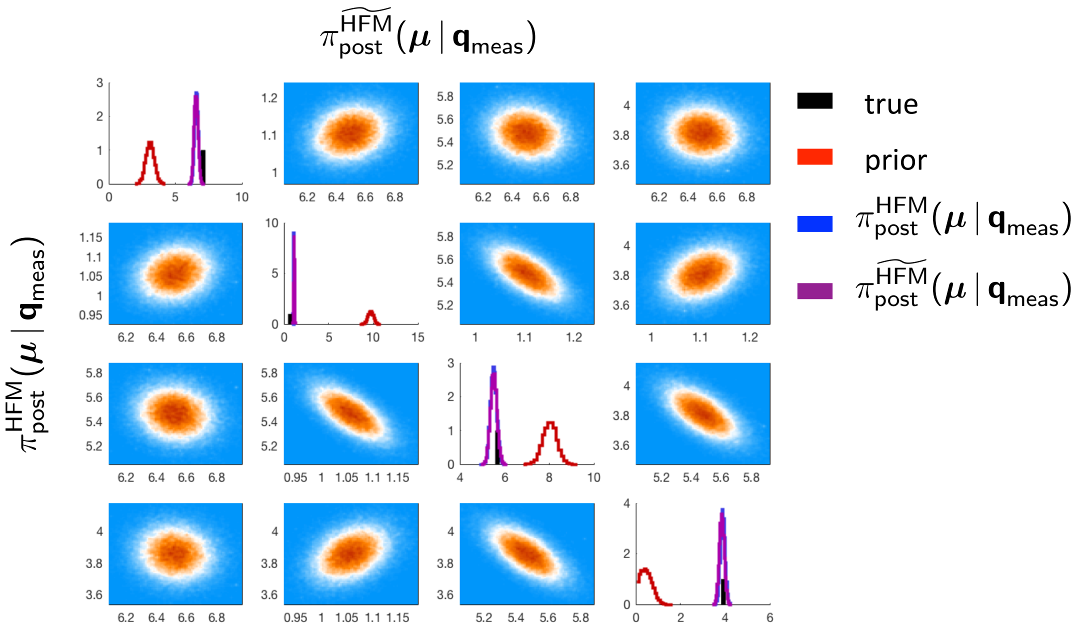
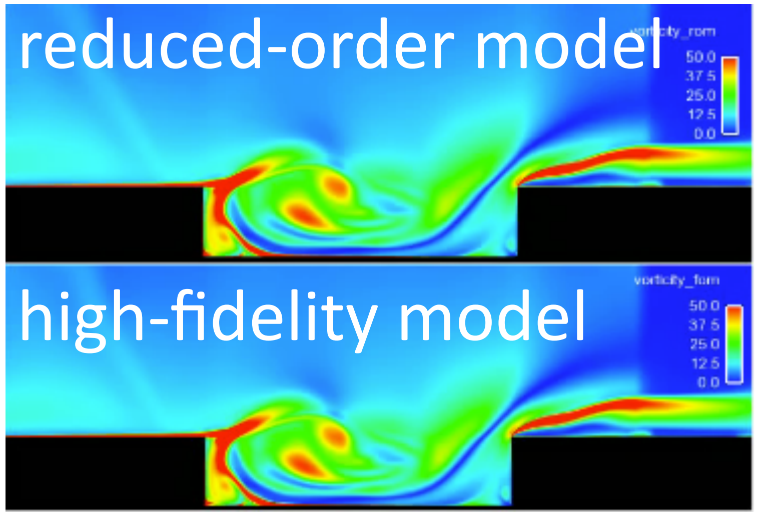
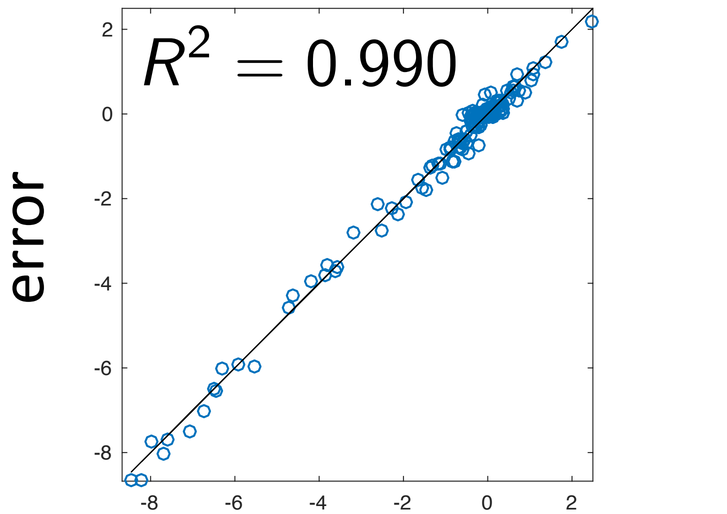
Advances in nonlinear model reduction:
least-squares Petrov–Galerkin projection and machine-learning error models







support vector machine error prediction

Kevin Carlberg

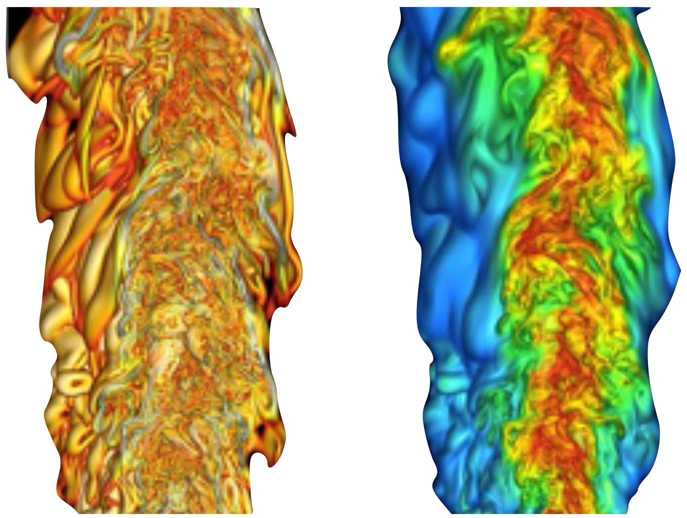
Sandia National Laboratories

SAMSI MUMS Opening Workshop
Duke University
August 21, 2018

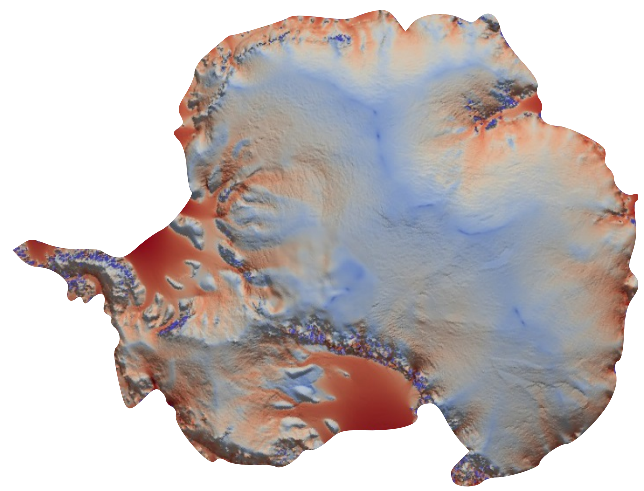
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

High-fidelity simulation

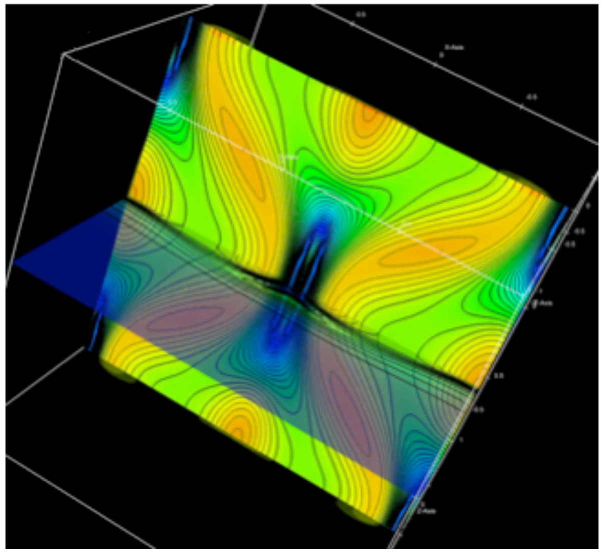
+ Indispensable across science and engineering
- *High fidelity*: extreme-scale nonlinear dynamical system models



Turbulent reacting flows
courtesy J. Chen, Sandia



Antarctic ice sheet modeling
courtesy R. Tuminaro, Sandia



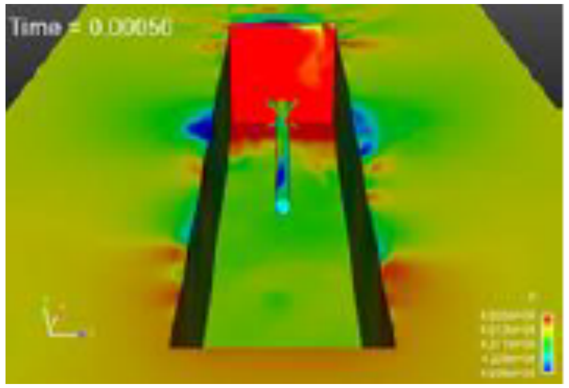
Magnetohydrodynamics
courtesy J. Shadid, Sandia

computational barrier

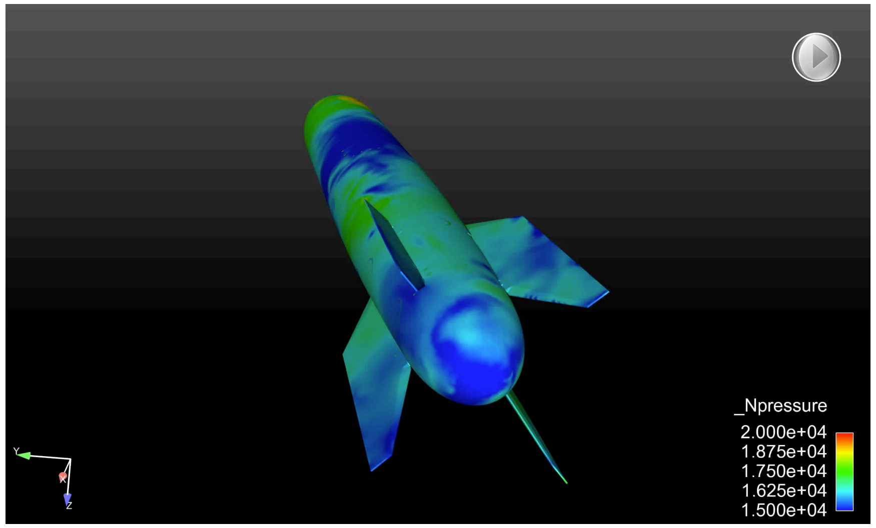
Many-query problems

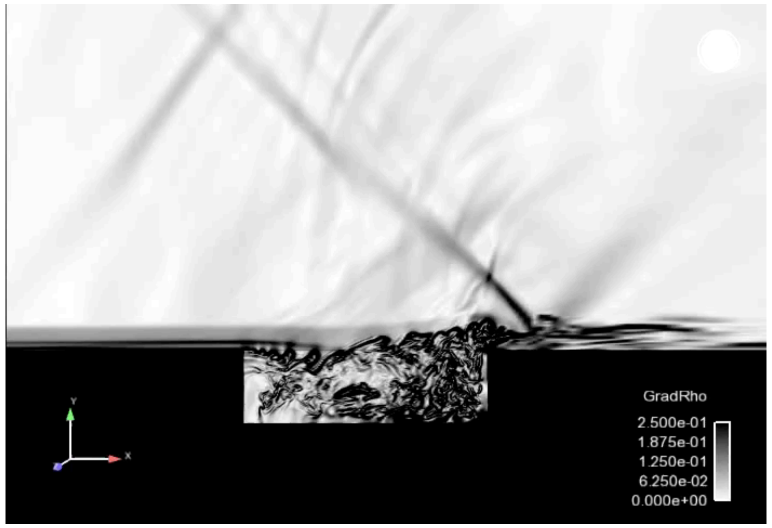
- uncertainty propagation
- Bayesian inference
- multi-objective optimization
- stochastic optimization

High-fidelity simulation: captive carry



High-fidelity simulation: captive carry





+ *Validated and predictive*: matches wind-tunnel experiments to within 5%
- *Extreme-scale*: 100 million cells, 200,000 time steps
- *High simulation costs*: 6 weeks, 5000 cores

computational barrier

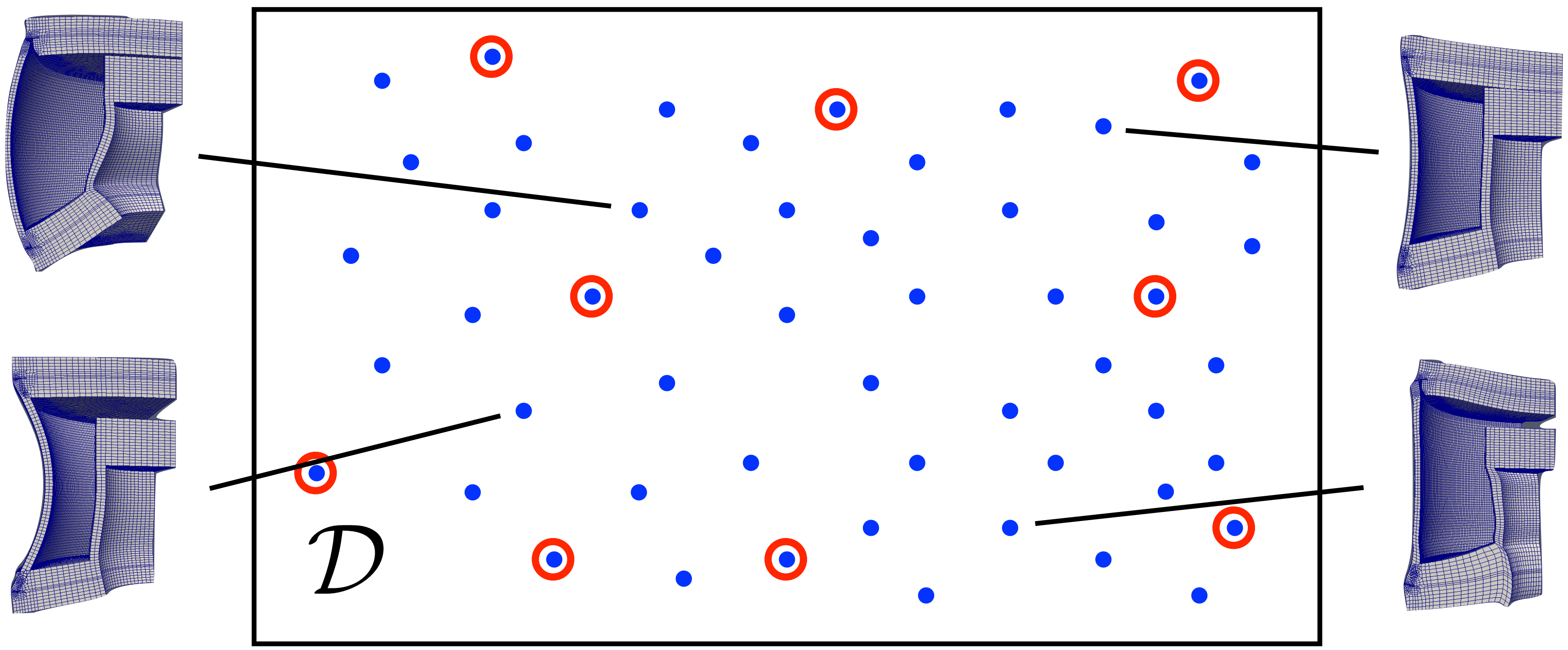
Many-query problems

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

Approach: exploit simulation data

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

***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

Our research

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

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

Our research

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

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

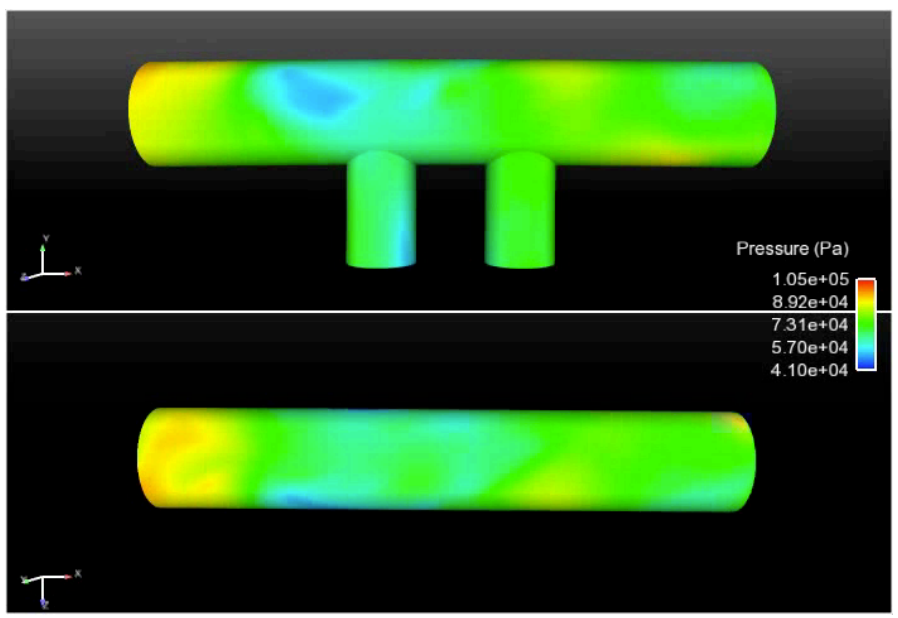
Collaborators:

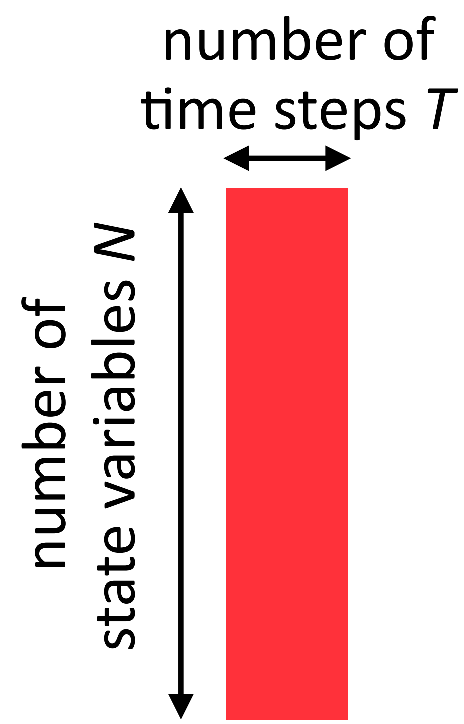
- Matthew Barone (Sandia)
- Harbir Antil (GMU)
- Charbel Farhat (Stanford University)
- Julien Cortial (Stanford University)

