PAINT: Parallel-in-time Neural Twins for Dynamical System Reconstruction

Andreas Radler [*1] **Vincent Seyfried** [*2 3] **Johannes Brandstetter** [1 4] **Thomas Lichtenegger** [2 3 4]

Abstract

Neural surrogates have shown great potential for simulating dynamical systems in real time. We envision *Neural Twins* as an evolution of neural surrogates that continuously consume measurements at test time, acting as digital replicas of real systems and enabling context-specific decision-making. We argue that a natural requirement for Neural Twins is that they remain *on-trajectory*: staying close to the true system state rather than drifting due to compounding errors. Yet most deep learning approaches for reconstructing high-dimensional dynamical systems are *autoregressive*: they predict states from previously predicted ones, which makes them inherently susceptible to such drift. We find that *parallel-in-time* modeling is a viable but underexplored alternative. Rather than sequential rollout, these models jointly predict a full window of states from a corresponding window of measurements. Such models are conceptually simple, scale to state spaces with thousands of dimensions, and avoid rollout degradation. We show theoretically and empirically that autoregressive models generally fail to satisfy the on-trajectory property, whereas parallel-in-time models satisfy it by construction. We call this family of methods **PAINT** (**Par**allel-**i**n-time **N**eural **T**wins). On a challenging two-dimensional turbulent fluid dynamics problem, our PAINT instantiation using Flow Matching closely follows the trajectory and reconstructs states from sparse measurements with high fidelity. These results establish parallel-in-time modeling as a viable alternative to autoregressive approaches and motivate further investigation of this direction.

[1]LIT AI Lab, Institute for Machine Learning, JKU Linz, Austria [2]Department of Particulate Flow Modelling, JKU Linz, Austria [3]Christian Doppler Laboratory for Data-Assisted Simulations of Complex Flows, Linz, Austria [4]Emmi AI GmbH, Linz, Austria. Correspondence to: Andreas Radler <radler@ml.jku.at>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

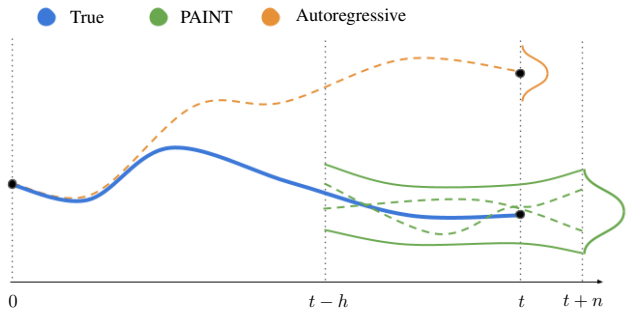


Figure 1. Conceptual depiction of trajectories from the ground truth (blue), our parallel-in-time method PAINT (green) and a naive autoregressive model (orange). Dashed lines represent sampled trajectories from the probability distribution (solid lines) given by the respective model.
The models ingest real world measurements (not depicted) at test time to reconstruct the true trajectory. Autoregressive models predict the probability of the next state given an autoregressive state additional to real world measurements. They typically depend on an initial state and are prone to drift off due to *over-reliance on their autoregressive state* (see Section 3.4). In contrast, parallel-in-time models are "on-trajectory" and do not depend on an initial state. They take a window of measurements and predict a distribution of subtrajectories which can be sampled and used for state and uncertainty estimation.

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.

Neural surrogates are neural network based predictive models which 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., 2022; 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. Almost all deep learning approaches to high-dimensional dynamical system reconstruction rely on autoregressive (AR) models: a network is trained to predict the next state from its previous prediction. While conceptually simple, this paradigm carries inherent difficulties. AR models require an initial state and are prone to error accumulation where small prediction errors compound over time (Orrell, 2005; Parthipan et al., 2024). This necessitates complex stabilization schemes such as scheduled sampling (Bengio et al., 2015) or the pushforward trick (Brandstetter et al., 2022).

In this work, we argue that a parallel-in-time approach offers a compelling and largely unexplored alternative for neural twin design. We instantiate this idea in **Pa**rallel-in-**t**ime **N**eural **T**wins (**PAINT**), a family of methods for reconstructing dynamical systems from measurements. Rather than predicting states sequentially, PAINT trains a generative model to directly capture the joint conditional distribution of a window of sequence of states given a window of measurements. At inference, it reconstructs sequences of states from measurements via a sliding-window approach. PAINT is conceptually simple, requires no stabilization techniques and no 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 AR models may possess this property under certain conditions, 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 AR model taking too much information from a drifted AR state to predict the next state.

Empirically, we show that PAINT leads to physically coherent predictions 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.

Summary of contributions.

- We introduce PAINT, a parallel-in-time framework for neural twins, and prove that it is on-trajectory under a finite temporal decay assumption on measurement information. We further show that autoregressive models do not satisfy this property in general, as their errors compound through a Jacobian product series, analogous to error growth in chaotic ODEs.
- We identify *over-reliance on the autoregressive state* as a common failure mode of autoregressive models, where the model uses too much information from a previous, potentially drifted state when predicting the next state. We show that parallel-in-time models avoid this failure mode by construction.
- We instantiate PAINT via Flow Matching and evaluate it on a 2D turbulent fluid dynamics problem with a 32K-dimensional state space. PAINT yields physically coherent reconstructions that remain stable over arbitrary long horizons, while the AR baseline drifts and leads to lower physical coherence even if regularization is applied. [1]

2. Background and related work

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 $[k, \ldots, t]$ including the endpoints k and t are denoted in brackets as $[k, t]$. We may also use brackets as subscripts over sequences, e.g., $\boldsymbol{x}_{[k,t]} = [\boldsymbol{x}_k, \ldots, \boldsymbol{x}_t]$. True or ground-truth probability distributions are denoted $p(\cdot)$, while the probability of a random variable $\boldsymbol{X}$ taking a specific value $\boldsymbol{x}$ is written as $p(\boldsymbol{X} = \boldsymbol{x})$ or, in shorthand, as $p(\boldsymbol{x})$. Functions or distributions parameterized by neural networks use subscripts θ for their parameters, e.g., f_θ or $p_\theta(\cdot)$.

2.1. Scalable probabilistic neural surrogates

Scalable neural surrogates. 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. Modern neural surrogates based on deep neural networks rely heavily on the transformer architecture (Vaswani et al., 2017). They have been successfully scaled from a few thousand (Li et al., 2022; Wu et al., 2024; Alkin et al., 2024b) to millions of input dimensions (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). These frameworks have set new state-of-the-art results when modeling PDEs (Sestak et al., 2025; Kohl et al., 2023). Additionally diffusion models seem to possess beneficial properties for modeling high frequencies (Lippe et al., 2023), leading to more accurate and physically plausible solutions.

[1] Code is available at https://github.com/ParticulateFlow/PAINT-Parallel_in_time_Neural_Twins

2.2. Reconstruction of dynamical system states

Bayesian inference. Updating model states by incorporating measurement data is known as Bayesian inference (Särkkä & Svensson, 2023). Bayesian inference methods can be broadly categorized into (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 $n \in \mathbb{N}$. Classical approaches to Bayesian inference take simplifying assumptions and usually arrive at analytical solutions. These approaches date back to the seminal works on Wiener and Kalman filters (Wiener, 1949; Kalman, 1960). However, neither classical approaches nor hybrid approaches using neural networks (Krishnan et al., 2015; Karl et al., 2016; Kim et al., 2025) have been successfully trained on tens of thousands or more input dimensions. They are therefore impractical for our use case.

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. A selection of relevant works is discussed in Appendix A.
Diffusion posterior sampling (Jalal et al., 2021; Chung et al., 2022) has been used (Li et al., 2024; Du et al., 2024; Amorós-Trepat et al., 2026) to reconstruct flow fields from measurements based on a prior diffusion model. However, the quality of the reconstruction often falls short of conditional diffusion models (Gupta et al., 2024). Additional relevant work stems from the domains of weather modeling and data assimilation (Kalnay, 2003; Rozet & Louppe, 2023), which are further elaborated in Appendix A.

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.
A key assumption of this approach is that simulated trajectories faithfully capture the behavior of the true underlying system. To validate this assumption, simulations are typically calibrated and verified against prior knowledge, empirical data, direct measurements, or observable physical effects such as the degree of mixing in liquid systems (American Institute of Aeronautics and Astronautics, 1998).

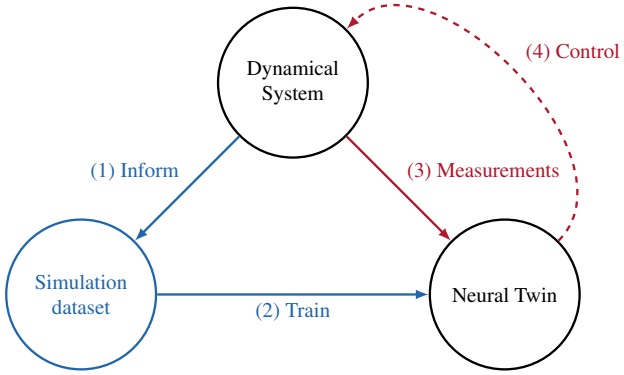


Figure 2. Interaction between a real-world dynamical system, the simulations dataset and a neural twin. The training phase is depicted in blue, the inference (test) phase in red and nodes involved in both phases in black. Dashed lines indicate optional steps. The workflow is structured in three phases: **(1)** The dynamical systems informs the design of the simulations and a diverse training dataset is generated. **(2)** The generated dataset 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 parallel-in-time modeling is *joint modeling* of this distribution. In contrast, AR models factorize this distribution (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, a measurement window of width l and may optionally predict n timesteps into the future:

$$p_\theta(\boldsymbol{x}_{[t-h+1,t+n]} \mid \boldsymbol{m}_{[t-l+1,t]}). \tag{1}$$

In this setup, future measurements can inform past states to obtain temporal consistency, such as in Bayesian filtering. We note, that there is no reliance on the existence or an assumption about an initial condition.

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:

$$p(\boldsymbol{x}_{[0,t]} \mid \boldsymbol{m}_{[0,t]}) = p(\boldsymbol{x}_0 \mid \boldsymbol{m}_{[0,t]}) \cdot \prod_{i=1}^{t} p(\boldsymbol{x}_i \mid \boldsymbol{x}_{i-1}, \boldsymbol{m}_{[0,t]}), \tag{2}$$

where the AR model learns the conditional probability $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$. However, in practice, due to finite compute but potentially arbitrarily long sequences, usually a measurement window $m_{t-l+1,t}$ is used.

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 $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$. To sample from this probability distribution $\boldsymbol{x}_t \sim p_\theta(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]})$ we rewrite the neural network to a deterministic function with a stochastic input η_t:

$$\boldsymbol{x}_t = f_\theta(\boldsymbol{x}_{t-1}, \boldsymbol{m}_{[0,t]}, \eta_t) \qquad \eta_t \sim p(\eta) \tag{3}$$

The Jacobians with respect to the AR state and the measurements are respectively defined as $\boldsymbol{J}_t^x := \partial f_\theta / \partial \boldsymbol{x}_{t-1}$ and $\boldsymbol{J}_t^m := \partial f_\theta / \partial \boldsymbol{m}_{[0,t]}$. In contrast to Orrell (2005), this is a discrete random ordinary differential equation (RODE), i.e., a discrete 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 measurements $\boldsymbol{m}_{[0,t]}$ 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:

$$\boldsymbol{x}_t = F_\theta^t \circ F_\theta^{t-1} \circ \ldots \circ F_\theta^{k+1}(\boldsymbol{x}_k) = F_\theta^{[k+1,t]}(\boldsymbol{x}_k) \tag{4}$$

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

$$F_\theta^{[k+1,t]}(\boldsymbol{x}_k + \epsilon) = F_\theta^{[k+1,t]}(\boldsymbol{x}_k) + \epsilon \prod_{i=k+1}^{t} J_i^x(\boldsymbol{x}_{i-1}) + o(\epsilon^2) \tag{5}$$

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 notion of a model being "on-trajectory". For this, we rely on Assumption 3.1, which states that the predictive relevance of past measurements decays within a finite window l.

Assumption 3.1. Temporal decay of measurement information: For the given dynamical system, there is a finite window size $l \in \mathbb{N}$, 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 a $l \ll t$ such that $\left| p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]}) \right| < \epsilon$.

The validity of assumption 3.1 depends on the dynamical system at hand. It is related to the autocorrelation function decay, which may be known or estimated from data (March-Leuba, 2015). Next, we proceed with our definition of on-trajectory.

Definition 3.2. On-trajectory: Assuming unbounded model size, compute, data a model is *on-trajectory* if the modeled distribution is arbitrarily close to the true distribution. Formally, for all $\epsilon > 0$ and all $t \in \mathbb{N}$, $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ there is model, data and compute such that: $\left| p_\theta(\boldsymbol{X}_t = \boldsymbol{x} \mid \boldsymbol{m}_{[0,t]}) - p(\boldsymbol{X}_t = \boldsymbol{x} \mid \boldsymbol{m}_{[0,t]}) \right| < \epsilon$.

Parallel-in-time models are on-trajectory under Assumption 3.1 . We derive this in Appendix B.1. Intuitively, for each ϵ we can choose the window l large enough to guarantee arbitrary closeness. Additionally, we assume a generative modeling approach capable of approximating arbitrary distributions (Lu & Lu, 2020), such as score-based models (Oko et al., 2023) or normalizing flows (Teshima et al., 2020). In contrast, AR models are in general not on-trajectory. As a constructed counterexample, consider an underlying chaotic system where the model learns to ignore the measurements (for further discussion, see Hess et al. (2023) and Appendix B.2).

3.4. Over-reliance on the autoregressive state

When predicting $\boldsymbol{x}_t$, the model f_θ can draw on two sources of information with different reliability. AR states are prone to drift over time (see Equation 5), whereas each measurement provides a fresh, independent observation of the underlying process.

Over-reliance on autoregressive state. We say that a model *over-relies* on the AR state when it extracts more information from $\boldsymbol{x}_{t-1}$ than is warranted to predict the next state $\boldsymbol{x}_t$. Over-reliance is inherently non-binary and difficult to quantify exactly, since even a drifted state may carry information about the next state. We therefore use the Jacobian norm $\|\boldsymbol{J}_t^x\|$ as a proxy: it measures how sensitively the predicted next state responds to perturbations in $\boldsymbol{x}_{t-1}$. High sensitivity is problematic when the AR state is unreliable. A well-calibrated model should exhibit a smaller $\|\boldsymbol{J}_t^x\|$ when $\boldsymbol{x}_{t-1}$ has drifted far from the true trajectory than when it lies on-trajectory. A model that maintains a uniformly large norm regardless of AR state quality is

more likely to over-rely on the autoregressive state. We note that over-reliance is less problematic in certain settings, e.g. where the dynamical system converges to a global steady state.

Parallel-in-time training as naive over-reliance mitigation. Parallel-in-time models do not factorize the trajectory distribution. Their modeling function is

$$\boldsymbol{x}_t = f_\theta(\boldsymbol{m}_{[0,t]}, \eta_t), \qquad \eta_t \sim p(\eta), \tag{6}$$

so that $\boldsymbol{J}_t^x = \mathbf{0}$ by construction. This eliminates error accumulation through the AR state, at the cost of discarding a potentially informative prior from the previous prediction. We argue that parallel-in-time architectures provide a simple mitigation strategy, which is often much simpler than AR architectures with mitigation strategies discussed next.

Autoregressive architectures which mitigate over-reliance. There exist adaptive strategies to mitigate over-reliance by adjusting the influence of the AR state depending on context. We distinguish two main variants present in contemporary work (Revach et al., 2022; Choi et al., 2022).
Uncertainty-based: The influence of the AR state can be adjusted by explicitly modelling its uncertainty: when the AR state $\boldsymbol{x}_{t-1}$ has high uncertainty (e.g. measured via covariance matrix), the model should rely more heavily on the incoming measurement. Kalman filters implement this principle natively through the Kalman gain, which trades off state and measurement uncertainty (Kalman, 1960; Särkkä & Svensson, 2023). However, accurate uncertainty estimation for high-dimensional states and long trajectories, remains a challenging research problem (Mücke et al., 2024).
Consistency-based: This class of approaches compares the AR state against the incoming measurement (Tandeo et al., 2020): if $\boldsymbol{x}_{t-1}$ is inconsistent with $\boldsymbol{m}_{[0,t]}$, its influence on the prediction should be reduced. This connects to the classical notion of the *innovation*, i.e. the residual between predicted and measured states in filtering (Kailath, 1971; Revach et al., 2022), as well as to variants that instead use the likelihood of the measurements under the model (Chen et al., 2023).

