Convolutional autoencoders and LSTMs
Using deep learning to overcome Kolmogorov-width limitations and accurately model errors in nonlinear model reduction

Kevin Carlberg
University of Washington
Mathematics of Reduced-Order Models
ICERM, Providence, Rhode Island
February 21, 2020

Kookjin Lee
Eric Parish
High-fidelity simulation

+ **Indispensable** in science and engineering
- **Extreme-scale** models required for high fidelity
High-fidelity simulation

+ Indispensable in science and engineering
- Extreme-scale models required for high fidelity

+ High fidelity: matches wind-tunnel experiments to within 5%
- Extreme scale: 100 million cells, 200,000 time steps
- High simulation costs: 6 weeks, 5000 cores

computational barrier

Many-query problems

- uncertainty propagation
- Bayesian inference
- stochastic optimization

Goal: break computational barrier
**Approach**: exploit simulation data

ODE: \[ \frac{dx}{dt} = f(x; t, \mu), \quad x(0, \mu) = x_0(\mu), \quad t \in [0, T_{\text{final}}], \quad \mu \in \mathcal{D} \]

**Many-query problem**: solve ODE for \( \mu \in \mathcal{D}_{\text{query}} \)

**Idea**: exploit simulation data collected at a few points
1. **Training**: Solve ODE for $\mu \in D_{\text{training}}$ and collect simulation data
2. **Machine learning**: Identify structure in data
3. **Reduction**: Reduce the cost of solving ODE for $\mu \in D_{\text{query}} \setminus D_{\text{training}}$

ODE: $$\frac{dx}{dt} = f(x; t, \mu)$$

- Number of state variables $N$
- Number of time steps $T$
1. **Training:** Solve ODE for $\mu \in D_{\text{training}}$ and collect simulation data

2. **Machine learning:** Identify structure in data

3. **Reduction:** Reduce the cost of solving ODE for $\mu \in D_{\text{query}} \setminus D_{\text{training}}$

ODE: \[
\frac{d\mathbf{x}}{dt} = f(\mathbf{x}; t, \mu)
\]
1. **Training:** Solve ODE for $\mu \in \mathcal{D}_{\text{training}}$ and collect simulation data

2. **Machine learning:** Identify structure in data

3. **Reduction:** Reduce the cost of solving ODE for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

\[
X = \Phi U \Sigma V^T
\]

*Φ* columns are principal components of the spatial simulation data
1. **Training:** Solve ODE for $\mu \in D_{\text{training}}$ and collect simulation data  

2. **Machine learning:** Identify structure in data  

3. **Reduction:** Reduce the cost of solving ODE for $\mu \in D_{\text{query}} \setminus D_{\text{training}}$ 

   - ODE residual: $r(\mathbf{v}, \mathbf{x}, t) := \mathbf{v} - f(\mathbf{x}, t)$  
   - O\(\Delta E\) residual: $r^n(\mathbf{w}) :=\alpha_0 \mathbf{w} - \Delta t \beta_0 f(\mathbf{w}, t^n) + \sum_{j=1}^k \alpha_j \mathbf{x}^{n-j} - \Delta t \sum_{j=1}^k \beta_j f(\mathbf{x}^{n-j}, t^{n-j})$  

   - Other residual-minimizing ROMs [LeGresley and Alonso, 2000; Bui-Thanh et al., 2008; Bui-Thanh et al., 2008; Constantine and Wang, 2012; Choi and C.; 2019; Parish and C., 2019]
Unsteady Navier–Stokes  \( \text{Re} = 6.3 \times 10^6 \)  \( \text{M}_\infty = 0.6 \)

Spatial discretization
\- 2nd-order finite volume
\- DES turbulence model
\- \( 1.2 \times 10^6 \) degrees of freedom

Temporal discretization
\- 2nd-order BDF
\- Verified time step \( \Delta t = 1.5 \times 10^{-3} \)
\- \( 8.3 \times 10^3 \) time instances

Captive carry
LSPG ROM with sample mesh [C., Farhat, Cortial, Amsallem, 2013]

\[ \Phi \hat{x}^n = \arg \min_{v \in \text{range}(\Phi)} \| r^n(v) \| \Theta \]

sample mesh

+ HPC on a laptop

vorticity field

pressure field

LSPG ROM
32 min, 2 cores

high-fidelity
5 hours, 48 cores

+ 229x savings in core–hours
+ < 1% error in time-averaged drag

... so why doesn’t everyone use ROMs?
Outstanding challenges in model reduction

1) Linear-subspace assumption is strong
\[ x(t) \approx \tilde{x}(t) = \Phi \hat{x}(t) \]


