Rollout-LaSDI: Enhancing the long-term accuracy of Latent Space Dynamics.

Robert Stephany
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, Ca 94550
`stephany1@llnl.gov`

Youngsoo Choi
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, Ca 94550
`choi15@llnl.gov`

Abstract

Solving complex partial differential equations is vital in the physical sciences, but often requires computationally expensive numerical methods. Reduced-order models (ROMs) address this by exploiting dimensionality reduction to create fast approximations. While modern ROMs can solve parameterized families of PDEs, their predictive power degrades over long time horizons. We address this by (1) introducing a flexible, high-order, yet inexpensive finite-difference scheme and (2) proposing a Rollout loss that trains ROMs to make accurate predictions over arbitrary time horizons. We demonstrate our approach on the 2D Burgers equation.

1 Introduction

The profound success of *Deep Neural Networks (DNNs)* [16, 22, 33, 8] has given rise to *Scientific Machine Learning (SciML)*, which uses machine learning to address challenges in science and engineering. SciML has engendered novel PDE solvers [27], algorithms to discover governing models [9, 29, 26], and *Reduced Order Models* (ROMs) to accelerate simulations [17, 7].

Numerically solving *Partial Differential Equations (PDEs) — full-order models (FOMs)* — is central to the physical sciences [31] and engineering [19, 1]. However, high-fidelity solvers are computationally expensive [3], hindering their usefulness in time-sensitive applications such as model predictive control [20]. ROMs address this by providing fast approximations to FOM solutions [14].

A ROM typically comprises: (1) an encoder mapping a FOM state (solution at a specific time) to a low-dimensional *latent space*, (2) *latent dynamics* describing the time evolution of the encoded states, and (3) a decoder returning latent states to FOM states [4, 30, 14, 11]. ROMs are predicated on the *manifold hypothesis*: many high-dimensional datasets lie on or near a low-dimensional manifold [15]. If the encoder-decoder pair can approximate a chart and its inverse, they can absorb much of the dataset's complexity, leaving simple latent dynamics to learn. We can use such a ROM to quickly approximate the FOM solution via the following workflow: (1) encode the initial FOM state, (2) evolve the latent dynamics, then (3) decode to approximate the final FOM state. Time-stepping occurs in latent space and can be orders of magnitude faster than solving the FOM [6].

[11] [9] [28] [12] [23] [1] [14]

LaSDI operates on a collection of time series, each one consisting of FOM states for a specific (and known) parameter value. LaSDI trains an autoencoder and learns distinct latent dynamics for each parameter instance[2]. At inference, LaSDI encodes the initial FOM state and interpolates among the learned latent dynamics to obtain dynamics for a new parameter. LaSDI has several extensions [25, 32, 2, 13, 18], including for interpolating the latent dynamics: an intrusive variant leverages

[1]Parameters may alter the initial condition and/or governing equation.

[2]Because parameters change the FOM dynamics, a single set of latent dynamics cannot serve all values.

known governing equations to greedily add parameters during training [17]; a non-intrusive variant, *GPLaSDI*, uses *Gaussian processes (GPs)* [34] to add parameters and interpolate dynamics without knowing the FOM [6, 5]. This paper builds on *GPLaSDI*.

While GPLaSDI represents significant progress, it faces two limitations that we address in this paper. First, for time series with nonuniform time steps, GPLaSDI estimates latent derivatives with a two-point finite difference scheme, yielding poor estimates and hindering learning. We introduce an inexpensive finite-difference scheme with higher-order accuracy that works on nonuniform time series. Second, GPLaSDI, like SINDy, approximates the latent dynamics *locally* by learning latent dynamics that approximately hold at each time step. Small errors can accumulate, degrading the predictive power of the ROM over long time horizons. [24] add a *Rollout loss* that trains the ROM to predict future FOM states over arbitrary horizons. These enhancements improve long-horizon prediction accuracy while maintaining the ROM's computational efficiency at inference.

2 Methodology

In this paper, the FOM is a parameterized family of PDEs:

$$\begin{aligned} &\tfrac{d}{dt}u_\theta\left(t,X\right)=F\left(u_\theta\left(t,X\right),t,X,\theta\right), \qquad (t,X)\in(0,T]\times\Omega,\\ &u_\theta(0,X)=u_0(X,\theta). \end{aligned} \tag{1}$$

Here, $\Omega\subseteq\mathbb{R}^{N_s}$ is the spatial portion of the problem domain, $T>0$ is the final time, and $\theta\in\Theta\subseteq\mathbb{R}^p$ is a parameter that defines PDE's initial condition and governing equation. We assume (1) can be solved by a high-fidelity solver for all $\theta\in\Theta$. We let $\{\vec{u}_\theta(t_j^\theta)\}_{j=0}^{N_t(\theta)}\subseteq\mathbb{R}^{N_u}$ denote a numerical solution to the FOM (1) at times $0=t_0^\theta\le\cdots\le t_{N_t(\theta)}^\theta\le T$ using N_u spatial nodes. [28] Our training set starts with time series for $N_\theta\in\mathbb{N}$ parameter values, $\{\theta_i\}_{i=1}^{N_\theta}$. We assume we can dynamically generate new time series, but do not know (1) (our approach is *non-intrusive*).

