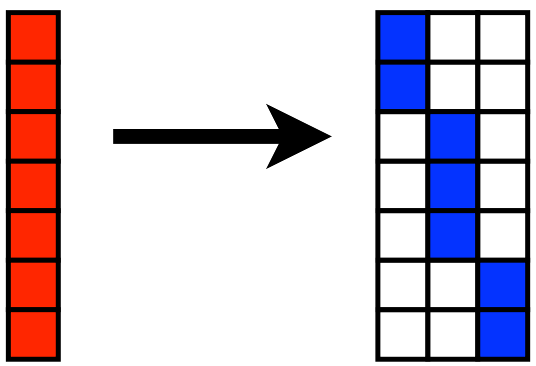
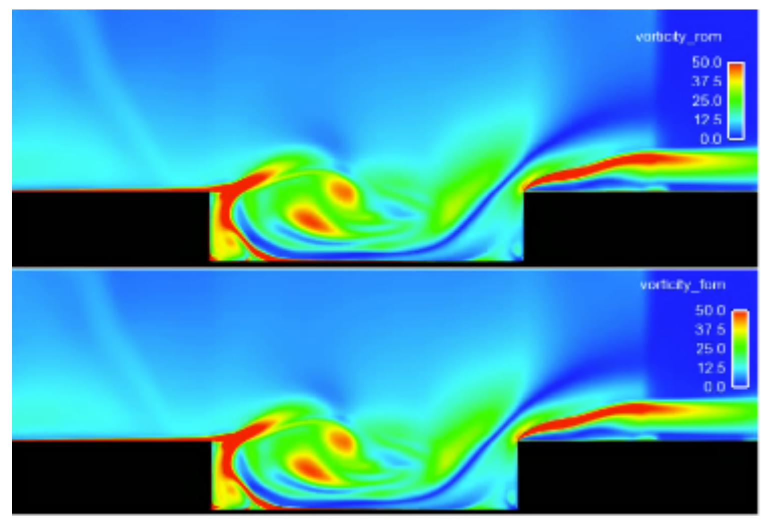
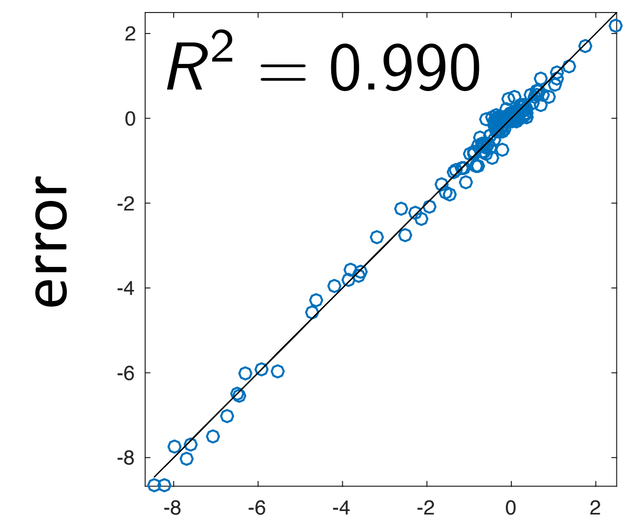
Reduced-order modeling

Using machine learning to enable large-scale simulations for many-query problems







support vector machine
error prediction

Kevin Carlberg

Sandia National Laboratories

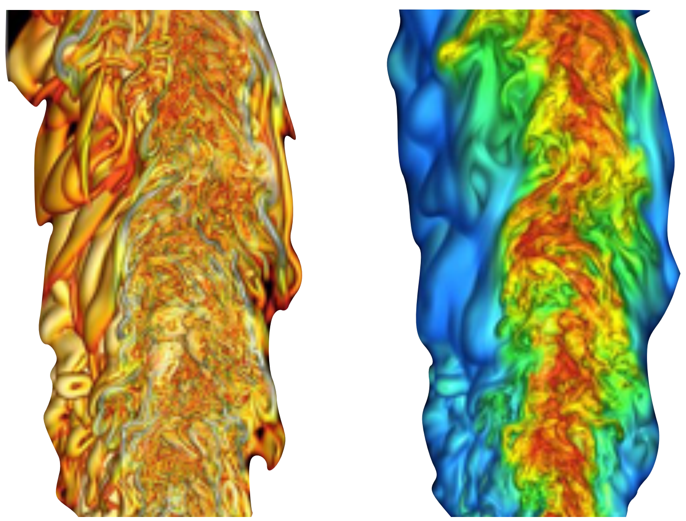
Advanced Modeling & Simulation Seminar Series

NASA Ames Research Center

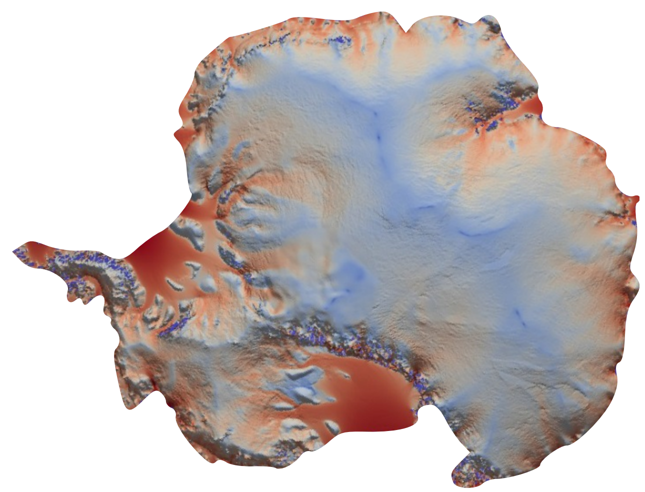
March 29, 2018

High-fidelity simulation

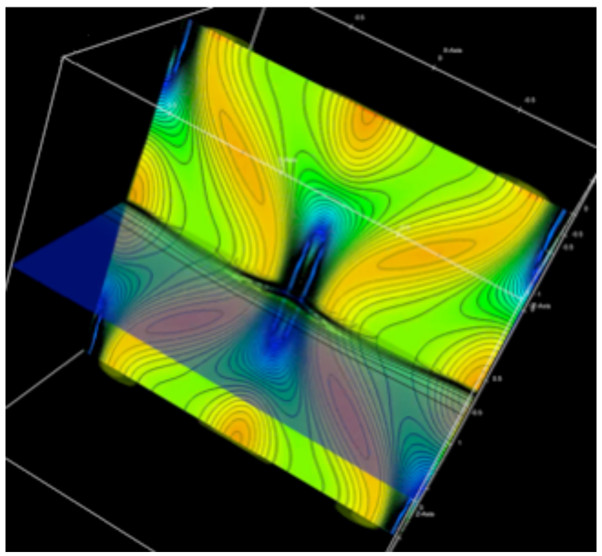
+ Indispensable across science and engineering
- *High fidelity*: extreme-scale nonlinear dynamical system models



Turbulent reacting flows
courtesy J. Chen, Sandia



Antarctic ice sheet modeling
courtesy R. Tuminaro, Sandia



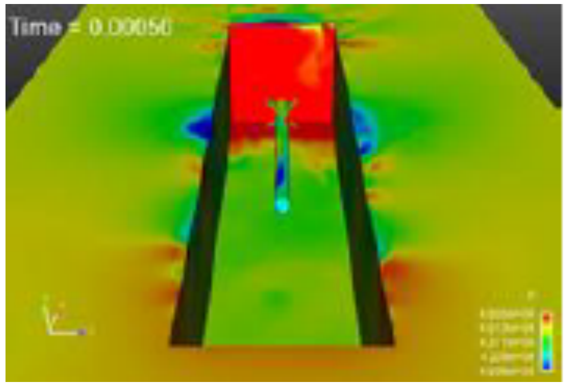
Magnetohydrodynamics
courtesy J. Shadid, Sandia

computational barrier

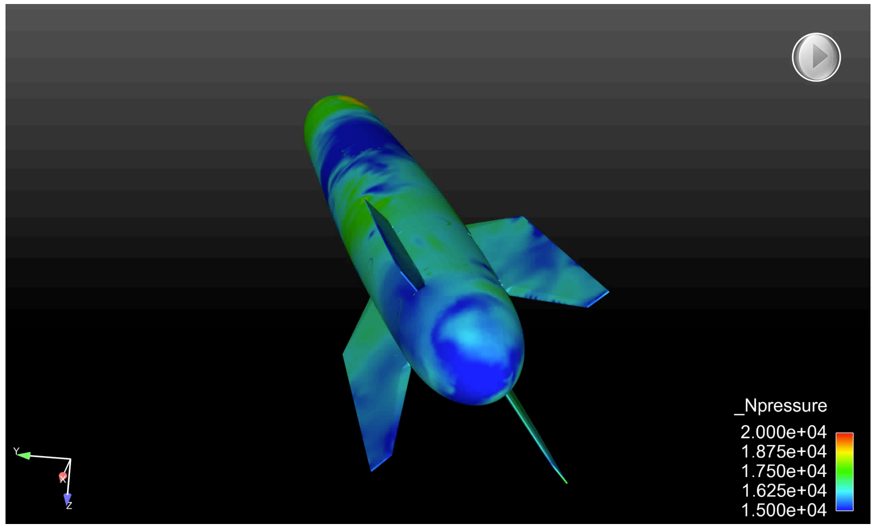
Many-query problems

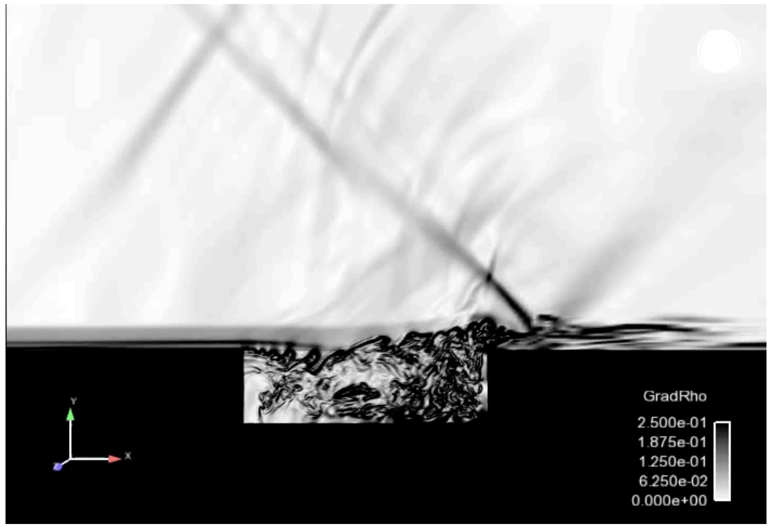
- uncertainty propagation
- Bayesian inference
- multi-objective optimization
- stochastic optimization

High-fidelity simulation: B61 LEP captive carry



High-fidelity simulation: B61 LEP 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

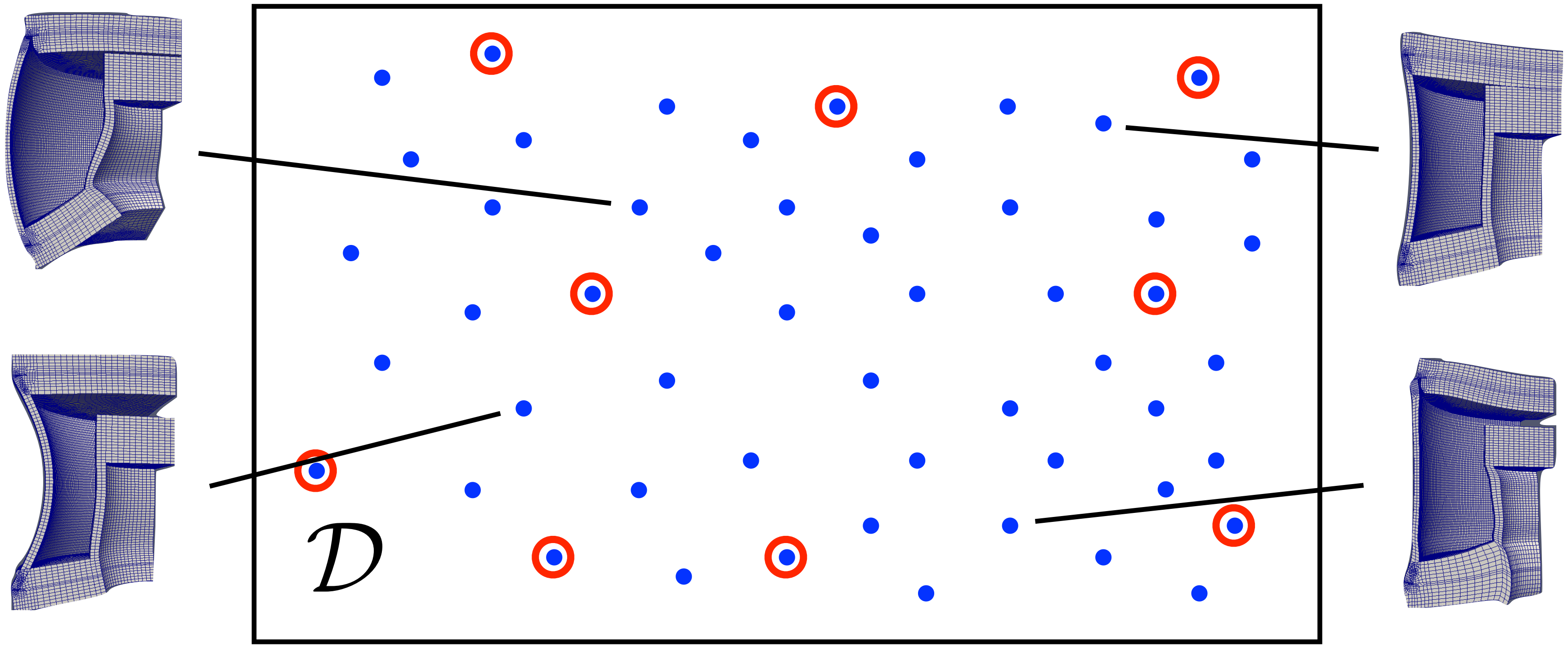
Many-query problems

- explore flight envelope
- quantify effects of uncertainties on store load
- robust design of store and cavity

Approach: exploit simulation data

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

***Many-query problem**: 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

1. ***Accuracy:*** achieves less than 1% error
2. ***Low cost:*** achieves at least 100x computational savings
3. ***Structure preservation:*** preserves important physical properties
4. ***Reliability:*** guaranteed satisfaction of any error tolerance (fail safe)
5. ***Certification:*** quantifies ROM-induced epistemic uncertainty

Model reduction: previous state of the art

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*, *reliable*, *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*, *reliable*, *certified*: sharp *a priori* error bounds, convergence

\+ *Inexpensive*: pre-assemble operators

\+ *Structure preservation*: preserve operator properties

Nonlinear dynamical systems: ineffective

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

\- *Inaccurate, unreliable*: often unstable

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

\- *Expensive*: projection insufficient for speedup

\- *Structure not preserved*: dynamical-system properties ignored

Model reduction: previous state of the art

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*, *reliable*, *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*, *reliable*, *certified*: sharp *a priori* error bounds, convergence

\+ *Inexpensive*: pre-assemble operators

\+ *Structure preservation*: preserve operator properties

Nonlinear dynamical systems: ineffective

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

– *Inaccurate, unreliable*: often unstable

– *Not certified*: error bounds grow exponentially in time

– *Expensive*: projection insufficient for speedup

– *Structure not preserved*: dynamical-system properties ignored

Our research

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

- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity
  [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models
  [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

Our research

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

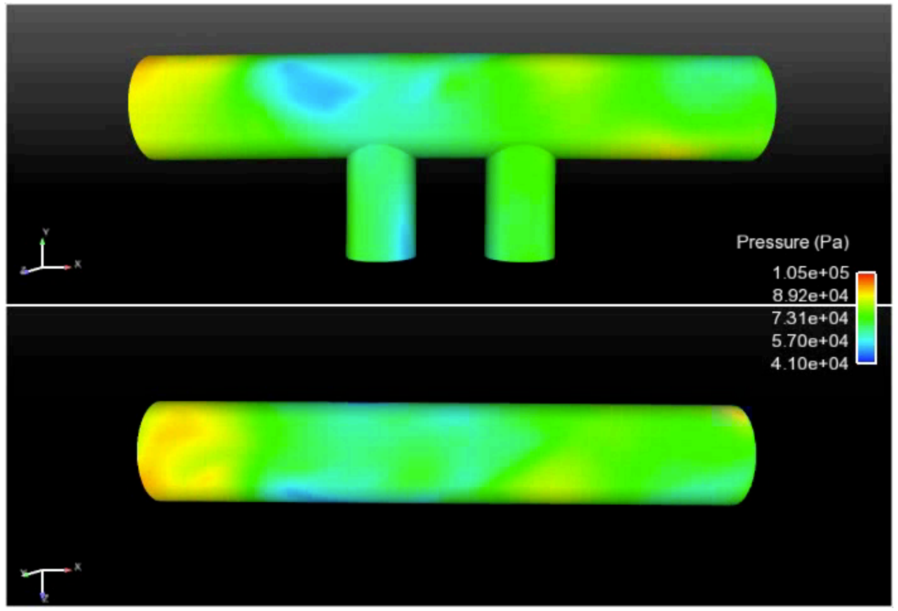
- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011*; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

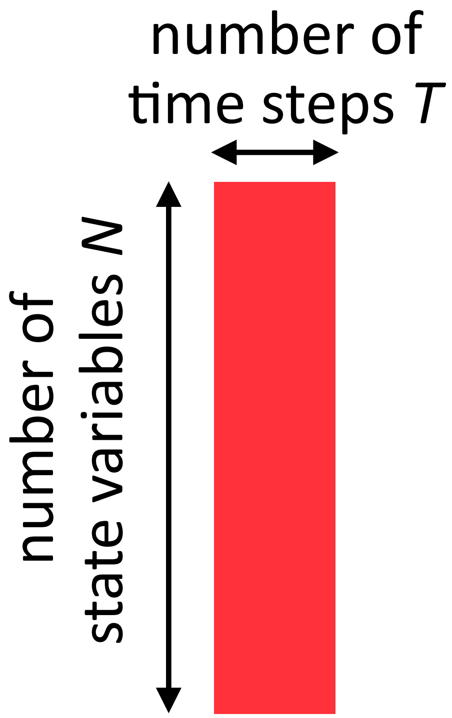
* #2 most-cited paper, Int J Numer Meth Eng, 2011

