

## 912 Appendix Table of Contents

913	• <b>A. Notation</b> .....	24
914	• <b>B. Related Works</b> .....	25
915	– Operator Learning .....	25
916	– Generative Models .....	25
917	– Parametric PDEs .....	26
918	• <b>C. Datasets Details</b> .....	27
919	– Combined Equation .....	27
920	– Advection Equation .....	29
921	– Wave Equation .....	31
922	– Gray-Scott Equation .....	32
923	– Vorticity Equation .....	34
924	• <b>D. Architecture Details</b> .....	35
925	– ENMA Encoder-Decoder .....	35
926	* Interpolation via Cross-Attention .....	35
927	* Compression via Causal Convolutions .....	37
928	– Tokenwise Autoregressive Generation .....	39
929	* Patchification of Latent Codes .....	39
930	* Causal Transformer .....	39
931	* Spatial Transformer .....	39
932	• <b>E. Implementation Details</b> .....	41
933	– Autoregressive Implementation .....	41
934	* Architecture Configuration .....	41
935	* Baseline Details .....	41
936	* Training Details .....	43
937	– Encoder-Decoder Implementation .....	44
938	* Architecture Details .....	44
939	* Baselines Details .....	44
940	* Training Details .....	45
941	• <b>F. Additional Experiments</b> .....	47
942	– ENMA Process Task .....	47
943	* Generative Ability of ENMA .....	47
944	* Forecasting with Less Compression .....	50
945	* Inference Speed Comparison .....	51
946	* Ablation Studies .....	51
947	– Experiments on the Encoder/Decoder .....	52
948	* OOD Encoding with 10% Input Grid .....	52
949	* Super-Resolution .....	53
950	* Ablation studies .....	54
951	* Geometry-Aware Attention Analysis .....	55
952	* Visualization in token space .....	56
953	• <b>G. Visualization</b> .....	59
954	– Combined Equation .....	59
955	– Advection Equation .....	60
956	– Gray-Scott Equation .....	61
957	– Wave Equation .....	62
958	– Vorticity Equation .....	64

## 959 A Notation

960 We summarize the main notations used throughout the paper:

Symbol	Description
$x$	Spatial coordinate
$t$	Temporal coordinate
$u(x, t)$	Solution of the PDE at $(x, t)$
$T$	Final time
$\pi$	Observation ratio (proportion of available inputs)
$\mathcal{X}$	Input spatial grid
$\mathcal{X}_{\text{te}}$	Test-time input spatial grid
$\gamma$	PDE parameters
$\mathcal{N}$	Differential operator
$\mathcal{B}$	Boundary condition operator
$\mathcal{G}$	True temporal evolution operator
$\hat{\mathcal{G}}$	Neural approximation of the evolution operator
$\mathbf{Z}$	True latent state from VAE tokenizer
$M$	Number of spatial tokens in $\mathbf{Z}$
$\mathbf{z}$	Spatial token, with $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_M)$
$L$	Number of observed time steps (history)
$\mathbf{Z}^{0:L-1}$	Latent sequence of past states
$\mathbf{Z}_{\text{DYN}}$	Predicted latent state from the Causal Transformer
$\tilde{\mathbf{Z}}$	Contextualized latent tokens from Spatial Transformer
$\tilde{\mathbf{z}}$	Spatial token in $\tilde{\mathbf{Z}} = (\tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_M)$
$\hat{\mathbf{Z}}$	Predicted latent state
$\hat{\mathbf{z}}$	Spatial token in $\hat{\mathbf{Z}} = (\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_M)$
$\mathcal{M}$	Masked token indices during training
$S$	Number of autoregressive steps
$s$	Autoregressive step index ( $s \in [0, S-1]$ )
$\mathcal{N}_s$	Masked token indices to predict at step $s$
$r$	Flow matching step index
$\mathbf{v}$	Velocity field for flow matching
$\epsilon$	Gaussian noise
$\Xi$	Regular interpolation grid
$\xi$	Interpolation grid point ( $\xi \in \Xi$ )
$\mathcal{L}$	Loss function
$Q$	Query tokens for attention
$K$	Key tokens for attention
$V$	Value tokens for attention
$m$	Scaling factor in cross-attention bias

## B Related Works

### B.1 Operator Learning

Operator learning has emerged as a powerful framework for modeling mappings between infinite-dimensional function spaces. Foundational works such as DeepONet and the Fourier Neural Operator (FNO) established neural operators (NOs) as effective tools for learning these mappings (Li et al., 2020a; Lu et al., 2021). Subsequent research has sought to improve both expressiveness and efficiency, exploring factorized representations for reduced complexity, wavelet-based techniques for multi-scale modeling (Gupta et al., 2021), and latent-space formulations to support general geometries (Tran et al., 2023; Li et al., 2023b). Implicit neural representations, as in CORAL, have also been proposed to accommodate variable spatial discretizations at inference time (Serrano et al., 2023).

A recent direction in operator learning leverages transformer-based architectures, inspired by their success in vision and language tasks. OFormer introduced a transformer model for embedding input-output function pairs, demonstrating the potential of transformers for operator learning (Li et al., 2023a). This has since led to more advanced architectures such as GNOT and Transolver, which improve input function encoding and generalization to irregular domains (Hao et al., 2023; Wu et al., 2024a). Aroma and UPT further adopt perceiver-style designs to learn compact and adaptable latent neural operators (Serrano et al., 2024b; Alkin et al., 2024b).

One close work is CViT, which learns compressed spatio-temporal representations using transformer encoders and decoders (Wang et al., 2024). In contrast, our approach first applies an attention-based encoder to map irregularly sampled fields to a uniform latent representation, followed by a causal spatio-temporal convolutional encoder that accommodates a flexible number of input states, enabling both regular and irregular conditioning.

### B.2 Generative Models

While most operator learning methods are deterministic, generative modeling introduces critical capabilities for modeling physical systems—most notably the ability to represent uncertainty and capture one-to-many mappings, which are especially relevant in chaotic or partially observed regimes. Two primary generative paradigms have emerged in this context: diffusion models and autoregressive transformers.

**Diffusion Transformers** Diffusion models synthesize data by learning to reverse a progressive noising process, typically through a sequence of denoising steps (Ho et al., 2020). In computer vision, Latent Diffusion Transformers (DiTs) have demonstrated strong performance by applying this principle in the latent space of a VAE (Peebles & Xie, 2022). More recently, diffusion-based techniques have been adapted to scientific modeling. For example, Kohl et al. (2024) proposed an autoregressive diffusion framework tailored for PDEs, particularly in turbulent settings where capturing stochasticity is essential. Similarly, Lippe et al. (2023) enhanced the modeling of high-frequency chaotic dynamics via noise variance modulation during denoising. Zhou et al. (2025) extended DiTs to the physical domain, generating PDE data from textual prompts.

Despite their expressiveness, diffusion models often require many sampling steps and lack efficient in-context conditioning. As an alternative, *Flow Matching* has recently gained attention. Instead of discrete denoising steps, it learns a continuous-time velocity field that transforms one distribution into another via an ODE. This leads to significantly faster sampling with far fewer steps, making it well-suited for scientific applications (Lipman et al., 2023, 2024).

**Autoregressive Transformers** Autoregressive models, originally designed for language modeling, have been successfully extended to image and video domains by treating spatial and temporal data as sequences. These models often couple a VQ-VAE with a causal or bidirectional transformer to model discrete token sequences (Oord et al., 2017; Esser et al., 2021; Chang et al., 2022). In video generation, frameworks such as *Magvit* and *Magvit2* (Yu et al., 2023, 2024) encode spatiotemporal information via 3D CNNs and generate frames autoregressively over quantized latent tokens. In the context of PDE modeling, *Zebra* adapts this paradigm by combining a spatial VQ-VAE with a causal transformer for in-context prediction (Serrano et al., 2024a). However, the use of discrete codebooks limits expressiveness and may hinder the ability to represent fine-grained physical phenomena, which are inherently continuous.

To address this, recent advances in vision have introduced masked autoregressive transformers that operate directly on continuous latent tokens (Li et al., 2024). These Masked Autoregressive (MAR) models predict subsets of tokens using a diffusion-style loss, allowing multi-token generation without relying on vector quantization. Notably, block-wise causal masking enables efficient inference while preserving temporal consistency (Deng et al., 2024). ENMA builds on this approach by adapting MAR for neural operator learning: our model learns to autoregress over continuous latent fields, supporting in-context generalization while maintaining high fidelity and computational efficiency.

### 1020 B.3 Parametric PDEs

1021 Generalizing to unseen PDE parameters is a central challenge in neural operator learning. Approaches  
1022 span classical data-driven training, gradient-based adaptation, and emerging in-context learning  
1023 strategies.

1024 **Classical ML Paradigm** A standard approach trains models on trajectories sampled from a dis-  
1025 tribution over PDE parameters, aiming to generalize to unseen configurations. This often involves  
1026 stacking past states as input channels (Li et al., 2021) or along a temporal axis, analogous to video  
1027 modeling (Ho et al., 2022; McCabe et al., 2023). However, such models typically degrade under dis-  
1028 tribution shifts, where small parameter changes lead to significantly different dynamics. Fine-tuning  
1029 has been proposed to address this (Subramanian et al., 2023), but often requires substantial data per  
1030 new PDE instance (Herde et al., 2024; Hao et al., 2024).

1031 **Gradient-Based Adaptation** To improve generalization, several methods train across multiple  
1032 PDE environments. *LEADS* (Yin et al., 2022a) employs a shared model with environment-specific  
1033 modules, updated during inference. Similarly, Kirchmeyer et al. (2022) introduce a hypernetwork  
1034 conditioned on learnable context vectors  $c^e$ , enabling adaptation to new environments. Building  
1035 on this, *NCF* (Nzoyem et al., 2024) uses a Taylor expansion to dynamically adjust context vectors,  
1036 offering both uncertainty estimation and inter-environment adaptation.

1037 **In-Context Learning for PDEs** Inspired by LLMs, recent work explores in-context learning (ICL)  
1038 for PDEs. Yang et al. (2023) propose a transformer that processes context-query pairs, but scalability  
1039 is limited to 1D or sparse 2D settings. Cao et al. (2024) extend this to PDEs via patches for a vision  
1040 transformer. They introduce a special conditioning mechanism, where input-output function are  
1041 given as inputs to the transformer, allowing to handle flexible time discretizations at inference. Chen  
1042 et al. (2024) propose unsupervised pretraining of neural operators, followed by ICL-style inference  
1043 via trajectory retrieval and averaging—without generative modeling. Serrano et al. (2024a) address  
1044 this by introducing a causal transformer over discrete latent tokens, paired with an effective in-context  
1045 pretraining strategy, allowing adaptation to new PDE regimes via in-context learning and probabilistic  
1046 predictions.



## 1047 C Datasets details

1048 To evaluate the performance of **ENMA**, we generate several synthetic datasets based on diverse  
 1049 **parametric PDEs** in 1D and 2D. Each dataset consists of 12,000 trajectories for training. For  
 1050 evaluation, we generate two test sets: 1,200 trajectories for in-distribution (*In-D*) and 120 for out-of-  
 1051 distribution (*Out-D*) evaluation. In the *In-D* case, test parameters  $\gamma$  differ from the training set but are  
 1052 sampled from the same distribution; in *Out-D*, they lie outside this range. Parameter ranges for the  
 1053 *in-D* and *Out-D* are presented in Table 3 and further detailed in each dataset section.

1054 Data generation proceeds as follows: for each sampled set of PDE parameters  $\gamma$  (as described in  
 1055 section 2), we simulate a batch of 10 trajectories using a numerical solver from 10 different initial  
 1056 conditions. Trajectories are computed over a time horizon  $t \in [0, T]$  on a spatial grid  $\mathcal{X}$ . This pipeline  
 1057 yields datasets with diverse and complex dynamics.

1058 We detail below the PDEs and parameter ranges used. In 1D, we consider the Combined equation  
 1059 (appendix C.1) and the Advection equation (appendix C.2); in 2D, we study the Wave equation  
 1060 (appendix C.3), the Gray-Scott system, and the Vorticity equation (appendix C.5).

Table 3: In-distribution (In-D) and out-of-distribution (Out-D) parameter ranges for each dataset.

Dataset	Parameter	In-D	Out-D
<b>Combined</b>	$\alpha$	$\mathcal{U}([0.3, 0.5])$	$\mathcal{U}([0.3, 0.5])$
	$\beta$	$\mathcal{U}([0.0005, 0.5])$	$\mathcal{U}([0.0005, 0.5])$
	$\gamma$	$\mathcal{U}([0.01, 1])$	$\mathcal{U}([0.01, 1])$
	$\delta$	—	$\mathcal{U}([0.5, 1])$
<b>Advection</b>	$\alpha$	$\mathcal{U}([-5, 5])$	$\mathcal{U}([-7, -5] \cup [5, 7])$
<b>Wave</b>	$c$	$\mathcal{U}([100, 500])$	$\mathcal{U}([500, 550])$
	$k$	$\mathcal{U}([0, 50])$	$\mathcal{U}([50, 60])$
<b>Gray-Scott</b>	$F$	$\mathcal{U}([0.023, 0.045])$	$\mathcal{U}([0.045, 0.0467])$
	$k$	$\mathcal{U}([0.0590, 0.0640])$	$\mathcal{U}([0.0570, 0.0590])$
<b>Vorticity</b>	$\nu$	$\mathcal{U}([10^{-3}, 10^{-2}])$	$\mathcal{U}([10^{-5}, 10^{-4}])$

### 1061 C.1 Combined Equation

1062 The **Combined equation** (Brandstetter et al., 2022) unifies several canonical PDEs—such as the heat  
 1063 and Korteweg–de Vries equations—by varying a set of coefficients  $(\alpha, \beta, \gamma)$ , enabling the modeling  
 1064 of diverse dynamical behaviors. It is defined as:

$$\frac{\partial u(x, t)}{\partial t} - \frac{\partial u(x, t)}{\partial x} \left( \alpha u(x, t)^2 - \beta \frac{\partial u(x, t)}{\partial x} + \gamma \frac{\partial^2 u(x, t)}{\partial x^2} \right) = 0, \quad (x, t) \in \Omega \times (0, T],$$

$$u(x, 0) = u^0(x), \quad x \in \Omega,$$

1065 with parameters sampled uniformly as  $\alpha \sim \mathcal{U}([0.3, 0.5])$ ,  $\beta \sim \mathcal{U}([0.0005, 0.5])$ , and  $\gamma \sim \mathcal{U}([0.01, 1])$   
 1066 for both training and in-domain test sets.

1067 Simulations are run over the domain  $\Omega \times [0, T] = [0, 2\pi] \times [0, 1]$ . The initial condition is defined as  
 1068 follows:

$$u^0(x) = \sum_{i=1}^N A_i \sin \left( \frac{2\pi l_i x}{L} + \phi_i \right),$$

1069 with domain length  $L = 2\pi$ , amplitudes  $A_i \sim \mathcal{U}([-0.5, 0.5])$ , phases  $\phi_i \sim \mathcal{U}([0, 2\pi])$ , and frequen-  
 1070 cies  $l_i \sim \mathcal{U}(\{1, 2, 3\})$ . Each trajectory is simulated for 100 time steps over a spatial grid of 256  
 1071 points. To train our models, we subsample each trajectory to retain only 20 time steps over a spatial  
 1072 grid of 128 points.

1073 **Out-Domain** For out-of-distribution evaluation, we consider a more complex scenario by adding a  
 1074 fourth-order spatial derivative. The resulting PDE is:

$$\frac{\partial u(x, t)}{\partial t} - \frac{\partial u(x, t)}{\partial x} \left( \alpha u(x, t)^2 - \beta \frac{\partial u(x, t)}{\partial x} + \gamma \frac{\partial^2 u(x, t)}{\partial x^2} + \delta \frac{\partial^3 u(x, t)}{\partial x^3} \right) = 0,$$

1075 with parameters sampled as  $\alpha \sim \mathcal{U}([0.3, 0.5])$ ,  $\beta \sim \mathcal{U}([0.0005, 0.5])$ ,  $\gamma \sim \mathcal{U}([0.01, 1])$ , and  $\delta \sim$   
 1076  $\mathcal{U}([0.5, 1])$ . Aside from this modification, simulations follow the same configuration as the in-  
 1077 distribution setup. Visualizations of the combined equation for in and out-domain trajectories are  
 1078 showed in Figure 4 and 5 respectively.

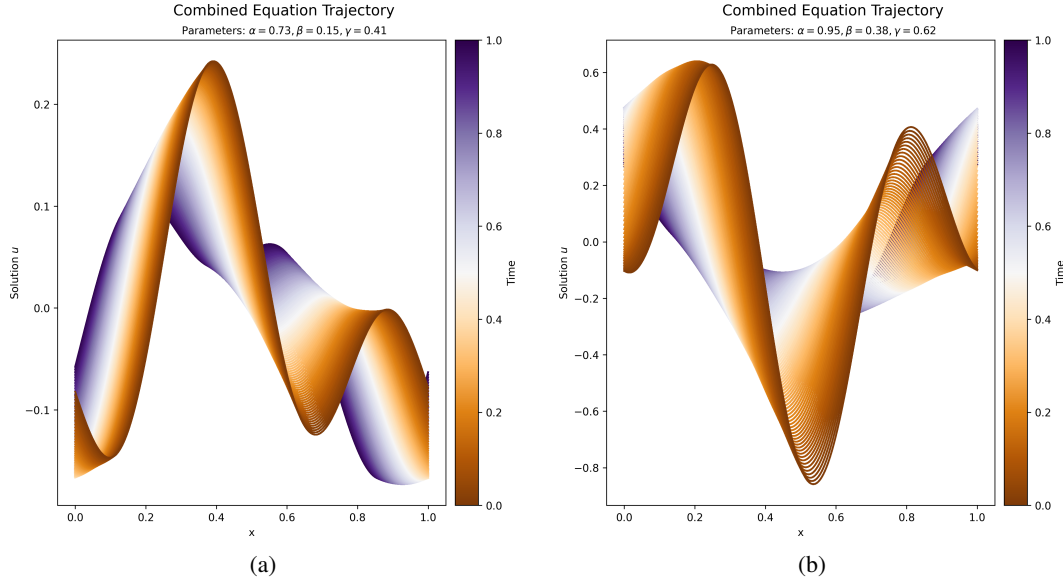


Figure 4: Samples from the Combined Dataset.

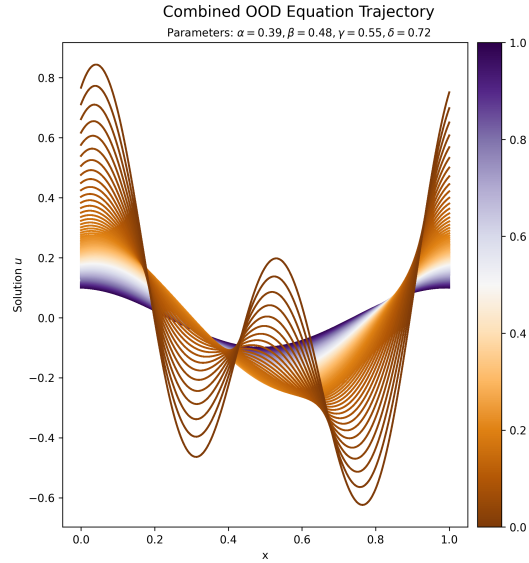


Figure 5: OOD sample of the Combined equation.

