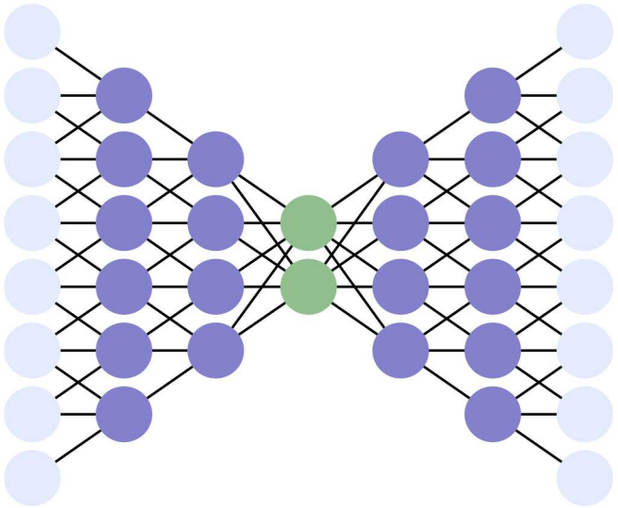
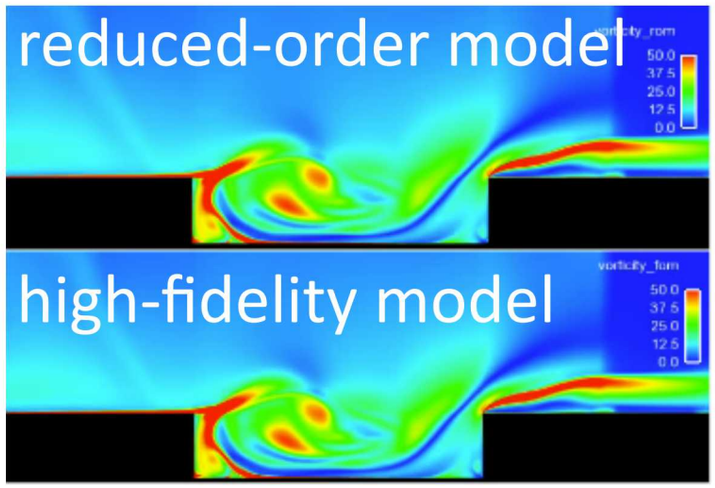
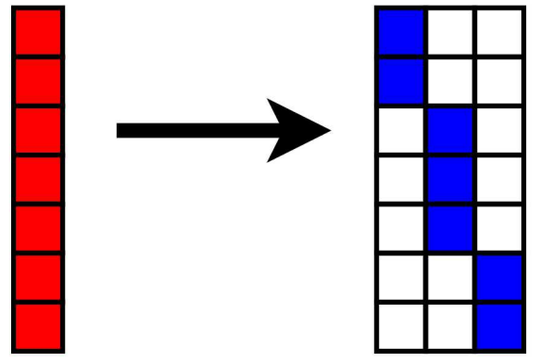
SAND2019-5917PE

Nonlinear model reduction

Using machine learning to enable rapid simulation of extreme-scale physics models







Kevin Carlberg

Sandia National Laboratories

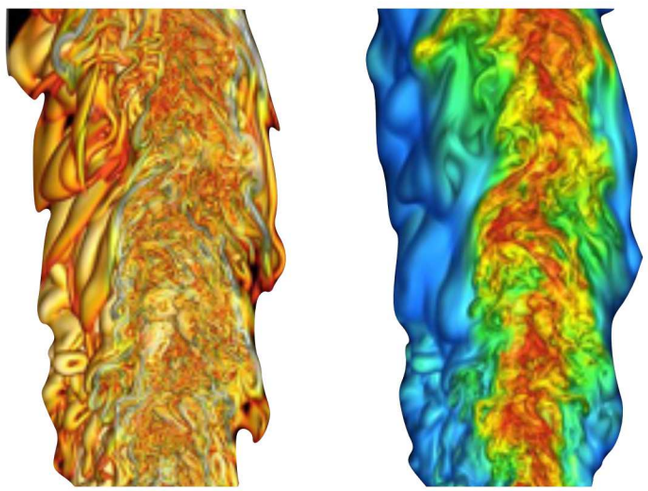
V&V review

May 23, 2019

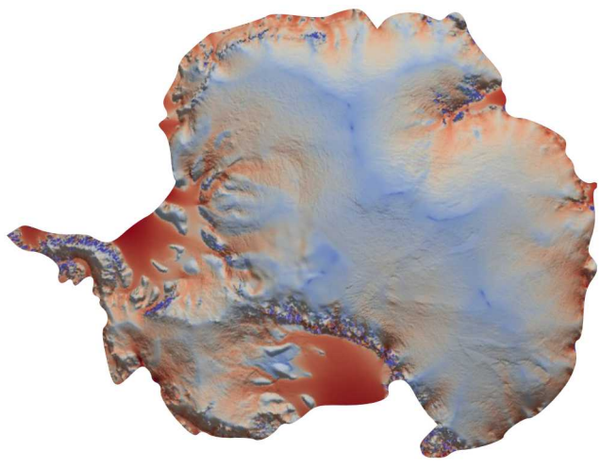
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

High-fidelity simulation

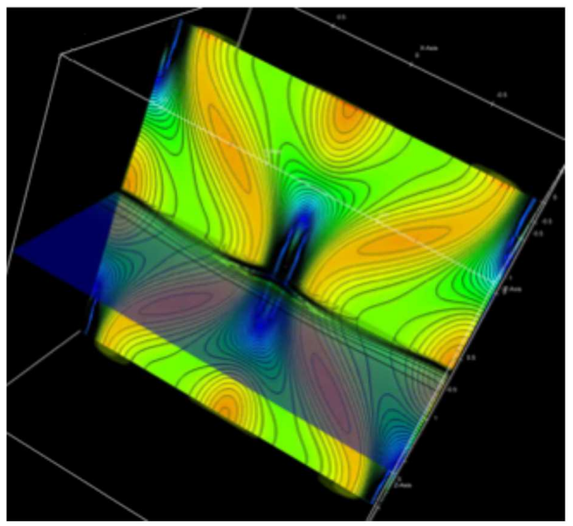
+ Indispensable across science, engineering, and entertainment
- *High fidelity*: extreme-scale computational models



Turbulent reacting flows
courtesy J. Chen, Sandia



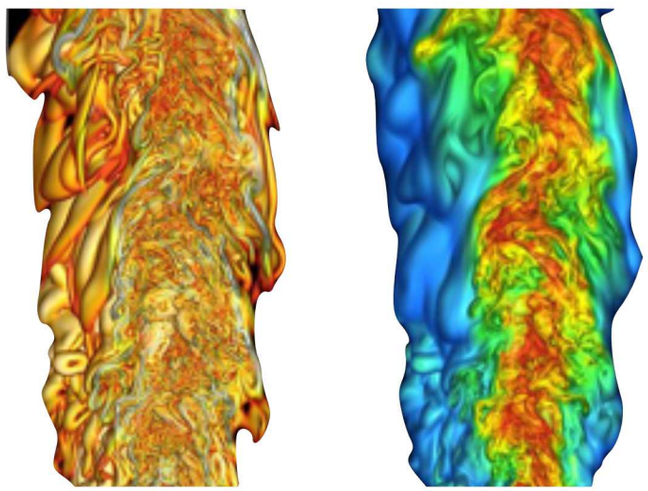
Antarctic ice sheet modeling
courtesy R. Tuminaro, Sandia



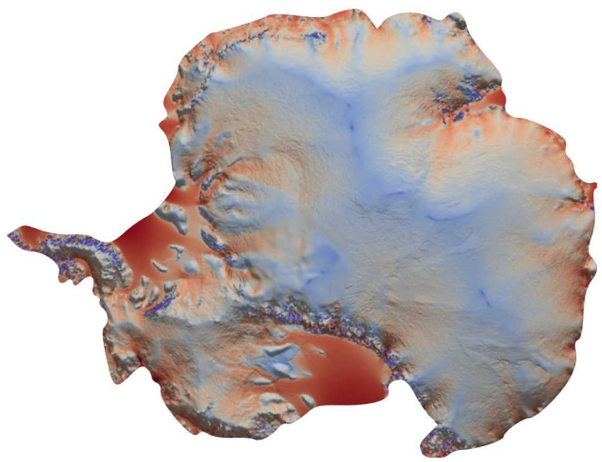
Magnetohydrodynamics
courtesy J. Shadid, Sandia

High-fidelity simulation

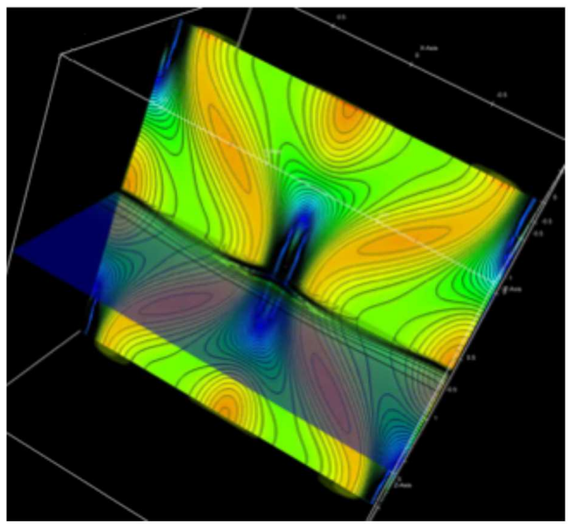
+ Indispensable across science, engineering, and entertainment
- *High fidelity*: extreme-scale computational models



Turbulent reacting flows
courtesy J. Chen, Sandia



Antarctic ice sheet modeling
courtesy R. Tuminaro, Sandia



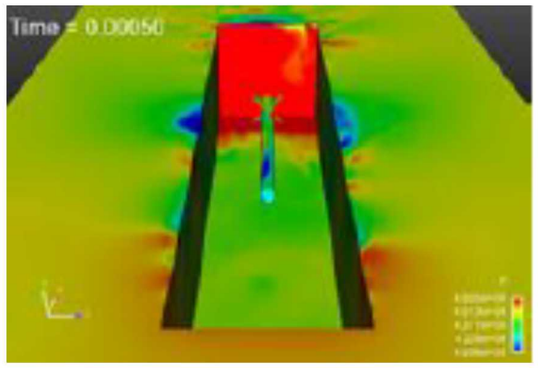
Magnetohydrodynamics
courtesy J. Shadid, Sandia

computational barrier

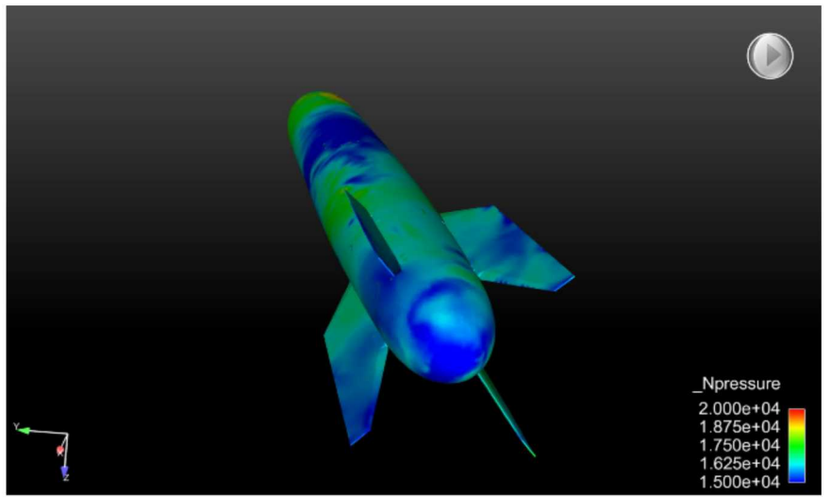
Time-critical problems

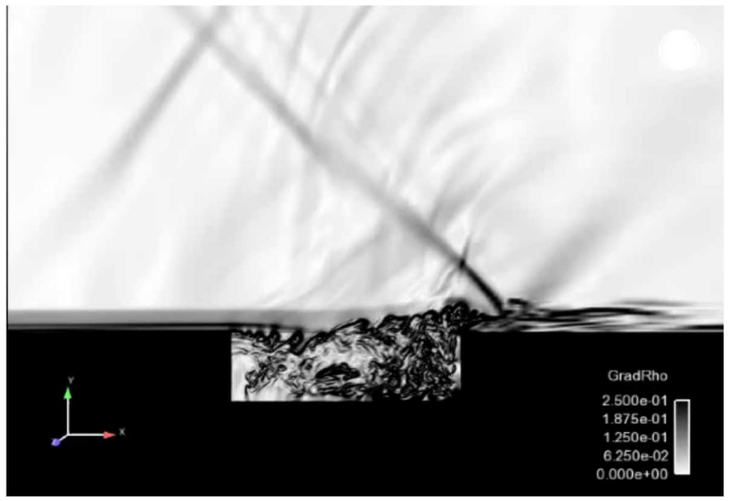
- model predictive control
- health monitoring
- interactive virtual environment
- design optimization

High-fidelity simulation: captive carry



High-fidelity simulation: captive carry

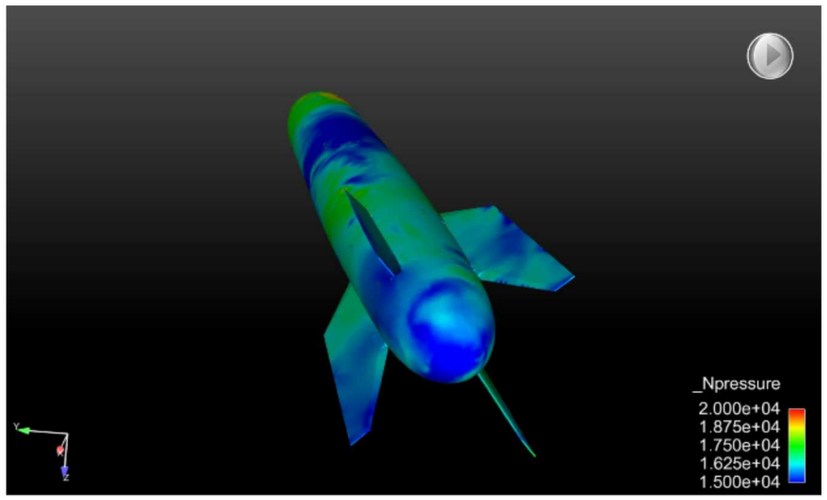


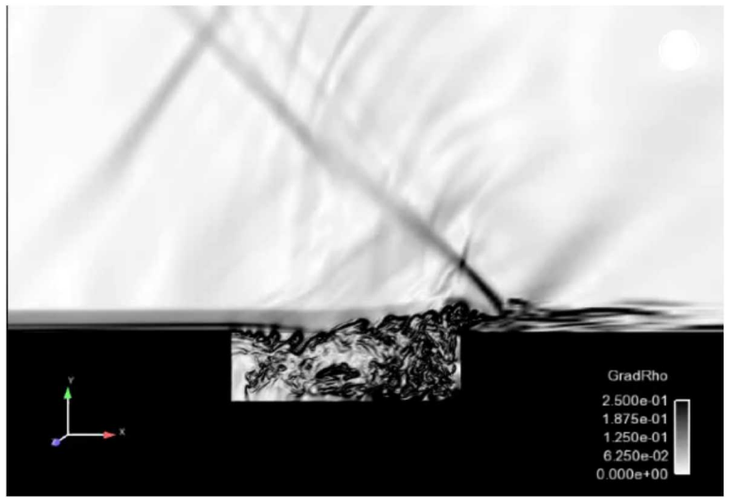


+ *Validated and predictive*: matches wind-tunnel experiments to within 5%

- *Extreme-scale*: 100 million cells, 200,000 time steps
- *High simulation costs*: 6 weeks, 5000 cores

High-fidelity simulation: captive carry





+ *Validated and predictive*: 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

Time-critical problems

- explore flight envelope
- uncertainty quantification
- model predictive control
- robust design of store and cavity

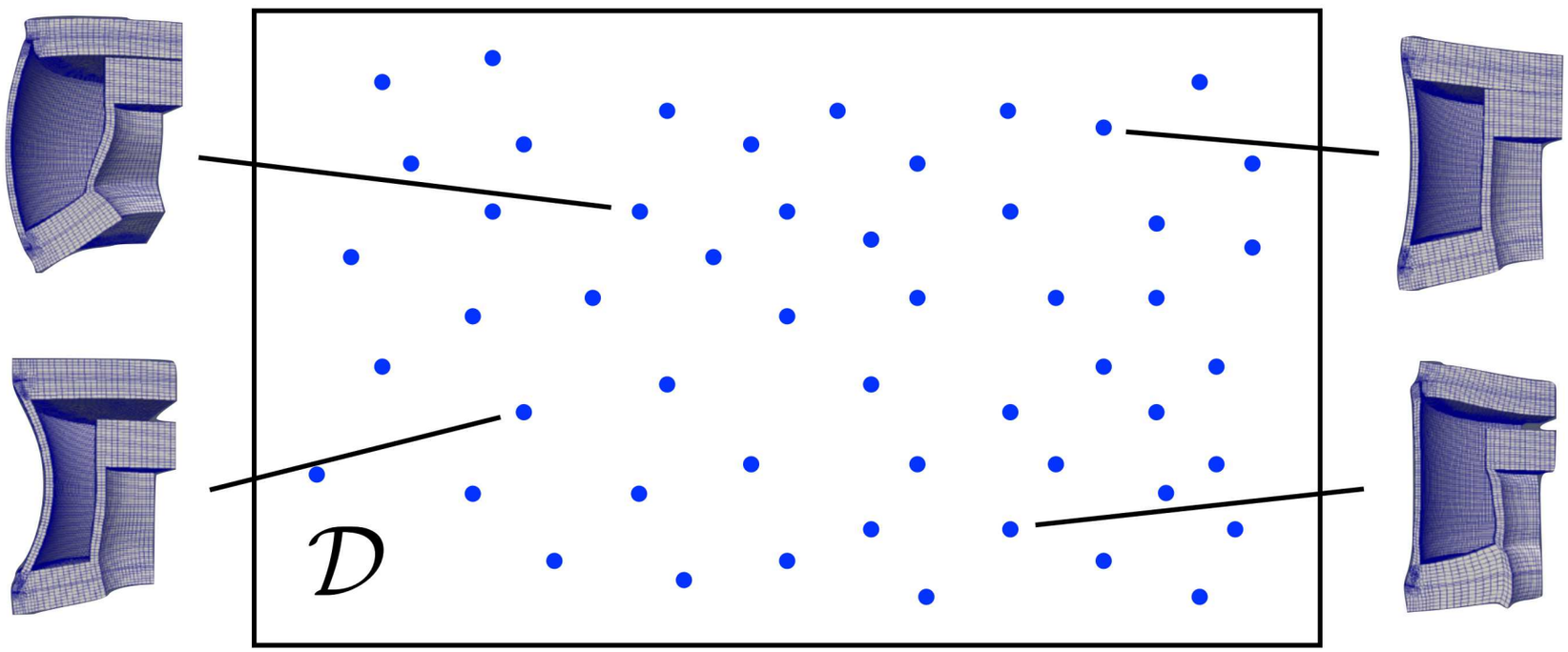
Approach: exploit simulation data

$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$$

Approach: exploit simulation data

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$

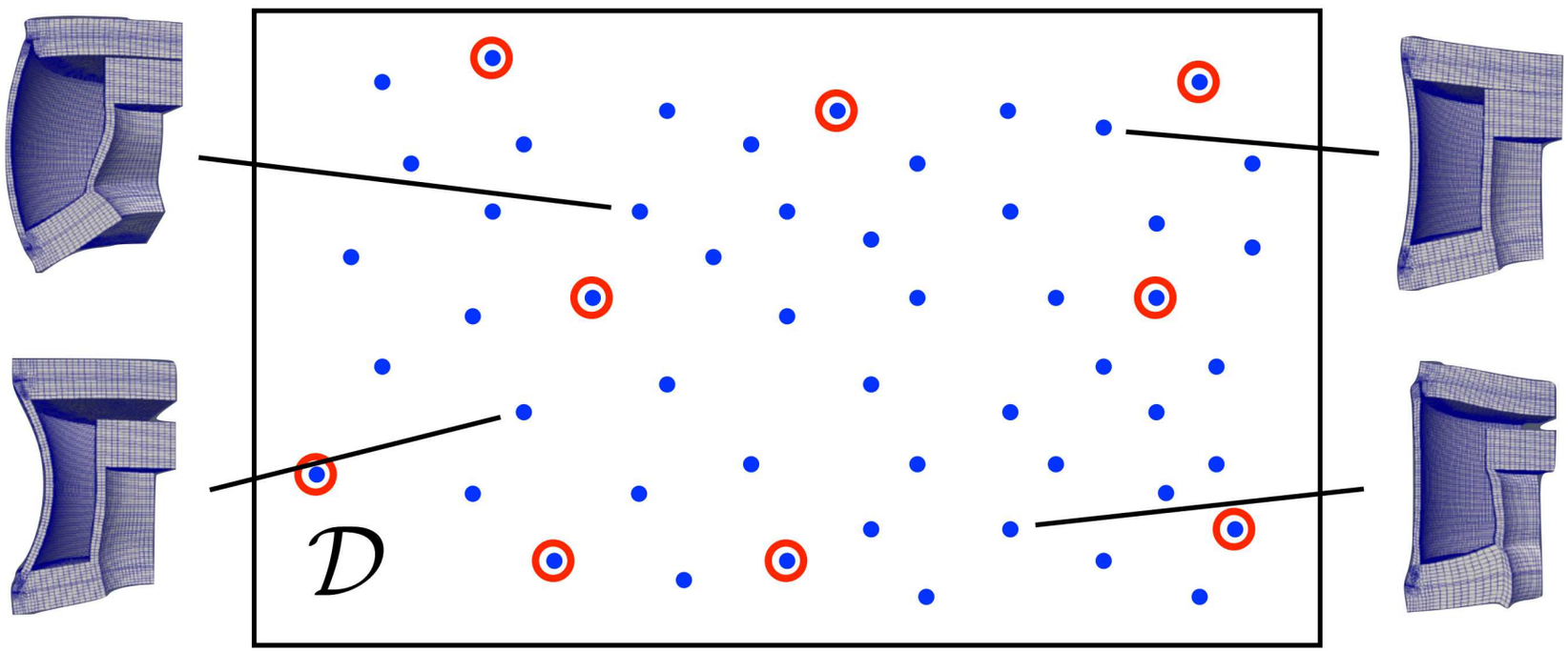
***Time-critical problem**: rapidly solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*



Approach: exploit simulation data

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$

***Time-critical problem**: rapidly solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*

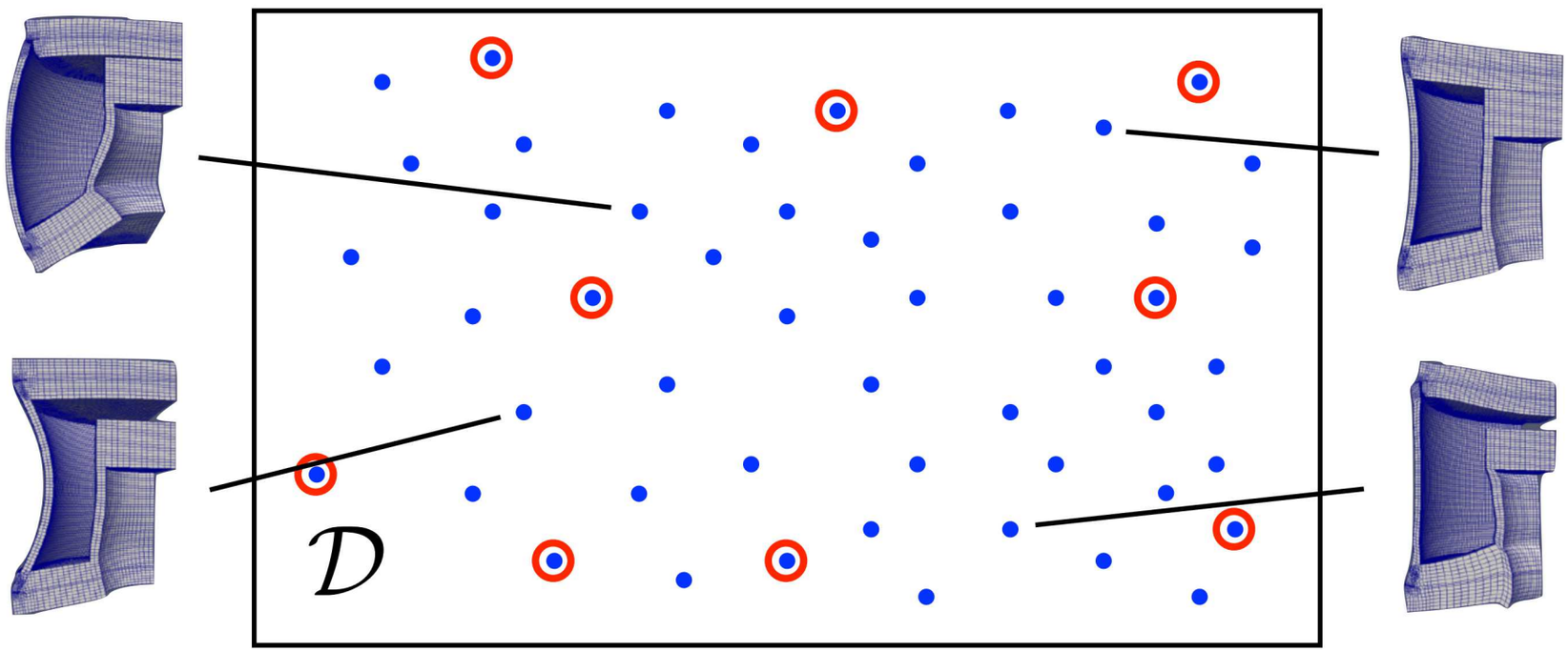


***Idea**: exploit simulation data collected at a few points*

Approach: exploit simulation data

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$

***Time-critical problem**: rapidly solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*



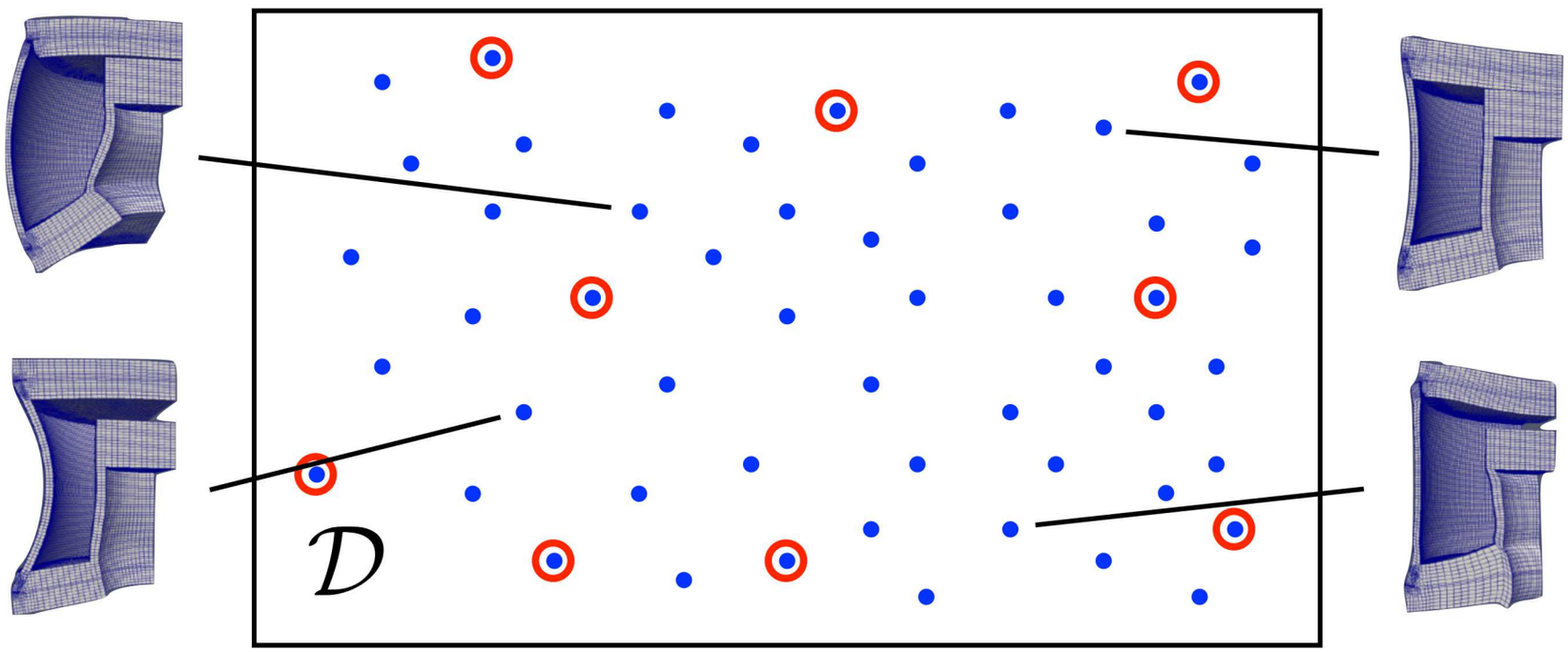
***Idea**: exploit simulation data collected at a few points*

