# Physics-informed reduced order model with conditional neural fields

**Minji Kim**
Department of Statistics and Operations Research
University of North Carolina at Chapel Hill
mkim5@unc.edu

**Tianshu Wen**
Aerospace and Mechanical Engineering
University of Notre Dame
twen2@nd.edu

**Kookjin Lee**
Computing and Augmented Intelligence
Arizona State University
kookjin.lee@asu.edu

**Youngsoo Choi**
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
choi15@llnl.gov

## Abstract

This study presents the conditional neural fields for reduced-order modeling (CNF-ROM) framework to approximate solutions of parametrized partial differential equations (PDEs). The approach combines a parametric neural ODE (PNODE) for modeling latent dynamics over time with a decoder that reconstructs PDE solutions from the corresponding latent states. We introduce a physics-informed learning objective for CNF-ROM, which includes two key components. First, the framework uses coordinate-based neural networks to calculate and minimize PDE residuals by computing spatial derivatives via automatic differentiation and applying the chain rule for time derivatives. Second, exact initial and boundary conditions (IC/BC) are imposed using approximate distance functions (ADFs) [Sukumar and Srivastava, *CMAME*, 2022]. However, ADFs introduce a trade-off as their second- or higher-order derivatives become unstable at the joining points of boundaries. To address this, we introduce an auxiliary network inspired by [Gladstone et al., *NeurIPS ML4PS* workshop, 2022]. Our method is validated through parameter extrapolation and interpolation, temporal extrapolation, and comparisons with analytical solutions.

## 1 Introduction

Numerical simulations for solving nonlinear partial differential equations (PDEs) have advanced scientific understanding by enabling accurate modeling of complex physical phenomena. However, the computational demands of high-fidelity simulations have led to the development of various surrogate and reduced-order modeling (ROM) techniques. Some of these techniques simplify the underlying physics, while another approach is to accelerate computations by leveraging data, as seen in linear subspace projection-based ROMs [1, 2, 3, 4, 5] and nonlinear manifold ROMs [6, 7, 8].

In recent years, neural networks have been employed increasingly to approximate PDE solutions, with physics-informed neural networks (PINNs) [9, 10, 11] emerging as an approach that learns physics by incorporating the governing PDE directly into the loss function. A more recent direction has emerged in coordinate-based neural networks, originally developed to learn implicit representations of complex signals, commonly referred to as implicit neural representations (INRs) [12, 13]. These networks are particularly attractive for PDE-related tasks because they can learn continuous functions over domains. Building on this capability, a series of works has been proposed that use data-driven approaches to learn PDE solutions using an INR-based decoder [14, 15, 16]. This approach leverages

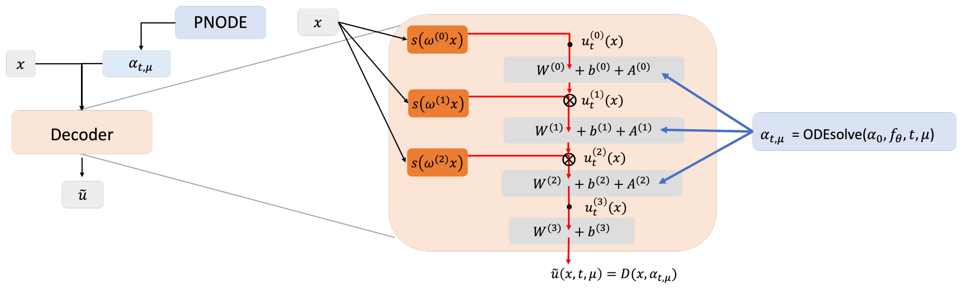
PNODE
x
$\alpha_{t,\mu}$
Decoder
$\tilde{u}$
x
$s(\omega^{(0)}x)$
$u_t^{(0)}(x)$
$W^{(0)} + b^{(0)} + A^{(0)}$
$s(\omega^{(1)}x)$
$u_t^{(1)}(x)$
$W^{(1)} + b^{(1)} + A^{(1)}$
$\alpha_{t,\mu} = \text{ODEsolve}(\alpha_0, f_\theta, t, \mu)$
$s(\omega^{(2)}x)$
$u_t^{(2)}(x)$
$W^{(2)} + b^{(2)} + A^{(2)}$
$u_t^{(3)}(x)$
$W^{(3)} + b^{(3)}$
$\tilde{u}(x,t,\mu) = D(x, \alpha_{t,\mu})$