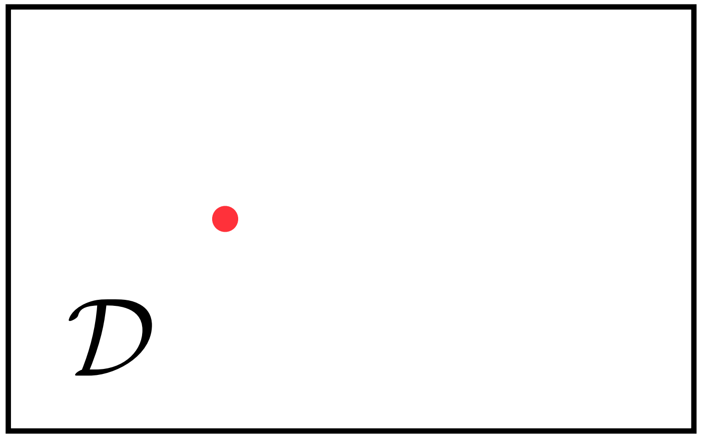
Training simulations: state tensor

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

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



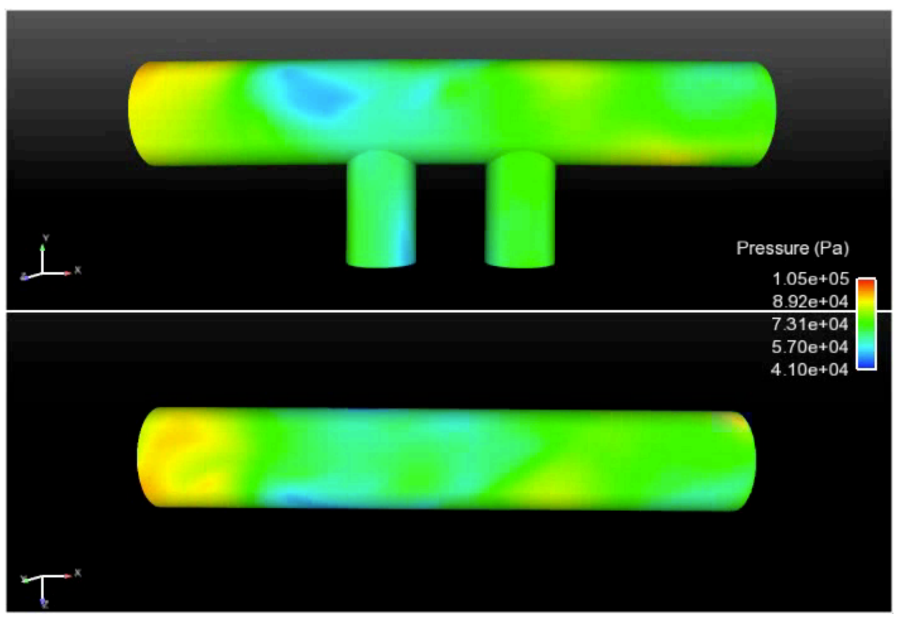


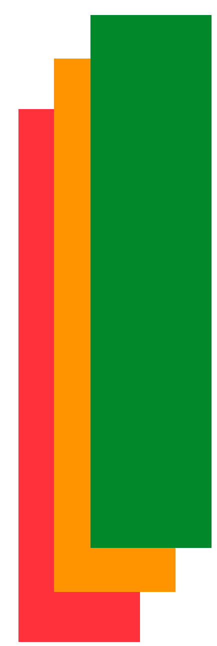


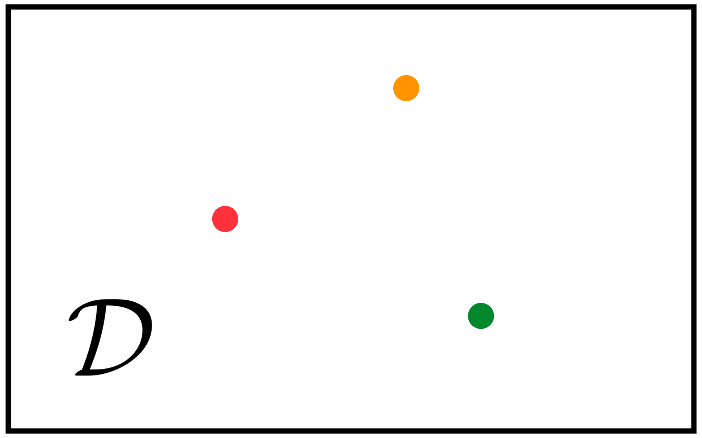
Training simulations: state tensor

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

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



$\mathcal{X}^{ijk} =$ 

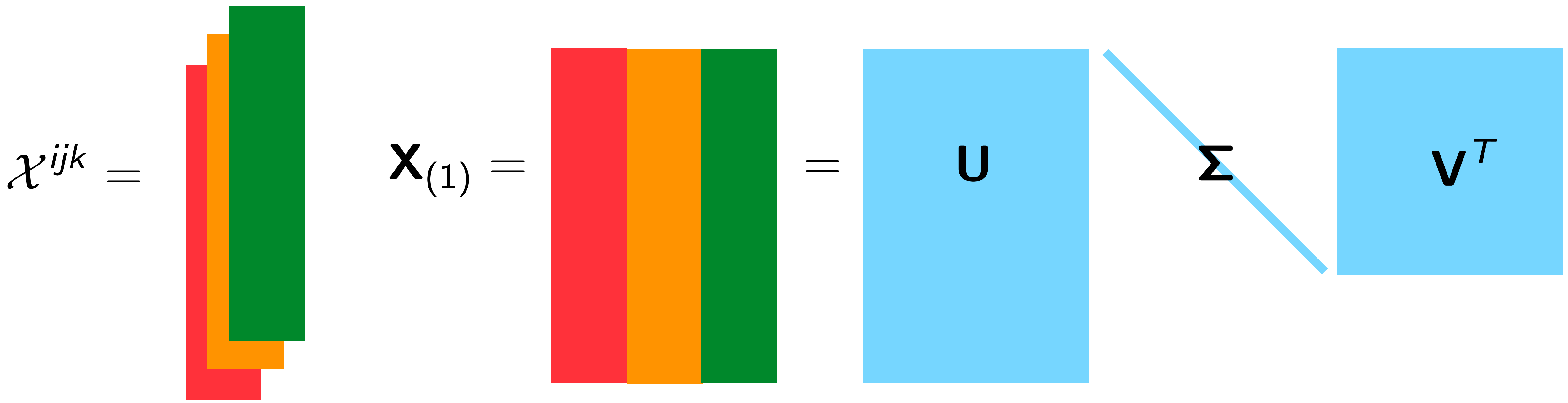


Tensor decomposition

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

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 values of mode-1 unfolding

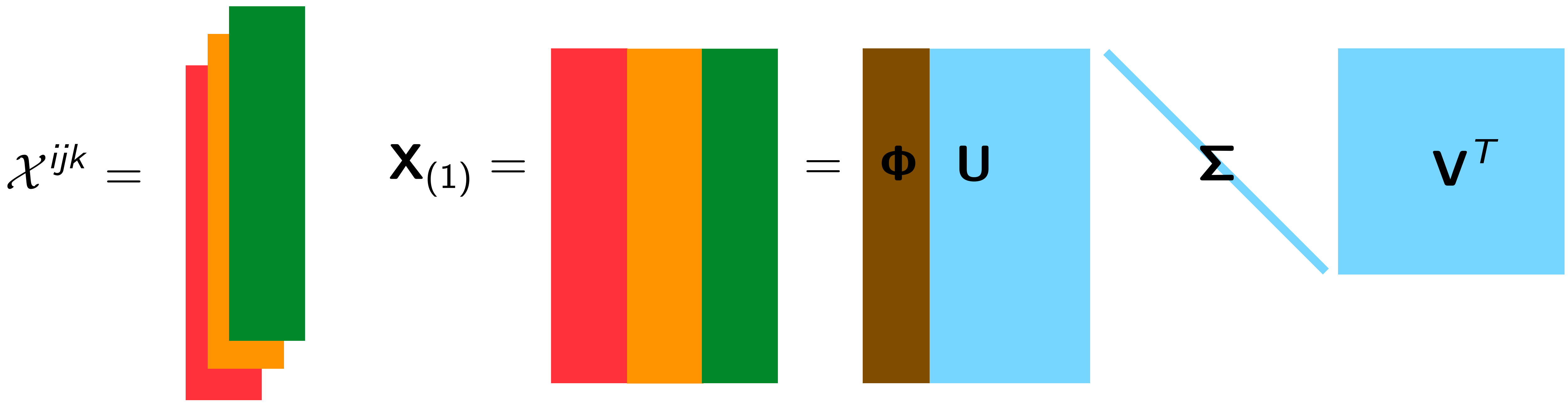


Tensor decomposition

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

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 values of mode-1 unfolding

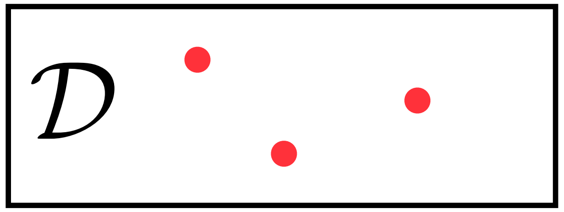


$\boldsymbol{\Phi}$ *columns are principal components of the spatial simulation data*

How to integrate these data with the computational model?

Previous state of the art: POD–Galerkin

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



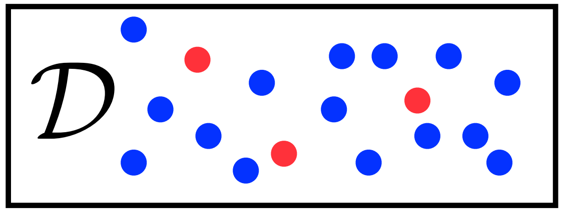
1. *Training:* Solve ODE for $\mu \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

1. Reduce the number of unknowns
2. Reduce the number of equations

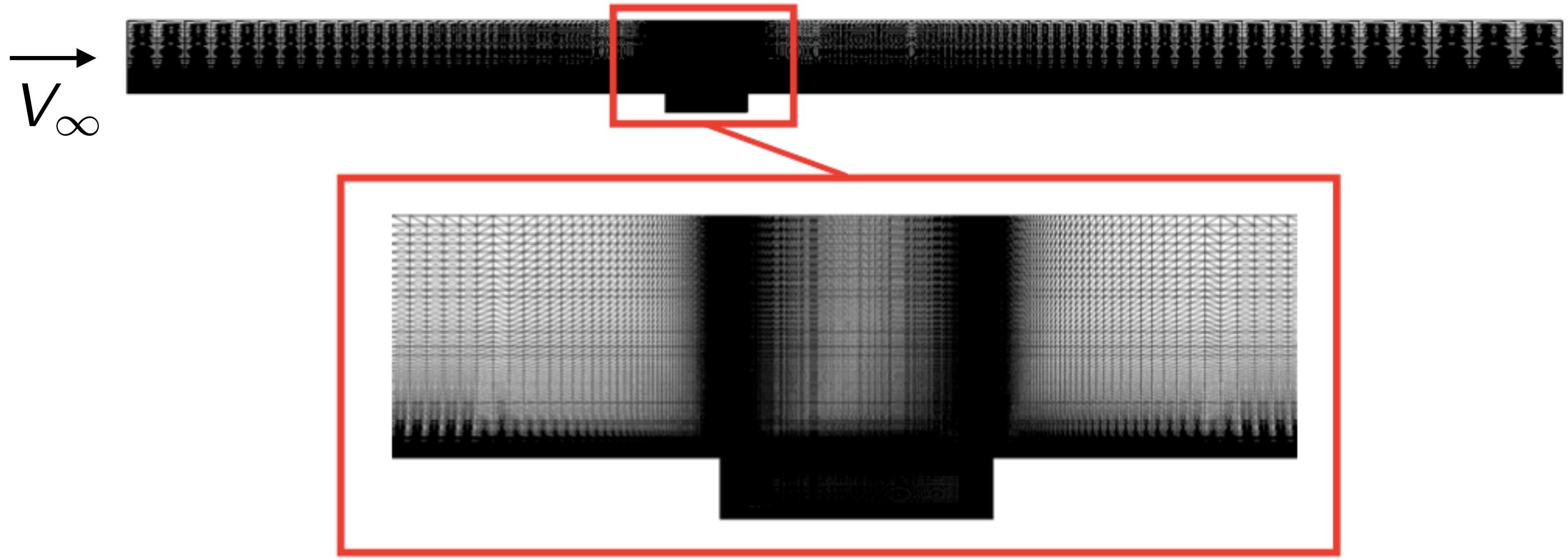
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\,\hat{\mathbf{x}}(t)$$

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

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



B61 captive carry



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

Spatial discretization

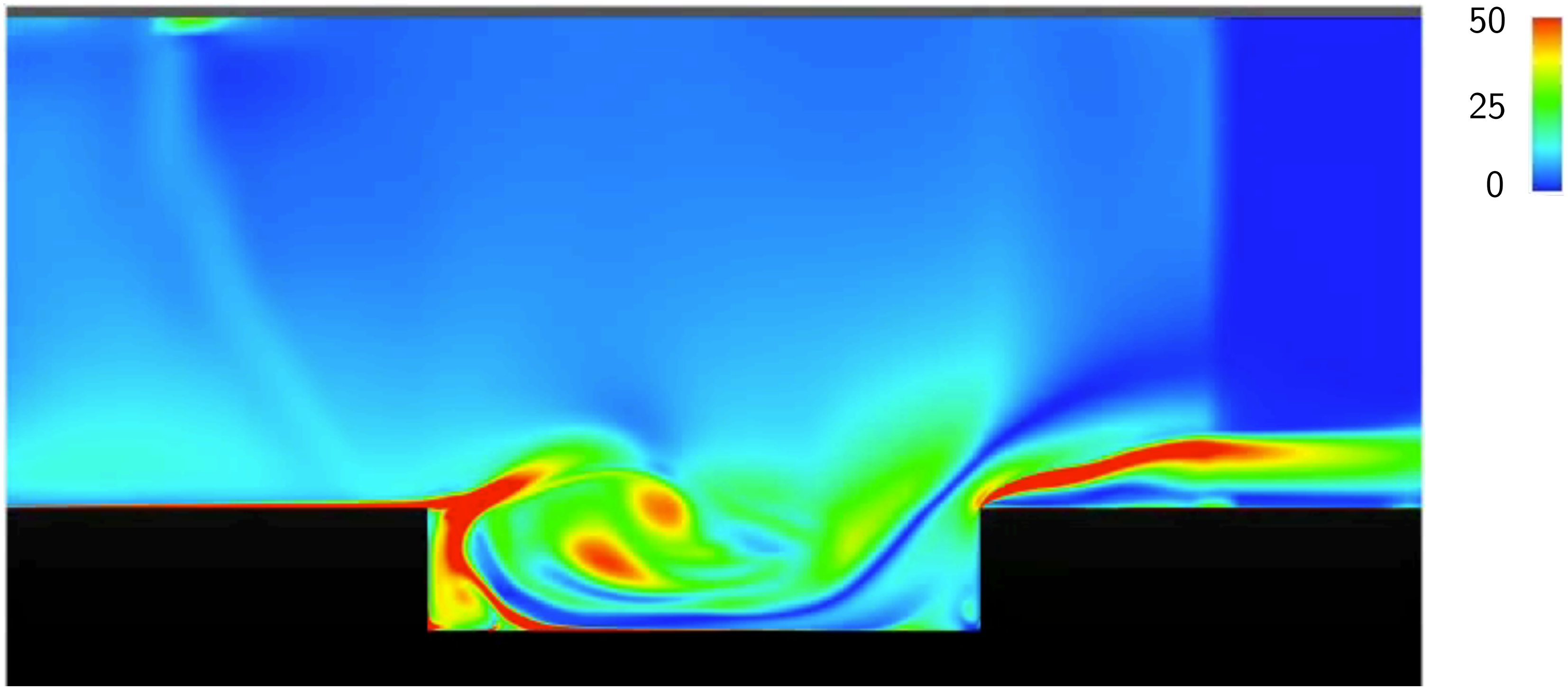
- 2nd-order finite volume
- DES turbulence model
- 1.2×10^6 degrees of freedom

Temporal discretization

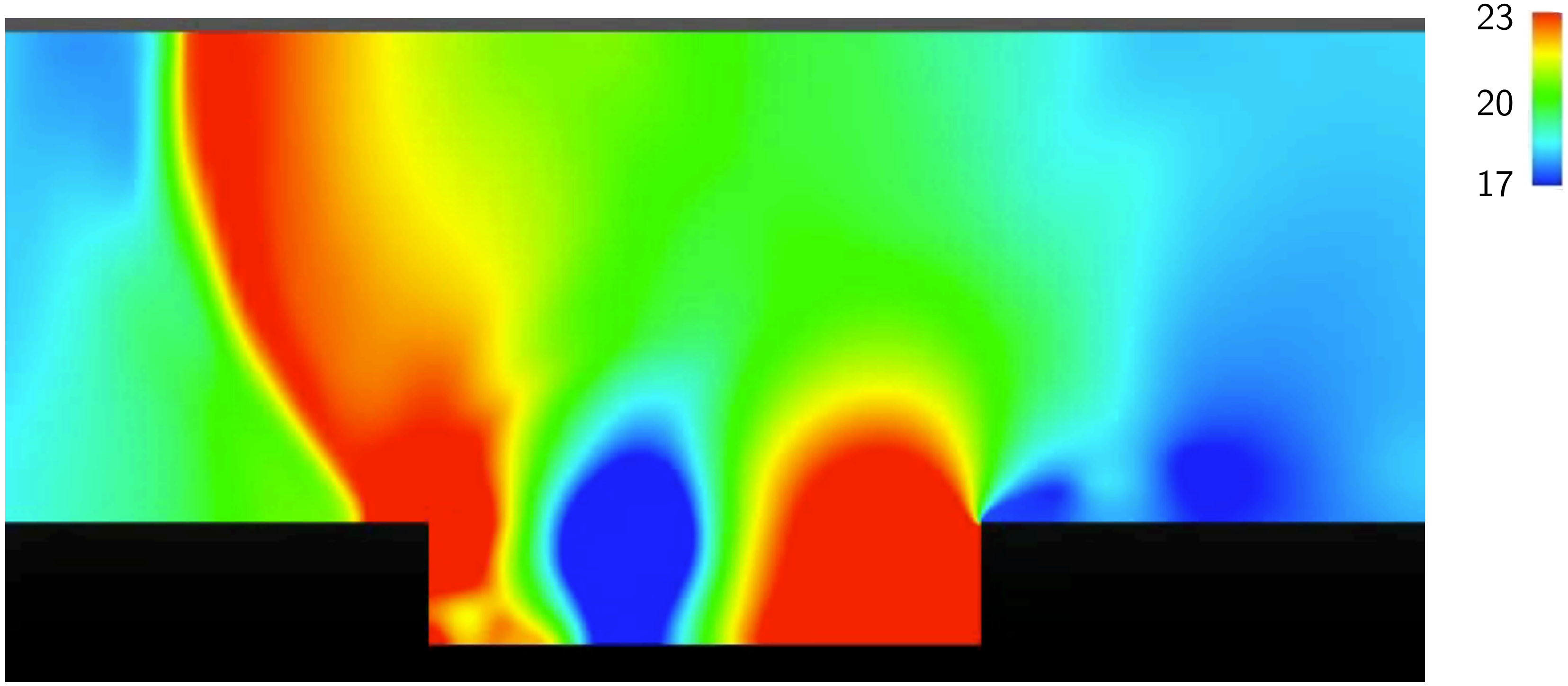
- 2nd-order BDF
- Verified time step $\Delta t = 1.5 \times 10^{-3}$
- 8.3×10^3 time instances

High-fidelity model solution

vorticity field

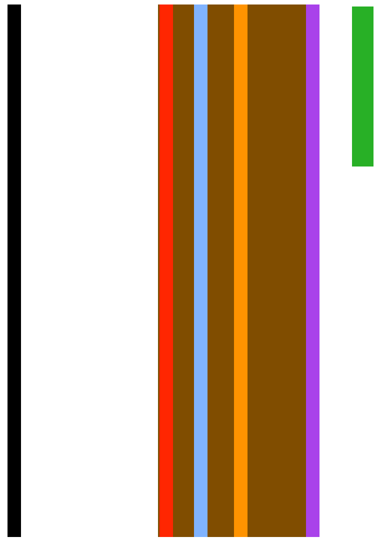


pressure field

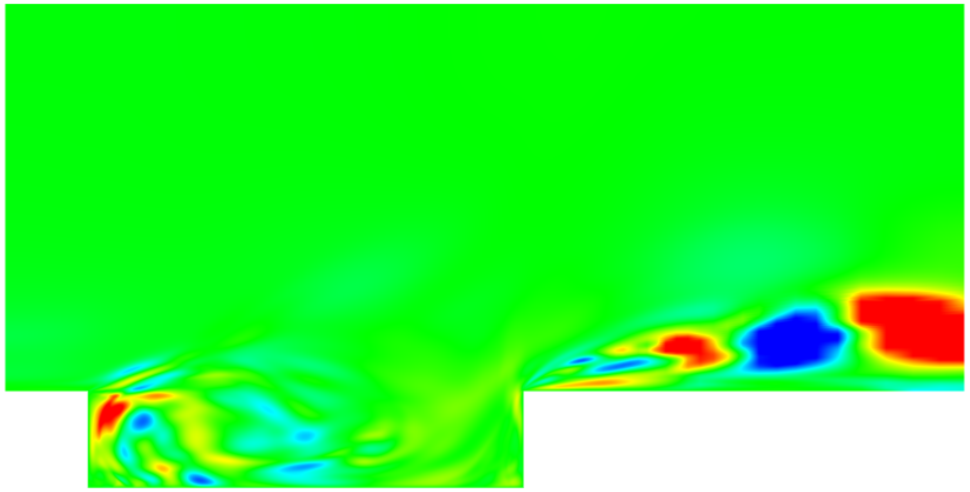


Principal components

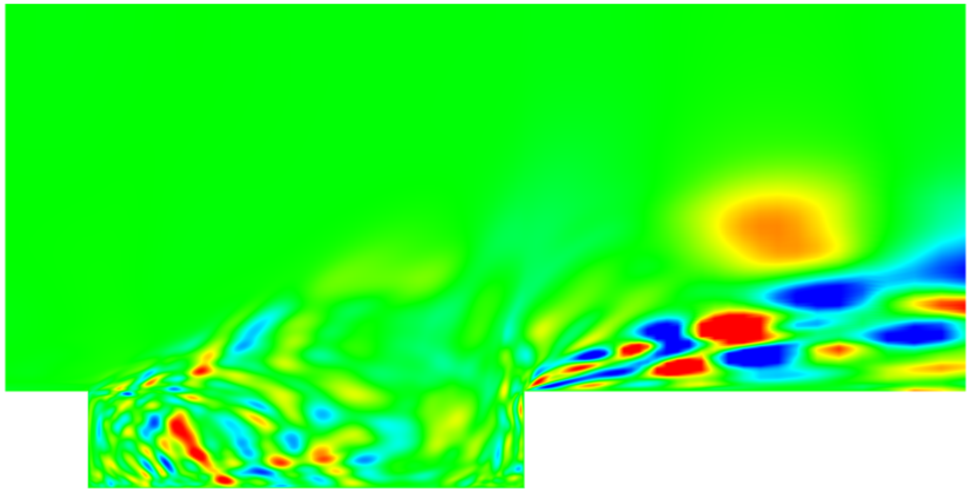
$\mathbf{x}(t) \approx \boldsymbol{\Phi}\ \hat{\mathbf{x}}(t)$



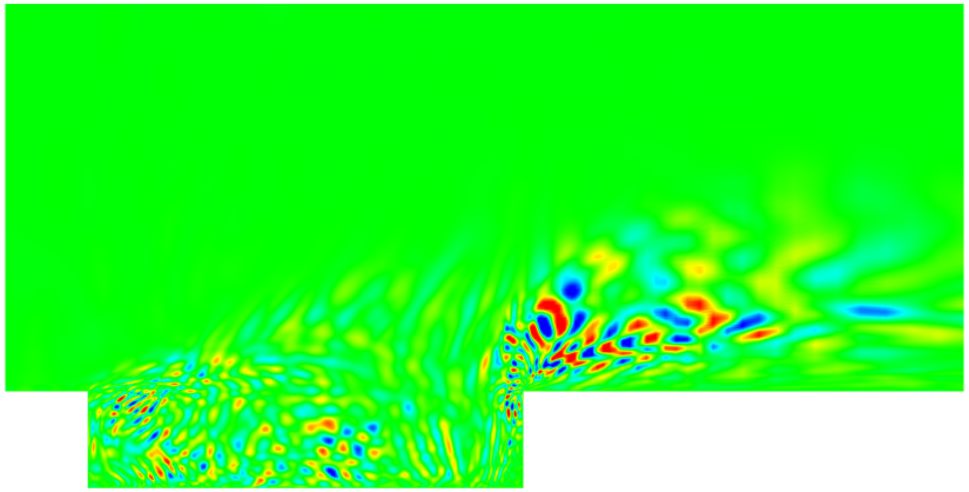
ϕ_1



ϕ_{21}

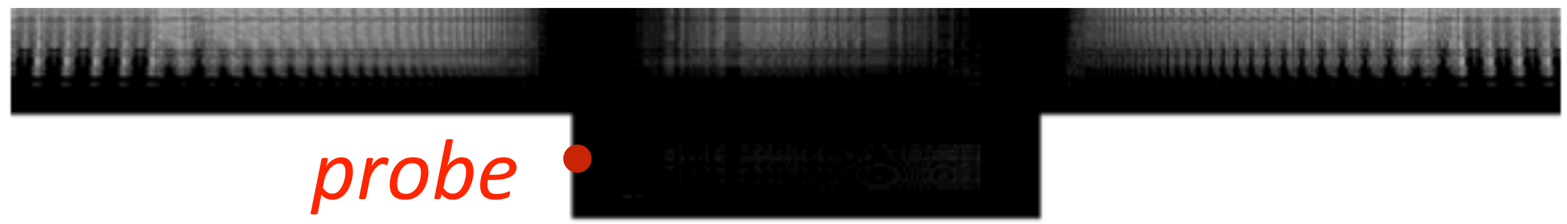


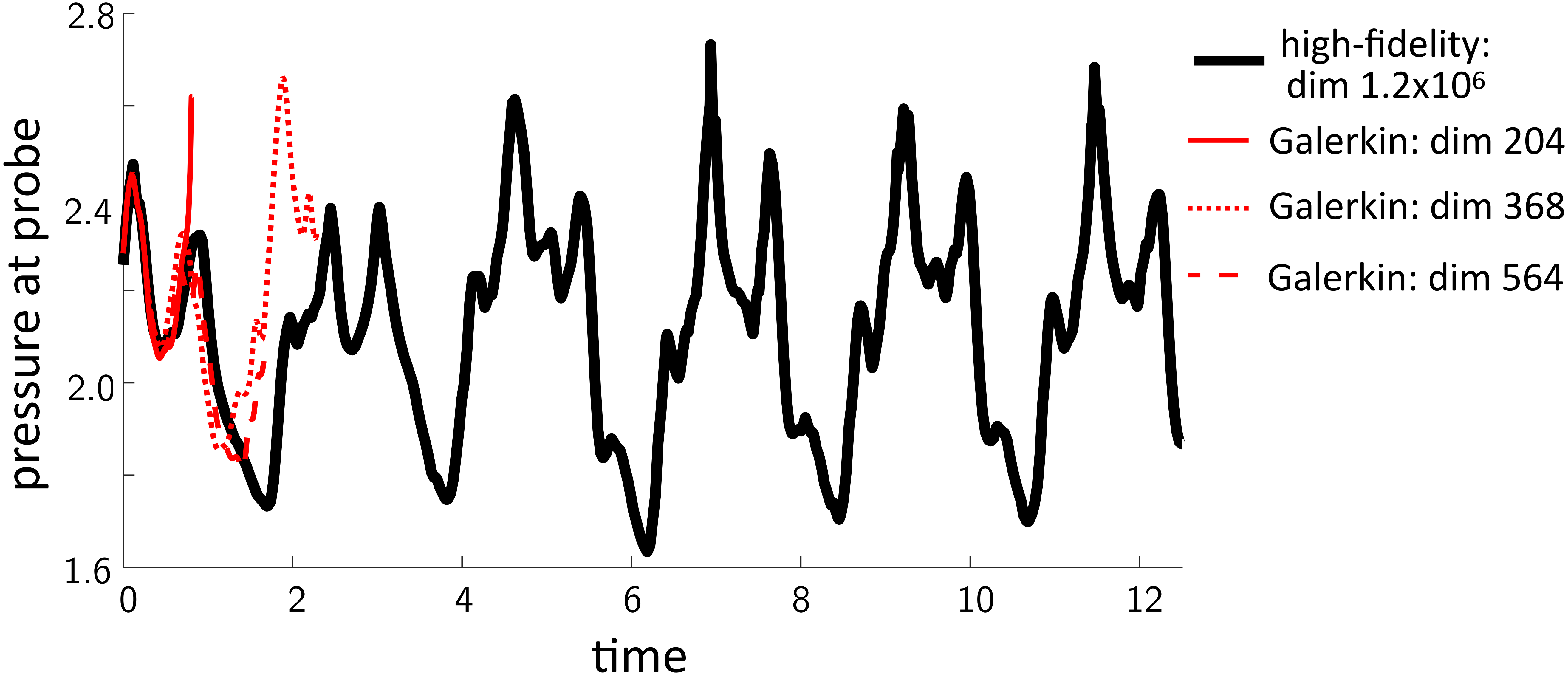
ϕ_{101}



ϕ_{401}

Galerkin performance





- Galerkin projection fails regardless of basis dimension

Can we construct a better projection?