Exposure bias and regularized training. Exposure bias (Bengio et al., 2015) refers to the train-test discrepancy that arises when an AR model is trained on ground-truth data but must condition on its own generated outputs at test time. Schmidt (2019) argues that this is a symptom of insufficient generalization and suggests regularization as a remedy, such as data augmentation via additive noise (Sanchez-Gonzalez et al., 2020), rollout (pushforward) training (Brandstetter et al., 2022), and ℓ_2 regularization (Bretherton et al., 2022). We argue that such techniques can improve generalization, but can not fully eliminate approximation errors, so AR error accumulation persists. For AR models, we therefore view regularization and over-reliance mitigation strategies as two orthogonal, complementary axes of improvement. In contrast, parallel-in-time models have no error accumulation, which avoids exposure bias entirely and eliminates the need for regularization. This results in a simpler training procedure with fewer tunable hyperparameters.

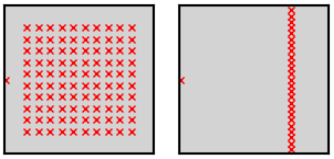
4. Experiments

4.1. 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 128×128 grid. Since the state-space represents 2D velocities, we will write $\boldsymbol{u}_t$ instead of $\boldsymbol{x}_t$ with $\boldsymbol{u}_t \in \mathbb{R}^{d_1 \times d_2 \times 2}$, yielding a state-space of 32,768 dimensions. The train/val/test splits contain 15/1/2 samples respectively, distributed across different Reynolds numbers. Further details about the dataset are provided in Appendix C.2.

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 $\overline{\boldsymbol{u}} := \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{u}_i$ 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 $\overline{\boldsymbol{u}'^2} := \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t}(\boldsymbol{u}_i - \overline{\boldsymbol{u}})^2$. 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 (more details in Appendix C.3).

Probe constellations and emission model. A probe point is a specific spatial location where sensory data such as velocity or pressure are collected. The model is trained using 25 random probe points that are resampled for each data point and every iteration. At train and test time an additional probe point is placed at the inlet. In our experiments we find that without the random sampling of probes, the model rapidly overfits to the training data. At inference, the model can still generalize to a different number and orderly arranged probe



(a) Grid. *(b)* Vertical.

Figure 3. Probe positions.

constellations (see Section 4.3).

For evaluation, we choose two fixed 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). The probes in the grid constellation cover a large space and should lead to a high-fidelity reconstruction. The vertical constellation has probes along a line, which is more challenging. The design of the vertical constellation is inspired by multi-port Pitot rake systems, as employed in Formula 1 aerodynamic testing (Argentini, 2008; F1 Framework, 2012).

Since the noise distribution is unknown, we model the emission process as a noise-free extraction from the ground truth, i.e., $m_t \subset x_t$. Although this simplifies the true observation process, future work may incorporate realistic noise by leveraging known sensor error tolerances. From a methodological standpoint, neural networks have demonstrated robustness to low signal-to-noise ratios (Lacy et al., 2024), suggesting the approach should generalize to more realistic noise conditions.

4.2. Models

Notation for learned distributions. To ensure comparability across models, we define a general formulation:

$$p_\theta\left(\boldsymbol{x}_{[t-h+1,t+n]} \mid \boldsymbol{m}_{[t-l,t]}, \boldsymbol{x}_{[t-k,t-1]}\right), \tag{7}$$

where h, n, l are defined as used in Section 3 and k is the number of autoregressive context steps.

We compactly describe a model configuration by the tuple $(h, n \mid l, k)$. For example $(16, 8 \mid 16, 0)$ denotes a parallel-in-time model that learns the probability $p_\theta\left(\boldsymbol{x}_{[t-15,\,t+8]} \mid \boldsymbol{m}_{[t-15,\,t]}\right)$, while $(1, 0 \mid 16, 1)$ denotes an AR model that learns $p_\theta\left(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-15,\,t]}, \boldsymbol{x}_{t-1}\right)$.

PAINT via Flow Matching. As a score-based generative modeling paradigm we use Flow Matching (FM) (Lipman et al., 2024) for PAINT. The task can be framed as unmasking of non-measurement points under an extreme masking regime, where only 26 out of 16K pixels are observed. The main experiments employ data-coupling Flow Matching (denoted as FM-DC) (Albergo et al., 2023b), but we provide comparisons against classical Flow Matching (denoted as FM) in Appendix C.5. In early experiments, we found that the gradient signal in the vicinity of the probe locations was too weak for the model to reliably incorporate probe information. To address this, we introduce a spatially-weighted Flow Matching loss that assigns higher weight to pixels in the neighborhood of each probe point. As architecture, we adopt a Diffusion Transformer (DiT) (Peebles & Xie, 2023) with small modifications described in Appendix C.4.

Autoregressive Baselines. PAINT differs fundamentally from autoregressive models: rather than regressing on its own previous output one step at a time, it predicts an entire sequence of consecutive timesteps in parallel. This distinction makes direct comparison inherently subtle. Our goal is not to demonstrate that PAINT universally outperforms AR models as we expect relative performance to depend on the dynamical system, the informativeness of the measurements, and the prediction horizon. Instead, the AR baselines serve as reference points that contextualise the results and highlight the strengths and weaknesses of each approach.

We therefore tested two baselines. First, we use the same DiT architecture as for PAINT, but trained autoregressively. However this does not yield stable rollouts for the 900 timesteps we evaluated on (see Appendix C.4 for details). Second, to provide a more informative point of comparison, we include the established AR UNet of Kohl et al. (2023). Following their paradigm, the model is conditioned on two past autoregressive states and on the probe values by concatenating the mask and probe values to the input channels, denoted as $(1, 0 \mid 2, 2)$. On this UNet architecture, we also tested training with a spatially weighted Flow Matching loss, but the model did not produce stable rollouts. Across all our experiments, the model sizes are matched closely, with the DiT comprising 19.8M parameters and the UNet 20.6M. Further architectural details and additional results are provided in Appendices C.4 and C.5.

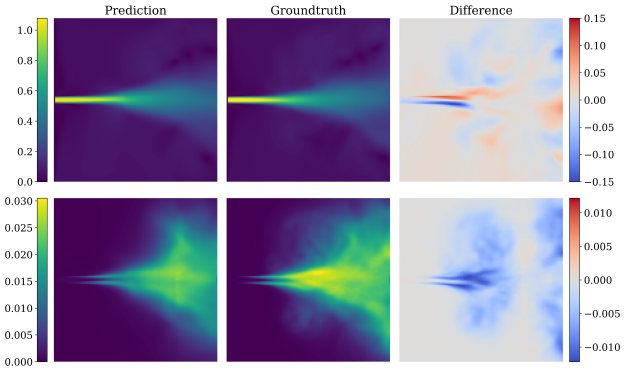


Figure 4. Mean velocity $\overline{u}$ (top row) and variance $\overline{u'^2}$ (bottom row) of the reconstructed $Re = 2100$ test trajectory using PAINT $(16, 8 \mid 16, 0)$ and the *vertical* probe constellation compared with the ground truth. All values are normalized by the mean inlet velocity.

4.3. Results

The quantitative results for the most difficult Reynolds number in the test set ($Re = 2100$) are reported in Table 1. Further results and ablations can be found in Appendix C.5 and Table 7.

Recovery of the physical flow characteristics. Most variants of PAINT show a better physical coherence than the AR UNet from Kohl et al. (2023) across all metrics (see Table 1),

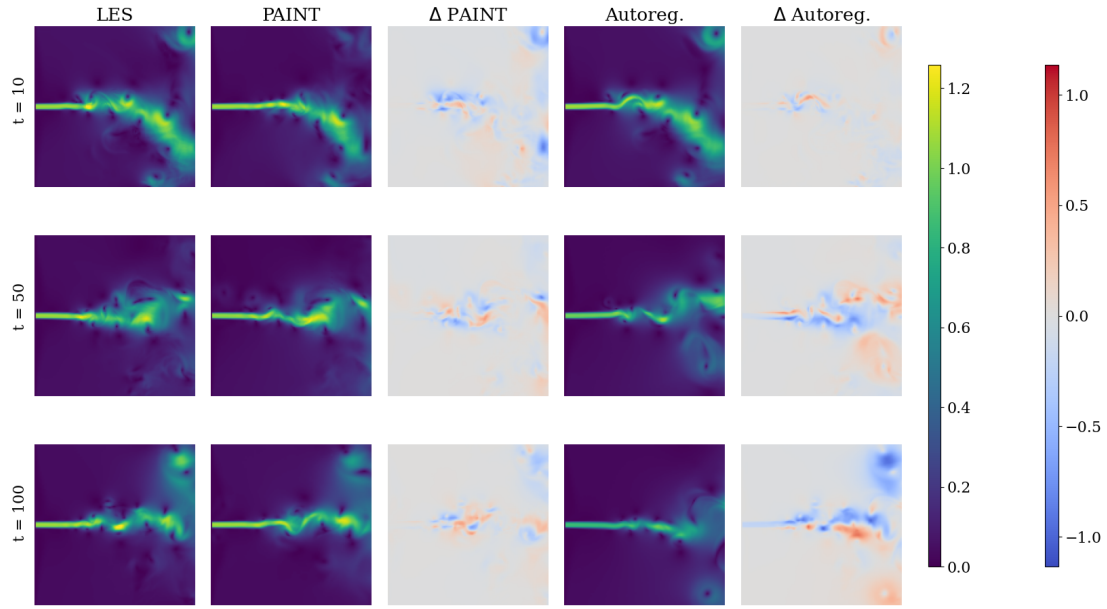


Figure 5. Ground truth vs. predicted states at different timesteps for $Re = 2100$ using the *vertical* probe constellation. The prediction labeled *PAINT* corresponds to PAINT $(16, 8 \mid 16, 0)$. Unlike the autoregressive U-Net $(1, 0 \mid 16, 2)$, PAINT does not accumulate errors over time, whereas the U-Net exhibits a pronounced temporal drift. All values are normalized by the mean inlet velocity.

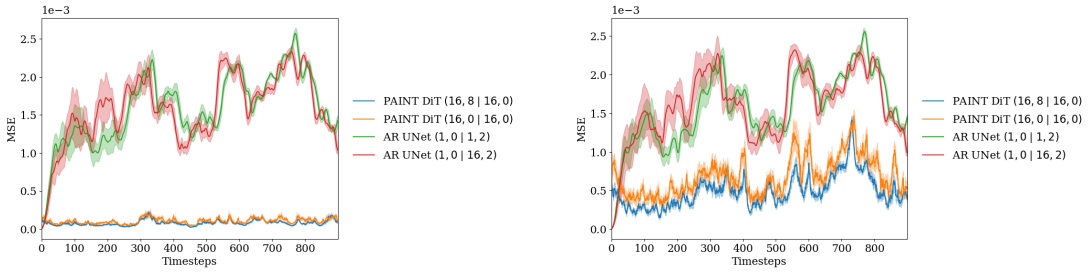
Table 1. Results for physical coherence of DiT and UNet architectures with parallel-in-time and AR modeling on the higher Reynolds number ($Re = 2100$) from the test set. No AR DiT runs led to stable rollouts and are therefore omitted. The full results are available in Table 7 and Appendix C.5. The MAE, MSE and RMSE are time-averaged quantities and measure the physical coherence. $\overline{\mathbf{u}}$ and $\overline{\mathbf{u}'^2}$ denote the mean and variance over time and $E(k)$ is the predicted kinetic energy spectrum. MSE$(u_{[0,t]})$ measures how well the model stays on-trajectory and is the MSE between the predicted and the ground-truth trajectory (see Appendix C.3). All results are reported as mean $\pm$ std across 10 seeds, with each seed evaluated over 900 timesteps of the $Re = 2100$ test trajectory. Diff abbreviates the diffusion training scheme of Kohl et al. (2023) and FM+DC denotes Flow Matching with data-coupling.

Probes	**Approach**	**Arch.**	**Method**	**Distribution**	**MAE**$(\overline{u})\downarrow$	**MAE**$(\overline{u'^2})\downarrow$	**RMSE**$(E(k))\downarrow$	**MSE**$(u_{[0,t]})\downarrow$
					$\times 10^{-3}$	$\times 10^{-5}$	$\times 10^{-5}$	$\times 10^{-7}$
Grid	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	1.2 ± 1.6	9.9 ± 21.7	68.5	8.1 ± 3.6
Grid	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	1.3 ± 2.6	8.9 ± 15.1	62.7	10.8 ± 3.9
Grid	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	2.3 ± 4.0	16.7 ± 29.2	24.9	41.8 ± 12.4
Grid	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	8.2 ± 13.7	93.0 ± 103.2	663.0	155.0 ± 50.1
Vertical	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	2.2 ± 3.1	30.2 ± 38.4	201.1	49.0 ± 21.0
Vertical	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	3.0 ± 3.2	29.9 ± 41.6	204.9	68.2 ± 25.3
Vertical	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	4.3 ± 4.8	29.5 ± 33.1	45.3	130.3 ± 34.7
Vertical	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	7.4 ± 12.5	87.6 ± 96.1	624.3	156.2 ± 50.9

probe constellations and Reynolds numbers in the test set. For PAINT with FM+DC, generally the mean and variance over time MAE$(\overline{u})$ and MAE$(\overline{u'^2})$ improve with more past and future prediction steps h and n, though improvements fall within the standard deviation. This seems to have the reverse effect on the kinetic energy spectrum RMSE$(E(k))$. For Flow Matching without data-coupling, the physical coherence metrics are consistently better for larger h and n (see Table 7). Although trained via randomly sampling probe points, the model generalized to different probe constellations (see Figures 4 and 18). Also, the model's ability to reproduce the correct physical statistics seems to depend directly on the number and position of the sample points (c.f. Figure 4 with Figures 15a, 16a and 17a).

PAINT models maintain stable error, while autoregressive UNet drifts. As measured by MSE$(u_{[0,t]})$ the PAINT variants do not suffer from autoregressive drift and stay closer to the true trajectory than AR models (see Figures 5, 6 and 13). Predicting more past and future steps generally seems to improve this property for PAINT. Notably, Figure 5 depicts a degradation of the AR model's state over time, which has been analyzed in prior work (Kohl et al., 2023).

Over-reliance on autoregressive state. There are several indications for over-reliance on AR state by the UNet $(1, 0 \mid 16, 2)$ The described drift in Figures 5 and 6 show that the



(a) Grid probe constellation. *(b) Vertical* probe constellation.

Figure 6. MSE over time of ground-truth vs. predicted trajectories for the $Re = 2100$ test trajectory and different probe point constellations, comparing the autoregressive baselines with a parallel-in-time DiT. Notably, the autoregressive approach yields nearly identical error profiles for both probe constellations, failing to leverage the additional information provided by the more dense probe placements. The autoregressive model indicates over-reliance on the autoregressive state.

measurement information is not effectively used, indicating that the model overly relies on its AR state.
We also analyzed the gradient norms of the loss with respect to the model input of the AR UNet (see Figure 10). The results indicate that over the course of training the model increases the sensitivity for the AR state, while decreasing it for the measurements. We leave further sensitivity analyses for future work.

Measure of uncertainty. One can obtain a measure for uncertainty by sampling several times and computing the variance (see Figure 14).

Sampling a sequence vs. a single state. PAINT in variantes where $h + n > 1$ for $(h, n \mid \cdot, \cdot)$ may be used for both sampling sequences or individual timesteps. Sampling a connected, temporally smooth sequence, $\boldsymbol{x}_{[t-h+1,t]} \sim p(\boldsymbol{x}_{[t-h+1,t])} \mid \boldsymbol{m}_{[t-l+1,t]})$ aids result interpretation (see Figure 23). Sampling individual timesteps $\boldsymbol{x}_t \sim p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]})$, may be used for real-time state estimation in a sliding window context, albeit without the same temporal smoothness. However, variants with $h + n > 1$ come with a stronger computational burden (see discussion below).

