PAINT: PARALLEL-IN-TIME NEURAL TWINS FOR DYNAMICAL SYSTEM RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural surrogates have shown great potential in simulating dynamical systems, while offering real-time capabilities. We envision *Neural Twins* as a progression of neural surrogates, aiming to create digital replicas of real systems. A neural twin consumes measurements at test time to update its state, thereby enabling context-specific decision-making. A critical property of neural twins is their ability to remain *on-trajectory*, i.e., to stay close to the true system state over time. We introduce **Pa**rallel-in-**i**me **N**eural **T**wins (**PAINT**), an architecture-agnostic family of methods for modeling dynamical systems from measurements. PAINT trains a generative neural network to model the distribution of states parallel over time. At test time, states are predicted from measurements in a sliding window fashion. Our theoretical analysis shows that PAINT is on-trajectory, whereas autoregressive models generally are not. Empirically, we evaluate our method on a challenging two-dimensional turbulent fluid dynamics problem. The results demonstrate that PAINT stays on-trajectory and predicts system states from sparse measurements with high fidelity. These findings underscore PAINT's potential for developing neural twins that stay on-trajectory, enabling more accurate state estimation and decision-making. [1]

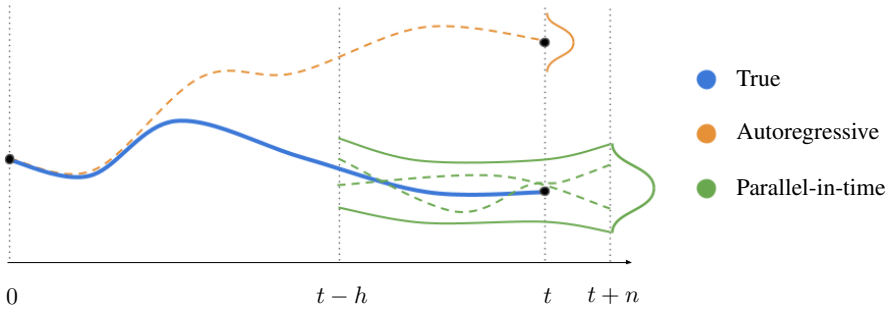


Figure 1: Real world measurements allow parallel-in-time models to stay on-trajectory, while autoregressive models may drift over time. Dashed lines represent sampled trajectories. Parallel-in-time models take a window of measurements and predict a distribution of subtrajectories which can be sampled and used for state and uncertainty estimation. Autoregressive models are prone to *over-reliance on their autoregressive state* (see Section 3.3) and typically depend on an initial state.

1 INTRODUCTION

Modern artificial intelligence is increasingly integrated into the modeling, monitoring, and control of complex dynamical systems. A key development in this domain are digital twins (Grieves, 2014): virtual models that run alongside a physical system, using live sensor data to update its predictions. Unlike standard simulations, a digital twin continuously adjusts its internal state to match the real system's behavior, thereby enabling better monitoring and automated decision-making.

[1] The code will be made public upon acceptance.

Neural surrogates emulate expensive, high-fidelity simulations and offer fast predictions while maintaining reasonable accuracy. Recent works have already demonstrated remarkable results on computational fluid dynamics (CFD) (Li et al., 2022a; Alkin et al., 2024a; Wu et al., 2024; Alkin et al., 2025), as well as on particle based systems as surrogate models for discrete element methods (Alkin et al., 2024c).

However, transforming neural surrogates into neural twins is a challenging endeavor. The current generation of neural surrogates can neither ingest nor adapt to real-time measurements during inference. Therefore, we ask the question: *How to design neural surrogates that adapt at test time using real-time measurements?*

We address this question by introducing **Pa**rallel-in-t**i**me **N**eural **T**wins (**PAINT**), a family of methods for reconstructing dynamical systems from measurements. At the core of PAINT is training a generative neural network to model the joint conditional probability of system states given the measurements. At inference, it reconstructs a sequence of states from measurements using a sliding-window approach. This makes our method interpretable and avoids any dependence on an initial condition.

Mathematically, we show that PAINT enables accurate state estimation from measurements due to being *on-trajectory*, i.e., staying close to the true trajectory even in presence of prediction errors. We also show that autoregressive models may possess this property under certain conditions (e.g. Kalman filters), but not in general. Our analysis also sheds light on a phenomenon we call *over-reliance on the autoregressive state*. It describes the problem of an autoregressive model over-relying on an off-trajectory state, whereas an ideal model would recognize such drift and reduce the state's influence on the prediction.

Empirically, we implement FlowPAINT, an instance of PAINT based on Flow Matching. FlowPAINT is evaluated on a dataset of state-of-the-art large eddy simulations with varying Reynolds numbers in 2D. This dataset is especially challenging due to the chaotic dynamics of the turbulent fluid. The results confirm the generalization capabilities of our method.

We show – theoretically and empirically – how to design neural twins which stay on-trajectory for arbitrary many timesteps. Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to train an on-trajectory neural twin for dynamical systems. Our method is interpretable, has an inherent measure of uncertainty and takes no prior assumptions on the initial state, therefore making it more widely applicable.
- Mathematically, we show that our method is on-trajectory, i.e., stays close to the true trajectory for arbitrarily long rollouts. Our analysis also sheds light on pitfalls when using autoregressive models with sparse measurements.
- We empirically demonstrate the effectiveness of our method on a challenging 2D turbulent fluid dynamics problem.

2 Background

Notation. Vectors and matrices are denoted in bold, e.g., $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. General sequences such as $[k, \dots, t]$ are denoted in brackets as $[k, t]$. We may also use brackets as subscripts over sequences, e.g., $\boldsymbol{x}_{[k,t]} = [\boldsymbol{x}_k, \dots, \boldsymbol{x}_t]$. True or ground-truth probability distributions are denoted $p(\cdot)$, while functions or distributions parameterized by neural networks use subscripts θ for their parameters, e.g., f_θ or $p_\theta(\cdot)$.

2.1 Digital twins and data integration

Digital twins as defined by Grieves (2014) interact with an industrial process by (a) consuming real-time data and (b) taking decisions based on the current state of the system. Ozan et al. (2025) marks a first end-to-end approach to incorporate and control a PDE with a learned policy. However as the state estimation is not interpretable, these models are hard to evaluate and can hardly be trusted for real-world production use cases. For a comprehensive introduction we refer to (Thelen et al., 2022).

Bayesian filtering. Updating model states by incorporating measurement data has a rich history, dating back to the seminal works on Wiener and Kalman filters (Wiener, 1949; Kalman, 1960). These types of Bayesian filters were extensively studied, leading to generalizations and extensions like the extended, unscented Kalman filters or particle methods. Bayesian filtering methods can be categorized by (1) Filtering: $p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})$, (2) Prediction: $p(\boldsymbol{x}_{t+n} \mid \boldsymbol{m}_{[0,t]})$, and (3) Smoothing: $p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,T]})$, where $\boldsymbol{x} \in \mathbb{R}^{d_x}, \boldsymbol{m} \in \mathbb{R}^{d_m}, t \in [0, T]$ and $n \in \mathbb{N}$. We refer to Särkkä & Svensson (2023) for a comprehensive overview on classical Bayesian filtering.
With the rise of deep learning in the mid-2010s, several works married Kalman or Bayesian filters with deep neural networks (Krishnan et al., 2015; Karl et al., 2016; Kim et al., 2025). However, similar to the original Kalman filter these works usually rely on simplifying assumptions, on the model, noise or initial condition, which is in stark contrast to our method.

Reconstructing flows from sparse measurements. Prior art has explored flow reconstruction from sparse measurements. These measurements are usually assumed to come from Particle Tracking Velocimetry (PTV), probes in the walls or at informative locations. Some works used generative modeling techniques to reconstruct flow fields from sparse measurements for a single timestep without temporal context (Güemes et al., 2022; Cuéllar et al., 2024; Oommen et al., 2025; Kim et al., 2021; Hemant Parikh et al., 2025). Similarly, Chakraborty et al. (2025) trained an instantaneous super-resolution weather model. Several works used physics-informed neural networks (PINNs) (Raissi et al., 2019) to reconstruct an instantaneous flow field from sparse measurements (Chaurasia & Chakraborty, 2024; Hosseini & Shiri, 2024; Toscano et al., 2024), however they lack the generative modeling which is necessary to rightfully model the distribution of possible solutions. Dang et al. (2024) used spatiotemporal information for reconstruction, but they did not model the target trajectories probabilistically and therefore learn the theoretical mean of the conditional distribution. From a conceptual point of view, the closest to our concrete implementation of PAINT is Sun & Wang (2020), however their construction based on Bayesian neural networks is unnecessarily complex, did only work on simple laminar flows and is not scalable to larger settings.

2.2 Scalable Neural Surrogates

Surrogates models. A surrogate model, or simply a surrogate (Forrester et al., 2008), is a simplified computational model designed to approximate a more intricate, computationally intensive system, such as a high-fidelity simulation or a physical experiment.

