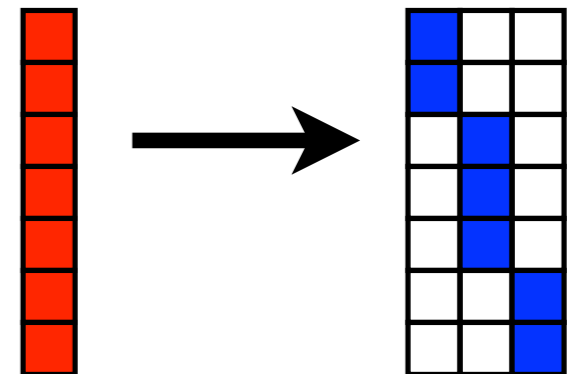
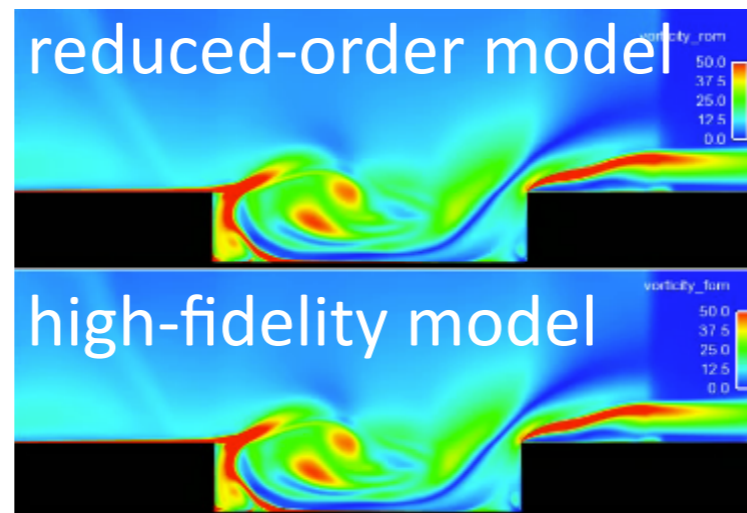
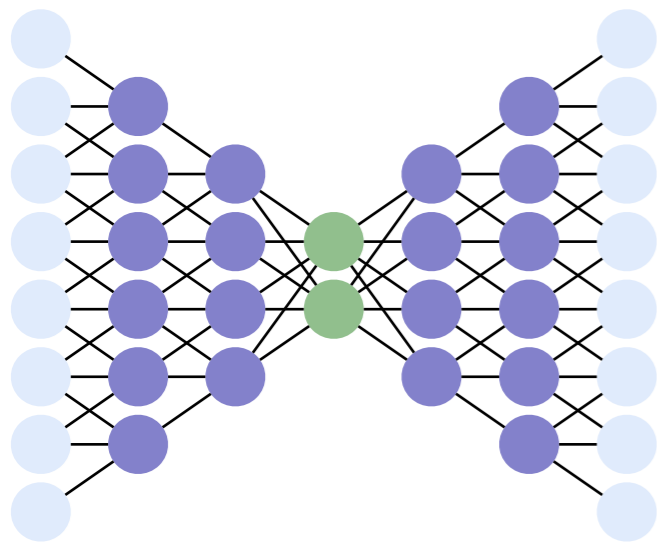


# Nonlinear model reduction

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



**Kevin Carlberg**

*Sandia National Laboratories*

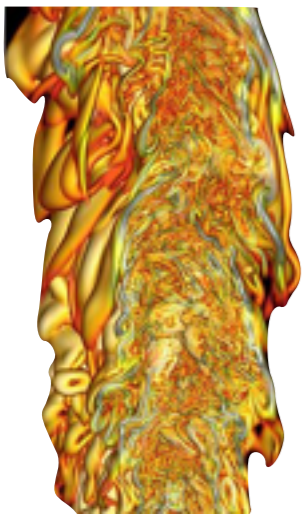
AI for Engineering Summer School

Toronto, Canada

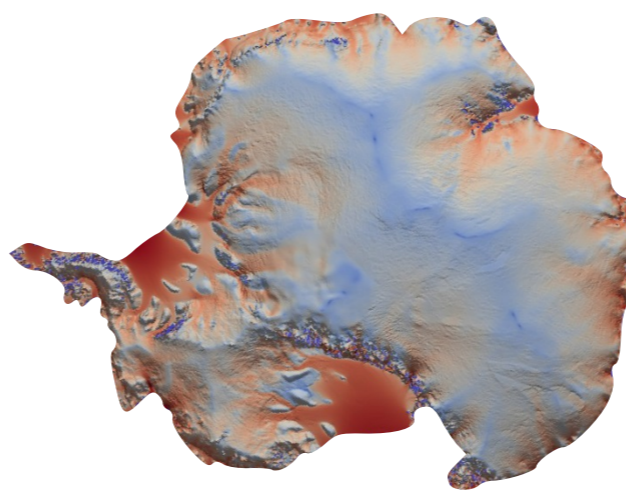
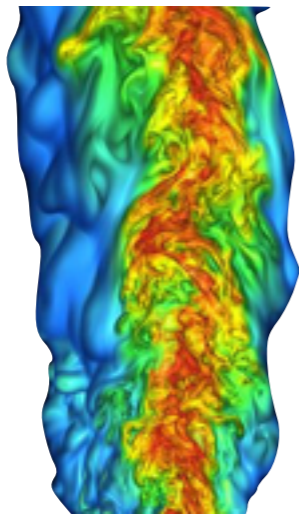
August 21, 2019

# High-fidelity simulation

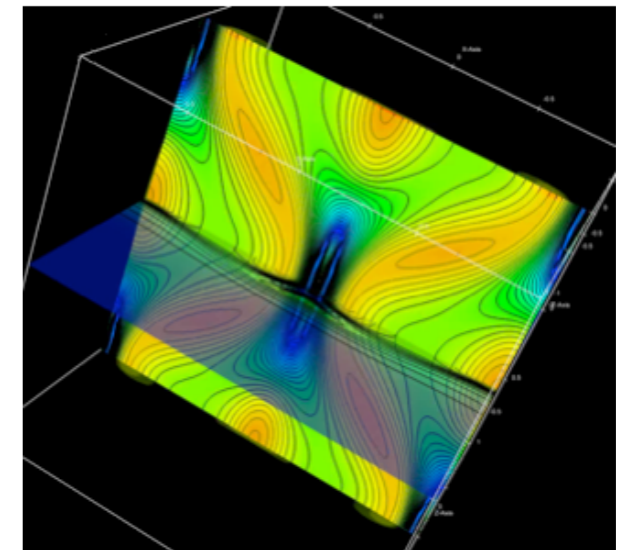
- + Indispensable across science and engineering
- *High fidelity*: extreme-scale computational models



*Turbulent reacting flows*  
courtesy J. Chen, Sandia



*Antarctic ice sheet modeling*  
courtesy R. Tuminaro, Sandia



*Magnetohydrodynamics*  
courtesy J. Shadid, Sandia

## computational barrier

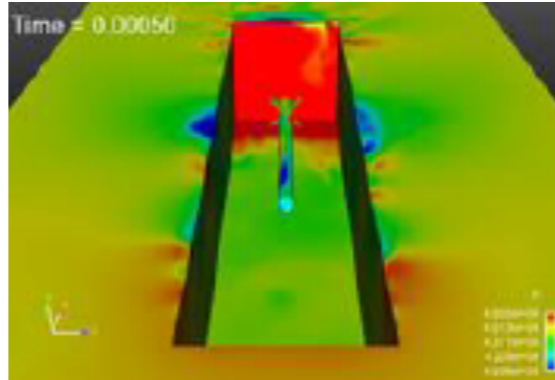
# Time-critical problems

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

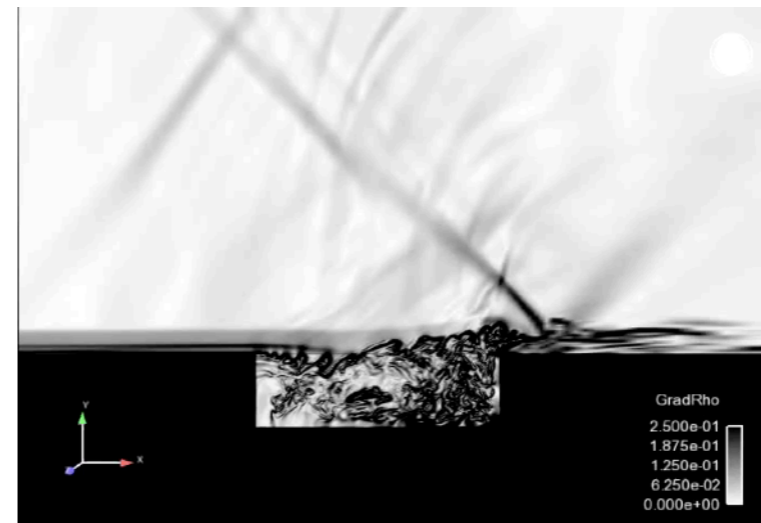
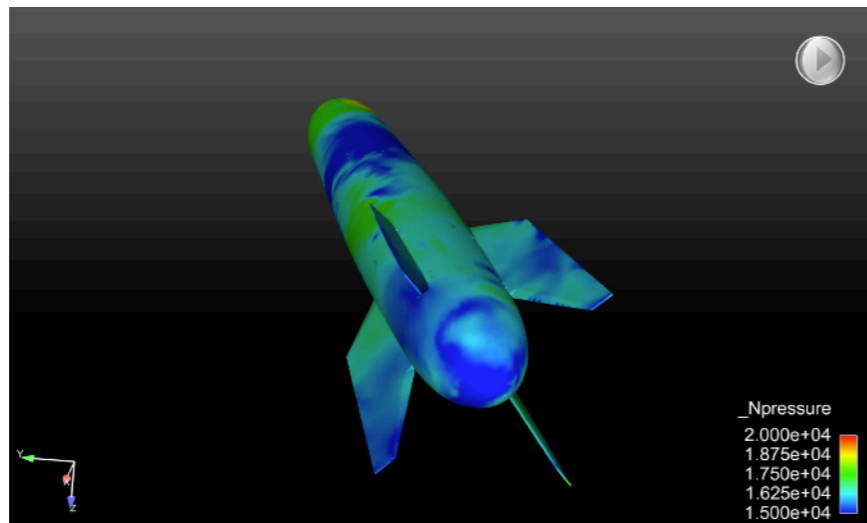


Sandia  
National  
Laboratories

# 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

**computational barrier**

## Time-critical problems

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

# Computational barrier at NASA

**The New York Times**

**Geniuses Wanted: NASA Challenges  
Coders to Speed Up Its Supercomputer**



*“Despite tremendous progress made in the past few decades, CFD tools are **too slow** for simulation of complex geometry flows... [taking] from **thousands** to **millions** of computational core-hours.”*

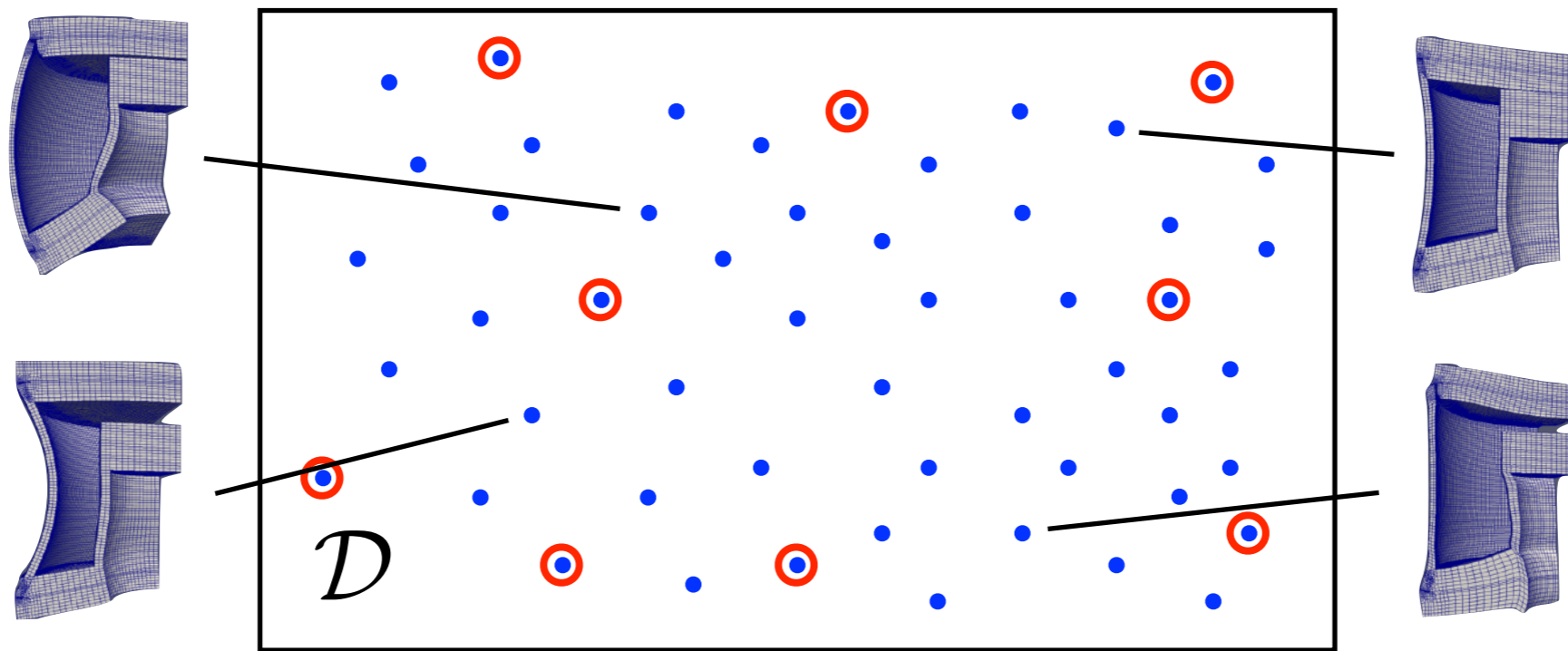
*“To enable high-fidelity CFD for **multi-disciplinary analysis and design**, the speed of computation must be increased by orders of magnitude.”*

*“The desired outcome is any approach that can **accelerate calculations by a factor of 10x to 1000x.**”*

# Approach: exploit simulation data

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

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



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

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

# Model reduction criteria

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

# 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

## ***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; Parish and C., 2019]

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

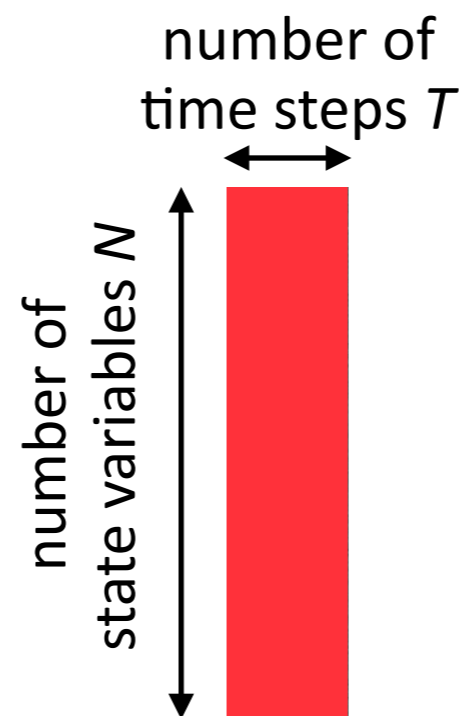
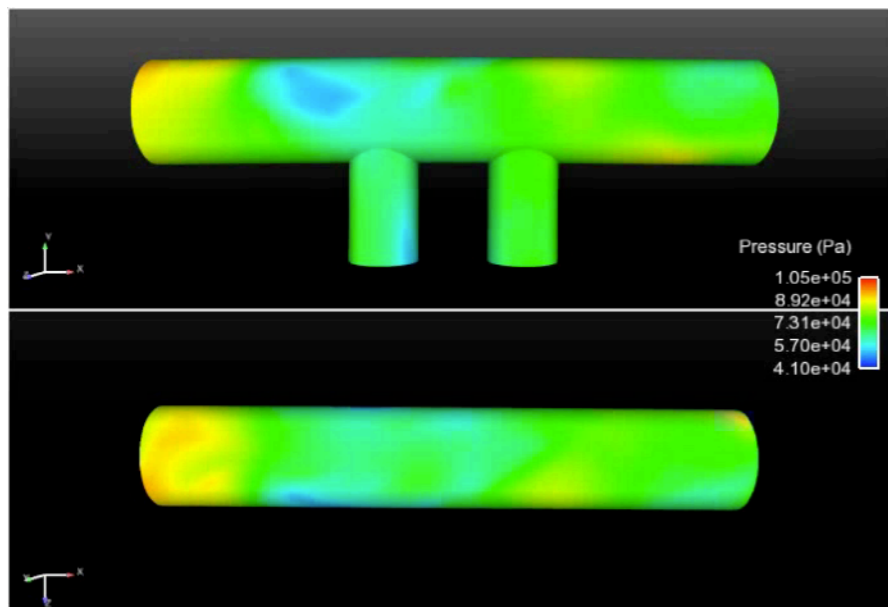
- ***accuracy***: LSPG projection [C., Bou-Mosleh, Farhat, 2011; C., Barone, Antil, 2017]
- *low cost*: sample mesh [C., Farhat, Cortial, Amsallem, 2013]
- *low cost*: space–time LSPG projection  
[C., Ray, van Bloemen Waanders, 2015; C., Brencher, Haasdonk, Barth, 2017; Choi and C., 2019]
- *structure 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; Parish and C., 2019]

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

# Training simulations: state tensor

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

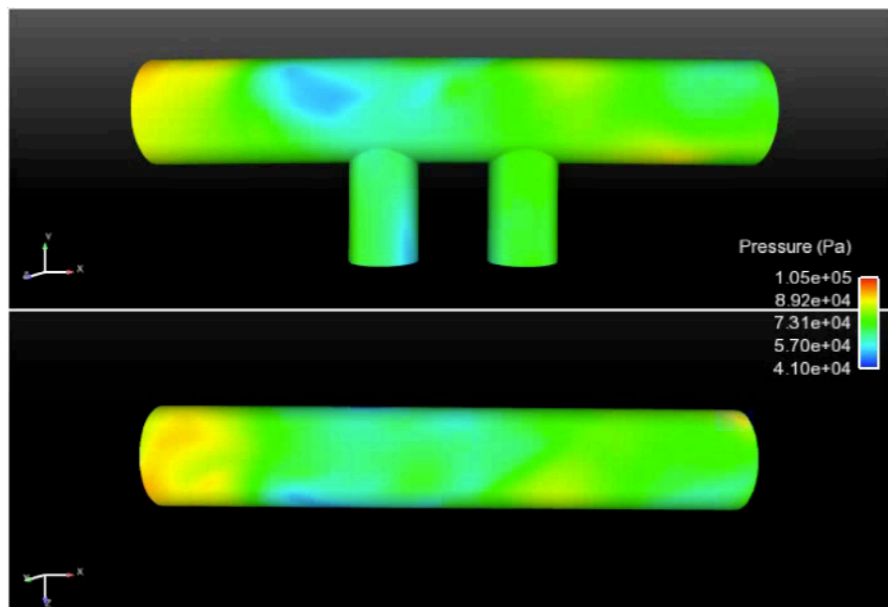
1. *Training*: Solve ODE for  $\mu \in \mathcal{D}_{\text{training}}$  and collect simulation data
2. *Machine learning*: Identify structure in data
3. *Reduction*: Reduce the cost of solving ODE for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



# Training simulations: state tensor

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

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



$\mathcal{X} =$

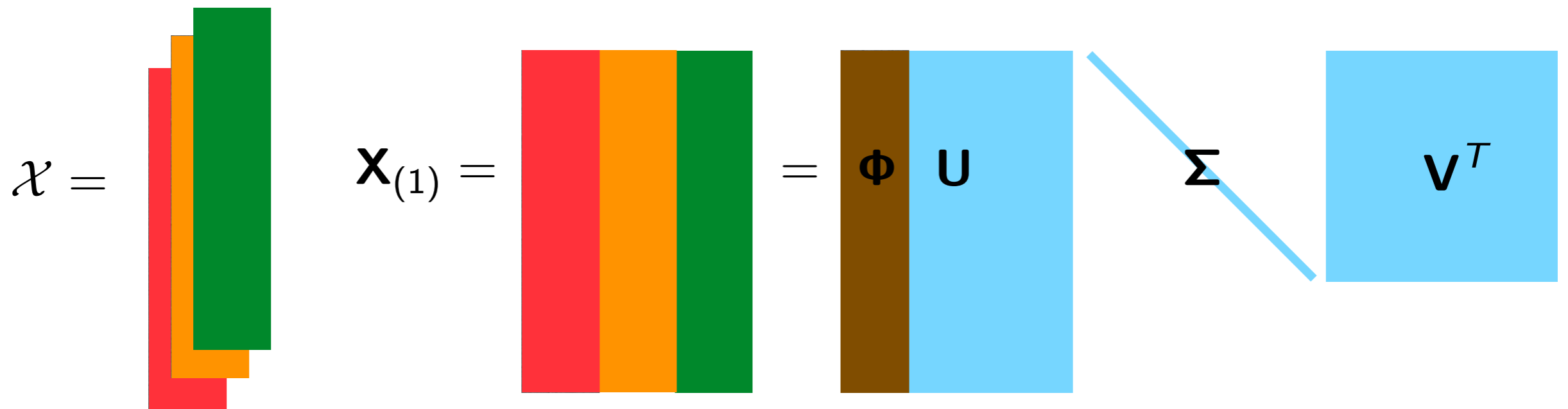


# Tensor decomposition

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

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

*Compute dominant left singular vectors of mode-1 unfolding*

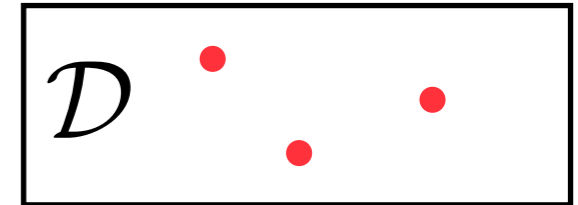


*$\Phi$  columns are principal components of the spatial simulation data*

***How to integrate these data with the computational model?***

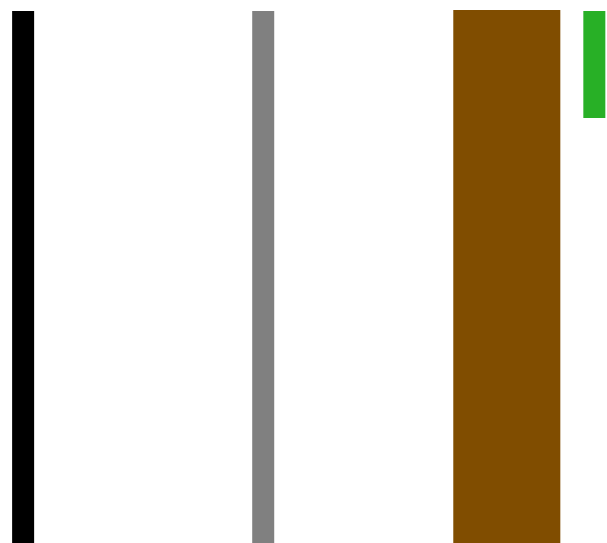
# Previous state of the art: POD–Galerkin

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

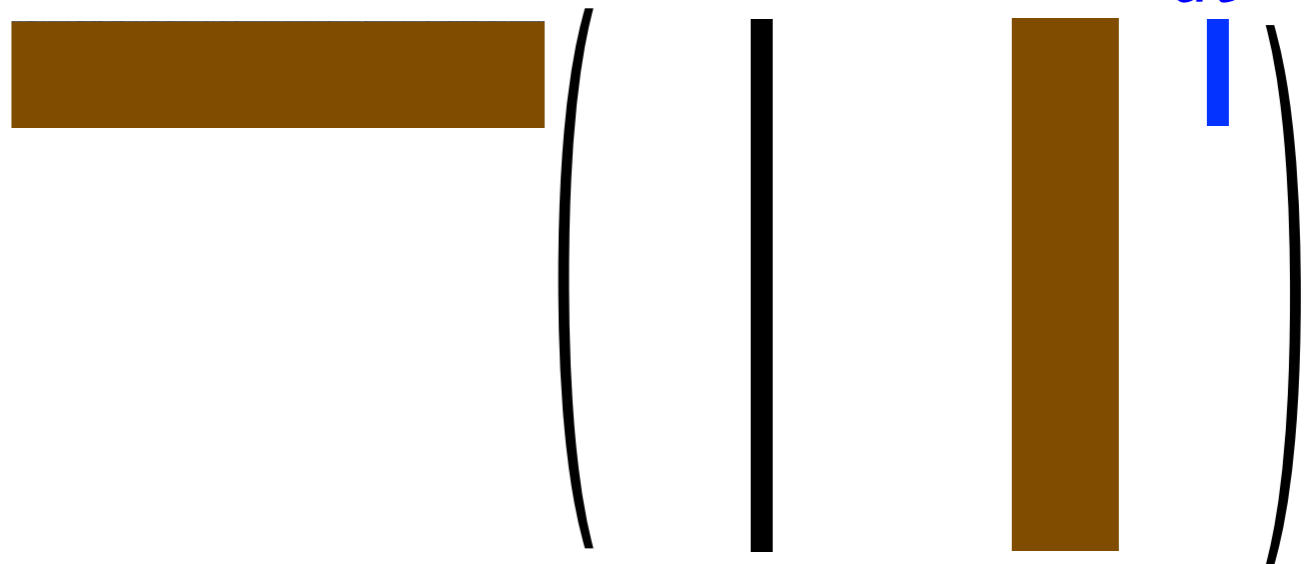


1. *Training*: Solve ODE for  $\mu \in \mathcal{D}_{\text{training}}$  and collect simulation data
2. *Machine learning*: Identify structure in data
3. *Reduction*: Reduce the cost of solving ODE for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$ 
  1. Reduce the number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

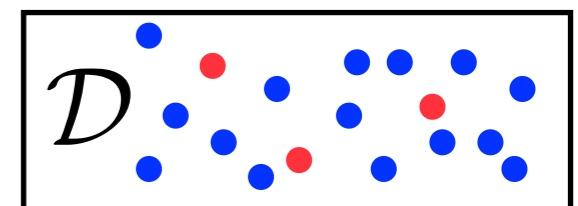
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \Phi \hat{\mathbf{x}}(t)$$



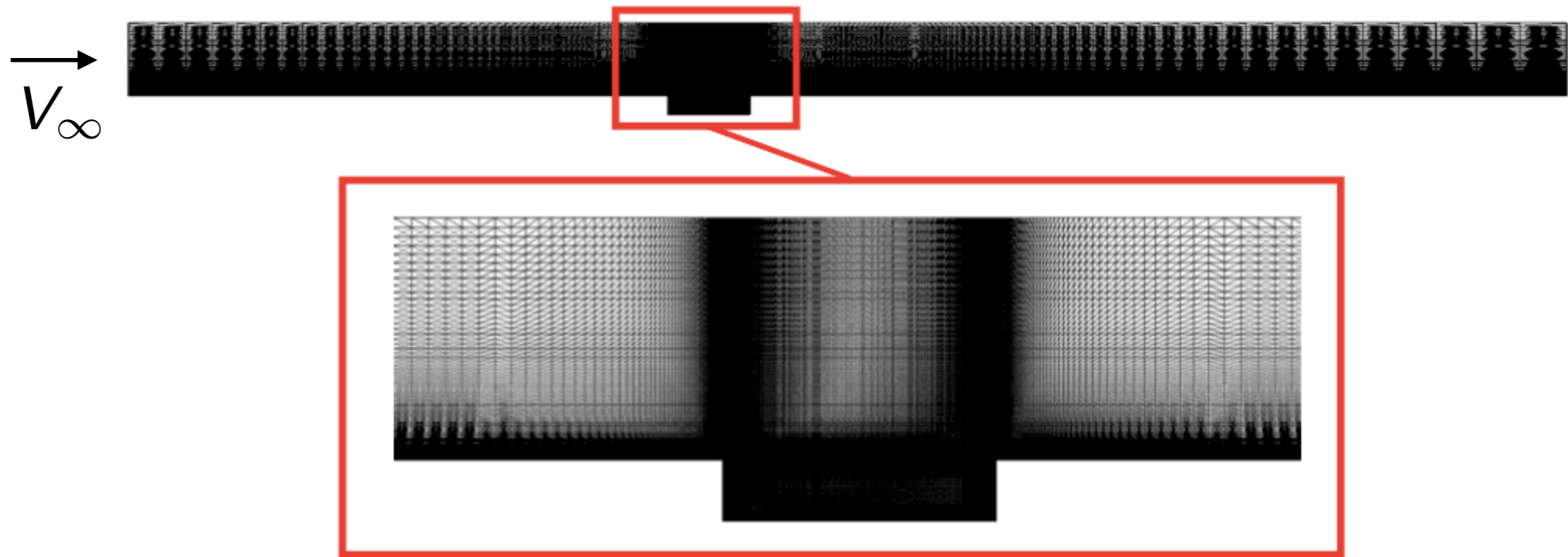
$$\Phi^T (\mathbf{f}(\Phi \hat{\mathbf{x}}; t, \mu) - \Phi \frac{d\hat{\mathbf{x}}}{dt}) = 0$$



$$\text{Galerkin ODE: } \frac{d\hat{\mathbf{x}}}{dt} = \Phi^T \mathbf{f}(\Phi \hat{\mathbf{x}}; t, \mu)$$