Our ROM consists of an autoencoder (with encoder $\varphi_e:\mathbb{R}^{N_u}\to\mathbb{R}^L$ and decoder $\varphi_d:\mathbb{R}^L\to\mathbb{R}^{N_u}$, both of which are trainable DNNs) with latent space $\mathbb{R}^L$ ($L\ll N_u$). The encoder maps each state, $\vec{u}_\theta(t_i^\theta,X)$, to a low-dimensional latent representation, $\varphi_e(\vec{u}_\theta(t_j^\theta))$. Restricted to the data manifold M, we train the autoencoder to satisfy $\varphi_d\circ\varphi_e|_M\approx\mathrm{Id}_M$ by minimizing the mean absolute error,

$$\mathcal{L}_{\text{Recon}}=\frac{1}{N_t}\sum_{i=1}^{N_\theta}\sum_{j=1}^{N_t(\theta_i)}\left\|\vec{u}_{\theta_i}\left(t_j^{\theta_i}\right)-(\varphi_d\circ\varphi_e)\left(\vec{u}_{\theta_i}\left(t_j^{\theta_i}\right)\right)\right\|_1, \quad N_t=\sum_{i=1}^{N_\theta}N_t(\theta_i). \tag{2}$$

The encoder can iteratively map a FOM time-series to the latent space, yielding a latent time series. Let $\vec{z}_\theta(t_j^\theta)=\varphi_e(\vec{u}_\theta(t_j^\theta))$. We posit there are unknown, parameter-specific *latent coefficients* - $A_\theta\in\mathbb{R}^{L\times L}$ and $b_\theta\in\mathbb{R}^L$, such that

$$\dot{\vec{z}}_\theta\left(t_i^\theta\right)\approx A_\theta\vec{z}_\theta\left(t_i^\theta\right)+b_\theta, \qquad t\in(0,T]. \tag{3}$$

We fit these coefficients by minimizing the *Latent Dynamics loss*,

$$\mathcal{L}_{\text{LD}}=\frac{1}{N_t}\sum_{i=1}^{N_\theta}\sum_{j=1}^{N_t(\theta_i)}\left\|\dot{z}_\theta\left(t_j^{\theta_i}\right)-\left[A_{\theta_i}\varphi_e\left(\vec{u}_{\theta_i}\left(t_j^{\theta_i}\right)\right)+b_{\theta_i}\right]\right\|_2^2. \tag{4}$$

We zero-initialize A_{θ_i} and b_{θ_i} and learn them jointly with φ_e,φ_d [3]. We learn distinct coefficients for each training parameter value.

Unlike GPLaSDI, we use an $\mathcal{O}(h^2)$ three-point finite difference scheme that supports nonuniform time steps (common in real data and stiff, adaptively stepped systems). Let U be a normed vector space, $f:\mathbb{R}\to U$ with U, and suppose we know $f(t-a)$, $f(t)$, $f(t+b)$ for some $a,b>0$. Then,

$$f'(t)=\tfrac{-b}{a(a+b)}f(t-a)+\tfrac{b-a}{ab}f(t)+\tfrac{a}{b(a+b)}f(t+b)+\mathcal{O}(h^2), \quad h=\max\{a,b\}. \tag{5}$$

This follows by taking third-order Taylor expansions of $f(t-a)$ and $f(t+b)$ and solving for $c_{-1},c_0,c_1\in\mathbb{R}$ such that $c_{-1}f(t-a)+c_0f(t)+c_1f(t+b)$ approximates $f'(t)$ with $\mathcal{O}(h^2)$ error

[3]Unlike GPLaSDI, which computes latent coefficients via differentiable linear regression at each step, we treat them as trainable parameters for stability.

[10]. We use analogous left/right three-point formulas, giving us one-sided, nonuniform stencils with $\mathcal{O}(h^2)$ accuracy.

Because the coefficients in (5) are rational functions of a and b, we can compute the derivative of an entire time series in linear time; in practice, runtime is just $2\times$ that of the standard three-point scheme. The resulting high-fidelity latent-velocity estimates improve our ability to learn and Rollout the latent dynamics, especially when using higher-order ODE solvers (e.g., RK4).