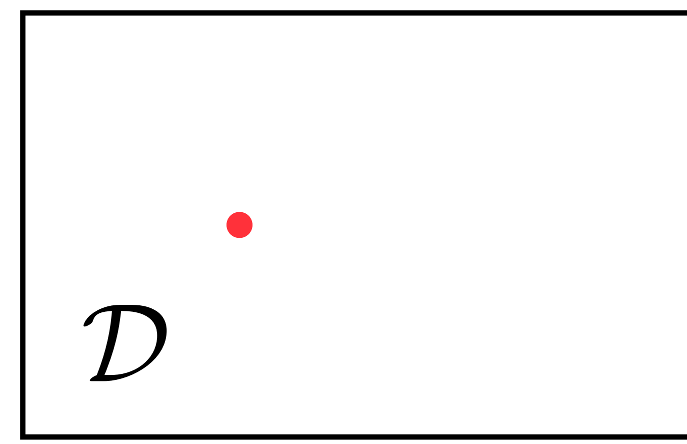
Training simulations: state tensor

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

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



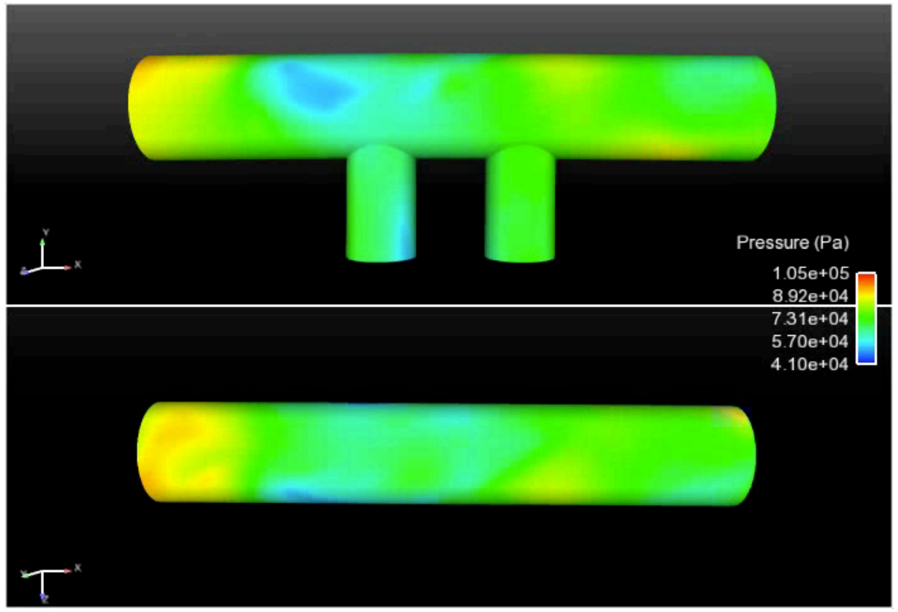




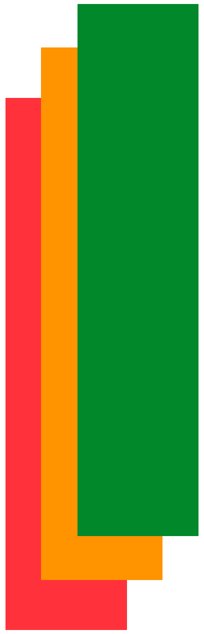
Training simulations: state tensor

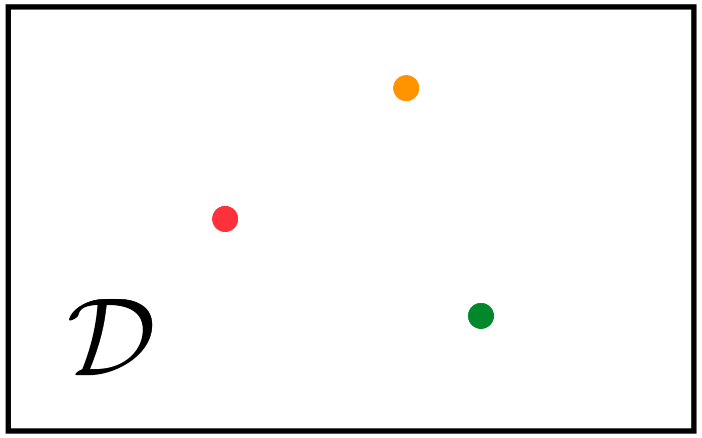
$$\text{ODE:} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$

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



$\mathcal{X} =$



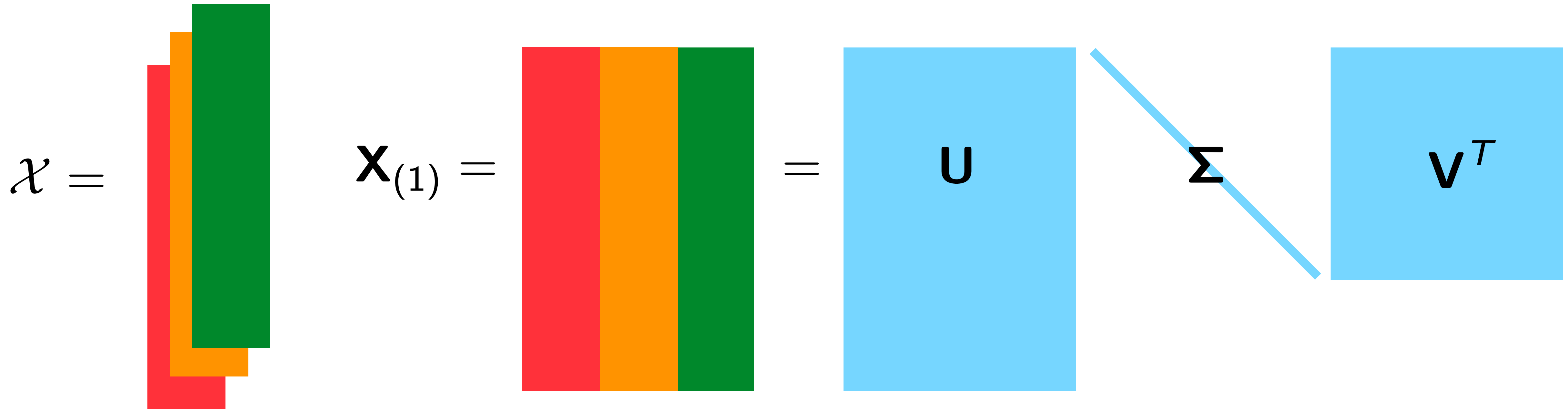


Tensor decomposition

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

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

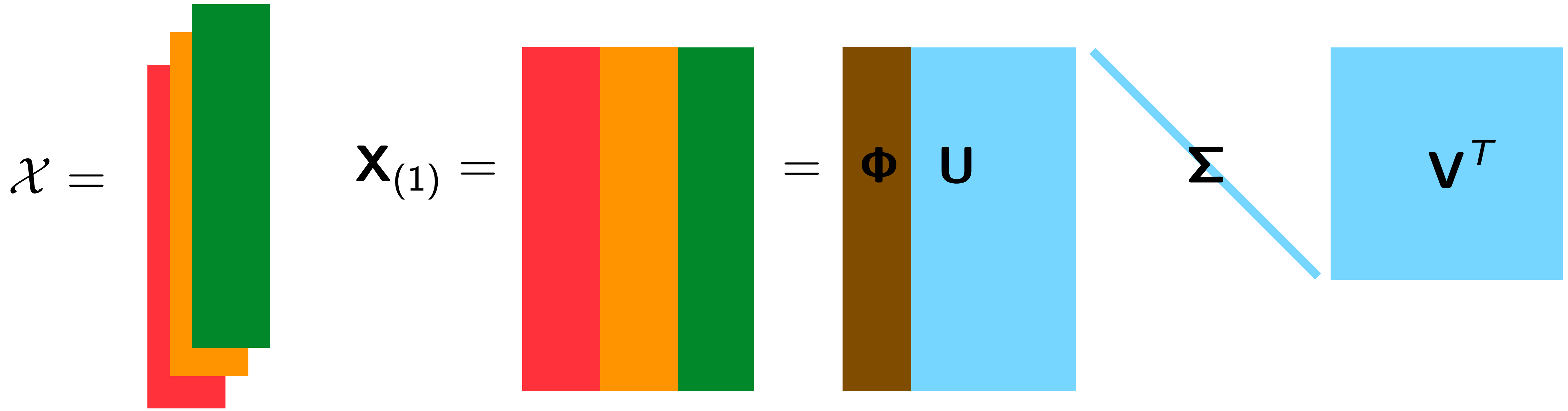


Tensor decomposition

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

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

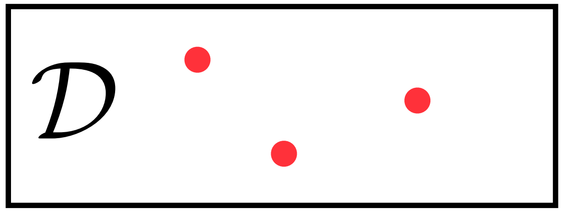


$\mathbf{\Phi}$ *columns are principal components of the spatial simulation data*

How to integrate these data with the computational model?

Previous state of the art: POD–Galerkin

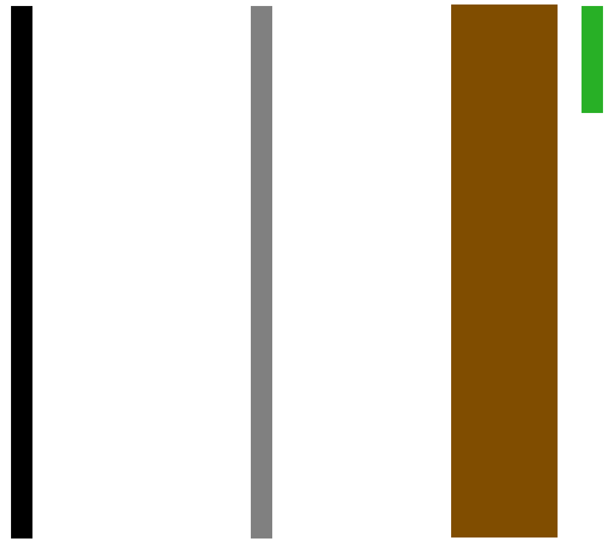
ODE: $$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}; t, \boldsymbol{\mu})$$



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

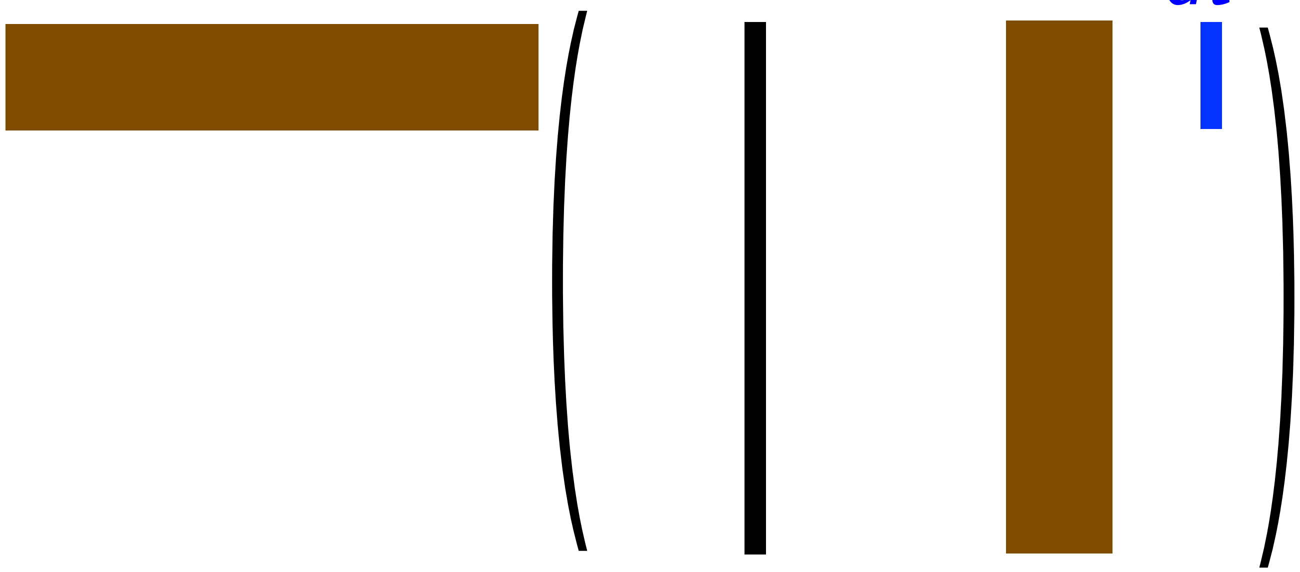
1. Reduce the number of unknowns

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

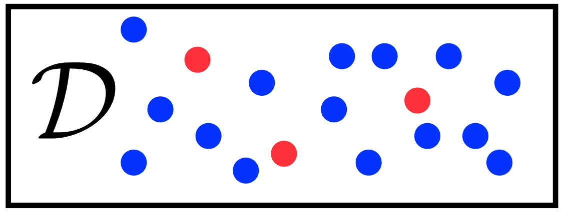


2. Reduce the number of equations

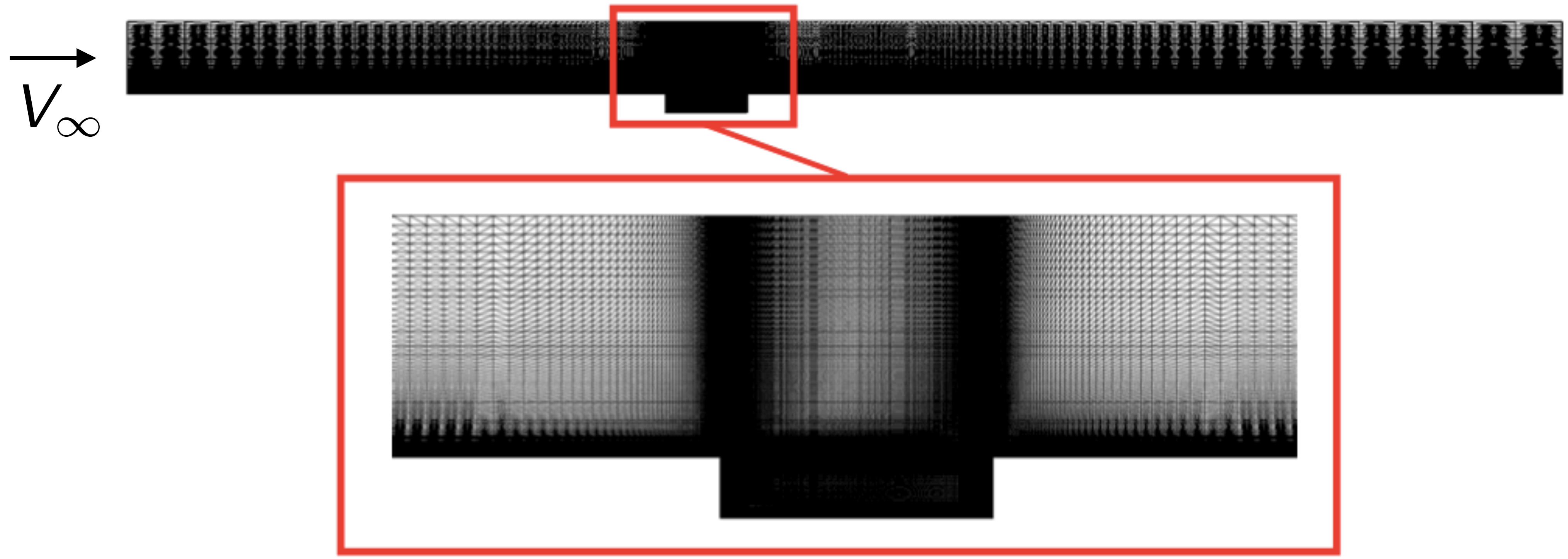
$$\boldsymbol{\Phi}^T\left(\mathbf{f}(\boldsymbol{\Phi}\hat{\mathbf{x}}; t, \boldsymbol{\mu}) - \boldsymbol{\Phi}\,\frac{d\hat{\mathbf{x}}}{dt}\right) = 0$$



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



Captive carry



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

Spatial discretization

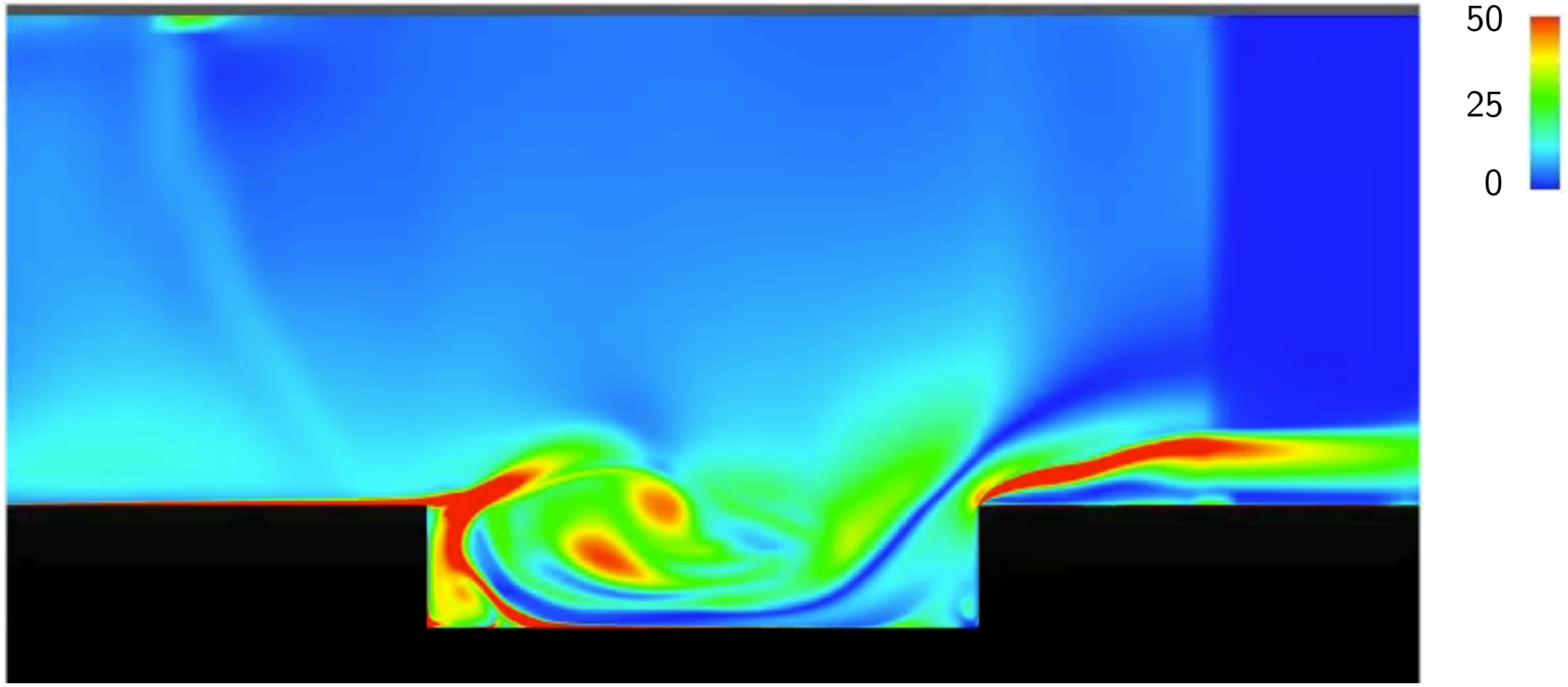
- 2nd-order finite volume
- DES turbulence model
- 1.2×10^6 degrees of freedom