Varying the number of measurement points. Although trained on 26 probes randomly resampled at each iteration, PAINT generalizes to varying numbers and locations of probes at test time (see Figure 12).

4.4. Limitations of PAINT

Discontinuities. A fundamental limitation of parallel-in-time models is the inherently discontinuous temporal nature of the predicted trajectory. This lack of temporal smoothness makes generated trajectories less interpretable and complicates the analysis of temporal properties. However, this limitation is acceptable in scenarios where the primary objective is to obtain an estimate of the system's current or near-future state. In our experiments, we also observe that for reasonably small time steps, the induced discontinuities relative to an autoregressive baseline are modest (see Figure 24 in the Appendix C.5).

Compute at training and inference. Depending on the variant, PAINT may need substantially more training resources, while being slower at inference than the autoregressive UNet. This is summarized in Table 5 and Appendix C.4. While currently the inference time of max. 1.62 seconds may be too slow for practical deployment, this can likely be reduced substantially through techniques such as hardware-level optimizations (Ekelund et al., 2025) or diffusion distillation (Luhman & Luhman, 2021).

5. Conclusion

In this work, we explored parallel-in-time models as a simple and flexible alternative to autoregressive models. We applied these to Neural Twins, a framework that combines neural surrogates with digital twins, that consume measurements at test time. We call the resulting family of methods Parallel-in-time Neural Twins (PAINT) and explore some of its properties, mathematically and experimentally. PAINT employs a generative neural network to model the joint conditional probability of system states, enabling it to provably stay on-trajectory without reliance on initial conditions. One of our key findings is the problem of over-reliance on autoregressive state in standard autoregressive models.
Taken together, our results illuminate the complementary strengths and limitations of autoregressive and parallel-in-time approaches. We hope this analysis sharpens the community's understanding of when each paradigm is most appropriate and spurs further progress in measurement-conditioned neural surrogates.

Limitations and future work A major limitation of parallel-in-time models are the higher computational costs when applied to longer prediction or reconstruction windows. Future work could explore more efficient architectures or alternative training paradigms, such as subsampling

timesteps, to improve scalability. Another limitation is the sliding window approach, which prevents continuous generations. This could potentially be mitigated by stitching techniques (Wei et al., 2019). Future work could also aim to combine parallel-in-time and autoregressive approaches, similar classical literature (Maday & Turinici, 2002).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aich, M., Fürst, A., Sestak, F., Ruiz-Gonzalez, C., Boers, N., and Brandstetter, J. Wind: Weather inverse diffusion for zero-shot atmospheric modeling. *arXiv preprint arXiv:2602.03924*, 2026.

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

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

Albergo, M. S., Goldstein, M., Boffi, N. M., Ranganath, R., and Vanden-Eijnden, E. Stochastic interpolants with data-dependent couplings. *arXiv preprint arXiv:2310.03725*, 2023b.

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers. *CoRR*, 2024a.

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37: 25152–25194, 2024b.

Alkin, B., Kronlachner, T., Papa, S., Pirker, S., Lichtenegger, T., and Brandstetter, J. Neuraldem–real-time simulation of industrial particulate flows. *arXiv preprint arXiv:2411.09678*, 2024c.

Alkin, B., Bleeker, M., Kurle, R., Kronlachner, T., Sonnleitner, R., Dorfer, M., and Brandstetter, J. Ab-upt: Scaling neural cfd surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *arXiv preprint arXiv:2502.09692*, 2025.

American Institute of Aeronautics and Astronautics. *AIAA Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*. American Institute of Aeronautics and Astronautics, 1998.

Amorós-Trepat, M., Medrano-Navarro, L., Liu, Q., Guastoni, L., and Thuerey, N. Guiding diffusion models to reconstruct flow fields from sparse data. *Physics of Fluids*, 38(1), 2026.

Anderson, J. L. An ensemble adjustment kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

Argentini, G. Pitot tube from euler equations point of view. application to formula 1 cars. *arXiv preprint arXiv:0809.3194*, 2008.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Bretherton, C. S., Henn, B., Kwa, A., Brenowitz, N. D., Watt-Meyer, O., McGibbon, J., Perkins, W. A., Clark, S. K., and Harris, L. Correcting coarse-grid weather and climate models by machine learning from global storm-resolving simulations. *Journal of Advances in Modeling Earth Systems*, 14(2):e2021MS002794, 2022.

Chakraborty, D., Guan, H., Stock, J., Arcomano, T., Cervone, G., and Maulik, R. Multimodal atmospheric super-resolution with deep generative models. *arXiv preprint arXiv:2506.22780*, 2025.

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

Chen, K., Bai, L., Ling, F., Ye, P., Chen, T., Luo, J.-J., Chen, H., Xiao, Y., Chen, K., Han, T., et al. Towards an end-to-end artificial intelligence driven global weather forecasting system. *arXiv preprint arXiv:2312.12462*, 2023.

Choi, G., Park, J., Shlezinger, N., Eldar, Y. C., and Lee, N. Split-kalmannet: A robust model-based deep learning approach for slam. *arXiv preprint arXiv:2210.09636*, 2022.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Courtier, P., Thépaut, J.-N., and Hollingsworth, A. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387, 1994.

Cuéllar, A., Güemes, A., Ianiro, A., Flores, Ó., Vinuesa, R., and Discetti, S. Three-dimensional generative adversarial networks for turbulent flow estimation from wall measurements. *Journal of Fluid Mechanics*, 991:A1, 2024.

Dang, H. V., Choi, J. B., and Nguyen, P. C. Flronet: Deep operator learning for high-fidelity fluid flow field reconstruction from sparse sensor measurements. *arXiv preprint arXiv:2412.08009*, 2024.

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

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

Dobrev, V. A., Kolev, T., Petersson, N. A., and Schroder, J. B. Two-level convergence theory for multigrid reduction in time (mgrit). *SIAM Journal on Scientific Computing*, 39 (5):S501–S527, 2017.

Du, P., Parikh, M. H., Fan, X., Liu, X.-Y., and Wang, J.-X. Confild: Conditional neural field latent diffusion model generating spatiotemporal turbulence. *arXiv preprint arXiv:2403.05940*, 2024.

Ekelund, J., Markidis, S., and Peng, I. Boosting performance of iterative applications on gpus: Kernel batching with cuda graphs. In *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 70–77. IEEE, 2025.

Emmett, M. and Minion, M. Toward an efficient parallel in time method for partial differential equations. *Communications in Applied Mathematics and Computational Science*, 7(1):105–132, 2012.

F1 Framework. Aero rake in F1, RB8. https://f1framework.blogspot.com/2012/02/aero-rake-in-f1-rb8.html, February 2012. Accessed: 2026-04-21.

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

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

Fukami, K., Fukagata, K., and Taira, K. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

Fukami, K., Fukagata, K., and Taira, K. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909:A9, 2021.

Gao, R., Hoogeboom, E., Heek, J., Bortoli, V. D., Murphy, K. P., and Salimans, T. Diffusion meets flow matching: Two sides of the same coin. 2024. URL https://diffusionflow.github.io/.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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

Güemes, A., Vila, C. S., and Discetti, S. Super-resolution gans of randomly-seeded fields. *arXiv preprint arXiv:2202.11701*, 2022.

Gupta, S., Jalal, A., Parulekar, A., Price, E., and Xun, Z. Diffusion posterior sampling is computationally intractable. *arXiv preprint arXiv:2402.12727*, 2024.

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

Hess, F., Monfared, Z., Brenner, M., and Durstewitz, D. Generalized teacher forcing for learning chaotic dynamics. *arXiv preprint arXiv:2306.04406*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hodyss, D. and Morzfeld, M. Using diffusion models to do data assimilation. *Monthly Weather Review*, 154(2): 165–182, 2026.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

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

Jalal, A., Arvinte, M., Daras, G., Price, E., Dimakis, A. G., and Tamir, J. Robust compressed sensing mri with deep generative priors. *Advances in neural information processing systems*, 34:14938–14954, 2021.

Johansen, A. A tutorial on particle filtering and smoothing: Fifteen years later. 2009.

Kailath, T. Some extensions of the innovations theorem. *The Bell System Technical Journal*, 50(4):1487–1494, 1971.

Kalman, R. E. A new approach to linear filtering and prediction problems. 1960.

Kalnay, E. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

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

Kim, H., Kim, J., Won, S., and Lee, C. Unsupervised deep learning for super-resolution reconstruction of turbulence. *Journal of Fluid Mechanics*, 910:A29, 2021.

Kim, S., Petrunin, I., and Shin, H.-S. 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.

Kohl, G., Chen, L.-W., and Thuerey, N. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv preprint arXiv:2309.01745*, 2023.

Krishnan, R. G., Shalit, U., and Sontag, D. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Lacy, F., Ruiz-Reyes, A., and Brescia, A. Machine learning for low signal-to-noise ratio detection. *Pattern Recognition Letters*, 179:115–122, 2024.

Lagemann, C., Mokbel, S., Gondrum, M., Rüttgers, M., Callaham, J., Paehler, L., Ahnert, S., Zolman, N., Lagemann, K., Adams, N., et al. Hydrogym: A reinforcement learning platform for fluid dynamics. *arXiv preprint arXiv:2512.17534*, 2025.

Lee, Y., Park, J., and Lee, C.-O. Parareal neural networks emulating a parallel-in-time algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5): 6353–6364, 2022.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

Li, Z., Han, W., Zhang, Y., Fu, Q., Li, J., Qin, L., Dong, R., Sun, H., Deng, Y., and Yang, L. Learning spatiotemporal dynamics with a pretrained generative model. *Nature Machine Intelligence*, 6(12):1566–1579, 2024.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T. Q., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Loshchilov, I., Hutter, F., et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5): 5, 2017.

Lu, Y. and Lu, J. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33: 3094–3105, 2020.

Luhman, E. and Luhman, T. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Maday, Y. and Turinici, G. A parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathematique*, 335(4):387–392, 2002.

March-Leuba, J. A. Autocorrelation function statistics and implication to decay ratio estimation. Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), 2015.

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

Meng, X., Li, Z., Zhang, D., and Karniadakis, G. E. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370:113250, 2020.

Mücke, N. T., Bohté, S. M., and Oosterlee, C. W. The deep latent space particle filter for real-time data assimilation with uncertainty quantification. *Scientific Reports*, 14(1): 19447, 2024.

Oko, K., Akiyama, S., and Suzuki, T. Diffusion models are minimax optimal distribution estimators. In *International Conference on Machine Learning*, pp. 26517–26582. PMLR, 2023.

Oommen, V., Khodakarami, S., Bora, A., Wang, Z., and Karniadakis, G. E. Learning turbulent flows with generative models: Super-resolution, forecasting, and sparse flow reconstruction. *arXiv preprint arXiv:2509.08752*, 2025.

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

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

Ozan, D. E., Nóvoa, A., and Magri, L. Data-assimilated model-based reinforcement learning for partially observed chaotic flows. In *International Conference on Computational Science*, pp. 65–72. Springer, 2025.

Pandya, D., Vachharajani, B., and Srivastava, R. A review of data assimilation techniques: Applications in engineering and agriculture. *Materials Today: Proceedings*, 62:7048–7052, 2022.

Parrish, D. F. and Derber, J. C. The national meteorological center's spectral statistical-interpolation analysis system. *Monthly Weather Review*, 120(8):1747–1763, 1992.

Parthipan, R., Anand, M., Christensen, H. M., Hosking, J. S., and Wischik, D. J. Defining error accumulation in ml atmospheric simulators. *arXiv preprint arXiv:2405.14714*, 2024.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

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

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., Van Sloun, R. J., and Eldar, Y. C. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70: 1532–1547, 2022.

Rozet, F. and Louppe, G. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36: 40521–40541, 2023.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Särkkä, S. and Svensson, L. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Schmidt, F. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 157–167, 2019.

Sestak, F., Toshev, A., Fürst, A., Klambauer, G., Mayr, A., and Brandstetter, J. Lam-slide: Latent space modeling of spatial dynamical systems via linked entities. *arXiv preprint arXiv:2502.12128*, 2025.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Tandeo, P., Ailliot, P., Bocquet, M., Carrassi, A., Miyoshi, T., Pulido, M., and Zhen, Y. A review of innovation-based methods to jointly estimate model and observation error covariance matrices in ensemble data assimilation. *Monthly Weather Review*, 148(10):3973–3994, 2020.

Teshima, T., Ishikawa, I., Tojo, K., Oono, K., Ikeda, M., and Sugiyama, M. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33:3362–3373, 2020.

Toscano, J. D., Käufer, T., Wang, Z., Maxey, M., Cierpka, C., and Karniadakis, G. E. 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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

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

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

A. Further related work

Reconstruction of flow fields 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. Earlier research focused on training models for super-resolution, i.e., recovering a high-fidelity output from a degraded input at a single point in time (Fukami et al., 2019; 2021; Chakraborty et al., 2025). Some works used generative modeling techniques such as GANs (Goodfellow et al., 2014), Diffusion or Flow Matching 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). 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.

Parallel-in-time methods. Classical parallel-in-time (PinT) methods are a family of numerical algorithms that exploit parallelism across the time dimension to accelerate the solution of time-dependent differential equations, complementing classical spatial parallelism. The parareal algorithm (Maday & Turinici, 2002) is among the most widely studied of these methods. It operates by decomposing the time domain into subintervals, using a cheap coarse solver to propagate an initial guess forward in time, and then refining it with accurate fine solvers running in parallel. The Parallel Full Approximation Scheme in Space and Time (PFASST) algorithm (Emmett & Minion, 2012) extends this idea by combining temporal parallelism with multigrid strategies across both space and time. Another foundational approach is multigrid reduction in time (MGRIT) (Dobrev et al., 2017), which applies multigrid principles directly to the time direction. These methods have also served as inspiration for neural network based surrogates (Meng et al., 2020; Lee et al., 2022).

Data assimilation. Data assimilation (DA) originated in numerical weather prediction (Kalnay, 2003) and has since been extended to diverse domains, including agriculture and dynamical systems (Pandya et al., 2022). Classical approaches include variational methods, such as 3D-Var and 4D-Var (Parrish & Derber, 1992; Courtier et al., 1994), particle-based filters (Johansen, 2009) or ensemble methods (Anderson, 2001).
With the recent rise of diffusion models, score-based data assimilation techniques usually leverage a form of diffusion posterior sampling (DPS) (Chung et al., 2022) to guide the diffusion process. This has been successfully applied to neural network models for weather forecasting and computational fluid dynamics (Rozet & Louppe, 2023; Hodyss & Morzfeld, 2026; Aich et al., 2026).

Control of dynamical systems. Instead of reconstructing the dynamical system from measurements, prior work controlled the dynamical system directly from the measurements using reinforcement learning (Ozan et al., 2025; Lagemann et al., 2025). While these are promising directions, the lack of interpretability is a major obstacle for a broader adoption and a limitation regarding applicability.

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 3.1. We will show that for an appropriate window size l, for all $t \in \mathbb{N}$ and any $\epsilon_1 > 0$:

$$\left|p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})\right| < \epsilon_1. \tag{8}$$

First, from Assumption 3.1 choose a window size l such that:

$$\left|p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[0,t]})\right| < \epsilon_2 < \frac{\epsilon_1}{2}. \tag{9}$$

Further we pick a generative modeling approach capable of approximating arbitrary distributions (Lu & Lu, 2020), such as score-based models (Oko et al., 2023) or normalizing flows (Teshima et al., 2020). Similar to the Universal Function Approximator Theorem (Hornik, 1991), we choose an approximation of the true distribution as follows:

$$\left|p_\theta(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]}) - p(\boldsymbol{x}_t \mid \boldsymbol{m}_{[t-l+1,t]})\right| < \epsilon_3 < \frac{\epsilon_1}{2}. \tag{10}$$

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. Rather than aiming for full mathematical rigor, our goal is to illustrate a key failure mode of AR models: their problem-dependent inability to stay on-trajectory. This issue is especially pronounced in chaotic dynamical systems, where small deviations compound rapidly over time, although it tends to be less severe for fixed-point dynamics. For clarity of exposition, we construct the counter-example in the setting where the model ignores the provided measurements. In practice, we also observe that in chaotic systems the velocities at points far from the probes typically drift off very quickly, which would provide an alternative counter-example; however, formalizing this case would require additional assumptions and obscure the core argument.
Being on-trajectory is defined as: for all $t \in \mathbb{N}$ and all $\epsilon > 0$:

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

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. Experimental details

C.1. Computational Fluid Dynamics

The dynamical system we will use in our experiments (see Section 4.1) is a 2D fluid dynamical system in the turbulent regime. Therefore, we give a brief introduction to computational fluid dynamics.