## 1079 C.2 Advection Equation

1080 The advection equation models the transport of a quantity at constant speed. For out-domain sets, the  
1081 advection speed is sampled from

1082 For our experiments, it is solved over the domain  $(\Omega \times [0, T]) = ([0, 1] \times [0, 1])$  and is defined as:

$$\begin{aligned} \frac{\partial u(x, t)}{\partial t} + \alpha \frac{\partial u(x, t)}{\partial x} &= 0 & (x, t) \in \Omega \times (0, T], \\ u(x, 0) &= u^0(x) & x \in \Omega, \end{aligned}$$

1083 where the advection speed  $\alpha$  is sampled from  $\mathcal{U}([-5, 5])$ . For *out-of-distribution* (Out-D) evaluation,  
1084 we increase the difficulty by sampling the advection speed  $\alpha$  from a uniform distribution  $\mathcal{U}([-7, -5] \cup$   
1085  $[5, 7])$ , explicitly excluding the training range  $[-5, 5]$ . For both in-domain and out-domain sets, we  
1086 generate trajectories defined by three types of initial conditions:

1087 • **Sine sum:**

$$u^0(x) = \sum_{i=1}^N A_i \sin\left(\frac{2\pi l_i x}{L} + \phi_i\right)$$

1088 • **Cosine sum:**

$$u^0(x) = \sum_{i=1}^N A_i \cos\left(\frac{2\pi l_i x}{L} + \phi_i\right)$$

1089 • **Sum of sine and cosine:**

$$u^0(x) = \sum_{i=1}^N A_i \left[ \sin\left(\frac{2\pi l_i x}{L} + \phi_i\right) + \cos\left(\frac{2\pi l_i x}{L} + \phi_i\right) \right]$$

1090 where  $L = 1$ ,  $A_i \sim \mathcal{U}([-0.5, 0.5])$ ,  $\phi_i \sim \mathcal{U}([0, 2\pi])$ , and  $l_i \sim \mathcal{U}(\{1, 2, 3\})$ . Each trajectory is  
1091 simulated for 100 time steps over a spatial grid of 1024 points. To train our models, we subsample  
1092 each trajectory to retain only 20 time steps over a spatial grid of 128 points. Visualizations of the  
advection equation for in and out-domain trajectories are shown in Figures 6 and 7.

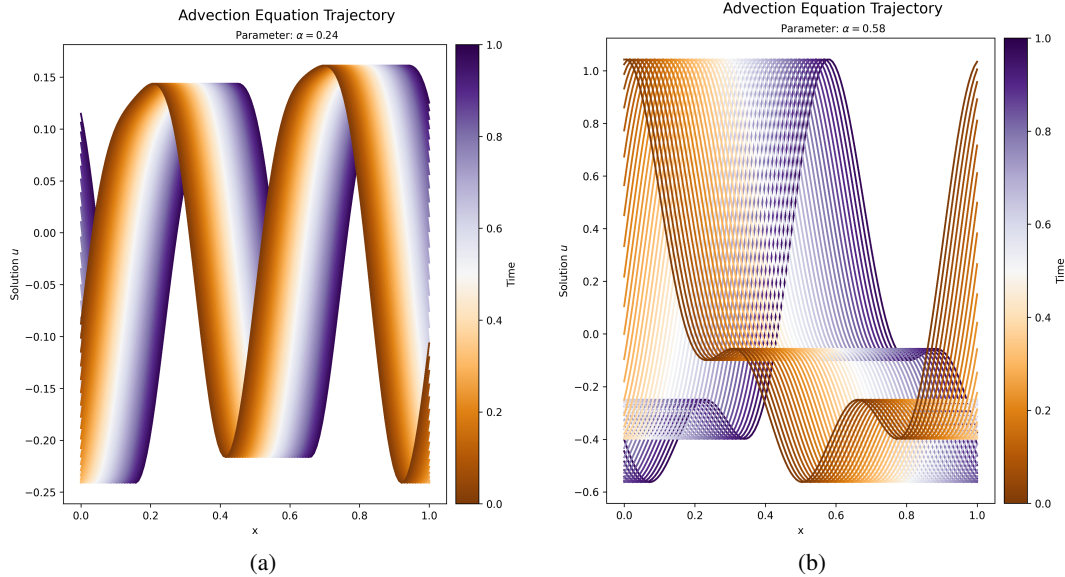


Figure 6: In-domain samples from the Advection Dataset.

1093

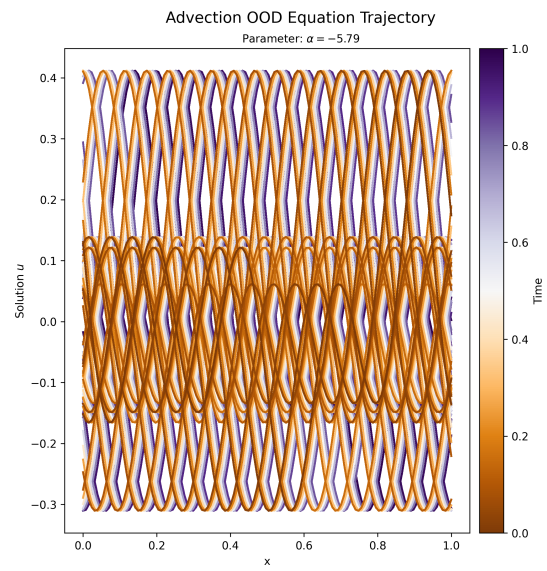


Figure 7: Out-domain sample from the Advection equation.

### 1094 C.3 Wave Equation

1095 We consider a 2D damped wave equation defined over the domain  $\Omega \times [0, T] = [0, 1]^2 \times [0, 0.005]$ ,  
 1096 given by:

$$\begin{aligned} \frac{\partial^2 \omega(x, t)}{\partial t^2} - c^2 \Delta \omega(x, t) + k \frac{\partial \omega(x, t)}{\partial t} &= 0, & (x, t) \in \Omega \times (0, T], \\ \omega(x, 0) &= \omega^0(x), & x \in \Omega, \end{aligned}$$

1097 where  $c \sim \mathcal{U}([100, 500])$  is the wave speed and  $k \sim \mathcal{U}([0, 50])$  the damping coefficient for the *In-D*  
 1098 datasets. For *Out-D*, we sample  $c \sim \mathcal{U}([500, 550])$  and  $k \sim \mathcal{U}[50, 60]$ . We focus on learning the  
 1099 scalar field  $\omega$ . The initial condition is defined as a sum of Gaussians:

$$\omega^0(x, y) = \sum_{i=1}^N \exp \left( -\frac{(x - x_i L)^2 + (y - y_i L)^2}{2\sigma_i^2} \right),$$

1100 with  $x_i, y_i \sim \mathcal{U}([0, 1])$ ,  $\sigma_i \sim \mathcal{U}([0.025, 0.1])$ ,  $L = 1$ , and  $N \sim \mathcal{U}(\{2, 3, 4\})$ . Simulations are  
 1101 performed on a  $64 \times 64$  spatial grid with 30 time steps. Two in-domain trajectories can be visualized  
 in Figure 8 and in Figure 9 for out-domain trajectories.

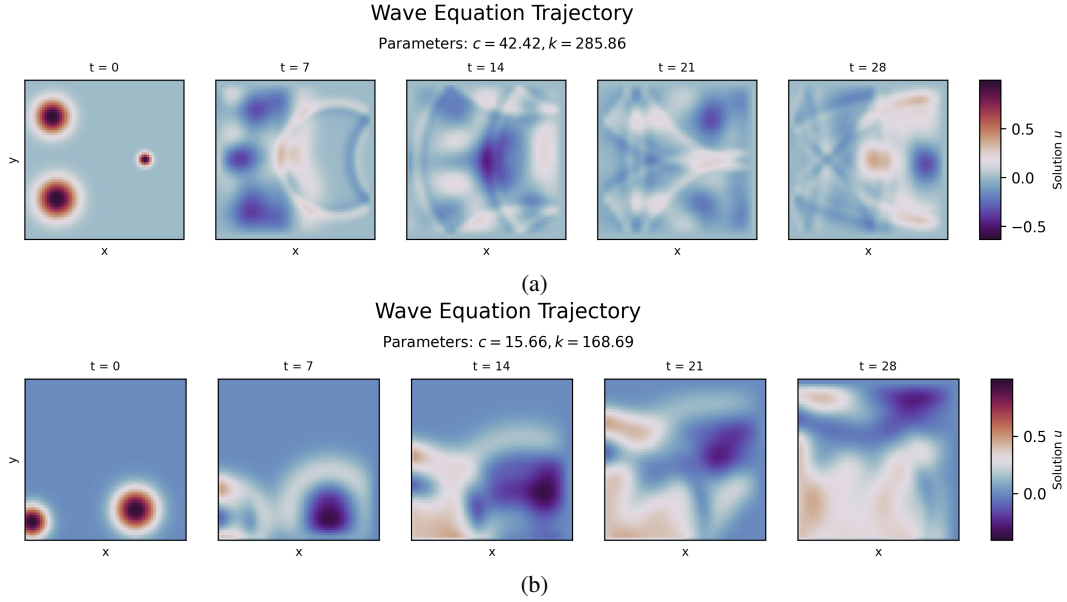


Figure 8: Samples from the Wave Dataset.

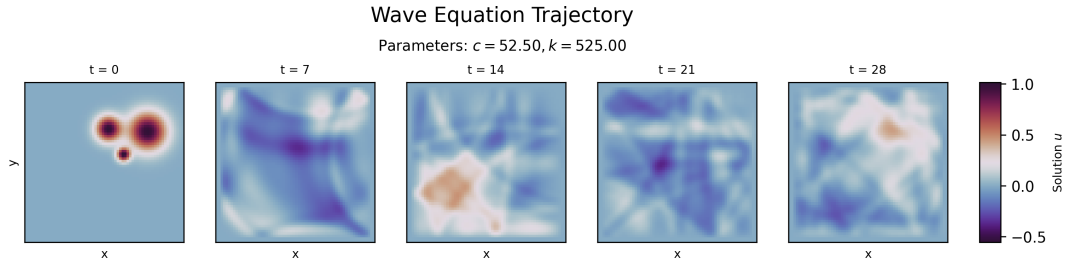


Figure 9: OOD sample of the Wave equation.

1102

#### 1103 C.4 Gray-Scott Equation

1104 The PDE describes a reaction-diffusion system generating complex spatiotemporal patterns, governed  
 1105 by the following 2D equations:

$$\begin{cases} \frac{du}{dt} = D_u \Delta u - uv^2 + F(1 - u), \\ \frac{dv}{dt} = D_v \Delta v - uv^2 - (F + k)v, \end{cases}$$

1106 where  $u, v$  represent the concentrations of two chemical species over a 2D spatial domain  $S$ , with  
 1107 periodic boundary conditions. The diffusion coefficients are fixed across all trajectories:  $D_u = 0.102$   
 1108 and  $D_v = 0.204$ . Each PDE instance  $\gamma$  is defined by variations in the reaction parameters. For *In-D*  
 1109 datasets, we sample  $F \sim \mathcal{U}([0.023, 0.045])$  and  $k \sim \mathcal{U}([0.0590, 0.0640])$ . For *Out-D* evaluation, we  
 1110 sample  $F \sim \mathcal{U}([0.045, 0.0467])$  and  $k \sim \mathcal{U}([0.0570, 0.00590])$ . The spatial domain is discretized  
 1111 as a  $32 \times 32$  grid with spatial resolution  $\Delta s = 2$ . Trajectories are presented in Appendix C.4 for  
 in-domain and in Figure 11 for out-domain.

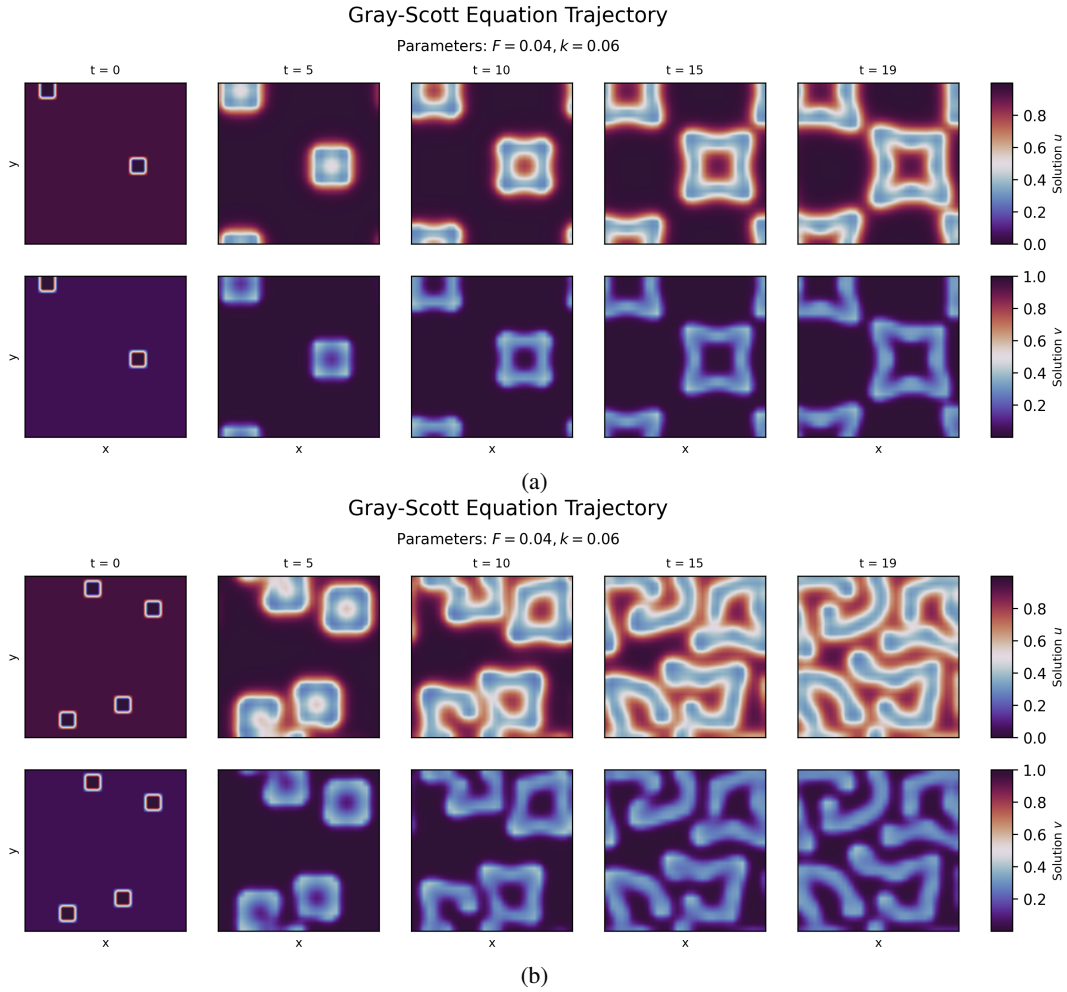


Figure 10: Samples from the Gray-Scott Dataset.

1112

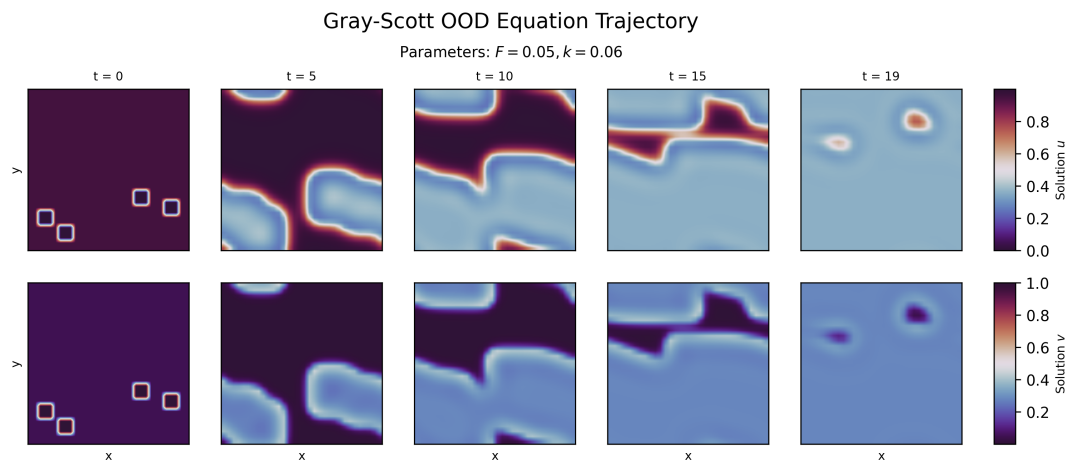


Figure 11: OOD sample of the Gray-Scott equation.



### 1113 C.5 Vorticity Equation

1114 We consider a 2D turbulence model and focus on the evolution of the vorticity field  $\omega$ , which captures  
 1115 the local rotation of the fluid and is defined as  $\omega = \nabla \times \mathbf{u}$ , where  $\mathbf{u}$  is the velocity field. The  
 1116 governing equation is:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega - \nu \nabla^2 \omega = 0, \quad (9)$$

1117 where  $\nu$  denotes the kinematic viscosity, defined as  $\nu = 1/\text{Re}$ . For *In-D* datasets, we sample  
 1118  $\nu \sim \mathcal{U}([10^{-3}, 10^{-2}])$ , while for *Out-D*, we consider a more challenging turbulent regime with  
 1119  $\nu \sim \mathcal{U}([10^{-5}, 10^{-4}])$ . The initial conditions are generated from the energy spectrum:

$$E(k) = \frac{4}{3} \sqrt{\pi} \left( \frac{k}{k_0} \right)^4 \frac{1}{k_0} \exp \left( - \left( \frac{k}{k_0} \right)^2 \right), \quad (10)$$

1120 where  $k_0$  denotes the characteristic wavenumber. Vorticity is linked to energy by the following  
 1121 equation :

$$\omega(k) = \sqrt{\frac{E(k)}{\pi k}} \quad (11)$$

1122 In-distribution and out-distribution trajectories can be visualized in Figure 12 and fig. 13 respectively.

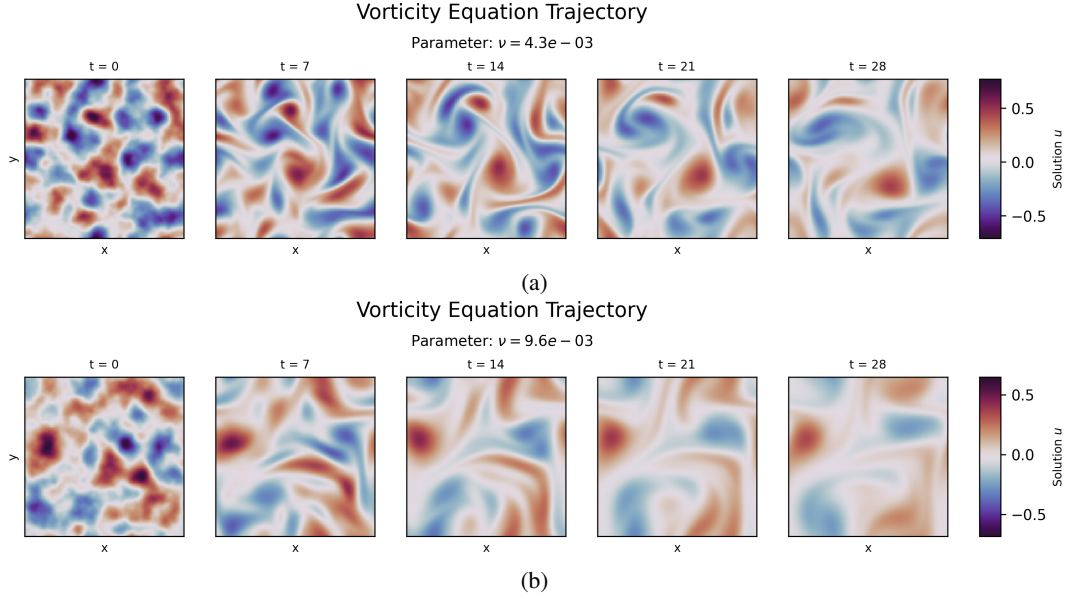


Figure 12: Samples from the Vorticity Dataset.

