



MOX-Report No. 76/2026

**Learning PDE solution operators with variable initial conditions via
Latent Dynamics Networks**

Pizzamiglio, S.M.; Pagani, S.; Regazzoni, F.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it

<https://mox.polimi.it>

Learning PDE solution operators with variable initial conditions via Latent Dynamics Networks

Stefano Maria Pizzamiglio¹, Stefano Pagani¹, and Francesco Regazzoni¹

¹MOX Laboratory, Department of Mathematics, Politecnico di Milano, Italy
{stefanomaria.pizzamiglio, stefano.pagani, francesco.regazzoni}@polimi.it

Abstract

In many-query scenarios, data-driven surrogate models provide an efficient alternative to high-fidelity solvers for simulating physical systems governed by Partial Differential Equations (PDEs). In this context, the Latent Dynamics Network (LDNet) has recently demonstrated remarkable performance in predicting the response of spatio-temporal systems, combining Neural Ordinary Differential Equations with nonlinear dimensionality reduction. However, the original formulation assumes a fixed initial condition, limiting its applicability to many real-world applications where a system evolves from varying starting states. In this work, we overcome this limitation while keeping the end-to-end training procedure of the original LDNet and its encoder-free nature, which preserves its intrinsic independence from spatial resolution and grid topology. We infer the initial latent state directly from a small set of early-time observations, treating latent-state initialization as an adaptation problem, and investigate two strategies: an auto-decoding formulation and a meta-learning approach in which the initial latent state acts as a task-specific context variable. We demonstrate the accuracy of the proposed methods across diverse physical phenomena, spanning advection-diffusion, fluid dynamics, and solid mechanics. Meta-learning markedly accelerates latent-state inference and induces smoother, better-conditioned optimization landscapes, and spontaneously organizes the latent space into a structured representation that reflects physically meaningful features of the underlying dynamics. The coordinate-based decoder enables training from spatially subsampled data while recovering high-resolution solution fields at inference. The resulting approach provides an efficient and resolution-independent surrogate modeling framework for many-query simulations of time-dependent PDEs with varying initial conditions.

1 Introduction

Partial differential equations (PDEs) are fundamental for modeling complex physical systems across various scientific and engineering domains. However, in most applications, obtaining an analytical solution is intractable, necessitating the use of numerical methods to compute approximate solutions [1]. Although high-fidelity methods, such as finite difference [2], finite element [3], or finite volume method [4], can provide accurate results, they can be computationally expensive, especially in many-query settings (e.g., inverse design) or for real-time predictions (e.g., control) [5, 6]. Recently, data-driven approaches have emerged as a promising alternative to address these challenges, leveraging the latest advancements in machine learning algorithms and High Performance Computing (HPC). In particular, machine learning models can be trained to approximate the input-output behavior of high-fidelity numerical solvers, providing computationally efficient surrogates [7]. These approaches fall within the broader field of Scientific Machine Learning (SciML), which develops learning-based methods for problems arising in scientific computing [8, 9].

Traditionally, machine learning algorithms have been applied to map finite-dimensional inputs to finite-dimensional outputs. However, differential problems are characterized by inputs and outputs that belong to infinite-dimensional function spaces. Specifically, given two function spaces $\mathcal{U}$ and $\mathcal{V}$, an operator (e.g., the data-to-solution map associated with a PDE problem) $\mathcal{G} : \mathcal{U} \rightarrow \mathcal{V}$ maps a function $u \in \mathcal{U}$ to a function $v \in \mathcal{V}$. The goal of *operator learning* is therefore to approximate the operator

$\mathcal{G}$ with $\tilde{\mathcal{G}}_{\theta}$, parameterized by a set of learnable parameters θ [9]. For example, the seminal work on Deep Operator Networks (DeepONets) employs two deep neural networks (i.e., *branch* and *trunk* networks) to approximate the operator $\mathcal{G}$ [10]. More recently, Neural Operators (NOs), characterized by a sequence of integral kernel layers, generalize the Feed-Forward Neural Network (FFNN) architecture to infinite-dimensional spaces [9, 11]. Various NOs have been proposed in the literature, with the Fourier Neural Operator (FNO) and its variants being among the most widely used, leveraging the Fast Fourier Transform (FFT) to efficiently compute the integrals in the frequency domain [12–14].

Due to their discretization invariance, Neural Fields (NFs) have been demonstrated to be effective for operator learning [15, 16]. Specifically, a neural field, also known as a coordinate-based neural network or an implicit neural representation, consists of a neural network that approximates a continuous field (e.g., scalar, vector, or tensor field) by mapping input coordinates to field values [17]. In addition to their popularity in visual computing applications (e.g., Neural Radiance Fields for 3D scene representation [18]), their ability to continuously represent functions makes them natural candidates to approximate inputs and solutions of differential problems in a discretization-invariant manner. Thus, many SciML applications have adopted the neural field framework, beginning with physics-informed approaches such as Physics-Informed Neural Networks (PINNs) [19] or the Deep Ritz Method (DRM) [20]. Alternatively, in the context of operator learning for dynamics problems, the Dynamics-aware Implicit Neural Representations (DINo) framework [21] introduced a neural-field-based encode-process-decode pipeline, which has been recently extended in other works [15, 22–27].

The Latent Dynamics Network (LDNet) [28] is a deep learning architecture designed to learn the solution operator of time-dependent PDEs. It can be seen as a process-decode neural field, where two fully connected neural networks (FCNNs) are used as a latent-dynamics processor and a decoder, respectively. Specifically, the physical state is represented by a compact latent vector, and the network is able to predict the value of the PDE solution at any time and at any spatial position within the PDE domain. The LDNet has been successfully applied to the surrogate modeling of several time-dependent PDEs, ranging from advection-diffusion-reaction problems to computational fluid dynamics (CFD) and cardiac electrophysiology [29–32]. Compared to other similar neural-field-based approaches, such as DINo or CORAL [15], the LDNet learns the latent dynamics and the mapping back to the physical space simultaneously, in an end-to-end fashion. Training the dynamics and the decoder simultaneously not only simplifies the procedure, but also allows the neural surrogate to learn a latent representation that is optimal for both efficiently compressing the physical state and correctly predicting the evolution in time. However, the original LDNet implementation is restricted to problems starting from a fixed initial condition. This hinders its applicability in real-world scenarios, where the goal is to predict the system evolution from an unseen initial state. Notably, a concurrent graph-based extension of LDNet [32] explicitly identifies this as an open limitation of encoder-free latent-dynamics architectures, while addressing a complementary limitation (i.e., the lack of geometric inductive bias in the decoder). Related architectures have tackled this limitation through different strategies: an explicit encoder [33, 34], or a shifted time convention in which the dynamics network itself bootstraps the latent state from the initial observation [31, 35, 36]. Unlike these works, our approach remains entirely encoder-free and resolution-independent, and it does not require a specific time-stepping convention. Encoder-free latent state initialization was also explored in the context of PINNs, with the physics-informed dynamics representation learner (PiDo) [37], combining learnable initial latent codes with fully unsupervised physics-informed training. Additionally, in the same context, the Meta Auto-Decoder (MAD) [38] introduced the idea of improving auto-decoders with meta-learning, adopting a two-step training procedure where an initial pre-training phase is followed by a fine-tuning adaptation stage. Unlike these works, our approach adopts a data-driven supervised training that does not require explicit knowledge of the underlying PDE and can be seamlessly applied to experimental observations. Additionally, we explore a different meta-learning framework than the one in [38]. Indeed, our approach does not require multiple training stages and ensures fast adaptation to unseen tasks.

The contributions of this work are threefold. First, we formalize latent-state initialization from variable initial conditions as an encoder-free adaptation problem within the LDNet framework, thereby preserving its independence from spatial resolution and grid topology. Second, we introduce and

systematically compare two encoder-free strategies, an auto-decoding formulation (AD-IC-LDNet) and a meta-learning formulation (Meta-IC-LDNet), across four benchmarks spanning advection-diffusion-reaction, fluid dynamics, and nonlinear solid mechanics. Third, in the test cases where both strategies are compared, we show that meta-learning accelerates latent-state inference by more than one order of magnitude, yields a better-conditioned adaptation loss, and leads to a latent space whose topology reflects that of the physical state space. Indeed, in all test cases, the organization of the meta-learned latent space reflects physically meaningful features of the underlying dynamics.

2 Methods

The Initial-Condition Latent Dynamics Network (IC-LDNet) overcomes one of the main limitations of the original LDNet architecture [28] by accommodating varying initial conditions into the learning framework. In this section, we first review the standard LDNet and its training procedure. Subsequently, we introduce various strategies to inform the model about the initial state, detailing the corresponding training paradigms. Finally, we introduce two latent-space regularization strategies.

2.1 Problem

Given a spatial domain $\Omega \subset \mathbb{R}^d$ and a time interval $[0, T]$, consider a dynamical system with input $\mathbf{u} : [0, T] \rightarrow U \subseteq \mathbb{R}^{d_u}$ and output $\mathbf{y} : \Omega \times [0, T] \rightarrow Y \subseteq \mathbb{R}^{d_y}$. The goal is to learn the mapping between the input function $\mathbf{u}$ and the output $\mathbf{y}$, provided that it is well-posed and causal (i.e., the output at any time t depends solely on the input history up to t).

2.2 LDNet: architecture

The LDNet architecture consists of two coupled neural networks, the dynamics network $\mathcal{NN}_{\text{dyn}}$, with parameters $\mathbf{w}_{\text{dyn}}$, and the decoder network $\mathcal{NN}_{\text{rec}}$, with parameters $\mathbf{w}_{\text{rec}}$. Specifically, the LDNet evolves a latent state through a Neural Ordinary Differential Equation (Neural ODE) [39], and then decodes it into the output space for each point in space and time. The continuous model is summarized by the following dynamical system:

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathcal{NN}_{\text{dyn}}(\mathbf{s}(t), \mathbf{u}(t); \mathbf{w}_{\text{dyn}}) & \text{for } t \in (0, T] \\ \mathbf{s}(0) = \mathbf{0} \\ \tilde{\mathbf{y}}(\mathbf{x}, t) = \mathcal{NN}_{\text{rec}}(\mathbf{s}(t), \mathbf{u}(t), \mathbf{x}; \mathbf{w}_{\text{rec}}) & \text{for } \mathbf{x} \in \Omega \text{ and } t \in [0, T], \end{cases} \quad (1)$$

where $\mathbf{s}(t) \in \mathbb{R}^{d_s}$ is the vector of latent states at time t . In this work, we integrate the Neural ODE in time using a forward Euler scheme.

2.3 LDNet: training

The LDNet is trained in a supervised manner [40], given a training dataset of input and output pairs $\mathcal{S}_{\text{train}} = \{(\mathbf{u}_i(\tau), \mathbf{y}_i(\boldsymbol{\xi}, \tau))\}_{i=1}^{N_{\text{tr}}}$, where $\tau \in \mathcal{T}^i$ and $\boldsymbol{\xi} \in \mathcal{P}_{\tau}^i$, with $\mathcal{T}^i$ and $\mathcal{P}_{\tau}^i$ being the sets of time and space points, respectively, for the i -th training sample. These spatial points are not restricted to a specific grid or mesh, but can vary from sample to sample, and even from time point to time point. The loss function is defined as:

$$\mathcal{L}(\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}) = \sum_{i \in \mathcal{S}_{\text{train}}} \sum_{\tau \in \mathcal{T}^i} \sum_{\boldsymbol{\xi} \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\boldsymbol{\xi}, \tau; \mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}), \mathbf{y}_i(\boldsymbol{\xi}, \tau)) + \alpha_{\text{dyn}} \mathcal{R}(\mathbf{w}_{\text{dyn}}) + \alpha_{\text{rec}} \mathcal{R}(\mathbf{w}_{\text{rec}}), \quad (2)$$

where the symbol $\sum$ denotes the average operator, $\mathcal{E}$ is a suitable error measure (e.g., the squared error), $\mathcal{R}$ is a regularization term for the weights (e.g., L_2 -regularization [40]), with weighting coefficients α_{dyn} and α_{rec} . An optimizer is then used to minimize the loss function with respect to the network parameters:

$$(\mathbf{w}_{\text{dyn}}^*, \mathbf{w}_{\text{rec}}^*) = \underset{\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}}{\operatorname{argmin}} \mathcal{L}(\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}). \quad (3)$$

Note that both $\mathcal{NN}_{\text{dyn}}$ and $\mathcal{NN}_{\text{rec}}$ are trained simultaneously, in an end-to-end fashion. During inference, the trained LDNet can produce the output $\tilde{\mathbf{y}}(\mathbf{x}, t)$ (e.g., the approximate PDE solution) at any point in space and time, given an input function $\mathbf{u}(t)$. Specifically, starting from $\mathbf{s}(0) = \mathbf{0}$, the latent state is evolved by integrating in time the Neural ODE, and then decoded to the output space.

2.4 Initial-Condition Latent Dynamics Network

By leveraging the previously described LDNet architecture, it is possible to learn a surrogate reduced-order model for time-dependent PDEs. However, as evident from Eq. (1), the standard LDNet assumes that all trajectories start from the same initial condition, since the initial latent state $\mathbf{s}(0)$ is always an arbitrary constant (e.g., a zero vector). Note that the choice of a zero latent vector does not imply that the physical initial condition must be zero, since the decoder can learn to associate the fixed latent state with an arbitrary physical state. To overcome this limitation, we propose the Initial-Condition Latent Dynamics Network (IC-LDNet), embedding a set of initial observations into $\mathbf{s}(0)$. The dynamical system is thus modified as follows:

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathcal{NN}_{\text{dyn}}(\mathbf{s}(t), \mathbf{u}(t); \mathbf{w}_{\text{dyn}}) & \text{for } t \in (0, T] \\ \mathbf{s}(0) = \mathbf{s}_0 = \Psi(\{\mathbf{y}(\boldsymbol{\xi}, \tau)\}_{\boldsymbol{\xi} \in \mathcal{P}_\tau, \tau \in \mathcal{T}_{\text{obs}}}; \mathbf{w}_{\text{IC}}) \\ \tilde{\mathbf{y}}(\mathbf{x}, t) = \mathcal{NN}_{\text{rec}}(\mathbf{s}(t), \mathbf{u}(t), \mathbf{x}; \mathbf{w}_{\text{rec}}) & \text{for } \mathbf{x} \in \Omega \text{ and } t \in [0, T], \end{cases} \quad (4)$$

where $\mathcal{T}_{\text{obs}}$ denotes the set of initial observation times. The function Ψ , parametrized by $\mathbf{w}_{\text{IC}}$, maps the system outputs available at these observation time steps to the initial latent state $\mathbf{s}_0$. This generalized notation accommodates systems where only the snapshot at $t = 0$ is considered, while also allowing for a short sequence of initial time steps to encode higher-order or non-Markovian dynamics when necessary. In the following subsections, different strategies to define Ψ are presented.

2.4.1 IC-LDNet with encoder

A straightforward approach to defining Ψ is to use an encoder network, which maps the initial condition to the initial latent state:

$$\mathbf{s}_0 = \mathcal{NN}_{\text{enc}}(\{\mathbf{y}(\boldsymbol{\xi}, \tau)\}_{\boldsymbol{\xi} \in \mathcal{P}_\tau, \tau \in \mathcal{T}_{\text{obs}}}; \mathbf{w}_{\text{IC}}), \quad (5)$$

where $\mathcal{NN}_{\text{enc}}$ is a generic encoder network with parameters $\mathbf{w}_{\text{IC}}$. The encoder can be jointly trained with the dynamics and decoder networks in an end-to-end fashion. At inference time, the initial condition is then passed through the encoder to obtain $\mathbf{s}_0$, and then the latent state is evolved and decoded as in the original implementation. However, standard encoder architectures are tied to a specific data format (e.g., convolutional networks [41] for regular grids, graph neural networks [42] for unstructured meshes), which would compromise the discretization invariance of the LDNet. Although discretization-agnostic encoders exist (e.g., attention-based architectures [24]), the introduction of an additional network, with respect to the original LDNet, substantially increases the number of parameters and thus the computational and memory cost of training and inference. Hence, in this work, the IC-LDNet with encoder is not implemented, in favor of encoder-free strategies.

2.4.2 IC-LDNet with auto-decoder

An alternative strategy to define Ψ is to adopt a decoder-only or *auto-decoder* approach [37, 43–45]. In this case, Ψ associates each initial condition in the training set with a specific trainable vector. Specifically, given the training set $\mathcal{S}_{\text{train}} = \{(\mathbf{u}_i(\tau), \mathbf{y}_i(\boldsymbol{\xi}, \tau))\}_{i=1}^{N_{\text{tr}}}$, the initial latent state for the i -th training sample is defined as:

$$\mathbf{s}_0^{(i)} = \mathbf{e}_i^T \mathbf{W}_{\text{IC}}^{\text{train}} \quad \text{with} \quad \mathbf{W}_{\text{IC}}^{\text{train}} \in \mathbb{R}^{N_{\text{tr}} \times d_s}, \quad (6)$$

where $\mathbf{W}_{\text{IC}}^{\text{train}}$ stores all the initial latent codes for the training set, and $\mathbf{e}_i$ is the i -th element of the canonical basis of $\mathbb{R}^{N_{\text{tr}}}$. In particular, $\mathbf{W}_{\text{IC}}^{\text{train}}$ is initialized by sampling from a Gaussian distribution

$\mathcal{N}(0, 10^{-4})$ and optimized jointly with the network parameters. Hence, the loss function becomes:

$$\begin{aligned} \mathcal{L}(\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}, \mathbf{W}_{\text{IC}}^{\text{train}}) = & \sum_{i \in \mathcal{S}_{\text{train}}} \sum_{\tau \in \mathcal{T}^i} \sum_{\xi \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\xi, \tau; \mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}, \mathbf{W}_{\text{IC}}^{\text{train}}), \mathbf{y}_i(\xi, \tau)) \\ & + \alpha_{\text{dyn}} \mathcal{R}(\mathbf{w}_{\text{dyn}}) + \alpha_{\text{rec}} \mathcal{R}(\mathbf{w}_{\text{rec}}) + \alpha_{\text{IC}} \mathcal{R}(\mathbf{W}_{\text{IC}}^{\text{train}}), \end{aligned} \quad (7)$$

where $\mathcal{R}(\mathbf{W}_{\text{IC}}^{\text{train}})$ is a regularization term with weight α_{IC} . The optimization problem is thus:

$$(\mathbf{w}_{\text{dyn}}^*, \mathbf{w}_{\text{rec}}^*, \mathbf{W}_{\text{IC}}^{\text{train}*}) = \underset{\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}, \mathbf{W}_{\text{IC}}^{\text{train}}}{\text{argmin}} \quad \mathcal{L}(\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}, \mathbf{W}_{\text{IC}}^{\text{train}}). \quad (8)$$

This approach has the advantage of being encoder-free, thus avoiding the previously mentioned drawbacks. However, at inference time, it is necessary to fit a new latent code for the initial condition of each new example in the evaluation set $\mathcal{S}_{\text{eval}} = \{(\mathbf{u}_i(\tau), \mathbf{y}_i(\xi, \tau))\}_{i=1}^{N_{\text{ev}}}$. Specifically, we initialize the collection of codes for the new initial conditions (i.e., $\mathbf{W}_{\text{IC}}^{\text{eval}} \in \mathbb{R}^{N_{\text{ev}} \times d_s}$) by sampling from the same Gaussian distribution as before. Then, after freezing the trained network parameters $(\mathbf{w}_{\text{dyn}}^*, \mathbf{w}_{\text{rec}}^*)$, we minimize the following loss function with respect to the new latent codes:

$$\mathcal{L}(\mathbf{W}_{\text{IC}}^{\text{eval}}) = \sum_{i \in \mathcal{S}_{\text{eval}}} \sum_{\tau \in \mathcal{T}_{\text{eval}}^i} \sum_{\xi \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\xi, \tau; \mathbf{W}_{\text{IC}}^{\text{eval}}), \mathbf{y}_i(\xi, \tau)) + \alpha_{\text{IC}} \mathcal{R}(\mathbf{W}_{\text{IC}}^{\text{eval}}). \quad (9)$$

Notably, $\mathcal{T}_{\text{eval}}^i \subseteq [0, T_{\text{eval}}]$, with $T_{\text{eval}} \ll T$, since the initial latent state should be inferred from only a short initial portion of the trajectory. In the simplest case of $T_{\text{eval}} = 0$, only the initial condition is used. For non-Markovian systems, however, a short time window with $T_{\text{eval}} > 0$ may be required to provide sufficient information about the system state. Once optimized, these initial latent codes are evolved through the Neural ODE and decoded to predict the entire system trajectory. In the rest of this work, this approach is referred to as the Auto-Decoder IC-LDNet, or AD-IC-LDNet.

