Deep Conservation: A Latent-Dynamics Model for Exact Satisfaction of Physical Conservation Laws

Kookjin Lee[1], **Kevin T. Carlberg**[2]
[1]Sandia National Laboratories, [2]University of Washington
koolee@sandia.gov, ktcarlb@uw.edu

Abstract

This work proposes an approach for latent-dynamics learning that exactly enforces physical conservation laws. The method comprises two steps. First, the method computes a low-dimensional embedding of the high-dimensional dynamical-system state using deep convolutional autoencoders. This defines a low-dimensional nonlinear manifold on which the state is subsequently enforced to evolve. Second, the method defines a latent-dynamics model that associates with the solution to a constrained optimization problem. Here, the objective function is defined as the sum of squares of conservation-law violations over control volumes within a finite-volume discretization of the problem; nonlinear equality constraints explicitly enforce conservation over prescribed subdomains of the problem. Under modest conditions, the resulting dynamics model guarantees that the time-evolution of the latent state exactly satisfies conservation laws over the prescribed subdomains.

Introduction

Learning a latent-dynamics model for complex real-world physical processes (e.g., fluid dynamics (Morton et al. 2018; Wiewel, Becher, and Thuerey 2019; Lee and Carlberg 2020), deformable solid mechanics (Fulton et al. 2019)) comprises an important task in science and engineering, as it provides a mechanism for modeling the dynamics of physical systems and can provide a rapid simulation tool for time-critical applications such as control and design optimization. Two main ingredients are required to learn a latent-dynamics model: (1) an *embedding*, which provides a mapping between high-dimensional dynamical-system states and low-dimensional latent variables, and (2) a *dynamics model*, which prescribes the time evolution of the latent variables in the latent space.

There are two primary classes of methods for learning a latent-dynamics model. The first class comprises *data-driven dynamics learning*, which aims to learn both the embedding and the dynamics model in a purely data-driven manner that requires only measurements of the state/velocity. As such, this class of methods does not require *a priori* knowledge of the system of ordinary differential equations (ODEs) governing the high-dimensional dynamical system. These methods traditionally learn a nonlinear embedding (e.g., via autoencoders (Lusch, Kutz, and Brunton 2018; Morton et al. 2018; Otto and Rowley 2019; Takeishi, Kawahara, and Yairi 2017)), and—inspired by Koopman operator theory—learn a dynamics model that is constrained to be linear. In a control (Lesort et al. 2018) or reinforcement-learning context (Böhmer et al. 2015), the embedding and locally linear dynamics models can be learned simultaneously from observations of the state (Goroshin, Mathieu, and LeCun 2015; Karl et al. 2017; Watter et al. 2015; Banijamali et al. 2018). More recently, deep-learning-based techniques for constructing nonlinear embeddings and nonlinear dynamics, e.g., via long short-term memory (Hochreiter and Schmidhuber 1997), neural ordinary differential equations (Chen et al. 2018), have been proposed (Gonzalez and Balajewicz 2018; Maulik, Lusch, and Balaprakash 2020; Portwood et al. 2019).

The second class of methods corresponds to *projection-based dynamics learning*, which learns the embedding in a data-driven manner, but computes the dynamics model via a projection process executed on the governing system of ODEs, which must be known *a priori*. As opposed to the data-driven dynamics learning, projection-based methods almost always employ a linear embedding, which is typically defined by principal component analysis (or "proper orthogonal decomposition" (Holmes et al. 2012)) performed on measurements of the state. The projection process that produces the latent-dynamics model requires substituting the linear embedding in the governing ODEs and enforcing orthogonality of the resulting residual to a low-dimensional linear subspace (Benner, Gugercin, and Willcox 2015), yielding a (Petrov–) Galerkin projection formulation.

Each approach suffers from particular shortcomings. Because data-driven dynamics learning lacks explicit *a priori* knowledge of the governing ODEs—and thus predicts latent dynamics separately from any computational-physics code—these methods risk severe violation of physical principles underpinning the dynamical system. On the other hand, projection-based dynamics-learning methods heavily rely on linear embeddings and, thus, exhibit limited dimensionality reduction compared with what is achievable with nonlinear embeddings (Ohlberger and Rave 2016). Recently, this limitation has been resolved by employing a nonlinear embedding (learned by deep convolutional autoenoders) and projecting the governing ODEs onto the resulting low-dimensional nonlinear manifold (Lee and Carlberg 2020). Another shortcoming of many projection-based dynamics-learning methods is that the (Petrov–)Galerkin projection process that they employ does not preclude the violation of important physical properties such as conservation. To mitigate this issue, a recent work has proposed a projection technique that explicitly enforces conservation over subdomains by adopting a constrained least-squares formulation to define the projection (Carlberg, Choi, and Sargsyan 2018).

In this study, we consider problems characterized by physical conservation laws. Such problems are ubiquitous in science and engineering.[1] For such problems, we propose *Deep Conservation: a projection-based dynamics learning method* that combines the advantages of Refs. (Lee and Carlberg 2020) and (Carlberg, Choi, and Sargsyan 2018), as the method (1) learns a nonlinear embedding via deep convolutional autoencoders, and (2) defines a dynamics model via a projection process that explicitly enforces conservation over subdomains. The method assumes explicit *a priori* knowledge of the ODEs governing the conservations laws in integral form, and an associated finite-volume discretization. In contrast to existing methods for latent-dynamics learning, this is the only method that both employs a nonlinear embedding and computes nonlinear dynamics for the latent state in a manner that guarantees the satisfaction of prescribed physical properties. Moreover, we propose a *physics-informed training objective* for training the autoencoder, which enforces physical conservation laws directly into the autoencoder during the training. Numerical experiments on a benchmark advection problem illustrate the method's ability to drastically reduce the dimensionality while successfully enforcing the physical conservation laws.

Relatedly, deep-learning-based approaches for enforcing conservations laws include (1) designing neural networks that can learn arbitrary conservation laws (hyperbolic conservation laws (Raissi, Perdikaris, and Karniadakis 2019), Hamiltonian dynamics (Greydanus, Dzamba, and Yosinski 2019; Toth et al. 2019; Chen et al. 2019), Lagrangian dynamics (Cranmer et al. 2020)), or (2) designing a loss function or adding an extra neural network constraining linear conservations laws (Beucler et al. 2019). These approaches, however, approximate states in (semi-)supervised-learning settings and the resulting approximations do not guarantee exact satisfaction of conservation laws. Instead, the proposed latent-dynamics model associates with the approximate states with the solution to a constrained residual minimization problem, and guarantees the exact satisfaction of conservation laws over the prescribed subdomains under modest conditions.

[1]In physics, conservation laws state that certain physical quantities of an isolated physical system do not change over time. In fluid dynamics, for example, the Euler equations (LeVeque 2002) governing inviscid flow are a set of equations representing the conservation of mass, momentum, and energy of the fluid.