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data

Approach: exploit simulation data

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$

***Time-critical problem**: rapidly solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*



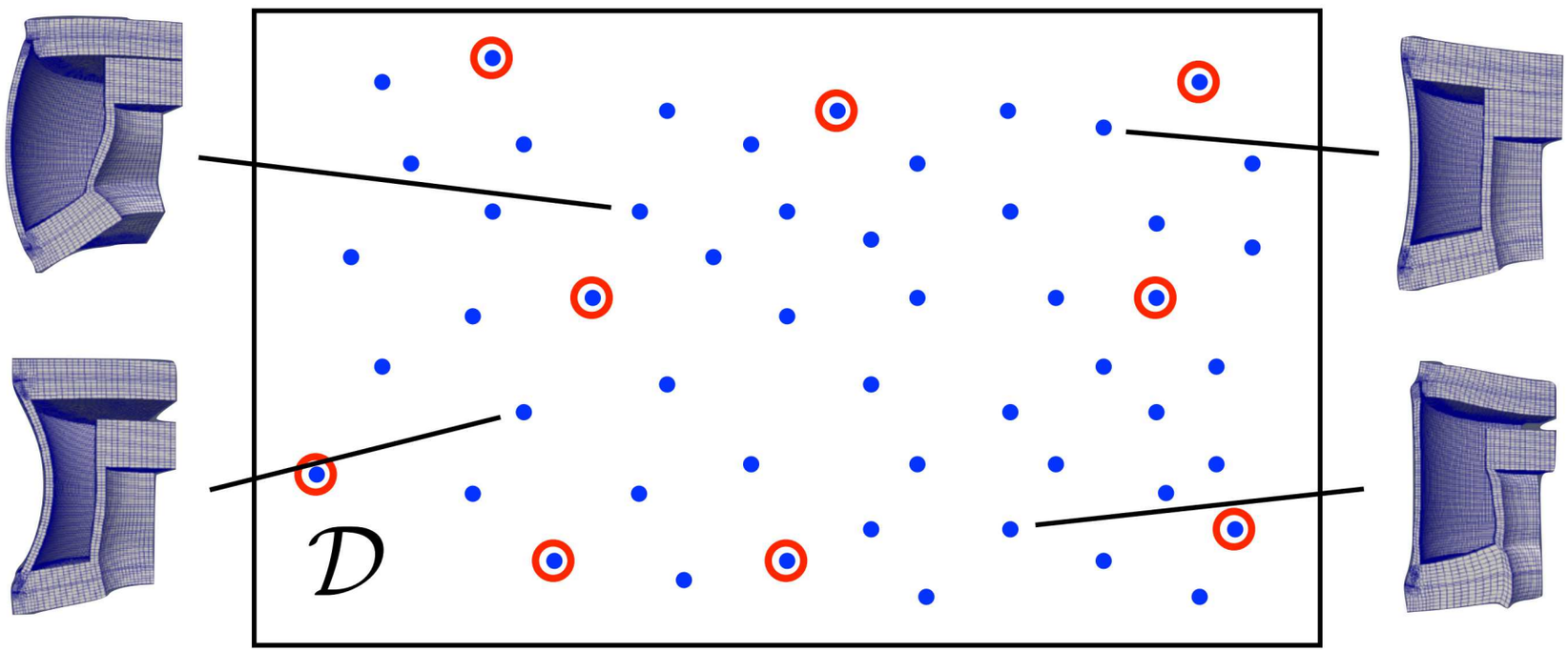
***Idea**: exploit simulation data collected at a few points*

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data

Approach: exploit simulation data

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu}), \quad \mathbf{x}(0, \boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \quad t \in [0, T_{\text{final}}], \quad \boldsymbol{\mu} \in \mathcal{D}$

***Time-critical problem**: rapidly solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*



***Idea**: exploit simulation data collected at a few points*

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 cost of ODE solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Model reduction criteria

Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error

Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error
2. ***Low cost:*** achieves at least 100x computational savings

Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error
2. ***Low cost:*** achieves at least 100x computational savings
3. ***Structure preservation:*** preserves important physical properties

Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error
2. ***Low cost:*** achieves at least 100x computational savings
3. ***Structure preservation:*** preserves important physical properties
4. ***Generalization:*** should work even in difficult cases

Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error
2. ***Low cost:*** achieves at least 100x computational savings
3. ***Structure preservation:*** preserves important physical properties
4. ***Generalization:*** should work even in difficult cases
5. ***Certification:*** accurately quantify the ROM error

Model reduction: existing approaches

Model reduction: existing approaches

Linear time-invariant systems: mature [Antoulas, 2005]

- Balanced truncation [Moore, 1981; Willcox and Peraire, 2002; Rowley, 2005]
- Transfer-function interpolation [Bai, 2002; Freund, 2003; Gallivan et al, 2004; Baur et al., 2001]

+ *Accurate*, *generalizes*, *certified*: sharp *a priori* error bounds
+ *Inexpensive*: pre-assemble operators
+ *Structure preservation*: guaranteed stability

Model reduction: existing approaches

Linear time-invariant systems: mature [Antoulas, 2005]

- Balanced truncation [Moore, 1981; Willcox and Peraire, 2002; Rowley, 2005]
- Transfer-function interpolation [Bai, 2002; Freund, 2003; Gallivan et al, 2004; Baur et al., 2001]

+ *Accurate*, *generalizes*, *certified*: sharp *a priori* error bounds
+ *Inexpensive*: pre-assemble operators
+ *Structure preservation*: guaranteed stability

Elliptic/parabolic PDEs: mature [Prud'Homme et al., 2001; Barrault et al., 2004; Rozza et al., 2008]

- Reduced-basis method

+ *Accurate*, *generalizes*, *certified*: sharp *a priori* error bounds
+ *Inexpensive*: pre-assemble operators
+ *Structure preservation*: preserve operator properties

Model reduction: existing approaches

Linear time-invariant systems: mature [Antoulas, 2005]

- Balanced truncation [Moore, 1981; Willcox and Peraire, 2002; Rowley, 2005]
- Transfer-function interpolation [Bai, 2002; Freund, 2003; Gallivan et al, 2004; Baur et al., 2001]

\+ *Accurate*, *generalizes*, *certified*: sharp *a priori* error bounds

\+ *Inexpensive*: pre-assemble operators

\+ *Structure preservation*: guaranteed stability

Elliptic/parabolic PDEs: mature [Prud'Homme et al., 2001; Barrault et al., 2004; Rozza et al., 2008]

- Reduced-basis method

\+ *Accurate*, *generalizes*, *certified*: sharp *a priori* error bounds

\+ *Inexpensive*: pre-assemble operators

\+ *Structure preservation*: preserve operator properties

Nonlinear dynamical systems: ineffective

- Proper orthogonal decomposition (POD)–Galerkin [Sirovich, 1987; Colonius, 2004]

\- *Inaccurate, doesn't generalize*: often unstable

\- *Not certified*: error bounds grow exponentially in time

\- *Expensive*: projection insufficient for speedup

\- *Structure not preserved*: physical properties ignored

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

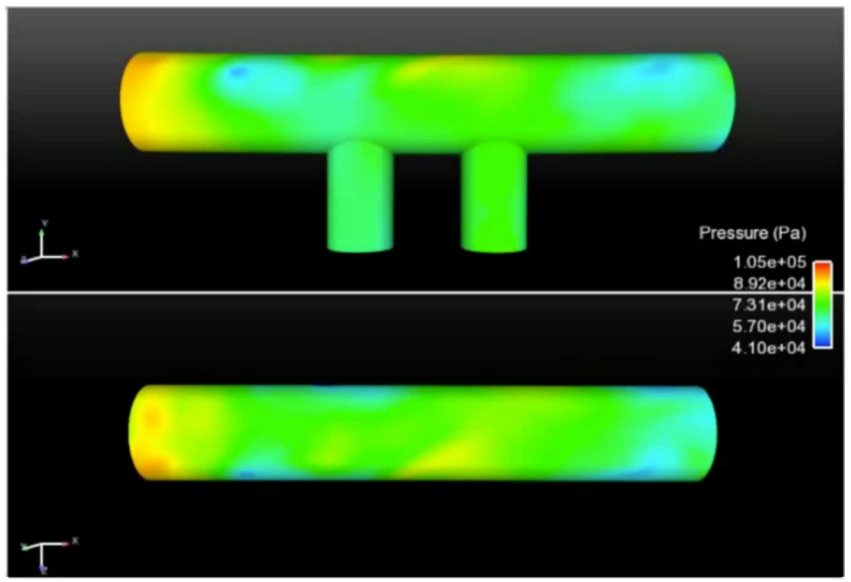
Collaborators: *Matthew Barone (Sandia), Harbir Antil (GMU)*

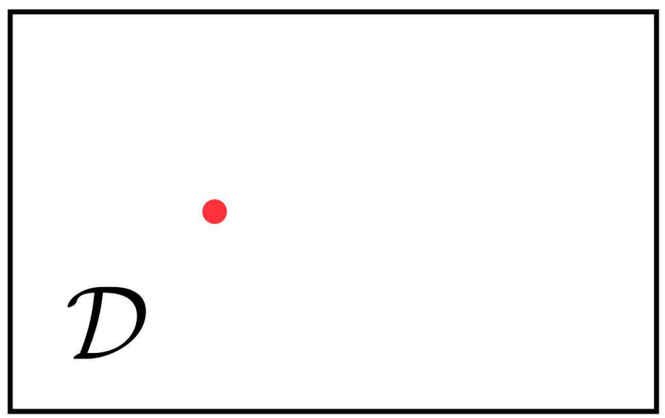
* #1 most-cited paper, Int J Numer Meth Eng, 2011

Training simulations: state tensor

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

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

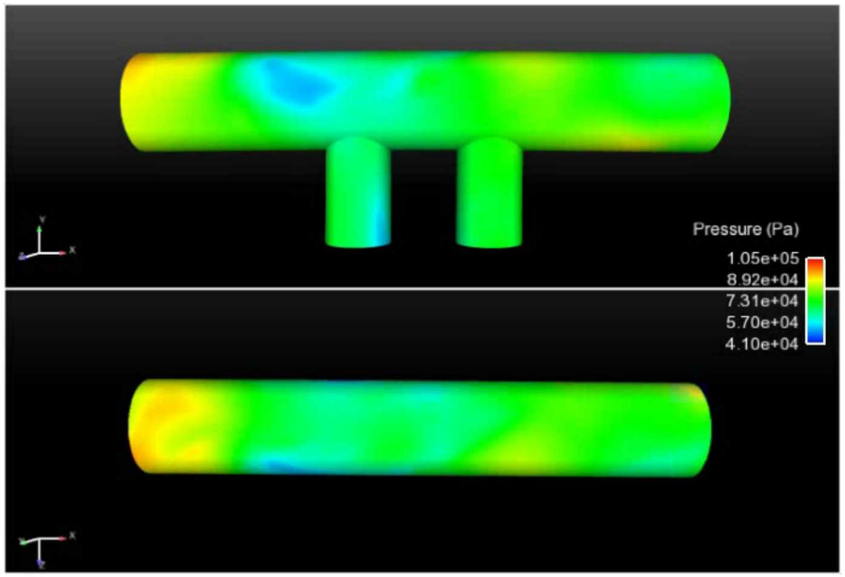


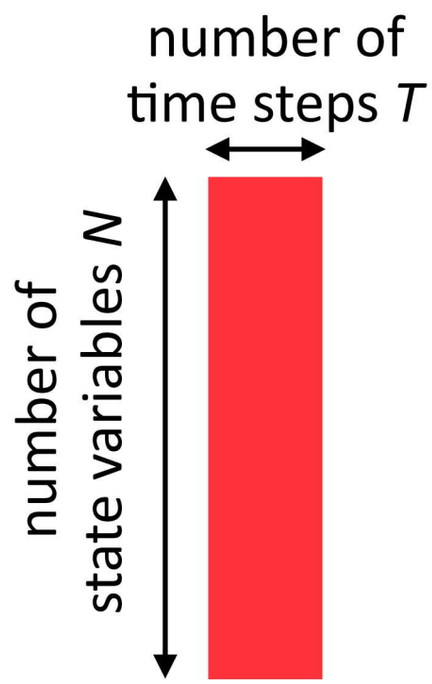


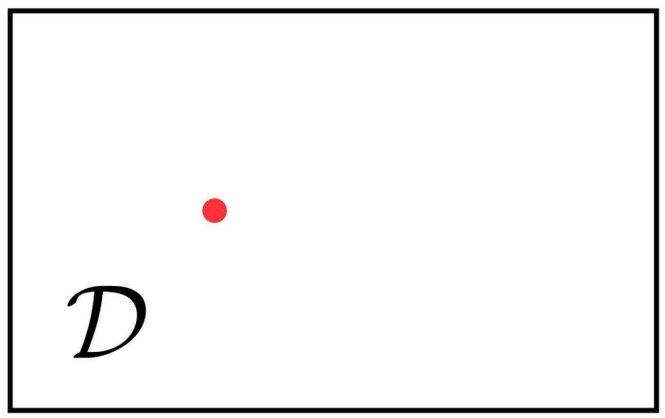
Training simulations: state tensor

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

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



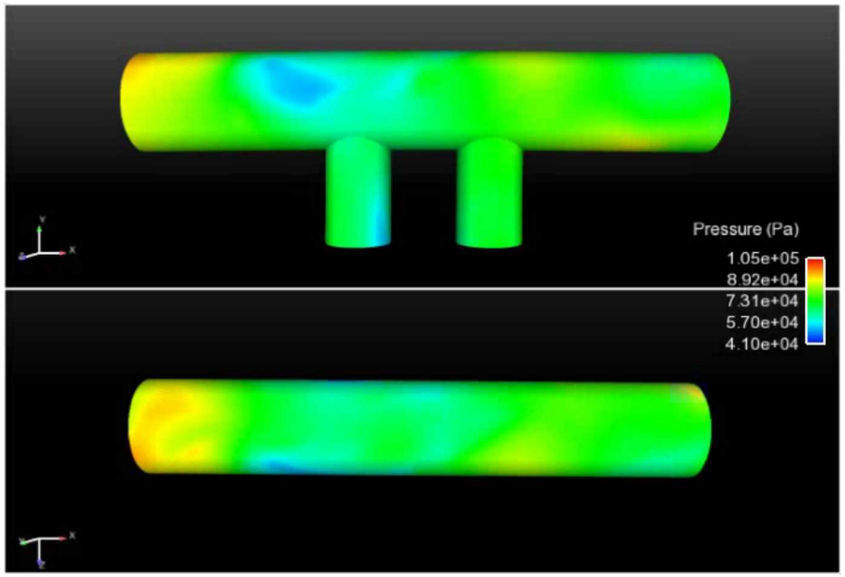




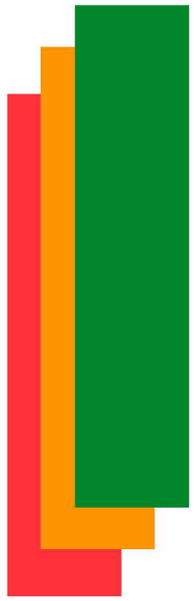
Training simulations: state tensor

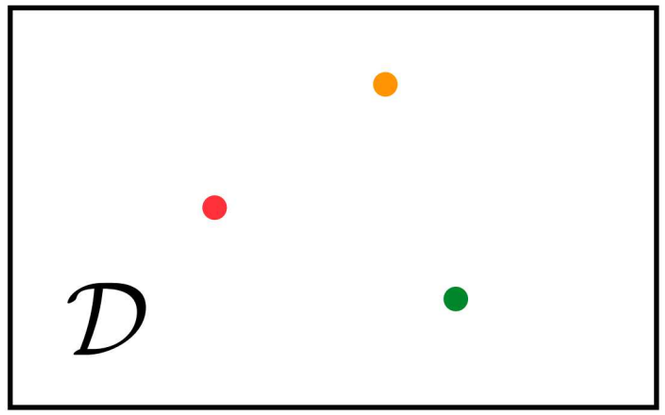
$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

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



$$\mathcal{X} =$$





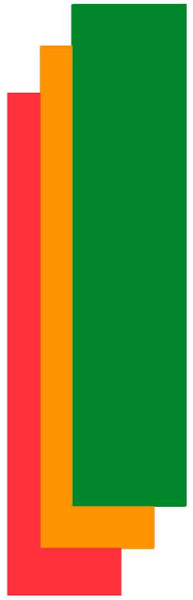
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$

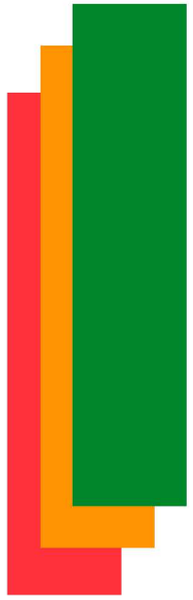


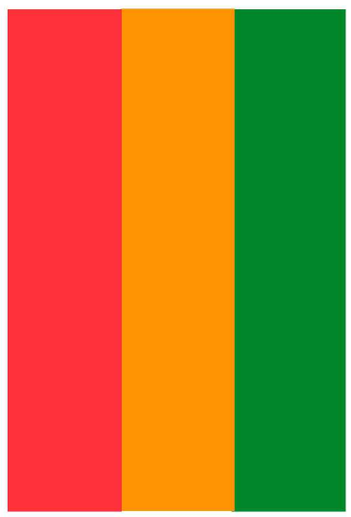
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$ 

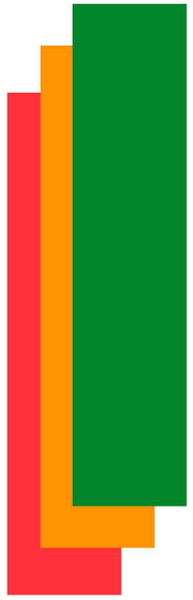
$\mathbf{X}_{(1)} =$ 

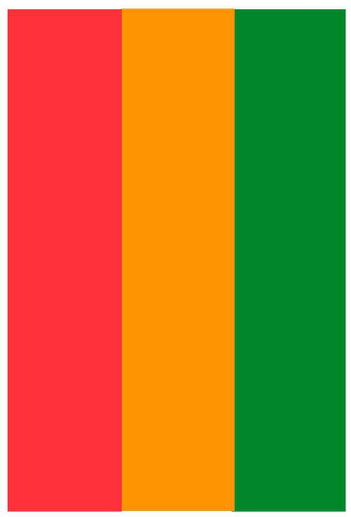
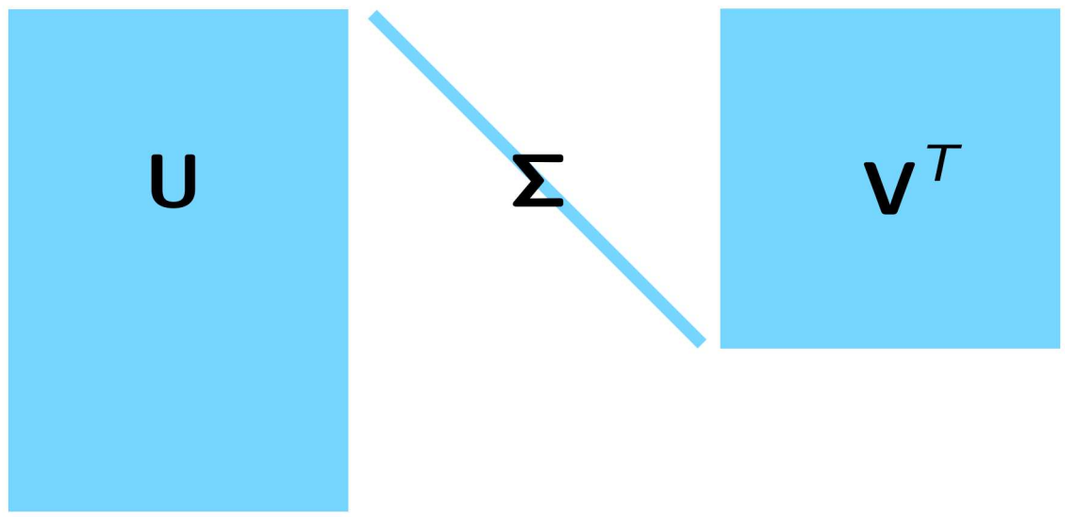
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$ 

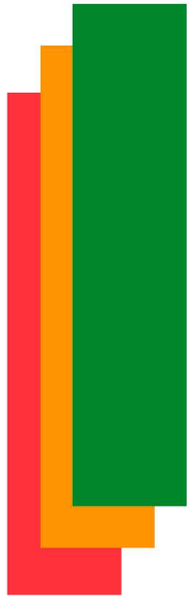
$\mathbf{X}_{(1)} =$  $=$ 

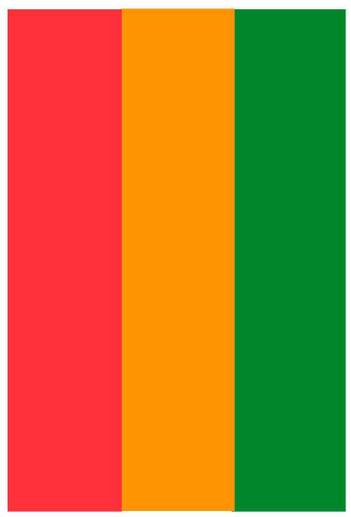
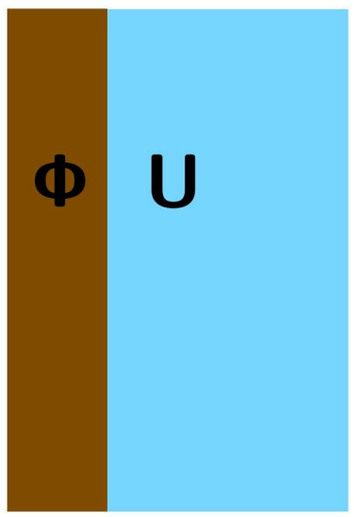
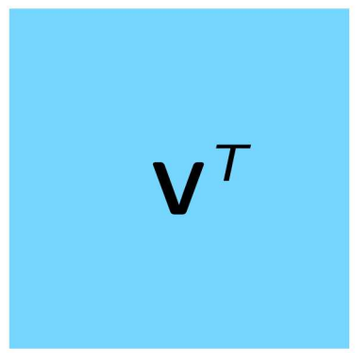
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$ 

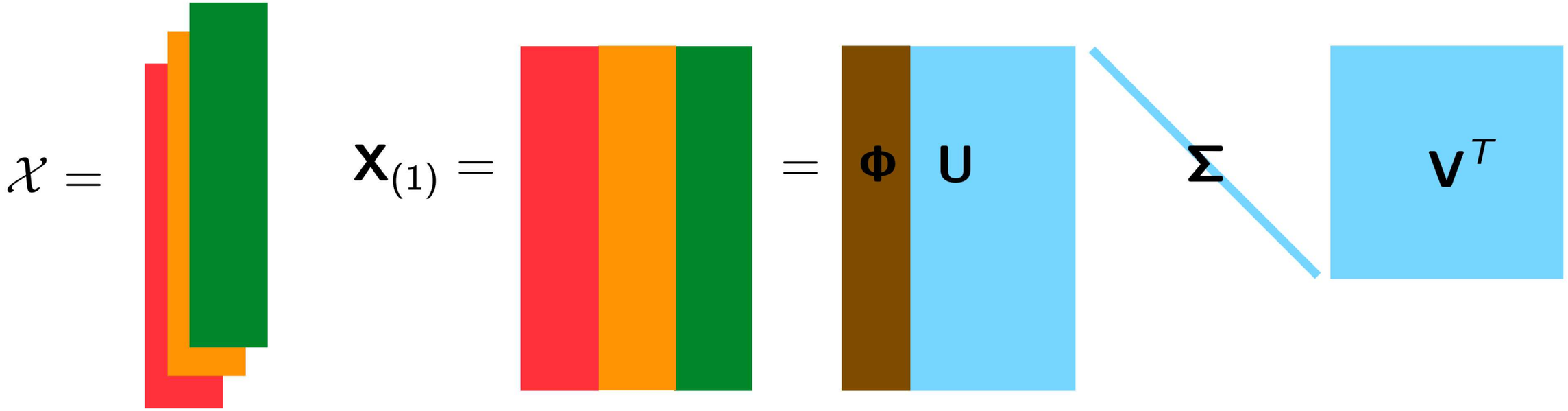
$\mathbf{X}_{(1)} =$  $=$  $\mathbf{\Phi}$ $\mathbf{U}$ $\mathbf{\Sigma}$  $\mathbf{V}^T$

Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding



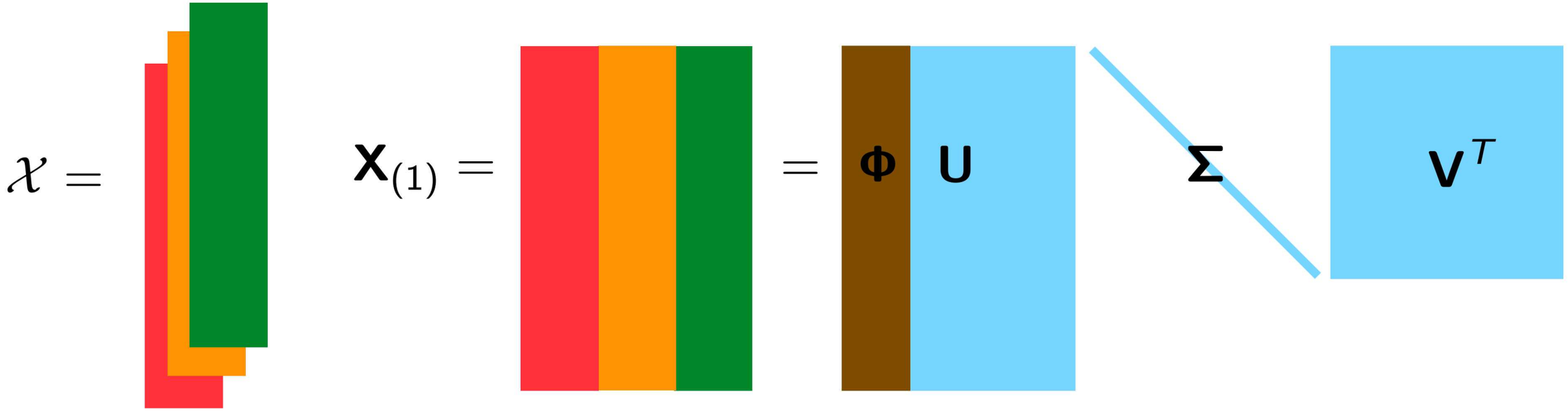
Φ *columns are principal components of the spatial simulation data*

Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

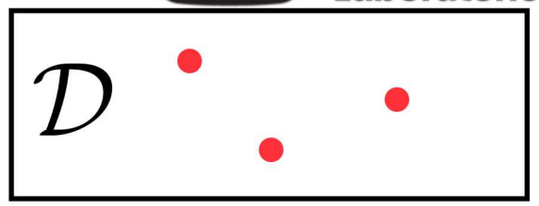


Φ *columns are principal components of the spatial simulation data*

How to integrate these data with the computational model?