2.4.3 IC-LDNet with meta-learning

A second encoder-free strategy relies on meta-learning [38, 46], specifically on the CAVIA framework [47], following the idea presented for CORAL [15]. While CORAL applies CAVIA-style meta-learning to a two-stage encode-process-decode pipeline, we integrate it into the end-to-end LDNet training scheme, where latent dynamics and reconstruction are learned jointly rather than sequentially. The overall structure is similar to the IC-LDNet with auto-decoder presented in Section 2.4.2, but it differs in the

Algorithm 1 Meta-IC-LDNet: training procedure

Require: Training dataset $\mathcal{S}_{\text{train}} = \{(\mathbf{u}_i(\tau), \mathbf{y}_i(\xi, \tau))\}_{i=1}^{N_{\text{tr}}}$

Require: Initial meta-parameters $\theta = \{\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}\}$, randomly initialized

Require: Learning rates α and β , and number of inner-loop iterations K

Require: Inner loss $\mathcal{L}_i^{\text{I}}(\theta, \mathbf{s}_0^{(i)}) = \sum_{\tau \in \mathcal{T}_{\text{inner}}^i} \sum_{\xi \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\xi, \tau; \theta, \mathbf{s}_0^{(i)}), \mathbf{y}_i(\xi, \tau))$ with $i \in [1, N_{\text{tr}}]$

Require: Outer loss $\mathcal{L}_i^{\text{O}}(\theta, \mathbf{s}_0^{(i)}) = \sum_{\tau \in \mathcal{T}_{\text{outer}}^i} \sum_{\xi \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\xi, \tau; \theta, \mathbf{s}_0^{(i)}), \mathbf{y}_i(\xi, \tau))$ with $i \in [1, N_{\text{tr}}]$

1: **while** not converged **do**

2: **for** $i \in \{1, \dots, N_{\text{tr}}\}$ **do**

3: Initialize task-specific parameters: $\mathbf{s}_0^{(i,0)} \leftarrow \mathbf{0}$ ▷ Reset codes

4: **for** $k \in \{1, \dots, K\}$ **do**

5: $\mathbf{s}_0^{(i,k)} \leftarrow \mathbf{s}_0^{(i,k-1)} - \alpha \nabla_{\mathbf{s}_0^{(i,k-1)}} \mathcal{L}_i^{\text{I}}(\theta, \mathbf{s}_0^{(i,k-1)})$ ▷ Find optimal $\mathbf{s}_0$

6: **end for**

7: **end for**

8: $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{i=1}^{N_{\text{tr}}} \mathcal{L}_i^{\text{O}}(\theta, \mathbf{s}_0^{(i,K)}(\theta))$ ▷ Update meta-parameters

9: **end while**

Algorithm 2 Meta-IC-LDNet: inference procedure

Require: Evaluation dataset $\mathcal{S}_{\text{eval}} = \{(\mathbf{u}_i(\tau), \mathbf{y}_i(\boldsymbol{\xi}, \tau))\}_{i=1}^{N_{\text{ev}}}$

Require: Trained meta-parameters $\boldsymbol{\theta} = \{\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}\}$

Require: Learning rate α , and number of inner-loop iterations K

Require: Inner loss $\mathcal{L}_i^{\text{I}}(\boldsymbol{\theta}, \mathbf{s}_0^{(i)}) = \sum_{\tau \in \mathcal{T}_{\text{inner}}^i} \sum_{\boldsymbol{\xi} \in \mathcal{P}_{\tau}^i} \mathcal{E}(\tilde{\mathbf{y}}_i(\boldsymbol{\xi}, \tau; \boldsymbol{\theta}, \mathbf{s}_0^{(i)}), \mathbf{y}_i(\boldsymbol{\xi}, \tau))$ with $i \in [1, N_{\text{ev}}]$

- 1: **for** $i \in \{1, \dots, N_{\text{ev}}\}$ **do**
 - 2: Initialize task-specific parameters: $\mathbf{s}_0^{(i,0)} \leftarrow \mathbf{0}$ ▷ Initialize codes
 - 3: **for** $k \in \{1, \dots, K\}$ **do**
 - 4: $\mathbf{s}_0^{(i,k)} \leftarrow \mathbf{s}_0^{(i,k-1)} - \alpha \nabla_{\mathbf{s}_0^{(i,k-1)}} \mathcal{L}_i^{\text{I}}(\boldsymbol{\theta}, \mathbf{s}_0^{(i,k-1)})$ ▷ Find optimal $\mathbf{s}_0$
 - 5: **end for**
 - 6: Compute $\tilde{\mathbf{y}}_i(\mathbf{x}, t; \boldsymbol{\theta}, \mathbf{s}_0^{(i,K)})$ with $\mathbf{x} \in \Omega$ and $t \in [0, T]$ ▷ Predict spatio-temporal dynamics
 - 7: **end for**
-

training procedure. Specifically, $\mathbf{s}_0^{(i)}$ represents a set of context parameters specific to the i -th training example, while $\boldsymbol{\theta} = \{\mathbf{w}_{\text{dyn}}, \mathbf{w}_{\text{rec}}\}$ are the meta-parameters, shared across all tasks. In the context of the IC-LDNet, the training procedure is reported in Algorithm 1. Note that the gradient with respect to the meta-parameters is computed by back-propagating through the inner loop, since $\mathbf{s}_0^{(i,K)}$ depends on $\boldsymbol{\theta}$. Moreover, meta-parameter update can be performed using more sophisticated approaches than gradient descent, such as Adam [48].

In the original CAVIA implementation, the training batch is divided into two parts, one used in the inner loop and one in the final gradient step [47]. In contrast, the CORAL architecture uses the entire batch for both phases [15]. In this work, in the inner loop, the loss function is computed only on a short time interval $[0, T_{\text{inner}}^{\text{train}}]$, with $T_{\text{inner}}^{\text{train}} \ll T$, and at discrete times $\tau \in \mathcal{T}_{\text{inner}}^i \subseteq [0, T_{\text{inner}}^{\text{train}}]$. After the inner loop, the meta-parameters are updated using the loss computed on the entire trajectory (i.e., $\tau \in \mathcal{T}_{\text{outer}}^i \subseteq [0, T]$). Alternatively, the outer loss function can be computed only on the remainder of the trajectory (i.e., $\tau \in \mathcal{T}_{\text{outer}}^i \subseteq (T_{\text{inner}}^{\text{train}}, T]$). For each test case in the results section, the specific choice of the outer-loss time domain will be specified. At inference time, as summarized in Algorithm 2, the meta-parameters $\boldsymbol{\theta}$ are fixed, and only the context parameters $\mathbf{s}_0^{(i)}$ are updated. Similarly to the training procedure, the inner loss is computed only on a short time interval $[0, T_{\text{inner}}^{\text{eval}}]$ to fit the context parameters to the initial condition. For simplicity, the same notation as in the training procedure is used, with $\mathcal{T}_{\text{inner}}^i \subseteq [0, T_{\text{inner}}^{\text{eval}}]$. After the inner loop, the optimized context parameters are used to predict the entire trajectory. Importantly, in the meta-learning approach, the procedure used to infer unseen initial latent states is already embedded in the training process, unlike in the AD-IC-LDNet. This provides the basis for a faster and more robust adaptation to unseen initial conditions. In this work, this approach is called Meta-IC-LDNet.

2.4.4 Regularization strategies for training stability and latent-space uniqueness

Depending on the specific test case, the IC-LDNet may suffer from training instability and may produce non-unique latent representations. Specifically, the former issue is mainly observed when training the Meta-IC-LDNet, possibly due to the reset of the context parameters at each iteration. Instead, the latter is particularly problematic for the AD-IC-LDNet, due to the random initialization of the trainable initial codes. We therefore propose two regularization strategies. For each of the four test cases in Section 3, we discuss whether regularization is needed and, if so, which strategy is adopted.

Curriculum learning The first strategy is to introduce a *continuation* or *curriculum-learning* approach [49], where the loss function (the outer one for the meta-learning approach) is computed on a progressively longer time horizon as training proceeds. Indeed, the random initialization of the dynamics network, combined with long time horizons, may initially produce diverging trajectories that hinder training. This strategy requires a careful choice of hyperparameters, such as the initial time

horizon and the rate of increase. Moreover, it requires longer training, for the model gradually learns to predict longer trajectories.

Latent-state penalty The second strategy is to actively prevent the latent trajectory from diverging excessively from the initial state. To achieve this, a straightforward approach is to add an L_2 penalty on the final latent state, with a suitable weighting coefficient. For the Meta-IC-LDNet, a more sophisticated approach is to penalize the distance (e.g., Mean Squared Error, MSE) between a subset of the trajectory states and the latent codes that would be obtained by inner-loop optimization at the same time instants. The advantage of the latter approach is that the penalty term converges to zero in the limit of perfect optimization. However, it requires additional inner-loop optimization steps, increasing training time and cost.

3 Results

To demonstrate the performance of the IC-LDNet and compare the auto-decoder and meta-learning frameworks, the proposed methods are evaluated on four benchmark problems, as summarized in Table 1. Due to the observed advantages of the Meta-IC-LDNet over the AD-IC-LDNet, we tackle the last two test cases with just the former approach. These tests are designed to thoroughly assess the model predictive accuracy and generalization capabilities across diverse physical domains. Unless otherwise specified, training is performed on a single custom NVIDIA A100 GPU (64 GB VRAM) hosted on CINECA’s LEONARDO HPC cluster [50], and the training budgets and inference times are summarized in Table 2.

For each test case, we performed a preliminary hyperparameter-tuning campaign, where several combinations of architectural and optimization parameters were tested to find those that correspond to the best generalization on the validation set. Additionally, the robustness of the hyperparameter combinations to the choice of the specific random seeds was verified during tuning. Furthermore, the test set was always set aside for just the final inference. In Appendix A, the final hyperparameter configurations are reported for each test case and IC-LDNet variant.

Test case	Architecture	NRMSE (Test set)		
ADR	AD-IC-LDNet	3.5×10^{-3}		
	Meta-IC-LDNet	2.9×10^{-3}		
Static cylinder	AD-IC-LDNet	p	u	v
		4.1×10^{-3}	4.7×10^{-3}	7.1×10^{-3}
	Meta-IC-LDNet	p	u	v
		2.6×10^{-3}	2.7×10^{-3}	4.1×10^{-3}
Rotating cylinder	Meta-IC-LDNet	p	u	v
		7.5×10^{-3}	1.2×10^{-2}	1.3×10^{-2}
Nonlinear beam	Meta-IC-LDNet	d_x	d_y	d_z
		4.8×10^{-2}	4.7×10^{-2}	4.5×10^{-2}

Table 1: Comparison of per-component test-set range-Normalized Root Mean Square Error (NRMSE) across the four test cases.

3.1 Test case 1: 1D advection-diffusion-reaction

The Advection-Diffusion-Reaction (ADR) equation is a second-order PDE that provides a fundamental benchmark for evaluating the performance of neural networks in approximating PDE solutions. Physically, the solution can be interpreted as the concentration of a substance being transported and diffused in a medium, while also undergoing a reaction process [1].

Test case	Architecture	Hardware (Training budget)	Inference mini-batch size	Inference time ^a
ADR	AD-IC-LDNet	HPC cluster (<24 h) ^b	100	1.14 s/ex.
	Meta-IC-LDNet	HPC cluster (24 h)	100	0.04 s/ex.
Static cylinder	AD-IC-LDNet	HPC cluster (24 h)	6	21.11 s/ex.
	Meta-IC-LDNet	HPC cluster (24 h)	6	0.23 s/ex.
Rotating cylinder	Meta-IC-LDNet	HPC cluster (24 h) ^c	2	0.87 s/ex.
Nonlinear beam	Meta-IC-LDNet	Workstation (Convergence) ^d	8	0.15 s/ex.

Unless otherwise noted, HPC training runs were executed until the end of the 24-hour queue.

^a Measured in seconds per example (s/ex.) on the test set.

^b The loss reached a plateau before the 24 h HPC queue limit and training was interrupted.

^c Fine-tuning is then performed for an additional 24 hours.

^d No fixed training budget was set: training was run until observed convergence of the losses (see Appendix B).

Table 2: Training hardware and inference speed for different test cases and architectures.

Specifically, we consider the following differential 1D problem:

$$\begin{cases} \frac{\partial z(x, t)}{\partial t} - \mu_1 \frac{\partial^2 z(x, t)}{\partial x^2} - \mu_2 \frac{\partial z(x, t)}{\partial x} + \mu_3 z(x, t) = f(x, t) & x \in [-1, 1], t \in (0, T] \\ z(x, 0) = z_0(x) & x \in [-1, 1] \\ \text{(Periodic BCs)}, \end{cases} \quad (10)$$

with constants μ_1, μ_2, μ_3 defining the diffusion, advection, and reaction coefficients, respectively. To create the dataset, the PDE coefficients (i.e., constant inputs $\mathbf{u}(t)$) are randomly sampled as follows:

$$\begin{cases} \mu_1 \sim \mathcal{U}(0, 0.05) \\ \mu_2 \sim \mathcal{U}(-0.1, 0.1) \\ \mu_3 \sim \mathcal{U}(0, 0.01), \end{cases} \quad (11)$$

with $\mathcal{U}(a, b)$ denoting the uniform distribution in the interval $[a, b]$. The initial conditions, on the other hand, are defined as sine waves with randomly sampled amplitude and phase shift:

$$z_0(x) = A \sin(\pi x + \varphi), \quad (12)$$

where the parameters are sampled as:

$$\begin{cases} A \sim \mathcal{U}(0.5, 1.5) \\ \varphi \sim \mathcal{U}(-\pi, \pi). \end{cases} \quad (13)$$

Given $f(x, t) = 0$ and the initial condition defined in (12), the PDE in (10) admits the following analytical solution:

$$z(x, t) = A \exp\left(-(\mu_1 \pi^2 + \mu_3)t\right) \sin(\pi(x + \mu_2 t) + \varphi). \quad (14)$$

Equation (14) shows that the evolution of the problem can be fully described by two states (i.e., amplitude and phase). Although the analytical expression of the solution is available, we generate the dataset using the same FFT-based solver adopted in the original LDNet implementation [28]. Specifically, the dataset comprises 1000 samples; for each sample, the solution is evaluated at 101 equally spaced spatial points and 100 time steps. The dataset is subsequently randomly split into training, validation, and test sets, with 800, 100, and 100 samples, respectively.

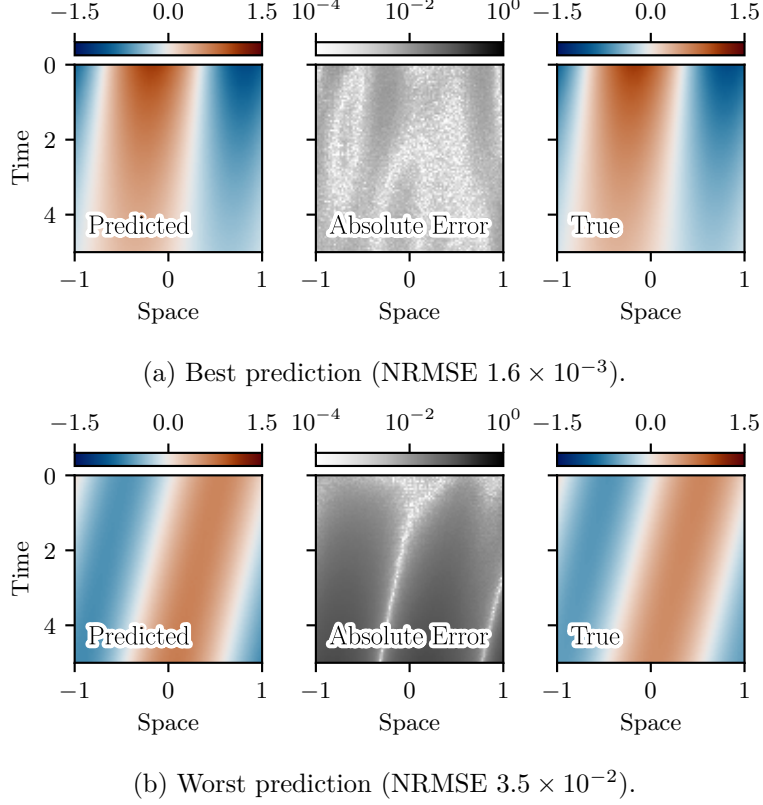


Figure 1: ADR test case: inference on the test set (AD-IC-LDNet). Predicted solution, absolute error, and true solution in the space-time domain for the test-set samples with the lowest (a) and highest (b) NRMSE.

3.1.1 AD-IC-LDNet on the ADR test case

We first train the AD-IC-LDNet surrogate defined in Section 2.4.2. In this case, the architecture of the model and the hyperparameters are similar to those used in the original LDNet paper [28] and are summarized in Table 3 (see Appendix A). Specifically, the latent dimension is $d_s = 2$, which corresponds to the intrinsic dimension of the solution manifold, as observed previously. The trained model is evaluated on the test set by first optimizing the initial latent states of the unseen samples, followed by the evolution of the latent dynamics and the reconstruction to the physical space. In this

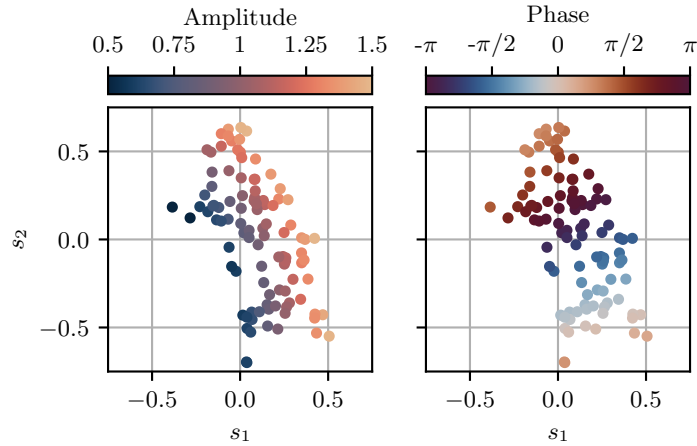


Figure 2: ADR test case: initial latent states (AD-IC-LDNet). Test-set initial codes, colored by the amplitude (left) and the phase (right) of the corresponding initial conditions.