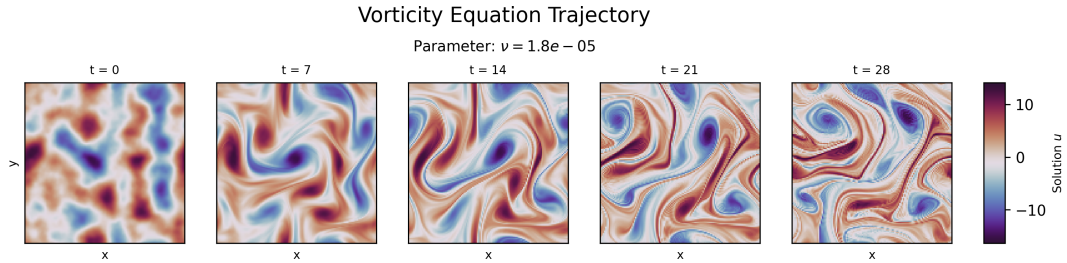


Figure 13: OOD sample of the Vorticity equation.

1123



## D Architecture details

### D.1 ENMA encoder-decoder

We describe the architecture of the encoder–decoder module in detail below. The overall encoding and decoding pipeline is illustrated in Figure 14. Starting from an input physical field  $u^{0:L-1}$  defined on a (possibly irregular) spatial grid  $\mathcal{X}_{in}$ , the encoder first applies an attention-based interpolation module (appendix D.1.1) to project the field onto a regular intermediate grid  $\Xi$ , producing  $u^{0:L-1}(\Xi)$ . This interpolation operates purely in space and is applied independently at each timestep.

Next,  $u^{0:L-1}(\Xi)$  is passed through a causal convolutional network that encodes the data in both space and time (appendix D.1.2), yielding latent tokens  $Z^{0:L_e-1}$  of length  $L_e < L$  when temporal compression is used. Temporal compression is optional, for notation simplicity, we will assume we do not compress time dimension. These latent tokens form the input to the autoregressive dynamics model.

The decoder mirrors the encoder’s structure. It first upsamples the latent tokens back to the intermediate grid  $\Xi$  (appendix D.1.2), then uses a cross-attention module to reconstruct the physical field on any desired output grid  $\mathcal{X}_{out}$ . This yields the predicted trajectory  $\hat{u}^{0:L-1}(\mathcal{X}_{out})$ . The final stage employs the same cross-attention mechanism as the encoder’s initial interpolation module (appendix D.1.1).

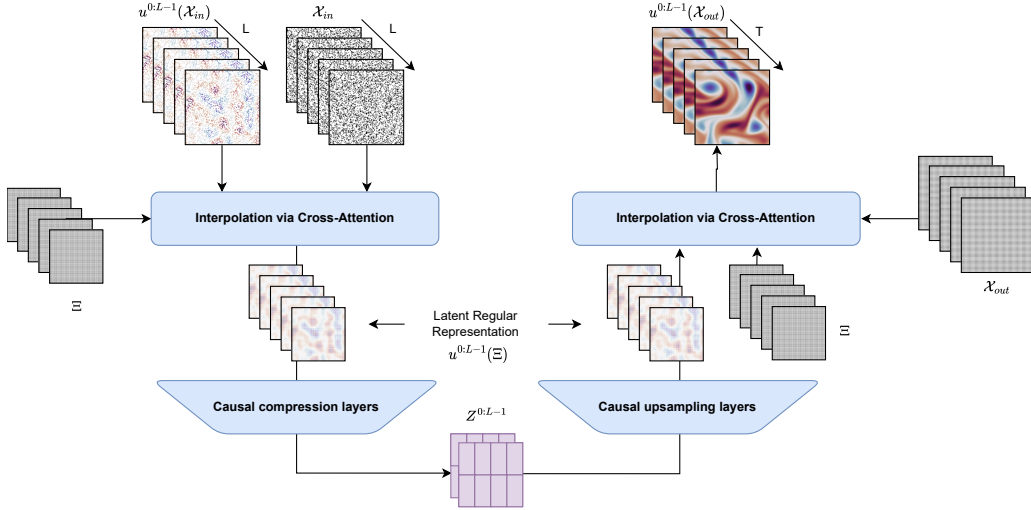


Figure 14: General architecture of ENMA’s encoder/decoder.

#### D.1.1 Interpolation via cross-attention

Figure 15 shows the detailed architecture of the interpolation module used in the encoder. The decoder mirrors this structure by directly reversing its operations.

The interpolation module takes as input a spatio-temporal field  $u^{0:L-1} \in \mathbb{R}^{|\mathcal{X}_{in}| \times L \times c}$  defined on a potentially irregular spatial grid  $\mathcal{X}_{in}$ . Positional encodings are first applied to the input coordinates as in Mildenhall et al. (2021), and the physical field is projected into a higher-dimensional representation using a linear layer, yielding embeddings  $p_e$  and  $h$ , which serve as the keys and values for the cross-attention module. Following Serrano et al. (2024b), the queries are learned latent embeddings that act as an intermediate representation.

To promote structured and localized interactions, we also initialize a learned coordinate grid  $\Xi$  for these latent queries. These coordinates are used to compute attention biases, encouraging stronger attention between spatially adjacent tokens. The cross-attention module outputs an intermediate field  $\in \mathbb{R}^{|\Xi| \times h}$ , which is further refined using Physics Attention (Wu et al., 2024a) to reduce computational overhead.

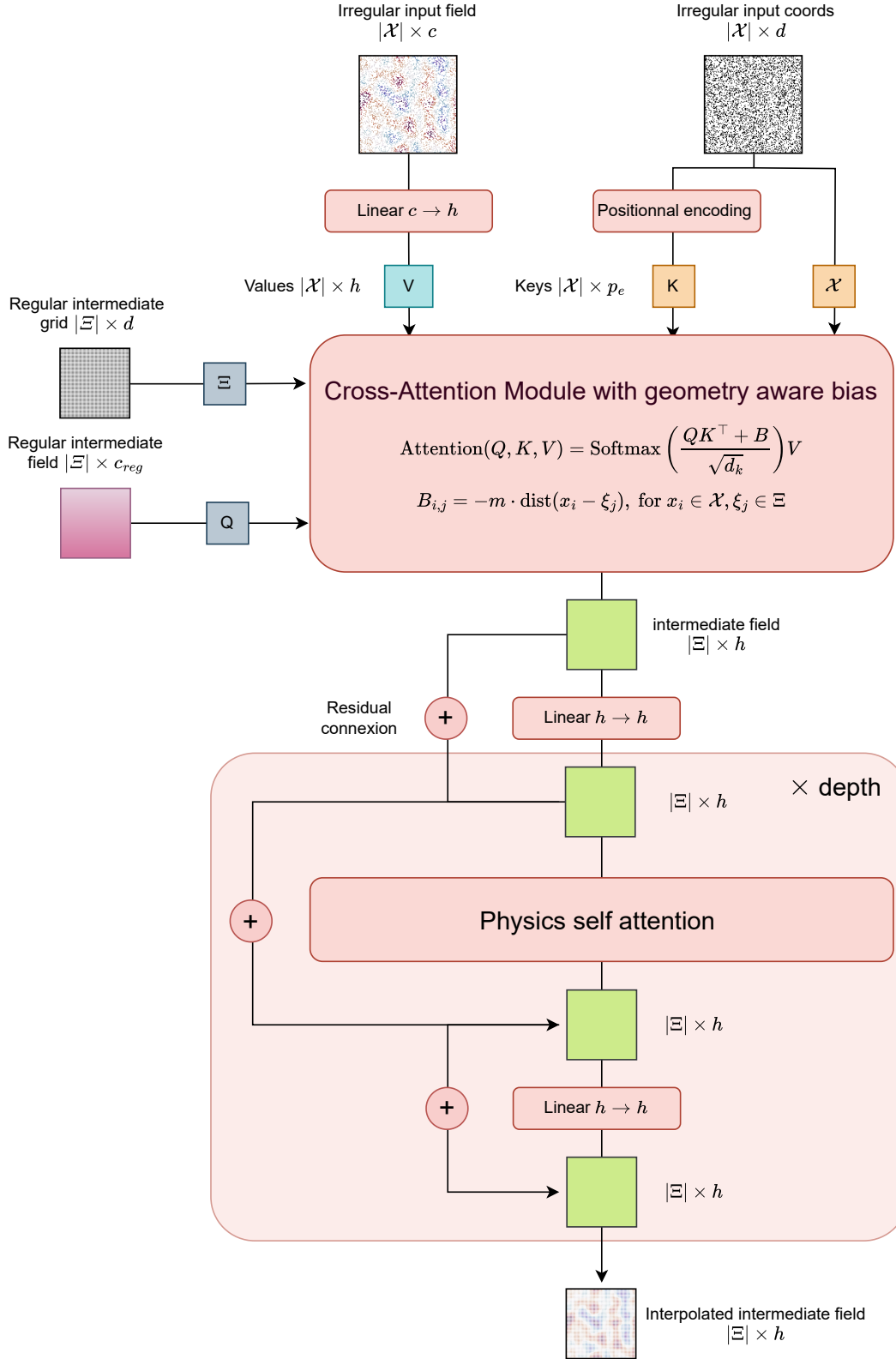


Figure 15: Detailed architecture of the Interpolation module.

1154 **Geometry-aware encoding** As presented in section 3.2, we introduce a geometry-aware attention  
 1155 bias that promotes attention locality. Appendix D.1.1 illustrates how this penalization is applied.  
 1156 Formally, we define the cross-attention from queries located at  $\mathcal{X}$  to keys and values located at  $\Xi$  as:

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{QK^\top + B}{\sqrt{d_k}} \right) V, \quad B_{i,j} = -m \cdot \text{dist}(x_i - \xi_j), \text{ for } x_i \in \mathcal{X}, \xi_j \in \Xi$$

1157 where  $Q = q(\mathcal{X})$ ,  $K = k(\Xi)$ , and  $V = v(\Xi)$  are the query, key, and value embeddings defined over  
 1158  $\mathcal{X}$  and  $\Xi$ , respectively;  $d_k$  is the key/query dimensionality; and  $m$  is a scaling factor, either fixed  
 1159 or learned as in ALiBi (Press et al., 2022). Intuitively, the bias  $B$  imposes stronger penalties for  
 1160 distant pairs  $(x_i, \xi_j)$ , reducing their attention weights and encouraging the model to focus on local  
 interactions during interpolation.

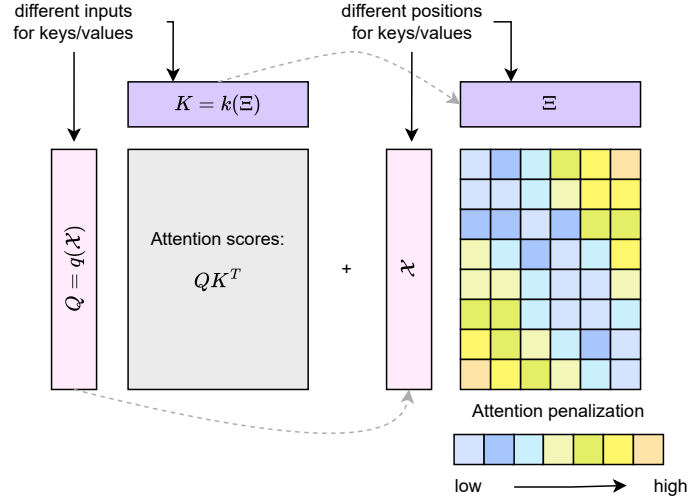


Figure 16: Geometry-aware Alibi bias.

1161

### 1162 D.1.2 Compression via causal convolutions

1163 For the architecture of the CNN, we adopt a design similar to that of Yu et al. (2023). In our  
 1164 case, it takes as input the output of the interpolation module,  $u^{0:L-1}(\Xi)$ , and lifts it to a higher-  
 1165 dimensional representation of size  $h_{\text{comp}}$  using a causal convolution layer (see fig. 3). The lifted  
 1166 tensor is then compressed through a stack of three building blocks: `residual`, `compress_space`,  
 1167 and `compress_time`. Finally, a concluding residual block projects the representation back to the  
 1168 target latent dimension. The up-sampling module strictly mirrors this compression pipeline. This  
 1169 type of architecture has been shown to be effective for spatio-temporal compression (Serrano et al.,  
 1170 2024a; Yu et al., 2023). The three types of layers used in the compression module are detailed below,  
 1171 and the full architecture is shown in fig. 17.

1172 **residual blocks:** The residual block processes the input while preserving its original shape. It  
 1173 consists of a causal convolution with kernel size  $k$ , followed by a linear layer and a Global Context  
 1174 layer adapted from Cao et al. (2020). If the output dimensionality differs from the input's, an  
 1175 additional convolution is used to project the spatial channels to the desired size.

1176 **compress\_space blocks:** The `compress_space` block reduces spatial resolution by a factor of  
 1177  $2^d$ , i.e., each spatial dimension is downsampled by a factor of 2 using a convolutional layer with  
 1178 stride  $s = 2$ . The kernel size  $k$  and padding  $p$  are set accordingly, with  $p = k/2$ . To ensure that  
 1179 only spatial dimensions are compressed, inputs are reshaped so that this operation does not affect the  
 1180 temporal axis.

1181 **compress\_time blocks:** The `compress_time` block performs temporal compression similarly to  
 1182 the spatial case, but operates along the time dimension. To preserve causality, padding is applied

1183 only to the past, with size  $p = k - 1$ , so that a frame at time  $t$  attends only to frames at times  $< t$ . A convolution with stride  $s = 2$  is used to reduce the temporal resolution by a factor of 2.

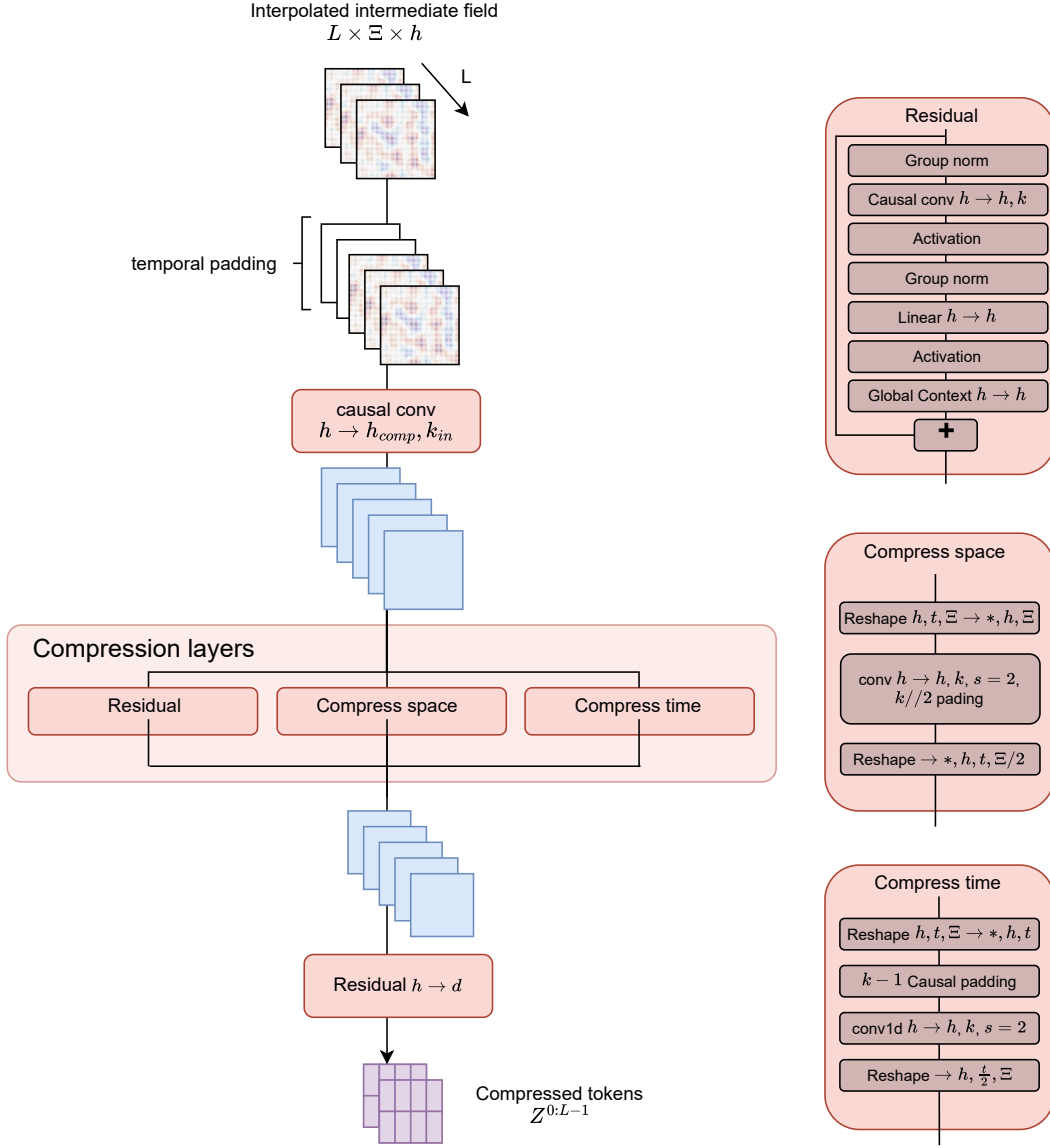


Figure 17: Detailed architecture of the Compression module.

1184

## D.2 Tokenwise Autoregressive Generation

We provide additional details about our generative process for tokenwise autoregression of physical fields.

### D.2.1 Patchification of Latent Codes

Given a sequence of physical states  $\mathbf{u}_{\mathcal{X}}^{0:L-1}$ , the VAE encoder produces a sequence of latent representations  $\mathbf{Z}^{0:L-1} \in \mathbb{R}^{M \times L \times d}$ , where  $M$  is the number of spatial tokens. For 2D physical systems, the latent representation has spatial dimensions  $\mathbf{Z}_{M_1 \times M_2}$ , with  $M = M_1 \times M_2$  and  $M_1 = M_2$  in our setting. The number of tokens per frame plays a critical role in capturing fine-grained spatial details. However, it also directly impacts the computational cost of tokenwise autoregressive generation, which scales with the number of tokens in each latent state.

To mitigate this cost while preserving spatial structure, we adopt a *patchify* strategy, commonly used in vision transformers. Specifically, we apply spatial down-sampling to the latent representation by dividing it into non-overlapping patches, reducing the token count per frame while retaining local coherence (see Figure 18). This significantly improves inference efficiency without sacrificing modeling fidelity.

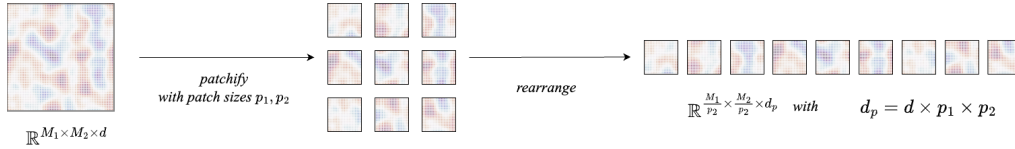


Figure 18: Illustration of latent patchification. The spatial latent map  $\mathbf{Z}_{M_1 \times M_2}$  is partitioned into coarser patches to reduce the number of autoregressed tokens.