Least-squares Petrov–Galerkin (LSPG)

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

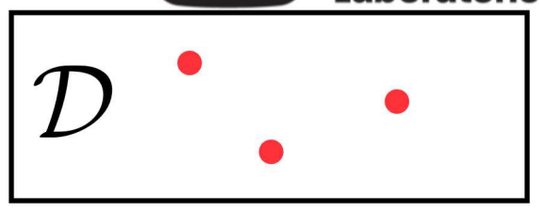


1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \ldots, T$

Least-squares Petrov–Galerkin (LSPG)

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



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

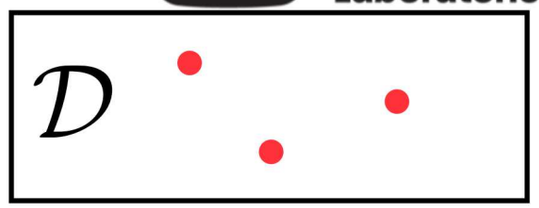
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

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

Least-squares Petrov–Galerkin (LSPG)

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



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

Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

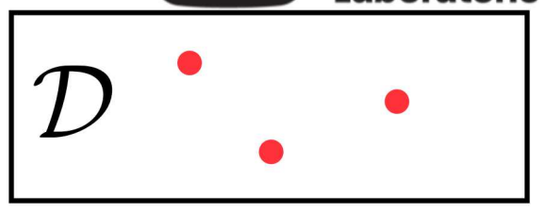
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$$

2. Minimize the OΔE residual

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\, \mathbf{r}^n(\boldsymbol{\Phi}\, \hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2$$

Least-squares Petrov–Galerkin (LSPG)

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



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

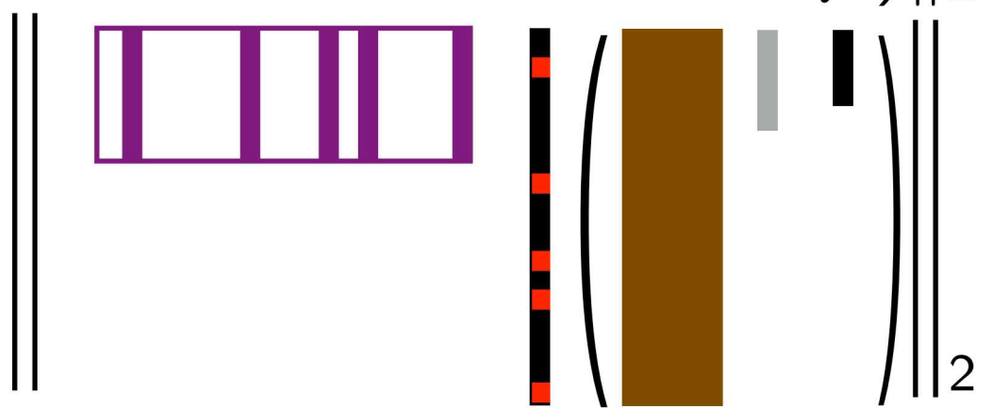
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

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

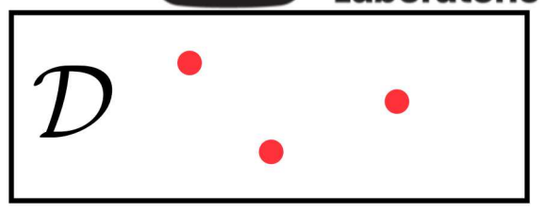
2. Minimize the OΔE residual

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\, \mathbf{r}^n(\boldsymbol{\Phi}\, \hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2$$



Least-squares Petrov–Galerkin (LSPG)

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



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

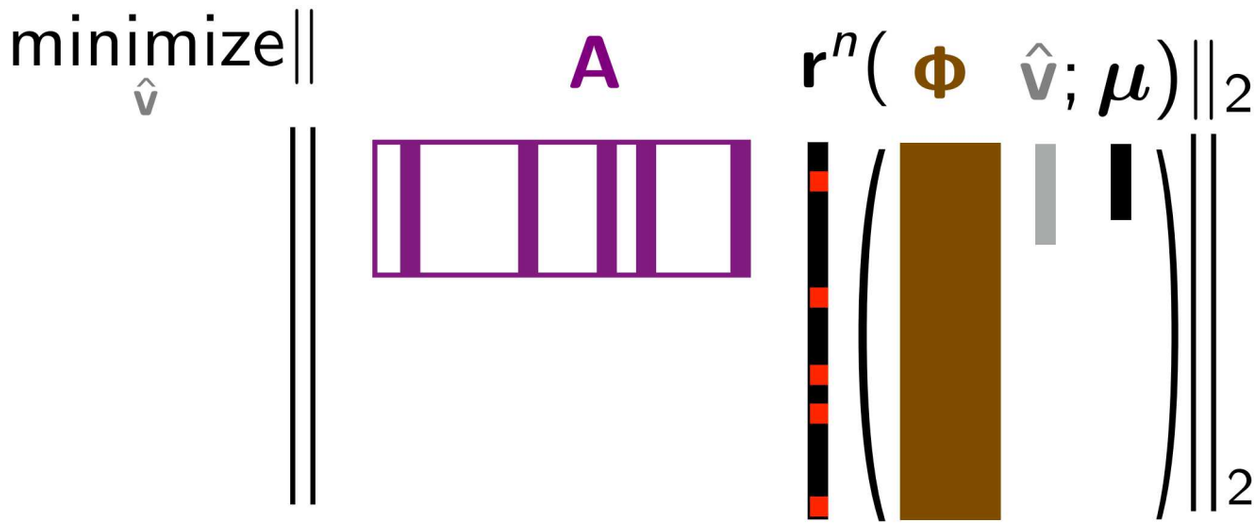
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

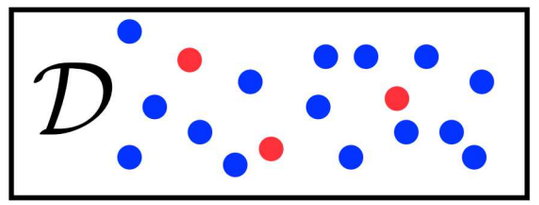
$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$

2. Minimize the OΔE residual

$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\, \mathbf{r}^n(\boldsymbol{\Phi}\, \hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2$

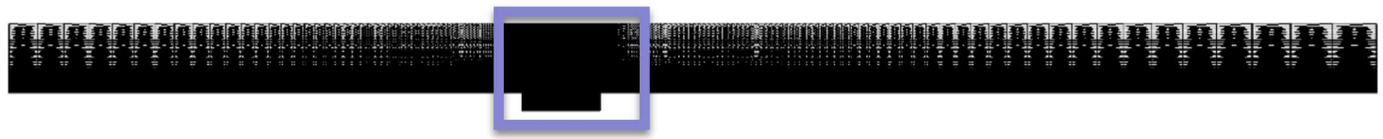


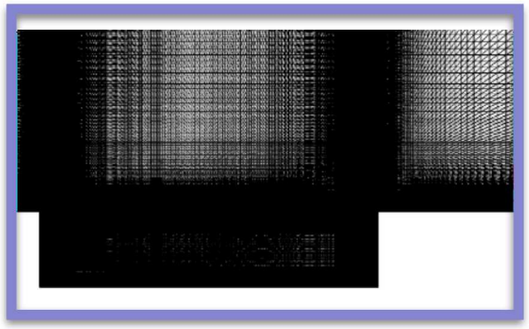
LSPG OΔE: $\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2^2$



Captive carry results [C., Barone, Antil, 2017]

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \; \left\| \mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2^2$$

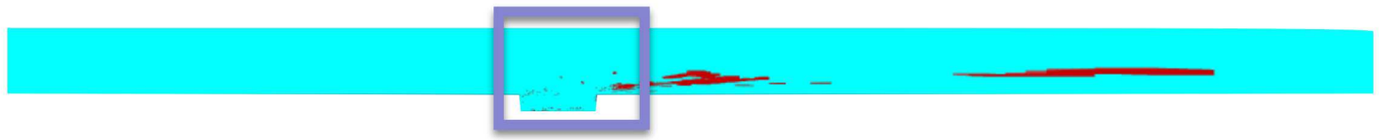


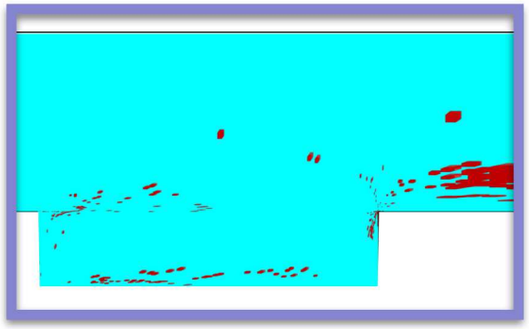


Captive carry results [C., Barone, Antil, 2017]

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \; \left\| \mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2^2$$

sample mesh



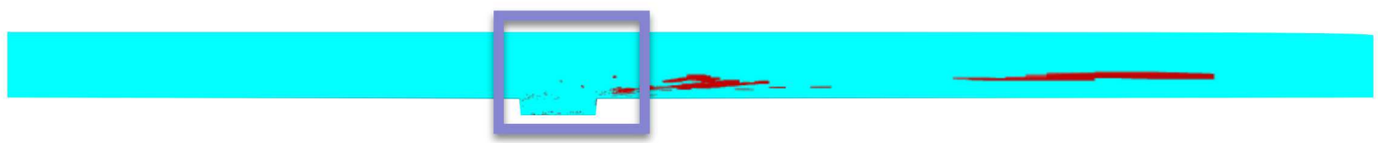


+ *HPC on a laptop*

Captive carry results [C., Barone, Antil, 2017]

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \; \left\| \mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2^2$$

sample mesh



+ *HPC on a laptop*

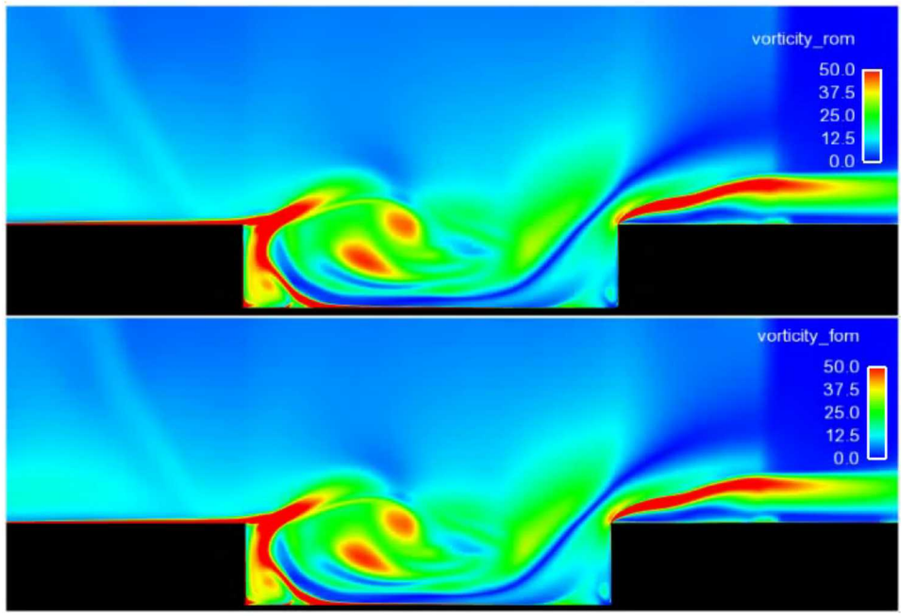
vorticity field *pressure field*

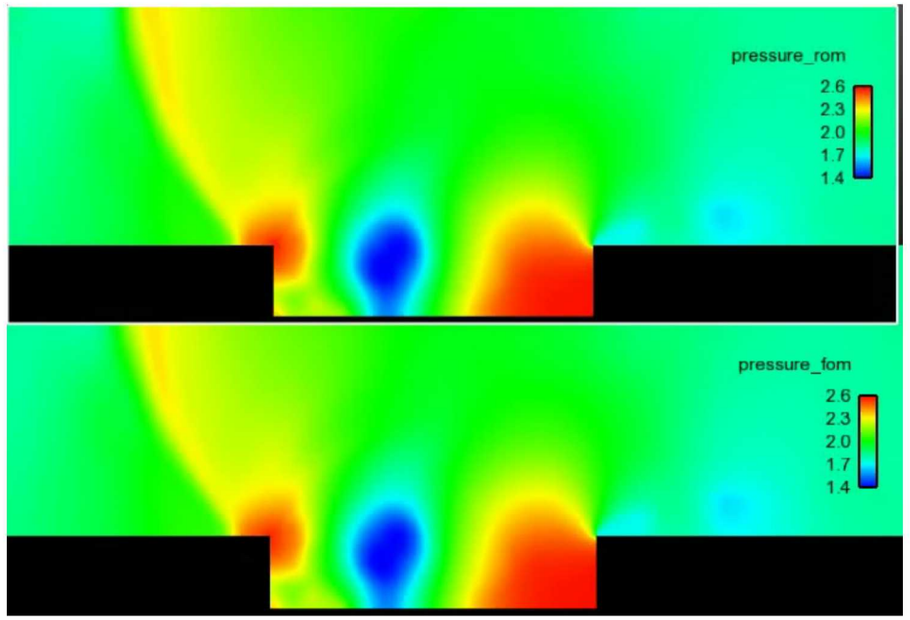
LSPG ROM

32 min, 2 cores

high-fidelity

5 hours, 48 cores





+ *229x savings in core–hours*

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

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
- [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

Collaborator: *Youngsoo Choi*

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

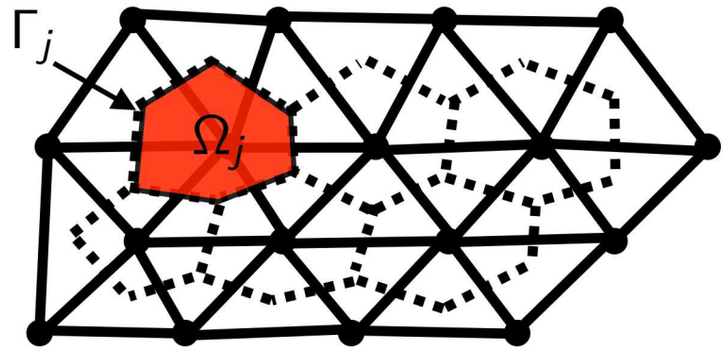
Collaborators: *Youngsoo Choi (Sandia), Syuzanna Sargsyan (UW)*

Finite-volume method

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

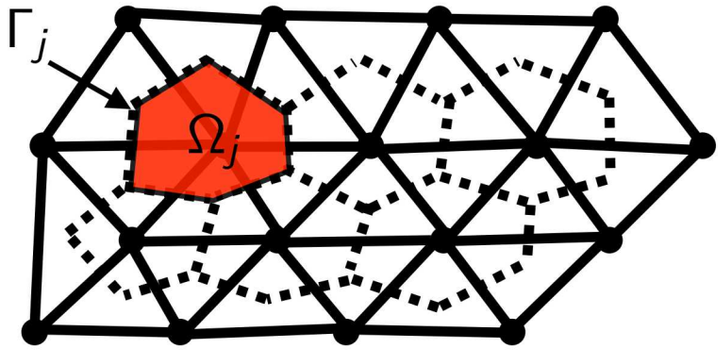
Finite-volume method

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t)$



Finite-volume method

ODE: $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t)$



$$x_{\mathcal{I}(i,j)}(t) = \frac{1}{|\Omega_j|}\int_{\Omega_j} u_i(\vec{x}, t)\,\mathrm{d}\vec{x}$$

- average value of conserved variable *i* over control volume *j*

$$f_{\mathcal{I}(i,j)}(\mathbf{x}, t) = -\frac{1}{|\Omega_j|}\int_{\Gamma_j} \underbrace{\mathbf{g}_i(\mathbf{x}; \vec{x}, t)}_{\text{flux}} \cdot \mathbf{n}_j(\vec{x})\,\mathrm{d}\vec{s}(\vec{x}) + \frac{1}{|\Omega_j|}\int_{\Omega_j} \underbrace{s_i(\mathbf{x}; \vec{x}, t)}_{\text{source}}\,\mathrm{d}\vec{x}$$

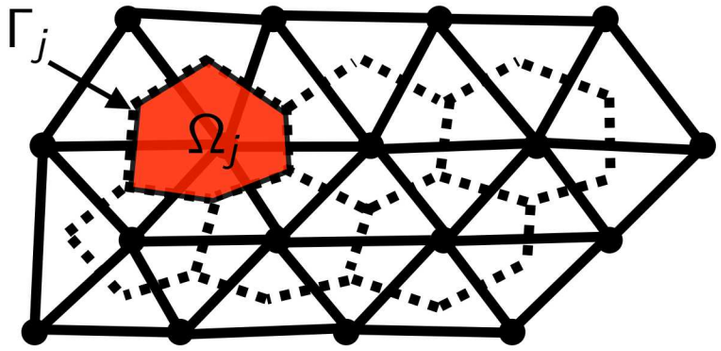
- flux and source of conserved variable *i* within control volume *j*

$$r_{\mathcal{I}(i,j)} = \frac{dx_{\mathcal{I}(i,j)}}{dt}(t) - f_{\mathcal{I}(i,j)}(\mathbf{x}, t)$$

- **rate of conservation violation** of variable *i* in control volume *j*

Finite-volume method

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



$$x_{\mathcal{I}(i,j)}(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} u_i(\vec{x}, t)\, \mathrm{d}\vec{x}$$

- average value of conserved variable *i* over control volume *j*

$$f_{\mathcal{I}(i,j)}(\mathbf{x}, t) = -\frac{1}{|\Omega_j|} \int_{\Gamma_j} \underbrace{\mathbf{g}_i(\mathbf{x}; \vec{x}, t)}_{\text{flux}} \cdot \mathbf{n}_j(\vec{x})\, \mathrm{d}\vec{s}(\vec{x}) + \frac{1}{|\Omega_j|} \int_{\Omega_j} \underbrace{s_i(\mathbf{x}; \vec{x}, t)}_{\text{source}}\, \mathrm{d}\vec{x}$$

- flux and source of conserved variable *i* within control volume *j*

$$r_{\mathcal{I}(i,j)} = \frac{dx_{\mathcal{I}(i,j)}}{dt}(t) - f_{\mathcal{I}(i,j)}(\mathbf{x}, t)$$

- **rate of conservation violation** of variable *i* in control volume *j*

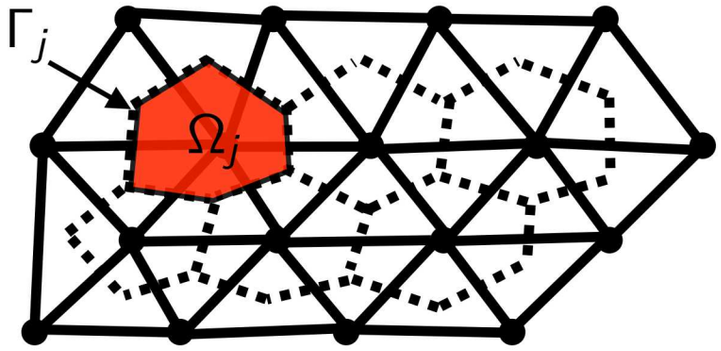
$$\text{O}\Delta\text{E:} \quad \mathbf{r}^n(\mathbf{x}^n) = 0, \;\; n = 1, \ldots, N$$

$$r^n_{\mathcal{I}(i,j)} = x_{\mathcal{I}(i,j)}(t^{n+1}) - x_{\mathcal{I}(i,j)}(t^n) + \int_{t^n}^{t^{n+1}} f_{\mathcal{I}(i,j)}(\mathbf{x}, t)dt$$

- **conservation violation** of variable *i* in control volume *j* over time step *n*

Finite-volume method

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



$$x_{\mathcal{I}(i,j)}(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} u_i(\vec{x}, t)\, \mathrm{d}\vec{x}$$

- average value of conserved variable *i* over control volume *j*

$$f_{\mathcal{I}(i,j)}(\mathbf{x}, t) = -\frac{1}{|\Omega_j|} \int_{\Gamma_j} \underbrace{\mathbf{g}_i(\mathbf{x}; \vec{x}, t)}_{\text{flux}} \cdot \mathbf{n}_j(\vec{x})\, \mathrm{d}\vec{s}(\vec{x}) + \frac{1}{|\Omega_j|} \int_{\Omega_j} \underbrace{s_i(\mathbf{x}; \vec{x}, t)}_{\text{source}}\, \mathrm{d}\vec{x}$$

- flux and source of conserved variable *i* within control volume *j*

$$r_{\mathcal{I}(i,j)} = \frac{dx_{\mathcal{I}(i,j)}}{dt}(t) - f_{\mathcal{I}(i,j)}(\mathbf{x}, t)$$

- **rate of conservation violation** of variable *i* in control volume *j*

$$\text{O}\Delta\text{E:} \quad \mathbf{r}^n(\mathbf{x}^n) = 0, \;\; n = 1, \ldots, N$$

$$r^n_{\mathcal{I}(i,j)} = x_{\mathcal{I}(i,j)}(t^{n+1}) - x_{\mathcal{I}(i,j)}(t^n) + \int_{t^n}^{t^{n+1}} f_{\mathcal{I}(i,j)}(\mathbf{x}, t)dt$$

- **conservation violation** of variable *i* in control volume *j* over time step *n*

Conservation is the intrinsic structure enforced by finite-volume methods

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

LSPG

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

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

- Minimize sum of squared conservation-violation **rates**

LSPG

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

- Minimize sum of squared conservation violations **over time step *n***

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

- Minimize sum of squared conservation-violation **rates**

LSPG

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

- Minimize sum of squared conservation violations **over time step *n***

- Neither enforces conservation!

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

- Minimize sum of squared conservation-violation **rates**

LSPG

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

- Minimize sum of squared conservation violations **over time step *n***

- Neither enforces conservation!

Conservative Galerkin

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}(\mathbf{v},\mathbf{x};t)\right\|_2$$

$$\text{subject to } \mathbf{Cr}(\mathbf{v},\mathbf{x};t) = \mathbf{0}$$

- Minimize sum of squared conservation-violation **rates**
  subject to zero conservation-violation **rates** over subdomains

Conservative LSPG

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

$$\text{subject to } \mathbf{Cr}^n(\mathbf{v}) = \mathbf{0}$$

- Minimize sum of squared conservation violations **over time step *n***
  subject to zero conservation violations **over time step *n*** over subdomains

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

- Minimize sum of squared conservation-violation **rates**

LSPG

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

- Minimize sum of squared conservation violations **over time step *n***

- Neither enforces conservation!

Conservative Galerkin

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}(\mathbf{v},\mathbf{x};t)\right\|_2$$

$$\text{subject to } \mathbf{Cr}(\mathbf{v},\mathbf{x};t) = \mathbf{0}$$

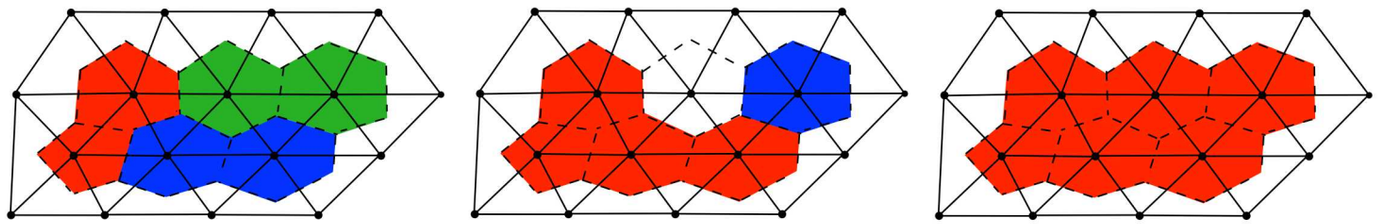
- Minimize sum of squared conservation-violation **rates**
  subject to zero conservation-violation **rates** over subdomains

Conservative LSPG

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

$$\text{subject to } \mathbf{Cr}^n(\mathbf{v}) = \mathbf{0}$$

- Minimize sum of squared conservation violations **over time step *n***
  subject to zero conservation violations **over time step *n*** over subdomains



+ Conservation enforced over prescribed subdomains

Conservative model reduction [C., Choi, Sargsyan, 2018]

Galerkin

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

- Minimize sum of squared conservation-violation **rates**

LSPG

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\arg\min} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

- Minimize sum of squared conservation violations **over time step *n***

- Neither enforces conservation!

Conservative Galerkin

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}(\mathbf{v},\mathbf{x};t)\right\|_2$$

$$\text{subject to } \mathbf{Cr}(\mathbf{v},\mathbf{x};t) = \mathbf{0}$$

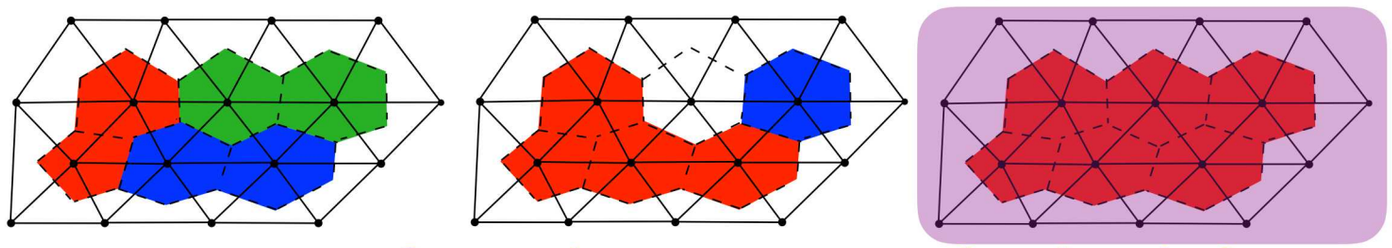
- Minimize sum of squared conservation-violation **rates**
  subject to zero conservation-violation **rates** over subdomains

Conservative LSPG

$$\underset{\mathbf{v}\in\text{range}(\boldsymbol{\Phi})}{\text{minimize}} \left\|\mathbf{r}^n(\mathbf{v})\right\|_2$$

$$\text{subject to } \mathbf{Cr}^n(\mathbf{v}) = \mathbf{0}$$

- Minimize sum of squared conservation violations **over time step *n***
  subject to zero conservation violations **over time step *n*** over subdomains



+ Conservation enforced over prescribed subdomains

- **Experiments**: enforcing global conservation can reduce error by 10X

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

Collaborator: *Kookjin Lee*

Model reduction can work well…

vorticity field | *pressure field*

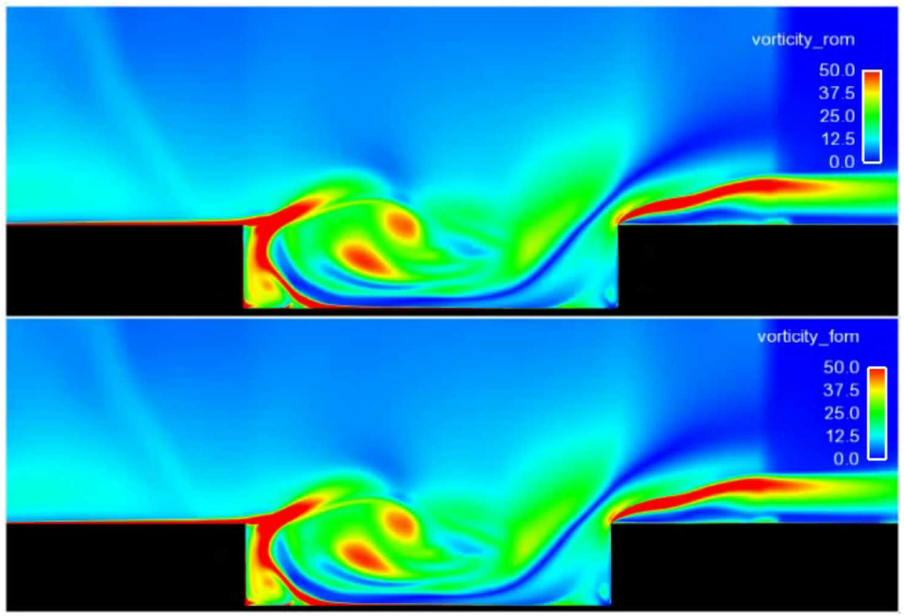
LSPG ROM with

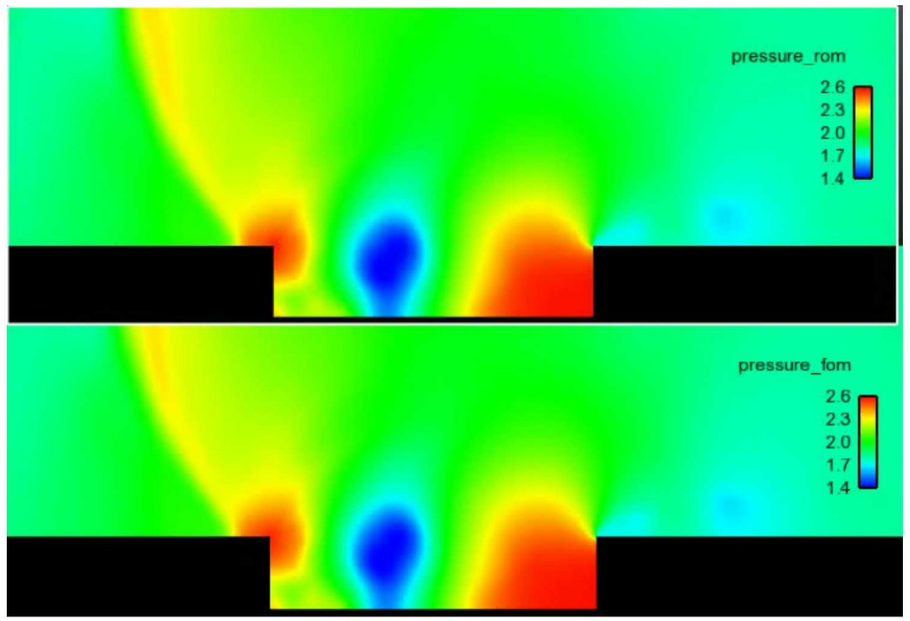
$$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}$$