To improve long-horizon accuracy, we introduce a *Rollout loss*. At each epoch and for each parameter θ, we choose a maximum Rollout horizon $\Delta t^{\theta}_{\max}$ (annealed from near zero during training) and define all frames $\vec{u}_\theta(t^\theta)$ with $t^\theta + \Delta t^\theta_{\max} \le T$ as *rollable*. We encode each rollable frame, $\vec{u}_\theta(t)$, to get a latent vector, $\vec{z}_\theta(t)$, which we use as the initial condition to the latent dynamics. We then integrate using a differentiable RK4 solver with a random[4] future time $t + \Delta t^\theta$. Here, $\Delta t^\theta \sim \mathcal{U}\big(0, \Delta t^\theta_{\max}\big)$. We decode the final latent state, $\hat{z}_\theta(t + \Delta t^\theta)$, to get $\hat{u}_\theta(t + \Delta t^\theta) = \varphi_d\big(\hat{z}_\theta(t + \Delta t^\theta)\big)$, and compare it with $\tilde{u}_\theta(t + \Delta t^\theta)$, a cubic-spline interpolation of the FOM solution. Doing this for each frame and parameter gives us the Rollout loss:

$$\mathcal{L}_{\text{Rollout}} = \frac{1}{N_{ro}} \sum_{i=1}^{N_\theta} \sum_{j=1}^{N_{ro}(\theta_i)} \left\| \tilde{u}_{\theta_i}\left(t_j^{\theta_i} + \Delta t_{\text{ro}}^{\theta_i}(j)\right) - \hat{u}_{\theta_i}\left(t_j^{\theta_i} + \Delta t_{ro}^{\theta_i}(j)\right) \right\|_1, \tag{6}$$

where $N_{ro}(\theta_i)$ is the number of rollable frames for θ_i and $N_{ro} = \sum_{i=1}^{N_\theta} N_{ro}(\theta_i)$. $\mathcal{L}_{\text{Rollout}}$ depends on the encoder/decoder parameters and the latent coefficients. Figure 1 illustrates the procedure.

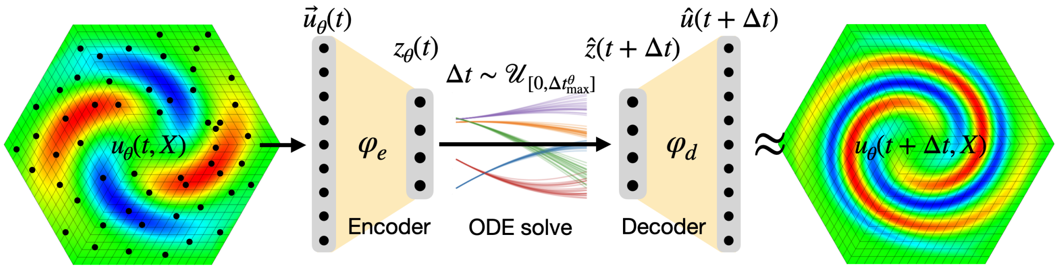

Figure 1: Rolling out $\vec{u}_\theta(t)$, the discretization of $u_\theta(t, X)$. From left to right, we begin with $\vec{u}_\theta(t)$, a numerical approximation of the FOM solution, $u_\theta(t, X)$ (black circles represent the spatial nodes). We encode $\vec{u}_\theta(t)$, use the encoding as the latent dynamics' initial condition. We solve the dynamics over a random time horizon, then decode the final latent state to predict the future FOM state.

We train our ROM using Adam [21] to minimize the following loss:

$$\mathcal{L}\left(\varphi_e, \varphi_d, \{A_{\theta_i}, b_{\theta_i}\}_{i=1}^{N_\theta}\right) = \eta_1 \mathcal{L}_{\text{Recon}} + \eta_2 \mathcal{L}_{\text{LD}} + \eta_3 \mathcal{L}_{\text{Rollout}} + \eta_4 \sum_{i=1}^{N_\theta} \left(\|A_{\theta_i}\|_F^2 + \|b_{\theta_i}\|_2^2 \right), \tag{7}$$

where $\eta_1, \dots, \eta_4$ are hyperparameters. We adopt GPLaSDI's GP-based coefficient interpolation to adaptively add parameters to the training set; selecting those whose latent-coefficient posteriors induce the largest predictive variance in FOM space. At test time, we use the GP posterior mean to infer latent coefficients for new parameters [6]. We refer to our algorithm as *Rollout-LaSDI*. An open source implementation of Rollout-LaSDI is available at https://github.com/punkduckable/Rollout-LaSDI.