### D.2.2 Causal Transformer

The causal Transformer introduced in section 3.1.1 captures the dynamics necessary to predict the next latent state  $\hat{\mathbf{Z}}^L$  from the past sequence  $\mathbf{Z}^{0:L-1}$ . We implement a next-state prediction strategy using a block-wise causal attention mask, where tokens within each frame can attend to one another, and tokens in frame  $i$  can attend to all tokens from earlier frames  $j < i$ . The attention mask used is defined as:

$$\mathcal{M}_{\text{attn}} = \underbrace{\begin{bmatrix} \mathbf{1}_{M \times M} & \mathbf{0}_{M \times M} & \cdots & \mathbf{0}_{M \times M} \\ \mathbf{1}_{M \times M} & \mathbf{1}_{M \times M} & \cdots & \mathbf{0}_{M \times M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{1}_{M \times M} & \mathbf{1}_{M \times M} & \cdots & \mathbf{1}_{M \times M} \end{bmatrix}}_{T \text{ columns}} \left. \vphantom{\begin{bmatrix} \mathbf{1}_{M \times M} & \mathbf{0}_{M \times M} & \cdots & \mathbf{0}_{M \times M} \\ \mathbf{1}_{M \times M} & \mathbf{1}_{M \times M} & \cdots & \mathbf{0}_{M \times M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{1}_{M \times M} & \mathbf{1}_{M \times M} & \cdots & \mathbf{1}_{M \times M} \end{bmatrix}} \right\} T \text{ rows}$$

This structure ensures that temporal causality is respected while allowing rich intra-frame interactions. It predicts a latent code  $\mathbf{Z}_{\text{DYN}}^L = \text{CausalTransformer}(\mathbf{Z}_{\text{BOS}}, \mathbf{Z}^{0:L-1})$  that captures the dynamics given the input sequence history.

For positional information, we apply standard sine-cosine embeddings along spatial dimensions. In the temporal direction, we omit explicit positional encodings, relying instead on the causal structure to encode temporal ordering implicitly (Kazemnejad et al., 2023).

## D.3 Spatial Transformer

At inference time, given a target number of steps  $S$ , the spatial Transformer generates the final latent frame  $\hat{\mathbf{Z}}^L$  in an autoregressive manner over  $S$  steps. Each step selectively predicts a subset of masked tokens, conditioned on the dynamic context  $\mathbf{Z}_{\text{DYN}}^L = \text{CausalTransformer}(\mathbf{Z}_{\text{BOS}}, \mathbf{Z}^{0:L-1})$ . The token prediction schedule follows a cosine scheduler, progressively revealing tokens in a smooth, annealed fashion. Figure 19 illustrates how tokens are generated at inference for  $S = 4$  steps.

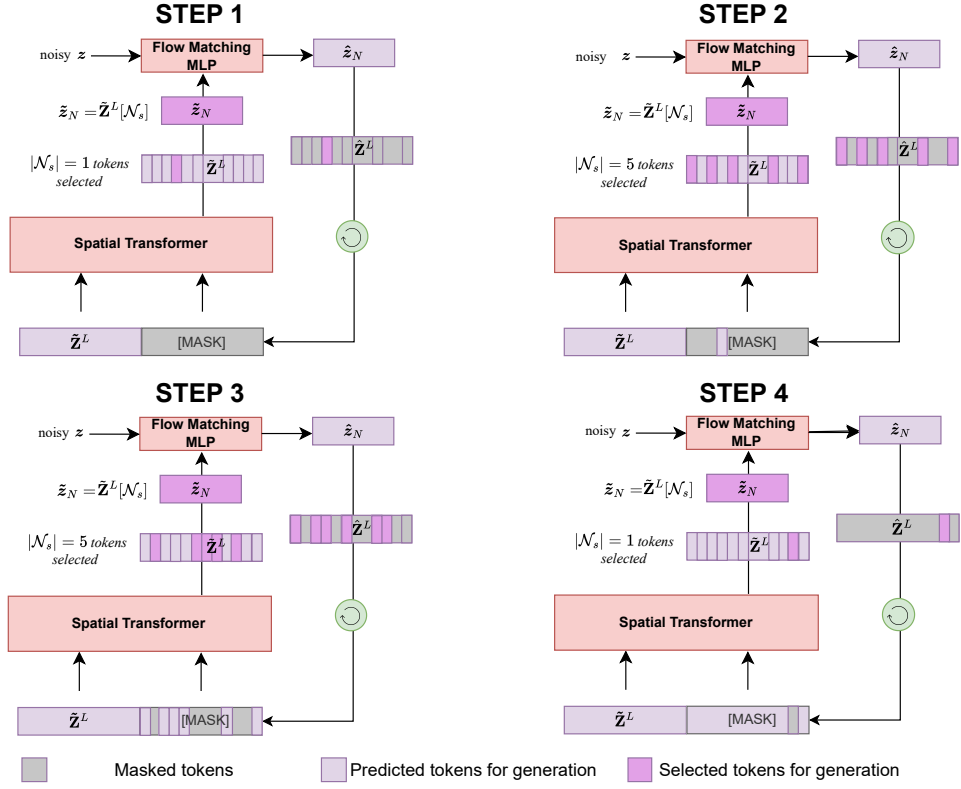


Figure 19: Illustration of tokenwise autoregressive generation at inference with a spatial Transformer over  $S = 4$  steps. At each step, a subset of tokens is selected for generation based on a cosine schedule.

## E Implementation details

The code has been written in Pytorch (Paszke et al., 2019). All experiments were conducted on a A100. We estimate the total compute budget—including development and evaluation—to be approximately 1000 GPU-days.

### E.1 Autoregressive Implementation Details

This section outlines the implementation details related to the autoregressive time-stepping and generative experiments with ENMA, along with the corresponding baseline configurations.

#### E.1.1 Architecture Configuration

To enable a fair comparison with existing baselines, we disable the interpolation component of ENMA’s encoder-decoder when conducting autoregression on a fixed grid, consistent with the setup used in Alkin et al. (2024a). The hyperparameter settings for ENMA’s generation architecture across all datasets are summarized in table 4.

Table 4: Model hyper-parameters configuration for all datasets for the ENMA’s generation process.

Hyperparameters	Combined	Advection	GS	Wave	Vorticity
Vae embedding dimension	4	4	4	8	8
Number of tokens	16	16	64	256	256
Patch size	1	1	1	4	4
Spatial Transformer depth	6	6	6	6	6
Causal Transformer depth	6	6	6	6	6
hidden size	512	512	512	512	512
mlp ratio	2	2	2	2	2
num heads	8	8	8	8	8
dropout	0	0	0	0	0
QK normalization	True	True	True	True	True
Normalization Type	RMS	RMS	RMS	RMS	RMS
activation	Swiglu	Swiglu	Swiglu	Swiglu	Swiglu
Layer Norm Rank	24	24	24	24	24
Positional embedding	Sinus	Sinus	Sinus	Sinus	Sinus
MLP depth	3	3	3	3	3
MLP width	512	512	512	512	512
number of steps $S$	6	6	16	16	16
FM steps	10	10	10	10	10

#### E.1.2 Baseline details

We detail here the architecture of the baselines used to evaluate ENMA’s dynamics forecasting experiments. We compared our method to FNO (Li et al., 2020a), BCAT (Liu et al., 2025), AViT (McCabe et al., 2023), AR-DiT (Kohl et al., 2024), Zebra (Serrano et al., 2024a), an In-Context ViT and a [CLS] ViT as done in (Serrano et al., 2024a).

**FNO** For FNO, we followed the authors guidelines and concatenated the temporal historic directly with the channels. We considered 10 modes for both 1D and 2D datasets and used a width of 128. We stacked 4 Fourier layers.

**BCAT** BCAT is a deterministic block-wise causal transformer approach for learning spatio-temporal dynamics. It has been proposed for tackling multi-physics problems, and we adapted it for parametric PDEs. BCAT performs autoregression in the physical space. As vision transformers (Dosovitskiy

et al., 2021), it relies on patches to reduce the number of tokens. We report the model hyper-parameters details for the Transformer in table 5.

Table 5: Model hyper-parameters configuration for all datasets for the BCAT process.

Hyperparameters	Combined	Advection	GS	Wave	Vorticity
Patch size	8	8	8	8	8
Transformer depth	6	6	6	6	6
hidden size	512	512	512	512	512
mlp ratio	2	2	2	2	2
num heads	8	8	8	8	8
dropout	0	0	0	0	0
QK normalization	True	True	True	True	True
Normalization Type	RMS	RMS	RMS	RMS	RMS
activation	Swiglu	Swiglu	Swiglu	Swiglu	Swiglu
Positional embedding	Sinus	Sinus	Sinus	Sinus	Sinus

**AViT** We evaluate against the Axial Vision Transformer (AViT) introduced in (Müller et al., 2022), which performs attention separately along spatial and temporal dimensions. Their proposed MPP model extends AViT to multi-physics settings by incorporating strategies for handling multi-channel inputs and system-specific normalization. In our parametric setting, such mechanisms are unnecessary. We adopt the same hyperparameter configuration as in table 5, which we found to yield the best results.

**AR-DiT** Our autoregressive diffusion transformer follows the setup proposed in Kohl et al. (2024), where known frames are concatenated with noise and denoised progressively to generate the next physical state. As in (Rombach et al., 2022), we employ AdaLayerNorm to condition the model during the diffusion process. The corresponding hyperparameters are provided in table 6.

Table 6: Model hyper-parameters configuration for all datasets for the AR-DiT process.

Hyperparameters	Combined	Advection	GS	Wave	Vorticity
Patch size	8	8	8	8	8
Transformer depth	6	6	6	6	6
hidden size	512	512	512	512	512
mlp ratio	4	4	4	4	4
num heads	8	8	8	8	8
dropout	0	0	0	0	0
Normalization Type	AdaLayer	AdaLayer	AdaLayer	AdaLayer	AdaLayer
Positional embedding	Sinus	Sinus	Sinus	Sinus	Sinus
Diffusion steps	100	100	100	100	100

**Zebra** Zebra is a token-based autoregressive model that combines a VQ-VAE for discretizing spatial fields with a causal transformer for modeling temporal dynamics. At each time step, it autoregressively predicts discrete latent tokens corresponding to the next frame, allowing for uncertainty quantification through stochastic sampling in latent space. The full set of hyperparameters for both the VQ-VAE and the transformer components is provided in table 7.

**In-Context ViT** We implement an in-context Vision Transformer (Dosovitskiy et al., 2020) to address the initial value problem using a context trajectory. The model is trained to forecast the next frame in a target sequence, conditioned on both the observed context and preceding frames. For architectural configuration, we adopt the same hyperparameters as reported in table 5.



Table 7: Hyperparameters for Zebra’s VQVAE and Transformer components.

Hyperparameters	Advection	Combined	GS	Wave	Vorticity
VQ-VAE					
start_hidden_size	64	64	128	128	128
max_hidden_size	256	256	1024	1024	1024
num_down_blocks	4	4	2	3	2
codebook_size	256	256	2048	2048	2048
code_dim	64	64	16	16	16
num_codebooks	2	2	1	2	1
shared_codebook	True	True	True	True	True
tokens_per_frame	32	32	256	128	256
start_learning_rate	3e-4	3e-4	3e-4	3e-4	3e-4
weight_decay	1e-4	1e-4	1e-4	1e-4	1e-4
scheduler	Cosine	Cosine	Cosine	Cosine	Cosine
num_epochs	1000	1000	300	300	300
Transformer					
max_context_size	2048	2048	8192	8192	8192
batch_size	4	4	2	2	2
num_gradient_accumulations	1	1	4	4	4
hidden_size	256	256	384	512	384
mlp_ratio	4.0	4.0	4.0	4.0	4.0
depth	8	8	8	8	8
num_heads	8	8	8	8	8
vocabulary_size	264	264	2056	2056	2056
start_learning_rate	1e-4	1e-4	1e-4	1e-4	1e-4
weight_decay	1e-4	1e-4	1e-4	1e-4	1e-4
scheduler	Cosine	Cosine	Cosine	Cosine	Cosine
num_epochs	100	100	30	30	30

1262 **[CLS] ViT** This variant of the Vision Transformer is adapted to a meta-learning setting, where a  
1263 dedicated [CLS] token captures the environment-specific variations across different PDE settings.  
1264 During inference, the [CLS] token is updated via 100 gradient steps to adapt to new environments.  
1265 We use the same model configuration as in table 5.

### 1266 E.1.3 Training details

1267 Models for 1D datasets were trained for approximately 8 hours, while training on 2D datasets took  
1268 around 20 hours. Unless otherwise specified, all datasets followed the same training objective.

1269 **Optimizer and Learning Rate Schedule** We use the AdamW optimizer with  $\beta_1 = 0.9$  and  
1270  $\beta_2 = 0.95$  for all experiments. The learning rate follows a cosine decay schedule, starting from an  
1271 initial value of  $10^{-3}$  and annealing to  $10^{-5}$  over the course of training. To stabilize the early training  
1272 phase, we apply a linear warmup over the first 500 optimization steps.

1273 **Baselines** All baselines were trained following the same protocol as ENMA. Each model was trained  
1274 from scratch for 1000 epochs, and the best-performing checkpoint was selected based on training  
1275 performance. A cosine learning rate scheduler was used across all methods, which consistently  
1276 improved prediction quality. All models were trained with a learning rate of  $1e^{-3}$ , except for AR-DiT,  
1277 which required a lower rate of  $1e^{-4}$  due to unstable training dynamics.

Table 8: Training hyper-parameters configuration for all datasets for the ENMA’s generation process.

Hyperparameters	Combined	Advection	GS	Wave	Vorticity
Epochs	1000	1000	1000	1000	1000
batch size	128	128	128	64	64
learning rate	1e-3	1e-3	1e-3	1e-3	1e-3
weight Decay	1e-4	1e-4	1e-4	1e-4	1e-4
grad clip norm	1	1	1	1	1
betas	[0.9, 0.95]	[0.9, 0.95]	[0.9, 0.95]	[0.9, 0.95]	[0.9, 0.95]

## E.2 Encoder-Decoder implementation protocol

### E.2.1 Architecture details

We present in table 9 the hyper parameters for the architecture of ENMA’s auto encoder.

Table 9: Architecture details of the auto encoder as presented in table 1. We refer the reader to appendices D.1.1 and D.1.2 for the notation.

Module	Block	Parameter	Advection	Vorticity
Interpolation module	intermediate grid size	$ \Xi $	128	$16 \times 16$
	Positional encoding	positional encoding	Nerf	Nerf
		positional encoding num freq	12	12
		positional encoding max freq	4	4
	Cross attention block	intermediate grid dim $c_{reg}$	64	16
		hidden dim $h$	16	16
		n cross-attention heads	4	4
		cross-attention dim heads	4	4
	Geometry-aware bias	alibi scaling $m$	1, 2, 3, 4	1, 2, 3, 4
		n alibi heads	4	4
Compression module	Physics self attention	depth	2	2
		n attention heads	4	4
		attention dim heads	4	4
		physics-attention n slice	16	64
	Compression layers	Compression layers	residual	residual
			compress_space	compress_space
			residual	residual
			compress_space	compress_time
			residual	residual
			compress_space	-
			residual	-
			compress_time	-
			residual	-
		$h_{comp}$	16	16
		compression kernel size $k$	7	7
	causal input layer	kernel size $k_{in}$	7	7
	causal output layer	kernel size $k_{out}$	7	7
	token dim	-	4	8

### E.2.2 Baselines details

We detail here the architecture of the baselines used to evaluate ENMA’s auto encoding. Hyper-parameters are chosen from the original papers and/or closest available implementation. We recall that Oformer acts pointwise in the physical space, and thus does not make any spatial nor temporal compression. CORAL, AROMA and GINO act at a frame level *i.e.* they compress space but not time. Finally, our ENMA’s auto encoder architecture performs both a spatial and a temporal compression. We provide a comparison in table 10.

Table 10: Token sizes used to evaluate auto encoders as presented in table 1. We show sizes using the following formatting: temporal size  $\times$  spatial size  $\times$  token dimension.

Model	<i>Advection</i>	<i>Vorticity</i>
<i>Trajectory sizes</i>	$50 \times 128 \times 1$	$30 \times 64 \times 64 \times 1$
Oformer	$50 \times 128 \times 4$	$30 \times 64 \times 64 \times 8$
GINO	$50 \times 64 \times 4$	$30 \times 16 \times 16 \times 8$
AROMA	$50 \times 16 \times 4$	$30 \times 64 \times 8$
CORAL	$50 \times 64$	$30 \times 2048$
ENMA	$26 \times 16 \times 4$	$16 \times 8 \times 8 \times 8$

**Oformer** Oformer models are trained point wise *i.e.* no spatial neither temporal compression are processed. We used 4-depth layers with Galerkin attention type and 128 latent channels.

**GINO** GINO architecture is a combination of 2 well-known neural operator architecture (GNO (Li et al., 2020b) and FNO (Li et al., 2020a)). The GNO uses the following architecture parameters: a GNO radius of 0.033 with linear transform. The latent grid size is 16 for 1d datasets and  $16 \times 16$  for 2d. As a consequence, we adapt the FNO modes to 8 for 1d datasets and  $8 \times 8$  for 2d's.

**AROMA** AROMA makes use of a latent token of size 16 in 1d and 64 in 2d. Other architecture parameters are kept similar *i.e.* 4 cross and latent heads with dimension 32. The hidden dimension of AROMA's architecture is set to 128. The INR has 3 layers of width 64.

**CORAL** Finally, CORAL is a meta-learning baseline that represents solution using an INR (SIREN (Sitzmann et al., 2020)), conditioned from a hyper-network. The hyper-network takes as input a latent code, optimized with 3 gradient descent steps. The inner learning rate is set to 0.01. The hyper network has 1 layer with 128 neurons. SIREN models have 6 layers with 256 neurons.

As a comparison between baselines, we show below the latent token sizes that are created with the hyper parameters detailed above.

### E.2.3 Training details

**Reconstruction performances** For training the auto-encoder (as well as the baseline models, unless stated otherwise), we followed a unified training procedure. The loss function combines a relative mean squared error term with a KL divergence term:  $\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta \cdot \mathcal{L}_{\text{KL}}$ , where  $\beta = 0.0001$ .

To enhance robustness to varying levels of input sparsity, we randomly subsample the spatial input grid during training. The number of input points ranges from 20% to 100% of the full grid  $\mathcal{X}$ . The output grid used for loss computation remains fixed to the full grid. Grid subsampling is performed independently for each sample in a batch and is refreshed at every iteration.

Models are optimized using the AdamW optimizer with an initial learning rate of 0.001, scheduled via a cosine decay down to  $1e^{-7}$  over the course of training. Training batch sizes for each encoder-decoder architecture are listed in table 11.

- **ENMA:** Trained with a learning rate of 0.001 and batch size of 64 in 1D. The autoencoder is trained for 1,000 epochs in 1D and 150 epochs in 2D.
- **AROMA:** Training procedure follows ENMA. 2D datasets are trained for 1,000 epochs.
- **CORAL:** To stabilize training, the KL loss term is removed. The model is trained as a standard autoencoder minimizing relative MSE, with an initial learning rate of  $1e^{-6}$ .
- **Oformer:** Follows the same training procedure as ENMA. 2D datasets are trained for 1,000 epochs.
- **GINO:** Training setup is similar to ENMA. 2D datasets are trained for 100 epochs due to the inner loop over batches in the forward pass.

Table 11: Batch size used for training autoencoders as presented in table 1.

Model	<i>Advection</i>	<i>Vorticity</i>
Oformer	64	4
GINO	64	32
AROMA	128	8
CORAL	128	8
ENMA	64	8

**Time-stepping** The time-stepping task consists in training a small FNO to unroll the dynamics in the latent space created by the different models. This allows us to assess the quality of the extracted features for dynamic modeling. Training proceeds as follows: Starting from a trajectory, we encode the entire trajectory. We then take the 2 first tokens in the temporal dimension as input for the FNO. By concatenating this to the PDE parameters, we unroll the dynamic in the latent space, to build the full sequence of tokens. We compute the loss (Relative MSE) **in the latent space** directly and at each step in the auto regressive process. All models are trained using an initial learning rate of 0.0001 with and AdamW optimizer. The learning is also scheduled using a cosine scheduler. 250 epochs are performed for all baselines in 1d and in 2 GINO and CORAL performs 50 training epochs, while Oformer and CORAL are trained for 150 epochs. This trainings can differs depending on the computational cost of the encoder.