32 min, 2 cores

high-fidelity

5 hours, 48 cores





+ *229x savings in core–hours*

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

… however, this is not guaranteed

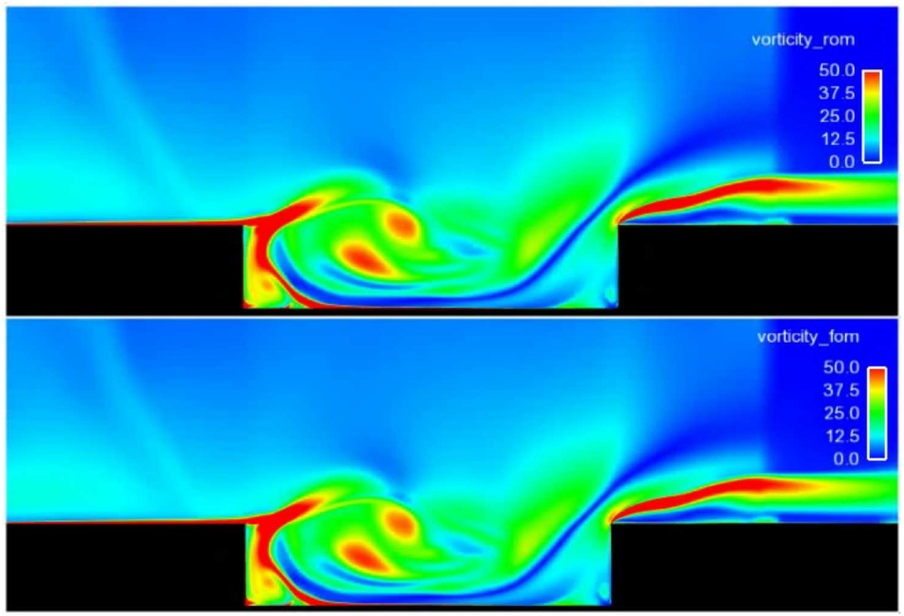
Model reduction can work well…

vorticity field | *pressure field*

LSPG ROM with

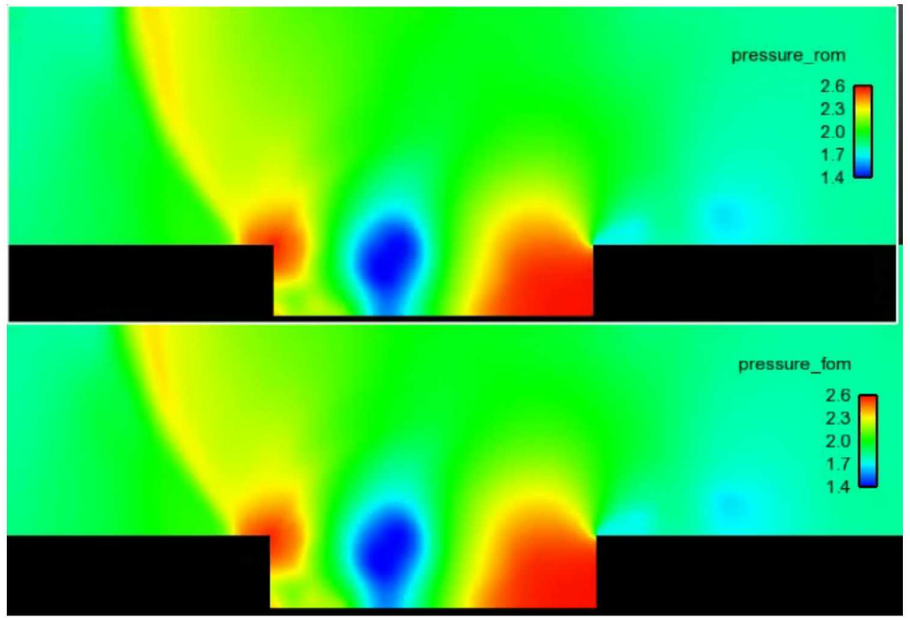
$$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}$$

32 min, 2 cores



high-fidelity

5 hours, 48 cores



+ *229x savings in core–hours*

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

… however, this is not guaranteed

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

Model reduction can work well…

vorticity field | *pressure field*

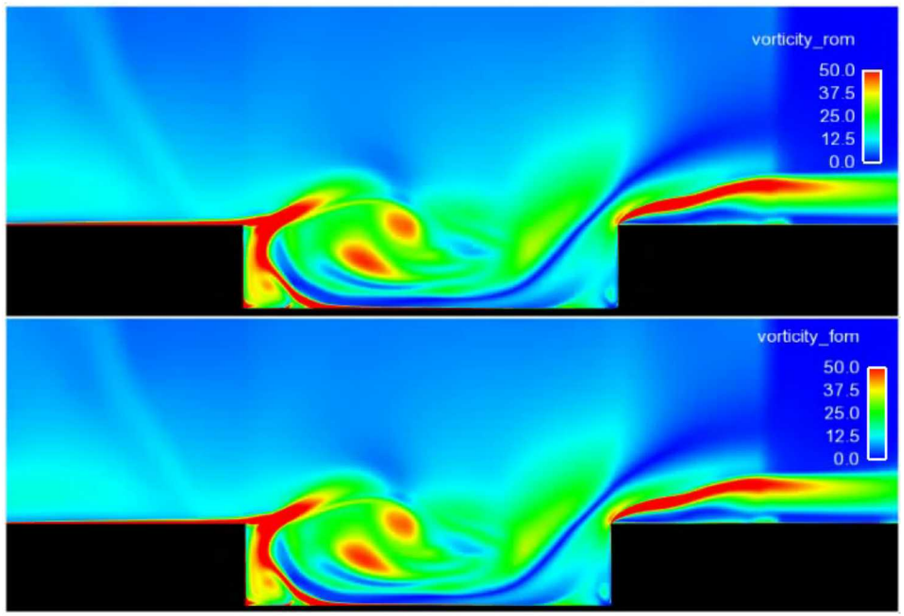
LSPG ROM with

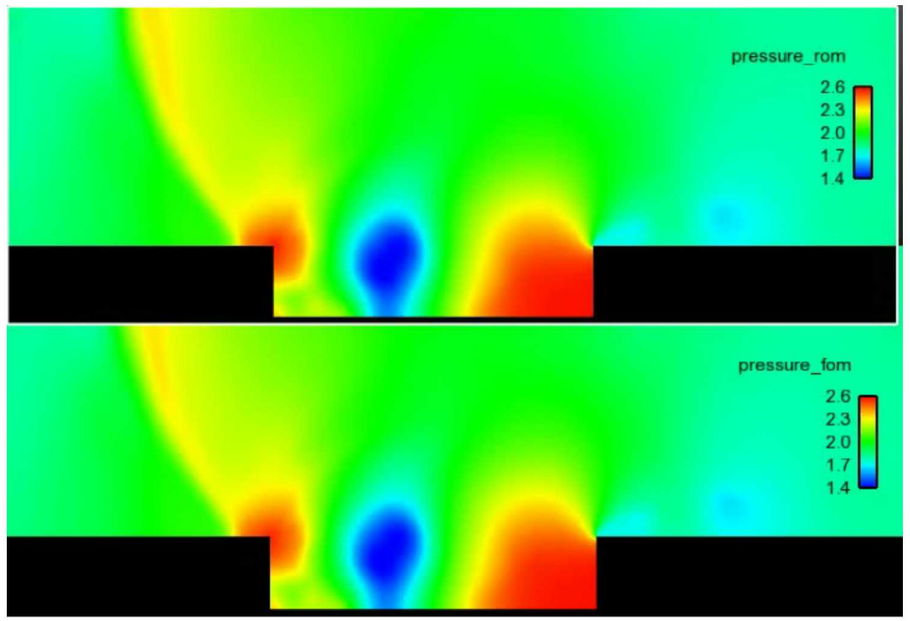
$$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}$$

32 min, 2 cores

high-fidelity

5 hours, 48 cores





+ *229x savings in core–hours*

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

… however, this is not guaranteed

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

1) Linear-subspace assumption is strong

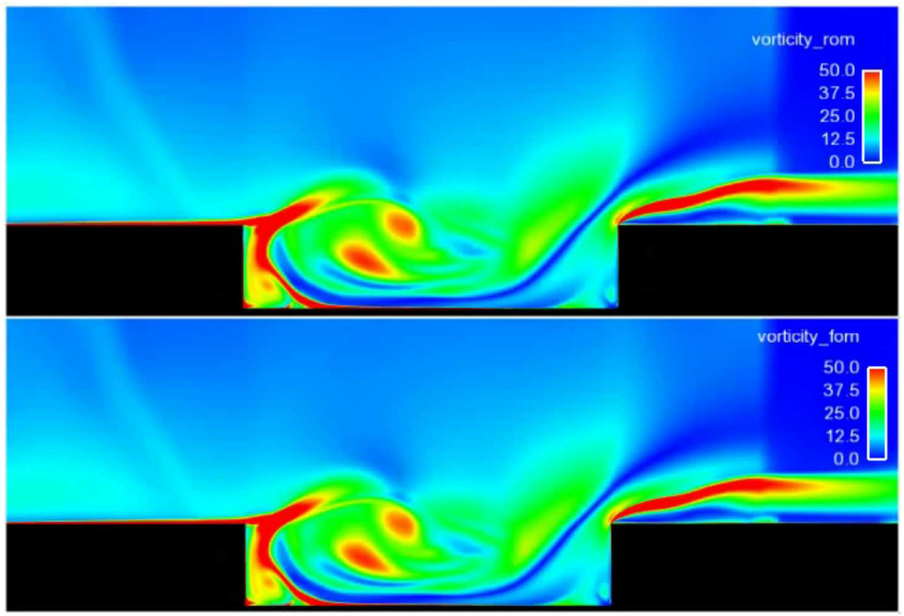
Model reduction can work well…

vorticity field | *pressure field*

LSPG ROM with

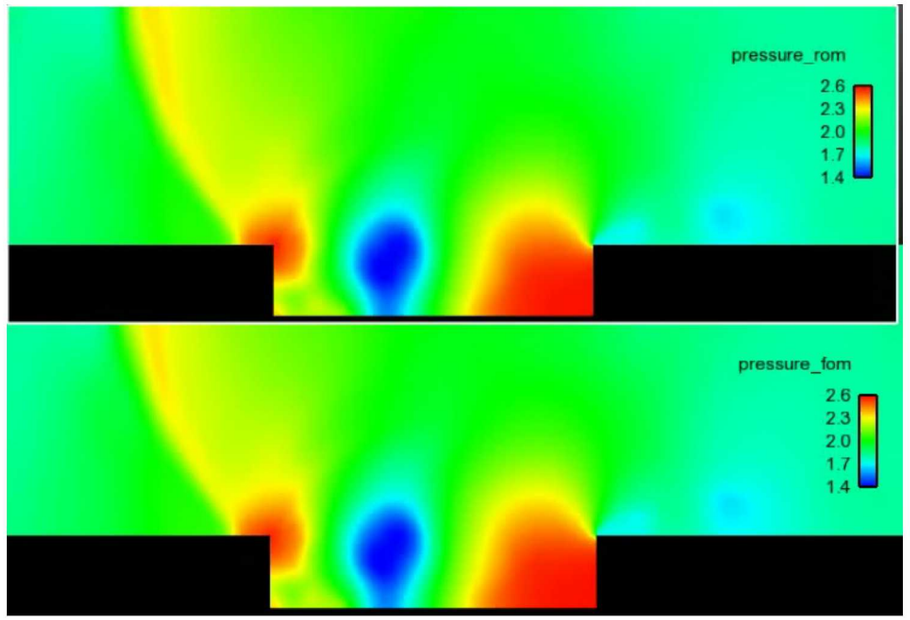
$$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}$$

32 min, 2 cores



high-fidelity

5 hours, 48 cores



+ *229x savings in core–hours*

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

… however, this is not guaranteed

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

1) Linear-subspace assumption is strong

2) Accuracy limited by information in $\boldsymbol{\Phi}$

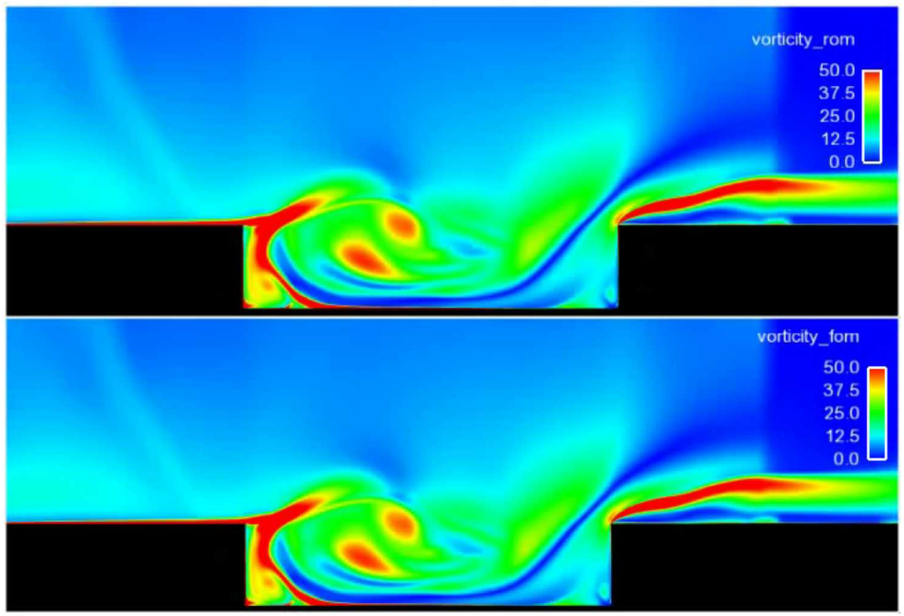
Model reduction can work well…

vorticity field | *pressure field*

LSPG ROM with

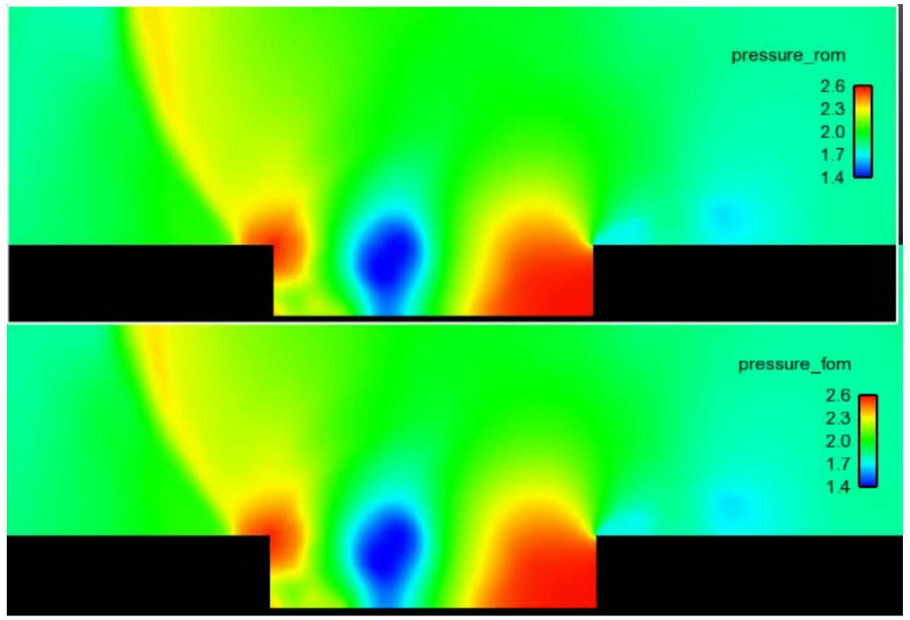
$$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}$$

32 min, 2 cores



high-fidelity

5 hours, 48 cores



+ *229x savings in core–hours*

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

… however, this is not guaranteed

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

1) Linear-subspace assumption is strong ←

2) Accuracy limited by information in $\boldsymbol{\Phi}$

Machine learning

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

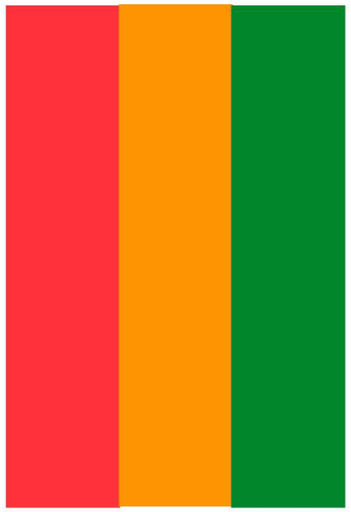
1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{X} =$$

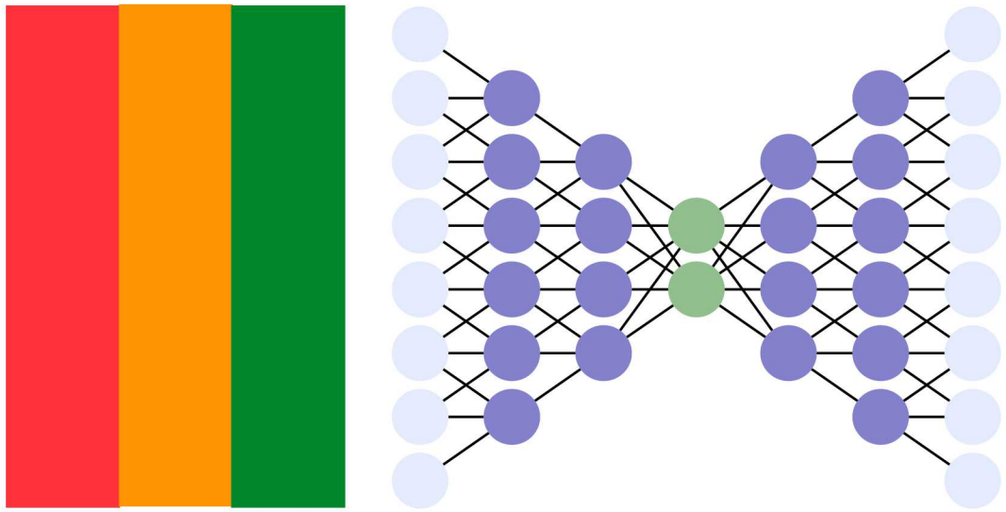


Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$\mathbf{x} =$

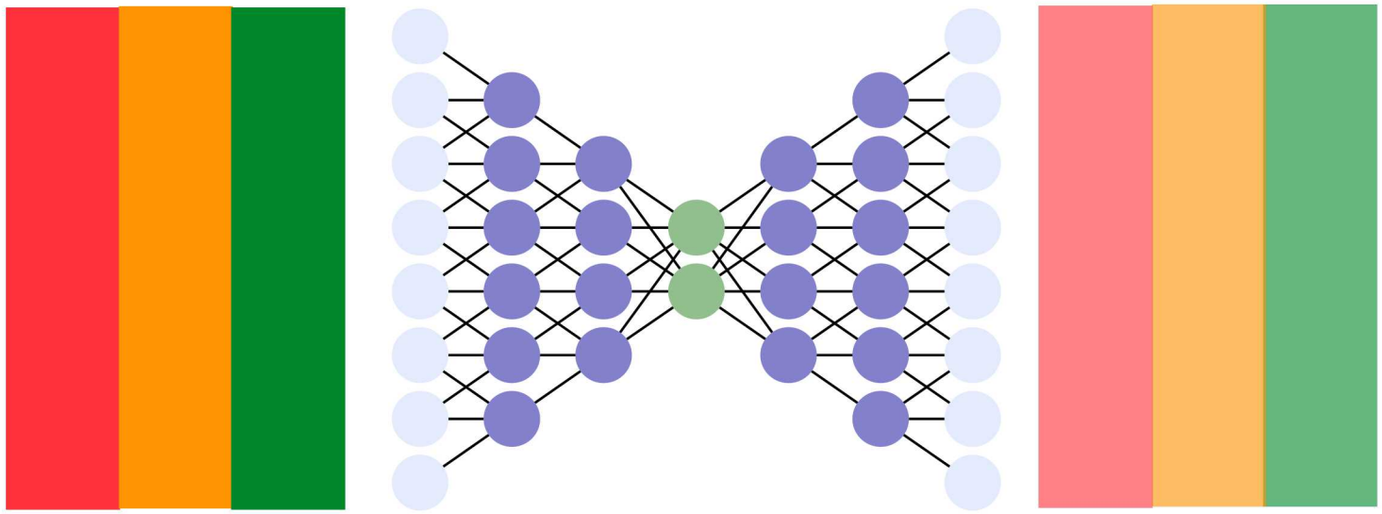


Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$

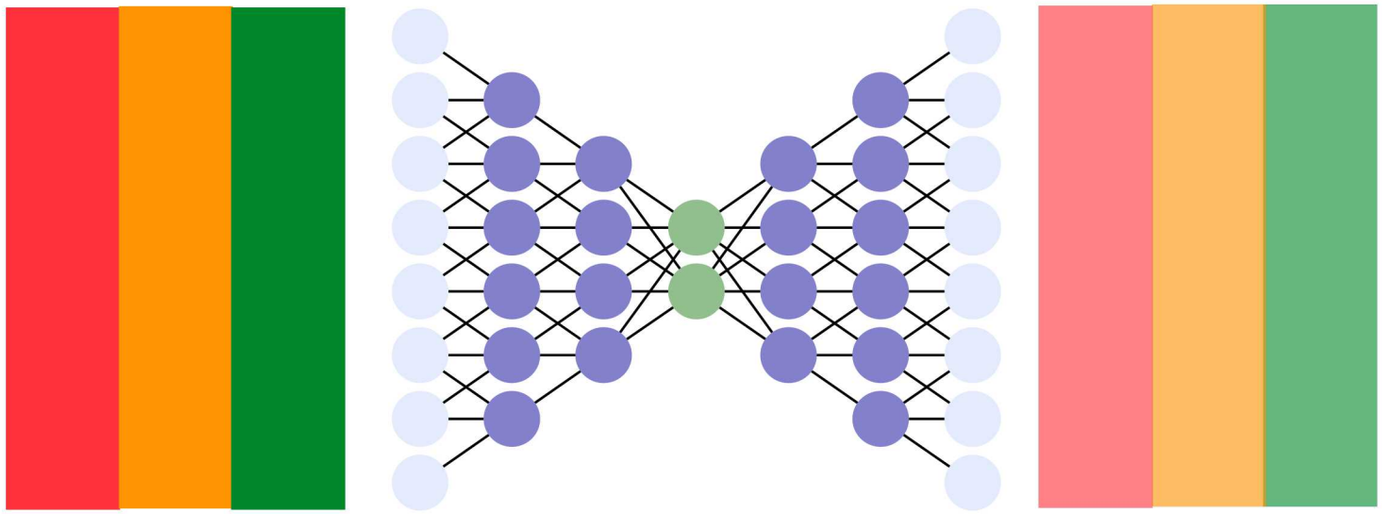


$$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

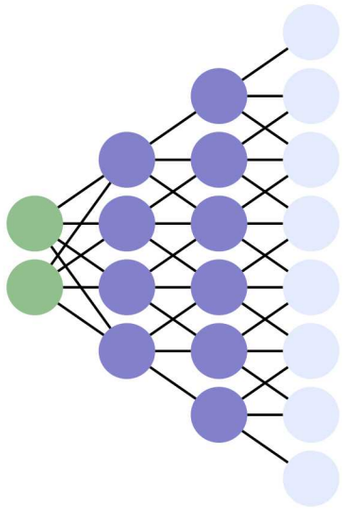
Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$  $$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

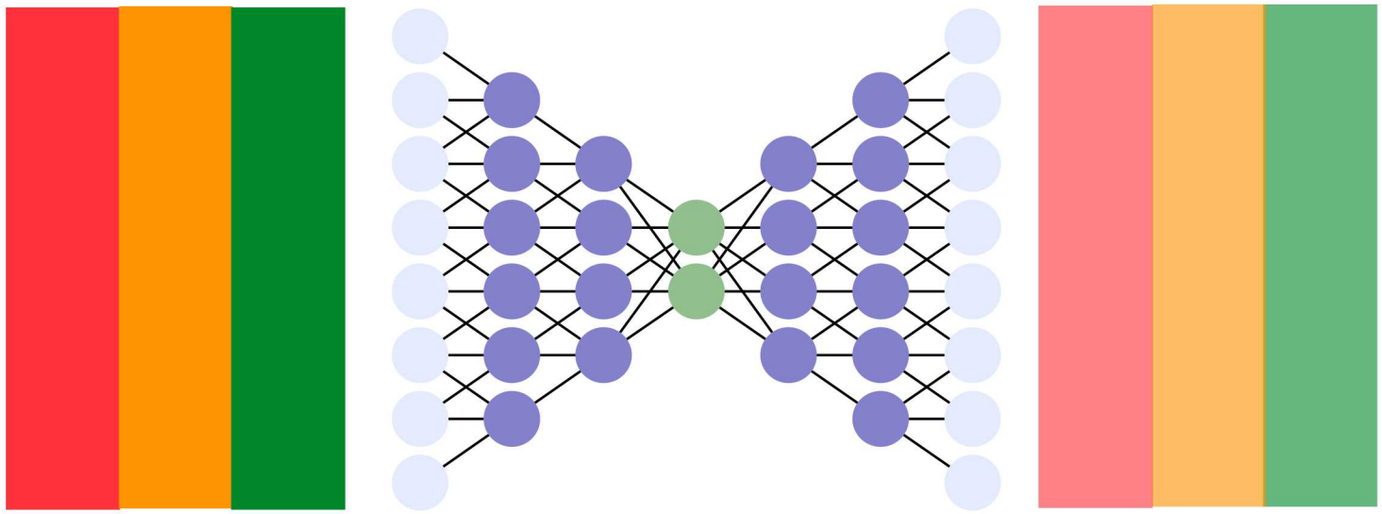
- Define low-dim manifold from decoder:



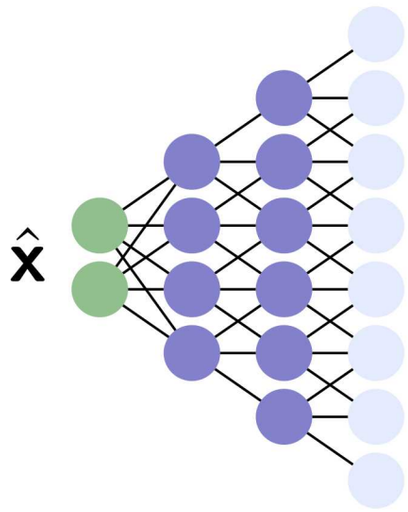
Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$  $$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

