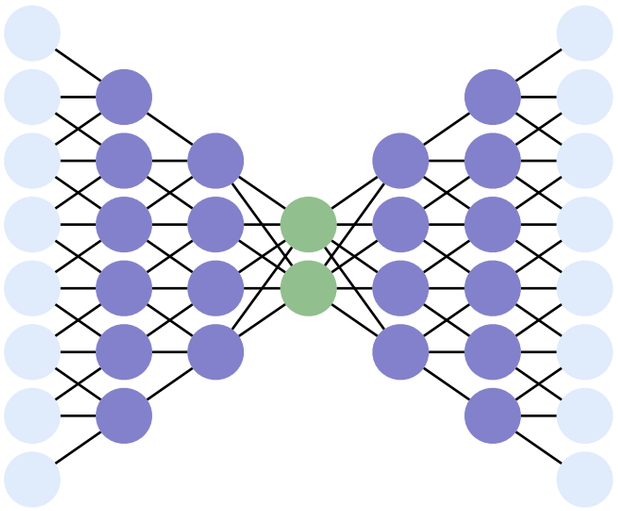
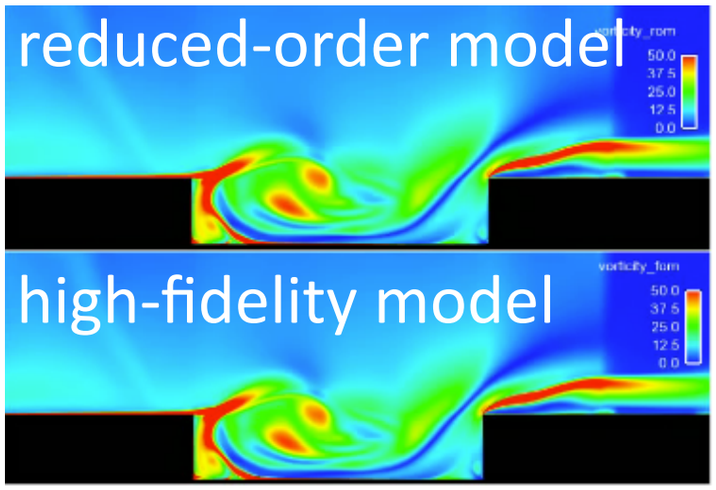
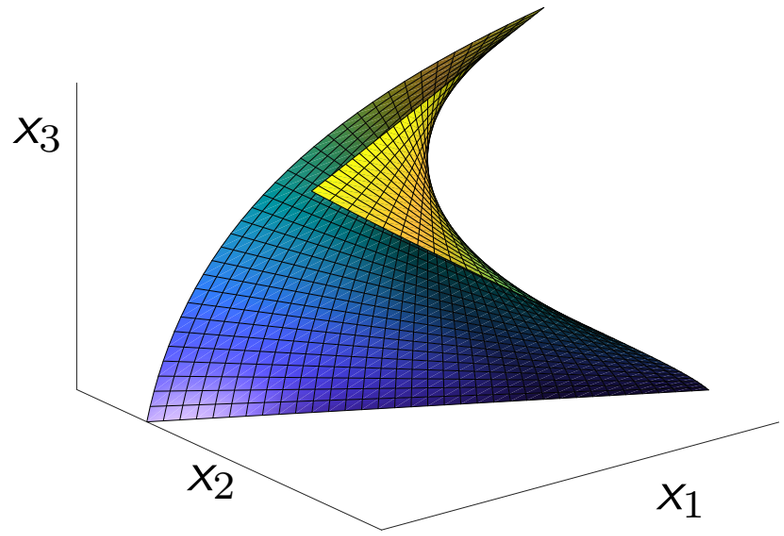
Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders







Kookjin Lee and Kevin Carlberg

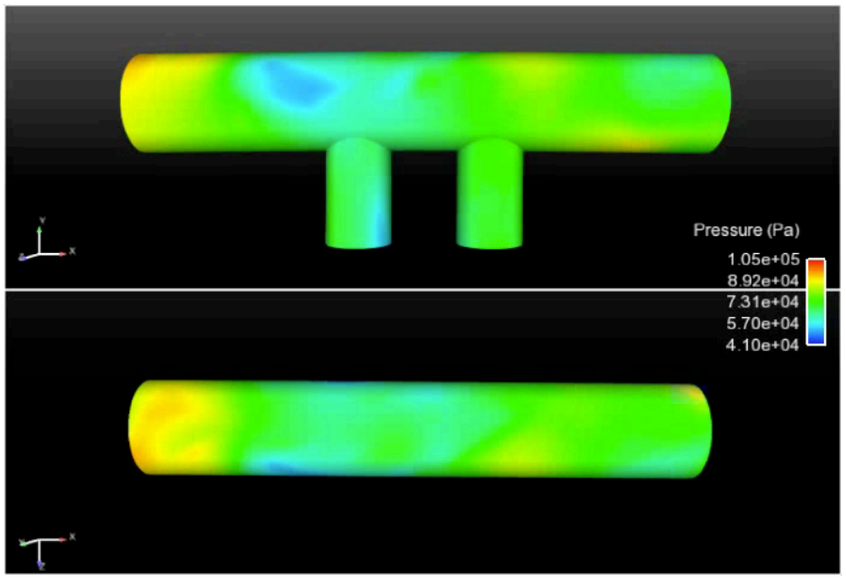
Sandia National Laboratories

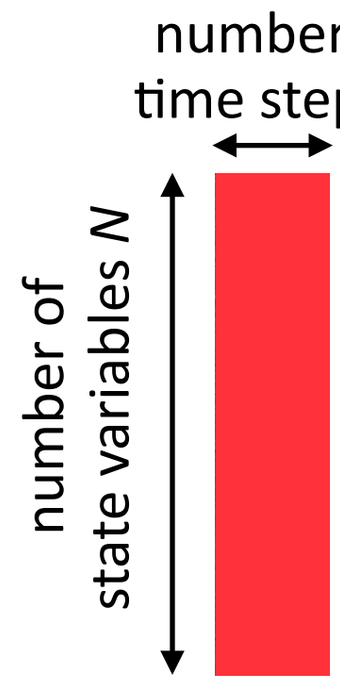
USNCCM

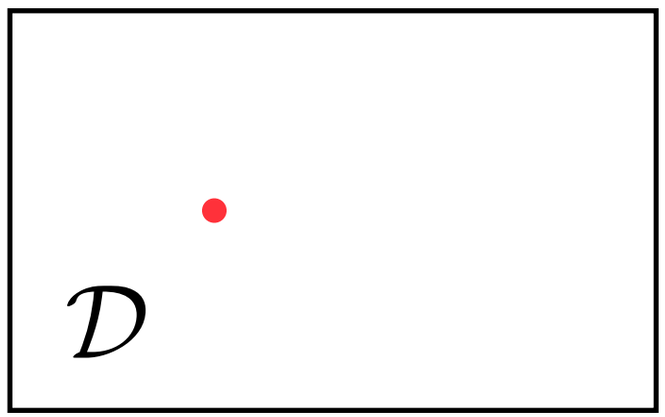
June 29, 2019

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

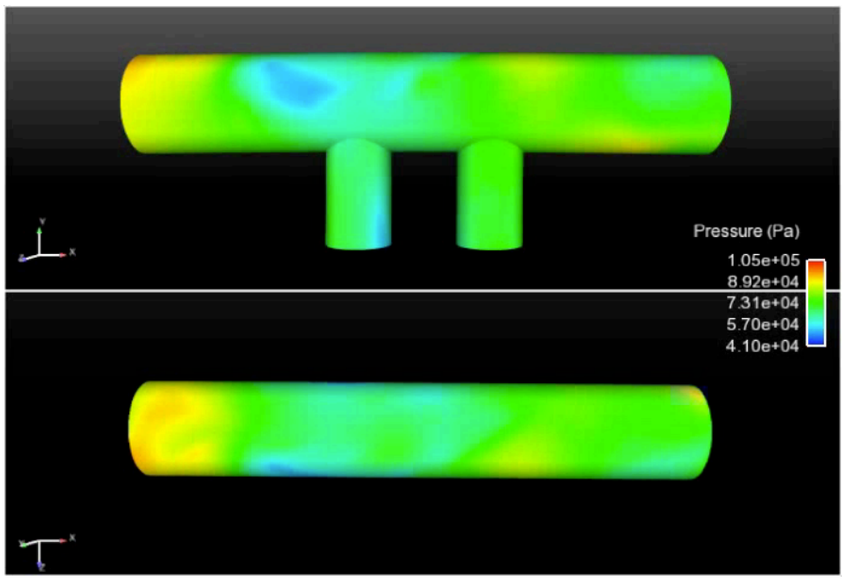




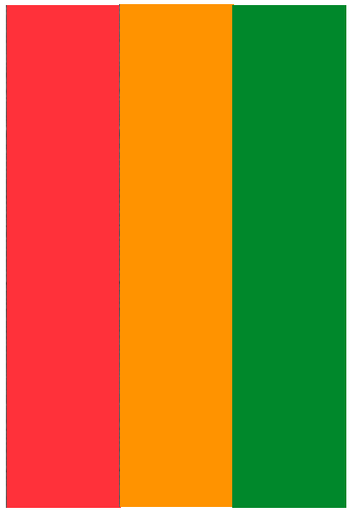


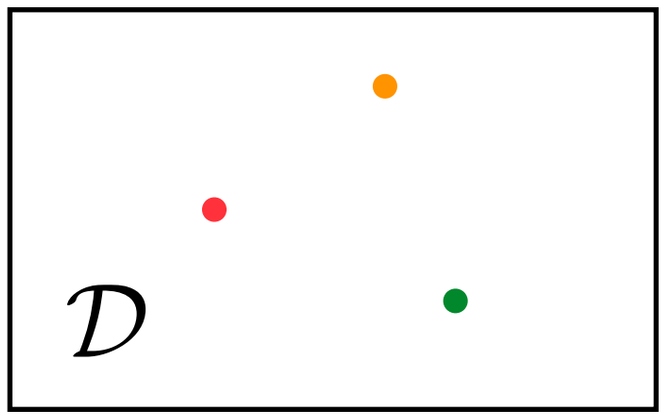
1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

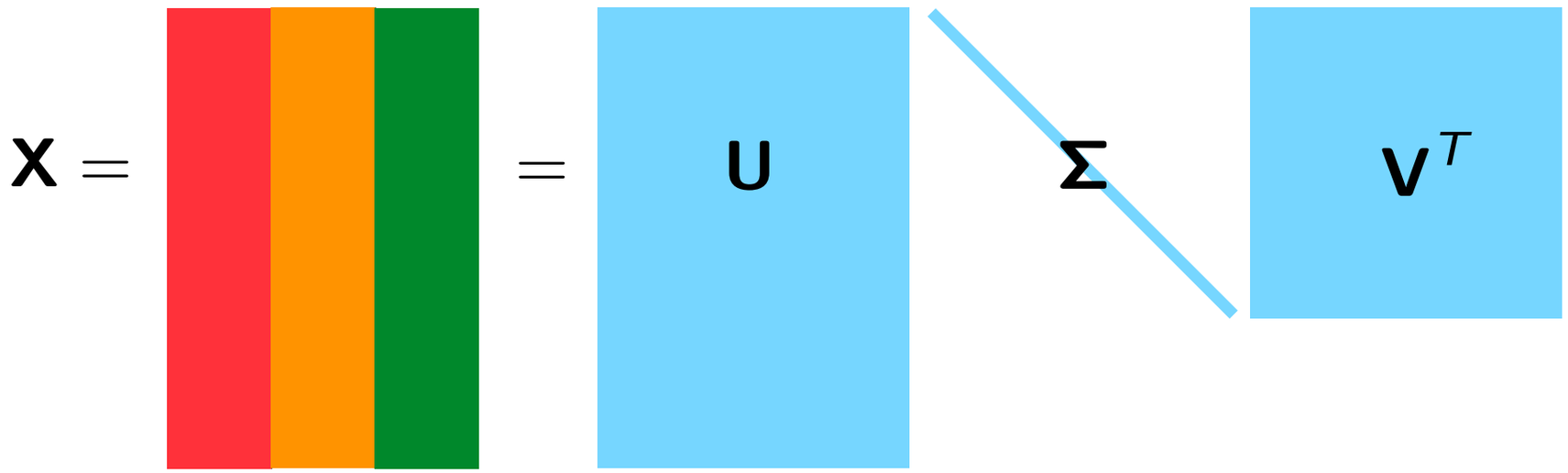


$\mathbf{x} =$

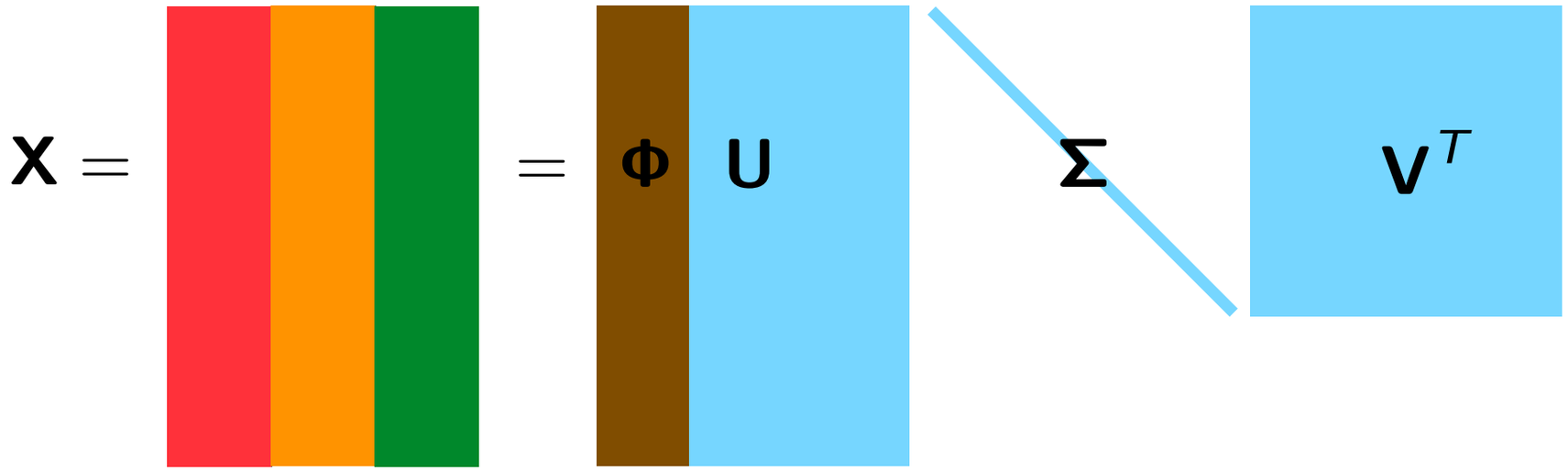




1. *Training:* Solve ODE for $\mu \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



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 $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



Φ *columns are principal components of the spatial simulation data*

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 $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$$

ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t)$$

$$\mathbf{r}\left(\frac{d\mathbf{x}}{dt}, \mathbf{x}, t\right) = \mathbf{0}$$

residual minimization →

Galerkin ODE

$$\frac{d\hat{\mathbf{x}}}{dt} = \boldsymbol{\Phi}^T \mathbf{f}(\boldsymbol{\Phi}\hat{\mathbf{x}}, t)$$

$$\boldsymbol{\Phi}\frac{d\hat{\mathbf{x}}}{dt}(\boldsymbol{\Phi}\hat{\mathbf{x}}, t) = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \|\mathbf{r}(\mathbf{v}, \boldsymbol{\Phi}\hat{\mathbf{x}}, t)\|_2$$

time discretization ↓

OΔE

$$\mathbf{r}^n(\mathbf{x}^n) = 0, \quad n = 1, \ldots, T$$

residual minimization ←

LSPG OΔE

[C., Bou-Mosleh, Farhat, 2011]