Figure 1: Diagram of the CNF-ROM structure, combining PNODE with the architecture of [14].

the idea of modeling a vector field with a neural function that is conditioned on a latent state, a framework we refer to as conditional neural fields ROMs (CNF-ROMs).

Building on these developments, our work extends the CNF-ROM framework to support both data-driven and physics-informed learning. To the best of our knowledge, this is the first application of a space- and time-separated CNF for training PINNs. Our contributions are summarized as follows:

- Establishing a physics-informed learning objective for CNF-ROM framework,
- Enabling the framework to handle parametrized PDEs using PNODEs [17, 18],
- Addressing challenges in enforcing exact initial and boundary conditions (IC and BC) by introducing an auxiliary network to approximate first-order derivatives, following [19, 20],
- Introducing training objectives with simultaneous optimization of decoder and PNODE parameters,
- Verifying the performance in parameter interpolation/extrapolation (with PINN fine-tuning for unseen parameters), temporal extrapolation, and comparison with analytical solutions.

## 2 Models: Parameterized CNF-ROMs

We consider PDEs parametrized by $\boldsymbol{\mu} \in \mathcal{D}$, with $\mathcal{D} \in I\!R^{N_\mu}$, taking the form

$$\partial_t u = \mathcal{L}(u; \boldsymbol{\mu}), \quad u(\boldsymbol{x}, 0, \boldsymbol{\mu}) = u_0(\boldsymbol{x}, \boldsymbol{\mu}), \quad \mathcal{B}(u; \boldsymbol{\mu}) = 0, \quad t \in [0, T], \quad \boldsymbol{x} \in \Omega \in I\!R^d, \tag{1}$$

where $\mathcal{L}$ is a differential operator and $\mathcal{B}$ is a boundary operator. $u : \Omega \times (0, T] \times \mathcal{D} \to I\!R$ is the solution of the PDE that represents a physical quantity such as velocity or pressure. $u(\boldsymbol{x}, t, \boldsymbol{\mu})$ can be approximated by a neural network $u_{\boldsymbol{\theta}}(\boldsymbol{x}, t, \boldsymbol{\mu})$, where $\boldsymbol{\theta} \in I\!R^{d_{\boldsymbol{\theta}}}$ is a vector of tunable parameters.

**Conditional neural fields** When coordinate-based neural networks specifically operate in spatial and temporal domains, we refer to them as neural fields (NFs) [21], borrowing the term "fields" from physics. CNFs [22] extend NFs by introducing a conditioning latent factor $\boldsymbol{\alpha}$. A common approach uses a hypernetwork $h_{\boldsymbol{\psi}} : I\!R^{d_{\boldsymbol{\alpha}}} \to I\!R^{d_{\boldsymbol{\theta}}}$ to generate (a part of) high-dimensional parameters $\boldsymbol{\theta} = h_{\boldsymbol{\psi}}(\boldsymbol{\alpha}) \in I\!R^{d_{\boldsymbol{\theta}}}$ from a low-dimensional latent state $\boldsymbol{\alpha} \in I\!R^{d_{\boldsymbol{\alpha}}}$. This use of a hypernetwork enables reduced-order modeling by casting the problem as one of modeling latent states. We apply this framework to approximate solutions of spatio-temporal governing PDEs, referring to it as the conditional neural field for reduced-order modeling (CNF-ROM). This structure is illustrated in the diagram shown in Figure 1. The approach combines a parametric neural ODE (PNODE) [17], which models latent dynamics, with a decoder that reconstructs PDE solutions. The decoder maps the latent state and coordinates to the solution, while the PNODE captures distinct latent trajectories for each $\boldsymbol{\mu}$.