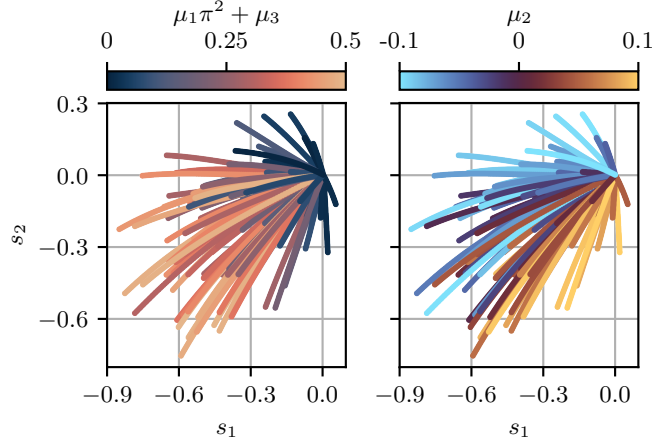


Figure 3: ADR test case: latent trajectories (AD-IC-LDNet). Test-set latent trajectories, translated to start from the origin, colored by $\mu_1\pi^2 + \mu_3$, which combines diffusion and reaction (left), and by the advection coefficient μ_2 (right).

context, in Figures 1a and 1b, respectively, we present the best and worst test-set predictions in terms of range-Normalized Root Mean Square Error (NRMSE). Interestingly, the predictive performance varies considerably between samples, with the worst sample characterized by an NRMSE of 3.5×10^{-2} , significantly higher than the 1.6×10^{-3} of the best sample. However, an average test-set NRMSE of 3.5×10^{-3} demonstrates that the AD-IC-LDNet still accurately predicts the ADR dynamics for the majority of samples.

Additionally, we analyze how the learned latent space is organized. As anticipated, a latent dimension of two would have a natural correspondence with the amplitude and phase of the exact solution. However, it is not guaranteed that the learned latent variables will directly correspond to these physical parameters, but rather to one of the infinitely many equivalent parametrizations. To investigate this, in Figure 2, the initial test-set states are plotted in the 2D latent space, colored according to the amplitude and phase of the corresponding initial conditions. Interestingly, we observe that the two physical parameters vary along two virtually perpendicular directions. However, the latent codes are not periodically arranged according to the phase, as an optimal encoding reflecting the topology of the physical space would require. Indeed, the learned representation is redundant, as a single state in the physical space may admit two alternative representations in the latent space. Finally, Figure 3 shows the link between the latent trajectories, moved to a common starting point, and the PDE coefficients. Here, we observe that the combined effect of diffusion and reaction is dominant in examples that move in the amplitude direction, while the advection coefficient influences the drift in the phase direction. This is in accordance with the physical interpretation of the analytical solution in Equation (14).

3.1.2 Meta-IC-LDNet on the ADR test case

As an alternative to the AD-IC-LDNet, we investigate the Meta-IC-LDNet described in Section 2.4.3. To have a fair comparison, the number of parameters of $\mathcal{NN}_{\text{dyn}}$ and $\mathcal{NN}_{\text{rec}}$, as well as the latent dimension d_s , are kept the same as in Section 3.1.1 and in the original LDNet paper [28]. Here, the main difference is the use of SiLU activations [51], due to training issues with the original tanh. Overall, the full set of hyperparameters is summarized in Table 3 in Appendix A. With respect to Algorithm 1, the inner loss $\mathcal{L}_i^I$ is computed only on the initial condition (i.e., $T_{\text{inner}}^{\text{train}} = 0$), while the outer loss $\mathcal{L}_i^O$ considers the remainder of the trajectory (i.e., $\mathcal{T}_{\text{outer}}^i = (0, T]$).

Following Algorithm 2, the model is evaluated on the test set, and Figure 4 shows the best and worst predictions in terms of NRMSE. By comparing Figures 1b and 4b, we can observe that the worst prediction of the Meta-IC-LDNet is considerably more accurate than that of the AD-IC-LDNet.

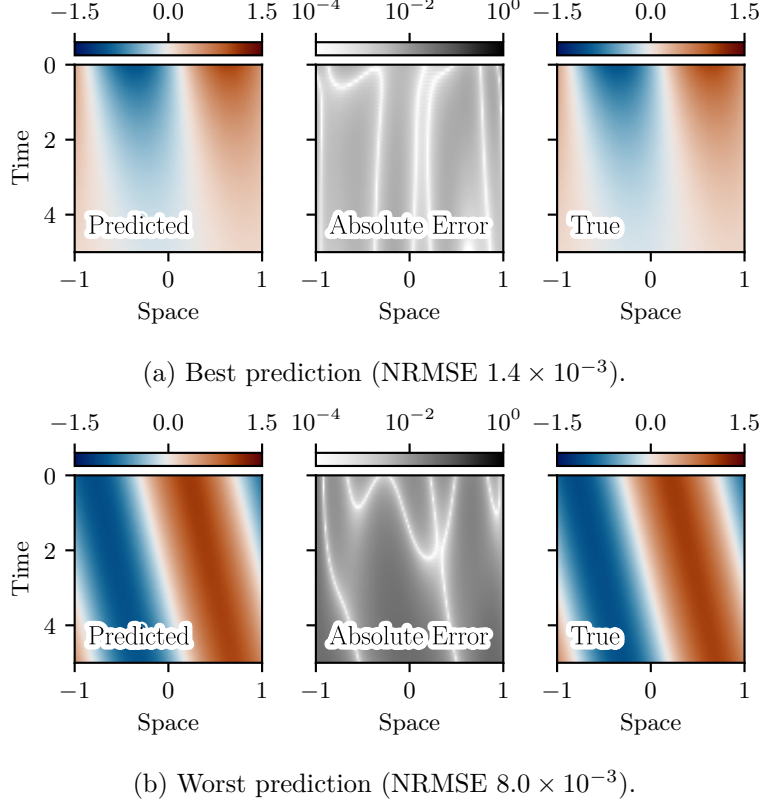


Figure 4: ADR test case: inference on the test set (Meta-IC-LDNet). Predicted solution, absolute error, and true solution in the space-time domain for the test-set samples with the lowest (a) and highest (b) NRMSE.

Remarkably, the meta-learning approach leads to a more evenly distributed error across the test set and an overall better accuracy, with an average NRMSE of 2.9×10^{-3} .

As before, the 2D latent space can be conveniently visualized. Specifically, Figure 5a shows the initial test-set latent states, colored according to the amplitude and phase of the corresponding initial conditions. Interestingly, compared to Figure 2, the latent space shows the desired topology, with the phase periodically distributed. This suggests that meta-learning encourages a more structured organization of the latent space, avoiding performance degradation during long roll-outs. In fact, when considering advection-dominated evolutions in the physical space, the closed latent space topology corresponds to trajectories that remain within the training region, where the decoder performs the best, as visualized in Figure 5b. Finally, in Figure 6, we depict a nearly perfect correlation between the PDE coefficients and the initial velocity in the latent space, using polar coordinates ρ and ϑ . As observed in Figure 3, the diffusion and reaction coefficients influence the movement in the amplitude direction, while the advection coefficient affects the motion in the phase direction, consistently with the physical interpretation of ADR.

3.1.3 AD-IC-LDNet versus Meta-IC-LDNet: initial-code fitting on the ADR test case

Both the AD-IC-LDNet and the Meta-IC-LDNet share the same approach of treating the initial latent codes as learnable vectors (i.e., auto-decoding). However, at training time, the former learns them together with the model parameters, while the latter adopts a two-loop approach. At inference time, both strategies require an optimization procedure to find the optimal initial codes for the unseen samples. Since $d_s = 2$, the loss function and the optimization trajectory can be conveniently visualized. In particular, Figure 7 shows the loss-function contours for both the AD-IC-LDNet and the Meta-IC-LDNet, along with the optimization paths to fit the initial codes for the first nine test-set examples. Comparing Figures 7a and 7b, we observe that meta-learning favors a better conditioned

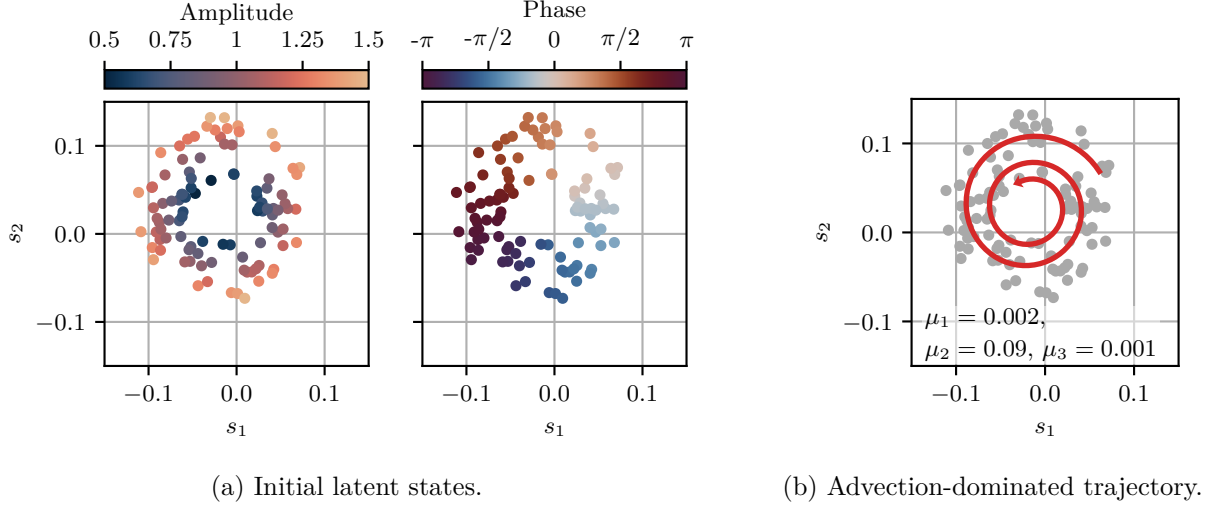


Figure 5: ADR test case: latent space (Meta-IC-LDNet). (a) Test-set initial codes, colored by the amplitude (left) and the phase (right) of the corresponding initial conditions. (b) Latent trajectory (red) of an advection-dominated case with the reported PDE coefficients, integrated over a longer time horizon than in training (i.e, $T = 50$), superimposed on the test-set initial codes (gray).

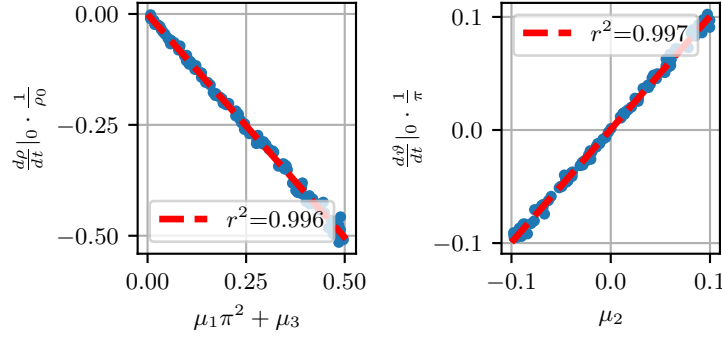
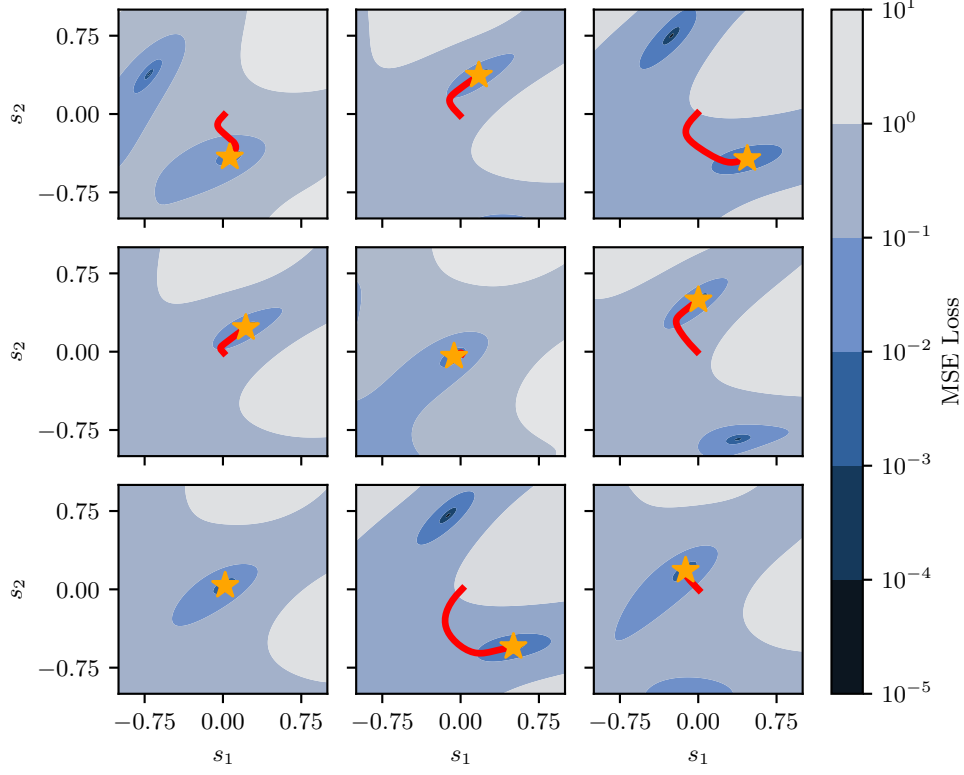


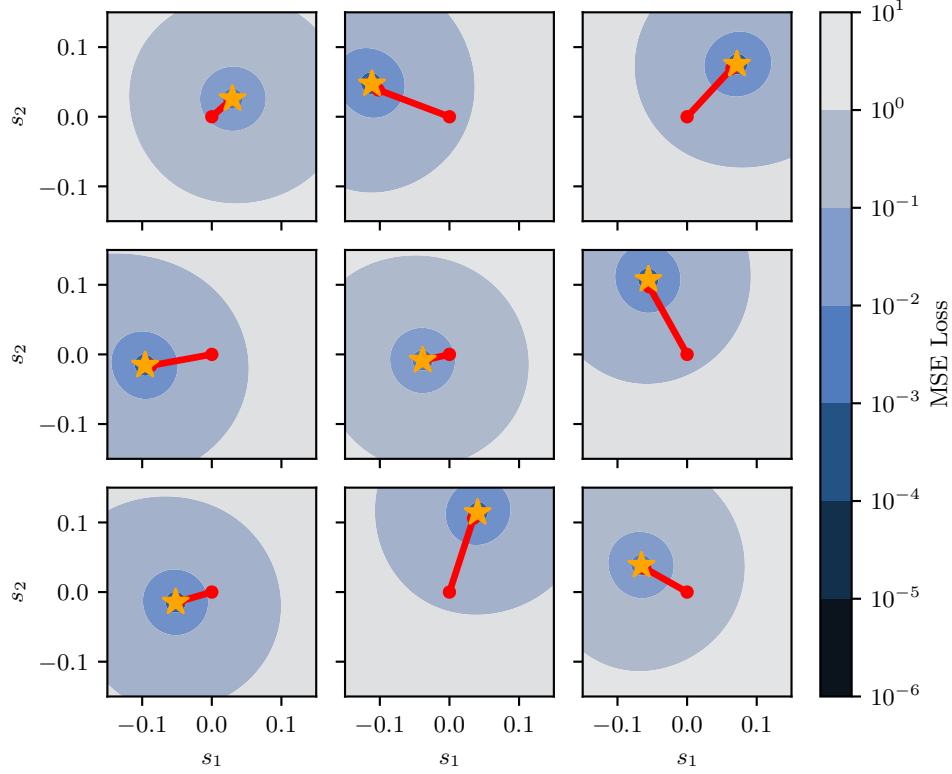
Figure 6: ADR test case: initial latent velocity against PDE parameters (Meta-IC-LDNet). For each test-set sample, the initial latent velocity is expressed in polar coordinates (ρ, ϑ) with respect to the center of the initial codes. The radial velocity, normalized by the initial radius ρ_0 , is plotted against $\mu_1\pi^2 + \mu_3$ (left), and the angular velocity, normalized by π , against μ_2 (right). The red dashed lines are linear fits, with the corresponding coefficient of determination r^2 .

and more regular loss. Indeed, the circular level curves correspond to a well-conditioned Hessian (i.e., similar maximum and minimum eigenvalues) near the minimum, guaranteeing a rapid convergence of gradient-based optimization [52]. In contrast, the AD-IC-LDNet exhibits an irregular landscape, with multiple local minima. These differences are the result of forcing the Meta-IC-LDNet to learn new tasks in only three optimization steps, which requires a well-conditioned landscape. Thus, the Meta-IC-LDNet is significantly more efficient in adapting to new tasks, allowing for rapid online inference.

It is also interesting to analyze how the inner-loop loss function of the Meta-IC-LDNet evolves during training to observe how it achieves the final well-conditioned and convex landscape. In particular, Figure 8 plots the inner-loop loss function at selected epochs, for the same validation-set sample, and should be read together with Figure 28 (see Appendix B). After an initial phase where the loss is essentially flat, it develops into a narrow valley, reflecting an ill-conditioned Hessian. As



(a) AD-IC-LDNet



(b) Meta-IC-LDNet

Figure 7: ADR test case: initial-code fitting on the test set. Each panel refers to one of the first nine test-set samples (same in (a) and (b)) and shows the contours of the MSE loss on the initial condition as a function of the initial latent code (s_1, s_2) . The red line is the optimization path and the orange star is the final code (i.e., 10^4 Adam iterations from a random initialization close to the origin for the AD-IC-LDNet, and three gradient-descent steps from $\mathbf{s}_0 = \mathbf{0}$ for the Meta-IC-LDNet).

training progresses, between 17 000 and 18 000 epochs, the condition number of the Hessian suddenly improves, leading to the final regular and smooth loss landscape. At the same time, the minimum value of the loss function continues to decrease. From preliminary experiments, we observed that this sudden improvement of the loss landscape is a crucial step to achieve good training performance and generalization, and the Meta-IC-LDNet appears to be very sensitive to the choice of hyperparameters. Hence, fast adaptation and correct latent-space topology come at the expense of robustness and ease of training.