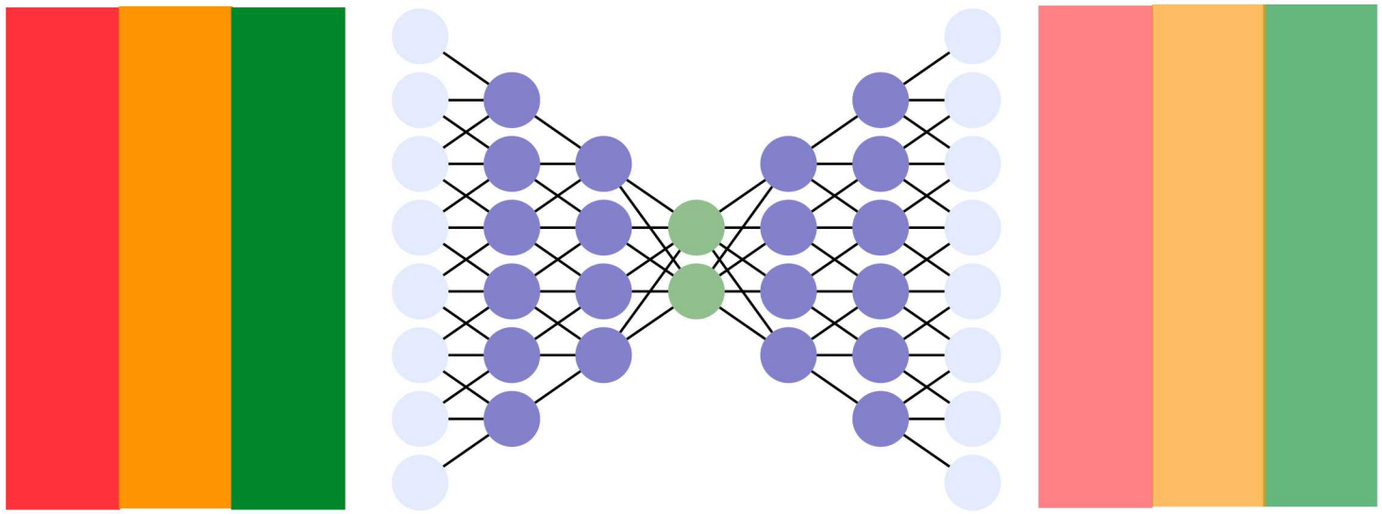
- Define low-dim manifold from decoder:



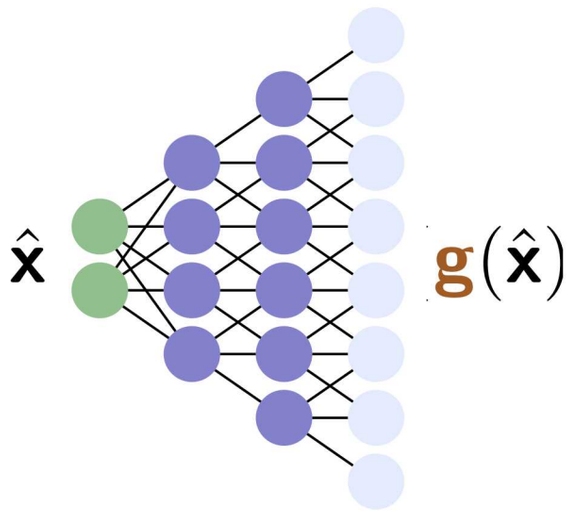
Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$  $$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

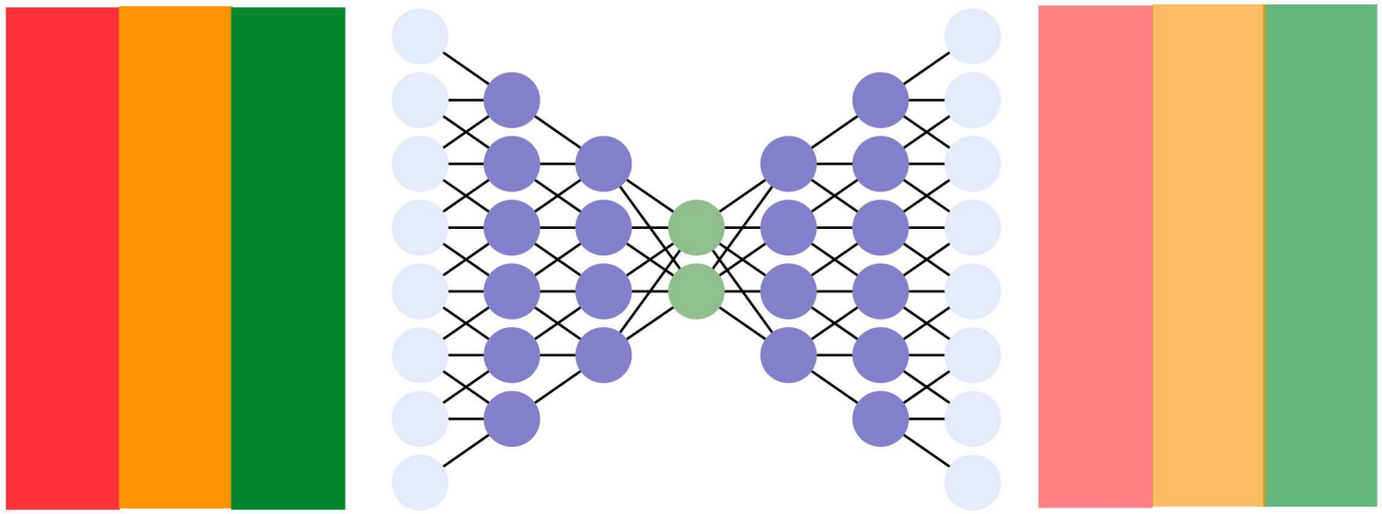
- Define low-dim manifold from decoder:



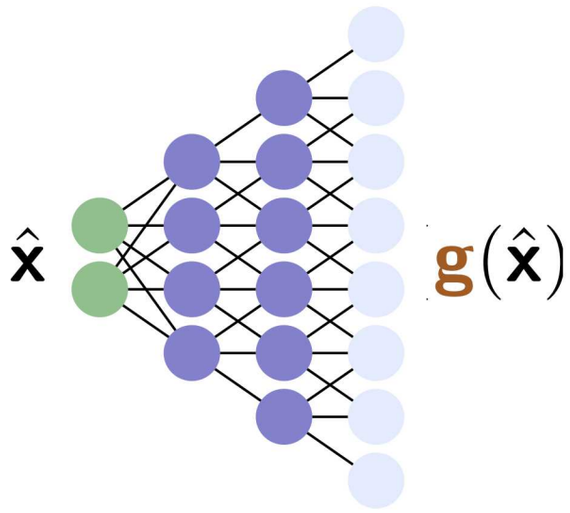
Machine learning

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$  $$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

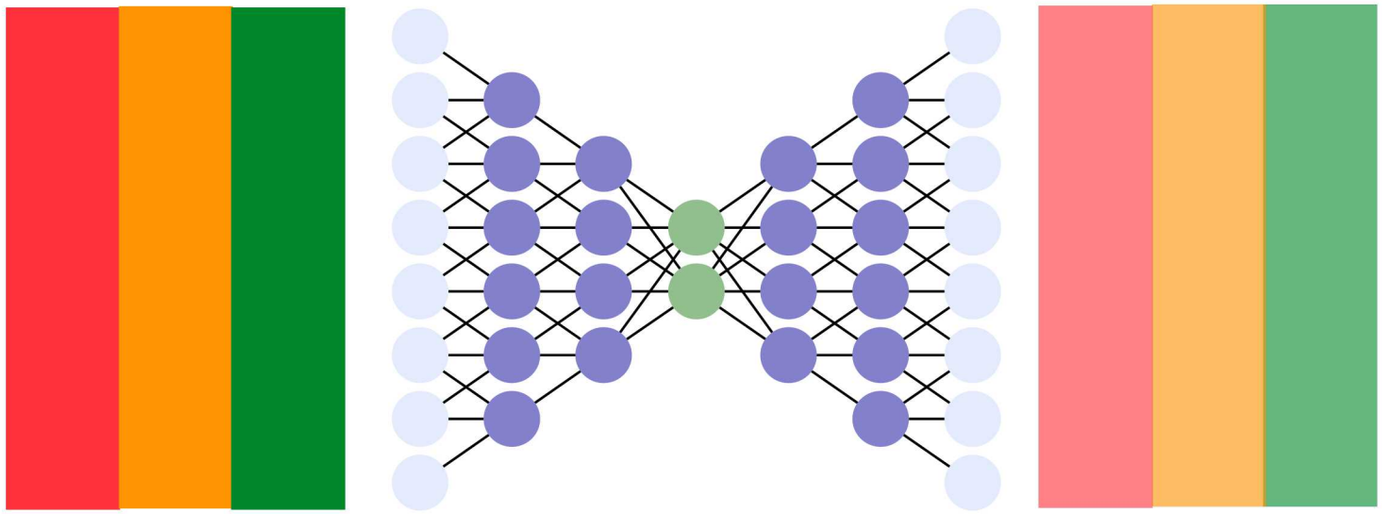
- Define low-dim manifold from decoder: $\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\} \subseteq \mathbb{R}^N$



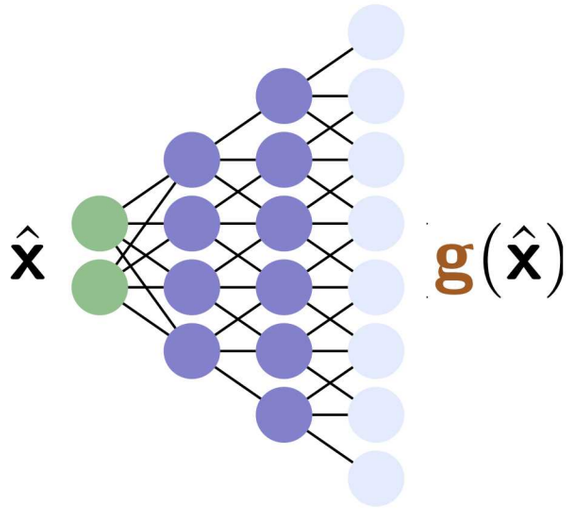
Machine learning

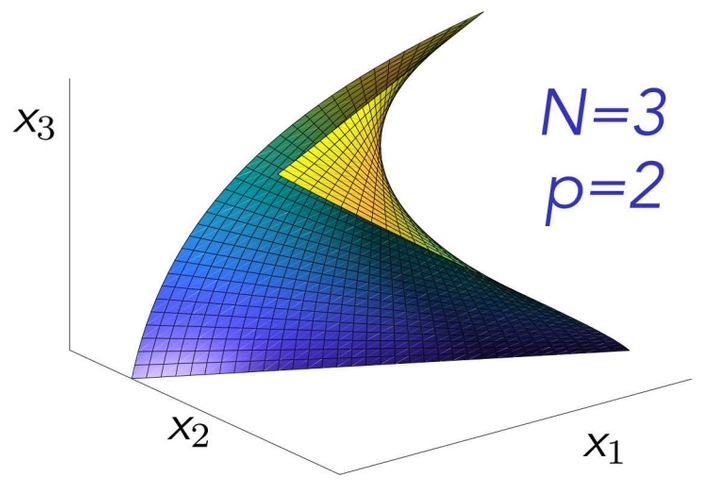
$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. ***Machine learning:*** Identify low-dimensional manifold
3. *Reduction:* Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

$$\mathbf{x} =$$  $$= \tilde{\mathbf{x}}(\boldsymbol{\theta})$$

- Define low-dim manifold from decoder: $\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\} \subseteq \mathbb{R}^N$





Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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

Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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

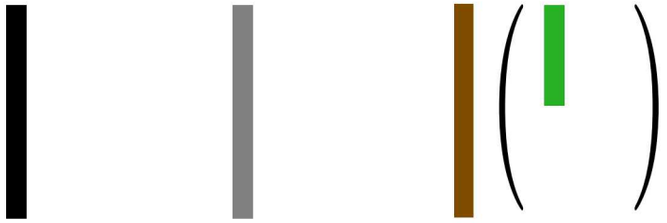
Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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



Perform optimal projection

$$\frac{d\tilde{\mathbf{x}}}{dt}(\hat{\mathbf{x}}) \quad \text{satisfies} \quad \underset{\mathbf{v} \in T_{\hat{\mathbf{x}}}\mathcal{S}}{\text{minimize}} \; \|\mathbf{v} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); \mathrm{t}, \boldsymbol{\mu})\|_2$$

Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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

Perform optimal projection with physics constraints

$$\frac{d\tilde{\mathbf{x}}}{dt}(\hat{\mathbf{x}}) \quad \text{satisfies} \quad \underset{\mathbf{v}\in T_{\hat{\mathbf{x}}}\mathcal{S}}{\text{minimize}} \ \|\mathbf{v} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); \mathrm{t}, \boldsymbol{\mu})\|_2$$

$$\text{subject to } \mathbf{c}(\mathbf{v}, \mathbf{g}(\hat{\mathbf{x}}); t, \boldsymbol{\mu}) = \mathbf{0}$$

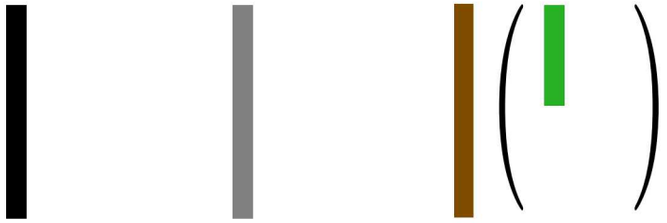
Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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



Perform optimal projection with physics constraints

$$\frac{d\tilde{\mathbf{x}}}{dt}(\hat{\mathbf{x}}) \quad \text{satisfies} \quad \underset{\mathbf{v} \in T_{\hat{\mathbf{x}}}\mathcal{S}}{\text{minimize}} \ \|\mathbf{v} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); \mathrm{t}, \boldsymbol{\mu})\|_2$$

$$\text{subject to } \mathbf{c}(\mathbf{v}, \mathbf{g}(\hat{\mathbf{x}}); t, \boldsymbol{\mu}) = \mathbf{0}$$

\+ Model integrates computational physics with deep learning

Reduction

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

1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify low-dimensional manifold
3. ***Reduction:*** Project ODE onto manifold and solve for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

Reduce the number of unknowns

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

Perform optimal projection with physics constraints

$$\frac{d\tilde{\mathbf{x}}}{dt}(\hat{\mathbf{x}}) \quad \text{satisfies} \quad \underset{\mathbf{v}\in T_{\hat{\mathbf{x}}}\mathcal{S}}{\text{minimize}} \ \|\mathbf{v} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); \mathrm{t}, \boldsymbol{\mu})\|_2$$

$$\text{subject to } \mathbf{c}(\mathbf{v}, \mathbf{g}(\hat{\mathbf{x}}); t, \boldsymbol{\mu}) = \mathbf{0}$$

+ Model integrates computational physics with deep learning
+ Physics constraints exactly satisfied

Numerical results

2D Chemically 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

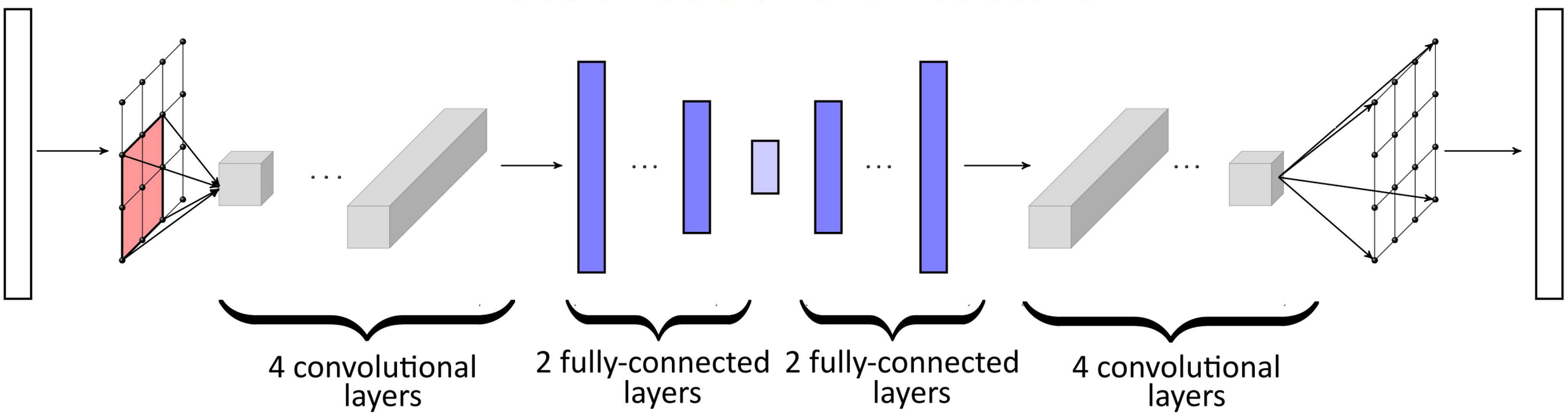
Numerical results

2D Chemically 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



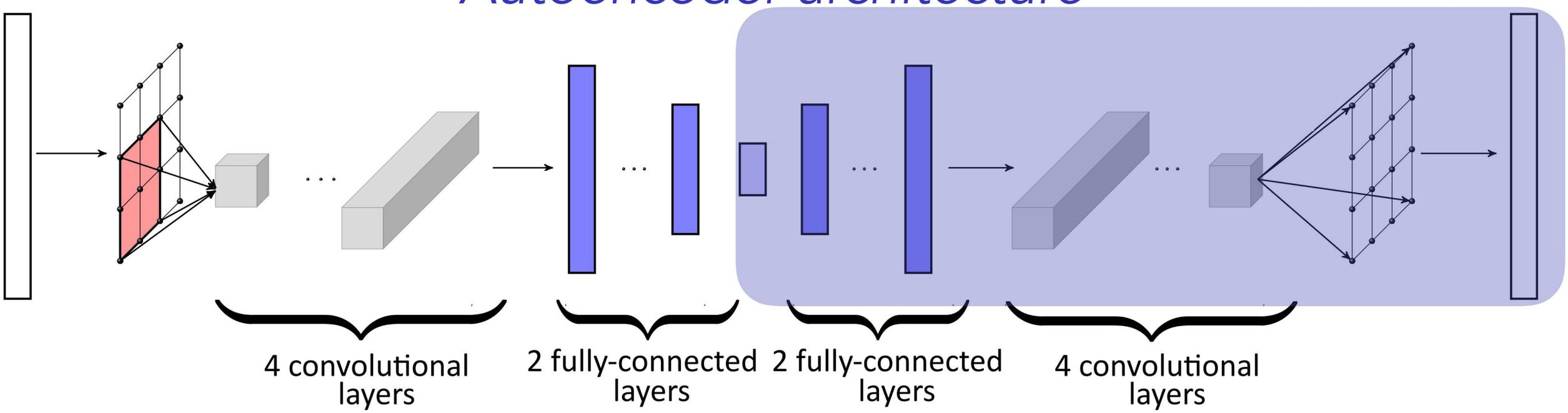
Numerical results

2D Chemically 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

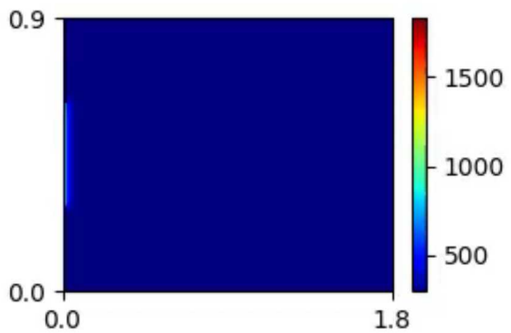


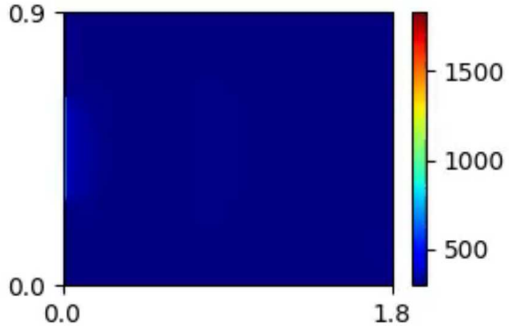
Manifold LSPG outperforms linear subspace LSPG

high-fidelity model

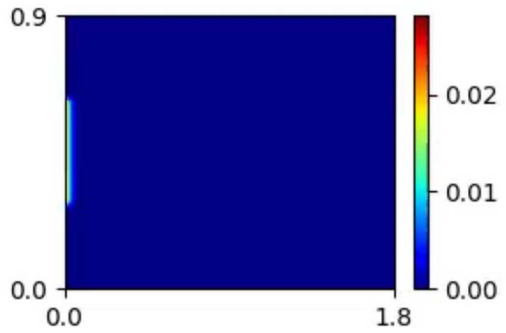
LSPG w/ PCA p=5

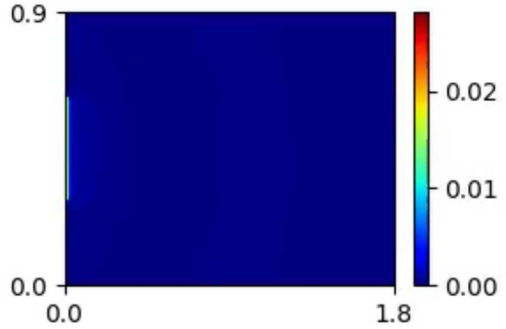
temperature





H_2 fraction



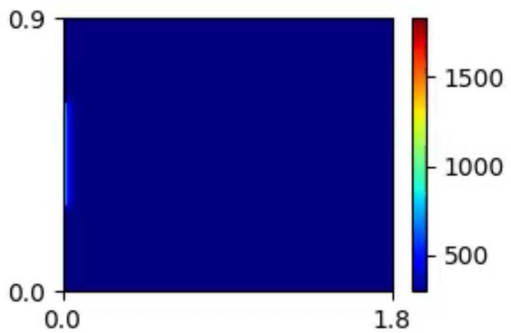


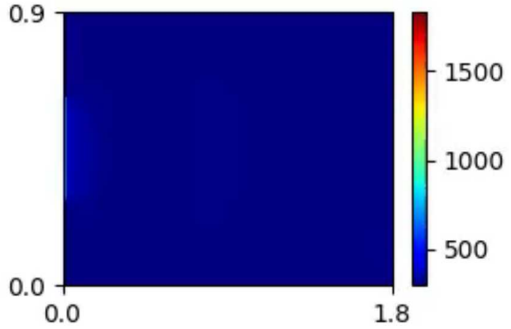
Manifold LSPG outperforms linear subspace LSPG