2) Important physical properties not satisfied
\[ \Phi \frac{d\hat{x}}{dt}(x, t) = \arg\min_{\mathbf{v} \in \text{range}(\Phi)} \| \mathbf{r}(\mathbf{v}, x; t) \|_2 \]
\[ \Phi\hat{x}^n = \arg\min_{\mathbf{v} \in \text{range}(\Phi)} \| \mathbf{r}^n(\mathbf{v}) \|_2 \]


3) Error analysis difficult

Outstanding challenges in model reduction

1) Linear-subspace assumption is strong

\[ x(t) \approx \tilde{x}(t) = \Phi \hat{x}(t) \]


2) Important physical properties not guaranteed


3) Error analysis difficult

Kolmogorov-width limitation of linear subspaces

- $\mathcal{M} := \{x(t, \mu) \mid t \in [0, T_{\text{final}}], \mu \in \mathcal{D}\}$: solution manifold
- $S_p$: set of all $p$-dimensional linear subspaces
- $d_p(\mathcal{M}) := \inf_{S \in S_p} P_\infty(\mathcal{M}, S)$, $P_\infty(\mathcal{M}, S) := \sup_{x \in \mathcal{M}} \inf_{y \in S} \|x - y\|$
Kolmogorov-width limitation of linear subspaces

- $\mathcal{M} := \{ \mathbf{x}(t, \mu) \mid t \in [0, T_{\text{final}}], \mu \in \mathcal{D} \}$: solution manifold
- $S_p$: set of all $p$-dimensional linear subspaces
- $\tilde{d}_p(\mathcal{M}) := \inf_{S \in S_p} P_2(\mathcal{M}, S)$, $P_2(\mathcal{M}, S) := \sqrt{\sum_{x \in \mathcal{M}} \inf_{y \in S} \|x - y\|^2} / \sqrt{\sum_{x \in \mathcal{M}} \|x\|^2}$

Graph:
- $\tilde{d}_p(\mathcal{M})$
- $P_2(\mathcal{M}, \text{range}(\Phi))$

**Training deficiency**
Kolmogorov-width limitation of linear subspaces

- $\mathcal{M} := \{x(t, \mu) \mid t \in [0, T_{\text{final}}], \mu \in \mathcal{D}\}$: solution manifold
- $S_p$: set of all $p$-dimensional linear subspaces
- $\tilde{d}_p(\mathcal{M}) := \inf_{S \in S_p} P_2(\mathcal{M}, S)$, $P_2(\mathcal{M}, S) := \sqrt{\sum_{x \in \mathcal{M}} \inf_{y \in S} \|x - y\|^2} / \sqrt{\sum_{x \in \mathcal{M}} \|x\|^2}$

$$\tilde{d}_p(\mathcal{M})$$

$$P_2(\mathcal{M}, \text{range}(\Phi))$$

$\sqrt{\sum_{x \in \mathcal{M}} \|x - \tilde{x}_{\text{LSPG}}\|^2} / \sqrt{\sum_{x \in \mathcal{M}} \|x\|^2}$
Kolmogorov-width limitation of linear subspaces

- \( \mathcal{M} := \{ \mathbf{x}(t, \mu) \mid t \in [0, T_{\text{final}}], \mu \in \mathcal{D} \} \): solution manifold
- \( S_p \): set of all \( p \)-dimensional linear subspaces
- \( \tilde{d}_p(\mathcal{M}) := \inf_{S \in S_p} \ P_2(\mathcal{M}, S) \), \( P_2(\mathcal{M}, S) := \sqrt{\sum_{\mathbf{x} \in \mathcal{M}} \inf_{\mathbf{y} \in S} \| \mathbf{x} - \mathbf{y} \|^2} / \sqrt{\sum_{\mathbf{x} \in \mathcal{M}} \| \mathbf{x} \|^2} \)

- Kolmogorov-width limitation: significant error for \( p = \dim(\mathcal{M}) \)

**Goal:** overcome limitation via projection onto a nonlinear manifold
Overcoming Kolmogorov-width limitation

Transform/update the linear subspace

[Ohlberger and Rave, 2013; Iollo and Lombardi, 2014; Gerbeau and Lombardi, 2014; Peherstorfer and Willcox, 2015; Welper, 2017; Mojgani and Balajewicz, 2017; Reiss et al., 2018; Zimmermann et al., 2018; Peherstorfer, 2018; Rim and Mandli, 2018; Rim and Mandli, 2018; Nair and Balajewicz, 2019; Cagniart et al., 2019]