The FNO used for processing is a simple 1d/2d FNO, with 3 layers of width 64. FNO modes are set to 8 and the activation function is a GELU fonction.

## F Additional experiments

We conduct a wide range of experiments to evaluate the capabilities of our model **ENMA**:

- Generative capabilities (appendix F.1.1): evaluation on uncertainty quantification, sample diversity, solver fidelity, and latent distribution alignment.
- Compression study (appendix F.1.2): Dynamics forecasting evaluation with reduced latent compression on 2D datasets.
- Inference speed comparison (appendix F.1.3): benchmarking ENMA against generative baselines.
- Ablation studies (appendix F.1.4): investigating the impact of history length, autoregressive steps, and flow matching iterations.
- OOD reconstruction (appendix F.2.1): testing under high sparsity with only 10% of the spatial input grid.
- Super-resolution (appendix F.2.2): evaluation on finer output resolutions beyond the training grid.
- Encoder-decoder variants (appendix F.2.3): ablations on positional bias and time-stepping architectures.
- Geometry-Aware Attention Analysis (appendix F.2.4): qualitative inspection of the geometry-aware attention module
- Token space visualization (appendix F.2.5): qualitative visualization of the encoded tokens

### F.1 ENMA process task

#### F.1.1 Generative ability of ENMA

We illustrate here additional benefits from the generative capabilities of **ENMA** through two example tasks: *uncertainty quantification* and *new trajectory generation*. For all uncertainty experiments, we focus on the Combined equation, as generating multiple samples can be costly for 2D data.

**Uncertainty quantification** ENMA naturally enables uncertainty quantification by generating multiple stochastic samples from the learned conditional distribution. This is achieved by sampling multiple candidates in the latent space via different noise realizations during the spatial decoding process. Specifically, for each autoregressive step, ENMA injects Gaussian noise into the flow matching sampler, producing diverse plausible predictions for the spatial tokens. Repeating this process yields an ensemble of trajectories. In contrast, discrete AR models rely on sampling from categorical distributions (e.g., top- $k$  or nucleus sampling), which often leads to overconfident outputs, limited diversity, and poorly calibrated uncertainty estimates.

An illustration is shown in fig. 20, where the red curve denotes the ground truth, the blue curve is the predicted mean at the final time step, and the shaded region indicates the empirical confidence interval defined by  $\pm 3$  standard deviations.

To evaluate uncertainty quality, we report two standard metrics:

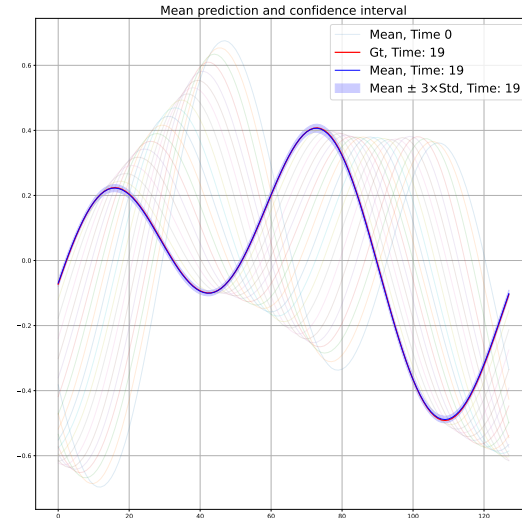


Figure 20: Uncertainty quantification using ENMA. Multiple trajectories are sampled, and the final time step is used to compute the point-wise mean (blue), standard deviation (shaded), and ground truth (red).

- **CRPS** (Continuous Ranked Probability Score) measures the accuracy and sharpness of probabilistic forecasts by comparing the predicted distribution to the true outcome. Lower CRPS indicates better probabilistic calibration.
- **RMSCE** (Root Mean Squared Calibration Error) quantifies calibration by comparing predicted confidence intervals with empirical coverage. A low RMSCE suggests well-calibrated uncertainty estimates.

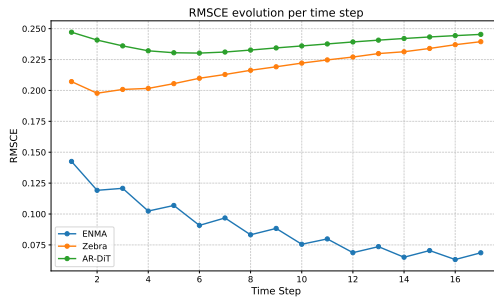
Method	Metric	Combined
AR-DiT	RMSCE	0.2679
	CRPS	0.0127
Zebra	RMSCE	<u>0.2190</u>
	CRPS	<u>0.0090</u>
ENMA (ours)	RMSCE	<b>0.08683</b>
	CRPS	<b>0.0017</b>

Table 12: Comparison of uncertainty metrics ( $\downarrow$  is better) for Combined.

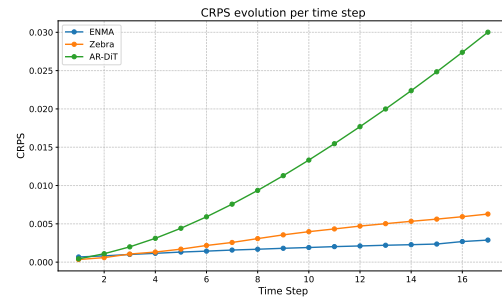
modeling for uncertainty-aware forecasting.

**CRPS and RMSCE Results** ENMA significantly outperforms both **AR-DiT** and **Zebra** in terms of uncertainty calibration and probabilistic accuracy, as shown in table 12. These results indicate that ENMA provides both sharper and more reliable uncertainty estimates. The stochastic sampling in latent space enabled by flow matching allows ENMA to model diverse plausible trajectories while maintaining strong calibration. In contrast, discrete token-based methods like Zebra and AR-DiT tend to produce overconfident or underdispersed forecasts. This demonstrates the advantage of per continuous latent-space

**Temporal Uncertainty Evaluation.** To further assess the quality of uncertainty calibration over time, we report the evolution of RMSCE and CRPS across autoregressive prediction steps in fig. 21. Ideally, both metrics should remain low and stable over time, indicating that the model maintains well-calibrated uncertainty estimates and accurate probabilistic forecasts throughout the trajectory. For CRPS, all models exhibit increasing scores over time due to the compounding errors typical of neural PDE solvers. However, ENMA consistently achieves substantially lower CRPS values than both Zebra and AR-DiT, indicating sharper and more accurate trajectory forecasts. In the case of RMSCE, ENMA displays a distinct trend: its calibration error decreases over time. This behavior reflects the model’s evolving confidence—initial predictions, informed by ground truth history, tend to be overly confident, resulting in narrower (and sometimes miscalibrated) confidence intervals. As the model progresses through the trajectory and relies more on its own predictions, it becomes more conservative, yielding better-calibrated uncertainty estimates. ENMA maintains the lowest RMSCE at every step, underscoring its robustness in providing reliable confidence intervals throughout the rollout. Together, these results highlight ENMA’s superiority in delivering both accurate and well-calibrated probabilistic forecasts over time.



(a) RMSCE evolution over time. Lower values indicate better uncertainty calibration.



(b) CRPS evolution over time. Lower values reflect sharper and more accurate forecasts.

Figure 21: Evolution of uncertainty metrics over time for the Combined dataset. (a) RMSCE and (b) CRPS demonstrate that ENMA outperforms baselines in both calibration and predictive sharpness.

**Data generation** As a second demonstration of ENMA’s generative capabilities, we evaluate its ability to produce plausible and diverse trajectories *without conditioning on the initial state*  $\mathbf{u}^0$ .

Specifically, we consider the setting of *generation conditioned on a context example*. Unlike classical neural approaches that typically require an initial condition and generate future states given  $\gamma$ , this setup assumes no access to either  $\mathbf{u}^0$  or  $\gamma$ . Instead, the model is provided with a context trajectory from which it must infer the underlying parameter  $\gamma$  and generate a coherent trajectory, including a plausible initial condition. This setting highlights ENMA’s capacity to function as a generative solver for parametric PDEs.

To benchmark generative performance, we compare ENMA against baseline models using standard metrics from the generative modeling literature. We introduce the *Fréchet Physics Distance* (FPD), a physics-adapted variant of Fréchet Inception Distance (FID) Heusel et al. (2017), along with Precision and Recall Kynkäänniemi et al. (2019). These metrics are computed in a compressed feature space extracted by a lightweight CNN encoder trained to regress the underlying PDE parameters  $\gamma$ , treated as instance classes.

The CNN processes the full trajectory and encodes it into a 64-dimensional feature vector, enabling semantically meaningful comparisons while mitigating the curse of dimensionality. In this space, FPD estimates the 2-Wasserstein distance between real and generated distributions, while Precision and Recall respectively measure sample fidelity and diversity. Following standard protocol, we use the training set as the reference distribution. Results for the Combined dataset are reported in table 13.

Table 13: Comparison of generative (FPD, Precision and Recall) metrics on the Combined dataset.

Model	FPD ↓	Precision ↑	Recall ↑
Zebra	0.1029	0.7700	<b>0.8612</b>
ENMA	<b>0.0095</b>	<b>0.7917</b>	0.7802

**Results** ENMA achieves the lowest FPD, significantly outperforming Zebra— indicating that ENMA’s generated trajectories are statistically closer to the real data distribution in the learned feature space. This reflects ENMA’s ability to generate high-fidelity samples that are well-aligned with physical ground truth. In terms of Precision, ENMA also outperforms Zebra, demonstrating superior sample quality. However, Zebra obtains a higher Recall, suggesting that it covers a broader range of modes from the data distribution. This highlights a trade-off: ENMA prioritizes fidelity and realism, whereas Zebra exhibits slightly more diversity, potentially at the cost of precision. Overall, these results illustrate ENMA’s strength in producing high-quality, physically plausible trajectories, while maintaining reasonable diversity in its generative predictions.

**Latent distribution alignment** To further analyze the generative behavior of ENMA and Zebra, we visualize PCA projections of feature representations extracted by a CNN trained on ground truth data. As shown in fig. 22, ENMA’s samples (orange) closely align with the real data (blue), indicating that the generated trajectories remain well-covered within the training distribution. In contrast, Zebra exhibits a large number of outliers that fall outside the support of the real data distribution. This mismatch suggests poorer calibration and lower fidelity to the training dynamics, which aligns with the lower FPD and precision scores reported in table 13. These findings confirm ENMA’s ability to generate physically plausible and distributionally consistent trajectories.

**Fidelity with respect to the numerical solver** To evaluate the fidelity of ENMA’s generations, we take advantage of the known parametric structure of the PDEs. For each sample, we condition ENMA on a context trajectory and let it generate a full trajectory, including the initial state. Since the true PDE parameters  $\gamma$  used to generate the context are known, we can pair ENMA’s generated initial condition with the ground-truth  $\gamma$  and run the numerical solver used during dataset creation. This produces a reference trajectory governed by the true physical dynamics. By comparing ENMA’s generated trajectory with the solver-based rollout, we can assess how well ENMA captures the underlying PDE. A close match indicates that ENMA has learned to infer physically consistent initial conditions and dynamics from context.

**Results** As shown in fig. 23, the trajectories produced by ENMA closely match those from the solver, both qualitatively and quantitatively. We also report a relative L2 error of **0.081** between the



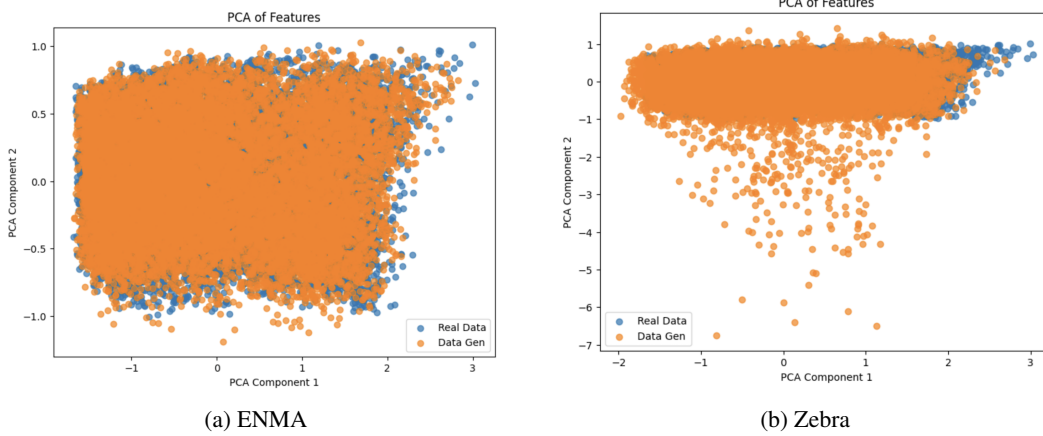


Figure 22: PCA projections of CNN features from generated (orange) and real (blue) trajectories at the final timestep.

1466 trajectories generated by ENMA and by the solver, demonstrating that ENMA’s generated states are not only coherent but also physically meaningful.

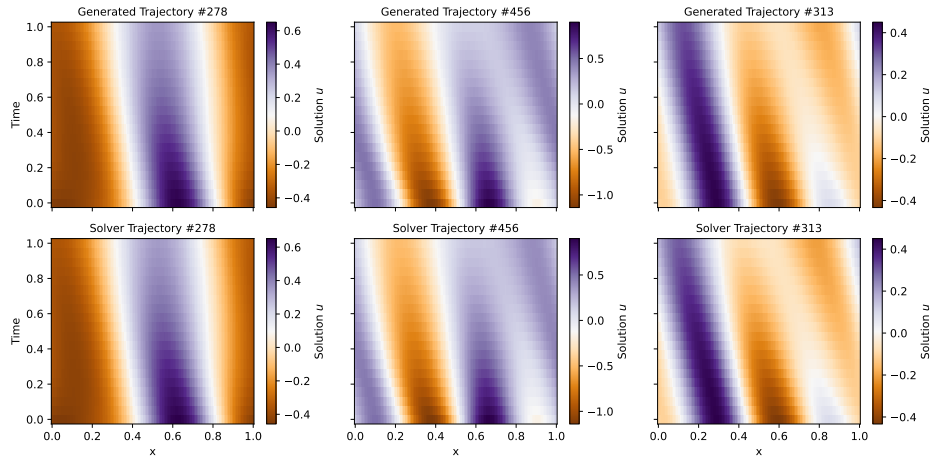


Figure 23: Comparison between trajectories generated by ENMA (top row) and the corresponding rollouts obtained from the PDE solver using the generated initial condition (bottom row). ENMA’s predictions yield physically consistent rollouts.

1467

## 1468 F.1.2 Dynamics forecasting with less compression

1469 As discussed in section 4.3, ENMA exhibits lower performance on the Vorticity dataset compared to  
 1470 baseline models. We hypothesize that this is due to the aggressive latent compression, which limits  
 1471 the VAE’s ability to capture fine-scale, high-frequency structures—ultimately constraining generation  
 1472 quality at decoding time. To investigate this, we evaluate ENMA under reduced compression settings.  
 1473 While the main experiments used a spatial compression factor of 4 for Vorticity, we report additional  
 1474 results on 2D datasets in the temporal conditioning setting, comparing against AVIT and Zebra with a  
 1475 relaxed compression factor.

1476 **Results** Table 14 shows results on Gray-Scott, Wave, and Vorticity under relaxed compression to  
 1477 match baseline settings. This setup addresses ENMA’s lower performance on Vorticity observed in  
 1478 section 4.3, attributed to overly aggressive compression. ENMA outperforms AVIT and Zebra on  
 1479 Gray-Scott and Wave across both In-D and Out-D settings. On Vorticity, ENMA remains competitive,



Table 14: Comparison of model performance on Gray-Scott, Wave, and Vorticity under the temporal conditioning setting. Metrics in Relative MSE ( $\downarrow$ ).

Model	Gray-Scott		Wave		Vorticity	
	In-D	Out-D	In-D	Out-D	In-D	Out-D
AVIT	0.0426	0.168	0.157	0.588	0.176	0.377
Zebra	0.0421	0.182	0.140	0.315	<b>0.0443</b>	<b>0.223</b>
ENMA	<b>0.0223</b>	<b>0.152</b>	<b>0.070</b>	<b>0.283</b>	0.0521	0.282

1480 closely matching Zebra. These results highlight ENMA’s strong predictive performance under  
1481 comparable constraints.

### 1482 F.1.3 Inference speed against generative approaches

1483 One of the primary limitations of generative models compared to deterministic surrogates lies  
1484 in their inference speed. Diffusion-based methods, for instance, require multiple iterative passes  
1485 through a large model—typically a Transformer—to generate a single sample, resulting in high  
1486 computational cost. Similarly, autoregressive (AR) approaches like (Serrano et al., 2024a) generate  
1487 tokens sequentially, which becomes especially expensive when modeling high-dimensional spatio-  
1488 temporal data.

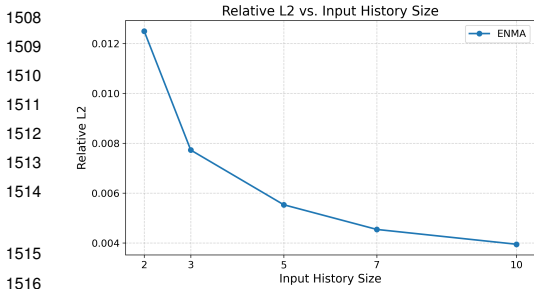
1489 ENMA addresses these inefficiencies by combining the strengths of both paradigms. It adopts an AR  
1490 framework that supports multi-token generation and leverages key-value caching for faster inference,  
1491 while also benefiting from the expressiveness of generative methods through its use of Flow Matching.  
1492 Crucially, rather than relying on a large model at each step, ENMA performs flow matching through a  
1493 lightweight MLP, substantially reducing the computational overhead compared to traditional diffusion  
1494 models.

1495 **Results** We compare the inference time per sample for AR-DiT, Zebra, and ENMA in table 15.  
1496 ENMA achieves the fastest inference across both the Combined and Vorticity datasets, with an average  
1497 of 2s and 4.5s per sample, respectively. This represents a 2× speedup over AR-DiT and a 3×  
1498 speedup over Zebra on Combined. Zebra particularly exhibits slower inference on the Vorticity  
1499 dataset where it reaches 31s per sample. These results highlight the efficiency of ENMA’s continuous  
1500 latent autoregressive generation, which scales favorably compared to existing generative baselines.

Table 15: Inference speed comparison on the Combined and Vorticity datasets. Lower is better.

Model	Combined	Vorticity
AR-DiT	4s	6.7s
Zebra	6s	31s
ENMA	<b>2s</b>	<b>4.5s</b>

### 1507 F.1.4 Ablation Studies



1517 Figure 24: Relative L2 error vs. input history size.  
1518 ENMA benefits from longer histories.

1519  
1520 tion of the number of past frames provided as context and evaluate the predictions quality on the last  
1521 10 steps of the trajectory. As shown in fig. 24, the relative L2 error consistently decreases as the input

To better understand the behavior and flexibility of our generative framework, we conduct ablations examining ENMA’s performance across three key dimensions: (i) sensitivity to the length of the input history, (ii) the number of autoregressive spatial steps  $S$ , and (iii) the number of Flow Matching (FM) steps during generation.

**Impact of Input History Length** In many real-world scenarios, the number of available historical observations varies, making it important for a model to adapt seamlessly to different history lengths. We evaluate ENMA’s predictive performance as a function

history length increases. This trend confirms that ENMA effectively leverages additional temporal context to refine its predictions. Notably, the model already achieves competitive performance with as few as 2–3 input frames and continues to improve as more information is made available. This highlights the model’s capacity for conditional generalization across varying input regimes.