Galerkin: time-continuous optimality

ODE

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

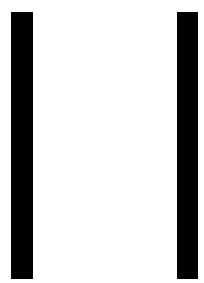
Galerkin ODE

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

Galerkin: time-continuous optimality

ODE

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



Galerkin ODE

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

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

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

OΔE

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

Galerkin OΔE

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

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

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

Residual minimization and time discretization

ODE

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

residual minimization →

Galerkin ODE

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

time discretization ↓

time discretization ↓

LSPG OΔE

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\mathrm{range}(\boldsymbol{\Phi})}{\mathrm{argmin}} \|\mathbf{A}\mathbf{r}^n(\mathbf{v})\|_2$$

$$n = 1, \ldots, T$$

← *residual minimization*

OΔE

$$\mathbf{r}^n(\mathbf{x}^n) = 0$$

$$n = 1, \ldots, T$$

Galerkin OΔE

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

$$n = 1, \ldots, T$$

[Carlberg, Bou-Mosleh, Farhat, 2011]

$$\boldsymbol{\Phi}\hat{\mathbf{x}}^n = \underset{\mathbf{v}\in\mathrm{range}(\boldsymbol{\Phi})}{\mathrm{argmin}} \|\mathbf{A}\mathbf{r}^n(\mathbf{v})\|_2 \quad \Leftrightarrow \quad \boldsymbol{\Psi}^n(\hat{\mathbf{x}}^n)^T\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{x}}^n) = 0$$

$$\boldsymbol{\Psi}^n(\hat{\mathbf{x}}^n) := \mathbf{A}^T\mathbf{A}\left(\alpha_0\mathbf{I} - \Delta t\beta_0\frac{\partial\mathbf{f}}{\partial\mathbf{x}}(\boldsymbol{\Phi}\hat{\mathbf{x}}^n; t)\right)\boldsymbol{\Phi}$$

*Least-squares Petrov–Galerkin (**LSPG**) projection*

Discrete-time error bound

Theorem [Carlberg, Antil, Barone, 2017]

If the following conditions hold:

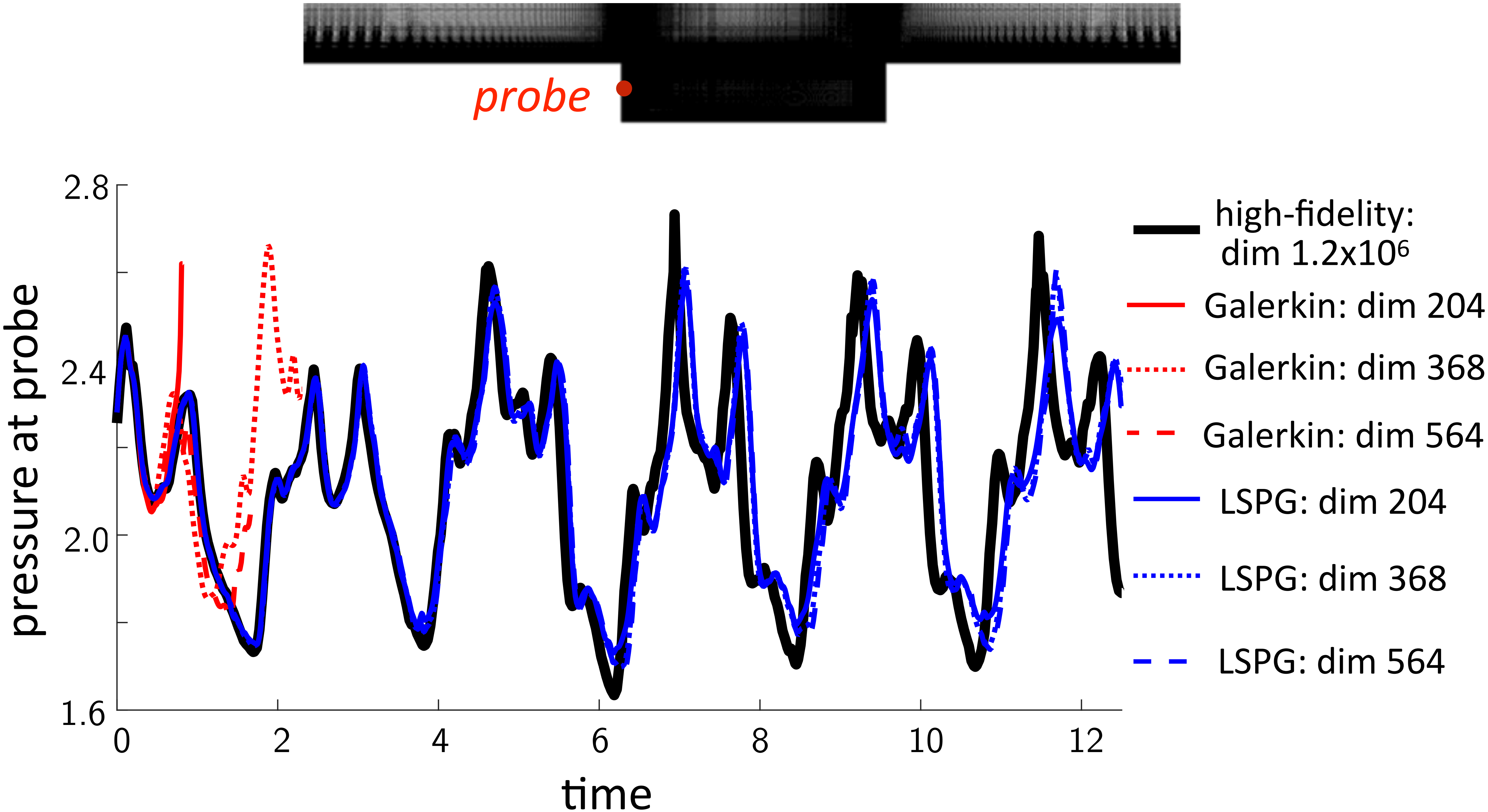
1. $\mathbf{f}(\cdot\,;t)$ is Lipschitz continuous with Lipschitz constant κ
2. The time step Δt is small enough such that $0 < h := |\alpha_0| - |\beta_0|\kappa\Delta t$,
3. A backward differentiation formula (BDF) time integrator is used,
4. LSPG employs $\mathbf{A} = \mathbf{I}$, then

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

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

+ LSPG sequentially minimizes the error bound

LSPG performance



+ LSPG is far more accurate than Galerkin

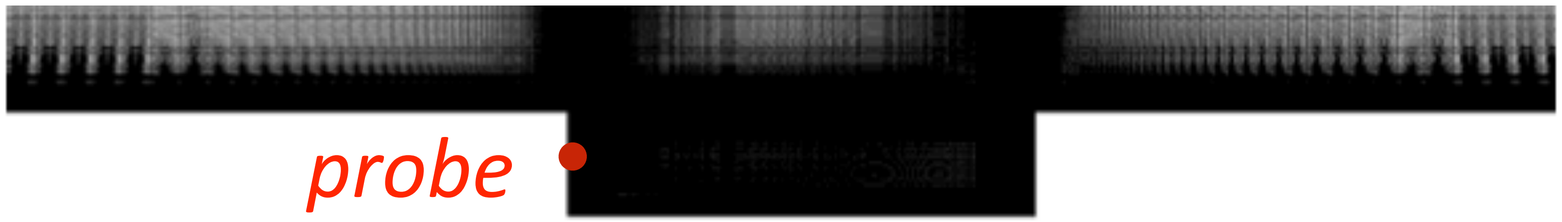
Our research

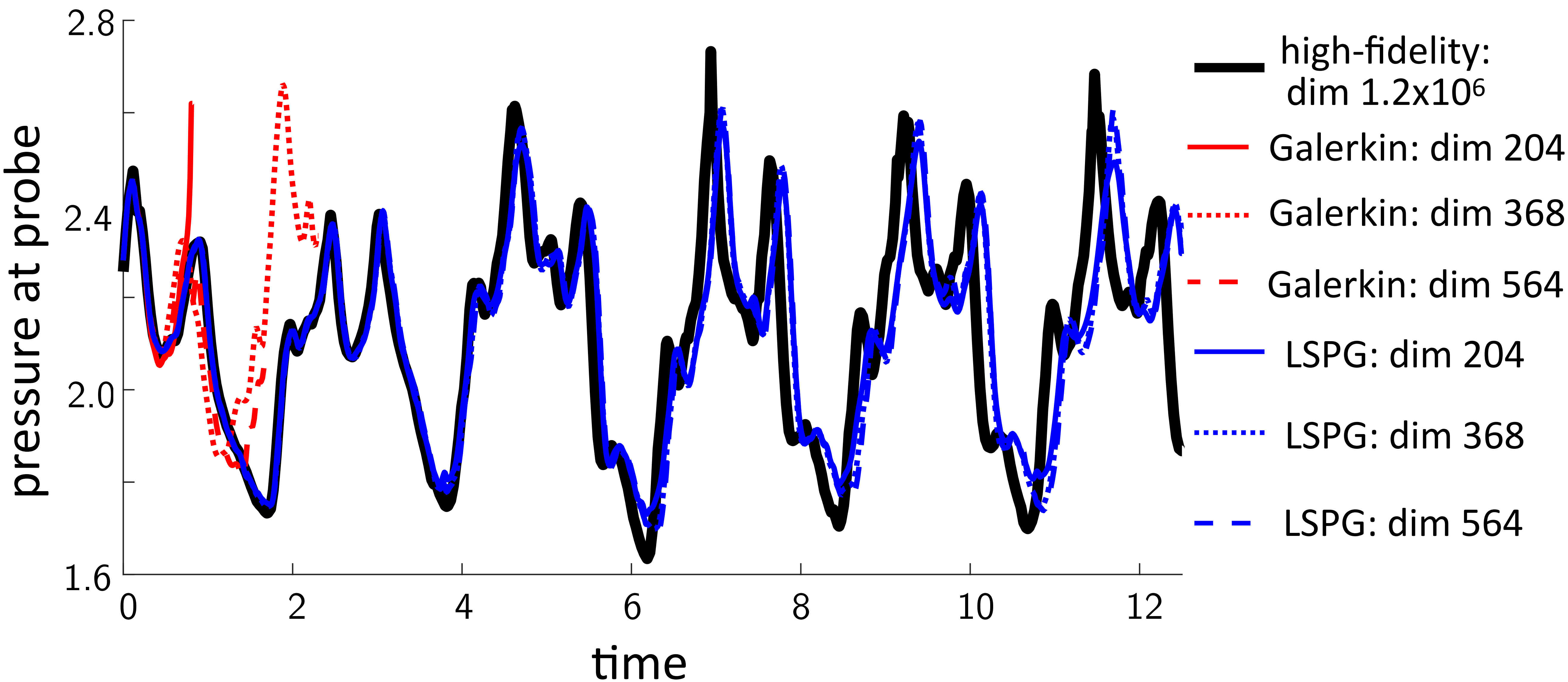
Accurate, low-cost, structure-preserving, reliable, certified nonlinear model reduction

- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013*]
- *low cost*: reduce temporal complexity [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

* #1 most-cited paper, J Comp Phys, 2013

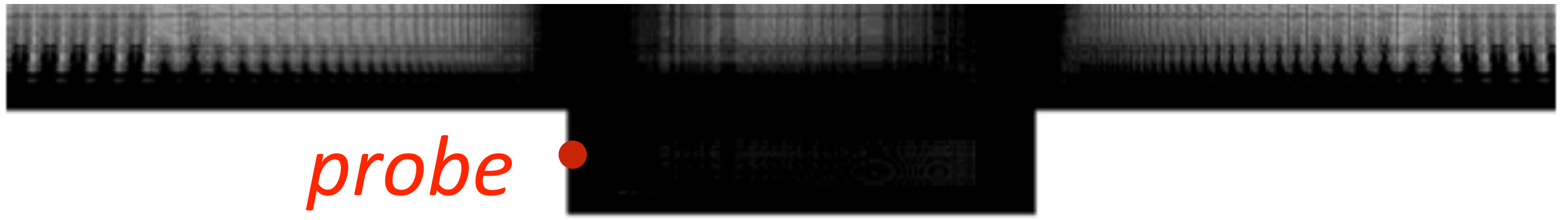
Wall-time problem

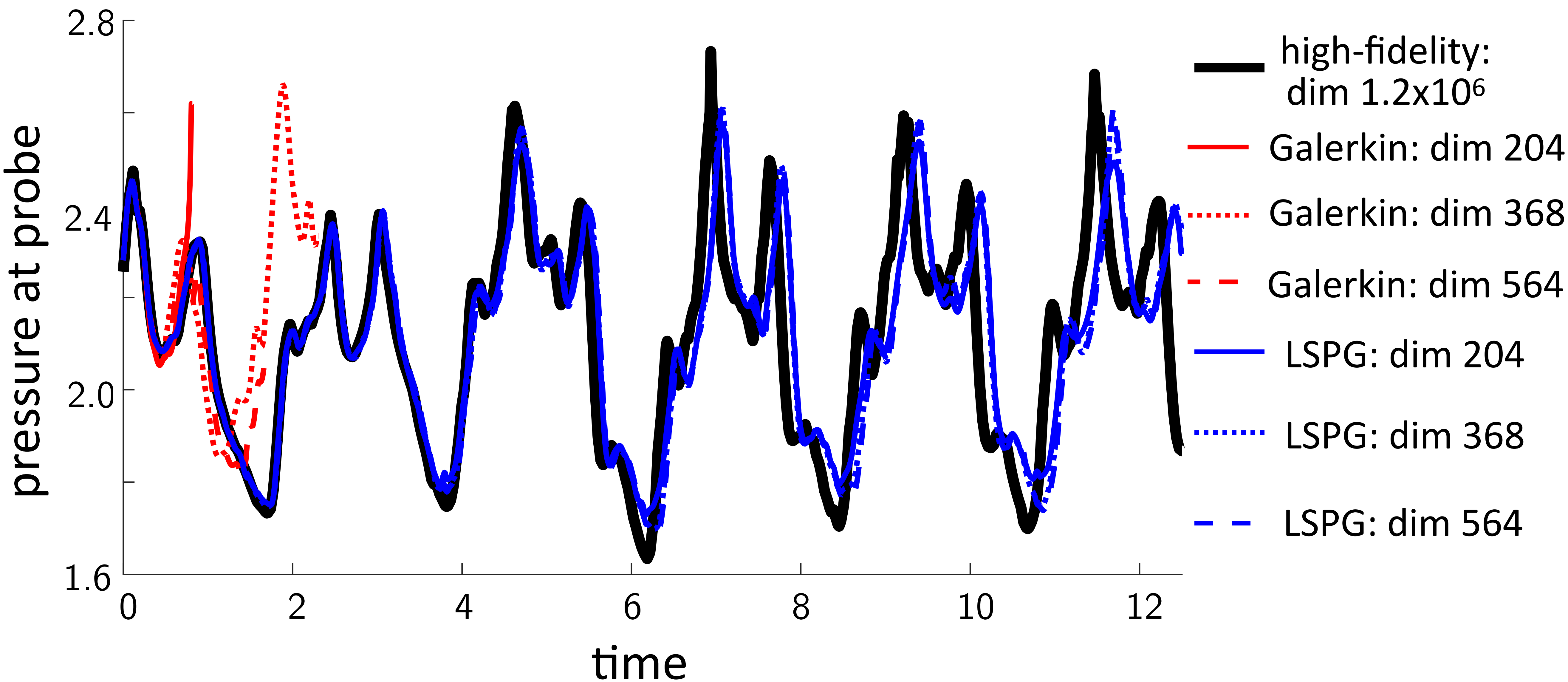




- *High-fidelity simulation*: 1 hour, 48 cores
- *Fastest LSPG simulation*: 1.3 hours, 48 cores

Wall-time problem





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

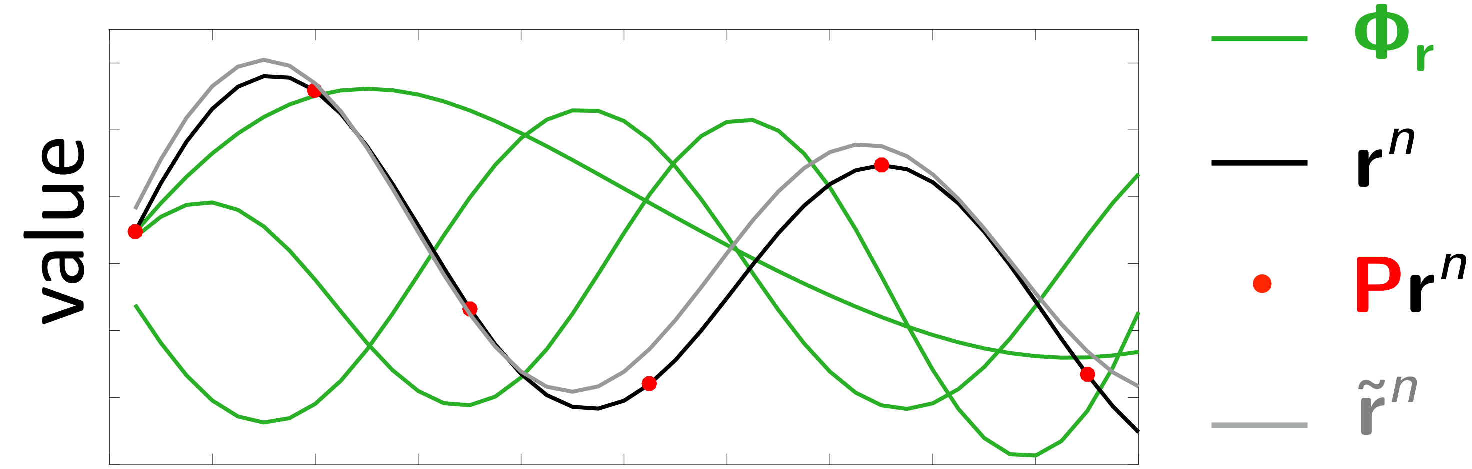
Why does this occur? Can we fix it?