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \|\mathbf{r}^n(\mathbf{v})\|_2, \quad n = 1, \ldots, T$$

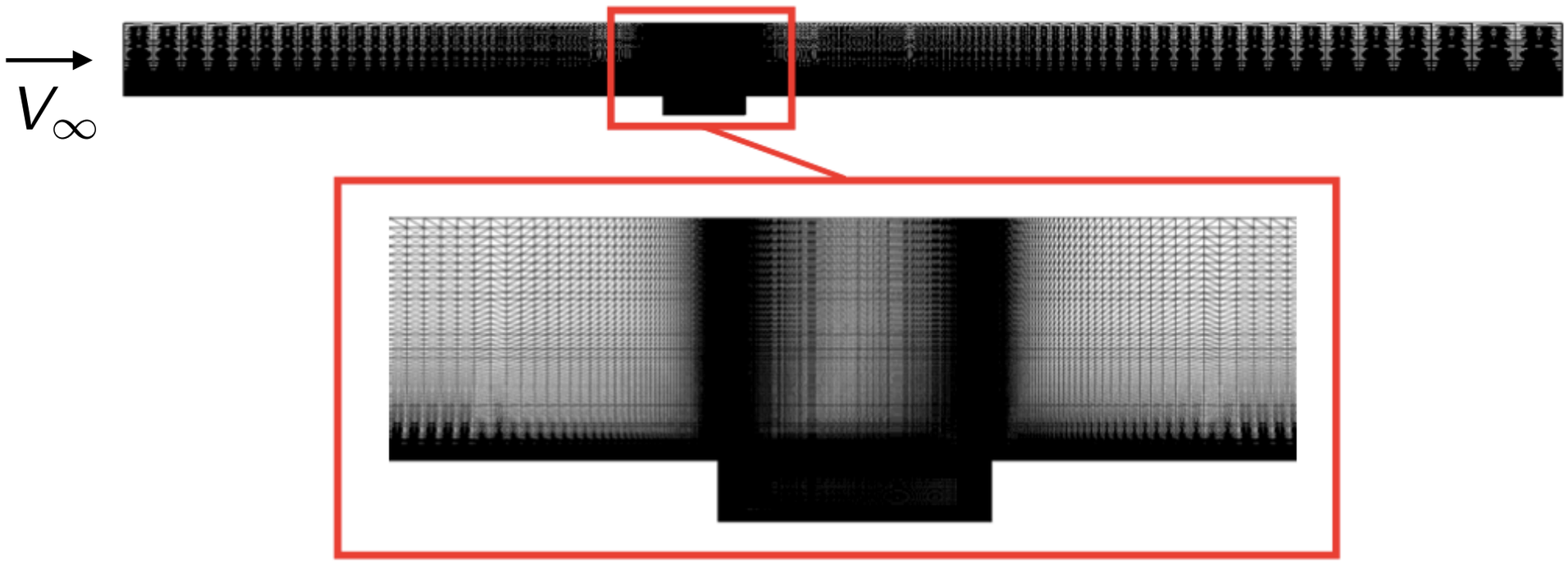
time discretization ↓

Galerkin OΔE

$$\boldsymbol{\Phi}^T \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{x}}^n) = 0, \quad n = 1, \ldots, T$$

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

Captive carry



- Unsteady Navier–Stokes
- Re = 6.3 x 10^6
- $M_\infty = 0.6$

Spatial discretization

- 2nd-order finite volume
- DES turbulence model
- 1.2×10^6 degrees of freedom

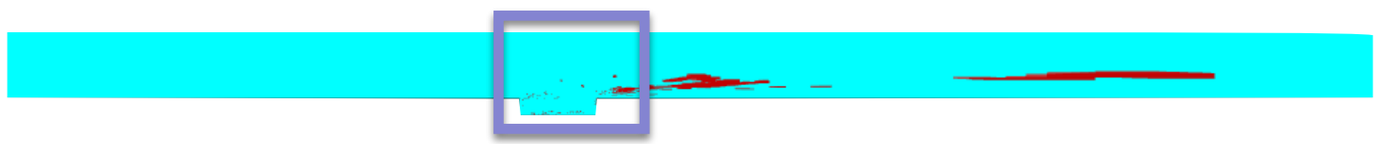
Temporal discretization

- 2nd-order BDF
- Verified time step $\Delta t = 1.5 \times 10^{-3}$
- 8.3×10^3 time instances

LSPG ROM with sample mesh [C., Barone, Antil, 2017]

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_{\Theta}$$

sample mesh

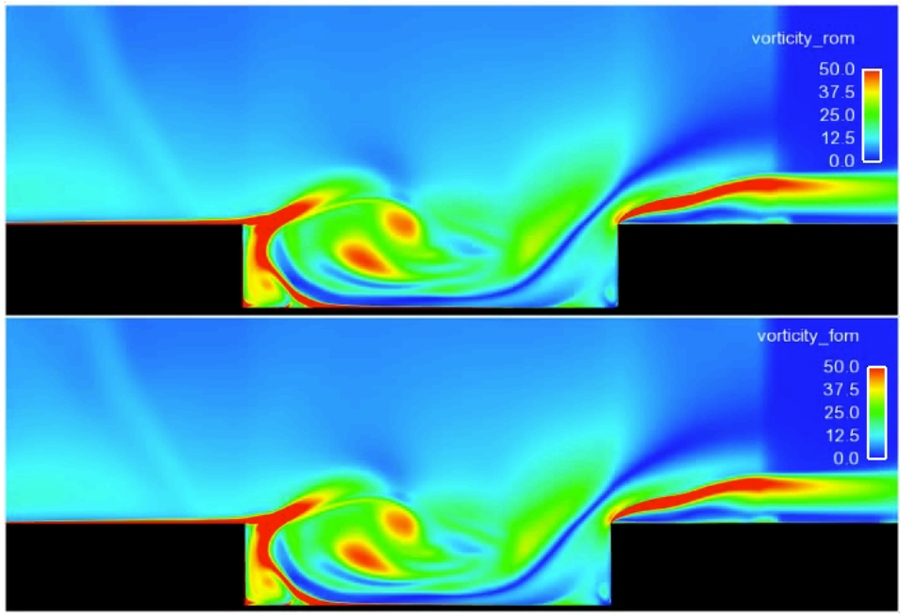


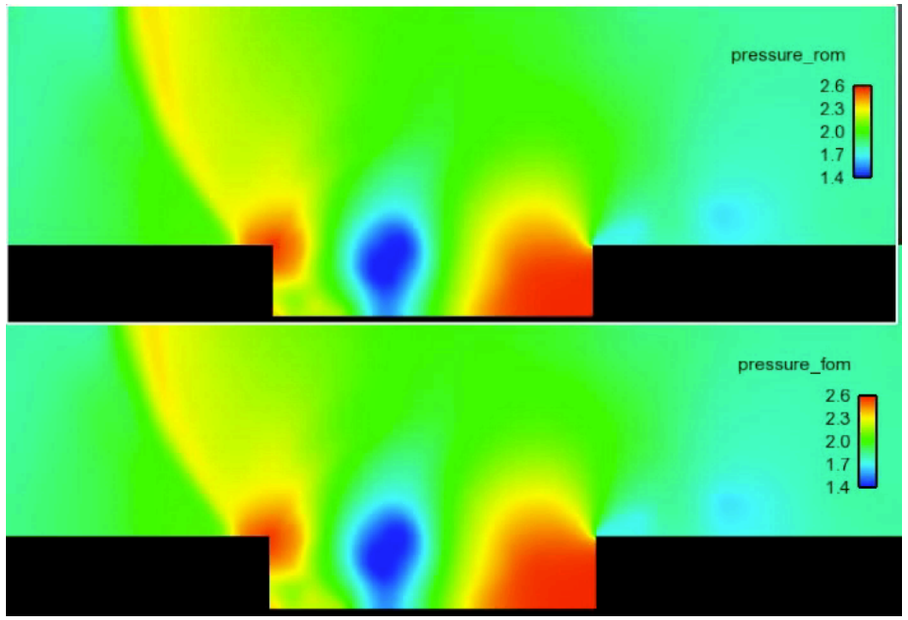
+ *HPC on a laptop*

vorticity field *pressure field*

LSPG ROM
32 min, 2 cores

high-fidelity
5 hours, 48 cores





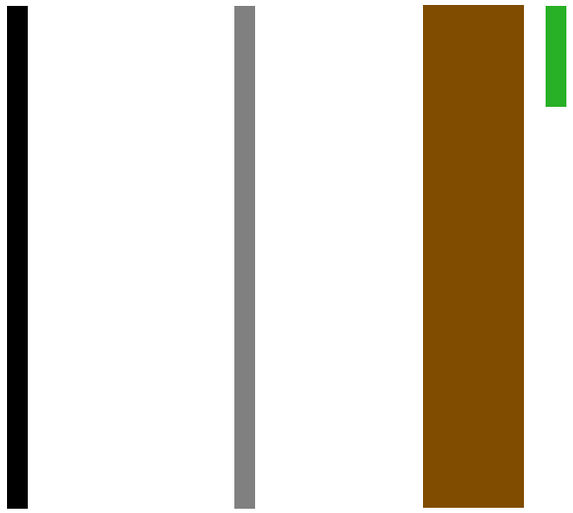
+ *229x savings in core–hours*

+ *< 1% error in time-averaged drag*

… so why doesn't everyone use ROMs?

Good generalization performance is not guaranteed

$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$$



1) Linear-subspace assumption is strong ← *This talk*

- Lee and C. “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders,” arXiv e-Print, 1812.08373 (2018).

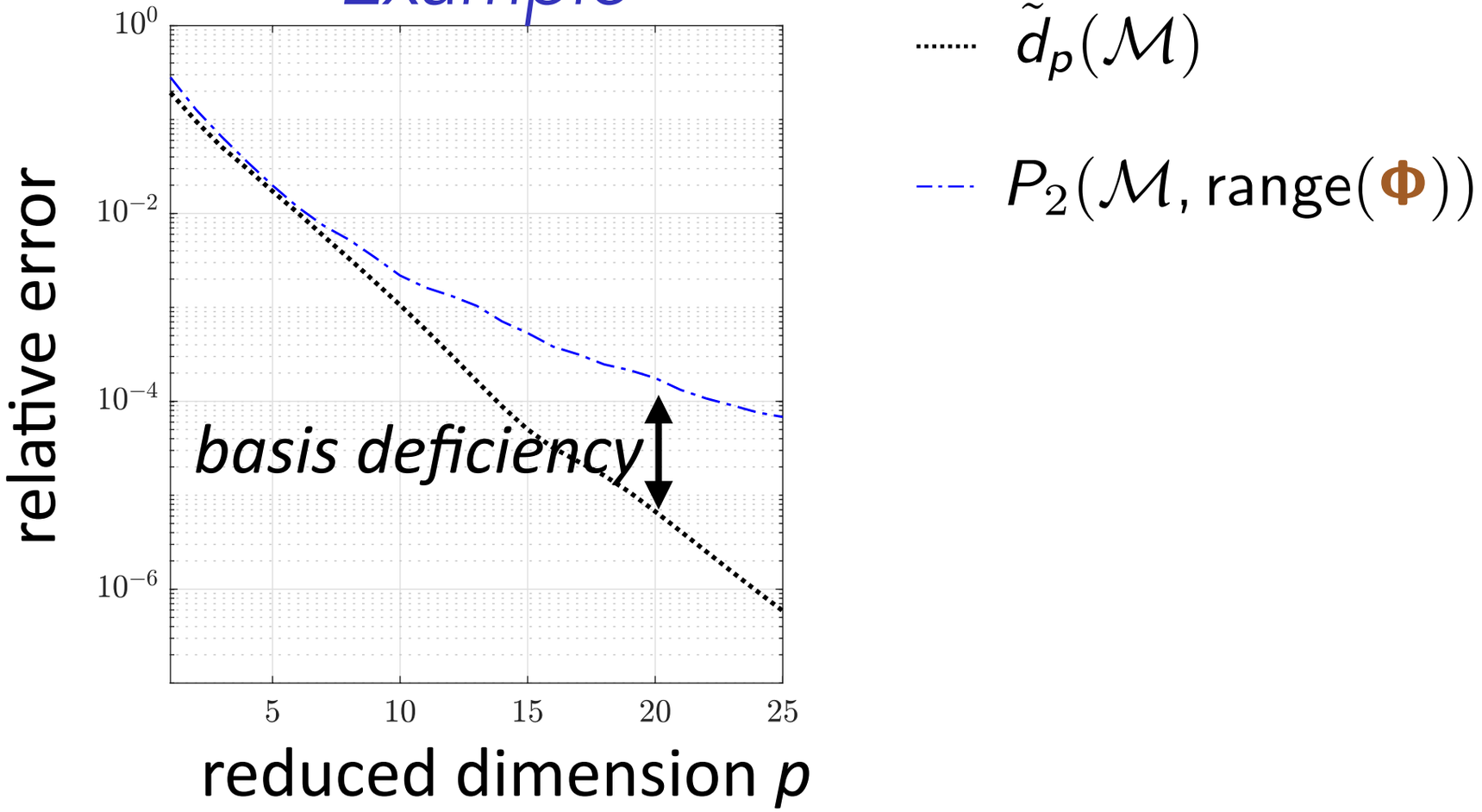
2) Accuracy limited by training data used to construct Φ

- Etter and C. “Online adaptive basis refinement and compression for reduced-order models,” arXiv e-Print, 1902.10659 (2019).
- C. “Adaptive h-refinement for reduced-order models,” Int J Numer Meth Eng, 102(5):1192–1210, 2015.