Full-order Model

Physical Conservation Laws

This work considers parameterized systems of *physical conservation laws*. The equations governing a system of conservations laws in integral form correspond to

$$\frac{d}{dt}\int_{\omega} w_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x} + \int_{\gamma} \boldsymbol{g}_i(\vec{x},t;\boldsymbol{\mu})\cdot\boldsymbol{n}(\vec{x})\,\mathrm{d}\vec{s}(\vec{x}) = \int_{\omega} s_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x}, \tag{1}$$

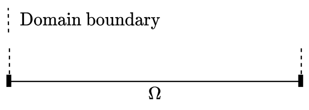
for $i = \{1,\ldots,n_w\}$, which is solved in time domain $t \in [0,T]$ given an initial condition such that $w_i(\vec{x},0;\boldsymbol{\mu}) = w_i^0(\vec{x};\boldsymbol{\mu})$, $i = \{1,\ldots,n_w\}$. Here, ω denotes any subset of the spatial domain $\Omega \subset \mathbb{R}^d$ with $d \leq 3$; $\gamma := \partial\omega$ denotes the boundary of the subset ω, while $\Gamma := \partial\Omega$ denotes the boundary of the domain Ω; $\mathrm{d}\vec{s}(\vec{x})$ denotes integration with respect to the boundary; and $w_i \in \mathbb{R}$, $\boldsymbol{g}_i \in \mathbb{R}^d$, and $s_i \in \mathbb{R}$ denote the ith conserved variable, the flux associated with w_i, and the source associated with w_i. The parameters $\boldsymbol{\mu} \in \mathcal{D}$ characterize physical properties of the governing equations, where $\mathcal{D} \subset \mathbb{R}^{n_\mu}$ denotes the parameter space. Finally, $\boldsymbol{n} \in \mathbb{R}^d$ denotes the outward unit normal to ω. **We emphasize that Eq.** (1) **describe conservation of *any* set of variables w_i (e.g., mass, momentum, and energy), given their respective flux $\boldsymbol{g}_i$ and source s_i functions**, e.g., without source term, the rate of change of mass w inside a certain region ω, $\frac{d}{dt}\int_\omega w$, equals the incoming flux (the flux entering minus the flux leaving the domain ω) $-\int_\gamma \boldsymbol{g}(\vec{x},t;\boldsymbol{\mu})\cdot\boldsymbol{n}(\vec{x})\,\mathrm{d}\vec{s}(\vec{x})$.

Finite-volume Discretization

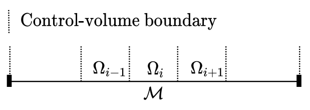
To discretize the governing equations (1), we apply the finite-volume method (LeVeque 2002), as it enforces conservation numerically by decomposing the spatial domain into many control volumes, numerically approximating the sources and fluxes, and then enforcing conservation (Eq. (1)) over the control volumes using the approximated quantities. In particular, we assume that the spatial domain Ω has been partitioned into a mesh $\mathcal{M}$ of $N_\Omega \in \mathbb{N}$ non-overlapping (closed, connected) control volumes $\Omega_i \subseteq \Omega$, $i = 1,\ldots,N_\Omega$. We define the mesh as $\mathcal{M} := \{\Omega_i\}_{i=1}^{N_\Omega}$, and denote the boundary of the ith control volume by $\Gamma_i := \partial\Omega_i$. Figure 1 depicts a one-dimensional spatial domain and a finite-volume mesh.

Enforcing conservation (1) on each control volume yields

$$\frac{d}{dt}\int_{\Omega_j} w_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x} + \int_{\Gamma_j} \boldsymbol{g}_i(\vec{x},t;\boldsymbol{\mu})\cdot\boldsymbol{n}_j(\vec{x})\,\mathrm{d}\vec{s}(\vec{x}) = \int_{\Omega_j} s_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x}, \tag{2}$$



(a) 1D spatial domain



(b) Finite-volume mesh

Figure 1: An example one-dimensional spatial domain Ω and an example finite-volume mesh $\mathcal{M} = \{\Omega_i\}_{i=1}^{N_\Omega}$. The conservation laws are enforced in each control volume Ω_i.

for the ith conserved variable on the j control volume, where $\boldsymbol{n}_j \in \mathbb{R}^d$ denotes the unit normal to control volume Ω_j. Finite-volume schemes complete the discretization by forming a state vector $\boldsymbol{x} \in \mathbb{R}^N$ with $N = N_\Omega n_w$ such that

$$x_{\mathcal{I}(i,j)}(t;\boldsymbol{\mu}) = \frac{1}{|\Omega_j|}\int_{\Omega_j} w_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x}, \tag{3}$$

where $\mathcal{I}$ denotes a mapping from conservation-law index and control-volume index to degree of freedom, and a velocity vector $\boldsymbol{f} : (\boldsymbol{\xi}, \tau; \boldsymbol{\nu}) \mapsto \boldsymbol{f}^g(\boldsymbol{\xi}, \tau; \boldsymbol{\nu}) + \boldsymbol{f}^s(\boldsymbol{\xi}, \tau; \boldsymbol{\nu})$ with $\boldsymbol{f}^g, \boldsymbol{f}^s \in \mathbb{R}^N$ whose elements consist of

$$f^g_{\mathcal{I}(i,j)}(\boldsymbol{x},t;\boldsymbol{\mu}) = -\frac{1}{|\Omega_j|}\int_{\Gamma_j} \boldsymbol{g}_i^{\mathrm{FV}}(\boldsymbol{x};\vec{x},t;\boldsymbol{\mu})\cdot\boldsymbol{n}_j(\vec{x})\,\mathrm{d}\vec{s}(\vec{x}),$$
$$f^s_{\mathcal{I}(i,j)}(\boldsymbol{x},t;\boldsymbol{\mu}) = \frac{1}{|\Omega_j|}\int_{\Omega_j} s_i^{\mathrm{FV}}(\boldsymbol{x};\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x},$$

where $\boldsymbol{g}_i^{\mathrm{FV}} \in \mathbb{R}^d$ and $s_i^{\mathrm{FV}} \in \mathbb{R}$ denote the approximated flux and source, respectively, associated with the ith conserved variable. **The conservation laws are enforced in each control volume with the approximated flux and source**, e.g., the rate of change of the mass inside each control volume equals the incoming flux to that control volume.

Substituting $\int_{\Omega_j} w_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x} \leftarrow |\Omega_j| x_{\mathcal{I}(i,j)}(t;\boldsymbol{\mu})$, $\boldsymbol{g}_i \leftarrow \boldsymbol{g}_i^{\mathrm{FV}}$, and $s_i \leftarrow s_i^{\mathrm{FV}}$ in Eq. (2) and dividing by $|\Omega_j|$ yields

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},t;\boldsymbol{\mu}), \qquad \boldsymbol{x}(0;\boldsymbol{\mu}) = \boldsymbol{x}^0(\boldsymbol{\mu}), \tag{4}$$

where $x^0_{\mathcal{I}(i,j)}(\boldsymbol{\mu}) := \frac{1}{|\Omega_i|}\int_{\Omega_j} w_i^0(\vec{x};\boldsymbol{\mu})\,\mathrm{d}\vec{x}$ denotes the parameterized initial condition. This is a parameterized system of nonlinear ordinary differential equations (ODEs) characterizing an initial value problem, which is our full-order model (FOM). We refer to Eq. (4) as the FOM ODE.

Numerically solving the FOM ODE (4) requires application of a time-discretization method. For simplicity, this work restricts attention to linear multistep methods. A linear k-step method applied to numerically solve the FOM ODE (4) leads to solving the system of algebraic equations

$$\boldsymbol{r}^n(\boldsymbol{x}^n;\boldsymbol{\mu}) = \boldsymbol{0}, \quad n = 1,\ldots,N_t, \tag{5}$$

where the time-discrete residual $\boldsymbol{r}^n : \mathbb{R}^N \times \mathcal{D} \to \mathbb{R}^N$, as a function of $\boldsymbol{\xi}$ parameterized by $\boldsymbol{\nu}$, is defined as

$$\begin{aligned}\boldsymbol{r}^n : (\boldsymbol{\xi};\boldsymbol{\nu}) \mapsto & \alpha_0\boldsymbol{\xi} - \Delta t\beta_0\boldsymbol{f}(\boldsymbol{\xi},t^n;\boldsymbol{\nu}) + \sum_{j=1}^{k}\alpha_j\boldsymbol{x}^{n-j} \\ & - \Delta t\sum_{j=1}^{k}\beta_j\boldsymbol{f}(\boldsymbol{x}^{n-j},t^{n-j};\boldsymbol{\nu}).\end{aligned} \tag{6}$$

Here, $\Delta t \in \mathbb{R}_+$ denotes the time step, $\boldsymbol{x}^k$ denotes the numerical approximation to $\boldsymbol{x}(k\Delta t;\boldsymbol{\mu})$, and the coefficients α_j and β_j, $j = 0,\ldots,k$ with $\sum_{j=0}^{k}\alpha_j = 0$ define a particular linear multistep scheme. These methods are implicit if $\beta_0 \neq 0$. We refer to Eq. (5) as the FOM OΔE.