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

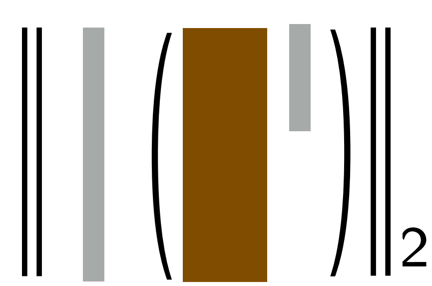
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\, \mathbf{r}^n(\boldsymbol{\Phi}\,\hat{\mathbf{v}}) \right\|_2$$

Can we select $\mathbf{A}$ *to make this less expensive?*

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



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

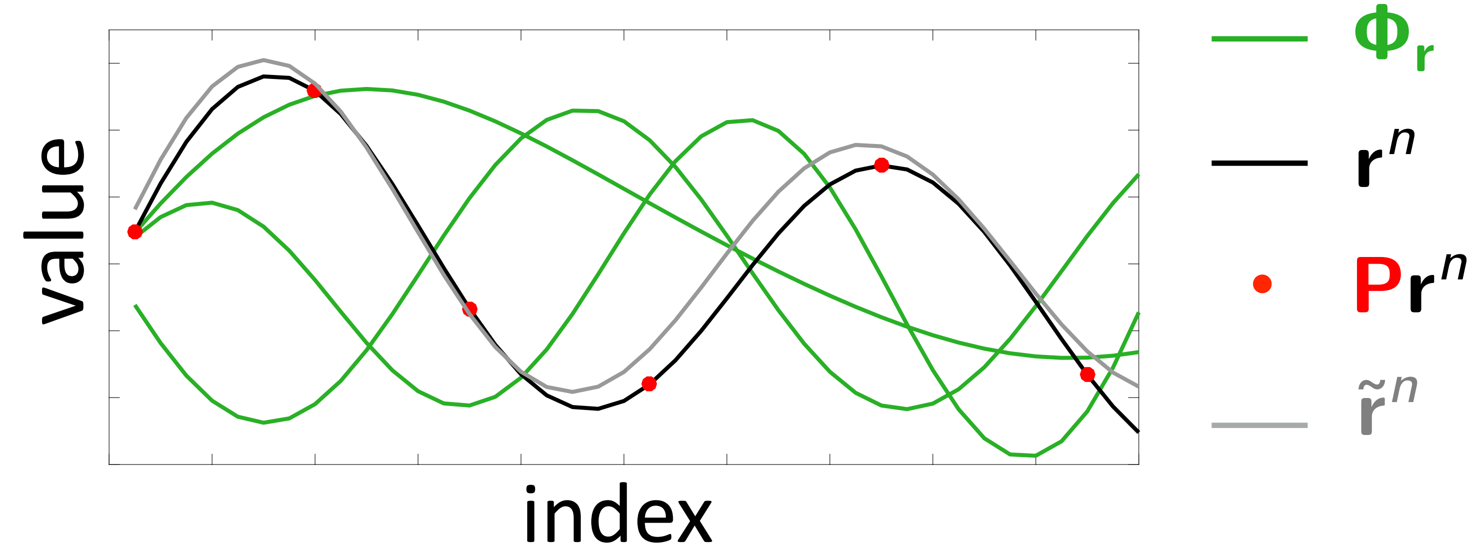


Cost reduction by gappy PCA [Everson and Sirovich, 1995]

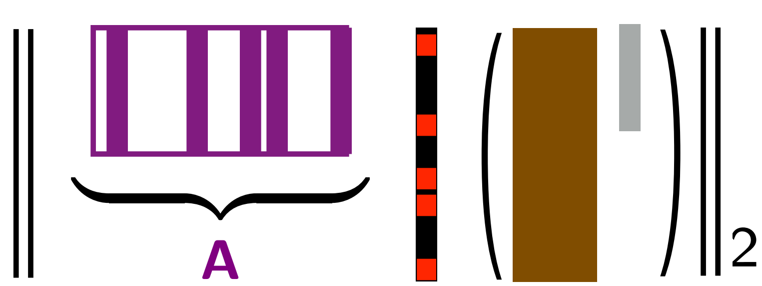
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \| \mathbf{A} \, \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) \|_2$$

Can we select **A** *to make this less expensive?*

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



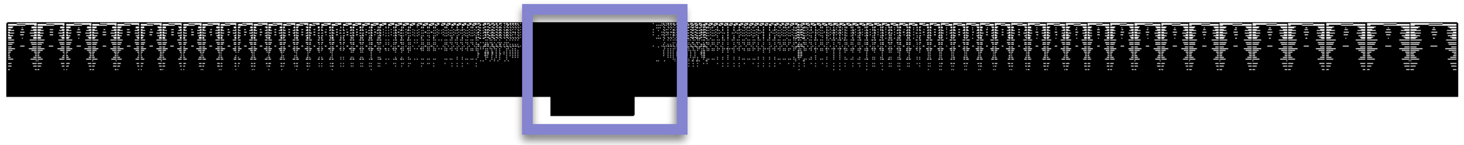
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \| \underbrace{(\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^+\mathbf{P}}_{\mathbf{A}} \, \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) \|_2$$

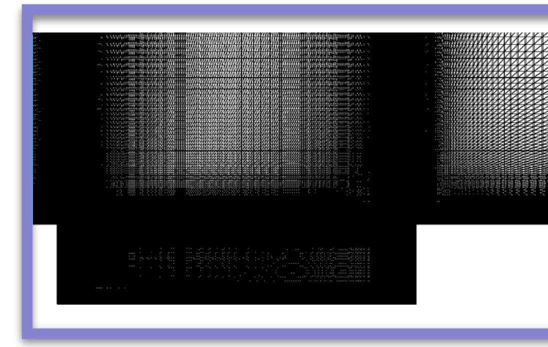


\+ *Only a few elements of* $\mathbf{r}^n$ *must be computed*

Sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]

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

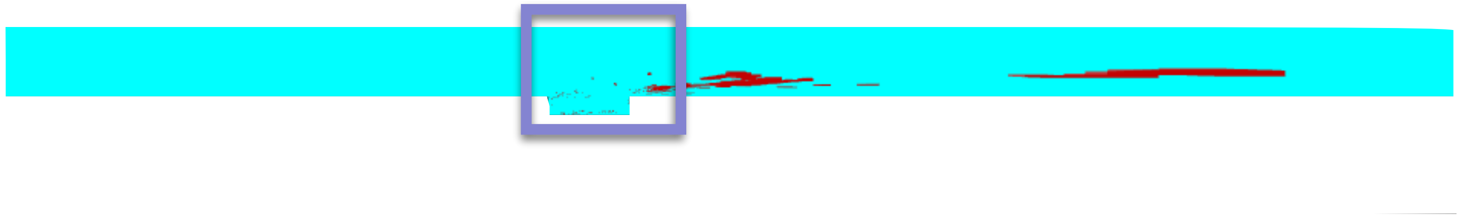




Sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]

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

sample mesh

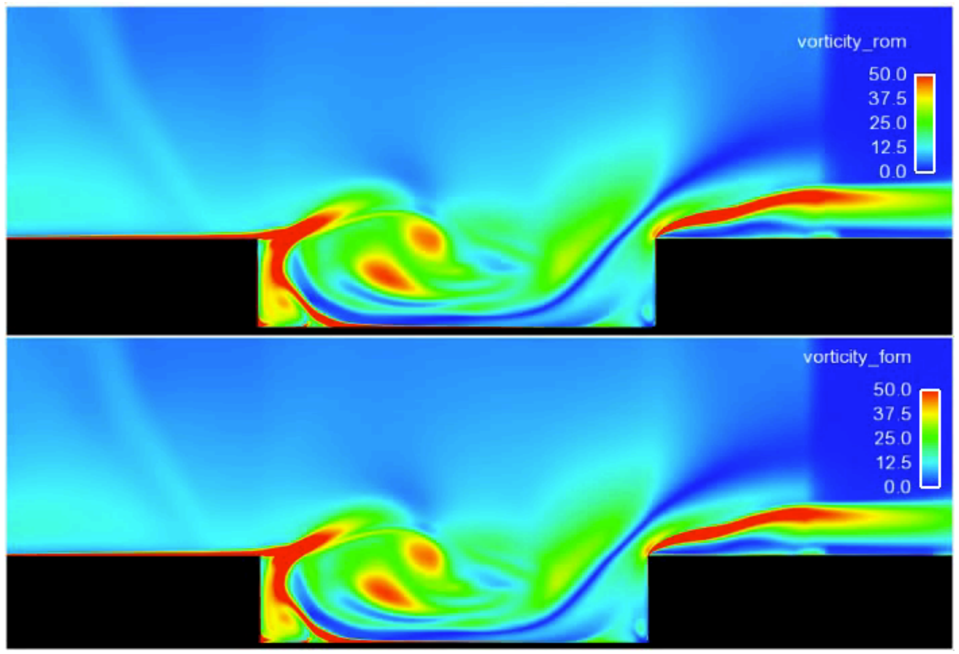


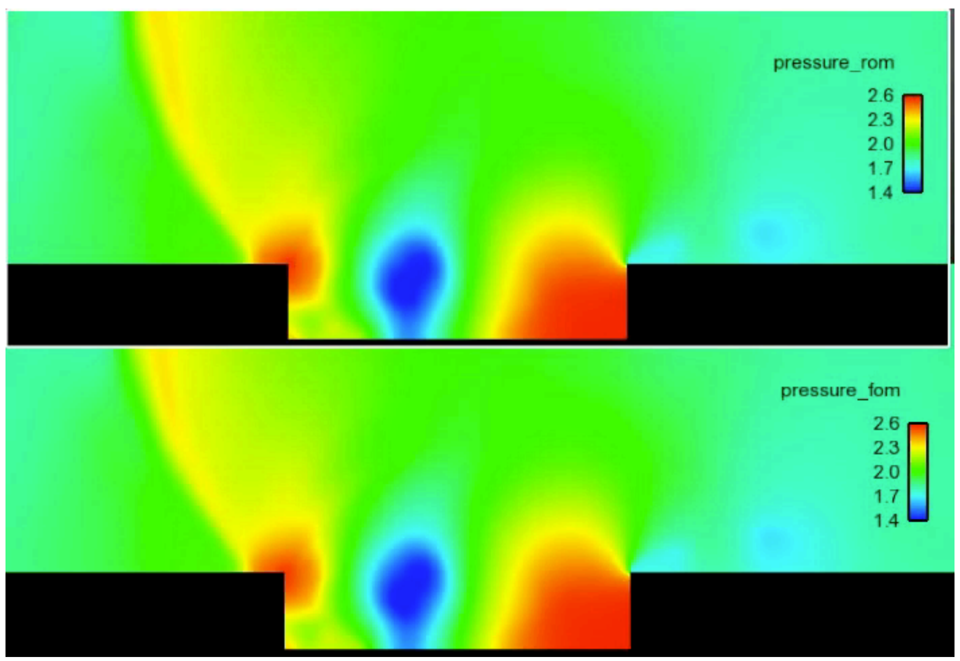
+ *HPC on a laptop*

vorticity field | *pressure field*

LSPG ROM with
$\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^{+}\mathbf{P}$
32 min, 2 cores

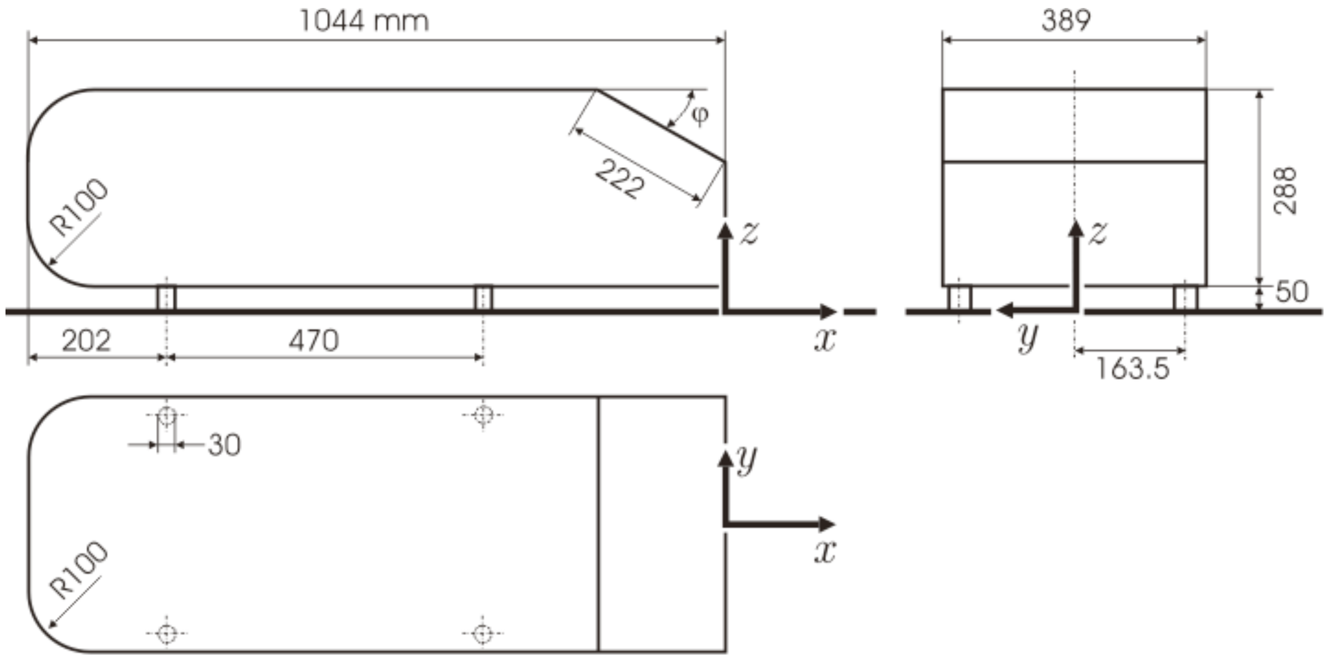
high-fidelity
5 hours, 48 cores

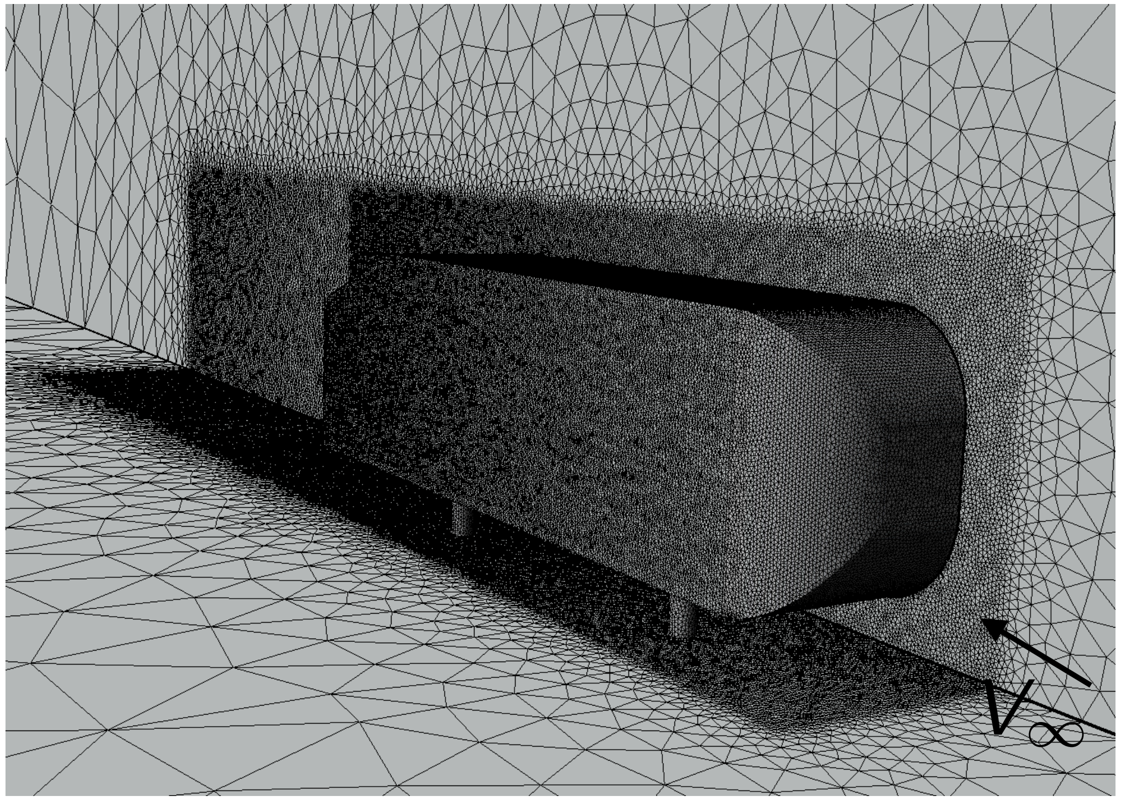




+ *229x savings in core–hours*
+ *< 1% error in time-averaged drag*
- implemented in three computational-mechanics codes

Ahmed body [Ahmed, Ramm, Faitin, 1984]





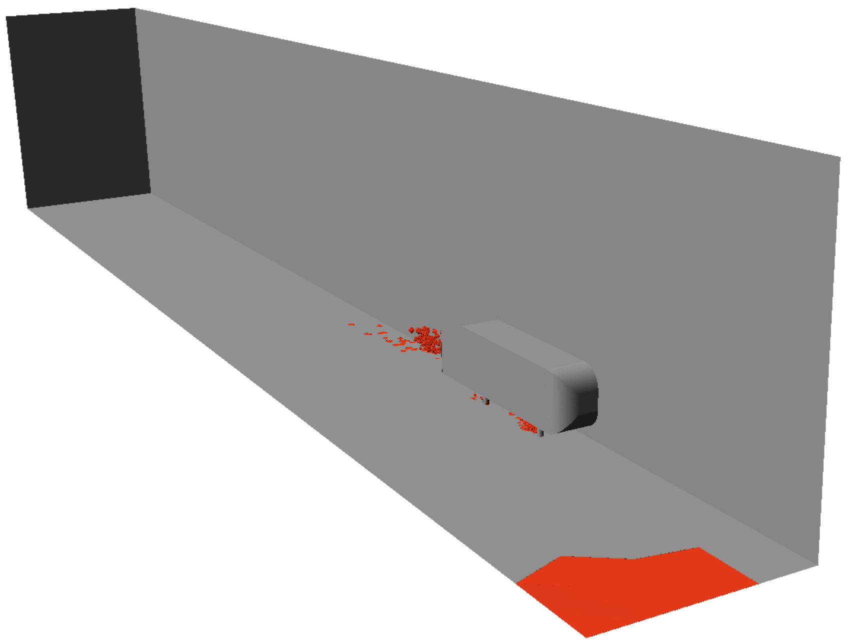
- Unsteady Navier–Stokes
- Re = 4.3 x 10^6
- $M_\infty = 0.175$

Spatial discretization
- 2nd-order finite volume
- DES turbulence model
- 1.7×10^7 degrees of freedom

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

Ahmed body results [Carlberg, Farhat, Cortial, Amsallem, 2013]

sample mesh



+ *HPC on a laptop*

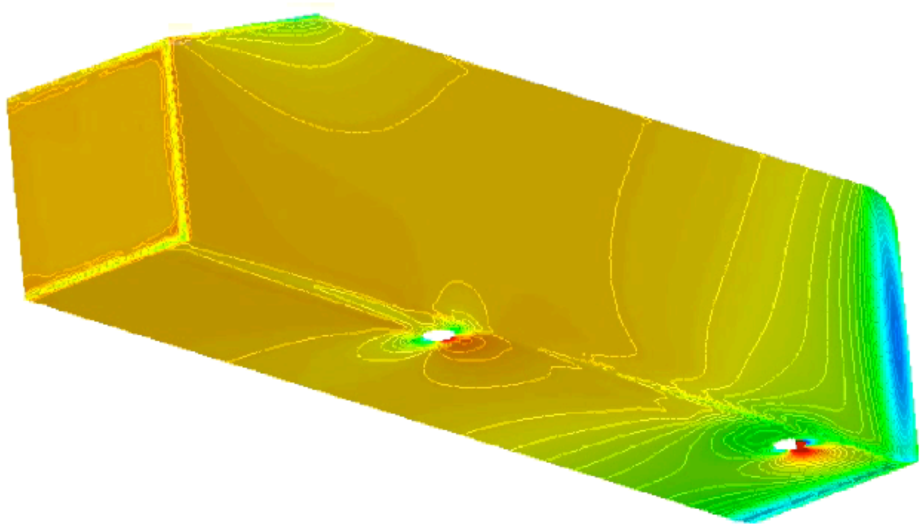
LSPG ROM with $\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^{+}\mathbf{P}$

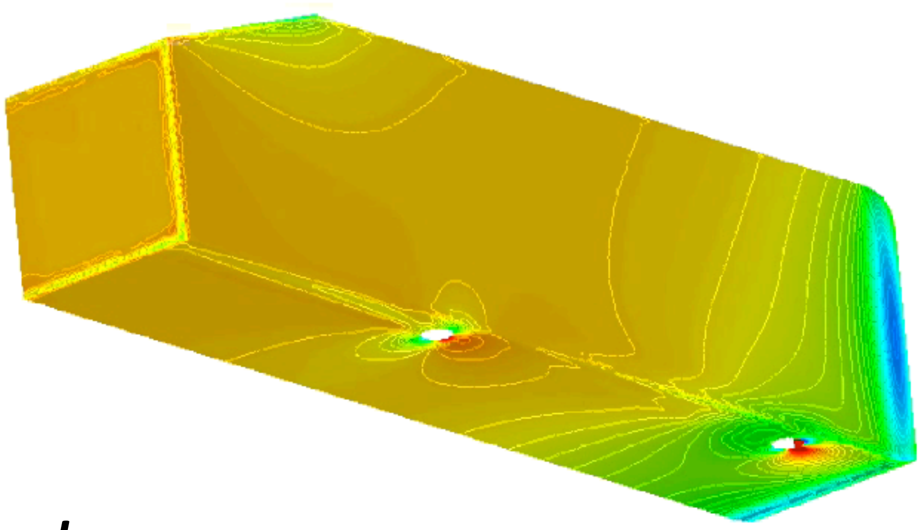
4 hours, 4 cores

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*

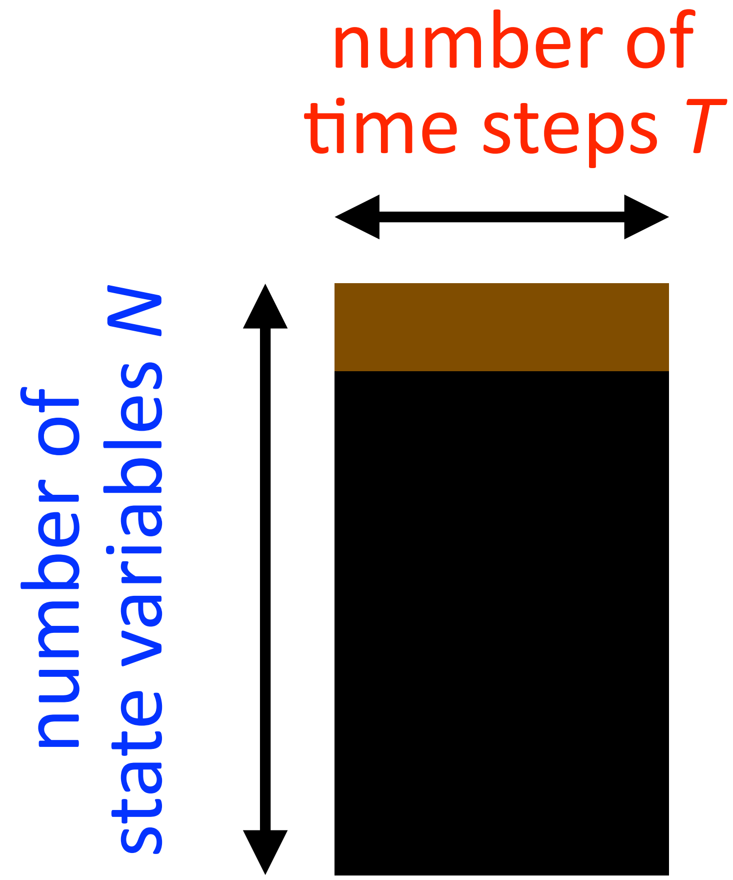
Our research

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

- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013*]
- *low cost*: reduce temporal complexity
  [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models
  [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

Temporal complexity

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



So far, we have focused on reducing the spatial complexity

What about the temporal complexity?