**Decoder** Building on the DINo architecture [14], we implement a decoder by employing a spatial coordinate-based network based on FourierNet [23]. To separate the spatial and temporal domains, DINo introduces the hypernetwork whose parameters are determined conditionally on the time-dependent latent state, which in our framework is extended to also depend on PDE parameters. The expanded section on the right side of Figure 1 illustrates the detailed DINo architecture integrated with PNODE. Following this approach, we define our decoder as

$$\tilde{u}(\boldsymbol{x}, t, \boldsymbol{\mu}) = D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}}), \tag{2}$$

where sinusoidal filters $s(\omega^{(l)}x) = [\sin(\omega^{(l)}x), \cos(\omega^{(l)}x)]^\top$ are used as Fourier basis. At each time step $t$ and parameter $\boldsymbol{\mu}$, a latent state $\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}$ determines the high-dimensional parameters of the decoder, thereby shifting the learning of the decoder to the training of the hypernetwork parameters $\boldsymbol{\psi}$.

**PNODE** Recent advancements in PINNs emphasize the importance of learning parameterized solutions to generalize across multiple PDE parameter values ($\boldsymbol{\mu}$) [11]. These extensions aim to capture diverse solution behaviors as a function of parameters, avoid retraining for every new parameter, and identify patterns across the parameter space, enabling generalization to unseen parameter values. Parameterized neural ODEs (PNODEs) [17] address these challenges by extending neural ODEs (NODEs) [24] to incorporate $\boldsymbol{\mu}$ as inputs, allowing them to model the evolution of latent states for multiple trajectories. In our model, PNODEs learn the function $f_\theta$:

$$\frac{d\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}}{dt} = f_{\boldsymbol{\theta}}(\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}, t, \boldsymbol{\mu}), \tag{3}$$

where $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_{\boldsymbol{\alpha}}} \times [0, T] \times \mathcal{D} \to \mathbb{R}^{d_{\boldsymbol{\alpha}}}$ represents the velocity of the latent state based on the current state, time, and PDE parameters. By integrating PNODEs with the decoder in Eq. (2), we model the evolution of parametrized PDE solutions over time in a reduced-order latent space.

## 3 Methods: Physics-informed training with exact satisfaction of IC/BC

**Physics-informed learning with parametrized CNF-ROM** With the CNF-ROM framework introduced in Section 2, we propose a physics-informed learning objective for PINNs. A physics-informed learning objective typically includes loss terms for the governing PDE residual, as well as for IC and BC. These terms ensure that the solution satisfies the PDE and adheres to the given IC and BC [9]. To compute the necessary spatial and temporal derivatives for the PDE residual loss, the CNF-ROM leverages coordinate-based neural networks to directly compute spatial derivatives via automatic differentiation. Temporal dynamics, on the other hand, are modeled through a latent state $\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}$ that evolves over time for each $\boldsymbol{\mu}$. Using the chain rule, temporal derivatives are expressed as:

$$\partial_t D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}}) = \partial_{\boldsymbol{\alpha}} D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}}) \cdot f_{\boldsymbol{\theta}}(\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}, t, \boldsymbol{\mu}), \tag{4}$$

where $\partial_{\boldsymbol{\alpha}} D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}}) = \partial_{\boldsymbol{\alpha}} D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha})\big|_{\boldsymbol{\alpha}_{t,\boldsymbol{\mu}}}$ can also be computed using automatic differentiation. This combination of coordinate-based neural networks and latent-state dynamics enables efficient and accurate computation of spatio-temporal derivatives.

Instead of incorporating IC and BC as loss terms, we address strategies for imposing exact IC and BC and discuss the trade-offs in this approach in the following paragraphs.