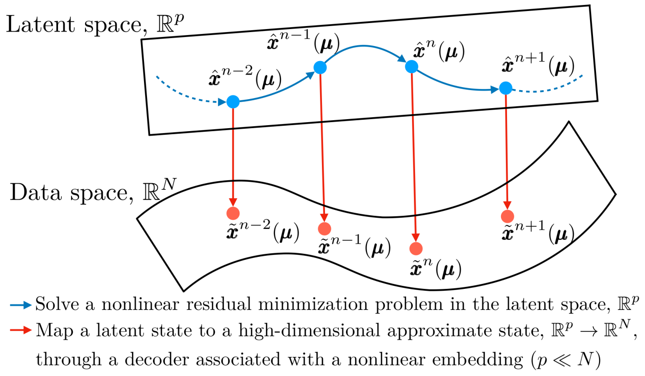

Latent space, $\mathbb{R}^p$
$\hat{\boldsymbol{x}}^{n-2}(\boldsymbol{\mu})$
$\hat{\boldsymbol{x}}^{n-1}(\boldsymbol{\mu})$
$\hat{\boldsymbol{x}}^{n}(\boldsymbol{\mu})$
$\hat{\boldsymbol{x}}^{n+1}(\boldsymbol{\mu})$
Data space, $\mathbb{R}^N$
$\tilde{\boldsymbol{x}}^{n-2}(\boldsymbol{\mu})$
$\tilde{\boldsymbol{x}}^{n-1}(\boldsymbol{\mu})$
$\tilde{\boldsymbol{x}}^{n}(\boldsymbol{\mu})$
$\tilde{\boldsymbol{x}}^{n+1}(\boldsymbol{\mu})$
Solve a nonlinear residual minimization problem in the latent space, $\mathbb{R}^p$
Map a latent state to a high-dimensional approximate state, $\mathbb{R}^p \to \mathbb{R}^N$, through a decoder associated with a nonlinear embedding ($p \ll N$)

Figure 2: Deep Conservation model – the second stage: a latent dynamics (blue arrows) and a decoder (red arrows, the decoder associated with a nonlinear embedding).

Computational Barrier: Time-critical Problems

Many problems in science and engineering are *time critical* in nature, meaning that the solution to the FOM OΔE (5) must be computed within a specified computational budget (e.g., less than 0.1 core–hours) for arbitrary parameter instances $\boldsymbol{\mu} \in \mathcal{D}$. When the FOM is truly high fidelity, the computational mesh $\mathcal{M}$ often becomes very fine, which can yield an extremely large state-space dimension N (e.g., $N \sim 10^7$). This introduces a *de facto* computational barrier: the FOM is too computationally expensive to solve within the prescribed computational budget. Such cases demand a method for *approximately* solving the FOM while retaining high levels of accuracy.

We now present a two-stage method that (1) computes a nonlinear embedding of the state using deep convolutional autoencoders, and (2) computes a dynamics model for latent states that exactly satisfies the physical conservation laws over *subdomains* comprising unions of control volumes of the mesh. Figure 2 depicts the second stage of the proposed method, where the latent space is identified by convolutional autoencoders during the first stage; the method computes latent states $\{\hat{\boldsymbol{x}}^n(\boldsymbol{\mu})\}_{n=1}^{N_t}$ via conservation-enforcing projection (Section Latent-dynamics model), and computes high-dimensional approximate states $\{\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})\}_{n=1}^{N_t}$ through a decoder associated with the nonlinear embedding (Section Nonlinear Embedding).

Nonlinear Embedding

Deep Convolutional Autoencoders

Autoencoders (DeMers and Cottrell 1993; Hinton and Salakhutdinov 2006) consist of two parts: an encoder $\boldsymbol{h}_{\mathrm{enc}}(\cdot;\boldsymbol{\theta}_{\mathrm{enc}}) : \mathbb{R}^N \to \mathbb{R}^p$ and a decoder $\boldsymbol{h}_{\mathrm{dec}}(\cdot;\boldsymbol{\theta}_{\mathrm{dec}}) : \mathbb{R}^p \to \mathbb{R}^N$ with latent-state dimension $p \ll N$ such that

$$\boldsymbol{h} : (\boldsymbol{x};\boldsymbol{\theta}_{\mathrm{enc}},\boldsymbol{\theta}_{\mathrm{dec}}) \mapsto \boldsymbol{h}_{\mathrm{dec}}(\cdot;\boldsymbol{\theta}_{\mathrm{dec}}) \circ \boldsymbol{h}_{\mathrm{enc}}(\boldsymbol{x};\boldsymbol{\theta}_{\mathrm{enc}}),$$

where $\boldsymbol{\theta}_{\mathrm{enc}}$ and $\boldsymbol{\theta}_{\mathrm{dec}}$ denote parameters associated with the encoder and decoder, respectively.

Because we are considering finite-volume discretizations of conservation laws, the state elements $x_{\mathcal{I}(i,j)}$, $j =$

$1, \ldots, N_\Omega$ correspond to the value of the ith conserved variable w_i distributed across the N_Ω control volumes characterizing the mesh $\mathcal{M}$. As such, we can interpret the state $\boldsymbol{x} \in \mathbb{R}^N$ as representing the distribution of spatially distributed data with n_w channels (i.e., $N = N_\Omega n_w$). This is precisely the format required by convolutional neural networks, which often generalize well to unseen test data (LeCun, Bengio, and Hinton 2015) because they exploit local connectivity, employ parameter sharing, and exhibit translation equivariance (Goodfellow et al. 2016; LeCun, Bengio, and Hinton 2015). Thus, we leverage the connection between conservation laws and image data, and employ convolutional autoencoders.

Offline Training

The first step of the offline training is snapshot-based data collection. This requires solving the FOM OΔE (5) for training-parameter instances $\boldsymbol{\mu} \in \mathcal{D}_{\text{train}} \equiv \{\boldsymbol{\mu}^i_{\text{train}}\}_{i=1}^{n_{\text{train}}} \subset \mathcal{D}$ and assembling the snapshot matrix,

$$\mathbf{X} := \left[\mathbf{X}(\boldsymbol{\mu}^1_{\text{train}}) \ \cdots \ \mathbf{X}(\boldsymbol{\mu}^{n_{\text{train}}}_{\text{train}})\right] \in \mathbb{R}^{N \times n_{\text{snap}}} \qquad (7)$$

with $n_{\text{snap}} := N_t n_{\text{train}}$ and $\mathbf{X}(\boldsymbol{\mu}) \equiv [\mathbf{x}^1(\boldsymbol{\mu}) \ \cdots \ \mathbf{x}^{N_t}(\boldsymbol{\mu})] := [\boldsymbol{x}^1(\boldsymbol{\mu}) - \boldsymbol{x}^0(\boldsymbol{\mu}) \ \cdots \ \boldsymbol{x}^{N_t}(\boldsymbol{\mu}) - \boldsymbol{x}^0(\boldsymbol{\mu})] \in \mathbb{R}^{N \times N_t}$.

