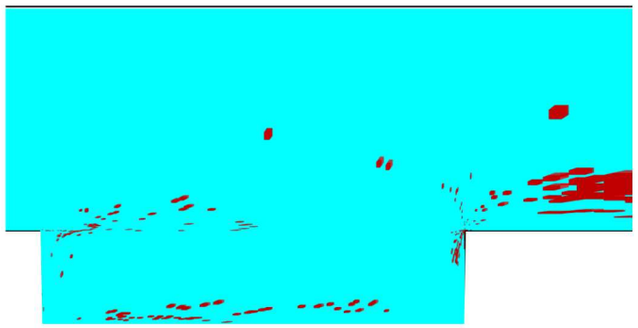
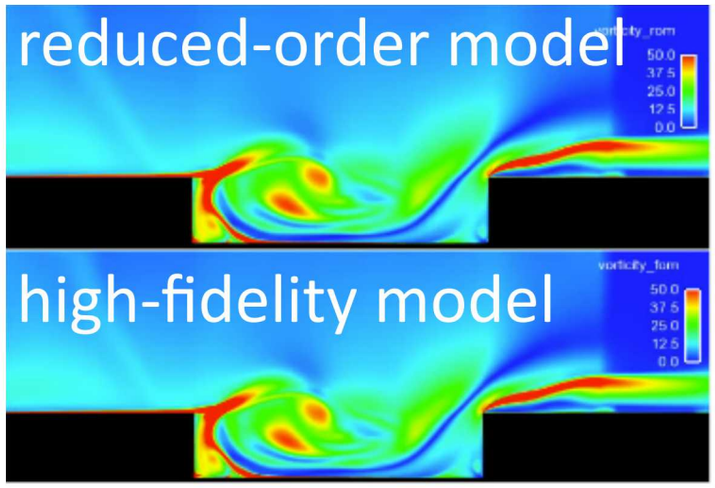
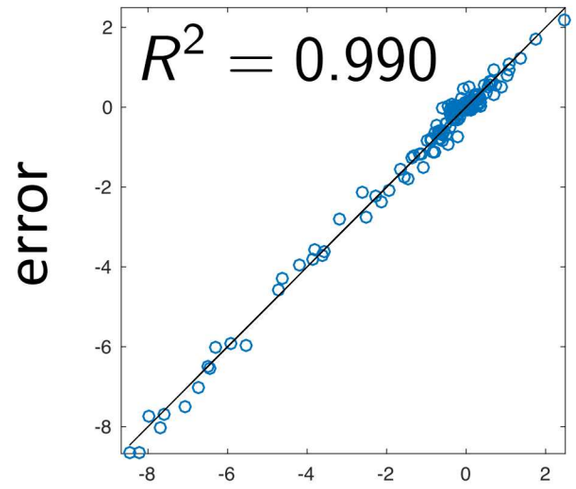
SAND2018-9130PE

Nonlinear reduced-order modeling

Using machine learning to enable extreme-scale simulation for many-query problems







support vector machine
error prediction

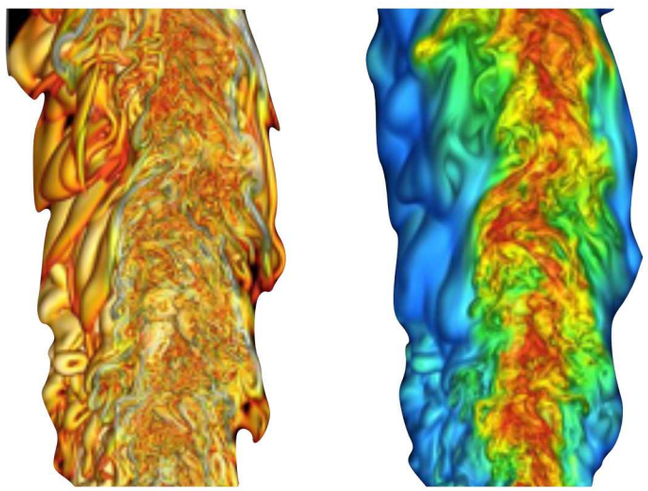
Kevin Carlberg

Sandia National Laboratories

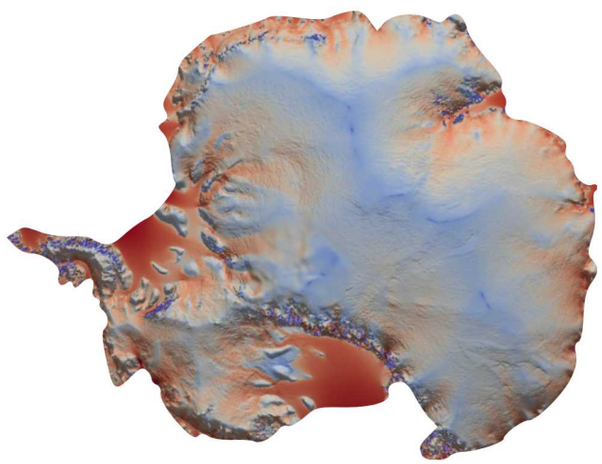
ASC Cognitive Simulation Initiative Meeting
Lawrence Livermore National Laboratory
August 16, 2018

High-fidelity simulation

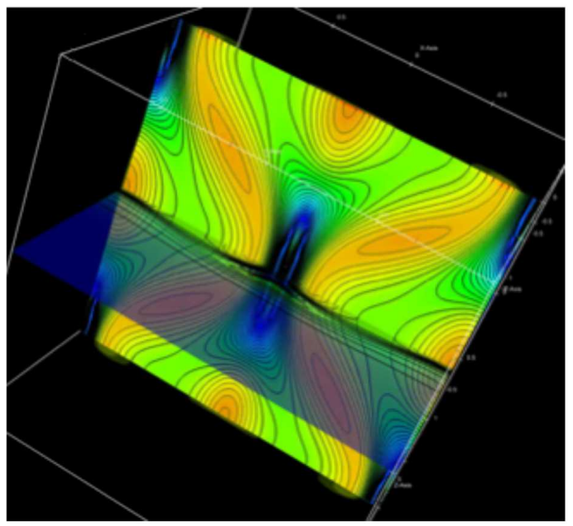
+ Indispensable for our NW and national-security missions
- *High fidelity*: extreme-scale nonlinear dynamical system models



Turbulent reacting flows
courtesy J. Chen, Sandia



Antarctic ice sheet modeling
courtesy R. Tuminaro, Sandia



Magnetohydrodynamics
courtesy J. Shadid, Sandia

computational barrier

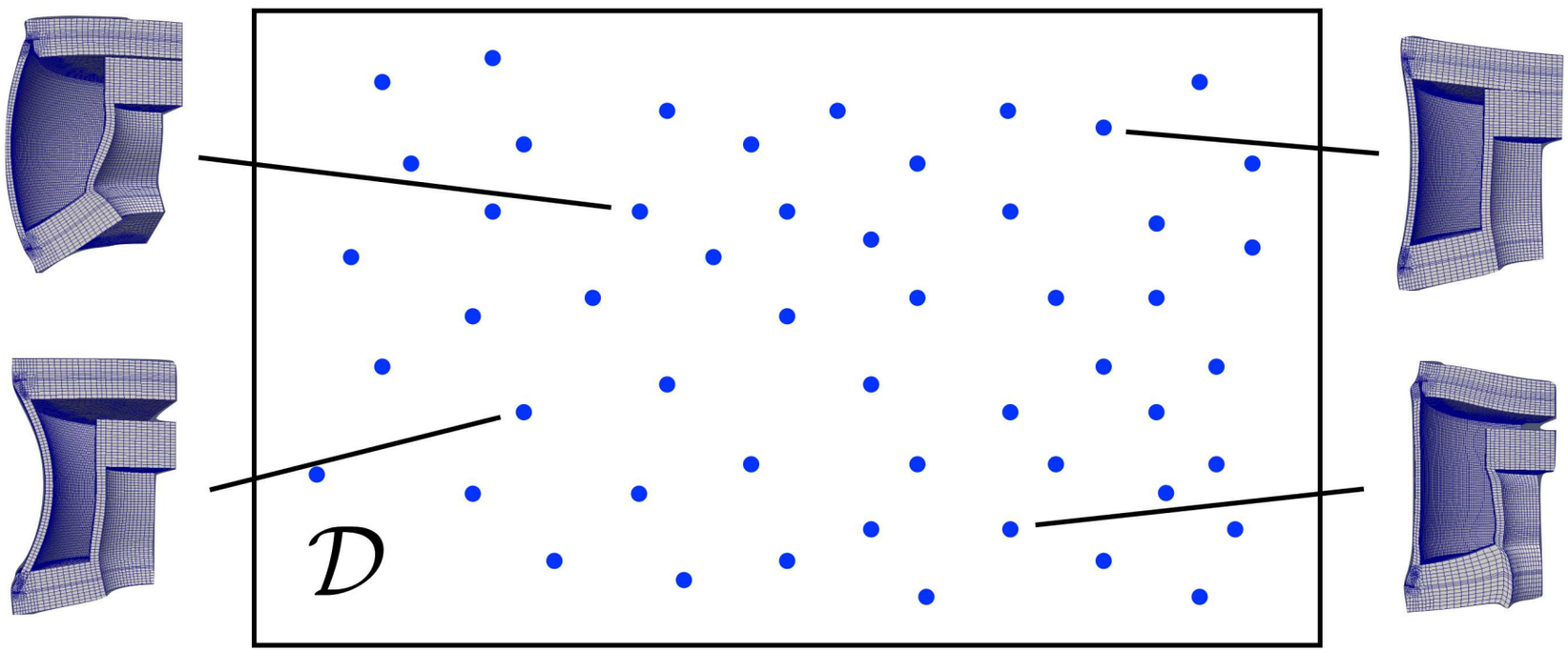
Many-query problems

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

Approach: exploit simulation data

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

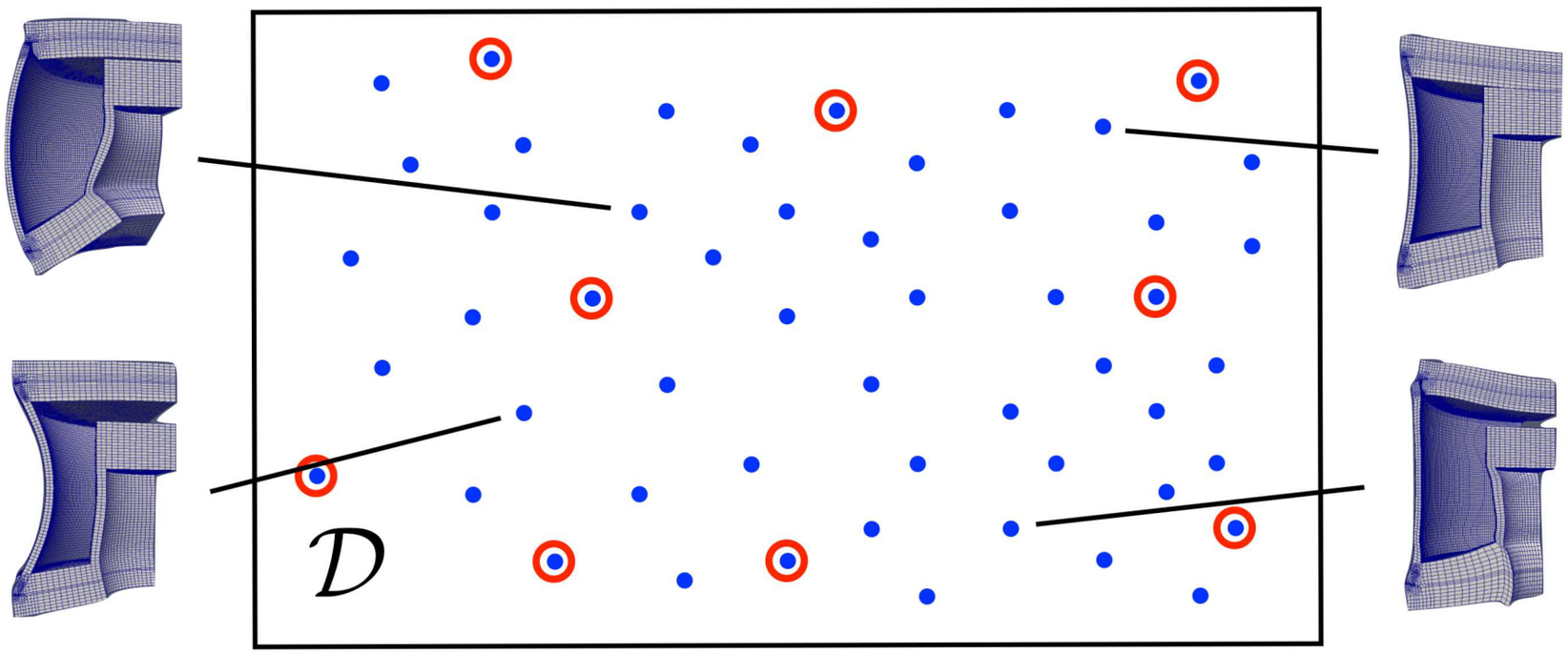
***Many-query problem**: solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}}$*