+ Can work much better than a fixed basis
- Some require problem-specific knowledge or characteristics
- Do not consider manifolds of general nonlinear structure
Overcoming Kolmogorov-width limitation

Transform/update the linear subspace
[Ohlberger and Rave, 2013; Iollo and Lombardi, 2014; Gerbeau and Lombardi, 2014; Peherstorfer and Willcox, 2015; Welper, 2017; Moigani and Balajewicz, 2017; Reiss et al., 2018; Zimmermann et al., 2018; Peherstorfer, 2018; Rim and Mandli, 2018; Rim and Mandli, 2018; Nair and Balajewicz, 2019; Cagniart et al., 2019]

+ Can work much better than a fixed basis
- Some require problem-specific knowledge or characteristics
- Do not consider manifolds of general nonlinear structure

A priori construction of local linear subspaces
[Dihlmann et al., 2011; Drohmann et al., 2011; Amsallem, Zahr, Farhat, 2012; Peherstorfer et al., 2014; Taddei et al., 2015]

+ Tailored bases for local regions of space/time domain, state space
- Do not consider manifolds of general nonlinear structure
Overcoming Kolmogorov-width limitation

Transform/update the linear subspace
[Ohlberger and Rave, 2013; Iollo and Lombardi, 2014; Gerbeau and Lombardi, 2014; Peherstorfer and Willcox, 2015; Welper, 2017; Moigani and Balajewicz, 2017; Reiss et al., 2018; Zimmermann et al., 2018; Peherstorfer, 2018; Rim and Mandli, 2018; Rim and Mandli, 2018; Nair and Balajewicz, 2019; Cagniart et al., 2019]

+ Can work much better than a fixed basis
- Some require problem-specific knowledge or characteristics
- Do not consider manifolds of general nonlinear structure

A priori construction of local linear subspaces
[Dihlmann et al., 2011; Drohmann et al., 2011; Amsallem, Zahr, Farhat, 2012; Peherstorfer et al., 2014; Taddei et al., 2015]

+ Tailored bases for local regions of space/time domain, state space
- Do not consider manifolds of general nonlinear structure

Model reduction on nonlinear manifolds [Gu, 2011; Kashima, 2016; Hartman and Mestha, 2017]
- Kinematically inconsistent [Kashima, 2016; Hartman and Mestha, 2017]
- Limited to piecewise linear manifolds [Gu, 2011]
Goals

Overcome shortcomings of existing methods
+ Enable manifolds with general nonlinear structure
+ Kinematically consistent
+ Satisfy optimality property

*Manifold Galerkin and LSPG projection*

Practical nonlinear-manifold construction
+ No problem-specific knowledge required
+ Use same training data as POD

*Deep convolutional autoencoders*
Goals

Overcome shortcomings of existing methods
+ Enable manifolds with general nonlinear structure
+ Kinematically consistent
+ Satisfy optimality property

*Manifold Galerkin and LSPG projection*

Practical nonlinear-manifold construction
+ No problem-specific knowledge required
+ Use same training data as POD

*Deep convolutional autoencoders*
Nonlinear trial manifold

**Linear trial subspace**

\[
\text{range}(\Phi) := \{ \Phi \hat{x} \mid \hat{x} \in \mathbb{R}^p \}
\]

**Nonlinear trial manifold**

\[
\mathcal{S} := \{ g(\hat{x}) \mid \hat{x} \in \mathbb{R}^p \}
\]

element example

\(N=3\)

\(p=2\)

\[
\begin{align*}
\dot{x}(t) & \approx \ddot{x}(t) = \Phi \hat{x}(t) \in \text{range}(\Phi) \\
\dot{x}(t) & \approx \ddot{x}(t) = g(\hat{x}(t)) \in \mathcal{S} \\
\text{state} & \\
\text{velocity} & \\
\frac{dx}{dt} & \approx \frac{d\hat{x}}{dt} = \Phi \frac{d\hat{x}}{dt} \in \text{range}(\Phi) \\
\frac{dx}{dt} & \approx \frac{d\hat{x}}{dt} = \nabla g(\hat{x}) \frac{d\hat{x}}{dt} \in T_{\hat{x}} \mathcal{S} \\
\text{+ Manifold has general structure} & \\
\text{+ Kinematically consistent} &
\end{align*}
\]
1. **Training:** Solve ODE for \( \mu \in D_{\text{training}} \) and collect simulation data

2. **Machine learning:** Identify structure in data

3. **Reduction:** Reduce the cost of solving ODE for \( \mu \in D_{\text{query}} \setminus D_{\text{training}} \)