To improve numerical stability of the gradient-based optimization for training, the first layer of the proposed autoencoder applies data standardization through an affine scaling operator $\boldsymbol{S}$, which ensures that all elements of the training data lie between zero and one. Then the autoencoder reformats the input vector into a tensor compatible with convolutional layers; the last layer applies the inverse scaling operator $\boldsymbol{S}^{-1}$ and reformats the data into a vector.

Given the network architecture $\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}})$, we compute parameter values $(\boldsymbol{\theta}^\star_{\text{enc}}, \boldsymbol{\theta}^\star_{\text{dec}})$ by approximately solving

$$\underset{\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}}{\text{minimize}} \sum_{i=1}^{n_{\text{snap}}} \|\mathbf{x}^i - \boldsymbol{h}(\mathbf{x}^i; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}})\|_2^2 \qquad (8)$$

using stochastic gradient descent (SGD) with minibatching and early stopping.

Along with this vanilla autoencoder, inspired by the formulation of physics-informed neural networks (Raissi, Perdikaris, and Karniadakis 2019), we devise a *physics-informed training constraint*, which is added to the optimization objective (8),

$$\rho \sum_{i=1}^{n_{\text{snap}}} \|\boldsymbol{r}^i(\boldsymbol{x}^0(\boldsymbol{\mu}) + \ \boldsymbol{S}^{-1}(\boldsymbol{h}(\mathbf{x}^i; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}); \boldsymbol{\mu}^i))\|_2^2,$$

which enforces minimization of the time-discrete residuals. The advantage of this approach is that it aligns the training objective more closely with the online objective (described in the next Section); the disadvantage is that evaluating the objective function requires evaluating the underlying finite-volume model, whereas the objective function (8) can be computed by accessing only the snapshot matrix.

Nonlinear Embedding

Given the trained autoencoder $\boldsymbol{h} : (\boldsymbol{x}; \boldsymbol{\theta}^\star_{\text{enc}}, \boldsymbol{\theta}^\star_{\text{dec}}) \mapsto \boldsymbol{h}_{\text{dec}}(\cdot; \boldsymbol{\theta}^\star_{\text{dec}}) \circ \boldsymbol{h}_{\text{enc}}(\boldsymbol{x}; \boldsymbol{\theta}^\star_{\text{enc}})$, we construct a nonlinear embedding by defining a low-dimensional nonlinear "trial manifold" on which we will restrict the approximated state to evolve. In particular, we define this manifold from the extrinsic view as $\mathcal{S} := \{\boldsymbol{d}(\hat{\boldsymbol{\xi}}) \,|\, \hat{\boldsymbol{\xi}} \in \mathbb{R}^p\}$, where the parameterization function is defined from the decoder as $\boldsymbol{d} : \hat{\boldsymbol{\xi}} \mapsto \boldsymbol{h}_{\text{dec}}(\hat{\boldsymbol{\xi}}; \boldsymbol{\theta}^\star_{\text{dec}})$ with $\boldsymbol{d} : \mathbb{R}^p \to \mathbb{R}^N$. We subsequently approximate the state as $\boldsymbol{x}(t; \boldsymbol{\mu}) \approx \tilde{\boldsymbol{x}}(t; \boldsymbol{\mu}) \in \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \mathcal{S}$, where $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \boldsymbol{x}^0(\boldsymbol{\mu}) - \boldsymbol{d}(\hat{\boldsymbol{x}}^0(\boldsymbol{\mu}))$ is the reference state. This approximation can be expressed algebraically as

$$\tilde{\boldsymbol{x}}(t; \boldsymbol{\mu}) = \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{x}}(t; \boldsymbol{\mu})), \qquad (9)$$

which elucidates the mapping between the latent state $\hat{\boldsymbol{x}} : \mathbb{R}_+ \times \mathcal{D} \to \mathbb{R}^p$ and the approximated state $\tilde{\boldsymbol{x}} : \mathbb{R}_+ \times \mathcal{D} \to \mathbb{R}^N$.

Remark 1 (Linear embedding via POD) *Classical methods for projection-based dynamics learning employ proper orthogonal decomposition (POD) (Holmes et al. 2012)—which is closely related to principal component analysis—to construct a linear embedding. Using the above notation, POD computes the singular value decomposition of the snapshot matrix* $\mathbf{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ *and sets a "trial basis matrix"* $\boldsymbol{\Phi} \in \mathbb{R}^{N \times p}$ *to be equal to the first* p *columns of* $\boldsymbol{U}$. *Then, these methods define the low-dimensional affine "trial subspace" such that the state is approximated as* $\boldsymbol{x}(t; \boldsymbol{\mu}) \approx \tilde{\boldsymbol{x}}(t; \boldsymbol{\mu}) \in \boldsymbol{x}^0(\boldsymbol{\mu}) + \text{Ran}(\boldsymbol{\Phi})$, *which is equivalent to the approximation in Eq.* (9) *with* $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \boldsymbol{x}^0(\boldsymbol{\mu})$ *and a linear parameterization function* $\boldsymbol{d} : \hat{\boldsymbol{\xi}} \mapsto \boldsymbol{\Phi}\hat{\boldsymbol{\xi}}$.

Latent-dynamics Model: Conservation-enforcing Projection

We now describe the proposed projection-based dynamics model, starting with deep least-squares Petrov–Galerkin (LSPG) projection (proposed in Ref. (Lee and Carlberg 2020)), and proceeding with the proposed Deep Conservation projection.

Deep LSPG Projection

To construct a latent-dynamics model for the approximated state $\tilde{\boldsymbol{x}}$, the Deep LSPG method (Lee and Carlberg 2020) simply substitutes $\boldsymbol{x} \leftarrow \tilde{\boldsymbol{x}}$ defined in Eq. (9) into the FOM OΔE (5) and minimizes the ℓ^2-norm of the residual, i.e.,

$$\hat{\boldsymbol{x}}^n(\boldsymbol{\mu}) = \underset{\hat{\boldsymbol{\xi}} \in \mathbb{R}^p}{\arg\min} \left\| \boldsymbol{r}^n \left(\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{\xi}}); \boldsymbol{\mu} \right) \right\|_2^2, \qquad (10)$$

which is solved sequentially for $n = 1, \ldots, N_t$.

Eq. (10) defines the (discrete-time) dynamics model for the latent states associated with Deep LSPG projection. The nonlinear least-squares problem (10) can be solved using, for example, the Gauss–Newton method, which leads to the iterations, for $k = 0, \ldots, K$,

$$\begin{aligned} \boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)}; \boldsymbol{\mu})^\intercal \boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)}; \boldsymbol{\mu}) \boldsymbol{p}^{n(k)} &= \\ -\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)}; \boldsymbol{\mu})^\intercal \boldsymbol{r}^n &\left(\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{x}}^{n(k)}); \boldsymbol{\mu} \right), \\ \hat{\boldsymbol{x}}^{n(k+1)} &= \hat{\boldsymbol{x}}^{n(k)} + \alpha^{n(k)} \boldsymbol{p}^{n(k)}. \end{aligned}$$