**Impact of the Number of Autoregressive Steps** The number of autoregressive steps  $S$  directly impacts inference efficiency: fewer steps accelerate generation, but may reduce accuracy. In the case of the Combined equation, each spatial state of 128 points is compressed into 16 latent tokens. We evaluate ENMA’s performance as we vary  $S$  from 1 to 16 in fig. 25, where  $S = 1$  corresponds to generating all tokens at once, and  $S = 16$  to generating one token per step.

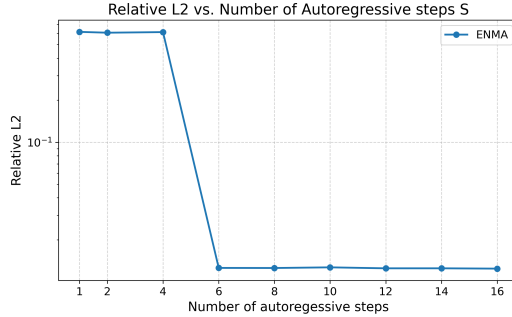


Figure 25: Relative L2 error as a function of the number of autoregressive steps  $S$ . Performance improves markedly around  $S = 6$  and plateaus thereafter.

We observe that performance is poor when  $S \leq 4$ , but significantly improves at  $S = 6$ , after which it quickly plateaus. This behavior reflects the masking ratio used during training: the model is trained to reconstruct frames with 75%–100% of tokens masked. Hence, inference scenarios where a majority of tokens are already visible (i.e., large  $S$  values) fall outside the model’s training regime, which may limit gains from further increasing  $S$ . These results suggest that a modest number of autoregressive steps—around 6—balances speed and accuracy. Expanding the training schedule to include lower masking ratios could further improve performance for large  $S$ , but would increase training time due to the larger space of masking patterns.

**Impact of the Number of Flow Matching Steps** The number of flow matching (FM) steps governs the granularity of sampling from the learned conditional distribution at each autoregressive step. While more FM steps can, in principle, yield smoother and more accurate trajectories, they also increase inference cost—even if the MLP used for token decoding remains lightweight. In fig. 26, we evaluate ENMA’s performance as a function of FM steps. We observe a sharp drop in relative L2 error between 2 and 5 steps, after which the performance plateaus. Beyond 10 steps, improvements are marginal or even slightly inconsistent. This indicates that ENMA captures most of the necessary detail with very few sampling steps.

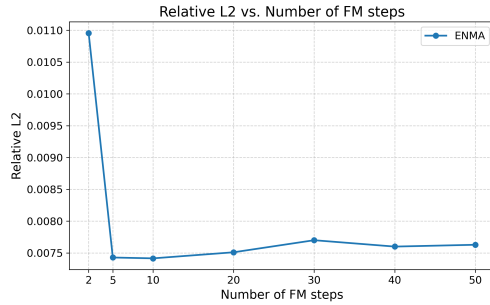


Figure 26: Relative L2 error versus number of flow matching steps per token. Performance quickly stabilizes after 5 steps.

From a practical standpoint, this suggests that using as few as 5 FM steps can offer an optimal trade-off between speed and accuracy, making ENMA efficient at inference without compromising quality.

## F.2 Experiments on the encoder/decoder

### F.2.1 OOD encoding on 10% of the input grid

We evaluate the performance of the auto-encoder in an out-of-distribution (OOD) reconstruction setting by reducing the input grid size to  $\pi = 10\%$  of the original spatial grid. This requires models to reconstruct the full field from very sparse observations—only 12 points on the Advection dataset and 409 points on the Vorticity dataset. Notably, such extreme sparsity was not encountered during training, where input sampling ratios ranged from  $\pi = 20\%$  to  $\pi = 100\%$ .

Table 16: **Reconstruction error** using  $\pi = 10\%$  of the initial grid Test results. Metrics in Relative MSE.

$\downarrow \mathcal{X}_{te}$	Dataset $\rightarrow$ Model $\downarrow$	<i>Advection</i>	<i>Vorticity</i>
		Reconstruction	Reconstruction
$\pi = 10\%$	OFormer	5.82e-1	1.00
	GINO	<u>2.63e-1</u>	6.46e-1
	AROMA	4.69e-1	<u>3.28e-1</u>
	CORAL	1.13	1.38
	ENMA	<b>2.45e-1</b>	<b>2.78e-1</b>

1574 This reconstruction experiment demonstrates that ENMA maintains superior reconstruction quality  
1575 even when provided with extremely sparse input fields. Among the baselines, AROMA and GINO  
1576 also exhibit robust performance under such challenging conditions. In contrast, OFormer and CORAL  
1577 fail to reconstruct the physical field, highlighting their limited generalization in low-data regimes.

## 1578 F.2.2 Super-resolution

1579 In this section, we assess the models' ability to generalize to finer spatial grids. For the Advection  
1580 dataset, we evaluate reconstructions starting from input subsamplings of  $\pi = 20\%, 50\%, 100\%$ ,  
1581 and query the model on denser grids with  $\pi_{sr} = 200\%, 400\%, 800\%$  of the original resolution. On  
1582 the Vorticity dataset, we use the same input subsamplings but perform super-resolution only at  
1583  $\pi_{sr} = 200\%$  due to data availability constraints.

1584 For comparison, we also report reconstruction results at  $\pi_{sr} = 100\%$ , as in table 1, and present the  
1585 full results in table 17. The table is structured as follows: rows correspond to the input grid ratio  $\pi$ ,  
1586 matching the setup of table 1, while columns indicate the relative query grid size  $\pi_{sr}$ . For instance, on  
1587 the Advection dataset with an original grid size of 128, the second row ( $\pi = 50\%$ ) corresponds to 64  
1588 input points, and the third column ( $\pi_{sr} = 400\%$ ) corresponds to querying 512 points. Other rows and  
1589 columns follow the same interpretation.

Table 17: **Reconstruction error** on super resolution task  
Test results. Metrics in Relative MSE.

$\downarrow \mathcal{X}_{te}$	Dataset $\rightarrow$ $\pi_{sr} \rightarrow$ Model $\downarrow$	<i>Advection</i>				<i>Vorticity</i>	
		100%	200%	400%	800%	100%	200%
$\pi = 100\%$	OFormer	1.70e-1	5.20	5.19	5.19	9.99e-1	1.00
	GINO	5.74e-2	7.77e-2	8.83e-2	9.43e-2	5.63e-1	5.76e-1
	AROMA	<u>5.41e-3</u>	3.78e-2	<u>5.62e-2</u>	6.54e-2	<u>1.45e-1</u>	<u>1.71e-1</u>
	CORAL	1.34e-2	4.00e-2	5.76e-2	6.66e-2	4.50e-1	4.55e-1
	ENMA	<b>1.83e-3</b>	<b>3.71e-2</b>	<b>5.56e-2</b>	<b>6.49e-2</b>	<b>9.20e-2</b>	<b>1.36e-1</b>
$\pi = 50\%$	OFormer	1.79e-1	5.20	5.19	5.18	9.99e-1	1.00
	GINO	6.64e-2	8.38e-2	9.37e-2	9.95e-2	5.69e-1	5.82e-1
	AROMA	<u>2.34e-2</u>	4.44e-2	<u>6.09e-2</u>	6.94e-2	<u>1.64e-1</u>	<u>1.89e-1</u>
	CORAL	7.57e-2	8.80e-2	9.96e-2	1.06e-1	4.93e-1	4.98e-1
	ENMA	<b>4.60e-3</b>	<b>3.74e-2</b>	<b>5.59e-2</b>	<b>6.51e-2</b>	<b>9.90e-2</b>	<b>1.41e-1</b>
$\pi = 20\%$	OFormer	2.50e-1	5.18	5.17	5.16	9.99e-1	1.00
	GINO	<u>9.13e-2</u>	<u>1.04e-1</u>	<u>1.12e-1</u>	<u>1.17e-1</u>	5.90e-1	6.01e-1
	AROMA	1.67e-1	1.72e-1	1.78e-1	1.81e-1	<u>2.29e-1</u>	2.45e-1
	CORAL	4.77e-1	4.79e-1	4.82e-1	4.84e-1	7.59e-1	7.62e-1
	ENMA	<b>3.05e-2</b>	<b>4.94e-2</b>	<b>6.49e-2</b>	<b>7.31e-2</b>	<b>1.37e-1</b>	<b>1.69e-1</b>

1590 **Results** As shown in table 17, all models exhibit performance degradation under the super-resolution  
1591 setting. Despite this, ENMA consistently outperforms other encoder-decoder architectures across all

resolutions. CORAL demonstrates stable performance as output resolution increases, while OFormer struggles significantly when queried on unseen grids for both datasets. ENMA and AROMA show similar trends as the super-resolution difficulty increases; however, AROMA’s performance degrades more rapidly with lower input grid densities. Overall, ENMA’s auto-encoder proves to be the most robust, both to variations in input sparsity and to changes in query resolution.

### F.2.3 Ablation studies

We conduct two additional ablation studies to evaluate specific architectural components of the encoder–decoder framework: (i) the effect of using a geometry-aware attention bias (table 18), and (ii) the impact of the time-stepping process used in the decoder (table 19).

**Geometry-aware attention bias.** Table 18 reports the reconstruction error when ENMA is trained with and without the geometry-aware attention bias described in appendix D.1. This positional bias encourages spatial locality in attention by penalizing interactions between distant query–key pairs. The results show consistent improvements across all input sparsity levels, with relative gains ranging from 25% to 40%.

Table 18: **Reconstruction error** with and without geometry-aware attention bias. Metrics are Relative MSE. Relative improvement is reported in parentheses.

$\pi$	Input Ratio	Model Variant	Reconstruction ↓
100%	Full	w/o bias	3.09e-3
		w/ bias	<b>1.83e-3</b> (-40%)
50%	Half	w/o bias	6.84e-3
		w/ bias	<b>4.60e-3</b> (-33%)
20%	Sparse	w/o bias	4.13e-2
		w/ bias	<b>3.05e-2</b> (-26%)

**Time-stepping architecture.** We further examine the impact of the time-stepping module by replacing the FNO used in ENMA with a 4-layer U-Net. Both models use identical lifting and projection layers to match the latent token dimensions. The U-Net adopts convolutional layers with kernel size 3, stride 1, and padding 1.

Table 19 summarizes the reconstruction performance on the Advection dataset across three input sparsity levels. We observe that ENMA maintains superior accuracy under both architectures. While the U-Net variant exhibits slightly degraded performance compared to the FNO, it remains competitive and stable. A dash (–) indicates that the model diverged and produced NaNs during training.

These results confirm the robustness of ENMA across different architectural choices for the time-stepping component and further emphasize the benefits of the geometry-aware attention mechanism.

Table 19: **Reconstruction error** for different time-stepping processes (FNO vs U-Net) on the Advection dataset. Metrics are Relative MSE.

$\pi$	Model	FNO	U-Net
100%	OFormer	1.11	1.54
	GINO	7.89e-1	7.52e-1
	AROMA	<u>2.23e-1</u>	<u>6.43e-1</u>
	CORAL	9.64e-1	–
	ENMA	<b>1.64e-1</b>	<b>3.51e-1</b>
50%	OFormer	1.11	1.04
	GINO	7.87e-1	7.50e-1
	AROMA	<u>2.29e-1</u>	<u>6.49e-1</u>
	CORAL	9.74e-1	–
	ENMA	<b>1.72e-1</b>	<b>3.58e-1</b>
20%	OFormer	1.13	1.03
	GINO	7.96e-1	7.53e-1
	AROMA	<u>3.21e-1</u>	<u>7.32e-1</u>
	CORAL	1.06	–
	ENMA	<b>3.13e-1</b>	<b>4.15e-1</b>

## 1629 F.2.4 Geometry-Aware Attention Analysis

1630 We visualize the attention mechanism in ENMA’s encoder-decoder using the Advection dataset to  
 1631 better understand the effect of the geometry-aware attention bias.

1632 **Attention Bias** Figure 28 displays the geometry-aware bias  $B$   
 1633 introduced in Section D.1. Each plot corresponds to one attention  
 1634 head, with rows representing different input sizes (128, 64, and 25  
 1635 points from top to bottom) and columns corresponding to different  
 1636 heads. Bias values near zero (white) indicate minimal distance be-  
 1637 tween query and key coordinates, promoting attention, while larger  
 1638 distances produce strongly negative biases (dark blue), which sup-  
 1639 press it. Because the input grid is irregular, attention patterns in the  
 1640 second and third rows are not strictly diagonal. Additionally, since  
 1641 the intermediate coordinates are learned (see fig. 27), the model is  
 1642 not constrained to maintain a perfectly regular grid, which further  
 1643 contributes to these deviations. Notably, the later attention heads  
 1644 appear to penalize distant tokens more strongly—an effect explained  
 1645 by the learned head-specific scaling factors, following the approach  
 1646 of Press et al. (2022).

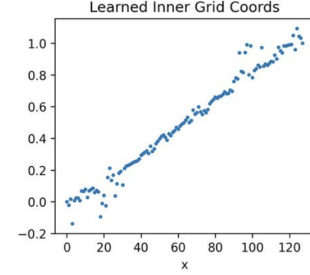


Figure 27: Learned coordinates of the intermediate regular grid.

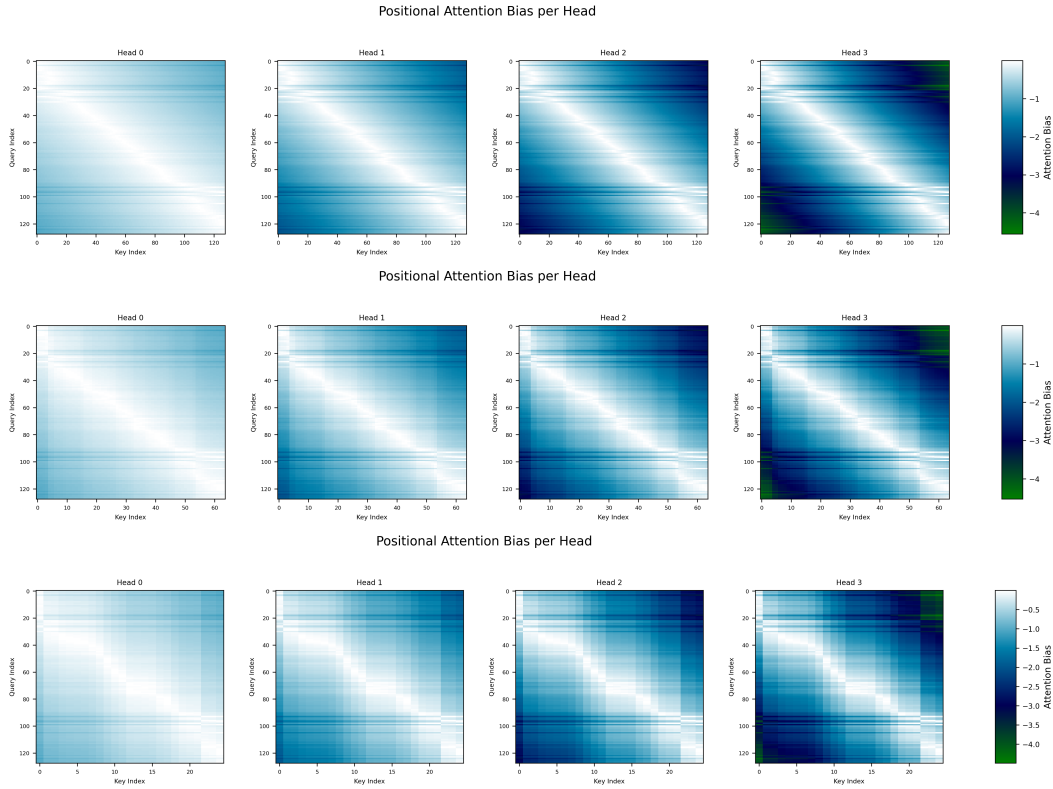


Figure 28: Geometry-aware attention bias  $B$  for each attention head, shown for 128 (top), 64 (middle), and 25 (bottom) input points.

1647 **Attention Scores** Figures 29 and 30 present the actual attention scores produced using the geometry-  
 1648 aware bias. These visualizations confirm that attention is concentrated on nearby points, enforcing a  
 1649 spatially local inductive bias during interpolation. We provide visualization for the advection dataset  
 1650 fig. 29 and vorticity fig. 30.

1651 These results demonstrate that the attention mechanism learns to prioritize spatially proximate inputs,  
 1652 even under varying levels of spatial sparsity.

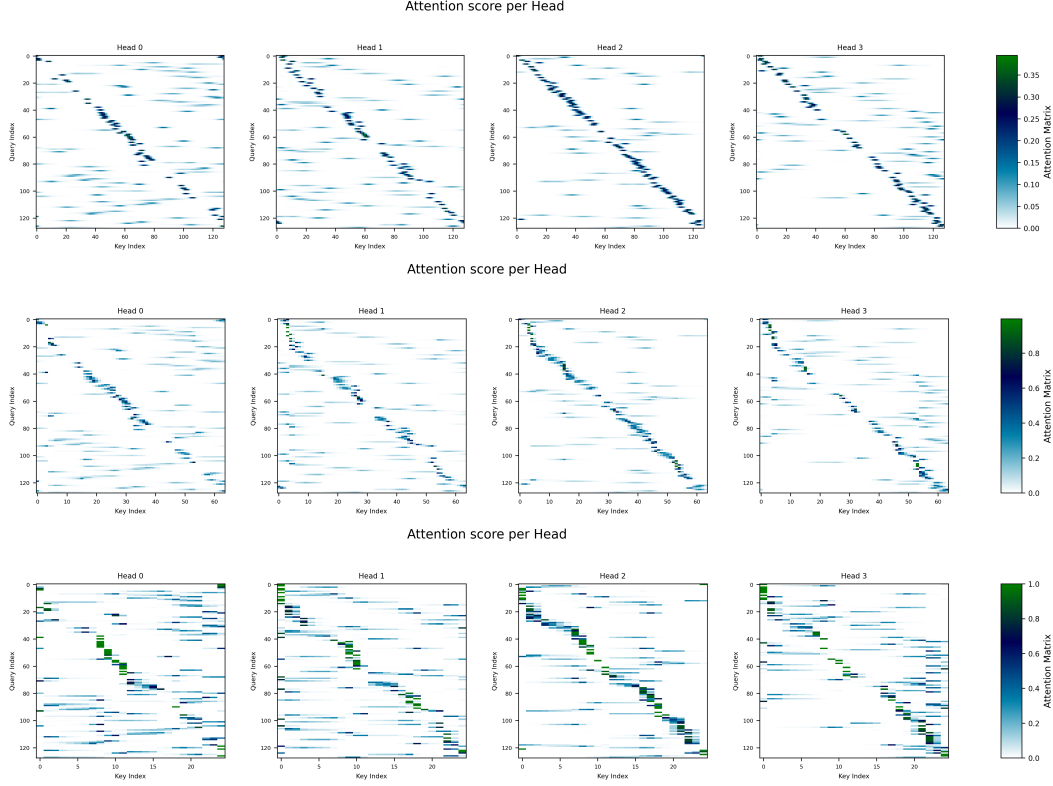


Figure 29: Attention scores per head using the geometry-aware bias for 128 (top), 64 (middle), and 25 (bottom) input points.

**Attention visualization in the physical space** We visualize the final attention weights in physical space in fig. 30. For a selected timestep of a given trajectory, we choose a set of spatial tokens (left panel) and display their corresponding attention scores across the physical domain (middle panel). The resulting patterns clearly demonstrate that the cross-attention mechanism, guided by the geometry-aware bias, preserves locality by assigning high attention weights to nearby regions—effectively focusing on spatially relevant information.

