
Sequential decoder training for improved latent space dynamics identification

William Anderson

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94550
anderson316@llnl.gov

Seung Whan Chung

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94550
chung28@llnl.gov

Youngsoo Choi

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94550
choi15@llnl.gov

Abstract

Accurate numerical solutions of partial differential equations are essential in many scientific fields but often require computationally expensive solvers, motivating reduced-order models (ROMs). Latent Space Dynamics Identification (LaSDI) is a data-driven ROM framework that combines autoencoders with equation discovery to learn interpretable latent dynamics. However, enforcing latent dynamics during training can compromise reconstruction accuracy of the model for simulation data. We introduce multi-stage LaSDI (mLaSDI), a framework that improves reconstruction and prediction accuracy by sequentially learning additional decoders to correct residual errors from previous stages. Applied to the 1D-1V Vlasov equation, mLaSDI consistently outperforms standard LaSDI, achieving lower prediction errors and reduced training time across a wide range of architectures.

1 Introduction

Simulating time-dependent partial differential equations (PDEs) is central to advances in engineering [9, 18, 29], physics [38, 40], and biology [34], but high-fidelity methods are often computationally expensive. Reduced-order models (ROMs) address this issue by approximating the dynamics at far lower computational cost. Here, we develop ROMs for PDEs with parametric dependence affecting the initial conditions or underlying physics.

Projection-based ROMs [2, 4, 6, 19, 30, 13, 11, 33, 16, 12, 31, 14] are generally interpretable, but require knowledge of the underlying governing equations or source code. Non-intrusive ROMs are purely data-driven, but often lack interpretability. To address this gap, Fries et al. [20] introduced Latent Space Dynamics Identification (LaSDI), which combines an autoencoder with Sparse Identification of Nonlinear Dynamics (SINDy) [8] to learn interpretable dynamics in the compressed latent space. LaSDI generates ordinary differential equations (ODEs) for each training input parameter and interpolates these ODEs for unseen input parameters.

Despite many improvements and variations [5, 21, 22, 35, 39, 24, 15, 23, 7], LaSDI requires autoencoders to balance accurate data reconstruction with enforcing prescribed latent dynamics. This compromise can lead to poor reconstruction and prediction accuracy. Additionally, training remains costly due to large autoencoders and extensive hyperparameter tuning. While multi-stage

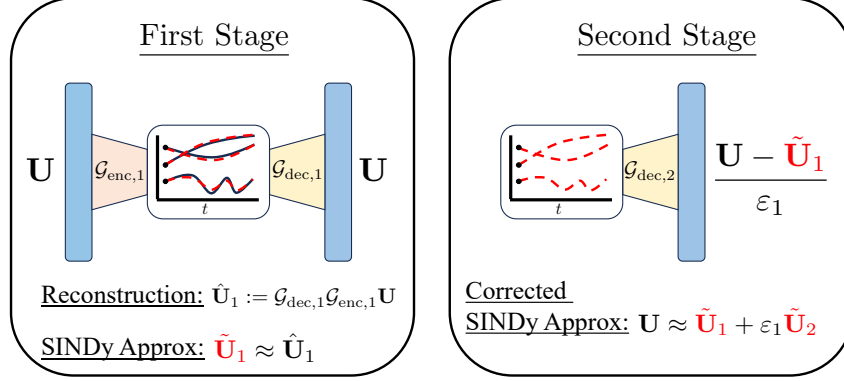


Figure 1: Schematic of mLaSDI. The first stage learns an autoencoder which reconstructs the training data \mathbf{U} , and latent space trajectories (solid black lines) are approximated using SINDy (dashed red lines). The second stage learns a new decoder which maps the SINDy trajectories in the latent space to the normalized residual error from the first stage.

architectures exist in other contexts [1, 42, 25, 41, 43, 44], none of these methods address the interpretability-accuracy trade-off inherent to LaSDI and other models with an interpretable latent space [3, 10, 17, 27, 37].

To overcome these limitations, we introduce **multi-stage Latent Space Dynamics Identification (mLaSDI)**, a general framework that improves any LaSDI variant by sequentially reducing residual errors. Each additional stage learns a decoder which maps latent space trajectories to the residual error of previous stages, providing additional representation power while inheriting the interpretable latent space from the first stage. This allows us to significantly improve reconstruction and prediction accuracy, reducing the need for large autoencoders or increasing model robustness to hyperparameters.

