ZERO-SHOT GENERALIZATION OF TRANSFORMER NEURAL OPERATORS TO LARGER DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer-based neural operators have shown remarkable performances for approximating solution operators of partial differential equations on complex geometries. However, existing approaches implicitly assume a fixed domain size, which limits their ability to generalize at inference. In this work, we investigate domain extension, namely zero-shot inference on spatial domains that are significantly larger than those encountered during training. We argue that this setting fundamentally requires spatial locality and translation equivariance. We propose to implement this locality via a decomposable bias in the attention logits computation, enabling finely controllable locality while remaining fully decomposable into query–key inner products and directly compatible with optimized attention kernels. Combined with rotary positional embeddings, it enables expressive embeddings with controllable spatial support without altering the transformer architecture. We empirically show that our approach substantially improves zero-shot generalization to larger domains across two PDE benchmarks and a 3D industrial atmospheric flow application.

1 INTRODUCTION

Partial differential equations (PDE) are ubiquitous in physics and engineering, but are often computationally intensive to solve with traditional numerical solvers. Deep learning based *neural operators* (Kovachki et al., 2023) have emerged as an attractive alternative: once trained, they offer fast inference while capturing complex nonlinear behaviors in high-dimensional systems. Yet this perspective hinges on large amounts of high-fidelity training data, which can be prohibitively expensive to obtain. As the computational cost of a simulation scales with the domain size, training a model on simulations over small domains while retaining accuracy over larger domains is a promising alternative.

Among existing neural operator architectures, transformer-based models (Cao, 2021) have shown strong performance across a variety of benchmarks (Alkin et al., 2024; Wu et al., 2024; de Villeroché et al., 2026; Wen et al., 2025). By applying attention directly to unstructured point clouds, transformers provide flexibility for complex geometries and can scale to real-world industrial problems (Alkin et al., 2025). Despite these advances, current neural operator architectures are only trained and evaluated on domains of predefined size, which limits their applicability at inference time.

Domain extension We consider the problem of *domain extension*, defined as zero-shot generalization to a substantially larger domain than the training domain. This problem is particularly challenging: while tasks such as super-resolution or inference from partial observations correspond to an interpolation problem, domain extension is fundamentally an extrapolation problem. Despite its practical importance, this setting is barely explored for neural operators with prior work limited to domain decomposition (Huang et al., 2025) or graph neural network models (Pfaff et al., 2020).

Locality vs globality To be universal, a neural operator must be non-local (Lanthaler et al., 2025; Calvello et al., 2025), i.e. point-wise predictions must depend on data over the entire domain. Yet in practice neural operators are trained on domains of finite size, and hence can only learn dependencies *up to that size*. For transformer-based neural operators, non-locality is achieved via an

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.

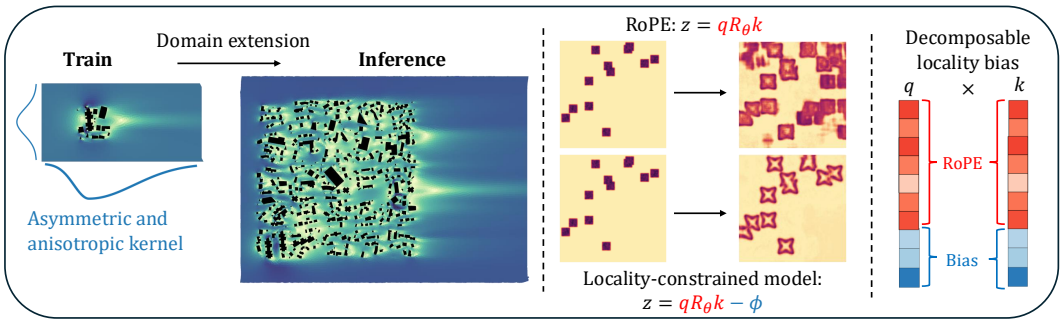


Figure 1: Domain upscaling. When increasing the domain size, RoPE creates spurious interactions for distances larger than the training domain. Our approach removes these interactions by enforcing locality via a decomposable bias term, enabling zero-shot generalization to larger domains without modification to the attention mechanism. This constraint must be fitted anisotropically and asymmetrically to the training domain to preserve long-range interactions.

integration operator, whose underlying integration kernel support will be *implicitly bounded* by the finite training domain. However, this implicit assumption breaks if the size of the domain changes at inference time. Hence, we propose to make this support bounded *explicitly*. This makes the operator's locality to the size of the training domain explicit and controllable, rather than implicit, and brings robustness to changes of the domain shape at inference, such as increasing the domain size.

Locality is hard to implement in practice Non-periodic physical fields have long been represented using localized representations such as wavelets (Daubechies, 1992), which offer a principled way to model spatially bounded interactions. However, integrating true wavelet-like or compactly supported locality into attention mechanisms remains difficult: numerically efficient attention kernels (Dao et al., 2022) assume that attention logits decompose into simple query–key inner products. Any positional structure that violates this constraint typically requires custom GPU kernels, which are challenging to generalize to multi-dimensional unstructured point clouds and become a serious barrier to scalability in large-scale neural operator applications.

This tension between the need for spatial locality, expressiveness and the constraints imposed by efficient attention implementations motivates the search for positional encodings that are simultaneously relative, localized, expressive, and fully compatible with optimized computational kernels. Ideally, such encodings should enable a tunable spatial support while preserving the efficiency of high-throughput matrix-multiplication-based attention.

To address this challenge, we combine Rotary Positional Encoding (RoPE) (Su et al., 2024) with a decomposable distance bias (Wu et al., 2025) (Figure 1). This creates a highly expressive relative position embedding with a tunable localized support, while ensuring compatibility with existing efficient attention kernels. We further extend existing positional biases using hyperbolic functions, creating numerically stable, asymmetric biases while retaining decomposability properties. Our main contributions are:

- we formulate the problem of *domain extension*, which is critical for scaling up to industrial-level applications, and show that it can be solved assuming a *local* neural operator;
- we combine RoPE with decomposable distance biases to create expressive position embeddings with localized kernel support that remains compatible with numerically efficient attention implementations. We propose formulations for both symmetric kernel support and asymmetric kernel support;
- we demonstrate empirically that this approach enables inference on substantially larger domains than training across two 1D and 2D benchmark problems and a challenging 3D industrial use case with a complex geometry.

2 LOCALITY VIA DECOMPOSABLE POSITIONAL BIASES

2.1 PRELIMINARY

Notations Let $\Omega \subset \mathbb{R}^p$ be a bounded domain. We adopt a neural operator view, and consider a continuous *latent field* $u : \Omega \to \mathbb{R}^d$, with $\mathbb{R}^d$ the latent space. We write $q, k, v : \Omega \to \mathbb{R}^d$ the query, key and value fields, obtained by learnable point-wise linear projections from the latent field u, and $z : \mathbb{R}^d \times \mathbb{R}^d \times \Omega \times \Omega \to \mathbb{R}$ the function used to compute the attention logits between two points. We note that the expression of z will depend on the positional embedding used. Finally, we adopt the variables $\xi \in \Omega$ to write the position of a key point and $c \in \Omega$ to write the position of a query point.

Attention as an integral operator Standard single-head scaled dot-product attention (Vaswani et al., 2017) is a Monte-Carlo approximation of the integral operator $\mathcal{G}$ defined as (Cao, 2021; Kovachki et al., 2023):

$$\forall c \in \Omega,\ \mathcal{G}(u)(c) = \int_\Omega \kappa(z(q(c), k(\xi), c, \xi))\, v(\xi)\, \mathrm{d}\xi, \tag{1}$$

where the kernel κ is defined by

$$\kappa(z(q(c), k(\xi), c, \xi)) = \frac{\exp(z(q(c), k(\xi), c, \xi)/\sqrt{d})}{\int_\Omega \exp(z(q(c), k(\xi'), c, \xi')/\sqrt{d})\, \mathrm{d}\xi'}. \tag{2}$$

2.2 REQUIREMENTS

Decomposability Numerical computation of the kernel κ implies a quadratic memory cost with the number of points over which u has been discretized, which is prohibitive for a large sequence of points. To reduce the memory footprint, optimized computation kernels, such as FlashAttention (Dao et al., 2022), decompose the computation using tiled matrix multiplications. While bringing considerable memory overhead reduction and computational speedup, fused GPU implementations of these algorithms have limited support for logits computation operations that are not written as a scalar product of a query and a key vector, making such an operation a barrier for scalability. Hence, to avoid this limitation, the logits function z must be *decomposable* as the scalar product of a function of $(q(c), c)$ and a function of $(k(\xi), \xi)$.

Locality and translation invariance We aim at preserving the translation invariance of κ — i.e., $\kappa = \kappa(\cdot, \cdot, c - \xi)$ depends only on relative positional differences. This property is particularly important for zero-shot generalization in large-scale physical domains. We therefore construct the new attention mechanism on top of RoPE as the backbone.