# Captive carry



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

## Spatial discretization

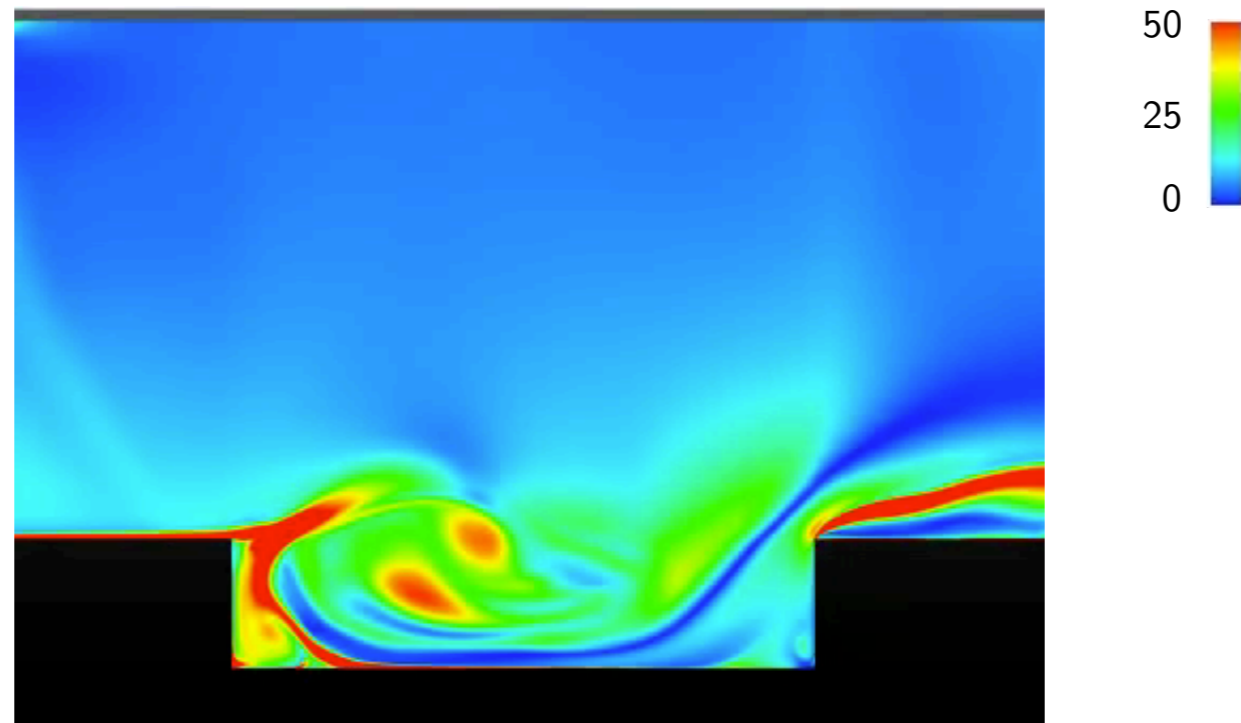
- 2nd-order finite volume
- DES turbulence model
- $1.2 \times 10^6$  degrees of freedom

## Temporal discretization

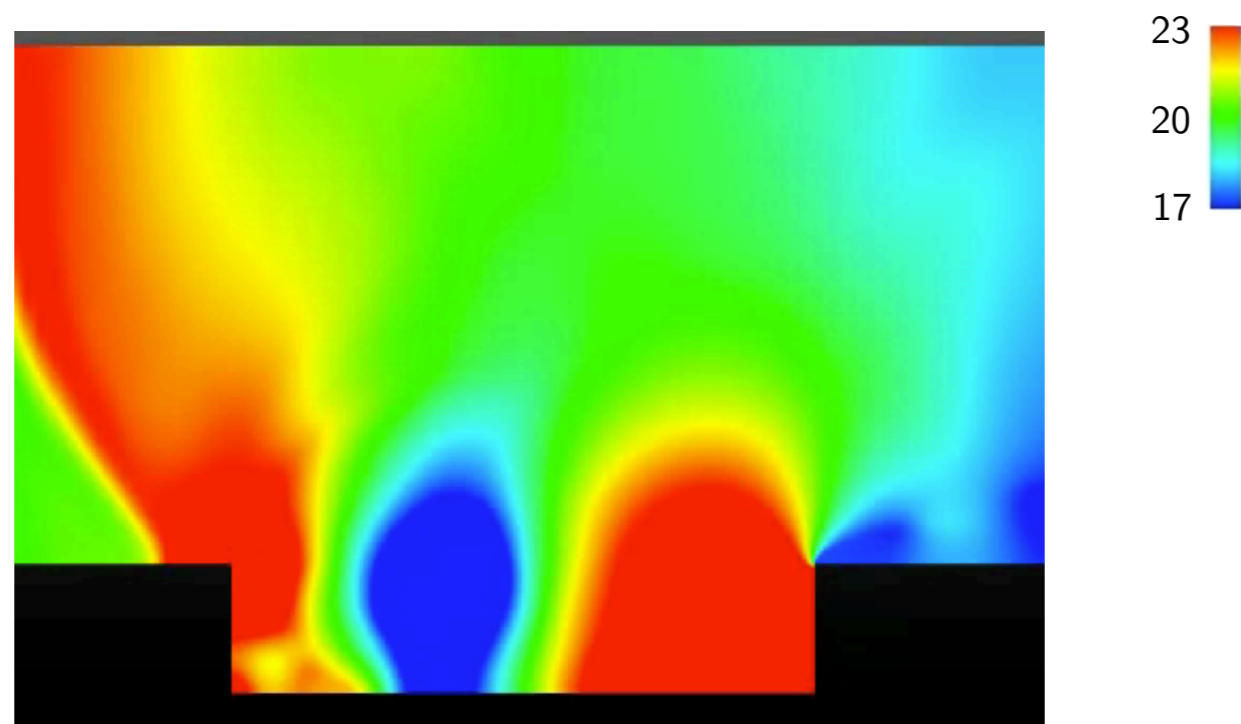
- 2nd-order BDF
- Verified time step  $\Delta t = 1.5 \times 10^{-3}$
- $8.3 \times 10^3$  time instances

# High-fidelity model solution

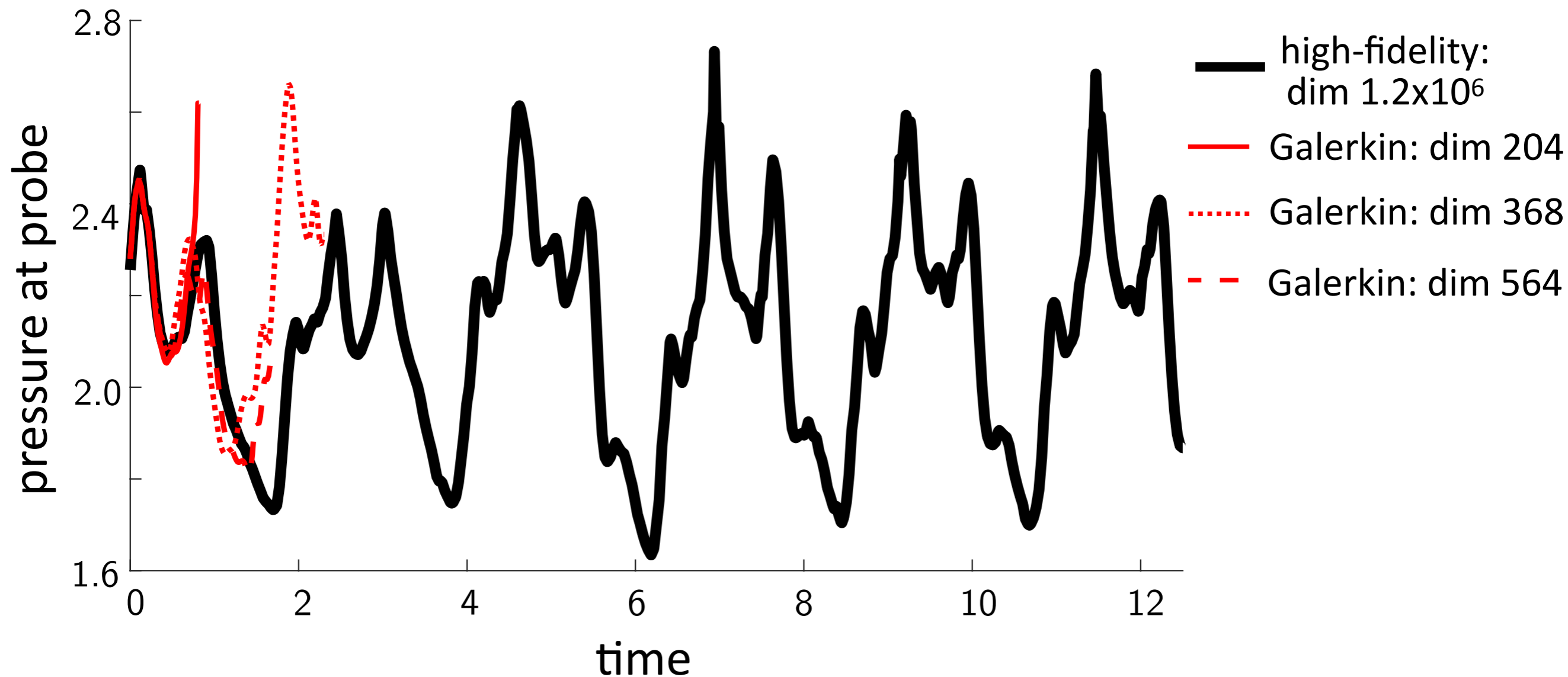
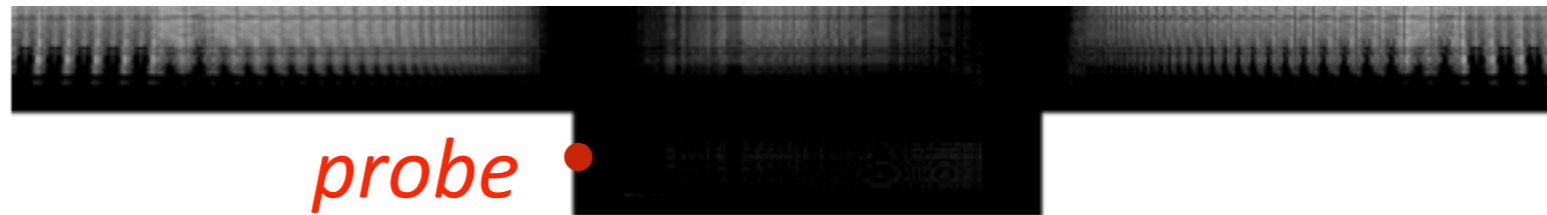
*vorticity field*



*pressure field*



# Galerkin performance

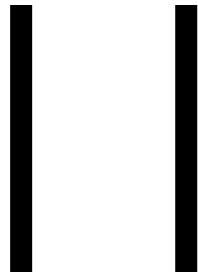


- *Galerkin projection fails* regardless of basis dimension  
***Can we construct a better projection?***

# Galerkin: time-continuous optimality

ODE

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



Galerkin ODE

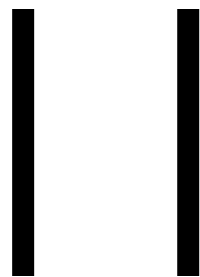
$$\frac{d\hat{\mathbf{x}}}{dt} = \boldsymbol{\Phi}^T \mathbf{f}(\boldsymbol{\Phi} \hat{\mathbf{x}}; t)$$



# Galerkin: time-continuous optimality

ODE

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



Galerkin ODE

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



+ *Time-continuous Galerkin solution: optimal* in the minimum-residual sense:

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

$$\mathbf{r}(\mathbf{v}, \mathbf{x}; t) := \mathbf{v} - \mathbf{f}(\mathbf{x}; t)$$

OΔE

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

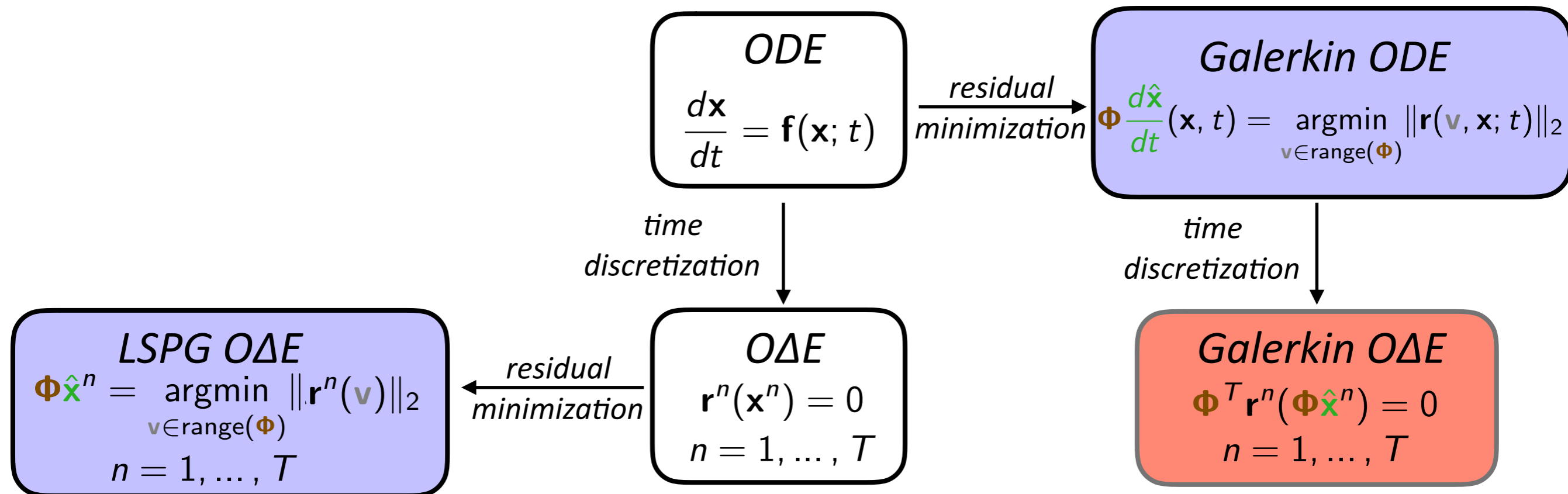
Galerkin OΔE

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

$$\mathbf{r}^n(\mathbf{x}) := \alpha_0 \mathbf{x} - \Delta t \beta_0 \mathbf{f}(\mathbf{x}; t^n) + \sum_{j=1}^k \alpha_j \mathbf{x}^{n-j} - \Delta t \sum_{j=1}^k \beta_j \mathbf{f}(\mathbf{x}^{n-j}; t^{n-j})$$

- *Time-discrete Galerkin solution: not generally optimal* in any sense

# Residual minimization and time discretization



**Least-squares Petrov–Galerkin (LSPG) projection** [C., Bou-Mosleh, Farhat, 2011]

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

If the following conditions hold:

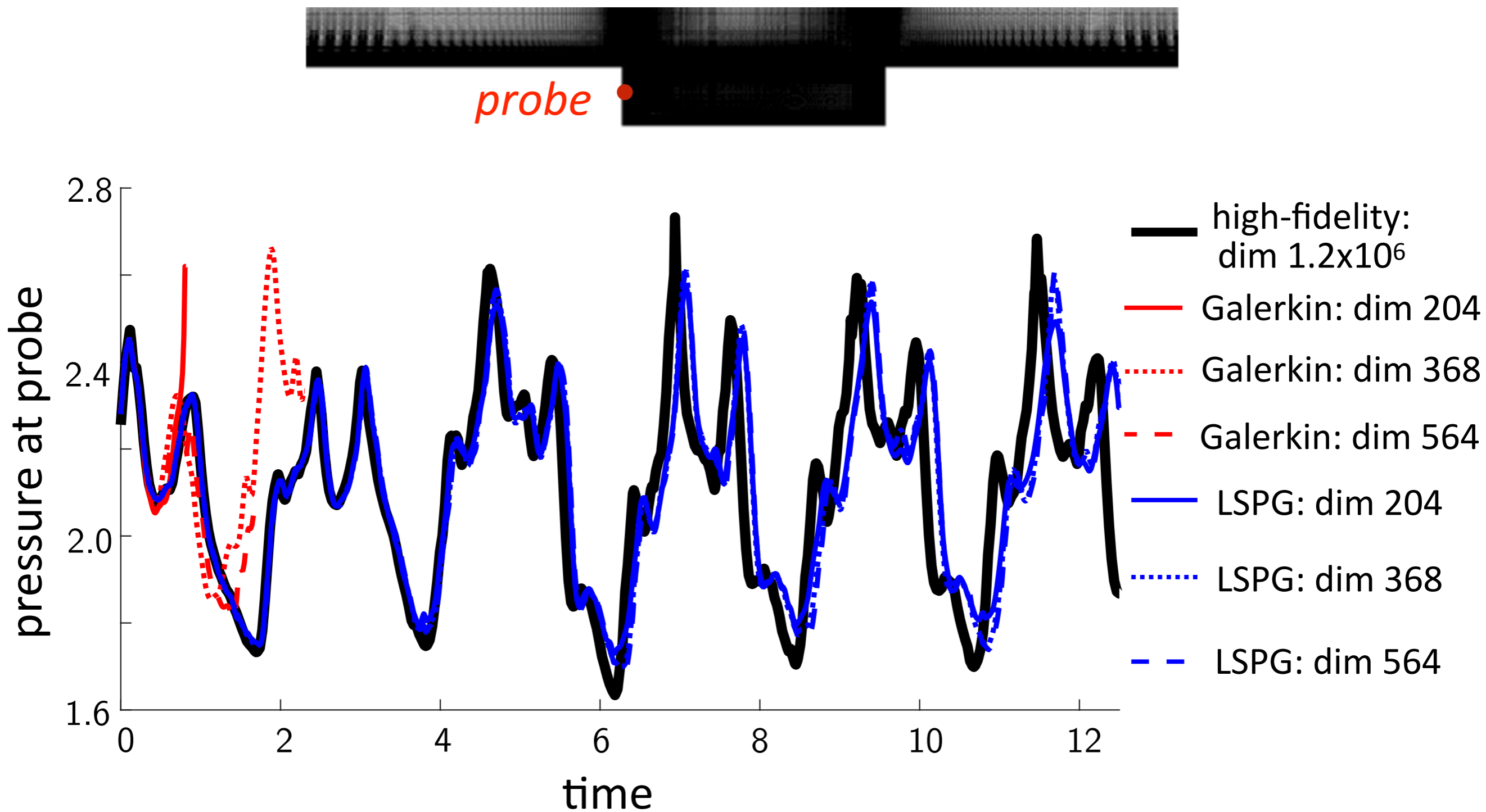
1.  $\mathbf{f}(\cdot; t)$  is Lipschitz continuous with Lipschitz constant  $\kappa$
2.  $\Delta t$  is small enough such that  $0 < h := |\alpha_0| - |\beta_0|\kappa\Delta t$ , then

$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_G^n\|_2 \leq \frac{1}{h} \|\mathbf{r}_G^n(\Phi \hat{\mathbf{x}}_G^n)\|_2 + \frac{1}{h} \sum_{\ell=1}^k |\alpha_\ell| \|\mathbf{x}^{n-\ell} - \Phi \hat{\mathbf{x}}_G^{n-\ell}\|_2$$

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

*+ LSPG sequentially minimizes the error bound*

# LSPG performance



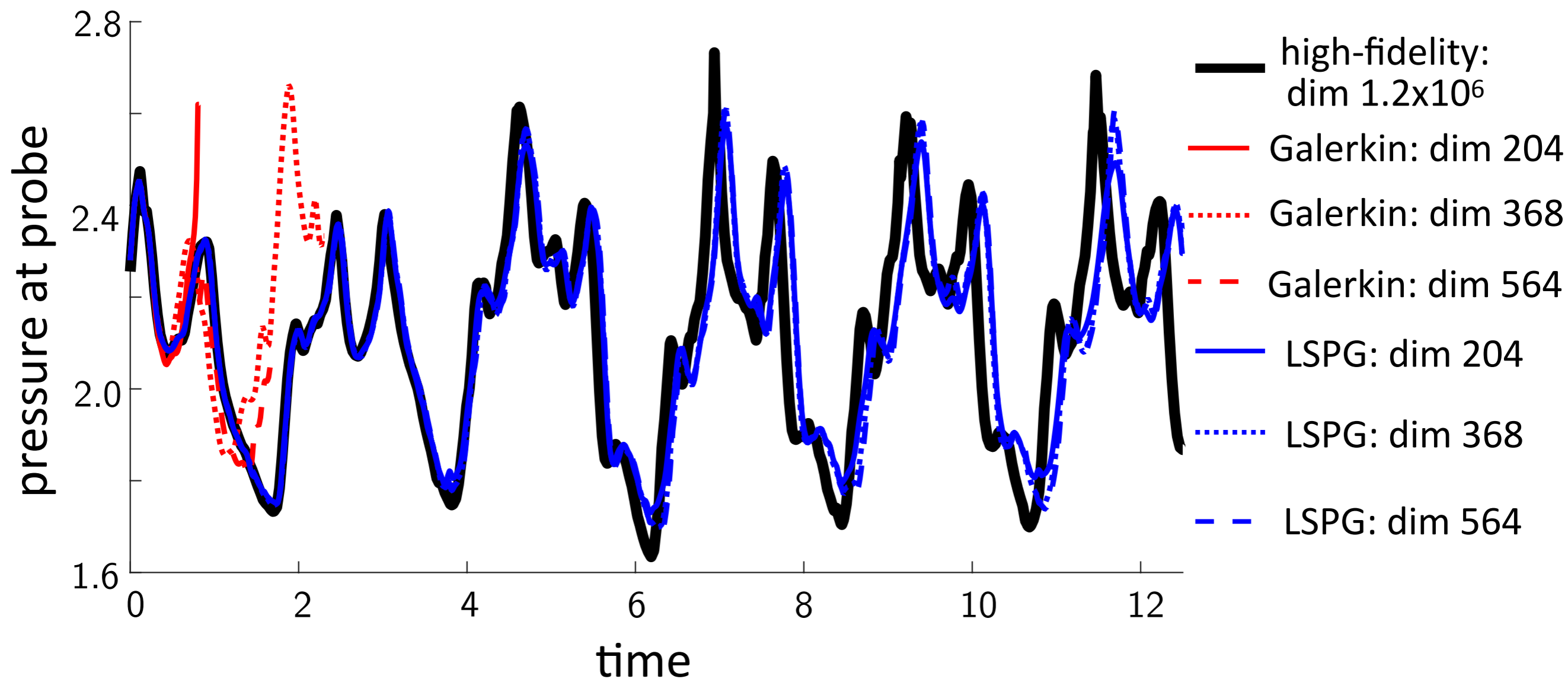
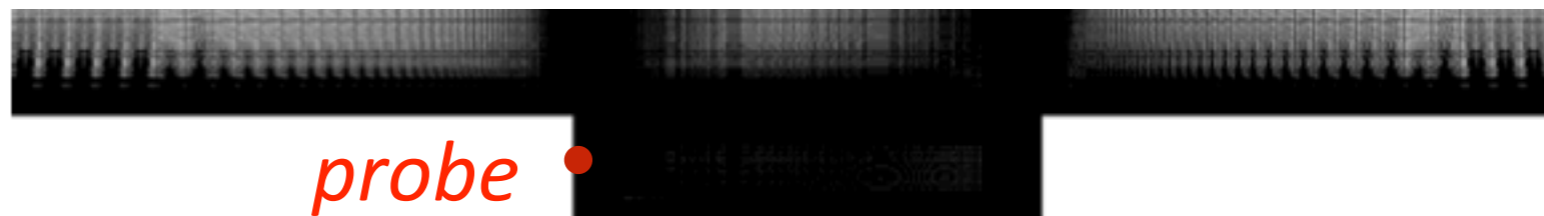
*+ LSPG is far more accurate than Galerkin*

## ***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; Parish and C., 2019]

***Collaborators: Julien Cortial (Stanford), Charbel Farhat (Stanford)***

# Wall-time problem



- High-fidelity simulation: 1 hour, 48 cores
- Fastest LSPG simulation: 1.3 hours, 48 cores

***Why does this occur?***  
***Can we fix it?***

# Cost reduction by gappy PCA [Everson and Sirovich, 1995]

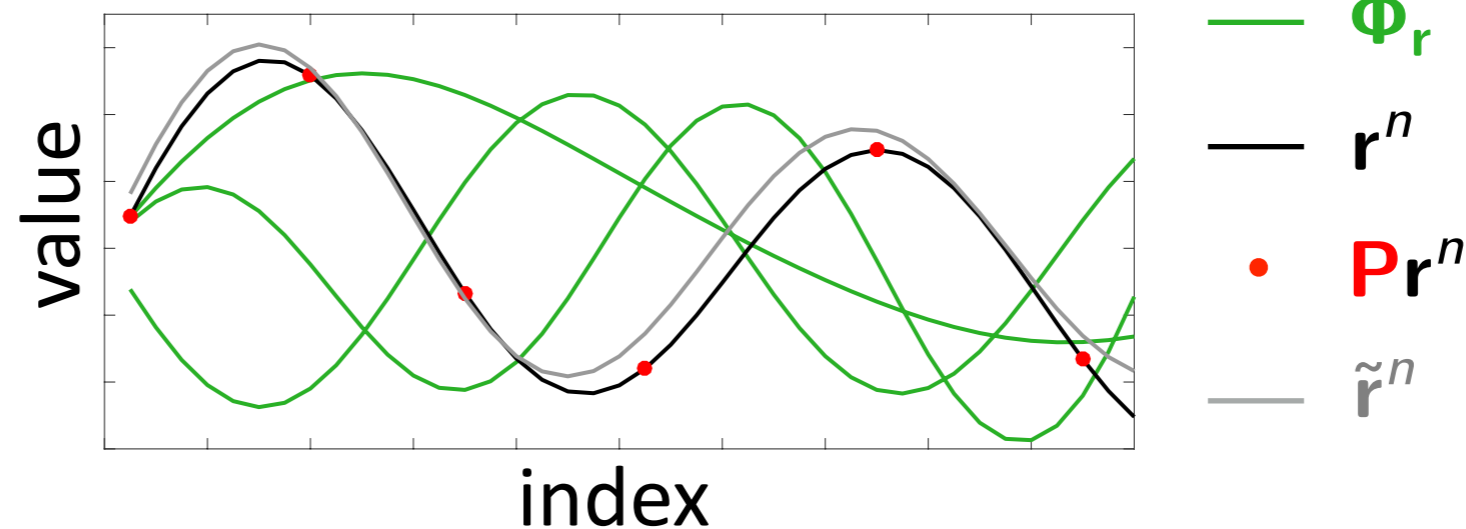
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{r}^n(\Phi \hat{\mathbf{v}}) \right\|_2$$

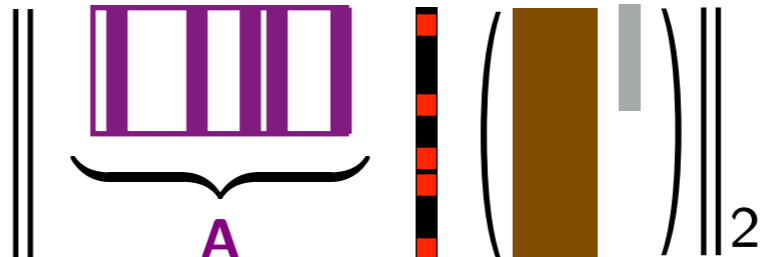
# Cost reduction by gappy PCA [Everson and Sirovich, 1995]

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


Can we introduce a weighting matrix  $\mathbf{A}$  to make this less expensive?