Approach: exploit simulation data

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

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

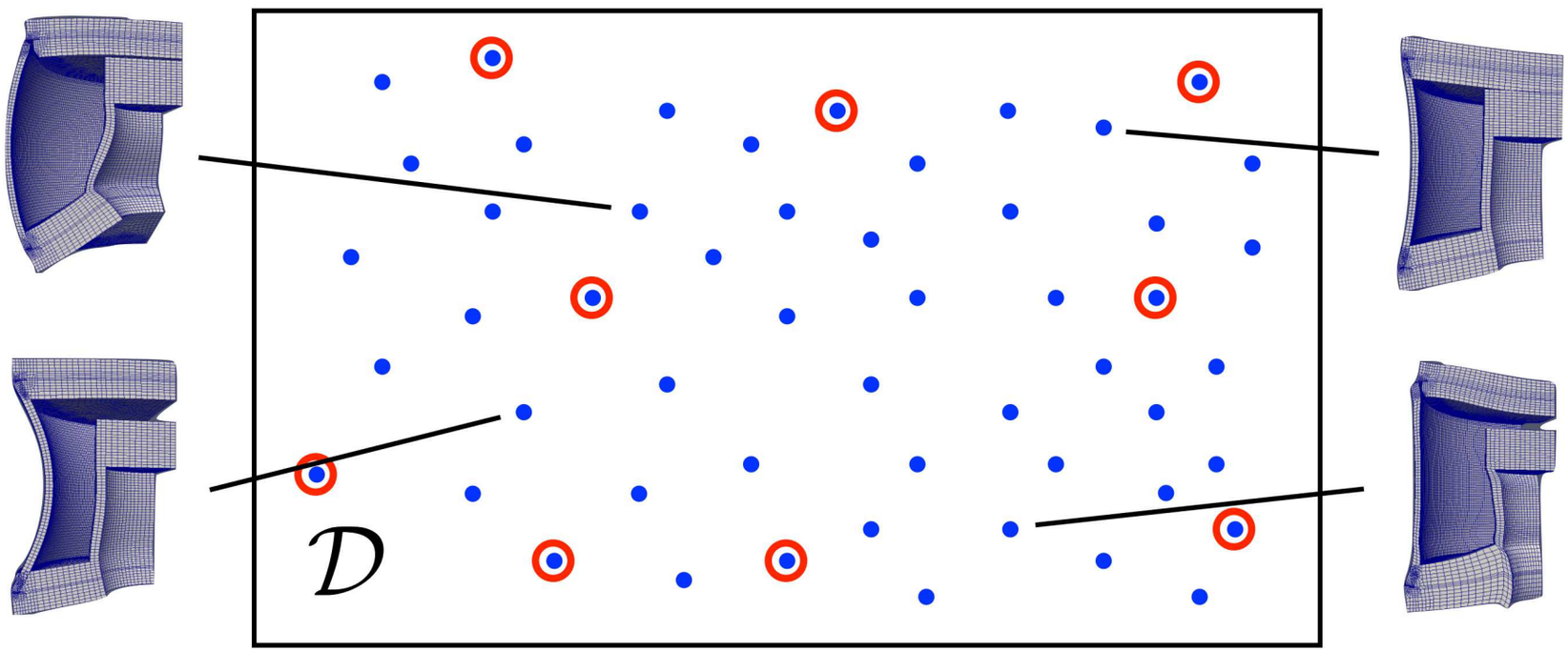


***Idea**: exploit simulation data collected at a few points*

Approach: exploit simulation data

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

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



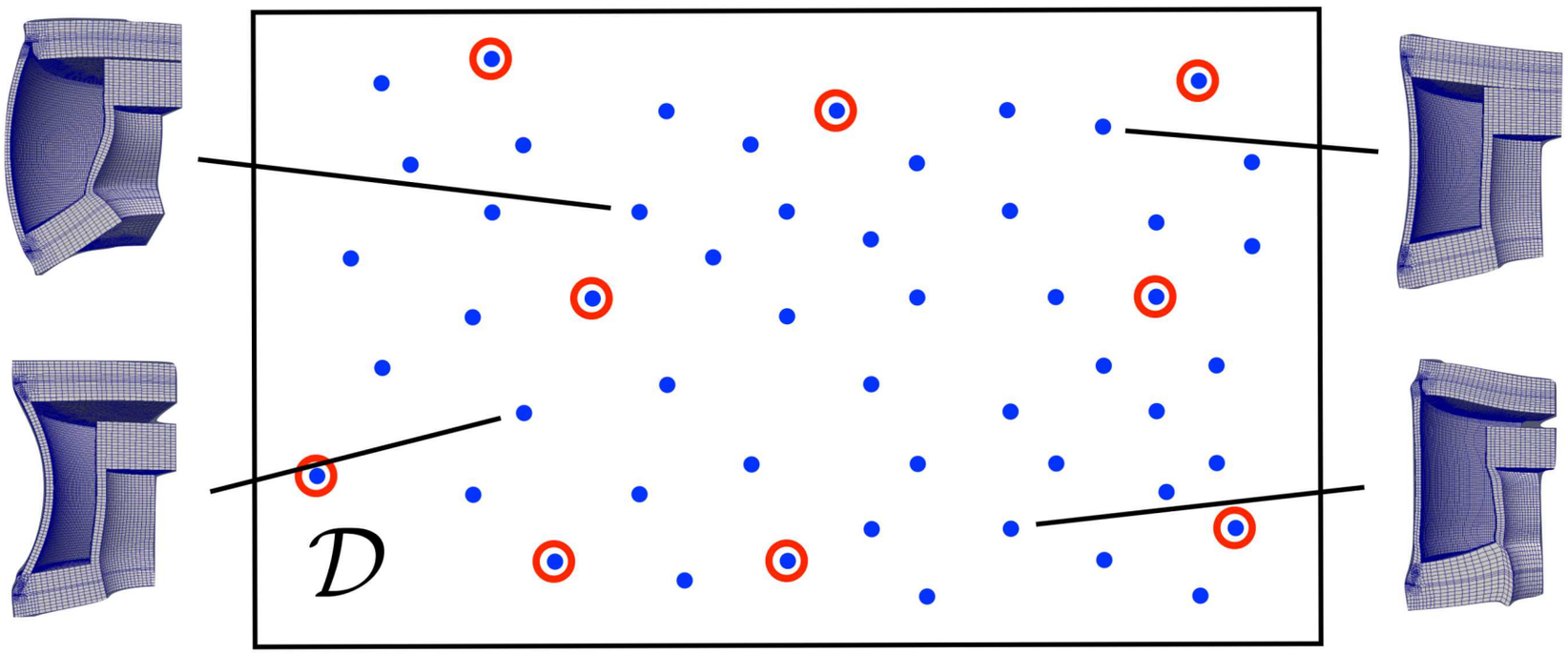
***Idea**: exploit simulation data collected at a few points*

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

Approach: exploit simulation data

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

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



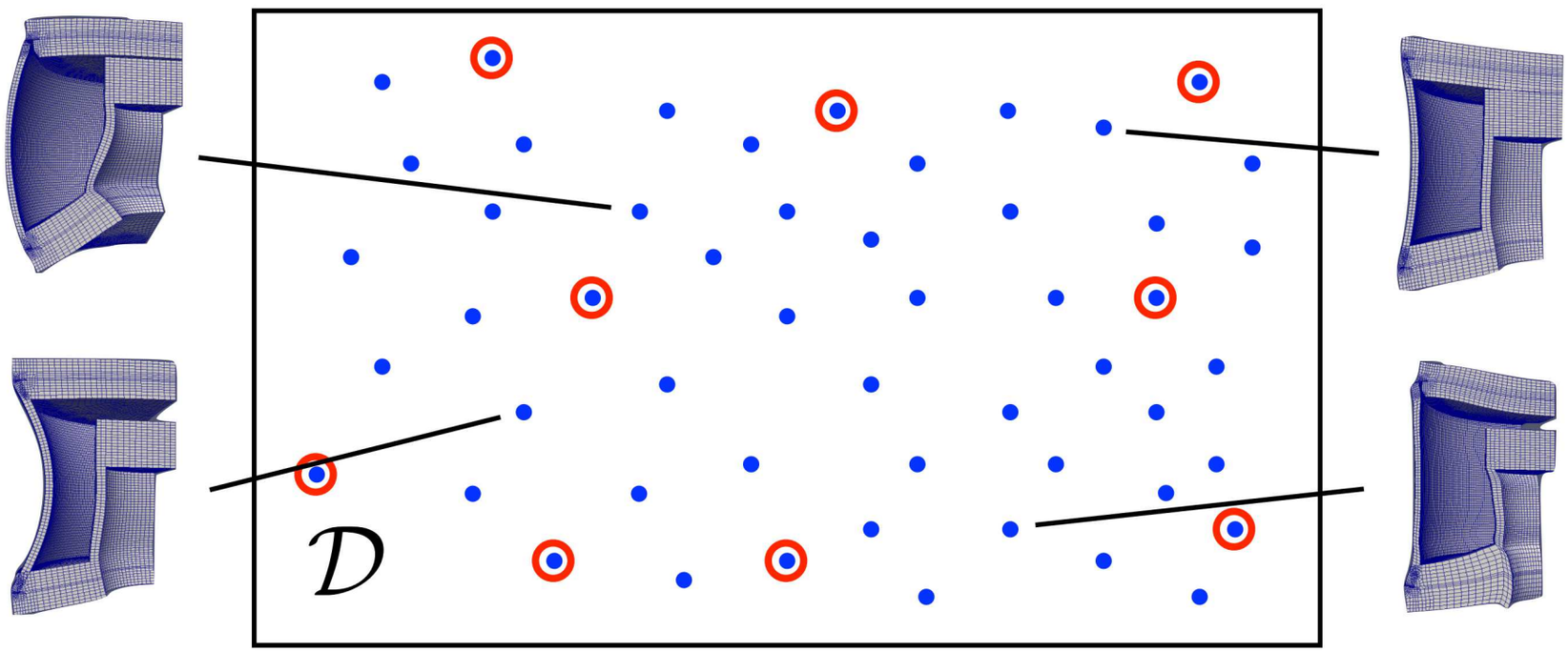
***Idea**: exploit simulation data collected at a few points*

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

Approach: exploit simulation data

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

***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

Model reduction criteria

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

Model reduction criteria

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

2. ***Low cost:*** achieves at least 100x computational savings

Model reduction criteria

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

2. ***Low cost:*** achieves at least 100x computational savings

3. ***Structure preservation:*** preserves important physical properties

Model reduction criteria

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

2. ***Low cost:*** achieves at least 100x computational savings

3. ***Structure preservation:*** preserves important physical properties

4. ***Reliability:*** guaranteed satisfaction of any error tolerance (fail safe)

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 criteria

1. ***Accuracy:*** achieves less than 1% error
   - *LSPG projection* [C., Bou-Mosleh, Farhat, 2011; C., Antil, Barone, 2017]
2. ***Low cost:*** achieves at least 100x computational savings
   - *Sample mesh* [C., Farhat, Cortial, Amsallem, 2013]
3. ***Structure preservation:*** preserves important physical properties
   - *Conservative model reduction* [C., Choi, Sargsyan, 2018]
4. ***Reliability:*** guaranteed satisfaction of any error tolerance (fail safe)
   - *Adaptive h-refinement* [C., 2015]
5. ***Certification:*** quantifies ROM-induced epistemic uncertainty
   - *Machine-learning error models*
     [Drohmann, C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2017]

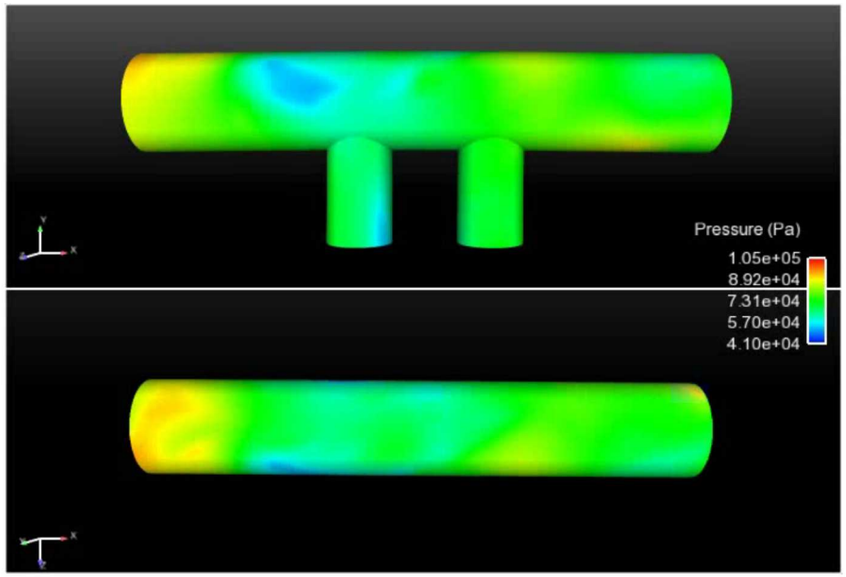
Model reduction criteria

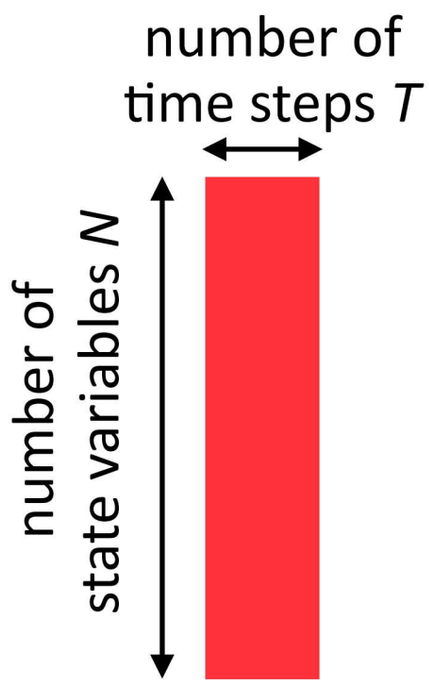
1. ***Accuracy:*** achieves less than 1% error
   - *LSPG projection* [C., Bou-Mosleh, Farhat, 2011; C., Antil, Barone, 2017]
2. ***Low cost:*** achieves at least 100x computational savings
   - *Sample mesh* [C., Farhat, Cortial, Amsallem, 2013]
3. ***Structure preservation:*** preserves important physical properties
   - *Conservative model reduction* [C., Choi, Sargsyan, 2018]
4. ***Reliability:*** guaranteed satisfaction of any error tolerance (fail safe)
   - *Adaptive h-refinement* [C., 2015]
5. ***Certification:*** quantifies ROM-induced epistemic uncertainty
   - *Machine-learning error models*
     [Drohmann, C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2017]