Nevertheless, RoPE and its variants rely on periodic functions to encode relative distances. When a model is trained on a bounded domain Ω_{train} and scaled up in a zero-shot manner to a significantly larger domain Ω_{test}, the periodic nature of the embedding can induce ambiguities in distance representation (Press et al., 2021; Peng et al., 2023). As a result, large relative distances in Ω_{test} may be mapped to similar embeddings as for shorter distances observed during training, thereby artificially inducing attention between points at distances that were not encountered within Ω_{train}.

Instead, we seek to restrict the support of κ to only interactions that were seen during the training phase. We argue that this can be archived by a *local* model. We can define the ensemble of oriented distances seen during training as $S_{\text{train}} = \{c - \xi \mid (c, \xi) \in \Omega^2_{\text{train}}\}$. We then require the kernel's support to be local, and bounded by S_{train}: $\kappa(\cdot, \cdot, c - \xi) = 0$ whenever $c - \xi \notin S_{\text{train}}$. As a consequence,

$$\mathcal{G}(u)(c) = \int_{\Omega_{\text{test}}} \kappa(z(q(c), k(\xi), c - \xi)) v(\xi) \mathrm{d}\xi = \int_{S(c)} \kappa(z(q(c), k(\xi), c - \xi)) v(\xi) \mathrm{d}\xi,$$

where $S(c) = \{\xi \mid c - \xi \in S_{\text{train}}\}$. Thus, at a given position c, $\mathcal{G}(u)(c)$ depends only on information within its neighborhood $S(c)$ and is therefore independent of the test domain Ω_{test}. Thus

$\mathcal{G}$ remains robust under enlargements of Ω_{test} relative to Ω_{train}, without requiring retraining. Intuitively, this approach can be understood as *preserving the same kernel* when changing the domain size at inference time.

Shape of the kernel support Ideally, the kernel κ should have a support that closely matches S_{train}, such that it is sufficiently large to capture all interactions present in Ω_{train} during training, but avoids introducing spurious long-range interactions during zero-shot domain extension. To achieve this, the support of the kernel should be able to match closely S_{train}. We highlight here two desirable properties of κ:

- *Anisotropy*: as Ω_{train} may be non-isotropic in complex geometries, S_{train} may reflect this structure. Consequently, κ should be anisotropically restrictable.

- *Asymmetry*: asymmetry may arise when Ω_{train} is not homogeneous and contains a sub-area $\Omega^b_{\text{train}} \subset \Omega_{\text{train}}$ which presents a specific behavior that determines the final solution. A common example is fluid-solid interactions: obstacles impacting the flow are often placed upwind in a larger simulated domain. This situation can also arise from the neural operator structure when cross-attention or perceiver attention is used to couple different parts of the simulation domain, as is done in (Alkin et al., 2025; de Villeroché et al., 2026). Under this assumption, the solution's behavior will be determined by values of the key function within Ω^b_{train}: consequently, relevant interactions will be dominated by $S^b_{\text{train}} = \{c - \xi \mid c \in \Omega_{\text{train}}, \xi \in \Omega^b_{\text{train}}\}$. Unlike the generic homogeneous case, S^b_{train} is *not necessarily centered around* 0. To represent this, κ may need a different, asymmetric support that accounts for the relative position of c and ξ.

2.3 Locality via a position bias

Commonly used RoPE (Su et al., 2024) embeds positions via a rotation matrix $R_\theta \in \mathbb{R}^{d\times d}$, formally defined in Appendix A.1. This formulation corresponds to a Fourier basis decomposition of the logits function on frequency basis θ with modulated coefficients constructed from $q(c)$ and $k(\xi)$ (see Appendix A.1). While particularly expressive and decomposable, this approach is inherently non-local.

Table 1: Different bias functions. $m \in \mathbb{R}^+, r \in \mathbb{R}^+$ and A_+, A_- diagonal matrices with positive coefficients are hyperparameters associated with each method, α being a token-wise learnable weight, and $\{e_i\}$ an orthonormal basis of Ω. *FlashBias in its original formulation is isotropic; in this work, we extend it to an anisotropic formulation.

			Kernel support	
Method	Logits bias ($z = q^\top R_\theta^\top k + \sqrt{d}b$)	Decomp.	Aniso.	Asym.
Window attention	$b = -\infty \mathbf{1}(\|c-\xi\|_1 > r)$	✗	✗	✗
ALiBi	$b = -m\|c-\xi\|_1$	✗	✗	✗
FlashBias	$b = -\alpha(u(c))\|c-\xi\|_2^2$	✓	✓*	✗
Hyperbolic bias (ours)	$b = -\frac{1}{2}\sum_i e^{A_-^{1/2}(c-\xi)\cdot e_i} + e^{-A_+^{1/2}(c-\xi)\cdot e_i}$	✓	✓	✓

To create a localized embedding that preserves the high expressiveness of RoPE, we add locality via a distance based position bias added to the logits. Unlike previous bias-based position embeddings (Press et al., 2021; Wu et al., 2025) which rely on a bias as the main position embedding methods, we view the bias purely as a geometrical term controlling locality, and retain RoPE as the position embedding backbone.

Table 1 summarizes commonly used position biases in transformers. Window attention (Beltagy et al., 2020) and ALiBi (Press et al., 2021) respectively apply a sliding attention window of half-width $r \in \mathbb{R}^+$ or a linear bias with distance with coefficient $m \in \mathbb{R}^+$. While these methods could be applied to enforce locality, they are not decomposable and hence would require significant implementation and computation overhead. We instead focus on decomposable bias functions: FlashBias (Wu et al., 2025) propose to use the L2 distance to create a decomposable distance bias. We extend

this bias to anisotropic configurations and define the associated logits as:

$$z^{\mathrm{L2}} = q^\top(c) R_\theta^\top (c-\xi) k(\xi) - \frac{\sqrt{d}}{2}(c-\xi)^\top A (c-\xi), \tag{3}$$

where the hyperparameter $A = \mathrm{diag}(\lambda_1^{-2}, \dots \lambda_p^{-2}) \in \mathbb{R}^{p\times p}$ is a diagonal matrix with positive coefficients controlling anisotropy. For $A = \mathrm{Id}$, the bias term corresponds to FlashBias with a fixed coefficient $\alpha = 1/2$. As this expression relies on a symmetric distance function $(c-\xi)^\top A(c-\xi)$, it cannot be used to define an asymmetric bias. We further propose a new formulation based on the hyperbolic function $\cosh$:

$$z^{\mathrm{hyp}} = q^\top(c) R_\theta^\top (c-\xi) k(\xi) - \frac{\sqrt{d}}{2} \sum_i \left\{ \exp(A_-^{1/2}(c-\xi)\cdot e_i) + \exp(A_+^{1/2}(\xi - c)\cdot e_i) \right\}, \tag{4}$$

where $A_+ = \mathrm{diag}(\lambda_{1,+}^{-2}, \dots \lambda_{p,+}^{-2}) \in \mathbb{R}^{p\times p}$ and $A_- = \mathrm{diag}(\lambda_{1,-}^{-2}, \dots \lambda_{p,-}^{-2}) \in \mathbb{R}^{p\times p}$ are two different anisotropy matrices inducing *asymmetric* behaviors depending on the relative position of c and ξ, and $\{e_i\}$ can be any orthonormal basis of $\mathbb{R}^p$. We show in Appendix A.2 that both methods can be decomposed into query-key inner products.

In the symmetric case $A_+ = A_- = A$, our formulation closely matches the original symmetric formulation of an L2-distance bias: indeed we show in Appendix A.2 that $z^{\mathrm{L2}} \sim z^{\mathrm{hyp}}$ up to an additive constant if $(c-\xi)\cdot e_i \ll \lambda_i$. For $(c-\xi)\cdot e_i \gg \lambda_i$, both decay to $-\infty$. Consequently, we consider our formulation to be a natural extension of the L2 symmetric bias for asymmetric cases.

Choosing the kernel width Selecting the width of the kernel correctly is critical in our method: a too small kernel will not model long distance interactions present in the training domain, while a too large kernel will be sensitive to an increase in domain size. An ideal kernel' support should match closely the ensemble of interactions present in the training domain S_{train}. However, a position bias does not enforce a strict limit of the kernel support, but instead imposes exponential decay to 0. To better build an intuition, we consider the $1D$, symmetric and isotropic case with a coefficient $\lambda \in \mathbb{R}^+$. Then the integration kernel reads:

$$f_\lambda(c-\xi) = e^{-\frac{(c-\xi)^2}{2\lambda^2}}, \qquad \kappa(z(q(c), k(\xi), c, \xi)) = \frac{f_\lambda(c-\xi) e^{q^\top(c) R_\theta^\top (c-\xi) k(\xi)/\sqrt{d}}}{\int_\Omega f_\lambda(c-\xi') e^{q^\top(c) R_\theta^\top (c-\xi') k(\xi')/\sqrt{d}} \mathrm{d}\xi'}.$$

Hence, the integration kernel support is controlled via a *Gaussian function of standard deviation of* λ. Under this insight, we consider the attention weights to become statistically insignificant for a distance exceeding 3λ. Consequently, λ *should be chosen at most as a third of the training domain size* to restrict the integration kernel to interactions present in the training data.