Source code for this project is based on Gaussian Process-based LaSDI (GPLaSDI): <https://github.com/LLNL/GPLaSDI>.

2 Multi-stage latent space dynamics identification

Although mLaSDI is applicable to any variant of LaSDI, here we focus on comparisons to GPLaSDI. We briefly describe the GPLaSDI algorithm and refer the reader to [5] for full details.

GPLaSDI We consider dynamical systems of the form

$$\frac{d}{dt}\mathbf{u}(t; \boldsymbol{\mu}) = \mathbf{f}(\mathbf{u}), \quad \mathbf{u}(0; \boldsymbol{\mu}) = \mathbf{u}_0(\boldsymbol{\mu}), \quad (1)$$

where $\mathbf{u} : \mathbb{R}^+ \rightarrow \mathbb{R}^{N_u}$ is the state vector, $\mathbf{f} : \mathbb{R}^{N_u} \rightarrow \mathbb{R}^{N_u}$ is a vector-valued function, and $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^N$ is an input parameter affecting physics of the simulation or the initial condition. We assume no knowledge of the underlying dynamical system, and only consider snapshots of the state vector \mathbf{u} taken at uniform times t_i , $i = 0, 1, \dots, N_t$, for each of the N_μ training parameters. Given an input parameter $\boldsymbol{\mu}^{(i)}$, we form snapshots of the state vector into the training data matrix $U^{(i)} = [\mathbf{u}^{(i)}(0), \mathbf{u}^{(i)}(t_1), \dots, \mathbf{u}^{(i)}(N_t)]$. Concatenating the snapshots from each of our N_μ training parameters, we obtain the training data tensor $\mathbf{U} = [U^{(1)}, U^{(2)}, \dots, U^{(N_\mu)}] \in \mathbb{R}^{N_\mu \times (N_t+1) \times N_u}$.

After obtaining our snapshots, an autoencoder compresses the spatial dimension of the training data through encoder and decoder mappings $\mathcal{G}_{\text{dec}} : \mathbb{R}^{N_u} \rightarrow \mathbb{R}^{N_z}$ and $\mathcal{G}_{\text{enc}} : \mathbb{R}^{N_z} \rightarrow \mathbb{R}^{N_u}$, where $N_z \ll N_u$. The autoencoder reconstruction of the training data is defined by $\hat{\mathbf{U}} = \mathcal{G}_{\text{dec}} \mathcal{G}_{\text{enc}} \mathbf{U}$, where the encoder and decoder minimize the training reconstruction loss $\mathcal{L}_{\text{AE}}(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}) = \|\mathbf{U} - \hat{\mathbf{U}}\|^2$. Here $\|\cdot\|$ is the element-wise ℓ^2 -norm and $\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}$ are model parameters of the encoder and decoder.

With the goal of obtaining an interpretable latent space, we consider the compressed data tensor $\mathbf{Z} = \mathcal{G}_{\text{enc}} \mathbf{U} \in \mathbb{R}^{N_\mu \times (N_t+1) \times N_z}$. For the snapshots of each compressed simulation $Z^{(i)}$, we follow

Table 1: Model architectures used to train GPLaSDI and mLaSDI for 1D-1V Vlasov experiments.

Component	Choices	Description
mLaSDI Hidden Layers	50, 500, 1000, 500-50, 1000-500-50	Fully connected layers
GPLaSDI Hidden Layers	75, 750, 1500, 750-75, 1500-750-75	Fully connected layers
Latent Dimension	4, 5, 6, 7	Bottleneck dimension
GPLaSDI Training Config	25k, 50k, 75k, 100k	Training checkpoints
mLaSDI Training Config.	(25k,25k), (25k,50k), (50k,50k), (75k,25k)	(Stage 1, Stage 2) iter. pairs