- **Training:** collect residual tensor  $\mathcal{R}^{ijk}$  while solving ODE for  $\mu \in \mathcal{D}_{\text{training}}$
- **Machine learning:** compute residual PCA  $\boldsymbol{\Phi}_r$  and sampling matrix  $\mathbf{P}$
- **Reduction:** compute regression approximation  $\mathbf{r}^n \approx \tilde{\mathbf{r}}^n = \boldsymbol{\Phi}_r(\mathbf{P}\boldsymbol{\Phi}_r)^+ \mathbf{P}\mathbf{r}^n$



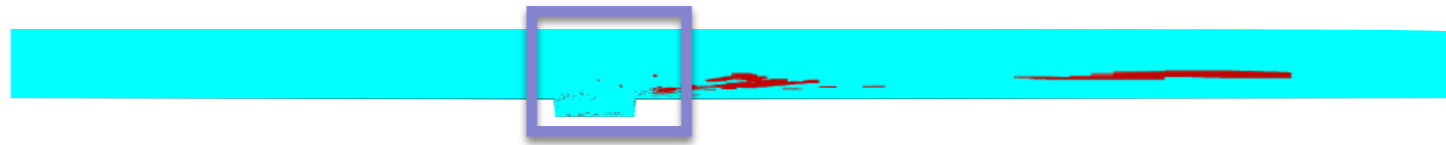
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \underbrace{(\mathbf{P}\boldsymbol{\Phi}_r)^+ \mathbf{P}}_{\mathbf{A}} \mathbf{r}^n(\boldsymbol{\Phi} \hat{\mathbf{v}}) \right\|_2$$


+ Only a *few elements* of  $\mathbf{r}^n$  must be computed

# Sample mesh [C., Farhat, Cortial, Amsallem, 2013]

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \|(\mathbf{P}\Phi_r)^+ \mathbf{P} \mathbf{r}^n(\Phi \hat{\mathbf{v}})\|_2$$

sample  
mesh



+ *HPC on a laptop*

*vorticity field*

*pressure field*

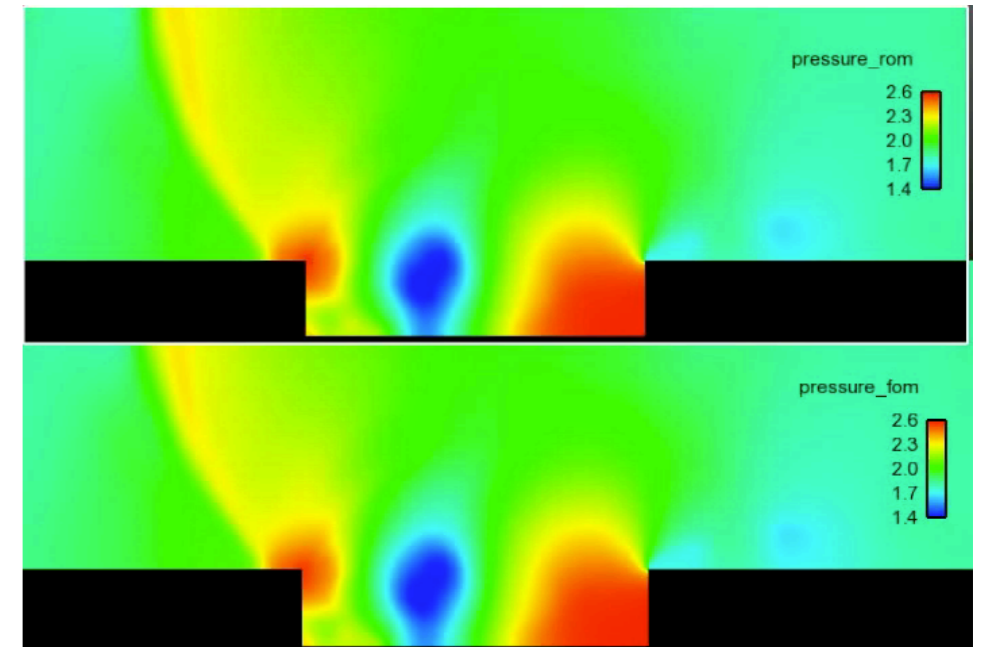
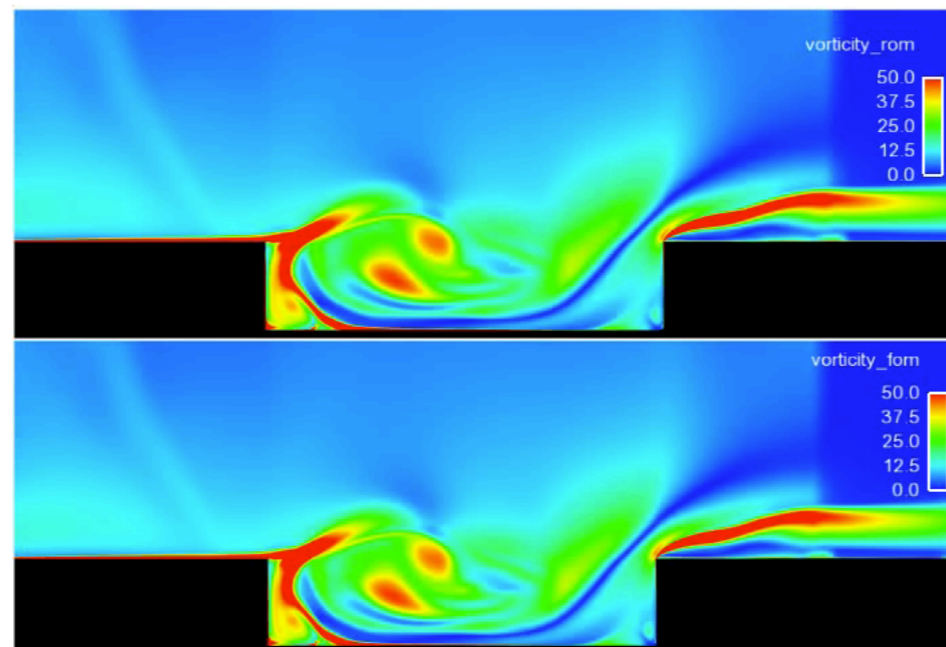
LSPG ROM with

$$\mathbf{A} = (\mathbf{P}\Phi_r)^+ \mathbf{P}$$

32 min, 2 cores

high-fidelity

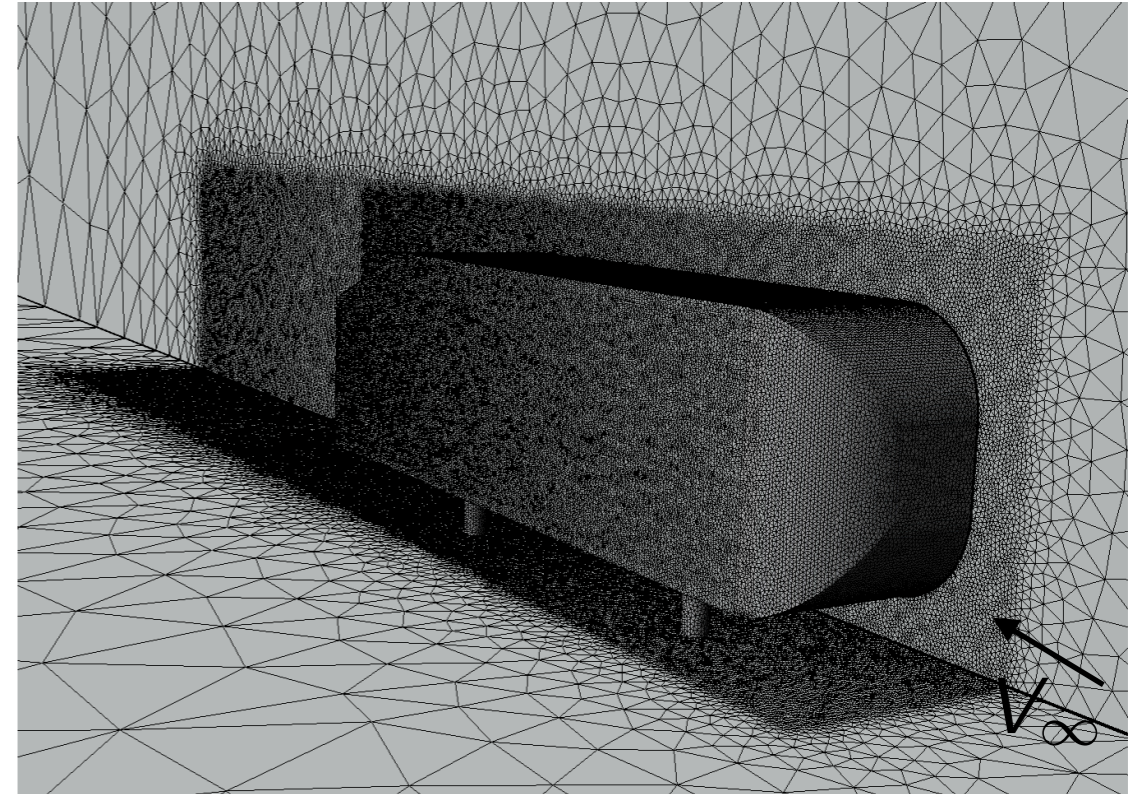
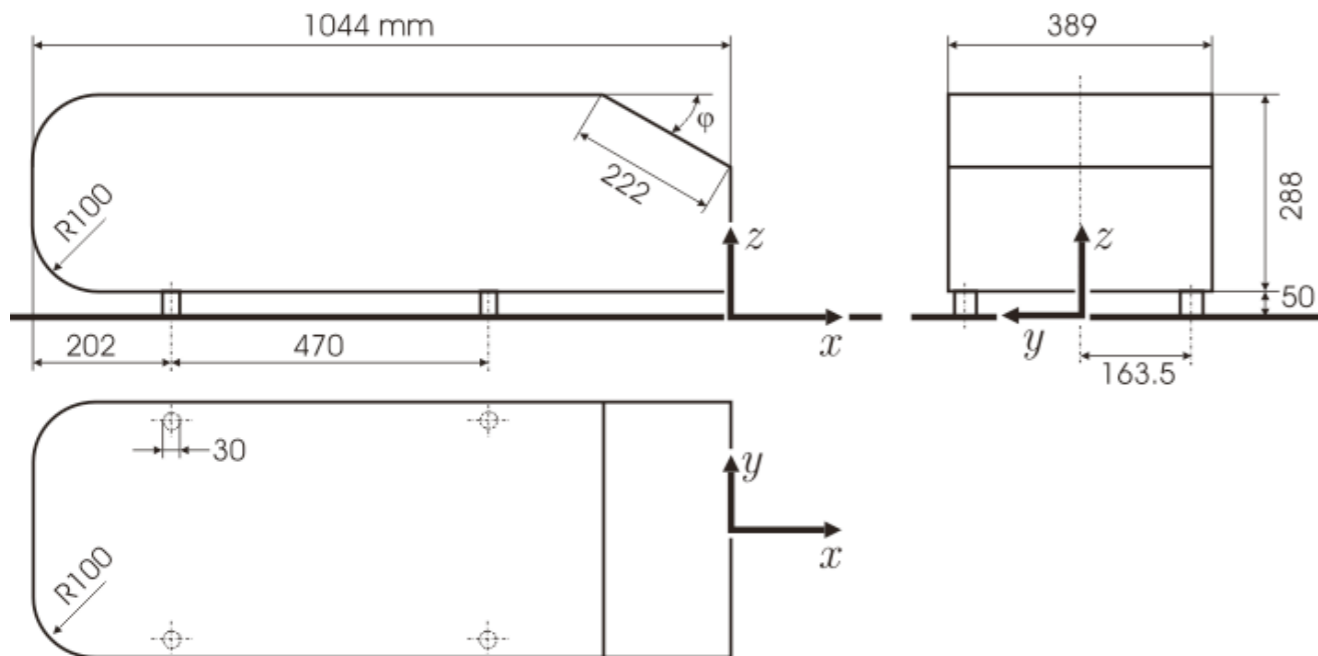
5 hours, 48 cores



+ *229x savings in core-hours*

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

# Ahmed body [Ahmed, Ramm, Faitin, 1984]



- Unsteady Navier–Stokes
- $Re = 4.3 \times 10^6$
- $M_\infty = 0.175$

## Spatial discretization

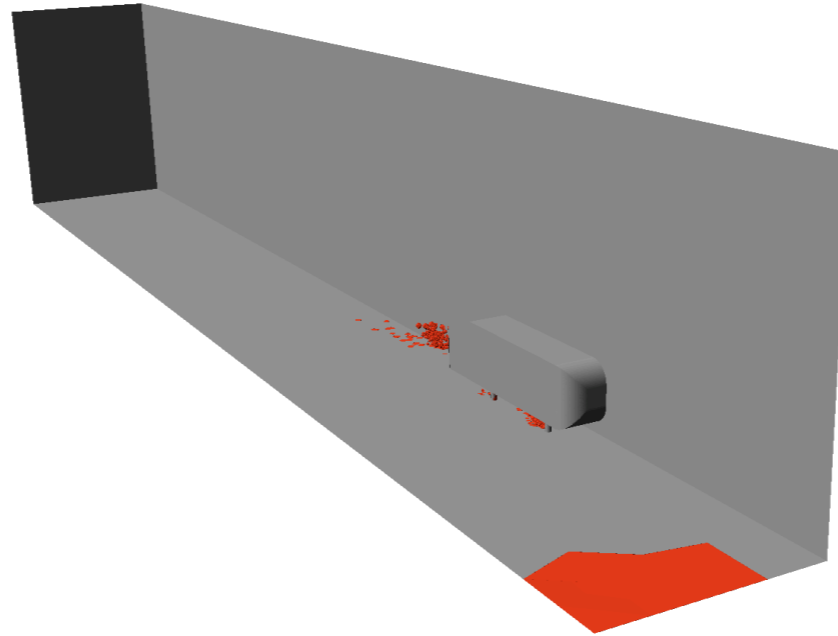
- 2nd-order finite volume
- DES turbulence model
- $1.7 \times 10^7$  degrees of freedom

## Temporal discretization

- 2nd-order BDF
- Time step  $\Delta t = 8 \times 10^{-5} s$
- $1.3 \times 10^3$  time instances

# Ahmed body results [C., Farhat, Cortial, Amsallem, 2013]

sample  
mesh

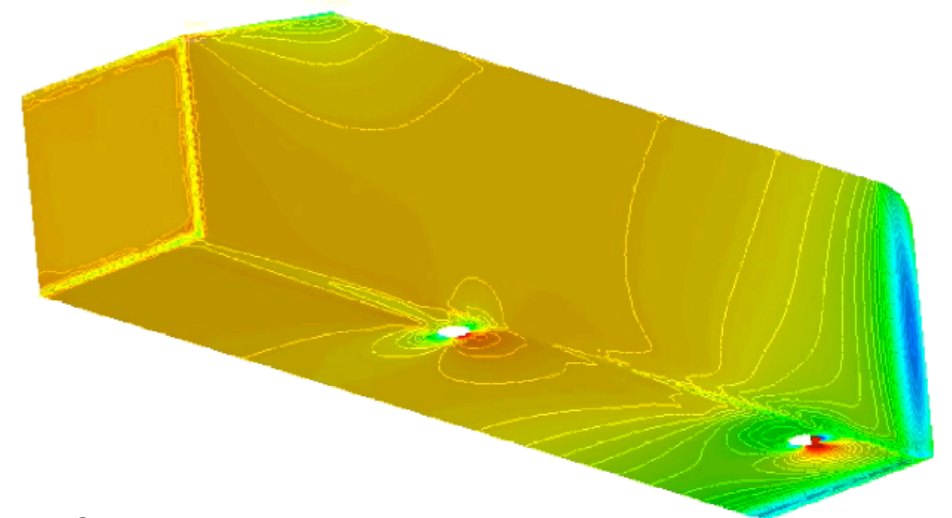
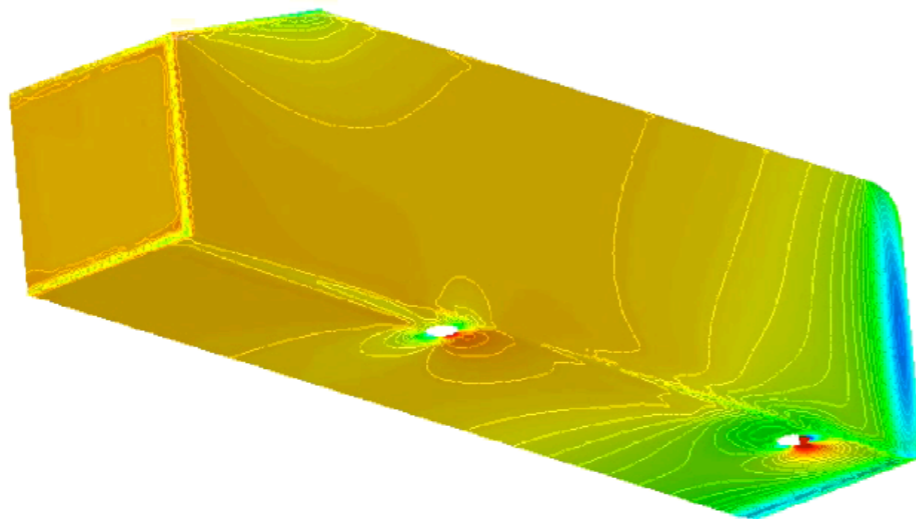


+ *HPC on a laptop*

LSPG ROM with  $\mathbf{A} = (\mathbf{P}\Phi_r)^+ \mathbf{P}$   
4 hours, 4 cores

high-fidelity model  
13 hours, 512 cores

*pressure  
field*



+ *438x savings in core-hours*

+ *Largest nonlinear dynamical system on which ROM has ever had success*

***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; Parish and C., 2019]

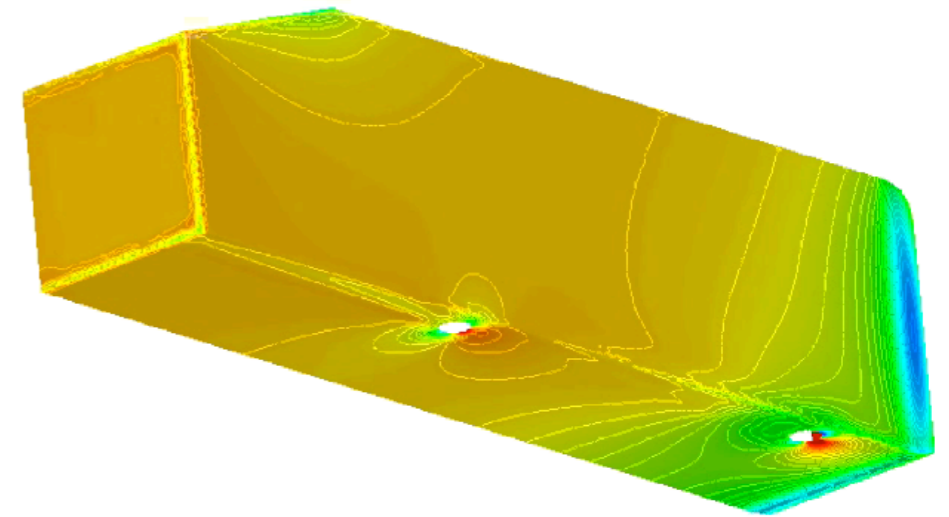
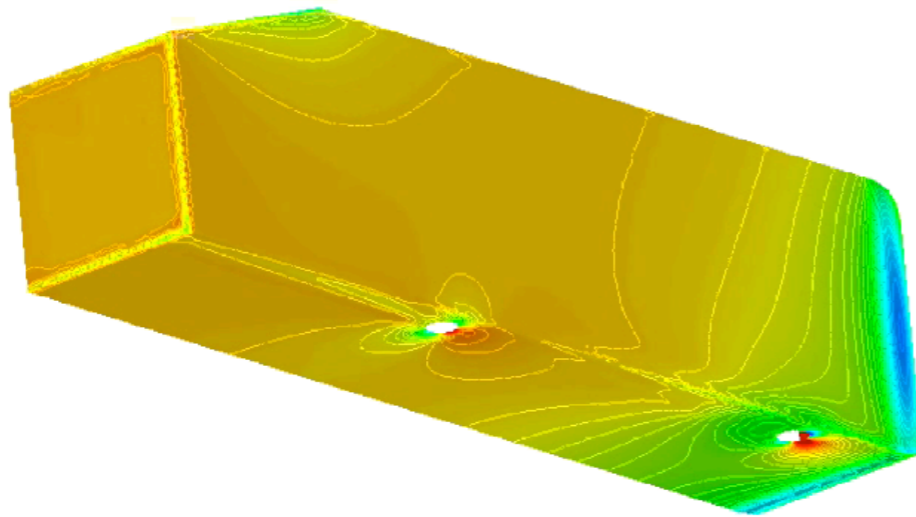
***Collaborator: Youngsoo Choi***

# Ahmed body results [C., Farhat, Cortial, Amsallem, 2013]

GNAT ROM ( $\mathbf{A} = (\mathbf{P}\Phi_r)^+ \mathbf{P}$ )  
4 hours, 4 cores

high-fidelity model  
13 hours, 512 cores

*pressure  
field*



*spatial dim: 283  
temporal dim:  $1.3 \times 10^3$*

*spatial dim:  $1.7 \times 10^7$   
temporal dim:  $1.3 \times 10^3$*

- + 438X computational-cost reduction
- + 60,500X spatial-dimension reduction
- Zero temporal-dimension reduction

**How can we significantly reduce the *temporal dimensionality*?**

# Reducing temporal complexity:

## Larger time steps with ROM

[Krysl et al., 2001; Lucia et al., 2004; Taylor et al., 2010; C. et al., 2017]

- Developed for explicit and implicit integrators
- Limited reduction of time dimension: <10X reductions typical

## Space–time ROMs

- Reduced basis [Urban, Patera, 2012; Yano, 2013; Urban, Patera, 2014; Yano, Patera, Urban, 2014]
- POD–Galerkin [Volkwein, Weiland, 2006; Baumann, Benner, Heiland, 2016]
- ODE-residual minimization [Constantine, Wang, 2012]
- + Reduction of time dimension
- + Linear time-growth of error bounds<sup>^</sup>
- Requires space–time finite element discretization<sup>^</sup>
- No hyper-reduction
- Only one space–time basis vector per training simulation

<sup>^</sup> Only reduced-basis methods

## **Preserve attractive properties of existing space–time ROMs**

- + Reduce both space and time dimensions
- + Slow time-growth of error bound

## **Overcome shortcomings of existing space–time ROMs**

- + Applicability to general nonlinear dynamical systems
- + Hyper-reduction
- + Extract multiple space–time basis vectors from each training simulation

***Space–time least-squares Petrov–Galerkin (ST-LSPG) projection*** [Choi and C., 2019]

# Spatial v. spatiotemporal subspaces

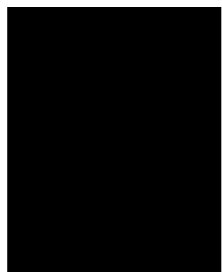
## *High-fidelity-model trial subspace*

$$[\mathbf{x}^1 \ \dots \ \mathbf{x}^T] \in \mathbb{R}^N \otimes \mathbb{R}^T$$



## *Spatial trial subspace*

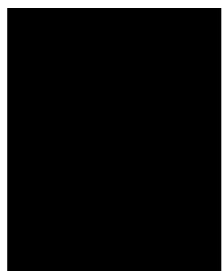
$$[\tilde{\mathbf{x}}^1 \ \dots \ \tilde{\mathbf{x}}^T] = \Phi [\hat{\mathbf{x}}^1 \ \dots \ \hat{\mathbf{x}}^T] \in \mathcal{S} \otimes \mathbb{R}^T \subseteq \mathbb{R}^N \otimes \mathbb{R}^T$$



- + Spatial dimension reduced
- Temporal dimension large

## *Space-time trial subspace*

$$[\tilde{\mathbf{x}}^1 \ \dots \ \tilde{\mathbf{x}}^T] = \sum_{i=1}^{n_{st}} \pi_i \hat{\mathbf{x}}_i(\boldsymbol{\mu}) \in \mathcal{ST} \subseteq \mathbb{R}^N \otimes \mathbb{R}^T$$



- + Spatial dimension reduced
- + Temporal dimension reduced

# Space-time LSPG projection

## LSPG

$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A} \mathbf{r}^n(\boldsymbol{\Phi} \hat{\mathbf{v}}, \tilde{\mathbf{x}}^{n-1}, \dots, \tilde{\mathbf{x}}^{n-k}; \boldsymbol{\mu}) \right\|_2, \quad n = 1, \dots, T$$

$$\left\| \begin{pmatrix} \text{[purple square]} \\ \vdots \\ \text{[purple square]} \end{pmatrix} \right\|_2$$

## ST-LSPG

$$\bar{\mathbf{r}}(\hat{\mathbf{v}}; \boldsymbol{\mu}) := \begin{bmatrix} \mathbf{r}^1 \left( \sum_{i=1}^{n_{st}} \boldsymbol{\pi}_i(t^1) \hat{\mathbf{v}}_i, \sum_{i=1}^{n_{st}} \boldsymbol{\pi}_i(t^0) \hat{\mathbf{v}}_i; \boldsymbol{\mu} \right) \\ \vdots \\ \mathbf{r}^T \left( \sum_{i=1}^{n_{st}} \boldsymbol{\pi}_i(t^T) \hat{\mathbf{v}}_i, \sum_{i=1}^{n_{st}} \boldsymbol{\pi}_i(t^{T-1}) \hat{\mathbf{v}}_i, \dots, \sum_{i=1}^{n_{st}} \boldsymbol{\pi}_i(t^{T-k}) \hat{\mathbf{v}}_i; \boldsymbol{\mu} \right) \end{bmatrix}$$

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

$$\left\| \begin{pmatrix} \text{[purple square]} & \dots & \text{[purple square]} \\ \vdots & \ddots & \vdots \\ \text{[purple square]} & \dots & \text{[purple square]} \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \right\|_2$$

- + applicable to **general** nonlinear dynamical systems
- **prohibitive cost**: minimizing residual over all space and time

# ST-LSPG hyper-reduction

minimize  $\|\bar{\mathbf{A}} \bar{\mathbf{r}}(\hat{\mathbf{v}}; \mu)\|_2$

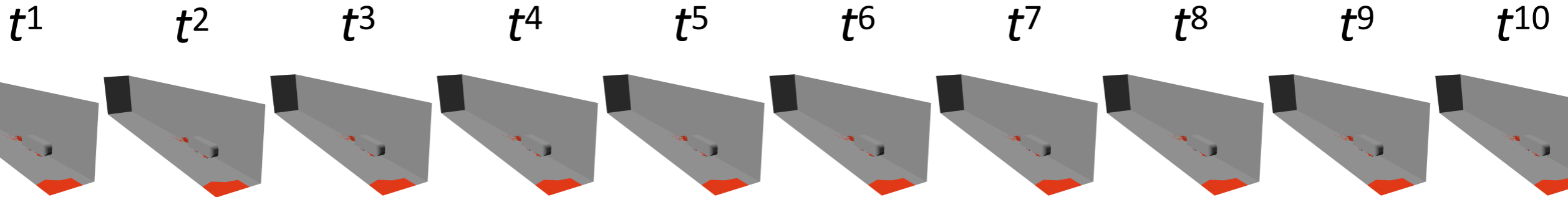
$\bar{\mathbf{r}} \approx \tilde{\mathbf{r}} = \bar{\Phi}_r (\bar{\mathbf{P}} \bar{\Phi}_r)^+ \bar{\mathbf{P}} \bar{\mathbf{r}}$

minimize  $\|(\bar{\mathbf{P}} \bar{\Phi}_r)^+ \bar{\mathbf{P}} \bar{\mathbf{r}}(\hat{\mathbf{v}}; \mu)\|_2$

+ Residual computed at a *few space–time degrees of freedom*

# Sample mesh

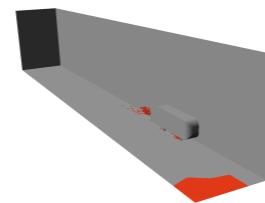
## *LSPG*



- + Residual computed at a few spatial degrees of freedom
- Residual computed at all time instances

## *ST-LSPG*

- $\bar{\mathbf{P}}$ : Kronecker product of space sampling and time sampling

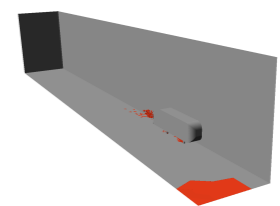
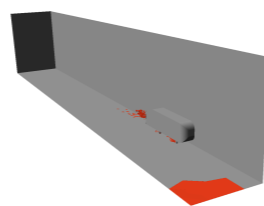
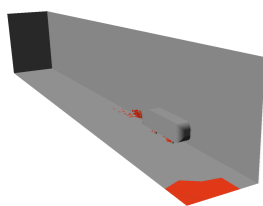


$t^1, t^5, t^9$

$t^1$

$t^5$

$t^9$



- + Residual computed at a few space–time degrees of freedom

# Error bound

## LSPG

- *Sequential solves*: sequential accumulation of time-local errors

$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_{\text{LSPG}}^n\|_2 \leq \frac{\gamma_1 (\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \underbrace{\max_{j \in \{1, \dots, n\}} \min_{\hat{\mathbf{v}}} \|\mathbf{r}_{\text{LSPG}}^j(\Phi \hat{\mathbf{v}})\|_2}_{\text{worst best time-local approximation residual}}$$

- *Stability constant*: exponential time growth
- bounded by the worst (over time) best residual

## ST-LSPG

- + *Single solve*: no sequential error accumulation

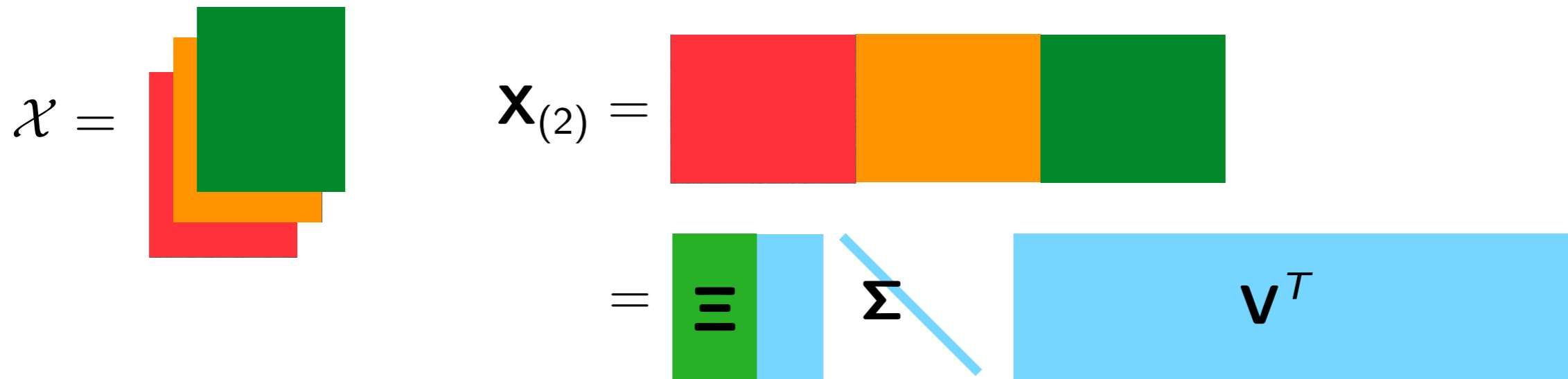
$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_{\text{ST-LSPG}}^n\|_2 \leq \sqrt{T}(1 + \Lambda) \underbrace{\min_{\mathbf{w} \in \mathcal{ST}} \max_{j \in \{1, \dots, T\}} \|\mathbf{x}^n - \mathbf{w}^n\|_2}_{\text{best space-time approximation error}}$$

- + *Stability constant*: polynomial growth in time with degree 3/2
- + bounded by best space–time approximation error

**How to construct space–time trial basis  $\{\boldsymbol{\pi}_i\}_{i=1}^{n_{\text{st}}}$  from training data?**

# Algorithm

1. *Training*: Solve ODE for  $\mu \in \mathcal{D}_{\text{training}}$  and collect simulation data
2. *Machine learning*: Compute truncated high-order SVD (T-HOSVD)
3. *Reduction*: Solve space–time LSPG for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



$\Xi$  columns are principal components of the **temporal** simulation data

$$\pi_{\mathcal{J}(i,j)} = \phi_i \otimes \xi_j$$

The diagram shows the tensor product structure of the basis vectors. A purple rectangle represents the spatial domain, a brown vertical line represents the temporal domain, and a green horizontal line represents the parameter domain. The equation  $\pi_{\mathcal{J}(i,j)} = \phi_i \otimes \xi_j$  indicates that the basis vectors are the tensor product of spatial and temporal components.