The above expression can be easily extended to multiple dimensions, for which f will be a multivariate Gaussian function with separate standard deviations per axis determined by the coefficients of A.

3 Numerical experiments

We validate our approach on 1D and 2D academic datasets and a 3D industrial use case. For each dataset, we evaluate the zero-shot robustness of the trained model on progressively larger test domains.

3.1 Academic datasets

As currently existing benchmarks do not have validation simulations over large domains, we generate our own datasets on classically-evaluated PDEs: Shallow Water equations and Gray-Scott equations. Shallow water equations model the evolution of the water height h and the velocity v of a water surface in 1D. Gray-Scott equations model the evolution of concentrations U and V in

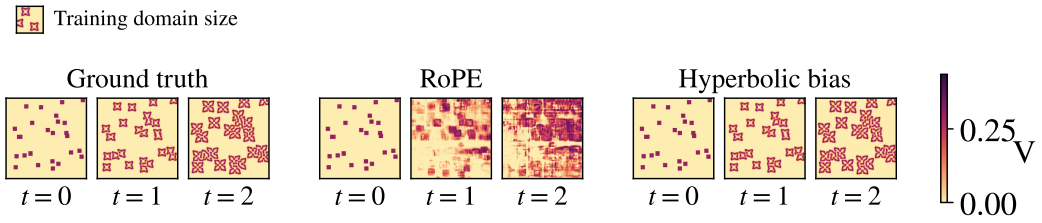


Figure 2: Autoregressive prediction of V for 3 steps of the Gray-Scott PDE for $s_x \times s_y = 3 \times 3$.

a reaction-diffusion system in 2D. For each dataset, we generate a training set of simulations over domain $\Omega_{\text{train}} = [0, 1000]^p$ in normalized units, with p the physical system dimension. We then generate large scale test datasets, with a per-dimension scaling of the domain size controlled by a factor s_p: $\Omega^s_{\text{test}} = [0, 1000 s_p]^p$. Further information on the solved PDE and datasets can be found in Appendix D, and visualization of the simulations are available in Appendix E.

To show the universality of our approach, we use a generic transformer neural operator architecture consisting of stacked transformer blocks with skip connections, linear input embeddings, and linear output projections. The model and hyperparameters are further described in Appendix C. We do not use absolute position embeddings on the initial embeddings, and only embed the previous time step and a boundary indicator. Combined with relative attention positional embeddings, this makes the model equivariant by translation, and robust to new coordinates when changing the domain size. We compare models trained with RoPE, an L2 bias, and a hyperbolic bias in Table 2. For the L2 and the hyperbolic biases, we use an isotropic decay length of $\lambda = 250$, yielding a kernel support comparable to the training domain size. For both datasets, RoPE performance degrades significantly as the inference domain size increases, while errors for locality-constrained models remain nearly constant. The L2 bias shows increased error at higher scalings ($s \geq 7$), which we attribute to numerical instabilities and further discuss in Appendix B. The relatively high base error for Gray–Scott is due to small average values of ΔV, which amplify the normalized L_1 metric. Figure 2 shows that an hyperbolic bias accurately predicts V autoregressively over 3 time steps for $s_x \times s_y = 3 \times 3$, whereas unbiased RoPE predictions degrade immediately.

Table 2: L1 error [%] for different test domain size where s denotes the scaling factor of the test domain size compared to the one of training. $\Delta\cdot$ denotes the difference between two time steps.

Shallow water equations (1D+time)										
s_x	1		2		3		7		10	
	Δv	Δh	Δv	Δh	Δv	Δh	Δv	Δh	Δv	Δh
RoPE	1.1	1.2	1.4	1.5	2.2	2.3	7.4	7.4	11.4	10.8
L2 bias	1.4	1.4	1.3	1.3	1.3	1.3	2.2	2.2	4.5	4.4
Hyperbolic bias	1.2	1.2	1.2	1.2	1.2	1.2	1.1	1.1	1.1	1.1
Gray-Scott equations (2D+time)										
$s_x \times s_y$	1×1		2×2		3×3		7×7		10×10	
	ΔU	ΔV	ΔU	ΔV	ΔU	ΔV	ΔU	ΔV	ΔU	ΔV
RoPE	31	33	153	175	347	373	436	483	386	430
L2 bias	28	30	25	28	26	29	67	78	119	136
Hyperbolic bias	29	32	26	30	26	30	27	30	27	31

3.2 Microscale atmospheric flow

We further evaluate our approach on a challenging industrial use case such as microscale atmospheric flow in urban environments. We rely on the AB-SWIFT model (de Villeroché et al., 2026), a specialized model for atmospheric flow built upon AB-UPT (Alkin et al., 2025), a well-validated recent architecture for PDE learning. We use the associated RandomBuildingsDataset, which consists of 3D steady-state simulations of wind flow around randomly generated urban geometries.

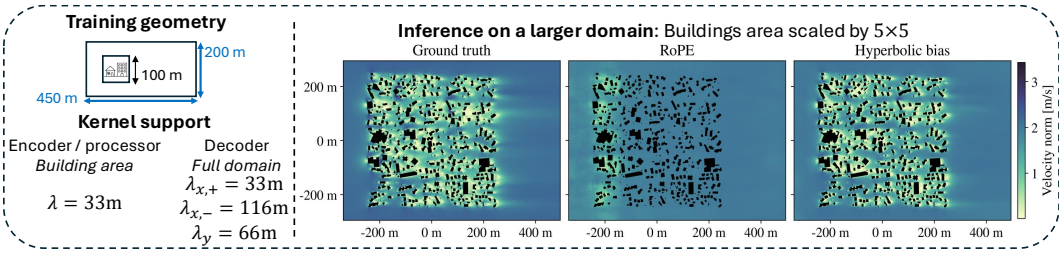


Figure 3: Left: geometry of the industrial use case with buildings located in an off-centered, squared area upwind of the domain. To reflect this, the geometry encoder and the processor are parametrized to have a kernel support limited to the size of the buildings area, and the volume decoder has an off-centered asymmetric kernel corresponding to the size of the full volumic mesh. Right: prediction of AB-SWIFT with an hyperbolic bias or unbiased RoPE, with a scaling factor of $s_x \times s_y = 5 \times 5$.

Beyond increased geometric and physical complexity, this task exhibits strong anisotropy due to the rectangular domain and a pronounced asymmetry, as buildings are located on the upwind side of the simulation domain. As a result, learnable interaction lengths differ substantially upstream and downstream of the building area. Moreover, predicting steady states involves long-range correlations, making this task a stringent test of locality-constrained operators.

AB-SWIFT follows a multi-block architecture with encoder, processor, and decoder modules, each operating over different spatial regions. The encoder embeds geometric information localized to the building area, the processor jointly handles geometric and volumetric information over the full domain, and the decoder operates solely on volumetric data. To reflect the heterogeneous geometry and interaction patterns of each module, we use distinct kernel supports per block. We use hyperbolic biases to allow for asymmetries, and to avoid numerical stability issues associated with L2 biases (Appendix B). The encoder and processor employ an isotropic and symmetric support matched to the size of the square building region. In contrast the decoder uses an asymmetric and anisotropic support reflecting the shape of the domain and the upstream/downstream asymmetry.

We compare AB-SWIFT models trained with RoPE or an hyperbolic bias on test domains with scalings of $s_x \times s_y = 2 \times 2$ and 5×5 of the buildings sub-area. Data generation is described in Appendix D. To preserve translation equivariance, the model omits absolute positional embeddings; all other hyperparameters are identical to the original AB-SWIFT implementation. Performance is evaluated using the normalized L1 error on the velocity field, reported separately for the building and wake regions due to their scale-dependent relative sizes. Table 3 shows that the error increases with domain size when using an hyperbolic bias, albeit much less than with RoPE. Furthermore, Figure 3 shows that the locality-constrained model preserve the overall flow structure, yielding a coherent prediction, while RoPE prediction fails catastrophically. Finally, the local model still captures the long-range wakes behind the buildings area, showing that it is still capable of modeling long-range correlations, up to the size of the kernel support. Consequently, we attribute the error increase for the local model metrics to new flow patterns arising from having buildings in a larger area, and which are not present in the original training data.

Table 3: L1 error [%] on the velocity field for different domain scaling factors. The building area is the square region occupied by buildings, while the wake area comprises the rest of the domain.

Microscale atmospheric flow						
	Buildings area			Wakes area		
$s_x \times s_y$	1×1	2×2	5×5	1×1	2×2	5×5
RoPE	4.9	13.8	33.0	1.3	5.5	8.2
Hyperbolic bias	4.5	9.4	16.7	1.2	3.8	5.3

4 LIMITATIONS

Sparsity An additional advantage of the proposed framework is that locality induces sparsity in the attention matrix, which has the potential to significantly reduce computational cost, ultimately enabling linear complexity in the number of points (Beltagy et al., 2020). However, in this proof-of-concept work, we do not yet exploit this structure, and our implementation retains the quadratic complexity of full self-attention. However, while windowed attention in natural language processing (NLP) yields an easily exploitable banded matrix structure, unstructured, multi-dimensional point clouds typically lead to randomly sparse attention matrices. This makes sparsity significantly harder to exploit in practice. An interesting direction for future work could be combining our work with algorithms reordering the mesh to enforce a banded structure such as space-filling curve, Cuthill-McKee mesh reordering (Cuthill & McKee, 1969) or ball tree partitioning, which have already been used for attention mechanisms (Zhdanov et al., 2025).