**Linear-subspace ROM**

Given \( \Phi \)

\[
\frac{d\hat{x}}{dt} = \text{argmin}_{\hat{v} \in \mathbb{R}^p} \| r(\Phi \hat{v}, \Phi \hat{x}; t) \|_2
\]

\[
\frac{d\hat{x}}{dt} = \Phi^T f(\Phi \hat{x}; t)
\]

**Nonlinear-manifold ROM**

Given \( g(\hat{x}) \)

\[
\frac{d\hat{x}}{dt} = \text{argmin}_{\hat{v} \in \mathbb{R}^p} \| r(\nabla g(\hat{x}) \hat{v}, g(\hat{x}); t) \|_2
\]

\[
\frac{d\hat{x}}{dt} = \nabla g(\hat{x})^T f(g(\hat{x}); t)
\]

\[\hat{x}^n = \text{argmin}_{\hat{v} \in \mathbb{R}^p} \| r^n(g(\hat{v})) \|_2 \]

+ Satisfy residual minimization

---

**Theorem** [Lee, C., 2020]

Manifold Galerkin and manifold LSPG are equivalent if

1. the nonlinear trial manifold \( S \) is twice continuously differentiable,
2. \( \| \hat{x}^{n-j} - \hat{x}^n \| = O(\Delta t) \) for \( n = 1, \ldots, T \) and \( j = 1, \ldots, k \), and
3. the limit \( \Delta t \to 0 \) is taken.
**Error bound**

**Theorem** [Lee, C., 2020]

If the following conditions hold:

1. \( f(\cdot; t) \) is Lipschitz continuous with Lipschitz constant \( \kappa \)
2. \( \Delta t \) is small enough such that \( 0 < h : = |\alpha_0| - |\beta_0| \kappa \Delta t \), then

\[
\| x^n - g(\hat{x}_G^n) \|_2 \leq \frac{1}{h} \| r^n_G(g(\hat{x}_G)) \|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\gamma_{\ell}| \| x^{n-\ell} - g(\hat{x}_G) \|_2
\]

\[
\| x^n - g(\hat{x}_{\text{LSPG}}^n) \|_2 \leq \frac{1}{h} \min_{\hat{v}} \| r^n_{\text{LSPG}}(g(\hat{v})) \|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\gamma_{\ell}| \| x^{n-\ell} - g(\hat{x}_{\text{LSPG}}) \|_2
\]

+ Manifold LSPG sequentially minimizes the error bound

---

**How to construct manifold** \( S := \{ g(\hat{x}) \mid \hat{x} \in \mathbb{R}^p \} \) **from training data?**
Goals

Overcome shortcomings of existing methods
+ Enable manifolds with general nonlinear structure
+ Kinematically consistent
+ Satisfy optimality property

*Manifold Galerkin and LSPG projection*

Practical nonlinear-manifold construction
+ No problem-specific knowledge required
+ Use same training data as POD

*Deep convolutional autoencoders*

\[ S := \{ g(\hat{x}) \mid \hat{x} \in \mathbb{R}^p \} \]
Deep autoencoders

Encoder $h_{\text{enc}}(\cdot; \theta_{\text{enc}})$  Decoder $h_{\text{dec}}(\cdot; \theta_{\text{dec}})$

$$\tilde{x} = h_{\text{dec}}(\cdot; \theta_{\text{dec}}) \circ h_{\text{enc}}(x; \theta_{\text{enc}})$$

+ If $\tilde{x} \approx x$ for $\theta_{\text{dec}}^*$, then $g = h_{\text{dec}}(\cdot; \theta_{\text{dec}}^*)$ is accurate manifold parameterization
1. **Training:** Solve ODE for $\mu \in \mathcal{D}_{\text{training}}$ and collect simulation data

2. **Machine learning:** Identify structure in data

3. **Reduction:** Reduce the cost of solving ODE for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

- Compute $\theta^*$ by approximately solving $\min_{\theta} \| X - \tilde{X}(\theta) \|_F$
- Define nonlinear trial manifold by setting $g = h_{\text{dec}}(\cdot; \theta^*_{\text{dec}})$

+ Same snapshot data, no specialized problem knowledge
1. **Training:** Solve ODE for $\mu \in D_{\text{training}}$ and collect simulation data

2. **Machine learning:** Identify structure in data

3. **Reduction:** Reduce the cost of solving ODE for $\mu \in D_{\text{query}} \setminus D_{\text{training}}$