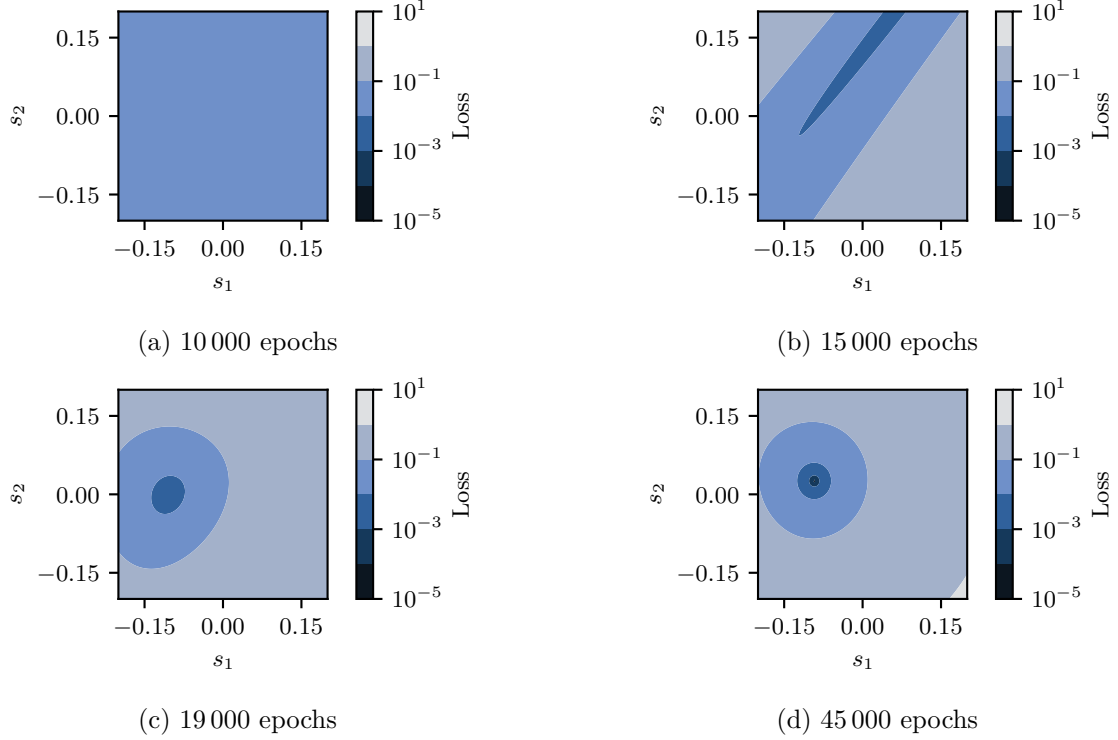


Figure 8: ADR test case: evolution of the inner-loop loss function (Meta-IC-LDNet). Contours of the inner-loop loss as a function of the initial latent code (s_1, s_2) , for the same validation-set sample at selected training epochs. The corresponding training and validation losses are shown in Figure 28.

3.2 Test case 2A: 2D flow around a static circular cylinder

The second test case covers the 2D flow around a static circular cylinder for a Reynolds number $Re_D = 100$, corresponding to the regime of laminar vortex shedding [53]. The computational domain and boundary conditions are schematically represented in Figure 9. Specifically, the domain is a rectangle of length $L = 100$ and height $H = 20$, with a circular-cylinder cutout of diameter $D = 1$ placed at a distance $L_{in} = 20$ from the inlet and centered vertically. The top and bottom boundaries are treated as slip walls, while a uniform velocity profile U_∞ is imposed at the inlet, ramping up from zero over one second of simulation time. At the outlet, the pressure is set to zero, together with a zero-gradient condition on the velocity field. Finally, the cylinder surface is treated as a no-slip wall. The fluid properties are set to $\nu = 1$, resulting in $Re_D = U_\infty$. The case is simulated with OpenFOAM v2506, using a structured mesh with approximately one million cells, with progressive refinement boxes to ensure adequate resolution around the cylinder and in its proximal wake region. The simulation is run using the `pimpleFoam` laminar-flow solver, for a total of four seconds of simulation time, with an adaptive time step to maintain a Courant number $Co \leq 1.0$. Figure 10a shows the time history of the drag and lift coefficients, highlighting the periodic vortex shedding behavior, while Figure 10b shows the FFT of the lift coefficient for the fully developed flow. The numerical Strouhal number is approximately $St = 0.167$, in good agreement with experimental values [53]. Finally, in order to build

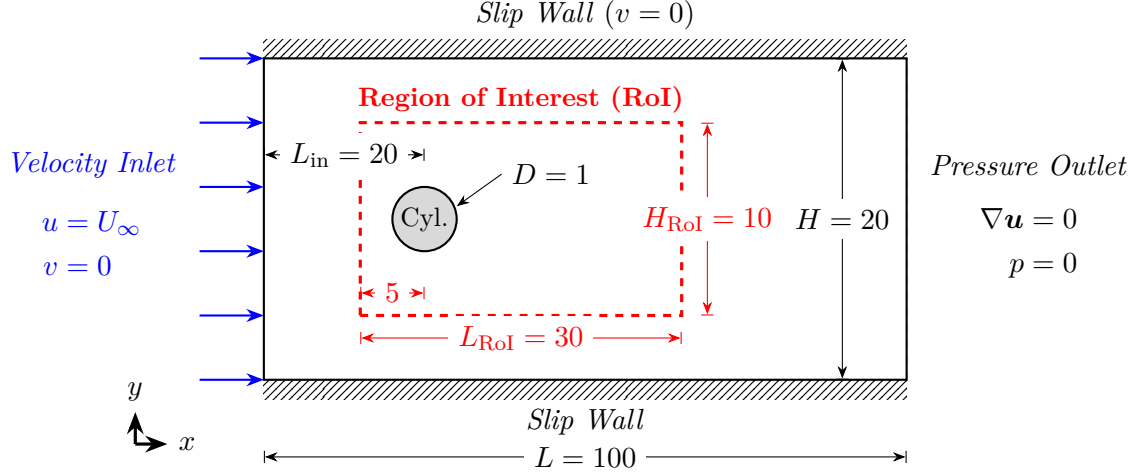


Figure 9: 2D flow around a circular cylinder: domain and boundary conditions (not to scale). The dataset fields are sampled in the region of interest (RoI, red dashed box).

the deep-learning dataset, the pressure and velocity fields are sampled every 10^{-3} seconds of simulation time, on a uniform 600×200 structured grid of points, covering the red-dashed box in Figure 9. In all cases, the flow variables are extracted from the value at the nearest cell center. Since the flow has a period of approximately 60 time steps, the dataset consists of 60 overlapping sequences, starting from consecutive moments within the shedding period, each of length 61 time steps. The dataset is randomly split into training, validation, and test sets, with an 80%-10%-10% ratio. Since each training sequence spans a full shedding period, every test initial condition also appears as an intermediate state of some training trajectory. This test case therefore primarily assesses the identification of the correct latent state from a single snapshot, whereas generalization to unseen states is assessed in the other test cases. Additionally, in this test case, $\mathbf{u}(t)$ is absent, because the dynamical system is autonomous.

3.2.1 AD-IC-LDNet on the static-cylinder test case

On the cylinder-flow dataset, the AD-IC-LDNet is trained to predict the evolution of pressure and velocity fields from an initial snapshot. To reduce the computational burden, the original 200×600 grid of the region of interest is downscaled to a coarser 50×150 resolution. As summarized in Table 4 (see Appendix A) the architecture consists of a fully connected $\mathcal{NN}_{\text{dyn}}$ and a FiLM-modulated [54]

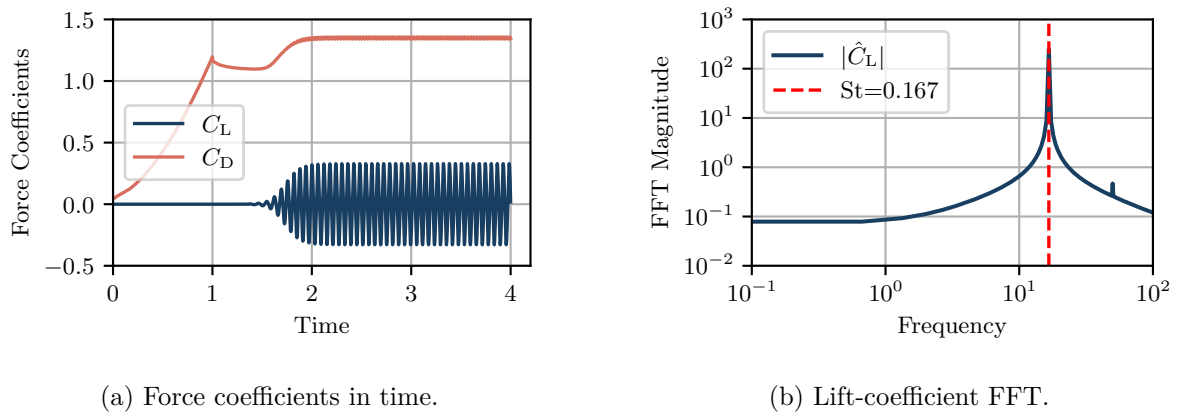


Figure 10: 2D flow around a circular cylinder: force coefficients from the OpenFOAM simulation at $\text{Re}_D = 100$. (a) Time history of the lift and drag coefficients. (b) FFT magnitude of the lift coefficient for the developed flow. The dashed line identifies the shedding frequency, corresponding to $\text{St} = 0.167$.

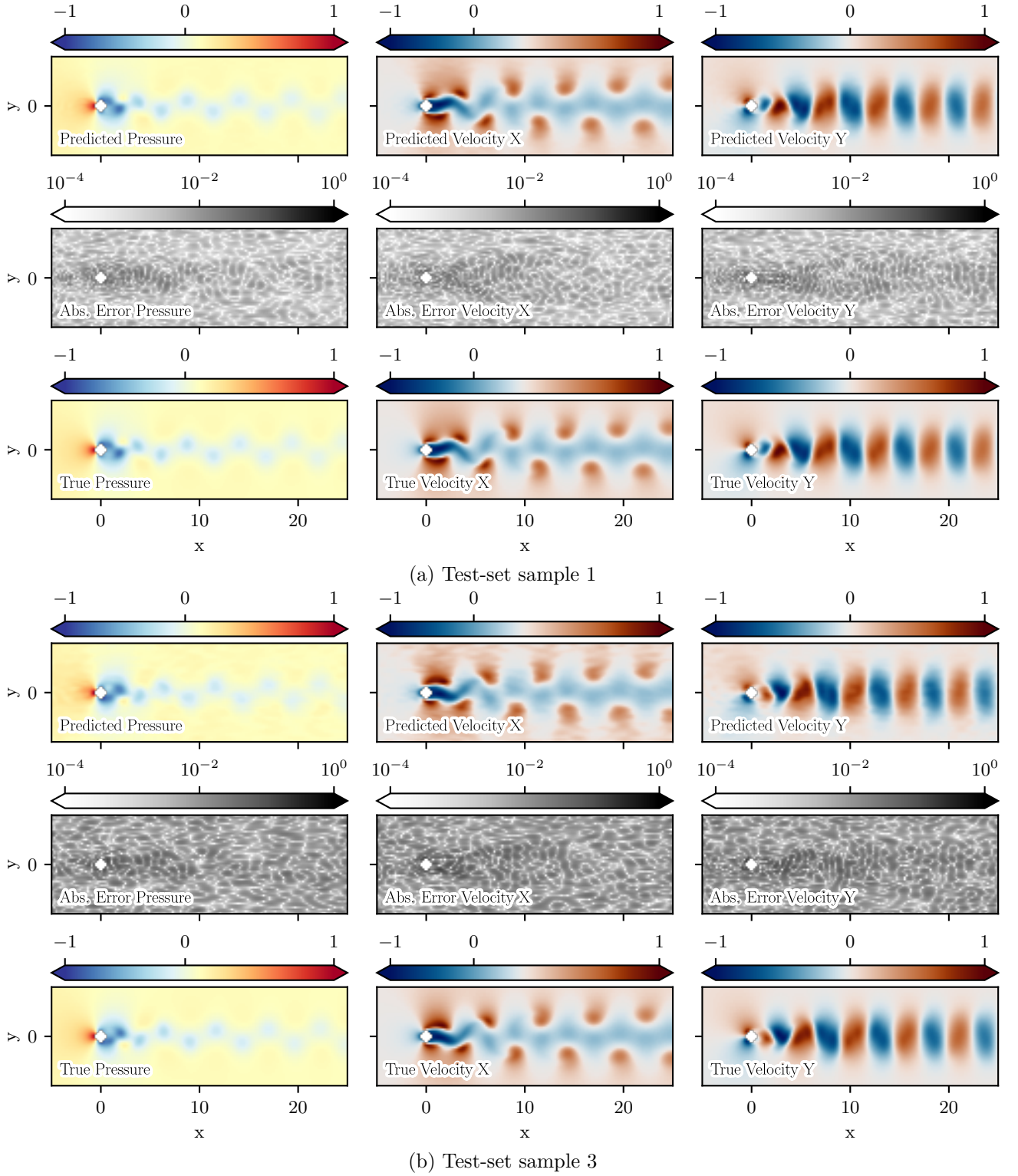


Figure 11: Static-cylinder test case: final-time normalized test-set fields (AD-IC-LDNet). Predicted (top), absolute error (middle), and true (bottom) pressure, x -velocity, and y -velocity at the last time step of test-set samples 1 (a) and 3 (b), numbered as in Figure 13. The fields are normalized to $[-1, 1]$ using the training-set minimum and maximum values.

SIREN-based [55] $\mathcal{NN}_{\text{rec}}$. Specifically, the modulation approach consists of a phase shift of the sinusoidal activations, similar to that proposed in [56], and is summarized as follows:

$$\begin{cases} \mathbf{h}_0 = \mathbf{x}(t) \\ \mathbf{h}_i = \sin\left(\omega_i \cdot (\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i + \mathbf{W}_i^{\text{cond}} \mathbf{s}(t) + \mathbf{b}_i^{\text{cond}})\right) \quad \forall i = 1, \dots, N_{\text{hidden}}^{\text{dec}} \\ \tilde{\mathbf{y}} = \mathbf{W}_{\text{out}} \mathbf{h}_{N_{\text{hidden}}^{\text{dec}}} + \mathbf{b}_{\text{out}} \end{cases} \quad (15)$$

where $\mathbf{h}_i$ is the output of the i -th hidden layer and ω_i is a frequency hyperparameter (set to 30 for every layer). $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weights and biases of the i -th hidden layer, while $\mathbf{W}_i^{\text{cond}}$ and $\mathbf{b}_i^{\text{cond}}$ are the weights and biases of the conditioning branch, which takes as input the latent state $\mathbf{s}(t)$ and produces a modulation signal for the activations. Note that, although the original SIREN features ω_i only in the first layer, our implementation is equivalent in terms of initialization. Due to the essentially periodic nature of the flow, similar to a simple harmonic oscillator, we expect two latent variables to be necessary and sufficient to capture the main dynamics. The two-dimensional state should provide a representation equivalent to the one that considers the phase and the angular velocity of the vortex shedding. Hence, the latent space dimension is set to $d_s = 2$.

The trained model is evaluated on the test set and Figure 11 shows the predicted fields for two selected test-set samples at the final time step. Note that the training-set minimum and maximum values are used to linearly normalize the data. The first example (i.e., Figure 11a) exhibits a smaller final error, while the second (i.e., Figure 11b) shows a more noticeable degradation of the surrogate solution. In particular, the error is dominated by components characterized by a high spatial frequency. Overall, the average NRMSE across the test dataset is 4.1×10^{-3} for the pressure channel and 4.7×10^{-3} and 7.1×10^{-3} for the two velocity components, respectively. Furthermore, since the IC-LDNet is a neural field, we can compute derived quantities, such as the vorticity field, by means of automatic

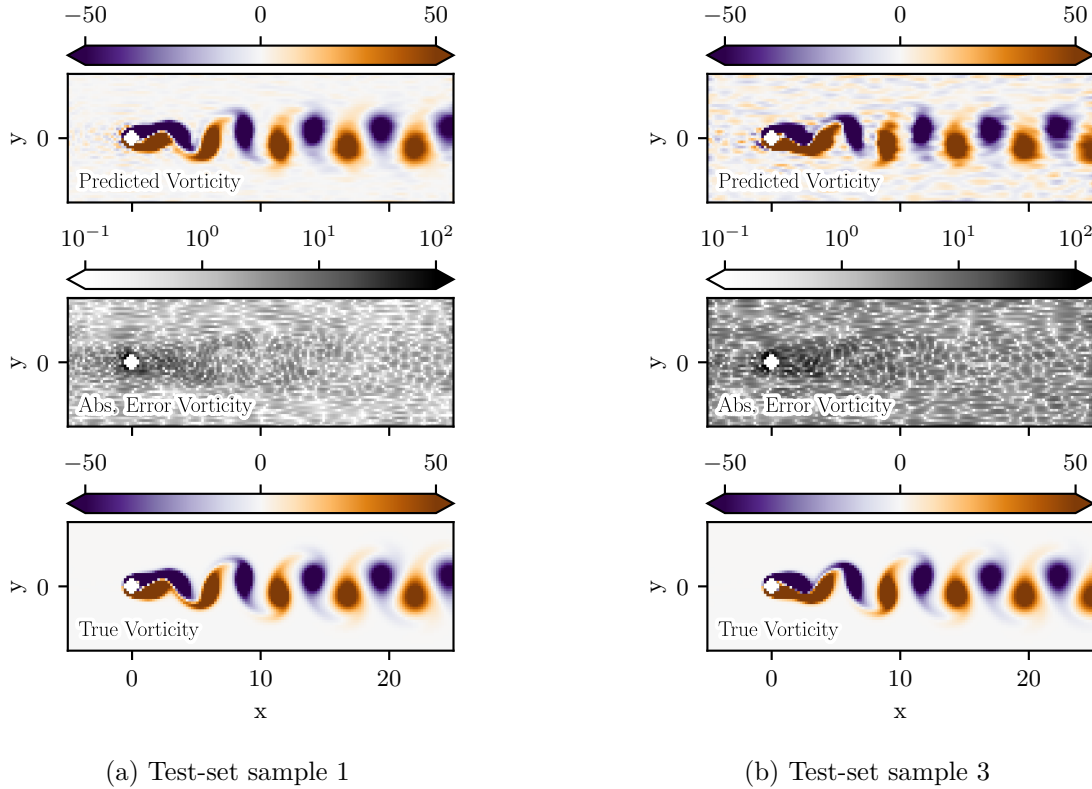


Figure 12: Static-cylinder test case: final-time test-set vorticity (AD-IC-LDNet). Predicted (top), absolute error (middle), and true (bottom) vorticity at the last time step of test-set samples 1 (a) and 3 (b). The predicted vorticity is obtained by automatic differentiation of the predicted velocity field, while the true one is computed by OpenFOAM.

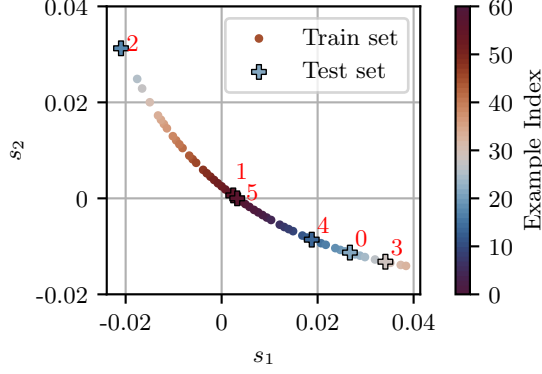


Figure 13: Static-cylinder test case: initial latent states (AD-IC-LDNet). Training-set initial codes, colored by the index of the initial time step within the shedding period, and test-set initial codes (crosses), numbered for reference in Figures 11 and 12.

differentiation. In particular, Figure 12 shows the predicted vorticity field for the same two test-set samples at the final time step, where the high-frequency error is even more evident, as the differentiation amplifies it.