Tensor decomposition

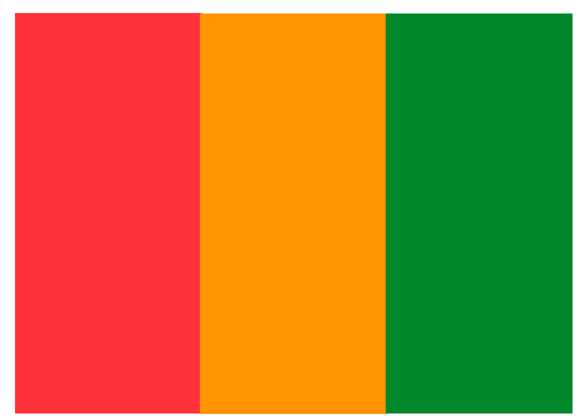
$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

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

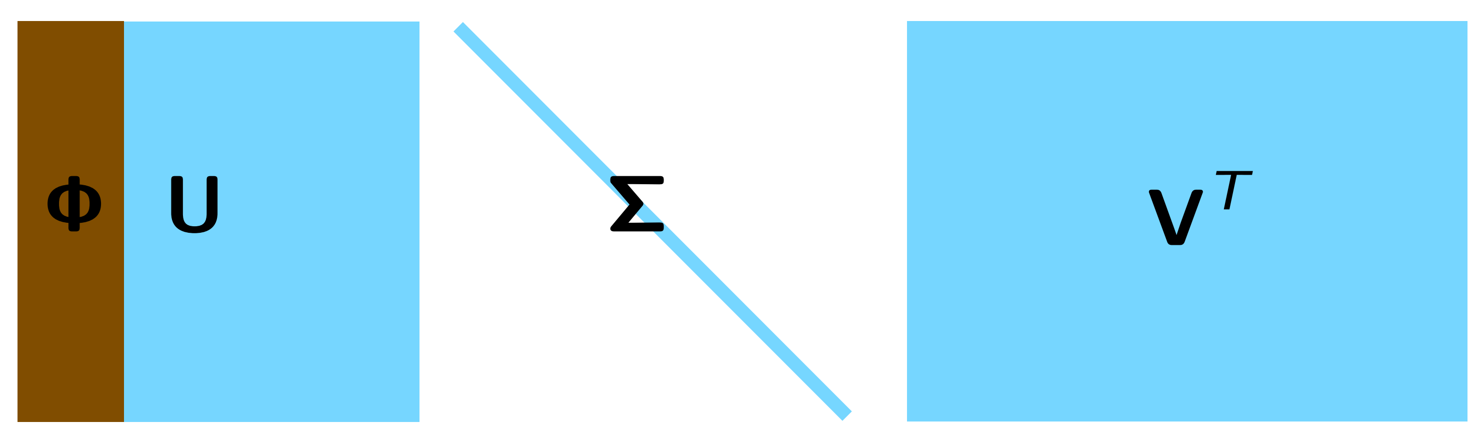
*Compute dominant left singular values of **mode-1** unfolding*

$\mathcal{X}^{ijk} =$

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



$=$



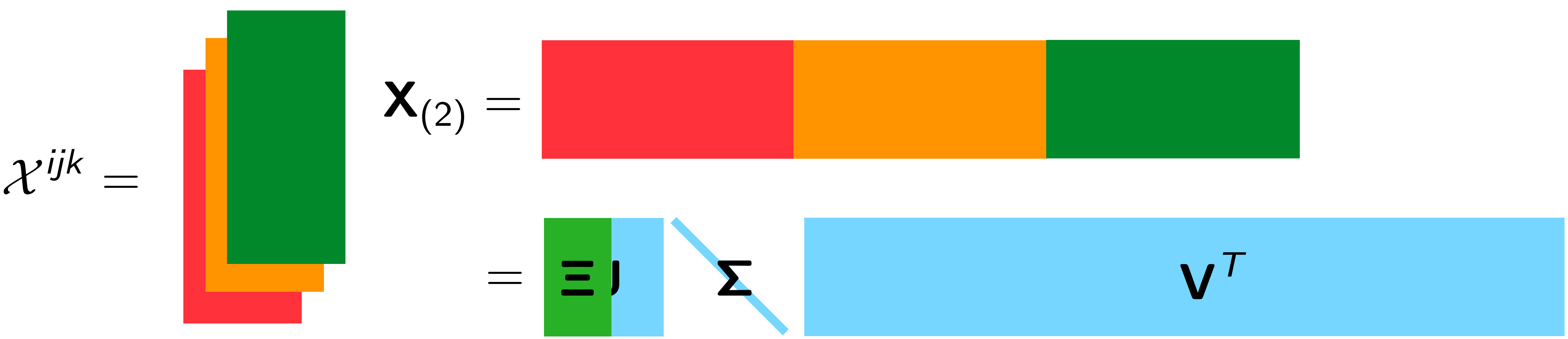
$\boldsymbol{\Phi}$ *columns are principal components of the **spatial** simulation data*

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 values of ***mode-2*** *unfolding*



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

How to integrate these data with the computational model?

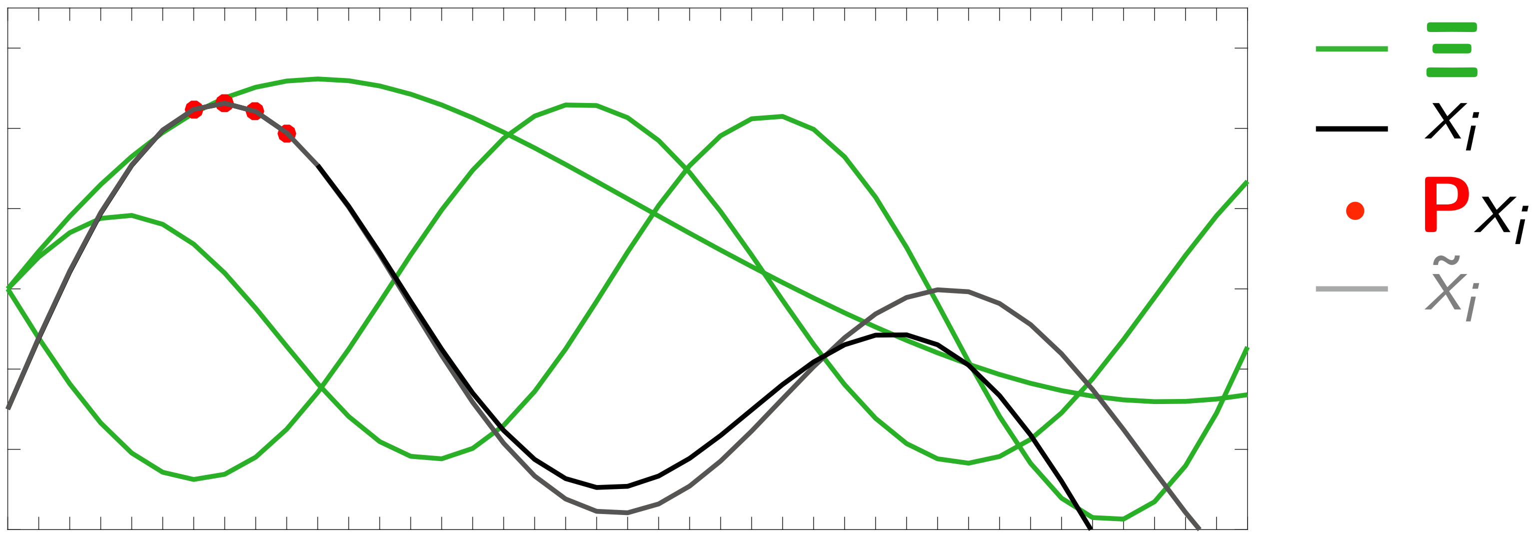
1. Space–time LSPG projection [Choi and Carlberg, 2017]
2. Data-driven time integration [Carlberg, Ray, v B Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017]

Idea: forecasting via Gappy PCA in time

[Carlberg, Ray, van Bloemen Waanders, 2015]

$$x_i(t) \approx \tilde{x}_i(t) = \boldsymbol{\Xi}(t)(\mathbf{P}\boldsymbol{\Xi}(t))^{+}\mathbf{P}x_i(t)$$

state variable x_i



Data-driven initial guess

[Carlberg, Ray, van Bloemen Waanders, 2015]

- use forecast $\tilde{x}_i$ as accurate initial guess for the Newton solver
- \+ 50% speedup improvement observed; no accuracy loss

Data-driven time-parallel solver

[Carlberg, Brencher, Haasdonk, Barth, 2016]

- use forecast $\tilde{x}_i$ as accurate coarse propagator
- \+ provably stable; superlinear convergence; ideal speedups possible
- \+ 10x speedup improvements observed; no accuracy loss

Our research

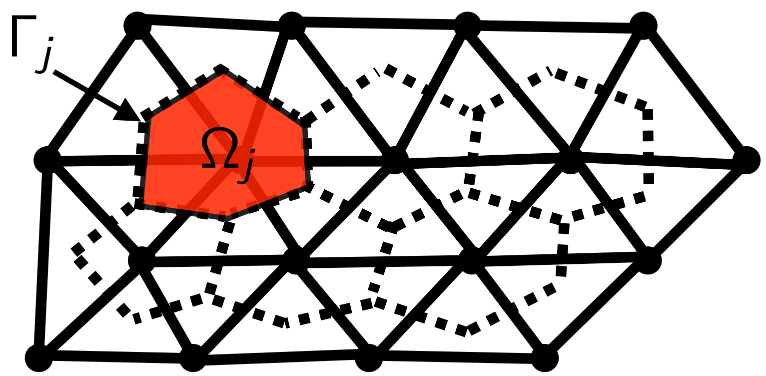
Accurate, low-cost, structure-preserving,
reliable, certified nonlinear model reduction

- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity
  [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015*; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models
  [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

* Featured Article, SIAM J Sci Comp, 2015

LSPG for finite-volume models

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



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

- average value of conserved variable i over control volume j

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

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

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

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

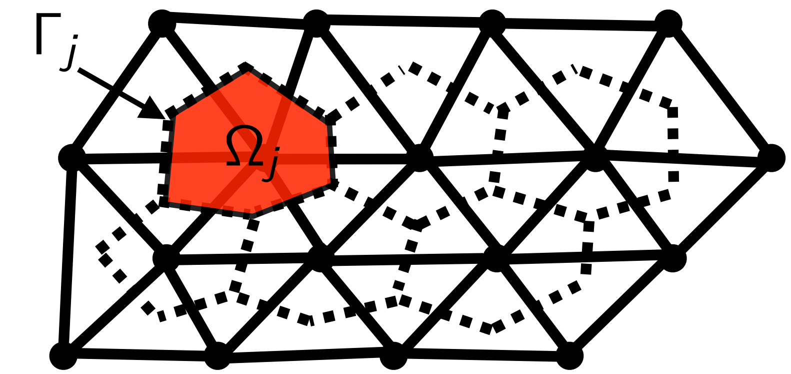
- **conservation violation** of variable i in control volume j over time step n

$$\boxed{\text{LSPG O}\Delta\text{E:} \quad \underset{\hat{\mathbf{v}}}{\text{minimize}} \; \|\mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}})\|_2}$$

- minimize weighted sum of squared conservation violations over time step n
  - Does not guarantee conservation anywhere

Enforce global conservation [Carlberg, Choi, Sargsyan, 2017]

LSPG: $\underset{\hat{\mathbf{v}}}{\text{minimize}} \; \|\mathbf{A}\mathbf{r}^n(\mathbf{\Phi}\hat{\mathbf{v}})\|_2$

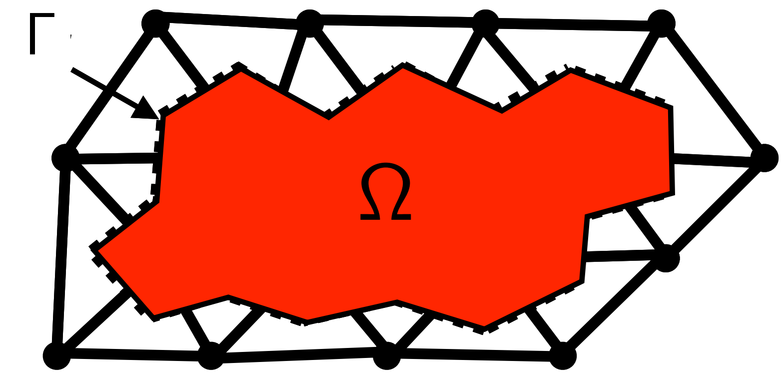


- minimize weighted sum of squared conservation-law violations over time step n

Enforce global conservation [Carlberg, Choi, Sargsyan, 2017]

$$\text{LSPG-FV:}\quad \underset{\hat{\mathbf{v}}}{\text{minimize}}\ \|\mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}})\|_2$$

$$\text{subject to } \bar{\mathbf{r}}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) = 0$$



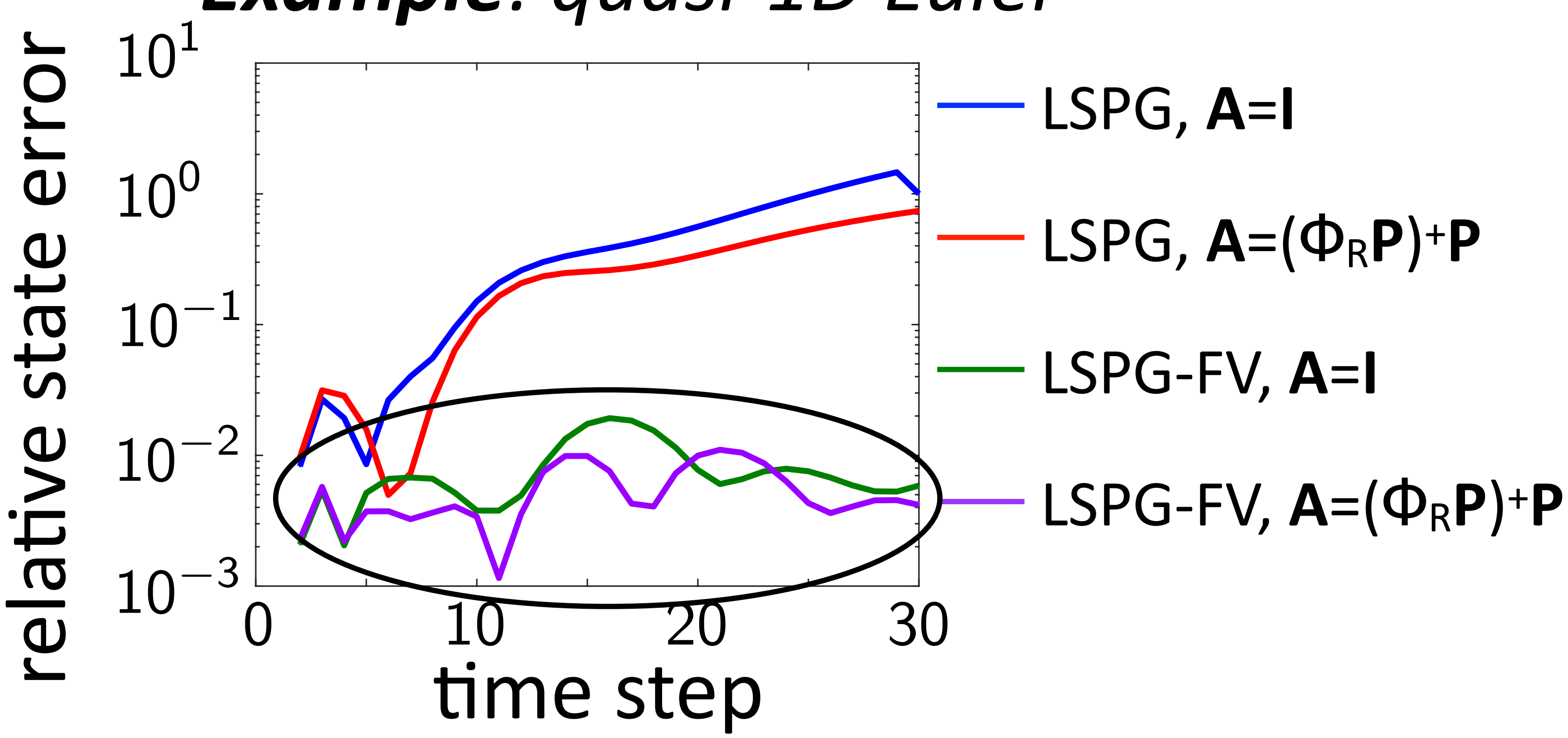
- minimize weighted sum of squared conservation-law violations over time step n

subject to global conservation

$$\bar{x}_i(t) = \frac{1}{|\Omega|}\int_\Omega u_i(\vec{x}, t)\,\mathrm{d}\vec{x}$$

$$\bar{f}_i(\mathbf{x}, t) = -\frac{1}{|\Omega|}\int_\Gamma \mathbf{g}_i(\mathbf{x}; \vec{x}, t)\cdot \mathbf{n}_j(\vec{x})\,\mathrm{d}\vec{s}(\vec{x}) + \frac{1}{|\Omega|}\int_\Omega s_i(\mathbf{x}; \vec{x}, t)\,\mathrm{d}\vec{x}$$

***Example**: quasi-1D Euler*



speedup

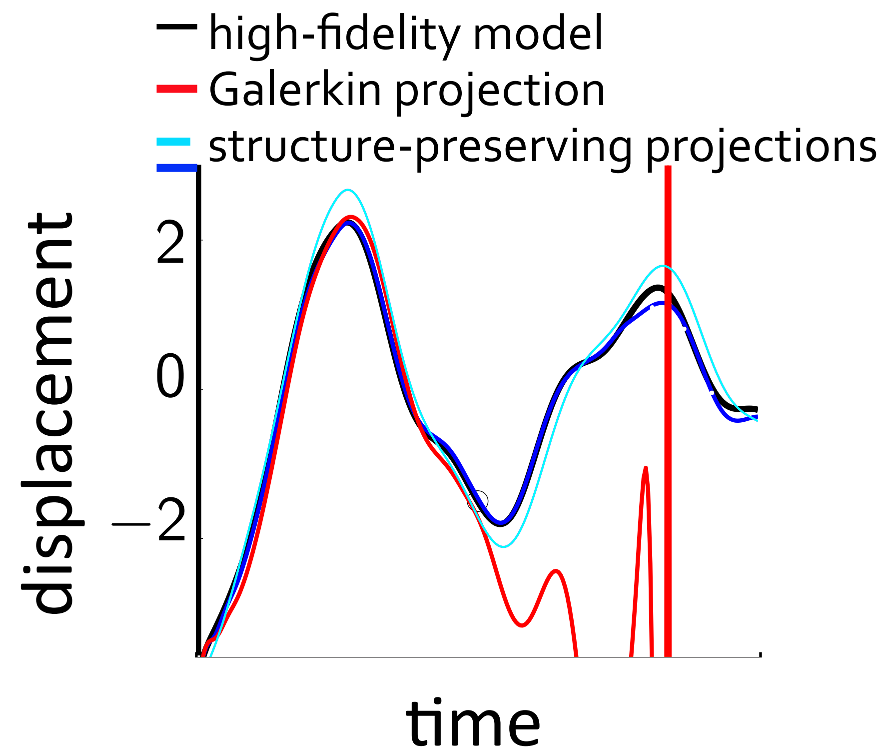
	LSPG	LSPG-FV
A=I	0.57	0.44
A=(Φ$_R$P)$^+$P	4.4	5.3

- structure preservation improves accuracy
- sample mesh improves wall time

Structure preservation

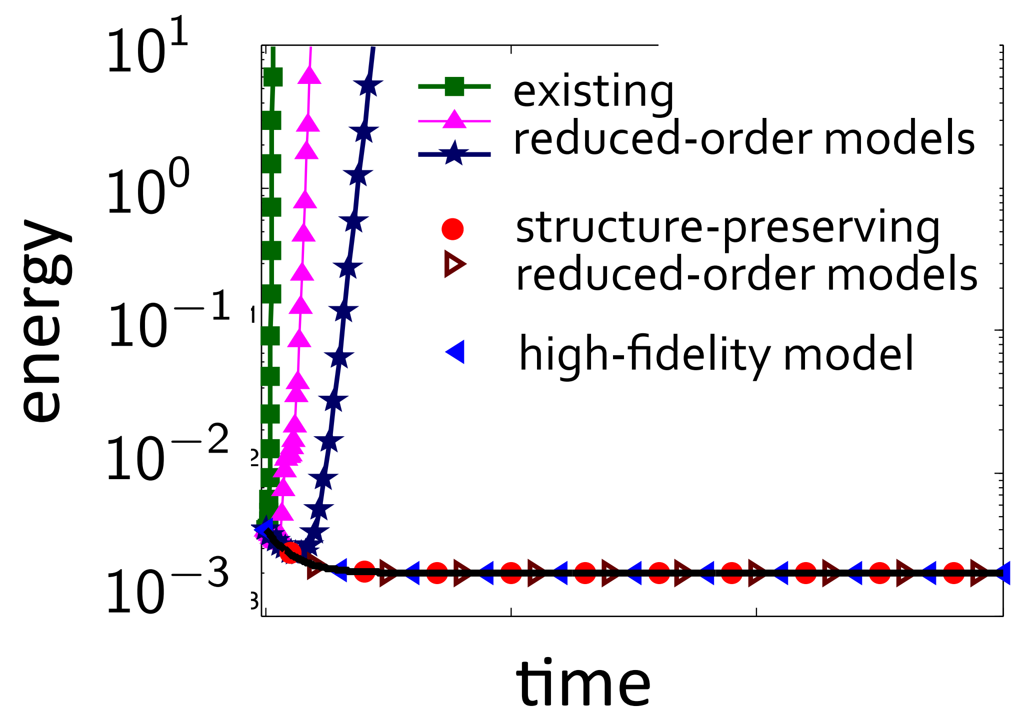
Nonlinear Lagrangian dynamical systems
[Carlberg, Tuminaro, Boggs, 2015]