Kolmogorov-width limitation of linear subspaces

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

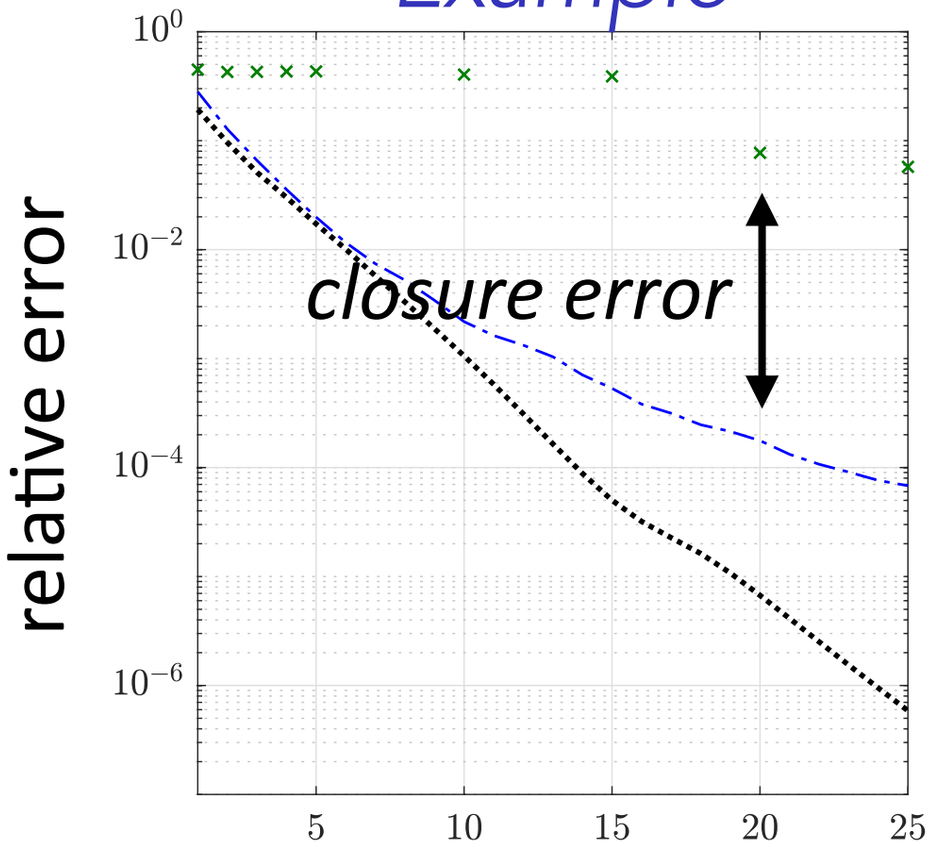
Example



Kolmogorov-width limitation of linear subspaces

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

Example



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

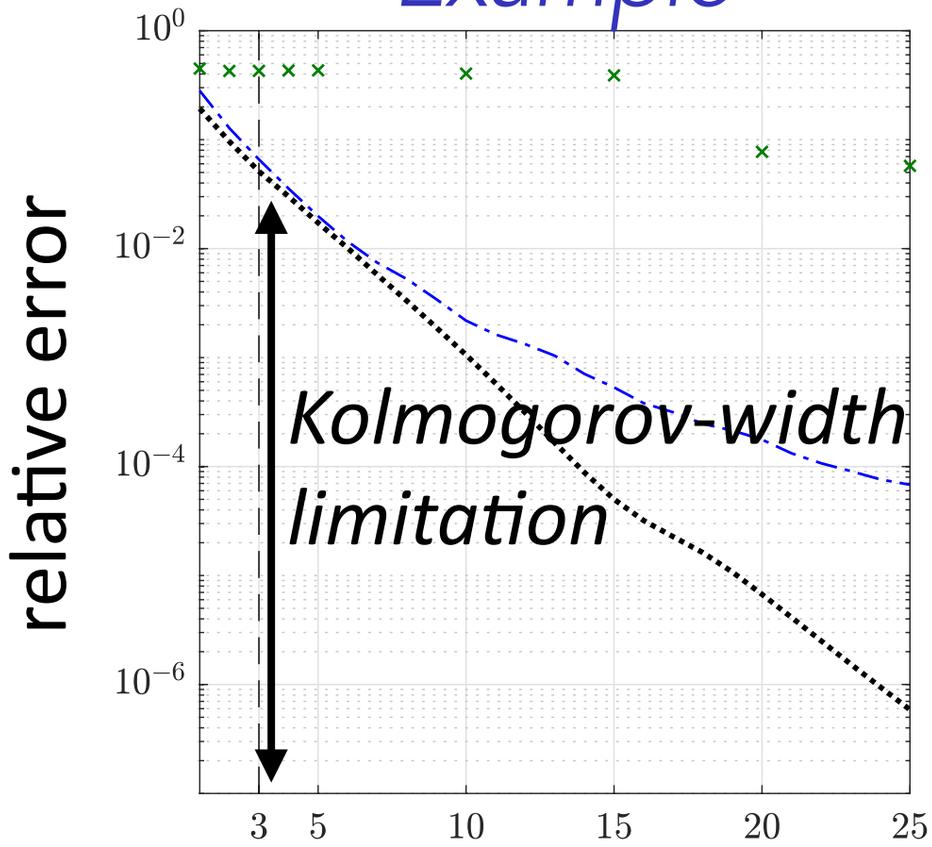
$P_2(\mathcal{M}, \text{range}(\boldsymbol{\Phi}))$

$$\frac{\sqrt{\sum_{\mathbf{x} \in \mathcal{M}} \|\mathbf{x} - \tilde{\mathbf{x}}_{\text{LSPG}}\|^2}}{\sqrt{\sum_{\mathbf{x} \in \mathcal{M}} \|\mathbf{x}\|^2}}$$

Kolmogorov-width limitation of linear subspaces

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

Example



- 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

A priori construction of local linear subspaces

[Dihlmann et al., 2011; Drohmann et al., 2011; Amsallem, Zahr, Farhat, 2012; Peherstorfer et el., 2014; Taddei et al., 2015]

+ Tailored bases for local regions of time/spatial 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]
- Solutions lack optimality [Gu, 2011; Kashima, 2016; Hartman and Mestha, 2017]

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 snapshot 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 snapshot data as POD

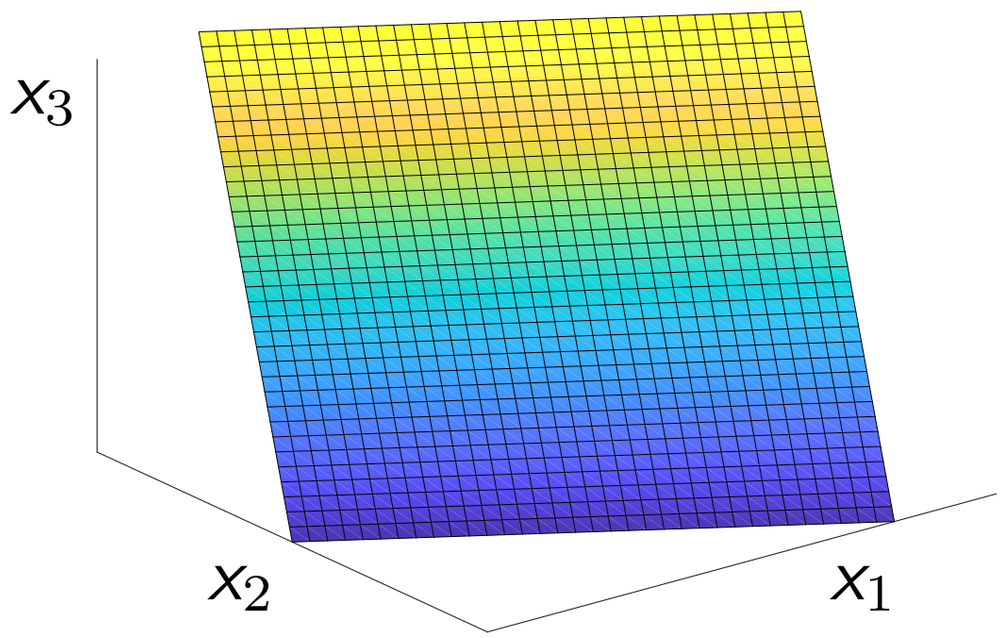
Deep convolutional autoencoders

Nonlinear trial manifold

Linear trial subspace

$$\text{range}(\boldsymbol{\Phi}) := \{\boldsymbol{\Phi}\hat{\mathbf{x}} \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\}$$

example
N=3
p=2



state

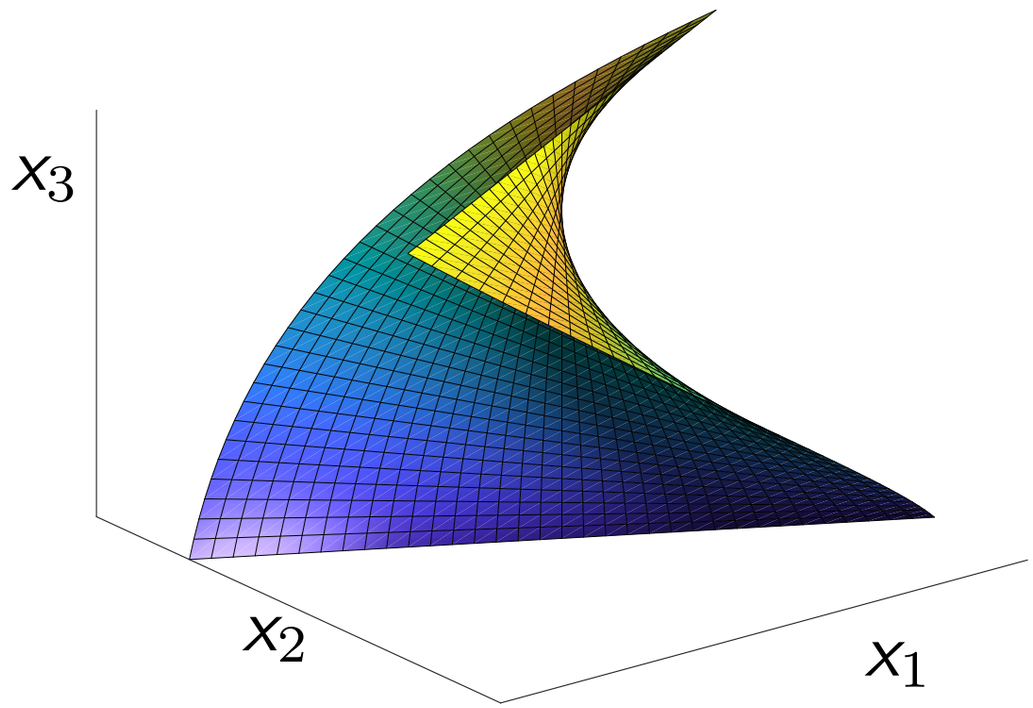
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\,\hat{\mathbf{x}}(t) \in \text{range}(\boldsymbol{\Phi})$$

velocity

$$\frac{d\mathbf{x}}{dt} \approx \frac{d\tilde{\mathbf{x}}}{dt} = \boldsymbol{\Phi}\frac{d\hat{\mathbf{x}}}{dt} \in \text{range}(\boldsymbol{\Phi})$$

Nonlinear trial manifold

$$\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\}$$



$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \mathbf{g}(\hat{\mathbf{x}}(t)) \in \mathcal{S}$$

+ manifold has general structure

$$\frac{d\mathbf{x}}{dt} \approx \frac{d\tilde{\mathbf{x}}}{dt} = \nabla\mathbf{g}(\hat{\mathbf{x}})\frac{d\hat{\mathbf{x}}}{dt} \in T_{\hat{\mathbf{x}}}\mathcal{S}$$

+ kinematically consistent

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 $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

	Subspace ROM Given $\boldsymbol{\Phi}$	***Manifold ROM*** Given $\mathbf{g}(\hat{\mathbf{x}})$
Galerkin	$\frac{d\hat{\mathbf{x}}}{dt} = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\text{argmin}} \left\| \mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{v}}, \boldsymbol{\Phi}\hat{\mathbf{x}}; t) \right\|_2$	$\frac{d\hat{\mathbf{x}}}{dt} = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\text{argmin}} \left\| \mathbf{r}(\nabla\mathbf{g}(\hat{\mathbf{x}})\hat{\mathbf{v}}, \mathbf{g}(\hat{\mathbf{x}}); t) \right\|_2$
	$\Updownarrow$	$\Updownarrow$
	$\frac{d\hat{\mathbf{x}}}{dt} = \boldsymbol{\Phi}^T \mathbf{f}(\boldsymbol{\Phi}\hat{\mathbf{x}}; t)$	$\frac{d\hat{\mathbf{x}}}{dt} = \nabla\mathbf{g}(\hat{\mathbf{x}})^+ \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); t)$
LSPG	$\hat{\mathbf{x}}^n = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\text{argmin}} \left\| \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) \right\|_2$	$\hat{\mathbf{x}}^n = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\text{argmin}} \left\| \mathbf{r}^n(\mathbf{g}(\hat{\mathbf{v}})) \right\|_2$

\+ Satisfy residual minimization

Error bound

Theorem

If the following conditions hold:

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

$$\|\mathbf{x}^n - \mathbf{g}(\hat{\mathbf{x}}^n_{\mathrm{G}})\|_2 \leq \frac{1}{h}\|\mathbf{r}^n_{\mathrm{G}}(\mathbf{g}(\hat{\mathbf{x}}_{\mathrm{G}}))\|_2 + \frac{1}{h}\sum_{\ell=1}^{k}|\gamma_\ell|\|\mathbf{x}^{n-\ell} - \mathbf{g}(\hat{\mathbf{x}}_{\mathrm{G}})\|_2$$

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

+ Manifold LSPG sequentially minimizes the error bound

Goals

Overcome shortcomings of existing methods

+ Enable manifolds with general nonlinear structure
+ Kinematically consistent
+ Satisfy optimality property

Manifold Galerkin and LSPG projection

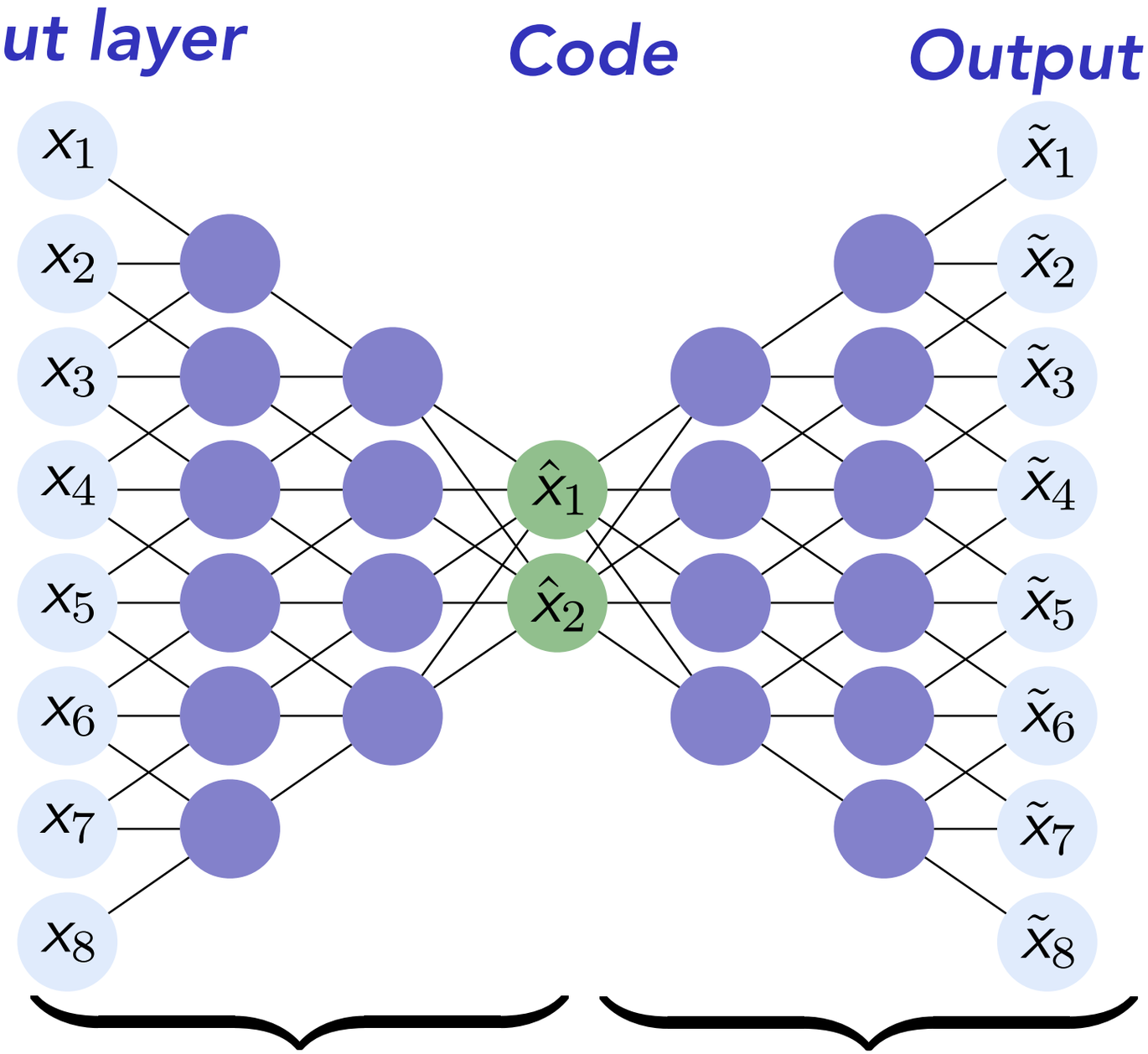
Practical nonlinear-manifold construction