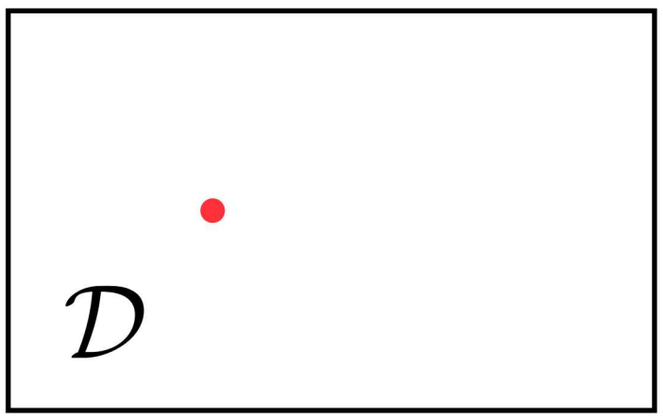
Training simulations: state tensor

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

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



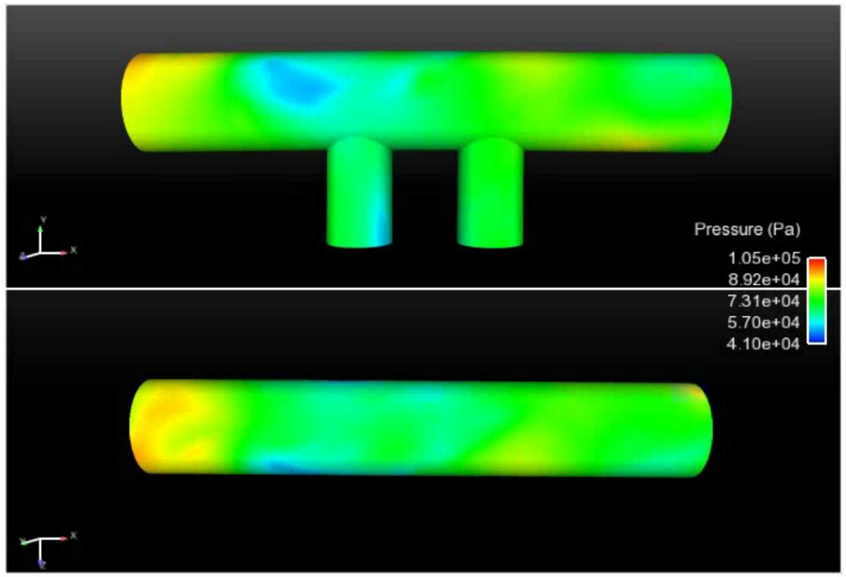




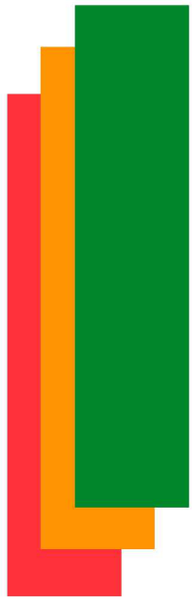
Training simulations: state tensor

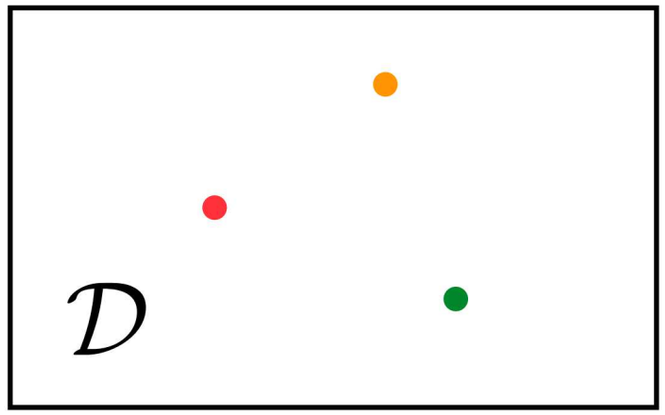
$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

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



$$\mathcal{X} =$$





Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

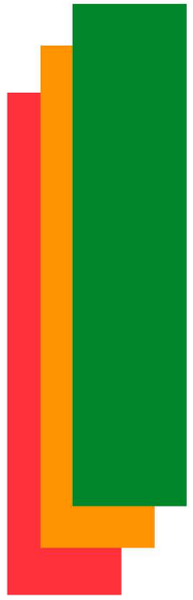
$\mathcal{X} =$

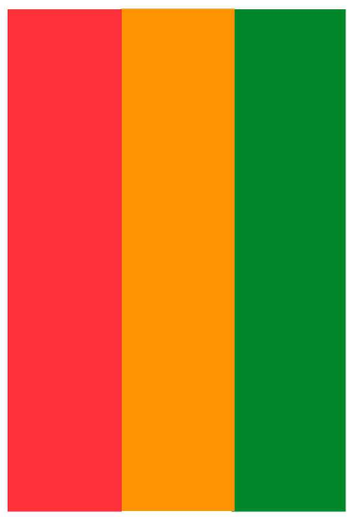
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$ 

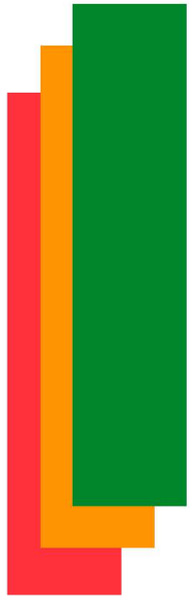
$\mathbf{X}_{(1)} =$ 

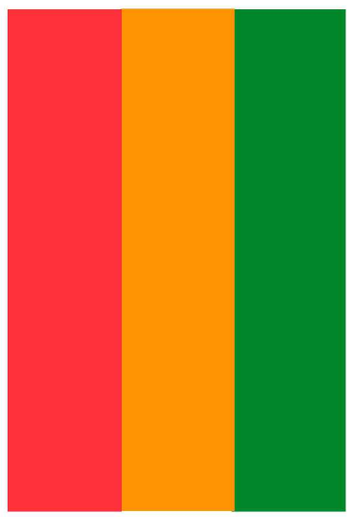
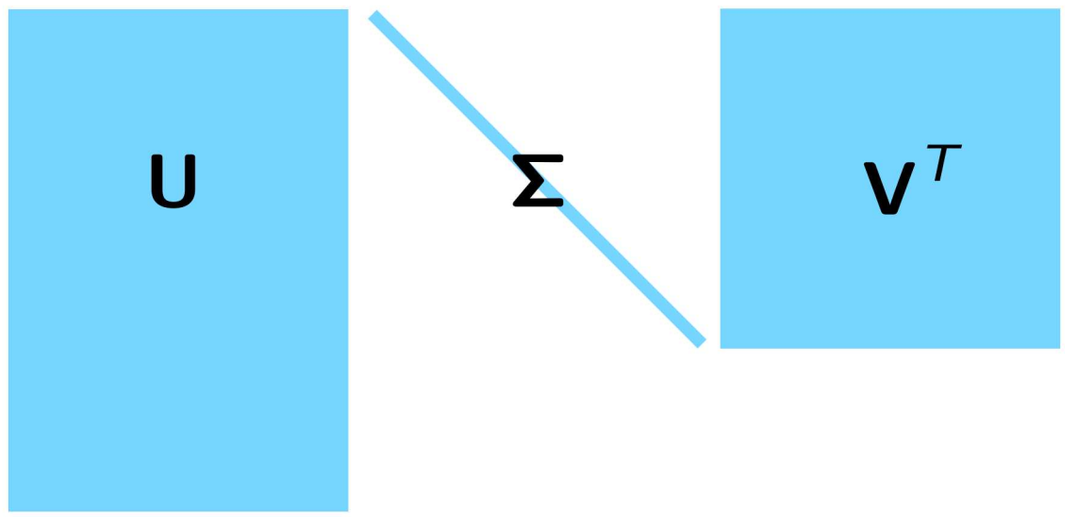
Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

$\mathcal{X} =$ 

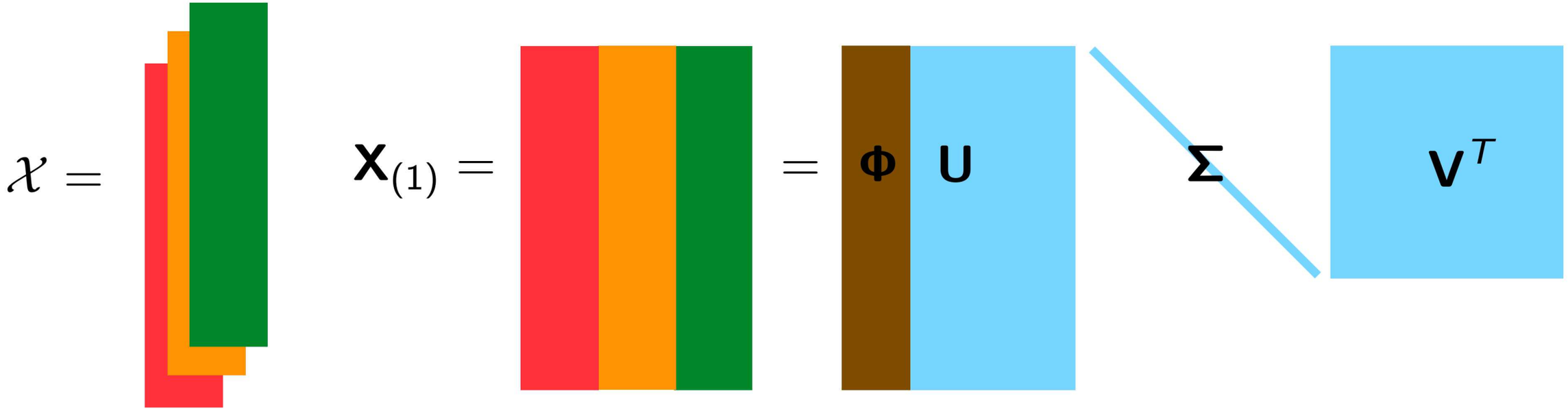
$\mathbf{X}_{(1)} =$  $=$ 

Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

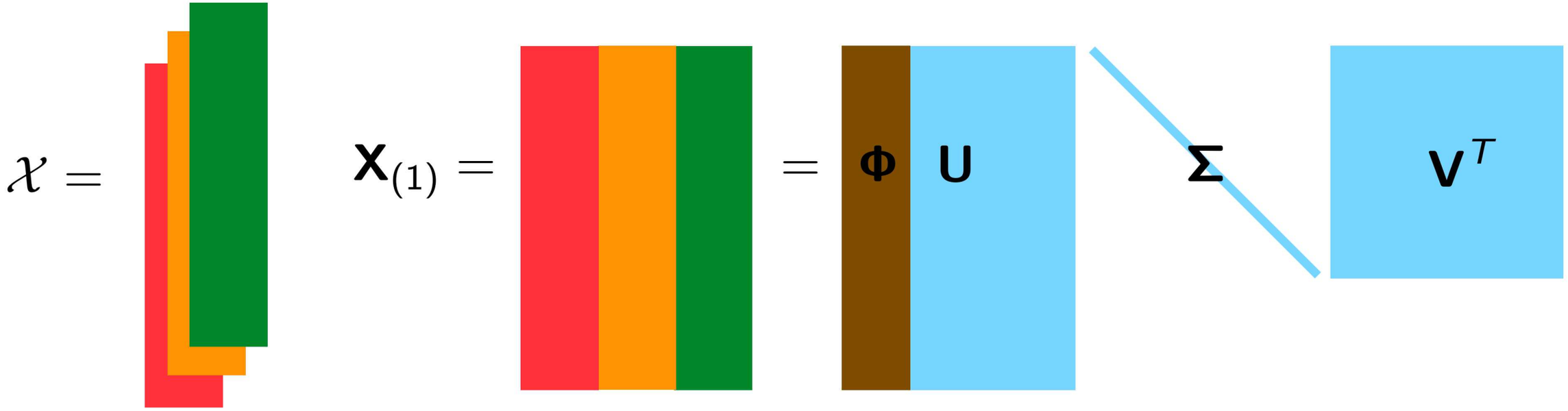


Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding



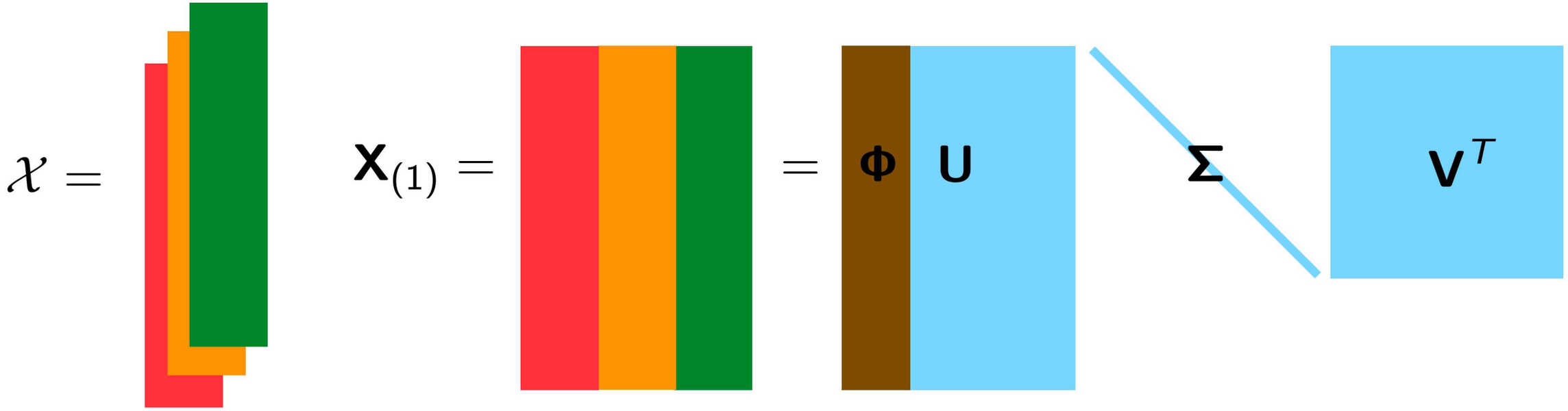
Φ *columns are principal components of the spatial simulation data*

Tensor decomposition

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

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

Compute dominant left singular vectors of mode-1 unfolding

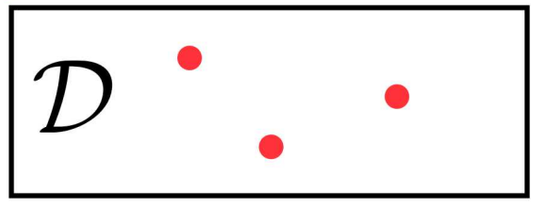


Φ *columns are principal components of the spatial simulation data*

How to integrate these data with the computational model?