+ **N+T storage** per basis vector

‣ **Experiments**: for fixed error, ST-LSPG **almost 100X faster** than LSPG

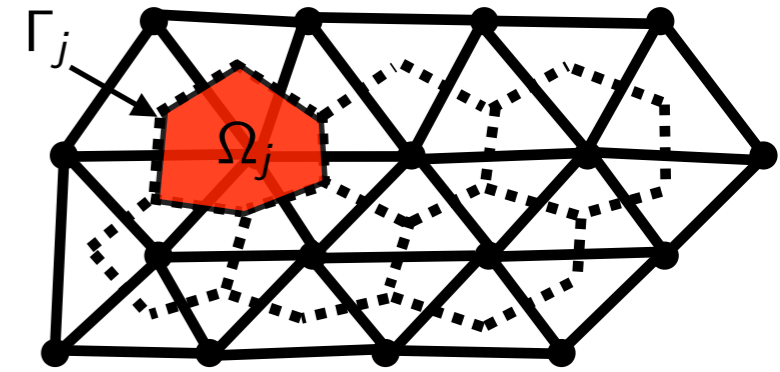
***Accurate, low-cost, structure-preserving,  
generalizable, certified nonlinear model reduction***

- *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; Parish and C., 2019]

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

# Finite-volume method

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



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

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

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

- 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  $j$

$$\text{ODE: } \mathbf{r}^n(\mathbf{x}^n) = 0, \quad n = 1, \dots, N$$

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

- 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

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

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

*- Neither enforces conservation!*

## LSPG

$$\Phi \hat{\mathbf{x}}^n = \operatorname{argmin}_{\mathbf{v} \in \operatorname{range}(\Phi)} \|\mathbf{r}^n(\mathbf{v})\|_2$$

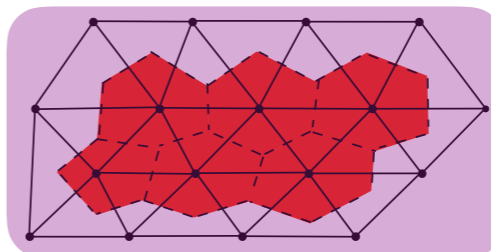
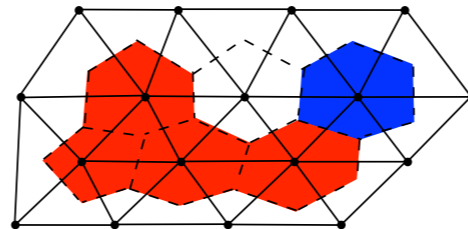
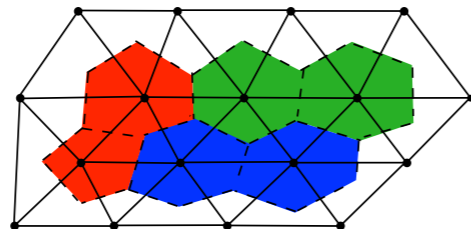
- Minimize sum of squared conservation violations **over time step  $n$**

## Conservative Galerkin

$$\operatorname{minimize}_{\mathbf{v} \in \operatorname{range}(\Phi)} \|\mathbf{r}(\mathbf{v}, \mathbf{x}; t)\|_2$$

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

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



*+ Conservation enforced over prescribed subdomains*

## Conservative LSPG

$$\operatorname{minimize}_{\mathbf{v} \in \operatorname{range}(\Phi)} \|\mathbf{r}^n(\mathbf{v})\|_2$$

subject to  $\mathbf{C}\mathbf{r}^n(\mathbf{v}) = \mathbf{0}$

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

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

## ***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; Parish and C., 2019]

***Collaborator: Kookjin Lee***

# Model reduction can work well...

*vorticity field*

*pressure field*

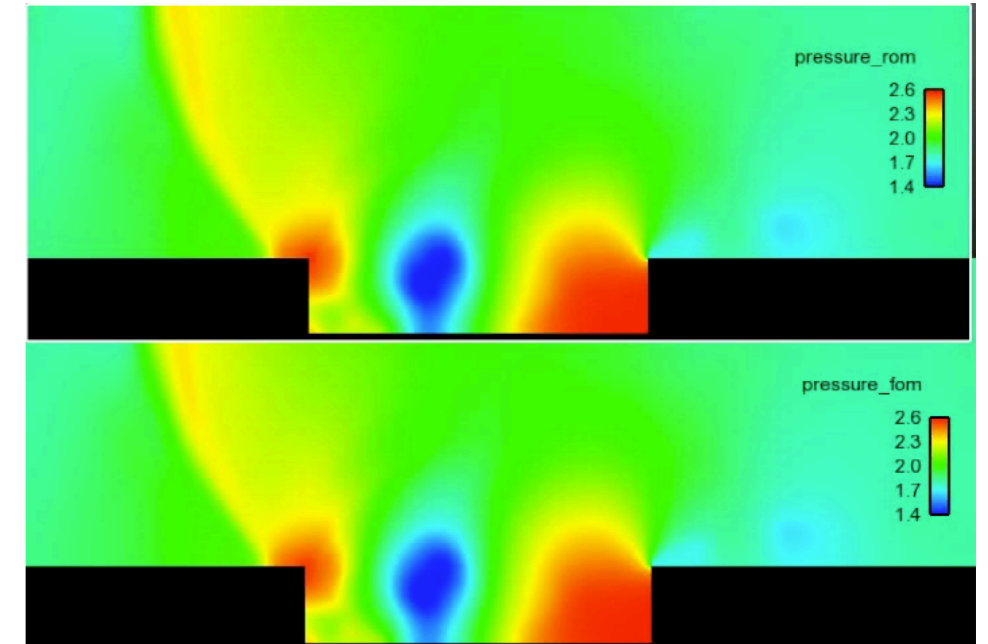
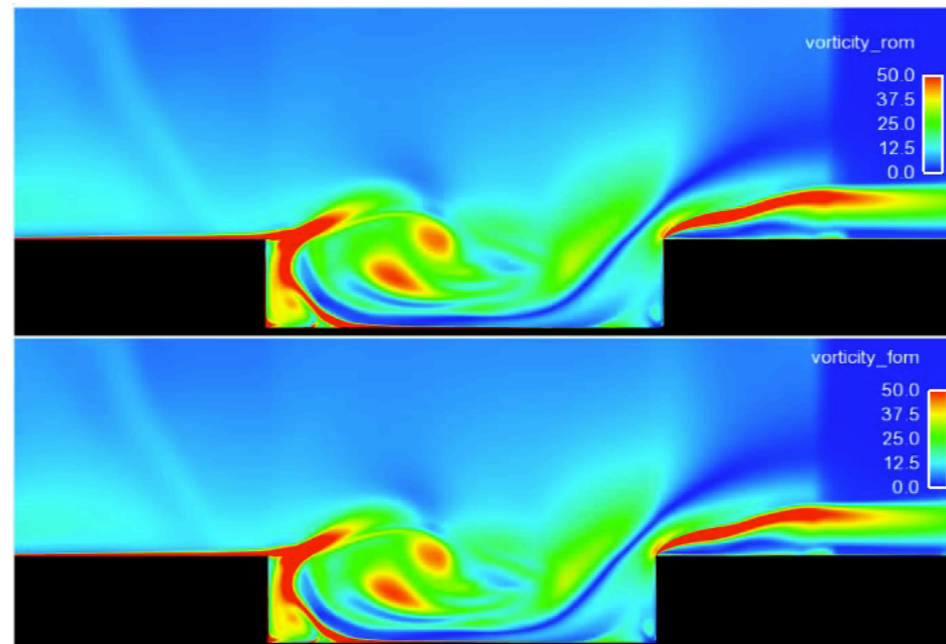
LSPG ROM with

$$\mathbf{A} = (\mathbf{P}\Phi_r)^+ \mathbf{P}$$

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**

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

1) *Linear-subspace assumption is strong* ←

2) *Accuracy limited by information in  $\Phi$*

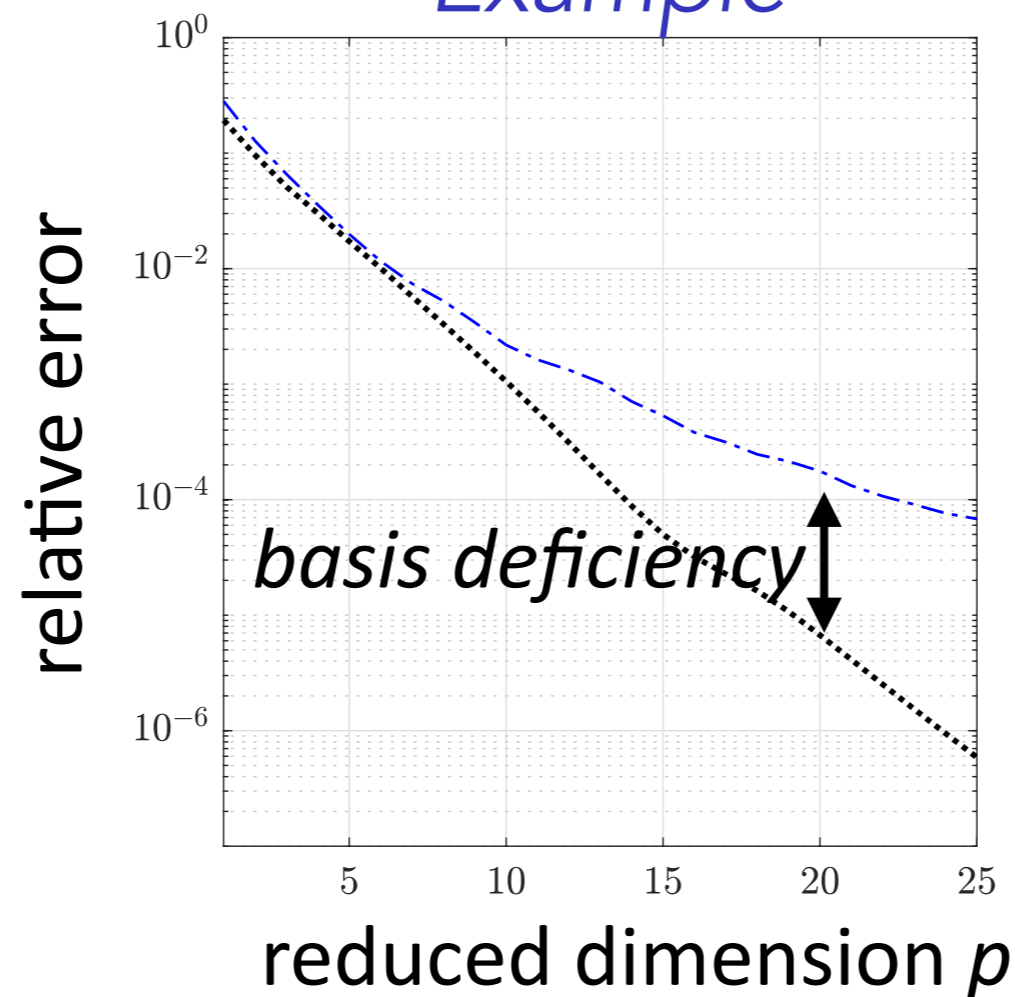
# Kolmogorov-width limitation of linear subspaces

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

# Kolmogorov-width limitation of linear subspaces

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

Example



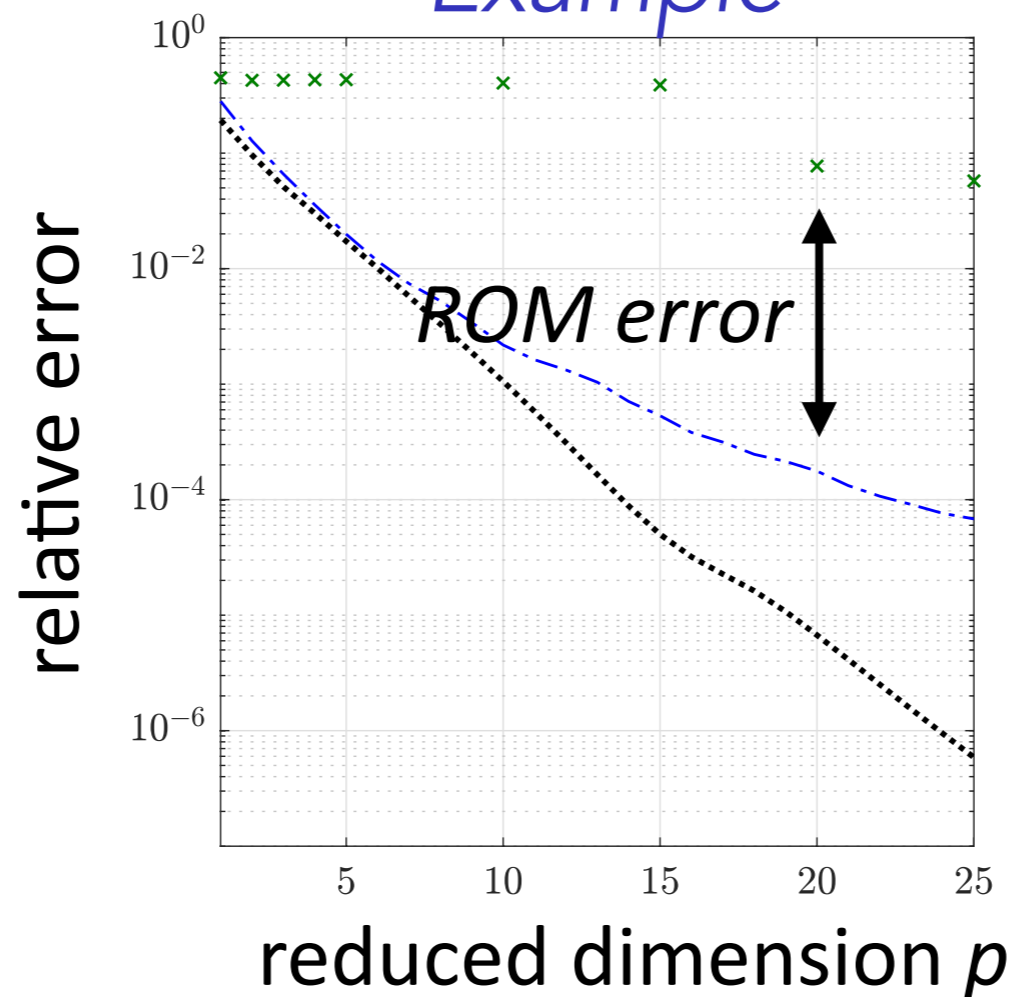
.....  $\tilde{d}_p(\mathcal{M})$

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

# Kolmogorov-width limitation of linear subspaces

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

Example



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

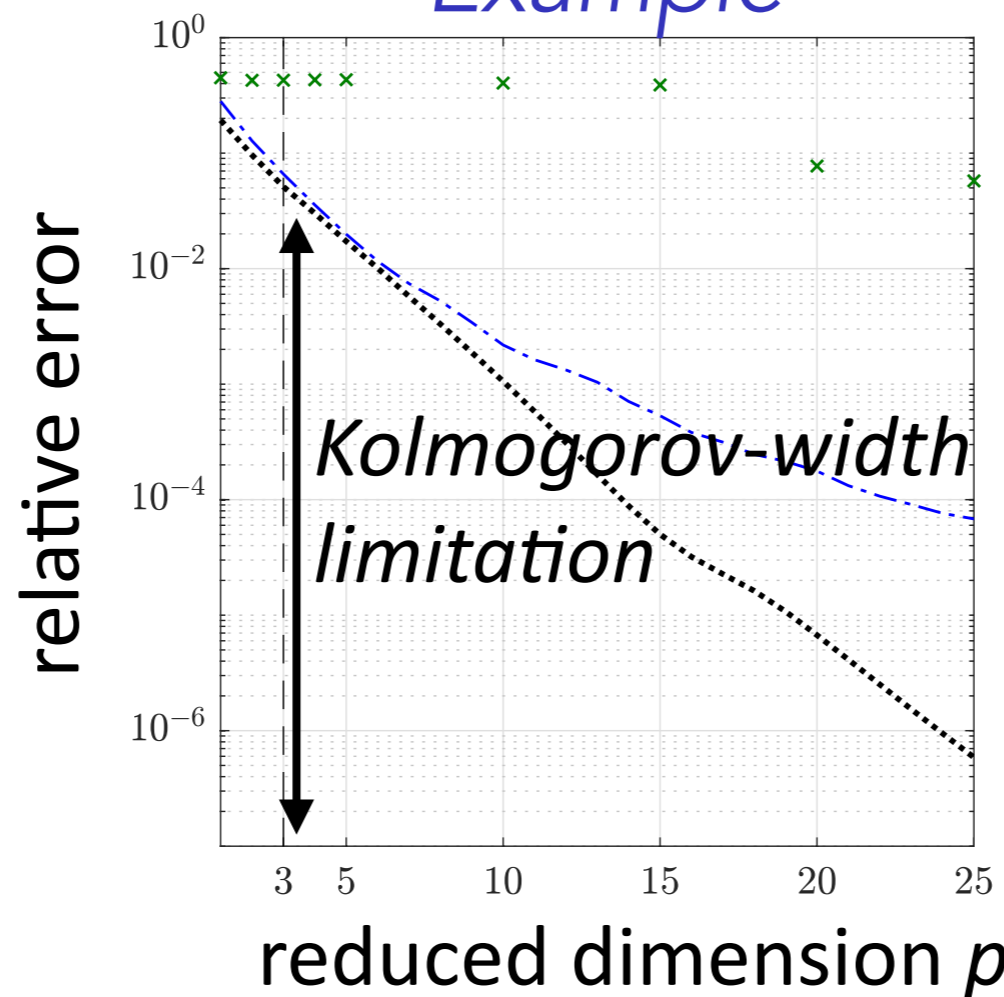
- - -  $P_2(\mathcal{M}, \text{range}(\Phi))$

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

# Kolmogorov-width limitation of linear subspaces

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

Example



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

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

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

|  $\dim(\mathcal{M})$

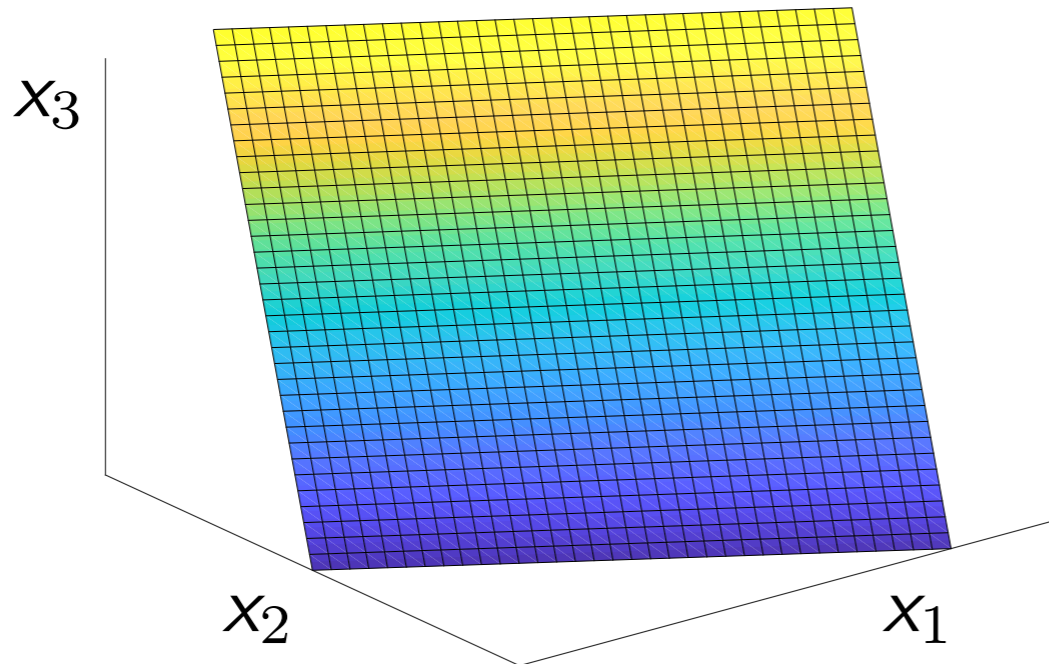
- Kolmogorov-width limitation: **significant error** for  $p = \dim(\mathcal{M})$   
**Goal:** overcome limitation via projection onto a nonlinear manifold

# Nonlinear trial manifold

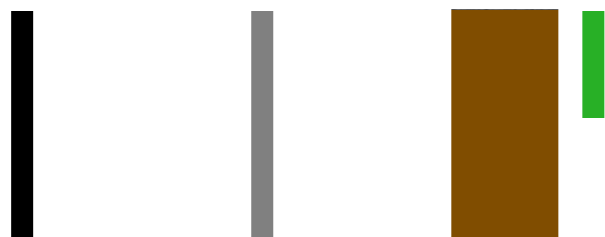
## Linear trial subspace

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

example  
 $N=3$   
 $p=2$

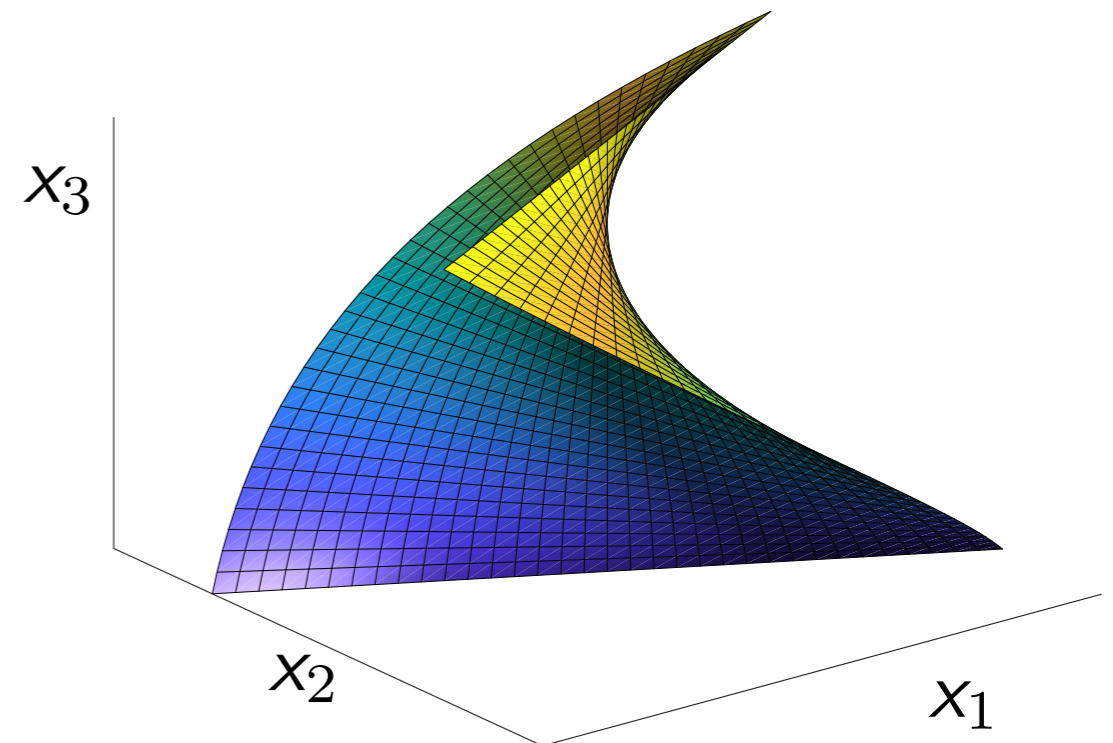


state  $\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \Phi \hat{\mathbf{x}}(t) \in \text{range}(\Phi)$



## Nonlinear trial manifold

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



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



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

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

# Manifold Galerkin and LSPG projection

## Linear-subspace ROM

## Nonlinear-manifold ROM

*Galerkin*  $\frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in \mathbb{R}^n} \|\mathbf{r}(\Phi \hat{\mathbf{v}}, \Phi \hat{\mathbf{x}}; t)\|_2$

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



$$\Phi \frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in \operatorname{range}(\Phi)} \|\hat{\mathbf{v}} - \mathbf{f}(\Phi \hat{\mathbf{x}}; t)\|_2$$

$$\nabla \mathbf{g}(\hat{\mathbf{x}}) \frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in T_{\hat{\mathbf{x}}} \mathcal{S}} \|\hat{\mathbf{v}} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); t)\|_2$$



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



$$\frac{d\hat{\mathbf{x}}}{dt} = \nabla \mathbf{g}(\hat{\mathbf{x}})^+ \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); t)$$

## LSPG

$$\hat{\mathbf{x}}^n = \operatorname{argmin}_{\hat{\mathbf{v}} \in \mathbb{R}^p} \|\mathbf{r}^n(\Phi \hat{\mathbf{v}})\|_2$$