+ No problem-specific knowledge required
+ Use same snapshot data as POD

Deep convolutional autoencoders

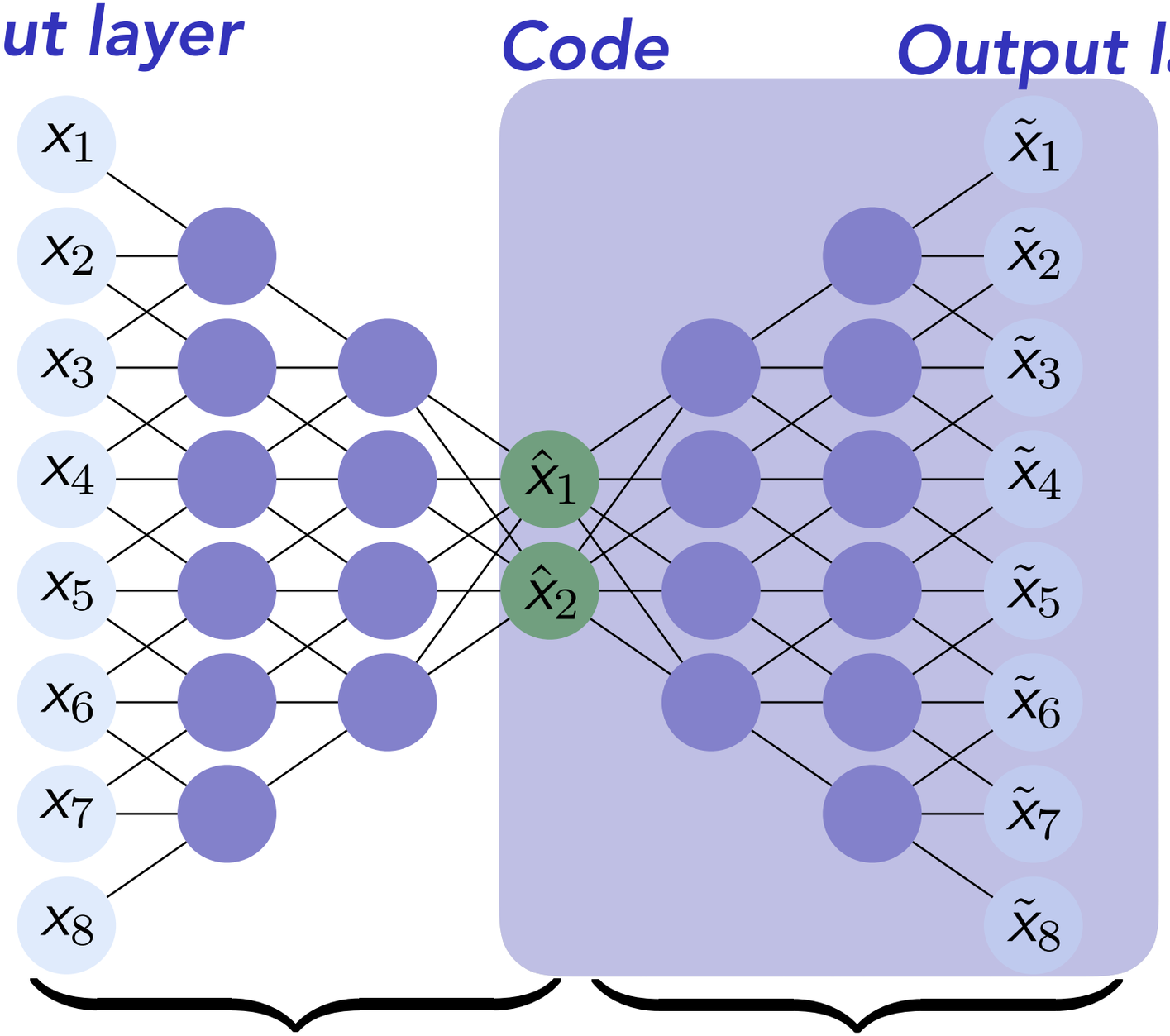
$$\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\}$$

Deep autoencoders



$$\tilde{\mathbf{x}} = \mathbf{h}_{\text{dec}}(\cdot;\boldsymbol{\theta}_{\text{dec}}) \circ \mathbf{h}_{\text{enc}}(\mathbf{x};\boldsymbol{\theta}_{\text{enc}})$$

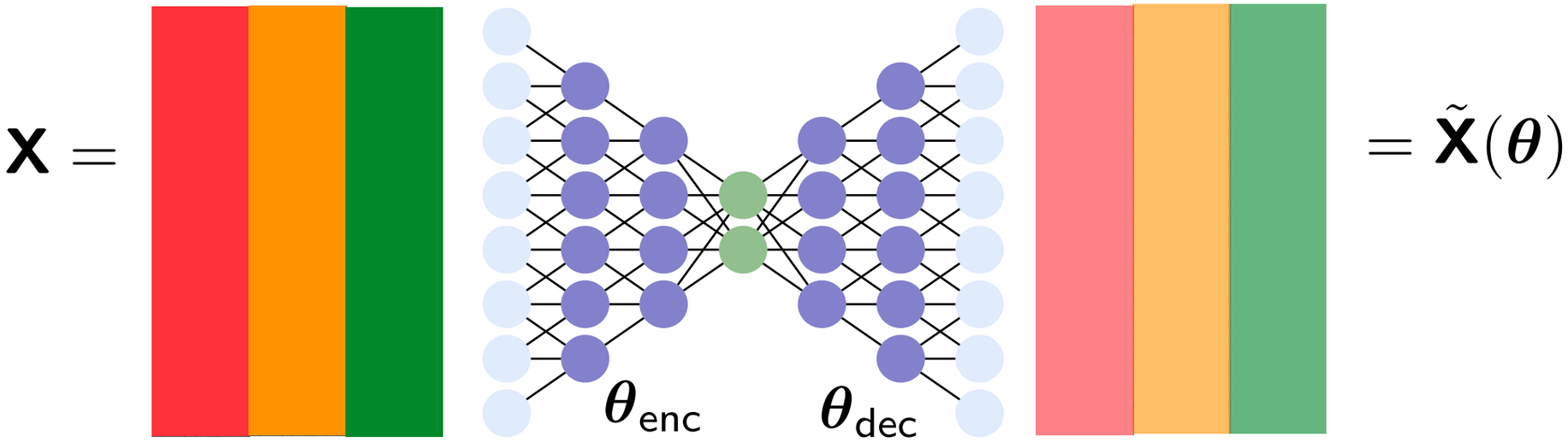
Deep autoencoders



$$\tilde{\mathbf{x}} = \mathbf{h}_{\text{dec}}(\cdot;\boldsymbol{\theta}_{\text{dec}}) \circ \mathbf{h}_{\text{enc}}(\mathbf{x};\boldsymbol{\theta}_{\text{enc}})$$

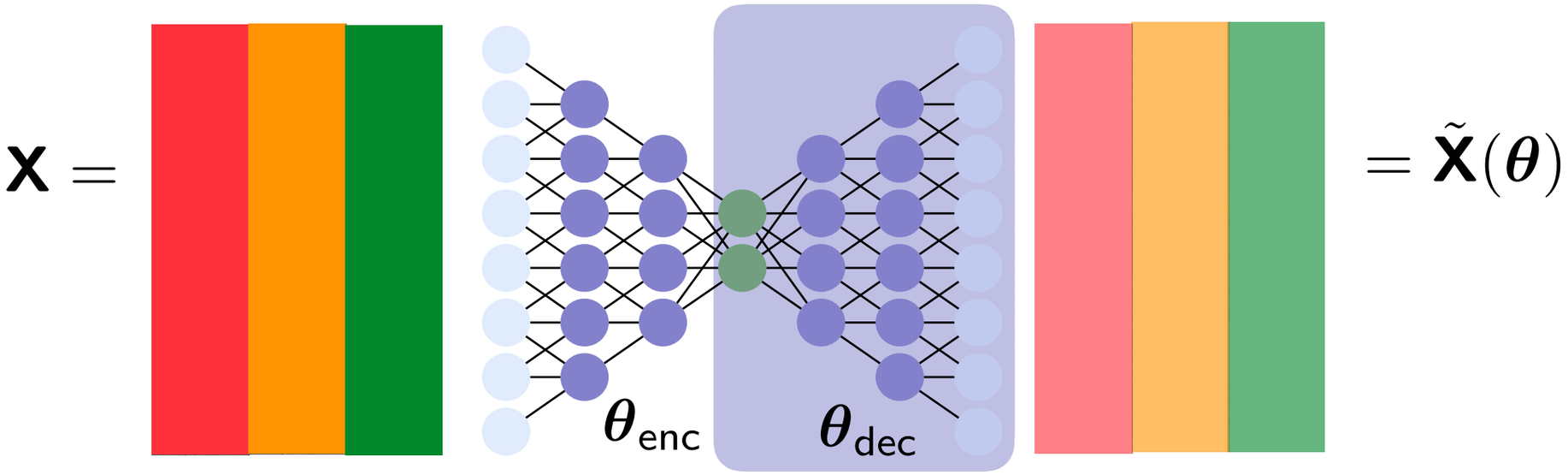
+ If $\tilde{\mathbf{x}} \approx \mathbf{x}$ for parameters $\boldsymbol{\theta}^{\star}_{\text{dec}}$, $\mathbf{g} = \mathbf{h}_{\text{dec}}(\cdot;\boldsymbol{\theta}^{\star}_{\text{dec}})$ produces an accurate manifold

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



- Compute $\boldsymbol{\theta}^\star$ by approximately solving $\underset{\boldsymbol{\theta}}{\text{minimize}} \|\mathbf{X} - \tilde{\mathbf{X}}(\boldsymbol{\theta})\|_F$

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



- Compute $\boldsymbol{\theta}^\star$ by approximately solving $\underset{\boldsymbol{\theta}}{\text{minimize}} \|\mathbf{X} - \tilde{\mathbf{X}}(\boldsymbol{\theta})\|_F$
- Define nonlinear trial manifold by setting $\mathbf{g} = \mathbf{h}_{\text{dec}}(\cdot; \boldsymbol{\theta}^\star_{\text{dec}})$

\+ Same snapshot data, no specialized problem knowledge

Numerical results

1D Burgers' equation

$$\frac{\partial w(x,t;\boldsymbol{\mu})}{\partial t} + \frac{\partial f(w(x,t;\boldsymbol{\mu}))}{\partial x} = 0.02e^{\alpha x}$$

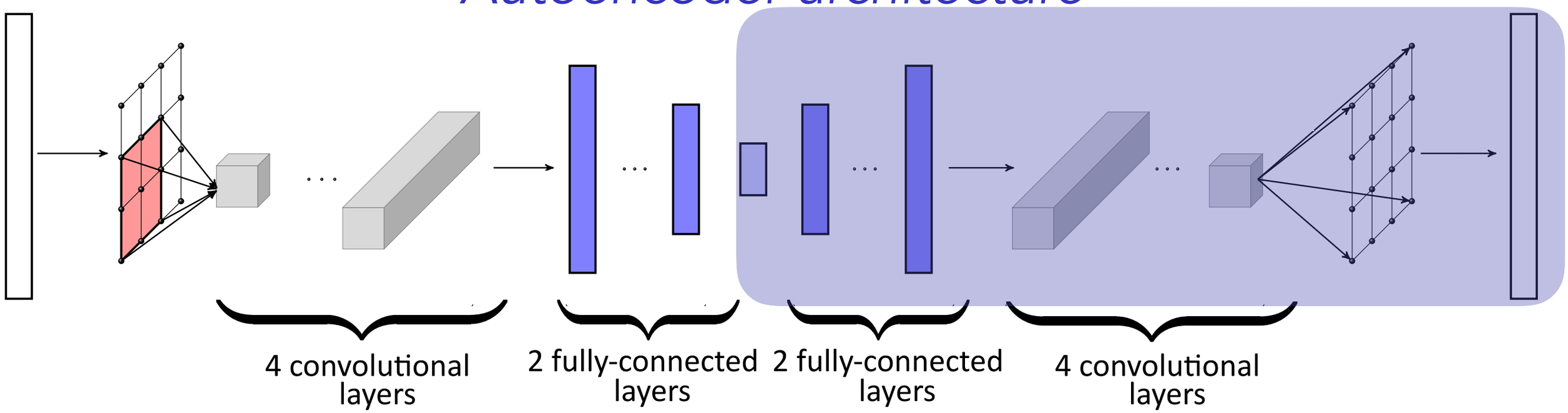
- $\boldsymbol{\mu}$: α, inlet boundary condition
- *Spatial discretization*: finite volume
- *Time integrator:* backward Euler

2D reacting flow

$$\frac{\partial \mathbf{w}(\vec{x},t;\boldsymbol{\mu})}{\partial t} = \nabla \cdot (\kappa \nabla \mathbf{w}(\vec{x},t;\boldsymbol{\mu})) - \mathbf{v} \cdot \nabla \mathbf{w}(\vec{x},t;\boldsymbol{\mu}) + \mathbf{q}(\mathbf{w}(\vec{x},t;\boldsymbol{\mu});\boldsymbol{\mu})$$

- $\boldsymbol{\mu}$: two terms in reaction
- *Spatial discretization*: finite difference
- *Time integrator*: BDF2

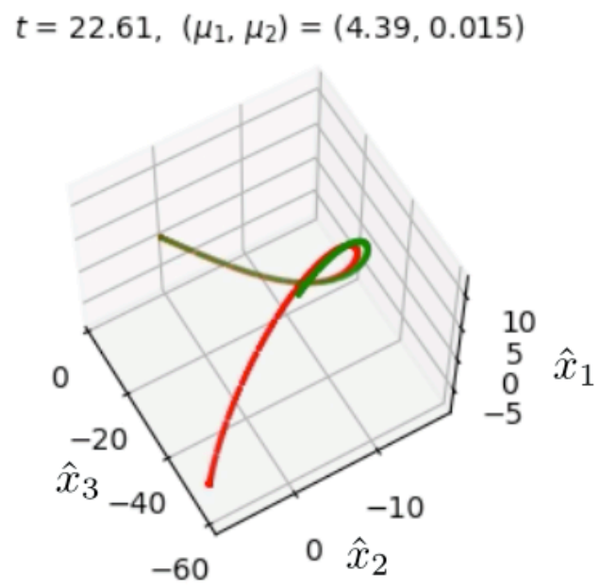
Autoencoder architecture



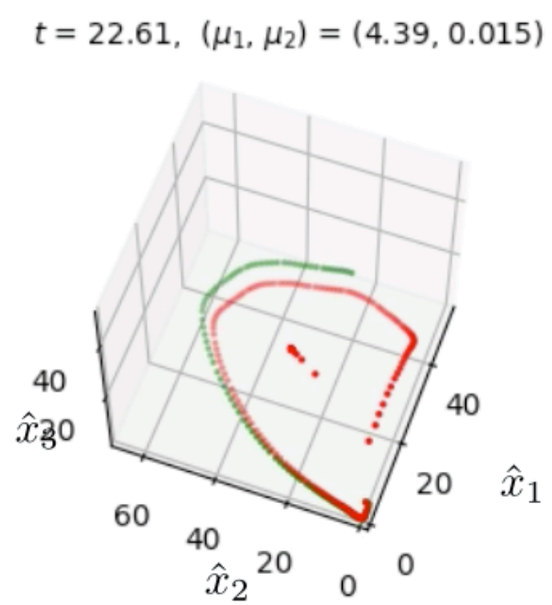
Manifold interpretation: Burgers' equation