**Exact imposition of IC and BC** Exact imposition of IC and BC reduces the number of loss constraints while ensuring the uniqueness of the solution, which improves training convergence and predictive accuracy. The R-function-based approximate distance functions (ADFs) $\phi$ proposed by [19] were developed to enforce boundary conditions while satisfying other desirable properties such as differentiability. These functions can be constructed to satisfy Dirichlet, Neumann, and Robin conditions a priori, even on complex and irregular geometries; see [19] for more details. Building on this foundation, we extend ADFs to the temporal domain to address initial conditions as well by ensuring $\phi(\boldsymbol{x}, t) = 0$ when $\boldsymbol{x} \in \partial\Omega$ or $t = 0$. For Dirichlet boundary conditions, where $\widehat{u} = g$ is imposed on boundaries, we use the following construction:

$$\widehat{u}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\psi}, \boldsymbol{\theta}) = g(\boldsymbol{x}, t; \boldsymbol{\mu}) + \phi(\boldsymbol{x}, t) D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}}(\boldsymbol{\theta})). \tag{5}$$

Here, $\phi$ is an ADF and $D_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\alpha}_{t,\boldsymbol{\mu}})$ denotes the decoder output in Eq. (2).

When using the R-function-based ADFs, the following challenges arise when the input dimension is greater than two. In our case, this holds as the ADF includes both time and space coordinates. To be specific, (i) ADF $\phi$ is not well-defined on the boundary, and (ii) its second (or higher) derivatives explode near boundary junctions, as noted in [19, 20, 25]. To mitigate these challenges, (i) we exclude the boundary set from the PINN loss definition in Eq. (8) and (ii) adopt a more delicate approach proposed in [20]. This approach addresses the trade-off in imposing IC and BC on $\widehat{u}$ by approximating its first-order derivatives. For example, to compute $\partial_{xx}\widehat{u}$, instead of directly computing the second-order derivatives, we first train an auxiliary network to approximate the first-order derivatives and then compute the derivative of this network's output. To achieve this, we train another CNF-ROM $\widetilde{v}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\xi}, \boldsymbol{\theta}) = D_{\boldsymbol{\xi}}(\boldsymbol{x}, \boldsymbol{\beta}_{t,\boldsymbol{\mu}}(\boldsymbol{\theta}))$ and add an additional loss term matching the derivatives,

$$L_{\text{deriv}}(\boldsymbol{\xi}, \boldsymbol{\theta}) = \frac{1}{N_o} \sum_{\boldsymbol{\mu} \in \mathcal{D}} \sum_{t \in \mathcal{T} \setminus \{0\}^c} \sum_{\boldsymbol{x} \in \mathcal{X} \setminus \partial\Omega^c} \|\partial_x \widehat{u}(\boldsymbol{x}, t, \boldsymbol{\mu}) - \widetilde{v}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\xi}, \boldsymbol{\theta})\|^2, \tag{6}$$

so that the second or higher-order derivatives of $\widehat{u}$ are approximated using the derivatives of $\widetilde{v}$. Note that $\widetilde{v}$ is constructed without ADF constraints, and thus higher-order derivatives are obtainable.

In the following paragraph, we describe two learning modes: data-driven learning and physics-informed learning. In both modes, the solution is constructed as in Eq. (5) to impose IC and BC.

**Training objectives** One key advantage of imposing exact IC and BC is that it eliminates the need for an encoder. Since the output satisfies the initial condition for any arbitrary $\boldsymbol{\alpha}_0$, we initialize $\boldsymbol{\alpha}_0 = \mathbf{0} \in \mathbb{R}^{d_{\boldsymbol{\alpha}}}$. Once the initial latent state is obtained, we use ODE solvers [24] to evolve it according to Eq. (3). The use of PNODEs allows the initial trajectory to adapt based on different values of $\boldsymbol{\mu}$, enabling the model to generate distinct outputs. Note that for $\widetilde{v}$ in Eq. (6), $\boldsymbol{\beta}_{0,\boldsymbol{\mu}}$ needs to be learned, and we use an auto-decoding approach as suggested in [26].

We propose the following training objectives for CNF-ROM, employing an efficient simultaneous optimization that jointly updates the decoder and PNODE parameters.