Incompressible Navier-Stokes. The fluid mechanics problem under the continuum assumption is governed by the Navier–Stokes equations in their incompressible form as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \mu \Delta \boldsymbol{u} + \boldsymbol{f_e}, \quad \nabla \cdot \boldsymbol{u} = 0 \tag{15}$$

where $\boldsymbol{u}$ denotes the velocity, p denotes the pressure, μ denotes the viscosity and $\boldsymbol{f_e}$ denotes any kind of external body forces. These equations are complemented by appropriate boundary and initial conditions. This constitutes a system of equations of highly complex, second-order nonlinear PDEs with no known general analytical solution. Therefore, they are typically solved numerically.

Precision–computation tradeoff. Numerical solutions of the Navier–Stokes equations involve a tradeoff between accuracy and computational cost. Direct numerical simulation (DNS) resolves the full flow field but is prohibitively expensive for most

practical flows. Large-eddy simulation (LES) resolves only the largest, energy-containing scales, while Reynolds-averaged Navier–Stokes (RANS) approaches provide only mean flow fields under strong modeling assumptions (Davidson, 2015; Fröhlich, 2006). LES represents a compromise between fidelity and computational cost within this spectrum of methods and was therefore used for the experiments in this paper.

C.2. 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 7, 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

$$Re = \langle u \rangle_{\text{inlet}} h_{\text{inlet}} \nu^{-1}. \tag{16}$$

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.

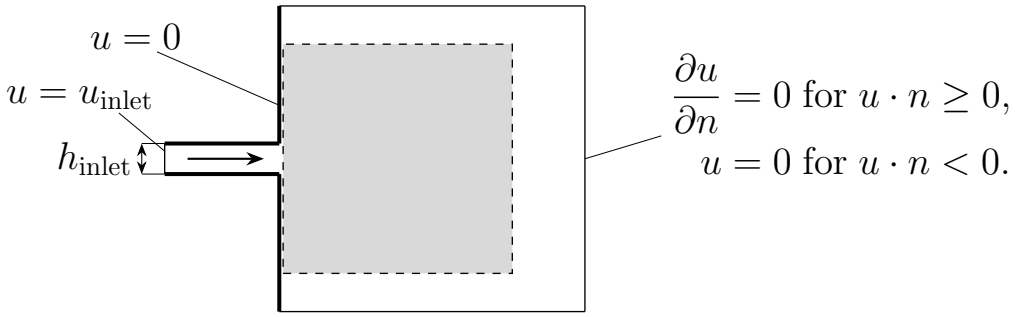


Figure 7. Computational domain of the CFD simulation, featuring a thin channel with a prescribed inlet velocity profile, transitioning into an expanded outflow region zone. The channel and outflow orifice walls enforce a no-slip boundary condition, while the top, bottom and right boundaries of the outflow 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		

C.3. Metrics

For the following we denote ground truth velocities with $\tilde{\boldsymbol{u}}$ and predicted velocities with $\boldsymbol{u}$. For computing the metrics, both these quantities are collected over a trajectory. Therefore these are tensors of shape $T \times H \times W \times D$, where T are the number of timesteps within a trajectory, H and W are the height and width of the domain and D is the dimensionality of an velocity vector, where $D = 2$ for our experiments.

As described in Section 4, $\boldsymbol{u}$ can be decomposed into a time-averaged mean component and a turbulent fluctuation component:

$$\boldsymbol{u}_{[0,t]} = \bar{\boldsymbol{u}} + \boldsymbol{u}'_{[0,t]}, \qquad \bar{\boldsymbol{u}} := \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{u}_i. \tag{17}$$

The variance over time is denoted as

$$\overline{\boldsymbol{u}'^2} := \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{u}_i'^2. \tag{18}$$

Both $\bar{\boldsymbol{u}}$ and $\boldsymbol{u}'$ are time-averaged quantities and therefore tensors of shape $H \times W \times D$.

Measures for physical coherence. We provide three measures for physical coherence.

MAE($\overline{\boldsymbol{u}}$): The mean absolute error of the time-averaged velocity field is then given by

$$\mathrm{MAE}(\bar{\boldsymbol{u}}) := \frac{1}{H\,W\,D} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{d=1}^{D} \left| \bar{\tilde{\boldsymbol{u}}}_{h,w,d} - \bar{\boldsymbol{u}}_{h,w,d} \right|. \tag{19}$$

Note that the time dimension T is already collapsed by the averaging in $\bar{\boldsymbol{u}}$, so the summation only runs over the spatial and velocity-component axes. This metric therefore evaluates the quality of the predicted mean flow, independent of errors in the turbulent fluctuations.

MAE($\boldsymbol{u}'$): Analogously, the mean absolute error of the time-averaged velocity variance is defined as

$$\mathrm{MAE}(\overline{\boldsymbol{u}'^2}) := \frac{1}{H\,W\,D} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{d=1}^{D} \left| \overline{\tilde{\boldsymbol{u}}'^2}_{h,w,d} - \overline{\boldsymbol{u}'^2}_{h,w,d} \right|. \tag{20}$$

As with $\mathrm{MAE}(\bar{\boldsymbol{u}})$, the time dimension T is already collapsed by the averaging in $\overline{\boldsymbol{u}'^2}$, so the summation runs only over the spatial and velocity-component axes. While $\mathrm{MAE}(\bar{\boldsymbol{u}})$ measures how well the predicted mean flow matches the ground truth, $\mathrm{MAE}(\overline{\boldsymbol{u}'^2})$ instead quantifies how accurately the model reproduces the second-order statistics of the turbulent fluctuations.

Mean kinetic energy:

For a 2D velocity field (u, v) defined on a grid of size $H \times W$, the mean kinetic energy spectrum $\bar{E}(k)$ is computed as follows: For each time step t, a fast Fourier transform is computed for the horizontal (u) and vertical (v) velocity components independently. From these, the spectral kinetic energy density in the Fourier domain is calculated as:

$$E_{2D}(k_x, k_y, t) = \frac{1}{2}\left(|FFT\{u(x, y, t)\}|^2 + |FFT\{v(x, y, t)\}|^2\right) \tag{21}$$

To obtain the one-dimensional spectrum $E(k)$, the 2D density is radially integrated (binned) by summing all energy components falling within a shell of magnitude k. Finally, the mean kinetic energy spectrum is obtained by averaging over the total number of time steps T in the rollout:

$$\bar{E}(k) = \frac{1}{T}\sum_{t=1}^{T} E(k, t) \tag{22}$$

The discrepancy between the predicted and ground-truth mean spectra is then quantified using the Root Mean Square Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_k}\sum_{k=0}^{N_k-1}\left(\bar{E}_{\text{pred}}(k) - \bar{E}_{\text{true}}(k)\right)^2} \tag{23}$$

Measure for a model being on-trajectory. In contrast to the previous metrics, $\mathrm{MSE}(\boldsymbol{u}_{[0,t]})$ evaluates the pointwise squared error at every timestep of the trajectory:

$$\mathrm{MSE}(\boldsymbol{u}_{[0,t]}) := \frac{1}{T\,H\,W\,D}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{d=1}^{D}\left(\tilde{\boldsymbol{u}}_{t,h,w,d} - \boldsymbol{u}_{t,h,w,d}\right)^2. \tag{24}$$

This metric measures the instantaneous deviation between predicted and ground-truth velocity fields at each timestep rather than the error of a time-averaged statistic. It is therefore a measure for how well the reconstructed state is on-trajectory.

C.4. Implementation and architecture

Architecture and hyperparameters. The DiT and the UNet were both implemented in Pytorch (Paszke et al., 2017). For the DiT we trained in float16 mixed precision. The most important training parameters are provided in Table 4. Our code also uses FlashAttention (Dao et al., 2022; Dao, 2024). For both models we use 20 denoising steps for generating samples.

Diffusion Transformer. We use a standard Diffusion Transformer (Peebles & Xie, 2023) with a few modifications. We use convolutional layers for patching and unpatching, and alternate spatial and temporal attention blocks to attend over the time dimension. For probe conditioning, we insert a Cross-Attention layer after the Self-Attention and before the MLP within each attention block.

Autoregressive UNet. For the UNet we followed (Kohl et al., 2023) and took the implementation of the public Github repository [2]. The UNet uses convolutional, linear attention (Shen et al., 2021) and classical softmax attention blocks. 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. We conditioned on a 16-timestep probe history, to ensure a fair comparison with DiT runs.

Optimization hyperparameters. We employed well-established techniques and hyperparameters for optimization, including AdamW (Loshchilov et al., 2017), cosine annealing of the learning rate, etc. These choices were grounded in their proven effectiveness and widespread adoption in the literature. Although we acknowledge that tuning these hyperparameters could potentially yield marginal performance improvements, we do not anticipate that such modifications would substantially

[2] https://github.com/tum-pbs/autoreg-pde-diffusion/blob/main/acdm-demo.ipynb

impact the primary conclusions and insights drawn from our experiments. Given the computational constraints and the dimensionality of the hyperparameter search space, we chose to use the established settings for the sake of experimental tractability. Future work may explore the potential benefits of hyperparameter tuning in this context to validate this assumption.

Spatially Weighted Flow Matching Loss. To promote accurate reconstruction in the vicinity of regions of interest, we employ a spatially weighted flow matching loss. Let $\mathcal{M} \in \{0,1\}^{H\times W}$ be a binary mask indicating the pixel locations of the probes. We first compute a soft spatial weight map by convolving the mask with a Gaussian kernel $\mathcal{G}_{\sigma,k}$ of size k and standard deviation σ:

$$W_{\text{mask}} = \mathcal{G}_{\sigma,k} * \mathcal{M}, \tag{25}$$

which assigns large weights to unmasked pixels near the mask boundary, with values decaying smoothly with distance at a rate governed by σ and k. The final per-pixel weight map is defined as a convex combination of a uniform base weight and the boundary weight map:

$$W = (1-\alpha)\,\frac{W_{\text{bias}}}{\sum_{h,w} W_{\text{bias}}} + \alpha\,\frac{W_{\text{mask}}}{\sum_{h,w} W_{\text{mask}}}, \tag{26}$$

where $W_{\text{bias}} = \mathbf{1}$ is a uniform map that ensures all pixels contribute to the loss, and both terms are ℓ^1-normalised so that their magnitudes are commensurate. The scalar $\alpha \in [0,1]$ controls the relative emphasis placed on the boundary region. In all experiments we use a Gaussian kernel of size $k = 3$ with $\sigma = 4$ and set $\alpha = 0.2$.

We adopt the *optimal transport* (OT) conditional vector field (Lipman et al., 2022), which constructs straighter probability paths between the source and target distributions. Under this formulation, the interpolant is defined as $x_t = (1-t)\,x_0 + t\,x_1$, where $x_0 \sim \mathcal{N}(0, I)$ is the source noise and x_1 is the clean target image. The OT conditional vector field is then:

$$u_t^{\text{OT}}(x_t \mid x_1) = x_1 - x_0, \tag{27}$$

which is constant in t and points directly from the noise sample to the clean image. This yields straighter flow trajectories and more efficient transport compared with diffusion-based vector fields. We incorporate the spatial weight map into this objective, yielding the spatially weighted OT flow matching loss:

$$\mathcal{L}_{\text{FM}} = \sum_{h,w} W_{h,w} \cdot \left\| v_\theta(x_t, t)_{h,w} - (x_1 - x_0)_{h,w} \right\|^2, \tag{28}$$

where $v_\theta(x_t, t)$ is the velocity predicted by the network at noised input $x_t = (1-t)\,x_0 + t\,x_1$ and timestep $t \sim \mathcal{U}[0,1]$, and $x_1 - x_0$ is the optimal transport target velocity conditioned on the clean image x_1.

Data-Coupled Spatially Weighted Flow Matching Loss. We additionally define a data-coupled variant of the spatially weighted flow matching loss. The construction mirrors the procedure above, with a single modification to the weight map: after convolving the binary mask with the Gaussian kernel to obtain W_{mask}, we zero out the weight map at the mask locations themselves:

$$W_{\text{mask}} = W_{\text{mask}} \odot (1 - \mathcal{M}), \tag{29}$$

where $\odot$ denotes element-wise multiplication. This ensures that the probe pixels receive no direct gradient signal from the flow matching objective, reflecting the fact that these locations are unchanged by the inpainting procedure. The surrounding boundary region still receives an elevated gradient signal through the spatially decayed weights, encouraging the model to produce coherent transitions at the probe boundaries. The resulting data-coupled spatially weighted OT flow matching loss is:

$$\mathcal{L}_{\text{FM}}^{\text{dc}} = \sum_{h,w} W_{h,w} \cdot \left\| v_\theta(x_t, t)_{h,w} - (x_1 - x_0)_{h,w} \right\|^2. \tag{30}$$

Selection of batch size and timesteps per batch. As mentioned in Section 4, a direct comparison between PAINT and the autoregressive baseline is not entirely feasible, given PAINT's prediction of a full sequence of states. Therefore, we carefully evaluated and chose the *batch size* and the *timesteps per batch* according to the following procedure

For the autoregressive UNet, we opted for the maximum batch size that could be accommodated by a single GPU, which amounted to 144. We also tested a batch size of 288 on 2 GPUs via data-parallel training. However, as there was no significant change in training nor validation loss, we settled on a batch size of 144 for all experiments.

In the case of PAINT $(16, 8 \mid 16, 0)$, we maintained the same batch size of 144. For PAINT $(16, 0 \mid 16, 0)$, we tested two

variants: one with a batch size of 144, and another with an increased batch size of $144 \times 1.5 = 216$. The latter was chosen to align the number of predicted timesteps with the 16/8 setting. Given the negligible difference observed between the two variants (see Table 3), we proceeded with a batch size of 144 for the remainder of the experiments.

Table 3. Results on different batch sizes. With further scaling of the batch size, the results do not improve further. We therefore conclude that the batch size of 144 is large enough.

Re	**Probes**	**Arch.**	**Distribution**	**BZ**	**MAE**$(\overline{u})\downarrow$	**MAE**$(\overline{u'^2})\downarrow$	**RMSE**$(E(k))\downarrow$	**MSE**$(u_{[0,t]})\downarrow$
					$\times 10^{-3}$	$\times 10^{-5}$	$\times 10^{-5}$	$\times 10^{-7}$
1100	Grid	DiT	$(16, 0 \mid 16, 0)$	144	1.2 ± 2.8	6.2 ± 10.7	46.3	4.6 ± 1.6
1100	Grid	DiT	$(16, 0 \mid 16, 0)$	216	1.0 ± 2.2	4.9 ± 8.2	33.8	3.8 ± 1.3
1100	Vertical	DiT	$(16, 0 \mid 16, 0)$	144	3.0 ± 5.8	8.2 ± 15.8	99.0	24.2 ± 6.5
1100	Vertical	DiT	$(16, 0 \mid 16, 0)$	216	3.4 ± 6.2	10.2 ± 17.5	117.3	27.2 ± 6.9
2100	Grid	DiT	$(16, 0 \mid 16, 0)$	144	1.3 ± 2.6	8.9 ± 15.1	62.7	10.8 ± 3.9
2100	Grid	DiT	$(16, 0 \mid 16, 0)$	216	1.5 ± 3.2	9.1 ± 15.5	69.1	11.5 ± 4.9
2100	Vertical	DiT	$(16, 0 \mid 16, 0)$	144	3.0 ± 3.2	29.9 ± 41.6	204.9	68.2 ± 25.3
2100	Vertical	DiT	$(16, 0 \mid 16, 0)$	216	2.9 ± 2.9	25.7 ± 34.6	189.2	70.6 ± 26.6

Table 4. Experimental details for DIT and the UNet.