Interestingly, the unevenly distributed error can be explained by the suboptimal latent-space structure produced by AD-IC-LDNet. In particular, Figure 13 shows the initial codes for the training and test sets, colored according to the time index of the initial condition within the shedding period. For clarity, the test codes are also uniquely identified by a number. We observe that the initial codes are arranged in a one-dimensional manifold, where the position along the manifold is related to the time within the shedding period. Hence, although the flow is periodic, the neural surrogate produces open trajectories in the latent space, leading to degraded predictive performance outside the training region. That explains why, in Figures 11 and 12, the predictions for sample 3 were significantly less accurate than those for sample 1.

3.2.2 Meta-IC-LDNet on the static-cylinder test case

The Meta-IC-LDNet is also trained on the cylinder-flow dataset with the hyperparameters summarized in Table 4 (see Appendix A). In this case, the inner loss function only covers the initial condition (i.e., $T_{\text{inner}}^{\text{train}} = 0$), while the outer loss is computed on the entire time sequence (i.e., $\mathcal{T}_{\text{outer}}^i = [0, T]$).

Tuning experiments showed sudden instabilities during the training process, which culminated in the model failing to train properly. Hence, we employed the curriculum-learning strategy described in Section 2.4.4. In particular, the outer loss is computed on the first two time steps for 500 epochs, then a new time step is added every 100 epochs, eventually covering the entire sequence of 61 time steps. Hence, the model first learns the initial latent codes and velocities, which, for this test case, are crucial to capture the periodic behavior. Indeed, as shown in Figure 14, this approach leads to a latent space with the expected topology, where the codes are arranged in a closed manifold, revealing a correspondence with C_L and dC_L/dt , and therefore with the angular position and velocity of the shedding, respectively.

The correct topology and the physical adherence of the latent space allow for long roll-outs without loss of accuracy, as the latent trajectory is closed and remains within the training region. In this regard, Figure 15 shows the predicted fields for test-set sample 1 at the last time step, demonstrating the stability over time. The topologically consistent latent-space also leads to an even distribution of the prediction error across the test set, with all the examples characterized by a similar NRMSE of around 2.6×10^{-3} , 2.7×10^{-3} , and 4.1×10^{-3} , for pressure and velocity components, respectively. Furthermore, Figure 16 shows the predicted vorticity for the same sample and time step, highlighting the ability of the model to accurately capture the whirling structures. Hence, these results demonstrate that the Meta-IC-LDNet, with the help of curriculum learning, trains a surrogate model that captures the periodicity and the physics of the flow, leading to accurate and stable predictions.

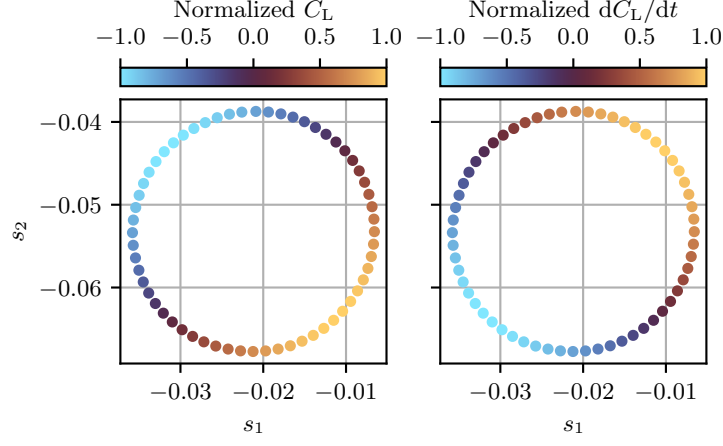


Figure 14: Static-cylinder test case: physical interpretation of initial latent states (Meta-IC-LDNet). Initial codes of the training, validation, and test sets combined, colored by the lift coefficient C_L (left) and its time derivative dC_L/dt (right) at the initial time, both min-max normalized to $[-1, 1]$.

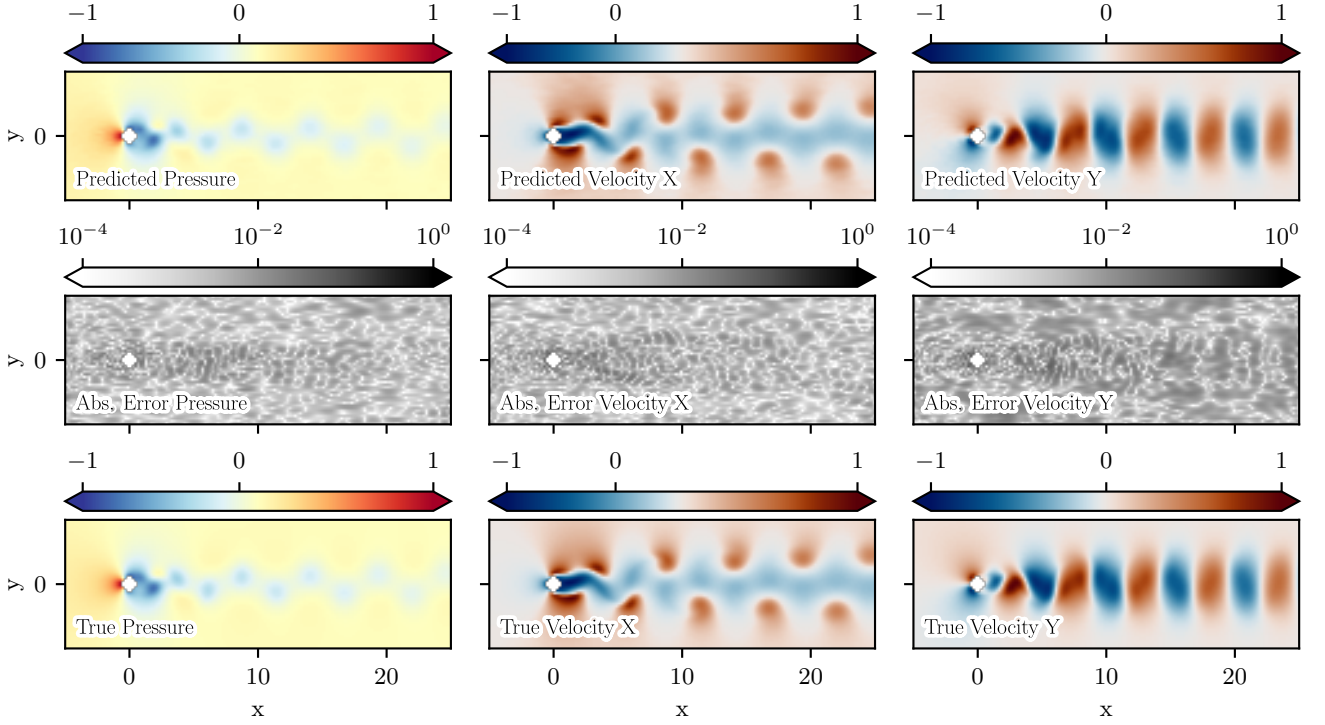


Figure 15: Static-cylinder test case: final-time normalized test-set fields (Meta-IC-LDNet). Predicted (top), absolute error (middle), and true (bottom) normalized fields at the last time step of test-set sample 1, arranged as in Figure 11.

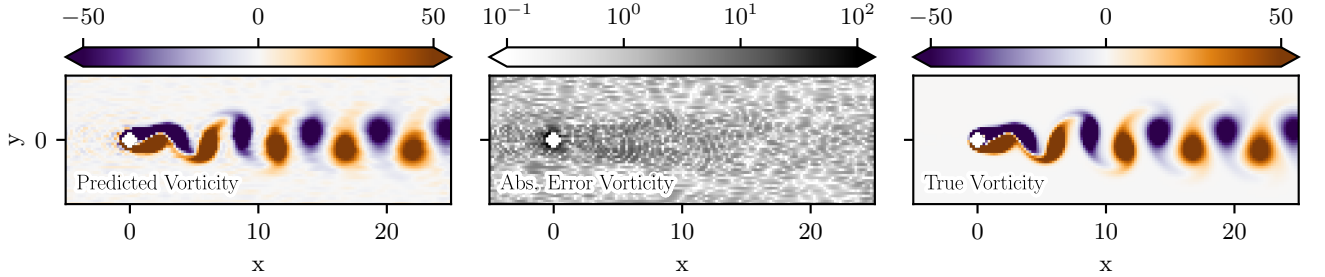


Figure 16: Static-cylinder test case: final-time test-set vorticity (Meta-IC-LDNet). Predicted (left), absolute error (center), and true (right) vorticity at the last time step of test-set sample 1, computed as in Figure 12.

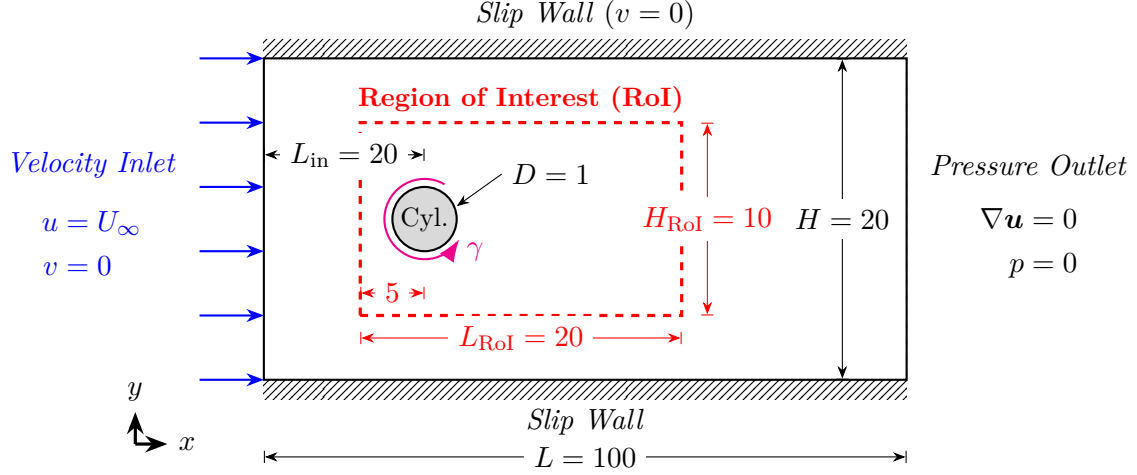


Figure 17: 2D flow around a rotating circular cylinder: domain and boundary conditions (not to scale). Same configuration as in Figure 9, with the cylinder rotating at a time-varying angular velocity γ and a smaller region of interest.

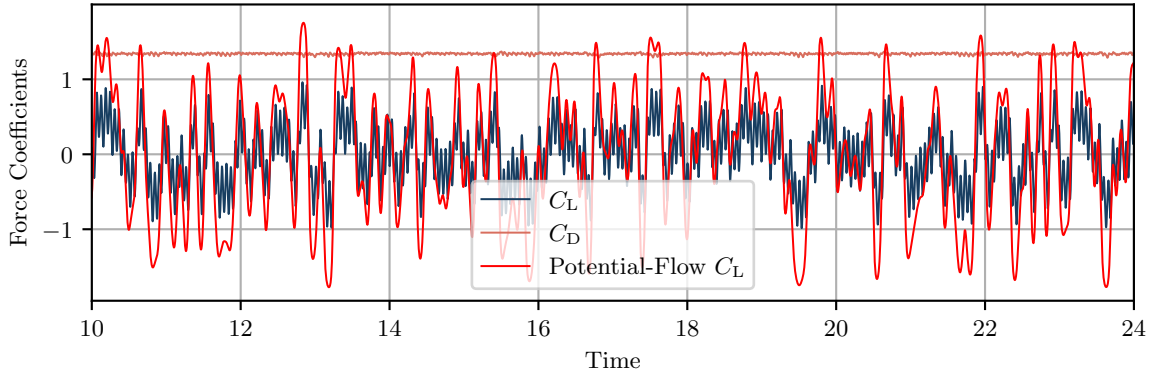


Figure 18: 2D flow around a rotating circular cylinder: force coefficients. Lift and drag coefficients from the OpenFOAM simulation over the time window used to build the dataset, together with the angular velocity of the cylinder. To match the scale of the lift and drag coefficients, the angular velocity is reported as the equivalent potential-flow C_L .

3.3 Test case 2B: 2D flow around a rotating circular cylinder

Following the test on the surrogate modeling of the flow around a static circular cylinder, the IC-LDNet is further assessed on a more challenging extension of the same problem, where the cylinder is rotating with an angular velocity γ that varies randomly in time. To build the dataset, a similar OpenFOAM setup to the one described in Section 3.2 is considered, with the only difference being that a moving-wall boundary condition is applied to the cylinder wall, as summarized in Figure 17. Here, the region of interest only covers the $[-5, 15] \times [-5, 5]$ domain (with a resolution of 400×200) to reduce the memory footprint of the stored snapshots. The simulation is run for 24 seconds of physical time, with an adaptive time step to maintain a Courant number below 1.0, and the snapshots are stored every 2×10^{-3} seconds. To develop the flow, the cylinder is initially stationary for five seconds, and then starts rotating at a random angular velocity. This kinematic profile is generated by sampling a zero-mean Gaussian process (squared-exponential kernel, correlation length 0.05 s), subsequently bounded within ± 60 rad/s using a tanh filter.

To assemble the final dataset, only the snapshots corresponding to the last 14 seconds of simulation time are considered, resulting in a total of 7 000 snapshots. Specifically, Figure 18 summarizes the evolution of the force coefficients over time, alongside the corresponding angular velocity of the cylinder, representing the input $\mathbf{u}(t)$. The simulation is partitioned into 70 non-overlapping sequences of 100

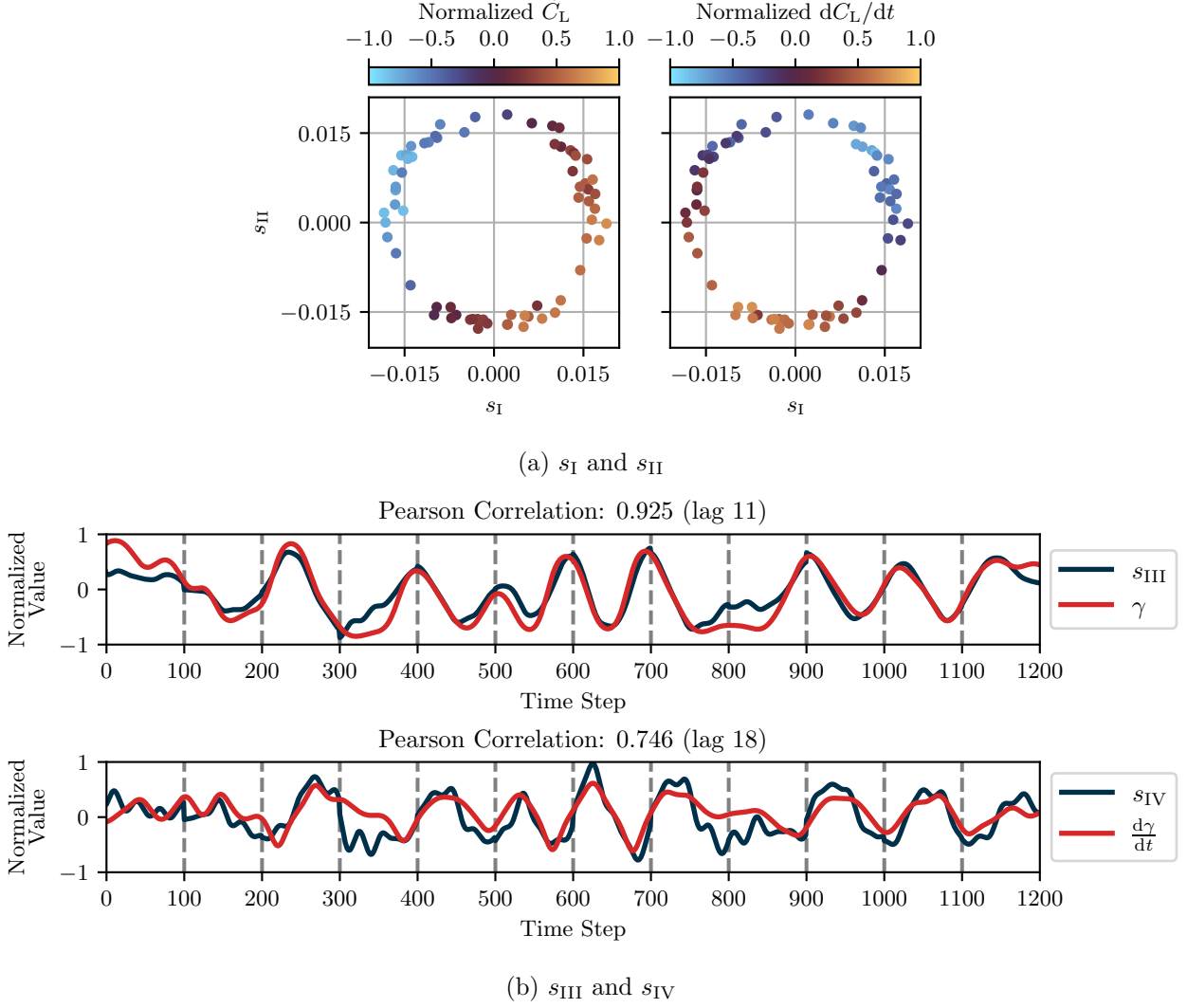


Figure 19: Rotating-cylinder test case: latent-space PCA (Meta-IC-LDNet). The principal components $s_I, \dots, s_{IV}$ are computed on the latent trajectories of the training, validation, and test sets combined. (a) Initial codes, colored by $\tilde{C}_L$ (i.e., the lift coefficient minus its sliding average over 30 time steps) and by dC_L/dt . (b) Third and fourth principal components along twelve consecutive sequences (separated by dashed lines), compared with the angular velocity γ and its time derivative, delayed by the lag (in time steps) that maximizes the Pearson correlation. The colors in (a) and the curves in (b) are min-max normalized to $[-1, 1]$.

snapshots each, and the resulting dataset is randomly split into training, validation, and test sets, in a ratio of 80%, 10%, and 10%, respectively.

An instance of the Meta-IC-LDNet, whose hyperparameters are summarized in Table 5 (see Appendix A), is trained on the rotating-cylinder dataset. Unlike Test Case 2A, the dataset is not downsized to a fixed spatial resolution, but the loss function is computed on 1250 randomly sampled points from the region of interest, instead of the full-resolution grid (i.e., 80 000 points). Every snapshot within each time series is spatially subsampled independently. At test time, the approximate solution is instead computed at 40 000 randomly sampled points, shared across samples and time steps. Moreover, as before, the curriculum-learning strategy described in Section 2.4.4 is applied, gradually exposing the model to the entire simulations.