high-fidelity model

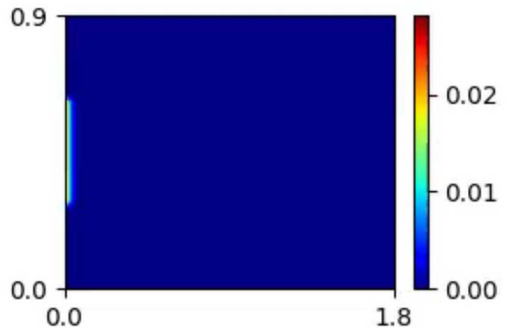
LSPG w/ PCA p=5

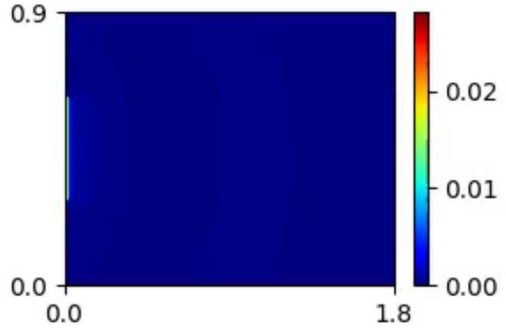
temperature





H_2 fraction



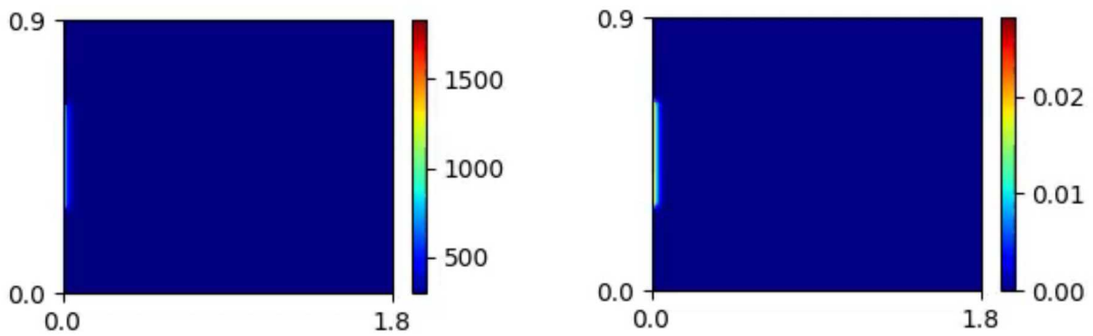


Manifold LSPG outperforms linear subspace LSPG

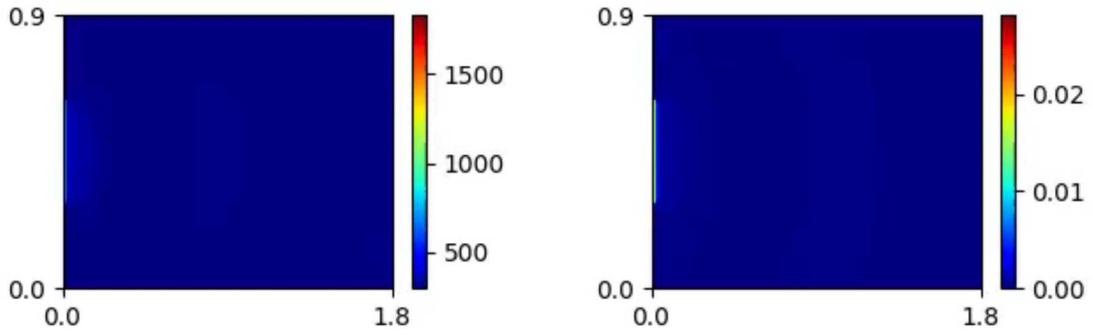
temperature

H_2 fraction

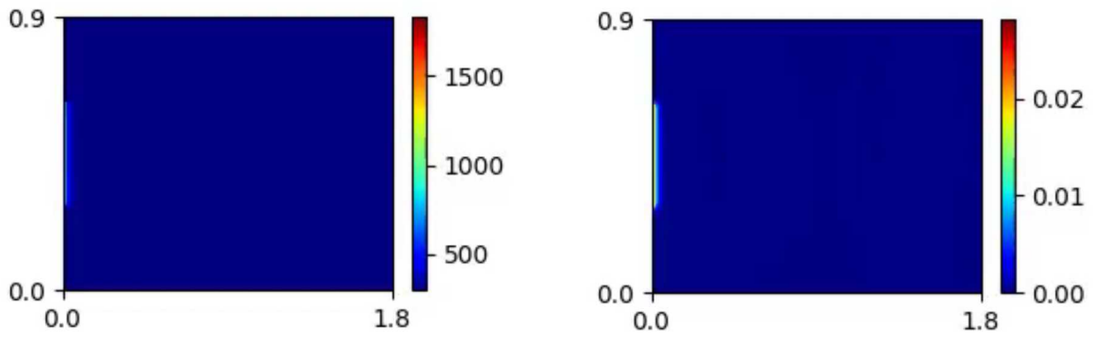
high-fidelity model



LSPG w/ PCA p=5



Manifold LSPG w/ autoencoder p=5

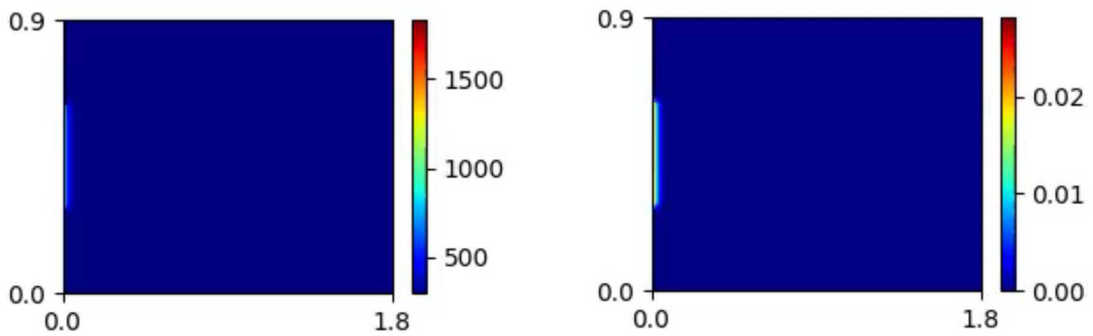


Manifold LSPG outperforms linear subspace LSPG

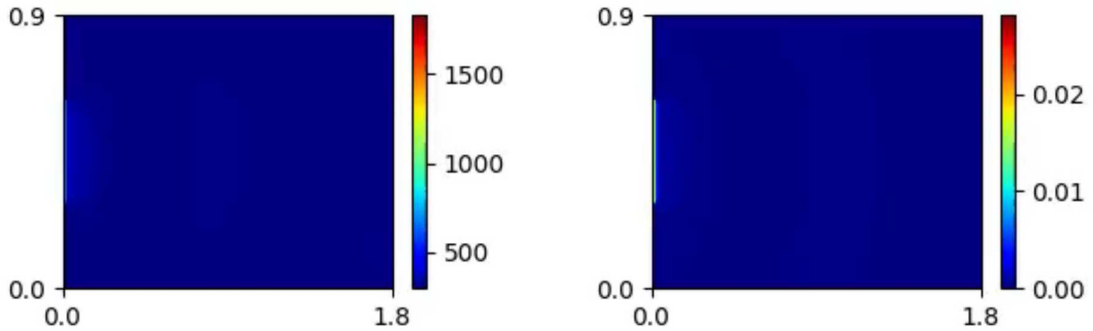
temperature

H_2 fraction

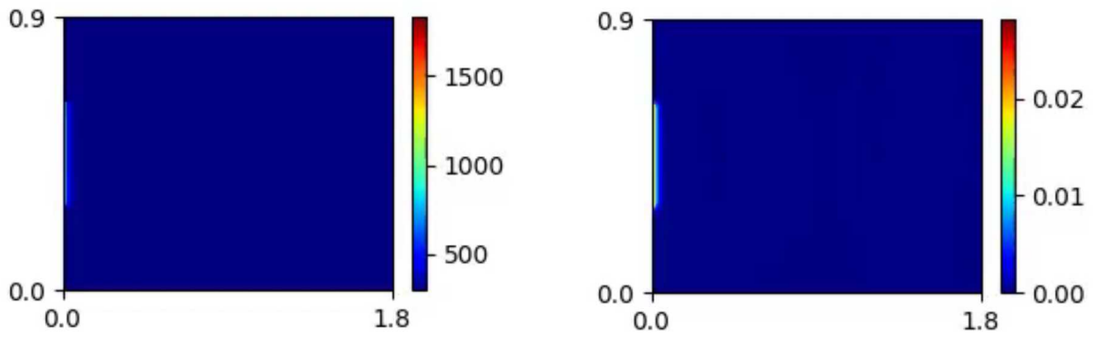
high-fidelity model



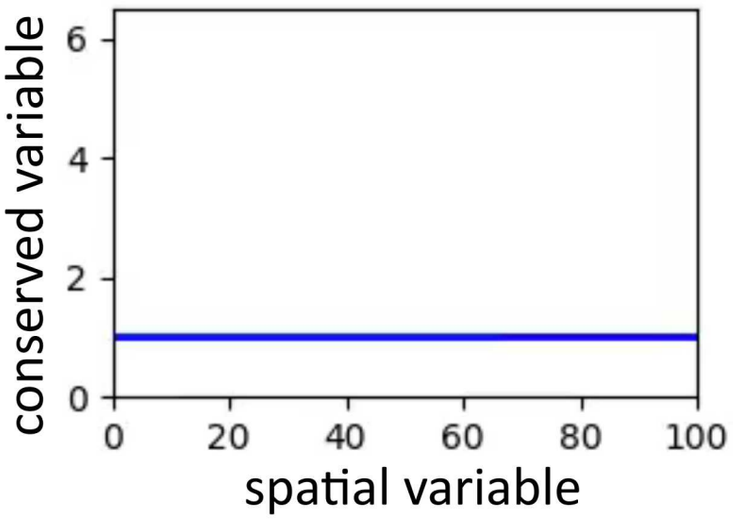
LSPG w/ PCA p=5



Manifold LSPG w/ autoencoder p=5

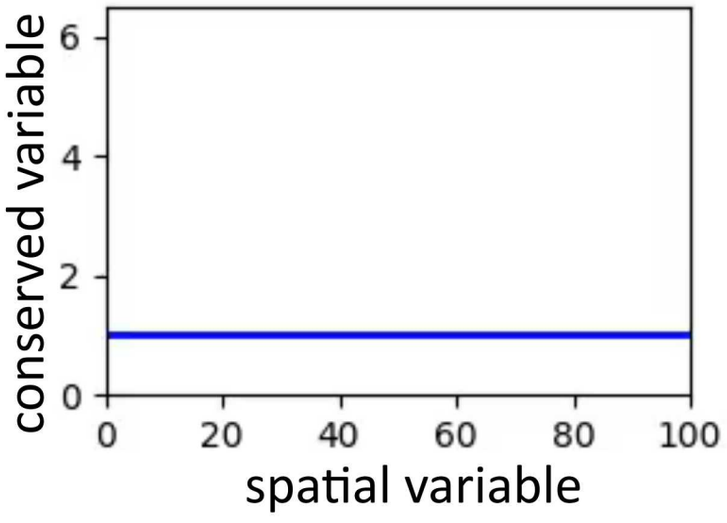


High-fidelity model

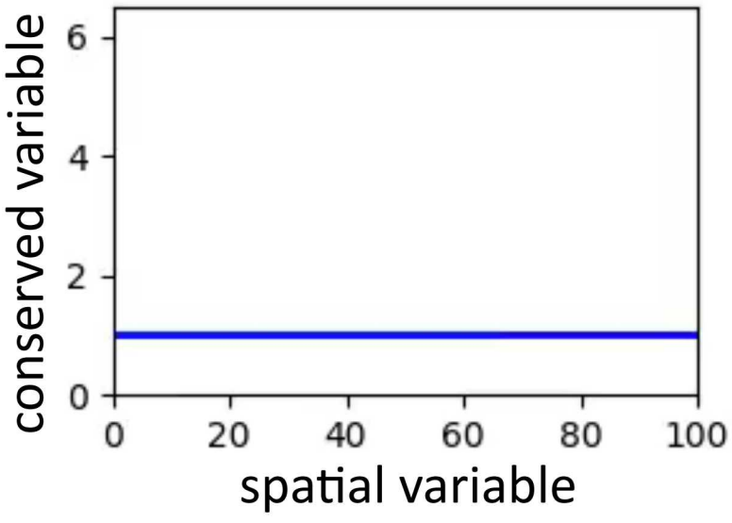


Reduced-order models

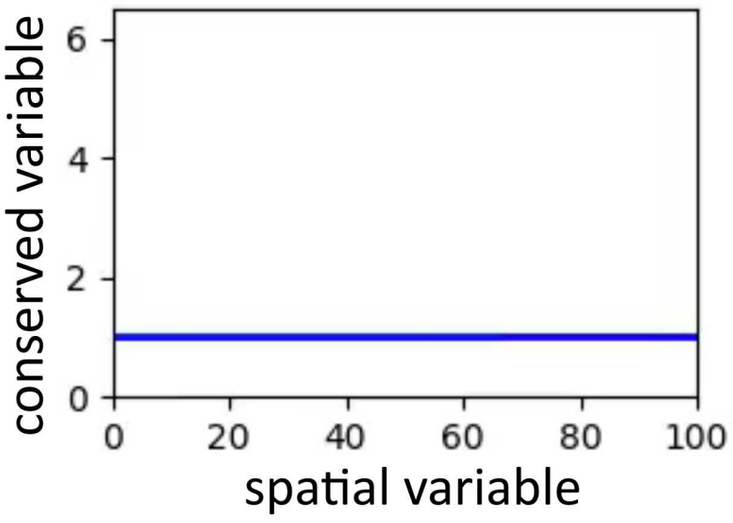
PCA subspace



Autoencoder manifold

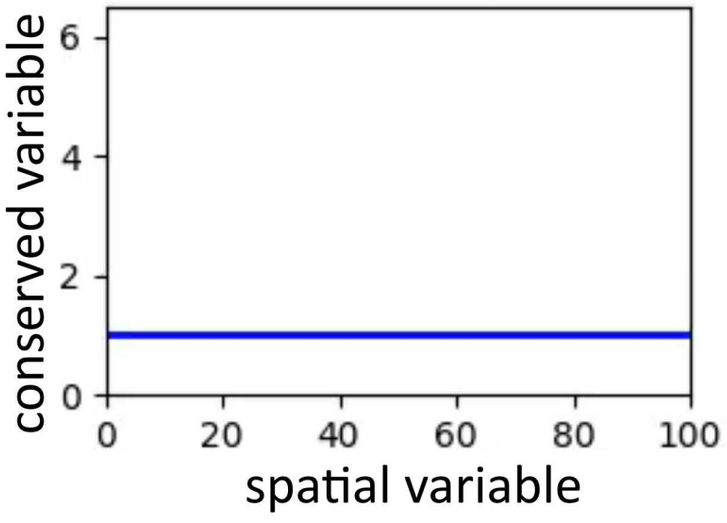


High-fidelity model

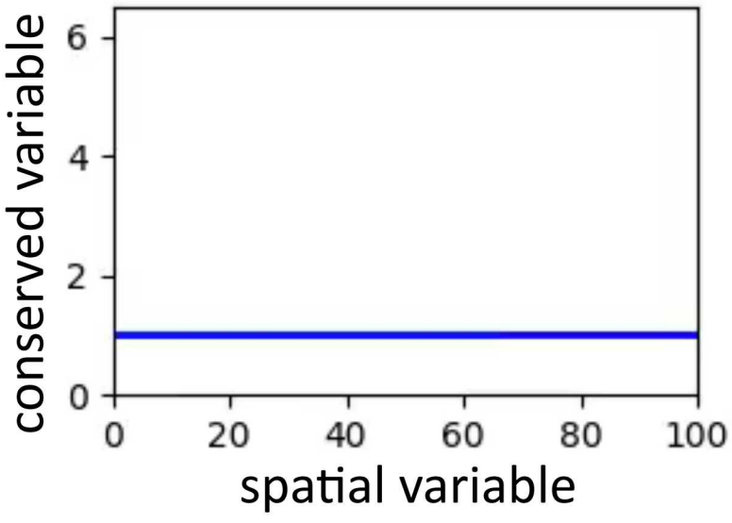


Reduced-order models

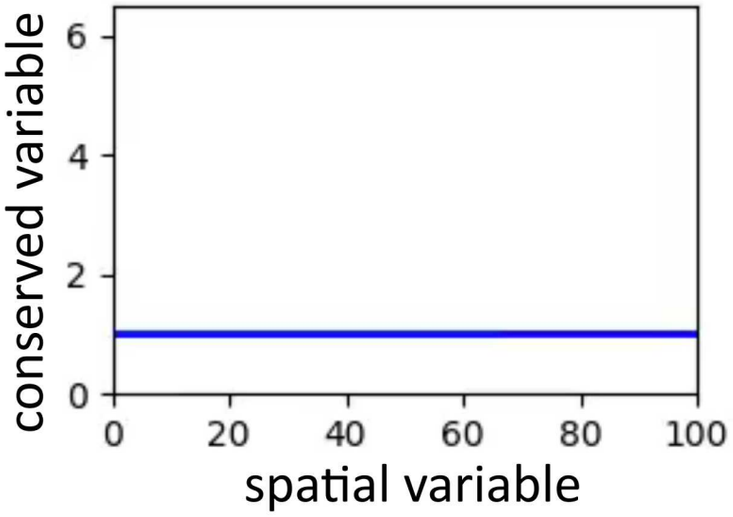
PCA subspace



Autoencoder manifold

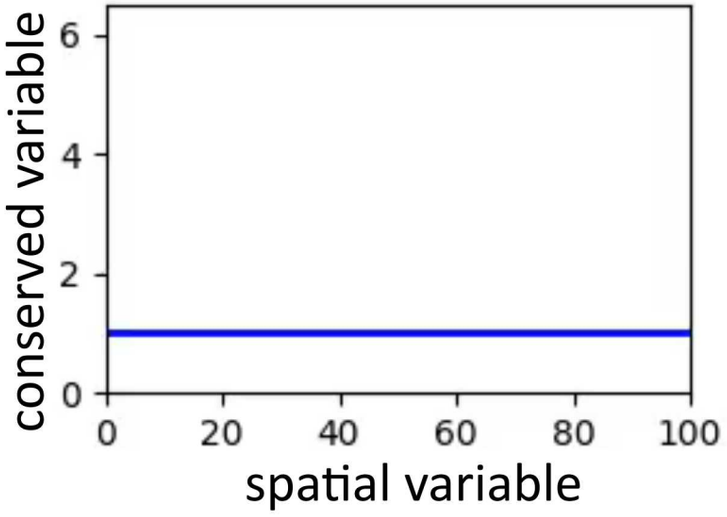


High-fidelity model



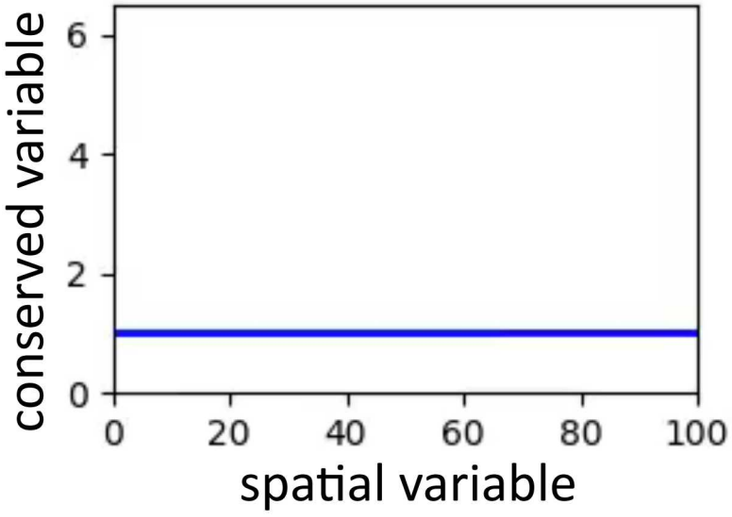
Reduced-order models

PCA subspace



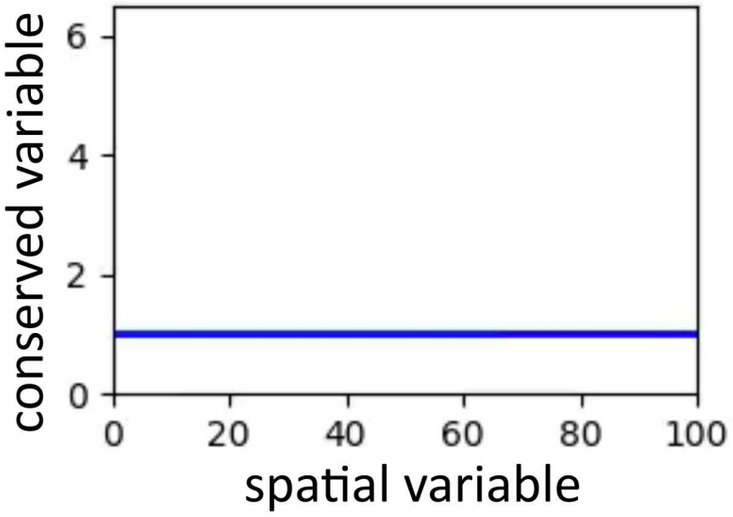
Solution error: 13%
Conservation violation: 16%

Autoencoder manifold



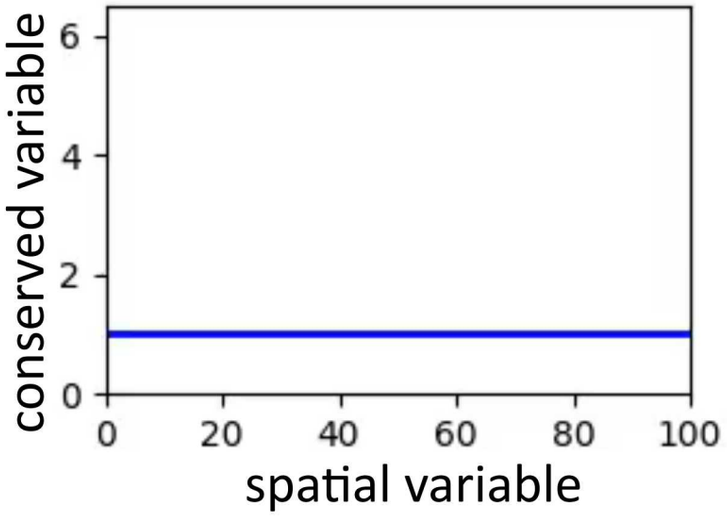
Solution error: 0.5%
Conservation violation: 1%

High-fidelity model



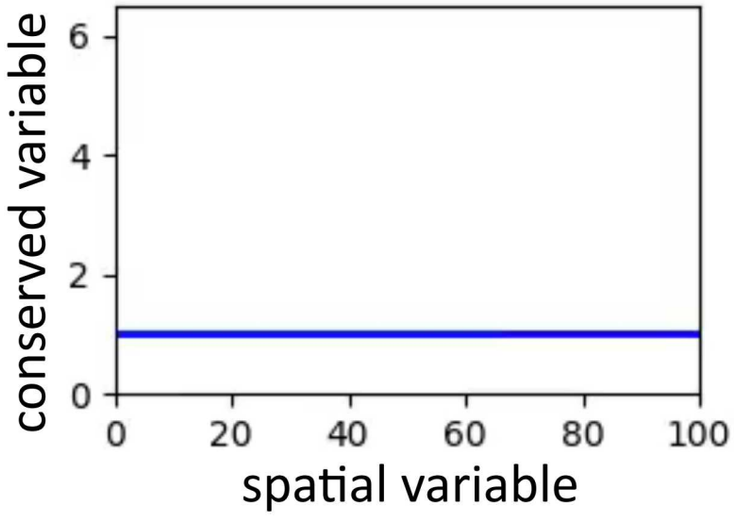
Reduced-order models

PCA subspace



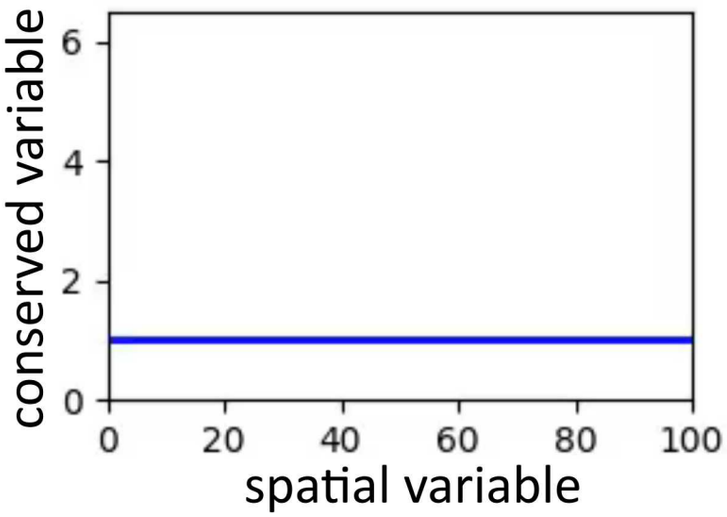
Solution error: 13%
Conservation violation: 16%

Autoencoder manifold

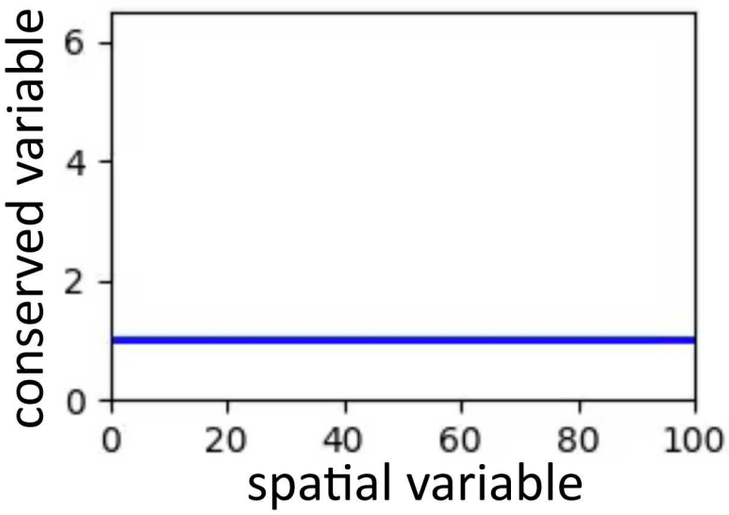


Solution error: 0.5%
Conservation violation: 1%

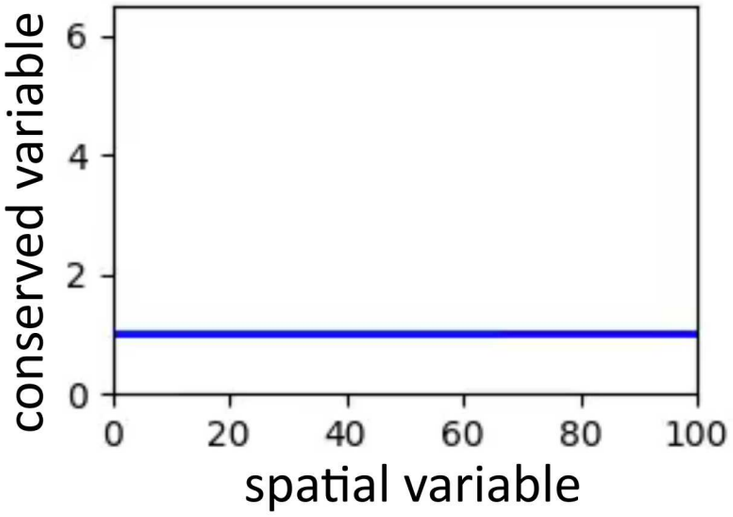
PCA subspace with conservation constraints



Autoencoder manifold with conservation constraints

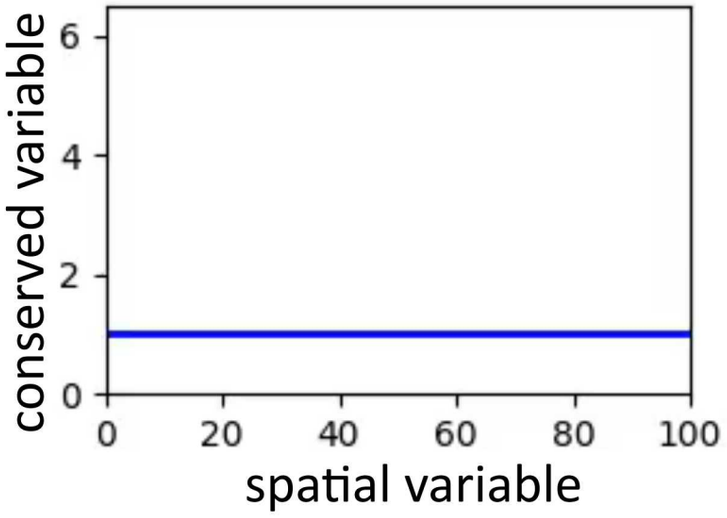


High-fidelity model



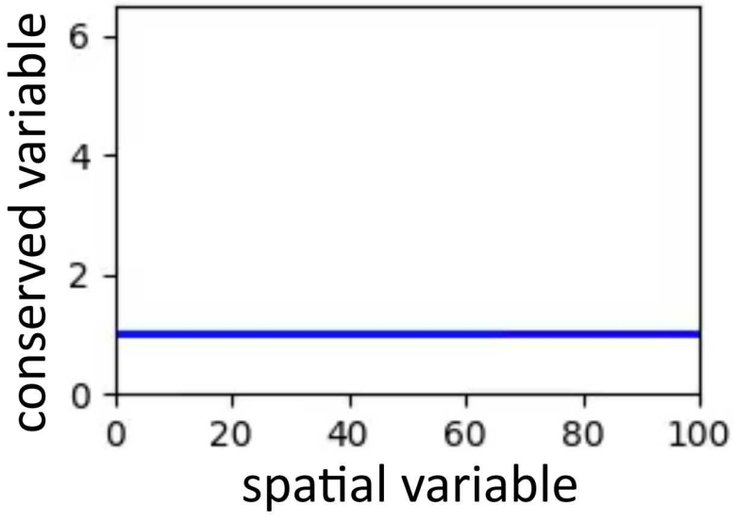
Reduced-order models

PCA subspace



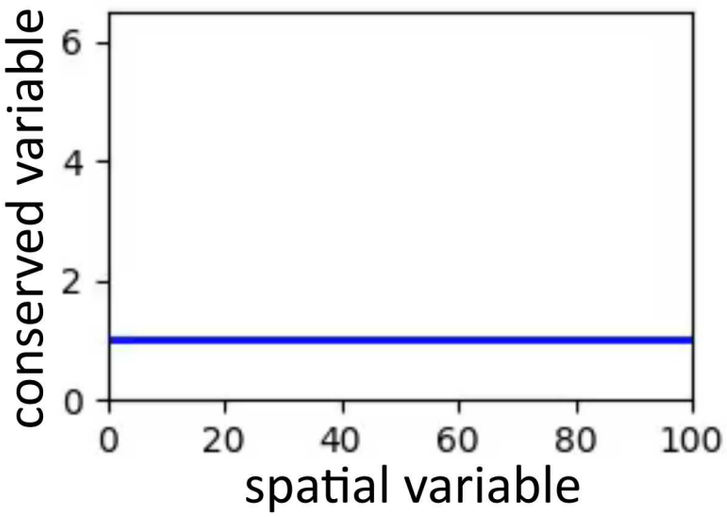
Solution error: 13%
Conservation violation: 16%

Autoencoder manifold

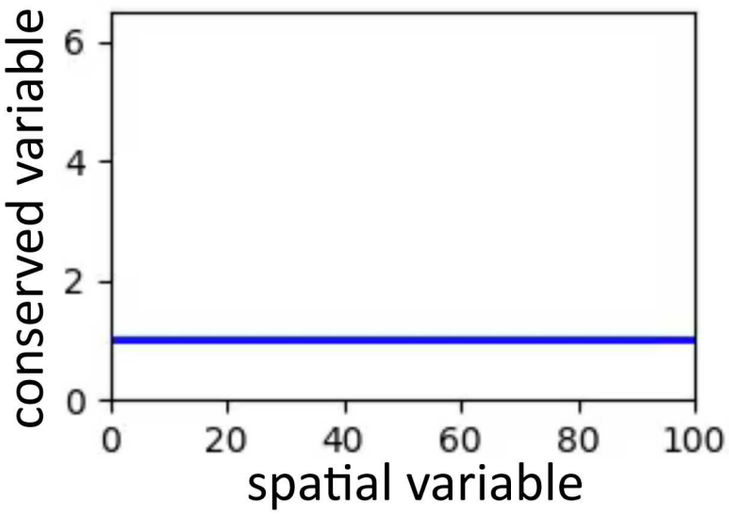


Solution error: 0.5%
Conservation violation: 1%

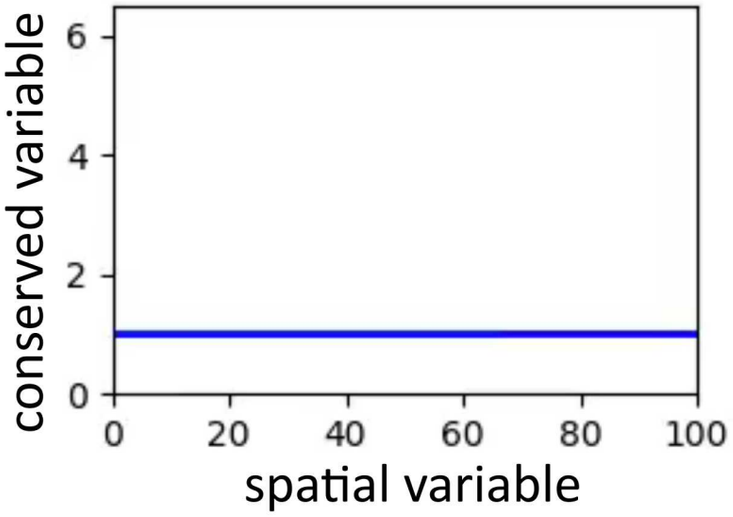
PCA subspace with conservation constraints