		DiT	UNet
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

Compute requirements. PAINT can be instantiated with $(h, n \mid l, k)$ for $p_\theta\big(\boldsymbol{x}_{[t-h+1,\, t+n]} \mid \boldsymbol{m}_{[t-l,\, t]}, \boldsymbol{x}_{[t-k,\, t-1]}\big)$ The computational costs depend on the architecture at hand. For the DiT the measurement condition length l is relatively cheap, as the measurements are sparse and are fed in via Cross-Attention (Vaswani et al., 2017). The AR state length k is somewhat more expensive, as it is concatenated to the input of the Flow Matching model. The most expensive settings use large h or n, as these define the output timestep dimensions.

PAINT $(16, 8 \mid 16, 0)$ is the most expensive during training, as it needs to reconstruct a video in parallel. We had access to a cluster environment and trained PAINT $(16, 8 \mid 16, 0)$ on 3 nodes each consisting of 4 NVIDIA Ampere A100 GPUs with 64GB VRAM. Furthermore the PAINT $(16, 0 \mid 16, 0)$ variants where trained on 2 nodes each consisting of 4 NVIDIA Ampere A100 GPUs with 64GB VRAM. Training took around 30 hours for 100K iterations for both the configurations.

For the smaller PAINT variants with $h + n = 1$ are cheaper. We used either a single H100 with 80GB or a single A100 with 60GB and gradient checkpointing.

The autoregressive baseline is comparably cheap. It was trained on either a single Nvidia RTX-PRO6000 Blackwell 96GB or a single Nvidia H100-SXM-80GB for ca. 24 hours. The inference times are reported in the main paper.

Inference times. The inference times across different models and distributions are given in Table 5. All times were measured on an H100 with batch size 1.

Table 5. Inference times for different models and learned distributions.

Model	**Distribution**	**Time**	**GPUs**	**Memory / GPU**	**Training Time**
DiT	(24,0|16,0)	1.62 sec	12x A100	60 GB	30 hours
DiT	(16,0|16,0)	1.02 sec	8x A100	60 GB	30 hours
DiT	(1,0|16,0)	0.36 sec	1x H100	80 GB	24 hours
UNet	(1.0|16,1)	0.86 sec	1x H100	80 GB	24 hours

Table 6. Architectural details of the diffusion transformer (DiT).

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

C.5. Additional results and ablations

We show additional results and ablations in Table 7. Further evaluations are shown in Figures 21, 22, 19 and 20.

Autoregressive Diffusion Transformer. We tested the Diffusion Transformer (Peebles & Xie, 2023) architecture used for PAINT as an autoregressive model. We tested this with and without datacoupled Flow Matching in the variants $(1, 0|16, 1)$ and $(1, 0|1, 1)$. In the datacoupled (Albergo et al., 2023b) runs the ground truth velocities at the probes are directly input to the prediction. For the autoregressive model with datacoupling, this leads to the measurement values being different to the neighboring pixels, which makes them visible (see top row in Figure 9). We concluded that data-coupling is inherently not suited for AR models that may drift.
Therefore, we also tried a Flow Matching run without data-coupling. While the FM based run (see Figure 8) does not have such a failure case and initial generations appear to be of high quality, the model becomes unstable over the course of 900 timesteps.

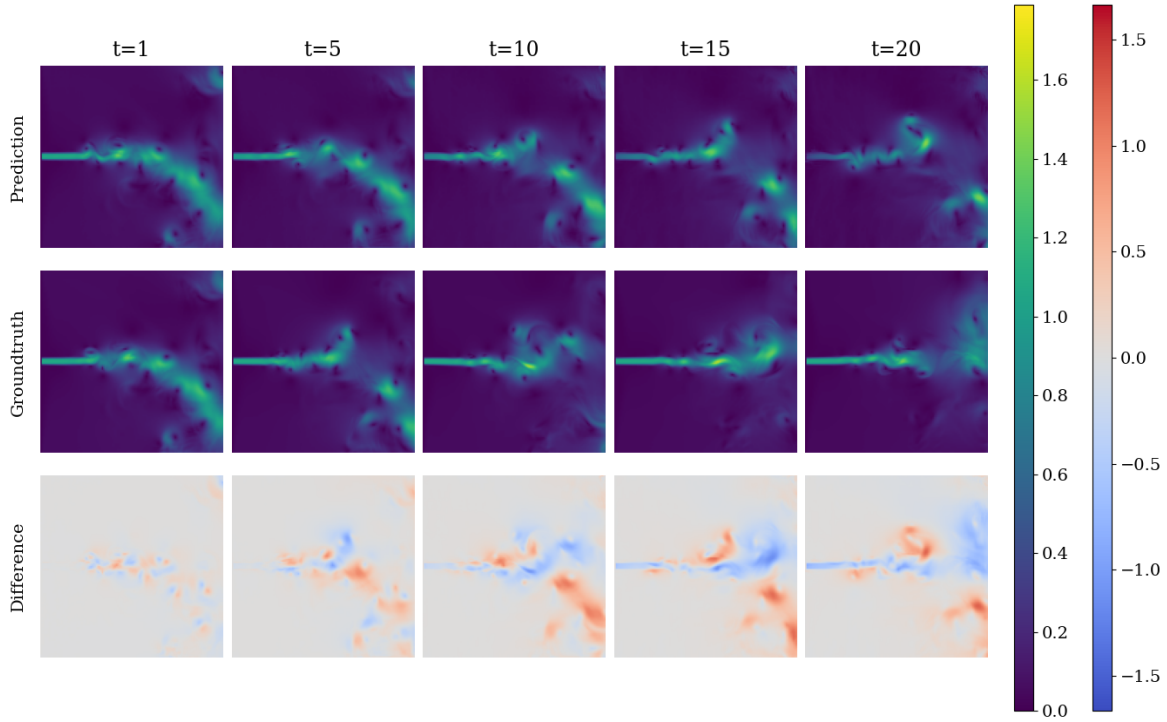


Figure 8. Rollout comparison of an AR DiT with classical Flow Matching in the $(1, 0 \mid 16, 1)$ setting.

Stability through future predictions. Comparing the smoothing-and-filtering variant, PAINT $(16, 0 \mid 16, 0)$ with the

Table 7. Results for physical coherence of PAINT DiT models and the autoregressive UNet. We also trained DiT in the settings $(1, 0 \mid 16, 0)$ and $(1, 0 \mid 1, 0)$ however without leading to stable rollouts. 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.

Re	**Probes**	**Approach**	**Arch.**	**Method**	**Distribution**	**MAE**$(\overline{u})\downarrow$	**MAE**$(\overline{u'^2})\downarrow$	**RMSE**$(E(k))\downarrow$	**MSE**$(u_{[0,t]})\downarrow$
						$\times 10^{-3}$	$\times 10^{-5}$	$\times 10^{-5}$	$\times 10^{-7}$
1100	Grid	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	0.9 ± 1.3	2.2 ± 5.5	19.8	2.0 ± 1.0
1100	Grid	PAINT	DiT	FM	$(16, 0 \mid 16, 0)$	1.0 ± 1.5	3.6 ± 6.9	4.2	7.7 ± 2.8
1100	Grid	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	1.2 ± 2.8	6.2 ± 10.7	46.3	4.6 ± 1.6
1100	Grid	PAINT	DiT	FM	$(1, 0 \mid 16, 0)$	1.8 ± 2.3	4.4 ± 8.5	6.9	13.1 ± 3.6
1100	Grid	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	1.0 ± 1.4	3.6 ± 6.1	4.5	8.2 ± 2.9
1100	Grid	PAINT	DiT	FM	$(1, 0 \mid 1, 0)$	1.6 ± 2.3	2.9 ± 6.0	6.3	8.0 ± 3.1
1100	Grid	PAINT	DiT	FM+DC	$(1, 0 \mid 1, 0)$	1.5 ± 1.7	4.2 ± 8.3	5.5	10.6 ± 3.2
1100	Grid	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	3.7 ± 6.3	16.0 ± 24.1	122.8	29.9 ± 8.8
1100	Vertical	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	1.1 ± 1.6	6.3 ± 11.1	36.8	8.9 ± 3.4
1100	Vertical	PAINT	DiT	FM	$(16, 0 \mid 16, 0)$	1.6 ± 2.4	5.7 ± 10.1	6.6	18.6 ± 5.1
1100	Vertical	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	3.0 ± 5.8	8.2 ± 15.8	99.0	24.2 ± 6.5
1100	Vertical	PAINT	DiT	FM	$(1, 0 \mid 16, 0)$	1.8 ± 2.5	5.5 ± 9.8	7.4	23.1 ± 6.6
1100	Vertical	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	1.9 ± 2.3	4.7 ± 8.7	7.6	22.6 ± 6.3
1100	Vertical	PAINT	DiT	FM	$(1, 0 \mid 1, 0)$	2.1 ± 2.7	4.7 ± 8.4	7.5	23.5 ± 6.6
1100	Vertical	PAINT	DiT	FM+DC	$(1, 0 \mid 1, 0)$	1.5 ± 1.8	4.7 ± 8.5	7.0	21.3 ± 6.0
1100	Vertical	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	3.7 ± 6.1	15.9 ± 23.9	122.6	29.5 ± 8.8
2100	Grid	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	1.2 ± 1.6	9.9 ± 21.7	68.5	8.1 ± 3.6
2100	Grid	PAINT	DiT	FM	$(16, 0 \mid 16, 0)$	2.1 ± 3.3	26.5 ± 36.2	27.4	48.5 ± 17.6
2100	Grid	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	1.3 ± 2.6	8.9 ± 15.1	62.7	10.8 ± 3.9
2100	Grid	PAINT	DiT	FM	$(1, 0 \mid 16, 0)$	3.4 ± 4.6	32.2 ± 43.2	36.1	75.8 ± 18.7
2100	Grid	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	2.3 ± 4.0	16.7 ± 29.2	24.9	41.8 ± 12.4
2100	Grid	PAINT	DiT	FM	$(1, 0 \mid 1, 0)$	2.6 ± 4.5	22.6 ± 34.6	30.0	34.1 ± 10.9
2100	Grid	PAINT	DiT	FM+DC	$(1, 0 \mid 1, 0)$	3.0 ± 4.5	33.2 ± 42.6	34.4	58.3 ± 15.5
2100	Grid	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	8.2 ± 13.7	93.0 ± 103.2	663.0	155.0 ± 50.1
2100	Vertical	PAINT	DiT	FM+DC	$(16, 8 \mid 16, 0)$	2.2 ± 3.1	30.2 ± 38.4	201.1	49.0 ± 21.0
2100	Vertical	PAINT	DiT	FM	$(16, 0 \mid 16, 0)$	3.5 ± 4.5	33.7 ± 44.6	35.9	101.7 ± 29.7
2100	Vertical	PAINT	DiT	FM+DC	$(16, 0 \mid 16, 0)$	3.0 ± 3.2	29.9 ± 41.6	204.9	68.2 ± 25.3
2100	Vertical	PAINT	DiT	FM	$(1, 0 \mid 16, 0)$	4.1 ± 4.4	32.9 ± 43.7	40.2	126.0 ± 33.9
2100	Vertical	PAINT	DiT	FM+DC	$(1, 0 \mid 16, 0)$	4.3 ± 4.8	29.5 ± 33.1	45.3	130.3 ± 34.7
2100	Vertical	PAINT	DiT	FM	$(1, 0 \mid 1, 0)$	4.0 ± 4.2	30.0 ± 36.1	41.3	119.9 ± 32.5
2100	Vertical	PAINT	DiT	FM+DC	$(1, 0 \mid 1, 0)$	3.8 ± 4.8	36.3 ± 48.2	39.2	111.3 ± 31.2
2100	Vertical	AR	UNet	Diff	$(1, 0 \mid 16, 2)$	7.4 ± 12.5	87.6 ± 96.1	624.3	156.2 ± 50.9

smoothing-filtering-and-prediction variant, $(16, 8 \mid 16, 0)$ highlights the stabilizing effect of incorporating future-state predictions. While the $(16, 0 \mid 16, 0)$ approach relies exclusively on reconstructing the given measurement series, the inclusion of predictive task seems to enable $(16, 8 \mid 16, 0)$ to produce more stable reconstructions. PAINT with $(16, 0 \mid 16, 0)$ performs in general worse than $(16, 8 \mid 16, 0)$, though most metrics lie within the standard deviation.

Jacobian as measure for over-reliance. In Figure 10 we show that over the course of teacher-forced Diffusion model training $\|\boldsymbol{J}_t^x\|$ increases, while $\|\boldsymbol{J}_t^m\|$ decreases (see also Section 3.4). We think that analyses like these at training and inference time can shed more light on autoregressive model dynamics, failure modes and how to mitigate exposure bias (Schmidt, 2019). We leave a more comprehensive investigation of this technique to future work.

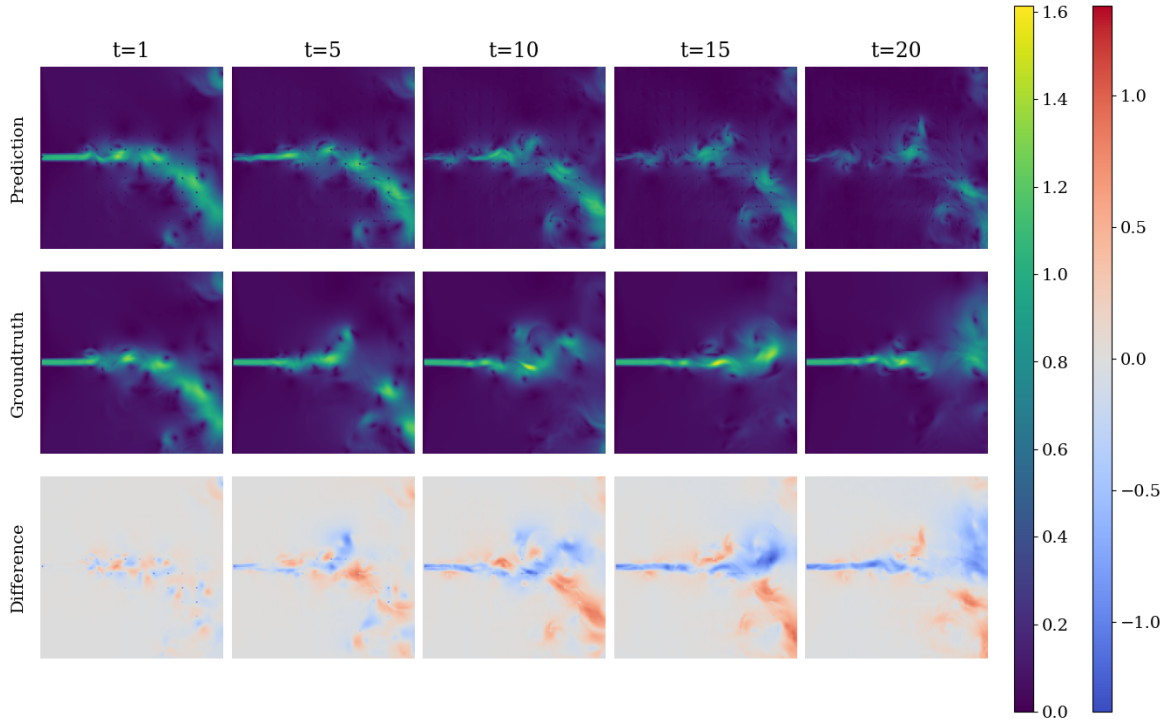


Figure 9. Rollout comparison of an AR DiT with data-coupled Flow Matching in the $(1, 0 \mid 16, 1)$ setting.