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

+ Satisfy residual-minimization properties

# Manifold Galerkin and LSPG projection

## Linear-subspace ROM

## Nonlinear-manifold ROM

*Galerkin*  $\frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in \mathbb{R}^n} \|\mathbf{r}(\Phi \hat{\mathbf{v}}, \Phi \hat{\mathbf{x}}; t)\|_2$

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



$$\Phi \frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in \operatorname{range}(\Phi)} \|\hat{\mathbf{v}} - \mathbf{f}(\Phi \hat{\mathbf{x}}; t)\|_2$$

$$\nabla \mathbf{g}(\hat{\mathbf{x}}) \frac{d\hat{\mathbf{x}}}{dt} = \operatorname{argmin}_{\hat{\mathbf{v}} \in T_{\hat{\mathbf{x}}} \mathcal{S}} \|\hat{\mathbf{v}} - \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); t)\|_2$$



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



$$\frac{d\hat{\mathbf{x}}}{dt} = \nabla \mathbf{g}(\hat{\mathbf{x}})^+ \mathbf{f}(\mathbf{g}(\hat{\mathbf{x}}); t)$$

## LSPG

$$\hat{\mathbf{x}}^n = \operatorname{argmin}_{\hat{\mathbf{v}} \in \mathbb{R}^p} \|\mathbf{r}^n(\Phi \hat{\mathbf{v}})\|_2$$

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

+ Satisfy residual-minimization properties

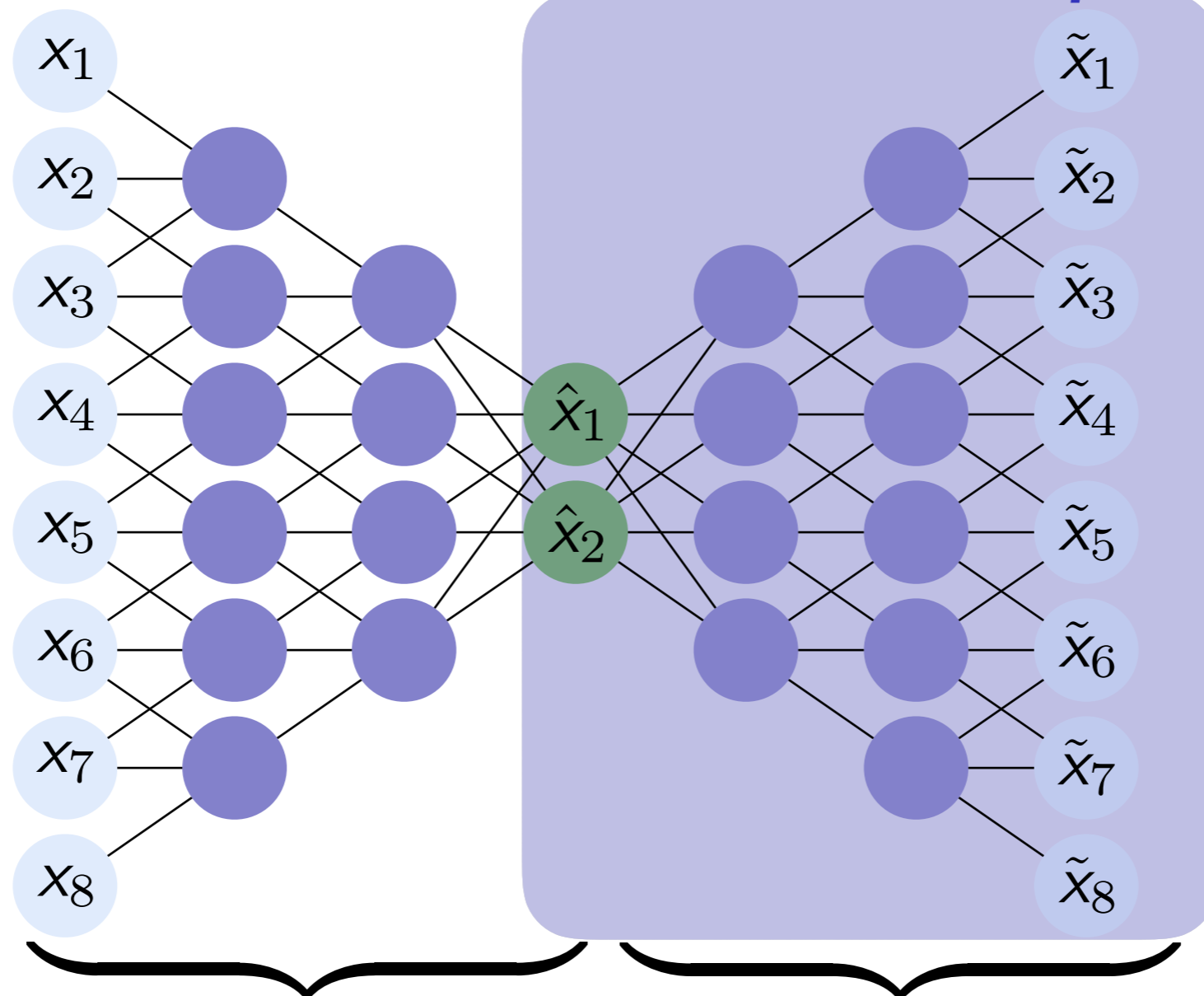
**How to construct manifold  $\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \mid \hat{\mathbf{x}} \in \mathbb{R}^p\}$  from training data?**

# Deep autoencoders

*Input layer*

*Code*

*Output layer*



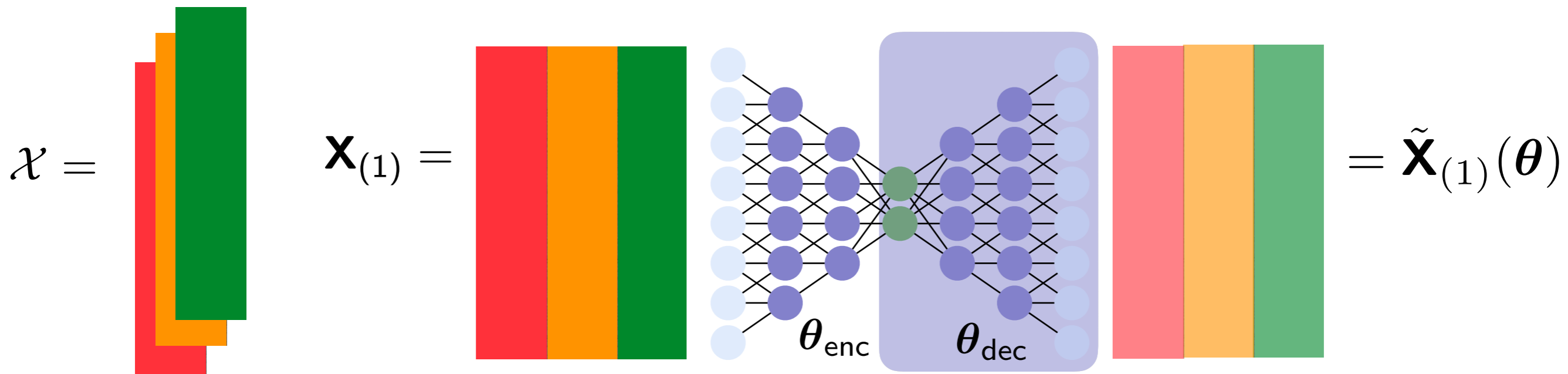
**Encoder**  $\mathbf{h}_{\text{enc}}(\cdot; \boldsymbol{\theta}_{\text{enc}})$  **Decoder**  $\mathbf{h}_{\text{dec}}(\cdot; \boldsymbol{\theta}_{\text{dec}})$

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

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

# Algorithm

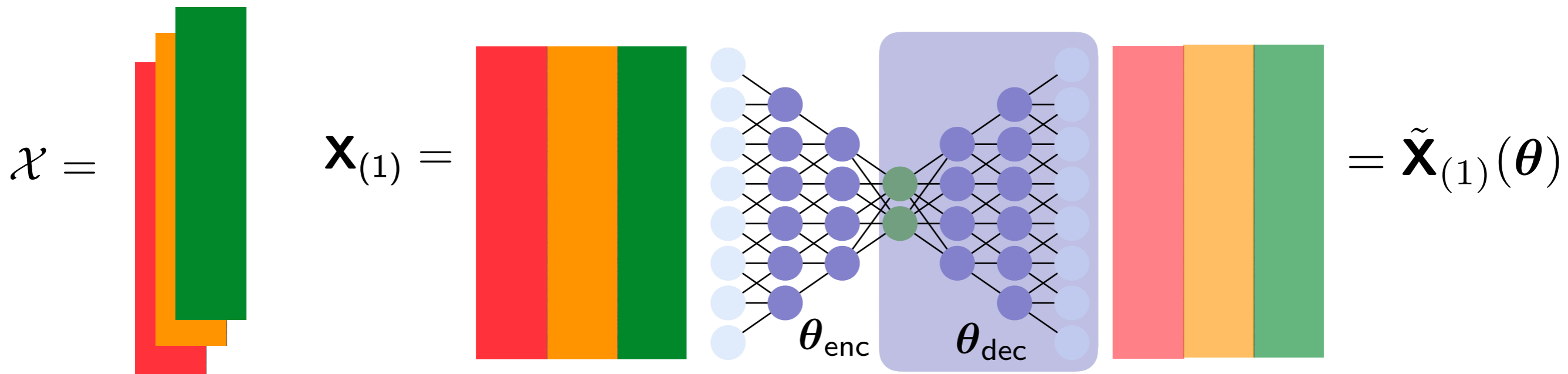
1. *Training*: Solve ODE for  $\mu \in \mathcal{D}_{\text{training}}$  and collect simulation data
2. *Machine learning*: Train deep convolutional autoencoder
3. *Reduction*: Solve manifold Galerkin or LSPG for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



- Compute  $\theta^*$  by approximately solving  $\min_{\theta} \|\mathbf{X}_{(1)} - \tilde{\mathbf{X}}_{(1)}(\theta)\|_F$
- Define nonlinear trial manifold by setting  $\mathbf{g} = \mathbf{h}_{\text{dec}}(\cdot; \theta_{\text{dec}}^*)$

# Algorithm

1. *Training*: Solve ODE for  $\mu \in \mathcal{D}_{\text{training}}$  and collect simulation data
2. *Machine learning*: Train deep convolutional autoencoder
3. *Reduction*: Solve manifold Galerkin or LSPG for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



- Compute  $\boldsymbol{\theta}^*$  by approximately solving  $\min_{\boldsymbol{\theta}} \|\mathbf{X}_{(1)} - \tilde{\mathbf{X}}_{(1)}(\boldsymbol{\theta})\|_F$
- Define nonlinear trial manifold by setting  $\mathbf{g} = \mathbf{h}_{\text{dec}}(\cdot; \boldsymbol{\theta}_{\text{dec}}^*)$

# Numerical results

## 1D Burgers' equation

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

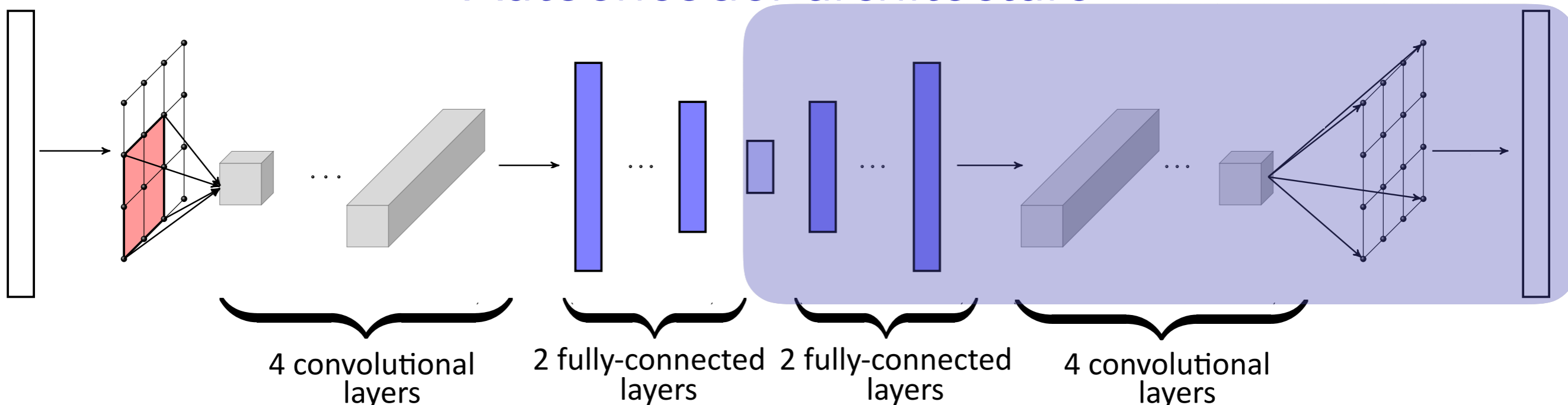
- $\mu$ :  $\alpha$ , inlet boundary condition
- *Spatial discretization*: finite volume
- *Time integrator*: backward Euler

## 2D Chemically reacting flow

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

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

## Autoencoder architecture



# Manifold LSPG outperforms optimal linear subspace

*1D Burgers' equation*

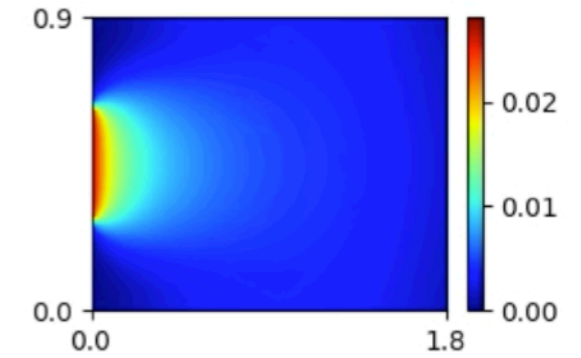
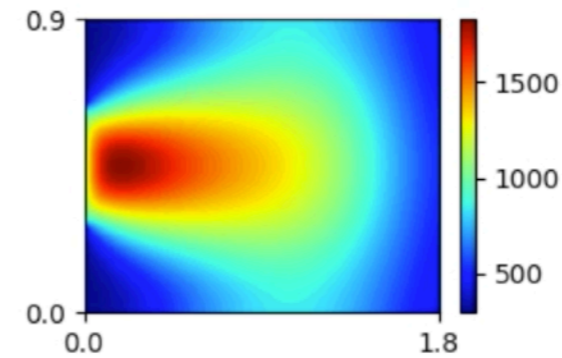
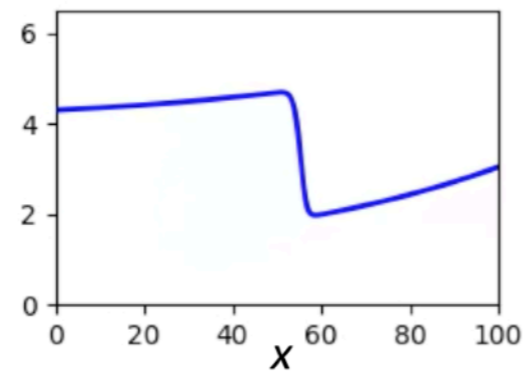
*2D reacting flow*

conserved variable

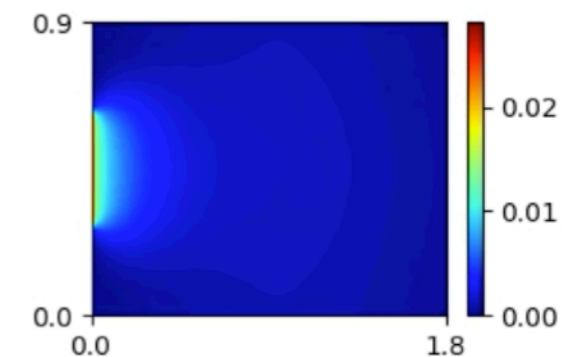
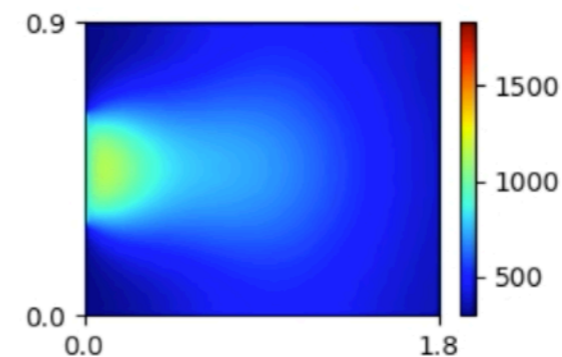
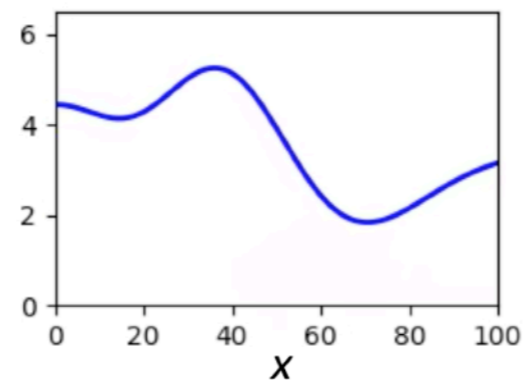
temperature

$H_2$  fraction

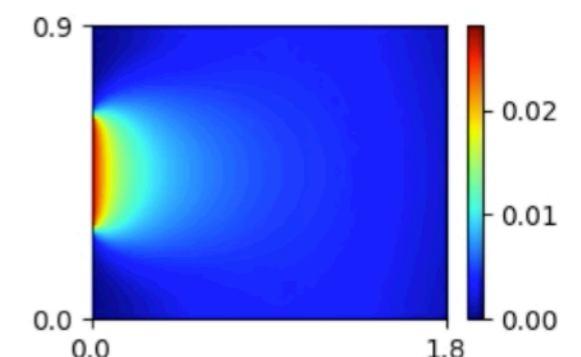
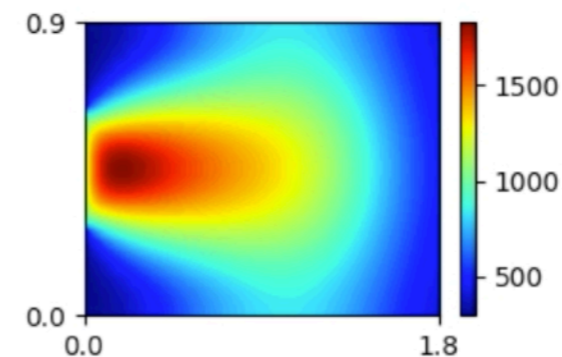
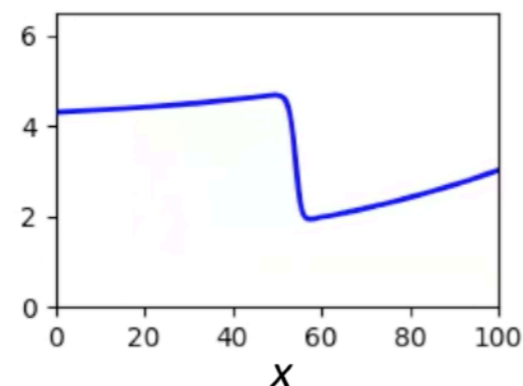
*high-fidelity  
model*



*POD-LSPG  
 $p=5$*



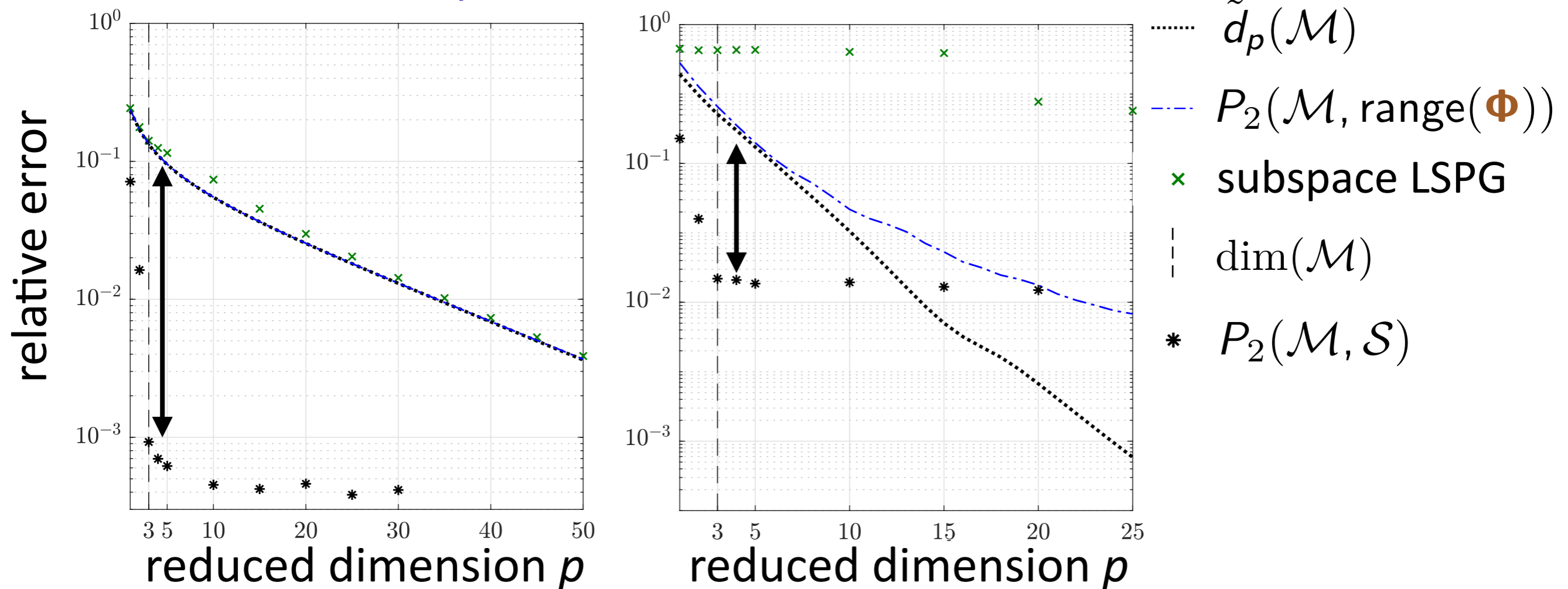
*Manifold LSPG  
 $p=5$*



# Method overcomes Kolmogorov-width limitation

1D Burgers' equation

2D reacting flow

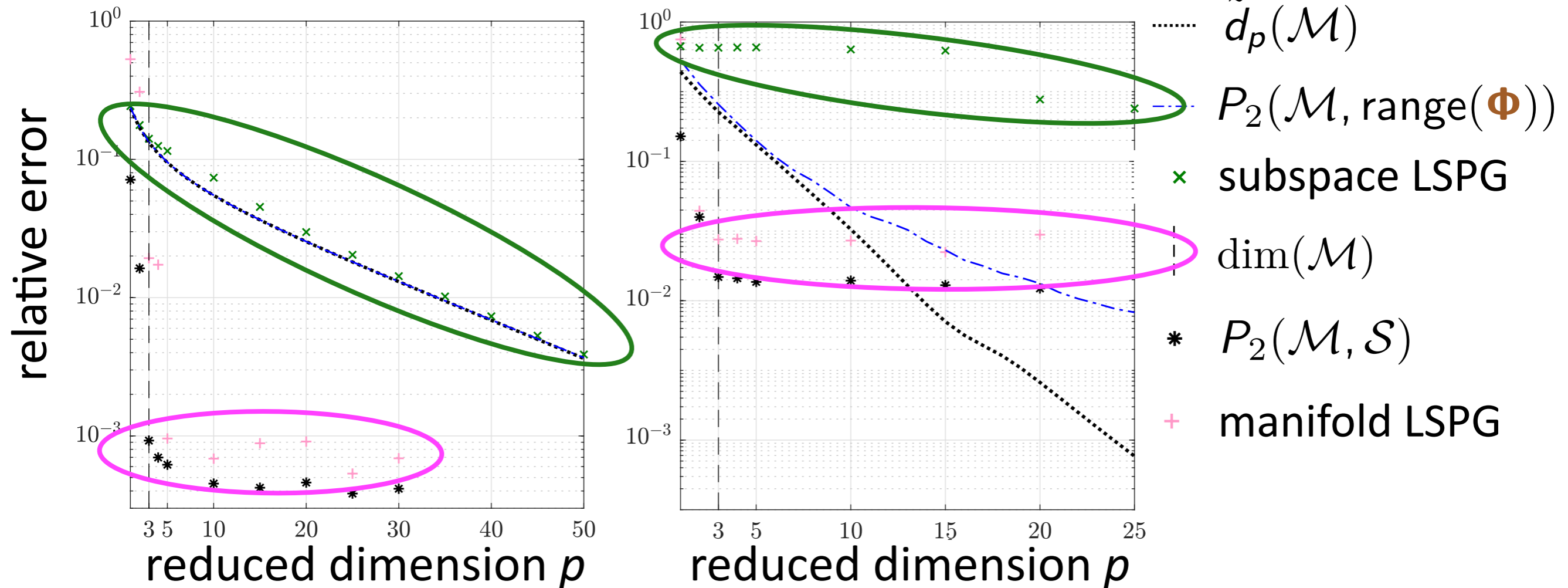


+ Autoencoder manifold **significantly better** than optimal linear subspace

# Method overcomes Kolmogorov-width limitation

1D Burgers' equation

2D reacting flow

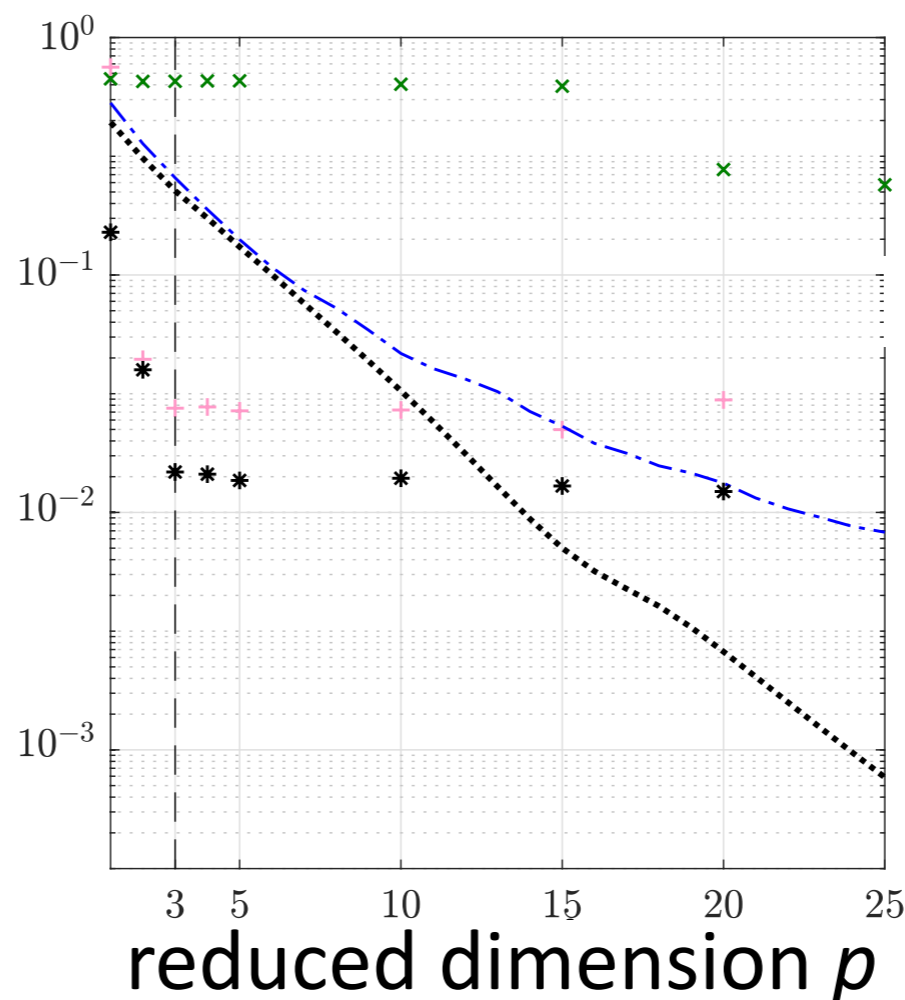
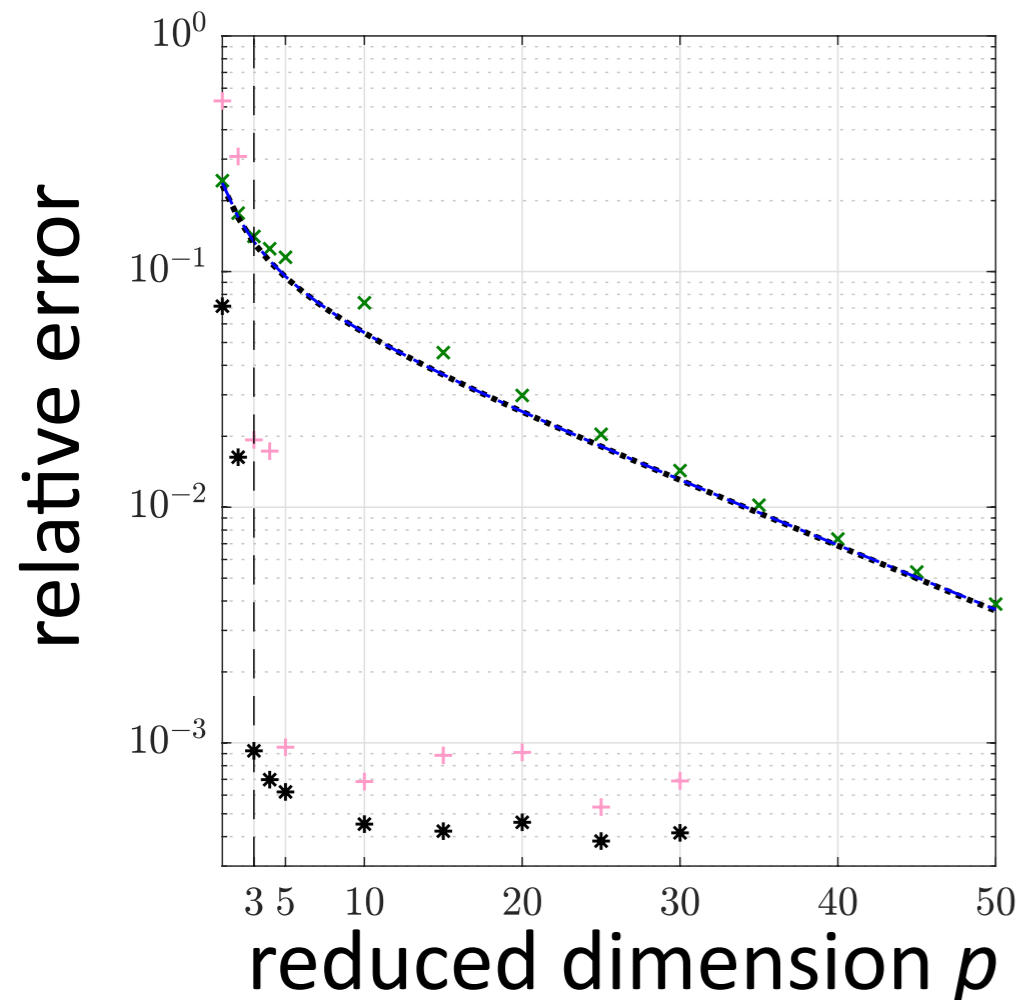