Temporal discretization

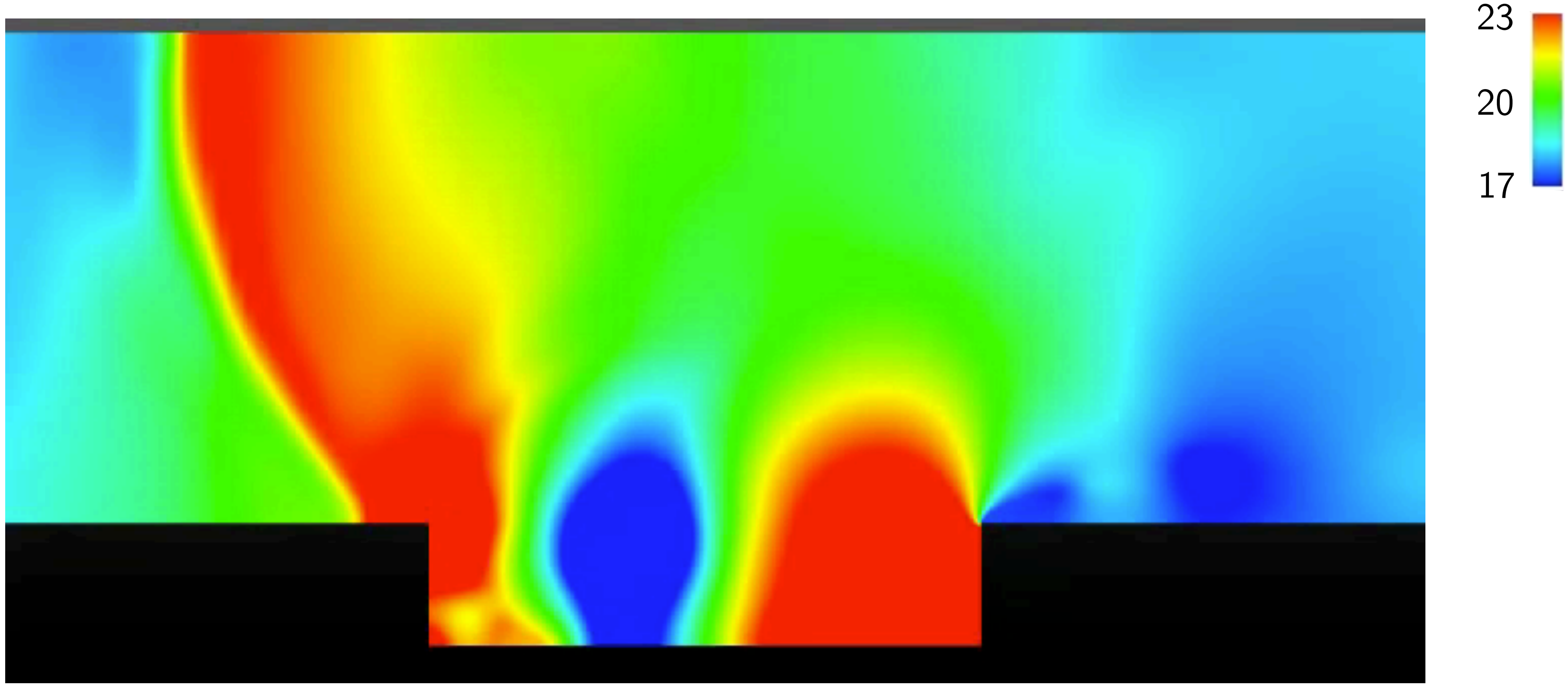
- 2nd-order BDF
- Verified time step $\Delta t = 1.5 \times 10^{-3}$
- 8.3×10^3 time instances

High-fidelity model solution

vorticity field

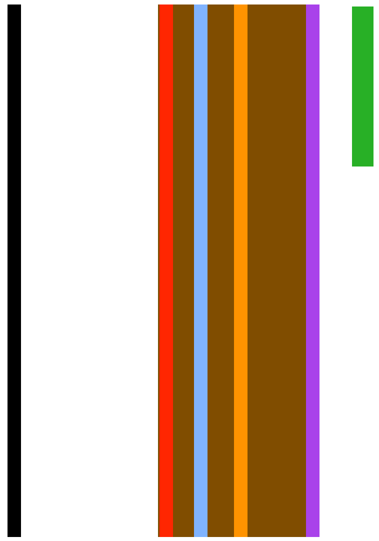


pressure field

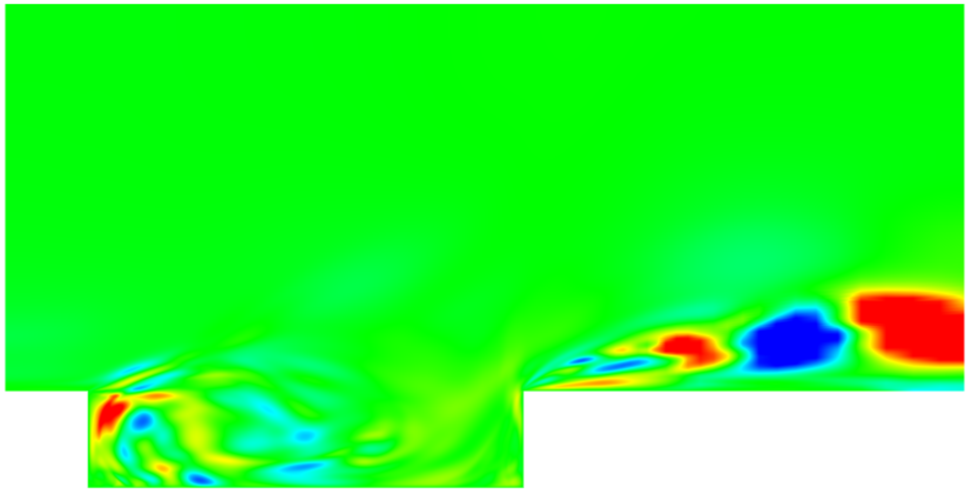


Principal components

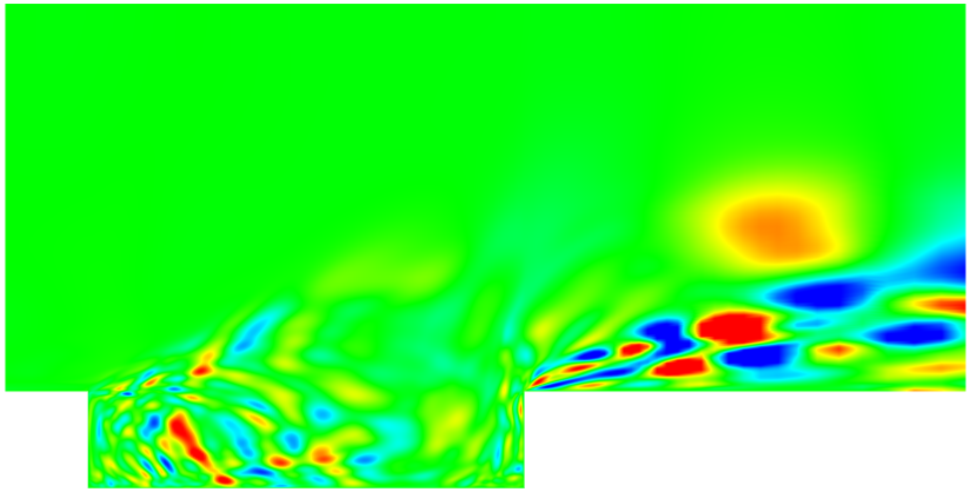
$\mathbf{x}(t) \approx \boldsymbol{\Phi}\ \hat{\mathbf{x}}(t)$



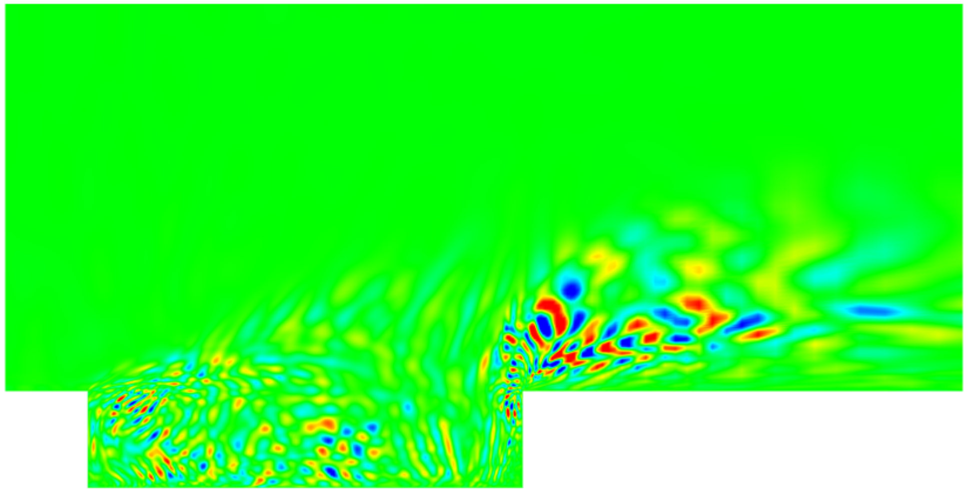
ϕ_1



ϕ_{21}

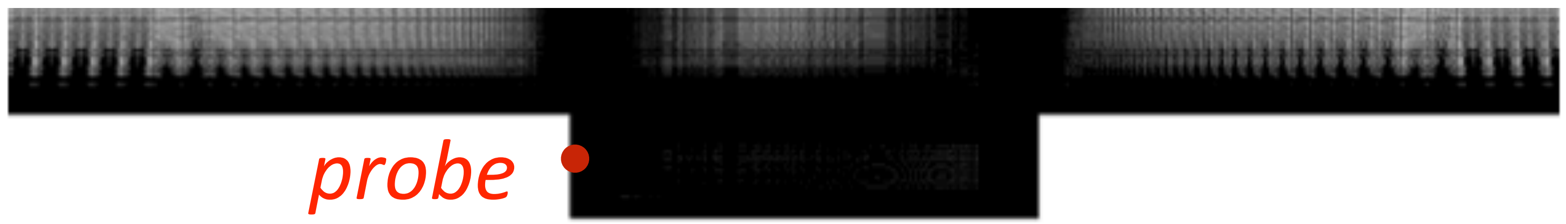


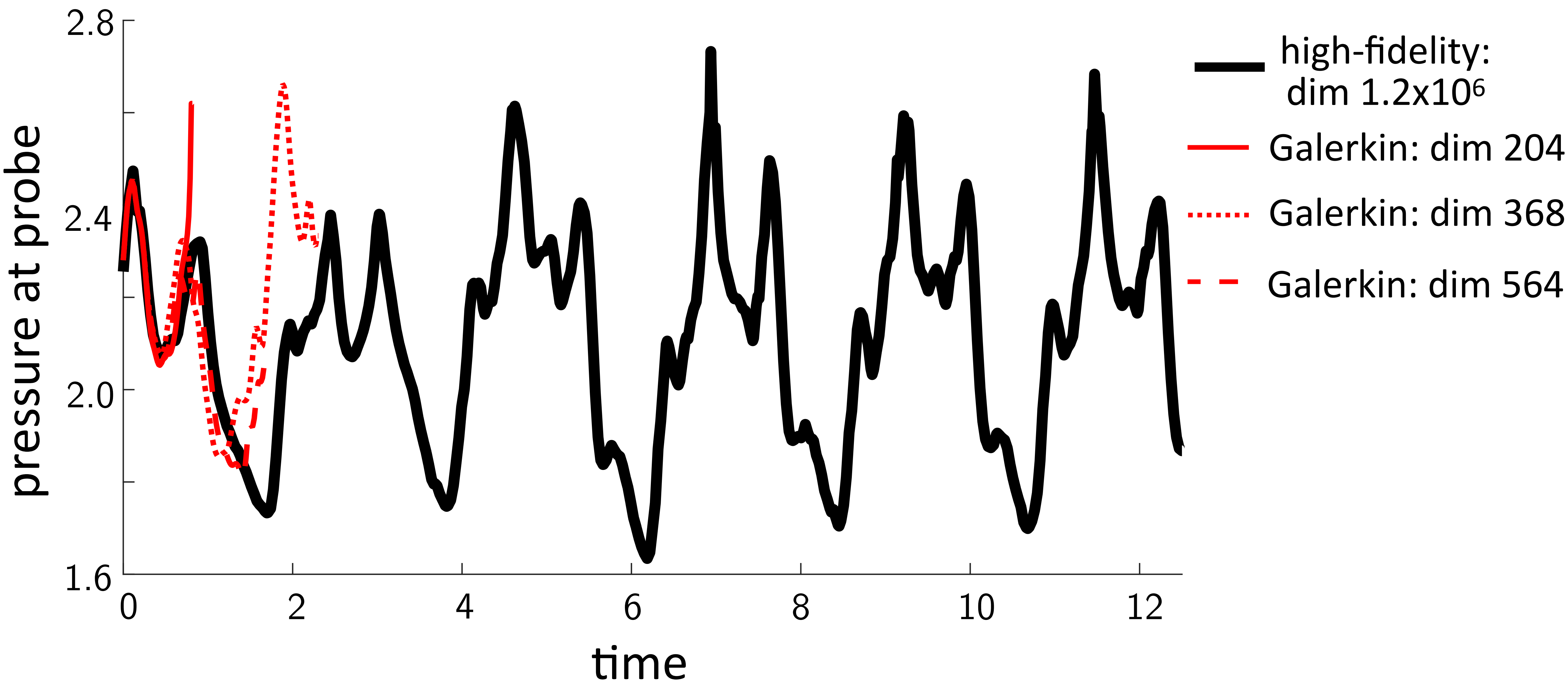
ϕ_{101}



ϕ_{401}

Galerkin performance





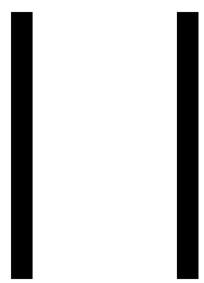
- *Galerkin projection fails regardless of basis dimension*

Can we construct a better projection?

Galerkin: time-continuous optimality

ODE

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



Galerkin ODE

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

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

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

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

OΔE

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

Galerkin OΔE

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

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

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

Residual minimization and time discretization

ODE

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

$\xrightarrow{\textit{residual minimization}}$

Galerkin ODE

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

time discretization $\downarrow$

time discretization $\downarrow$

LSPG OΔE

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

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

$\xleftarrow{\textit{residual minimization}}$

OΔE

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

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

Galerkin OΔE

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

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

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

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

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

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

Discrete-time error bound

Theorem [C., Barone, Antil, 2017]

If the following conditions hold:

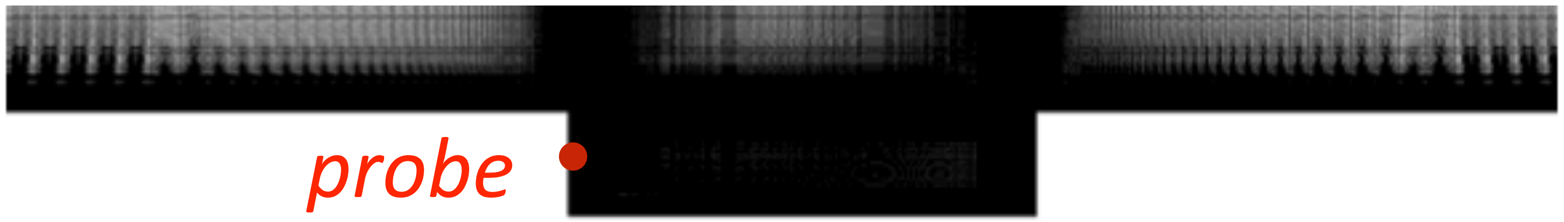
1. $\mathbf{f}(\cdot; t)$ is Lipschitz continuous with Lipschitz constant κ
2. The time step Δt is small enough such that $0 < h := |\alpha_0| - |\beta_0|\kappa\Delta t$,
3. A backward differentiation formula (BDF) time integrator is used,
4. LSPG employs $\mathbf{A} = \mathbf{I}$, then

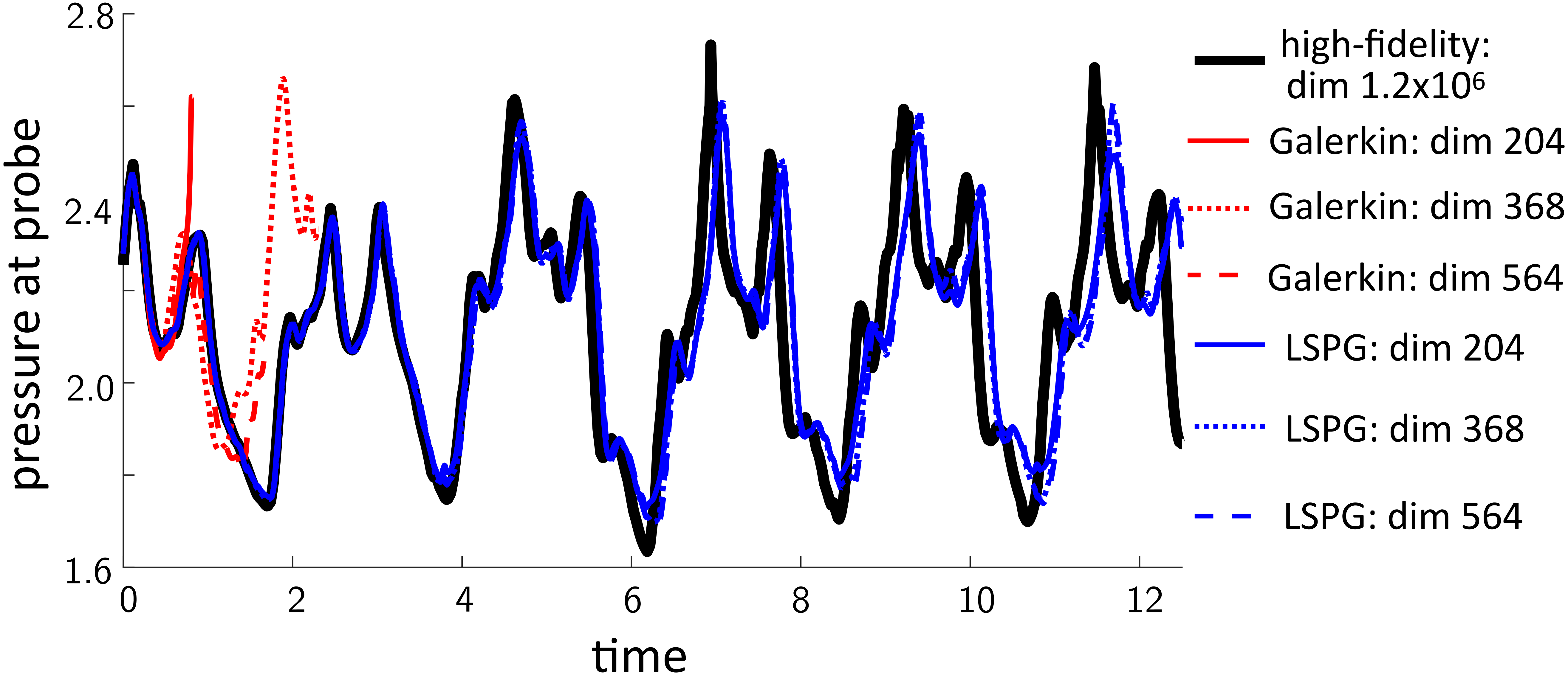
$$\|\mathbf{x}^n - \boldsymbol{\Phi}\hat{\mathbf{x}}_{\mathrm{G}}^n\|_2 \leq \frac{1}{h}\|\mathbf{r}_{\mathrm{G}}^n(\boldsymbol{\Phi}\hat{\mathbf{x}}_{\mathrm{G}}^n)\|_2 + \frac{1}{h}\sum_{\ell=1}^{k}|\alpha_\ell|\|\mathbf{x}^{n-\ell} - \boldsymbol{\Phi}\hat{\mathbf{x}}_{\mathrm{G}}^{n-\ell}\|_2$$

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

+ LSPG sequentially minimizes the error bound

LSPG performance





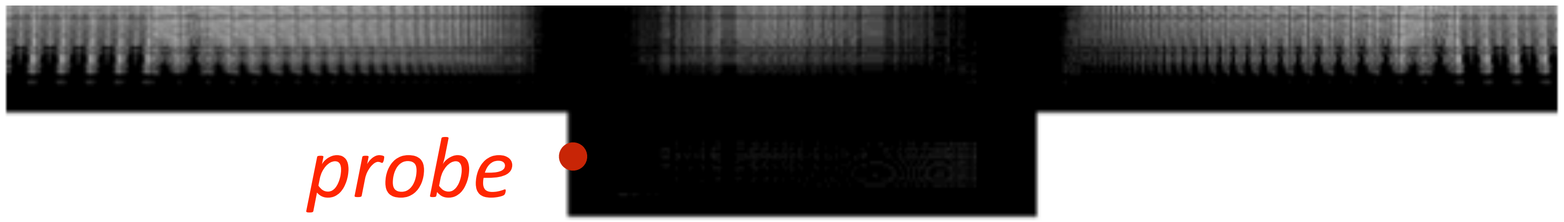
+ *LSPG is far more accurate than Galerkin*