Correlations longer than the training domain While we show that locality makes neural operators independent of the inference domain size, we note that the true solution operator may not be: the underlying physics may present correlations at a longer range than what is seen in the training domain, which will not be captured by our approach. However, to capture these correlations in a zero-shot generalization setting, the learned integration kernel would have to be extrapolated and generalized beyond its training support. While we note that some physics may well-behave enough to make this extrapolation possible, it is not true in general, as fundamental solution behaviors may vary depending on the scale of the problem. A more robust approach is to fine-tune a model on a few large cases, which we leave to future work.

5 RELATED WORK

We review related work through the lens of *domain extension* for neural operators. While many studies address generalization across resolutions, geometries, or partial observations, domain extension constitutes a fundamental extrapolation setting that has received comparatively little attention.

Extension in NLP via representation scaling A related problem arises in natural language processing, where transformers struggle to extrapolate beyond the context lengths seen during training. Methods such as ALiBi (Press et al., 2021) introduce distance-dependent biases into attention logits, while more recent approaches rely on position interpolation (Chen et al., 2023a; Peng et al., 2023) to rescale positional coordinates for longer contexts. Despite conceptual similarities, these approaches assume scale-invariant, one-dimensional inputs—an assumption that breaks for PDEs defined over multi-dimensional domains with physically meaningful interaction lengths.

Architectural scaling via domain decomposition Domain decomposition methods (Toselli & Widlund, 2004; Mathew, 2008) scale numerical solvers by partitioning large domains into smaller subdomains, and have recently been adapted to neural operators (Huang et al., 2025). However, these approaches operate at the application level rather than at the operator level, requiring explicit coupling mechanisms and introducing challenges when stitching subdomain interfaces. They do not directly address the behavior of a single learned operator under domain enlargement. In contrast, our work targets robustness to domain extension without requiring explicit partitioning or retraining.

Locality in other neural operator architectures Other neural operator architectures rely on localized convolution kernel. Graph neural operators (Li et al., 2020) are natively local, while Liu-Schiaffini et al. (2024) add localized kernel to Fourier Neural operators (Li et al., 2023). However these approaches are still limited by the shortcomings of their specific architectures; graph neural networks struggle to computationally scale to large meshes or model long-distance effects while fourier neural operators lack support for unstructured meshes.

Kernel-level control through locality and positional encodings Transformer-based neural operators (Cao, 2021; Alkin et al., 2024; 2025; Wu et al., 2024; Serrano et al., 2024) operate directly on unstructured point clouds and provide highly expressive non-local representations. Relative positional encodings such as RoPE (Su et al., 2024; Heo et al., 2024) are widely adopted due to

their translation equivariance and compatibility with optimized attention implementations such as FlashAttention (Dao et al., 2022). However, their quasi-periodic structure can associate large relative distances with shorter ones, inducing spurious interactions under domain extension. Locality can be enforced explicitly through window attention (Beltagy et al., 2020), but strong attention masks break decomposability and limit scalability on irregular geometries. More recently, FlashBias (Wu et al., 2025) introduces a decomposable L2 distance bias compatible with efficient attention kernels, providing controlled locality but restricting the kernel support to symmetric configurations.

Our work builds on these developments by introducing explicit, decomposable control over the spatial support of attention integration kernels. By extending distance-based biases to anisotropic and asymmetric configurations, we enforce physically meaningful locality while preserving expressiveness and compatibility with high-performance attention kernels, enabling robust zero-shot domain extension.

6 Conclusion and future work

In this study, we addressed the problem of domain extension for transformer-based neural operators by introducing explicit spatial locality into attention mechanisms. We have shown that an L2 bias restricts kernel support to a Gaussian-like shape, and proposed a novel hyperbolic decomposable bias that extends existing methods to asymmetric settings, while avoiding numerical stability issues of an L2 bias decomposition. Experiments on multiple PDE benchmarks, including a large-scale 3D industrial case, show that enforcing locality significantly improves zero-shot generalization to larger domains.

More broadly, our results suggest that enforcing structured locality in neural operators may provide a principled route toward appropriate scaling laws in machine learning. By explicitly constraining the effective support of interactions, models can generalize to larger domains without requiring proportional increases in training data or model capacity. This points to an alternative scaling paradigm where performance gains arise not only from data and model scale, but from modulating inductive biases that reflect how information propagates across space.

AI use statement

In this work we have only used generative AI for corrective editing and proofreading of the paper's text, and for creating the plotting scripts for some of the figures presented in this paper. We have not, in particular, used generative AI for any other tasks required for disclosure by the ICLR 2027 guidelines. All AI-generated texts and edits have been thoroughly reviewed and verified. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Section 2.3 describes the methodology used in this paper, while Annex A.2 describes the associated implementation. Appendix C describes the training and models setup, and Appendix D the dataset generation processes. Furthermore, all codes and datasets are made available for reviewers, and will be made public with the final version of the paper.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *arXiv e-prints*, pp. arXiv–2402, 2024.

Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. Ab-upt: Scaling neural cfd surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *arXiv preprint arXiv:2502.09692*, 2025.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Maurits Bleeker, Markus Hennerbichler, and Pavel Kuksa. Noether: A pytorch-based framework for engineering ai, 2026. URL https://github.com/Emmi-AI/noether. Equal contribution.

Edoardo Calvello, Nikola B Kovachki, Matthew E Levine, and Andrew M Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023a.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023b.

Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pp. 157–172, 1969.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

Armand de Villeroché, Rem-Sophia Mouradi, Vincent Le Guen, Sibo Cheng, Marc Bocquet, Alban Farchi, Patrick Armand, and Patrick Massin. Anchored-branched steady-state wind flow transformer (ab-swift): a metamodel for 3d atmospheric flow in urban environments. *arXiv preprint arXiv:2603.25635*, 2026.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Jianing Huang, Kaixuan Zhang, Youjia Wu, and Ze Cheng. Operator learning with domain decomposition for geometry generalization in pde solving. *arXiv preprint arXiv:2504.00510*, 2025.

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

Samuel Lanthaler, Zongyi Li, and Andrew M Stuart. Nonlocality and nonlinearity implies universality in operator learning. *Constructive Approximation*, 62(2):261–303, 2025.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. *arXiv preprint arXiv:2402.16845*, 2024.

Tarek Poonithara Abraham Mathew. *Domain decomposition methods for the numerical solution of partial differential equations*. Springer, 2008.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

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, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2004.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. *arXiv preprint arXiv:2505.18781*, 2025.

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, Yuezhou Ma, Yuanxu Sun, Jianmin Wang, Wojciech Matusik, and Mingsheng Long. Flashbias: Fast computation of attention with bias. *arXiv preprint arXiv:2505.12044*, 2025.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. *arXiv preprint arXiv:2502.17019*, 2025.

A Position embeddings

This appendix presents detailed formulations and decomposed expression of all position embeddings used in this work.

Setting Let p denote the spatial dimension of the physical domain (typically 2 or 3). We consider a canonical spatial basis $\{e_1, \dots, e_p\}$. Let $c, \xi \in \mathbb{R}^p$ be two spatial locations, with $c_i = c \cdot e_i$ and $\xi_i = \xi \cdot e_i$. Let $q(c), k(\xi) \in \mathbb{R}^d$ be the query and key embeddings of a given attention head.

We denote $\bigoplus$ the concatenation operation of two or more tensors.

A.1 Rotary positional embedding

Rotary positional embeddings (RoPE) were first introduced by Su et al. (2024) in the context of natural language processing, and later generalized to multi-dimensional settings by Heo et al. (2024). RoPE embeds positional information using rotation matrices. Let $\theta \in \mathbb{R}^{d/2 \times p}$ denote a (possibly learnable) basis of frequencies. We define the block-wise rotation matrix R_θ as

$$R_\theta(c) = \begin{pmatrix} r(\theta_0 \cdot c) & & & 0 \\ & r(\theta_1 \cdot c) & & \\ & & \ddots & \\ 0 & & & r(\theta_{d/2-1} \cdot c) \end{pmatrix},$$

where

$$r(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

is the rotation matrix of angle α.