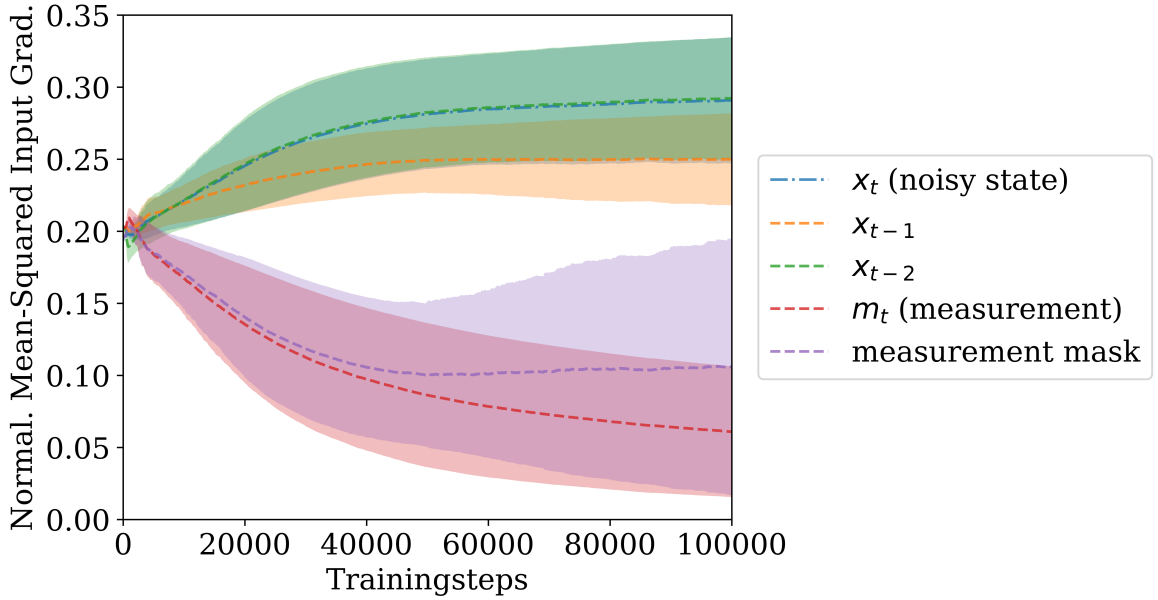


Figure 10. Relative input sensitivity of the loss with respect to different input channel groups, measured using mean squared gradient norms with respect to the model input. Gradients are aggregated over spatial and batch dimensions and grouped by input type, as the conditioning for the autoregressive model $(1, 0 \mid 2, 2)$ is provided via the channel dimension. The gradients are normalized to obtain relative contributions for each conditioning signal, and the running mean (dashed line) and standard deviation (shaded area) of these quantities are visualized. Over the course of the training, sensitivity with respect to the measurement inputs decreases, while sensitivity to the autoregressive states increases. This could suggest, that the model increasingly relies on the more informative autoregressive context relative to the measurements.

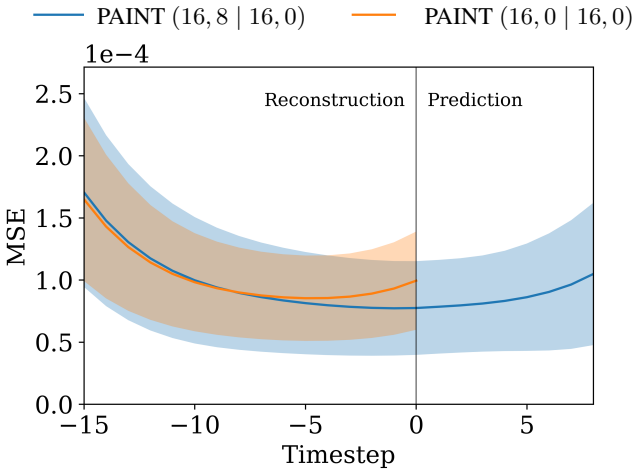


Figure 11. The solid lines indicate the mean MSE over a reconstructed sequence, with the shaded regions indicating $\pm$ one standard deviation across samples. While the average MSE for future timesteps remains relatively low, PAINT $(16, 8 \mid 16, 0)$ surprisingly exhibits the highest error for timesteps in the past. This data stems from an evaluation using Reynolds number 2100 and the *grid* probe point constellation.

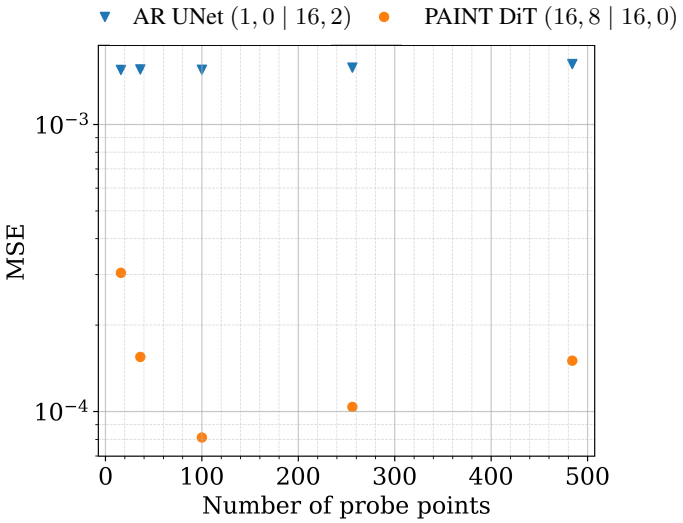
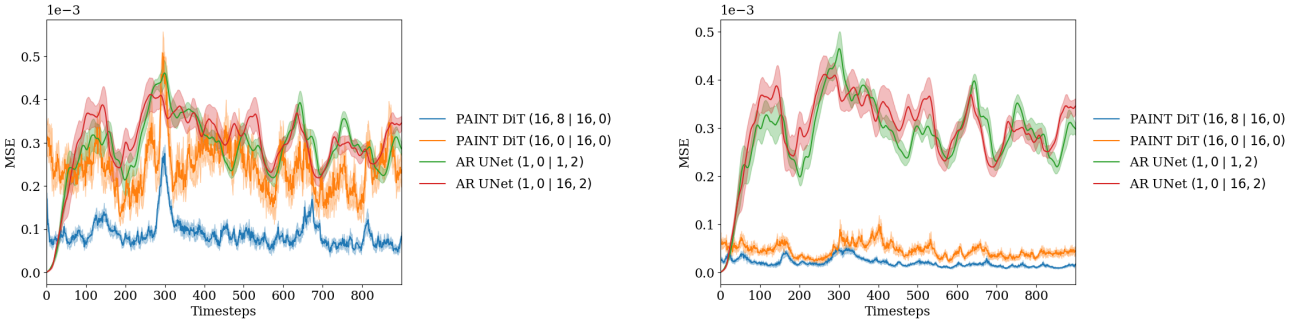


Figure 12. MSE over number of probe points provided during inference. Probe points are arranged in a pattern comparable to the *grid* probe constellation. For PAINT $(16, 8 \mid 16, 0)$, the MSE decreases as the number of probe points increases, reaching an optimum near 100. Beyond this point, the MSE increases again, as PAINT was only trained with 25 probe points. In contrast, the performance of the autoregressive baseline shows little sensitivity to the number of probe points provided.



(a) Vertical probe constellation.

(b) Grid probe constellation.

Figure 13. MSE over time of ground-truth vs. predicted trajectories for the $Re = 1100$ test trajectory and different probe point constellations, comparing the autoregressive baselines with PAINT $(16, 8 \mid 16, 0)$ and $(16, 0 \mid 16, 0)$. The autoregressive model indicates over-reliance on the autoregressive state.

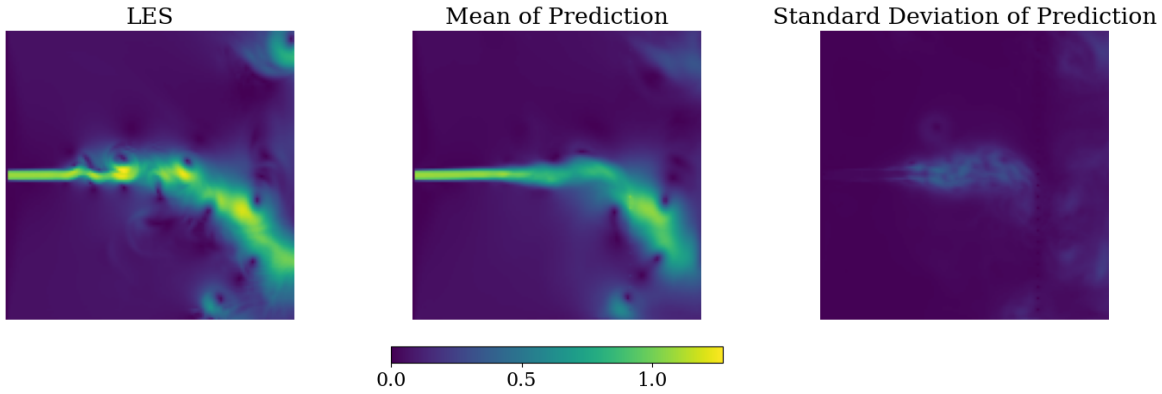
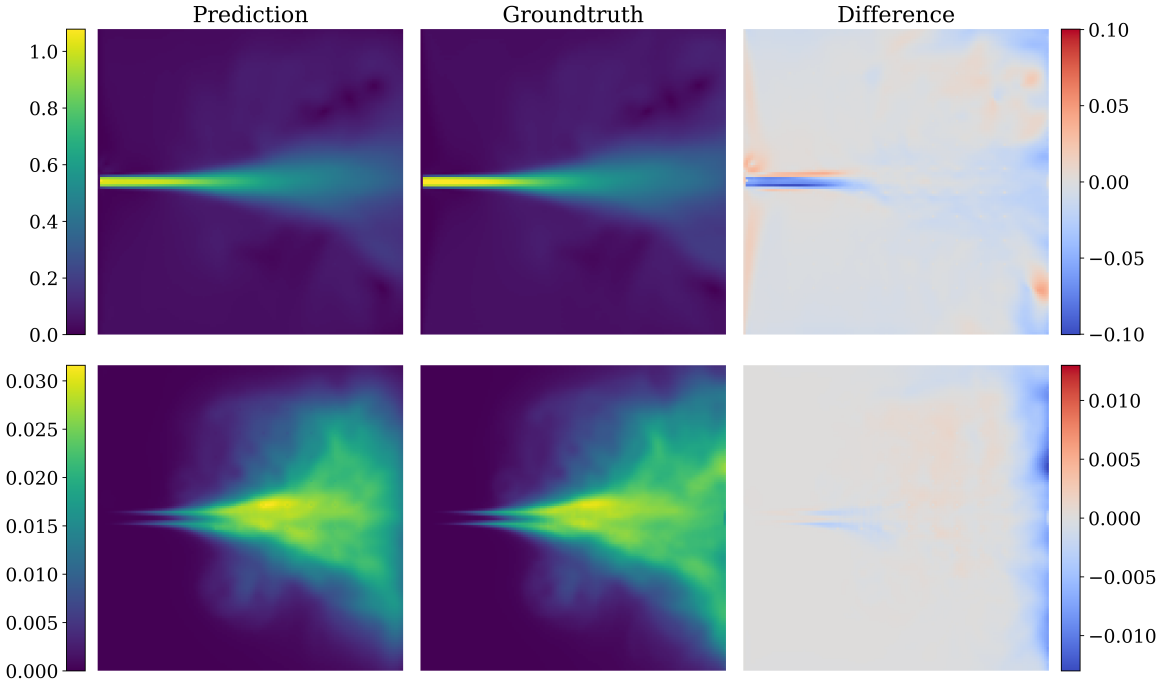
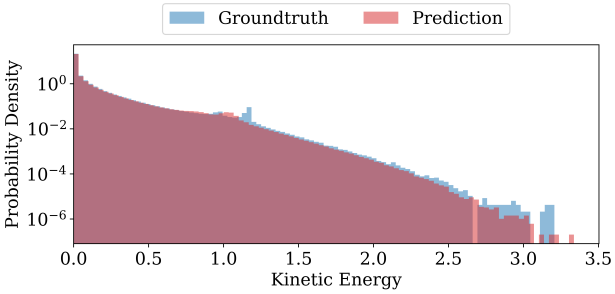


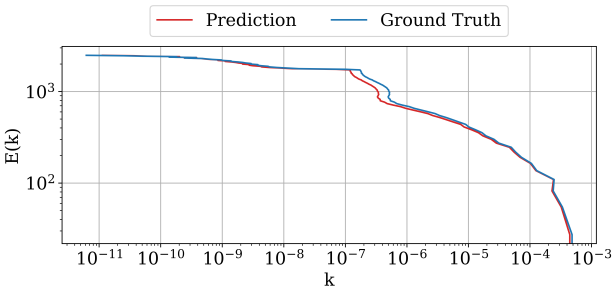
Figure 14. Left: Exemplary groundtruth velocity snapshot, for Reynolds number 2100 and using the *vertical* probe point constellation. Center: Mean of the predictions using PAINT $(16, 8 \mid 16, 0)$ over 10 independent seeds. Right: Standard deviation of the predictions using PAINT $(16, 8 \mid 16, 0)$ 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.

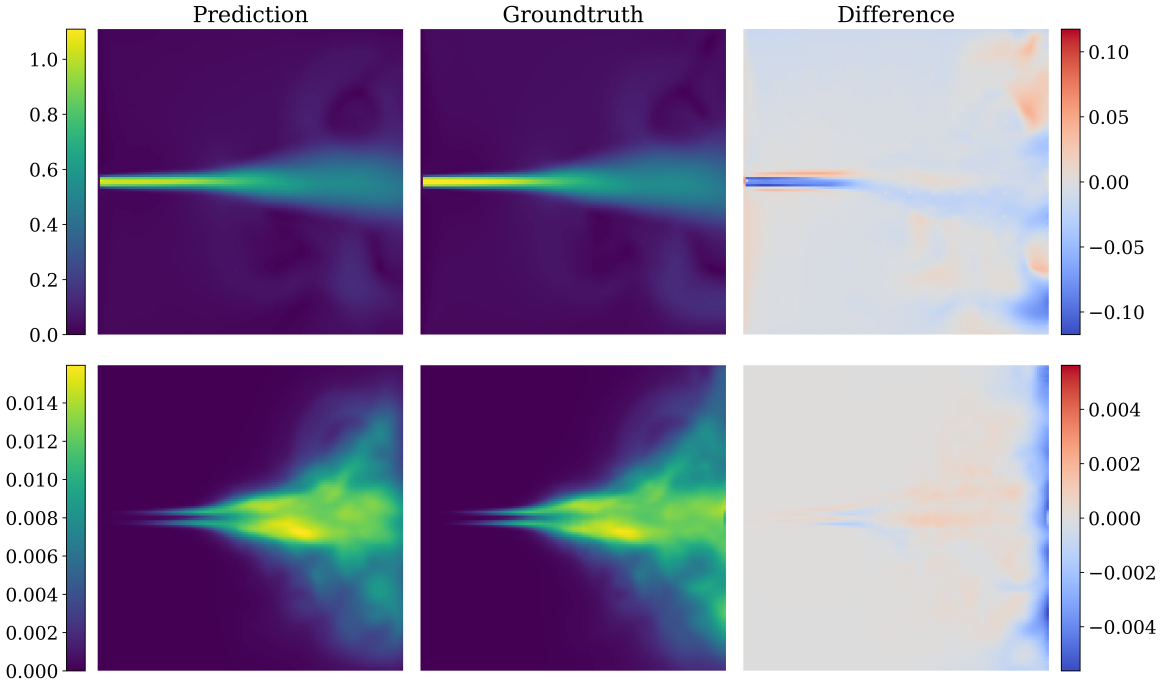


(b) Histogram of time-averaged kinetic energy.

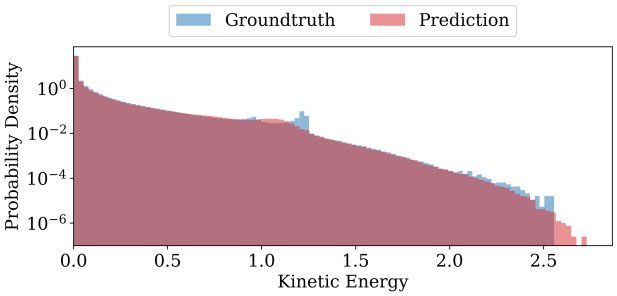


(c) Time-averaged kinetic energy spectrum.