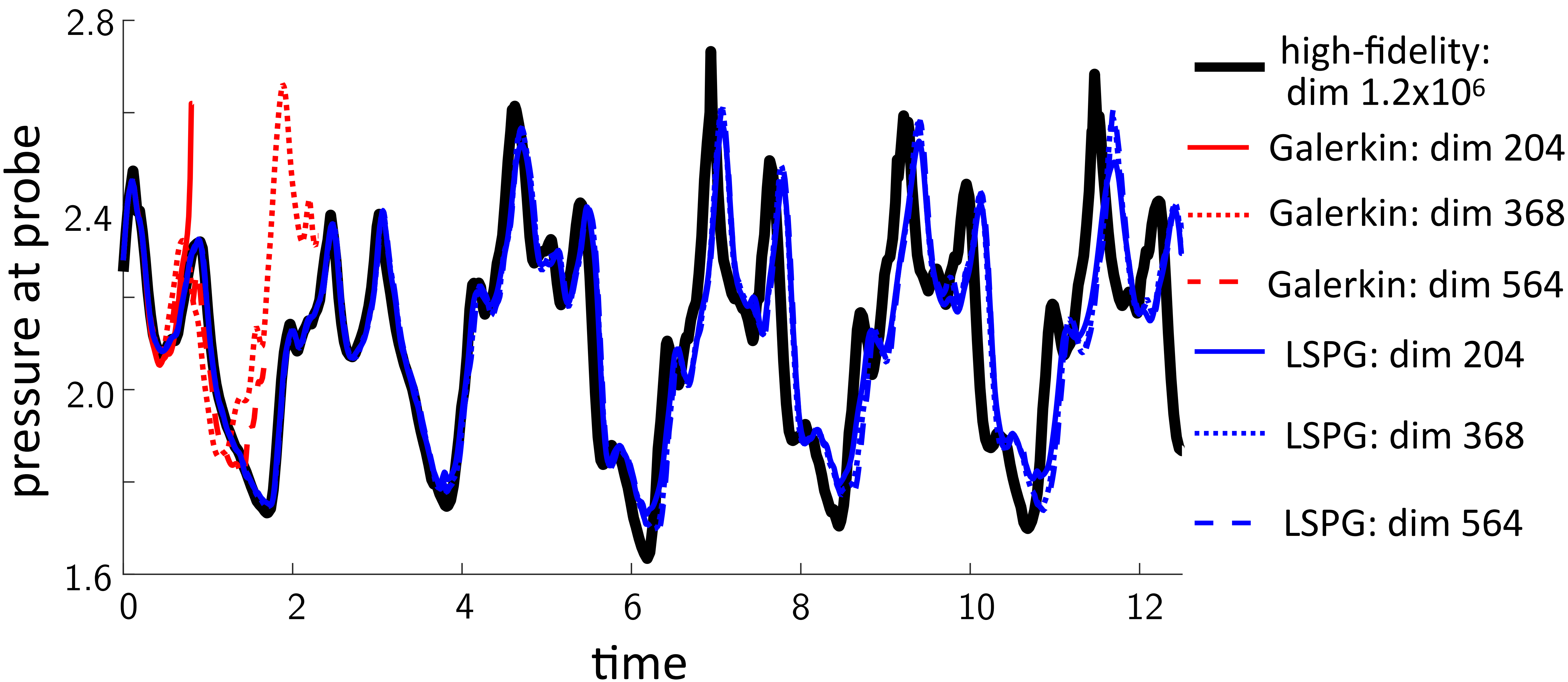
Our research

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

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

Wall-time problem





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

Why does this occur?

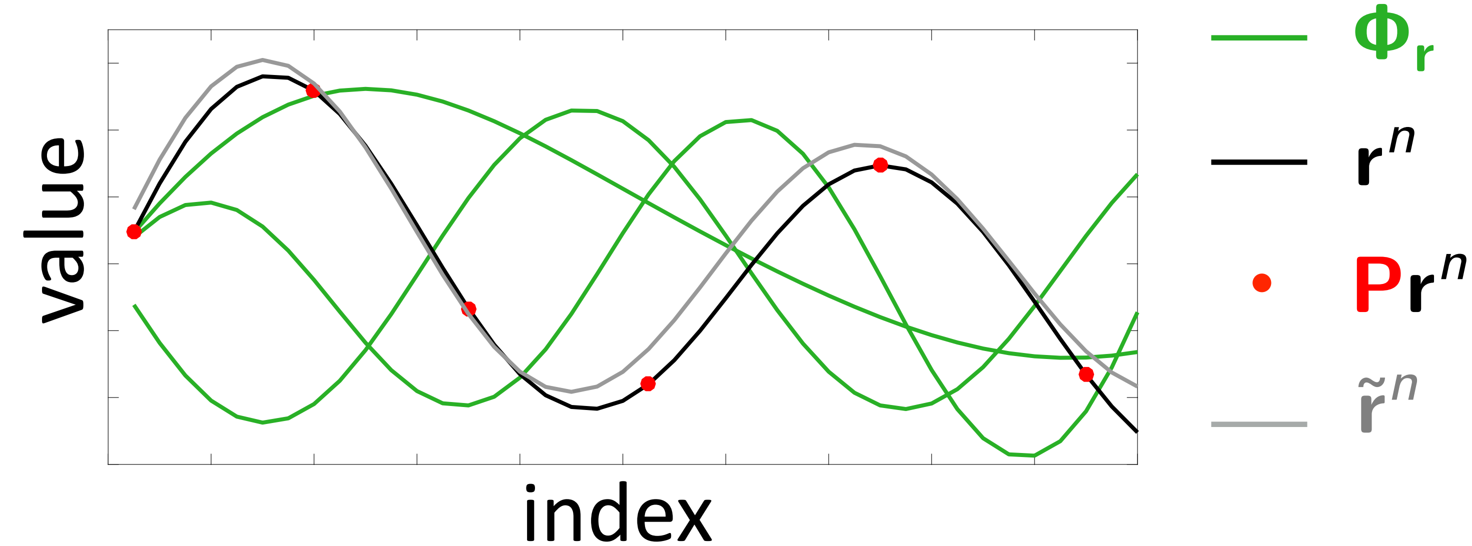
Can we fix it?

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

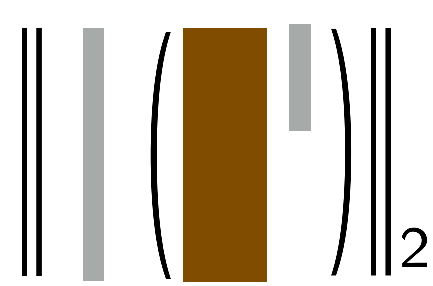
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \| \mathbf{A} \, \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) \|_2$$

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

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



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

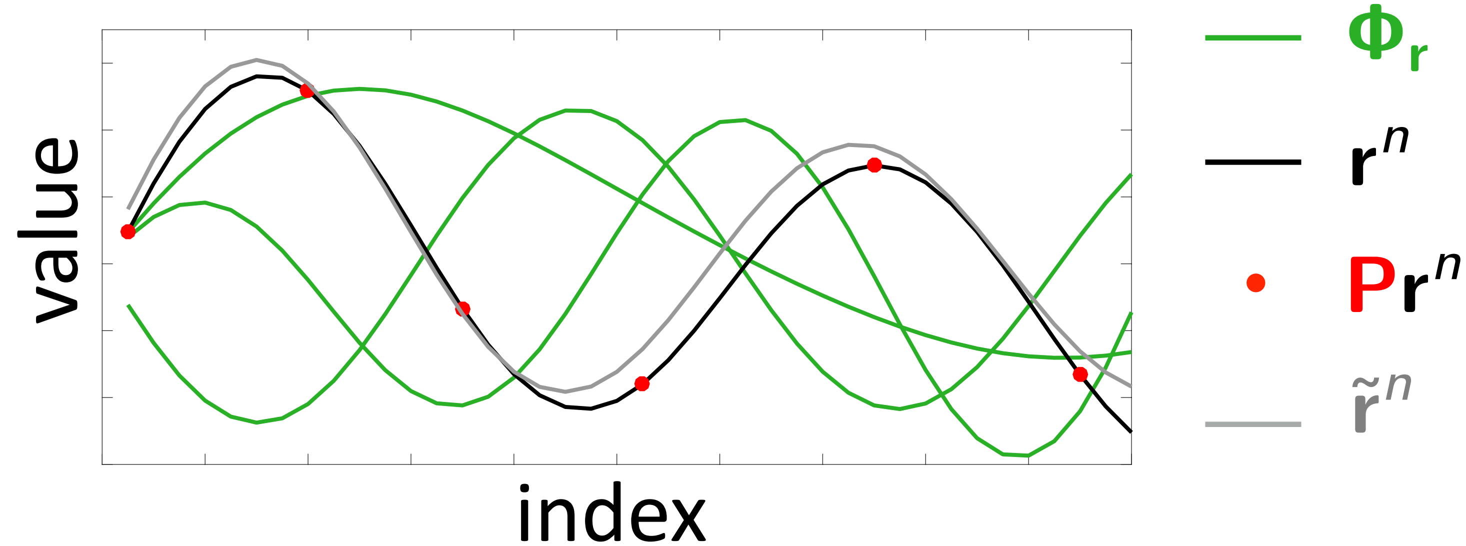


Cost reduction by gappy PCA [Everson and Sirovich, 1995]

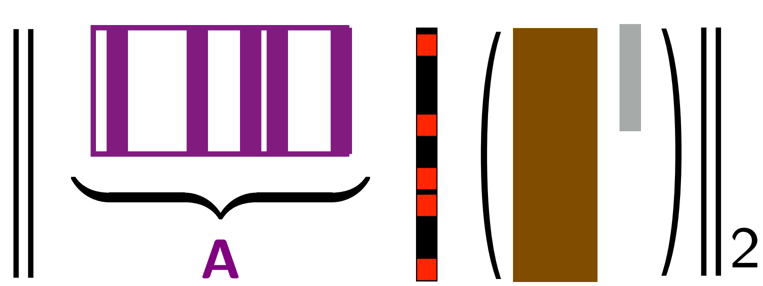
$$\underset{\hat{\mathbf{v}}}{\text{minimize}} \left\| \mathbf{A} \, \mathbf{r}^n(\boldsymbol{\Phi}\hat{\mathbf{v}}) \right\|_2$$

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

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



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

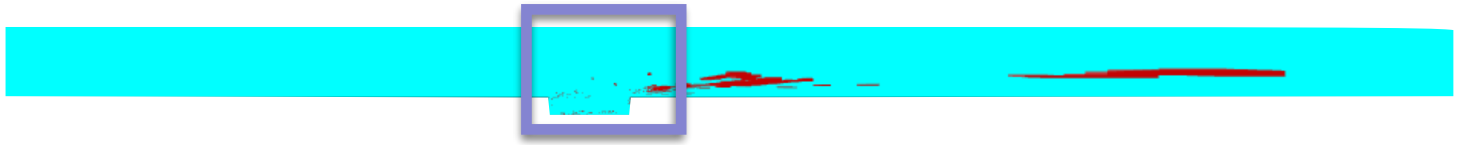


+ *Only a few elements of* $\mathbf{r}^n$ *must be computed*

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

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

sample mesh



+ *HPC on a laptop*

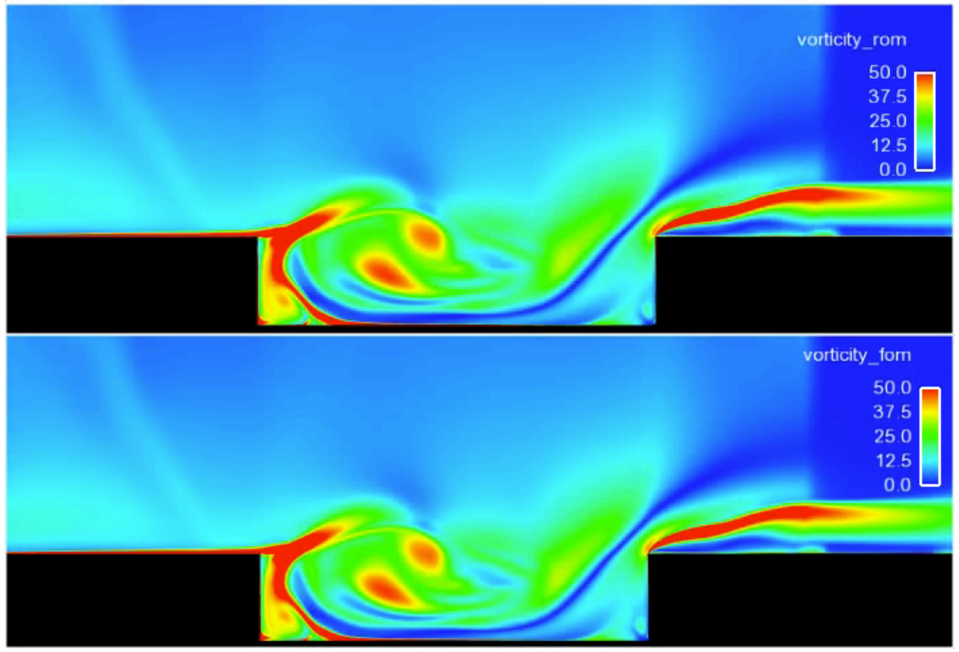
vorticity field *pressure field*

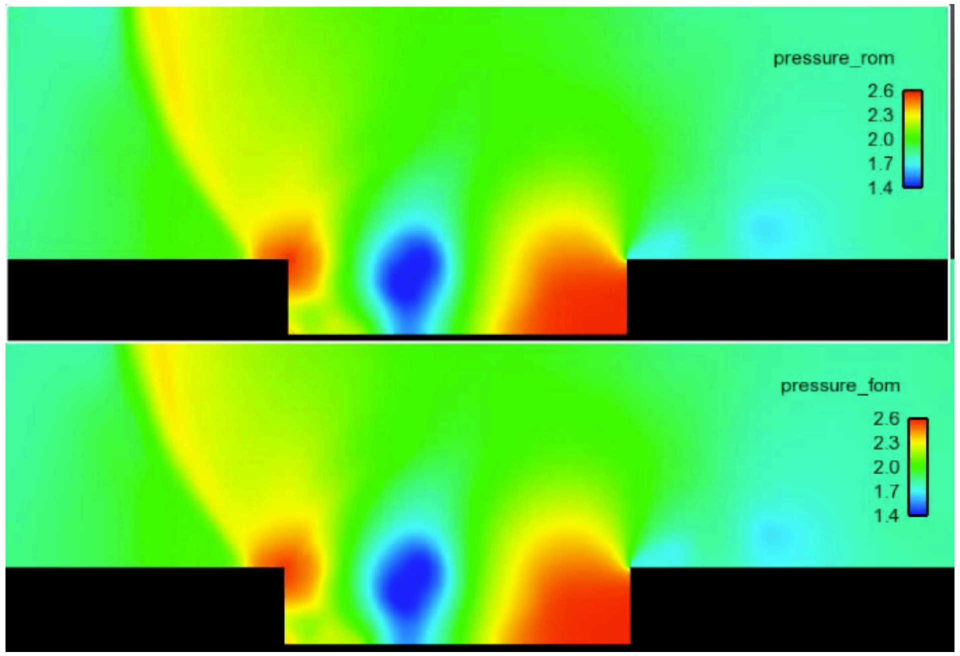
LSPG ROM with $\mathbf{A} = (\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^{+}\mathbf{P}$

32 min, 2 cores

high-fidelity

5 hours, 48 cores



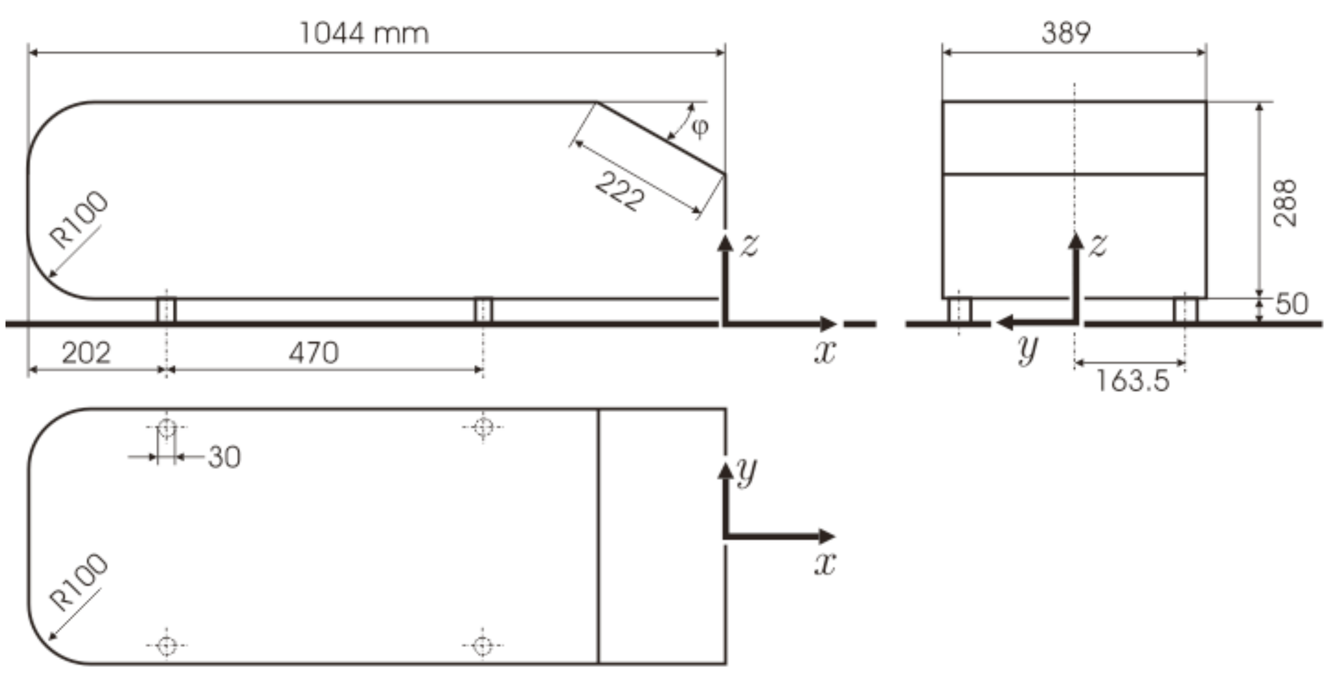


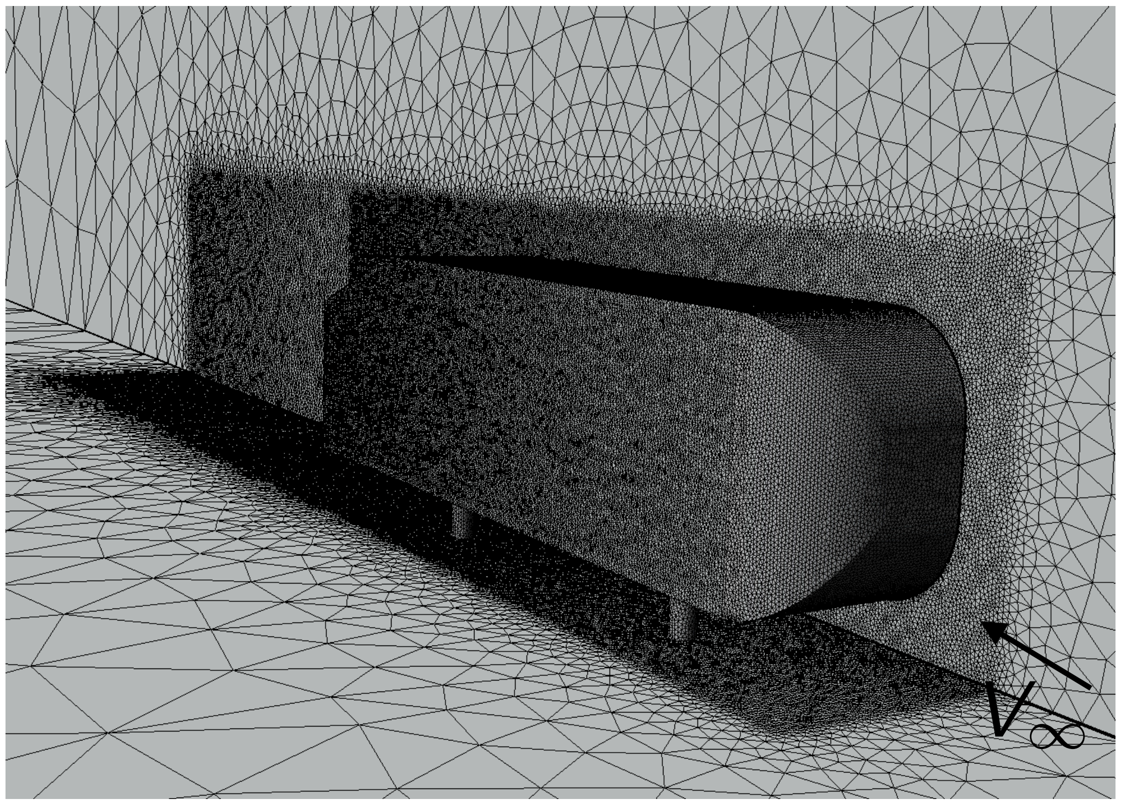
+ *229x savings in core–hours*

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

Implemented in three computational-mechanics codes at Sandia

Ahmed body [Ahmed, Ramm, Faitin, 1984]