- approximates Lagrangian ingredients, then derives equations of motion
- + ensures symplectic time evolution
- + conserves total energy



Preserving marginal stability (LTI systems)
[Peng and Carlberg, 2017]

- applies symplectic projection to ensure ROM has purely imaginary poles
- + guarantees finite infinite-time energy
- + enables extension of balanced truncation



Our research

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

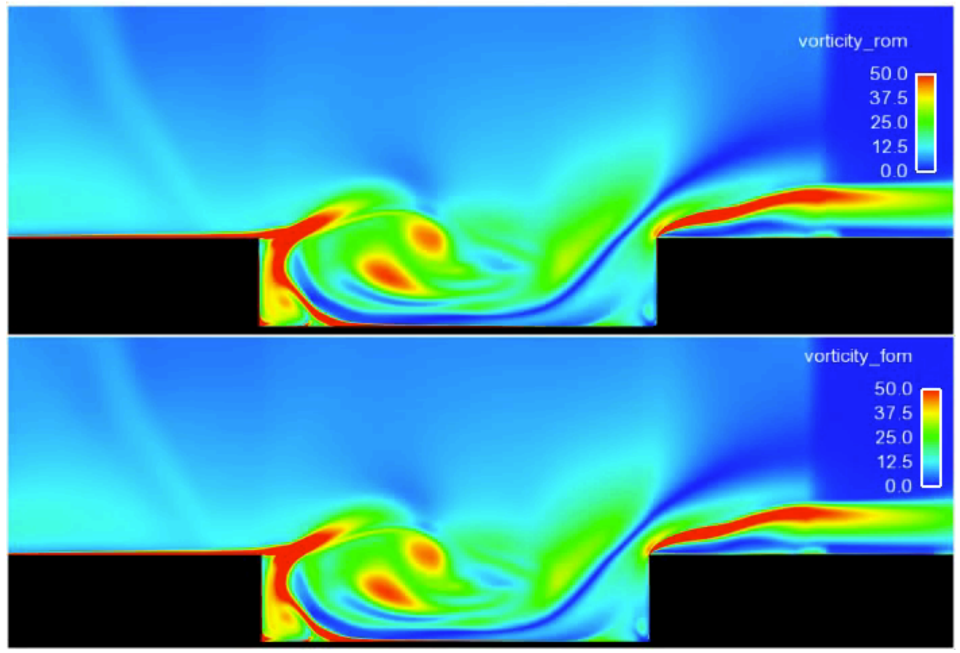
- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity
  [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models
  [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

Model reduction can work well…

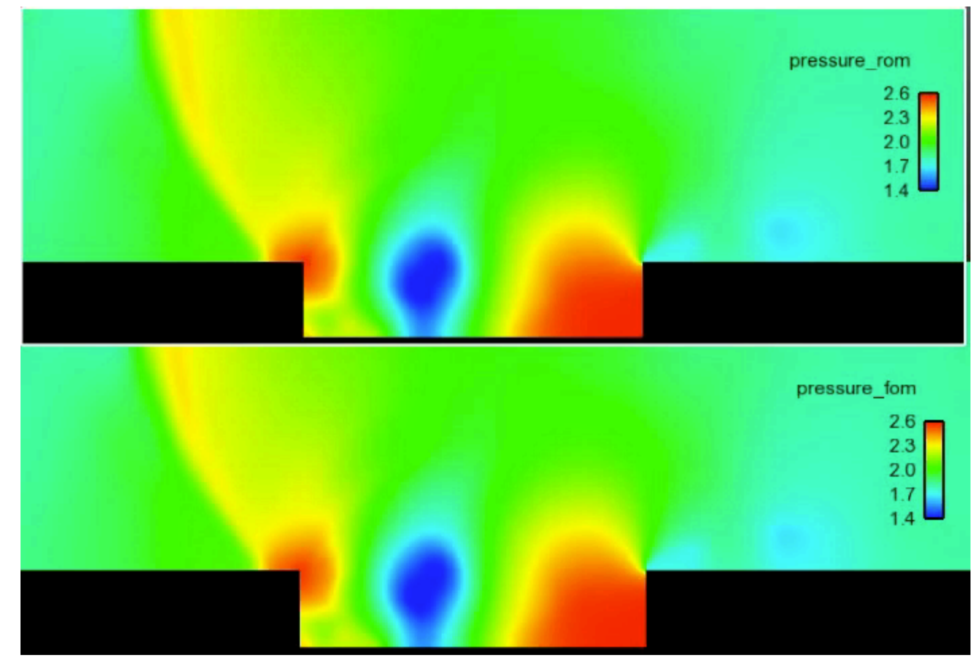
LSPG ROM with
$\mathbf{A} = (\mathbf{P\Phi_r})^{+}\mathbf{P}$
32 min, 2 cores

high-fidelity
5 hours, 48 cores

vorticity field



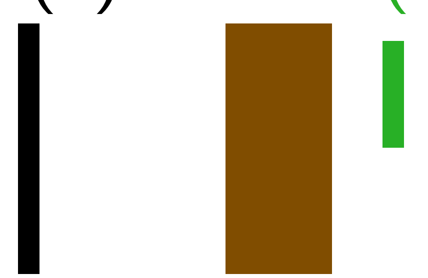
pressure field



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

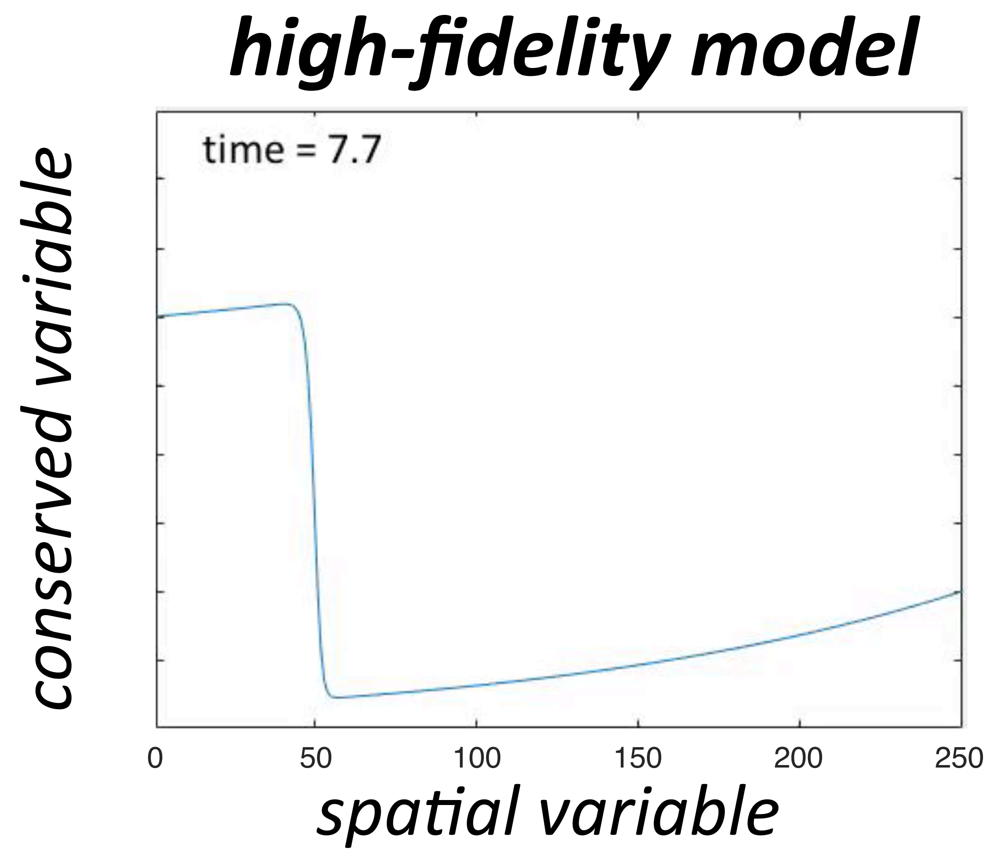
… however, this is not guaranteed

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



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

Illustration: inviscid 1D Burgers' equation



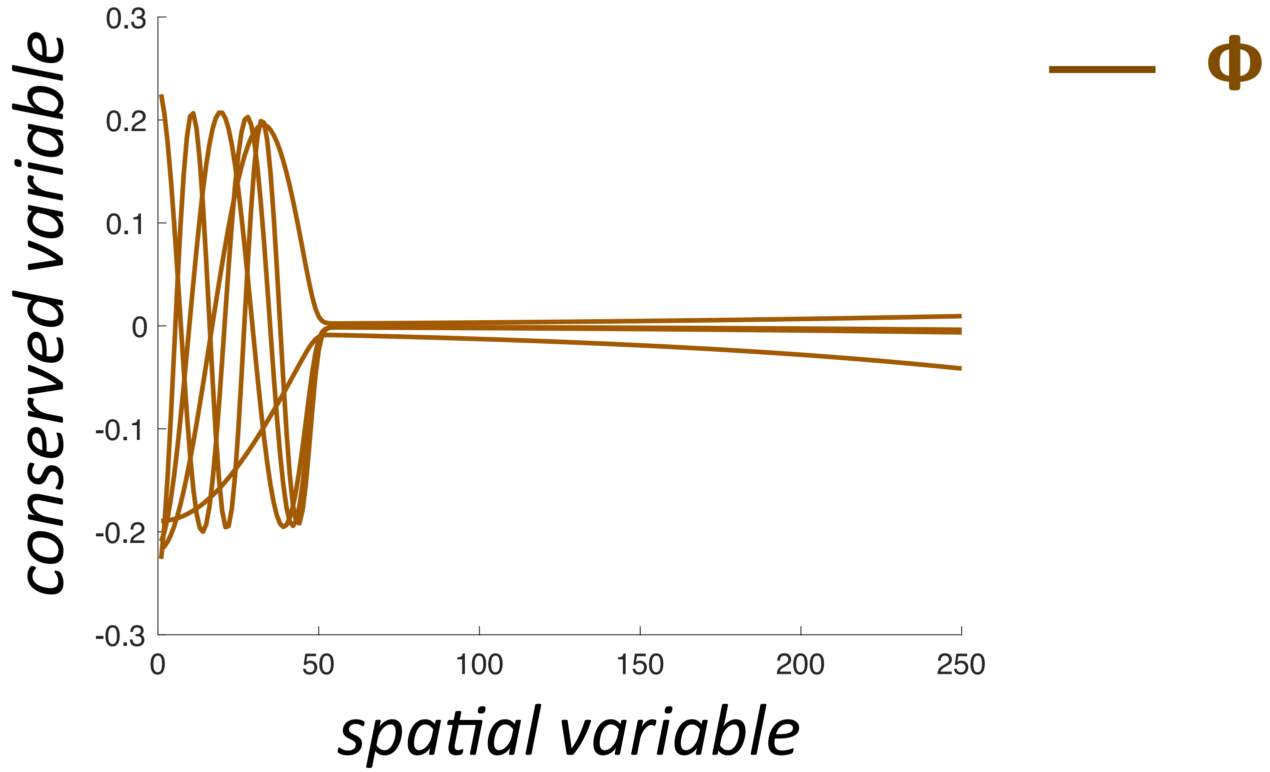
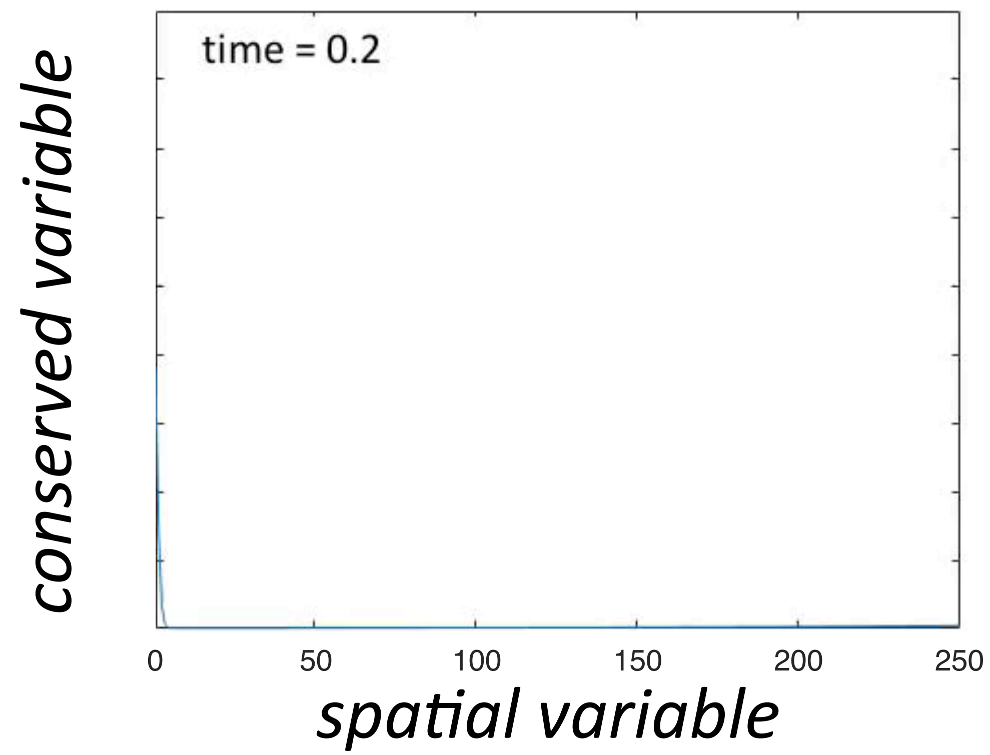
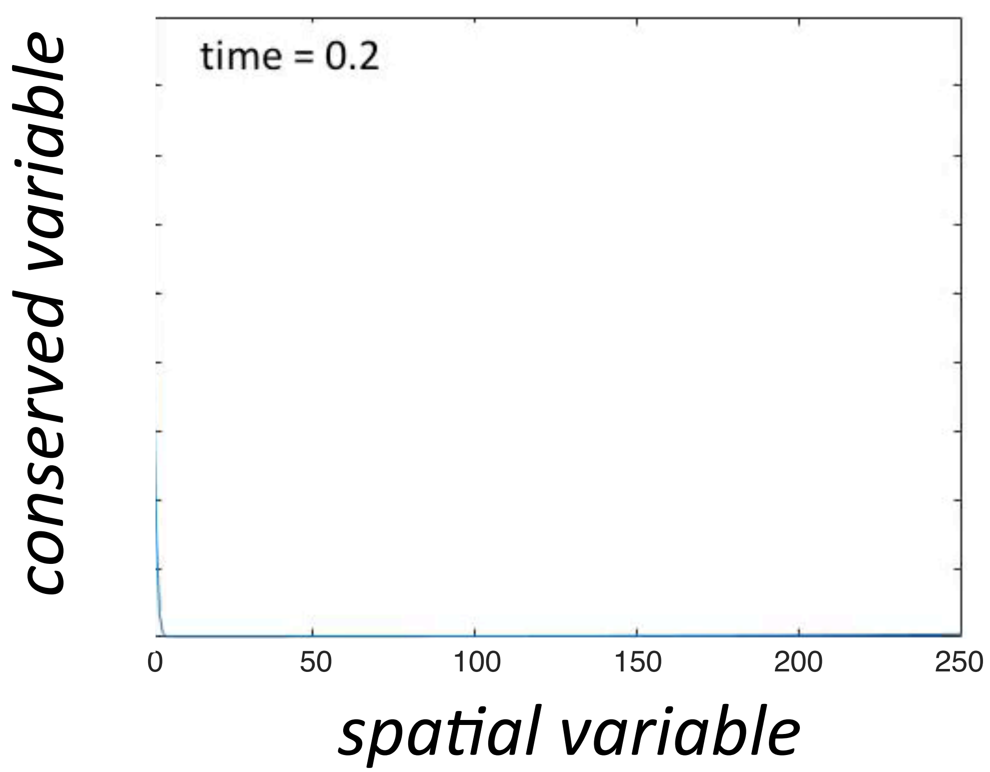


Illustration: inviscid 1D Burgers' equation

high-fidelity model



reduced-order model



reduced-order model inaccurate when Φ insufficient

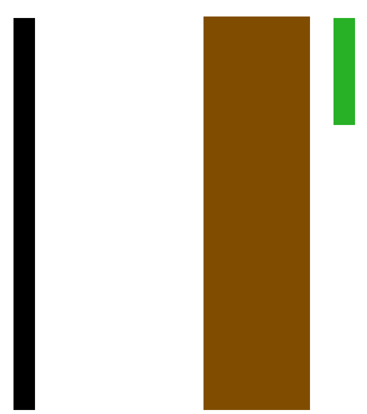
Key insight

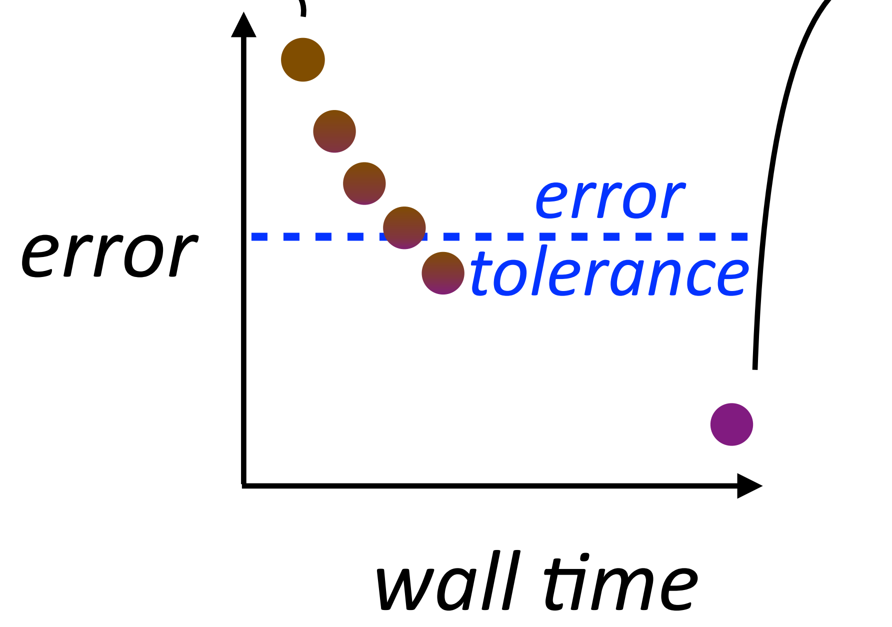
$$\mathbf{x}^n = \underset{\mathbf{v}\in\mathcal{S}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{r}^n(\mathbf{v})\|_2^2$$

reduced-order model

$\mathcal{S} = \operatorname{range}(\mathbf{\Phi}) \subset \mathbb{R}^N$

$\mathbf{x}^n = \mathbf{\Phi}\,\hat{\mathbf{x}}^n$

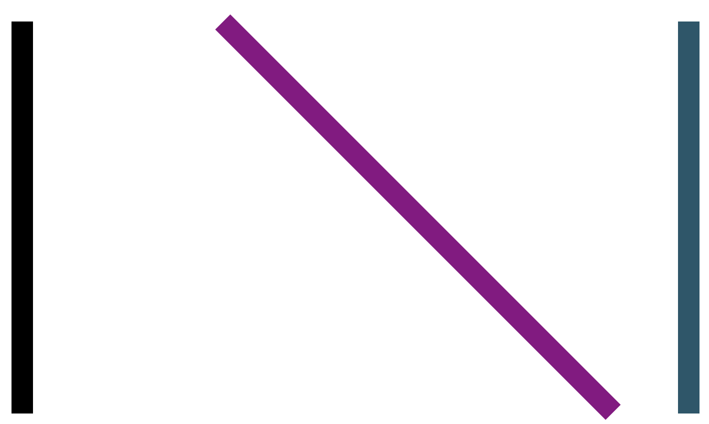




high-fidelity model

$\mathcal{S} = \operatorname{range}(\mathbf{I}) = \mathbb{R}^N$

$\mathbf{x}^n = \mathbf{I}\,\hat{\mathbf{x}}^n$



Idea: *the data provide an initial, low-dim subspace that* ***can be refined*** *to satisfy any error tolerance*

1. Generalization of mesh-adaptive *h*-refinement [Carlberg, 2015]

$$\mathcal{S} = \operatorname{range}(\mathbf{\Phi}_{h\text{-refine}}) \supset \operatorname{range}(\mathbf{\Phi})$$

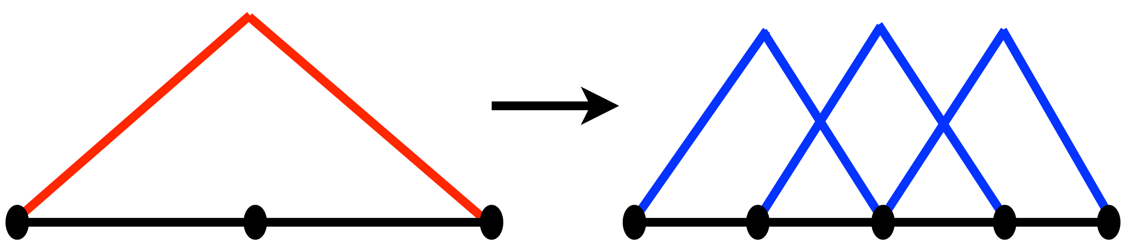
2. Augmented Krylov method [Carlberg, Forstall, Tuminaro, 2016]

$$\mathcal{S} = \operatorname{range}(\mathbf{\Phi}) + \mathcal{K}(A, b)$$

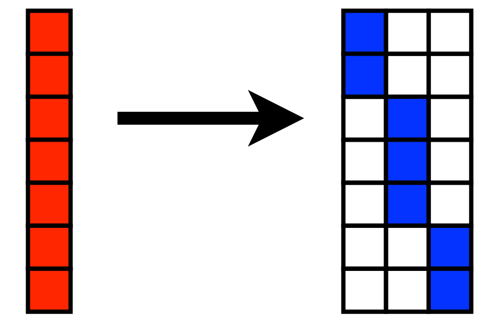
Main idea [Carlberg, 2015]