In line with the static-cylinder findings, the latent space learned by the Meta-IC-LDNet exhibits a topologically consistent arrangement of the initial codes, and a physically meaningful relationship is

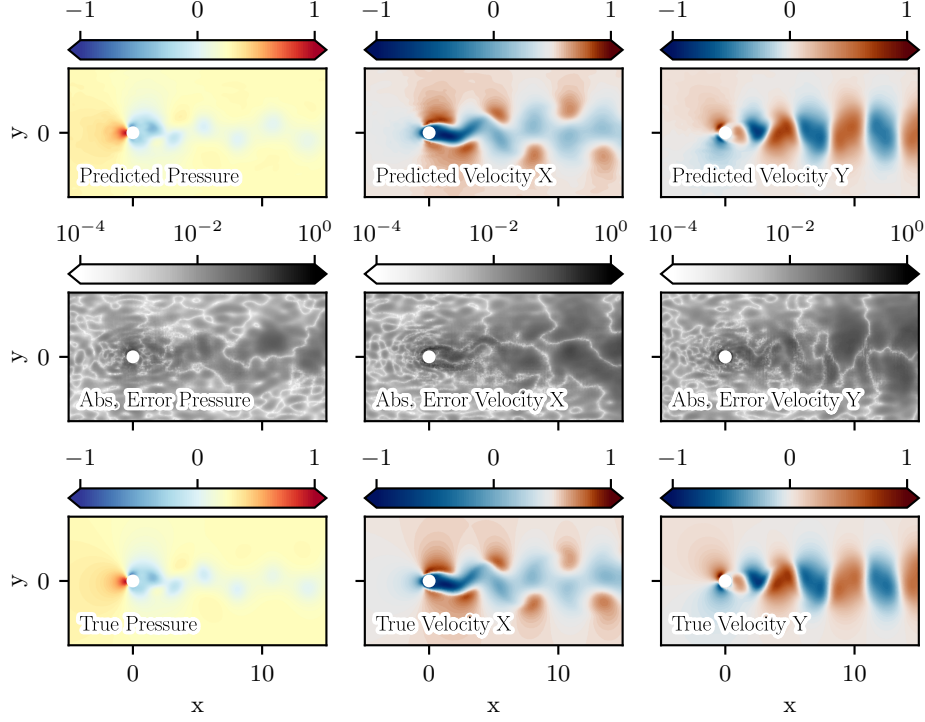


Figure 20: Rotating-cylinder test case: final-time normalized test-set fields (Meta-IC-LDNet). Predicted (top), absolute error (middle), and true (bottom) normalized fields at the last time step of the first test-set sample, arranged as in Figure 11.

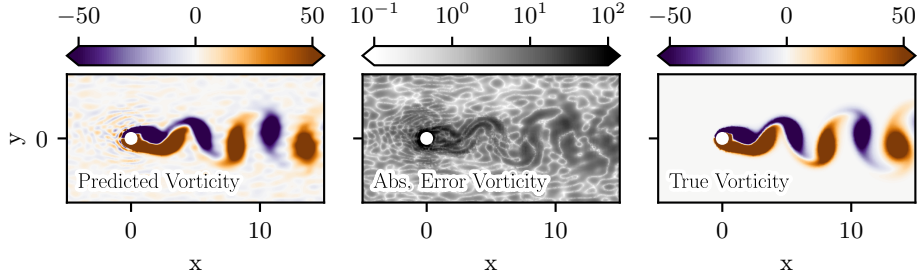


Figure 21: Rotating-cylinder test case: final-time test-set vorticity (Meta-IC-LDNet). Predicted (left), absolute error (center), and true (right) vorticity at the last time step of the first test-set sample, computed as in Figure 12.

observed between the latent codes and the flow features. To better analyze the structure of the latent space, whose dimension is here $d_s = 4$, Principal Component Analysis (PCA) is performed on the latent trajectories, considering the combined training, validation, and test sets. The first two principal components of the initial codes are shown in Figure 19a, while Figure 19b depicts a subset of latent trajectories projected onto the third and fourth principal components. The former plot suggests that the first two principal components are related to the high-frequency periodic shedding, similar to the observations in Figure 14. In particular, $\tilde{C}_L$ represents the difference between the instantaneous lift coefficient and a sliding average of C_L over a time window of 30 time steps (i.e., approximately one shedding period), highlighting high-frequency variations. Additionally, Figure 19b reveals a strong correlation of the third and fourth principal components with the cylinder angular velocity and its time derivative, respectively, provided that a certain time shift is considered (i.e., the one that maximizes the correlation). This suggests that the latent space contains past information on the cylinder rotation. Hence, the four-dimensional latent space encodes both the high-frequency von Kármán shedding and

the low-frequency Magnus effect [57].

As observed in previous tests, this well-organized latent space allows for good generalization and an accurate prediction of the flow evolution. For example, Figures 20 and 21 show the true and predicted flow fields and vorticity contours at the last time step of the first test-set example, along with the corresponding absolute error. As before, the vorticity is computed from the predicted velocity field through automatic differentiation. Although the error is noticeably larger than the one observed in the static-cylinder case, the model is still able to reconstruct the flow field with a good degree of accuracy, despite the more challenging nature of the problem. Overall, on the test set, the pressure is predicted with an average NRMSE of 7.5×10^{-3} , while the x and y velocity predictions are characterized by an average NRMSE of 1.2×10^{-2} and 1.3×10^{-2} , respectively.

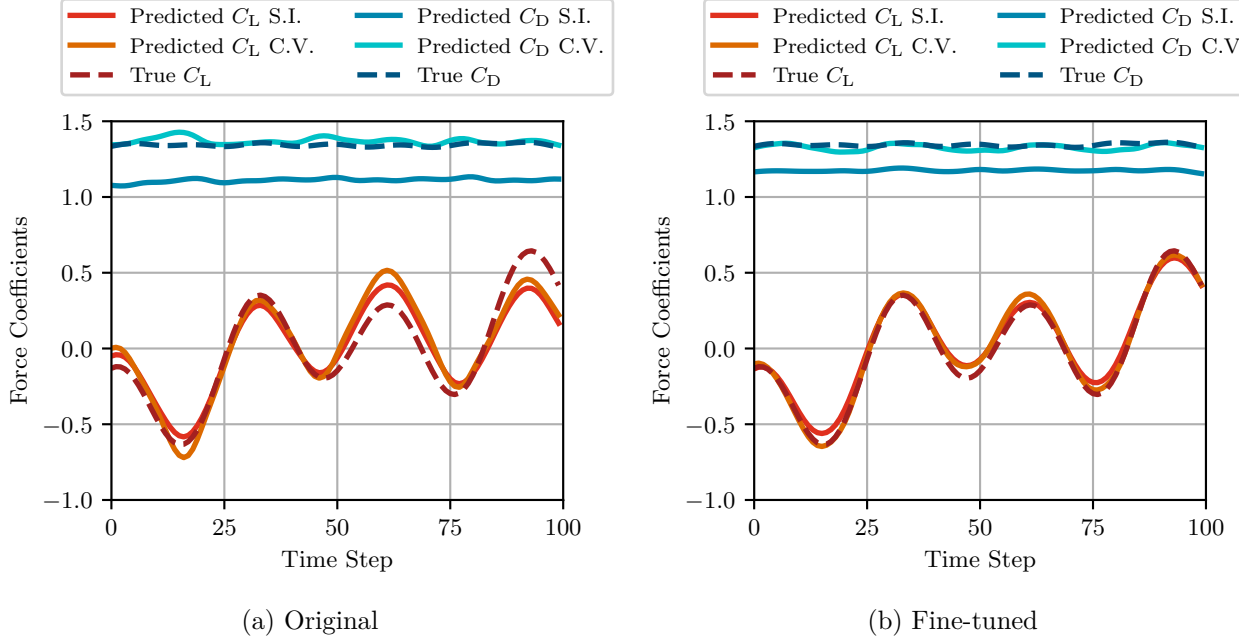


Figure 22: Rotating-cylinder test case: force coefficients (Meta-IC-LDNet). Lift and drag coefficients of the first test-set sample, computed from the predicted fields via surface integration (S.I.) on the cylinder wall and via momentum balance on a control volume (C.V.) around the cylinder, and compared with the OpenFOAM reference (dashed lines). (a) Original model. (b) Model fine-tuned on a denser grid of points, including an O-grid around the cylinder.

Finally, the approximate velocity and pressure fields are used to compute the force coefficients, which are compared to the ones computed in OpenFOAM. Figure 22 shows the time evolution of the lift and drag coefficients for the first test-set example. Specifically, two methods for computing the force coefficients are considered: surface integration (i.e., S.I.) of the pressure and viscous forces on the cylinder wall, and momentum balance on a fluid control volume (i.e., C.V.) surrounding the cylinder. Interestingly, while both methods provide similarly satisfactory results for the lift coefficient, the control-volume approach is highly accurate for the drag coefficient, which is underestimated by the surface-integration method, possibly due to the training-set resolution near the cylinder wall. Indeed, in Figure 22b, when considering an improved model fine-tuned for an additional 24 hours on a denser grid, the accuracy of both lift and drag coefficients considerably increases. Specifically, the fine-tuning grid consists of 40 000 randomly sampled points from the RoI (see Figure 17), as well as 14 500 additional randomly sampled points from a 58×500 (radial $\times$ angular) O-grid covering the annular region $0.48 \leq r \leq 1$, where r denotes the distance from the cylinder center. Note that the O-grid slightly extends inside the cylinder to better capture the boundary.

Accurate surrogate predictions of the force coefficients pave the way for the application of the Meta-IC-LDNet to optimal control (OC) problems, which require many evaluations of the flow field. As

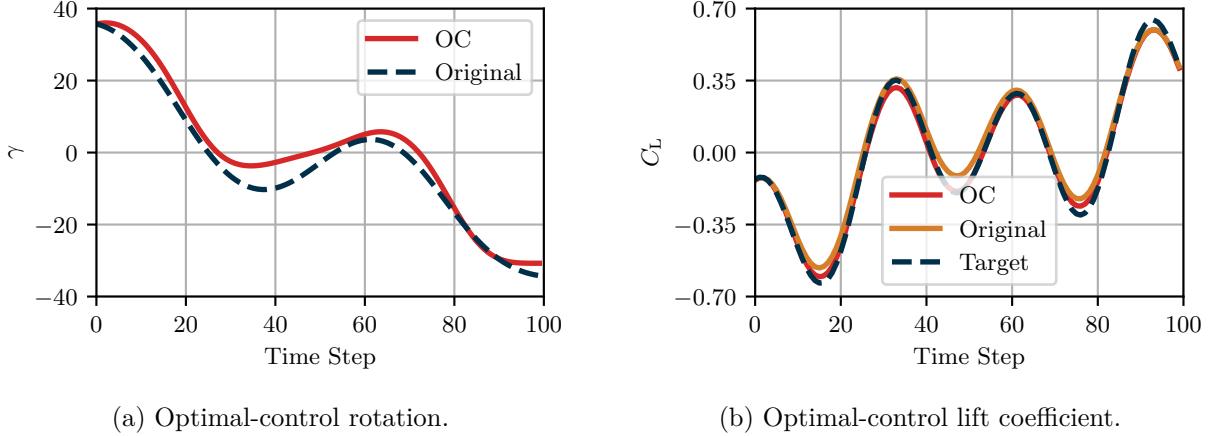


Figure 23: Rotating-cylinder test case: optimal control (Meta-IC-LDNet). The fine-tuned model is used to recover the angular velocity that reproduces the lift coefficient of the first test-set sample. (a) Angular velocity obtained by solving the optimal-control (OC) problem, compared with the original one. (b) Lift coefficient predicted by the surrogate model with the OC and with the original angular velocity, compared with the OpenFOAM target.

a proof of concept, the fine-tuned model is tasked with recovering the angular velocity that produces the lift-coefficient evolution of the first test-set example (i.e., the one in Figure 22) by minimizing the MSE loss between the predicted (via the S.I. method) and target C_L over the entire time series. The optimization is performed in 2×10^3 Adam iterations, using a learning rate of 10^{-3} and L_2 regularization. In Figure 23a, the original and recovered angular velocities are compared. Although the error between the two signals is not negligible, the optimal-control angular velocity follows the same trend as the original. Furthermore, in Figure 23b we compare the target lift coefficient (i.e., from OpenFOAM) to the one predicted by the surrogate model using the original rotation and the one using the just computed optimal-control rotation. Specifically, Figure 23b shows that the discrepancy between the original and reconstructed signal in Figure 23a does not originate from the solution of the optimal-control problem, since the OC C_L accurately follows the target. Crucially, the fast inference of the neural surrogate allows for solving the optimal-control problem in approximately 16 seconds, highlighting the potential of the proposed framework for real-world many-query applications.

3.4 Test case 3: nonlinear solid dynamics of a 3D cantilever beam

The final test case considered in this work deals with the nonlinear solid dynamics of a 3D cantilever beam. As summarized in Figure 24, the beam is clamped at one end and loads are applied on the free extremity. Specifically, the instantaneous intensities of these loads are randomly sampled using independent zero-mean Gaussian processes. All processes share a temporal correlation length of $l = 1.0$, with a standard deviation set to $\sigma_f = 0.05$ for the axial and shear tractions, and $\sigma_q = 500$ for the twisting moment intensity.

The beam has a length of $L = 0.1$ and a square cross-section with side length $a = 0.01$, and is made of a hyperelastic material modeled by a compressible Neo-Hookean constitutive law [58]. Specifically, the strain energy density Ψ is described as:

$$\Psi = \frac{\mu}{2} \left(J^{-2/3} I_1 - 3 \right) + \frac{\lambda}{4} \left((J - 1)^2 + (\ln J)^2 \right), \quad (16)$$

where λ is the first Lamé parameter, μ is the shear modulus (i.e., the second Lamé parameter), $J = \det(\mathbf{F})$ with $\mathbf{F}$ being the deformation gradient, and $I_1 = \text{tr}(\mathbf{C})$ with $\mathbf{C} = \mathbf{F}^T \mathbf{F}$ being the right Cauchy-Green tensor. The density of the material is $\rho = 10^3$ and the damping coefficient is $\eta = 10^2$. To introduce variability in the material properties, the shear modulus is sampled from a uniform

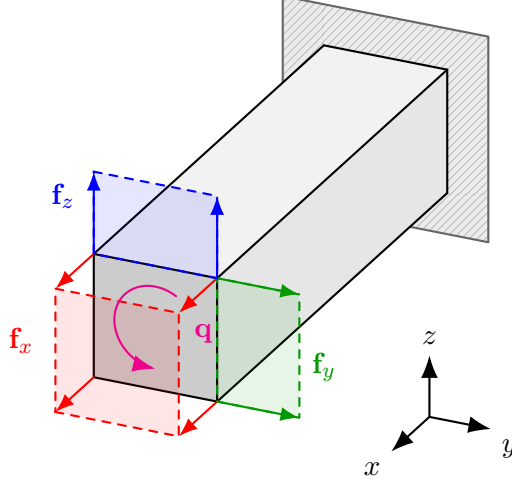


Figure 24: Nonlinear solid dynamics of a 3D cantilever beam: domain and boundary conditions (not to scale). The beam is clamped at $x = 0$, while axial ($\mathbf{f}_x$) and shear ($\mathbf{f}_y, \mathbf{f}_z$) tractions and a twisting moment ($\mathbf{q}$) are applied on the free end.

distribution in the range $[25, 250]$, while the first Lamé parameter is proportionally updated as $\lambda = 2.5\mu$. Note that we are considering non-dimensional physical quantities.

The dynamic evolution of the system is governed by the Principle of Virtual Work. Specifically, expressed in a total Lagrangian framework over the reference configuration Ω , the problem consists in finding a kinematically admissible displacement field $\mathbf{d}$ such that, for any valid test function $\mathbf{v}$, the following equation holds:

$$\int_{\Omega} \rho \ddot{\mathbf{d}} \cdot \mathbf{v} \, d\Omega + \int_{\Omega} \left(\mathbf{P} + \frac{\eta}{2} (\mathbf{F}\dot{\mathbf{C}}) \right) : \nabla_0 \mathbf{v} \, d\Omega = \int_{\Gamma_N} (\mathbf{f}_t + q_t \boldsymbol{\tau}) \cdot \mathbf{v} \, dS, \quad (17)$$

where $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$ is the first Piola-Kirchhoff stress tensor, ∇_0 denotes the gradient operator with respect to the material coordinates, $\ddot{\mathbf{d}}$ is the acceleration field, and $\dot{\mathbf{C}}$ is the rate of the right Cauchy-Green tensor accounting for viscous dissipation. The external work on the Neumann boundary Γ_N (i.e., the free extremity) depends on the applied surface tractions $\mathbf{f}_t$ and the twisting moment controlled by q_t through the lever-arm vector $\boldsymbol{\tau} = [0, Z, -Y]^T$.

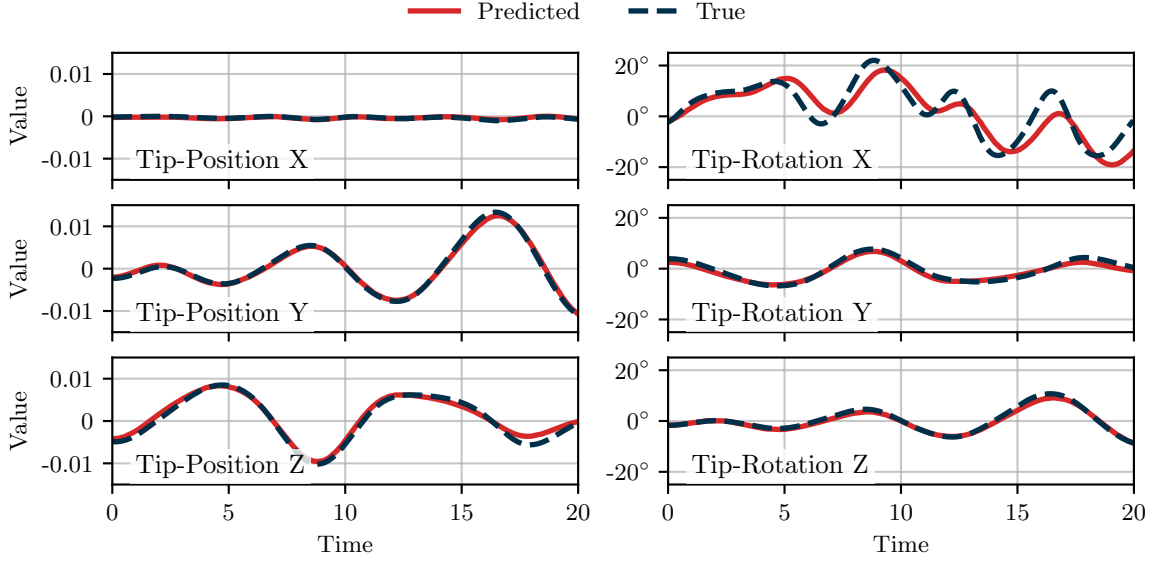
The spatial domain is discretized using a structured mesh of first-order tetrahedral elements with a characteristic side length of 2.5×10^{-3} . Time integration is performed using a fully implicit finite difference scheme, approximating the acceleration field and the kinematic rates with backward differences. The resulting fully discrete nonlinear system is solved at each time step t_n by means of the Newton-Raphson method, implemented within the FEniCS finite element framework [59, 60]. The simulation spans a total of 2500 time steps with a constant time increment of $\Delta t = 2 \times 10^{-2}$. Initially, at $t = 0$, the system starts from rest, assuming zero displacement and velocity fields.