**Subspace ROM**

Given $\Phi$

Galerkin: \[ \frac{d\hat{x}}{dt} = \arg\min_{\hat{v} \in \mathbb{R}^p} \| r(\Phi\hat{v}, \Phi\hat{x}; t) \|_2 \]

LSPG: \[ \hat{x}^n = \arg\min_{\hat{v} \in \mathbb{R}^p} \| r^n(\Phi\hat{v}) \|_2 \]

**Manifold ROM**

Given $g(\hat{x})$

Galerkin: \[ \frac{d\hat{x}}{dt} = \arg\min_{\hat{v} \in \mathbb{R}^p} \| r(\nabla g(\hat{x})\hat{v}, g(\hat{x}); t) \|_2 \]

LSPG: \[ \hat{x}^n = \arg\min_{\hat{v} \in \mathbb{R}^p} \| r^n(g(\hat{v})) \|_2 \]

+ Satisfy residual minimization

+ Predictions directly integrate deep learning with computational physics
Numerical results

1D Burgers’ equation

\[ \frac{\partial w(x, t; \mu)}{\partial t} + \frac{\partial f(w(x, t; \mu))}{\partial x} = 0.02e^{ax} \]

- \( \mu : \alpha \), inlet boundary condition
- Spatial discretization: finite volume
- Time integrator: backward Euler

2D reacting flow

\[ \frac{\partial w(x, t; \mu)}{\partial t} = \nabla \cdot (\kappa \nabla w(x, t; \mu)) - v \cdot \nabla w(x, t; \mu) + q(w(x, t; \mu); \mu) \]

- \( \mu : \) two terms in reaction
- Spatial discretization: finite difference
- Time integrator: BDF2

Autoencoder architecture

4 convolutional layers  2 fully-connected layers  2 fully-connected layers  4 convolutional layers
Manifold interpretation: Burgers’ equation

**FOM**

**POD, p=3 projection**

**Autoencoder, p=3 projection**

- Projection error onto 3-dimensional manifold **nearly perfect**
Manifold LSPG outperforms optimal linear subspace

1D Burgers’ equation: conserved variable

2D reacting flow: temperature, $H_2$ fraction

**Manifold LSPG**

$p=5$

**POD-LSPG**

$p=5$

**High-fidelity model**
Method improves generalization performance

**Burgers’ equation**

Relative error vs. reduced dimension $p$

- Autoencoder manifold **significantly better** than optimal linear subspace

**Reacting flow**

Subspace LSPG

- $\tilde{d}_p(\mathcal{M})$
- $P_2(\mathcal{M}, \text{range}(\Phi))$
- $\text{dim}(\mathcal{M})$
- $P_2(\mathcal{M}, S)$
Method improves generalization performance

**Burgers’ equation**

- Relative error vs. reduced dimension $p$

**Reacting flow**

- Relative error vs. reduced dimension $p$

\[ \tilde{d}_p(\mathcal{M}) \]

\[ P_2(\mathcal{M}, \text{range}(\Phi)) \]

- Subspace LSPG
- \( \dim(\mathcal{M}) \)
- \( P_2(\mathcal{M}, S) \)
- Manifold LSPG

+ Autoencoder manifold *significantly better* than optimal linear subspace
+ Manifold LSPG *orders-of-magnitude more accurate* than subspace LSPG
Method improves generalization performance

**Burgers’ equation**

**Reacting flow**

+ Autoencoder manifold **significantly better** than optimal linear subspace
+ Manifold LSPG **orders-of-magnitude more accurate** than subspace LSPG
+ Method **breaks Kolmogorov-width barrier**

\[ \tilde{d}_p(\mathcal{M}) \]
\[ P_2(\mathcal{M}, \text{range}(\Phi)) \]
\[ \text{subspace LSPG} \]
\[ \dim(\mathcal{M}) \]
\[ P_2(\mathcal{M}, S) \]
\[ + \text{ manifold LSPG} \]
Method improves generalization performance

+ Autoencoder manifold **significantly better** than optimal linear subspace
+ Manifold LSPG orders-of-magnitude more accurate than subspace LSPG
+ Method **breaks Kolmogorov-width barrier**
+ Manifold LSPG outperforms manifold Galerkin on 1D Burgers’ equation
Outstanding challenges in model reduction

1) Linear-subspace assumption is strong

\[ x(t) \approx \tilde{x}(t) = \Phi \hat{x}(t) \]


2) Important physical properties not satisfied

\[ \Phi \frac{d\hat{x}}{dt}(x, t) = \underset{v \in \text{range}(\Phi)}{\text{argmin}} \| r(v, x; t) \|_2 \]

\[ \Phi \hat{x}^n = \underset{v \in \text{range}(\Phi)}{\text{arg min}} \| r^n(v) \|_2 \]