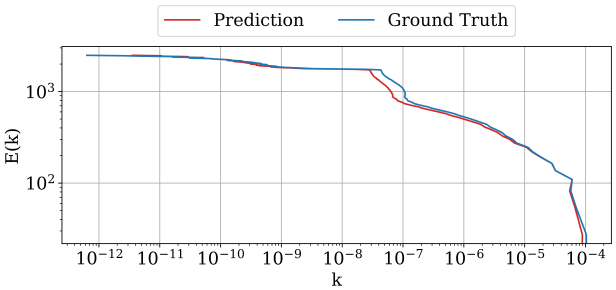
Figure 15. Statistical analysis of reconstructed flow fields using PAINT DiT $(16, 8 \mid 16, 0)$ for Reynolds number 2100 and a *grid* probe point constellation.



(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.

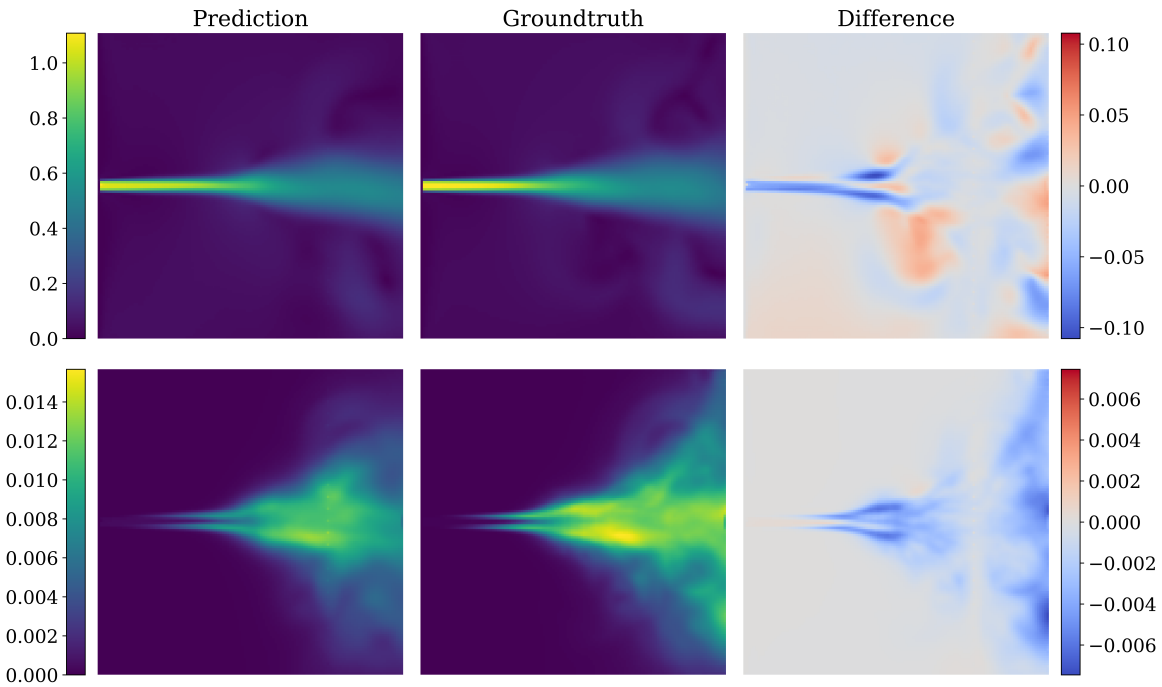


(b) Histogram of time-averaged kinetic energy.

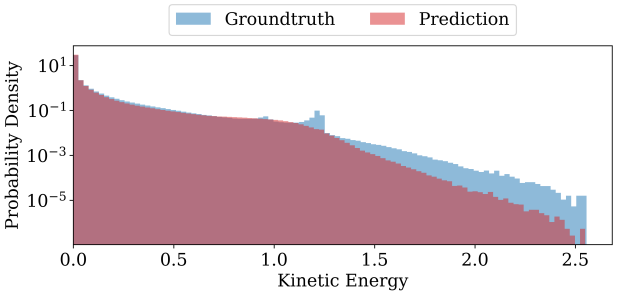


(c) Time-averaged kinetic energy spectrum.

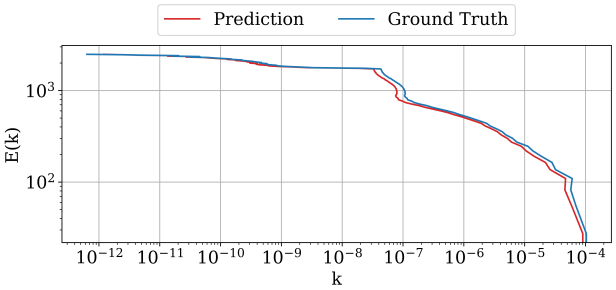
Figure 16. Statistical analysis of reconstructed flow fields using PAINT $(16, 8 \mid 16, 0)$ for Reynolds number 1100 and a *grid* probe point constellation.



(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.

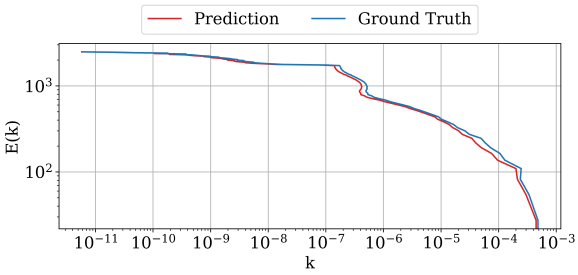


(b) Histogram of time-averaged kinetic energy.

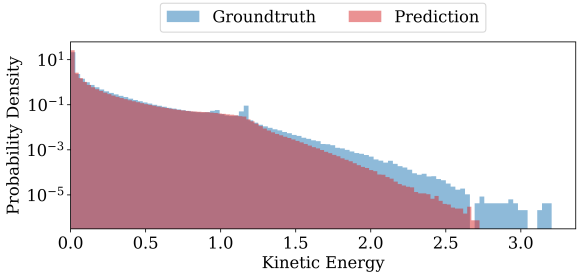


(c) Time-averaged kinetic energy spectrum.

Figure 17. Statistical analysis of reconstructed flow fields using PAINT $(16, 8 \mid 16, 0)$ for Reynolds number 1100 and a *vertical* probe point constellation.

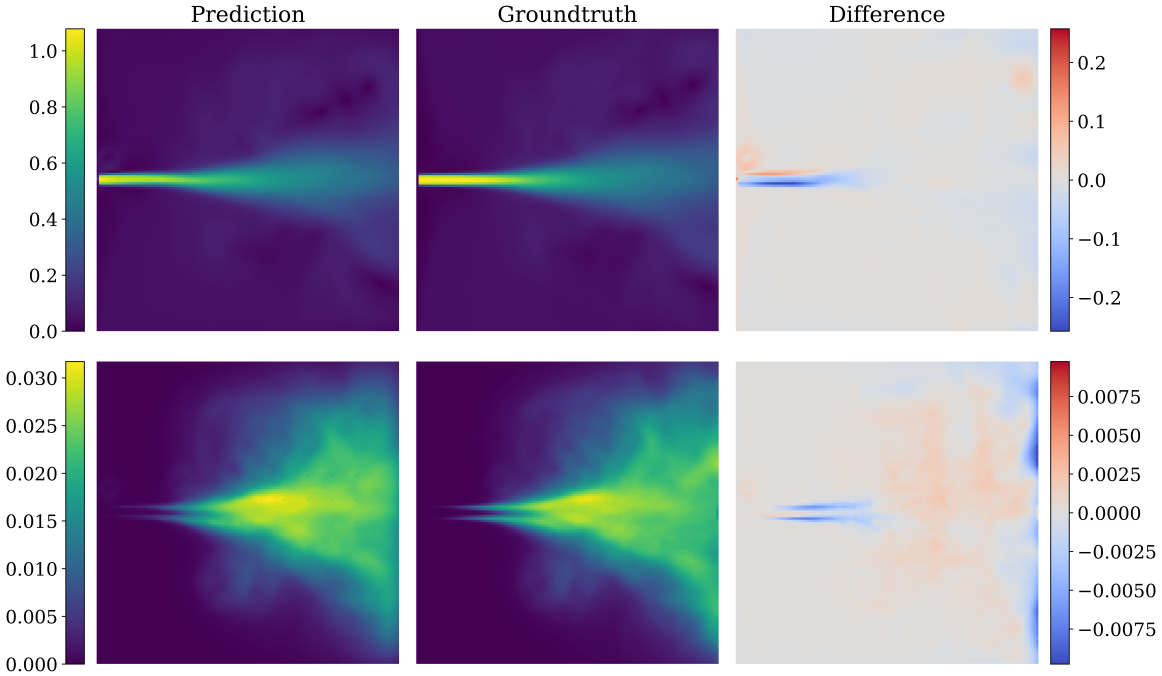


(a) Time-averaged kinetic energy spectrum.

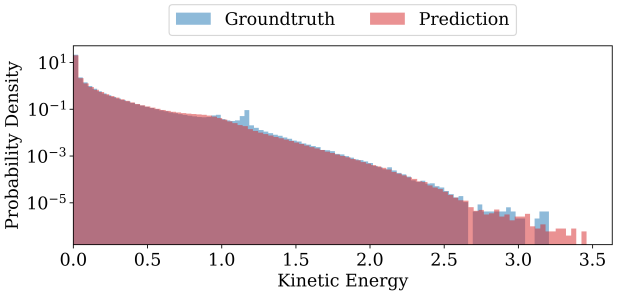


(b) Histogram of the kinetic energy.

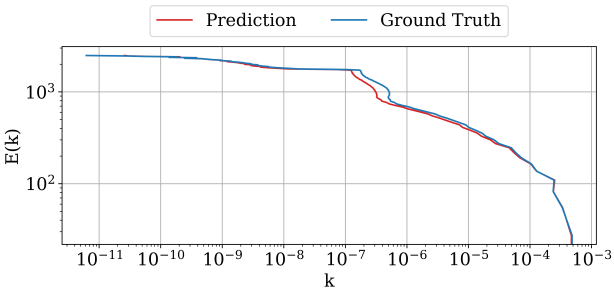
Figure 18. Statistical analysis of reconstructed flow fields using PAINT $(16, 8 \mid 16, 0)$ for Reynolds number 2100 and a *vertical* probe point constellation.



(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.

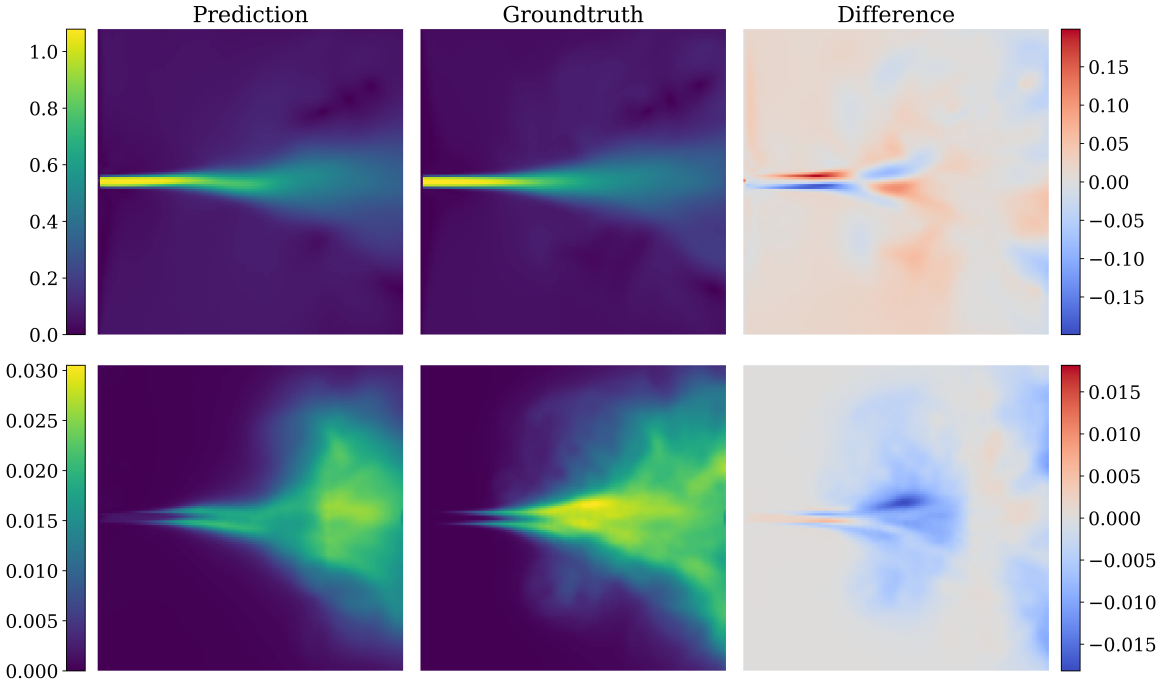


(b) Histogram of time-averaged kinetic energy.

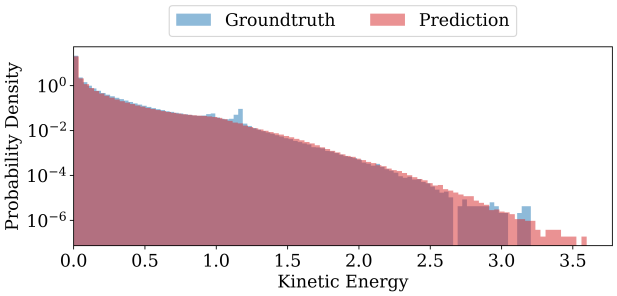


(c) Time-averaged kinetic energy spectrum.

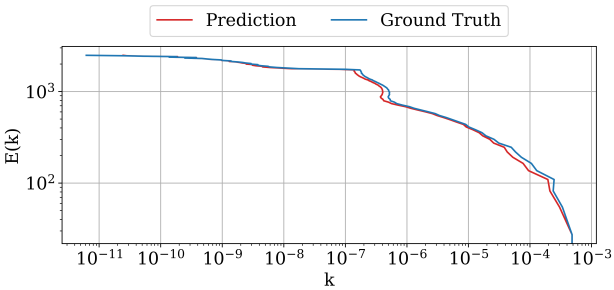
Figure 19. Statistical analysis of reconstructed flow fields using PAINT $(16, 0 \mid 16, 0)$ for Reynolds number 2100 and a *grid* probe point constellation.



(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.

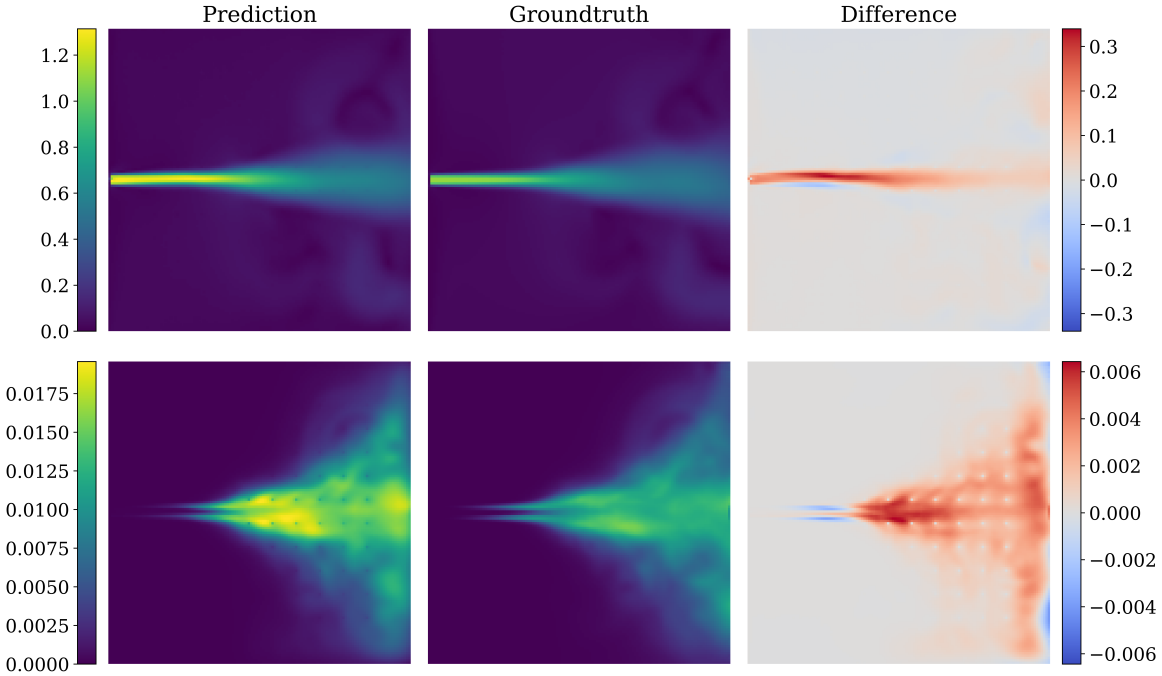


(b) Histogram of time-averaged kinetic energy.