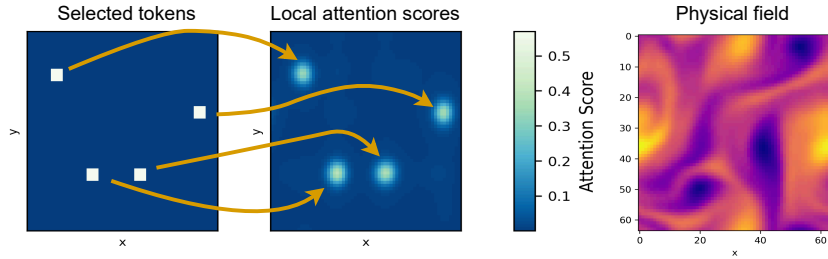


Figure 30: Attention scores on the vorticity dataset, on spatially selected tokens (left) at a given timestep ( $t = 15$ ). The resulting attention score on the output grid keeps the local behavior (middle). Attention score are averaged across heads for visualization. The left figure is the physical field considered.

## F2.5 Visualization in the Token Space

Figure 31 shows the latent token representations on the Advection dataset for varying input sparsity (128, 64, and 25 points from top to bottom). The final column displays the corresponding ground-truth physical trajectory.



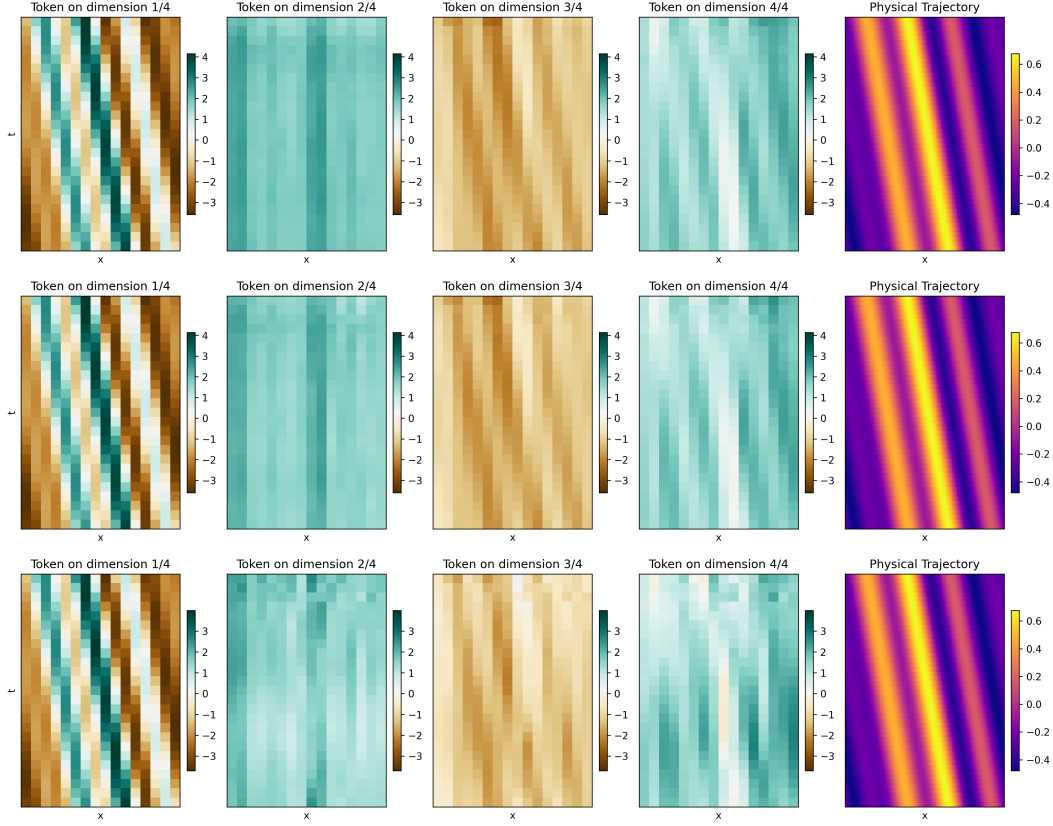


Figure 31: Token representation when using 128 (top), 64 (middle), and 25 (bottom) input points.

1663 The tokens effectively capture the dynamics in a compressed form, demonstrating the encoder’s strong  
 1664 inductive bias. Notably, the first latent dimension remains stable across input resolutions, encoding  
 1665 coarse global structure. In contrast, the remaining dimensions vary with input density, indicating  
 1666 progressive refinement of local details—a sign of partial disentanglement across token dimensions.  
 1667 This behavior supports the model’s ability to encode structure hierarchically and adaptively under  
 1668 compression.

1669 **Intermediate Fields in the Encoder–Decoder** Figure 32 visualizes the full trajectory of latent  
 1670 transformation within ENMA’s encoder–decoder. Each row corresponds to a processing stage,  
 1671 averaged across channels.

1672 Starting from a sparse, irregular input field (top row,  $\pi = 20\%$ ), the first cross-attention layer projects  
 1673 the observations onto a learned latent grid (row 2), effectively interpolating the missing spatial points.  
 1674 The causal CNN then encodes the trajectory into a compact token space (row 3), capturing the  
 1675 trajectory’s temporal evolution in a lower-dimensional representation. These tokens are subsequently  
 1676 upsampled (row 4) and decoded onto the full spatial grid (row 5). The final reconstruction is  
 1677 smooth and well-aligned with the underlying dynamics, highlighting the effectiveness of ENMA’s  
 1678 encoder–decoder architecture. Minor artifacts observed in intermediate stages—likely caused by  
 1679 learned positional embeddings (Yang et al., 2024)—are effectively corrected during decoding.

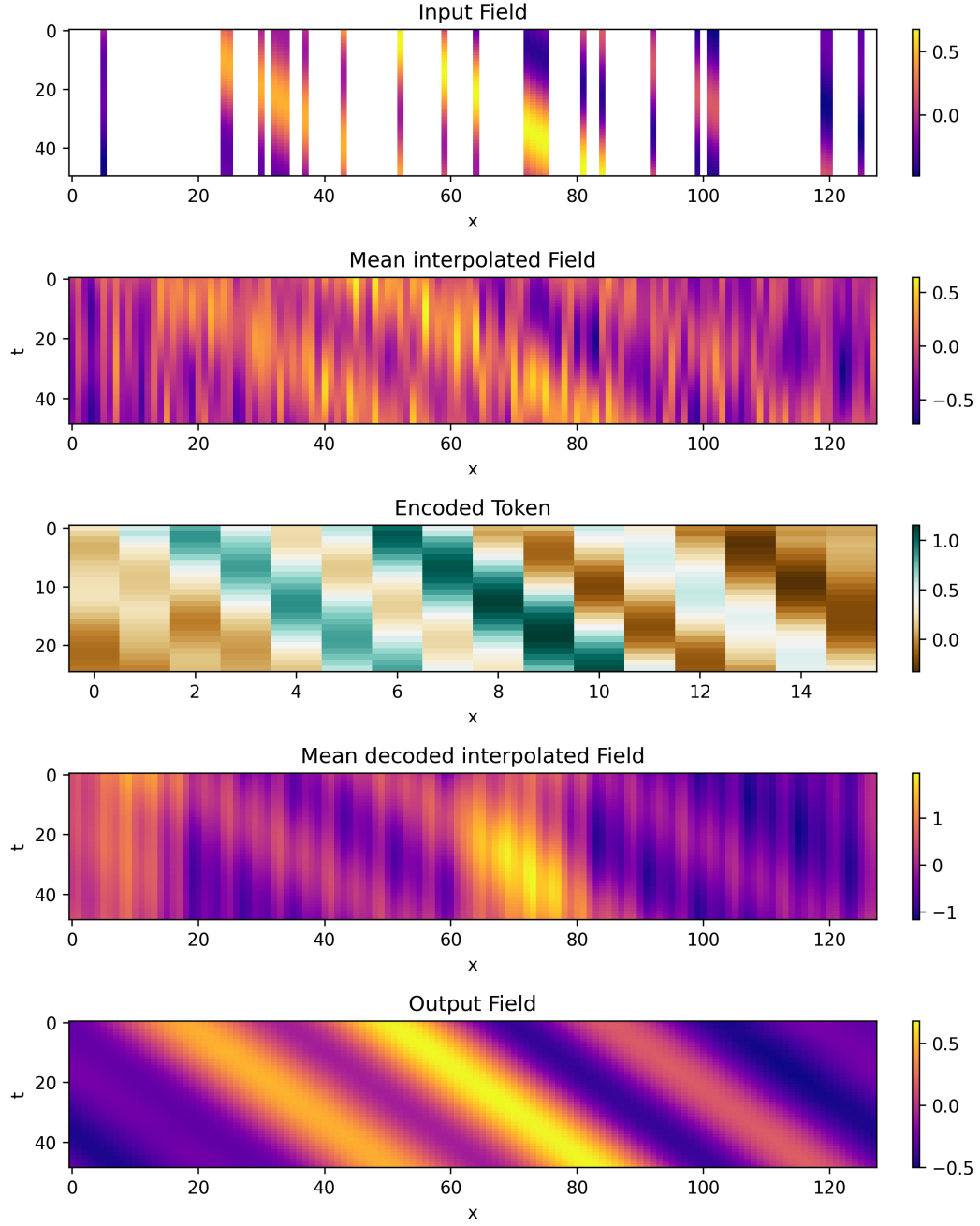


Figure 32: Intermediate fields in the encoder/decoder blocks: interpolation (rows 1–2), compression (rows 2–3), upsampling (rows 3–4), and reconstruction (rows 4–5).



## 1680 G Visualization

### 1681 G.1 Combined Equation

1682 Figure 33 and Figure 34 provide qualitative results on the Combined equation, showcasing ENMA's accuracy on in-distribution inputs and its generalization capability to out-of-distribution scenarios.

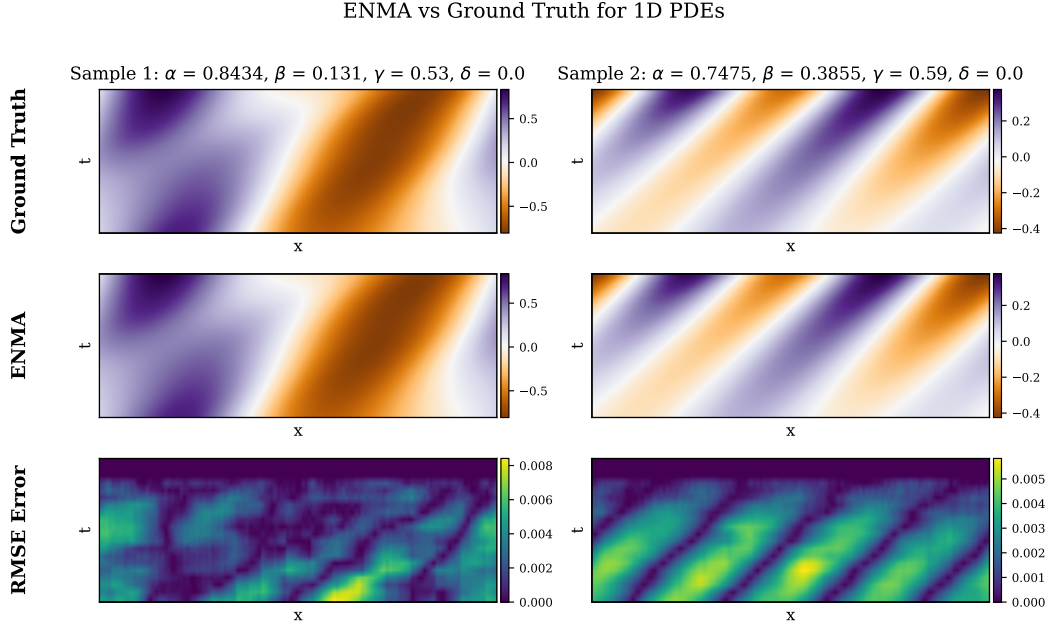


Figure 33: Qualitative comparison between ENMA prediction and ground truth for in-distribution examples from the Combined.

1683

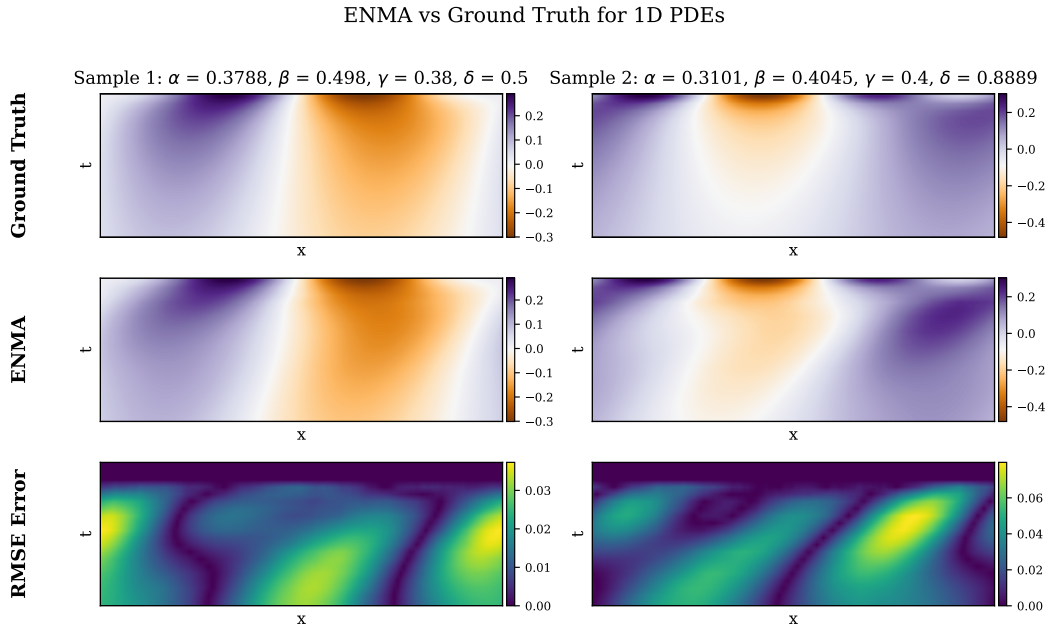


Figure 34: Qualitative comparison between ENMA prediction and ground truth for an OOD example from the Combined.

1684 **G.2 Advection Equation**

1685 Figure 35 and Figure 36 provide qualitative results on the Advection equation. These plots highlight  
 1686 ENMA's accurate forecasting on in-distribution samples and its ability to generalize to out-of-  
 1687 distribution scenarios.

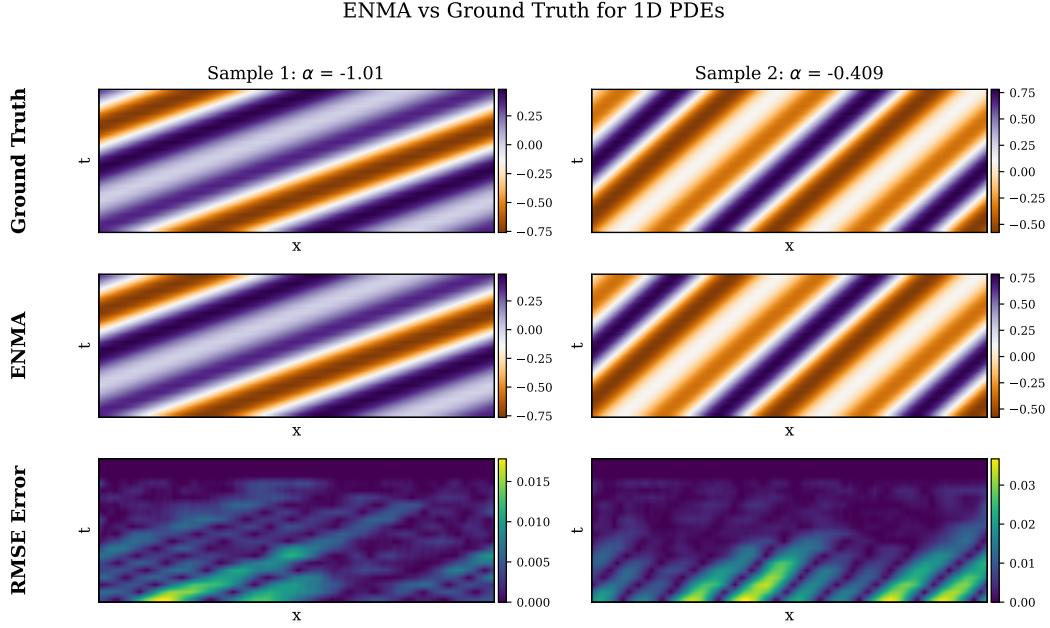


Figure 35: Qualitative comparison between ENMA prediction and ground truth for in-distribution examples from the Advection dataset.

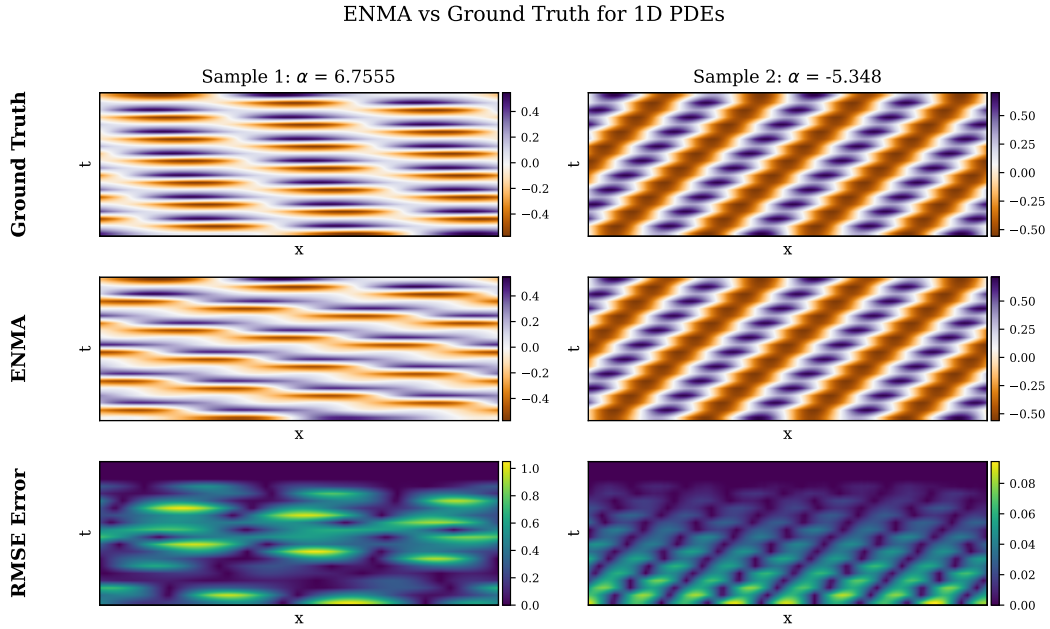


Figure 36: Qualitative comparison between ENMA prediction and ground truth for an OOD example from the Advection dataset.

### 1688 G.3 Gray-Scott Equation

1689 Figure 37, Figure 38, and Figure 39 present qualitative comparisons between ENMA and ground  
 1690 truth on the Gray-Scott equation. ENMA accurately reconstructs complex spatiotemporal patterns  
 1691 on in-distribution samples and demonstrates good qualitative performance on out-of-distribution  
 parameters.