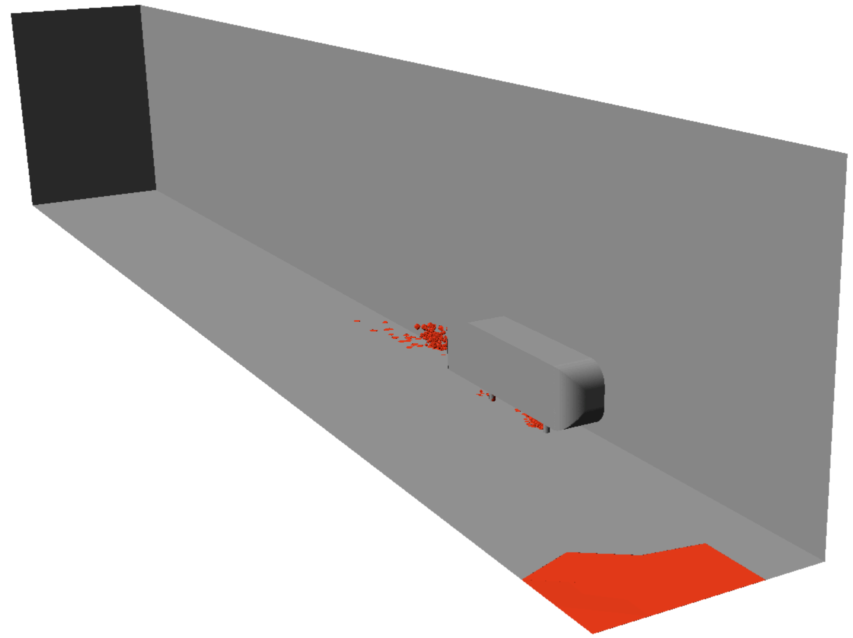
- Unsteady Navier–Stokes
- Re = 4.3 x 10^6
- M_∞ = 0.175

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 [C., Farhat, Cortial, Amsallem, 2013]

sample mesh



+ *HPC on a laptop*

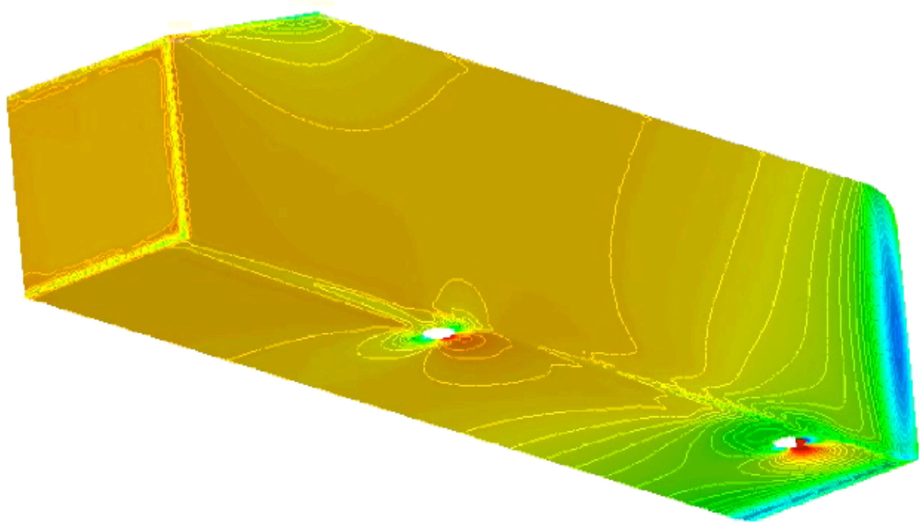
LSPG ROM with $\mathbf{A} = (\mathbf{P\Phi_r})^{+}\mathbf{P}$

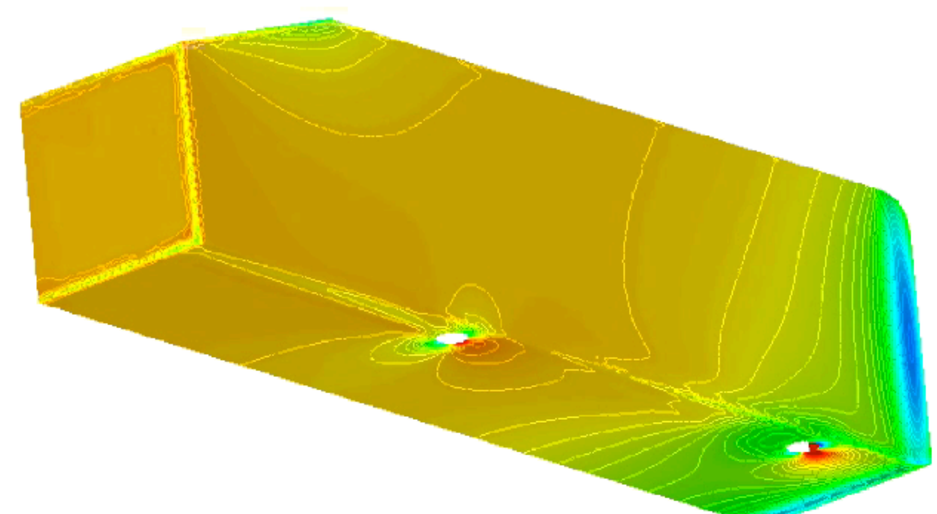
4 hours, 4 cores

high-fidelity model

13 hours, 512 cores

pressure field





+ *438x savings in core–hours*

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

Our research

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

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

Our research

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

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

Our research

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

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

Our research

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

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

Collaborators:

- Martin Drohmann (formerly Sandia)
- Wayne Uy (Cornell University)
- Fei Lu (Johns Hopkins University)
- Matthias Morzfeld (U of Arizona)
- Brian Freno (Sandia)

Surrogate modeling in UQ

inputs $\boldsymbol{\mu}$ $\rightarrow$ $\boxed{\textit{high-fidelity model}}$ $\rightarrow$ *outputs* $\mathbf{q}_{\mathrm{HFM}}$

- high-fidelity-model (HFM) noise model: $\mathbf{q}_{\mathrm{meas}} = \mathbf{q}_{\mathrm{HFM}}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$
- measurement noise $\boldsymbol{\varepsilon}$ has probability distribution $\pi_{\varepsilon}(\cdot)$
- HFM likelihood: $\pi_{\mathrm{HFM}}(\mathbf{q}_{\mathrm{meas}} \,|\, \boldsymbol{\mu}) = \pi_{\boldsymbol{\varepsilon}}(\mathbf{q}_{\mathrm{meas}} - \mathbf{q}_{\mathrm{HFM}}(\boldsymbol{\mu}))$

inputs $\boldsymbol{\mu}$ $\rightarrow$ $\boxed{\textit{surrogate model}}$ $\rightarrow$ *outputs* $\mathbf{q}_{\mathrm{surr}}$

- surrogate noise model: $\mathbf{q}_{\mathrm{meas}} = \mathbf{q}_{\mathrm{surr}}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$
- surrogate likelihood: $\pi_{\mathrm{surr}}(\mathbf{q}_{\mathrm{meas}} \,|\, \boldsymbol{\mu}) = \pi_{\boldsymbol{\varepsilon}}(\mathbf{q}_{\mathrm{meas}} - \mathbf{q}_{\mathrm{surr}}(\boldsymbol{\mu}))$
  - inconsistent with HFM noise model

Surrogate modeling in UQ

$$\mathbf{q}_{\mathsf{HFM}}(\boldsymbol{\mu}) = \mathbf{q}_{\mathsf{surr}}(\boldsymbol{\mu}) + \boldsymbol{\delta}(\boldsymbol{\mu})$$

- HFM noise model: $\mathbf{q}_{\mathsf{meas}} = \mathbf{q}_{\mathsf{HFM}}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$
  $= \mathbf{q}_{\mathsf{surr}}(\boldsymbol{\mu}) + \boldsymbol{\delta}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$
- HFM likelihood: $\pi_{\mathsf{HFM}}(\mathbf{q}_{\mathsf{meas}} \mid \boldsymbol{\mu}) = \pi_{\boldsymbol{\varepsilon}}(\mathbf{q}_{\mathsf{meas}} - \mathbf{q}_{\mathsf{HFM}}(\boldsymbol{\mu}))$
  $= \pi_{\boldsymbol{\varepsilon}}(\mathbf{q}_{\mathsf{meas}} - \mathbf{q}_{\mathsf{surr}}(\boldsymbol{\mu}) - \boldsymbol{\delta}(\boldsymbol{\mu}))$

\+ equivalent to HFM formulation

\+ not practical: the (deterministic) error $\boldsymbol{\delta}(\boldsymbol{\mu})$ is generally unknown

How can we account for the error $\boldsymbol{\delta}(\boldsymbol{\mu})$ in a manner that is consistent and practical?

Surrogate modeling in UQ

$$\mathbf{q}_{\text{HFM}}(\boldsymbol{\mu}) = \mathbf{q}_{\text{surr}}(\boldsymbol{\mu}) + \delta(\boldsymbol{\mu})$$

Approach: *statistical model* $\tilde{\delta}(\boldsymbol{\mu})$ *for the error that models its uncertainty*

$$\underbrace{\tilde{\mathbf{q}}_{\text{HFM}}(\boldsymbol{\mu})}_{\text{stochastic}} = \underbrace{\mathbf{q}_{\text{surr}}(\boldsymbol{\mu})}_{\text{deterministic}} + \underbrace{\tilde{\delta}(\boldsymbol{\mu})}_{\text{stochastic}}$$

- statistical HFM noise model: $\mathbf{q}_{\text{meas}} = \tilde{\mathbf{q}}_{\text{HFM}}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$ $= \mathbf{q}_{\text{surr}}(\boldsymbol{\mu}) + \tilde{\delta}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon}$
- stochastic HFM likelihood: $\pi_{\widetilde{\text{HFM}}}(\mathbf{q}_{\text{meas}} \mid \boldsymbol{\mu}) = \pi_{\boldsymbol{\varepsilon}+\tilde{\delta}}(\mathbf{q}_{\text{meas}} - \mathbf{q}_{\text{surr}}(\boldsymbol{\mu}))$

\+ consistent with HFM noise model

\+ practical if the statistical error model $\tilde{\delta}$ is computable

Desired properties in statistical error model $\tilde{\delta}(\boldsymbol{\mu})$

1. cheaply computable: similar cost to evaluating the surrogate
2. low variance: introduces little epistemic uncertainty
3. generalizable: correctly models the error

How can we construct a statistical error model for reduced-order models?

Approximate-solution surrogate models

High-fidelity model

- governing equations: $\mathbf{r}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \mathbf{0}$
- quantity of interest: $q_{\mathrm{HFM}}(\boldsymbol{\mu}) := q(\mathbf{x}(\boldsymbol{\mu}))$

Approximate-solution surrogate model

- approximate solution: $\tilde{\mathbf{x}}(\boldsymbol{\mu}) \approx \mathbf{x}(\boldsymbol{\mu})$
- quantity of interest: $q_{\mathrm{surr}}(\boldsymbol{\mu}) := q(\tilde{\mathbf{x}}(\boldsymbol{\mu}))$

Types of approximate solutions

- *Reduced-order model*:

$$\boldsymbol{\Psi}^T \mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}}; \boldsymbol{\mu}) = \mathbf{0}, \quad \tilde{\mathbf{x}} = \boldsymbol{\Phi}\hat{\mathbf{x}}$$

- *Low-fidelity model*:

$$\mathbf{r}_{\mathrm{LF}}(\mathbf{x}_{\mathrm{LF}}; \boldsymbol{\mu}) = \mathbf{0}, \quad \tilde{\mathbf{x}} = \mathbf{p}(\mathbf{x}_{\mathrm{LF}})$$

- *Inexact solution:* compute $\mathbf{x}^{(k)}$, $k = 1, \ldots, K$ such that

$$\|\mathbf{r}(\mathbf{x}^{(K)}; \boldsymbol{\mu}) = \mathbf{0}\|_2 \leq \epsilon, \quad \tilde{\mathbf{x}} = \mathbf{x}^{(K)}$$

What methods exist for quantifying the error $\delta(\boldsymbol{\mu}) := q_{\mathrm{HFM}}(\boldsymbol{\mu}) - q_{\mathrm{surr}}(\boldsymbol{\mu})$?

1) **Error indicators**: residual norm

- HFM governing equations: $\mathbf{r}(\mathbf{x}(\boldsymbol{\mu});\boldsymbol{\mu}) = \mathbf{0}$ (1)

- Approximate solution: $\tilde{\mathbf{x}}(\boldsymbol{\mu}) \approx \mathbf{x}(\boldsymbol{\mu})$ (2)

- Substitute (2) into the residual of (1) and take the norm:

$$\|\mathbf{r}(\tilde{\mathbf{x}};\boldsymbol{\mu})\|_2$$

- *Applications:* termination criterion, greedy methods, trust regions
  [Bui-Thanh et al., 2008; Hine and Kunkel, 2012; Wu and Hetmaniuk, 2015; Zahr, 2016]

\+ *Informative*: zero for high-fidelity model

\- *Deterministic*: not a statistical error model

\- *Low quality*: relationship to error depends on conditioning

1) **Error indicators**: dual-weighted residual

- Approximate HFM quantity of interest to first order

$$q(\mathbf{x}) = q(\tilde{\mathbf{x}}) + \frac{\partial q}{\partial \mathbf{x}}(\tilde{\mathbf{x}})(\mathbf{x} - \tilde{\mathbf{x}}) + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2) \qquad (1)$$

- Approximate HFM residual to first order

$$\mathbf{0} = \mathbf{r}(\mathbf{x}) = \mathbf{r}(\tilde{\mathbf{x}}) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\tilde{\mathbf{x}})(\mathbf{x} - \tilde{\mathbf{x}}) + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

- Solve for the error

$$\mathbf{x} - \tilde{\mathbf{x}} = -[\frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\tilde{\mathbf{x}})]^{-1}\mathbf{r}(\tilde{\mathbf{x}}) + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2) \qquad (2)$$

- Substitute (2) in (1): $\boxed{q(\mathbf{x}) - q(\tilde{\mathbf{x}}) = \mathbf{y}^T\mathbf{r}(\tilde{\mathbf{x}}) + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)}$

$$\boxed{\frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T\mathbf{y} = -\frac{\partial q}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T}$$

- *Applications:* adaptive mesh refinement
  [Babuska and Miller, 1984; Becker and Rannacher, 1996; Rannacher, 1999; Venditti and Darmofal, 2000; Fidkowski, 2007]

\+ *Accurate*: second-order-accurate approximation

\- *Deterministic*: not a statistical error model

2) Rigorous *a posteriori* error bound

Proposition

If the following conditions hold:

1. $\mathbf{r}(\cdot;\boldsymbol{\mu})$ is inf–sup stable, i.e., for all $\boldsymbol{\mu} \in \mathcal{D}$, there exists $\alpha(\boldsymbol{\mu}) > 0$ s.t.
$$\|\mathbf{r}(\mathbf{z}_1;\boldsymbol{\mu}) - \mathbf{r}(\mathbf{z}_2;\boldsymbol{\mu})\|_2 \geq \alpha(\boldsymbol{\mu})\|\mathbf{z}_1 - \mathbf{z}_2\|_2, \quad \forall \mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^N$$
2. $q(\cdot)$ is Lipschitz continuous, i.e., there exits $\beta > 0$ such that
$$|q(\mathbf{z}_1) - q(\mathbf{z}_2)| \leq \beta\|\mathbf{z}_1 - \mathbf{z}_2\|_2, \quad \forall \mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^N$$

then the quantity-of-interest error can be bounded as

$$|q(\mathbf{x}) - q(\tilde{\mathbf{x}})| \leq \frac{\beta}{\alpha}\|\mathbf{r}(\tilde{\mathbf{x}};\boldsymbol{\mu})\|_2$$

- *Applications*: reduced-order models
  [Rathinam and Petzold, 2003; Grepl and Patera, 2005; Antoulas, 2005; Hinze and Volkwein, 2005; C. et al., 2017]

\+ *Certification*: guaranteed bound

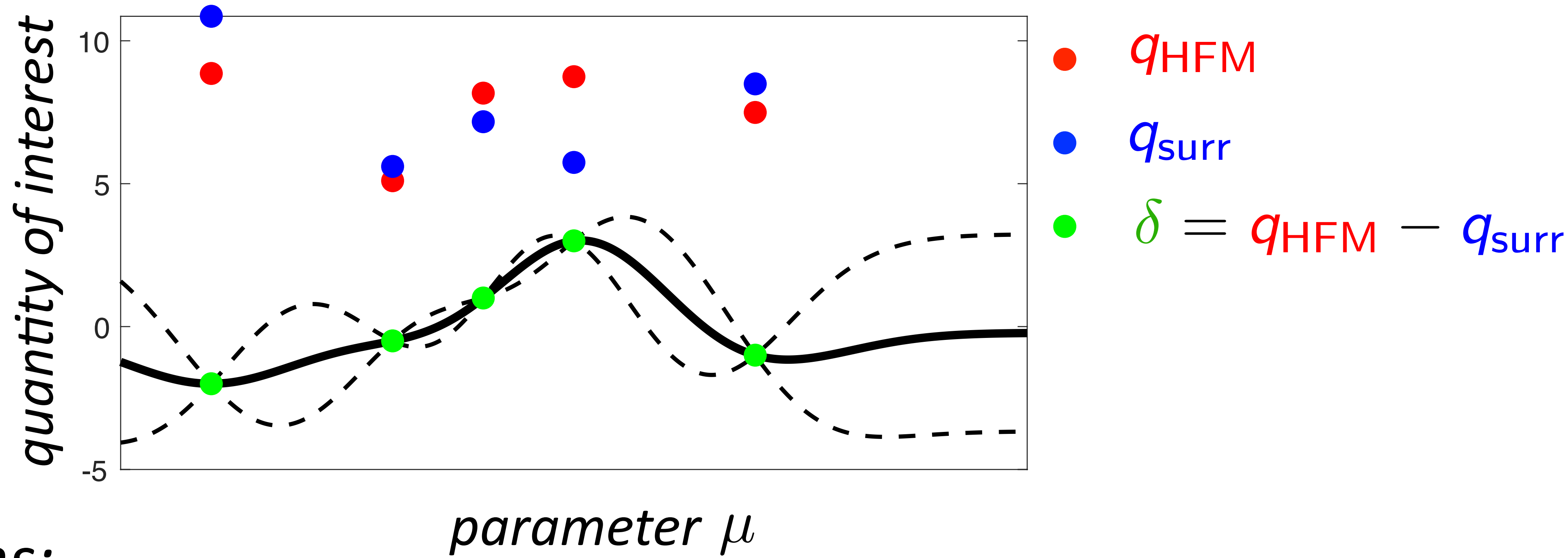
– *Lack sharpness*: orders-of-magnitude overestimation