While θ may be learned, in this work we use the axial RoPE formulation of Heo et al. (2024), which corresponds to:

$$\theta = \bigoplus_{i\in[1,p]} \bigoplus_{j\in[0,(d/2-1)/p]} \left(10000^{-2jp/d}\, e_i\right),$$

RoPE embeds coordinates directly into the attention logits by separately modifying the query and key vectors:

$$q^{\mathrm{RoPE}}(c) = R_\theta(c)q(c), \qquad k^{\mathrm{RoPE}}(\xi) = R_\theta(\xi)k(\xi).$$

Due to the properties of rotation matrices, the resulting attention logits depend only on the relative displacement $c - \xi$:

$$q^{\mathrm{RoPE}}(c)^\top\, k^{\mathrm{RoPE}}(\xi) = q^\top(c)\, R_\theta^\top(c-\xi)\, k(\xi).$$

A direct calculation shows that RoPE corresponds to a Fourier decomposition of the attention logits over the frequency basis θ, with modulated coefficients $\alpha_j(q(c), k(\xi))$ and $\beta_j(q(c), k(\xi))$:

$$\begin{aligned}
\alpha_j(q(c), k(\xi)) &= q_{2j}(c)k_{2j}(\xi) + q_{2j+1}(c)k_{2j+1}(\xi),\\
\beta_j(q(c), k(\xi)) &= q_{2j+1}(c)k_{2j}(\xi) - q_{2j}(c)k_{2j+1}(\xi),\\
z(q(c), k(\xi), c-\xi) &= \sum_{j=0}^{d/2-1} \alpha_j(q(c), k(\xi))\cos(\theta_j\cdot(c-\xi)) + \beta_j(q(c), k(\xi))\sin(\theta_j\cdot(c-\xi))\,.
\end{aligned}$$

A.2 Local positional embeddings

We build - on top of RoPE and FlashBias (Wu et al., 2025) - a localized embedding. We specifically separate the expressive relative attention term (handled by RoPE) and a purely geometrical locality potential Φ as an additive FlashBias-like term:

$$z(q(c), k(\xi), \delta) = q^\top(c)R_\theta^\top(\delta)k(\xi) - \sqrt{d}\Phi(\delta),$$

where $\delta = c - \xi$ is the relative displacement.

Anisotropic L2 locality potential A simple potential function can be written using the anisotropic L2 distance. With $A = \mathrm{diag}(\lambda_1^{-2}, \dots \lambda_p^{-2})$, the resulting potential is defined as:

$$\Phi_A^{\mathrm{L2}}(\delta) = \frac{1}{2}\delta^\top A\delta = \frac{1}{2}\sum_{i=1}^{p}(\delta\cdot e_i)^2/\lambda_i^2.$$

Each term of this potential is exactly separable with the following identity:

$$(c_i - \xi_i)^2 = \begin{pmatrix} c_i^2 & -2c_i & 1\end{pmatrix}\begin{pmatrix}1\\ \xi_i\\ \xi_i^2\end{pmatrix}.$$

This lets us augment the standard RoPE-modulated embeddings by concatenating $3p$ additional locality channels. Hence we define the query as:

$$q^{\mathrm{L2}}(c) = R_\theta(c)q(c)\bigoplus_{i=1}^{p}\frac{1}{2}\begin{pmatrix} c_i^2/\lambda_i^2\\ -2c_i/\lambda_i\\ 1\end{pmatrix}.$$

and the key embedding is defined as

$$k^{\mathrm{L2}}(\xi) = R_\theta(\xi)k(\xi)\bigoplus_{i=1}^{p}\begin{pmatrix} 1\\ \xi_i/\lambda_i\\ \xi_i^2/\lambda_i^2\end{pmatrix}.$$

Both embeddings lie in $\mathbb{R}^{d+3p}$.

Anisotropic and asymmetric hyperbolic locality potential. Let's now define a directional and anisotropic locality potential acting on relative displacements:

$$\Phi^{\mathrm{hyp}}_{A_+,A_-}(\delta) = \frac{1}{2}\sum_{i=1}^{p}\left[\exp\left(A_-^{1/2}\delta\cdot e_i\right) + \exp\left(-A_+^{1/2}\delta\cdot e_i\right)\right].$$

This potential is a sum of direction-wise exponential functions encoding anisotropic and asymmetric interaction ranges in physical space through diagonal matrices $A_+ = \mathrm{diag}(\lambda_{1,+}^{-2},\dots\lambda_{p,+}^{-2})$ and $A_- = \mathrm{diag}(\lambda_{1,-}^{-2},\dots\lambda_{p,-}^{-2})$.

For each spatial direction i, the exponential terms are exactly multiplicatively separable:

$$\exp\left(\frac{c_i-\xi_i}{\lambda_{i,-}}\right) = \exp\left(\frac{c_i}{\lambda_{i,-}}\right)\exp\left(-\frac{\xi_i}{\lambda_{i,-}}\right),$$

$$\exp\left(-\frac{c_i-\xi_i}{\lambda_{i,+}}\right) = \exp\left(-\frac{c_i}{\lambda_{i,+}}\right)\exp\left(\frac{\xi_i}{\lambda_{i,+}}\right).$$

Similarly to the L2 bias, this lets us augment the standard RoPE-modulated embeddings by concatenating $2p$ additional locality channels, corresponding to the positive and negative directions of each spatial axis.

The query embedding is defined as

$$q^{\mathrm{hyp}}(c) = R_\theta(c)q(c)\bigoplus_{i=1}^{p}-\frac{\sqrt{d}}{2}\,\exp(-c_i/\lambda_{i,+})\bigoplus_{i=1}^{p}-\frac{\sqrt{d}}{2}\exp(+c_i/\lambda_{i,-}),$$

and the key embedding is defined as

$$k^{\mathrm{hyp}}(\xi) = R_\theta(\xi)k(\xi)\bigoplus_{i=1}^{p}\exp(+\xi_i/\lambda_{i,+})\bigoplus_{i=1}^{p}\exp(-\xi_i/\lambda_{i,-}).$$

Both embeddings lie in $\mathbb{R}^{d+2p}$.

Hyperbolic bias is a natural extension of L2 bias for asymmetric cases .

Let us consider the behavior of hyperbolic embeddings for a symmetric choice of interaction ranges such that $A_+ = A_- = A$.

Then the hyperbolic bias expression simplifies with a $\cosh$ function:

$$\Phi^{\mathrm{hyp}}_{A} = \sum_{i=1}^{p}\cosh\left((\delta\cdot e_i/)\lambda_i\right).$$

Assuming small values of displacements δ, for which $\delta\cdot e_i \ll \lambda_i$ and $\cosh(\delta_i/\lambda_i)\sim 1+(\delta_i/\lambda_i)^2/2$, one gets,

$$\Phi^{\mathrm{hyp}}_{A}(\delta) \sim p + \frac{1}{2}\sum_{i=1}^{p}\frac{\delta_i^2}{\lambda_i^2} \sim p + \Phi^{\mathrm{L2}}_{A}(\delta).$$

Hence, for small displacements, both embeddings are similar up to an additive constant, which disappears under softmax normalization.

Furthermore, both potentials will diverge to $+\infty$ for large values of displacements. This leaves only a difference in the transitory region: a kernel with an hyperbolic bias has a slightly faster decay than with an L2 bias (see Figure 4). Consequently, we consider hyperbolic biases to naturally extend L2 biases to asymmetric cases.

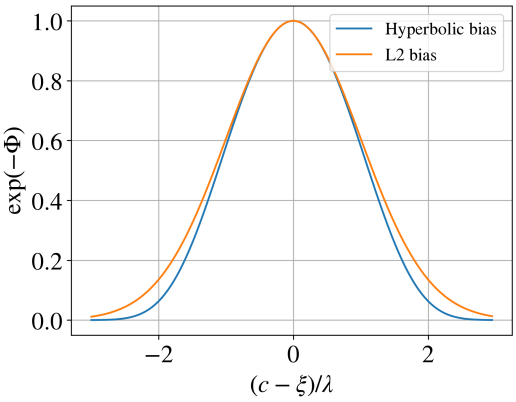


Figure 4: Comparison of the shape of the kernel enforced by an hyperbolic and an L2 bias, not accounting for the RoPE term, and assuming a 1D symmetric case.

B NUMERICAL STABILITY

Numerical errors of L2 decomposition We found experimentally that an L2 bias tends to exhibit numerical instabilities when reaching high coordinates values. Figure 5 shows this numerical error on the Gray Scott dataset for a scaling $s_x \times s_y = 7 \times 7$. The error of the L2 model's prediction is non-homogeneous and increases with the distance to the origin. By contrast, an hyperbolic bias remains stable even for large coordinate values.