FOM

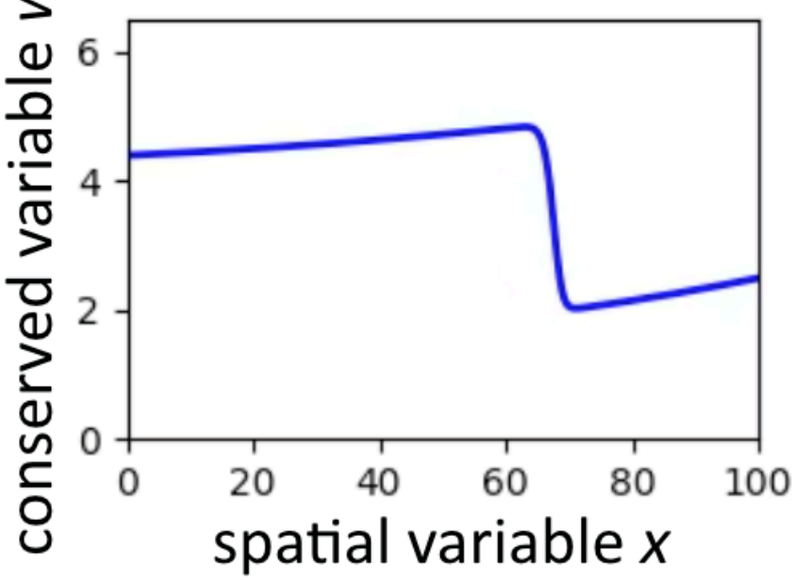
POD, p=3
projection



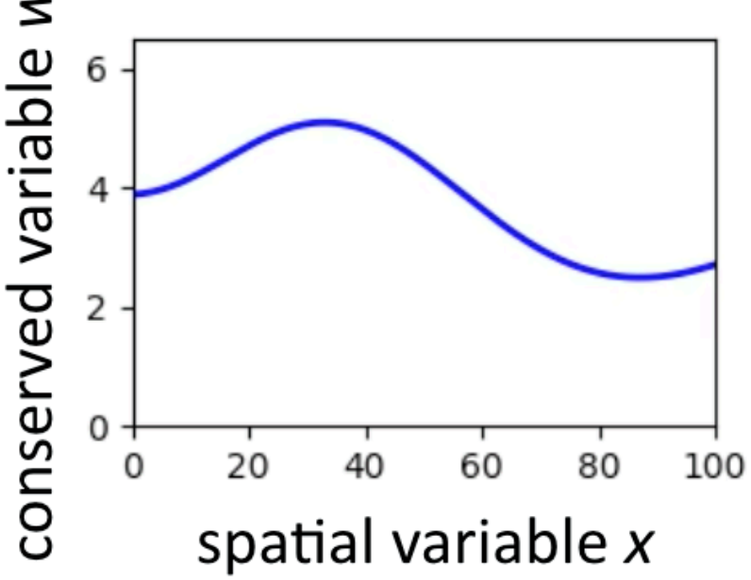
Autoencoder, p=3
projection



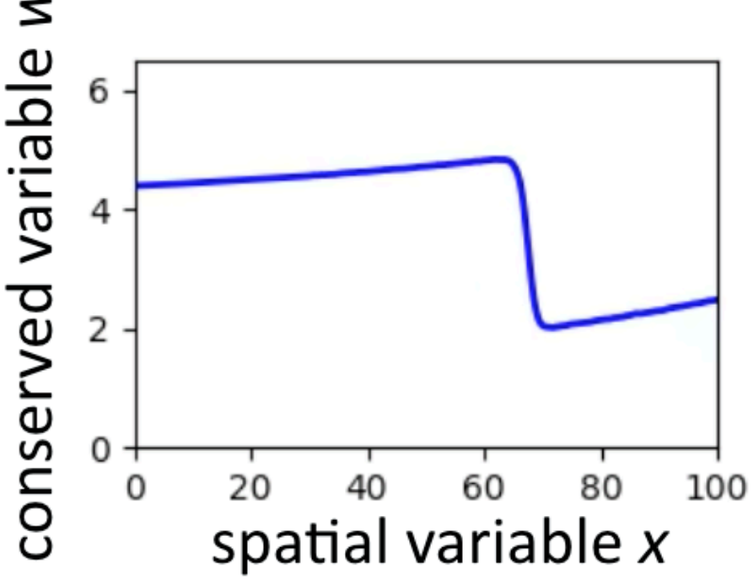
solution



reconstruction



reconstruction



\+ *Projection error onto 3-dimensional manifold nearly perfect*

Manifold LSPG outperforms optimal linear subspace

1D Burgers' equation

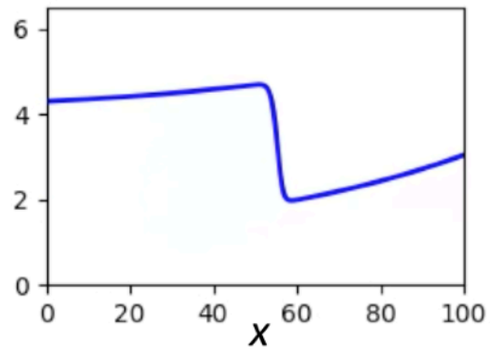
conserved variable

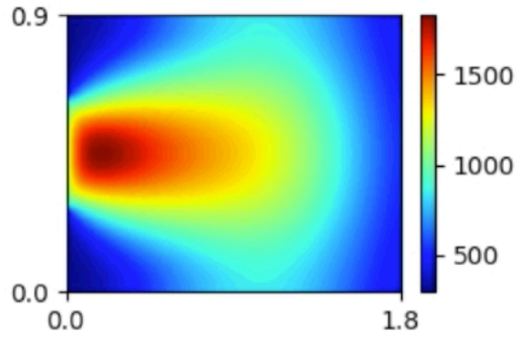
2D reacting flow

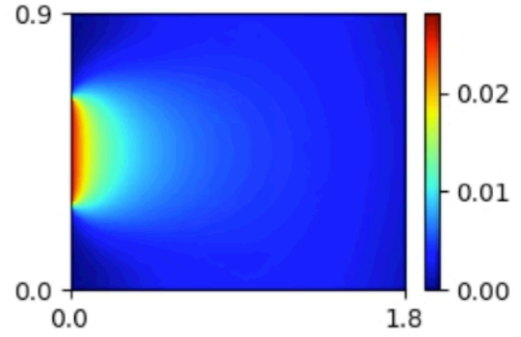
temperature

H_2 fraction

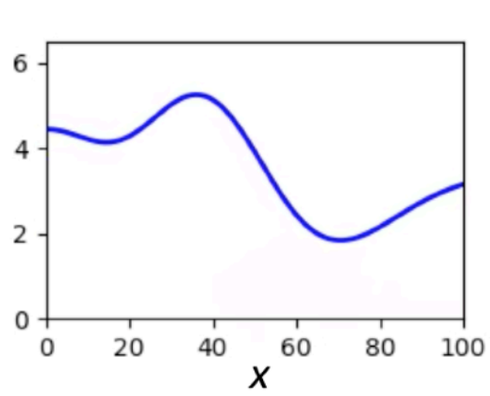
high-fidelity model

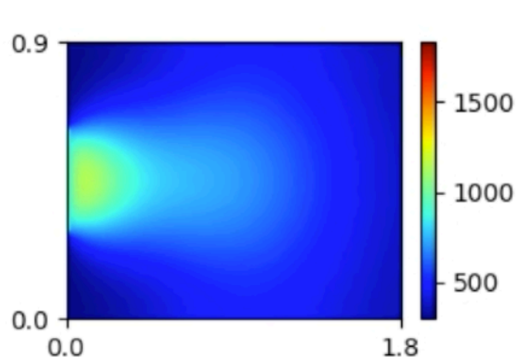


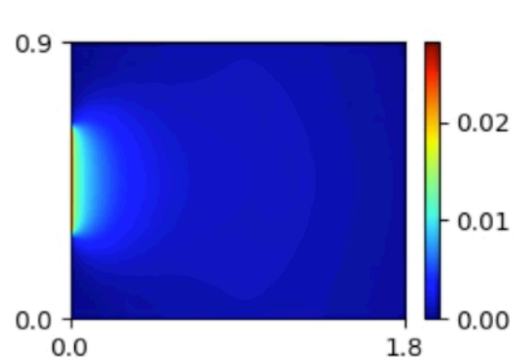




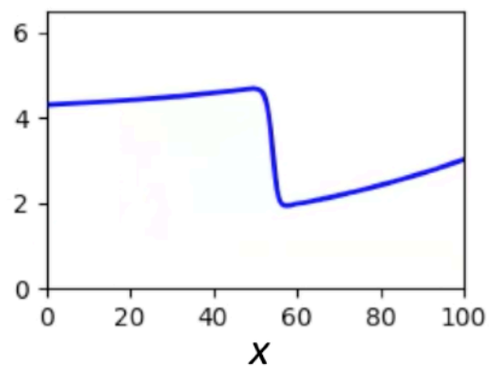
POD-LSPG p=5

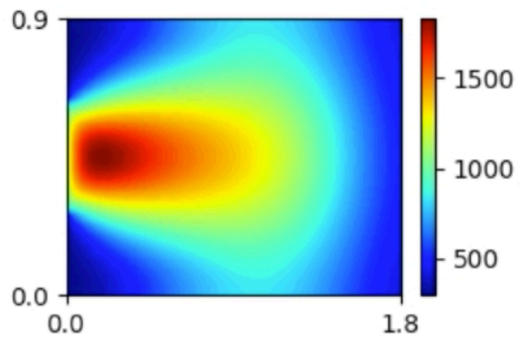


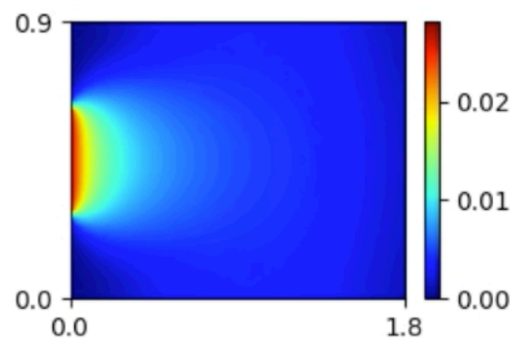




Manifold LSPG p=5

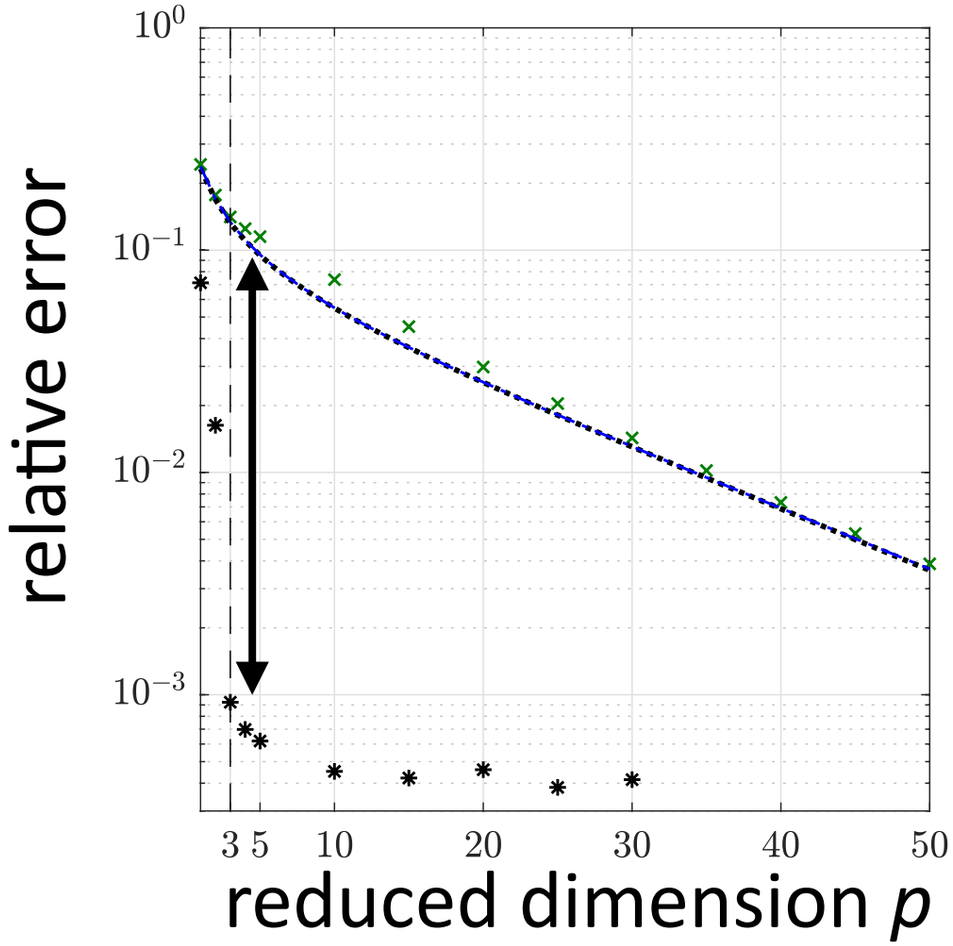




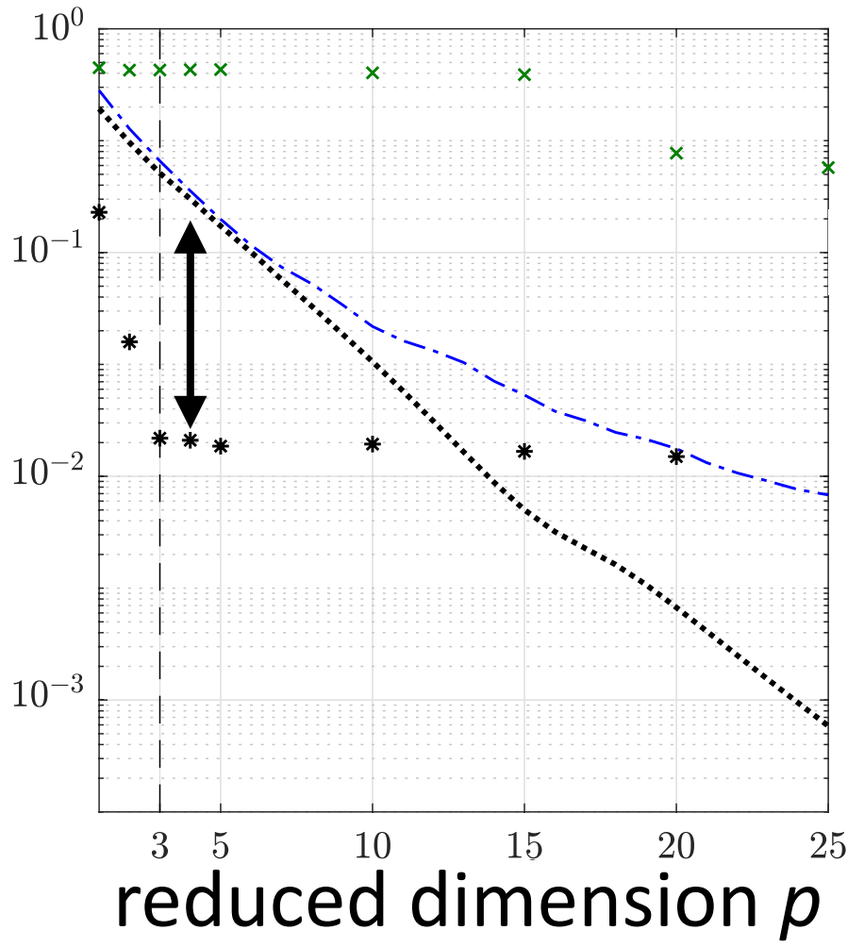


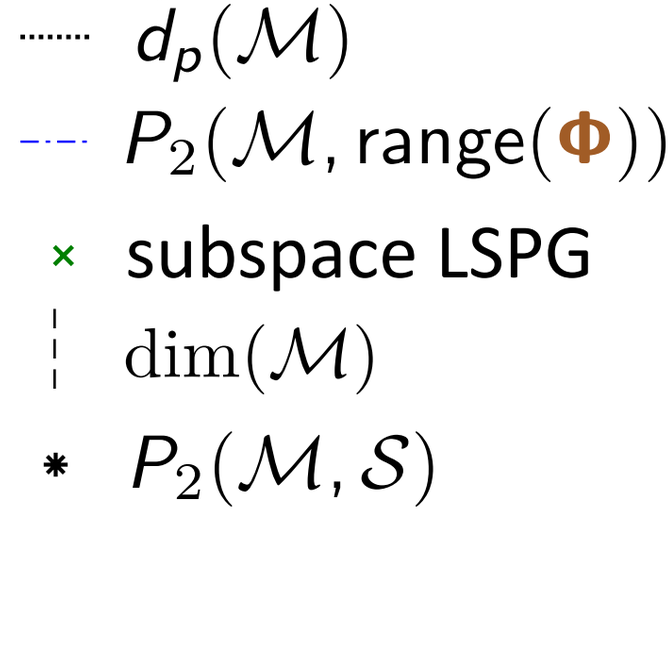
Method improves generalization performance