3) Error analysis difficult

Finite-volume method

\[ \text{ODE: } \frac{dx}{dt} = f(x; t) \]

\[ x_I(i,j)(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} u_i(\vec{x}, t) \, d\vec{x} \]

- average value of conserved variable \( i \) over control volume \( j \)

\[ f_I(i,j)(x, t) = -\frac{1}{|\Omega_j|} \int_{\Gamma_j} g_i(x; \vec{x}, t) \cdot n_j(\vec{x}) \, ds(\vec{x}) + \frac{1}{|\Omega_j|} \int_{\Omega_j} s_i(x; \vec{x}, t) \, d\vec{x} \]

- flux and source of conserved variable \( i \) within control volume \( j \)

\[ r_I(i,j) = \frac{dx_I(i,j)}{dt}(t) - f_I(i,j)(x, t) \]

- rate of conservation violation of variable \( i \) in control volume \( j \)

\[ \text{ODE: } r^n(x^n) = 0, \quad n = 1, \ldots, N \]

\[ r^n_I(i,j) = x_I(i,j)(t^{n+1}) - x_I(i,j)(t^n) + \int_{t^n}^{t^{n+1}} f_I(i,j)(x, t) \, dt \]

- conservation violation of variable \( i \) in control volume \( j \) over time step \( n \)

Conservation is the intrinsic structure enforced by finite-volume methods
Conservative manifold model reduction

**Manifold Galerkin**

\[
\min_{\hat{\nu} \in \mathbb{R}^p} || r(\nabla g(\hat{x})\hat{\nu}; g(\hat{x}); t) ||_2
\]

- Minimize conservation-violation rates

**Manifold LSPG**

\[
\hat{x}^n = \arg\min_{\hat{\nu} \in \mathbb{R}^p} || r^n(g(\hat{\nu})) ||_2
\]

- Minimize conservation violations over time step \( n \)

- Neither enforces conservation!

**Conservative manifold Galerkin**

\[
\min_{\hat{\nu} \in \mathbb{R}^p} || r(\nabla g(\hat{x})\hat{\nu}; g(\hat{x}); t) ||_2
\]

subject to \( \text{Cr}(\nabla g(\hat{x})\hat{\nu}; g(\hat{x}); t) = 0 \)

- Minimize conservation-violation rates

subject to zero conservation-violation rates over subdomains

**Conservative manifold LSPG**

\[
\min_{\hat{\nu} \in \mathbb{R}^p} || r^n(g(\hat{\nu})) ||_2
\]

subject to \( \text{Cr}^n(g(\hat{\nu})) = 0 \)

- Minimize conservation violations over time step \( n \)

subject to zero conservation violations over time step \( n \)

over subdomains

**Conservation enforced over prescribed subdomains**
Lemma: local conserved-quantity error bounds [C., Choi, Sargsyan, 2018]

The error in the conserved quantities computed with either conservative Galerkin or conservative LSPG can be bounded as:

\[
\| \bar{C}(x^n - \Phi \hat{x}^n) \|_2 \leq \sum_{\ell=0}^{k} \frac{|\beta^n_\ell| \Delta t}{|\alpha^n_0|} \| \bar{C}f(x^{n-\ell}) - \bar{C}f(\Phi \hat{x}^{n-\ell}) \|_2 \\
+ \sum_{\ell=1}^{k} \frac{|\alpha^n_\ell|}{|\alpha^n_0|} \| \bar{C}(x^{n-\ell} - \Phi \hat{x}^{n-\ell}) \|_2
\]

- Error depends only on velocity error on decomposed mesh
- No source, global conservation: error due to flux error along boundary!
High-fidelity model

Reduced-order models

**POD subspace**

- Solution error: 13%
- Conservation violation: 16%

**Autoencoder manifold**

- Solution error: 0.5%
- Conservation violation: 1%

**POD subspace with conservation constraints**

- Solution error: 12%
- Conservation violation: <0.001%

**Autoencoder manifold with conservation constraints**

- Solution error: 0.2%
- Conservation violation: <0.001%
**Outlook**

**Conservative manifold Galerkin**

\[
\begin{align*}
\text{minimize} & \| r(\nabla g(\hat{x})\hat{v}; g(\hat{x}); t) \|_2 \\
\text{subject to} & \mathbf{C}_r(\nabla g(\hat{x})\hat{v}; g(\hat{x}); t) = 0
\end{align*}
\]