To compose the dataset, 500 unique configurations of load histories and material properties are simulated, and the nodal displacements are recorded at each time step. Then, for each simulation, the first 500 time steps are discarded, and the remaining 2000 are split into two non-overlapping sequences of 1000 time steps each, resulting in a total of 1000 sequences. The input $\mathbf{u}(t)$ contains both the constant shear modulus and the variable intensities of the external loads. The final dataset is randomly partitioned into training, validation, and test sets with a split of 80%, 10%, and 10%, respectively.

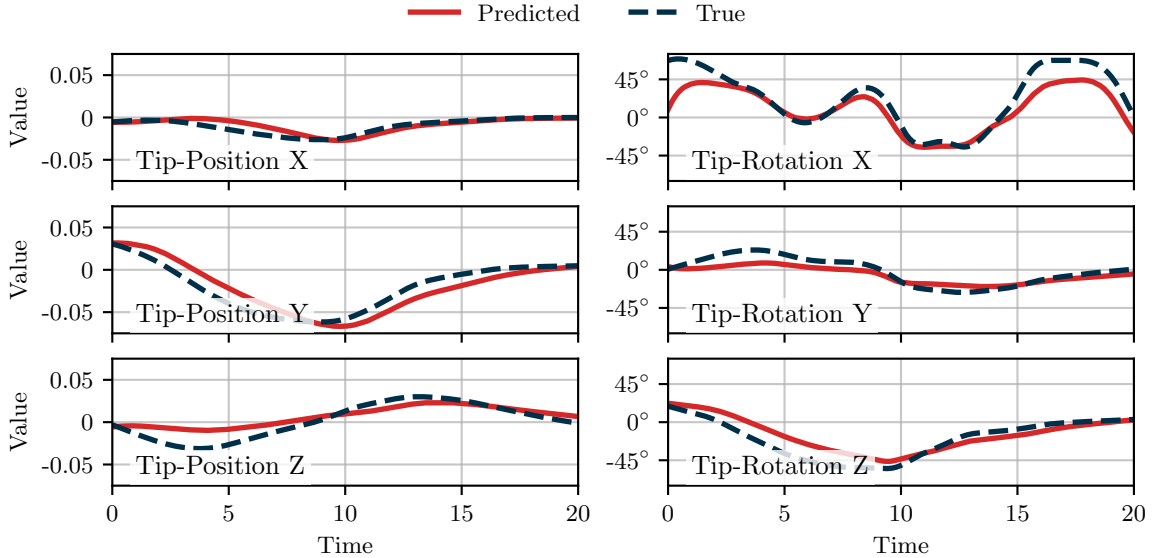
The Meta-IC-LDNet is trained as a surrogate for the FEniCS solver on the nonlinear beam dynamics. Unlike the previous test cases, the model is trained on a workstation equipped with an NVIDIA GeForce RTX 5090 GPU and an Intel Core Ultra 9 285K CPU with 64 GB of RAM. In this case, $\mathcal{NN}_{\text{dyn}}$ and $\mathcal{NN}_{\text{rec}}$ are both FCNNs with SiLU activation function, and the latent space has a dimension of $d_s = 12$ (refer to Table 6 for the full set of hyperparameters). To alleviate the memory burden, the original 1025 points of the mesh are randomly subsampled to 500 points during training, while the full set of points is used during inference. Finally, the second regularization strategy described in Section 2.4.4 is

adopted. Specifically, three additional inner-loop optimizations are performed at equidistant times within the sequence, and the MSE between the resulting codes and the corresponding state in the latent trajectory is added to the loss function.

The performance and generalization of the trained model are assessed on the test set. Specifically, Figure 25 presents the tip dynamics of the best and worst predictions within the test set. The position of the tip is extracted directly from the model outputs (i.e., the displacement components), while the beam-section rotation at the free extremity is computed during post-processing using the Kabsch algorithm [61]. In particular, Figure 25a shows that the model can accurately predict the beam dynamics with only a slight discrepancy in torsion. By contrast, in the worst-case scenario (i.e., Figure 25b), although the surrogate predictions exhibit a noticeable error, the overall dynamics still preserve the correct physical trends. Note that the samples where the model underperforms are



(a) Best test-set prediction: tip dynamics.



(b) Worst test-set prediction: tip dynamics.

Figure 25: Nonlinear-beam test case: inference on the test set (Meta-IC-LDNet). Predicted and true displacement at the center of the free-end cross-section (left) and rotation angles of the cross-section about the x , y , and z axes, computed with the Kabsch algorithm (right), for the test-set samples with the lowest (a) and highest (b) NRMSE.

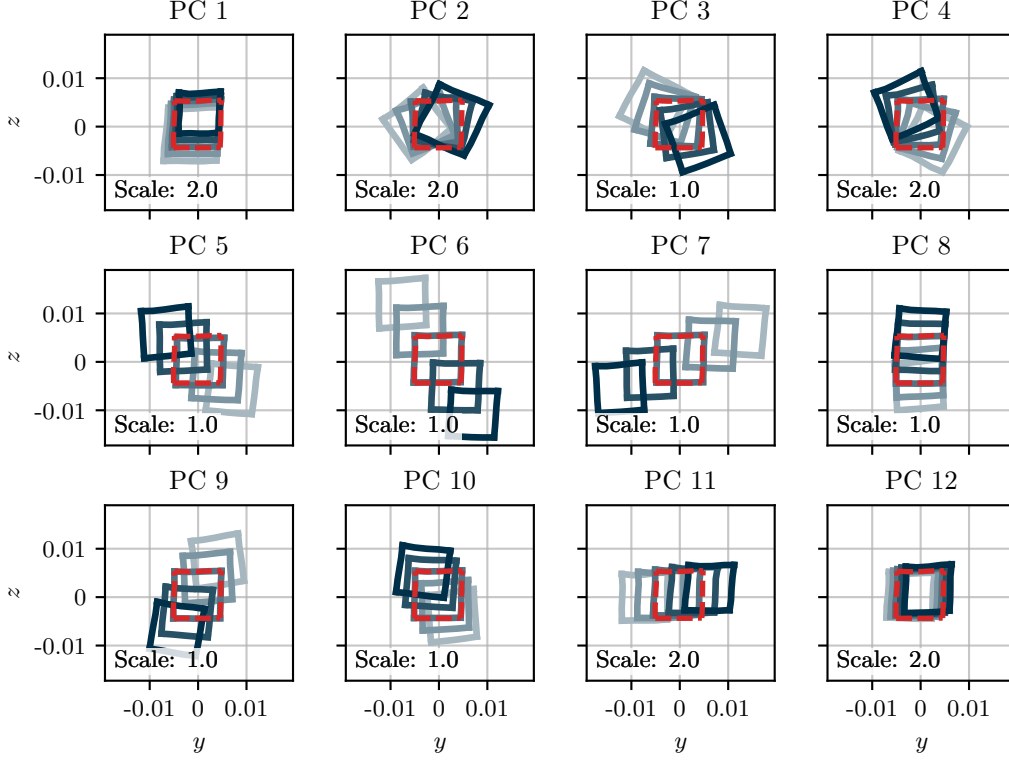


Figure 26: Nonlinear-beam test case: effect of latent-space principal components (Meta-IC-LDNet). Each panel shows the free-end cross-section in the yz plane, decoded from latent states where a single principal component (PC) varies between its 1st and 99th percentiles over all latent states (from light gray to dark blue), while the remaining components are fixed at their mean values. The red dashed square corresponds to all components at their mean values. Displacements are amplified by the indicated scale factor for visualization.

characterized by very large displacements and represent outliers within the data distribution. Hence, a larger error should be expected for these instances. Overall, the test-set NRMSE confirms that this is the most challenging benchmark among the ones presented in this work. Specifically, the x , y , and z displacement components are predicted with an average error of 4.8×10^{-2} , 4.7×10^{-2} , and 4.5×10^{-2} , respectively.

Finally, to investigate the latent space structure, the latent states along the trajectories are projected onto their principal components. Subsequently, the effect of each principal component is assessed by varying only one of them at a time while keeping the other components equal to their per-component means. These codes are then decoded to recover the corresponding beam deformation, as shown in Figure 26, where the deformed free extremity is depicted for selected intermediate interpolation steps. Interestingly, also in this case, the latent space learned by the Meta-IC-LDNet is physically interpretable. Indeed, the variation of each latent-state principal component mainly corresponds to a specific mechanical response of the beam, such as stretching, twisting, bending, or flexural-torsional coupling.

4 Conclusions

The Latent Dynamics Network learns the solution operator of time-dependent PDEs, but its original formulation assumes a fixed initial condition, limiting its applicability to real-world settings where the system evolves from varying, unseen initial states. This work addresses this limitation by formulating latent-state initialization as an encoder-free adaptation problem, and compares two strategies for solving it, auto-decoding and meta-learning. The comparison reveals that the effect of meta-learning

extends beyond inference speed: it regularizes the latent-state identification problem, turning an ill-conditioned, multi-modal optimization landscape into a smooth, well-conditioned one. Additionally, meta-learning is consistently associated with latent representations whose topology reflects that of the underlying physical space. More broadly, these results suggest that latent representation, state initialization, and latent evolution should not necessarily be designed as separate components. Training the representation for rapid state inference may itself provide a useful inductive bias for learning dynamically meaningful reduced coordinates.

Because the proposed IC-LDNet remains entirely encoder-free, it preserves the resolution independence of the original framework and integrates seamlessly with neural-field-based decoders, enabling training on highly subsampled spatial domains while accurately recovering high-fidelity fields at inference. Its versatility was demonstrated across distinct physical domains, from advection-diffusion-reaction and fluid dynamics to nonlinear solid mechanics. Moreover, its rapid inference was further leveraged for an optimal-control problem in fluid dynamics, a many-query task that would be computationally challenging with traditional numerical methods.

Some limitations of this work should be acknowledged. The training of the Meta-IC-LDNet can be sensitive to the choice of hyperparameters and, in some applications, it requires a regularization strategy, chosen on a case-by-case basis, to reach a stable, well-conditioned latent space. Moreover, as observed for the nonlinear-beam test case, predictive accuracy degrades when dealing with outliers with respect to the training distribution. Additionally, in the static- and rotating-cylinder test cases, the validation and test sets contain only 6 and 7 samples each, respectively, which limits the statistical significance of the reported generalization metrics. Finally, the proposed approach was not benchmarked against alternative surrogate models from the literature, as this work focuses on the comparison between encoder-free initialization strategies within the LDNet framework.

Future work will explore the application of the Meta-IC-LDNet to further optimization and control problems, a natural direction given its inference speed. It will also be worth investigating whether explicit physical or topological constraints on the latent space can further facilitate training and enforce desired representation properties, building on the inductive bias already observed to emerge from meta-learning itself.

Acknowledgements

The present research has received support from the project FIS, MUR, Italy 2025-2028, Project code: FIS-2023-02228, CUP: D53C24005440001, “SYNERGIZE: Synergizing Numerical Methods and Machine Learning for a new generation of computational models”. We acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy). We acknowledge the grant Dipartimento di Eccellenza 2023-2027, MUR, Italy. We acknowledge “INdAM - GNCS Project” code CUP E53C25002010001. The authors of this work are members of GNCS, “Gruppo Nazionale per il Calcolo Scientifico” (National Group for Scientific Computing) of INdAM (Istituto Nazionale di Alta Matematica), Italy.

References

- [1] Alfio Quarteroni. *Numerical Models for Differential Problems*. 2nd. MS&A - Modeling, Simulation and Applications 8. Milano: Springer, 2014. 658 pp. ISBN: 978-88-470-5521-6 978-88-470-5522-3. DOI: 10.1007/978-88-470-5522-3.
- [2] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Philadelphia: Society for Industrial and Applied Mathematics, 2007. ISBN: 978-0-89871-629-0.
- [3] Klaus-Jürgen Bathe. *Finite element procedures*. Englewood Cliffs, NJ: Prentice Hall, 1996. 1037 pp. ISBN: 978-0-13-301458-7.

- [4] Robert Eymard, Thierry Gallouët, and Raphaële Herbin. “Finite volume methods”. In: *Handbook of Numerical Analysis*. Vol. 7. Elsevier, 2000, pp. 713–1018. ISBN: 978-0-444-50350-3. DOI: 10.1016/S1570-8659(00)07005-8.
- [5] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced Basis Methods for Partial Differential Equations*. Vol. 92. UNITEXT. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-15430-5 978-3-319-15431-2. DOI: 10.1007/978-3-319-15431-2.
- [6] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-22469-5 978-3-319-22470-1. DOI: 10.1007/978-3-319-22470-1.
- [7] Nils Thuerey et al. “Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows”. In: *AIAA Journal* 58.1 (2020), pp. 25–36. ISSN: 0001-1452. DOI: 10.2514/1.J058291.
- [8] George Em Karniadakis et al. “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6 (June 2021), pp. 422–440. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00314-5.
- [9] Alfio Quarteroni, Paola Gervasio, and Francesco Regazzoni. “Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning”. In: *Mathematical Models and Methods in Applied Sciences* 35.4 (Apr. 2025), pp. 905–1071. ISSN: 0218-2025. DOI: 10.1142/S0218202525500125.
- [10] Lu Lu et al. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature Machine Intelligence* 3.3 (Mar. 2021), pp. 218–229. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00302-5.
- [11] Nikola Kovachki et al. “Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs”. In: *Journal of Machine Learning Research* 24.89 (2023), pp. 1–97. ISSN: 1533-7928. URL: <http://jmlr.org/papers/v24/21-1524.html>.
- [12] Zongyi Li et al. “Fourier Neural Operator for Parametric Partial Differential Equations”. In: *International Conference on Learning Representations*. International Conference on Learning Representations. 2021. URL: https://openreview.net/forum?id=c8P9NQVtmn0&spm=a2ty_o01.29997173.0.0.34fbc921RYdZHi.
- [13] Zongyi Li et al. “Fourier Neural Operator with Learned Deformations for PDEs on General Geometries”. In: *Journal of Machine Learning Research* 24.388 (2023), pp. 1–26. ISSN: 1533-7928. URL: <http://jmlr.org/papers/v24/23-0064.html>.
- [14] Zongyi Li et al. “Physics-Informed Neural Operator for Learning Partial Differential Equations”. In: *ACM / IMS Journal of Data Science* 1.3 (Sept. 30, 2024), pp. 1–27. ISSN: 2831-3194. DOI: 10.1145/3648506.
- [15] Louis Serrano et al. “Operator Learning with Neural Fields: Tackling PDEs on General Geometries”. In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 2023, pp. 70581–70611. DOI: 10.52202/075280-3093.
- [16] Sifan Wang et al. “CViT: Continuous Vision Transformer for Operator Learning”. In: *International Conference on Learning Representations*. Vol. 2025. May 1, 2025, pp. 24004–24039. URL: https://proceedings.iclr.cc/paper_files/paper/2025/hash/3bf4b55960aaa23553cd2a6bdc6e1b57-Abstract-Conference.html.
- [17] Yiheng Xie et al. “Neural Fields in Visual Computing and Beyond”. In: *Computer Graphics Forum* 41.2 (2022), pp. 641–676. ISSN: 1467-8659. DOI: 10.1111/cgf.14505.
- [18] Ben Mildenhall et al. “NeRF: representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (Dec. 17, 2021), pp. 99–106. ISSN: 0001-0782. DOI: 10.1145/3503250.

- [19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (Feb. 1, 2019), pp. 686–707. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2018.10.045.
- [20] Weinan E and Bing Yu. “The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems”. In: *Communications in Mathematics and Statistics* 6.1 (Mar. 1, 2018), pp. 1–12. ISSN: 2194-671X. DOI: 10.1007/s40304-018-0127-z.
- [21] Yuan Yin et al. “Continuous PDE Dynamics Forecasting with Implicit Neural Representations”. In: *International Conference on Learning Representations*. International Conference on Learning Representations. 2023. URL: <https://openreview.net/forum?id=B73niNjbPs>.
- [22] Louis Serrano et al. *INFINITY: Neural Field Modeling for Reynolds-Averaged Navier-Stokes Equations*. July 25, 2023. DOI: 10.48550/arXiv.2307.13538. arXiv: 2307.13538[cs].
- [23] Tianshu Wen, Kookjin Lee, and Youngsoo Choi. *Reduced-order modeling for parameterized PDEs via implicit neural representations*. Nov. 28, 2023. DOI: 10.48550/arXiv.2311.16410. arXiv: 2311.16410[math].
- [24] Louis Serrano et al. “AROMA: Preserving Spatial Structure for Latent PDE Modeling with Local Neural Fields”. In: *Advances in Neural Information Processing Systems*. Vol. 37. Curran Associates, Inc., 2024, pp. 13489–13521. DOI: 10.52202/079017-0431.
- [25] Honghui Wang, Shiji Song, and Gao Huang. “GridMix: Exploring Spatial Modulation for Neural Fields in PDE Modeling”. In: *International Conference on Learning Representations*. May 1, 2025, pp. 92717–92734. URL: https://proceedings.iclr.cc/paper_files/paper/2025/hash/e7938ede51225b490bb69f7b361a9259-Abstract-Conference.html.
- [26] Giovanni Catalani et al. “Neural fields for rapid aircraft aerodynamics simulations”. In: *Scientific Reports* 14.1 (Oct. 26, 2024), p. 25496. ISSN: 2045-2322. DOI: 10.1038/s41598-024-76983-w.
- [27] Giovanni Catalani et al. *Geometry aware inference of steady state PDEs using Equivariant Neural Fields representations*. Sept. 26, 2025. DOI: 10.48550/arXiv.2504.18591. arXiv: 2504.18591[cs.LG].
- [28] Francesco Regazzoni et al. “Learning the intrinsic dynamics of spatio-temporal processes through Latent Dynamics Networks”. In: *Nature Communications* 15.1 (Feb. 28, 2024), p. 1834. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45323-x.
- [29] Matteo Salvador and Alison Lesley Marsden. “Liquid Fourier Latent Dynamics Networks for fast GPU-based numerical simulations in computational cardiology”. In: *Computers in Biology and Medicine* 200 (Jan. 1, 2026), p. 111355. ISSN: 0010-4825. DOI: 10.1016/j.combiomed.2025.111355.
- [30] Edoardo Centofanti et al. *A Neural Latent Dynamics Approach for Solving Inverse Problems in Cardiac Electrophysiology*. May 2, 2026. DOI: 10.48550/arXiv.2605.01463. arXiv: 2605.01463[math.NA].
- [31] Phillip Si et al. “Toward AI-driven digital twins for metropolitan floods: A conditional latent dynamics network surrogate of the shallow water equations”. In: *Journal of Hydrology* 680 (Nov. 2026), p. 136461. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2026.136461.
- [32] Lorenzo Tomada, Federico Pichi, and Gianluigi Rozza. “Latent dynamics graph convolutional networks for model order reduction of parameterized time-dependent PDEs”. In: *Journal of Computational Physics* 564 (Nov. 1, 2026), p. 115150. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2026.115150.
- [33] Xinlei Lin and Dunhui Xiao. *Parametric Taylor series based latent dynamics identification neural networks*. Oct. 5, 2024. DOI: 10.48550/arXiv.2410.04193. arXiv: 2410.04193[cs.LG].
- [34] Zhangyong Liang, Jihong Zhu, and Huanhuan Gao. *Parameterized Temporal Extrapolation of PDEs via Disentangled Latent Dynamics Manifold Fusion*. Sept. 29, 2026. DOI: 10.48550/arXiv.2603.12676. arXiv: 2603.12676[cs.LG].