Burgers' equation



Reacting flow

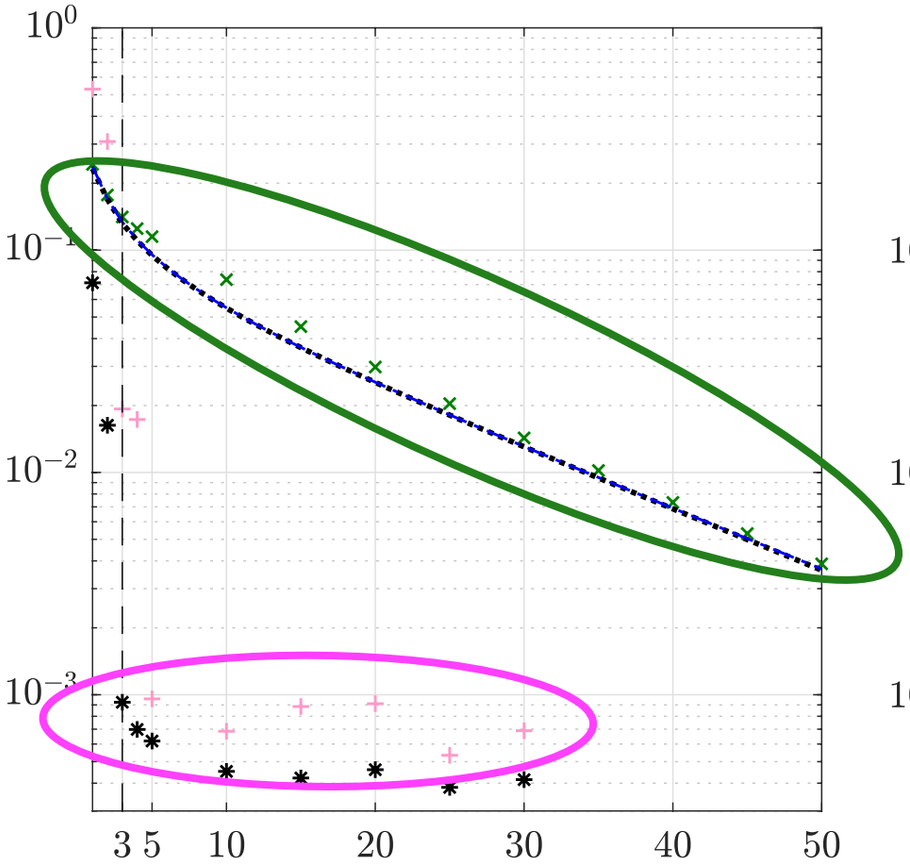




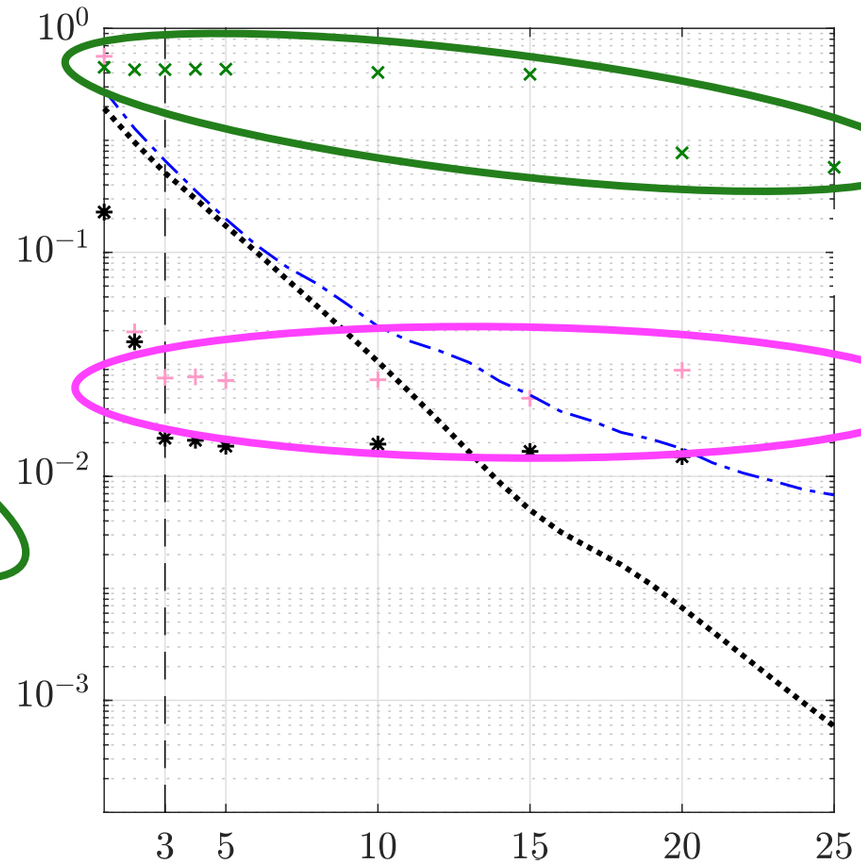
+ Autoencoder manifold significantly better than optimal linear subspace

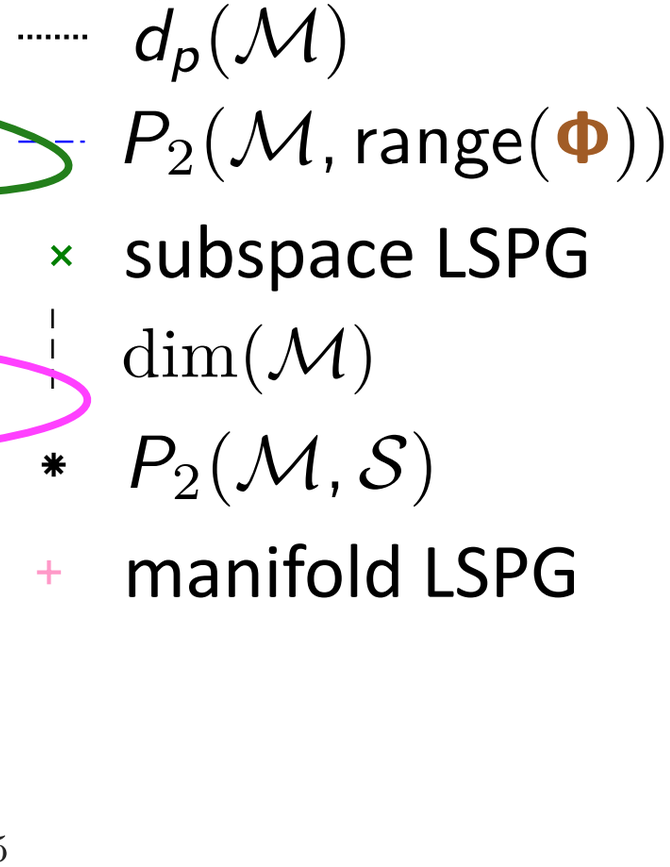
Method improves generalization performance

Burgers' equation



Reacting flow

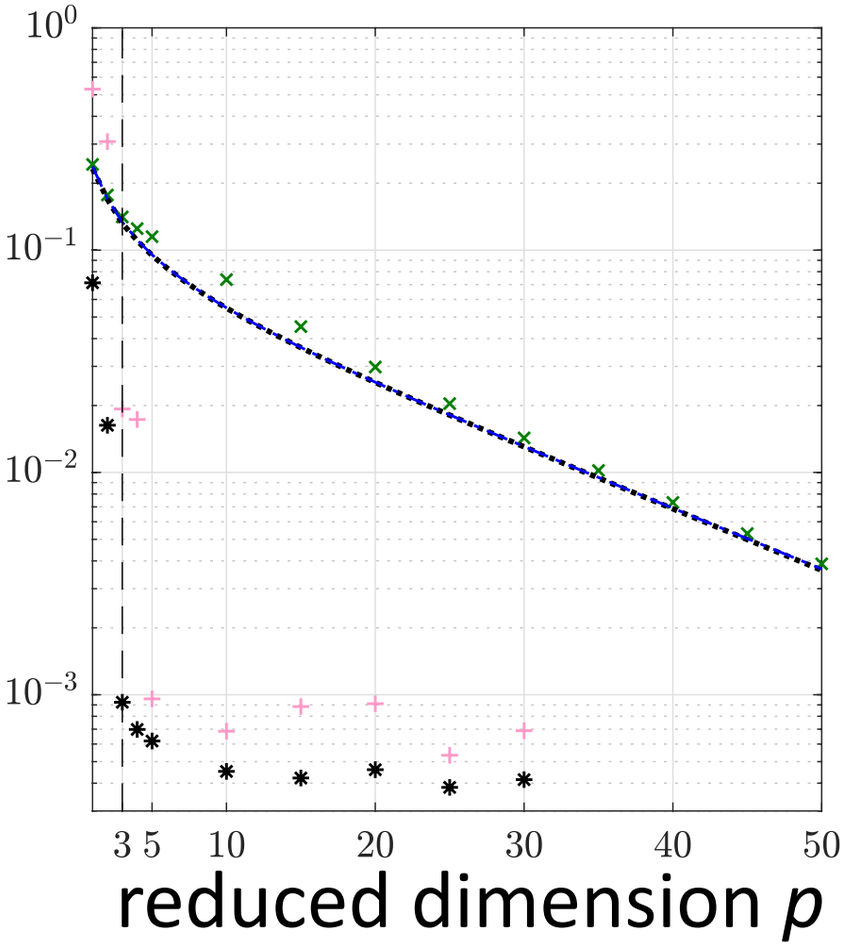




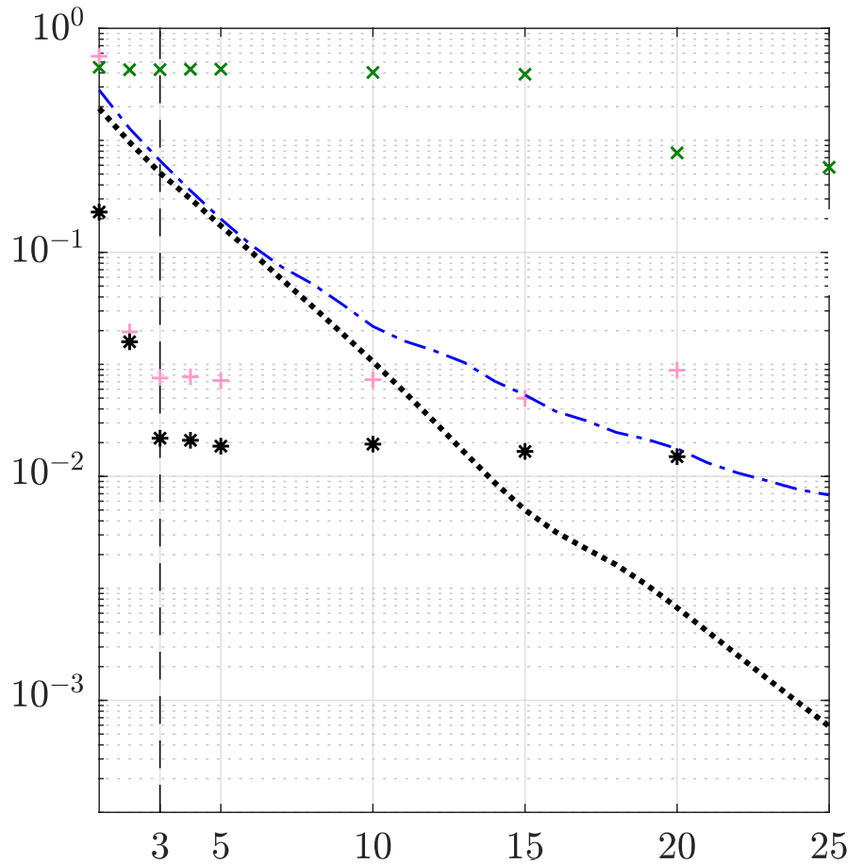
+ Autoencoder manifold significantly better than optimal linear subspace
+ Manifold LSPG orders-of-magnitude more accurate than subspace LSPG

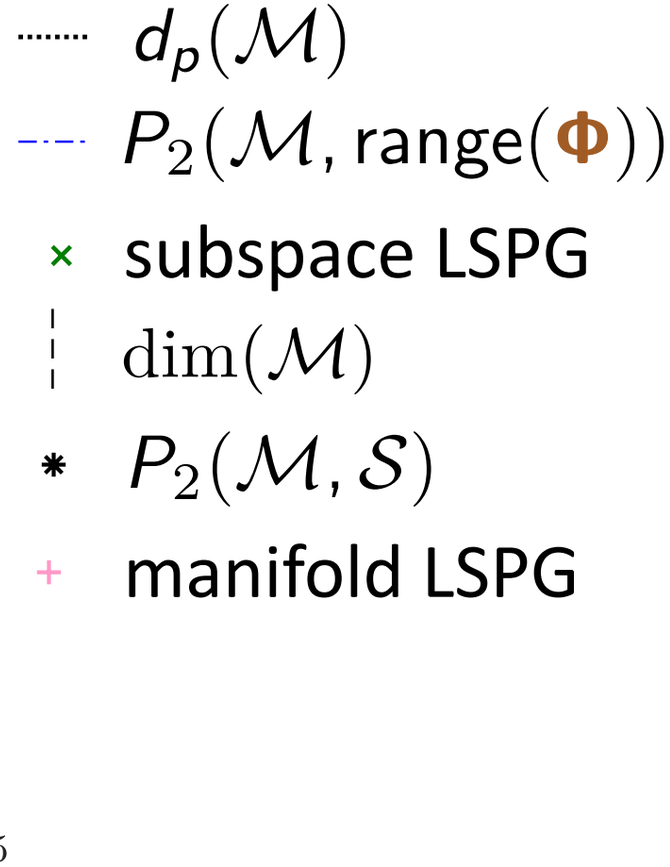
Method improves generalization performance