Here, $\hat{\boldsymbol{x}}^{n(0)}$ is the initial guess (often taken to be $\hat{\boldsymbol{x}}^{n-1}$); $\alpha^{n(k)} \in \mathbb{R}$ is a step length chosen to satisfy the strong Wolfe conditions for global convergence; and $\boldsymbol{\Psi}^n : \mathbb{R}^p \times \mathcal{D} \to \mathbb{R}^{N\times p}$, as a function of $\hat{\boldsymbol{\xi}} \in \mathbb{R}^N$ parameterized by $\boldsymbol{\nu}$, is

$$\begin{aligned}\boldsymbol{\Psi}^n(\hat{\boldsymbol{\xi}};\boldsymbol{\nu}) &= \frac{\partial \boldsymbol{r}^n}{\partial \boldsymbol{\xi}}(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\nu}) + \boldsymbol{d}(\hat{\boldsymbol{\xi}});\boldsymbol{\nu})\boldsymbol{J}(\hat{\boldsymbol{\xi}}) \\ &= \left(\alpha_0 \boldsymbol{I} - \Delta t \beta_0 \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\xi}}\left(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\nu}) + \boldsymbol{d}(\hat{\boldsymbol{\xi}}), t^n;\boldsymbol{\nu}\right)\right)\boldsymbol{J}(\hat{\boldsymbol{\xi}}),\end{aligned}$$

where $\boldsymbol{J} : \hat{\boldsymbol{\xi}} \mapsto \frac{d\boldsymbol{d}}{d\hat{\boldsymbol{\xi}}}(\hat{\boldsymbol{\xi}})$ is the Jacobian of the decoder.

Remark 2 (POD–LSPG projection) *POD–LSPG projection (Carlberg, Barone, and Antil 2017) employs the same residual-minimization projection* (10)*, but with reference state $\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) = \boldsymbol{x}^0(\boldsymbol{\mu})$ and linear parameterization function $\boldsymbol{d} : \hat{\boldsymbol{\xi}} \mapsto \boldsymbol{\Phi}\hat{\boldsymbol{\xi}}$ as described in Remark 1.*

Deep Conservation Projection

The Deep LSPG projection (10), however, violates the conservation laws in general, (which will be elaborated later in Remark 3). To overcome this, we derive the proposed Deep Conservation projection, which effectively combines Deep LSPG projection (Lee and Carlberg 2020) just described with conservative LSPG (Carlberg, Choi, and Sargsyan 2018), which was developed for linear embeddings only.

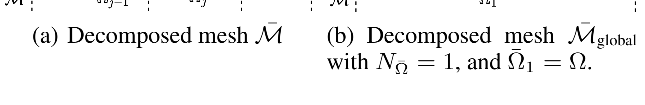
To begin, we decompose the mesh $\mathcal{M}$ into subdomains, each of which comprises the union of control volumes. That is, we define a decomposed mesh $\bar{\mathcal{M}}$ of $N_{\bar{\Omega}}(\le N_{\Omega})$ subdomains $\bar{\Omega}_i = \cup_{j\in\mathcal{K}\subseteq\mathbb{N}(N_\Omega)}\Omega_j$, $i \in \mathbb{N}(N_{\bar{\Omega}})$ with $\bar{\mathcal{M}} := \{\bar{\Omega}_i\}_{i=1}^{N_{\bar{\Omega}}}$ and denote the boundary of the ith subdomain by $\bar{\Gamma}_i := \partial\bar{\Omega}_i$.Note that the global domain can be considered by employing $\bar{\mathcal{M}} = \bar{\mathcal{M}}_{\mathrm{global}}$, which is characterized by $N_{\bar{\Omega}} = 1$ subdomain that corresponds to the global domain. Figure 3 depicts example decomposed meshes.

Enforcing conservation (1) on each subdomain in the decomposed mesh yields

$$\frac{d}{dt}\int_{\bar{\Omega}_j} w_i(\vec{x},t;\boldsymbol{\mu})\mathrm{d}\vec{x} + \int_{\bar{\Gamma}_j}\boldsymbol{g}_i(\vec{x},t;\boldsymbol{\mu})\cdot\bar{\boldsymbol{n}}_j(\vec{x})\mathrm{d}\vec{s}(\vec{x}) = \int_{\bar{\Omega}_j} s_i(\vec{x},t;\boldsymbol{\mu})\mathrm{d}\vec{x}, \tag{11}$$

for the ith conserved variable on the jth subdomain, where $\bar{\boldsymbol{n}}_j$ denotes the unit normal to subdomain $\bar{\Omega}_j$. We propose applying the same finite-volume discretization employed to discretize the control-volume conservation equations (2) to the subdomain conservation equations (11). To accomplish this, we introduce a "decomposed" state vector $\bar{\boldsymbol{x}} \in \mathbb{R}^{\bar{N}}$ with $\bar{N} = N_{\bar{\Omega}} n_w$ and elements

$$\bar{x}_{\bar{\mathcal{I}}(i,j)}(\boldsymbol{x},t;\boldsymbol{\mu}) = \frac{1}{|\bar{\Omega}_j|}\int_{\bar{\Omega}_j} w_i(\vec{x},t;\boldsymbol{\mu})\,\mathrm{d}\vec{x}, \tag{12}$$



(a) Decomposed mesh $\bar{\mathcal{M}}$ (b) Decomposed mesh $\bar{\mathcal{M}}_{\mathrm{global}}$ with $N_{\bar{\Omega}} = 1$, and $\bar{\Omega}_1 = \Omega$.

Figure 3: Examples of decomposed meshes $\bar{\mathcal{M}}$ for the finite-volume mesh shown in Figure 1. The conservation laws are enforced in each control volume Ω_i and each subdomain $\bar{\Omega}_j$.

where $\bar{\mathcal{I}}$ denotes a mapping from conservation-law index and subdomain index to decomposed degree of freedom. **With this formulation, the conservation laws are enforced in each subdomain with the approximated flux and source**, e.g., the rate of change of the mass inside each subdomain equals the incoming flux to that subdomain.

The decomposed state vector can be computed from the state vector $\boldsymbol{x}$ as

$$\bar{\boldsymbol{x}}(\boldsymbol{x}) = \bar{\boldsymbol{C}}\boldsymbol{x}$$

where $\bar{\boldsymbol{C}} \in \mathbb{R}_+^{\bar{N}\times N}$ has elements $\bar{\boldsymbol{C}}_{\bar{\mathcal{I}}(i,j),\mathcal{I}(l,k)} = \frac{|\Omega_k|}{|\bar{\Omega}_j|}\delta_{il}I(\Omega_k \subseteq \bar{\Omega}_j)$, where I is the indicator function, which outputs one if its argument is true, and zero otherwise.

Similarly, the velocity associated with the finite-volume scheme applied to subdomain can be expressed as

$$\bar{\boldsymbol{f}}(\boldsymbol{x},t;\boldsymbol{\mu}) = \bar{\boldsymbol{C}}\boldsymbol{f}(\boldsymbol{x},t;\boldsymbol{\mu}), \tag{13}$$

such that subdomain conservation can be expressed as

$$\bar{\boldsymbol{C}}\dot{\boldsymbol{x}} = \bar{\boldsymbol{C}}\boldsymbol{f}(\boldsymbol{x},t;\boldsymbol{\mu}). \tag{14}$$

Applying a linear multistep scheme to (14) yields

$$\bar{\boldsymbol{C}}\boldsymbol{r}^n(\boldsymbol{x}^n;\boldsymbol{\mu}) = \boldsymbol{0}. \tag{15}$$

Remark 3 (Lack of conservation for Deep LSPG) *We note that* ***the Deep LSPG dynamics model*** (10) ***in general violates the conservation laws underlying the dynamical system of interest.*** *This occurs because the objective function in* (10) *is generally nonzero at the solution, and thus conservation condition* (15) *is not generally satisfied for any decomposed mesh $\bar{\mathcal{M}}$. This lack of conservation can lead to spurious generation or dissipation of physical quantities that should be conserved in principle (e.g., mass, momentum, energy).*