Least-squares Petrov–Galerkin (LSPG) projection

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

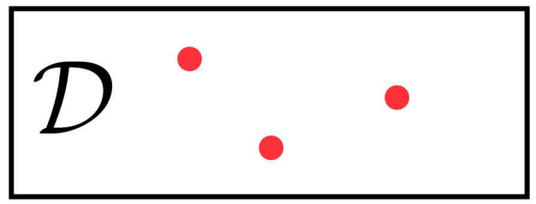


1. *Training:* Solve ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$ and collect simulation data
2. *Machine learning:* Identify structure in data
3. *Reduction:* Reduce the cost of solving ODE for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

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

Least-squares Petrov–Galerkin (LSPG) projection

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



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

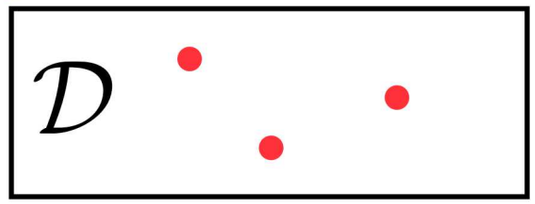
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

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

Least-squares Petrov–Galerkin (LSPG) projection

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



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

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

1. Reduce the number of unknowns

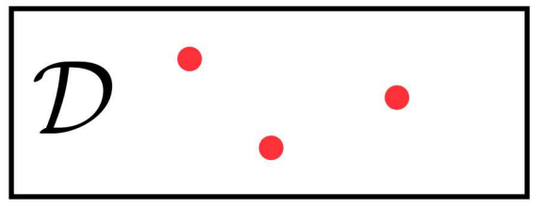
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$$

2. Minimize the OΔE residual

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

Least-squares Petrov–Galerkin (LSPG) projection

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



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

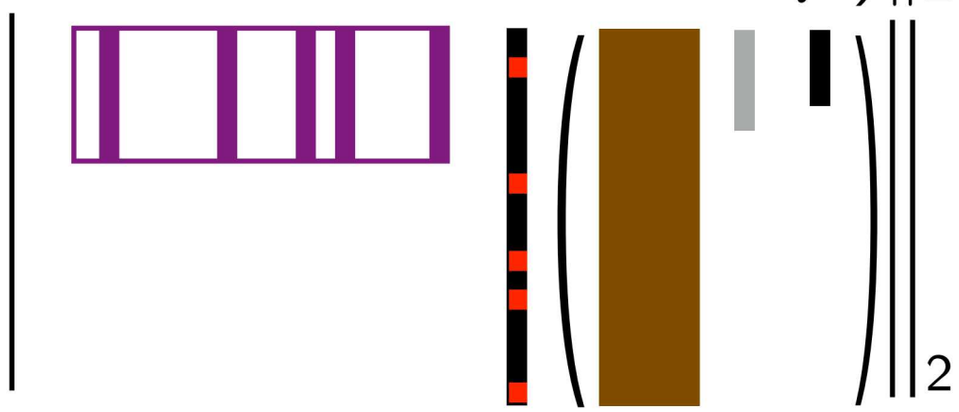
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \ldots, T$

1. Reduce the number of unknowns

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

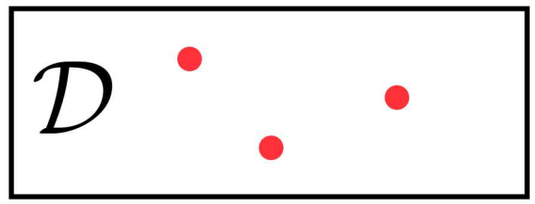
2. Minimize the OΔE residual

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



Least-squares Petrov–Galerkin (LSPG) projection

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



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

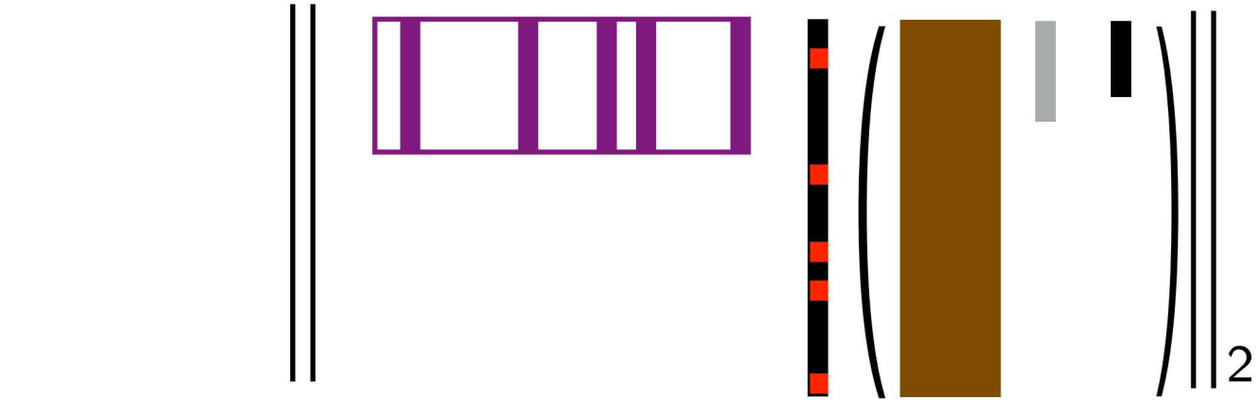
Apply time discretization to obtain OΔE: $\mathbf{r}^n(\mathbf{x}^n; \boldsymbol{\mu}) = \mathbf{0}, \quad n = 1, \dots, T$

1. Reduce the number of unknowns

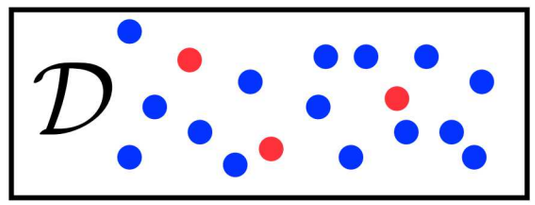
$$\mathbf{x}(t) \approx \tilde{\mathbf{x}}(t) = \boldsymbol{\Phi}\, \hat{\mathbf{x}}(t)$$

2. Minimize the OΔE residual

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

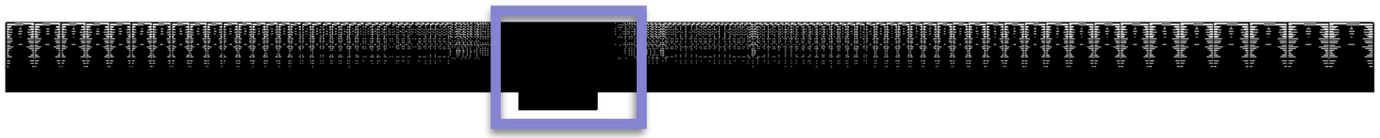


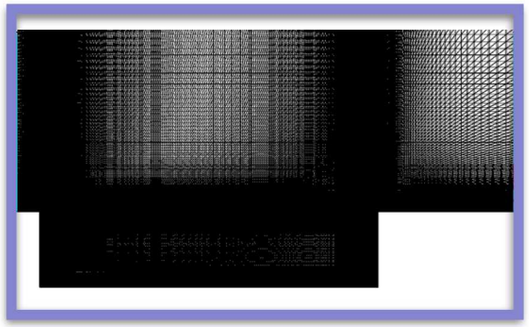
LSPG OΔE: $\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A}\mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}; \boldsymbol{\mu}) \right\|_2^2$



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

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

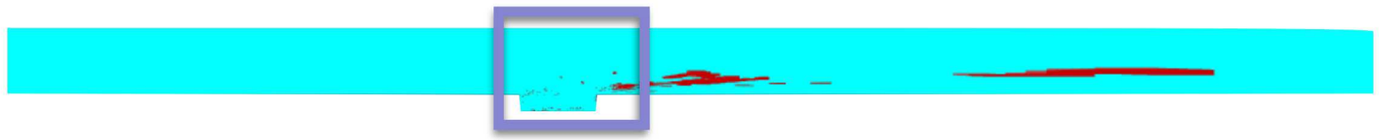


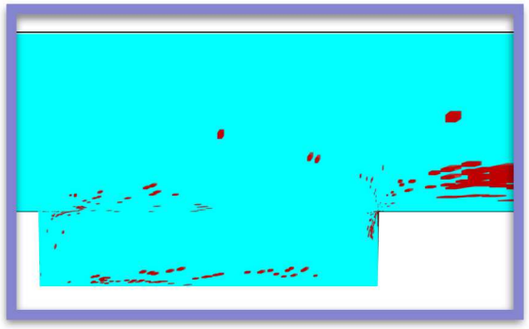


Captive carry results [C., Barone, Antil, 2017]