Burgers' equation



Reacting flow

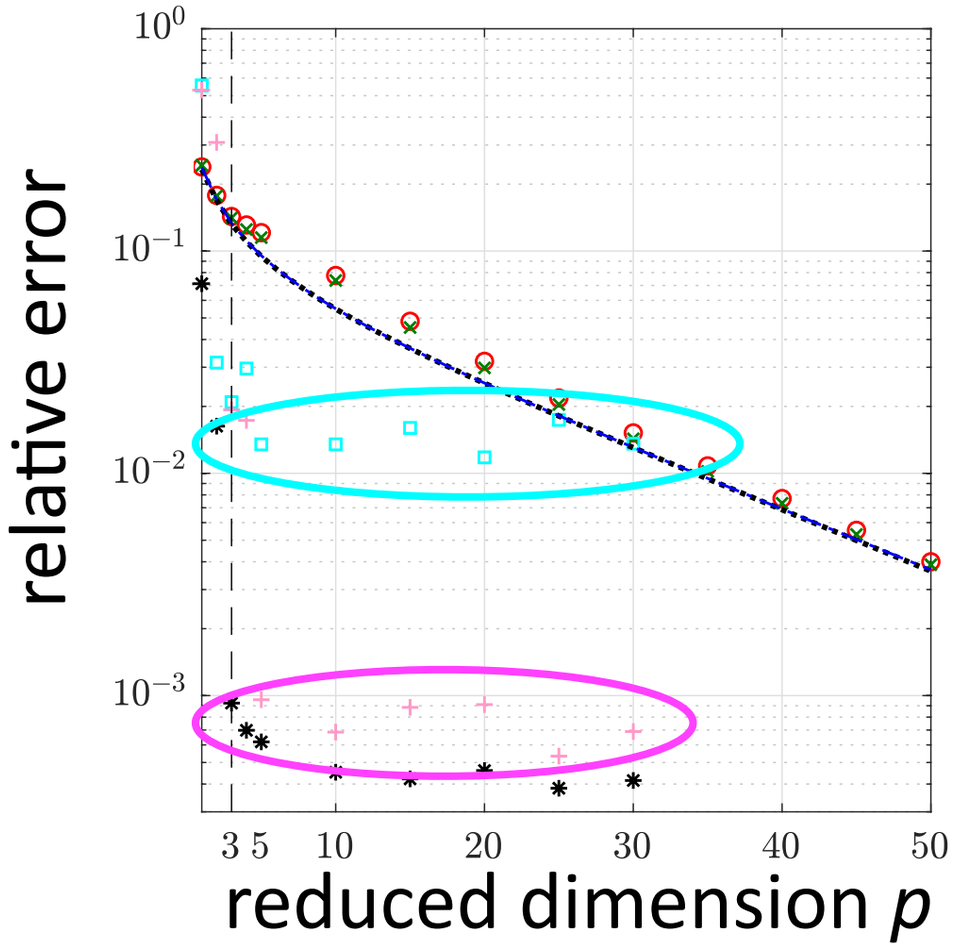




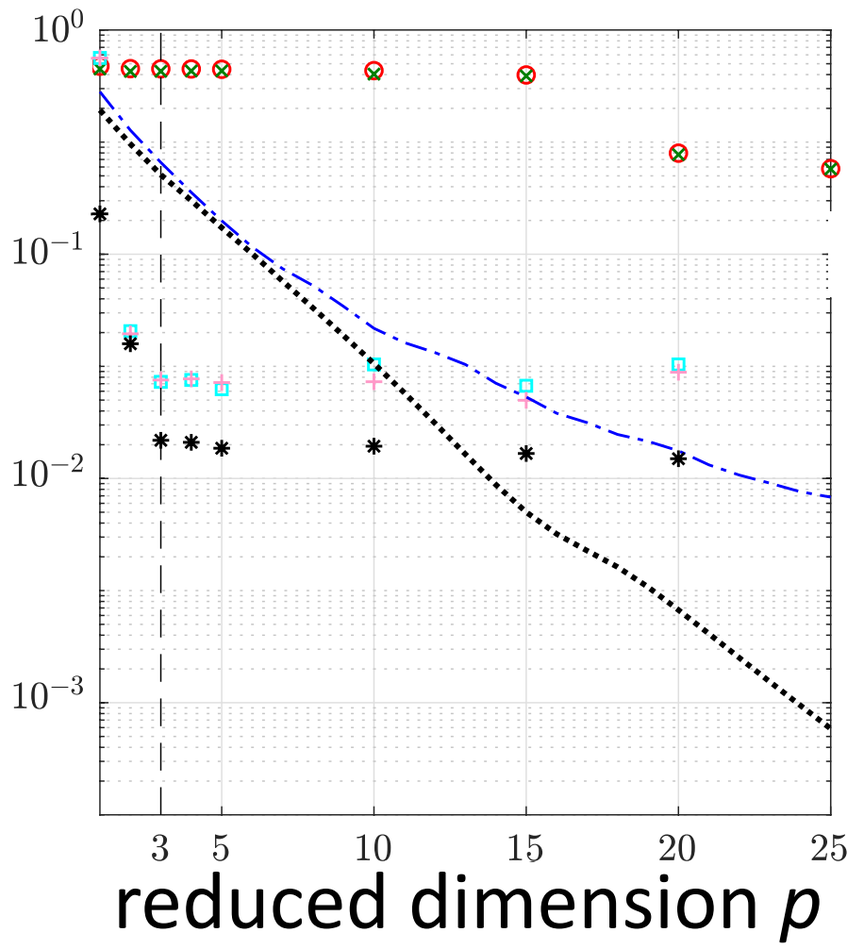
+ Autoencoder manifold significantly better than optimal linear subspace
+ Manifold LSPG orders-of-magnitude more accurate than subspace LSPG
+ Method breaks Kolmogorov-width barrier

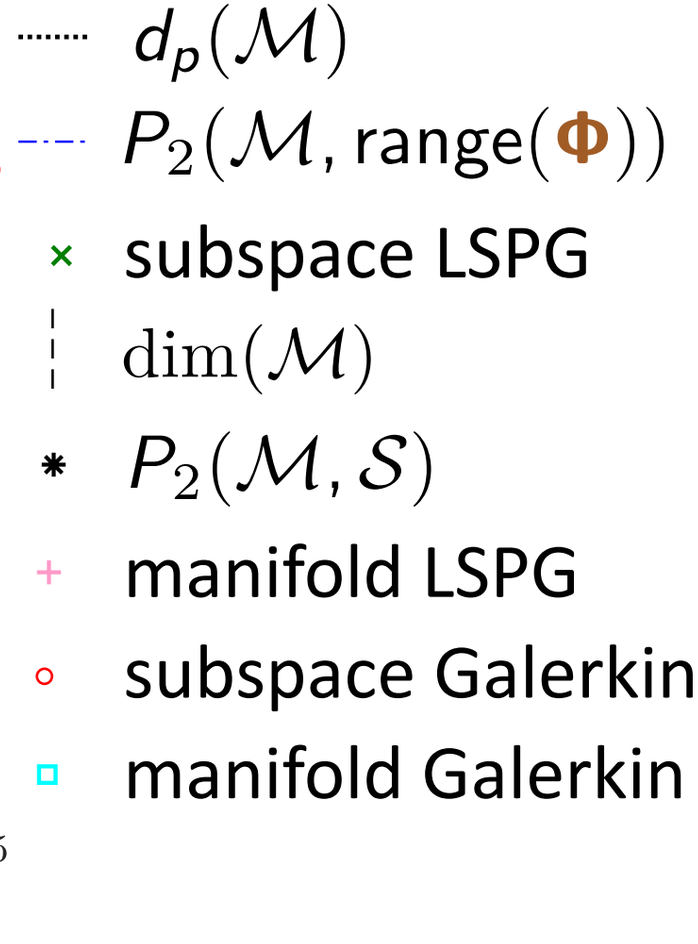
Method improves generalization performance

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
+ Manifold LSPG outperforms manifold Galerkin on 1D Burgers' equation

Outlook

Manifold Galerkin

$$\frac{d\hat{\mathbf{x}}}{dt} = \underset{\hat{\mathbf{v}}\in\mathbb{R}^n}{\operatorname{argmin}} \left\| \mathbf{r}\left(\nabla \mathbf{g}(\hat{\mathbf{x}})\hat{\mathbf{v}}, \mathbf{g}(\hat{\mathbf{x}}); t\right) \right\|_2$$

Manifold LSPG

$$\hat{\mathbf{x}}^n = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\operatorname{argmin}} \left\| \mathbf{r}^n\left(\mathbf{g}(\hat{\mathbf{v}})\right) \right\|_2$$

Interpretation

- Predictions directly integrate deep learning with computational physics

Outlook

Manifold Galerkin

$$\frac{d\hat{\mathbf{x}}}{dt} = \underset{\hat{\mathbf{v}}\in\mathbb{R}^n}{\operatorname{argmin}} \left\| \mathbf{r}(\nabla \mathbf{g}(\hat{\mathbf{x}})\hat{\mathbf{v}}, \mathbf{g}(\hat{\mathbf{x}}); t) \right\|_2$$

Manifold LSPG

$$\hat{\mathbf{x}}^n = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\operatorname{argmin}} \left\| \mathbf{r}^n(\mathbf{g}(\hat{\mathbf{v}})) \right\|_2$$

Interpretation

- Predictions directly integrate deep learning with computational physics
- Latent dynamics model via p*rojection of governing equations*
- Nearly all existing latent dynamics models are purely *data driven*

[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]

Outlook

Manifold Galerkin

$$\frac{d\hat{\mathbf{x}}}{dt} = \underset{\hat{\mathbf{v}}\in\mathbb{R}^n}{\operatorname{argmin}} \left\| \mathbf{r}(\nabla\mathbf{g}(\hat{\mathbf{x}})\hat{\mathbf{v}}, \mathbf{g}(\hat{\mathbf{x}}); t) \right\|_2$$

Manifold LSPG

$$\hat{\mathbf{x}}^n = \underset{\hat{\mathbf{v}}\in\mathbb{R}^p}{\operatorname{argmin}} \left\| \mathbf{r}^n(\mathbf{g}(\hat{\mathbf{v}})) \right\|_2$$

Interpretation

- Predictions directly integrate deep learning with computational physics
- Latent dynamics model via p*rojection of governing equations*
- Nearly all existing latent dynamics models are purely *data driven*

[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

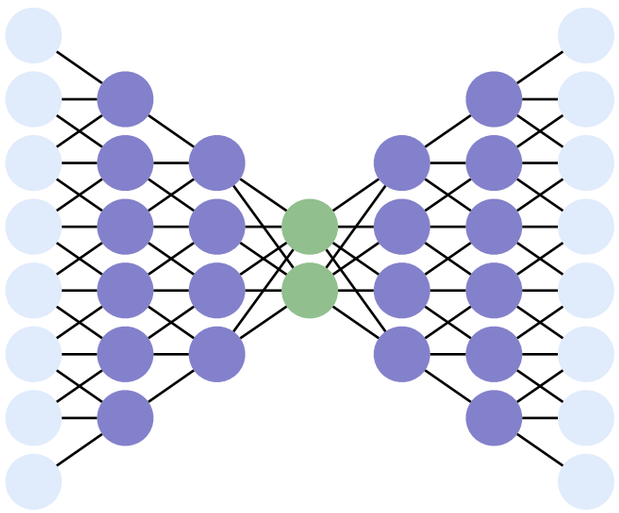
- Backpropagation used to compute decoder Jacobian $\nabla\mathbf{g}(\hat{\mathbf{x}})$
- Quasi-Newton solvers directly call TensorFlow

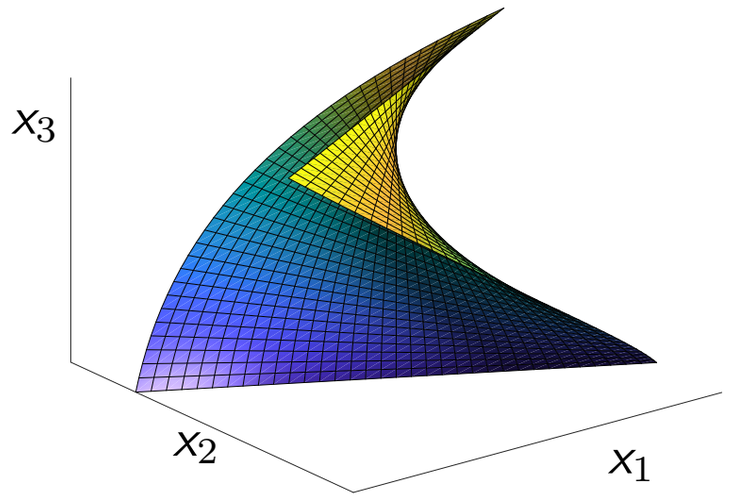
Forward-compatible extensions

- *Hyper-reduction*: convolutional layers preserve sparsity
- *Structure preservation*: equality constraints enforcing conservation

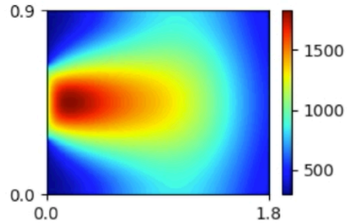
Questions?

Reference: *Lee and C. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," arXiv e-Print, 1812.08373 (2018).*