We aim to remedy this primary shortcoming of Deep LSPG with the proposed Deep Conservation projection method. In particular, we define the Deep Conservation dynamics model by equipping the nonlinear least-squares problem (10) with *nonlinear equality constraints* corresponding to Eq. (15), **which has the effect of explicitly enforcing conservation over the decomposed mesh $\bar{\mathcal{M}}$.** In particular, the Deep Conservation dynamics model computes latent states $\hat{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 1,\ldots,N_t$ that satisfy

$$\begin{aligned}&\underset{\hat{\boldsymbol{\xi}}\in\mathbb{R}^p}{\text{minimize}} \left\|\boldsymbol{r}^n\left(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{\xi}});\boldsymbol{\mu}\right)\right\|_2^2 \\ &\text{subject to } \bar{\boldsymbol{C}}\boldsymbol{r}^n\left(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{\xi}});\boldsymbol{\mu}\right) = \boldsymbol{0}.\end{aligned} \tag{16}$$

That is, **conservation laws over control volumes and subdomains are enforced via minimizing the objective function and satisfying the constraint, respectively** (see Figure 3(b), for instance, where the conservation laws are enforced over the global domain via the optimization constraint).

To solve the problem (16) at each time instance, we follow the approach considered in (Carlberg, Choi, and Sargsyan 2018) and apply sequential quadratic programming (SQP) with the Gauss–Newton Hessian approximation, which leads to iterations

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)};\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)};\boldsymbol{\mu}) & \boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)};\boldsymbol{\mu})^\mathsf{T}\bar{\boldsymbol{C}}^\mathsf{T} \\ \bar{\boldsymbol{C}}\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)};\boldsymbol{\mu}) & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\delta\hat{\boldsymbol{x}}^{n(k)} \\ \delta\boldsymbol{\lambda}^{n(k)}\end{bmatrix} \\ &\qquad = -\begin{bmatrix}\boldsymbol{\Psi}^n(\hat{\boldsymbol{x}}^{n(k)};\boldsymbol{\mu})^\mathsf{T}\boldsymbol{r}^n(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{x}}^{n(k)});\boldsymbol{\mu}) \\ \bar{\boldsymbol{C}}\boldsymbol{r}^n(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) + \boldsymbol{d}(\hat{\boldsymbol{x}}^{n(k)});\boldsymbol{\mu}))\end{bmatrix}\end{aligned}$$

$$\begin{bmatrix} \hat{\boldsymbol{x}}^{n(k+1)} \\ \boldsymbol{\lambda}^{n(k+1)} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}}^{n(k)} \\ \boldsymbol{\lambda}^{n(k)} \end{bmatrix} + \eta^{n(k)} \begin{bmatrix} \delta\hat{\boldsymbol{x}}^{n(k)} \\ \delta\boldsymbol{\lambda}^{n(k)} \end{bmatrix},$$

where $\boldsymbol{\lambda}^{n(k)} \in \mathbb{R}^{\bar{N}}$ denotes Lagrange multipliers at time instance n and iteration k and $\eta^{n(k)} \in \mathbb{R}$ is the step length that can be chosen, e.g., to satisfy the strong Wolfe conditions to ensure global convergence to a local solution of (16).

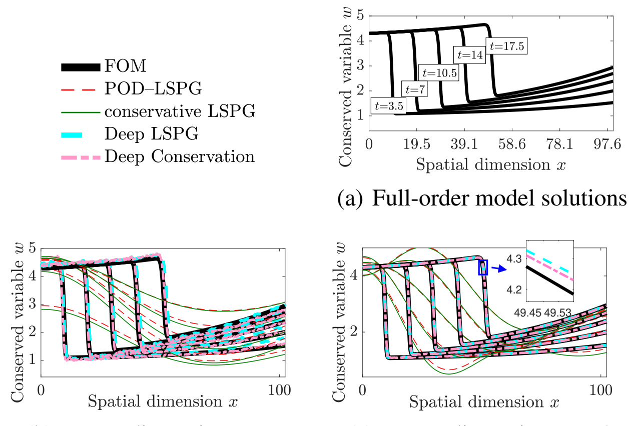
Remark 4 (Conservative LSPG projection) *Conservative LSPG projection (Carlberg, Choi, and Sargsyan 2018) employs the same formulation* (16)*, but with linear parameterization function* $\boldsymbol{d} : \hat{\boldsymbol{\xi}} \mapsto \boldsymbol{\Phi}\hat{\boldsymbol{\xi}}$ *as described in Remark 1.*

Numerical Experiments

This section assesses the performance of (1) the proposed Deep Conservation projection, (2) Deep LSPG projection, which also employs a nonlinear embedding but does not enforce conservation, (3) POD–LSPG projection, which employs a linear embedding and does not enforce conservation, and (4) conservative LSPG projection, which employs a linear embedding but enforces conservation. We consider a parameterized Burgers' equation, as it is a classical benchmark advection problem. We employ the numerical PDE tools and projection functionality provided by `pyMORTestbed` (Zahr and Farhat 2015), and we construct the autoencoder using `TensorFlow` 1.13.1 (Abadi et al. 2016).

Network architecture The Deep Conservation and Deep LSPG methods employ a 10-layer convolutional autoencoder. The encoder $\boldsymbol{h}_{\text{enc}}$ consists of 5 layers with 4 convolutional layers, followed by 1 fully-connected layer. The decoder $\boldsymbol{h}_{\text{dec}}$ consists of 1 fully-connected layer, followed by 4 transposed-convolution layers. The latent state is of dimension p, which will vary during the experiments to define different latent-state dimensions. The size of the convolutional kernels in the encoder and the decoder are $\{16, 8, 4, 4\}$ and $\{4, 4, 8, 16\}$; the numbers of kernel filters in each convolutional and transposed-convolutional layer are $\{8, 16, 32, 64\}$ and $\{64, 32, 16, 1\}$; the stride is configured as $\{2, 4, 4, 4\}$ and $\{4, 4, 4, 2\}$; the "SAME" padding strategy is used; and no pooling is used. For the nonlinear activation functions, we use ELU (Clevert, Unterthiner, and Hochreiter 2016), and no activation function in the output layer.

Data collection and training We consider a parameterized inviscid Burgers' equation (Hirsch 2007), where the system is governed by a conservation law of the form (1) with $n_w = 1$, $d = 1$, $\Omega = [0, 100]$, $\boldsymbol{g}(x, t; \boldsymbol{\mu}) = \frac{w(x,t;\boldsymbol{\mu})^2}{2}$, $s(x, t; \boldsymbol{\mu}) = 0.02e^{\mu_2 x}$ with initial and boundary conditions $w(x, 0; \boldsymbol{\mu}) = 1, \forall x \in [0, 100]$, $w(0, t; \boldsymbol{\mu}) = \mu_1, \forall t \in [0, T]$. There are $n_\mu = 2$ parameters (i.e., $\boldsymbol{\mu} \equiv (\mu_1, \mu_2)$) with the parameter domain $\mathcal{D} = [4.25, 5.5] \times [0.015, 0.03]$, and the final time is set to $T = 35$. We apply Godunov's scheme (Hirsch 2007), which is a finite-volume scheme, with $N_\Omega = 512$ control volumes, which results in a system of ODEs of the form (4) with $N = 512$ spatial degrees of freedom. For time discretization, we use the backward-Euler



(a) Full-order model solutions

(b) Latent dimension, $p = 2$

(c) Latent dimension, $p = 4$

Figure 4: Test stage: solution snapshots at time instances $t = \{3.5, 7.0, 10.5, 14, 17.5\}$ computed by the FOM, POD–LSPG, conservative LSPG, Deep LSPG, and Deep Conservation. All conservative methods enforce global conservation, i.e., they employ $N_{\bar{\Omega}} = 1$ subdomain with $\bar{\Omega}_1 = \Omega$.