- + Autoencoder manifold significantly better than optimal linear subspace
- + Manifold LSPG orders-of-magnitude more accurate than subspace LSPG

# Method overcomes Kolmogorov-width limitation

1D Burgers' equation

2D reacting flow



- $\cdots \tilde{d}_p(\mathcal{M})$
- $\cdots P_2(\mathcal{M}, \text{range}(\Phi))$
- $\times$  subspace LSPG
- $-$   $\dim(\mathcal{M})$
- $*$   $P_2(\mathcal{M}, \mathcal{S})$
- $+$  manifold LSPG

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

## Manifold Galerkin

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

## Manifold LSPG

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

## Interpretation

- First work demonstrating *physics-constrained* time evolution of codes

## Gradient computation

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

## Forward-compatible extensions

- *Sample mesh*: convolutional layers preserve sparsity
- *Structure preservation*: equality constraints enforcing conservation

## Future work

- Detailed study of architecture, amount of requisite training
- Integration in large-scale code

***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; Parish and C., 2019]

# Model reduction can work well...

*vorticity field*

*pressure field*

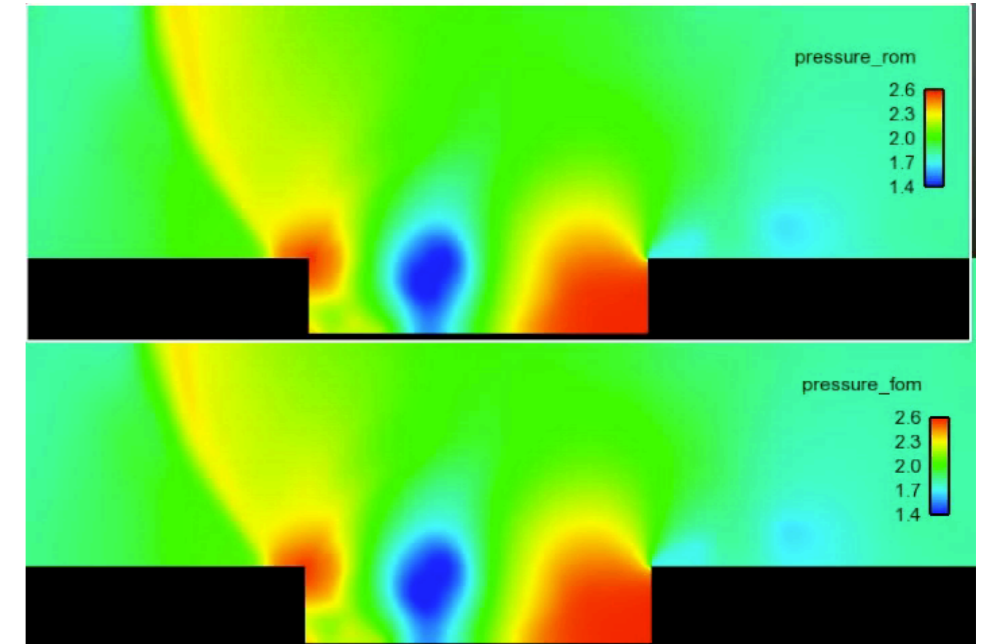
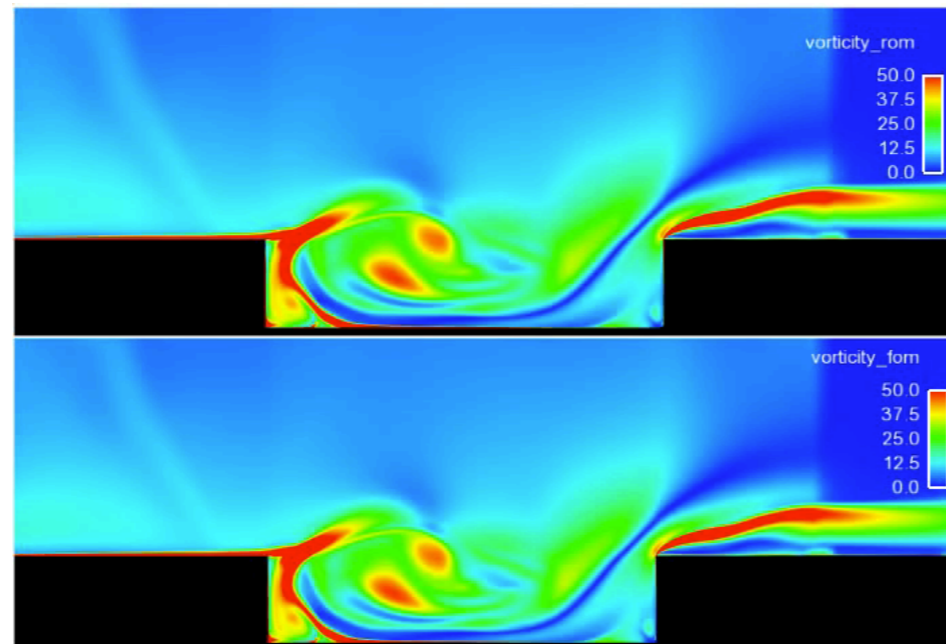
LSPG ROM with

$$\mathbf{A} = (\mathbf{P}\Phi_r)^+ \mathbf{P}$$

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**

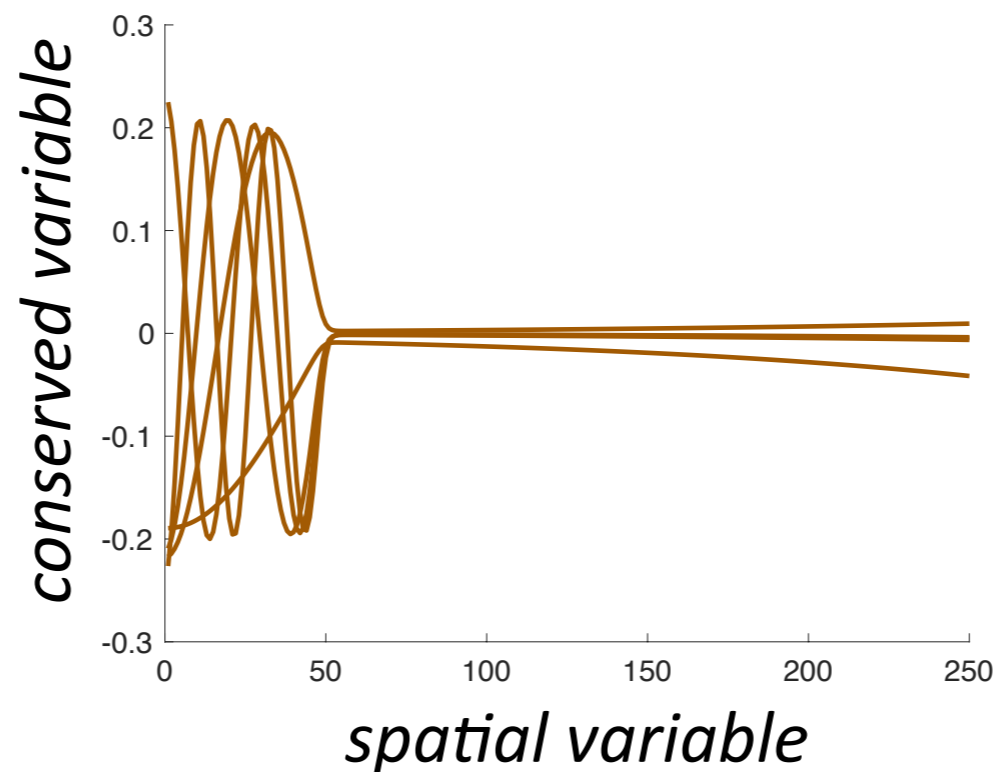
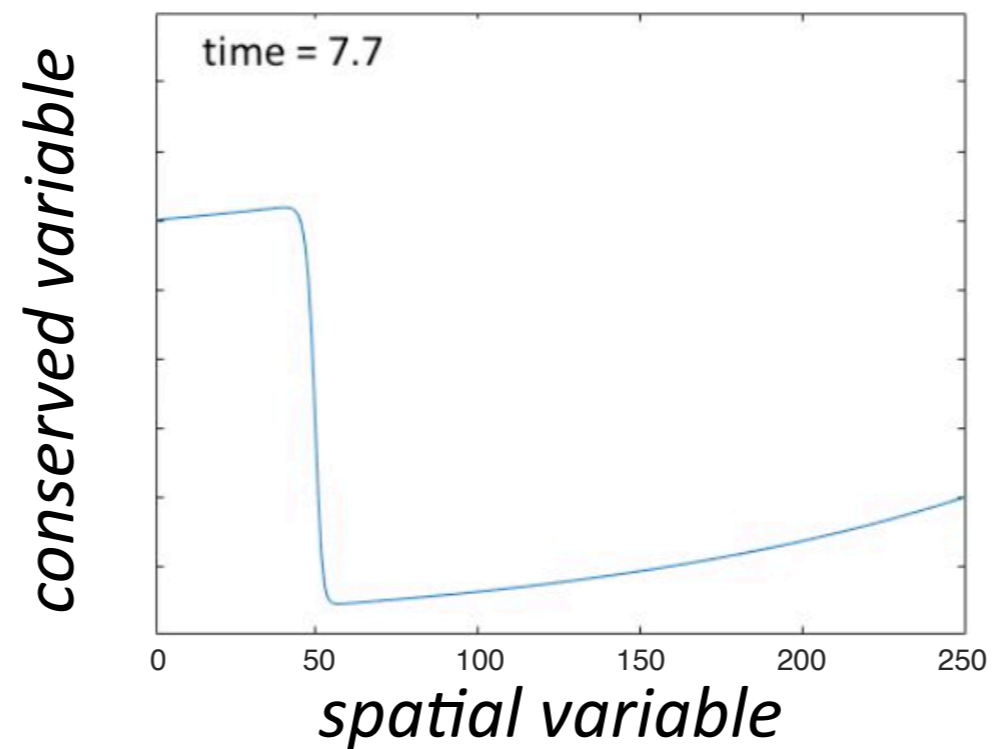
$$\mathbf{x}(t) \approx \Phi \hat{\mathbf{x}}(t)$$

1) *Linear-subspace assumption is strong*

2) *Accuracy limited by information in  $\Phi$*  ←

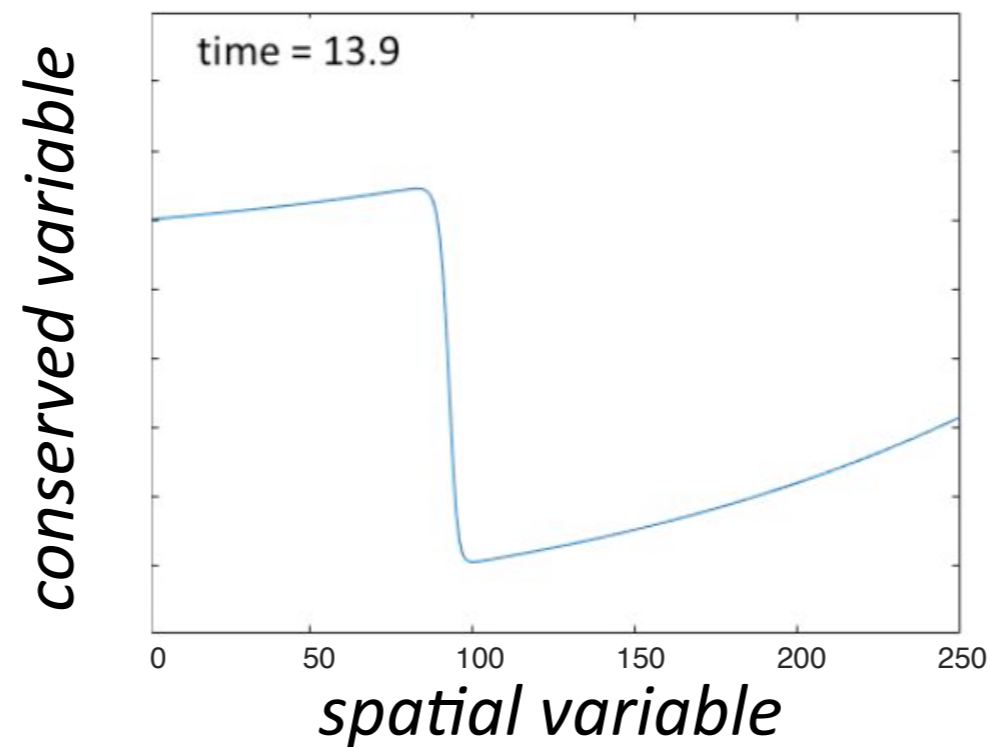
# Illustration: inviscid 1D Burgers' equation

*high-fidelity model*

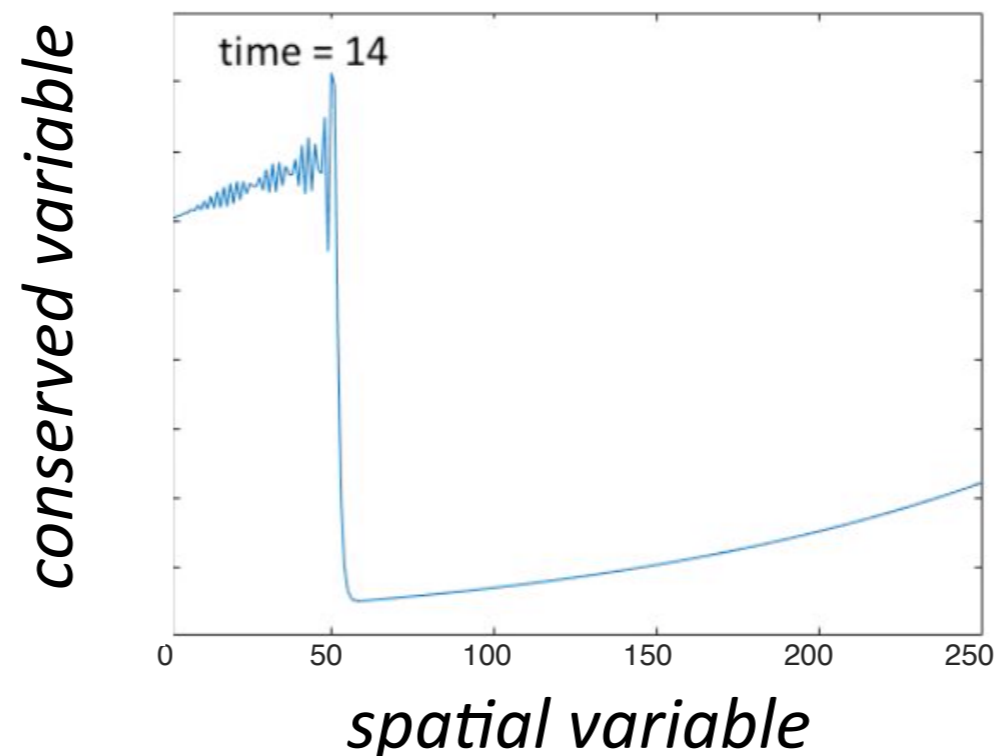


# Illustration: inviscid 1D Burgers' equation

*high-fidelity model*



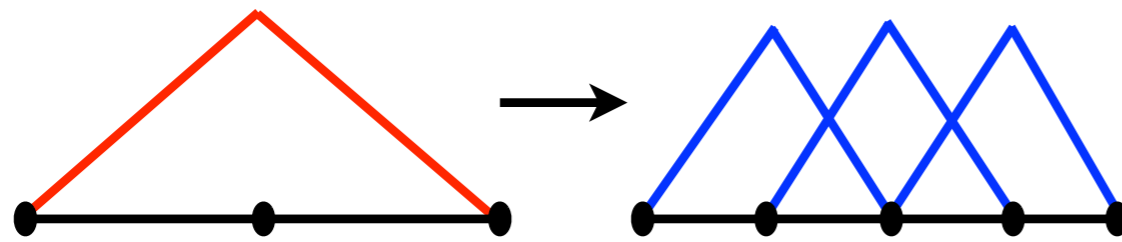
*reduced-order model*



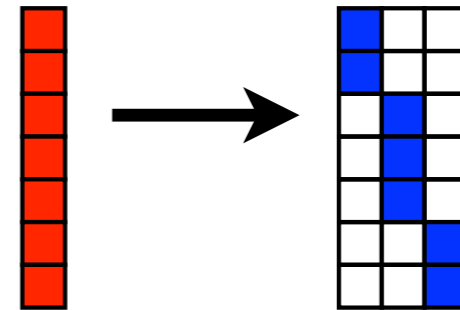
*reduced-order model*  
**inaccurate** when  $\Phi$   
**insufficient**

## *Model-reduction analogue to mesh-adaptive h-refinement*

- ‘Split’ basis vectors



*finite-element  
h-refinement*

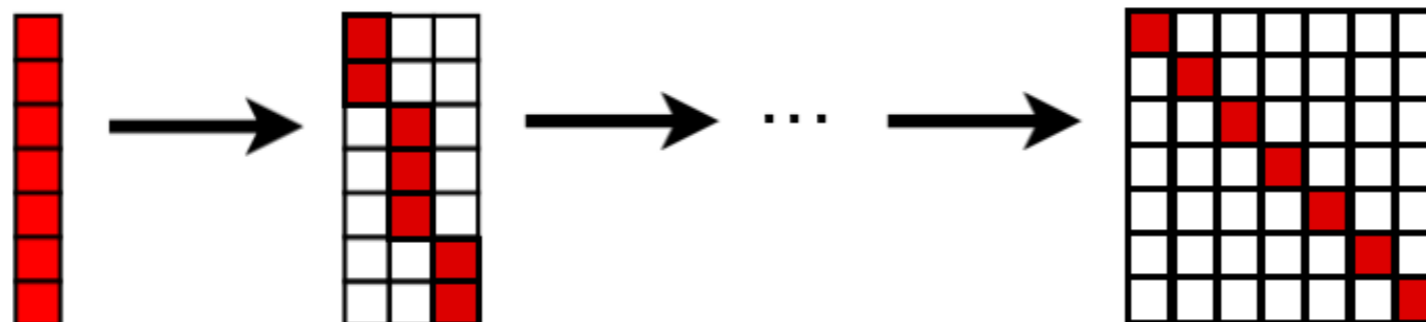


*reduced-order-model  
h-refinement*

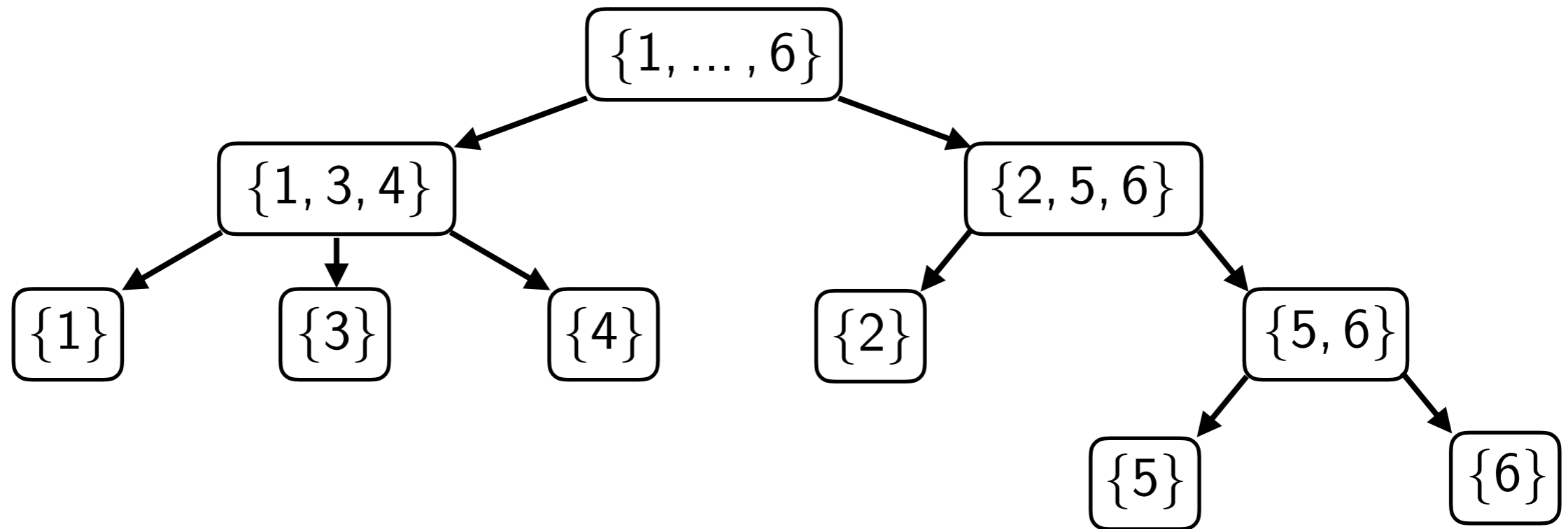
- Generate hierarchical subspaces

$$\text{range} \left( \begin{pmatrix} \text{red column} \end{pmatrix} \right) \subseteq \text{range} \left( \begin{pmatrix} \text{blue grid} \end{pmatrix} \right)$$

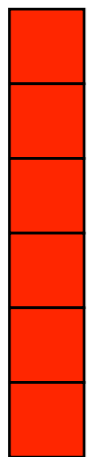
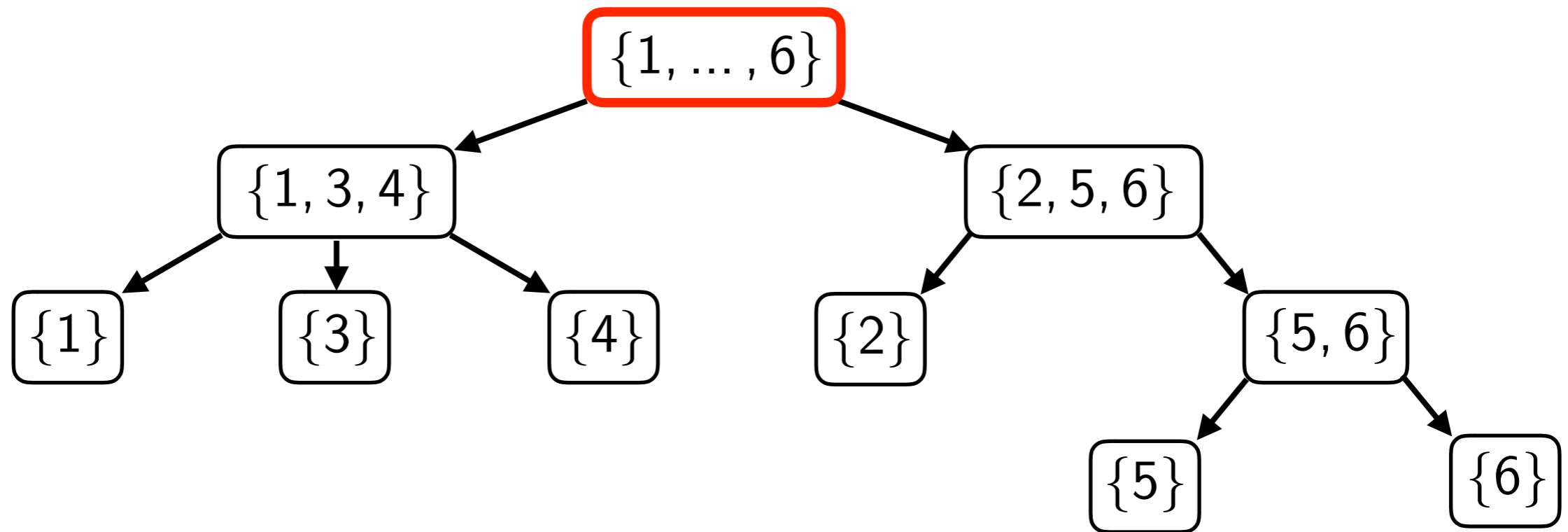
- Converges to the high-fidelity model



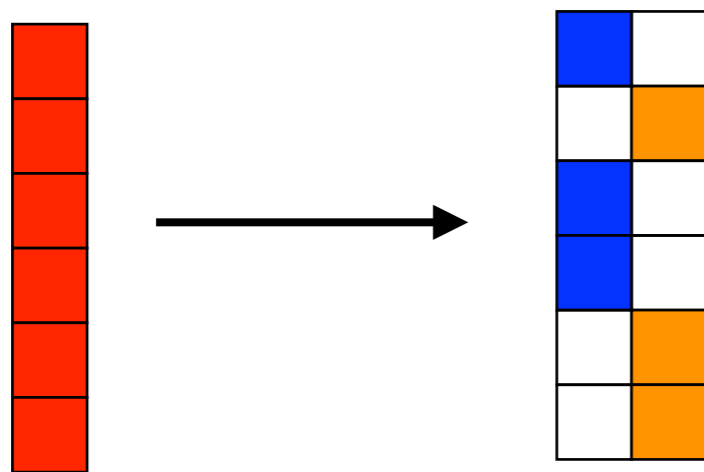
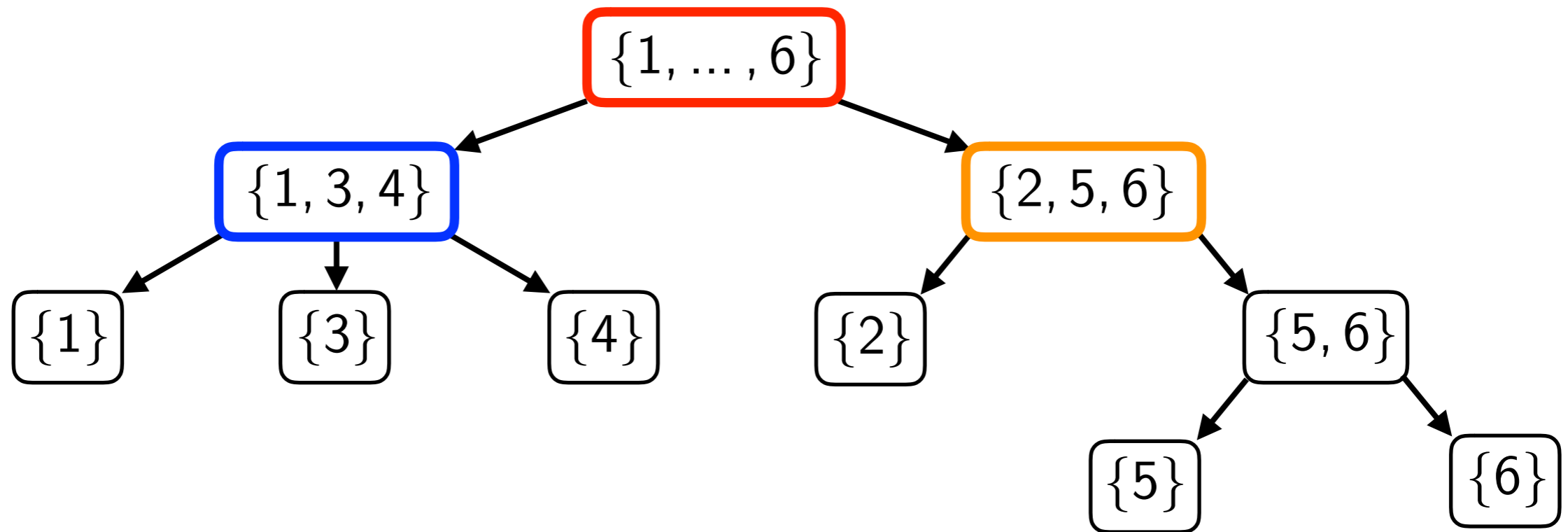
# Refinement tree encodes splitting