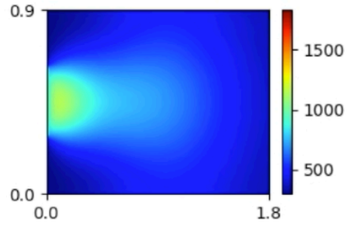




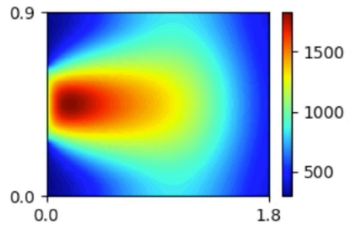
high-fidelity model



POD-LSPG p=5

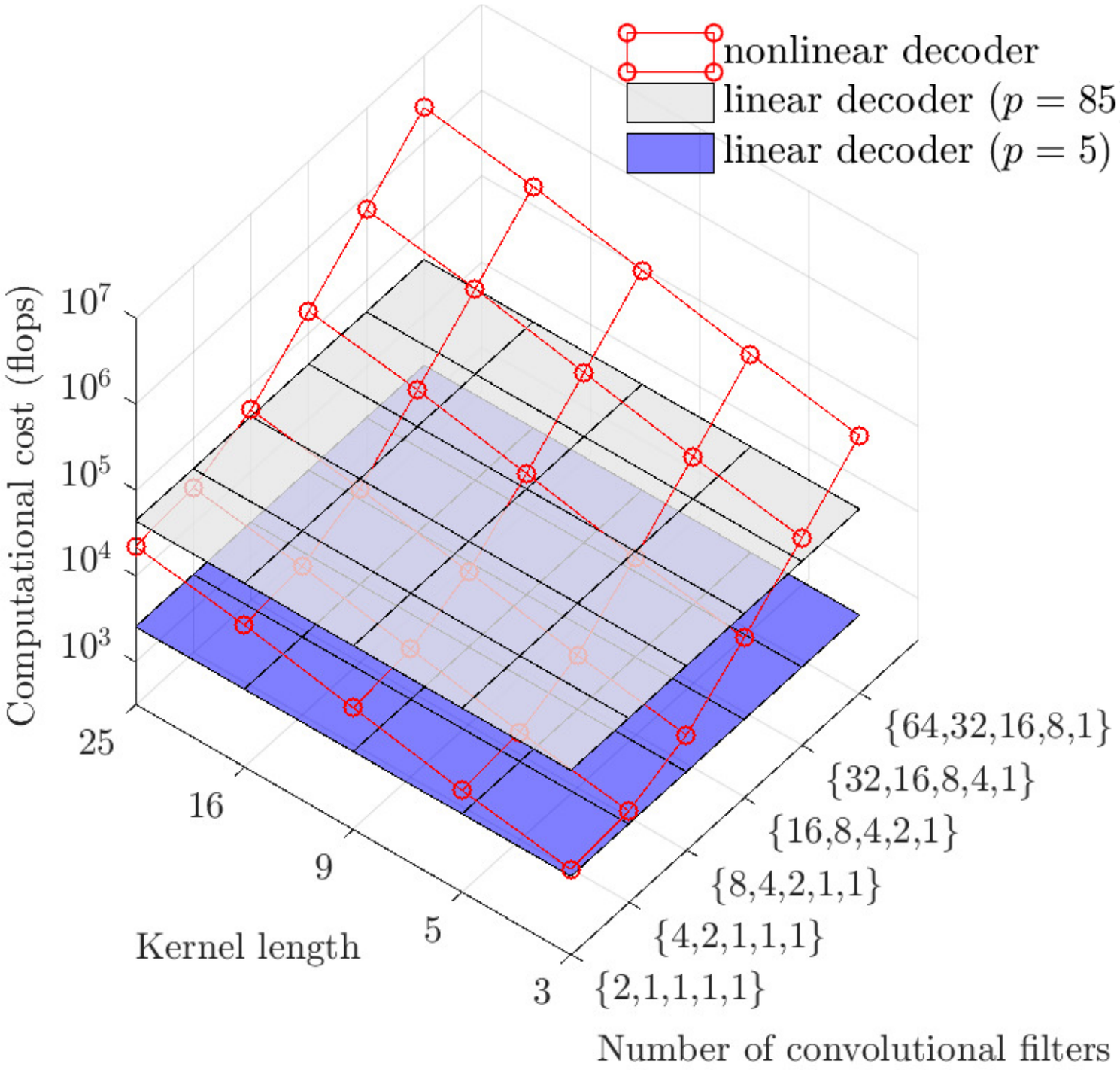


Manifold LSPG p=5



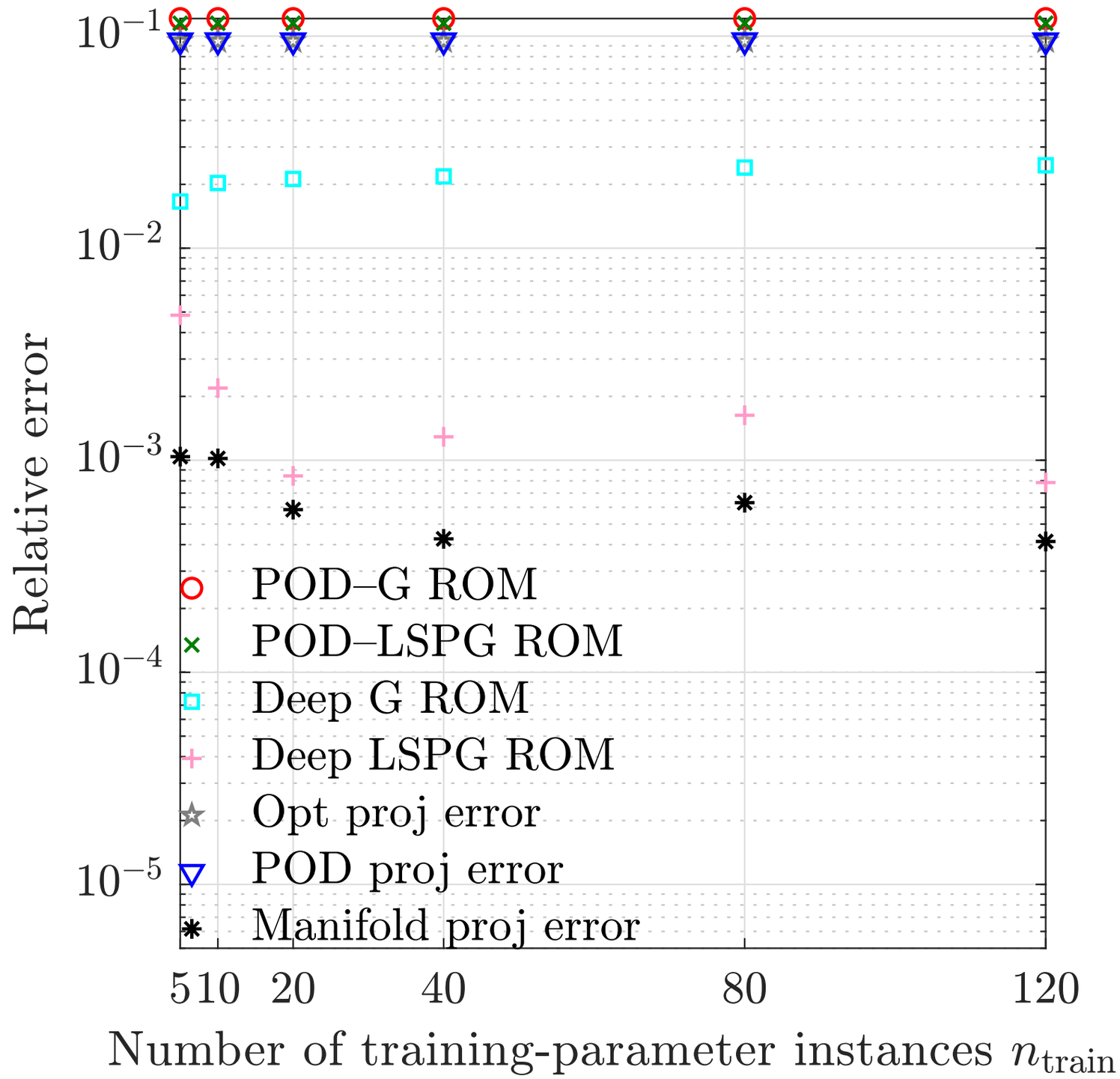
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

Computational cost



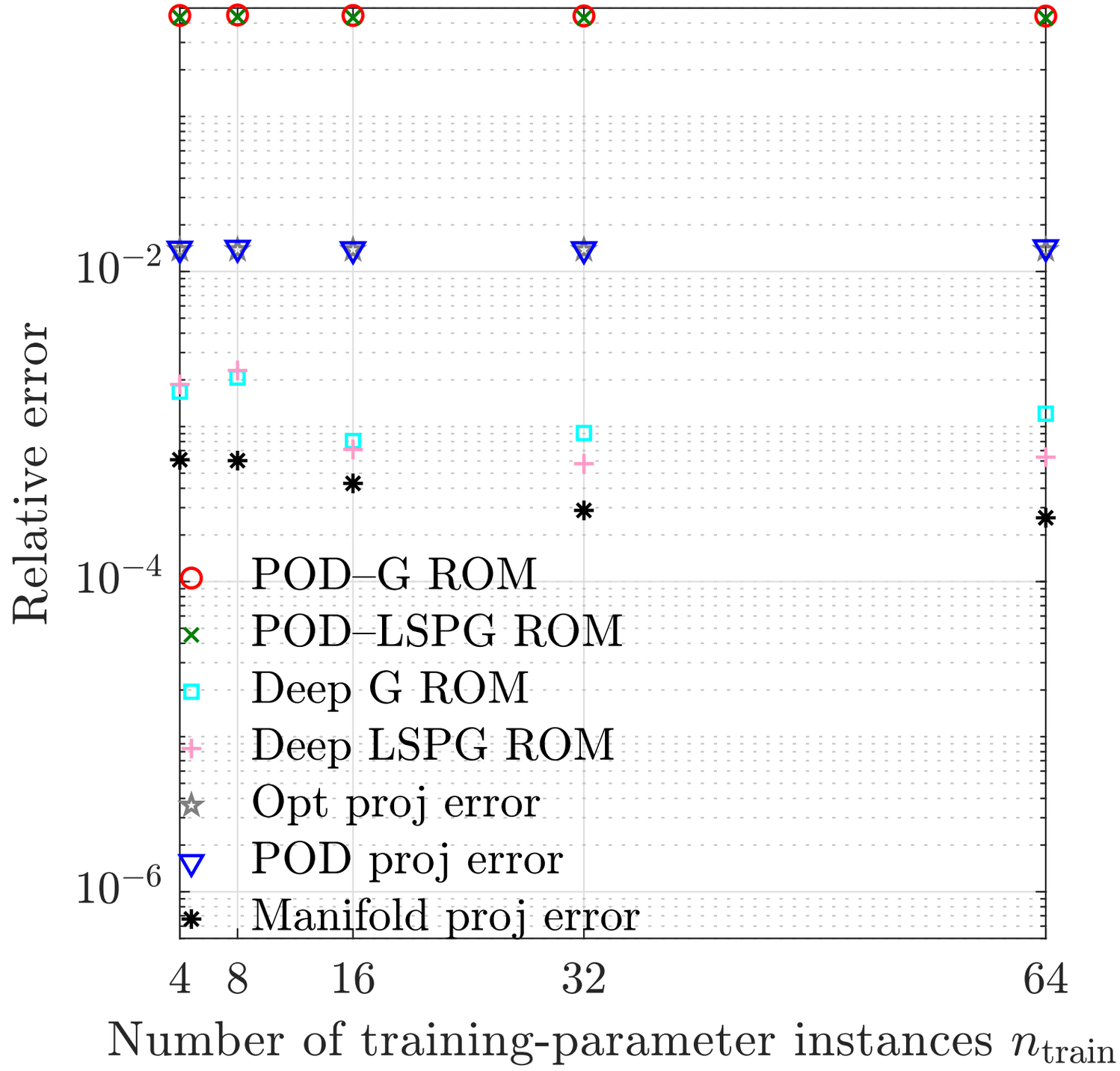
- Architecture dictates computational cost of nonlinear decoder
- \+ Can achieve costs comparable to linear decoders

Dependence on training data: Burgers'



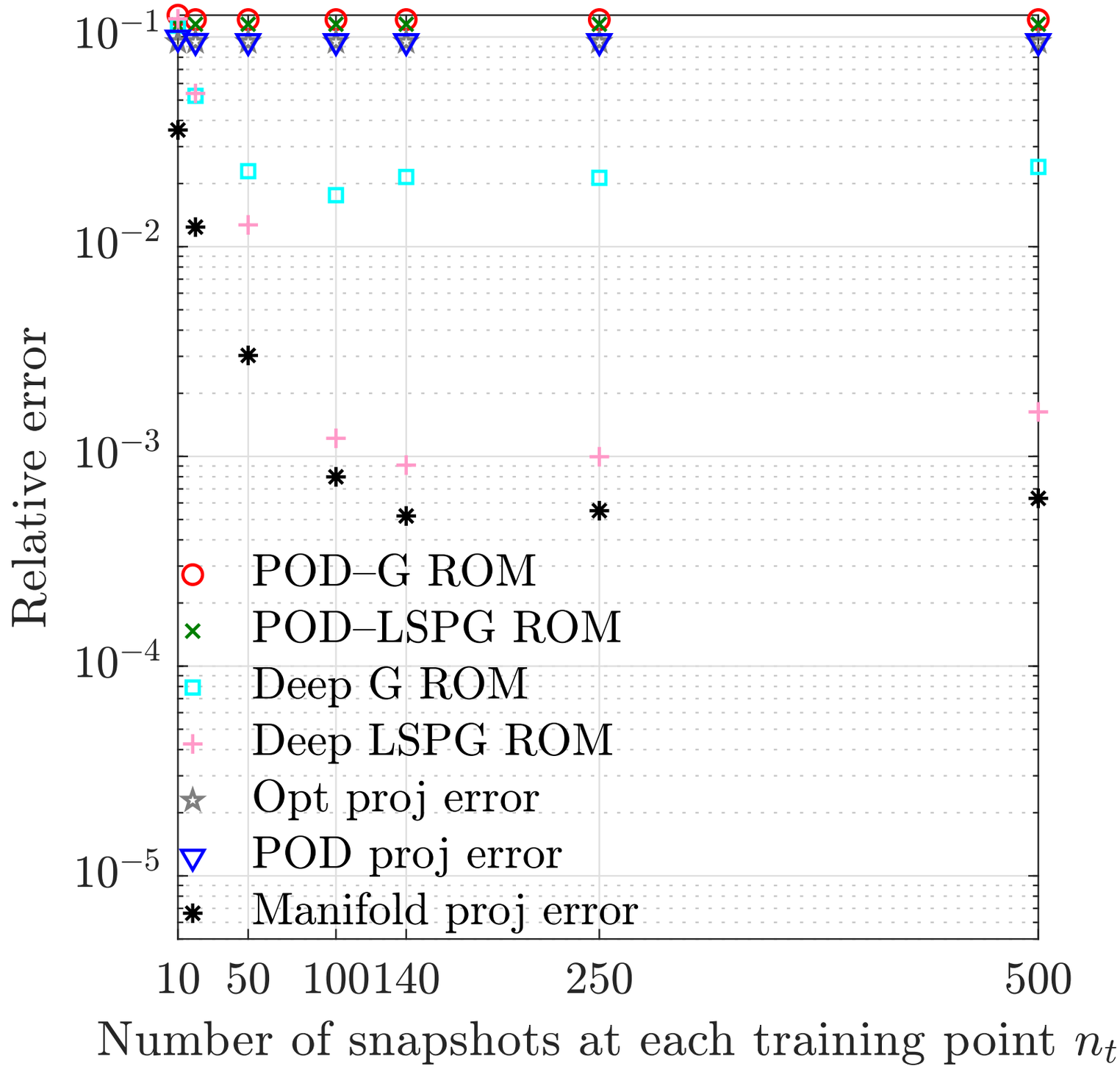
+ Manifold LSPG: <1% errors with only 5 training points

Dependence on training data: Chemically



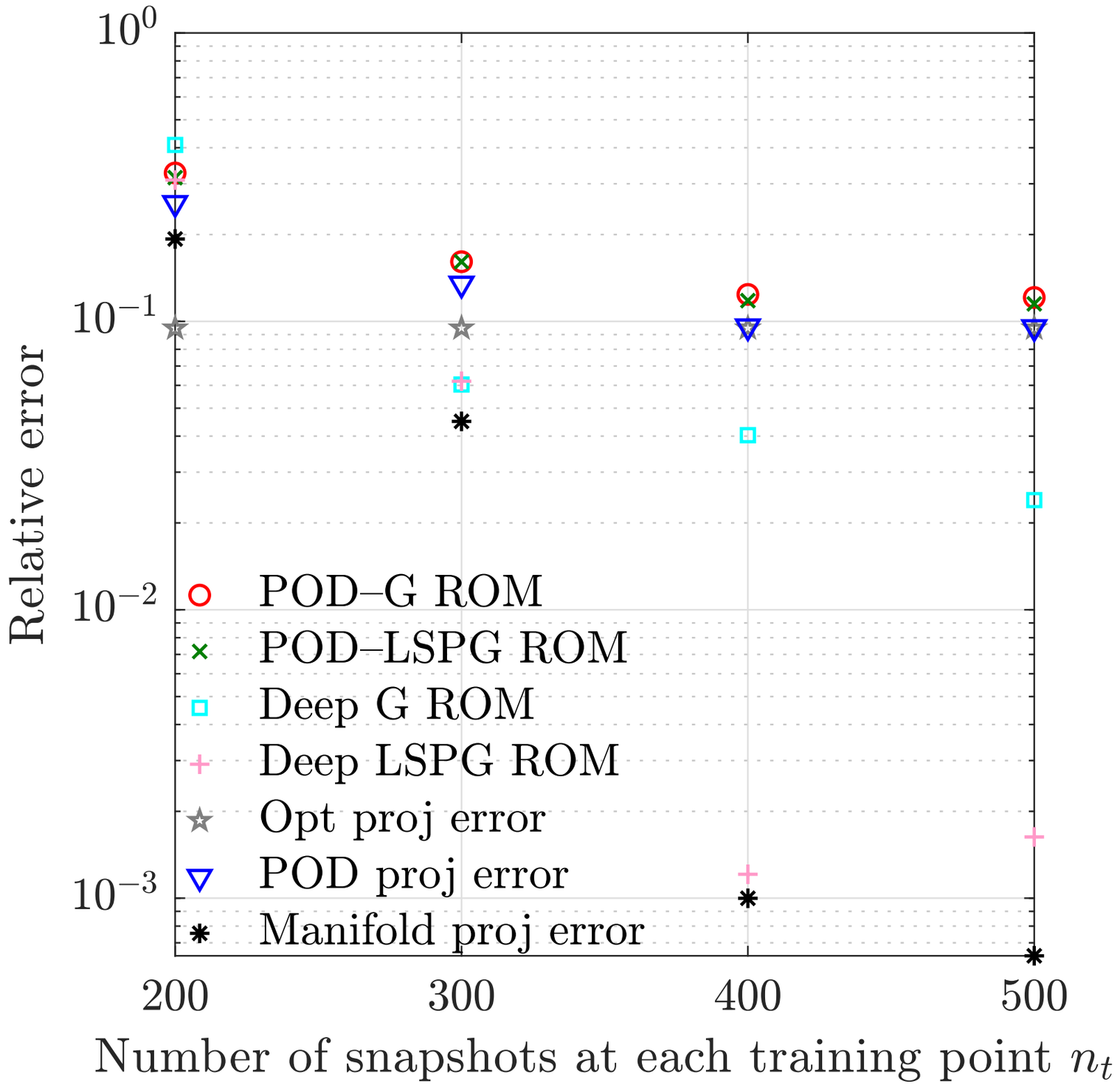
- Manifold LSPG and Galerkin: <1% errors with only 4 training points

Time interpolation: Burgers' equation



+ Manifold LSPG can interpolate well with at least 20% snapshots

Time extrapolation: Burgers' equation



+ Manifold LSPG can extrapolate well with at least 80% snapshots