3 Experiments and Discussion

We test Rollout-LaSDI on the 2D Burgers equation with periodic *Boundary Conditions (BCs)*:

$$\begin{aligned} \dot{u} &= -uu_x - uu_y + \nu\left(u_{xx} + u_{yy}\right) \\ u\left(0, (x, y)\right) &= \exp\left(-k(x^2 + y^2)\right) \sin\left(\tfrac{\pi}{2}\omega x\right) \sin\left(\tfrac{\pi}{2}\omega y\right). \end{aligned} \tag{8}$$

We consider two parameters: ν, scaling the Laplacian, and ω, controlling the spatial frequency of the initial condition. We generate FOM solutions using a high-fidelity finite-difference solver on

[4]Different frames will be simulated for different amounts of time. This changes each epoch.

$(0, T] \times \Omega = (0, 2] \times [-2, 2]^2$. We discretize this domain with 500 time points and 51×51 spatial nodes ($N_u = 2601$). The encoder is a fully connected DNN with widths [2601, 250, 100, 100, 100, 5] ($L = 5$) and sin activation functions; the decoder mirrors the encoder. Finally, during training, we set $\{\eta_1, \eta_2, \eta_3, \eta_4\} = \{1, 1, 1, 0.001\}$ (see (7)).

We train for $17,500$ epochs using Adam (lr $= 10^{-3}$), perform greedy sampling (to pick a θ and corresponding time-series to add to the training set) every 2500 epochs, and use 20 GP samples to estimate FOM variance [6]. The test set consists of an 11×11 grid (121 parameter combinations, with 11 values for both ν and ω). The training set begins with 4 parameters (red-bordered squares in Fig. 2) but dynamically grows (black-bordered squares) via greedy sampling as the ROM trains [6].

At test time, the final ROM and learned GPs predict the FOM solution for each θ: we encode the FOM initial condition[5], set latent coefficients to the GP posterior mean, integrate the latent dynamics on $(0, 2]$, and decode to obtain the predicted FOM time series. Because Rollout-LaSDI only uses $\mathcal{L}_{\text{Rollout}}$ during training, and because Runge–Kutta solvers (used in GPLaSDI and Rollout-LaSDI) handle variable step sizes, inference in Rollout-LaSDI is identical to Rollout with GPLaSDI.

To measure the performance of the final ROM, we compute the maximum relative error between a FOM frame, $\vec{u}_\theta(t_j)$, and Rollout-LaSDI's prediction, $\hat{u}_\theta(t_j)$:

$$\text{Error}(\theta) = \max_k \left(\frac{\text{mean}_i \, |\vec{u}_\theta(t_k)_i - \hat{u}_\theta(t_k)_i|}{\sigma_{i,j}\, \{|\vec{u}_\theta(t_i)_j|\}} \right). \tag{9}$$

We normalize by the standard deviation of FOM frame components to avoid division by zero, which could occur with node-specific or time-specific normalizations near the boundary of Ω or when the FOM state approach zero.

We run four experiments: two with variable time steps (each step size randomly varied) and two with fixed steps. In each subset, we train one ROM with Rollout loss and one without it (the two ROMs and their training procedure are identical otherwise). Figure 2 reports the results across all θ values.

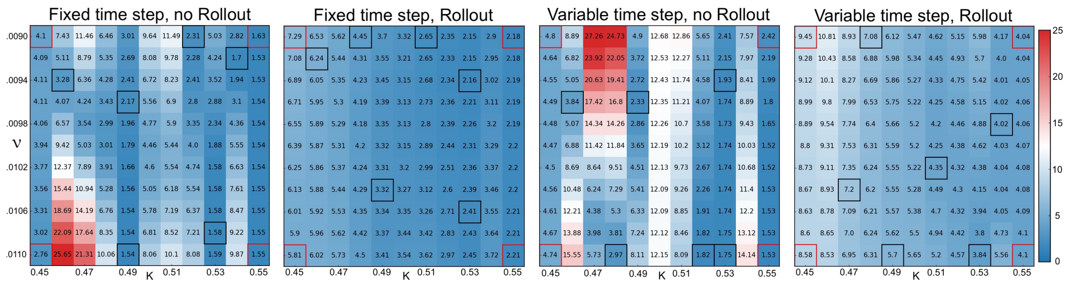

Figure 2: Error for all 121 parameter values with and without Rollout. Here, the four red cornered boxes represent the parameter values that are originally in the training set, while the black ones represent parameters that were dynamically added to the training set during training.