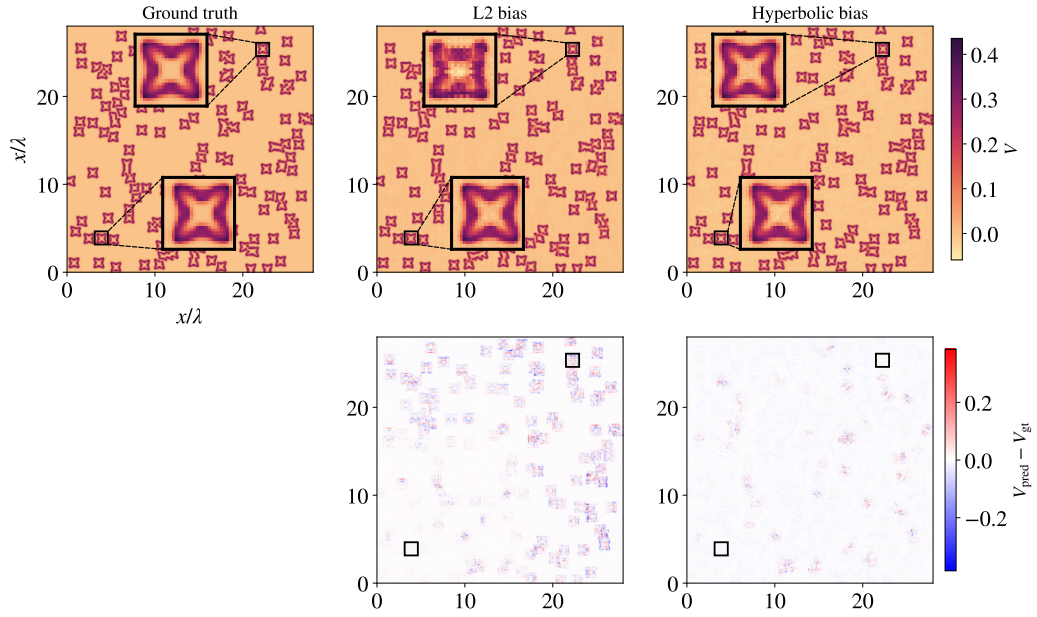


Figure 5: Comparative predictions of hyperbolic and L2 biases at $t = 1$ on the Gray Scott dataset for a scaling $s_x \times s_y = 7 \times 7$ with BFloat16 precision. L2 biases exhibits strong numerical errors for larger coordinates values, while hyperbolic biases remains stable.

To demonstrate that this error indeed comes from the numerical precision, we compare predictions using Bfloat16 and Float32 formats. A scale 7 inference is not computationally feasible with Float32. We instead perform this experiment on $s_x \times s_y = 2 \times 2$ and shift the input coordinates to reproduce coordinate values seen at larger scales. We present the result in Figure 6. For small coordinates, both predictions yield similar results. However, for large coordinates, prediction accuracy degrades significantly when using BFloat16 precision, while remaining stable with Float32.

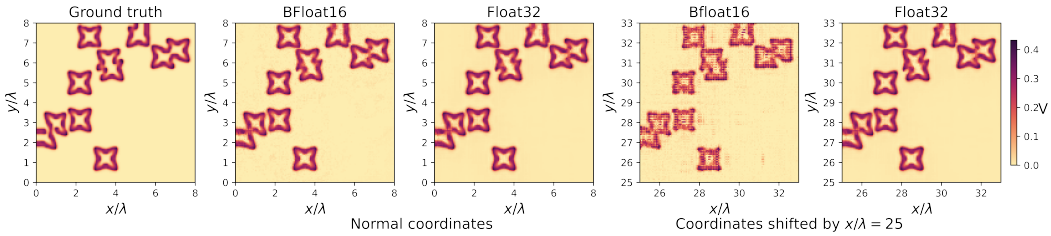


Figure 6: Numerical stability experiment. Left: Predictions using an L2 bias for Bfloat16 and Float32 precisions for coordinate values near the origin. Right: Predictions for coordinate values shifted by $x/\lambda = 25$ and $y/\lambda = 25$, mimicking values seen at large scaling.

Numerical overflow limit of hyperbolic biases The presented asymmetric bias relies on exponential functions, which introduces numerical limitations due to potential overflow and underflow. Specifically, the computation of the bias terms will overflow or underflow on positions that are too far from the domain center. Let $v_{\min}$ and $v_{\max}$ denote the smallest and largest representable positive numbers for a given precision. The maximum attainable domain size scales as

$$\lambda\big(\log v_{\max} - \log v_{\min}\big).$$

Table 4 summarizes the resulting limits for common floating-point formats, assuming a $1D$ isotropic and symmetric bias controlled by a decay length λ. Commonly used Bfloat16 format presents a maximum scaling of 181λ, which is very substantial in the context of physics simulations and covers most use cases.

Table 4: Maximum attainable scaling range of hyperbolic biases before numerical overflow or underflow.

Precision	underflow limit	overflow limit	max. domain size
Float16	6×10^{-8}	6.5×10^{4}	$27\,\lambda$
Float32 / Bfloat16	6×10^{-41}	3×10^{38}	$181\,\lambda$
Float64	2×10^{-308}	2×10^{308}	$1418\,\lambda$

C Training setup and model hyperparameters

This appendix presents detailed training setup and model hyperparameters. All conducted experiments were run using the Noether framework (Bleeker et al., 2026), and associated code and datasets will be made publicly available upon publication of this paper.

For all experiments, training is performed using the Lion optimizer (Chen et al., 2023b), with a 5% linear learning-rate warmup up to 5×10^{-5}, followed by a cosine decay to 1×10^{-6}. The weight decay parameter of Lion is set to 0.05. All models are trained using Bfloat16 precision. We train for 100 epochs with a batch size of 32 for the academic benchmarks, and for 500 epochs with a batch size of 1 for AB-SWIFT.

Input and output features are standardized, and positions are normalized anisotropically to $[0, 1000]$ in the training dataset. A log scale is used on the turbulence-related variables k and ϵ predicted by AB-SWIFT, following the choices of de Villeroché et al. (2026).

Academic case models We apply the same model for each dataset, only varying the positional embedding between experiments. We build a generic transformer neural operator architecture consisting of stacked transformer blocks and linear input embeddings and output projections. Each transformer block consists of a self-attention layer followed by a feedforward layer, with residual connections between each sublayer. We report the associated hyperparameters in Table 5.

AB-SWIFT We re-implement AB-SWIFT using the Noether framework, following the original implementation of de Villeroché et al. (2026). However, unlike the original implementation, we do

Table 5: Hyperparameters of the models used for all academic datasets experiments.

Parameter	value
Hidden dimension	192
Number of transformer blocks	6
Number of attention heads	3
Feedforward hidden layers	1
Feedforward expansion factor	4
Feedforward activation	GeLU
RoPE max frequency	10000

not use absolute positional embeddings to initially encode the latent state from absolute positions, and instead rely only on geometric information when available, or generate constant initial embeddings when no geometric information are available. We report details on hyperparameters in Table 6.

Table 6: AB-SWIFT hyperparameters.

Description	Value
Model parameters	
Hidden dimension	192
Obstacles supernode pooling radius	1
Terrain supernode pooling radius	5
Number of processor blocks	3
Number of decoder blocks	4
Number of attention head	3
Feedforward hidden layers	1
Feedforward expansion factor	4
Feedforward activation	GeLU
RoPE max frequency	10000
Training pipeline parameters	
Number of points describing obstacles	4096
Number of points describing the terrain	4096
Number of obstacles supernodes	1024
Number of terrain supernodes	1024
Number of volume anchor point	8192
Inference pipeline parameters	
All number of points are scaled proportionally to the area of the inference domain.	

Training cost Training takes approximately $1\,\mathrm{h}$ on a A100 GPU for the shallow water dataset, and $1.5\,\mathrm{h}$ for the Gray-Scott and the microscale atmospheric flow dataset. Running all experiments sequentially takes approximately $13\,\mathrm{h}$.

D Datasets

This appendix further describes the datasets used in this paper, summarized in Table 7.

D.1 Shallow water equations

Shallow water equations represent the movements of waves on a water surface, assuming a low water depth relative to the surface. The shallow water equations solved reads:

$$\frac{\partial h}{\partial t} + \frac{\partial (hv)}{\partial x} = 0, \qquad \frac{\partial (hv)}{\partial t} + \frac{\partial (hv^2)}{\partial x} + gh\frac{\partial h}{\partial x} = 0,$$

Table 7: Datasets considered in this work. For each dataset, we generate a large number of small-scale simulations for training, and a small number of large-scale simulations for evaluations, with a per-axis scaling ranging from 1 to 10. $\Delta\cdot$ denote the difference between two time steps.

Dataset	PDE	Dim	input variable	predicted variable
SWE 1D	Shallow water	1D+time	Water height h and velocity v	$\Delta h, \Delta v$
GrayScott	Gray Scott	2D+time	Concentrations U and V	$\Delta U, \Delta V$
RandomBuildingsDataset	Navier-Stokes	3D	Building topography meteorological conditions	3D steady-state atmospheric flow variables

where h and v respectively represent the water height and velocity, and $g = 9.81\,\mathrm{m\,s^{-2}}$ is the gravity acceleration constant. Homogeneous Neumann boundary conditions on h and v are used.

We generate an initial dataset of 800 training, 100 test and 100 validation simulations of a domain of width $100\,\mathrm{m}$, and an initial water height $h_0 = 1\,\mathrm{m}$. We randomly place 3 crenel perturbations of the initial water height in each simulation, with widths and heights uniformly sampled respectively between $4\,\mathrm{m}$ to $15\,\mathrm{m}$ and $0.02\,\mathrm{m}$ to $0.08\,\mathrm{m}$. We simulate the obtained system for $15\,\mathrm{s}$, and save the state every $0.3\,\mathrm{s}$.

We then generate large scale splits for scalings $s = 2, 3, 7$ and 10, and run 50 simulations per split. We scale the domain width and the number of initial perturbations by s.