**Conservative manifold LSPG**

\[
\begin{align*}
\text{minimize} & \| r^n(g(\hat{v})) \|_2 \\
\text{subject to} & \mathbf{C}_r^n(g(\hat{v})) = 0
\end{align*}
\]

**Interpretation**

- Integrates **computational physics** with **deep learning**
- **Projection-based latent dynamics model** that enforces conservation
- Nearly all existing methods are **data-driven latent dynamics models**
  
  [Böhmer et al., 2015; Goroshin et al., 2015; Watter et al., 2015; Karl et al., 2017; Takeishi et al., 2017; Banijamali et al., 2018; Lesort et al., 2018; Lusch et al., 2018; Morton et al., 2018 Otto and Rowley, 2019]

**Gradient computation**

- Backpropagation used to compute decoder Jacobian \( \nabla g(\hat{x}) \)
- Quasi-Newton solvers directly call TensorFlow

**Ongoing work**

- **Hyper-reduction**: “easy” because convolutional layers preserve sparsity
- Integration in large-scale code underway in Pressio
Shortcomings of state-of-the-art ROMs

1) Linear-subspace assumption is strong
\[ x(t) \approx \tilde{x}(t) = \Phi \hat{x}(t) \]


2) Important physical properties not guaranteed

\[
\Phi \frac{d\hat{x}}{dt}(x, t) = \text{argmin}_{v \in \text{range}(\Phi)} \| r(v, x; t) \|_2 \quad \text{Galerkin} \quad \quad \quad \quad \text{LSPG} \quad \Phi \hat{x}^n = \text{arg min}_{v \in \text{range}(\Phi)} \| r^n(v) \|_2
\]


3) Error analysis difficult

Discrete-time error bound

Theorem: error bound for BDF integrators [C., Barone, Antil, 2017]

If the following conditions hold:
1. \( f(\cdot; t) \) is Lipschitz continuous with Lipschitz constant \( \kappa \)
2. The time step \( \Delta t \) is small enough such that 
   \[
   0 < h := |\alpha_0| - |\beta_0| \kappa \Delta t,
   \]

\[
\|x^n - g(\hat{x}_G^n)\|_2 \leq \frac{1}{h}\|r_G^n(g(\hat{x}_G^n))\|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\gamma_\ell|\|x^{n-\ell} - g(\hat{x}_G)\|_2
\]

\[
\|x^n - g(\hat{x}_{LSPG}^n)\|_2 \leq \frac{1}{h} \min_{\hat{v}} \|r_{LSPG}^n(g(\hat{v}))\|_2 + \frac{1}{h} \sum_{\ell=1}^{k} |\gamma_\ell|\|x^{n-\ell} - g(\hat{x}_{LSPG})\|_2
\]

Can we use these error bounds for error estimation?
Discrete-time error bound

**Theorem:** error bound for BDF integrators [C., Barone, Antil, 2017]

If the following conditions hold:
1. $f(\cdot; t)$ is Lipschitz continuous with Lipschitz constant $\kappa$
2. The time step $\Delta t$ is small enough such that $0 < h := |\alpha_0| - |\beta_0| \kappa \Delta t$,

\[
\|x^n - g(\hat{x}_G^n)\|_2 \leq \gamma_1 (\gamma_2)^{n} \exp(\gamma_3 t^n) \frac{\gamma_4 + \gamma_5 \Delta t}{\max_{j\in\{1,\ldots,N\}} \|r_L^{i}\|_2} \\
\|x^n - g(\hat{x}_{LSPG}^n)\|_2 \leq \gamma_1 (\gamma_2)^{n} \exp(\gamma_3 t^n) \frac{\gamma_4 + \gamma_5 \Delta t}{\max_{j\in\{1,\ldots,N\}} \min_{\hat{v}} \|r_{LSPG}^{i}(g(\hat{v}))\|_2}
\]

**Can we use these error bounds for error estimation?**

- grow exponentially in time
- deterministic: not amenable to uncertainty quantification
Main idea

- **Observation**: ROMs generate quantities that are **informative** of the error

- **ML perspective**: these are **good features** for predicting the error

*Idea*: Apply **machine learning regression** to generate a mapping from residual-based quantities to a random variable for the error

**Machine-learning error models** [Freno and C., 2019; Parish and C., 2019]
Machine-learning error models: formulation

What attributes does the ROM error have?

\[
\|x^n - g(\hat{x}^n_{\text{LSPG}})\|_2 \leq \frac{\gamma_1 (\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j \in \{1, \ldots, T\}} \min_{\hat{v}} \|r^j_{\text{LSPG}}(g(\hat{v}))\|_2
\]