- *Data loss:* When data is available, we minimize the following data loss:

$$L_{\text{data}}(\boldsymbol{\psi}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{(\boldsymbol{x}, t, \boldsymbol{\mu}) \in \mathcal{C}} \|u(\boldsymbol{x}, t, \boldsymbol{\mu}) - \widehat{u}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\psi}, \boldsymbol{\theta})\|^2, \tag{7}$$

  where $\widehat{u}$ is defined in Eq. (5), $\mathcal{C} = \mathcal{X} \times \mathcal{T} \times \mathcal{D}$ is the set of collocation points, and $N = |\mathcal{C}|$.

- *PINN loss:* For physics-informed training, we minimize the following PDE residual loss:

$$L_{\text{PDE}}(\boldsymbol{\psi}, \boldsymbol{\theta}) = \frac{1}{N_o} \sum_{(\boldsymbol{x}, t, \boldsymbol{\mu}) \in \mathcal{C}_0} \|\partial_t \widehat{u}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\psi}, \boldsymbol{\theta}) - \mathcal{L}(\widehat{u}(\boldsymbol{x}, t, \boldsymbol{\mu}; \boldsymbol{\psi}, \boldsymbol{\theta}); v(\boldsymbol{x}, t, \boldsymbol{\mu}))\|^2, \tag{8}$$

  where $\mathcal{C}_0 = \mathcal{X} \setminus \partial\Omega^c \times \mathcal{T} \setminus \{0\}^c \times \mathcal{D}$, $N_o = |\mathcal{C}_0|$, and $\mathcal{L}$ is from Eq. (1). When the governing equation involves second or higher-order derivatives, these terms in Eq. (8) are replaced with those computed using the derivatives of $\widetilde{v}$. Note that IC and BC are imposed a priori.

We are now equipped with the training losses defined in Eq. 6, (7) and (8). In Section 4, we present training scenarios and numerical results for the 1D viscous Burgers equation.

## 4 Numerical results and discussion

We consider the 1D viscous Burgers equation on $\Omega \times [0, T] = [0, 2] \times [0, 1]$ with specific IC and BC, where a known solution exists. The governing equation, parametrized by the Reynolds number $\mu$, is:

$$\partial_t u + u \cdot \partial_x u - (1/\mu) \partial_{xx} u = 0, \quad u(0, t) = u(2, t) = 0 \text{ on } t \in [0, 1]. \tag{9}$$

The exact solution used to validate the performance of our models is given by

$$u_{ex}(x, t; \mu) = \frac{2\pi}{\mu} \left\{ \frac{\frac{1}{4} e^{-\pi^2 t/\mu} \sin(\pi x) + e^{-4\pi^2 t/\mu} \sin(2\pi x)}{1 + \frac{1}{4} e^{-\pi^2 t/\mu} \cos(\pi x) + \frac{1}{2} e^{-4\pi^2 t/\mu} \cos(2\pi x)} \right\}, \tag{10}$$

with the initial condition determined by (10) at $t = 0$. The sets of training and test parameters $\boldsymbol{\mu}_{\text{train}} = \{20, 30, 40, 50, 60, 70, 80, 100\}$ and $\boldsymbol{\mu}_{\text{test}} = \{15, 25, 45, 90, 110\}$ and the latent state dimension $d_{\boldsymbol{\alpha}} = 10$ are used. The numerical solution of the full-order model was obtained using a uniform spatial and temporal discretization ($n_x = 64$, $n_t = 100$) and a backward Euler time integrator. We consider the following training scenarios:

(a) Training with data loss: $L_{\text{data}}(\boldsymbol{\psi}, \boldsymbol{\theta}) + L_{\text{deriv}}(\boldsymbol{\xi}, \boldsymbol{\theta})$, $T = 1$, across $\boldsymbol{\mu}_{\text{train}}$.

(b) Fine-tuning with PINN loss: $L_{\text{PDE}}(\boldsymbol{\theta}) + L_{\text{deriv}}(\boldsymbol{\theta})$, $T = 1$, $\boldsymbol{\mu} \in \boldsymbol{\mu}_{\text{train}} \cup \boldsymbol{\mu}_{\text{test}}$.