Transformers. The application of transformers (Vaswani et al., 2017) as surrogate models has recently taken over, building on their established efficacy in diverse scientific domains, including protein folding (Jumper et al., 2021; Abramson et al., 2024) and weather forecasting (Bi et al., 2023; Bodnar et al., 2024). These transformer-based neural surrogates are engineered to synthesize information across varying spatial locations and scales by leveraging attention mechanisms (Vaswani et al., 2017). Transformer-based surrogates represent an extension of the neural operator framework (Lu et al., 2019; 2021; Li et al., 2020b;a; Kovachki et al., 2021), specifically by integrating self-attention, cross-attention, or perceiver blocks (Jaegle et al., 2021b;a). Notable examples of these models include OFormer (Li et al., 2022b), Transolver (Wu et al., 2024), and (AB-)UPT (Alkin et al., 2024b; 2025). They have paved the way for scalable neural surrogate learning, scaling from a few thousand (Li et al., 2022a; Wu et al., 2024; Alkin et al., 2024b) to millions of mesh points (Alkin et al., 2025).

Generative modeling of PDEs. Over the past years, a powerful group of related generative modeling frameworks were introduced around the works of Diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020), Flow Matching (Liu et al., 2022; Lipman et al., 2022) and stochastic interpolants (Albergo & Vanden-Eijnden, 2022; Albergo et al., 2023a; Gao et al., 2024).
Generative models were used to model PDEs (Sestak et al., 2025; Kohl et al., 2023). Some works emphasize the beneficial properties of diffusion models for modeling high frequencies (Lippe et al., 2023), leading to more accurate and physically plausible solutions. Our work differs, as prior works sample *any* trajectory $p(x_{[0,t]})$, while we model $p(x_{[0,t]} \mid m_{[0,t]})$ and aim to stay on the real trajectory from measurements $m_{[0,t]}$ (see Section 3.2).

2.3 COMPUTATIONAL FLUID DYNAMICS

The fluid mechanics problem, under the continuum assumption, is governed by the Navier–Stokes equations, a set of highly complex, second-order nonlinear PDEs. These equations are complemented by appropriate boundary and initial conditions and are for engineering problems typically solved numerically. Because turbulence in fluid flows is in most engineering applications inevitable and involves a wide range of spatial and temporal scales, direct numerical simulation (DNS) is almost never feasible in practice. Most industrial applications therefore rely on Reynolds-Averaged Navier–Stokes (RANS) approaches, which solve only for mean fields under strong modeling assumptions. Large-Eddy Simulation (LES) offers a compromise: it resolves the larger, energy-containing scales while modeling only the smallest, but remains computationally demanding (Davidson, 2015; Fröhlich, 2006).

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} &= -\nabla p + \mu \Delta \boldsymbol{u} + \rho \boldsymbol{f_e} \end{aligned} \tag{1}$$

3 METHOD

3.1 A NEURAL TWIN FRAMEWORK FOR DYNAMICAL SYSTEMS

We envision neural twins in a four-step process, depicted in Figure 2. First, design knowledge of the dynamical system is used to create the simulations. This ensures that the simulations reflect physical constraints and key behaviors. Second, a diverse training dataset is generated by exploring a broad spectrum of trajectories. This dataset is used for training a neural twin. Third, at test time the neural twin dynamically incorporates real-time measurement data, allowing it to stay aligned with actual system trajectories and support real-time decision-making. Fourth, via the state estimation of the neural twin, one can optionally control the dynamical system, e.g. via rule-based methods. The approach combines domain expertise with adaptive learning, enabling both accuracy and agility in complex environments.

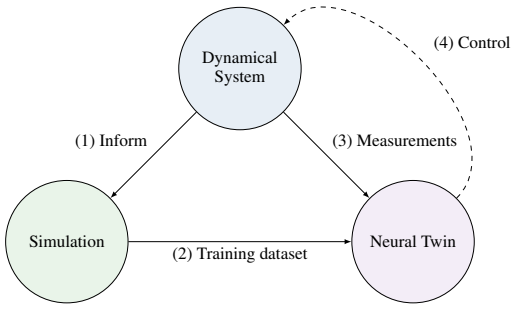


Figure 2: Interaction between a real-world dynamical system, simulations, and a neural twin. The workflow is structured in three phases: **(1)** The dynamical systems informs the design of the simulations. **(2)** Via the simulations, a diverse training dataset is generated, which is used for training the neural twin. **(3)** During inference, the neural twin integrates real-time measurement data to remain on-trajectory. **(4)** The state estimation of the neural twin can optionally be used for decision-making.

3.2 PARALLEL-IN-TIME NEURAL TWINS

Problem description. Consider a transient dynamical system where the state space is a compact subset $\mathcal{X} \subset \mathbb{R}^{d_x}, d_x \in \mathbb{N}$ and $\boldsymbol{x}_t \in \mathcal{X}$ denotes the system state at time t. We assume a $d_m \in \mathbb{N}$ dimensional compact measurement space $\mathcal{M} \in \mathcal{R}^{d_m}$. At each timestep $t \in \mathbb{N}$ the system provides a measurement $\boldsymbol{m}_t$ from the emission model p_e, where $\boldsymbol{m}_t \sim p_e(\boldsymbol{m}_t \mid \boldsymbol{x}_t)$.

PAINT. The goal of our generative model p_θ is to sample trajectories from $p(\boldsymbol{x}_{[0,t]} \mid \boldsymbol{m}_{[0,t]})$. At the core of our method is *joint modeling* of this distribution. In contrast, *autoregressive models* factorize this distribution which may lead to drifting (see Section 3.3).
As the number of timesteps may become very large, PAINT models the joint probability over a window (see Figure 1) of width h and may predict n timesteps into the future:

$$p_\theta(\boldsymbol{x}_{[t-h,t+n]} \mid \boldsymbol{m}_{[t-h,t]}). \tag{2}$$

In this setup, future measurements can inform past states to obtain temporal consistency. Furthermore, there is no reliance on the existence or an assumption about an initial condition.
PAINT is agnostic to the generative modeling framework and the neural network architecture. Common building blocks may be transformers, MLPs, CNNs (LeCun et al., 2002).

3.3 Theoretical Considerations

Autoregressive models factorize the joint distribution as a product of conditionals. Assuming the system is Markovian and conditioned on a sequence of variables $\boldsymbol{m}_{[0,t]}$, we obtain:

$$p(\boldsymbol{x}_{[0,t]} \mid \boldsymbol{m}_{[0,t]}) = p(\boldsymbol{x}_0 \mid \boldsymbol{m}_{[0,t]}) \cdot \prod_{i=1}^{t} p_\theta(\boldsymbol{x}_i \mid \boldsymbol{x}_{i-1}, \boldsymbol{m}_{[0,t]}), \tag{3}$$

where the autoregressive model learns the conditional probability $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$. Notice, this does not equal $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{m}_t)$, although e.g. Kalman filters use this form.

Autoregressive model drift. Our analysis is leaned on the analysis of error growth of dynamical systems (Orrell, 2005) and key observations made in (Hess et al., 2023). We assume a fully differentiable neural network , which learns the probability distribution $p_\theta(\boldsymbol{x}_t||\boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$. To sample from this probability distribution $\boldsymbol{x}_t \sim p_\theta(\boldsymbol{x}_t||\boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$ we rewrite the neural network to a deterministic function with a stochastic input η_t:

$$\boldsymbol{x}_t = f_\theta(\boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]}, \eta_t) \qquad \eta_t \sim p(\eta) \tag{4}$$

The Jacobian is then defined as $\boldsymbol{J}_t \coloneqq \partial f_\theta / \partial \boldsymbol{x}_t$. In contrast to Orrell (2005), this is a discrete random ordinary differential equation (RODE), i.e., an ODE with a random input at each timestep.
To show the model drift from small perturbations, we fix a path of the RODE by fixing $\eta_{[0,t]}$. For simplicity, we then write $F_\theta^t(\boldsymbol{x}_t) = f_\theta(\boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]}, \eta_t)$ where we absorb the measurement $\boldsymbol{m}_{[0,t]}$ and the stochastic input η_t into the function, such that $\boldsymbol{x}_t = F_\theta^t(\boldsymbol{x}_{t-1})$. Hence for the generation of a state $\boldsymbol{x}_t$ from a previous state $\boldsymbol{x}_k$ it follows from function composition:

$$\boldsymbol{x}_t = F_\theta^t \circ F_\theta^{t-1} \circ ... \circ F_\theta^{k+1}(\boldsymbol{x}_k) = F_\theta^{[k+1,t]}(\boldsymbol{x}_k) \tag{5}$$

The first-order Taylor approximation of a small input perturbation at $t = k$ is:

$$F_\theta^{[k+1,t]}(\boldsymbol{x}_k + \epsilon) = F_\theta^{[k+1,t]}(\boldsymbol{x}_k) + \epsilon \prod_{i=k+1}^{t} J_i(\boldsymbol{x}_{i-1}) + o(\epsilon^2) \tag{6}$$

This means, a deviation at timestep k induces a deviation multiplied by the *Jacobian product series* at timestep t. This has also been experimentally explored by Hess et al. (2023), who include visualizations of the Jacobian product series for chaotic dynamical systems.