1. Dependence on non-local quantities in time
2. Dependence on the residual

**Regression model**

\[
\hat{\delta}^n(\mu) = \hat{\delta}^n_f(\mu) + \hat{\delta}^n_\varepsilon(\mu)
\]

deterministic \hspace{1cm} stochastic

- regression function:
  \[
  \hat{\delta}^n_f(\mu) = \hat{f}(\rho^n(\mu), h^{n-1}(\mu), \hat{\delta}^{n-1}_f(\mu))
  \]
  \[
  h^n(\mu) = g(\rho^n(\mu), h^{n-1}(\mu), \hat{\delta}^{n-1}_f(\mu))
  \]

+ latent variables \(h^n(\mu)\): enable capturing non-local dependencies
+ features \(\rho^n(\mu)\): residual-based (and cheaply computable)
+ general formulation encompasses ARX, LARX, RNN, LSTM, GRU
**Example:** long short-term memory (LSTM)

\[
\hat{\delta}_f^n (\mu) = \hat{f}(\rho^n(\mu), h_{n-1}^n(\mu)) \\
\hat{h}^n(\mu) = g(\rho^n(\mu), h_{n-1}^{n-1}(\mu))
\]
Training and machine learning: error modeling

1. **Training**: Solve high-fidelity and reduced-order models for \( \mu \in D_{\text{training}} \)
2. **Machine learning**: Construct regression model
3. **Reduction**: predict reduced-order-model error for \( \mu \in D_{\text{query}} \setminus D_{\text{training}} \)

- randomly divide data into (1) training data and (2) testing data
- construct regression function \( \hat{\delta}_f^n \) via cross validation on **training data**
- construct noise model \( \hat{\delta}_\varepsilon^n \) from sample variance on **test data**
Reduction

1. **Training**: Solve high-fidelity and reduced-order models for $\mu \in D_{\text{training}}$

2. **Machine learning**: Construct regression model

3. **Reduction**: predict reduced-order-model error for $\mu \in D_{\text{query}} \setminus D_{\text{training}}$

\[
\begin{align*}
\text{inputs } \mu & \rightarrow \text{reduced-order model} \rightarrow \text{outputs } q^n_{\text{ROM}}, \\
& \quad n = 1, \ldots, T \\
& \quad \text{features } \rho^n, \quad n = 1, \ldots, T \\
& \quad \text{regression model} \\
& \quad \hat{\delta}^n(\mu) = \hat{\delta}^n_f(\mu) + \hat{\delta}^n_\epsilon(\mu) \\
& \quad h^n(\mu) = g(\rho^n(\mu), h^{n-1}(\mu), \hat{\delta}^{n-1}_f(\mu)) \\
& \quad \hat{\delta}^n_f(\mu) = f(\rho^n(\mu), h^{n-1}(\mu), \hat{\delta}^{n-1}_f(\mu)) \\
& \quad \hat{q}^n_{\text{HFM}}(\mu) = q^n_{\text{ROM}}(\mu) + \hat{\delta}^n(\mu)
\end{align*}
\]

- Latent dynamics learning
Application: Advection–diffusion equation

+ Regression methods: classical RNN and LSTM most accurate
**Application: Advection–diffusion equation**

![Image](image.png)

**regression methods**

<table>
<thead>
<tr>
<th>GP</th>
<th>kNN</th>
<th>ANN</th>
<th>ARX (NRT)</th>
<th>ARX (RT)</th>
<th>ANN-I (NRT)</th>
<th>ANN-I (RT)</th>
<th>LARX</th>
<th>RNN</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
<td><img src="color.png" alt="Color" /></td>
</tr>
</tbody>
</table>

**features**

+ **regression methods**: classical RNN and LSTM most accurate
+ **features**: only 7 residual samples needed for good accuracy

Table 4:

<table>
<thead>
<tr>
<th>$D$</th>
<th>$C$</th>
<th>$\log_{10}(FVU)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
</tr>
<tr>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
</tr>
<tr>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
</tr>
<tr>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
<td><img src="value.png" alt="Value" /></td>
</tr>
</tbody>
</table>

In summary, for the advection–diffusion equation, the LSTM model yielded best performance, followed by RNN and ANN. Including time slightly improves performance for these methods. Finally, we observe that for the deterministic regression-function models in Category 2, both training methods yield comparable results. We also observe that including time slightly improves performance for these methods. Finally, we observe that for the deterministic regression-function models in Category 2, both training methods yield comparable results. We also observe that including time slightly improves performance for these methods.
Questions?