Data generation takes a few seconds on an Intel i7-11850H laptop CPU.

D.2 Gray-Scott equations

The Gray-Scott equation system describes the evolution of two chemical substances U and V, with U being consumed by V and both substances diffusing over time. The Gray-Scott equation systems reads:

$$\frac{\partial U}{\partial t} = D_u \nabla^2 U - UV^2 + F(1-U), \qquad \frac{\partial V}{\partial t} = D_v \nabla^2 V + UV^2 - (F+k)V,$$

where $D_u = 0.2$ and $D_v = 0.1$ are respectively the diffusion coefficients of U and V, $k = 0.06$ is the kill rate of V and $F = 0.035$ is the feed rate. We use homogeneous Neumann boundary conditions on U and V.

We generate an initial dataset of 800 training 100 test and 100 validation simulations of a square domain of width 128×128 width grid cells of size 2×2 and, with homogeneous initial concentrations $U = 1$ and $V = 0$. For each simulation, we randomly place 3 initial sources of V within the simulated domain on squares of width 5 where we set $U = 0.5$ and $V = 0.25$. We simulated the obtained system for 5000 steps and save the state every 500 steps.

We then generate large scale splits for scalings $s = 2, 3, 7$ and 10, and run respectively 50, 50, 25, 10 simulations per scaling. We scale the domain width and height by s and the number of initial perturbations by s^2.

Data generation takes around $20\,\mathrm{min}$ on an Intel i7-11850H laptop CPU.

D.3 Random buildings dataset

The random buildings datasets contain steady-state atmospheric flows around randomly generated urban geometries with varied building geometries. Readers are referred to the original work of de Villeroché et al. (2026) for the detailed CFD simulation setup.

We generate large-scale validation cases by scaling the size of the buildings area width by s. For each simulation, we generate a new urban geometry on this enlarged area using the same generation method as de Villeroché et al. (2026). We scale the number of buildings by s^2, keeping the building density constant across all scalings. We generate large scale splits for scalings $s = 2$ and 5 and generate 10 simulations per scaling.

We restrict ourselves to neutral atmospheric stratifications because buildings are placed very close to the upstream boundary in the original dataset, making boundary conditions interact strongly with the buildings in other stratifications. As this effect depends on the exact distance between the buildings and the upwind boundary of the simulation domain, it could not be robustly reproduced when increasing the size of the mesh. In neutral stratification, this effect is much less pronounced. We note that this is a limit of the CFD setup and not of the proposed machine learning approach.

Data generation of large scale cases takes around 11 h on 4 Intel Platinum 8260 CPUs.

E VISUALIZATION OF DATASETS AND PREDICTIONS

The appendix presents visualizations of the different datasets and predictions.

E.1 SHALLOW WATER EQUATIONS

Figures 7, 8, and 9 present ground truth and models predictions for scalings $s = 1, 3$ and 10 respectively.

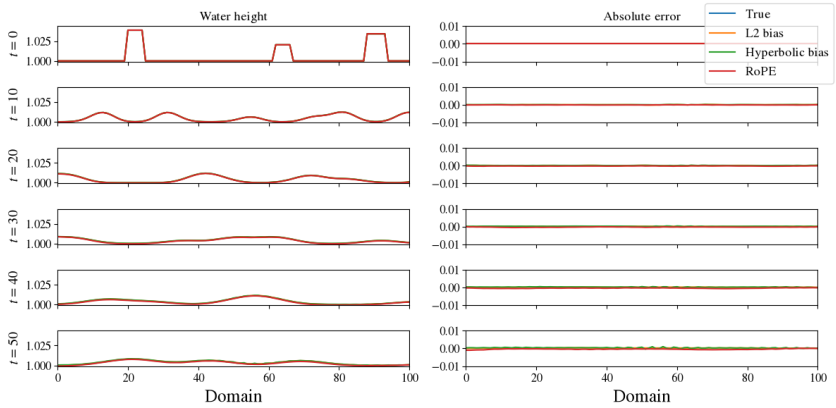


Figure 7: Shallow water ground truth and predictions for $s = 1$.

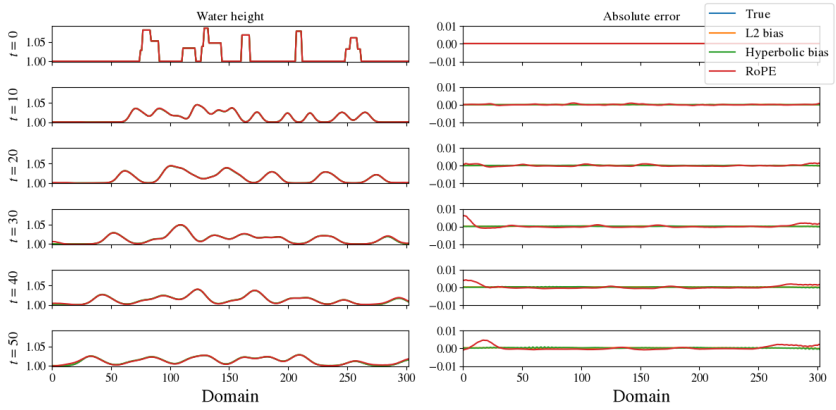


Figure 8: Shallow water ground truth and predictions for $s = 3$.

E.2 GRAY-SCOTT EQUATIONS

Figures 10, 11, and 12 present ground truth and models predictions for scalings $s_x \times s_y = 1 \times 1, 3 \times 3$ and 10×10 respectively.

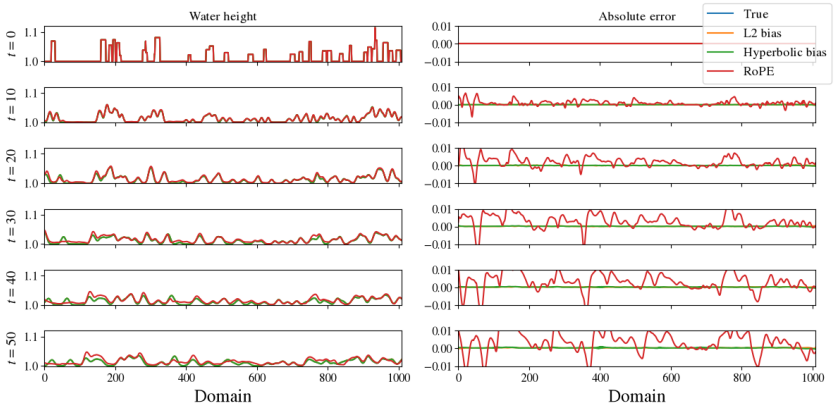


Figure 9: Shallow water ground truth and predictions for $s = 10$.

E.3 Microscale atmospheric flow

Figures 13, 14, and 15 presents ground truth and models predictions for scalings $s_x \times s_y = 1\times1, 2\times2$ and 5×5 respectively.

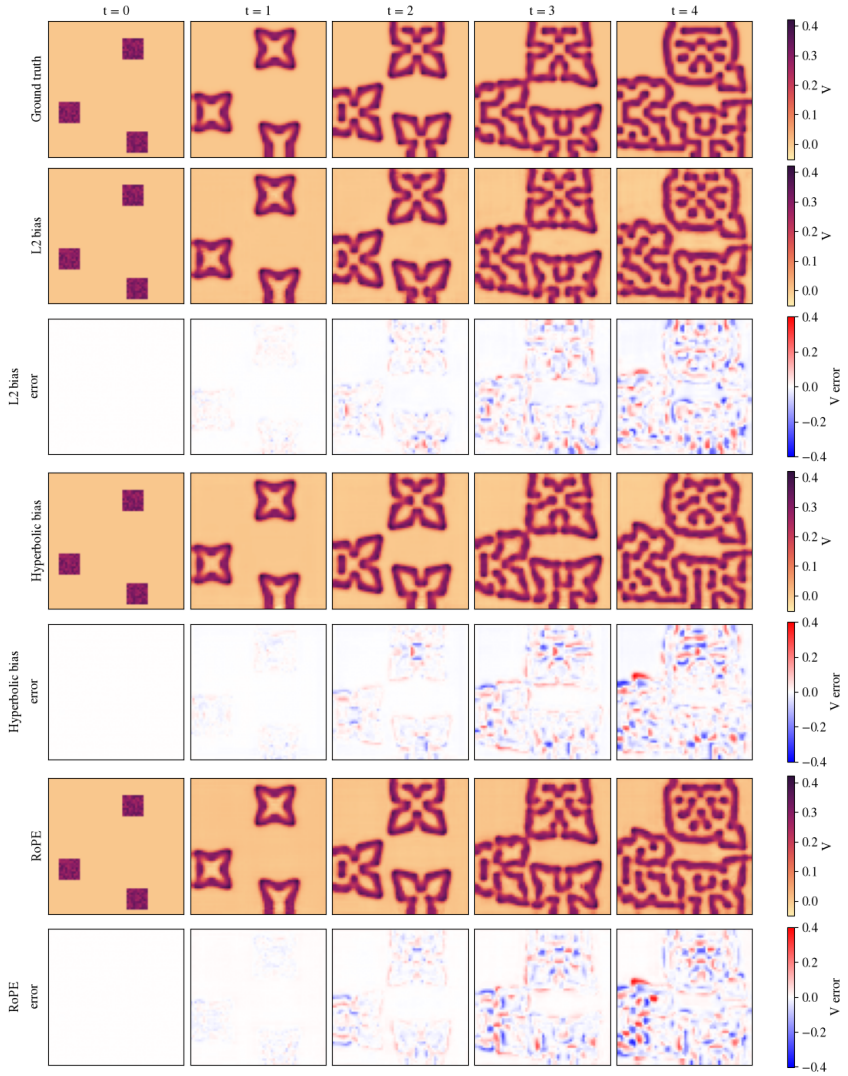


Figure 10: Gray-Scott ground truth and predictions for $s_x \times s_y = 1 \times 1$.