On-trajectory. In the following we will formalize the notions of a model being "on-trajectory". For this, we rely on Assumption 1, which states that the predictive relevance of past measurements decays within a finite window h.

Assumption 1. ***Temporal decay of measurement information****: For the given dynamical system, there is a finite window size $h \in \mathbb{N}$, such that almost all information from past measurements comes from measurements within this window. Formally, for all $\epsilon > 0$ and all $t \in \mathbb{N}$ there exists an $h \ll t$ such that $||p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})|| < \epsilon$.*

Next, we proceed with our definition of on-trajectory.

Definition 1. ***On-trajectory****: Assuming unbounded model size, compute and data, a model is on-trajectory if the modeled distribution is arbitrarily close to the true distribution. Formally, for all $t \in \mathbb{N}$ and any $\epsilon > 0$ it holds that: $||p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})|| < \epsilon$.*

Parallel-in-time models are on-trajectory under Assumption 1 . We derive this in Appendix B.1. Intuitively, for each ϵ we can choose the window h large enough to guarantee arbitrary closeness. In contrast, autoregressive models are in general not on-trajectory. As a counterexample, consider an underlying chaotic system where the model ignores the measurements (for further discussion, see Hess et al. (2023) and Appendix B.2).

Over-reliance on the autoregressive state. The above analysis highlights a fundamental problem we call over-reliance on the autoregressive state. Intuitively, it states that an autoregressive state plays an ambiguous role. If the state is on-trajectory, it helps the model as it provides useful information for the next state. If the state is off-trajectory, the model may still make use of it for prediction and drift off further. In principle, the only way for the model to correct an off-trajectory autoregressive state is with informative measurements. This excludes edge cases, e.g., where the model converges to a global steady state.

Ideally, a model would detect when the autoregressive state drifts off and decrease its influence on the prediction. Depending on the informativeness of the measurements, it could increase its reliance on measurement information or capture the inherent uncertainty in a different way.

Over-reliance on autoregressive state is also influenced by the training strategy. Autoregressive models are usually trained with teacher-forcing, where the ground-truth previous state $\boldsymbol{x}_t$ is used to predict the next state $\boldsymbol{x}_{t+1}$. As these states are highly correlated, the model might learn to over-rely on $\boldsymbol{x}_t$ and under-rely on $\boldsymbol{m}_{t+1}$. This could potentially be mitigated by training techniques such as generalized teacher forcing (Hess et al., 2023) or unrolled training (Kohl et al., 2023).

How much information is taken from the autoregressive state is indicated by $\boldsymbol{J}_t(\boldsymbol{x}_t)$. To avoid autoregressive error accumulation, the Jacobian product series should ideally have a small norm. Notice that parallel-in-time models predict $\boldsymbol{x}_{[t-h,t]} = f_\theta(\boldsymbol{m}_{[k,t]}, \eta)$ without any recurrence. Consequently the Jacobians are all zero-matrices.

Advice for practitioners

The selection of an *on-trajectory* neural twin approach depends on two key factors:
(1) Measurement Informativeness. Highly informative measurements often justify the use of autoregressive models (e.g., Kalman filters). If measurements lack sufficient information for state reconstruction, *parallel-in-time models* are preferable to ensure staying on-trajectory – **but** they require the selected window's measurements to be informative.
(2) System Dynamics. If measurements provide limited information, chaotic systems typically demand *parallel-in-time models* to stay on-trajectory. For stable or periodic systems (e.g., steady-state behavior) *autoregressive models* may suffice.

4 Experiments

4.1 Models

PAINT via Flow Matching. We implement FlowPAINT as a concrete instantiation of PAINT using Flow Matching (Albergo & Vanden-Eijnden, 2022; Albergo et al., 2023a; Lipman et al., 2022; Liu et al., 2022). PAINT is agnostic to the neural network architecture. We therefore aimed for simplicity and use established building blocks of the computer vision literature (Dosovitskiy et al., 2020; Hoogeboom et al., 2023; 2024; Peebles & Xie, 2023). Further experimental details can be found in Appendix A.2.

Data-coupled stochastic interpolants. PAINT is also agnostic to the generative modeling paradigm, as long as it maps a source to a target distribution. We follow the modeling from Albergo et al. (2023b) and model the unmasking as data-coupled distribution matching. However, the masking ratio of our method is very high (25 of 16K pixels are unmasked). In early experiments we observed that the gradient signal was too weak for the vicinity of the probes. Therefore, we introduce a spatially weighted loss that puts higher weight on pixels in the neighborhood of the probe points. As stochastic interpolants and Flow Matching in principle lead to the same training setup, we use a common Flow Matching implementation (Lipman et al., 2024).

Autoregressive UNet. To compare PAINT to an autoregressive model, we use the UNet architecture from Kohl et al. (2023). We follow their proposed paradigm and condition on the probe values

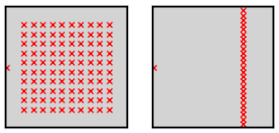
m_{t+1} by concatenating the mask and probe values to the channels. The model sizes are chosen to match parameters. FlowPAINT has 19.8M and the autoregressive UNet 20.6M. Further architectural details and hyperparameters are reported in Appendix A.2.

4.2 Dataset and evaluation

Turbulent single jet dataset. We generate a CFD dataset using the incompressible, pressure-based solver pimpleFOAM in OpenFOAM 8 (OpenCFD, 2009), with subgrid scales modeled by the Smagorinsky LES model (Pope, 2001; Fröhlich, 2006). Simulations are performed on a structured mesh, and for training purposes, a subregion is extracted and interpolated onto a regular grid. Since the state-space represents 2D velocities, we will write $\boldsymbol{u}_t$ instead of $\boldsymbol{x}_t$ with $\boldsymbol{u}_t \in \mathbb{R}^{d_1 \times d_2}$. The train/val/test splits across Reynolds numbers are provided in Appendix A.1.

Physical coherence. Due to the chaotic and rapidly diverging nature of turbulent flows, comparing a single predicted frame or a short sequence to the ground truth is not meaningful. Research in turbulence instead focuses on the flow's inherently reproducible statistical properties (Pope, 2001; Davidson, 2015). Given a possibly infinitely long trajectory $\boldsymbol{u}_{[0,t]}$, we follow a common notation in fluid dynamics and decompose $\boldsymbol{u}_{[0,t]}$ into a *time-averaged mean* component $\overline{\boldsymbol{u}} \coloneqq \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{u}_i$ and a turbulent fluctuation component $\boldsymbol{u}'_{[0,t]}$, such that $\boldsymbol{u}_{[0,t]} = \overline{\boldsymbol{u}} + \boldsymbol{u}'_{[0,t]}$. The *variance over time* is denoted as $\overline{\boldsymbol{u}'^2} \coloneqq \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} (\boldsymbol{u}_i - \overline{\boldsymbol{u}})^2$. Further, we define $E(k)$ as the *mean kinetic energy spectrum*. While these metrics are defined over an infinite-time, in practice they are computed over a large, finite horizon.

Probe constellations. The model was trained with 25 randomly sampled probe points for each data point in a batch. At inference, we investigate the two probe point constellations depicted in Figure 3. *Grid*: A 10×10 grid of 100 probe points evenly spaced across the domain (see Figure 3a). *Vertical*: A linear arrangement of 25 probe points positioned at the 3/4 axial location of the domain (see Figures 3b). In both configurations, an additional probe point is placed at the center of the inlet of the jet to capture the inlet velocity.



(a) Grid. (b) Vertical.

Figure 3: Probe positions.

4.3 Results

Table 1: Results for physical coherence of FlowPAINT (ours) and the autoregressive UNet baseline (AR UNet). The MAE, MSE and RMSE are taken between the ground truth and the predicted quantity. $\overline{\mathbf{u}}$ and $\overline{\mathbf{u}'^2}$ denote the mean and variance over time. $E(k)$ is the predicted kinetic energy spectra. MSE$(u_{[0,t]})$ is the mean squared error between the predicted and the ground-truth trajectory as a whole. All values are taken along 900 timesteps in the test trajectories. The values after $\pm$ are standard deviations from averaging over 10 seeds. Generally, the higher Reynolds number seems to be more difficult for both models.

Model	Re	**Probes**	MAE$(\overline{\mathbf{u}}) \downarrow$	MAE$(\overline{\mathbf{u}'^2}) \downarrow$	MSE$([\mathbf{u}_{[0,t]}]) \downarrow$	RMSE$(E(k)) \downarrow$
			$\times 10^{-3}$	$\times 10^{-5}$	$\times 10^{-5}$	$\times 10^{-6}$
FlowPAINT	2100	grid	1.2 ± 1.6	9.9 ± 21.7	8.1 ± 3.6	6.85
AR UNet	2100	grid	9.5 ± 14.3	98.7 ± 110.4	158.3 ± 46.8	70.9
FlowPAINT	2100	vertical	2.2 ± 3.1	30.2 ± 38.4	49 ± 21	20.1
AR UNet	2100	vertical	9.2 ± 13.8	97.8 ± 108.8	157.3 ± 70.0	71.8
FlowPAINT	1100	grid	0.9 ± 1.3	2.2 ± 5.5	2 ± 1	1.98
AR UNet	1100	grid	3.5 ± 5.6	17.3 ± 25.9	28.8 ± 8.3	12.4
FlowPAINT	1100	vertical	1.1 ± 1.6	6.3 ± 11.1	8.9 ± 3.4	3.68
AR UNet	1100	vertical	3.4 ± 5.5	16.9 ± 25.5	28.8 ± 8.1	12.2

Quantitative results. The quantitative results are reported in Table 1. FlowPAINT outperforms the autoregressive UNet across all metrics, probe constellations and Reynolds numbers in the test set. We conclude that FlowPAINT is more physically coherent than the probe-conditioned autoregressive baseline. Further results and analyses can be found in Appendix C.