– *Difficult to implement*: require bounds for inf–sup/Lipschitz constants

– *Deterministic*: not a statistical error model

3) Model-discrepancy approach

$$\tilde{\delta}(\boldsymbol{\mu}) \sim \mathcal{N}(\mu(\boldsymbol{\mu}); \sigma^2(\boldsymbol{\mu}))$$



- *Applications:*
  - Model calibration [Kennedy, O'Hagan, 2001; Higdon et al., 2003; Higdon et al., 2004]
  - Multifidelity optimization [Gano et al., 2005; Huang et al., 2006; March, Willcox, 2012; Ng, Eldred, 2012]

\+ *General*: applicable to any surrogate model

\+ *Statistical*: interpretable as a statistical error model

\+ *Epistemic uncertainty quantified*: through variance

– *Poorly informative inputs*: parameters $\boldsymbol{\mu}$ weakly related to the error

– *Poor scalability*: difficult in high-dimensional parameter spaces

– *Thus, can introduce large epistemic uncertainty*: large variance

Objective

***Goal**: combine the strengths of*

1. *error indicators,*
2. *rigorous a posteriori error bounds, and*
3. *the model-discrepancy approach*

A posteriori**:** use residual-based quantities computed by the surrogate

- strength of #1 and #2

+ *Informative inputs*: quantities are strongly related to the error
+ *Thus, can lead to lower epistemic uncertainty*: lower variance

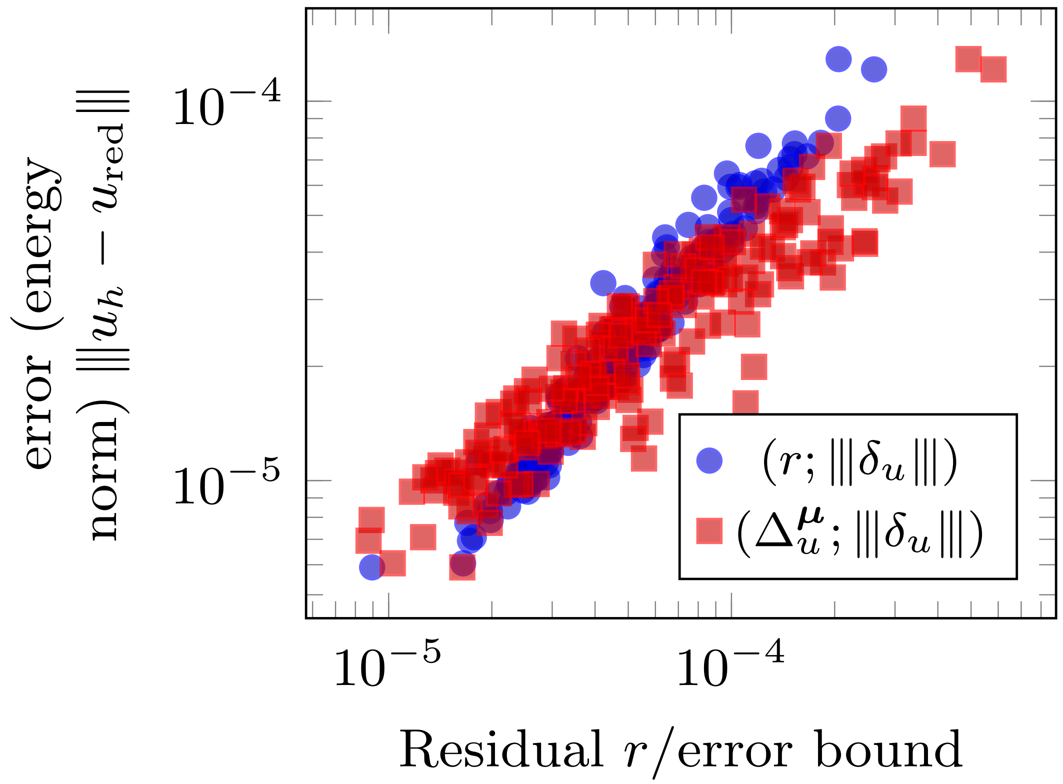
Error modeling: statistical model for the error

- strength of #3

+ *Statistical*: interpretable as a statistical error model
+ *Epistemic uncertainty quantified*: through variance

Main idea

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



- So, these are informative features: can predict the error with low variance

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

+ Can produce lower-variance models than the model-discrepancy approach

Machine-learning error models

Machine-learning error models: formulation

$$\delta(\boldsymbol{\mu}) = \underbrace{f(\boldsymbol{\rho}(\boldsymbol{\mu}))}_{\text{deterministic}} + \underbrace{\epsilon(\boldsymbol{\rho}(\boldsymbol{\mu}))}_{\text{stochastic}}$$

- features: $\boldsymbol{\rho}(\boldsymbol{\mu}) \in \mathbb{R}^{N_{\boldsymbol{\rho}}}$
- regression function: $f(\boldsymbol{\rho}) = \mathrm{E}[\delta \mid \boldsymbol{\rho}]$
- noise: $\epsilon(\boldsymbol{\rho})$
- *Note*: model-discrepancy approach uses $\boldsymbol{\rho} = \boldsymbol{\mu}$

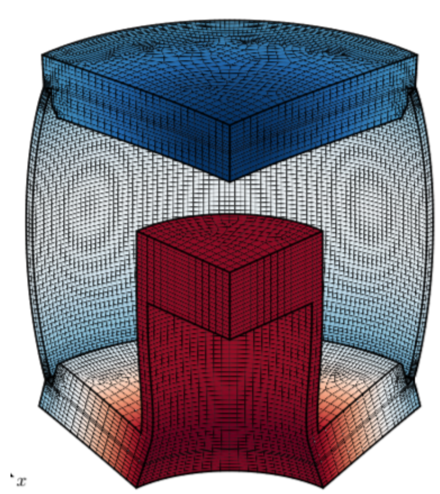
$$\tilde{\delta}(\boldsymbol{\mu}) = \underbrace{\tilde{f}(\boldsymbol{\rho}(\boldsymbol{\mu}))}_{\text{deterministic}} + \underbrace{\tilde{\epsilon}(\boldsymbol{\rho}(\boldsymbol{\mu}))}_{\text{stochastic}}$$

- regression-function model: $\tilde{f}(\approx f)$
- noise model: $\tilde{\epsilon}(\approx \epsilon)$

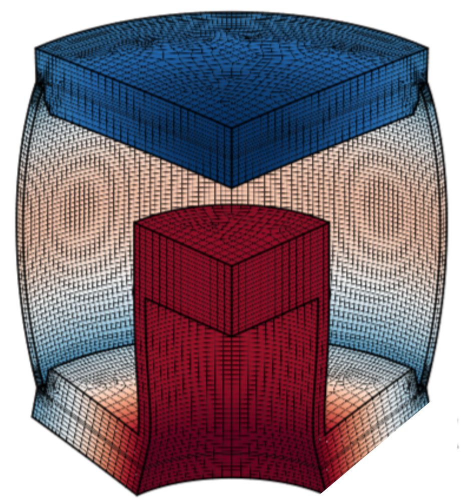
- Desired properties in error model $\tilde{\delta}$
  1. cheaply computable: features $\boldsymbol{\rho}(\boldsymbol{\mu})$ are inexpensive to compute
  2. low variance: noise model $\tilde{\epsilon}(\boldsymbol{\rho})$ has low variance
  3. generalizable: empirical distributions of δ and $\tilde{\delta}$ ‘close’ on test data

Training and machine learning

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

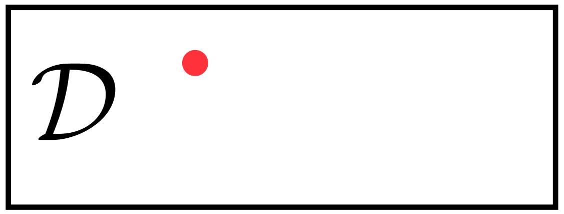


high-fidelity model



surrogate models

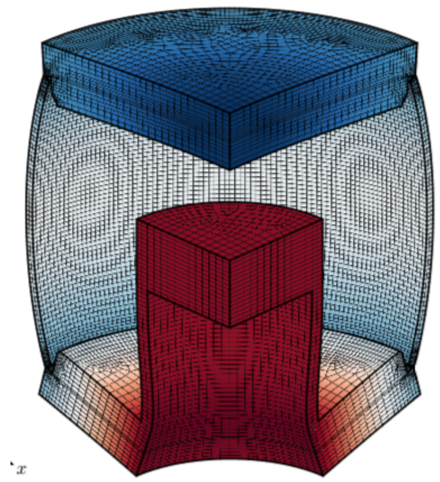
$$\delta = q_{\text{HFM}} - q_{\text{surr}}$$



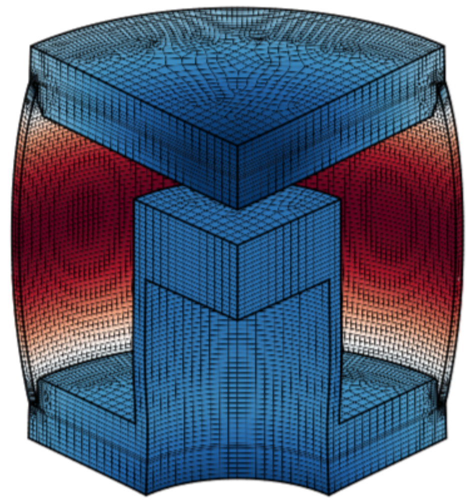
$\boldsymbol{\rho}$

Training and machine learning

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

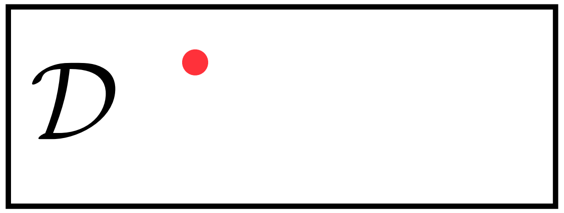


high-fidelity model



surrogate models

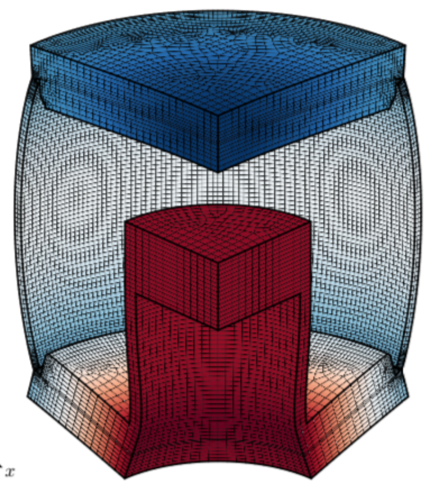
$$\delta = q_{\text{HFM}} - q_{\text{surr}}$$



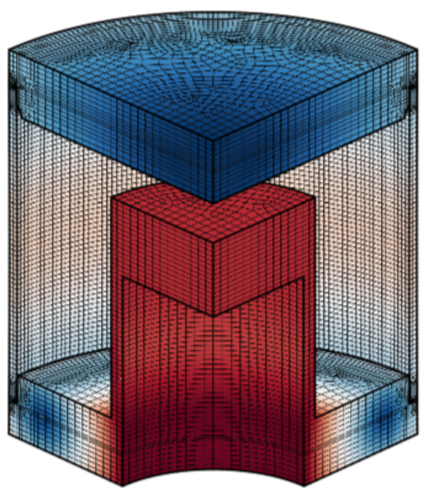
$\boldsymbol{\rho}$

Training and machine learning

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



high-fidelity model



surrogate models

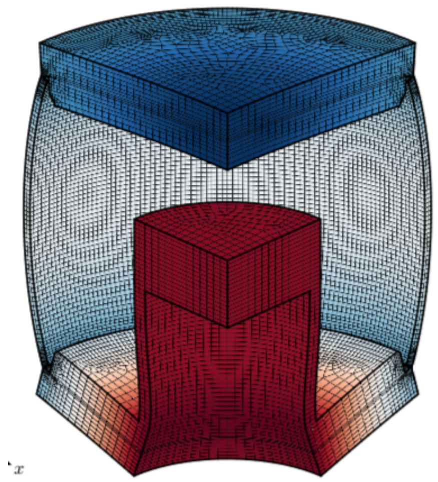
$\delta = q_{\text{HFM}} - q_{\text{surr}}$

$\boldsymbol{\rho}$

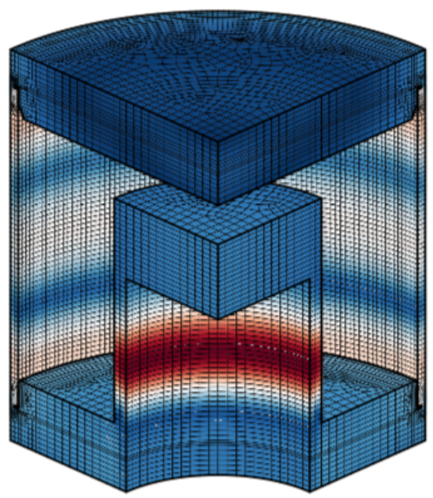
$\mathcal{D}$

Training and machine learning

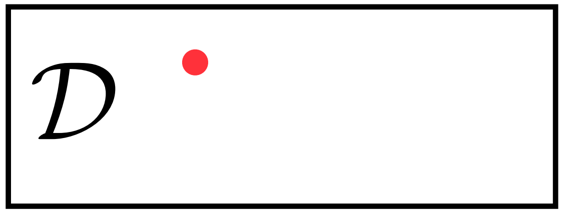
1. *Training:* Solve high-fidelity and multiple surrogates for $\boldsymbol{\mu} \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict surrogate-model error for $\boldsymbol{\mu} \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



high-fidelity model



surrogate models

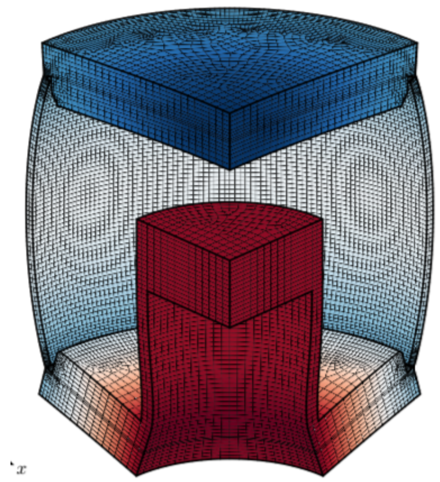


$\delta = q_{\text{HFM}} - q_{\text{surr}}$

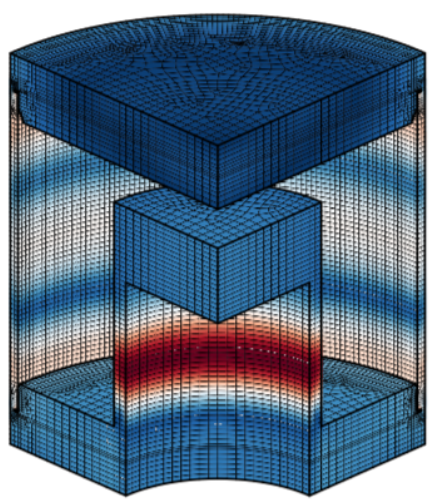
$\boldsymbol{\rho}$

Training and machine learning

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

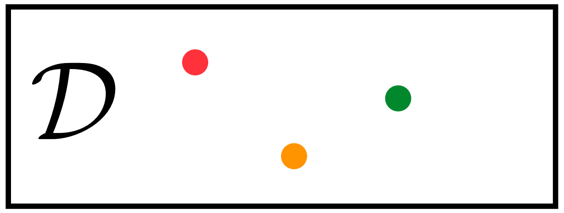


high-fidelity model



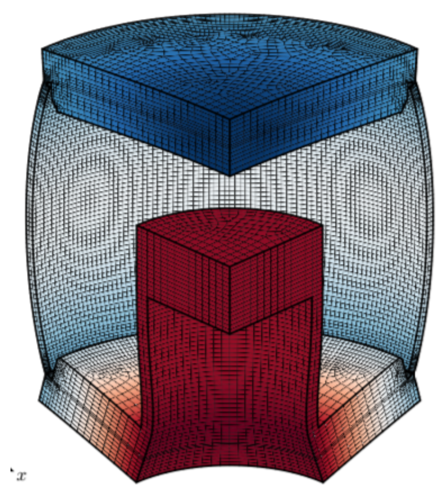
surrogate models

$\delta = q_{\text{HFM}} - q_{\text{surr}}$ $\boldsymbol{\rho}$

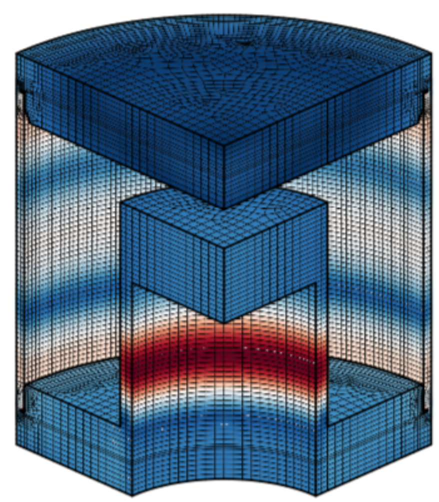


Training and machine learning

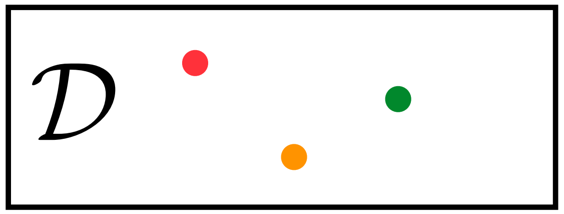
1. *Training:* Solve high-fidelity and multiple surrogates for $\mu \in \mathcal{D}_{\text{training}}$
2. *Machine learning:* Construct regression model
3. *Reduction:* predict surrogate-model error for $\mu \in \mathcal{D}_{\text{query}} \setminus \mathcal{D}_{\text{training}}$



high-fidelity model



surrogate models



$\delta = q_{\text{HFM}} - q_{\text{surr}}$

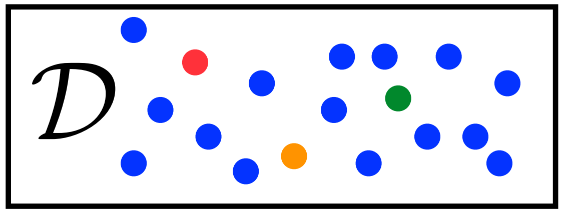
$\boldsymbol{\rho}$

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

Reduction

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

inputs $\boldsymbol{\mu}$ → *surrogate model* → *outputs* q_{surr}



↓

features $\boldsymbol{\rho}$

↓

regression model
$$\tilde{\delta}(\boldsymbol{\mu}) = \tilde{f}(\boldsymbol{\rho}(\boldsymbol{\mu})) + \tilde{\epsilon}(\boldsymbol{\rho}(\boldsymbol{\mu}))$$
→ *machine learning error model* $\tilde{\delta}$

$$\underbrace{\tilde{q}_{\text{HFM}}(\boldsymbol{\mu})}_{\text{stochastic}} = \underbrace{q_{\text{surr}}(\boldsymbol{\mu})}_{\text{deterministic}} + \underbrace{\tilde{\delta}(\boldsymbol{\mu})}_{\text{stochastic}}$$

Error-model construction

$$\tilde{\delta}(\boldsymbol{\mu}) = \tilde{f}(\boldsymbol{\rho}(\boldsymbol{\mu})) + \tilde{\epsilon}(\boldsymbol{\rho}(\boldsymbol{\mu}))$$

Feature engineering: select features $\boldsymbol{\rho}$ to trade off:

1. Number of features
   - *Large number*: costly, low variance, high-capacity regression
   - *Small number*: cheap, high variance, low-capacity regression
2. Quality of features
   - *High quality*: expensive, low variance
   - *Low quality*: cheap, high variance