[5, 21] and apply SINDy [8] to approximate dynamics of our latent space by

$$\dot{Z}^{(i)} \approx \dot{\hat{Z}}^{(i)} := \Theta(Z^{(i)})\Xi^{(i)}, \quad \Theta(Z^{(i)}) = (\mathbf{1} (Z^{(i)})^\top), \quad \Xi^{(i)} = (\mathbf{b}^{(i)} A^{(i)})^\top, \quad (2)$$

where we must learn the coefficients $\mathbf{b}^{(i)} \in \mathbb{R}^{N_z}$, and $A^{(i)} \in \mathbb{R}^{N_z \times N_z}$. Defining the the SINDy coefficient tensor $\Xi = [\Xi^{(1)} \Xi^{(2)} \dots \Xi^{(N_\mu)}] \in \mathbb{R}^{N_\mu \times (N_z+1) \times N_z}$, we determine our coefficients through minimizing the dynamics identification loss $\mathcal{L}_{\text{DI}}(\Xi) = \|\dot{\mathbf{Z}} - \dot{\hat{\mathbf{Z}}}\|^2$. Additionally, we penalize the norm of our SINDy coefficients to obtain the loss function for GPLaSDI

$$\mathcal{L}(\theta_{\text{enc}}, \theta_{\text{dec}}, \Xi) = \mathcal{L}_{\text{AE}}(\theta_{\text{enc}}, \theta_{\text{dec}}) + \beta_1 \mathcal{L}_{\text{DI}}(\Xi) + \beta_2 \|\Xi\|^2, \quad (3)$$

where β_1, β_2 are hyperparameters.

After training our model, we use the GP interpolation scheme of GPLaSDI [5] to interpolate SINDy coefficients $\Xi^{(*)}$ for a new test parameter $\mu^{(*)}$. Essentially, we perform Gaussian Process Regression on each of the SINDy coefficients in the training set to find $\mathcal{GP}_\Xi : \mu^{(*)} \mapsto \{m(\Xi^{(*)}), (\Xi^{(*)})\}$, where m is the predictive mean of $\Xi^{(*)}$ and s the predictive standard deviation. This allows us to obtain a mean prediction, but we can also sample the GP to get uncertainty for our predictions.

mLaSDI The key limitation of LaSDI is that training with loss function (3) limits the representation power of our autoencoders. In some cases, autoencoders trained by LaSDI fail to accurately reconstruct even the training data due to constraints imposed by our latent dynamics. This is the main motivation for mLaSDI, where we propose training additional decoders to improve reconstruction accuracy when one stage of LaSDI fails to provide acceptable accuracy. Decoders introduced after the first stage are not required to learn additional SINDy coefficients, but provide additional representation power while still inheriting the interpretable latent space from the first stage of LaSDI.

For training data \mathbf{U} , we obtain the autoencoder reconstruction of the data from the first stage $\hat{\mathbf{U}}_1$. We can also obtain SINDy reconstructions of the training data by decoding our SINDy approximations of the latent space dynamics, i.e. $\tilde{\mathbf{U}}_1 := \mathcal{G}_{\text{dec},1}(\hat{\mathbf{Z}})$, where the subscript refers to the fact that this is our first stage of training and $\hat{\mathbf{Z}}$ is our SINDy approximation of the latent trajectories \mathbf{Z} .

To improve our SINDy reconstruction accuracy, we consider the residual between training data \mathbf{U} and the first stage SINDy reconstructions of the training data $\tilde{\mathbf{U}}_1$, given by $\mathbf{R}_1 = \mathbf{U} - \tilde{\mathbf{U}}_1$. With mLaSDI, we introduce a second decoder which takes the first stage SINDy latent space trajectories as input, and attempts to reconstruct the (normalized) residual error. More precisely, we fix model parameters of the first autoencoder, and learn a mapping for the second stage $\mathcal{G}_{\text{dec},2} : \mathbb{R}^{N_z} \rightarrow \mathbb{R}^{N_u}$ by minimizing

$$\mathcal{L}_{\text{dec},2}(\theta_{\text{dec},2}) = \|\varepsilon_1^{-1} \mathbf{R}_1 - \mathcal{G}_{\text{dec},2} \hat{\mathbf{Z}}\|^2, \quad \varepsilon_1 = \text{std}(\mathbf{R}_1). \quad (4)$$

With this approach we can sequentially train multiple models to approximate the training data \mathbf{U} as

$$\mathbf{U} \approx \tilde{\mathbf{U}}_1 + \varepsilon_1 \tilde{\mathbf{U}}_2 + \varepsilon_2 \tilde{\mathbf{U}}_3 + \dots + \varepsilon_{n-1} \tilde{\mathbf{U}}_n, \quad (5)$$

where $\tilde{\mathbf{U}}_k$ is the residual approximation at the k^{th} stage $\tilde{\mathbf{U}}_k = \mathcal{G}_{\text{dec},k} \hat{\mathbf{Z}}$, and ε_k normalizes the k^{th} residual to have standard deviation 1. A schematic of mLaSDI describing this process is provided in Figure 1.