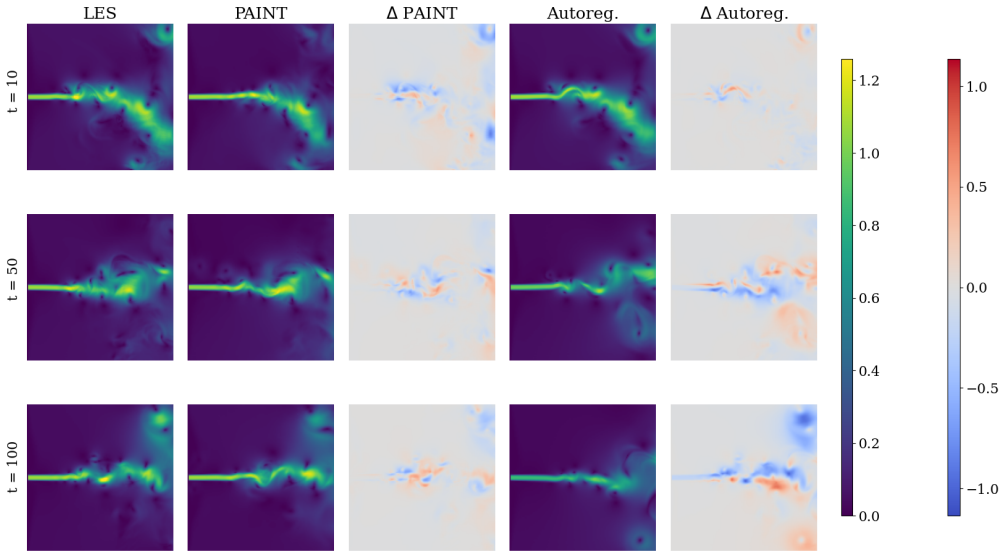


Figure 4: Ground truth vs. predicted states at different timesteps for $Re = 2100$ using the *vertical* probe constellation. PAINT (ours) does not accumulate errors over time. For the autoregressive UNet a pronounced drift over time is observable. All values are normalized by the mean inlet velocity.

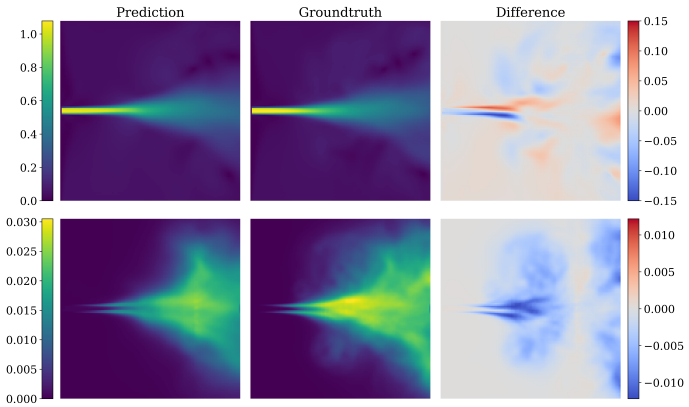
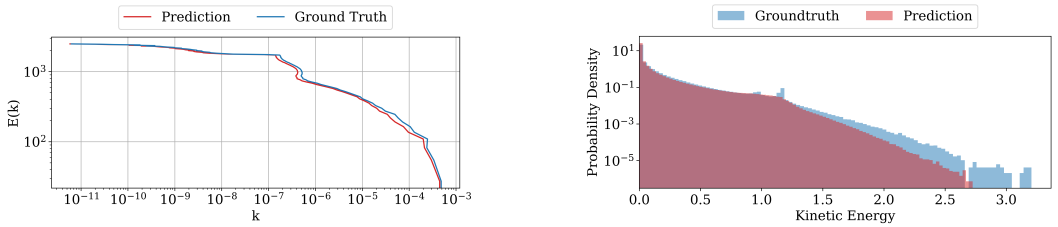


Figure 5: Mean velocity $\overline{u}$ and variance $\overline{u'^2}$ of the reconstructed $Re = 2100$ test trajectory using the *vertical* probe constellation compared with the ground truth. All values are normalized by the mean inlet velocity.

Recovery of the physical flow characteristics. We observe the models ability to reconstruct physical plausible states from different constellations of probe points (see Figures 5 and 6). The model's ability to reproduce the correct physical statistics seems to depend directly on the number and position of the sample points (as can be seen when comparing Figure 5 with Figures 10a, 11a and 12a).

Parallel-in-time model maintains stable error, while autoregressive baseline drifts. The autoregressive approach exhibits a drift in MSE when rolled out (compare Figures 4, 7, 13,14 and 15). Importantly, the autoregressive model benefits from a known, correct initial state, which is often unknown in practice.

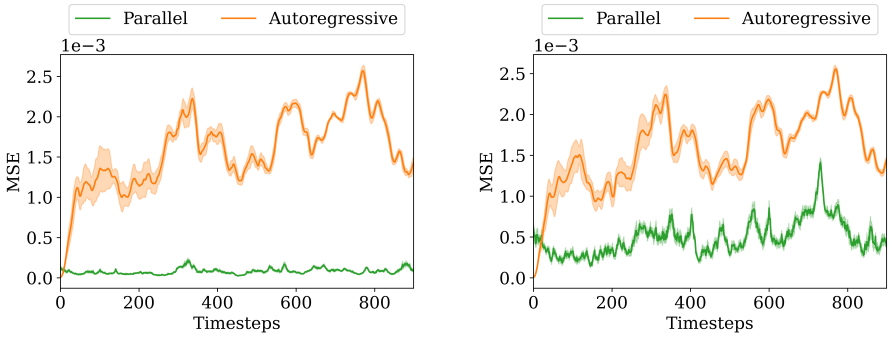
Sampling a sequence vs. a single state. PAINT can be used in two variants. The first samples a connected sequence, usually $\boldsymbol{x}_{[t-h,t]} \sim p(\boldsymbol{x}_{[t-h,t])} \mid \boldsymbol{m}_{[t-h,t]}$ (see Figure 16). This sequence is smooth over time and can be used by practitioners to interpret results.
The second variant samples a single timestep. It is used for state estimation in a sliding window



(a) Time-averaged kinetic energy spectrum.

(b) Histogram of the kinetic energy.

Figure 6: Statistical analysis of reconstructed flow fields for Reynolds number 2100 and a *vertical* probe point constellation.



(b) *Grid* probe constellation.

(a) *Vertical* probe constellation.

Figure 7: MSE over time of ground-truth vs. predicted trajectories for the $Re = 2100$ test trajectory and different probe point constellations. Interestingly, the MSE's of the two autoregressive trajectories are very similar, even if more informative probes are provided for the grid. The autoregressive model indicates over-reliance on the autoregressive state.

fashion by sampling $\boldsymbol{x}_t \sim p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]})$. In this case a sampled sequence of consecutive states is not smooth over time.

Measure of uncertainty. One can obtain a measure for uncertainty by sampling several times and computing the variance. In Figure 9 it is visible that the model is most certain around the probe positions and most uncertain to the left and the right.

Compute at training and inference. FlowPAINT needs substantially more training resources and is slower at inference than the autoregressive baseline. It was trained on 12 A100 (64 GB VRAM) for 30 hours, whereas the UNet was trained on a single H100 (80GB VRAM) for 24 hours. Notably, the UNet takes about 66 milliseconds and FlowPAINT around 6.6 seconds for 20 denoising steps. Further details can be found in Appendix A.2.

5 Conclusion

In this paper we introduced parallel-in-time neural twins (PAINT) a novel data-driven method for reconstructing dynamical systems from real-time measurements. PAINT employs a generative neural network to model the joint conditional probability of system states. Our method provable stays on-trajectory and does not rely on an initial condition. The empirical validation with FlowPAINT on 2D turbulent fluid dynamics confirms its effectiveness. This paper represents an advancement in designing interpretable and widely applicable neural twins for complex dynamical systems.

Limitations and future work The biggest limitation of our method are the high computational costs, which deserves further investigation. Another limitation is that the sliding window approach does not generate continuous generations. This could potentially be mitigated by stitching techniques (Wei et al., 2019).