Autoencoder manifold with conservation constraints

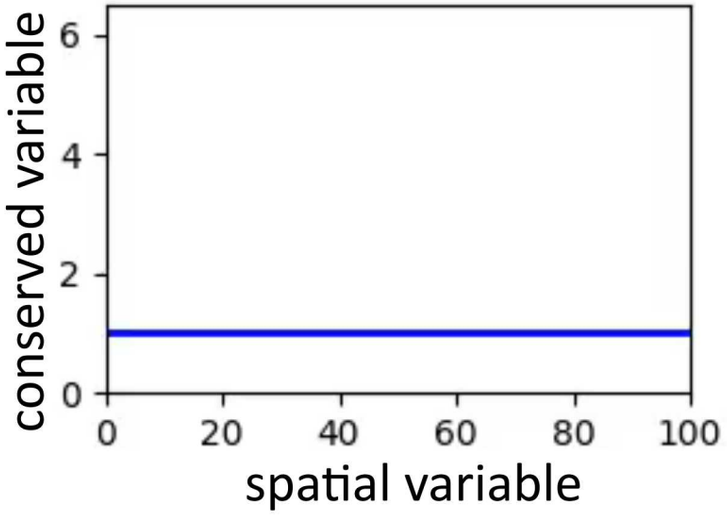


High-fidelity model



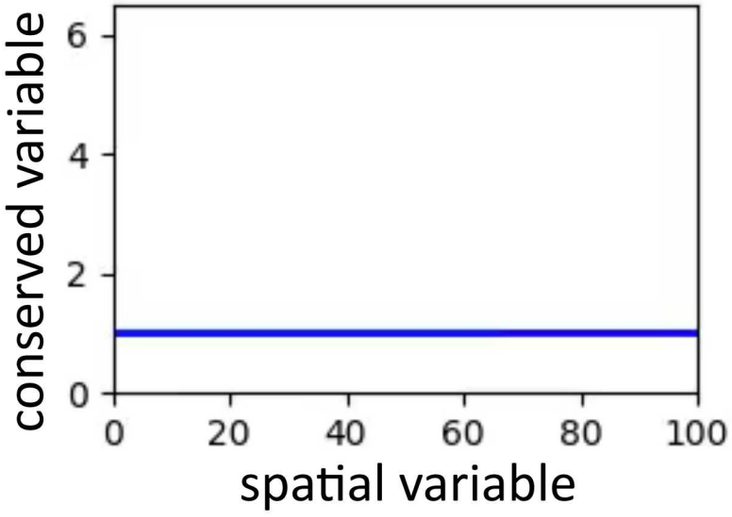
Reduced-order models

PCA subspace



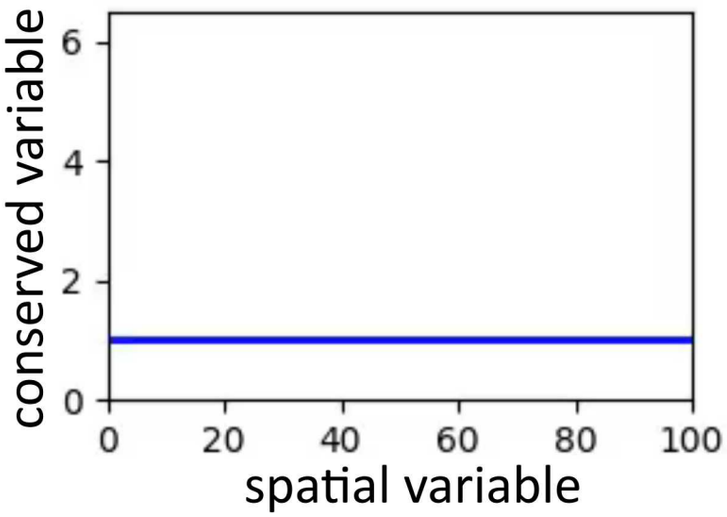
Solution error: 13%
Conservation violation: 16%

Autoencoder manifold



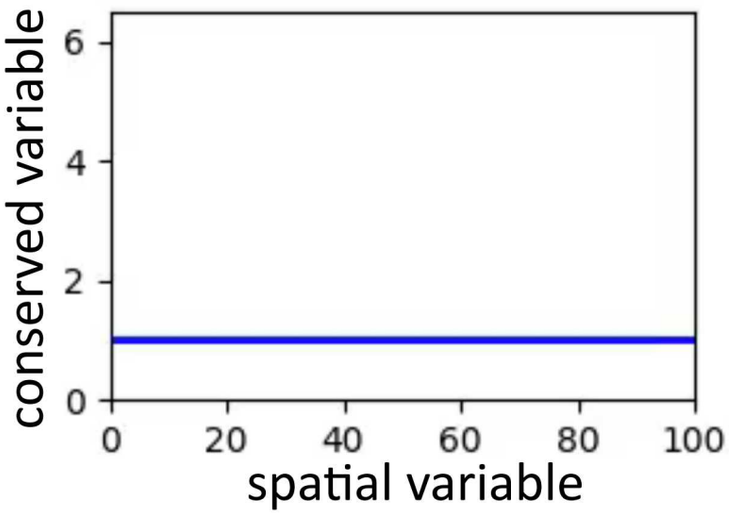
Solution error: 0.5%
Conservation violation: 1%

PCA subspace with conservation constraints



Solution error: 12%
Conservation violation: <0.001%

Autoencoder manifold with conservation constraints



Solution error: 0.2%
Conservation violation: <0.001%

Our research

Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2017]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

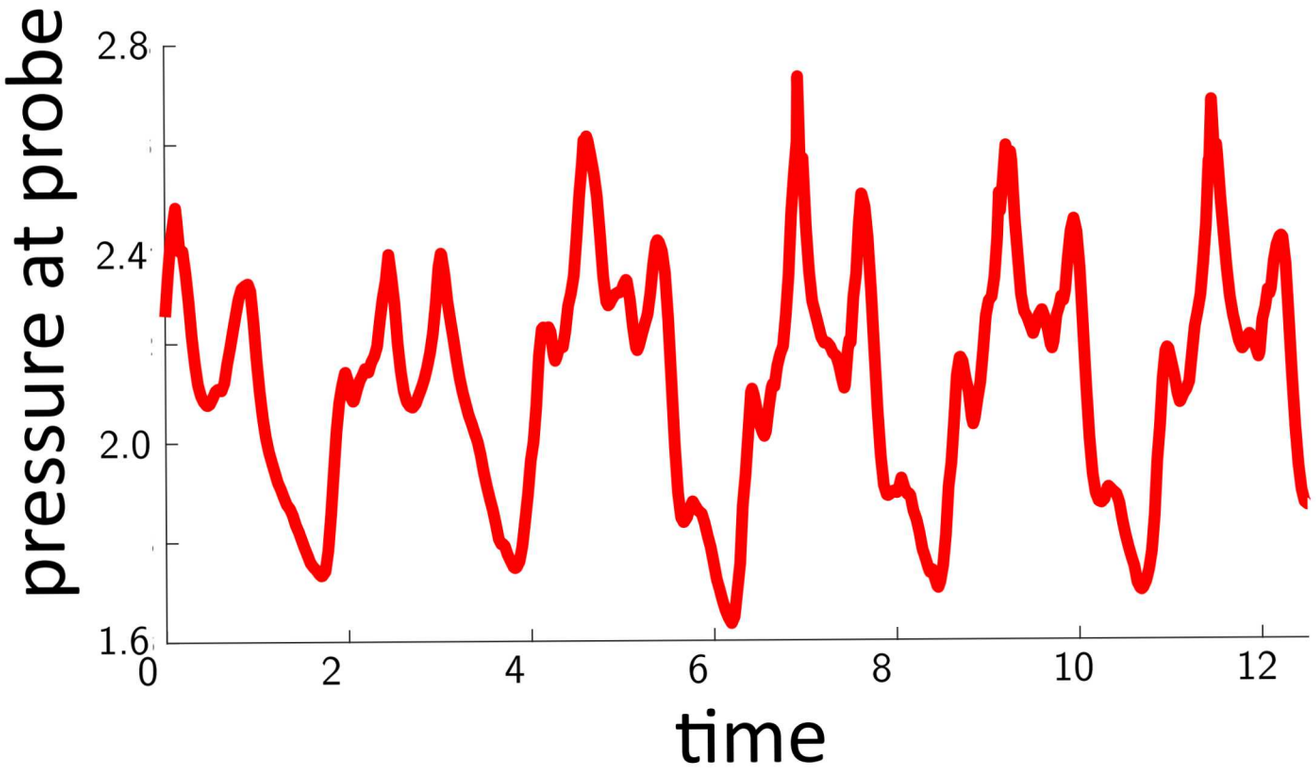
Our research

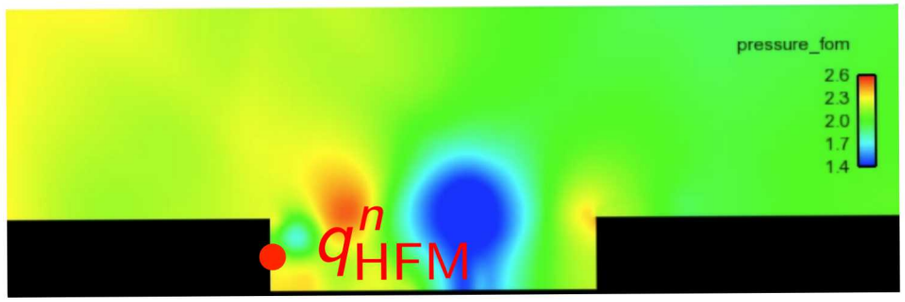
Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2017]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

Key insight

inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}}, \quad n = 1, \ldots, T$

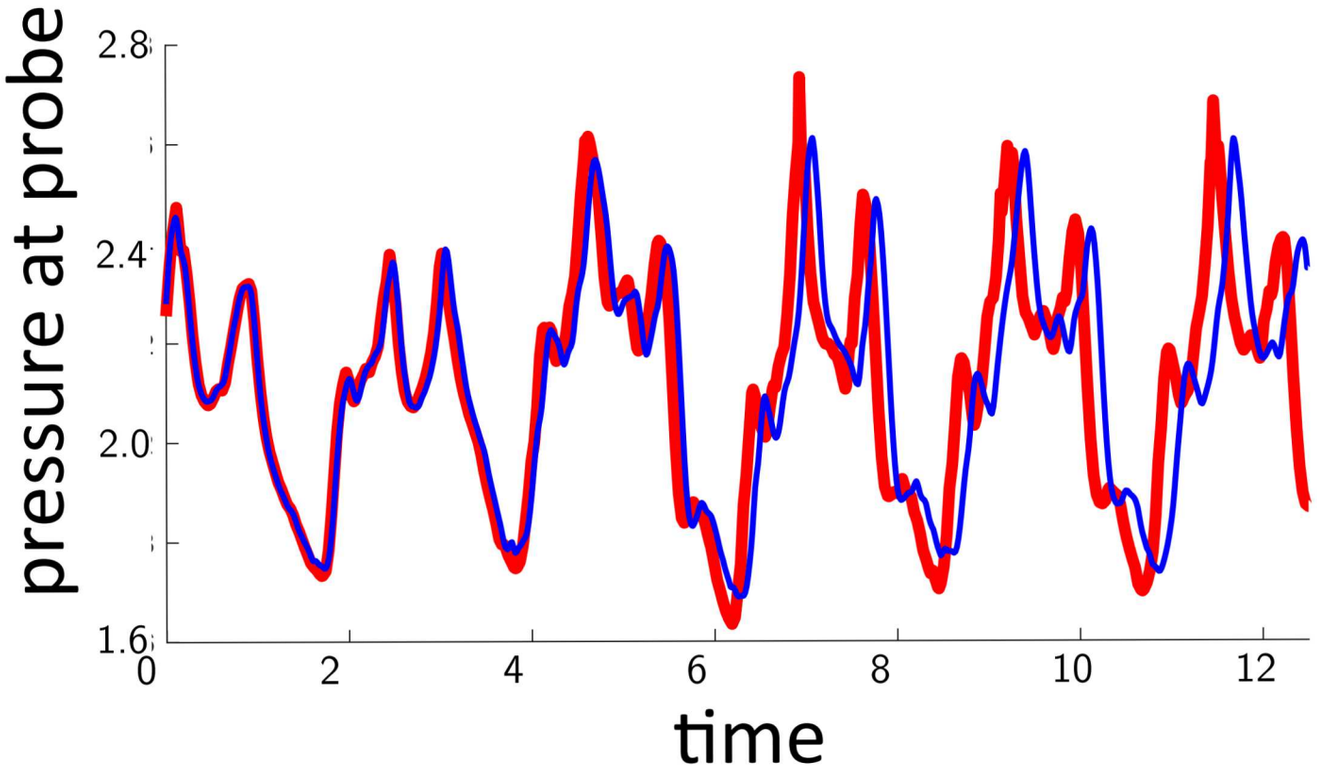


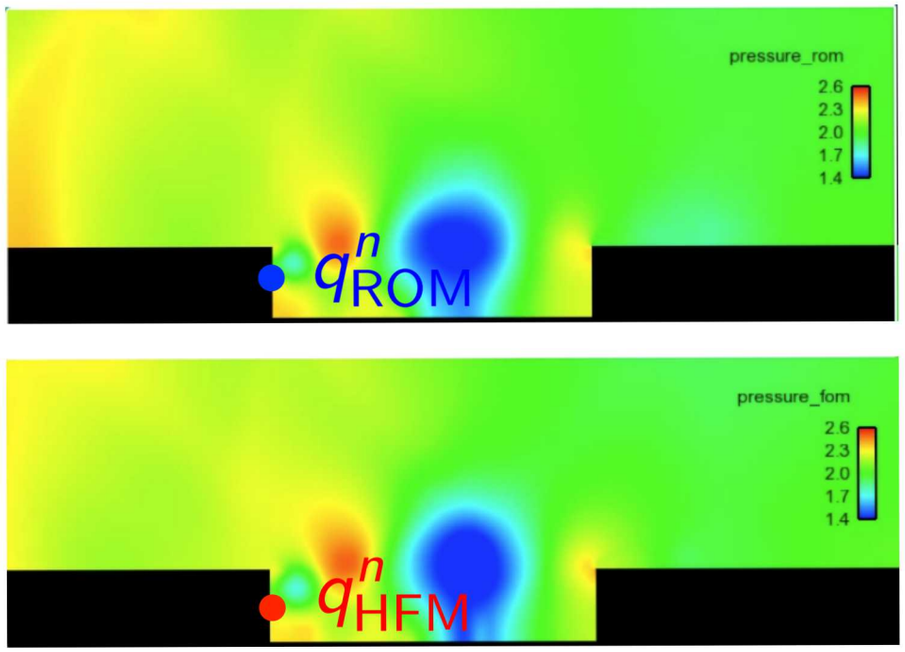


Key insight

inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}}, \quad n = 1, \ldots, T$

inputs μ → *reduced-order model* → *outputs* $q^n_{\text{ROM}}, \quad n = 1, \ldots, T$





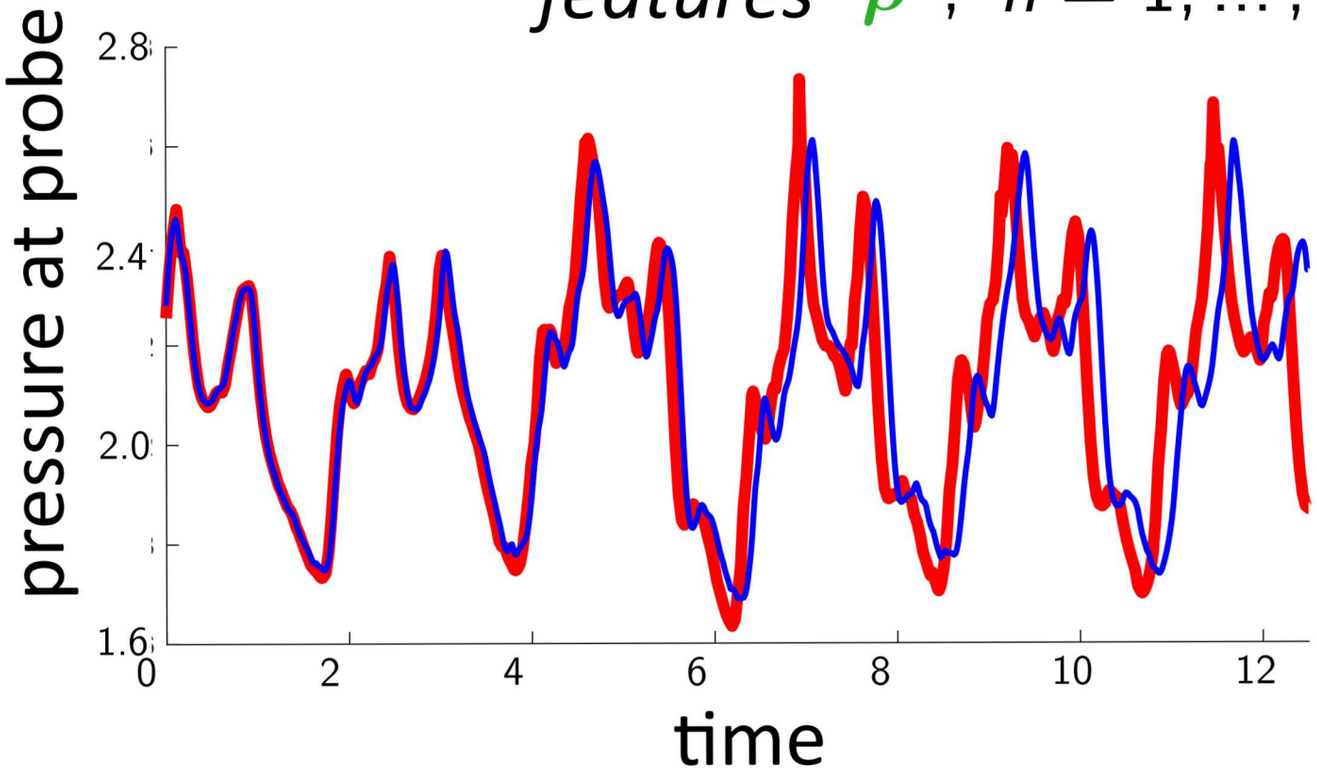
Key insight

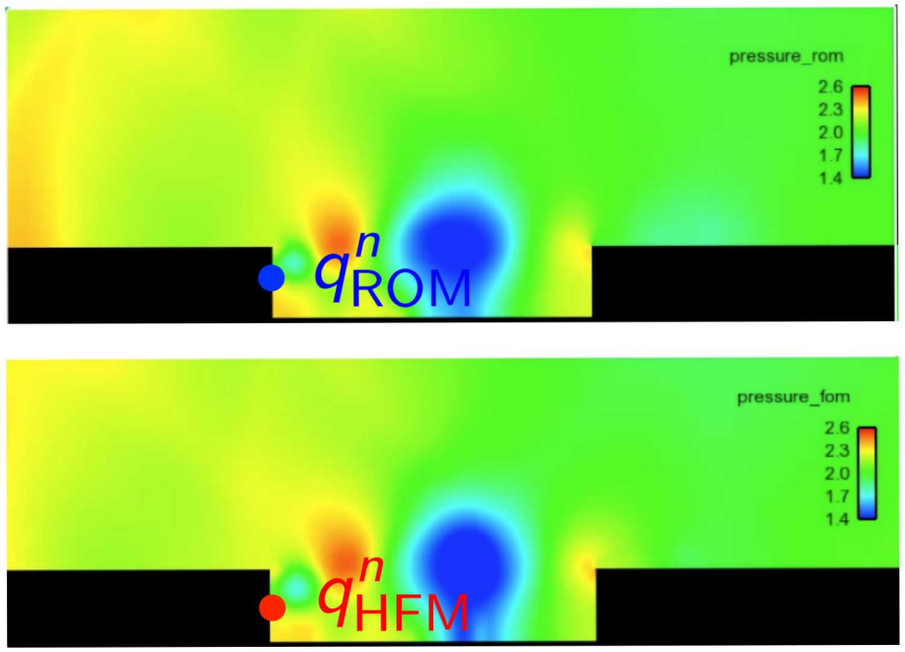
inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}}, \; n = 1, \ldots, T$

inputs μ → *reduced-order model* → *outputs* $q^n_{\text{ROM}}, \; n = 1, \ldots, T$

↓

features $\rho^n, \; n = 1, \ldots, T$





Reduced-order models generate features ρ^n ***that may inform its error***

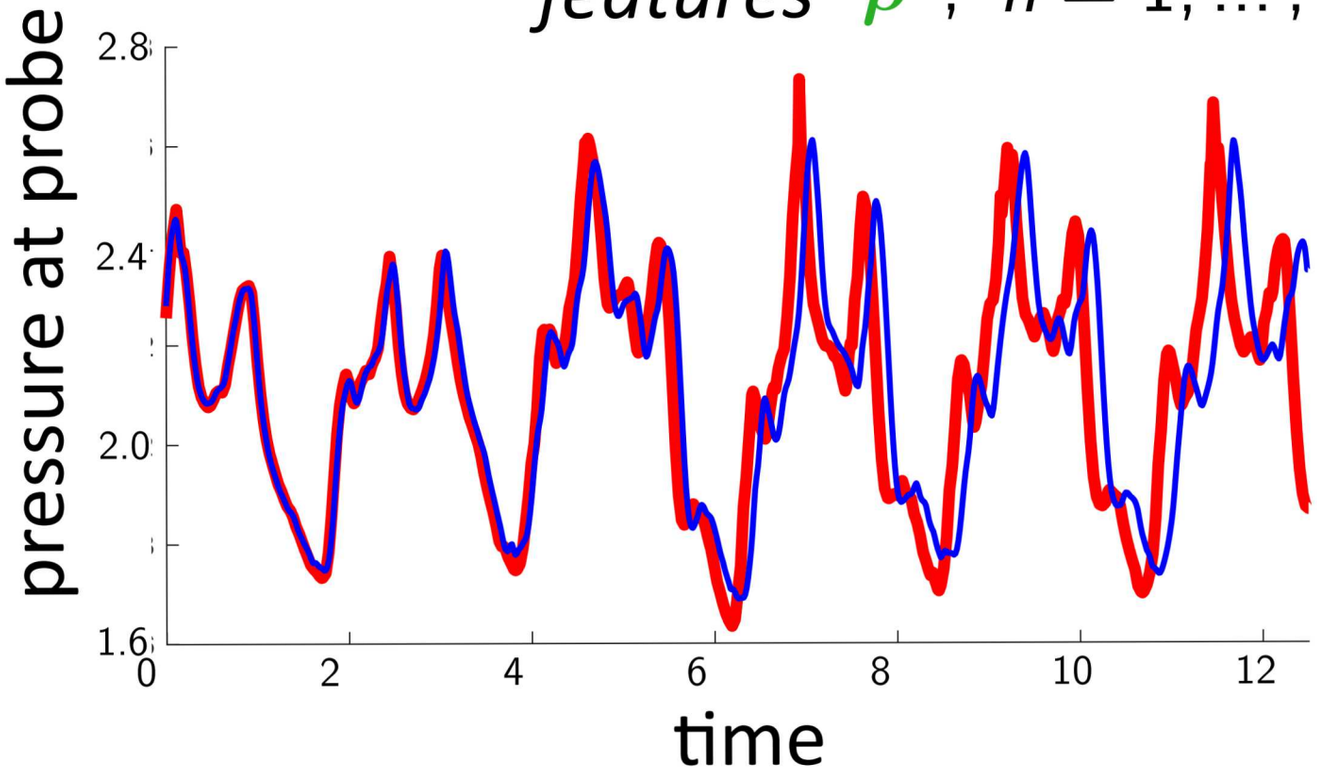
Key insight

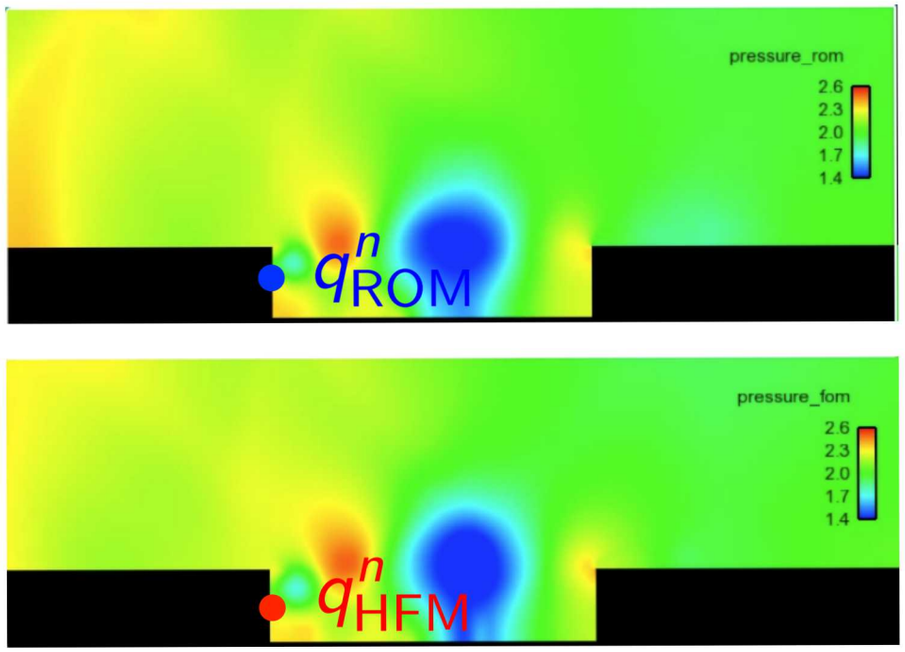
inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}}, \; n = 1, \ldots, T$

inputs μ → *reduced-order model* → *outputs* $q^n_{\text{ROM}}, \; n = 1, \ldots, T$

↓

features $\rho^n, \; n = 1, \ldots, T$



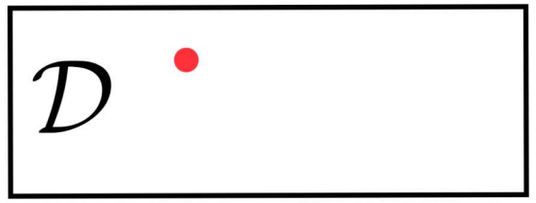


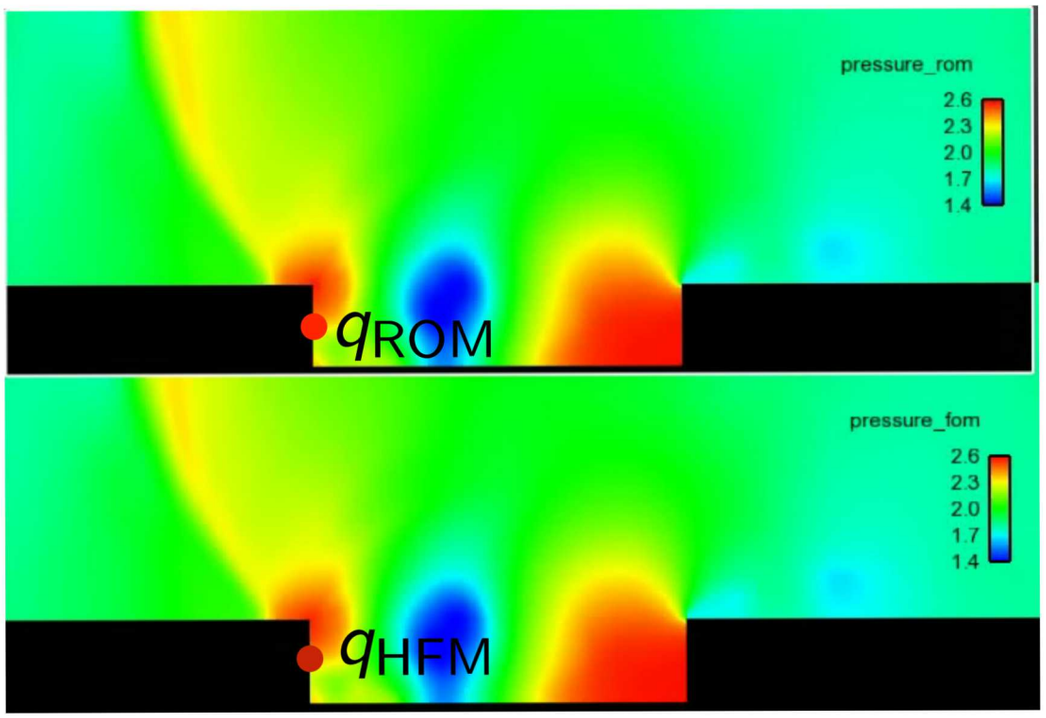
Reduced-order models generate features ρ^n ***that may inform its error***