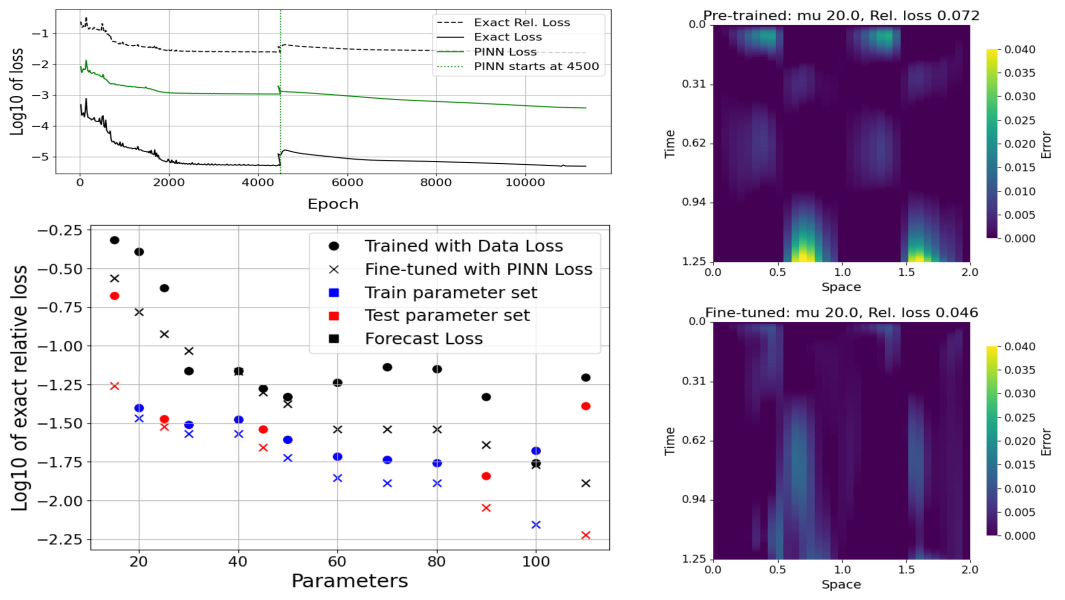

Figure 2: Performance verification through: (top left) loss trajectories, (bottom left) evaluation on training and test set parameters, and temporal extrapolation at $t = 1.25$; (top right) heatmap of loss for model (a) at $\boldsymbol{\mu} = 20$; (bottom right) heatmap of loss for model (b) at $\boldsymbol{\mu} = 20$.

The goal of these scenarios is to highlight the advantage of using PINN as a fine-tuning objective, leveraging its ability to learn without data. For scenario (a), both the decoder and the auxiliary network are optimized during training. For scenario (b), starting from the pre-trained model from (a), we freeze the decoder and use the PDE loss to update only the PNODE (latent dynamics) parameters for the target parameters. This PINN fine-tuning stage models only the low-dimensional latent state, aligning with the ROM perspective.

Figure 2 presents the performance evaluation from multiple perspectives. The top left panel of presents the loss trajectory, displaying the loss with respect to exact solutions in Eq. (10) (“Exact Loss", solid black), its relative error (“Exact Rel. Loss", dashed black) and the PDE loss for PINN (“PINN Loss", solid green). The model was pre-trained under scenario (a) up to epoch 4500, followed by fine-tuning under scenario (b) for the fixed parameter $\boldsymbol{\mu} = 20$. The purpose of this figure is to validate the PDE loss calculation by showing that both exact and PINN losses decrease in parallel, regardless of whether the training was based on (a) data loss or (b) PINN loss, confirming that the PDE residual loss aligns with the exact solution. The right panels of Figure 2 present heatmaps of the loss for $\mu = 20$. The top right heatmap corresponds to the model trained under scenario (a), while the bottom right heatmap shows the result of the fine-tuned model (b). We first observe that additional fine-tuning with PINN loss led to additional error reduction. Notably, the region $t > 1$ represents an extrapolation beyond the trained domain (i.e. forecasting). The error plots reveal that forecast errors are more pronounced in the pre-trained model, whereas the fine-tuned PINN loss model exhibits robust result in this region.