Regression model: construct regression model $\tilde{f}$ to trade off:

- *High capacity*: low variance, more data to generalize
- *Low capacity*: high variance, less data to generalize

Method 1: Dual-weighted residual and Gaussian process regression
[Drohmann, C., 2015; C., Uy, Lu, Morzfeld, 2018]

Method 2: Large number of features and high-dimensional regression
[Trehan, C., Durlofsky, 2017; Freno, C., 2018]

Error-model construction

$$\tilde{\delta}(\boldsymbol{\mu}) = \tilde{f}(\boldsymbol{\rho}(\boldsymbol{\mu})) + \tilde{\epsilon}(\boldsymbol{\rho}(\boldsymbol{\mu}))$$

Feature engineering: select features $\boldsymbol{\rho}$ to trade off:

1. Number of features
   - *Large number*: costly, low variance, high-capacity regression
   - *Small number*: cheap, high variance, low-capacity regression
2. Quality of features
   - *High quality*: expensive, low variance
   - *Low quality*: cheap, high variance

Regression model: construct regression model $\tilde{f}$ to trade off:

- *High capacity*: low variance, more data to generalize
- *Low capacity*: high variance, less data to generalize

Method 1: Dual-weighted residual and Gaussian process regression
[Drohmann, C., 2015; C., Uy, Lu, Morzfeld, 2018]

Method 2: Large number of features and high-dimensional regression
[Trehan, C., Durlofsky, 2017; Freno, C., 2018]

Feature: dual-weighted residual [Drohmann, C., 2015]

$$\boxed{q(\mathbf{x}) - q(\tilde{\mathbf{x}}) = \mathbf{y}^T \mathbf{r}(\tilde{\mathbf{x}}) + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)}$$

$$\boxed{\frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T \mathbf{y} = -\frac{\partial q}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T}$$

- Want to avoid HFM-scale solves, so approximate dual as

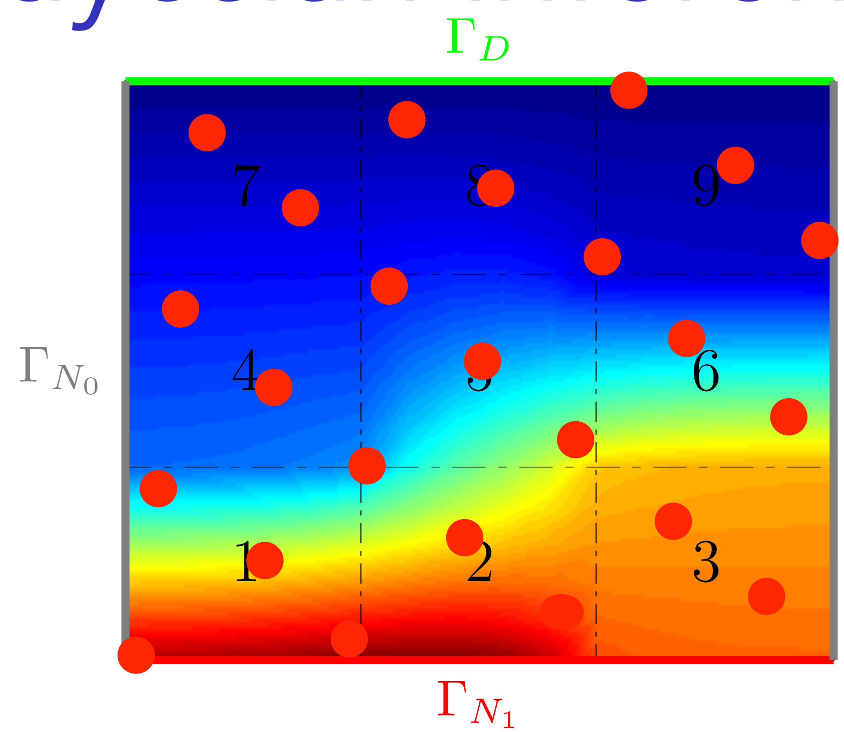
$$\mathbf{y} \approx \tilde{\mathbf{y}} = \mathbf{\Phi}_{\mathbf{y}} \hat{\mathbf{y}}$$

and construct a ROM for the dual

$$\mathbf{\Phi}_{\mathbf{y}}{}^T \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T \mathbf{\Phi}_{\mathbf{y}} \hat{\mathbf{y}} = -\mathbf{\Phi}_{\mathbf{y}}{}^T \frac{\partial q}{\partial \mathbf{x}}(\tilde{\mathbf{x}})^T$$

- **One feature**: $q(\mathbf{x}) - q(\tilde{\mathbf{x}}) \approx \boxed{\hat{\mathbf{y}}^T \mathbf{\Phi}_{\mathbf{y}}{}^T \mathbf{r}(\tilde{\mathbf{x}})}$
  - can control feature quality via dimension of $\mathbf{\Phi}_{\mathbf{y}}$
- **Regression model**: Gaussian process [Rasmussen, Williams, 2006]

Application: Bayesian inference



$$\triangle c(x;\boldsymbol{\mu})u(x;\boldsymbol{\mu}) = 0 \text{ in } \Omega \qquad \boldsymbol{x}(\boldsymbol{\mu}) = 0 \text{ on } \Gamma_D$$

$$\nabla c(\boldsymbol{\mu})\boldsymbol{x}(\boldsymbol{\mu}) \cdot n = 0 \text{ on } \Gamma_{N_0} \qquad \nabla c(\boldsymbol{\mu})\boldsymbol{x}(\boldsymbol{\mu}) \cdot n = 1 \text{ on } \Gamma_{N_1}$$

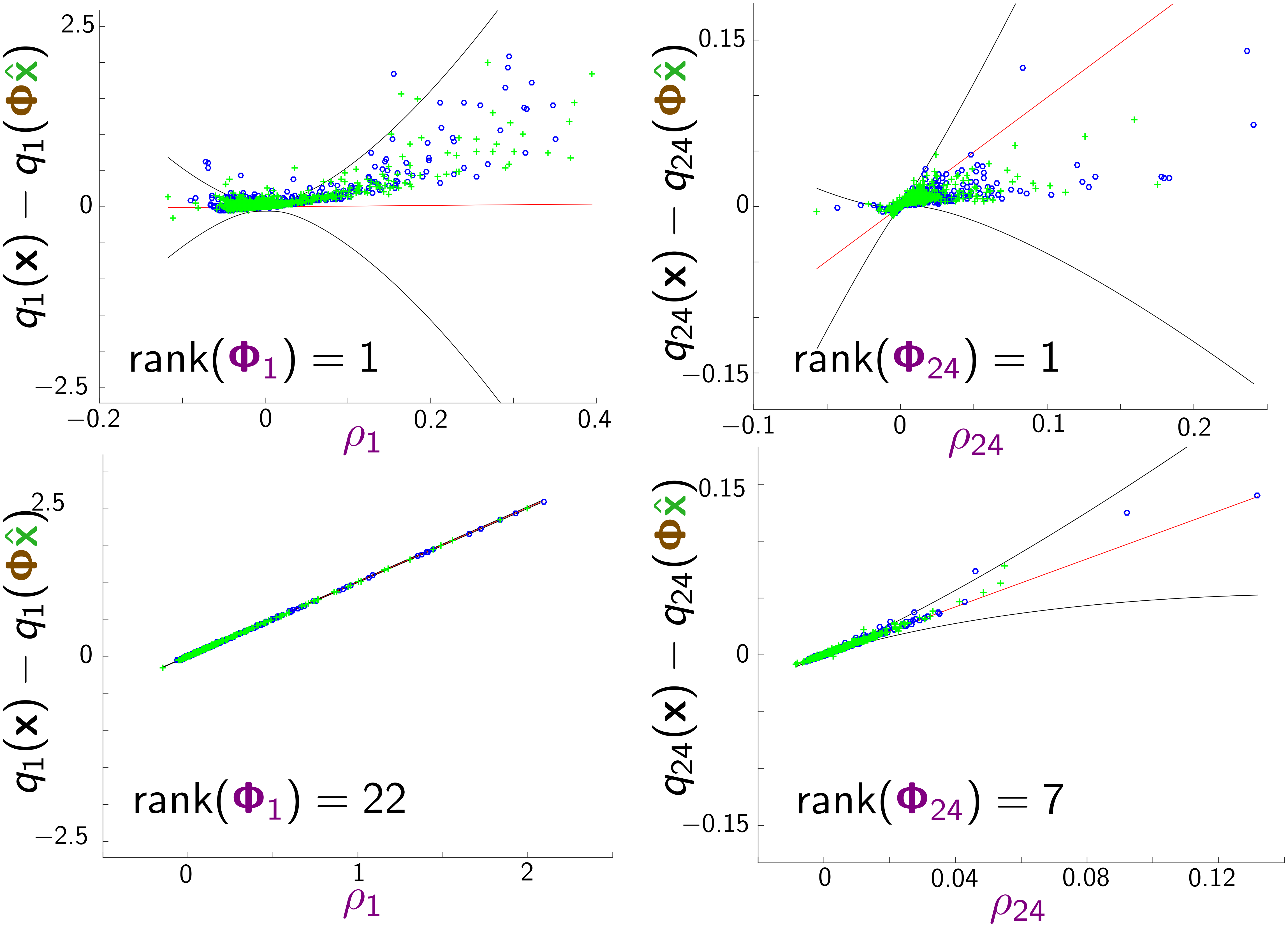
- Inputs $\boldsymbol{\mu} \in [0.1, 10]^9$ define diffusivity in c in subdomains
- Outputs $\mathbf{q}$ are 24 measured temperatures
- ROM constructed via RB-Greedy [Patera and Rozza, 2006]
- $\pi_{\text{prior}}(\boldsymbol{\mu})$: Gaussian with variance 0.1
- $\varepsilon \sim \mathcal{N}(0, 1 \times 10^{-3})$
- Posterior sampling: 1×10^5 samples w/ implicit sampling [Tu et al., 2013]

Machine learning error models

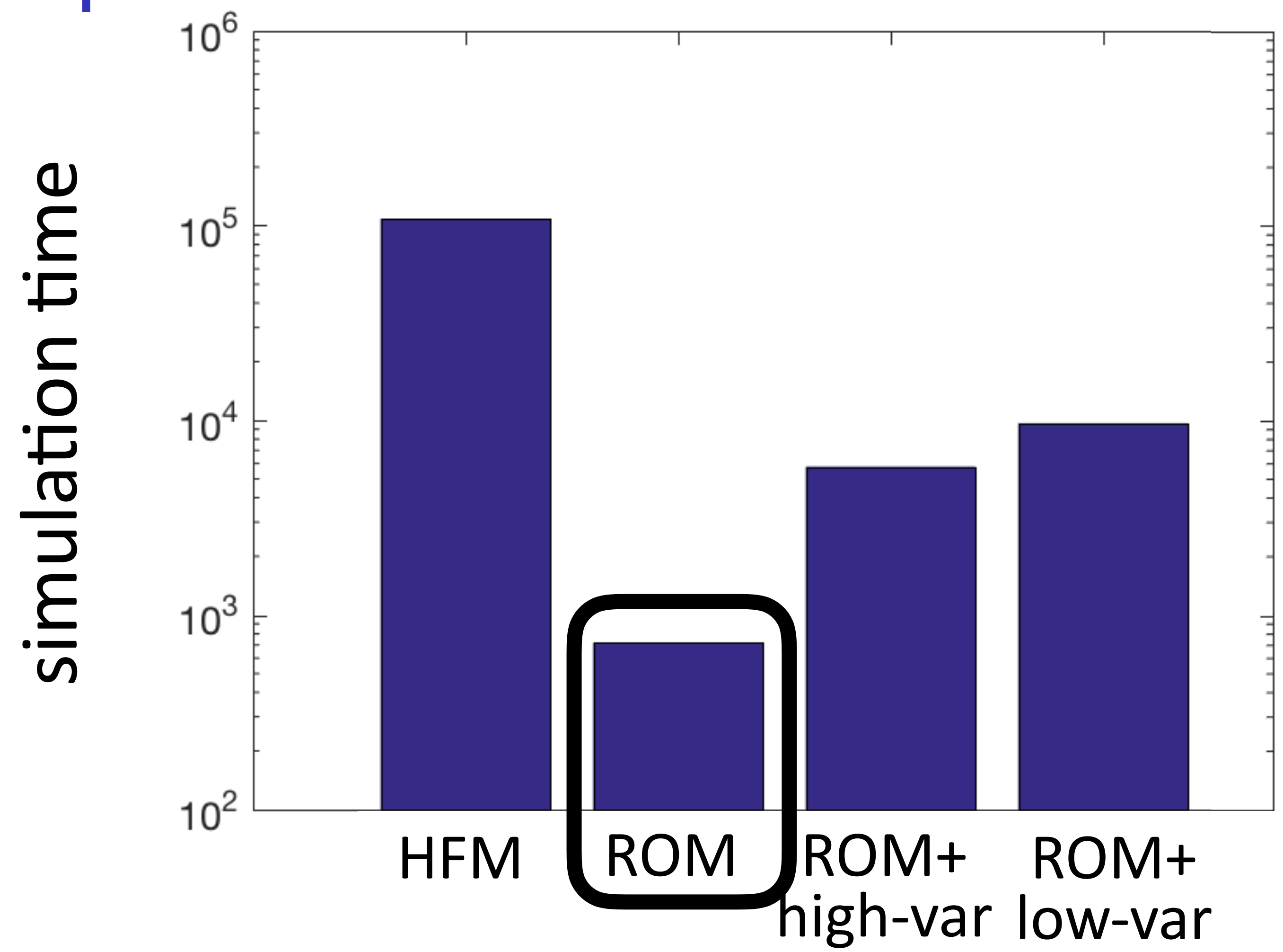
$$\tilde{\delta}_i(\boldsymbol{\mu}) \sim \mathcal{N}(\beta \rho_i(\boldsymbol{\mu}), \alpha_1 + \alpha_2 |\rho_i(\boldsymbol{\mu})|^{\alpha_3})$$

low quality
high variance
cheap

high quality
low variance
costly

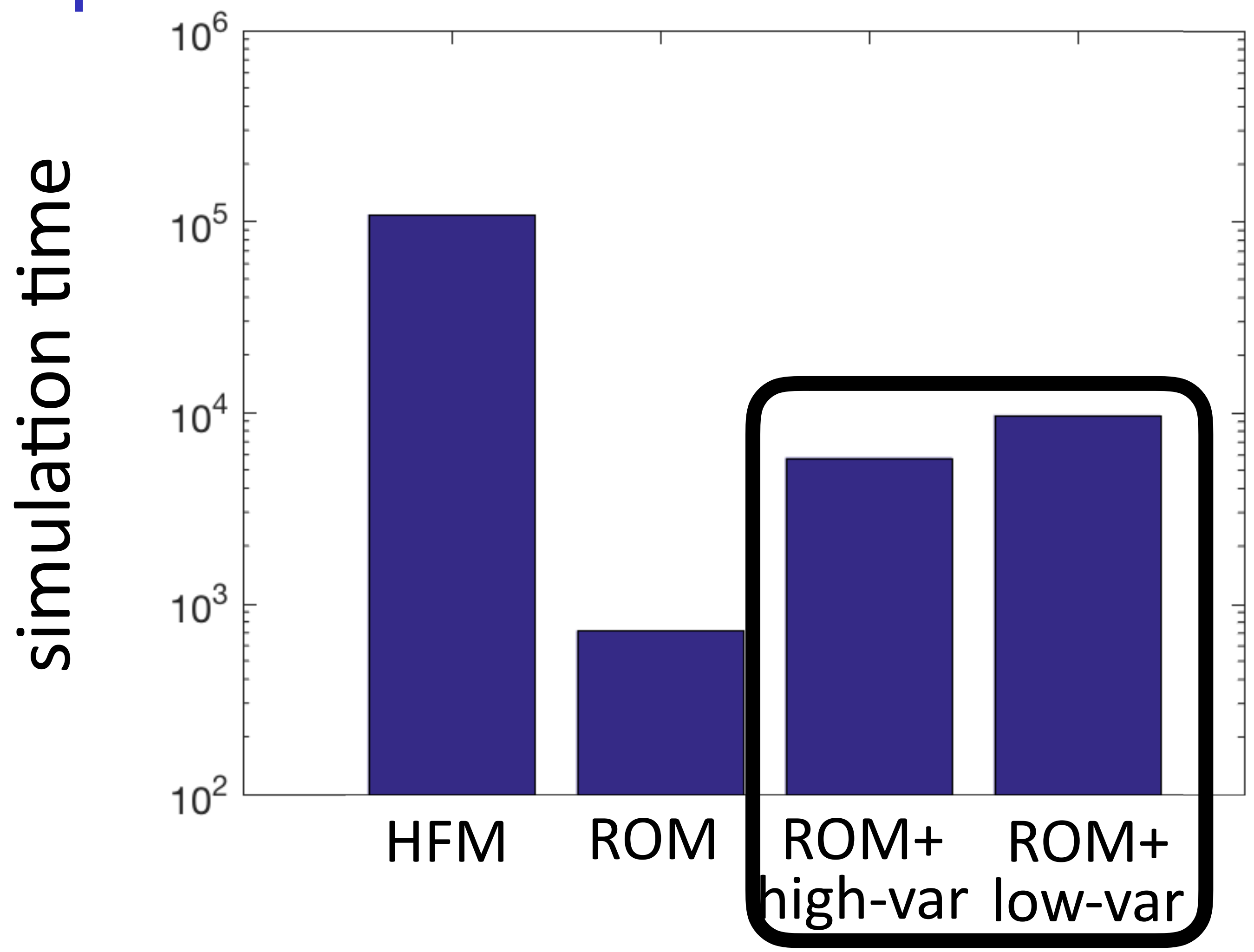


Wall-time performance



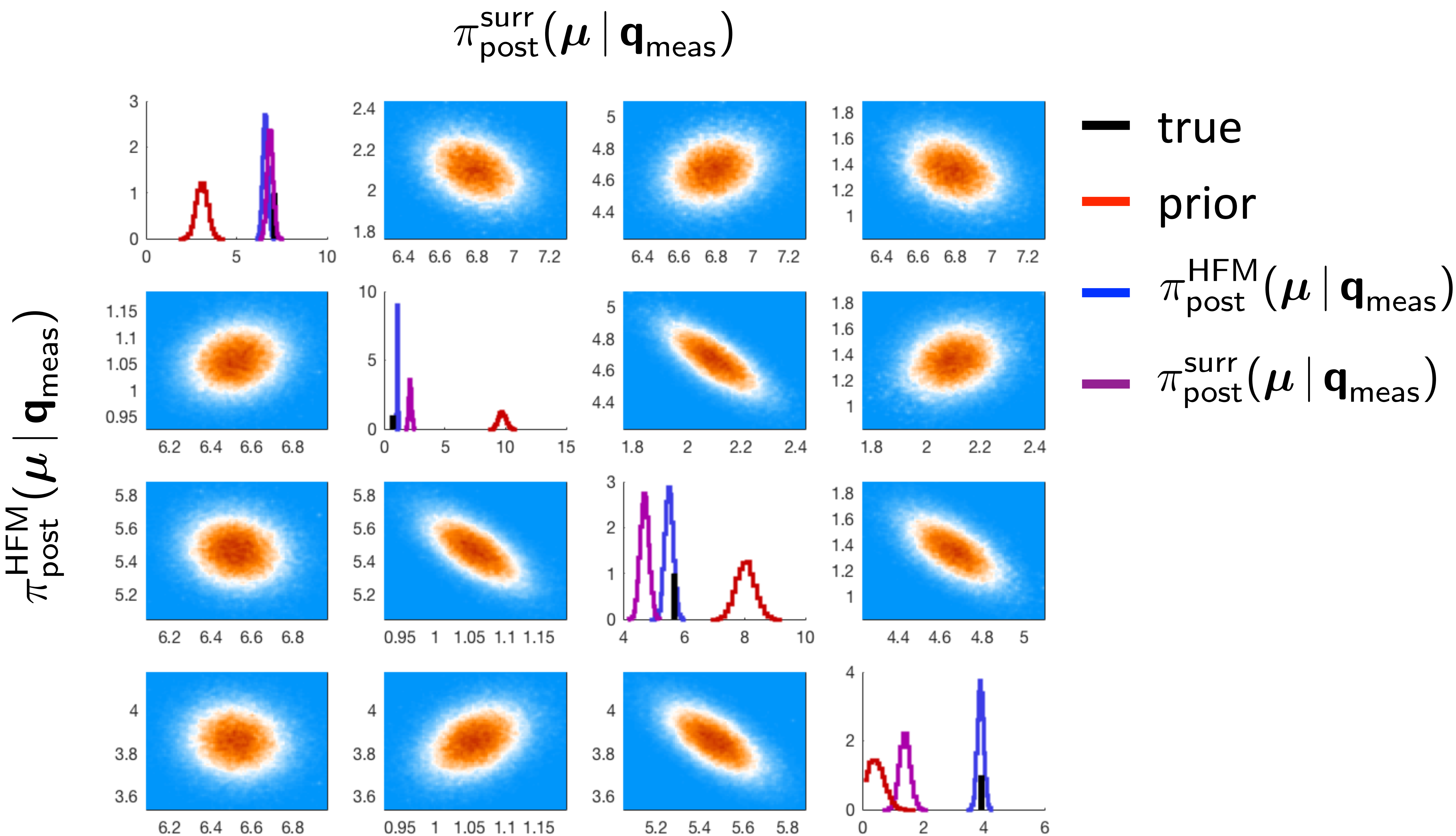
- ROM:
  + cheapest
  - inconsistent formulation