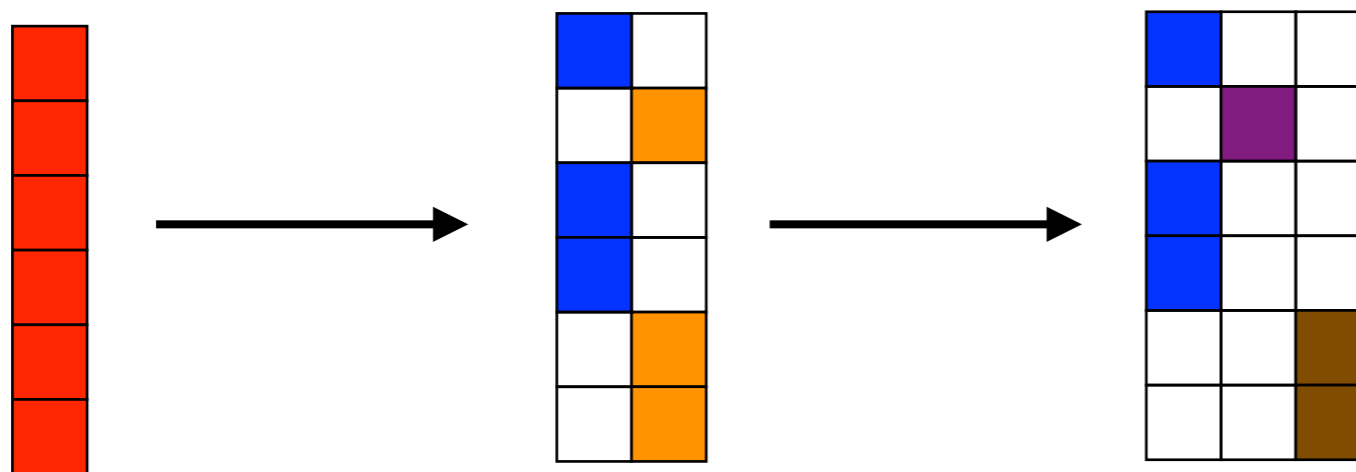
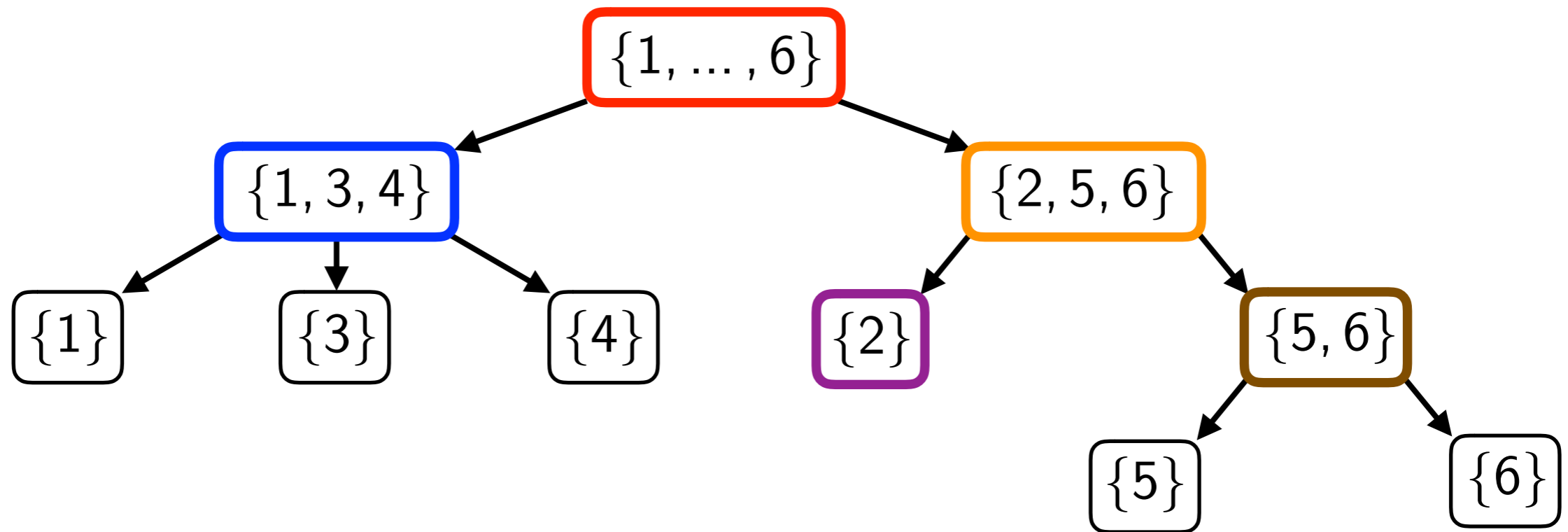
# Refinement tree encodes splitting



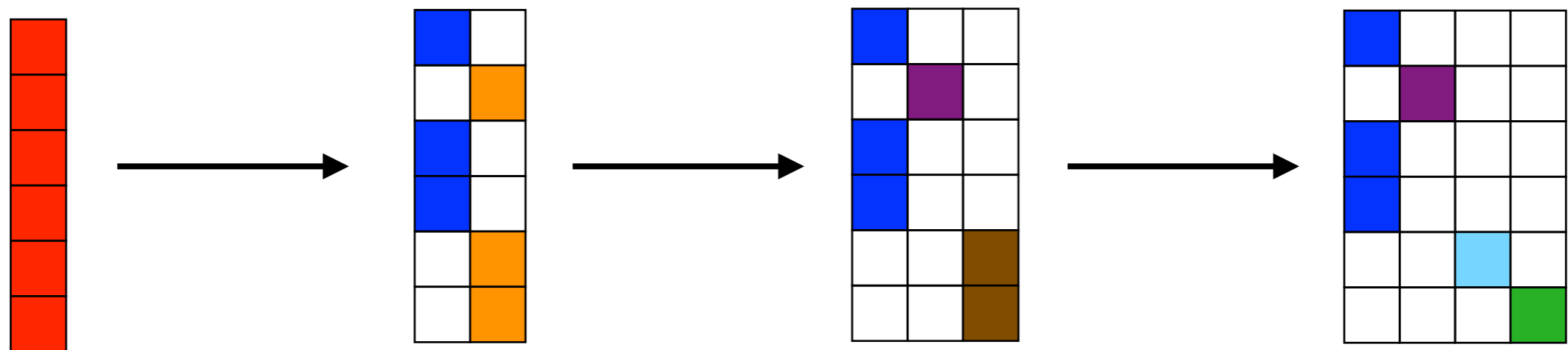
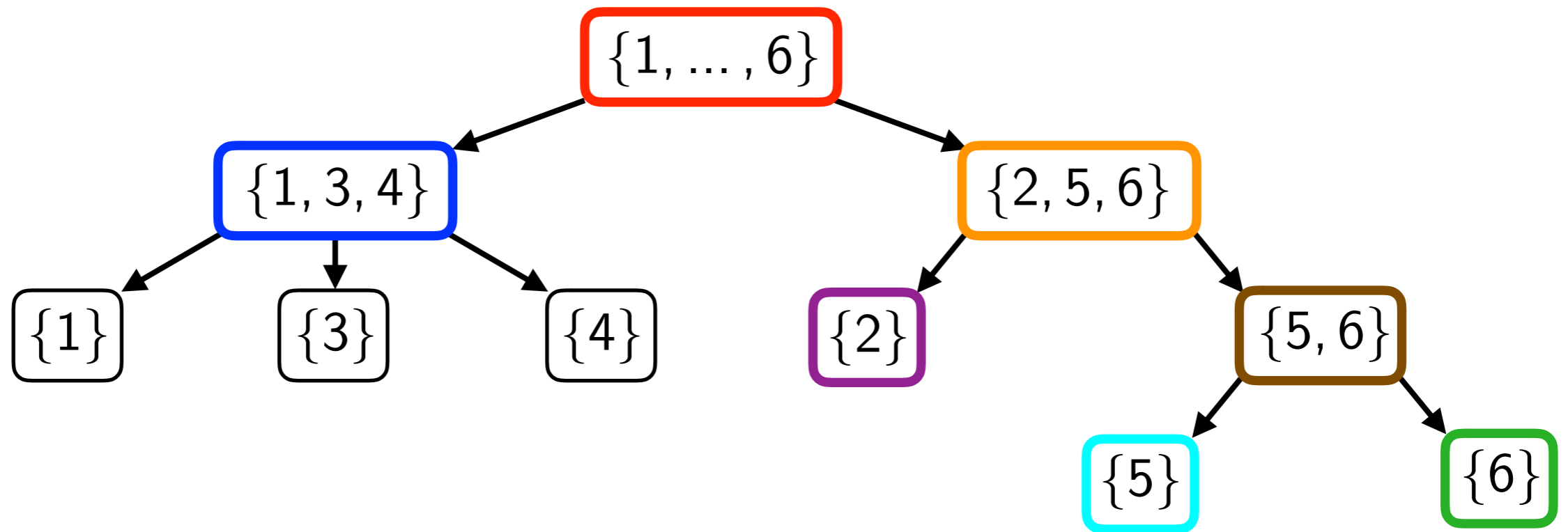
# Refinement tree encodes splitting



# Refinement tree encodes splitting



# Refinement tree encodes splitting

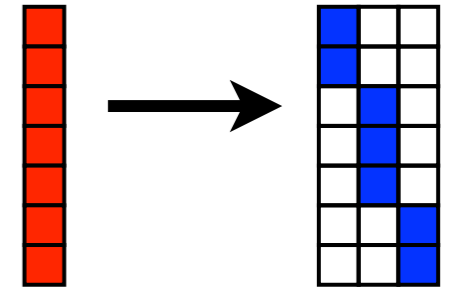


# Tree requirements

## Theorem [C., 2015]

$h$ -adaptivity generates a **hierarchy of subspaces** if:

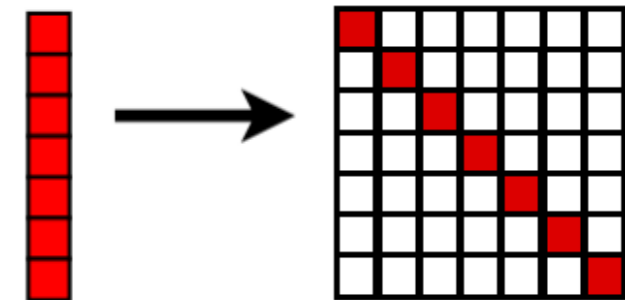
1. children have disjoint support, and
2. the union of the children elements is equal to the parent elements



## Theorem [C., 2015]

$h$ -adaptivity **converges to the high-fidelity model** if:

1. every element has a nonzero entry in  $>1$  basis vector,
2. the root node includes all elements, and
3. each element has a leaf node.



## Tree-construction algorithm

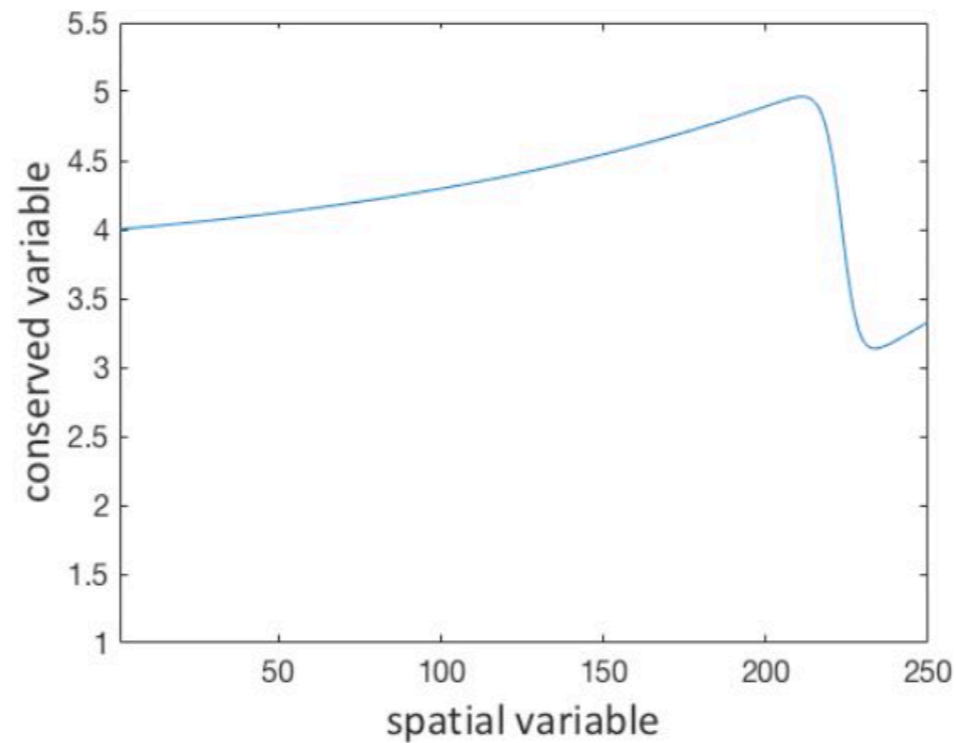
- Identifies hierarchy of correlated states via  $k$ -means clustering
- + Ensures **theorem conditions** are satisfied

## Which vectors to split?

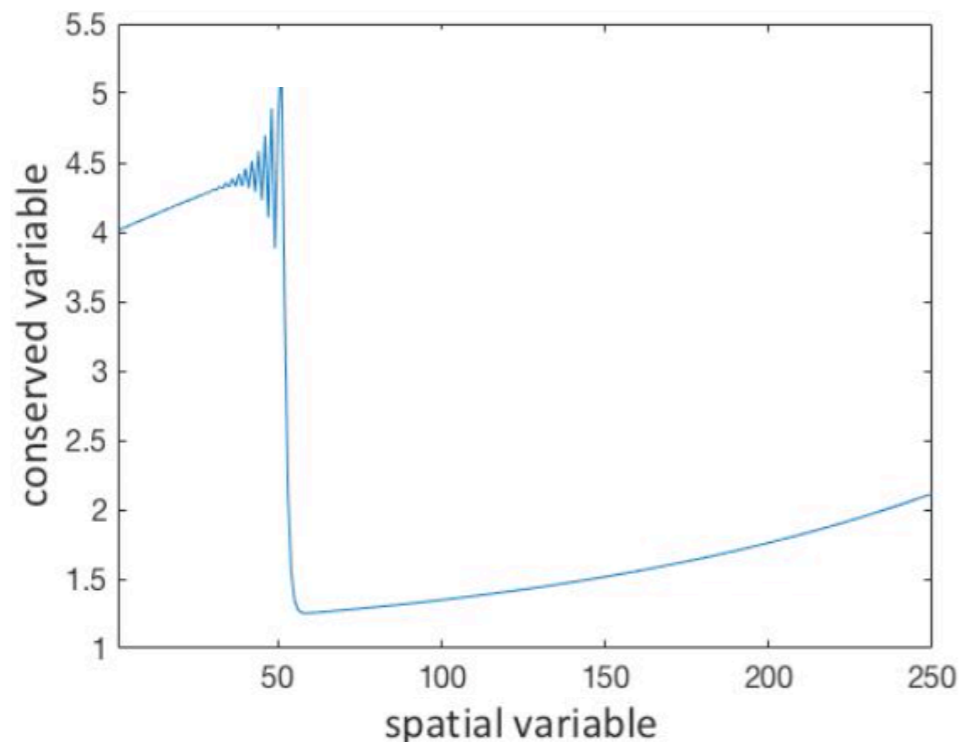
- Dual-weighted-residual error estimation

# Illustration: inviscid 1D Burgers' equation

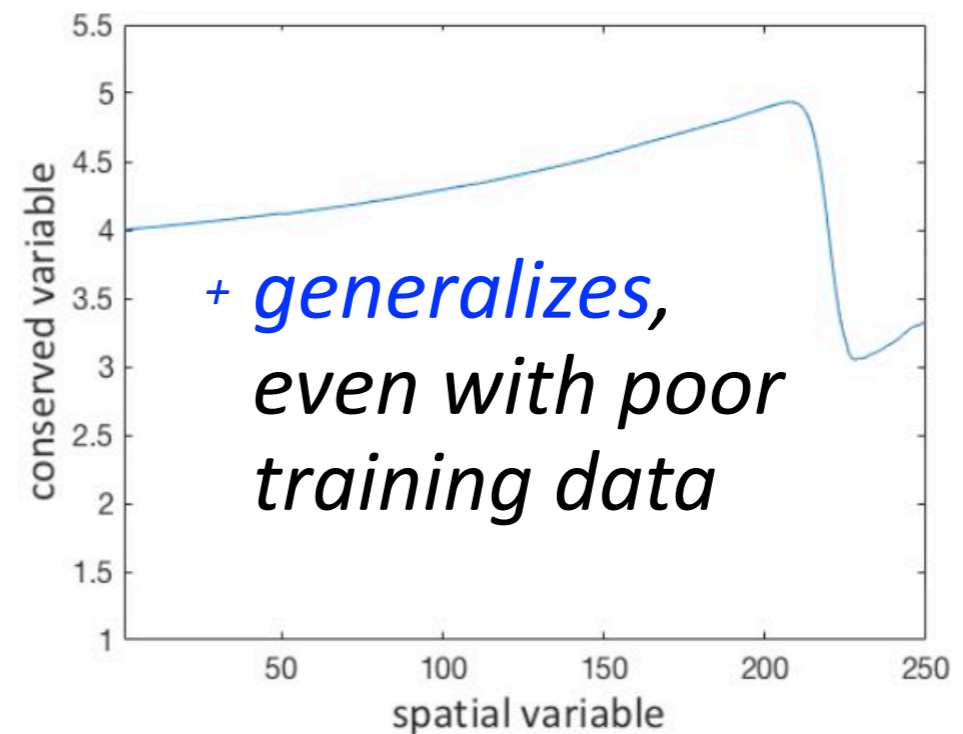
## *high-fidelity model*



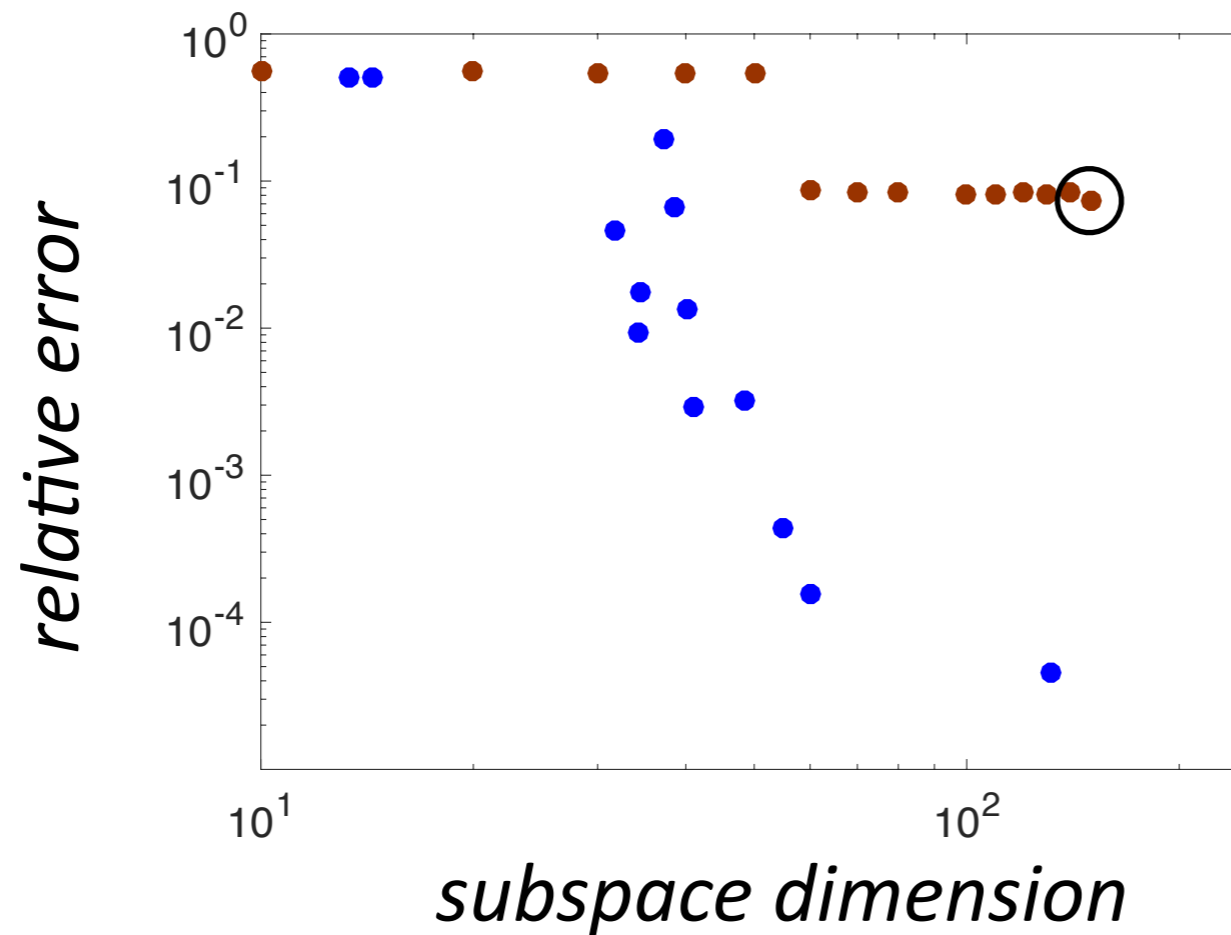
*reduced-order model (dim 50)*



*h-adaptive ROM (mean dim 48.5)*



# *h*-adaptivity provides an accurate, low-dim subspace

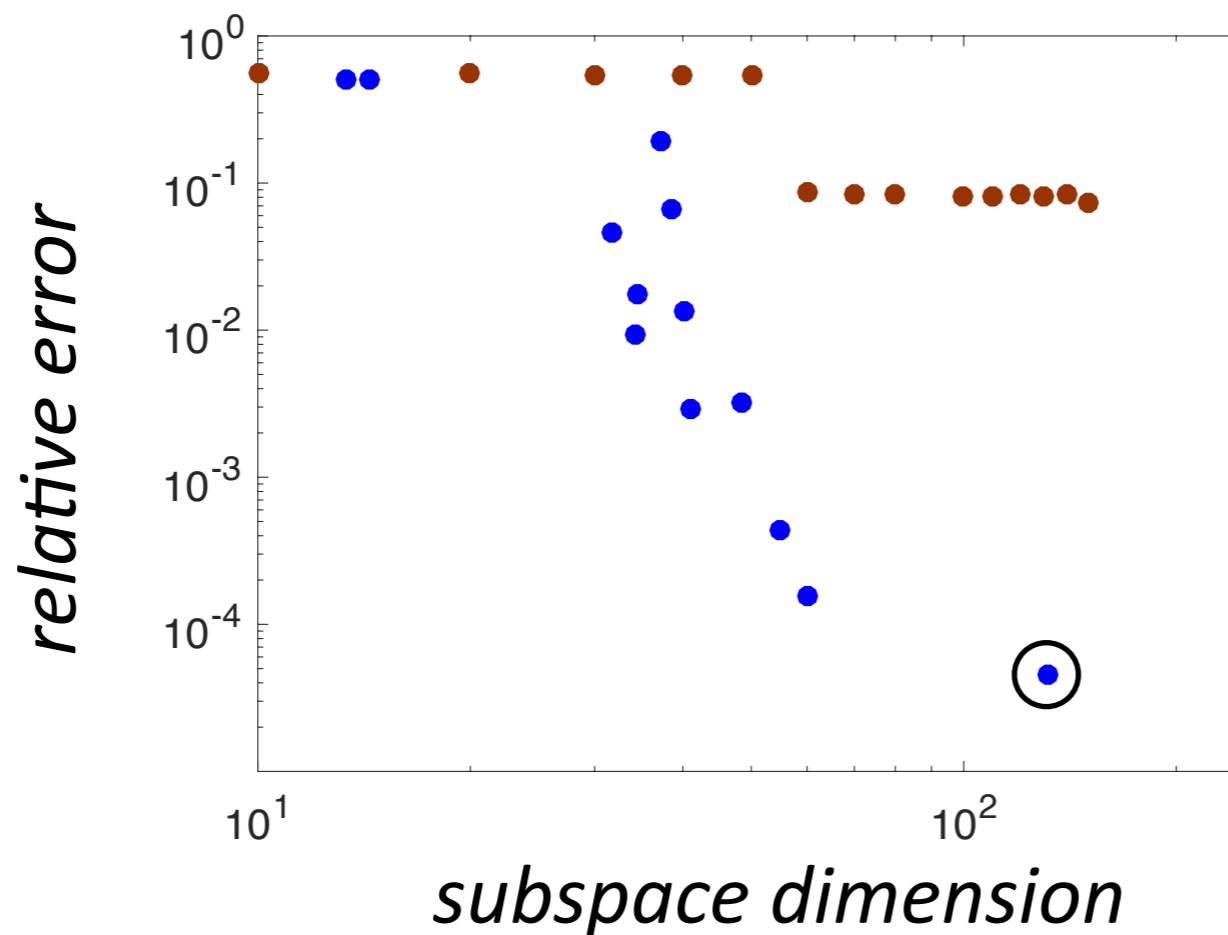


- reduced-order models
- *h*-adaptive ROMs

## ***Reduced-order models***

- minimum error **7.5%**
- **cannot overcome** insufficient training data

# *h*-adaptivity provides an accurate, low-dim subspace



- reduced-order models
- *h*-adaptive ROMs

## **Reduced-order models**

- minimum error **7.5%**
- **cannot overcome** insufficient training data

## ***h*-adaptive ROMs**

- + minimum error  **$<0.01\%$**  with **lower subspace dimension**
- + **generalizes** if insufficient training data
- + can satisfy **any prescribed error tolerance**

***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; Parish and C., 2019]

# Discrete-time error bound

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

If the following conditions hold:

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

$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_G^n\|_2 \leq \frac{1}{h} \|\mathbf{r}_G^n(\Phi \hat{\mathbf{x}}_G^n)\|_2 + \frac{1}{h} \sum_{\ell=1}^k |\alpha_\ell| \|\mathbf{x}^{n-\ell} - \Phi \hat{\mathbf{x}}_G^{n-\ell}\|_2$$

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

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

# Discrete-time error bound

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

If the following conditions hold:

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

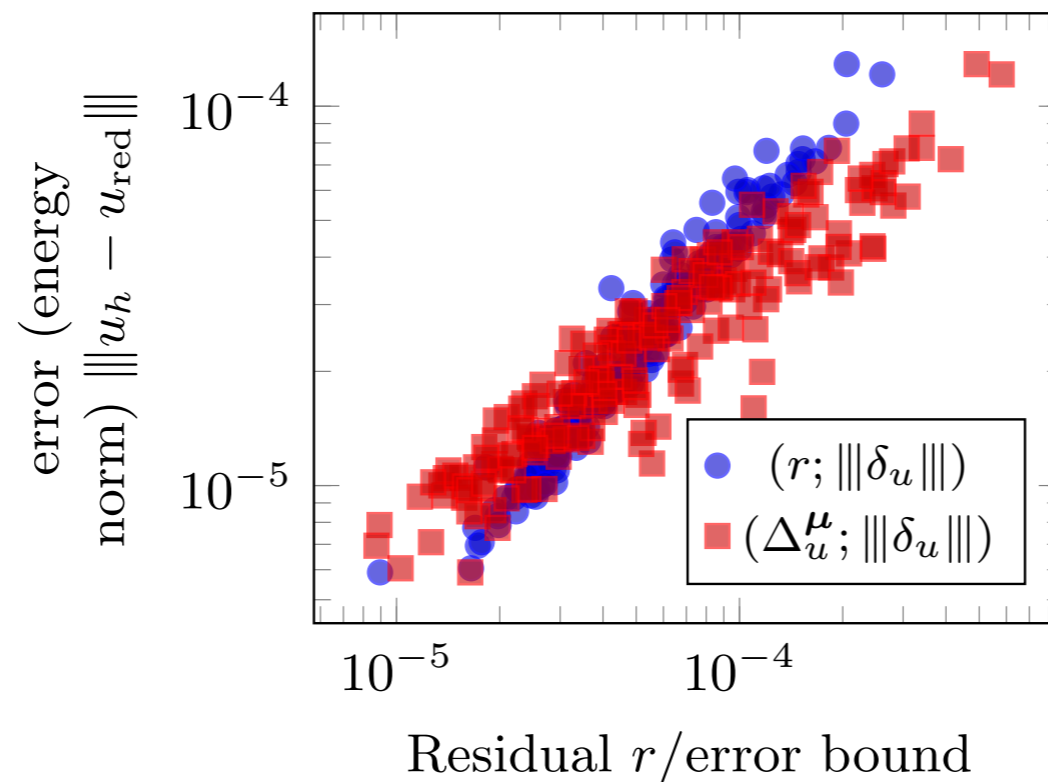
$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_G^n\|_2 \leq \frac{\gamma_1 (\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j \in \{1, \dots, N\}} \|\mathbf{r}_G^j(\Phi \hat{\mathbf{x}}_G^j)\|_2$$
$$\|\mathbf{x}^n - \Phi \hat{\mathbf{x}}_{\text{LSPG}}^n\|_2 \leq \frac{\gamma_1 (\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j \in \{1, \dots, N\}} \min_{\hat{\mathbf{v}}} \|\mathbf{r}_{\text{LSPG}}^j(\Phi \hat{\mathbf{v}})\|_2$$

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

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

# Main idea

- **Observation:** residual-based quantities are **informative** of the error



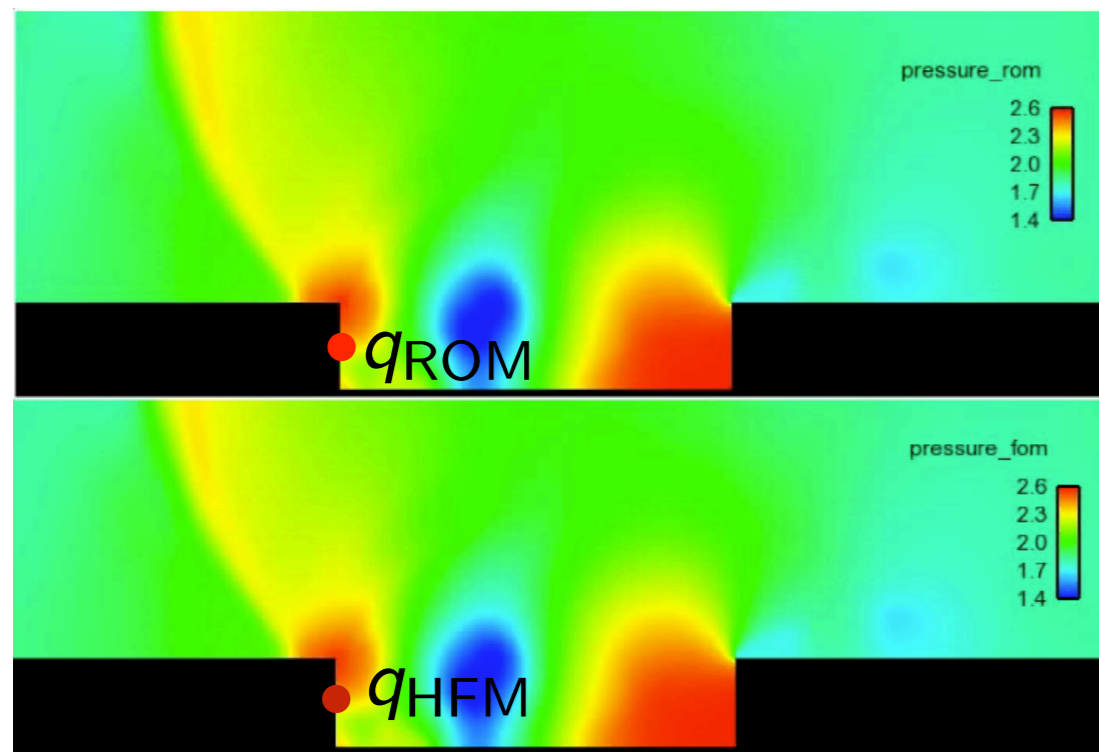
- **ML perspective:** these are **good features** for predicting the error

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

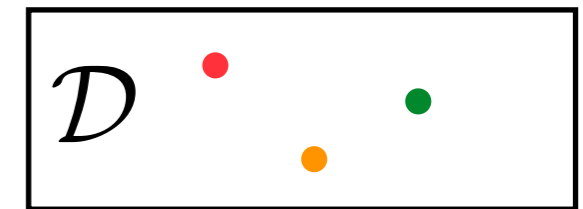
**Machine-learning error models** [Freno and C., 2019]

# Training and machine learning: error modeling

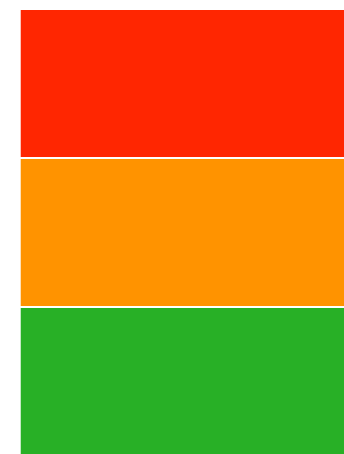
1. *Training*: Solve high-fidelity and reduced-order models for  $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning*: Construct regression model
3. *Reduction*: predict reduced-order-model error for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



$$q_{\text{HFM}}^n - q_{\text{ROM}}^n$$



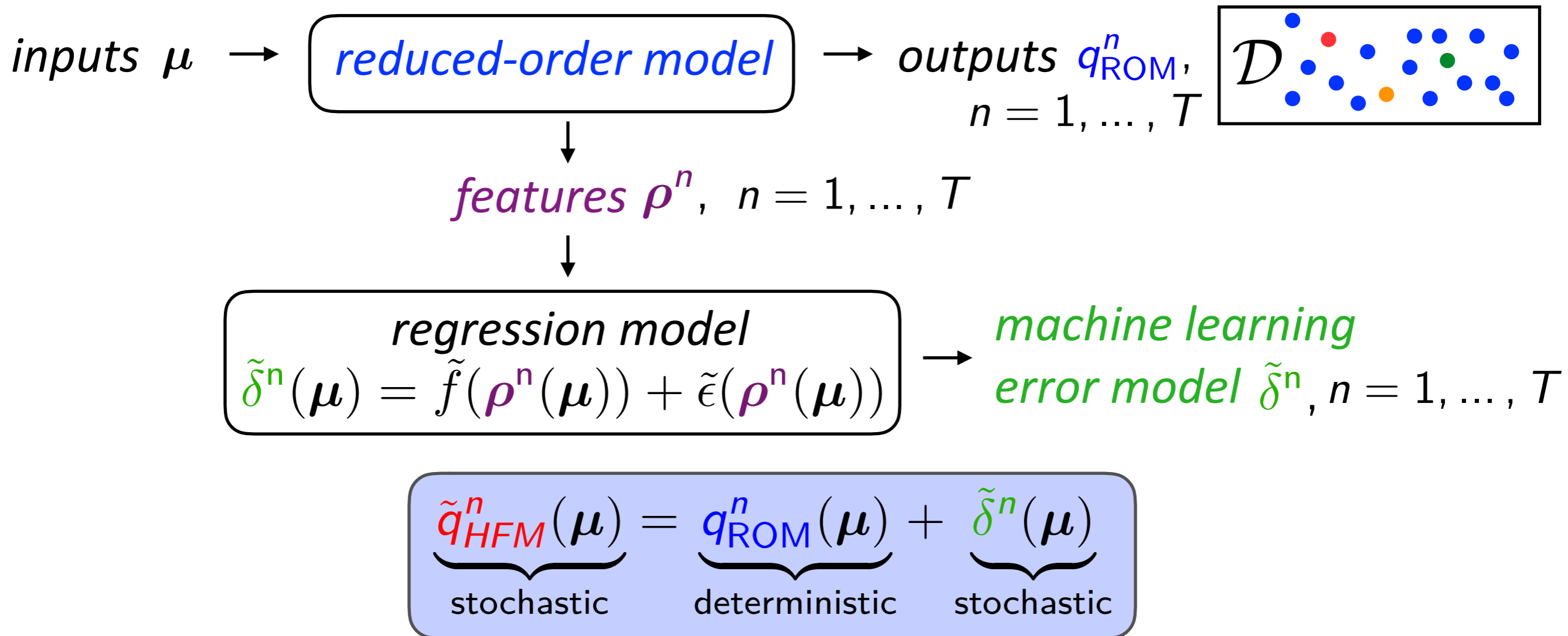
$$\rho^n$$



- ▶ randomly divide data into (1) training data and (2) testing data
- ▶ construct regression-function model  $\tilde{f}$  via cross validation on **training data**
- ▶ construct noise model  $\tilde{\epsilon}$  from sample variance on **test data**

# Reduction

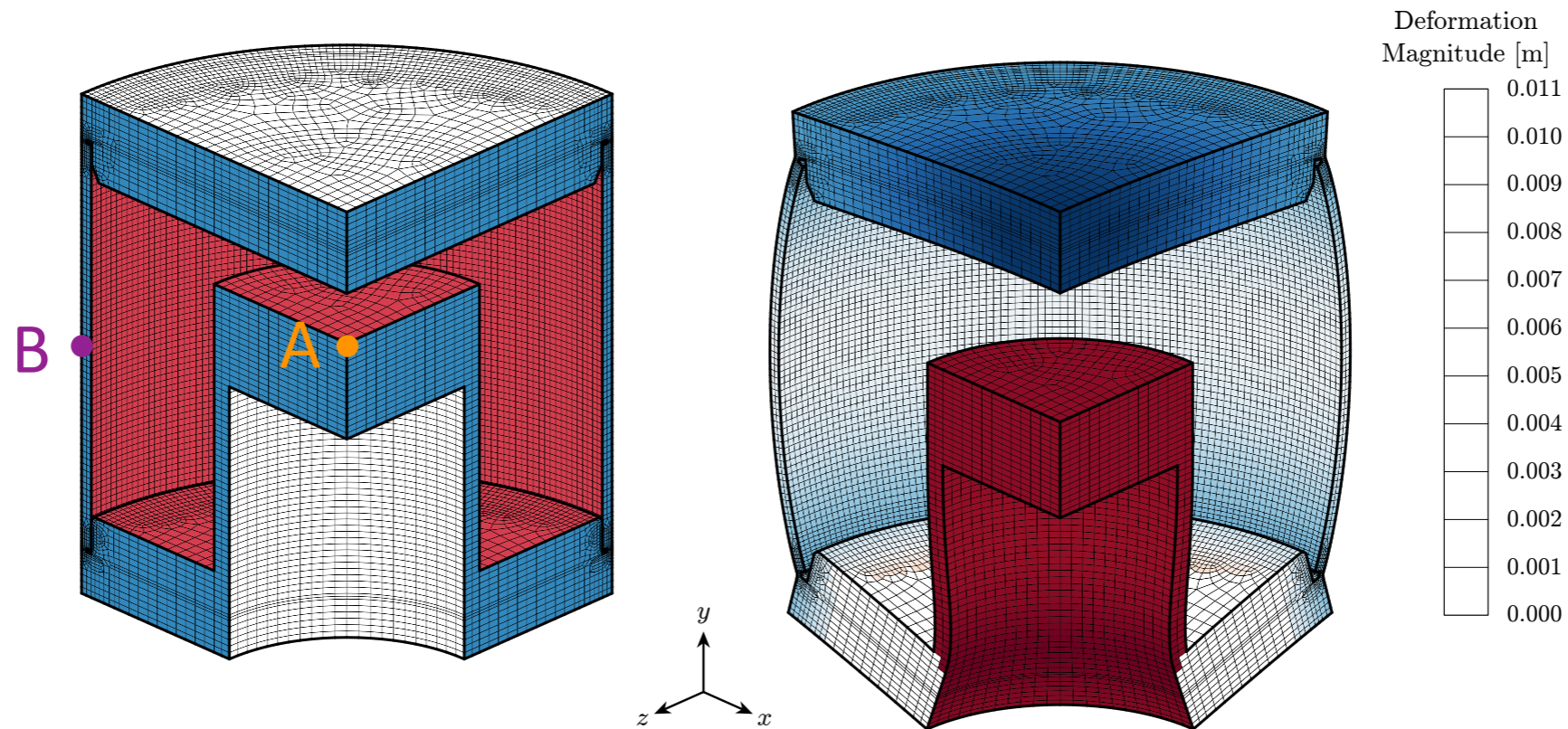
1. *Training*: Solve high-fidelity and reduced-order models for  $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning*: Construct regression model
3. *Reduction*: predict reduced-order-model error for  $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



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

***Use rigorous error analysis to engineer features  $\rho^n$***

# Application: Predictive capability assessment project



- *high-fidelity model dimension:*  $2.8 \times 10^5$
- *reduced-order model dimensions:*  $1, \dots, 5$
- *inputs  $\mu$ :* elastic modulus, Poisson ratio, applied pressure
- *quantities of interest:*  $y$ -displacement at A, radial displacement at B
- *training data:* 150 training examples, 150 testing examples

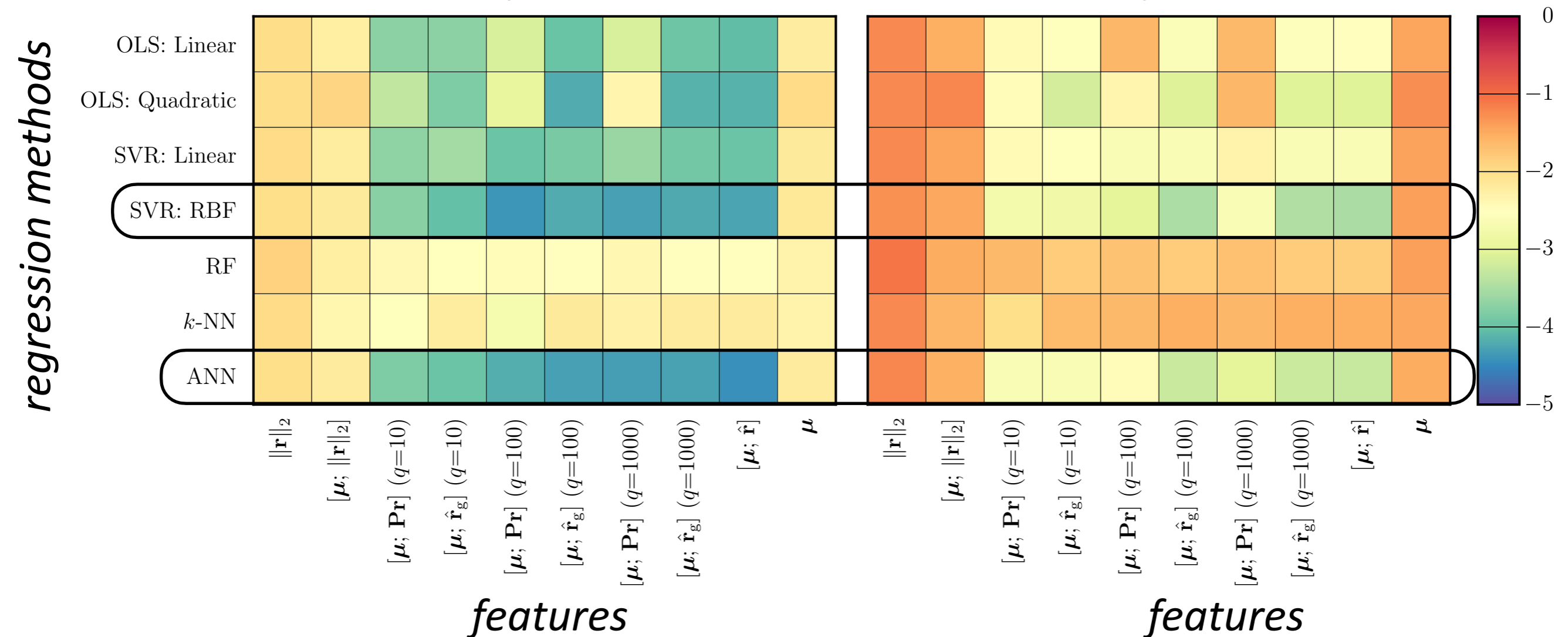
# Application: Predictive capability assessment project

y-displacement at **A**

$$\log_{10}(1 - R^2)$$

radial displacement at **B**

$$\log_{10}(1 - R^2)$$



+ regression methods: neural networks and SVR: RBF **most accurate**

# Application: Predictive capability assessment project

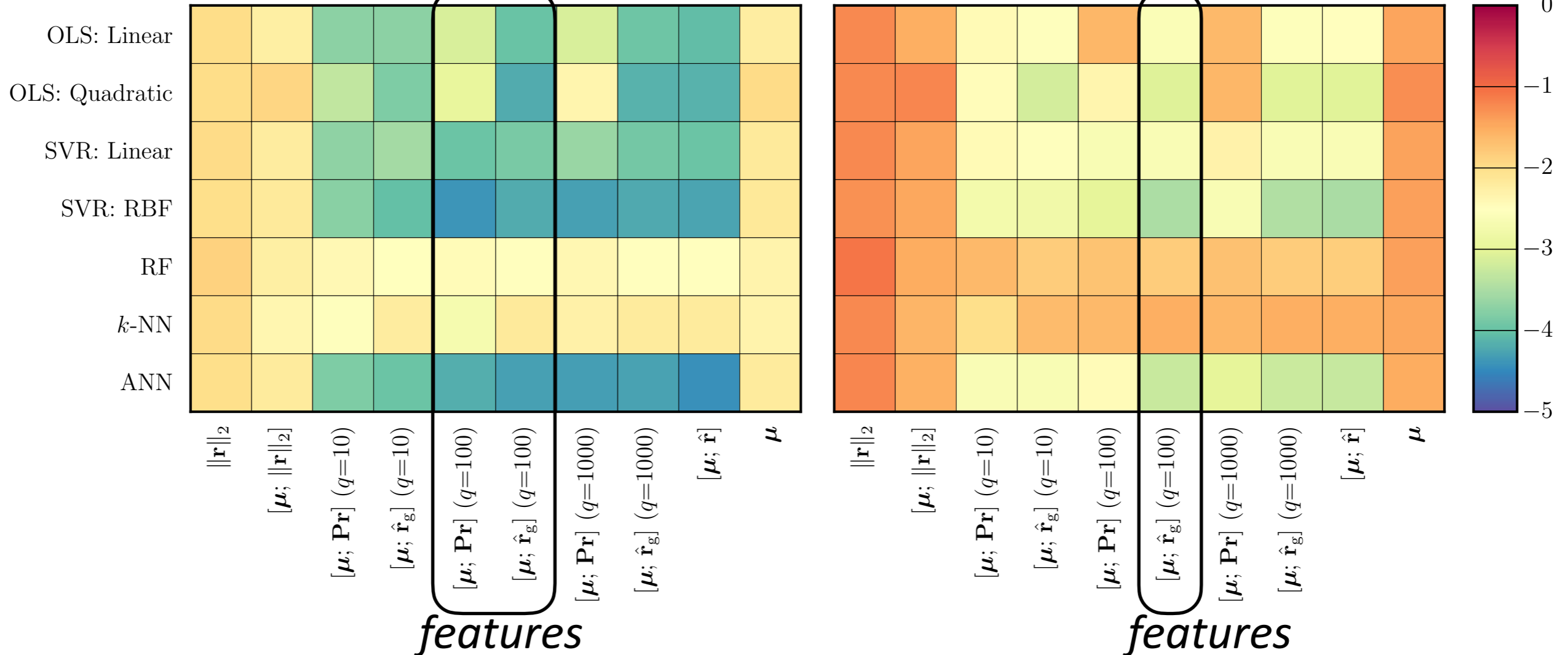
y-displacement at **A**

$$\log_{10}(1 - R^2)$$

radial displacement at **B**

$$\log_{10}(1 - R^2)$$

regression methods

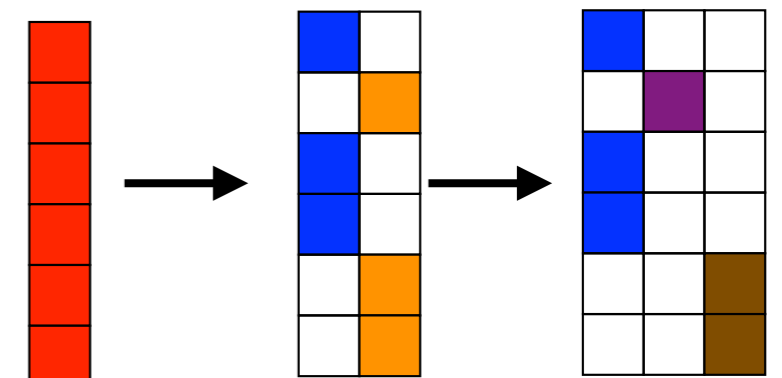
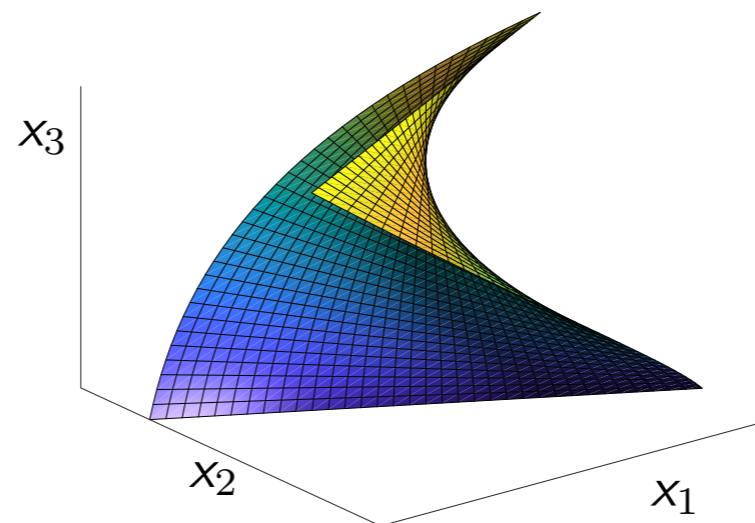
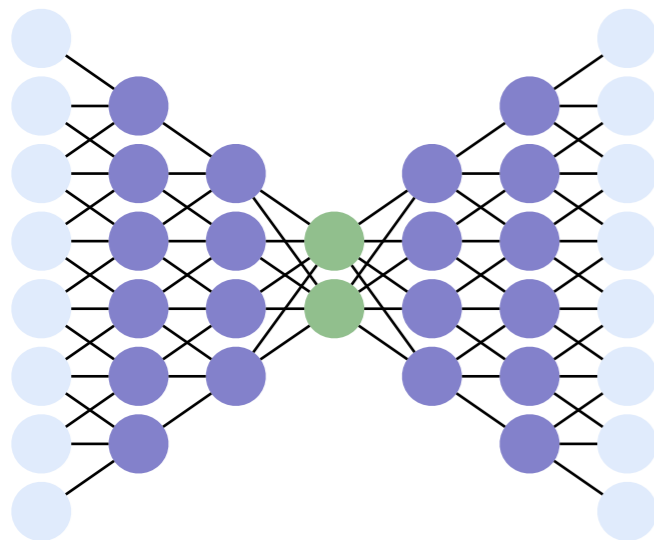
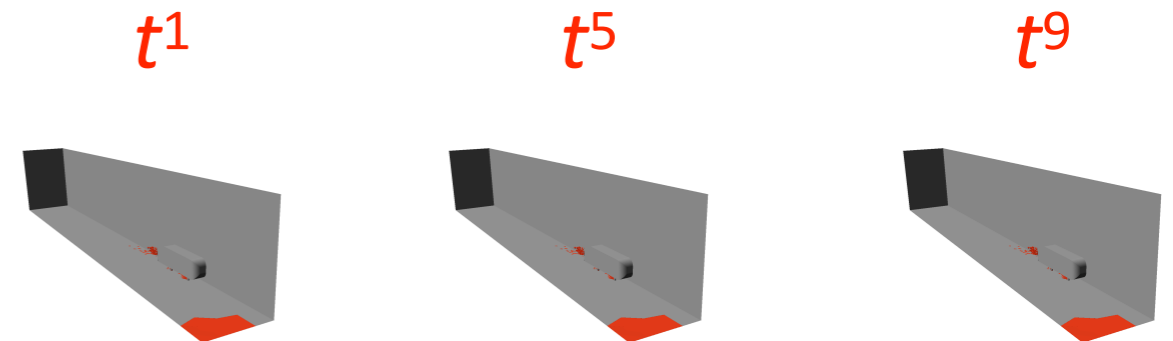
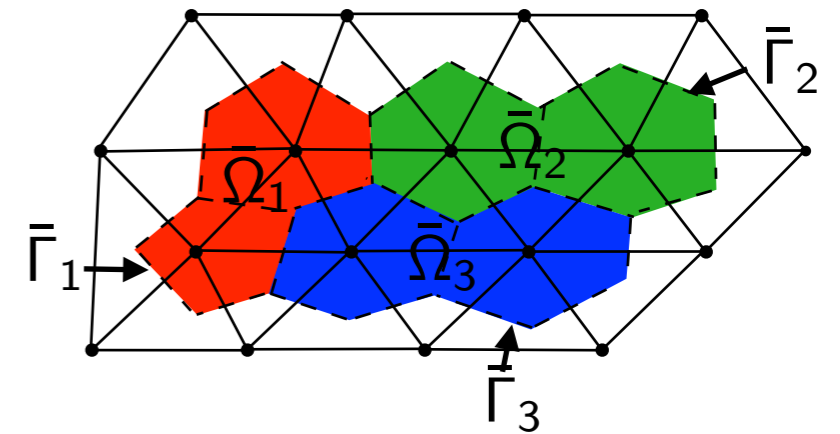
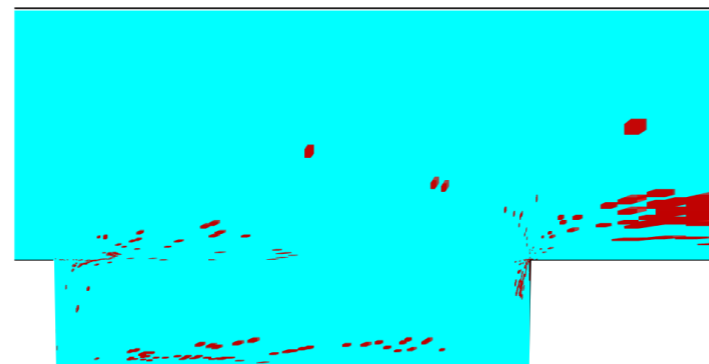
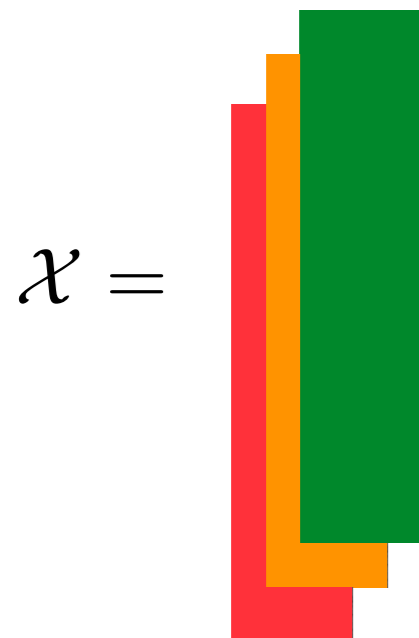


- + regression methods: neural networks and SVR: RBF most accurate
- + features: only 100 residual samples needed for good accuracy

## ***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; Parish and C., 2019]

# Questions?



*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*