Model-reduction analogue to mesh-adaptive h-refinement

- 'Split' basis vectors

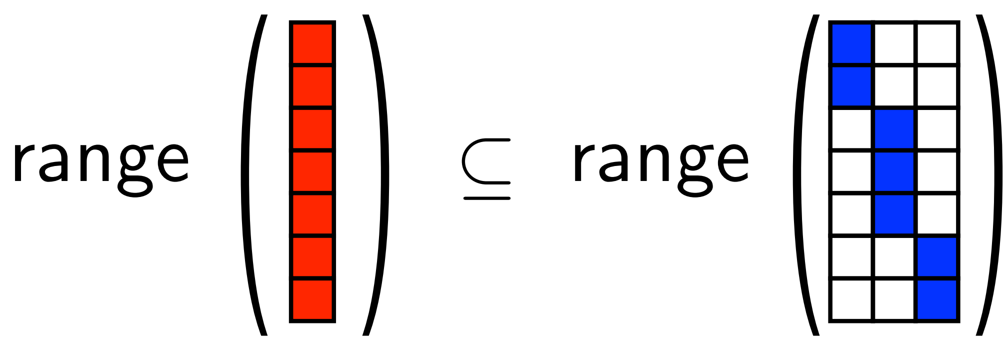


finite-element h-refinement

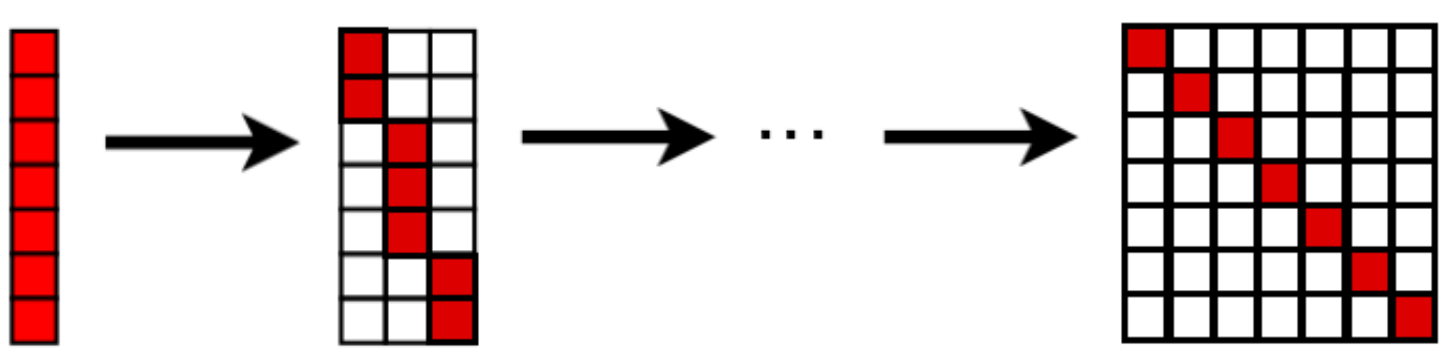


reduced-order-model h-refinement

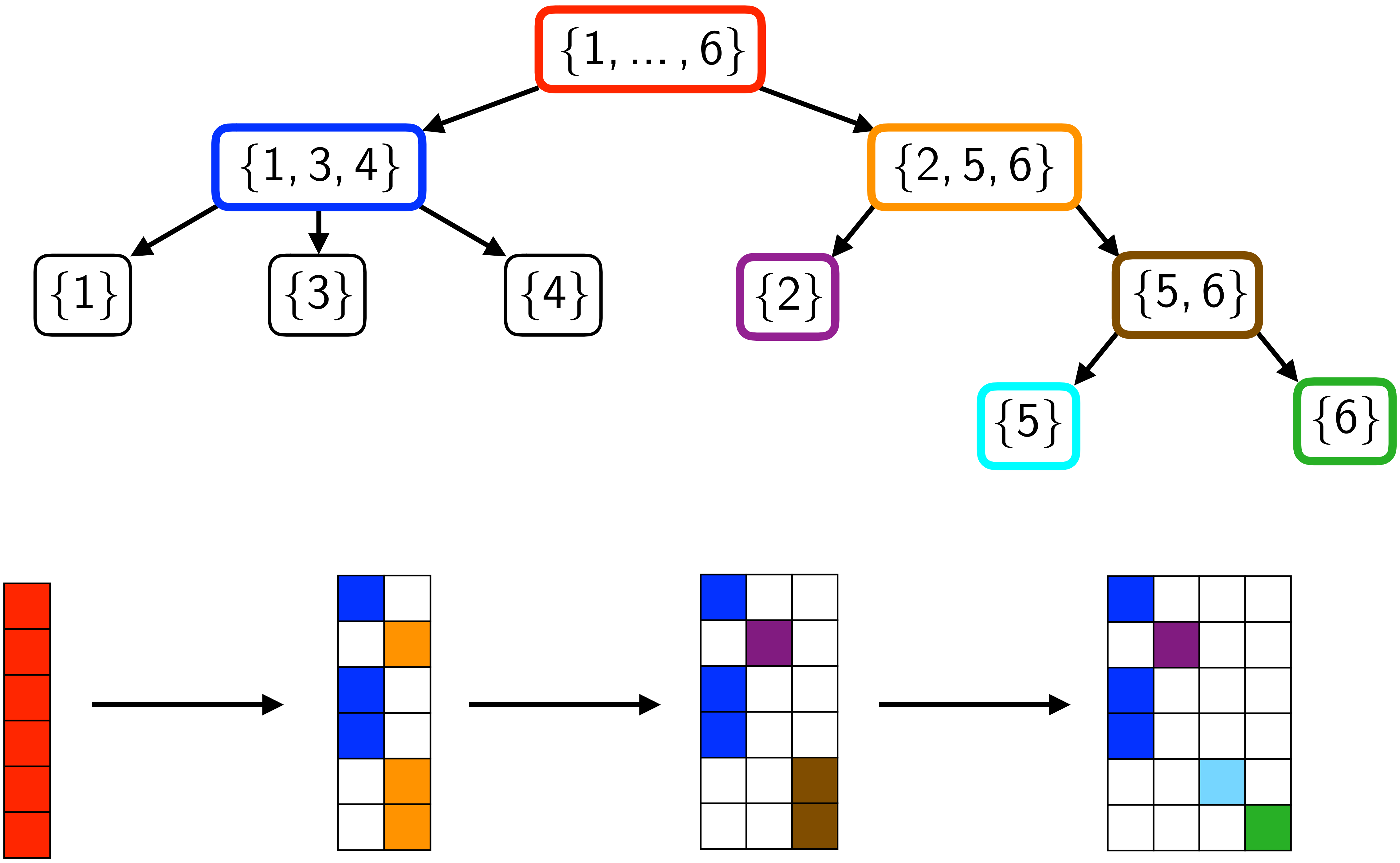
- Generate hierarchical subspaces



- Converges to the high-fidelity model



Tree encodes splitting

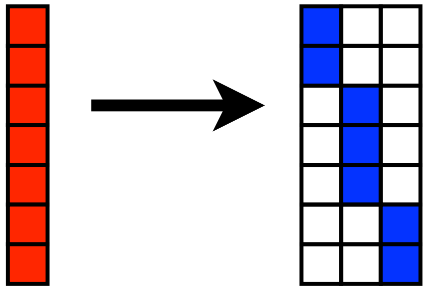


Tree requirements

Theorem [Carlberg, 2015]

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

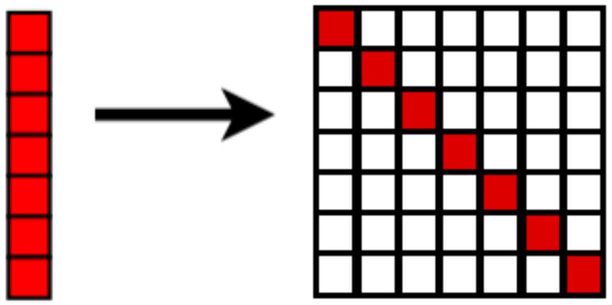
1. children have disjoint support, and
2. the union of the children elements is equal to the parent elements



Theorem [Carlberg, 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.



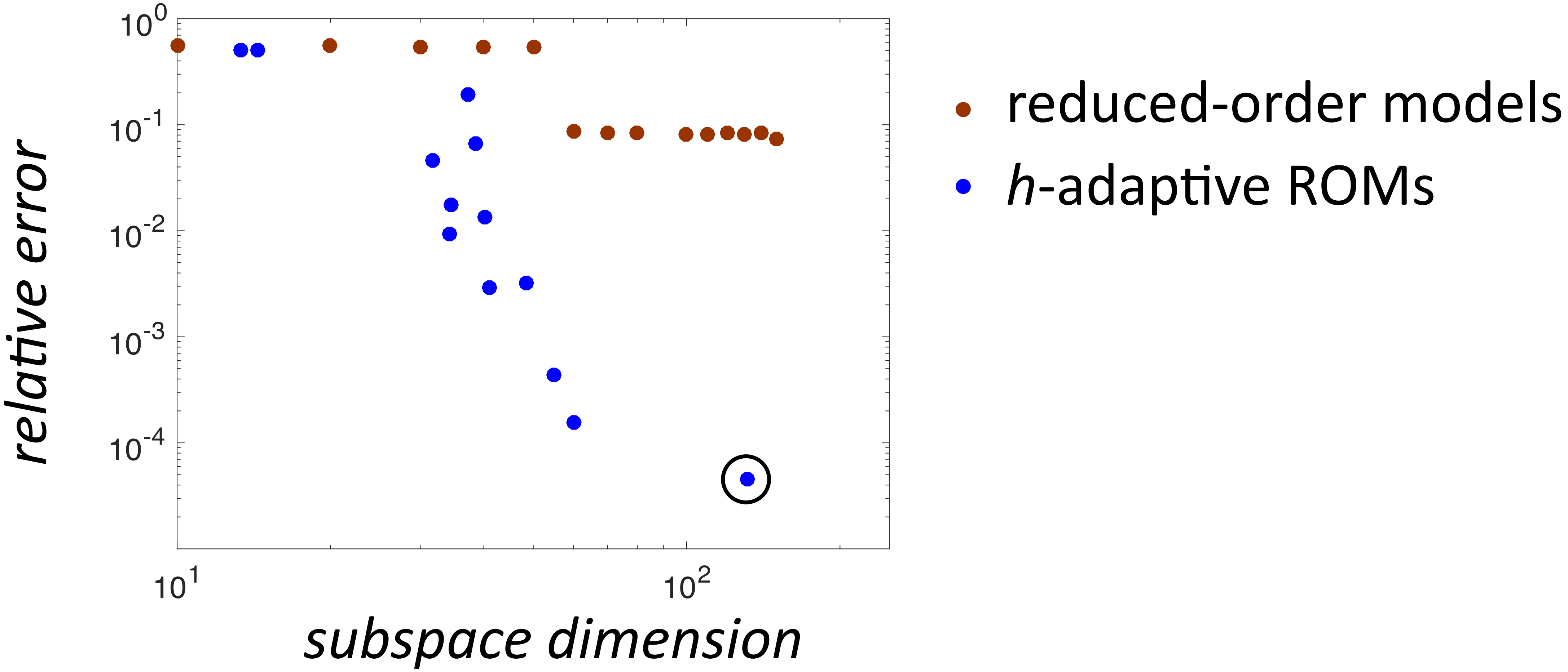
Tree-construction algorithm

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

Which vectors to split?

- Dual-weighted-residual error estimation

h-adaptivity provides an accurate, low-dim subspace



Reduced-order models

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

h-adaptive ROMs

+ *minimum error <0.01% with lower subspace dimension*
+ *can overcome insufficient training data without collecting more data*
+ *can satisfy any prescribed error tolerance*

Our research

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

- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models [Drohmann and Carlberg, 2015*; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2018]

* Top 5 most cited papers, SIAM/ASA JUQ, 2015

Discrete-time error bound

Theorem [Carlberg, Antil, Barone, 2017]

If the following conditions hold:

1. $\mathbf{f}(\cdot;t)$ is Lipschitz continuous with Lipschitz constant κ
2. The time step Δt is small enough such that $0 < h := |\alpha_0| - |\beta_0|\kappa\Delta t$,
3. A backward differentiation formula (BDF) time integrator is used,
4. LSPG employs $\mathbf{A} = \mathbf{I}$, then

$$\|\mathbf{x}^n - \boldsymbol{\Phi}\hat{\mathbf{x}}_{\mathrm{G}}^n\|_2 \leq \frac{\gamma_1(\gamma_2)^n \exp(\gamma_3 t^n)}{\gamma_4 + \gamma_5 \Delta t} \max_{j\in\{1,\ldots,N\}} \|\mathbf{r}_{\mathrm{G}}^j(\boldsymbol{\Phi}\hat{\mathbf{x}}_{\mathrm{G}}^j)\|_2$$

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

Can we use these error bounds for error estimation?

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

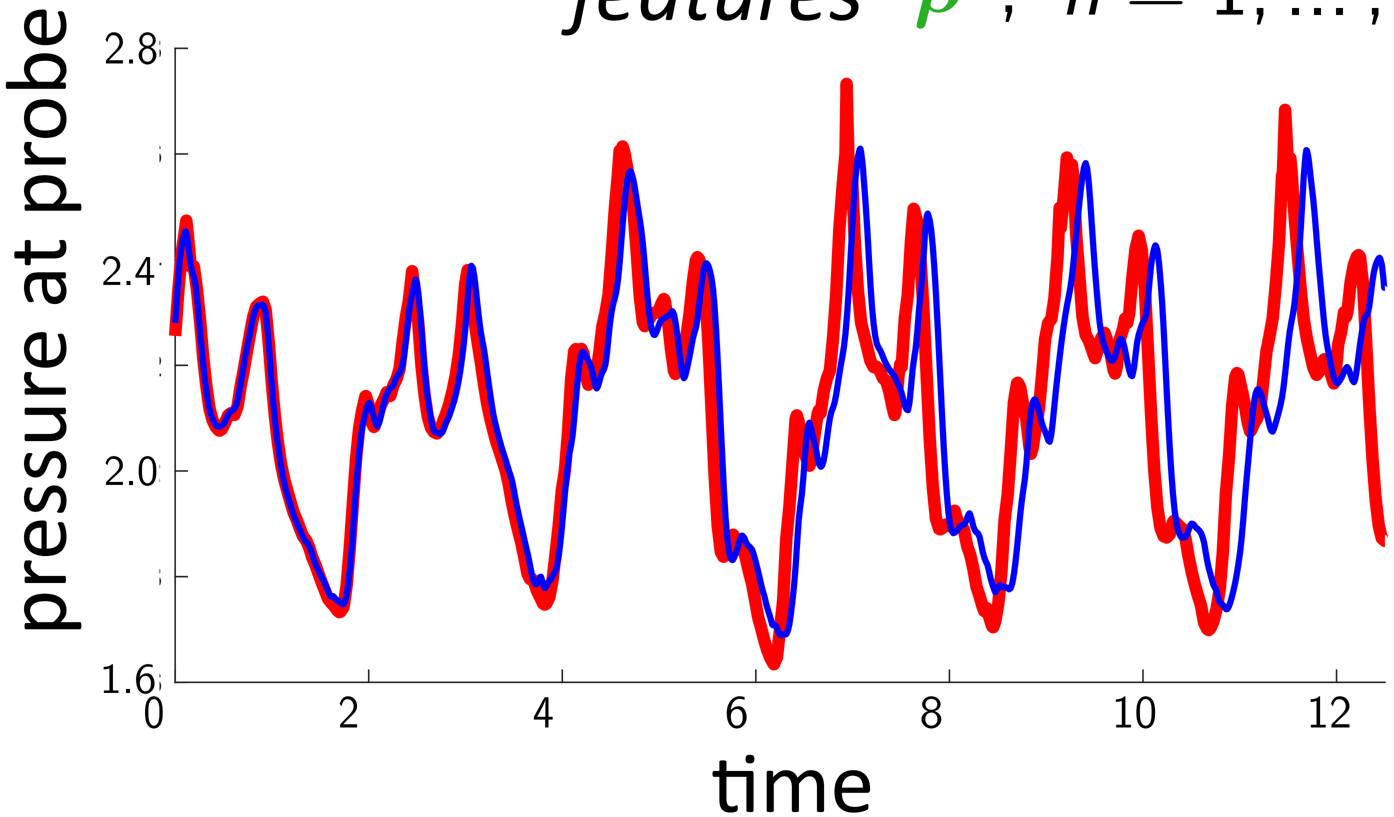
Key insight

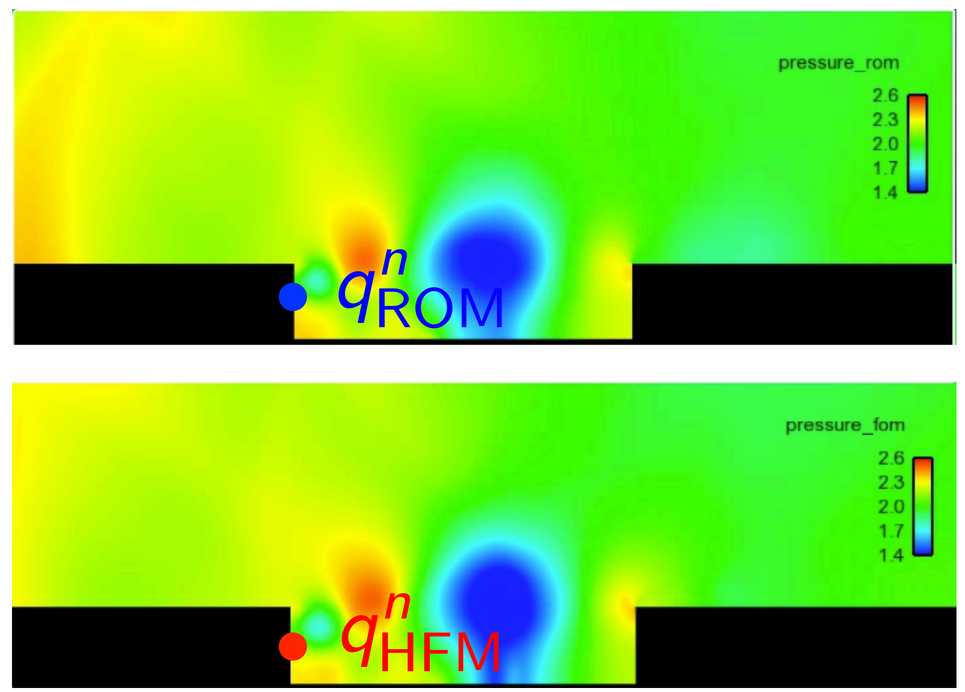
inputs $\boldsymbol{\mu}$ $\rightarrow$ *high-fidelity model* $\rightarrow$ *outputs* $q^n_{\mathrm{HFM}},\; n = 1, \ldots, T$

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

$\downarrow$

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



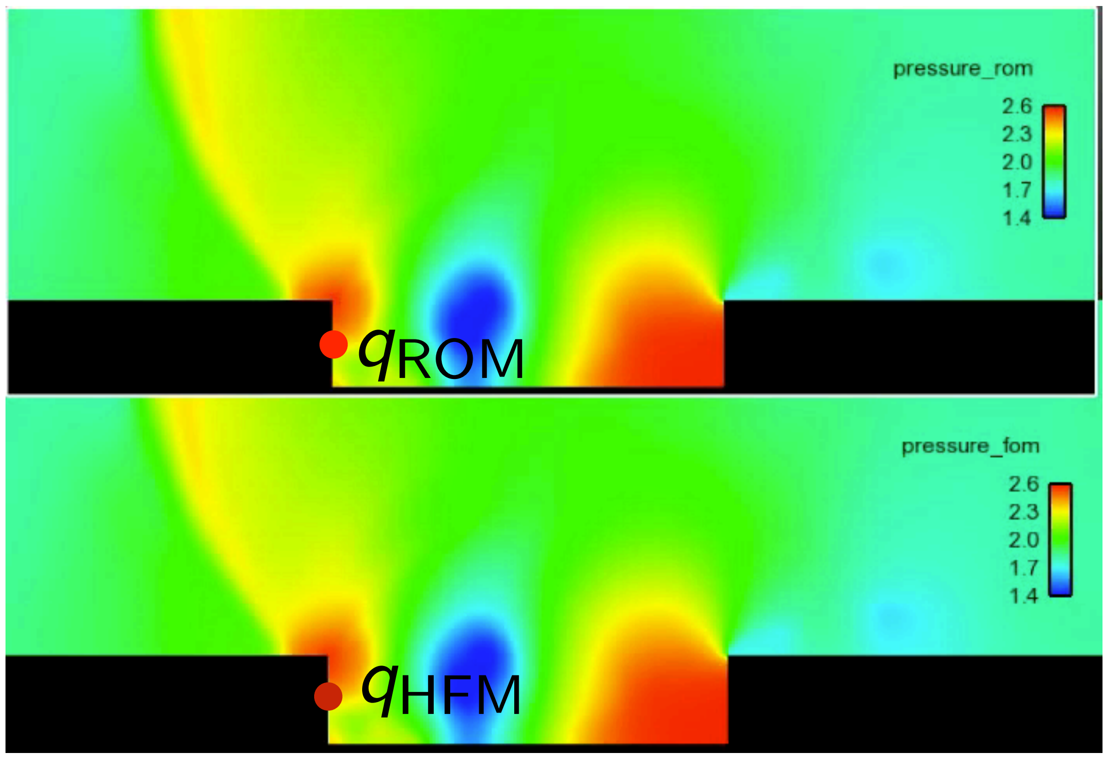


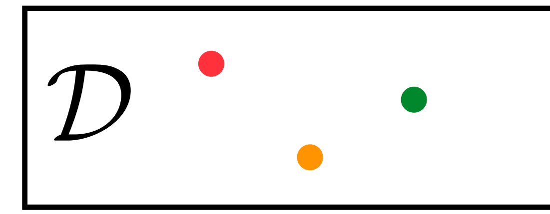
Reduced-order models generate features $\boldsymbol{\rho}^n$ that may inform its error

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

Training and machine learning: error modeling

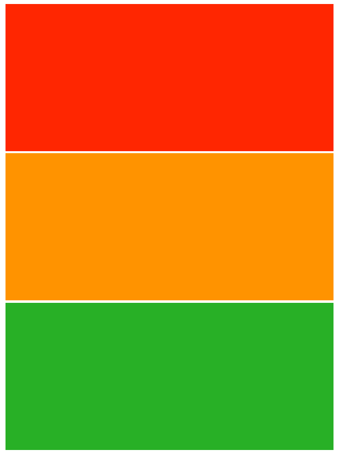
1. *Training:* Solve high-fidelity and reduced-order models for $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict reduced-order-model error for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$