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

sample mesh



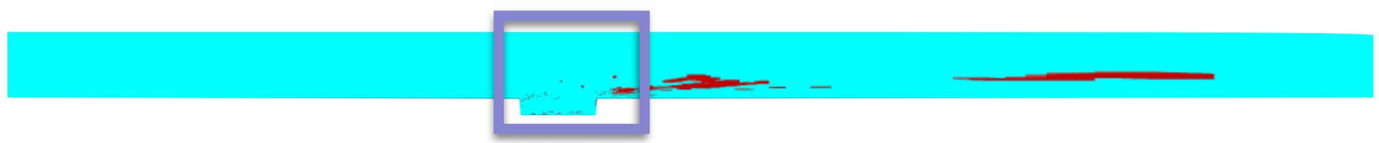


+ HPC on a laptop

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

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

sample mesh



+ *HPC on a laptop*

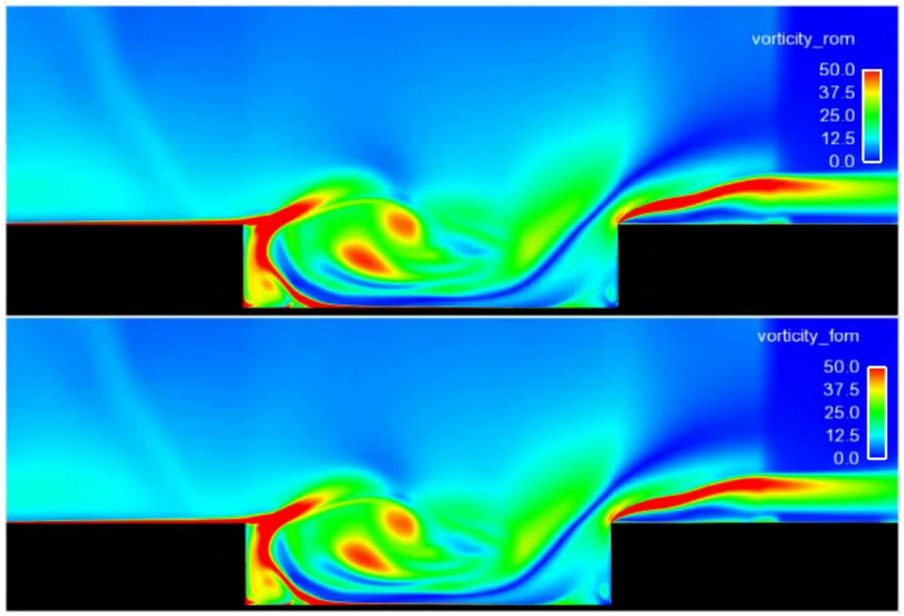
vorticity field *pressure field*

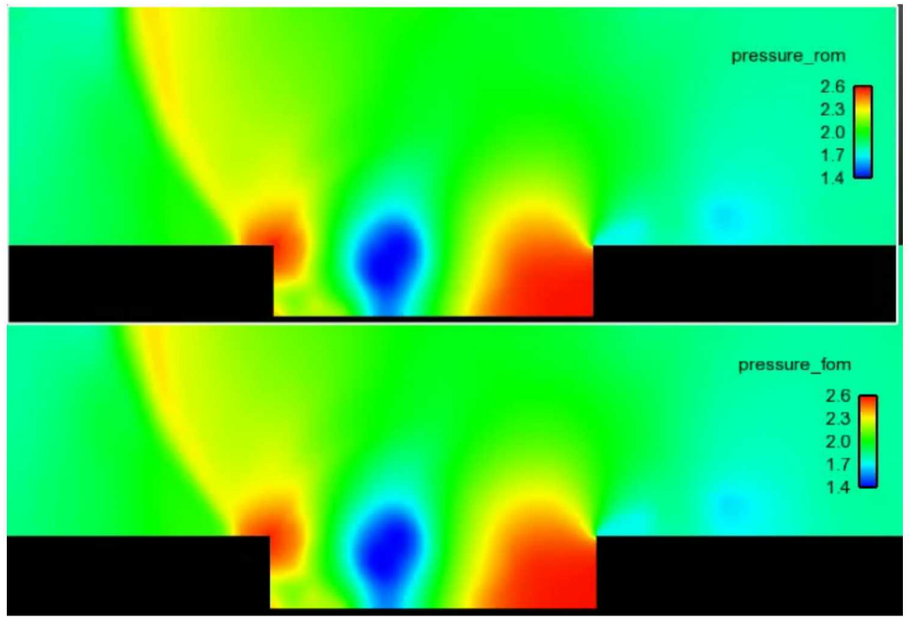
LSPG ROM

32 min, 2 cores

high-fidelity

5 hours, 48 cores





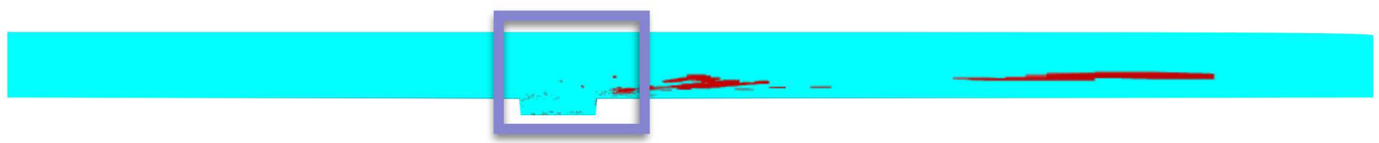
+ *229x savings in core–hours*

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

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

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

sample mesh



+ *HPC on a laptop*

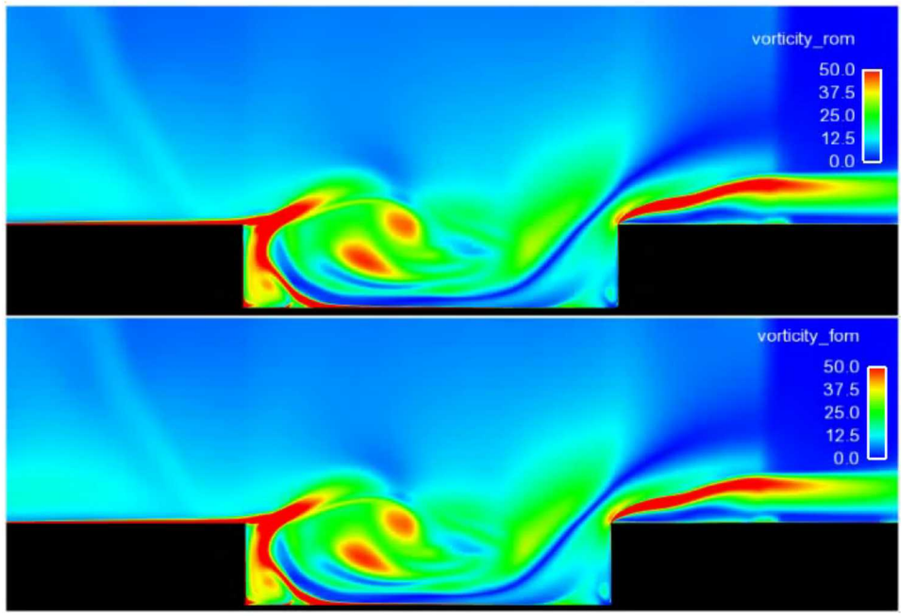
vorticity field *pressure field*

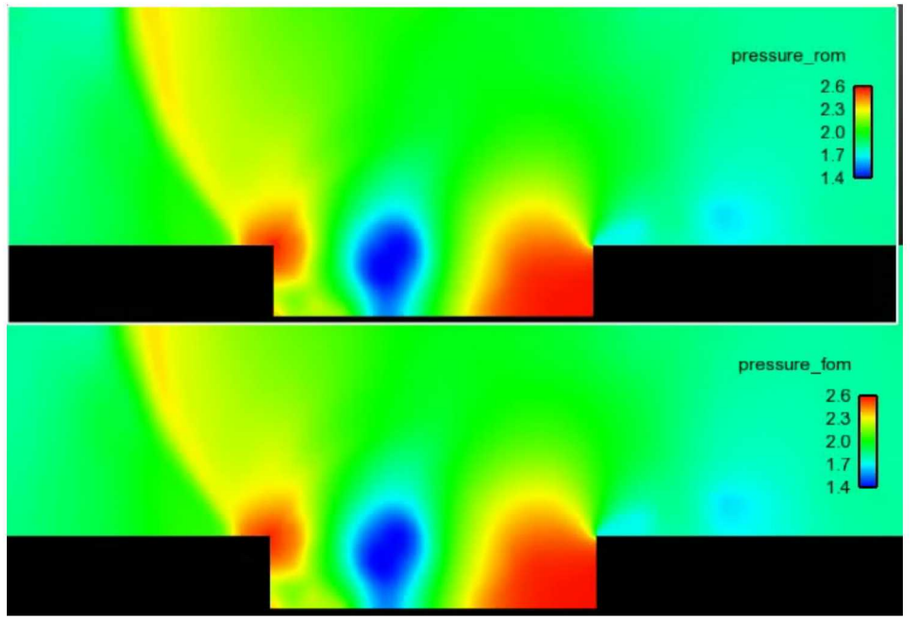
LSPG ROM

32 min, 2 cores

high-fidelity

5 hours, 48 cores





+ *229x savings in core–hours*

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

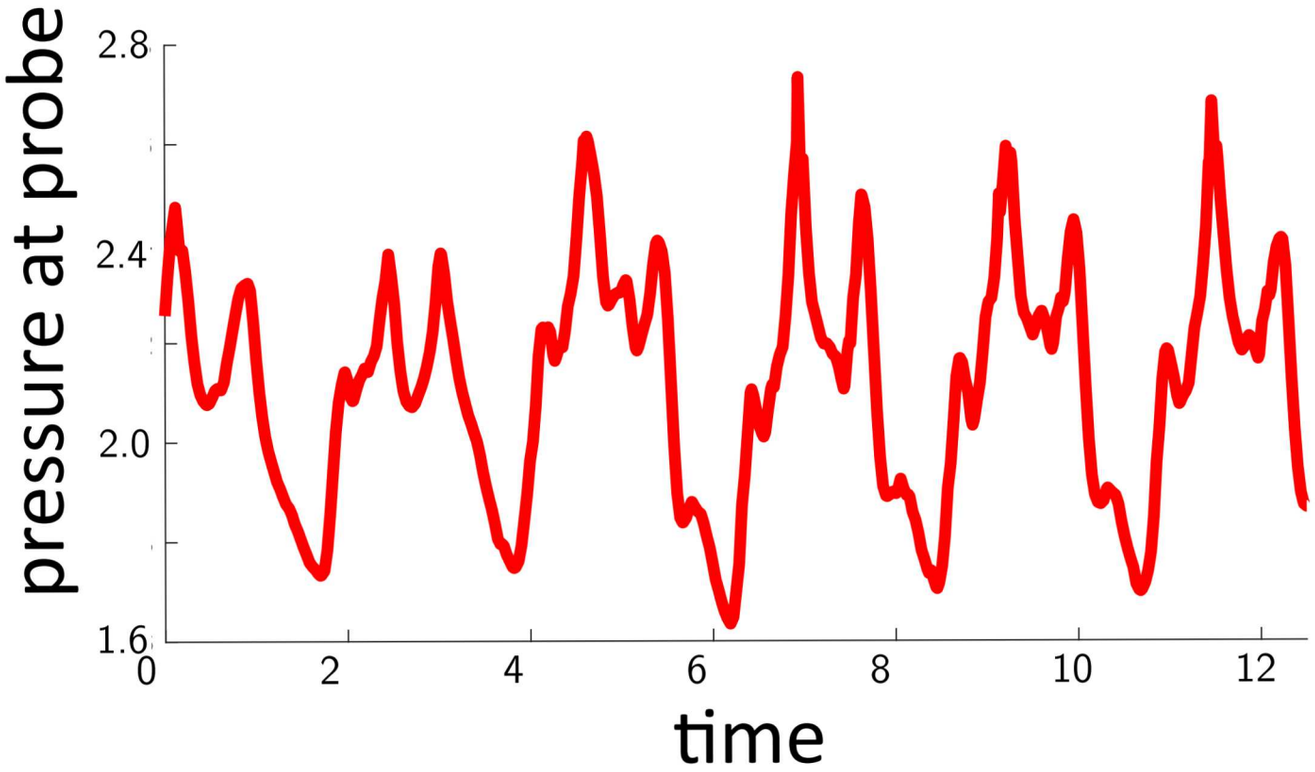
Implementation: Albany, SPARC, non-intrusive wrapper code

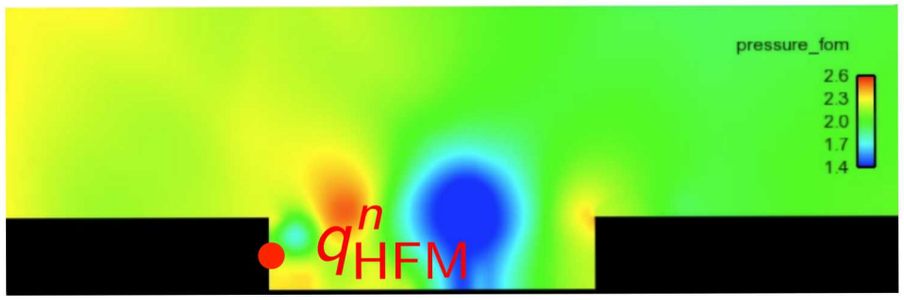
Model reduction criteria

1. ***Accuracy:*** achieves less than 1% error
   - *LSPG projection* [C., Bou-Mosleh, Farhat, 2011; C., Antil, Barone, 2017]
2. ***Low cost:*** achieves at least 100x computational savings
   - *Sample mesh* [C., Farhat, Cortial, Amsallem, 2013]
3. ***Structure preservation:*** preserves important physical properties
   - *Conservative model reduction* [C., Choi, Sargsyan, 2018]
4. ***Reliability:*** guaranteed satisfaction of any error tolerance (fail safe)
   - *Adaptive h-refinement* [C., 2015]
5. ***Certification:*** quantifies ROM-induced epistemic uncertainty
   - *Machine-learning error models*
     [Drohmann, C., 2015; Trehan, C., Durlofsky, 2017; Freno and C., 2017]