Ethics statement. More broadly, our work aims to advance the field of machine learning and may contribute to its broader societal impact. More specifically, our work advances reconstruction of dynamical systems, with a wide range of potential applications. While our method or future derivatives of it might enable transformative societal benefits it also introduces risks of misconduct or dual-use risks. To mitigate the risk of misconduct, we will strive to be transparent about the capabilities and limitations of this method and encourage the use in real-world applications only after rigorous testing. We welcome critical discussions on further safeguarding such technologies against adversarial use.

Reproducibility statement. We provide code to reproduce the main results in the supplementary material. Additionally, we report hyperparameters, and important implementation details to facilitate the reproduction of our results. Upon publication, the code will be made publicly available on GitHub.

Usage of LLMs. LLMs assisted in ideation and in refining and optimizing formulations on a sentence- and paragraph-level.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023a.

Michael S Albergo, Mark Goldstein, Nicholas M Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. *arXiv preprint arXiv:2310.03725*, 2023b.

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *CoRR*, 2024a.

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024b.

Benedikt Alkin, Tobias Kronlachner, Samuele Papa, Stefan Pirker, Thomas Lichtenegger, and Johannes Brandstetter. Neuraldem–real-time simulation of industrial particulate flows. *arXiv preprint arXiv:2411.09678*, 2024c.

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.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nat.*, 619(7970):533–538, 2023. doi: 10.1038/S41586-023-06185-3.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Anna Vaughan, Jayesh K. Gupta, Kit Thambiratnam, Alex Archibald, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. Aurora: A foundation model of the atmosphere. *CoRR*, abs/2405.13063, 2024. doi: 10.48550/ARXIV.2405.13063.

Dibyajyoti Chakraborty, Haiwen Guan, Jason Stock, Troy Arcomano, Guido Cervone, and Romit Maulik. Multimodal atmospheric super-resolution with deep generative models. *arXiv preprint arXiv:2506.22780*, 2025.

Nagendra Kumar Chaurasia and Shubhankar Chakraborty. Reconstruction of the turbulent flow field with sparse measurements using physics-informed neural network. *Physics of Fluids*, 36(8), 2024.

Antonio Cuéllar, Alejandro Güemes, Andrea Ianiro, Óscar Flores, Ricardo Vinuesa, and Stefano Discetti. Three-dimensional generative adversarial networks for turbulent flow estimation from wall measurements. *Journal of Fluid Mechanics*, 991:A1, 2024.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Hiep Vo Dang, Joseph B Choi, and Phong CH Nguyen. Flronet: Deep operator learning for high-fidelity fluid flow field reconstruction from sparse sensor measurements. *arXiv preprint arXiv:2412.08009*, 2024.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Peter Davidson. *Turbulence: an introduction for scientists and engineers*. Oxford university press, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

Jochen Fröhlich. *Large Eddy Simulation turbulenter Strömungen: Mit 145 Abbildungen und 14 Tabellen*. Springer, 2006.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. URL `https://diffusionflow.github.io/`.

Michael Grieves. Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1(2014):1–7, 2014.

Alejandro Güemes, Carlos Sanmiguel Vila, and Stefano Discetti. Super-resolution gans of randomly-seeded fields. *arXiv preprint arXiv:2202.11701*, 2022.

Meet Hemant Parikh, Xiantao Fan, and Jian-Xun Wang. Conditional flow matching for generative modeling of near-wall turbulence with quantified uncertainty. *arXiv e-prints*, pp. arXiv–2504, 2025.

Florian Hess, Zahra Monfared, Manuel Brenner, and Daniel Durstewitz. Generalized teacher forcing for learning chaotic dynamics. *arXiv preprint arXiv:2306.04406*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pp. 13213–13232. PMLR, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. *arXiv preprint arXiv:2410.19324*, 2024.

Mohammad Yasin Hosseini and Yousef Shiri. Flow field reconstruction from sparse sensor measurements with physics-informed neural networks. *Physics of Fluids*, 36(7), 2024.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2021a.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021b.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Hyojin Kim, Junhyuk Kim, Sungjin Won, and Changhoon Lee. Unsupervised deep learning for super-resolution reconstruction of turbulence. *Journal of Fluid Mechanics*, 910:A29, 2021.

Sukkeun Kim, Ivan Petrunin, and Hyo-Sang Shin. A review of bayes filters with machine learning techniques and their applications. *Information Fusion*, 114:102707, 2025. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2024.102707. URL https://www.sciencedirect.com/science/article/pii/S1566253524004858.

Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv preprint arXiv:2309.01745*, 2023.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022a.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022b.

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

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*, 2020b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

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

Robert M May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560): 459–467, 1976.

Vivek Oommen, Siavash Khodakarami, Aniruddha Bora, Zhicheng Wang, and George Em Karniadakis. Learning turbulent flows with generative models: Super-resolution, forecasting, and sparse flow reconstruction. *arXiv preprint arXiv:2509.08752*, 2025.

OpenFOAM OpenCFD. The open source cfd toolbox. *User Guide, OpenCFD Ltd*, 770, 2009.

D Orrell. Estimating error growth and shadow behavior in nonlinear dynamical systems. *International Journal of Bifurcation and Chaos*, 15(10):3265–3280, 2005.

Defne Ege Ozan, Andrea Nóvoa, and Luca Magri. Data-assimilated model-based reinforcement learning for partially observed chaotic flows. In *International Conference on Computational Science*, pp. 65–72. Springer, 2025.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Stephen B Pope. Turbulent flows. *Measurement Science and Technology*, 12(11):2020–2021, 2001.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Florian Sestak, Artur Toshev, Andreas Fürst, Günter Klambauer, Andreas Mayr, and Johannes Brandstetter. Lam-slide: Latent space modeling of spatial dynamical systems via linked entities. *arXiv preprint arXiv:2502.12128*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Luning Sun and Jian-Xun Wang. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, 10(3):161–169, 2020.

Adam Thelen, Xiaoge Zhang, Olga Fink, Yan Lu, Sayan Ghosh, Byeng D Youn, Michael D Todd, Sankaran Mahadevan, Chao Hu, and Zhen Hu. A comprehensive review of digital twin—part 1: modeling and twinning enabling technologies. *Structural and Multidisciplinary Optimization*, 65 (12):354, 2022.

Juan Diego Toscano, Theo Käufer, Zhibo Wang, Martin Maxey, Christian Cierpka, and George Em Karniadakis. Inferring turbulent velocity and temperature fields and their statistics from lagrangian velocity measurements using physics-informed kolmogorov-arnold networks. *arXiv preprint arXiv:2407.15727*, 2024.

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.

Lyu Wei, Zhou Zhong, Chen Lang, and Zhou Yi. A survey on image and video stitching. *Virtual Reality & Intelligent Hardware*, 1(1):55–83, 2019.

Norbert Wiener. The linear filter for a single time series. In *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, pp. 81–103. The MIT Press, 1949.

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.

A EXPERIMENTAL DETAILS

A.1 DATASET

Turbulent single jet dataset. We use an in-house dataset to ensure its quality and match the specific requirements of our study. As we have the capability to generate reliable high-fidelity data, additional external datasets were not needed. We consider the domain shown in Figure 8, which produces a two-dimensional turbulent free jet. The inlet is prescribed by a turbulent power-law velocity profile, with the system dynamics controlled by the mean inlet velocity. The Reynolds number is defined with respect to the inlet velocity as

$$Re = \langle u \rangle_{\text{inlet}} h_{\text{inlet}} \nu^{-1}. \tag{7}$$

The dataset is generated using the incompressible, pressure-based solver `pimpleFOAM` in `OpenFOAM 8` (OpenCFD (2009)), with subgrid scales modeled by the Smagorinsky LES model (Pope, 2001; Fröhlich, 2006). Simulations are performed on a structured mesh consisting of 19040 cells, and for training purposes, a subregion is extracted and interpolated onto a regular 128 × 128 grid to facilitate adaptability.. The numerical simulations made use of a timestep size of $\Delta t = 6.5 \times 10^{-4}$s. For training purposes, the data are sampled every 70th timestep, resulting in an effective temporal resolution of 4.55×10^{-2}s. For each Reynolds number, the dataset contains 1000 such snapshots.

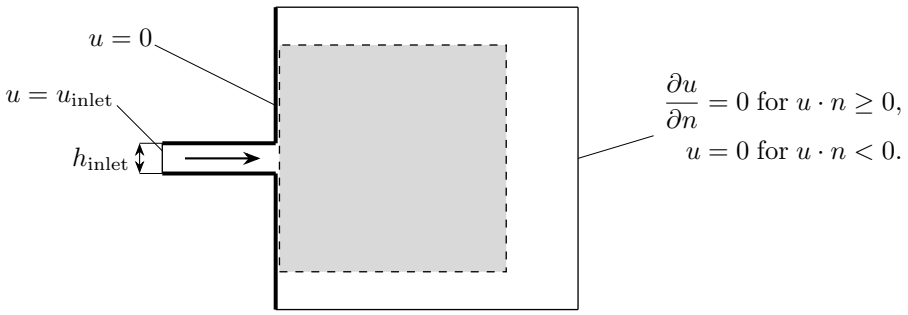


Figure 8: Computational domain of the CFD simulation, featuring a thin channel with a prescribed inlet velocity profile, transitioning into an expanded outflow region zone. The channel and outflow orifice walls enforce a no-slip boundary condition, while the top, bottom and right boundaries of the outflow region implement a conditional Neumann/Dirichlet outflow condition. The light-grey subregion denotes the part of the CFD domain considered for model training. The arrow shows the imposed inlet flow direction.