We are free to choose any architecture for the decoders learned after the first stage. Here, we take an approach similar to Wang and Lai [42] and use identical decoder architectures at all stages, varying only the activation functions. After the first stage, we use sine activation on the first layer which helps to capture high-frequency components of the residual. Any subsequent layers use hyperbolic tangent activation function.

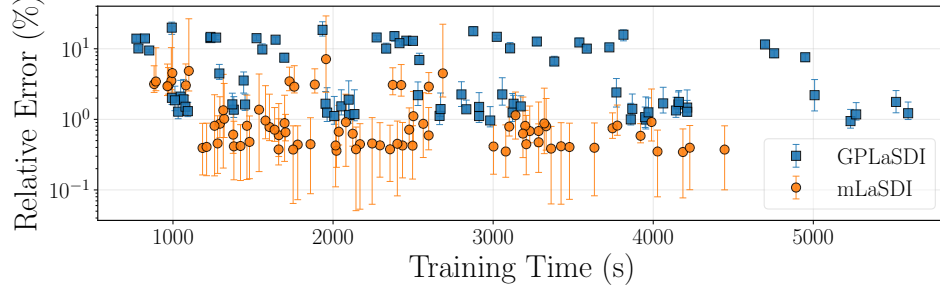


Figure 2: Applying GPLaSDI and mLaSDI with two stages to 1D-1V Vlasov equation using a wide range of model architectures in Table 1. Solid shapes indicate the mean relative error from (7). Error bars indicate the maximum and minimum relative error achieved across the parameter space.

3 Results and discussion

We compare mLaSDI and GPLaSDI on the 1D-1V Vlasov equation

$$\begin{cases} u_t + (vu)_x + (\Phi_x u)_v = 0, & t \in (0, 5], x \in [0, 2\pi], v \in [-7, 7], \\ \Phi_{xx} = \int_v u \, dv, \\ u(x, v, 0; \boldsymbol{\mu}) = \frac{8}{\sqrt{2\pi T}} [1 + 0.1 \cos(kx)] \left[\exp\left(-\frac{(v-2)^2}{2T}\right) + \exp\left(-\frac{(v+2)^2}{2T}\right) \right], \end{cases} \quad (6)$$

where $u(x, v, t)$ is the plasma distribution function which depends on space x and velocity v . The function $\Phi(x)$ is the electrostatic potential. The input parameter vector $\boldsymbol{\mu} = [T, k]^\top$ controls the spread T and periodicity k of the initial two-stream configuration.

Data Generation We solve (6) using HyPar [26] with a WENO spatial discretization scheme [28] and RK4 time integration scheme with timestep $\Delta t = 0.005$. We run full-order simulations for parameter values $T \in [0.9, 1.1]$ and $k \in [1.0, 1.2]$, where the parameter ranges are discretized by $\Delta T = \Delta k = 0.01$. We sample the solution at every timestep on a 64×64 grid in the space-velocity field to obtain 251 snapshots with state dimension $N_u = 4096$.

Training Details We train GPLaSDI and two stages of mLaSDI, varying the number of training iterations, hidden layers, and bottleneck dimension for each experiment, as summarized in Table 1. The hidden layers for GPLaSDI were widened by a factor 1.5 compared to mLaSDI, demonstrating that the multi-stage training achieves superior performance with fewer model parameters. For each experiment we uniformly sample 25 points from the 2D parameter space and predict on the full set of 441 simulations. All models are implemented in Pytorch and trained on a NVIDIA V100 (Volta) GPU using the Adam optimizer [32] with a learning rate of 10^{-3} . The mLaSDI loss weights are set to $\beta_1 = 0.1$, $\beta_2 = 0.001$. SINDy coefficients are interpolated via scikit-learn’s GaussianProcessRegressor with a Matérn kernel [36], and we only consider the mean predictions of the GPs.