Key insight

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

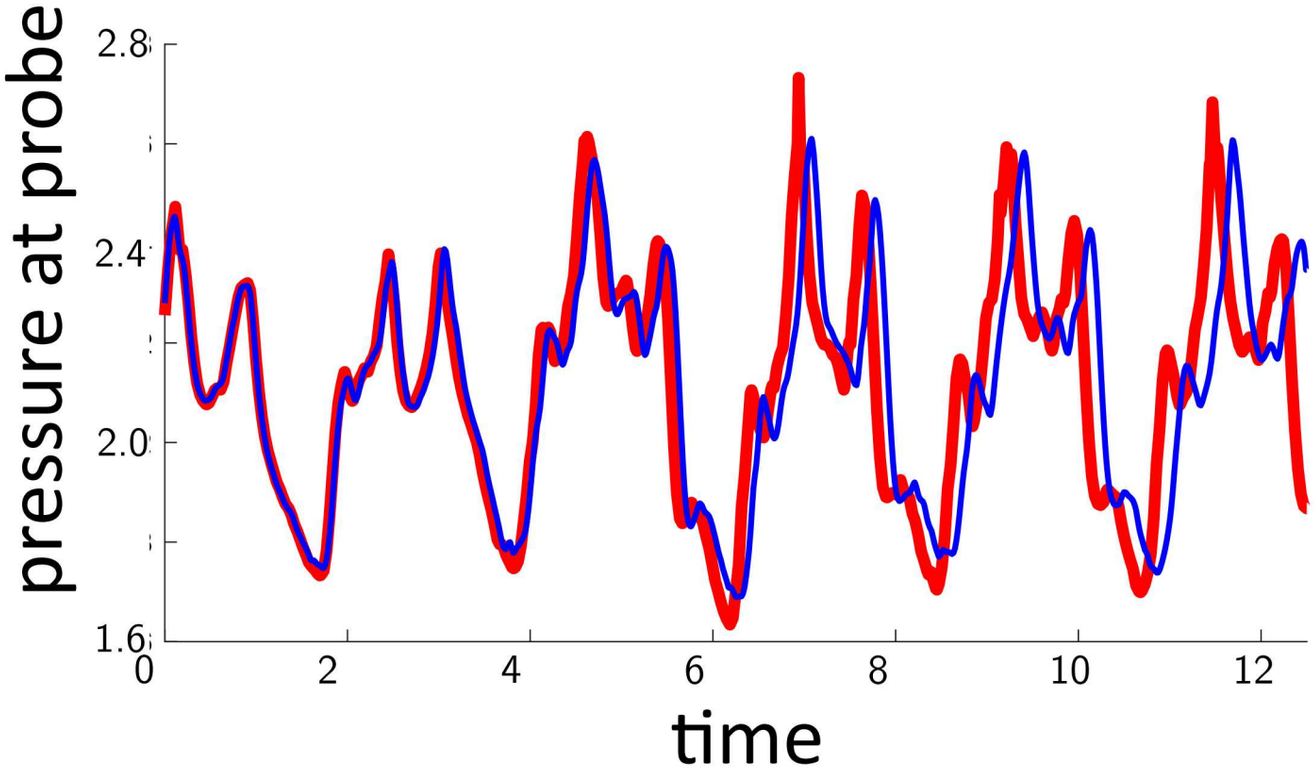


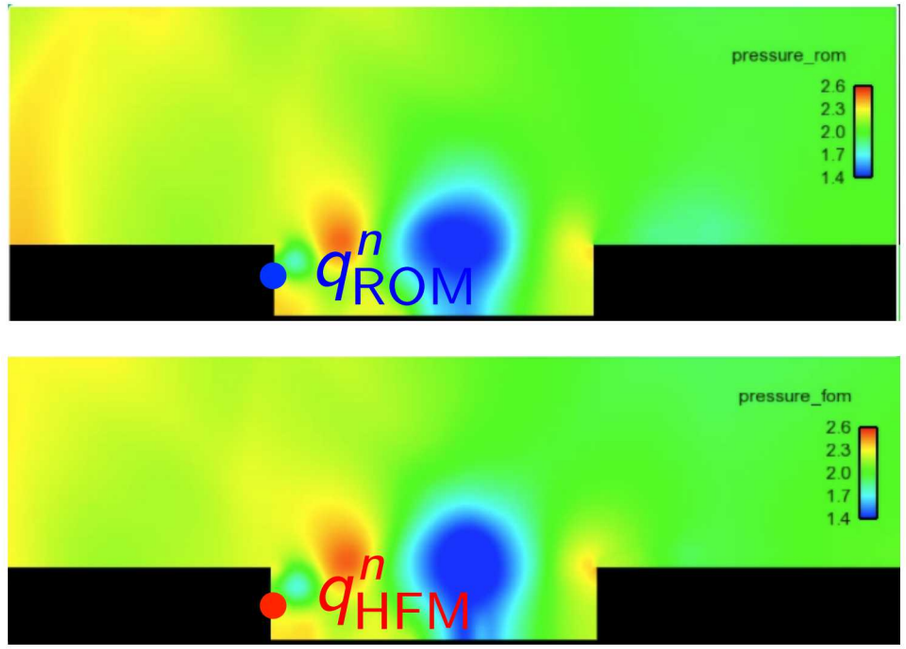


Key insight

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

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





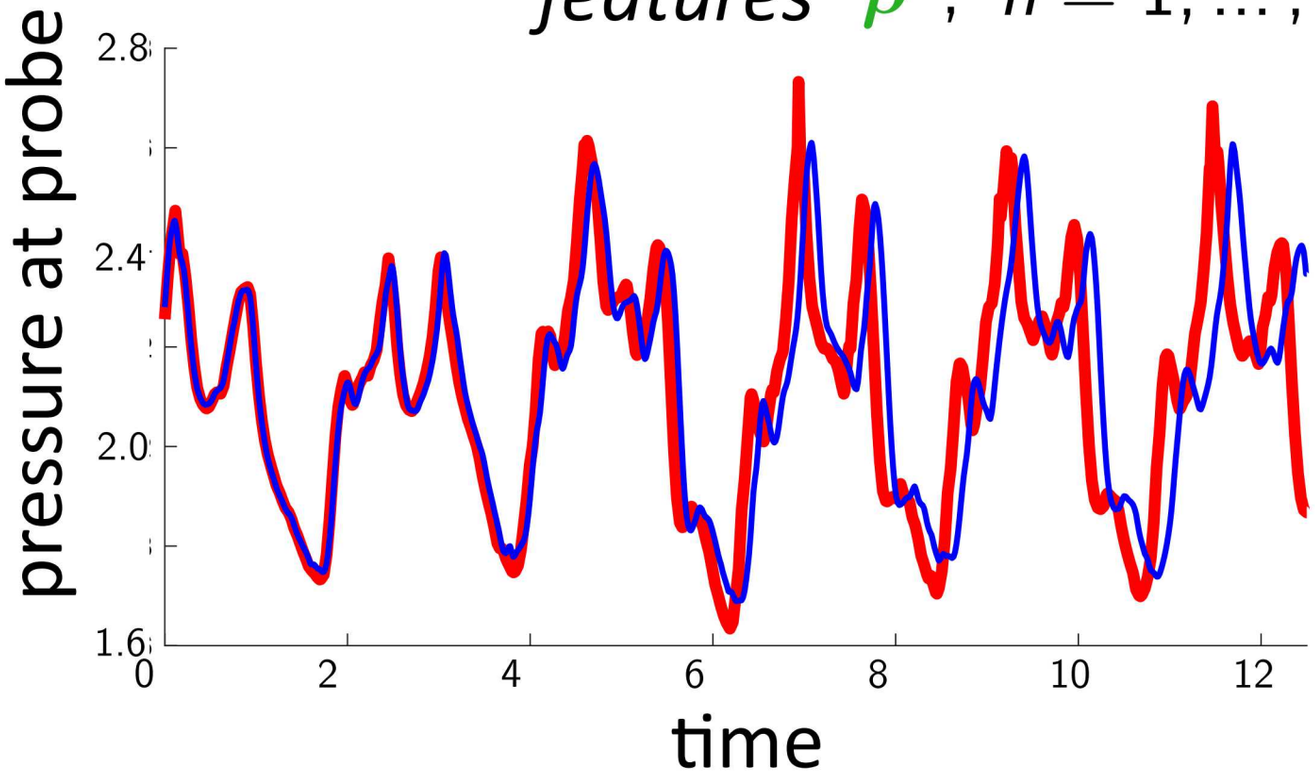
Key insight

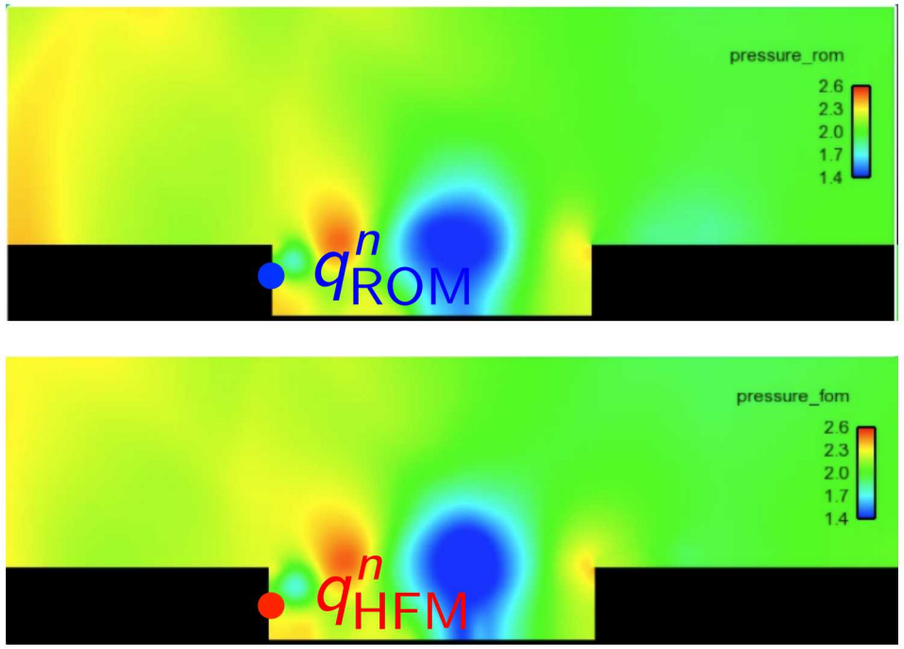
inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}},\ n = 1, \ldots, T$

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

↓

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





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

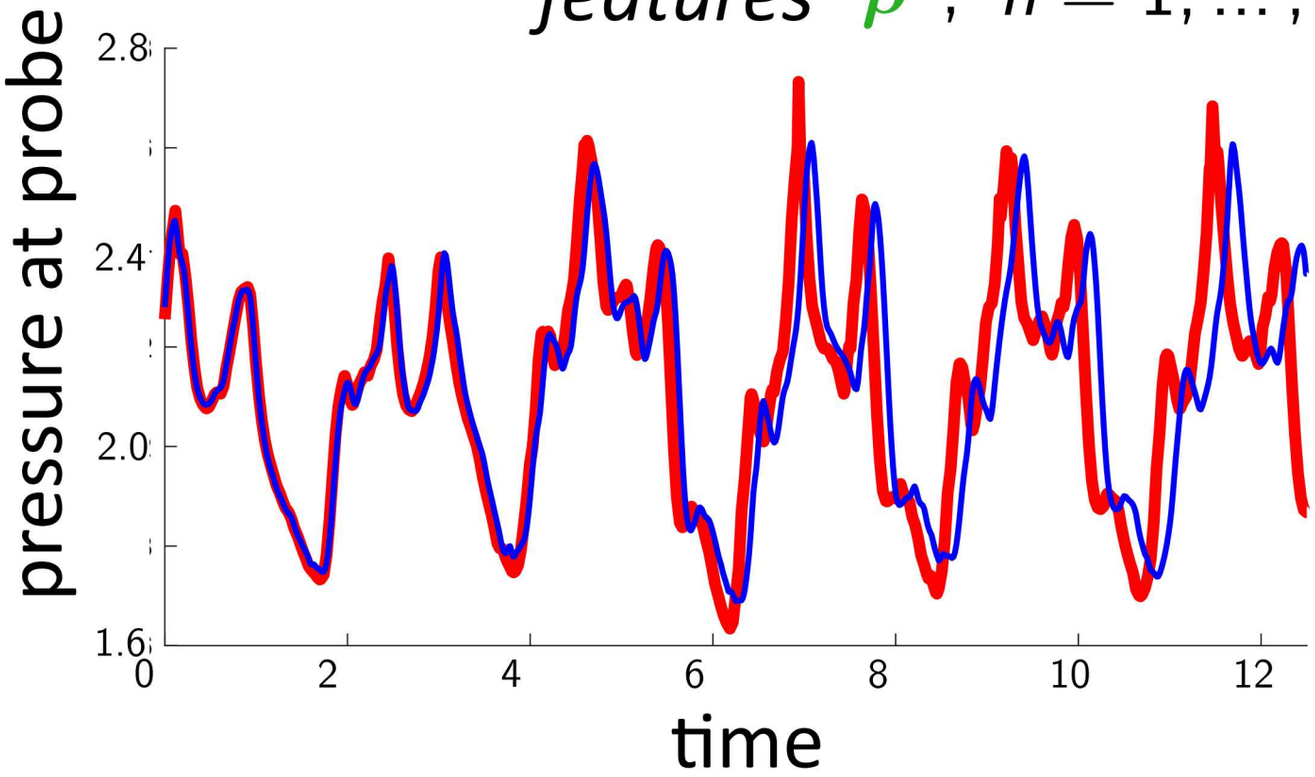
Key insight

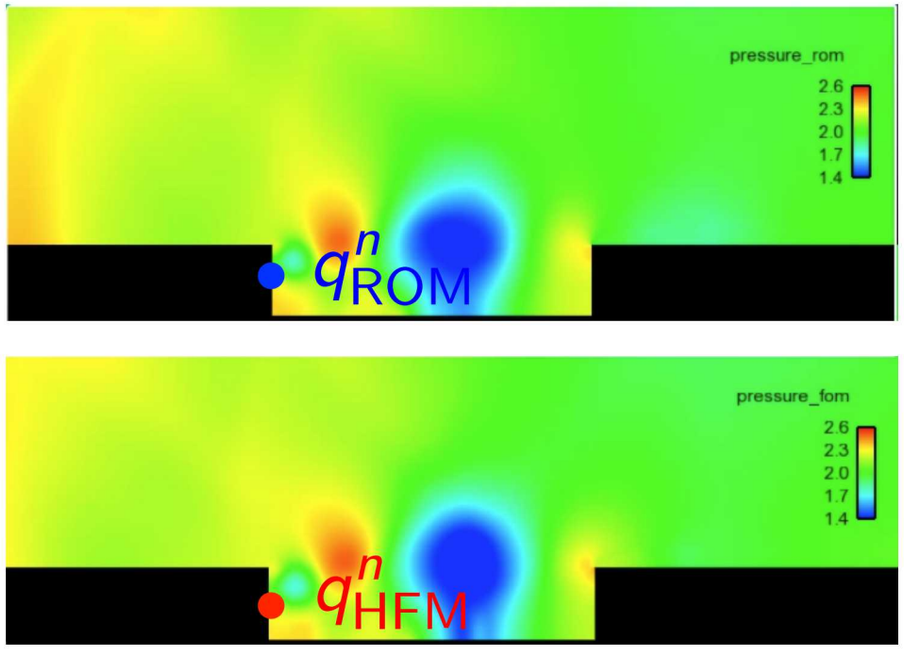
inputs μ → *high-fidelity model* → *outputs* $q^n_{\text{HFM}},\ n = 1, \ldots, T$