The bottom left panel of Figure 2 offers a summary of the performance of the model trained under the pre-trained scenario (a, ●) and the fine-tuned model (b, ✕) for each parameter $\boldsymbol{\mu} \in \boldsymbol{\mu}_{\text{train}} \cup \boldsymbol{\mu}_{\text{test}}$. This panel compares fine-tuning results for the training set (blue), inter-/extrapolation results in the parameter space (red), and temporal extrapolation (black). For temporal extrapolation, all models (a) and (b) were trained up to $T = 1$ and used to evaluate up to $t = 1.25$. Specifically, fine-tuning with PINN loss demonstrated clear improvements across all regions except temporal extrapolation performance at $\mu = 30$, with a particularly significant error reduction in parameter extrapolation areas where $\mu = 15$ or $110$. In general, we observed that the fine-tuning with PINN loss improves model performance by learning without additional data, enabling further learning for unseen parameters, and enhancing robustness in forecast regions. Finally, we conclude with the remark that the CNF-ROM framework can be extended to higher-dimensional problems, which we leave their exploration as a future direction.

## Acknowledgments and Disclosure of Funding

**K. Lee** was supported by NSF under grant IIS #2338909. **Y. Choi** was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, as part of the CHaRMNET Mathematical Multifaceted Integrated Capability Center (MMICC) program, under Award Number DE-SC0023164. Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344. IM release number: LLNL-CONF-869137.

## References

[1] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The gnat method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[2] Youngkyu Kim, Karen Wang, and Youngsoo Choi. Efficient space–time reduced order model for linear dynamical systems in python using less than 120 lines of code. *Mathematics*, 9(14):1690, 2021.

[3] Dylan Matthew Copeland, Siu Wun Cheung, Kevin Huynh, and Youngsoo Choi. Reduced order models for lagrangian hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 388:114259, 2022.

[4] Youngsoo Choi and Kevin Carlberg. Space–time least-squares petrov–galerkin projection for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 41(1):A26–A58, 2019.

[5] Youngsoo Choi, Deshawn Coombs, and Robert Anderson. SNS: A solution-based nonlinear subspace method for time-dependent model order reduction. *SIAM Journal on Scientific Computing*, 42(2):A1116–A1146, 2020.

[6] Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, volume 38, pages 379–391. Wiley Online Library, 2019.

[7] Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[8] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

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

[10] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[11] Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized PDEs. *Preprint, arXiv:2408.09446*, 2024.

[12] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[13] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[14] Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Thomas Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Zhong Yi Wan, Leonardo Zepeda-Nunez, Anudhyan Boral, and Fei Sha. Evolve smoothly, fit consistently: Learning smooth latent dynamics for advection-dominated systems. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] Kookjin Lee and Eric J. Parish. Parameterized neural ordinary differential equations: applications to computational physics problems. *Proceedings of the Royal Society A*, 477(2251):20210162, 2021.

[18] Tianshu Wen, Kookjin Lee, and Youngsoo Choi. Reduced-order modeling for parameterized pdes via implicit neural representations. In *Proceedings of the Machine Learning and the Physical Sciences Workshop at the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[19] N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

[20] Rini J. Gladstone, Mohammad A. Nabian, N. Sukumar, Ankit Srivastava, and Hadi Meidani. FO-PINNs: A first-order formulation for physics informed neural networks. In *Proceedings of the Machine Learning and the Physical Sciences Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[21] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[22] Jan Hagnberger, Marimuthu Kalimuthu, Daniel Musekamp, and Mathias Niepert. Vectorized conditional neural fields: A framework for solving time-dependent parametric partial differential equations. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17189–17223. PMLR, 21–27 Jul 2024.

[23] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021.

[24] Ricky T.Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[25] Stefano Berrone, Claudio Canuto, Moreno Pintore, and N. Sukumar. Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks. *Heliyon*, 9(8):e18820, 2023.

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.