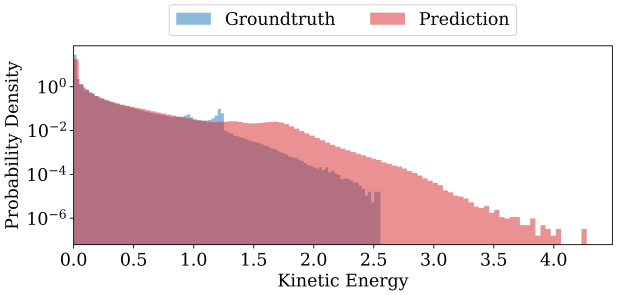


(c) Time-averaged kinetic energy spectrum.

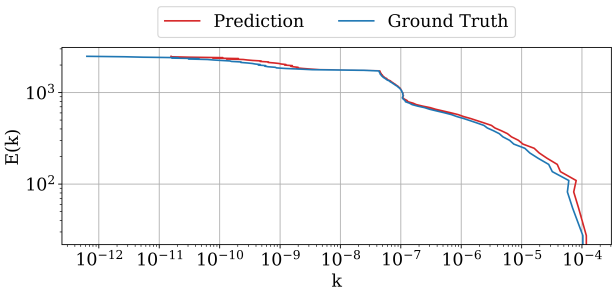
Figure 20. Statistical analysis of reconstructed flow fields using PAINT $(16, 0 \mid 16, 0)$ for Reynolds number 2100 and a *vertical* probe point constellation.



(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.

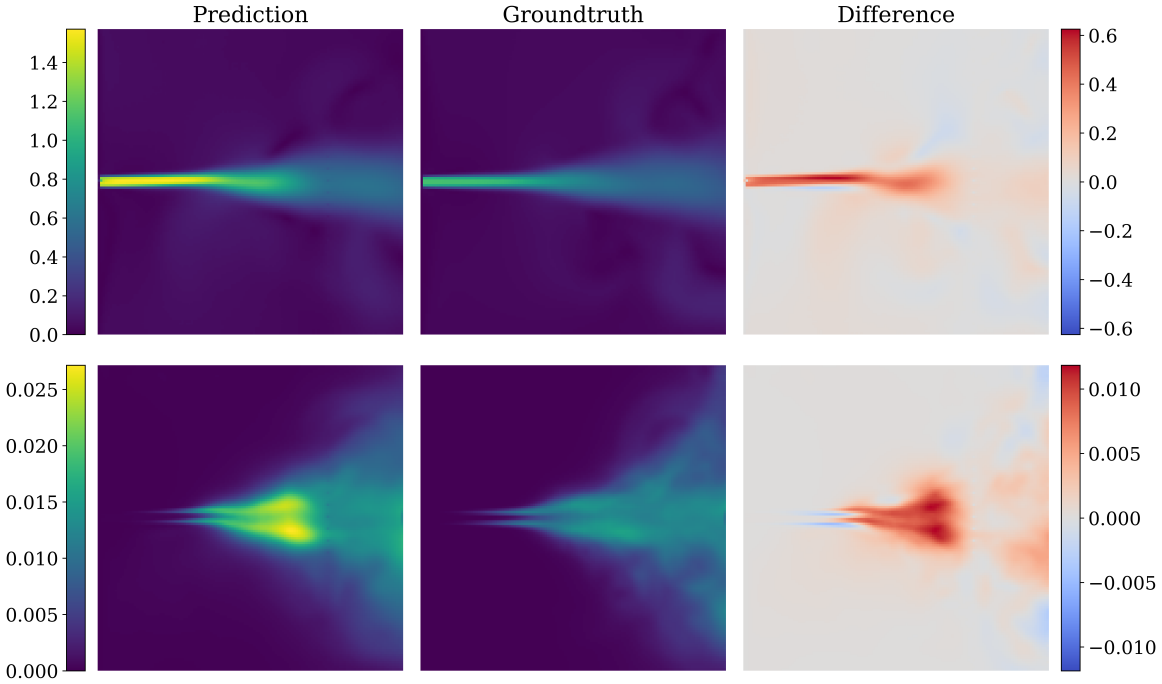


(b) Histogram of time-averaged kinetic energy.

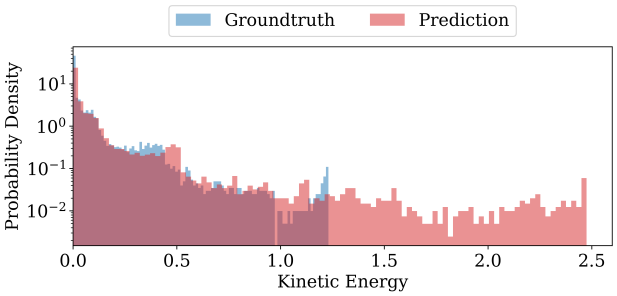


(c) Time-averaged kinetic energy spectrum.

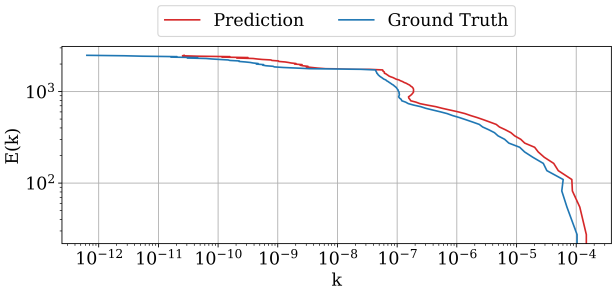
Figure 21. Statistical analysis of reconstructed flow fields using PAINT $(16, 0 \mid 16, 0)$ for Reynolds number 1100 and a *grid* probe point constellation.



(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.

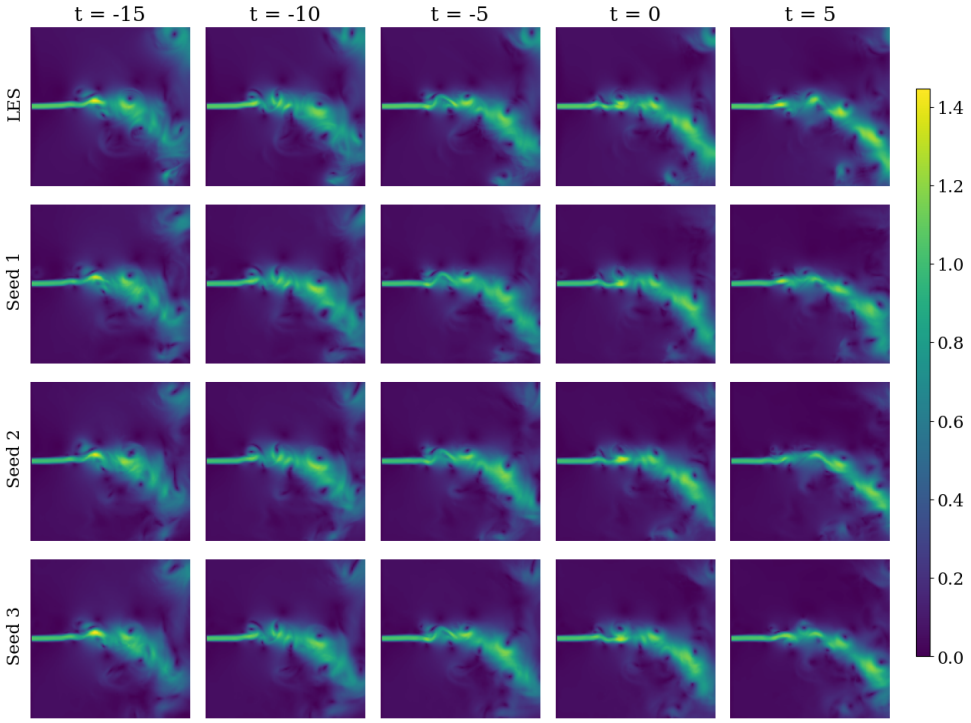


(b) Histogram of time-averaged kinetic energy.

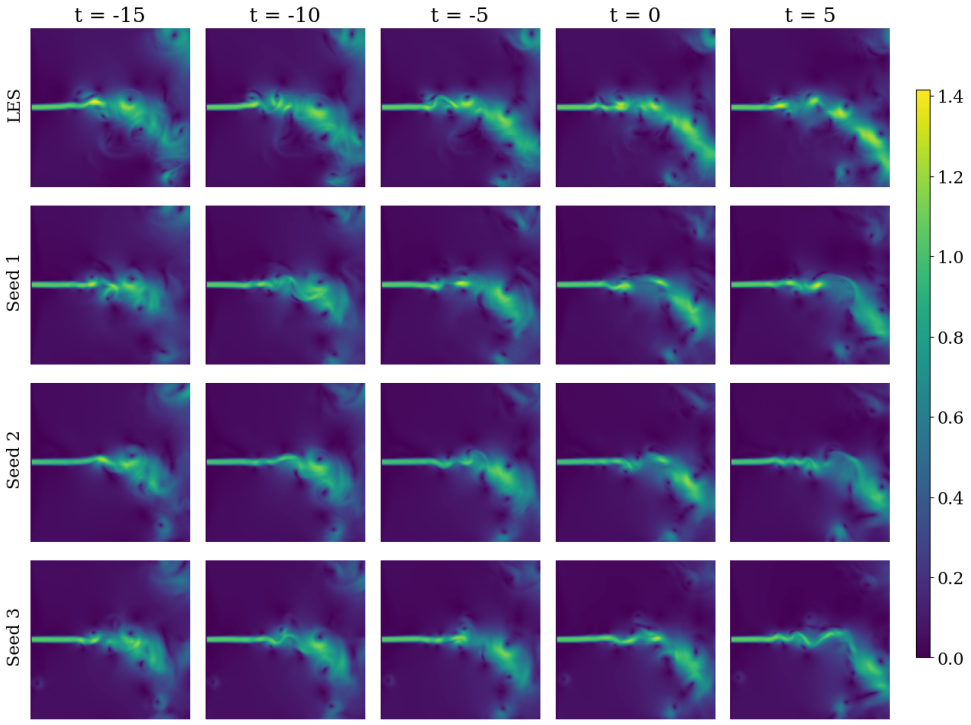


(c) Time-averaged kinetic energy spectrum.

Figure 22. Statistical analysis of reconstructed flow fields using PAINT $(16, 0 \mid 16, 0)$ for Reynolds number 1100 and a *vertical* probe point constellation.



(a) Grid probe constellation.



(b) Vertical probe constellation.

Figure 23. 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 and predictions using PAINT $(16, 8 \mid 16, 0)$ 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.

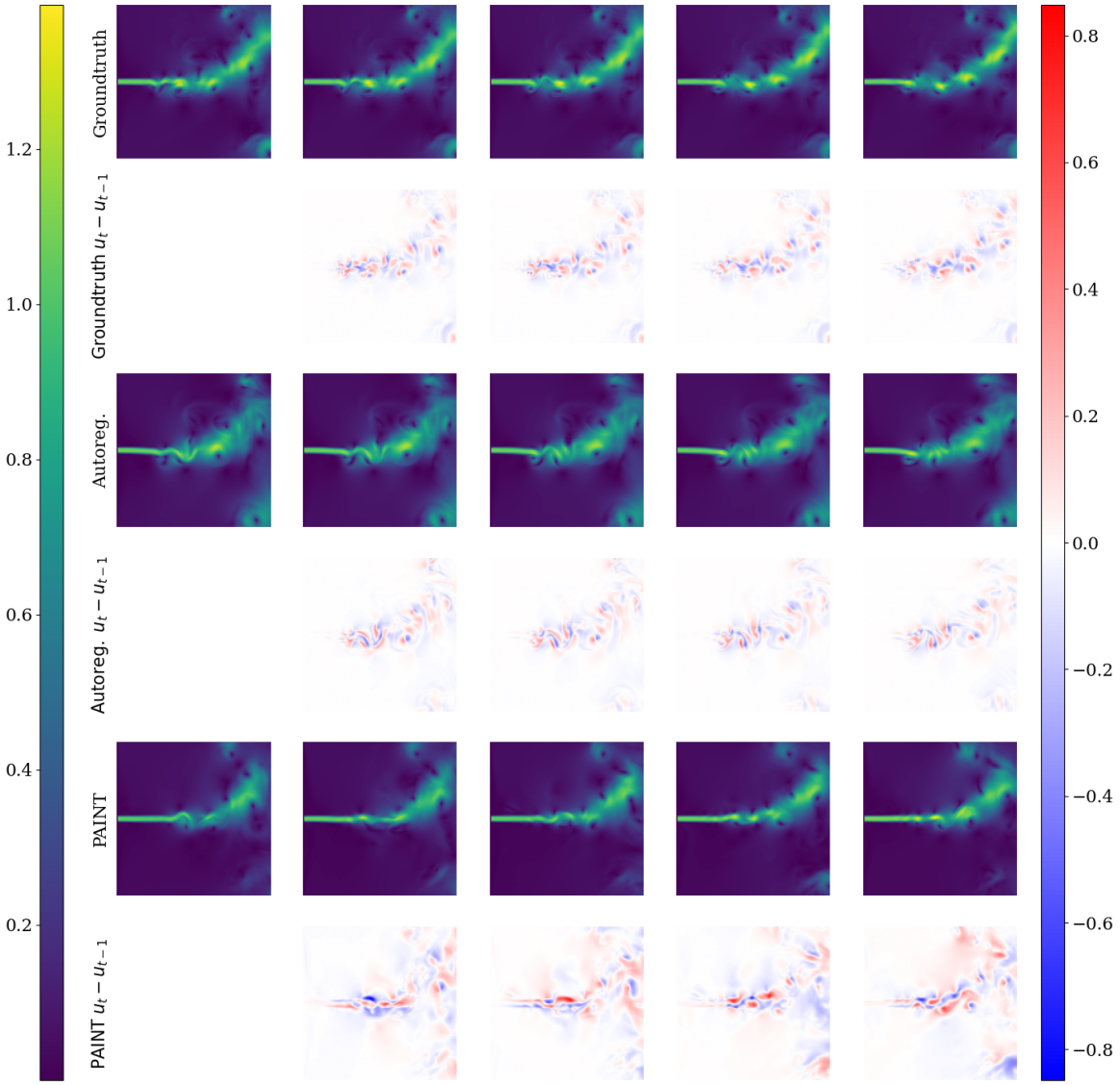


Figure 24. Qualitative illustration of the discontinuities induced by our parallel-in-time approach, using UNet $(1, 0 \mid 2, 2)$ and PAINT $(16, 8 \mid 16, 0)$ for Reynolds number 2100 and the *vertical* probe point constellation. Shown are consecutive prediction timesteps of a groundtruth trajectory, alongside the autoregressive baseline and the parallel-in-time approach. Due to its inherently temporally discontinuous nature, PAINT exhibits larger frame-to-frame changes. The autoregressive approach, on the other hand, displays frame-to-frame variations comparable to the ground truth, as each prediction is conditioned on previous outputs.