A note on 2D turbulence. In fluid mechanics, the behavior of turbulent flows differs fundamentally between three and two dimensions. In three dimensions, turbulence is characterized by the *energy cascade*: the largest eddies form due to inertial instabilities but tend to exist only briefly, breaking up into progressively smaller eddies. This cascade continues until viscous forces dissipate the smallest scales.

In contrast, in two dimensions, the energy cascade is reversed, transferring energy from smaller to larger eddies, which produces coherent, long-lived vortices. Intuitively, this can be explained by *vortex stretching*. Consider a thin tube of vorticity: if a velocity gradient along the tube axis is present, the tube stretches, and due to conservation of angular momentum, the vorticity magnitude increases. This mechanism is absent in two dimensions because there is no third dimension along which stretching can occur. As a result, vortices cannot break up and tend to merge into larger structures if they have the same rotational sense, which explains the inverse energy cascade.

Mathematically, this is seen in the vorticity transport equation derived from the curl of the Navier–Stokes equations. In 2D, the velocity and vorticity vectors are perpendicular, which makes the vortex stretching term identically zero.

In reality, there is no perfectly two-dimensional turbulence; even quasi-2D flows, such as atmospheric flows, exhibit three-dimensional structures at smaller scales. Nevertheless, we are confident that the proposed method, having demonstrated robust performance in two-dimensional turbulence,

will generalize effectively to three-dimensional turbulent flows. For a more detailed discussion of two-dimensional turbulence, see Davidson (2015).

Train/val/test splits. We consider 18 different training trajectories with varying Reynolds numbers, as explained above. The Reynolds numbers used range from 700 to 2400 in steps of 100. The split is performed at random, but avoiding that any validation or test data is placed in the extrapolation regime. The exact split can be seen in Table 2.

Table 2: Reynolds numbers by split.

Reynolds	train	val	test
700	X		
800	X		
900	X		
1000	X		
1100			X
1200	X		
1300	X		
1400	X		
1500	X		
1600		X	
1700	X		
1800	X		
1900	X		
2000	X		
2100			X
2200	X		
2300	X		
2400	X		

A.2 Implementation and architecture

Architecture and hyperparameters. FlowPAINT and the autoregressive UNet were both implemented in Pytorch (Paszke et al., 2017). For FlowPAINT we trained in float16 mixed precision. The most important training parameters are provided in Table 3. Our code also uses FlashAttention (Dao et al., 2022; Dao, 2024). For both models we use 20 denoising steps for generating samples.

Autoregressive UNet. For the UNet we followed (Kohl et al., 2023) and took the implementation of the public Github repository [2]. We matched the number of parameters by increasing the default model dimension from 128 to 224. For probe-conditioning, we followed the same recipe as for prior timestep conditioning and added mask and probe values as additional channels.

Compute requirements. FlowPaint is expensive during training, as it needs to reconstruct a video in parallel. We had limited access to a cluster environment and trained FlowPaint on 3 nodes each consisting of 4 NVIDIA Ampere A100 GPUs with 64GB VRAM. Training took around 30 hours for 100K iterations.
The autoregressive baseline is comparably cheap. It was trained on a single Nvidia H100-SXM-80GB for ca. 24 hours. The inference times are reported in the main paper.

B Theory

B.1 Parallel-in-time models are on-trajectory

Here we provide a simple derivation to show that parallel-in-time models are on-trajectory under Assumption 1. We will show that for an appropriate window size h, for all $t \in \mathbb{N}$ and any $\epsilon_1 > 0$:

$$||p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})|| < \epsilon_1. \tag{8}$$

[2] https://github.com/tum-pbs/autoreg-pde-diffusion

Table 3: Experimental details for FlowPaint and the autoregressive baseline.

		FlowPaint	AR baseline
Data			
	History length	16	2
	Forward prediction	8	1
Optimization			
	Optimizer	AdamW	AdamW
	AdamW decay	0.05	0.05
	Learning rate	10^{-4}	10^{-4}
	Learning rate start	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$
	Learning rate warmup steps	10000	10000
	Learning rate end	10^{-5}	10^{-5}
	Train steps	100K	100K
	Batch size	144	144

Table 4: Architectural details of FlowPaint.

Encoding		
	num_layers of 3x3x3 conv	3
	patch size in pixels	4x4
Transformer		
	model_dim	192
	layers	10
	num_heads	3
	Spatial Block	Yes
	Temporal Block	Yes
	Temporal Convolution	No
Decoding		
	num_layers of 3x3x3 conv	3

First, from Assumption 1 choose a window size h such that:

$$||p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})|| < \epsilon_2 < \frac{\epsilon_1}{2}. \tag{9}$$

Then, from the Universal Function Approximator Theorem (Cybenko, 1989), choose an approximation of the true distribution as follows:

$$||p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-h,t]})|| < \epsilon_3 < \frac{\epsilon_1}{2}. \tag{10}$$

Summing up the individual ϵ_1 and ϵ_2 the total approximation error is: $\epsilon_2 + \epsilon_3 < \frac{\epsilon_1}{2} + \frac{\epsilon_1}{2} < \epsilon_1$.

B.2 In general, autoregressive models are not on-trajectory

Here we show a more detailled counterexample to illustrate that autoregressive model of the form

$$p(\boldsymbol{x}_{[0,t]} \mid \boldsymbol{m}_{[0,t]}) = p(\boldsymbol{x}_0 \mid \boldsymbol{m}_{[0,t]}) \cdot \prod_{i=1}^{t} p_\theta(\boldsymbol{x}_i \mid \boldsymbol{x}_{i-1}, \boldsymbol{m}_{[0,t]}), \tag{11}$$

do not generally fulfill the on-trajectory property. We construct this example while demonstrating the over-reliance on autoregressive state. Being on-trajectory is defined as: for all $t \in \mathbb{N}$ and any $\epsilon > 0$:

$$||p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})|| < \epsilon. \tag{12}$$

From Section 3.3 it is known that autoregressive models accumulate errors with the product of Jacobians, similar to classical ODEs (Orrell, 2005).

Logistic map. Let $f : \mathbb{R} \mapsto \mathbb{R}$ be the logistic map, which is chaotic for $r = 3.8$ (May, 1976):

$$f(x_{t-1}, \eta_t) = r \cdot x_{t-1} \cdot (1 - x_{t-1}) \tag{13}$$

Then we assume the model f_θ ignores the measurements and learns to predict $\boldsymbol{x}_t$ from $\boldsymbol{x}_{t-1}$ alone, but with a small error $\epsilon' > 0$.

$$f_\theta(x_{t-1}, m_{[0,t]}, \eta_t) = r \cdot x_{t-1} \cdot (1 - x_{t-1}) + \epsilon' \tag{14}$$

This is an extreme case of over-reliance on autoregressive state, namely solely relying on it and not relying on the measurements at all. For infinitesimal perturbations in x_{t-1} exponentiate over time, causing $p_\theta(x_t \mid m_{[0,t]})$ to diverge from the true trajectory distribution $p(x_t \mid m_{[0,t]})$ even for arbitrarily small ϵ'. Thus, the autoregressive factorization is not on-trajectory because the model's sequential predictions amplify initial errors, violating the required closeness condition for all $t > 0$.

C ADDITIONAL RESULTS

We show additional results for other Reynolds numbers and probe constellations in Figures 9, 10, 11, 12, 14, 13, 15, and 16.