Results Our error metric is the maximum relative error between a full-order simulation $\mathbf{u}(t; \boldsymbol{\mu}^{(*)})$ and the mLaSDI approximation $\tilde{\mathbf{u}}(t; \boldsymbol{\mu}^{(*)})$, defined as

$$r^{(*)} := \max_{j=0, \dots, N_t} \frac{\|\mathbf{u}(t_j; \boldsymbol{\mu}^{(*)}) - \tilde{\mathbf{u}}(t_j; \boldsymbol{\mu}^{(*)})\|}{\|\mathbf{u}(t_j; \boldsymbol{\mu}^{(*)})\|}. \quad (7)$$

We then calculate the maximum relative errors for every parameter value $\{r^{(i)}\}_{i=1}^{N_\mu}$ in our test set. In Figure 2 we plot the errors from each model along with training time. The reported training times for the mLaSDI represent the total time to train both the first and second stages.

We see that mLaSDI consistently produces lower relative errors than GPLaSDI models that have more model parameters. Introduction of a second stage also allows us to achieve maximum relative errors below 1% for some architectures, which we never achieve by training only one stage of GPLaSDI. Across all tested models, mLaSDI provides a median increase in accuracy by a factor

of 2.54, 3.24, and 5.91 for the maximum, mean, and minimum errors, respectively. These results demonstrate that mLaSDI maintains high accuracy across a wide range of model choices, reducing sensitivity to hyperparameters while improving prediction accuracy. Consistent improvement in accuracy across different architectures demonstrates that mLaSDI’s multi-stage approach addresses a fundamental limitation in LaSDI methods more effectively than architectural modifications alone. Our results suggest that the interpretability-accuracy trade-off is better resolved through training methodology rather than model scaling. This framework opens promising directions for enhancing other interpretable latent space methods and warrants investigation across broader PDE families.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Lawrence Livermore National Laboratory (LLNL) under Project No. 50284. Y. Choi was also supported for this work 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. 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-2010449.

References

- [1] Z. Aldirany, R. Cottreau, M. Laforest, and S. Prudhomme. Multi-level neural networks for accurate solutions of boundary-value problems. *Computer Methods in Applied Mechanics and Engineering*, 419:116666, 2024. ISSN 0045-7825. doi: <https://doi.org/10.1016/j.cma.2023.116666>.
- [2] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015. doi: [10.1137/130932715](https://doi.org/10.1137/130932715).
- [3] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, and K. Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Computer Methods in Applied Mechanics and Engineering*, 372:113433, 2020.
- [4] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.
- [5] C. Bonneville, Y. Choi, D. Ghosh, and J.L. Belof. Gpladi: Gaussian process-based interpretable latent space dynamics identification through deep autoencoder. *Computer Methods in Applied Mechanics and Engineering*, 418:116535, 2024. ISSN 0045-7825. doi: <https://doi.org/10.1016/j.cma.2023.116535>.
- [6] C. Bonneville 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.
- [7] Aaron L Brown, Eric B Chin, Youngsoo Choi, Saad A Khairallah, and Joseph T McKeown. A data-driven, non-linear, parameterized reduced order model of metal 3d printing. *arXiv preprint arXiv:2311.18036*, 2023.
- [8] S.L. Brunton, J.L. Proctor, and J.N. 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. doi: [10.1073/pnas.1517384113](https://doi.org/10.1073/pnas.1517384113).
- [9] M. Calder, C. Craig, D. Culley, R. de Cani, C. A. Donnelly, R. Douglas, B. Edmonds, J. Gascoigne, N. Gilbert, C. Hargrove, F. Hinds, D. C. Lane, D. Mitchell, G. Pavay, D. Robertson, B. Rosewell, S. Sherwin, M. Walport, and A. Wilson. Computational modelling for decision-making: where, why, what, who and how. *Royal Society Open Science*, 5(6):172096, 2018. doi: [10.1098/rsos.172096](https://doi.org/10.1098/rsos.172096).
- [10] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