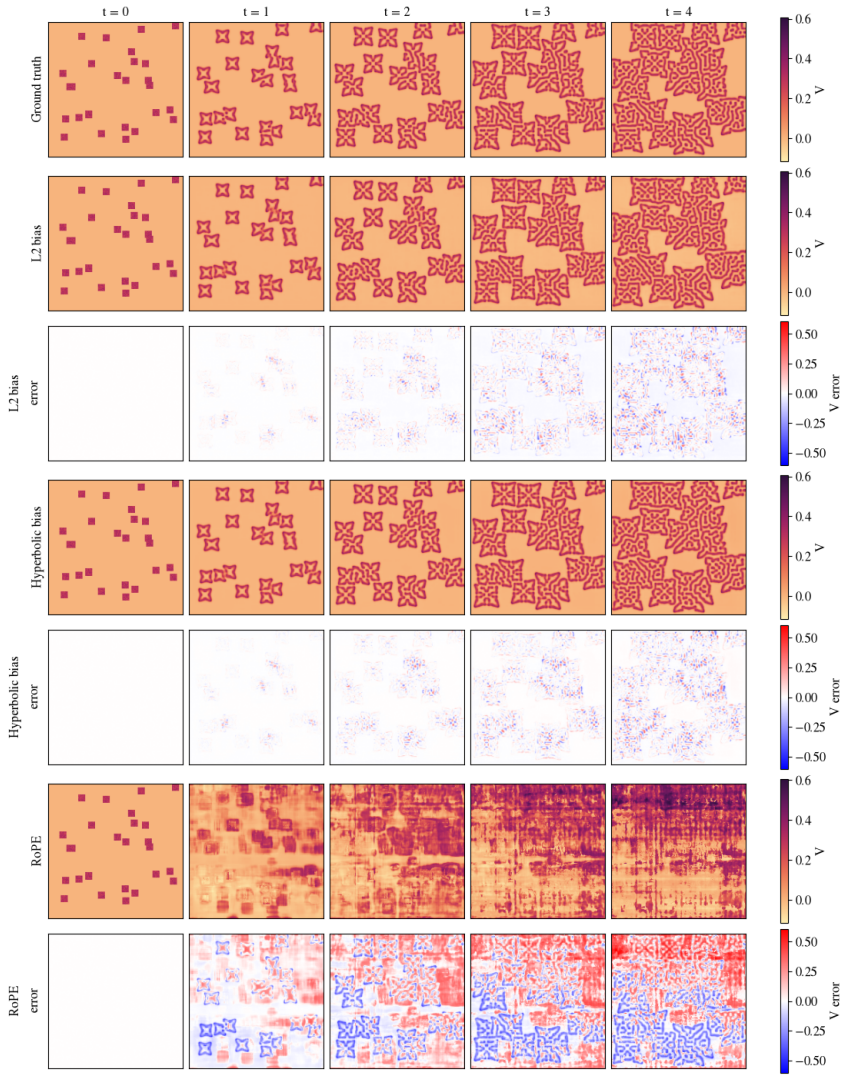


Figure 11: Gray-Scott ground truth and predictions for $s_x \times s_y = 3 \times 3$.

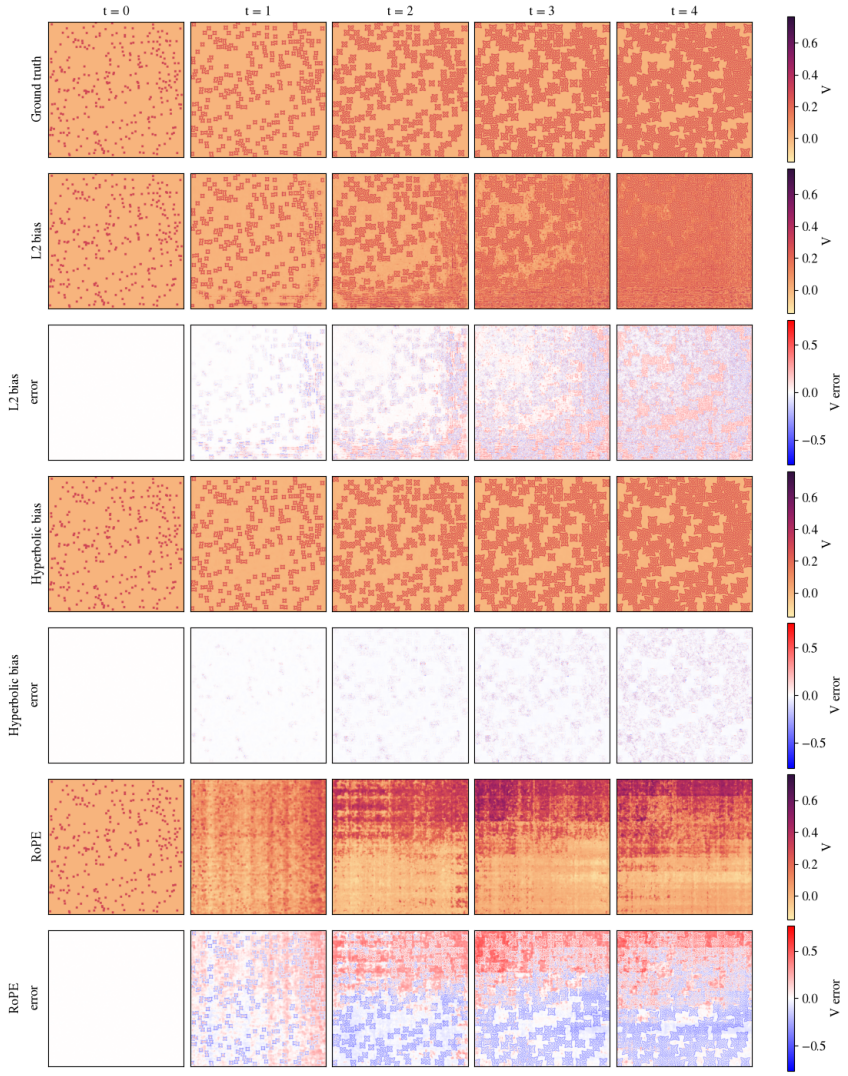


Figure 12: Gray-Scott ground truth and predictions for $s_x \times s_y = 10 \times 10$.

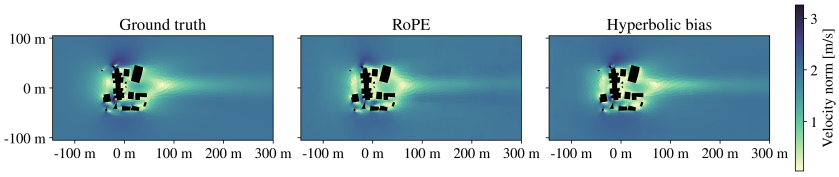


Figure 13: Microscale atmospheric flow ground truth and prediction for $s_x \times s_y = 1 \times 1$.

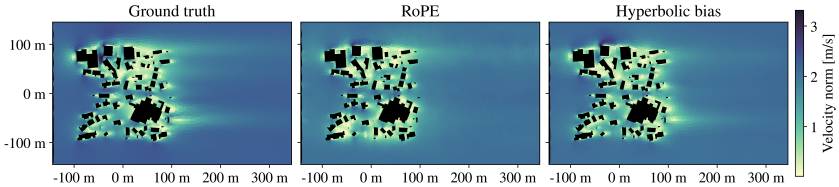


Figure 14: Microscale atmospheric flow ground truth and prediction for $s_x \times s_y = 2 \times 2$.

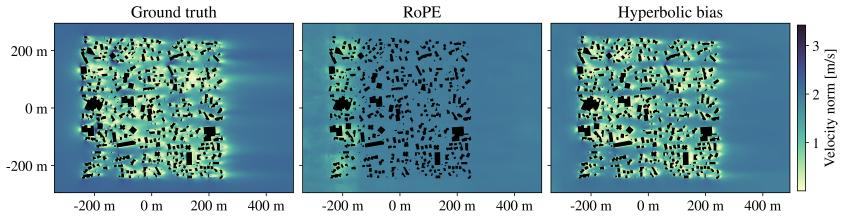


Figure 15: Microscale atmospheric flow ground truth and prediction for $s_x \times s_y = 5 \times 5$.