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

↓

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



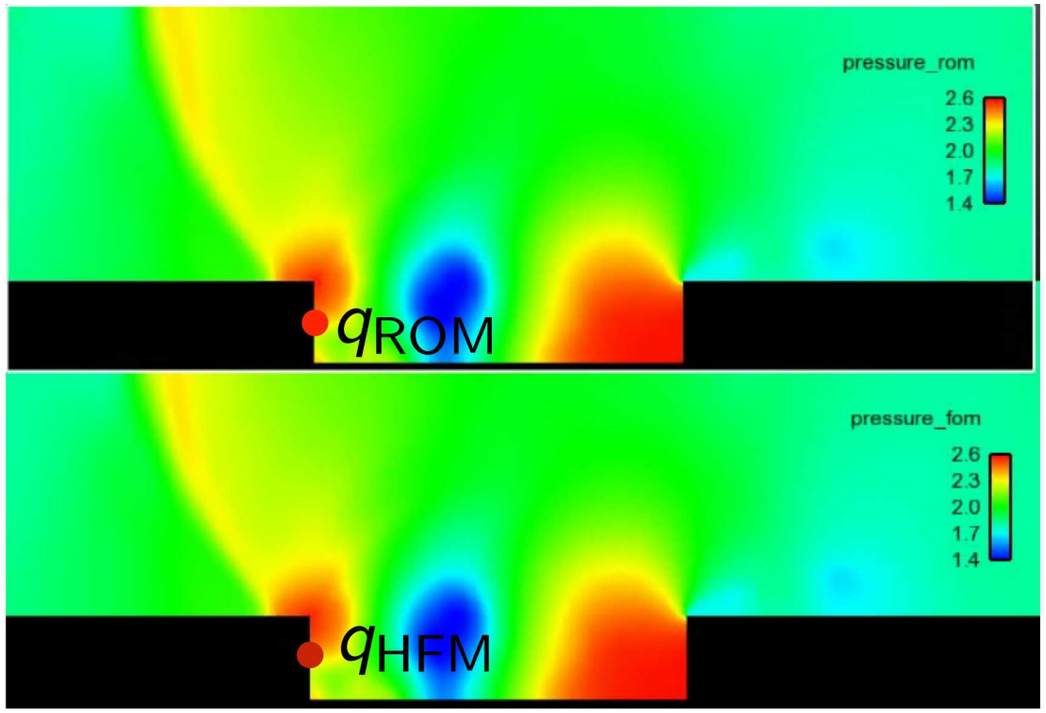


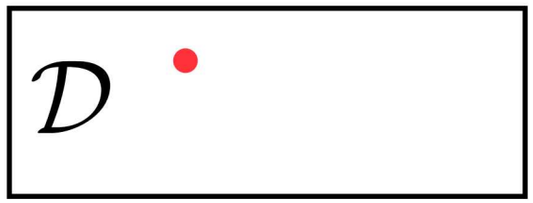
Reduced-order models generate features ρ^n ***that may inform its error***

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

Training and machine learning: error modeling

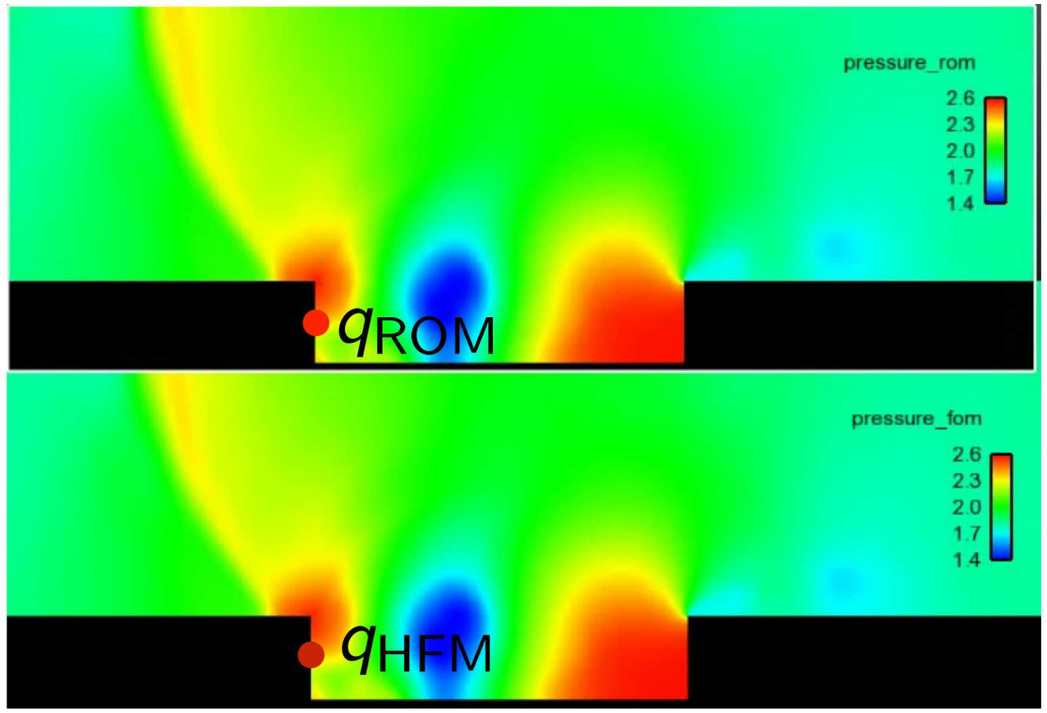
1. *Training:* Solve high-fidelity and reduced-order models for $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict reduced-order-model error for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$

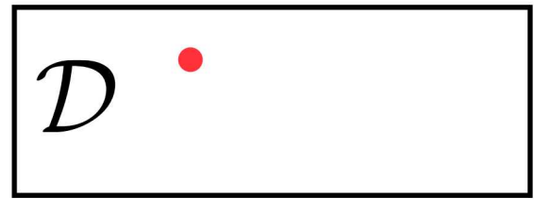




Training and machine learning: error modeling

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



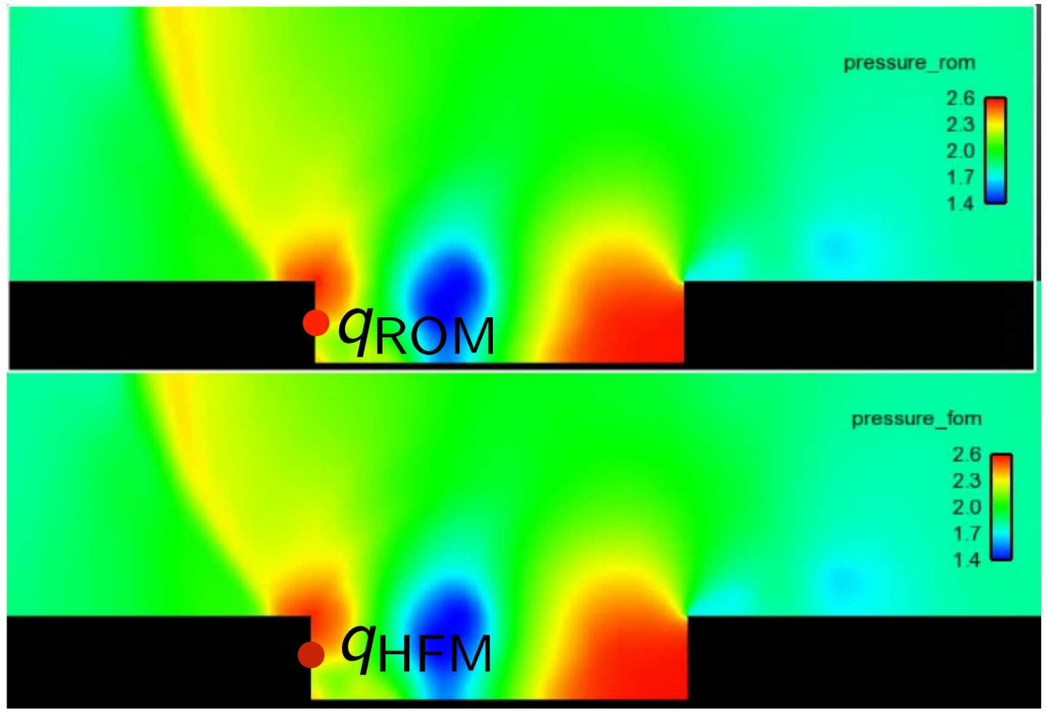


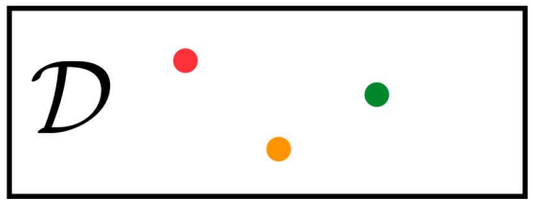
$q^n_{\text{HFM}} - q^n_{\text{ROM}}$

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



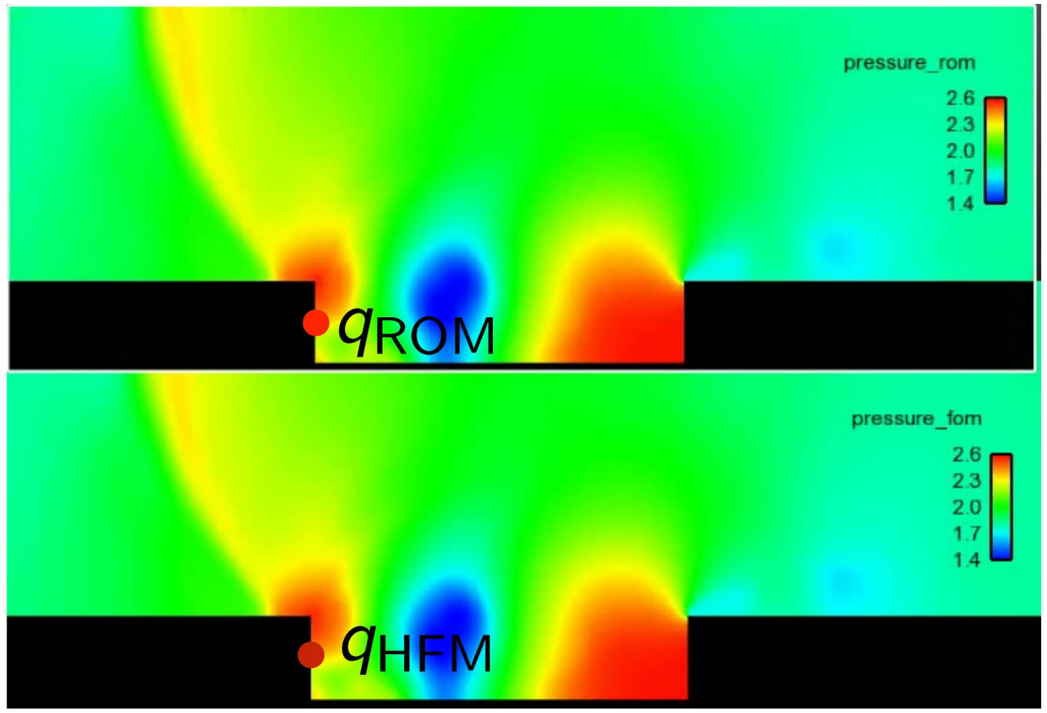


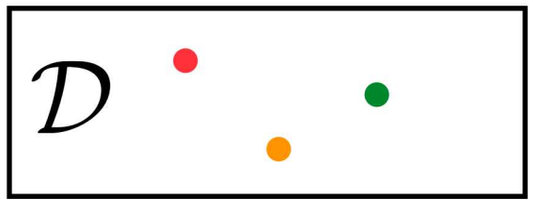
$q^n_{\text{HFM}} - q^n_{\text{ROM}}$

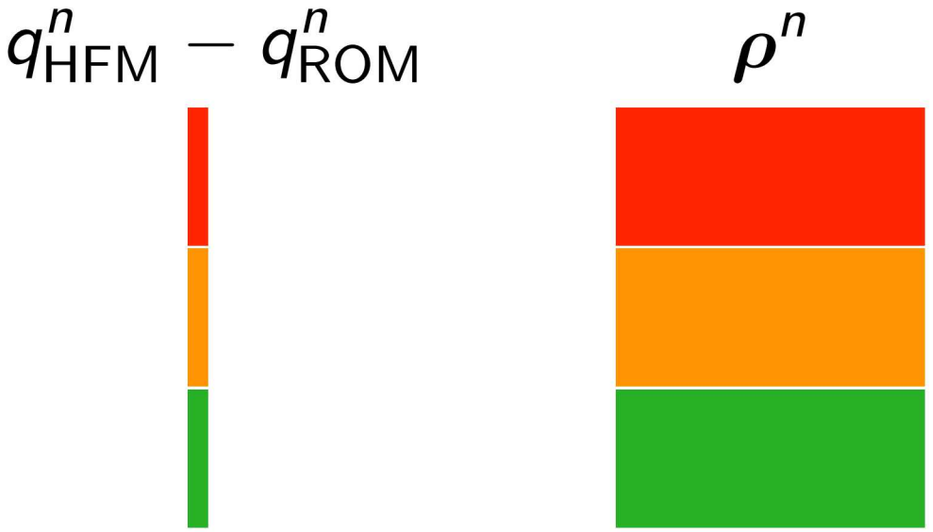
ρ^n

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





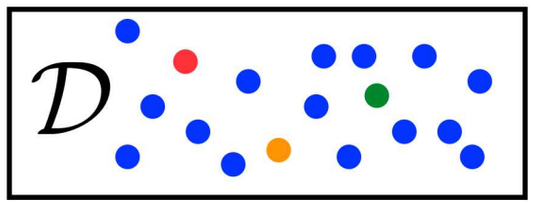


features ρ^n → *regression model* → *error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$