- [11] Youngsoo Choi, Gabriele Boncoraglio, Spenser Anderson, David Amsallem, and Charbel Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, 423:109787, 2020.
- [12] 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.
- [13] Youngsoo Choi, Peter Brown, William Arrighi, Robert Anderson, and Kevin Huynh. Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems. *Journal of Computational Physics*, 424:109845, 2021.
- [14] Seung Whan Chung, Youngsoo Choi, Pratanu Roy, Thomas Moore, Thomas Roy, Tiras Y Lin, Du T Nguyen, Christopher Hahn, Eric B Duoss, and Sarah E Baker. Train small, model big: Scalable physics simulators via reduced order modeling and domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 427:117041, 2024.
- [15] 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.
- [16] 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.
- [17] M. Cranmer et al. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020.
- [18] R. M. Cummings, W. H. Mason, S. A. Morton, and D. R. McDaniel. *Applied computational aerodynamics: A modern engineering approach*, volume 53. Cambridge University Press, 2015.
- [19] A. N. Diaz, Y. Choi, and M. Heinkenschloss. A fast and accurate domain decomposition nonlinear manifold reduced order model. *Computer Methods in Applied Mechanics and Engineering*, 425:116943, 2024.
- [20] W.D. Fries, X. He, and Y. Choi. Lasdi: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 399:115436, 2022. ISSN 0045-7825. doi: <https://doi.org/10.1016/j.cma.2022.115436>.
- [21] X. He, Y. Choi, W.D. Fries, J.L. Belof, and J. Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 489:112267, 2023. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2023.112267>.
- [22] X. He, A. Tran, D.M. Bortz, and Y. Choi. Physics-informed active learning with simultaneous weak-form latent space dynamics identification. *International Journal for Numerical Methods in Engineering*, 126(1):e7634, 2025. doi: <https://doi.org/10.1002/nme.7634>.
- [23] Xiaolong He, Youngsoo Choi, William D Fries, Jonathan L Belof, and Jiun-Shyan Chen. Certified data-driven physics-informed greedy auto-encoder simulator. *arXiv preprint arXiv:2211.13698*, 2022.
- [24] Xiaolong He, Yeonjong Shin, Anthony Gruber, Sohyeon Jung, Kookjin Lee, and Youngsoo Choi. Thermodynamically consistent latent dynamics identification for parametric systems. *arXiv preprint arXiv:2506.08475*, 2025.
- [25] A.A. Howard, S.H. Murphy, S.E. Ahmed, and P. Stinis. Stacked networks improve physics-informed training: Applications to neural networks and deep operator networks. *Foundations of Data Science*, 7(1):134–162, 2025. doi: 10.3934/fods.2024029.
- [26] HyPar. HyPar repository. <https://github.com/debog/hypar>.
- [27] O. Issan and B. Kramer. Predicting solar wind streams from the inner-heliosphere to earth via shifted operator inference. *arXiv preprint arXiv:2203.13372*, 2022.
- [28] G.S. Jiang and C.W. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. doi: 10.1006/jcph.1996.0130.
- [29] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29: 36–52, 2020. ISSN 1755-5817. doi: <https://doi.org/10.1016/j.cirpj.2020.02.002>.

- [30] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.
- [31] 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.
- [32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [33] Sean McBane and Youngsoo Choi. Component-wise reduced order model lattice-type structure design. *Computer methods in applied mechanics and engineering*, 381:113813, 2021.
- [34] D. Noble. The rise of computational biology. *Nature Reviews Molecular Cell Biology*, 3(6):459–463, 2002.
- [35] J. Sur R. Park, S.W. Cheung, Y. Choi, and Yeonjong Shin. tlasdi: Thermodynamics-informed latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 429:117144, 2024. ISSN 0045-7825. doi: <https://doi.org/10.1016/j.cma.2024.117144>.
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, November 2011. ISSN 1532-4435.
- [37] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.
- [38] J. Thijssen. *Computational physics*. Cambridge university press, 2007.
- [39] A. Tran, X. He, D. A. Messenger, Y. Choi, and D. M. Bortz. Weak-form latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 427:116998, Jul 2024.
- [40] D. Vasileska, S. M. Goodnick, and G. Klimeck. *Computational Electronics: semiclassical and quantum device modeling and simulation*. CRC press, 2017.
- [41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010.
- [42] Y. Wang and C. Lai. Multi-stage neural networks: Function approximator of machine precision. *Journal of Computational Physics*, 504:112865, 2024. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2024.112865>.
- [43] Yang Yu, Yong Ma, Xiaoguang Mei, Fan Fan, Jun Huang, and Hao Li. Multi-stage convolutional autoencoder network for hyperspectral unmixing. *International Journal of Applied Earth Observation and Geoinformation*, 113:102981, 2022. ISSN 1569-8432. doi: <https://doi.org/10.1016/j.jag.2022.102981>.
- [44] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, and S. Marshall. Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, 185:1–10, 2016. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2015.11.044>.