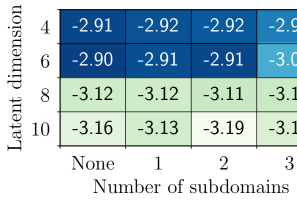
scheme (i.e., $k = 1$, $\alpha_0 = \beta_0 = 1$, $\alpha_1 = -1$, and $\beta_1 = 0$ in Eq. (6)). We consider a uniform time step $\Delta t = 0.07$, resulting in $N_t = 500$.

For offline training, we set the training-parameter instances to $\mathcal{D}_{\text{train}} = \{(4.25 + (1.25/9)i,\ 0.015 + (0.015/7)j)\}_{i=0,\dots,9;\ j=0,\dots7}$, resulting in $n_{\text{train}} = 80$ training-parameter instances. After collecting the snapshots, we split the snapshot matrix (7) into a training set and a validation set; the fraction of snapshots to use for validation is 10%. Then we compute optimal parameters $(\boldsymbol{\theta}^\star_{\text{enc}}, \boldsymbol{\theta}^\star_{\text{dec}})$ using the Adam optimizer (Kingma and Ba 2015) with an initial uniform learning rate $\eta = 10^{-4}$ and initial parameters $(\boldsymbol{\theta}^{(0)}_{\text{enc}}, \boldsymbol{\theta}^{(0)}_{\text{dec}})$ are computed via Xavier initialization (Glorot and Bengio 2010). The number of minibatches is determined by a fixed batch size of 20; a maximum number of epochs is $n_{\text{epoch}} = 1550$; and early-stopping is enforced if the loss on the validation set fails to decrease over 200 epochs.

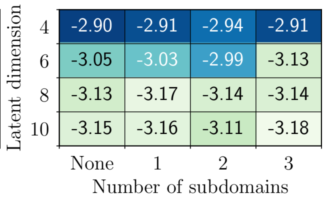
Testing In the online-test stage, solutions of the model problem at a parameter instance $\boldsymbol{\mu}^1_{\text{test}} = (4.3, 0.021) \notin \mathcal{D}_{\text{train}}$ are computed using all considered projection methods. The stopping criterion for all nonlinear solvers is the relative residual and the default stopping tolerance is $\tau = 10^{-5}$. For conservative LSPG and Deep Conservation methods, we consider decomposed meshes, where the subdomains are defined such that they have equal size ($|\bar{\Omega}_i| = |\bar{\Omega}_j|, \forall i, j$), do not overlap ($\text{meas}(\bar{\Omega}_i \cap \bar{\Omega}_j) = 0, \forall i \neq j$), and their union is equal to the full spatial domain ($\cup_i \bar{\Omega}_i = \Omega$).

Effective latent dimension Figure 4 reports solutions at five different time instances computed using FOM and all of the considered projection methods. Figure 4(a) shows the FOM solutions demonstrating that the location of the shock, where the discontinuity exists, moves from left to right as time evolves. All projection methods employ the same latent-state dimension of $p = 2$ and $p = 4$. These results clearly demonstrate that the projection-based methods using nonlinear embeddings yield significantly lower error than the methods using the classical linear embeddings. Moreover, Figure 4 demonstrates that the accuracy of the nonlinear embedding solutions is significantly improved as the latent dimension is increased from $p = 2$ (Figure 4(b)) to $p = 4$ (Figure 4(c)). **As the solutions of the problem are characterized by three factors (t, μ_1, μ_2), the intrinsic solution-manifold dimension is (at most) 3. Thus, autoencoders with the latent dimension larger than or equal to $p = 3$ will be able to reconstruct the original input data given sufficient capacity.**

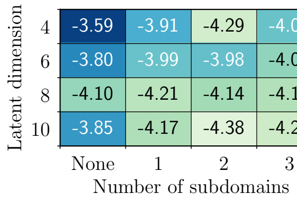
Approximation accuracy Now, we quantitatively assess the accuracy of of the approximated state $\tilde{\boldsymbol{x}}$ computed using Deep LSPG and Deep Conservation methods with the following metrics: 1) the state error, $\mathcal{E}_{\boldsymbol{x}} := \sqrt{\sum_{n=1}^{N_t} \|\boldsymbol{x}^n(\boldsymbol{\mu}) - \tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})\|_2^2 \Big/ \sum_{n=1}^{N_t} \|\boldsymbol{x}^n(\boldsymbol{\mu})\|_2^2}$, 2) the error in the globally conserved variables, $\mathcal{E}_{\boldsymbol{x},\text{global}} := \sqrt{\sum_{n=1}^{N_t} \|\bar{\boldsymbol{C}}_1(\boldsymbol{x}^n(\boldsymbol{\mu}) - \tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}))\|_2^2 \Big/ \sum_{n=1}^{N_t} \|\bar{\boldsymbol{C}}_1 \boldsymbol{x}^n(\boldsymbol{\mu})\|_2^2}$, and 3) the violation in global conservation, $\mathcal{E}_{\boldsymbol{r},\text{global}} := \sqrt{\sum_{n=1}^{N_t} \|\bar{\boldsymbol{C}}_1 \boldsymbol{r}^n(\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}); \boldsymbol{\mu})\|_2^2}$, where $\bar{\boldsymbol{C}}_1 \in \mathbb{R}_+^{n_w \times N}$ is the operator $\bar{\boldsymbol{C}}$ associated with the global conservation $\bar{\mathcal{M}}_{\text{global}} := \{\Omega\}$.



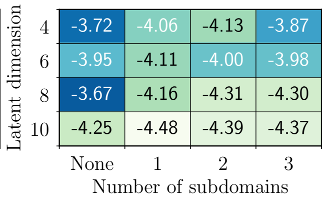
(a) State error $\mathcal{E}_{\boldsymbol{x}}$, $\rho = 0$



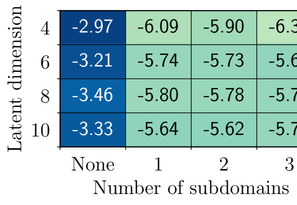
(d) State error $\mathcal{E}_{\boldsymbol{x}}$, $\rho = 1$



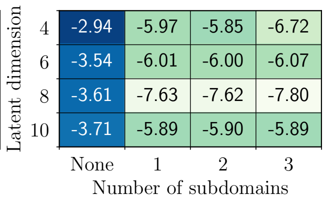
(b) Global conserved-variable error $\mathcal{E}_{\boldsymbol{x},\text{global}}$, $\rho = 0$



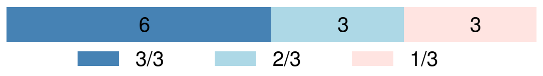
(e) Global conserved-variable error $\mathcal{E}_{\boldsymbol{x},\text{global}}$, $\rho = 1$



(c) Global conservation violation $\mathcal{E}_{\boldsymbol{r},\text{global}}$, $\rho = 0$



(f) Global conservation violation $\mathcal{E}_{\boldsymbol{r},\text{global}}$, $\rho = 1$



(g) Proportion of error metrics where Deep Conservation with the physics-informed constraint ($\rho = 1$) outperforms Deep Conservation with $\rho = 0$ in 1, 2, and all 3 metrics.