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

Regression model for the error

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



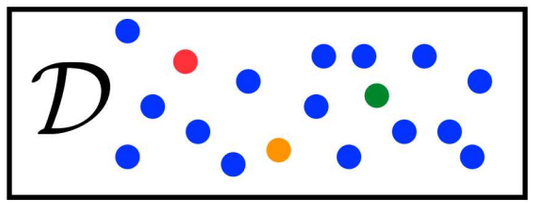
inputs $\boldsymbol{\mu}$ → *reduced-order model* → *outputs* q^n_{ROM}, $n = 1, \dots, T$

↓

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

Regression model for the error

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



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

↓

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

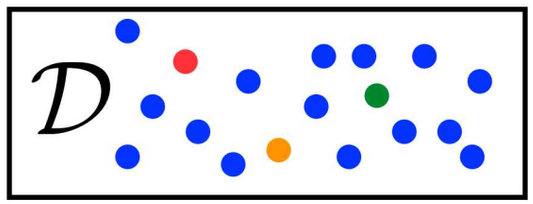
↓

regression model → *stochastic error prediction*

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

Regression model for the error

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



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

↓

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

↓

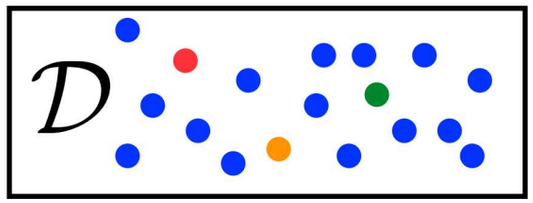
regression model → *stochastic error prediction* $\hat{\delta}^n \approx q^n_{\text{HFM}} - q^n_{\text{ROM}}$, $n = 1, \ldots, T$

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

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

Regression model for the error

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



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

↓

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

↓

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

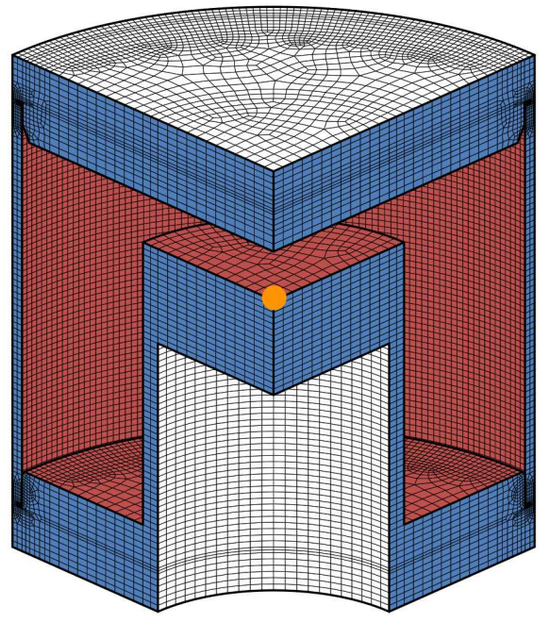
$$q^n_{\text{HFM}} \approx q^n_{\text{ROM}} + \hat{\delta}^n$$

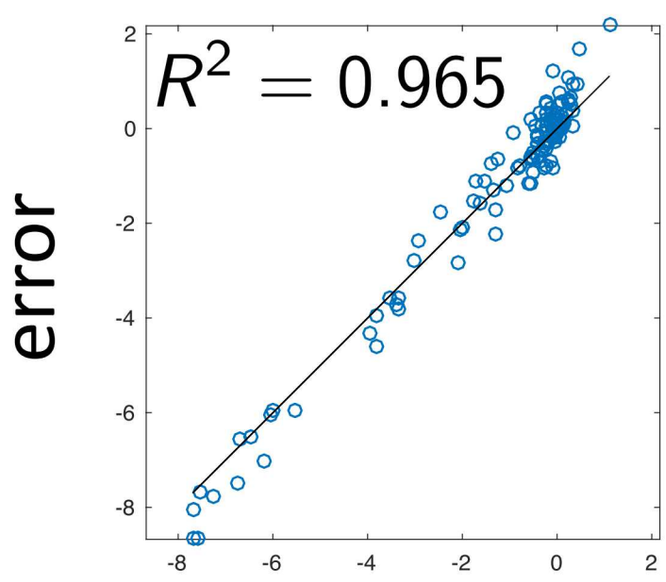
\+ *Statistical model of high-fidelity-model output*

Physics-based feature engineering to determine $\boldsymbol{\rho}^n$

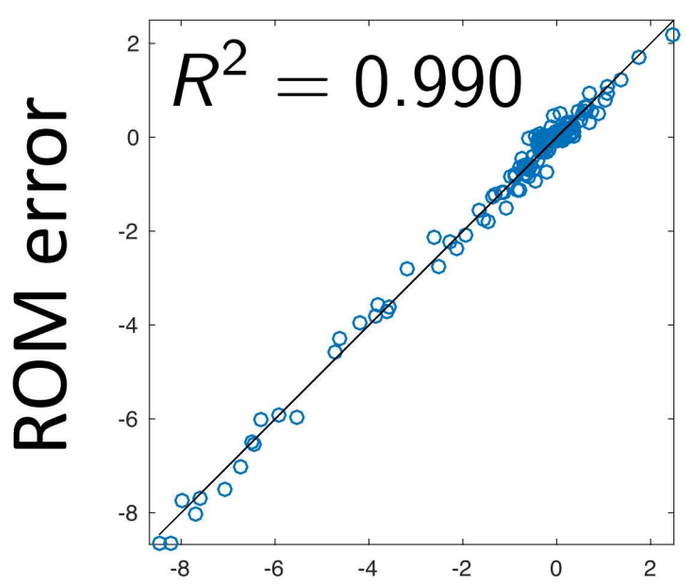
Predictive Capability Assessment Project results [Freno, C., 2018]

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

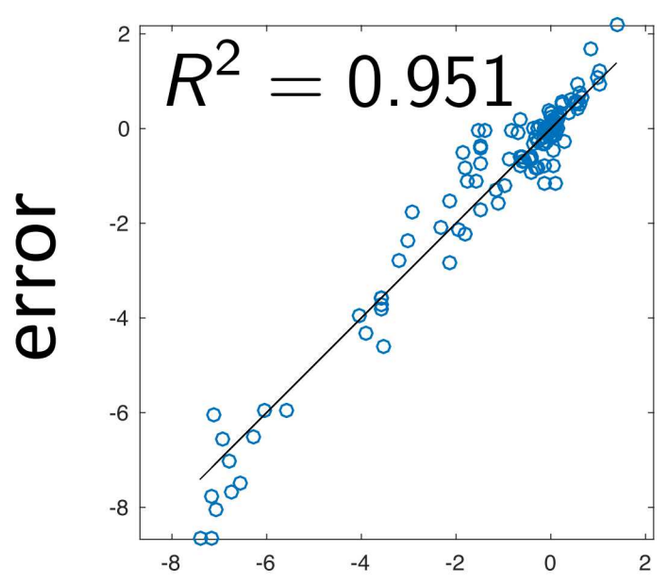




random forest
error prediction



support vector machine
error prediction

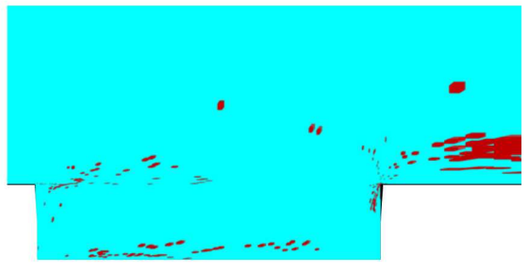


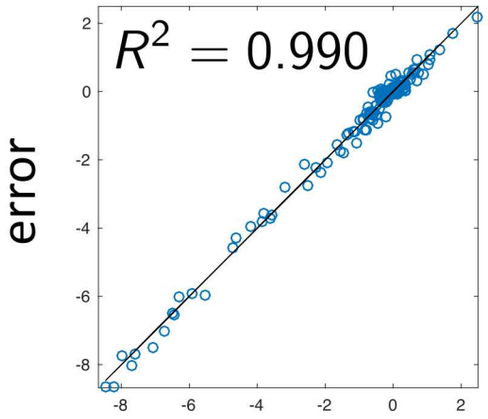
k-NN
error prediction

- *ML methods yield low-variance error predictions*

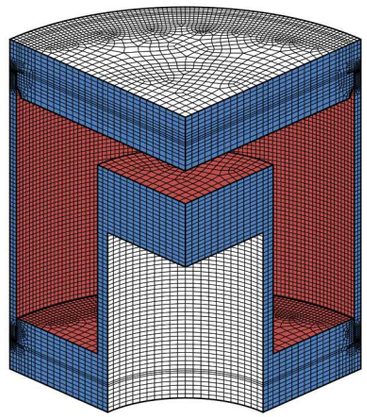
Questions?

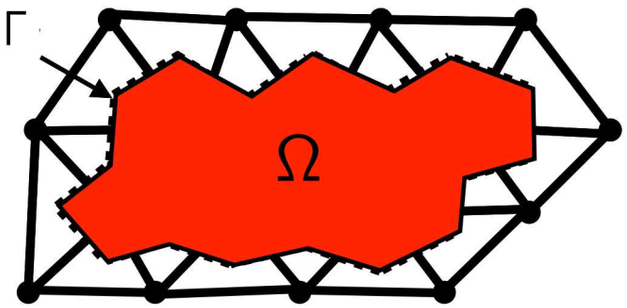
$\mathcal{X}^{ijk} =$

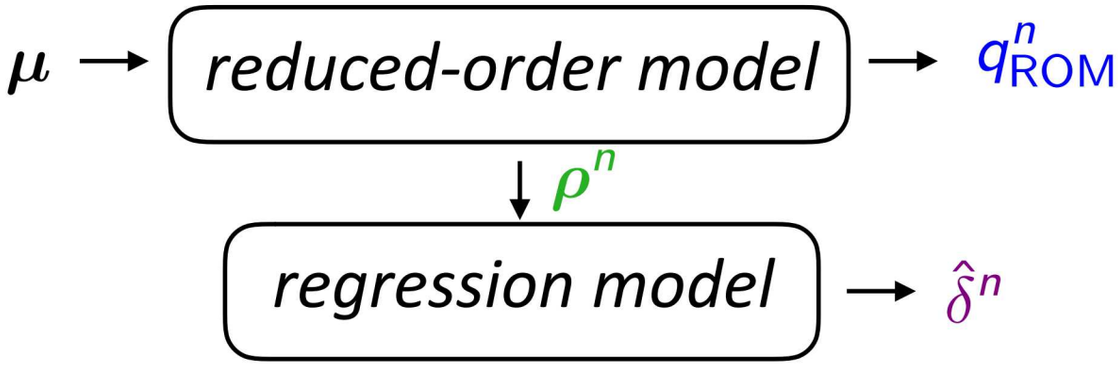


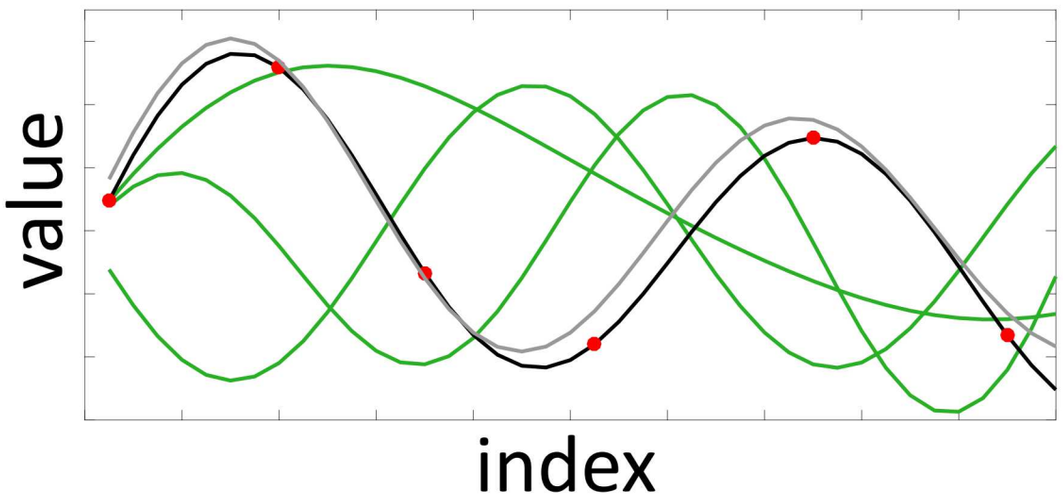


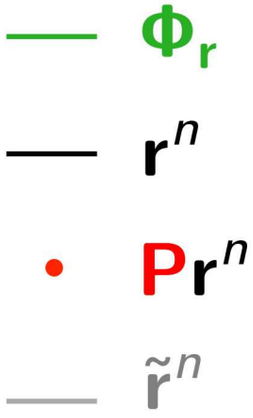
support vector machine
error prediction

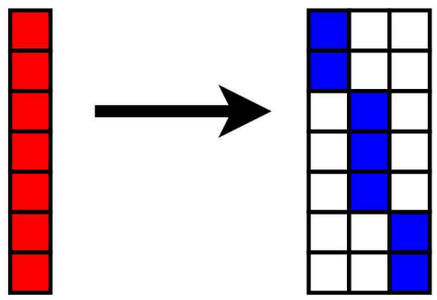












Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525