Idea*: regression model that predicts error* $q^n_{\text{HFM}} - q^n_{\text{ROM}}$ *from features* ρ^n

Training and machine learning

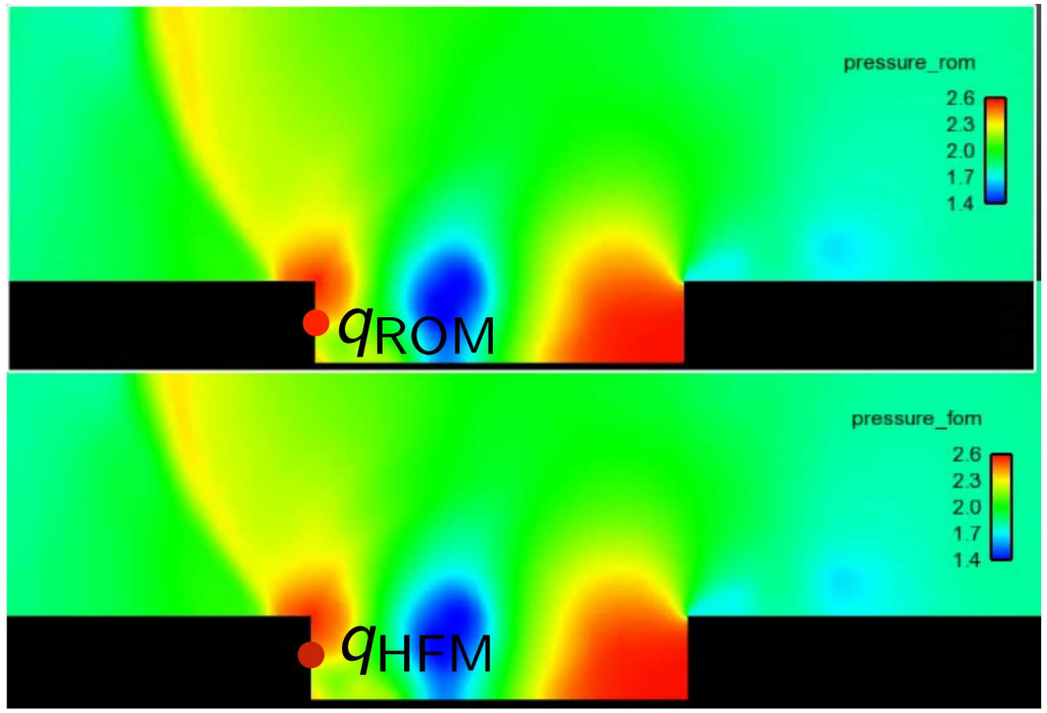
1. *Training:* Solve high-fidelity and reduced-order models for $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict reduced-order-model error for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

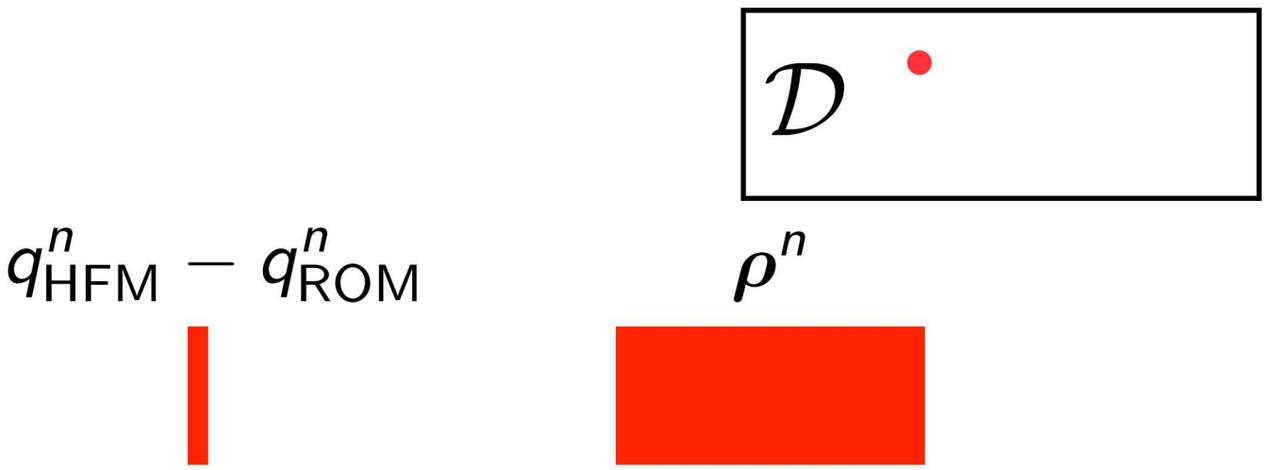




Training and machine learning

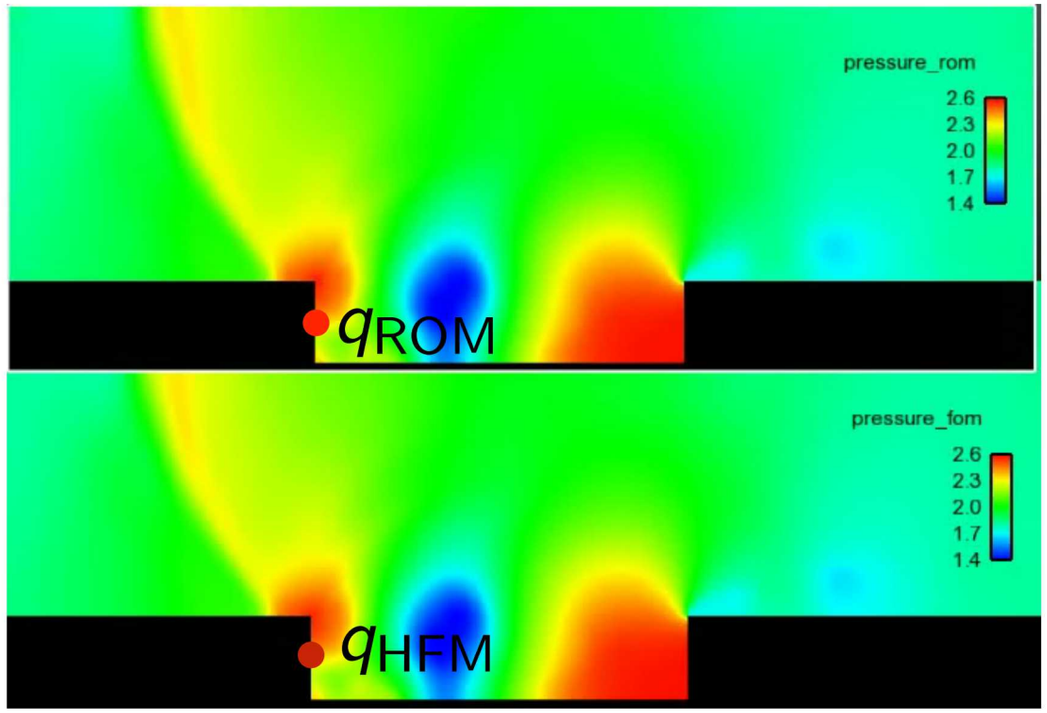
1. *Training:* Solve high-fidelity and reduced-order models for $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict reduced-order-model error for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

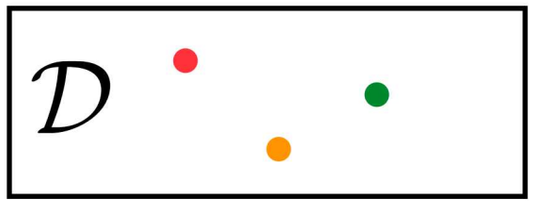


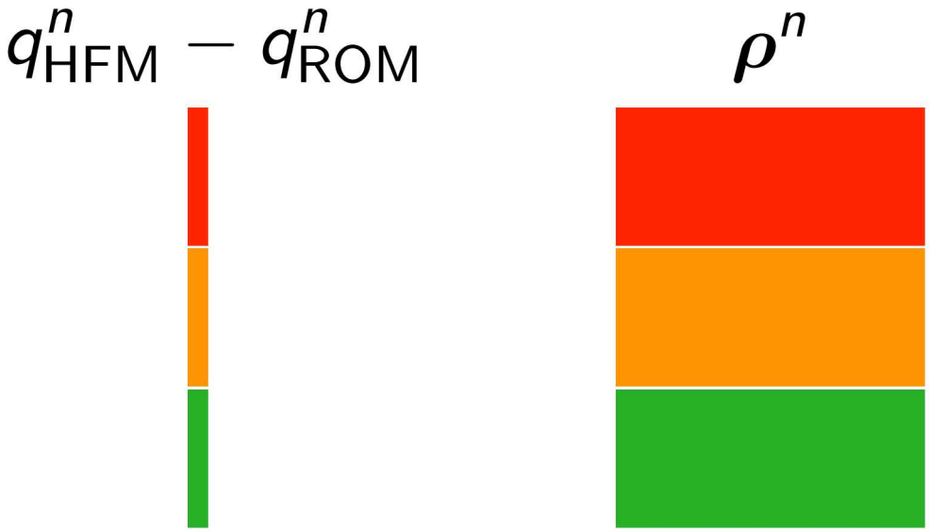


Training and machine learning

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

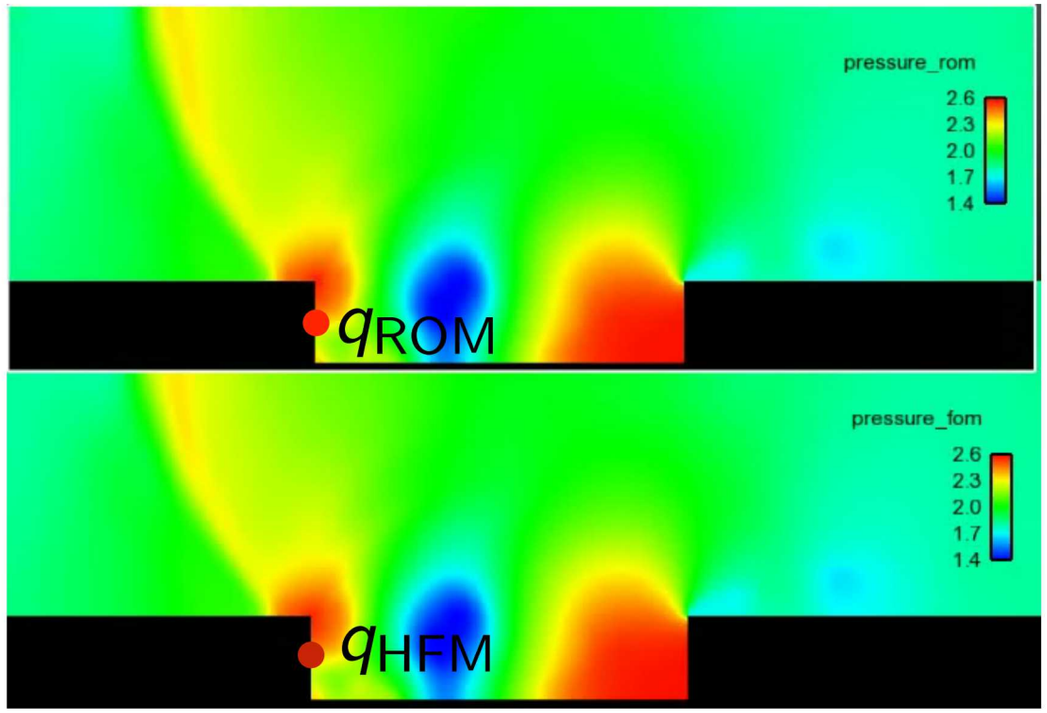


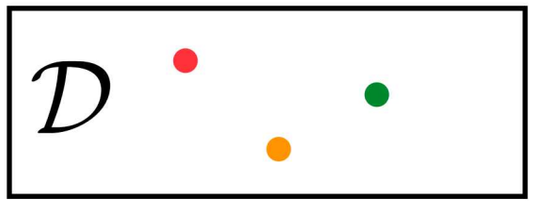


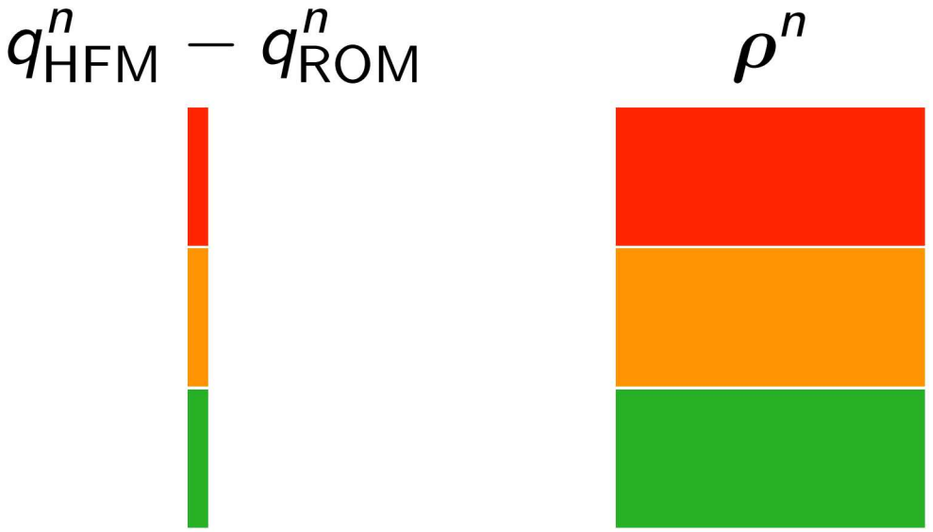


Training and machine learning

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





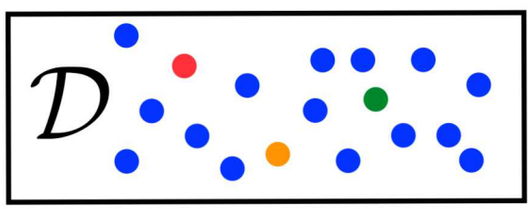


features ρ^n → *regression model* → *error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$

- *Regression methods*: Gaussian process, random forest, SVM, neural nets

Regression model for the error

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



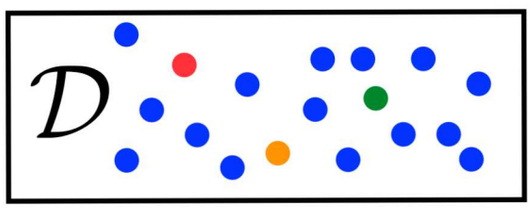
inputs μ → *reduced-order model* → *outputs* q^n_{ROM}, $n = 1, \dots, T$

↓

features ρ^n, $n = 1, \dots, N$

Regression model for the error

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



inputs μ → *reduced-order model* → *outputs* q^n_{ROM}, $n = 1, \ldots, T$

↓

features ρ^n, $n = 1, \ldots, N$

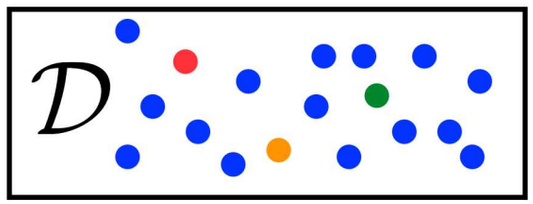
↓

regression model → *stochastic error prediction*

$\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$, $n = 1, \ldots, T$

Regression model for the error

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



inputs $\boldsymbol{\mu}$ → *reduced-order model* → *outputs* q^n_{ROM}, $n = 1, \ldots, T$

↓

features $\boldsymbol{\rho}^n$, $n = 1, \ldots, N$

↓

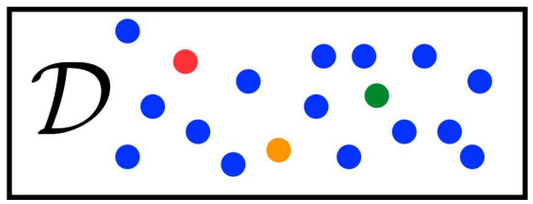
regression model → *stochastic error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$, $n = 1, \ldots, T$

$$q^n_{\text{HFM}} \approx q^n_{\text{ROM}} + \hat{\delta}^n$$

+ *Statistical model of high-fidelity-model output*

Regression model for the error

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



inputs μ → *reduced-order model* → *outputs* q^n_{ROM}, $n = 1, \ldots, T$

↓

features ρ^n, $n = 1, \ldots, N$

↓

regression model → *stochastic error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$, $n = 1, \ldots, T$

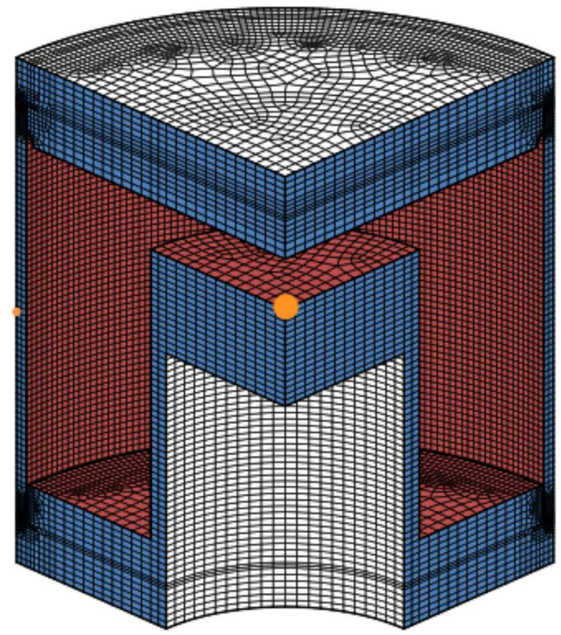
$$q^n_{\text{HFM}} \approx q^n_{\text{ROM}} + \hat{\delta}^n$$

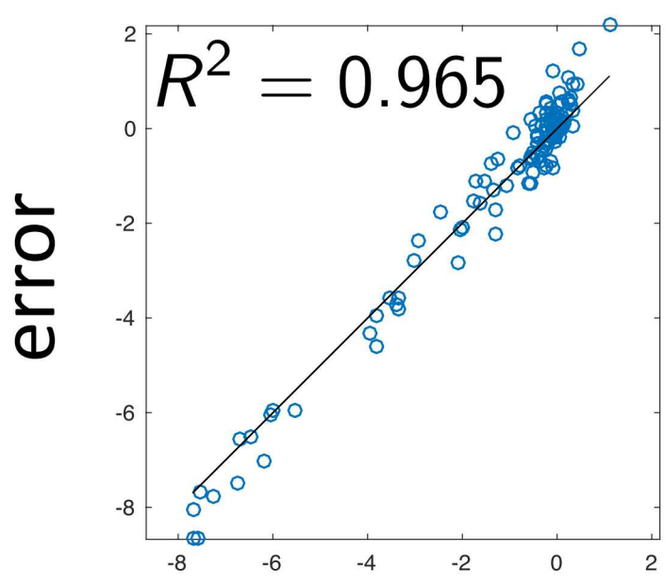
\+ *Statistical model of high-fidelity-model output*

Physics-based feature engineering to determine ρ^n

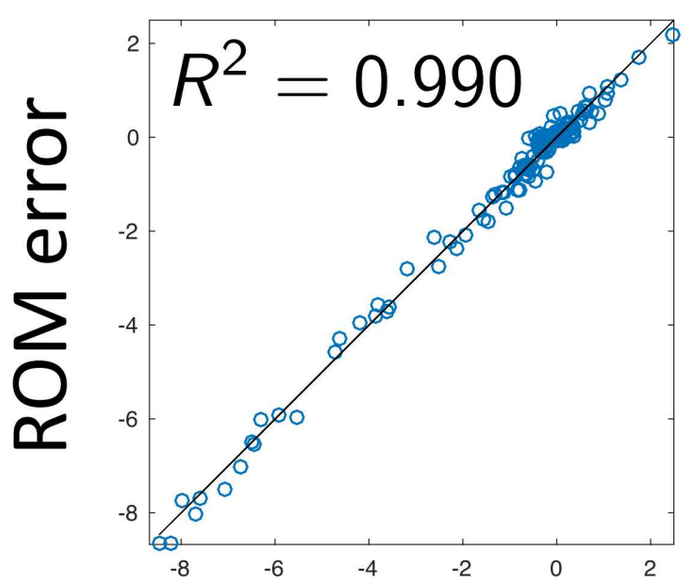
Predictive Capability Assessment Project [Freno, C., 2018]

- *high-fidelity model dimension:* 2.8×10^5
- *reduced-order model dimension:* 6
- *inputs* μ*:* elastic modulus, Poisson ratio
- *error:* error in *y*-displacement at point
- *50 features* ρ*:* residual approx $(\mathbf{P}\boldsymbol{\Phi}_\mathbf{r})^{+}\mathbf{P}\mathbf{r}^n$, inputs μ
- *regression:* random forest, SVM, *k*-NN

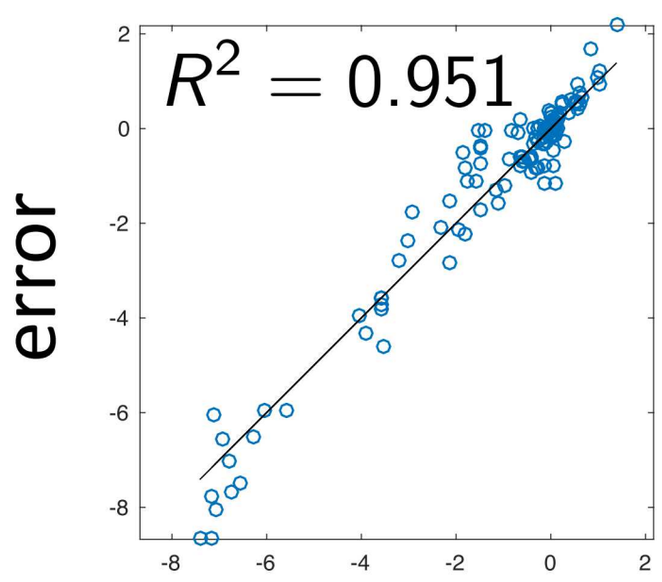




random forest
error prediction



support vector machine
error prediction



k-NN
error prediction

- *ML methods yield low-variance error predictions*

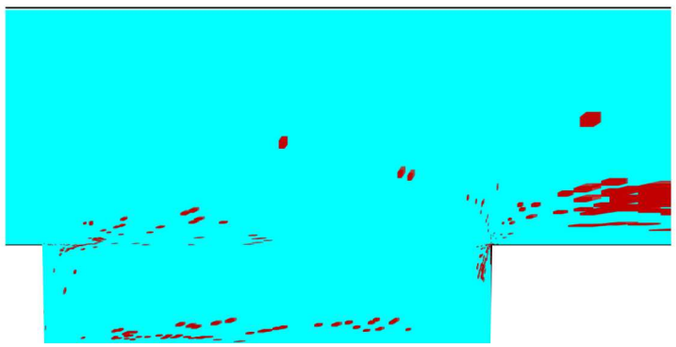
Our research

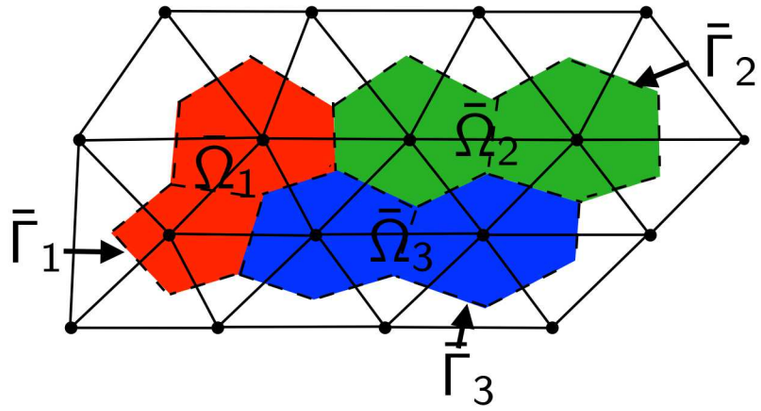
Accurate, low-cost, structure-preserving, generalizable, certified nonlinear model reduction

- *accuracy*: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection
  [C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure preservation* [C., Tuminaro, Boggs, 2015; Peng and C., 2017; C., Choi, Sargsyan, 2018]
- *generalization*: projection onto nonlinear manifolds [Lee, C., 2018]
- *generalization*: *h*-adaptivity [C., 2015; Etter and C., 2019]
- *certification*: machine learning error models
  [Drohmann and C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2019; Pagani, Manzoni, C., 2019]

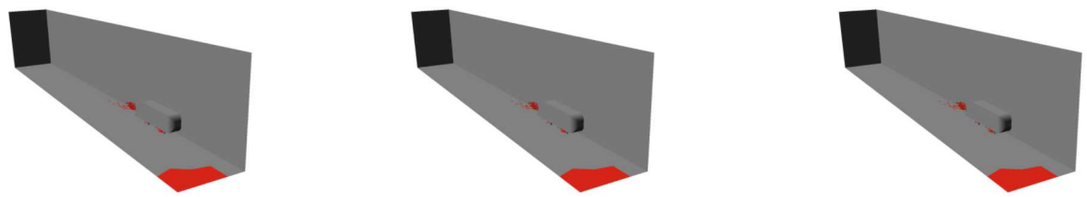
Questions?

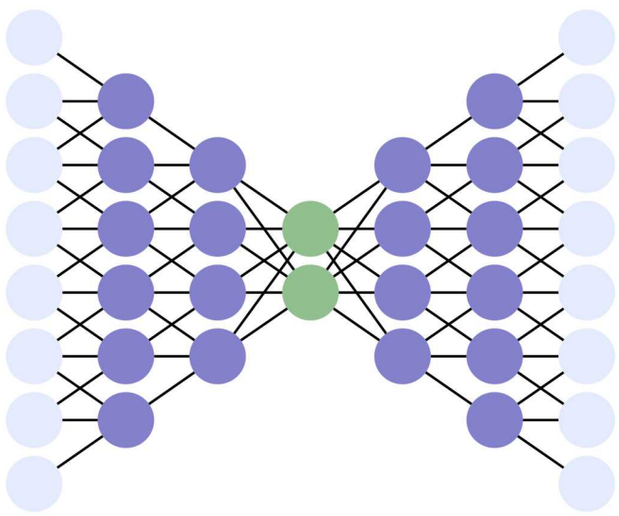
$$x =$$

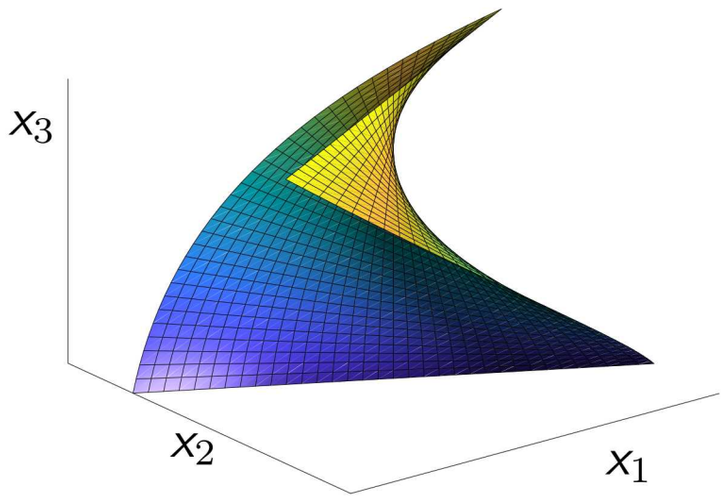


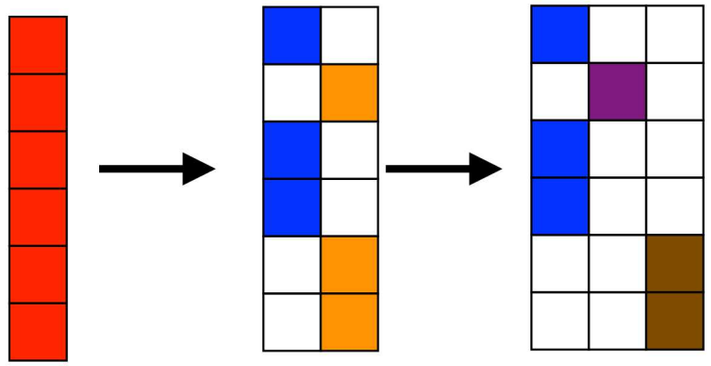


t^1 t^5 t^9









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-NA0003525