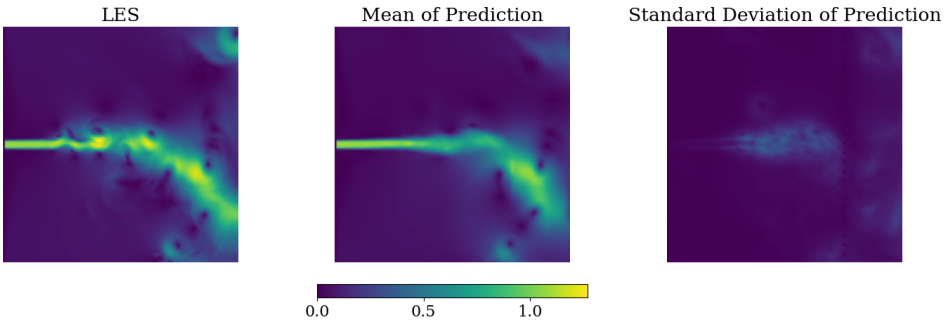
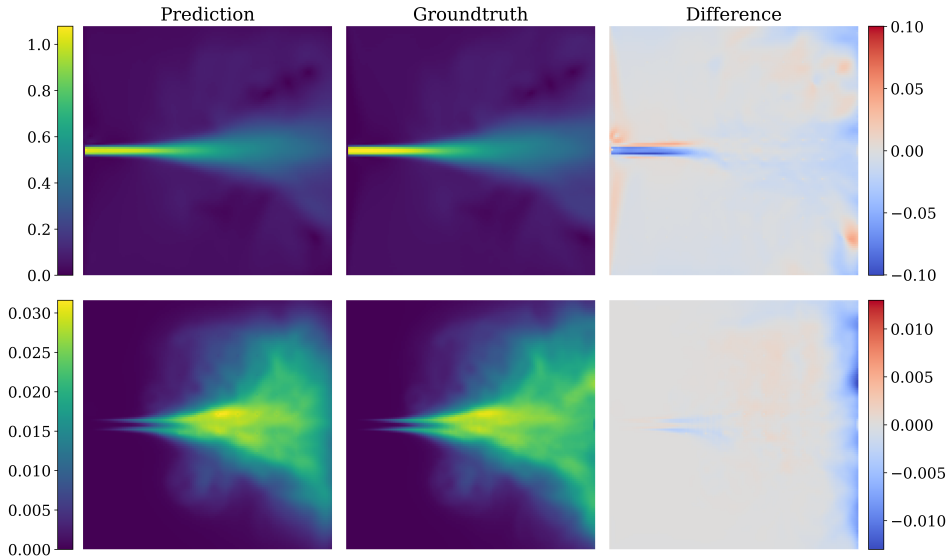
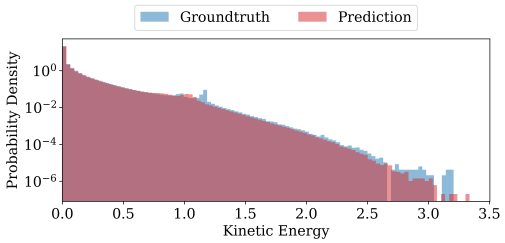


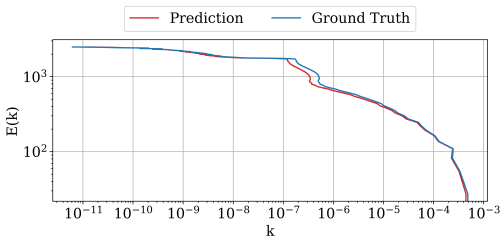
Figure 9: Left: Exemplary groundtruth velocity snapshot, for Reynolds number 2100 and using the *vertical* probe point constellation. Center: Mean of the predictions over 10 independent seeds. Right: Standard deviation of the predictions over the same 10 seeds. All values are normalized by the mean inlet velocity.



(a) Mean velocity ($\overline{u}$) and variance ($\overline{u'^2}$) of the reconstructed trajectories compared with the ground truth. All values are normalized by the mean inlet velocity.

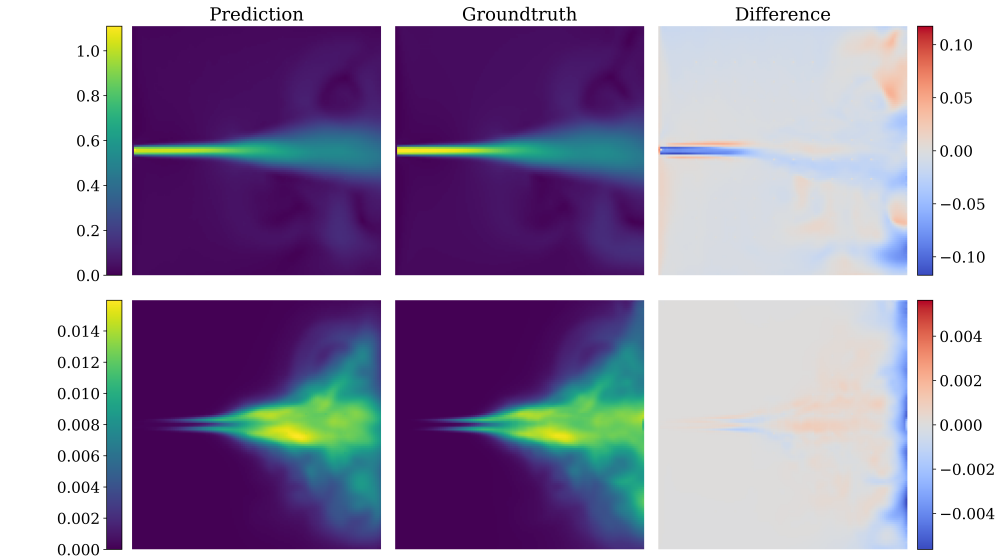


(b) Histogram of time-averaged kinetic energy.

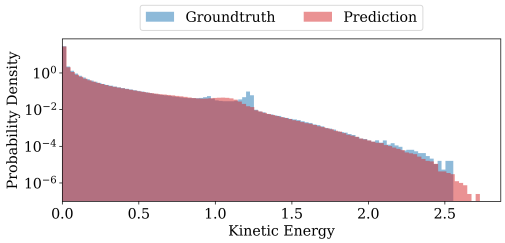


(c) Time-averaged kinetic energy spectrum.

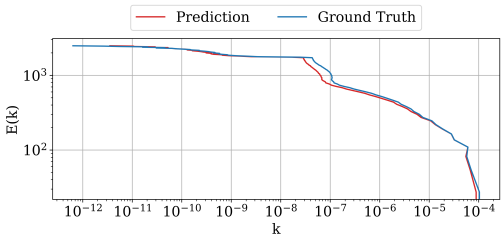
Figure 10: Statistical analysis of reconstructed flow fields for Reynolds number 2100 and a *grid* probe point constellation.



(a) Mean velocity ($\overline{u}$) and variance ($\overline{u'^2}$) of the reconstructed trajectories compared with the ground truth. All values are normalized by the mean inlet velocity.

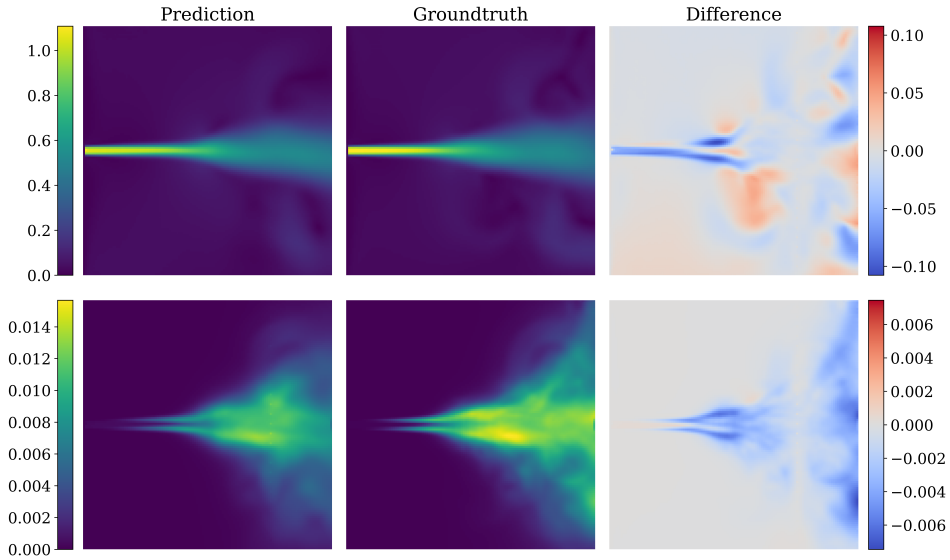


(b) Histogram of time-averaged kinetic energy.

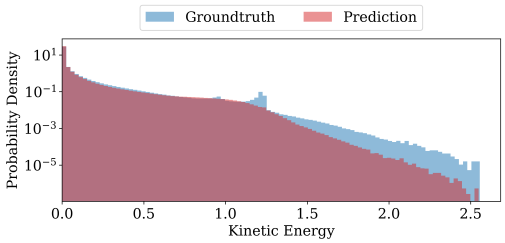


(c) Time-averaged kinetic energy spectrum.

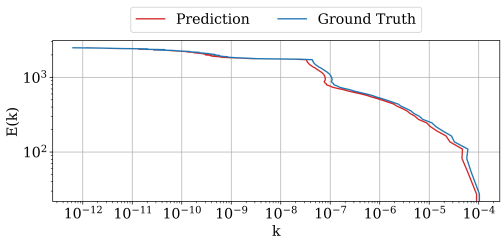
Figure 11: Statistical analysis of reconstructed flow fields for Reynolds number 1100 and a *grid* probe point constellation.



(a) Mean velocity ($\overline{u}$) and variance ($\overline{u'^2}$) of the reconstructed trajectories compared with the ground truth. All values are normalized by the mean inlet velocity.

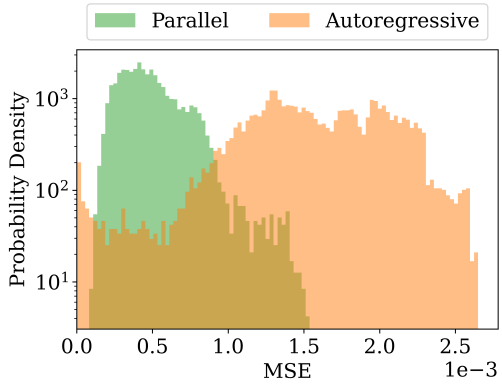


(b) Histogram of time-averaged kinetic energy.