Figure 5: Error metrics in $\log_{10}$ scale for Deep LSPG (None) and Deep Conservation ($N_{\bar{\Omega}} \geq 1$) for varying latent-space dimensions p (vertical axis) and for varying numbers of subdomains $N_{\bar{\Omega}}$ (horizontal axis) with the baseline ($\rho = 0$, left) and the hybrid ($\rho = 1$, right) training objective functions.

Figure 5 reports these quantities for the considered methods. **These results illustrate that the best performance in most cases is obtained through the combination of a nonlinear embedding and conservation enforcement as provided by the proposed Deep Conservation method**. That is, lower errors can be achieved by using the proposed Deep Conservation than the Deep LSPG projection. In particular, Deep Conservation reduces the global conservation violation $\mathcal{E}_{\boldsymbol{r},\text{global}}$ by more than an order of magnitude relative to that of Deep LSPG. As numerically demonstrated in (Carlberg, Choi, and Sargsyan 2018), minimizing the residual with the conservation constraint leads to smaller errors in states and globally conserved states.

Figure 5 also shows that **Deep Conservation with the physics-informed objective constraint ($\rho = 1$, right) can lead to smaller errors than Deep Conservation with the baseline autoencoder objective function ($\rho = 0$, left).** The hybrid objective function ($\rho = 1$) helps improving the accuracy in terms of violation in global conservation (Figures 5(c)–5(f)). Based on the 12 experimental settings used in Figure 5 (i.e., combinations of $p = \{4, 6, 8, 10\}$ and $N_{\bar{\Omega}} = \{1, 2, 3\}$), Figure 5(g) reports the proportions of the error metrics where the Deep Conservation with the hybrid objective ($\rho = 1$) outperforms Deep Conservation with the baseline objective ($\rho = 0$) in 1, 2, and, all 3 error metrics.

Conclusion

This work has proposed Deep Conservation: a novel latent-dynamics learning technique that learns a nonlinear embedding using deep convolutional autoencoders, and computes a dynamics model via a projection process that enforces physical conservation laws. The dynamics model associates with a nonlinear least-squares problem with nonlinear equality constraints, and the method requires the availability of a finite-volume discretization of the original dynamical system, which is used to define the objective function and constraints. Numerical experiments on an advection-dominated benchmark problem demonstrated that Deep Conservation both achieves significantly higher accuracy compared with classical projection-based methods, and guarantees the time evolution of the latent state satisfies prescribed conservation laws. In particular, the results highlight that *both* the nonlinear embedding *and* the particular latent-dynamics model associating with the solution to a constrained optimization problem are essential, as removing either of these two elements yields a substantial degradation in performance.

Acknowledgements

The authors thank anonymous reviewers for their helpful and insightful comments.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Broader Impact

Our study suggests that natural sciences can benefit from techniques developed in deep learning/artificial intelligence (DL/AI) and we believe that the study opens up an opportunity for more scientists in the computational science and engineering (CSE) community to participate in DL/AI research, who can contribute to developing models conforming physical laws. Although there have been many efforts to incorporate concepts of physics into neural networks and build more physically-aware models, such models often fail to provide optimal properties or to guarantee that physical laws are enforced, which are of great interest to many scientists. In this study, we proposed a method to resolve those issues; we incorporate deep-learning techniques (autoencoder) into physical laws (the governing equations describing conservation laws) and then, based on well-developed ideas from the computational physics community, we develop an algorithm that enforces exact conservations laws. We believe that this study brings up attention from the CSE community and opens up an opportunity for more scientists to get involved.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283.

Banijamali, E.; Shu, R.; Ghavamzadeh, M.; Bui, H.; and Ghodsi, A. 2018. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*.

Benner, P.; Gugercin, S.; and Willcox, K. 2015. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review* 57(4): 483–531.

Beucler, T.; Rasp, S.; Pritchard, M.; and Gentine, P. 2019. Achieving conservation of energy in neural network emulators for climate modeling. *arXiv preprint arXiv:1906.06622* .

Böhmer, W.; Springenberg, J. T.; Boedecker, J.; Riedmiller, M.; and Obermayer, K. 2015. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intelligenz* 29(4): 353–362.

Carlberg, K.; Barone, M.; and Antil, H. 2017. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *Journal of Computational Physics* 330: 693–734.

Carlberg, K.; Choi, Y.; and Sargsyan, S. 2018. Conservative model reduction for finite-volume models. *Journal of Computational Physics* 371: 280–314.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems*, 6571–6583.

Chen, Z.; Zhang, J.; Arjovsky, M.; and Bottou, L. 2019. Symplectic recurrent neural networks. *arXiv preprint arXiv:1909.13334* .

Clevert, D.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *the 4th International Conference on Learning Representations*.

Cranmer, M.; Greydanus, S.; Hoyer, S.; Battaglia, P.; Spergel, D.; and Ho, S. 2020. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630* .

DeMers, D.; and Cottrell, G. W. 1993. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems*, 580–587.

Fulton, L.; Modi, V.; Duvenaud, D.; Levin, D. I.; and Jacobson, A. 2019. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, volume 38, 379–391. Wiley Online Library.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Gonzalez, F. J.; and Balajewicz, M. 2018. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint arXiv:1808.01346* .

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep Learning*, volume 1. MIT press Cambridge.

Goroshin, R.; Mathieu, M. F.; and LeCun, Y. 2015. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, 1234–1242.

Greydanus, S.; Dzamba, M.; and Yosinski, J. 2019. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, 15379–15389.

Hinton, G. E.; and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786): 504–507.

Hirsch, C. 2007. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. Elsevier.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8): 1735–1780.

Holmes, P.; Lumley, J. L.; Berkooz, G.; and Rowley, C. W. 2012. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press.

Karl, M.; Soelch, M.; Bayer, J.; and van der Smagt, P. 2017. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *the 3rd International Conference on Learning Representations*.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553): 436.

Lee, K.; and Carlberg, K. T. 2020. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics* 404: 108973.

Lesort, T.; Díaz-Rodríguez, N.; Goudou, J.-F.; and Filliat, D. 2018. State representation learning for control: An overview. *Neural Networks* .

LeVeque, R. J. 2002. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press.

Lusch, B.; Kutz, J. N.; and Brunton, S. L. 2018. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications* 9(1): 4950.

Maulik, R.; Lusch, B.; and Balaprakash, P. 2020. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *arXiv preprint arXiv:2002.00470* .

Morton, J.; Jameson, A.; Kochenderfer, M. J.; and Witherden, F. 2018. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems*, 9258–9268.

Ohlberger, M.; and Rave, S. 2016. Reduced basis methods: Success, limitations and future challenges. In *Proceedings of ALGORITMY*, 1–12. Slovak University of Technology.

Otto, S. E.; and Rowley, C. W. 2019. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems* 18(1): 558–593.

Portwood, G. D.; Mitra, P. P.; Ribeiro, M. D.; Nguyen, T. M.; Nadiga, B. T.; Saenz, J. A.; Chertkov, M.; Garg, A.; Anandkumar, A.; Dengel, A.; et al. 2019. Turbulence forecasting via neural ODE. *arXiv preprint arXiv:1911.05180* .

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. 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.

Takeishi, N.; Kawahara, Y.; and Yairi, T. 2017. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, 1130–1140.

Toth, P.; Rezende, D. J.; Jaegle, A.; Racanière, S.; Botev, A.; and Higgins, I. 2019. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789* .

Watter, M.; Springenberg, J.; Boedecker, J.; and Riedmiller, M. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, 2746–2754.

Wiewel, S.; Becher, M.; and Thuerey, N. 2019. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*, volume 38, 71–82. Wiley Online Library.

Zahr, M. J.; and Farhat, C. 2015. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *International Journal for Numerical Methods in Engineering* 102(5): 1111–1135.