In both fixed- and variable-time-step experiments, adding Rollout reduces the maximum relative error by $3\times$ and the median error by $2\times$. Experiments with variable time steps have slightly higher relative error than those with a fixed time step size, as expected. These results demonstrate that Rollout improves the long-horizon accuracy of the final ROM without harming test-time performance.

Beyond accuracy improvements, our enhancements maintain GPLaSDI's computational efficiency. Our ROMs can achieve the same 10^5-factor speedup as GPLaSDI [5, 6] while providing superior long-horizon predictions. We find Rollout and nonuniform time-stepping crucial for robust, real-world ROMs; this study shows that both enhancements strengthen state-of-the-art methods like GPLaSDI.

Acknowledgments and Disclosure of Funding

RS was supported by the Sydney Fernbach Postdoctoral Fellowship under LDRD number 25-ERD-049. YC was supported by the U.S. Department of Energy (DOE), Office of Science, Office

[5]During testing, this is the only FOM information used by the ROM.

of Advanced Scientific Computing Research (ASCR), through the CHaRMNET Mathematical Multifaceted Integrated Capability Center (MMICC) under Award Number DE-SC0023164 and the LEADS SciDAC Institute under Project Number SCW1933. 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. LLNL document release number: LLNL-CONF-2010452.

References

[1] Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, et al. Mfem: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.

[2] William Anderson, Seung Whan Chung, and Youngsoo Choi. mlasdi: Multi-stage latent space dynamics identification. *arXiv preprint arXiv:2506.09207*, 2025.

[3] Julian Andrej, Nabil Atallah, Jan-Phillip Bäcker, Jean-Sylvain Camier, Dylan Copeland, Veselin Dobrev, Yohann Dudouit, Tobias Duswald, Brendan Keith, Dohyun Kim, et al. High-performance finite elements with mfem. *The International Journal of High Performance Computing Applications*, 38(5):447–467, 2024.

[4] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[5] Christophe Bonneville, Youngsoo Choi, Debojyoti Ghosh, and Jonathan L Belof. Data-driven autoencoder numerical solver with uncertainty quantification for fast physical simulations. *arXiv preprint arXiv:2312.01021*, 2023.

[6] Christophe Bonneville, Youngsoo Choi, Debojyoti Ghosh, and Jonathan L Belof. GPLaSDI: Gaussian process-based interpretable latent space dynamics identification through deep autoencoder. *Computer Methods in Applied Mechanics and Engineering*, 418:116535, 2024.

[7] Christophe Bonneville, Xiaolong He, April Tran, Jun Sur Park, William Fries, Daniel A Messenger, Siu Wun Cheung, Yeonjong Shin, David M Bortz, Debojyoti Ghosh, et al. A comprehensive review of latent space dynamics identification algorithms for intrusive and non-intrusive reduced-order-modeling. *arXiv preprint arXiv:2403.10748*, 2024.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[9] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[10] Richard L Burden and J Douglas Faires. Numerical analysis, brooks, 1997.

[11] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

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

[13] Seung Whan Chung, Christopher Miller, Youngsoo Choi, Paul Tranquilli, H Keo Springer, and Kyle Sullivan. Latent space dynamics identification for interface tracking with application to shock-induced pore collapse. *arXiv preprint arXiv:2507.10647*, 2025.

[14] William D Fries, Xiaolong He, and Youngsoo Choi. LaSDI: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 399:115436, 2022.

[15] Alexander N Gorban and Ivan Yu Tyukin. Blessing of dimensionality: mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118):20170237, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Xiaolong He, Youngsoo Choi, William D Fries, Jonathan L Belof, and Jiun-Shyan Chen. gLaSDI: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 489:112267, 2023.

[18] Xiaolong He, April Tran, David M Bortz, and Youngsoo Choi. Physics-informed active learning with simultaneous weak-form latent space dynamics identification. *International Journal for Numerical Methods in Engineering*, 126(1):e7634, 2025.

[19] Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20. Dubrovnik, Croatia), 2007.

[20] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474 (2219):20180335, 2018.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[23] Kookjin Lee and Eric J Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A*, 477(2253): 20210162, 2021.

[24] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[25] Jun Sur Richard Park, Siu Wun Cheung, Youngsoo Choi, and Yeonjong Shin. tlasdi: Thermodynamics-informed latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 429:117144, 2024.

[26] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

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

[28] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[29] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[30] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[31] Jos Thijssen. *Computational physics*. Cambridge university press, 2007.

[32] April Tran, Xiaolong He, Daniel A Messenger, Youngsoo Choi, and David M Bortz. Weak-form latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 427:116998, 2024.

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

[34] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.