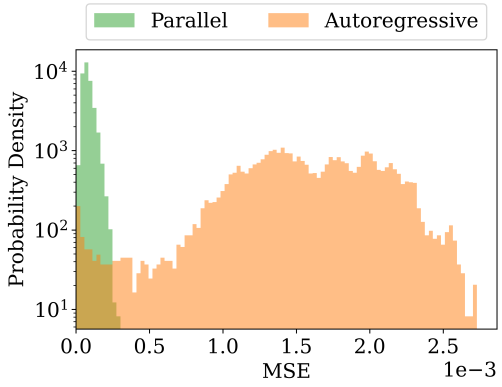


(c) Time-averaged kinetic energy spectrum.

Figure 12: Statistical analysis of reconstructed flow fields for Reynolds number 1100 and a *vertical* probe point constellation.

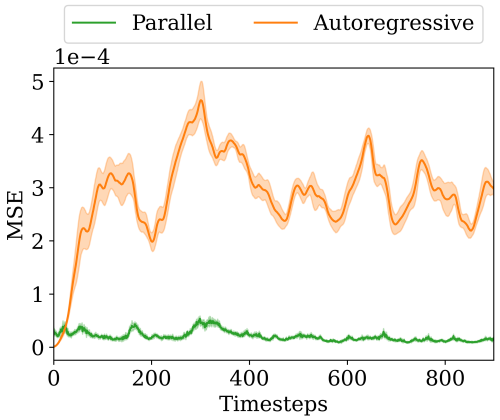


(a) *Vertical* probe constellation.

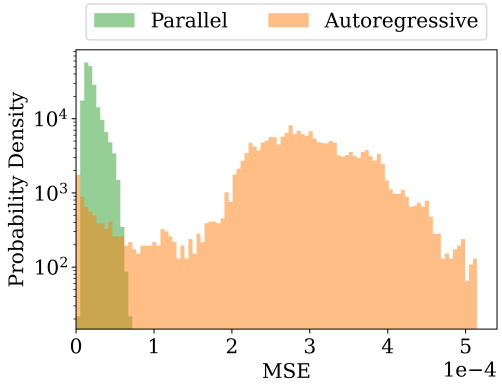


(b) *Grid* probe constellation.

Figure 13: Histogram of all MSE values of the Reynolds number 2100 test trajectory and a *grid* probe point constellation.

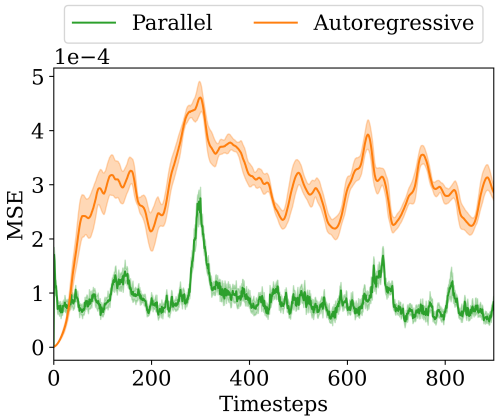


(a) Temporal evolution of the mean and standard deviation of the MSE when comparing the parallel-in-time and autoregressive approaches.

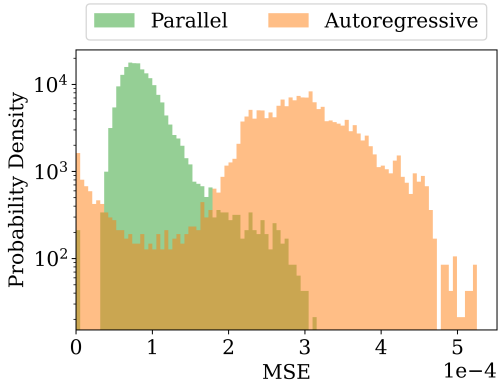


(b) Histogram of all MSE values accumulated over the rollout, comparing the parallel-in-time and autoregressive approaches.

Figure 14: Comparison of error characteristics between the parallel-in-time and autoregressive reconstruction approaches for Reynolds number 1100 and a *grid* probe point constellation.

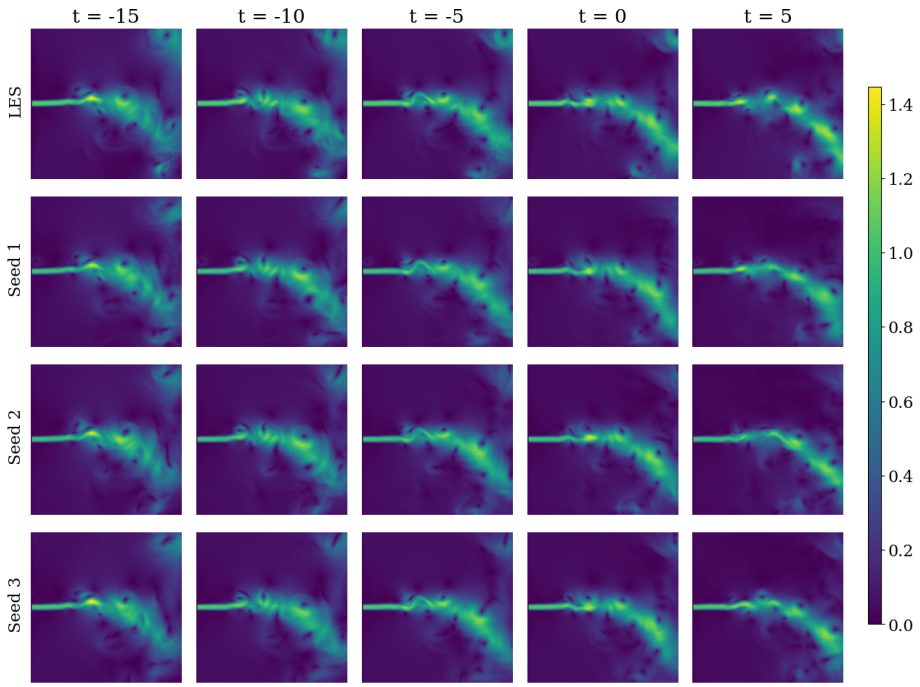


(a) Temporal evolution of the mean and standard deviation of the MSE when comparing the parallel-in-time and autoregressive approaches.

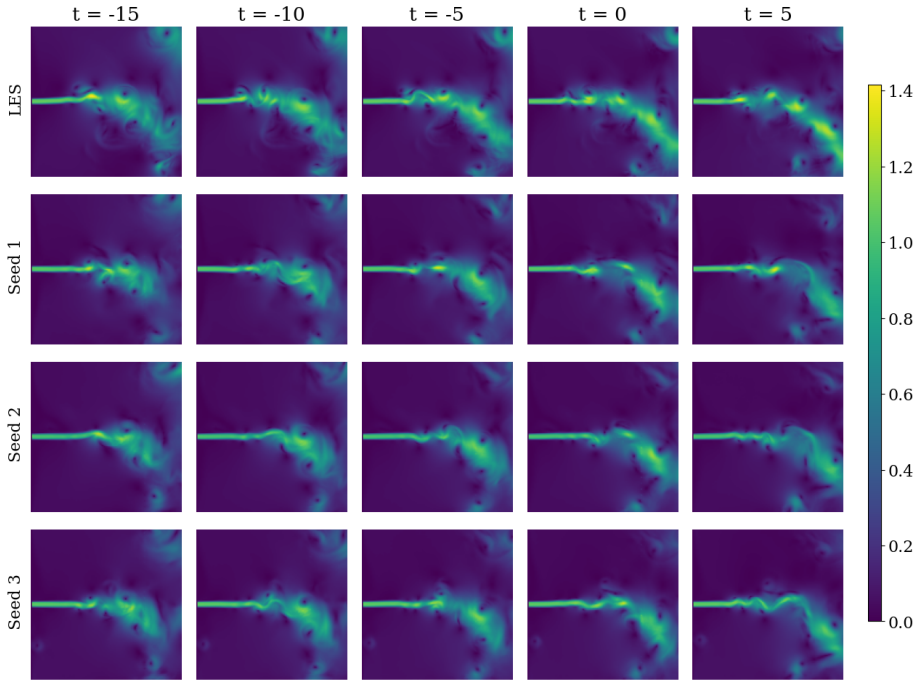


(b) Histogram of all MSE values accumulated over the rollout, comparing the parallel-in-time and autoregressive approaches.

Figure 15: Comparison of error characteristics between the parallel-in-time and autoregressive reconstruction approaches for Reynolds number 1100 and a *vertical* probe point constellation.



(a) *Grid* probe constellation.



(b) *Vertical* probe constellation.

Figure 16: The top rows show five exemplary LES ground truth snapshots. Time steps with $t \leq 0$ correspond to reconstructions based on measurements, while time steps with $t > 0$ represent predictions with no measurements present. The following three rows display reconstructions from three independent random seeds, each showing a connected sequence generated based on the probe information provided. All values are normalized by the mean inlet velocity.