$q^n_{\text{HFM}} - q^n_{\text{ROM}}$

$\boldsymbol{\rho}^n$

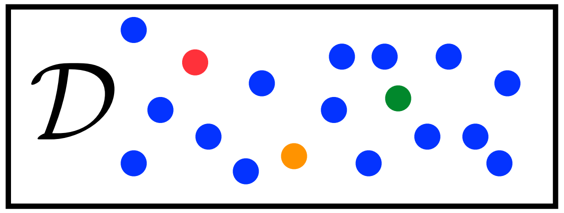


features $\boldsymbol{\rho}^n$ $\longrightarrow$ *regression model* $\longrightarrow$ *error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$

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

Regression model for the error

1. *Training:* Solve high-fidelity and reduced-order models for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$
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* $q^n_{\text{ROM}}, \; n = 1, \ldots, T$

↓

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

↓

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

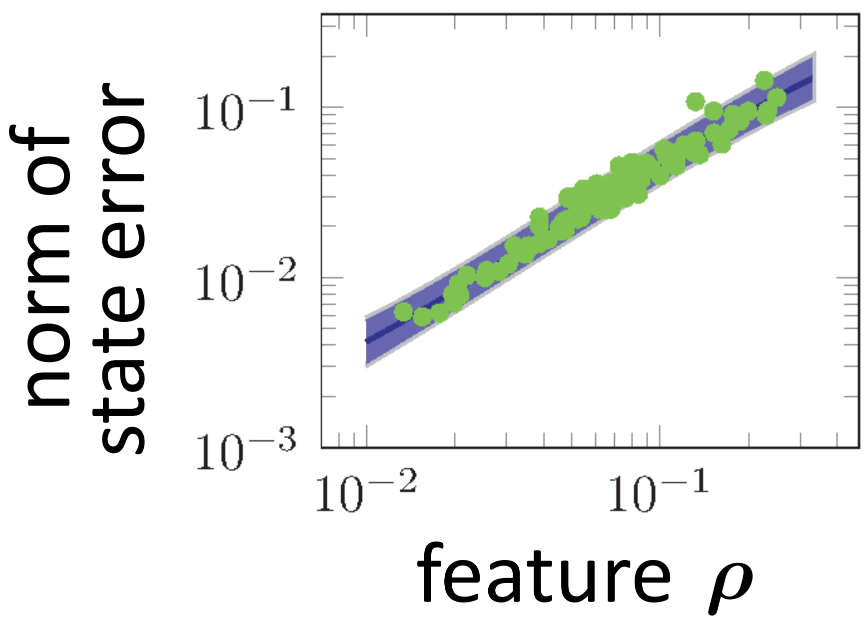
$$\boxed{q^n_{\text{HFM}} \approx q^n_{\text{ROM}} + \hat{\delta}^n}$$

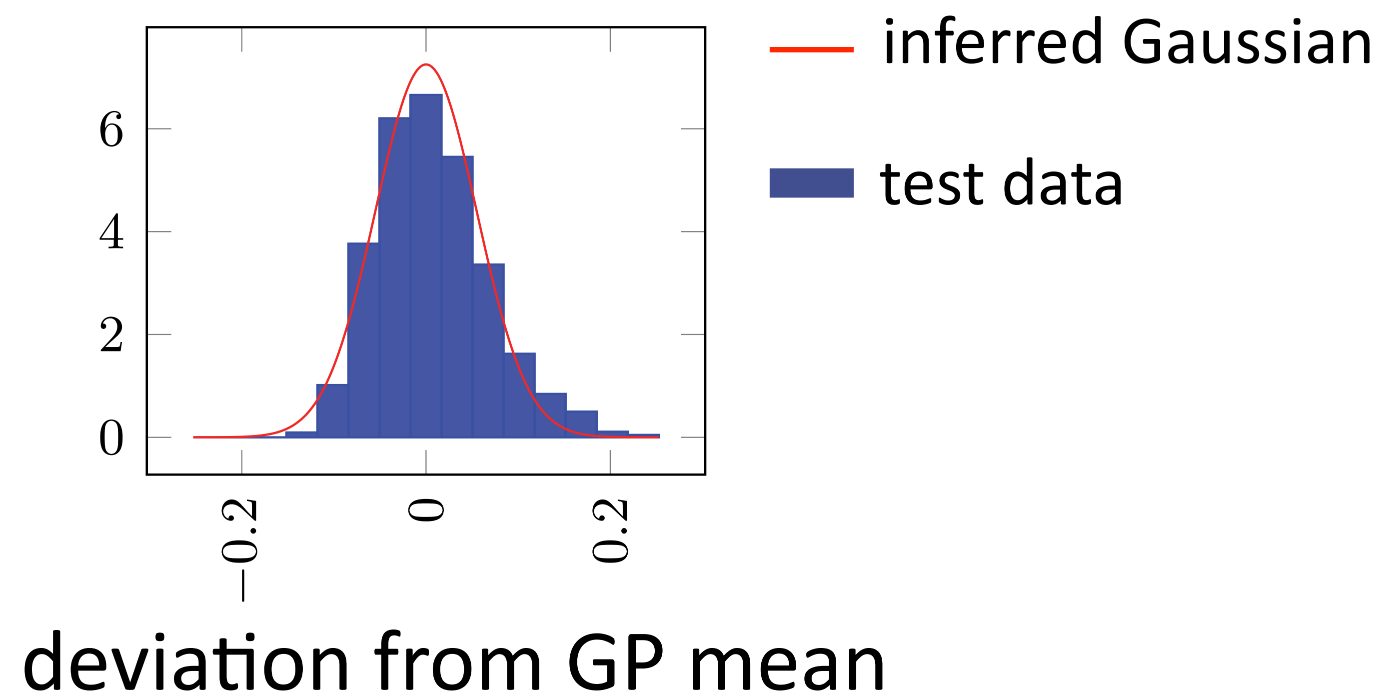
+ *Statistical model of high-fidelity-model output*

Physics-based feature engineering to determine ρ^n

Application 1: Poisson equation [Drohmann, Carlberg, 2015]

- *error:* norm of state error $\|\mathbf{x} - \boldsymbol{\Phi}\hat{\mathbf{x}}\|$
- *1 feature* ρ *:* residual norm $\|\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}})\|_2$
- *regression:* Gaussian process

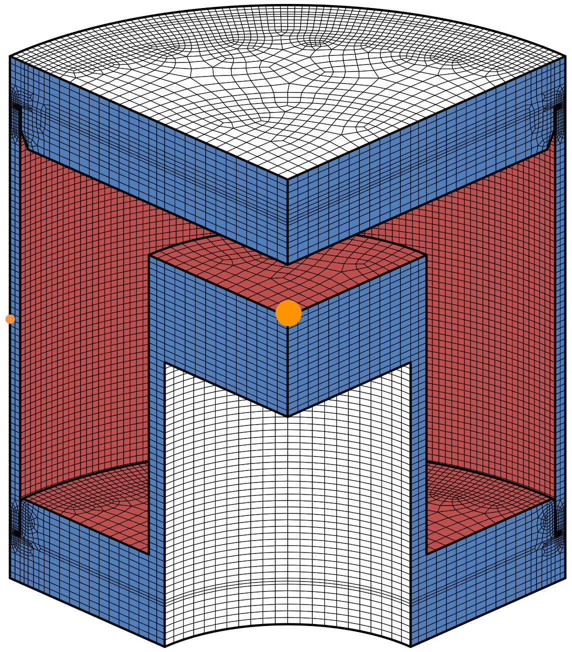




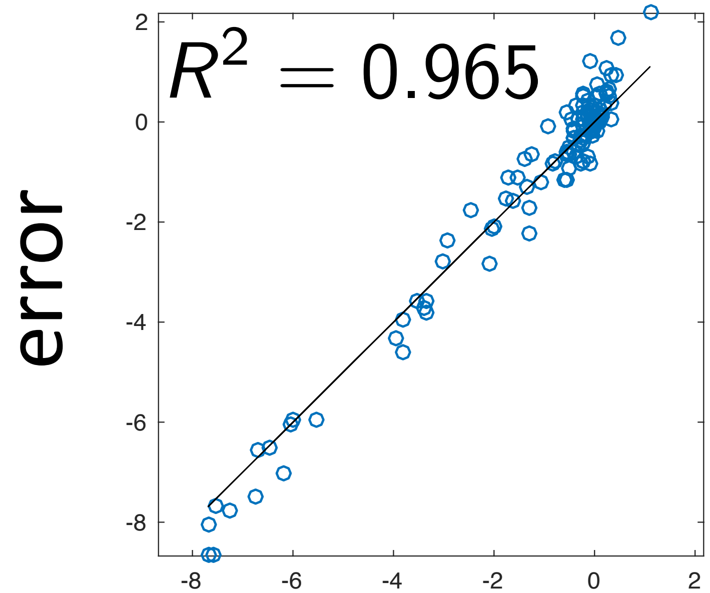
+ low-variance model of the error
+ numerically validated on test set
- error bound overproduction as high as 8.0

Application 2: nonlinear static mechanical response

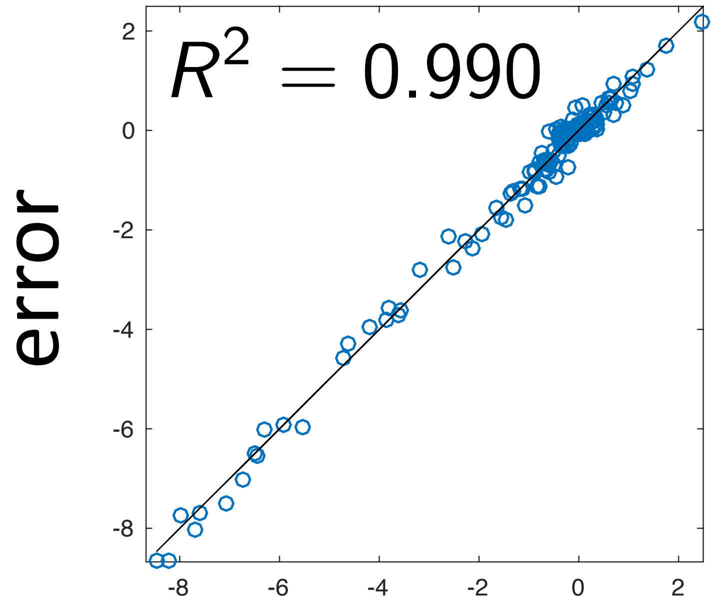
[Freno, Carlberg, 2017]



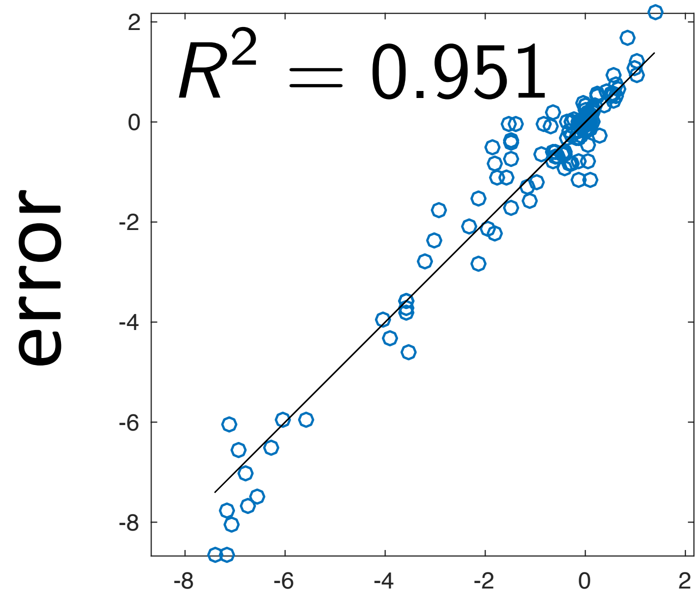
- *high-fidelity model dimension:* 2.8×10^5
- *reduced-order model dimension:* 6
- *inputs $\boldsymbol{\mu}$:* elastic modulus, Poisson ratio
- *error:* error in *y*-displacement at point
- *50 features $\boldsymbol{\rho}$:* residual approx $(\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^{+}\mathbf{P}\mathbf{r}^{n}$,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

- *Other application*: nonlinear oil–water flow [Trehan, Carlberg, Durlofsky, 2017]

Summary

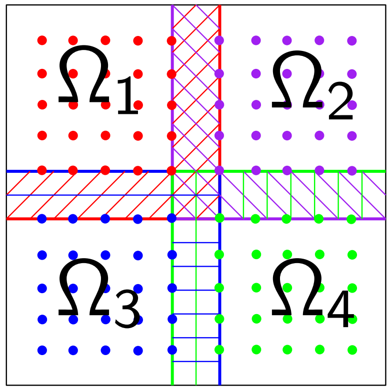
Accurate, low-cost, structure-preserving, reliable, certified nonlinear model reduction

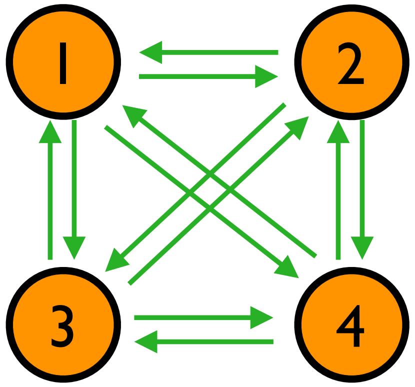
- *accuracy*: LSPG projection [Carlberg, Bou-Mosleh, Farhat, 2011; Carlberg, Antil, Barone, 2017]
- *low cost*: sample mesh [Carlberg, Farhat, Cortial, Amsallem, 2013]
- *low cost*: reduce temporal complexity
  [Carlberg, Ray, van Bloemen Waanders, 2015; Carlberg, Brencher, Haasdonk, Barth, 2017; Choi and Carlberg, 2017]
- *structure preservation* [Carlberg, Tuminaro, Boggs, 2015; Peng and Carlberg, 2017; Carlberg and Choi, 2017]
- *reliability*: adaptivity [Carlberg, 2015]
- *certification*: machine learning error models
  [Drohmann and Carlberg, 2015; Trehan, Carlberg, Durlofsky, 2017; Freno and Carlberg, 2017]

Moving forward

*Make extreme-scale simulations **pervasive** for many-query problems*

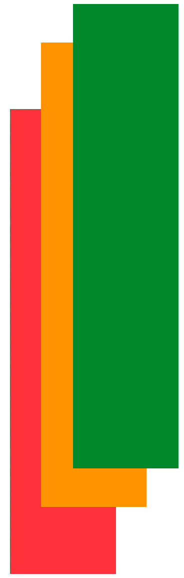
1. **Maximize impact for current methods**
   - implement promising techniques in large-scale codes
   - non-intrusive model reduction for rapid extensibility [Carlberg, Peng, Brunton, 2018]
2. **Extend model-reduction innovations to other applications**
   - stochastic LSPG [Lee, Carlberg, Elman, 2018]
   - data-driven iterative linear solvers [Carlberg, Forstall, Tuminaro, 2016]
3. **Reduce training requirements**
   - domain decomposition LSPG [Carlberg, Hoang 2018; Hoang, Carlberg, 2018]
     + training simulations for subsystems only
     + Primal–Schur solver: excellent weak scalability
4. **Extreme-scale uncertainty quantification**
   - domain decomposition UQ [Carlberg, Khalil, Guzzetti, Sargsyan, 2018]
     + uncertainty propagation for subsystems only
     + Multiplicative Schwarz: excellent weak scalability

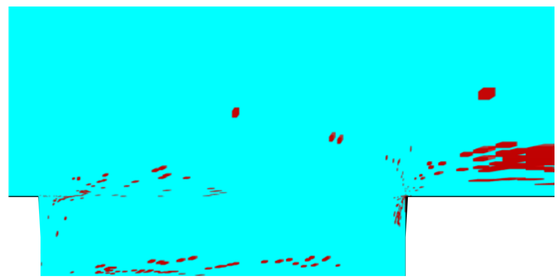


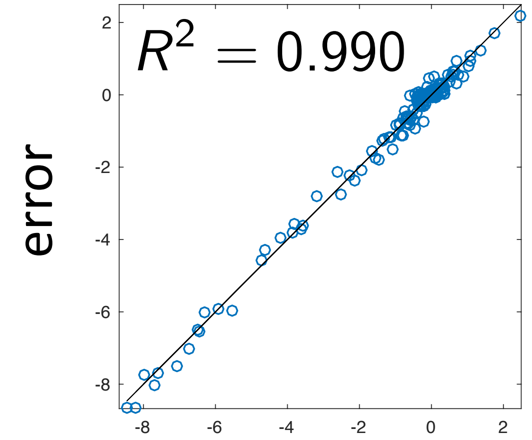


Questions?

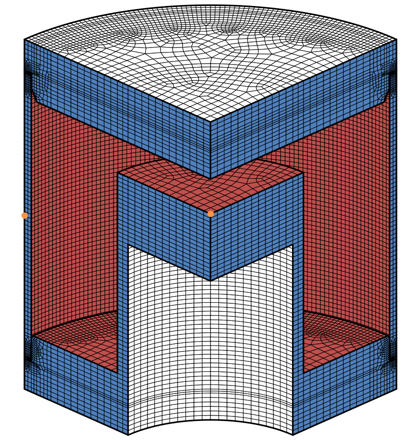
$\mathcal{X}^{ijk} =$

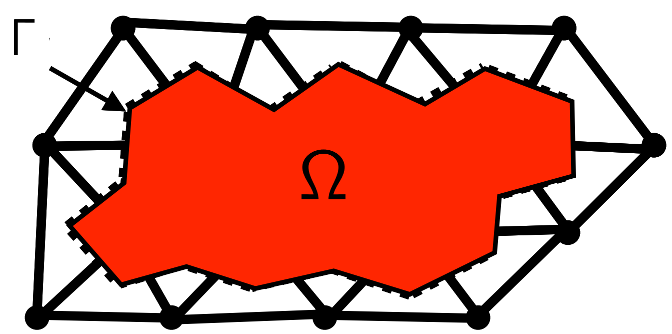


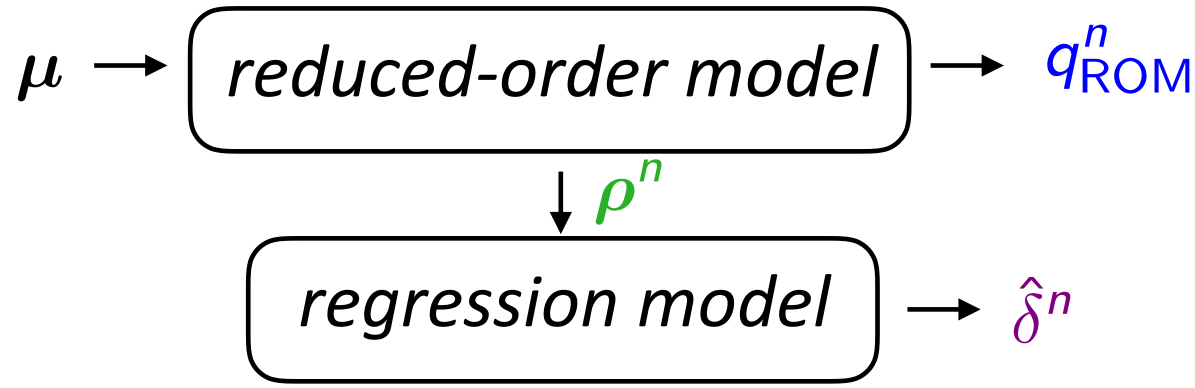


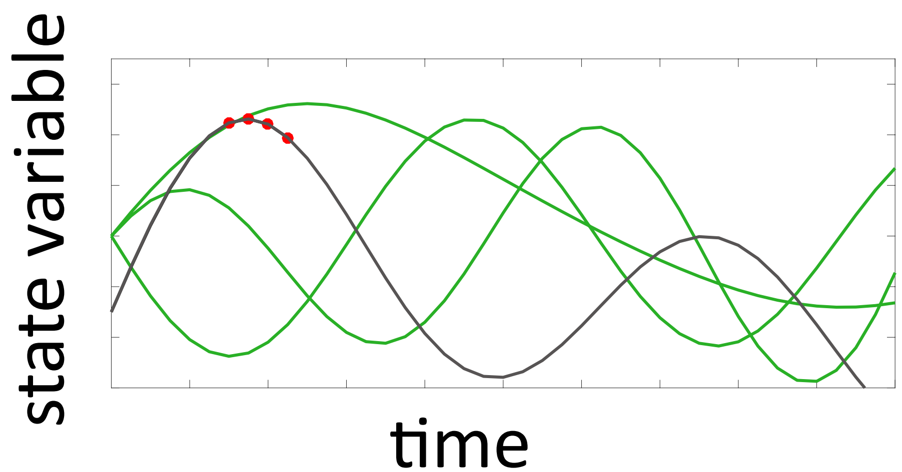


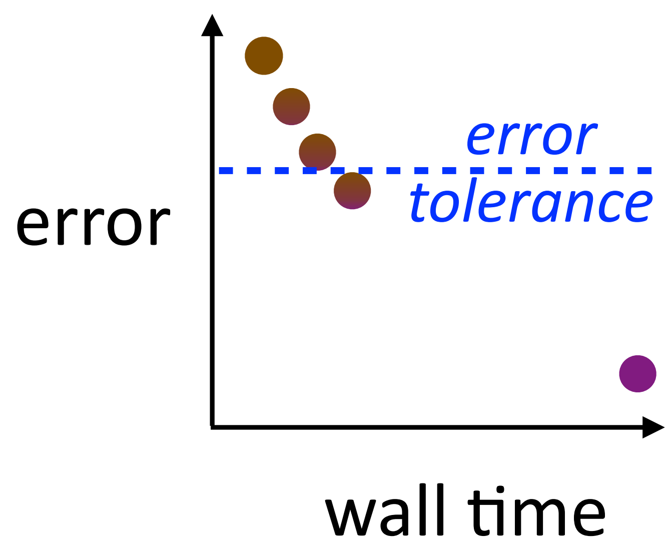
support vector machine
error prediction

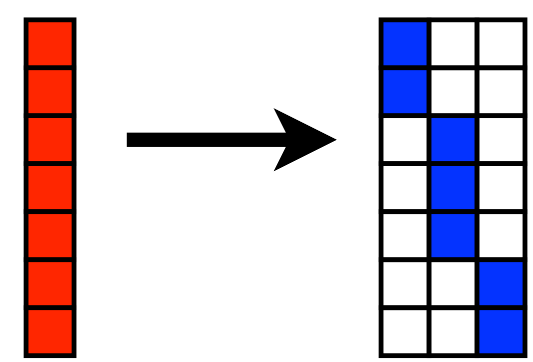












Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525