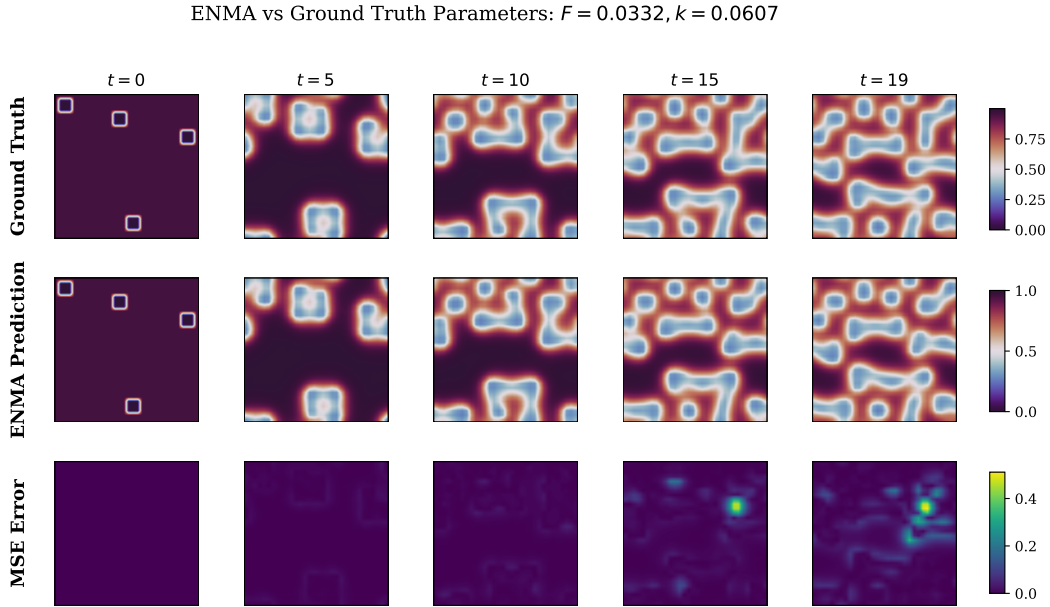


Figure 37: Qualitative comparison between ENMA prediction and ground truth for an in-distribution sample from the Gray-Scott dataset ( $F = 0.0323, k = 0.0606$ ).

1692

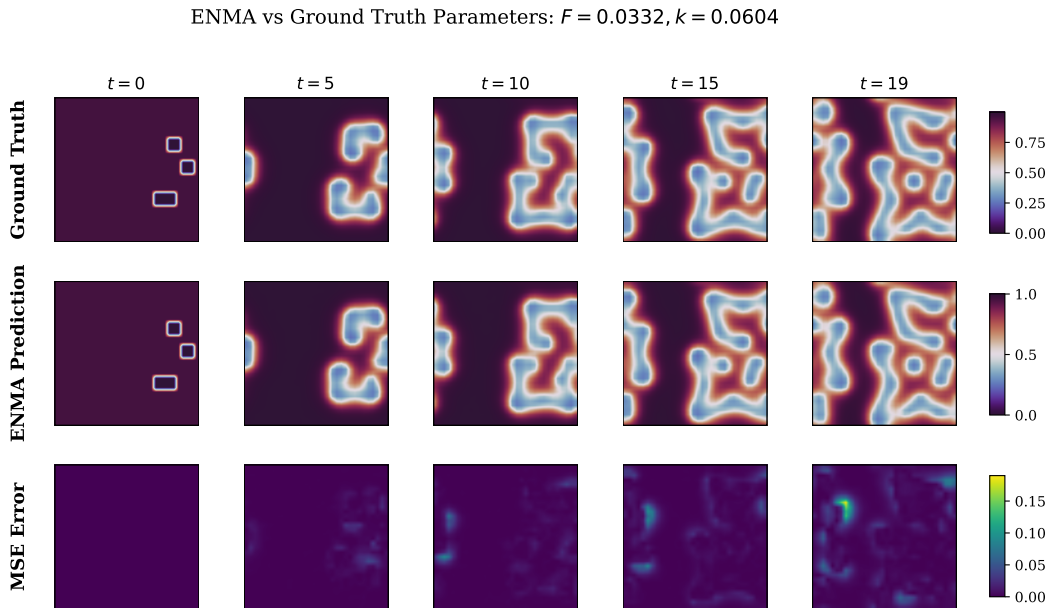


Figure 38: Another in-distribution Gray-Scott example showing the agreement between ENMA prediction and ground truth ( $F = 0.0316, k = 0.0597$ ).

ENMA vs Ground Truth Parameters:  $F = 0.0467, k = 0.058$

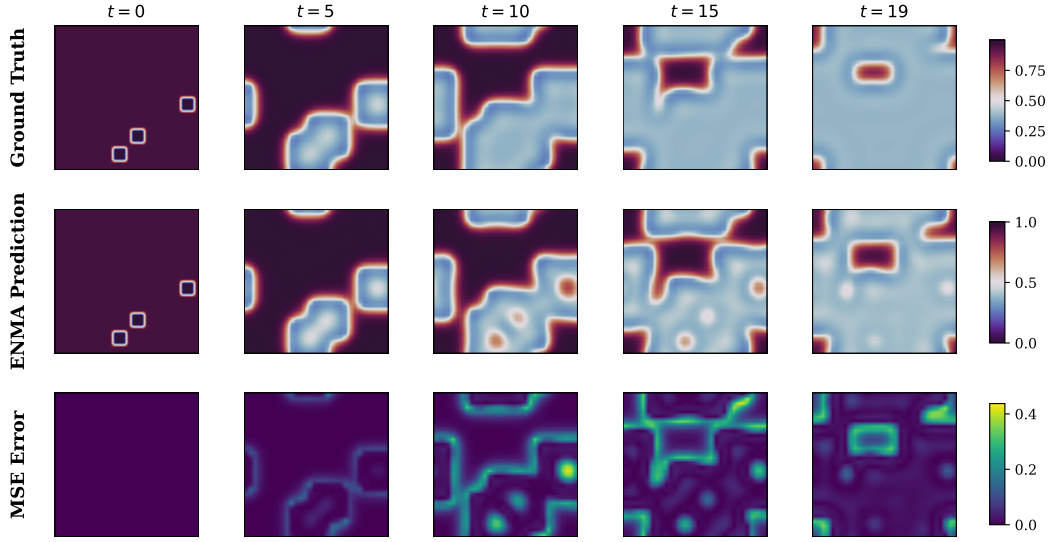


Figure 39: Out-of-distribution (OOD) generalization for the Gray-Scott equation. ENMA prediction remains consistent despite being evaluated at unseen parameters ( $F = 0.0467, k = 0.058$ ).

#### 1693 G.4 Wave Equation

1694 Figure 40, Figure 41, and Figure 42 present qualitative comparisons for the Wave equation. ENMA  
 1695 accurately captures wavefront propagation in in-distribution scenarios and generalizes well to out-of-distribution conditions.

ENMA vs Ground Truth Parameters:  $c = 140.404, k = 23.2323$

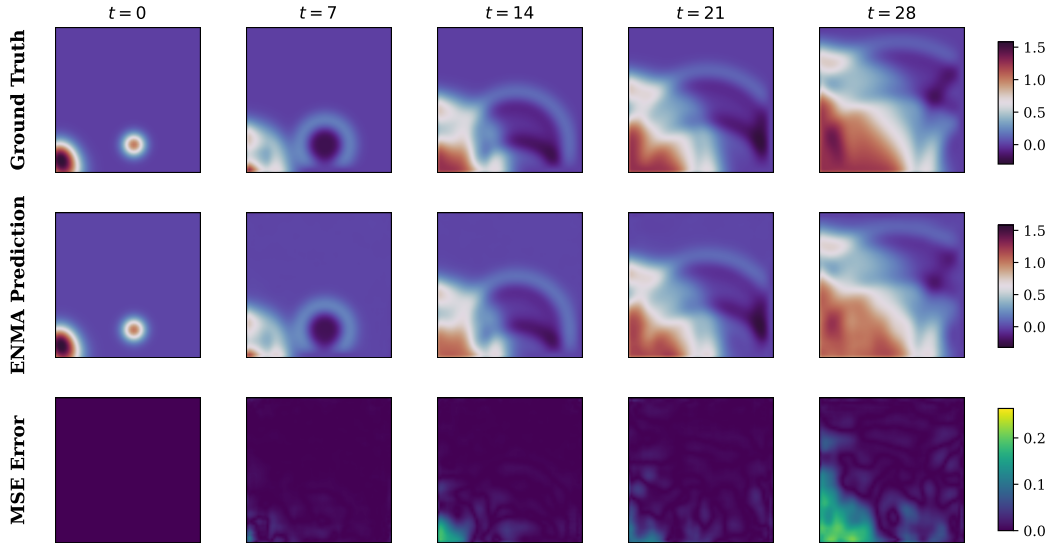


Figure 40: Qualitative comparison between ENMA prediction and ground truth for an in-distribution sample from the Wave dataset ( $c = 140.404, k = 23.2323$ ).

1696

ENMA vs Ground Truth Parameters:  $c = 144.4444, k = 21.2121$

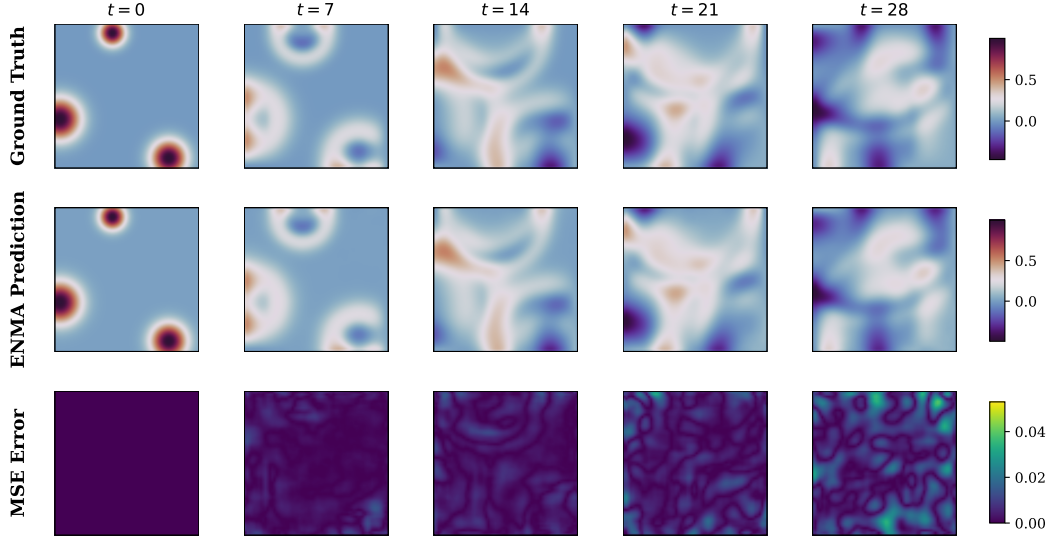


Figure 41: Second in-distribution sample from the Wave dataset ( $c = 144.4444, k = 21.2121$ ), showing ENMA's prediction, ground truth, and the corresponding MSE error.

ENMA vs Ground Truth Parameters:  $c = 500.0, k = 52.5$

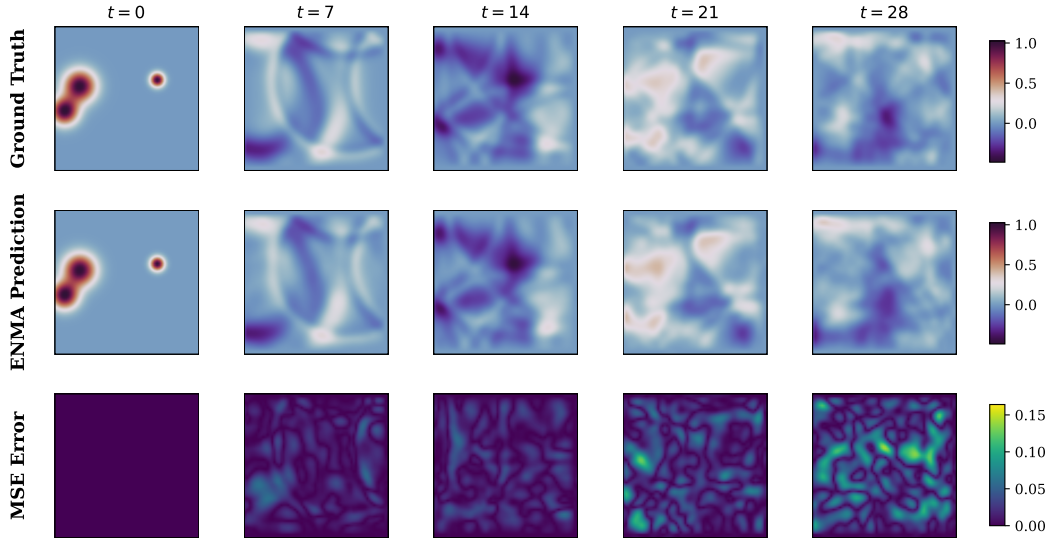


Figure 42: Out-of-distribution example from the Wave dataset ( $c = 500.0, k = 52.5$ ), demonstrating ENMA's generalization capabilities beyond the training regime.



## 1697 G.5 Vorticity Equation

1698 Figure 43, Figure 44, and Figure 45 show qualitative comparisons for the Vorticity equation. ENMA  
 1699 reliably captures fluid dynamics behavior on in-distribution samples and maintains accuracy in  
 1700 out-of-distribution regimes.

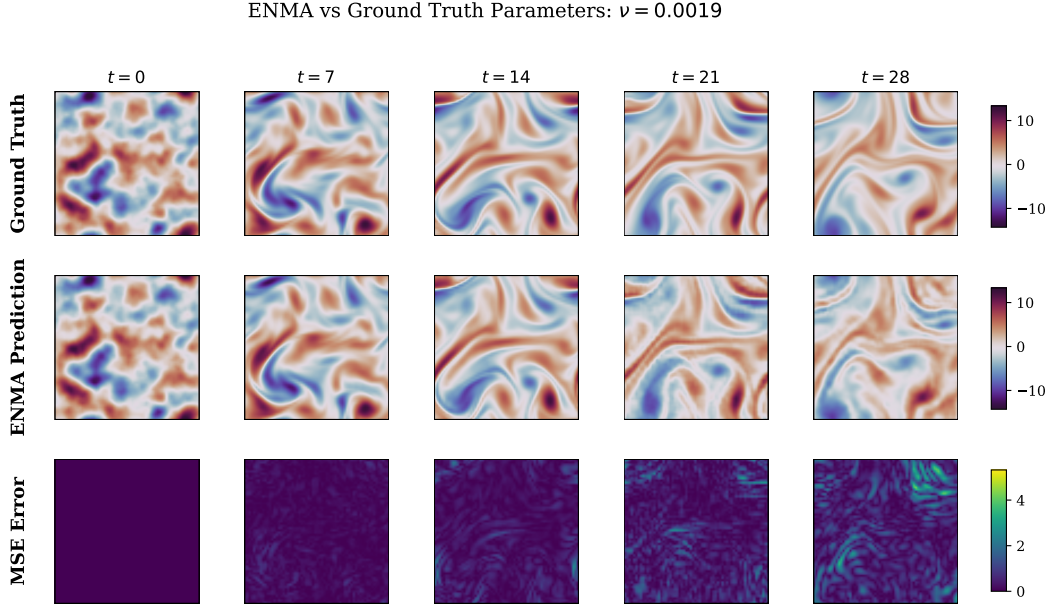


Figure 43: Qualitative comparison between ENMA prediction and ground truth for an in-distribution sample from the Vorticity dataset ( $\nu = 0.0019$ ).

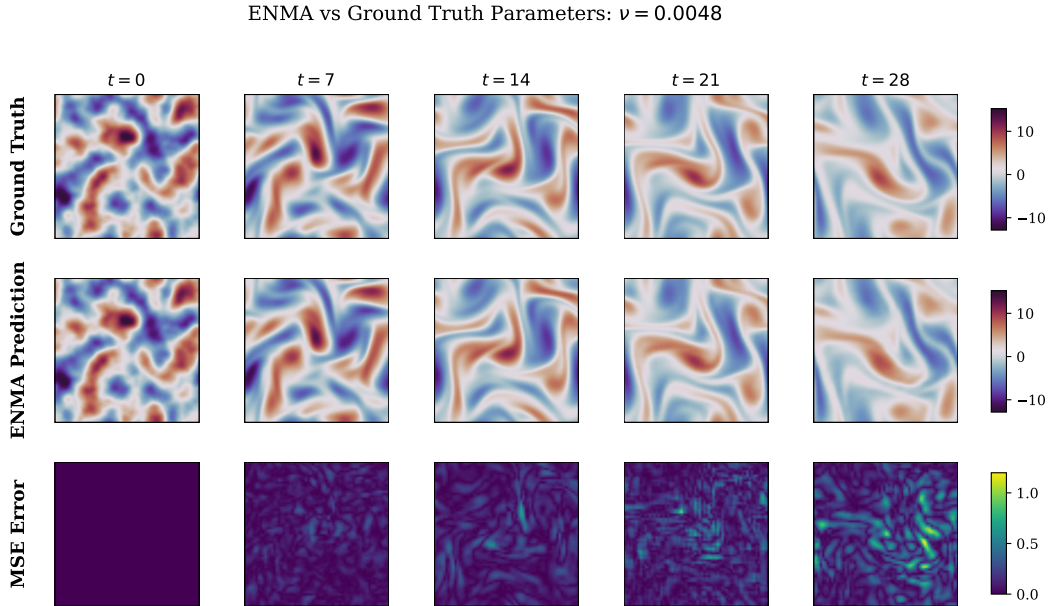


Figure 44: Second in-distribution sample from the Vorticity dataset ( $\nu = 0.0048$ ), illustrating ENMA's ability to reproduce complex vortex structures.

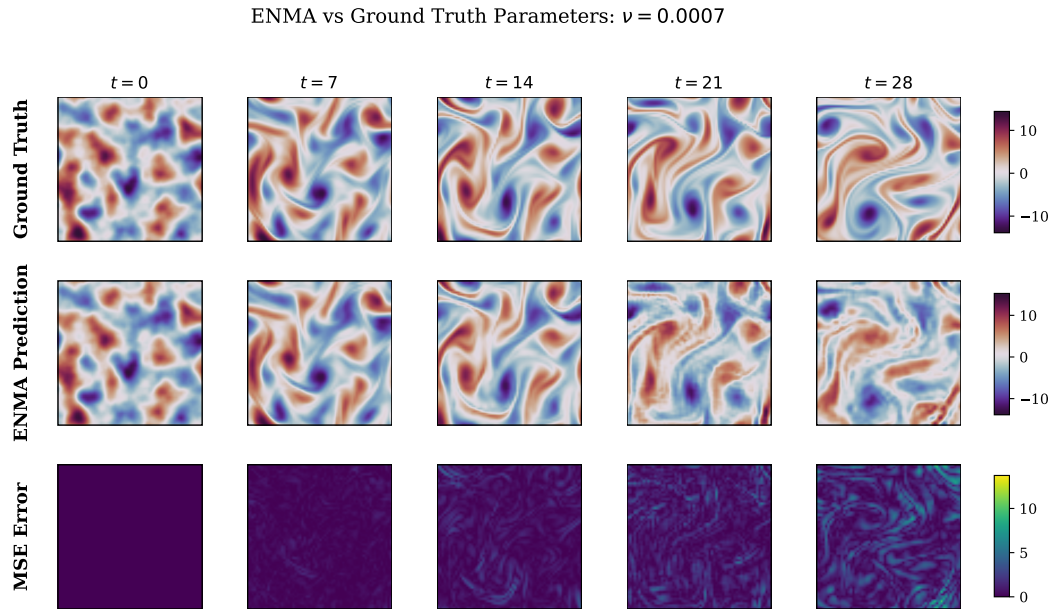


Figure 45: Out-of-distribution example from the Vorticity dataset ( $\nu = 0.0007$ ), highlighting ENMA's robustness in extrapolating vortex dynamics.