- [35] Pengpeng Xiao, Phillip Si, and Peng Chen. “LD-EnSF: Synergizing Latent Dynamics with Ensemble Score Filters for Fast Data Assimilation with Sparse Observations”. In: *International Conference on Learning Representations*. Apr. 20, 2026, pp. 152461–152491. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/f7883c88d25e8cd0f05f06c92375febc-Abstract-Conference.html.
- [36] Phillip Si and Peng Chen. *LEVDA: Latent Ensemble Variational Data Assimilation via Differentiable Dynamics*. Feb. 23, 2026. DOI: 10.48550/arXiv.2602.19406. arXiv: 2602.19406[cs.LG].
- [37] Honghui Wang et al. “Advancing Generalization in PINNs Through Latent-Space Representations”. In: *IEEE Transactions on Neural Networks and Learning Systems* 36.12 (Dec. 2025), pp. 20243–20257. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2025.3598617.
- [38] Zhanhong Ye et al. “Meta-Auto-Decoder: a Meta-Learning-Based Reduced Order Model for Solving Parametric Partial Differential Equations”. In: *Communications on Applied Mathematics and Computation* 6.2 (June 1, 2024), pp. 1096–1130. ISSN: 2661-8893. DOI: 10.1007/s42967-023-00293-7.
- [39] Ricky T. Q. Chen et al. “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html>.
- [40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [41] Zewen Li et al. “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (Dec. 2022), pp. 6999–7019. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2021.3084827.
- [42] William L. Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, Sept. 16, 2020. 161 pp. ISBN: 978-1-68173-964-9.
- [43] Shufeng Tan and Michael L. Mayrovouniotis. “Reducing data dimensionality through optimizing neural network inputs”. In: *AIChE Journal* 41.6 (1995), pp. 1471–1480. ISSN: 1547-5905. DOI: 10.1002/aic.690410612.
- [44] Piotr Bojanowski et al. “Optimizing the Latent Space of Generative Networks”. In: *Proceedings of the 35th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 3, 2018, pp. 600–609. URL: <https://proceedings.mlr.press/v80/bojanowski18a.html>.
- [45] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2019, pp. 165–174. DOI: 10.1109/CVPR.2019.00025.
- [46] Timothy Hospedales et al. “Meta-Learning in Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (Sept. 2022), pp. 5149–5169. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3079209.
- [47] Luisa Zintgraf et al. “Fast Context Adaptation via Meta-Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 7693–7702. URL: <https://proceedings.mlr.press/v97/zintgraf19a.html>.
- [48] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 30, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs.LG].
- [49] Yoshua Bengio et al. “Curriculum learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML ’09. New York, NY, USA: Association for Computing Machinery, June 14, 2009, pp. 41–48. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553380.

- [50] Matteo Turisini, Mirko Cestari, and Giorgio Amati. “LEONARDO: A Pan-European Pre-Exascale Supercomputer for HPC and AI applications”. In: *Journal of large-scale research facilities JLSRF* 9.1 (Jan. 15, 2024). ISSN: 2364-091X. DOI: 10.17815/jlsrf-8-186.
- [51] Stefan Elfving, Eiji Uchibe, and Kenji Doya. “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”. In: *Neural Networks* 107 (Nov. 2018), pp. 3–11. ISSN: 08936080. DOI: 10.1016/j.neunet.2017.12.012.
- [52] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. “Optimization Methods for Large-Scale Machine Learning”. In: *SIAM Review* 60.2 (Jan. 2018), pp. 223–311. ISSN: 0036-1445. DOI: 10.1137/16M1080173.
- [53] John H. Lienhard. *Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders*. Technical Extension Service, Washington State University, 1966.
- [54] Ethan Perez et al. “FiLM: Visual Reasoning with a General Conditioning Layer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 29, 2018). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v32i1.11671.
- [55] Vincent Sitzmann et al. “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473.
- [56] Pan Du et al. “Conditional neural field latent diffusion model for generating spatiotemporal turbulence”. In: *Nature Communications* 15.1 (Nov. 29, 2024), p. 10416. ISSN: 2041-1723. DOI: 10.1038/s41467-024-54712-1.
- [57] John David Anderson and Christopher P. Cadou. *Fundamentals of aerodynamics*. 7th. McGraw Hill series in aeronautical and aerospace engineering. New York, NY: McGraw Hill, 2024. 1142 pp. ISBN: 978-1-266-07644-2.
- [58] Gerhard A. Holzapfel. *Nonlinear solid mechanics: a continuum approach for engineering*. Chichester ; New York: Wiley, 2000. 455 pp. ISBN: 978-0-471-82304-9 978-0-471-82319-3.
- [59] Anders Logg, Kent-Andre Mardal, and Garth Wells, eds. *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Vol. 84. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-23098-1 978-3-642-23099-8. DOI: 10.1007/978-3-642-23099-8.
- [60] Martin Alnæs et al. “The FEniCS Project Version 1.5”. In: *Archive of Numerical Software* Vol 3 (2015). Ed. by Archive Of Numerical Software. Version Number: 1.0.0. DOI: 10.11588/ANS.2015.100.20553.
- [61] Jim Lawrence, Javier Bernal, and Christoph Witzgall. “A Purely Algebraic Justification of the Kabsch-Umeyama Algorithm”. In: *Journal of Research of the National Institute of Standards and Technology* 124 (Oct. 9, 2019), pp. 1–6. ISSN: 1044-677X. DOI: 10.6028/jres.124.028.

A Architecture details and hyperparameters

Here, for each test case presented in this work, we report the selected configurations of hyperparameters, resulting from tuning experiments, as described in Section 3.

Test case: ADR		
Parameter	AD-IC-LDNet	Meta-IC-LDNet
<i>Architecture</i>		
Latent dimension d_s		2
$\mathcal{NN}_{\text{dyn}}$ architecture (activ.)	FCNN (tanh)	FCNN (SiLU)
$\mathcal{NN}_{\text{dyn}}$ hidden layers (width)		2 (9)
$\mathcal{NN}_{\text{rec}}$ architecture (activ.)	FCNN (tanh)	FCNN (SiLU)
$\mathcal{NN}_{\text{rec}}$ hidden layers (width)		2 (11)
<i>Model training</i>		
Loss function		MSE
Mini-batch size		800 (full batch)
Optimizer		Adam
Learning rate		10^{-4}
Validation frequency		10^3 epochs
Latent-space regularization		None
<i>Latent-code optimization</i>		
$\mathcal{T}_{\text{obs}}$		Initial condition only
Outer-loss time domain	-	Remainder of the trajectory
Optimizer	Adam	Gradient Descent
Optimization steps	10^3 (Val), 10^4 (Test)	3
Learning rate	10^{-2} (Val), 10^{-3} (Test)	10^{-2}

Table 3: ADR test case: hyperparameters (AD-IC-LDNet and Meta-IC-LDNet).

Test case: static cylinder		
Parameter	AD-IC-LDNet	Meta-IC-LDNet
<i>Architecture</i>		
Latent dimension d_s		2
$\mathcal{NN}_{\text{dyn}}$ architecture (activ.)		FCNN (SiLU)
$\mathcal{NN}_{\text{dyn}}$ hidden layers (width)		2 (16)
$\mathcal{NN}_{\text{rec}}$ architecture (activ.)		FiLM-modulated SIREN (sin)
$\mathcal{NN}_{\text{rec}}$ hidden layers (width)		2 (128)
<i>Model training</i>		
Loss function		MSE
Mini-batch size		24
Optimizer		Adam
Learning rate		10^{-4}
Validation frequency		25 epochs
Latent-space regularization	None	Curriculum learning (first 2 snaps for 500 ep., +1 every 100 ep.)
<i>Latent-code optimization</i>		
$\mathcal{T}_{\text{obs}}$		Initial condition only
Outer-loss time domain	-	Entire trajectory
Optimizer	Adam	Gradient Descent
Optimization steps	10^3 (Val), 10^4 (Test)	3
Learning rate	10^{-2} (Val), 10^{-3} (Test)	10^{-2}

Table 4: Static-cylinder test case: hyperparameters (AD-IC-LDNet and Meta-IC-LDNet).

Test case: rotating cylinder	
Parameter	Meta-IC-LDNet
<i>Architecture</i>	
Latent dimension d_s	4
$\mathcal{NN}_{\text{dyn}}$ architecture (activ.)	FCNN (SiLU)
$\mathcal{NN}_{\text{dyn}}$ hidden layers (width)	2 (32)
$\mathcal{NN}_{\text{rec}}$ architecture (activ.)	FiLM-modulated SIREN (sin)
$\mathcal{NN}_{\text{rec}}$ hidden layers (width)	2 (128)
<i>Model training</i>	
Loss function	MSE
Mini-batch size	14
Random spatial subsampling	1250 points (out of 80 000)
Fine-tuning points	54 500
Optimizer	Adam
Learning rate	10^{-4}
Latent-space regularization	Curriculum learning (first 3 snaps for 10^3 ep., +1 every 50 ep.)
<i>Latent-code optimization</i>	
$\mathcal{T}_{\text{obs}}$	First 3 snapshots
Outer-loss time domain	Entire trajectory
Optimizer	Gradient Descent
Optimization steps	3
Learning rate	10^{-2}

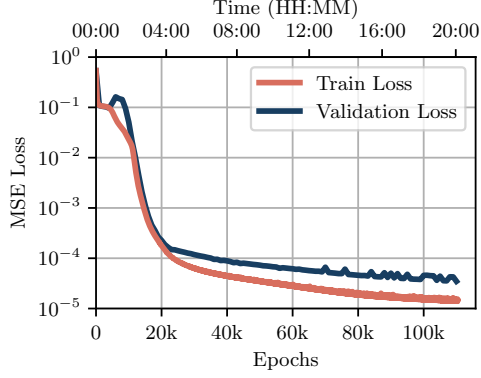
Table 5: Rotating-cylinder test case: hyperparameters (Meta-IC-LDNet).

Test case: nonlinear beam	
Parameter	Meta-IC-LDNet
<i>Architecture</i>	
Latent dimension d_s	12
$\mathcal{NN}_{\text{dyn}}$ architecture (activ.)	FCNN (SiLU)
$\mathcal{NN}_{\text{dyn}}$ hidden layers (width)	2 (32)
$\mathcal{NN}_{\text{rec}}$ architecture (activ.)	FCNN (SiLU)
$\mathcal{NN}_{\text{rec}}$ hidden layers (width)	4 (64)
<i>Model training</i>	
Loss function	MSE
Mini-batch size	24
Random spatial subsampling	500 points (out of 1025)
Optimizer	Adam
Learning rate	2×10^{-4}
Latent-space regularization	Latent-state penalty (3 additional inner loops at equidistant times)
<i>Latent-code optimization</i>	
$\mathcal{T}_{\text{obs}}$	First 3 snapshots
Outer-loss time domain	Entire trajectory
Optimizer	Gradient Descent
Optimization steps	3
Learning rate	10^{-2}

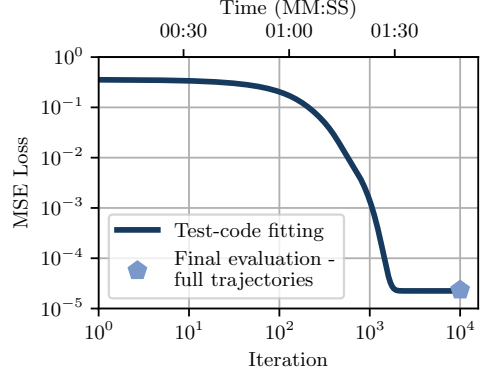
Table 6: Nonlinear-beam test case: hyperparameters (Meta-IC-LDNet).

B Training and validation

For each test case, this section presents the training and validation curves for the final hyperparameter configurations described in Appendix A. For the AD-IC-LDNet we also report the MSE evolution during the optimization of the initial codes for the unseen test-set examples. For the rotating-cylinder test case, we also report the fine-tuning run. In all figures, the top axis reports the elapsed time.



(a) Training and validation losses against epochs.



(b) Test-set latent-code optimization.

Figure 27: ADR test case: training, validation and testing (AD-IC-LDNet).

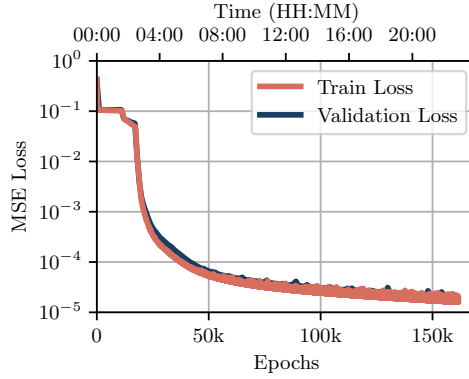
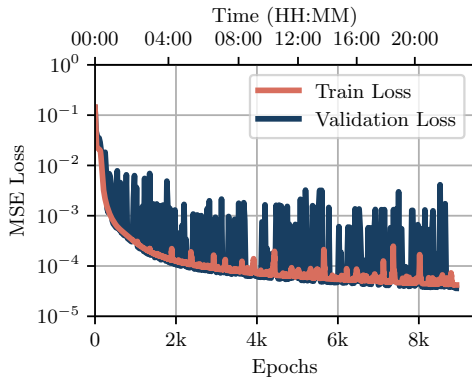
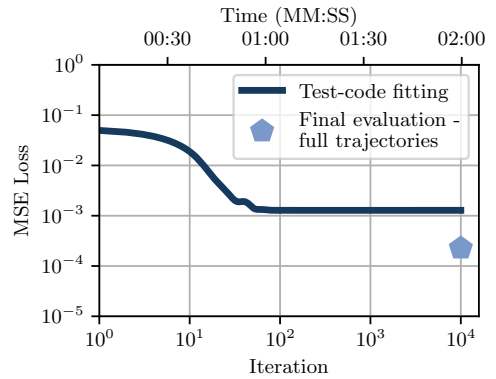


Figure 28: ADR test case: training and validation losses against epochs (Meta-IC-LDNet).



(a) Training and validation losses against epochs.



(b) Test-set latent-code optimization.

Figure 29: Static-cylinder test case: training, validation and testing (AD-IC-LDNet).

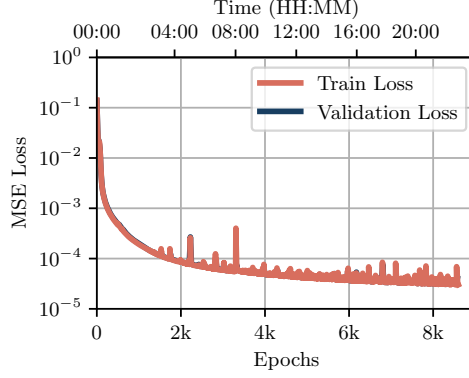
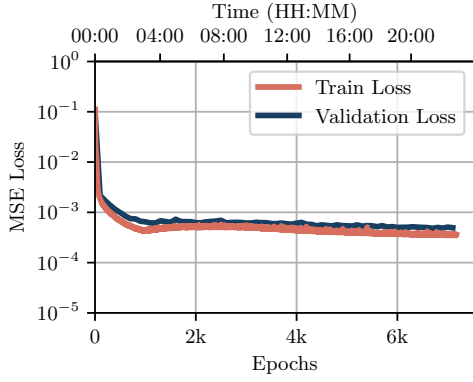
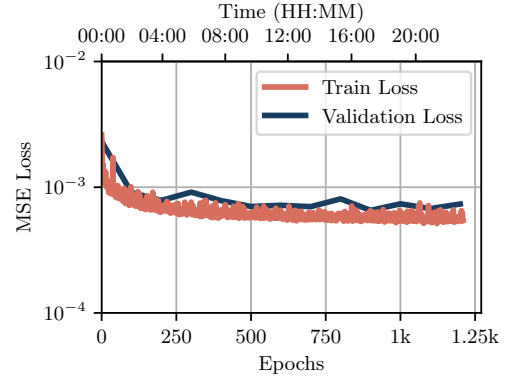


Figure 30: Static-cylinder test case: training and validation losses against epochs (Meta-IC-LDNet).



(a) Training and validation losses against epochs.



(b) Fine-tuning on a denser grid of points.

Figure 31: Rotating-cylinder test case: training and fine-tuning (Meta-IC-LDNet).

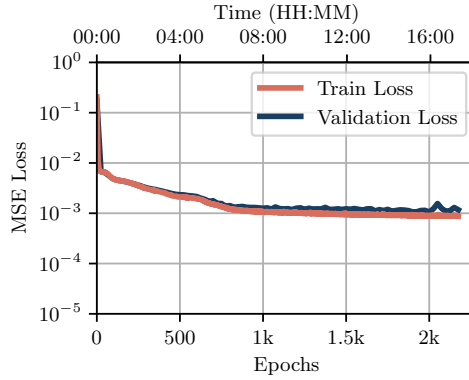


Figure 32: Nonlinear-beam test case: training and validation losses against epochs (Meta-IC-LDNet).

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

- 75/2026** Botteghi, N; Tomasetto, M; Fasel, U; Braghin, F; Manzoni, A
HypeMARL: Multi-Agent Reinforcement Learning For High-Dimensional, Parametric, and Distributed Systems
- 74/2026** Brivio, S.; Franco, N. R.
Deep Symmetric Autoencoders from the Eckart-Young-Schmidt Perspective
- 73/2026** Corti, M.
A weighted polygonal discontinuous Galerkin method for hierarchically coupled reaction–diffusion systems with cubic interactions
- 72/2026** Lorenzon, G.; Regazzoni, F.
Optimal Transport Dropout for Structured Predictive Uncertainty
- 70/2026** Micheletti, S.
Longitudinal versus quasi-P waves in weakly anisotropic elasticity: operator comparison and long-time dephasing
- 71/2026** Radisic, I.; Regazzoni, F.; Dede', L.; Quarteroni, A.
A mathematical model for irreversible damage of the collagen scaffold in the myocardium
- 68/2026** Patanè, G.; Greven, S.; Menafoglio, A.
Random mixtures in Bayes Hilbert spaces
- 69/2026** Zecchi, A. A.; Ferro, N.; Perotto, S.
Complexity-controlled QUBO formulation for topology optimization via anisotropic mesh adaptation
- 67/2026** Ciardulli, S.; Fontana, N.; Vantini, S.; Ieva F.
Generalized propensity score weighting for functional causal inference framework
- 66/2026** Antonietti, P.F.; Bonizzoni, F.; Corti, M.; De March, N.; Di Noto, S.; Regazzoni, F.
A stability-preserving polytopal discontinuous Galerkin method for the Fisher-Kolmogorov model with applications to neurodegenerative diseases