Wall-time performance



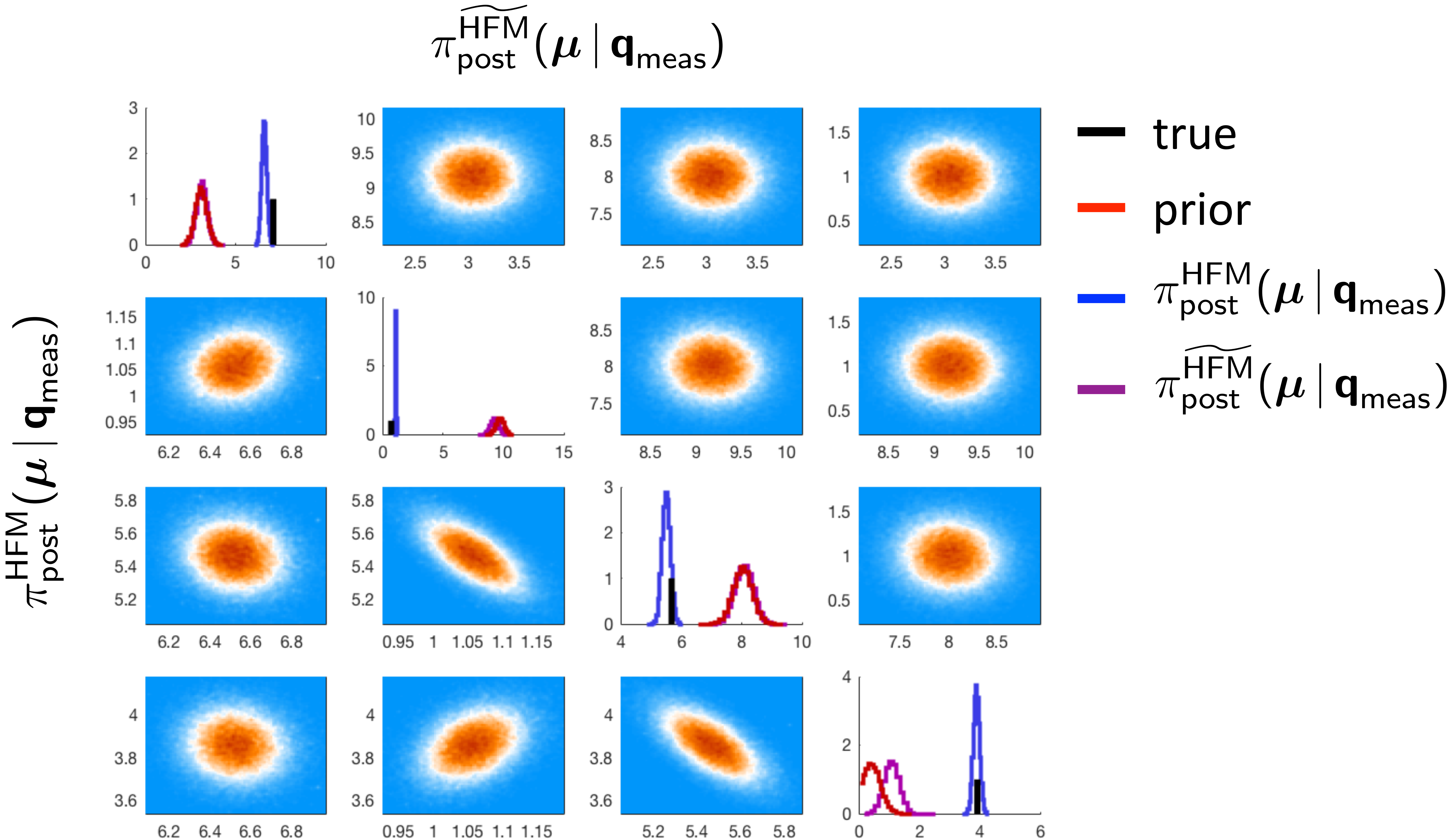
- ROM:
  - \+ cheapest
  - \- inconsistent formulation
- ROM + error models:
  - \+ cheaper than HFM
  - \- more expensive than ROM
  - \+ consistent formulation

Posteriors: ROM



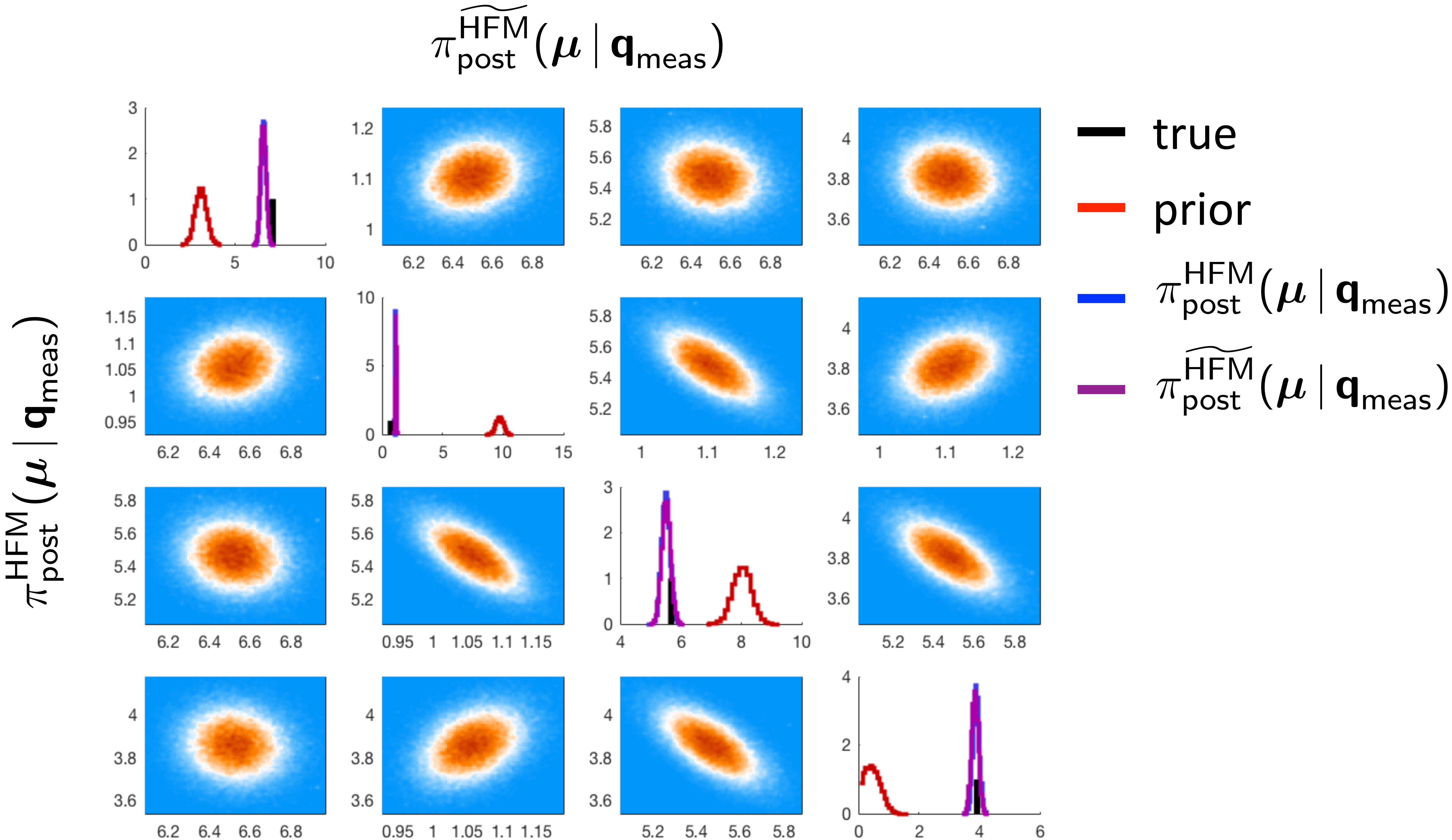
+ HFM posterior: close to **true parameters**
- ROM posterior: far from prior and **true parameters**

Posteriors: ROM + high-variance error model



+ ROM + high-variance error model posterior: close to prior

Posteriors: ROM + low-variance error model



+ ROM + low-variance error model posterior: close to HFM posterior

Error-model construction

$$\tilde{\delta}(\boldsymbol{\mu}) = \tilde{f}(\boldsymbol{\rho}(\boldsymbol{\mu})) + \tilde{\epsilon}(\boldsymbol{\rho}(\boldsymbol{\mu}))$$

Feature engineering: select features $\boldsymbol{\rho}$ to trade off:

1. Number of features
   - *Large number*: costly, low variance, high-capacity regression
   - *Small number*: cheap, high variance, low-capacity regression
2. Quality of features
   - *High quality*: expensive, low variance
   - *Low quality*: cheap, high variance

Regression model: construct regression model $\tilde{f}$ to trade off:

- *High capacity*: low variance, more data to generalize
- *Low capacity*: high variance, less data to generalize

Method 1: Dual-weighted residual and Gaussian process regression
[Drohmann, C., 2015; C., Uy, Lu, Morzfeld, 2018]

Method 2: Large number of features and high-dimensional regression
[Trehan, C., Durlofsky, 2017; Freno, C., 2018]

Feature engineering [Freno, C., 2018]

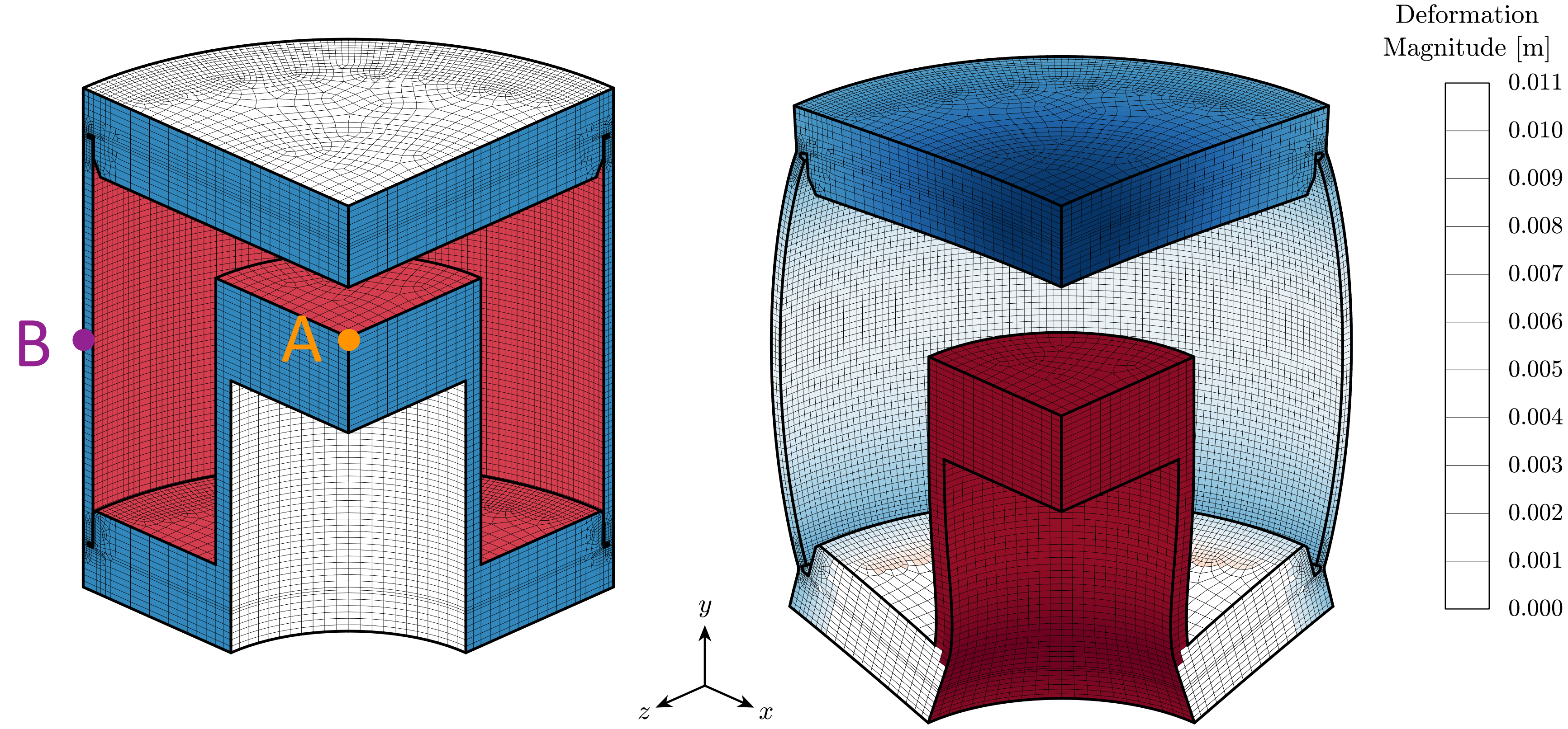
Idea: *Use traditional error quantification as inspiration for features*

1. **Error indicators:**
   - residual norm: $\|\mathbf{r}(\tilde{\mathbf{x}};\boldsymbol{\mu})\|_2$
   - dual-weighted residual: $q(\mathbf{x}) - q(\tilde{\mathbf{x}}) = \mathbf{y}^T\mathbf{r}(\tilde{\mathbf{x}}) + O(\|\mathbf{x}-\tilde{\mathbf{x}}\|^2)$
2. **Rigorous *a posteriori* error bound:** $|q(\mathbf{x}) - q(\tilde{\mathbf{x}})| \leq \dfrac{\beta}{\alpha}\|\mathbf{r}(\tilde{\mathbf{x}};\boldsymbol{\mu})\|_2$
3. **Model discrepancy:** $\tilde{\delta}(\boldsymbol{\mu}) \sim \mathcal{N}(\mu(\boldsymbol{\mu});\sigma^2(\boldsymbol{\mu}))$

Proposed features:

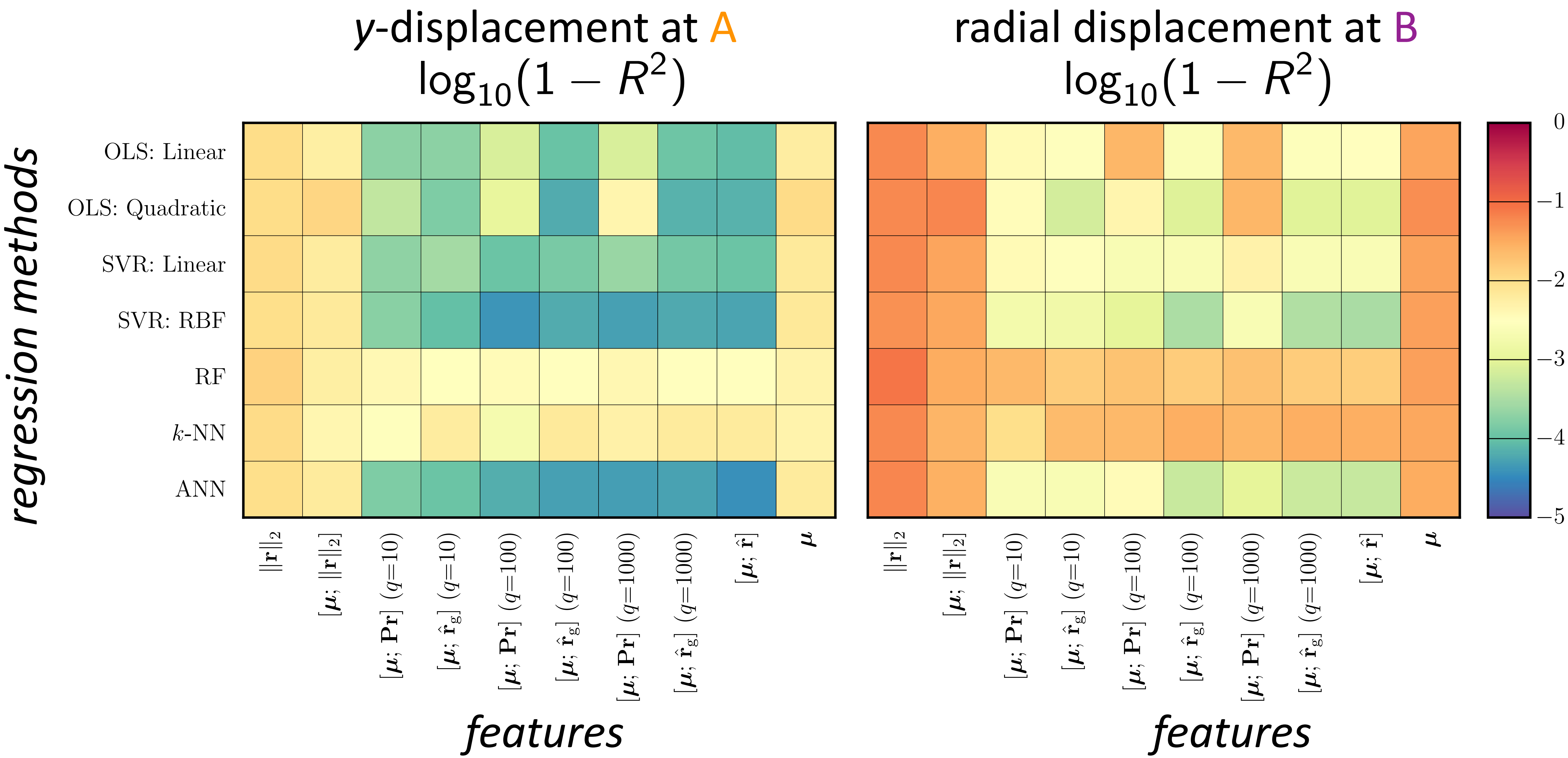
- parameters $\boldsymbol{\mu}$
  - low quality, cheap
  - used by model discrepancy
- residual norm $\|\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})\|_2$
  - small number, low quality, costly
- residual $\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})$
  - large number, low quality, costly
- residual samples $\mathbf{Pr}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})$
  - \+ moderate number, cheap
  - \- low quality
- residual PCA $\hat{\mathbf{r}} := \boldsymbol{\Phi}_{\mathbf{r}}^T\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})$
  - \+ moderate number, high-quality
  - \- costly
- gappy PCA $\hat{\mathbf{r}}_g := (\mathbf{P}\boldsymbol{\Phi}_{\mathbf{r}})^+\mathbf{Pr}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})$
  - \+ moderate number, high-quality
  - \+ cheap

Application: Predictive capability assessment project

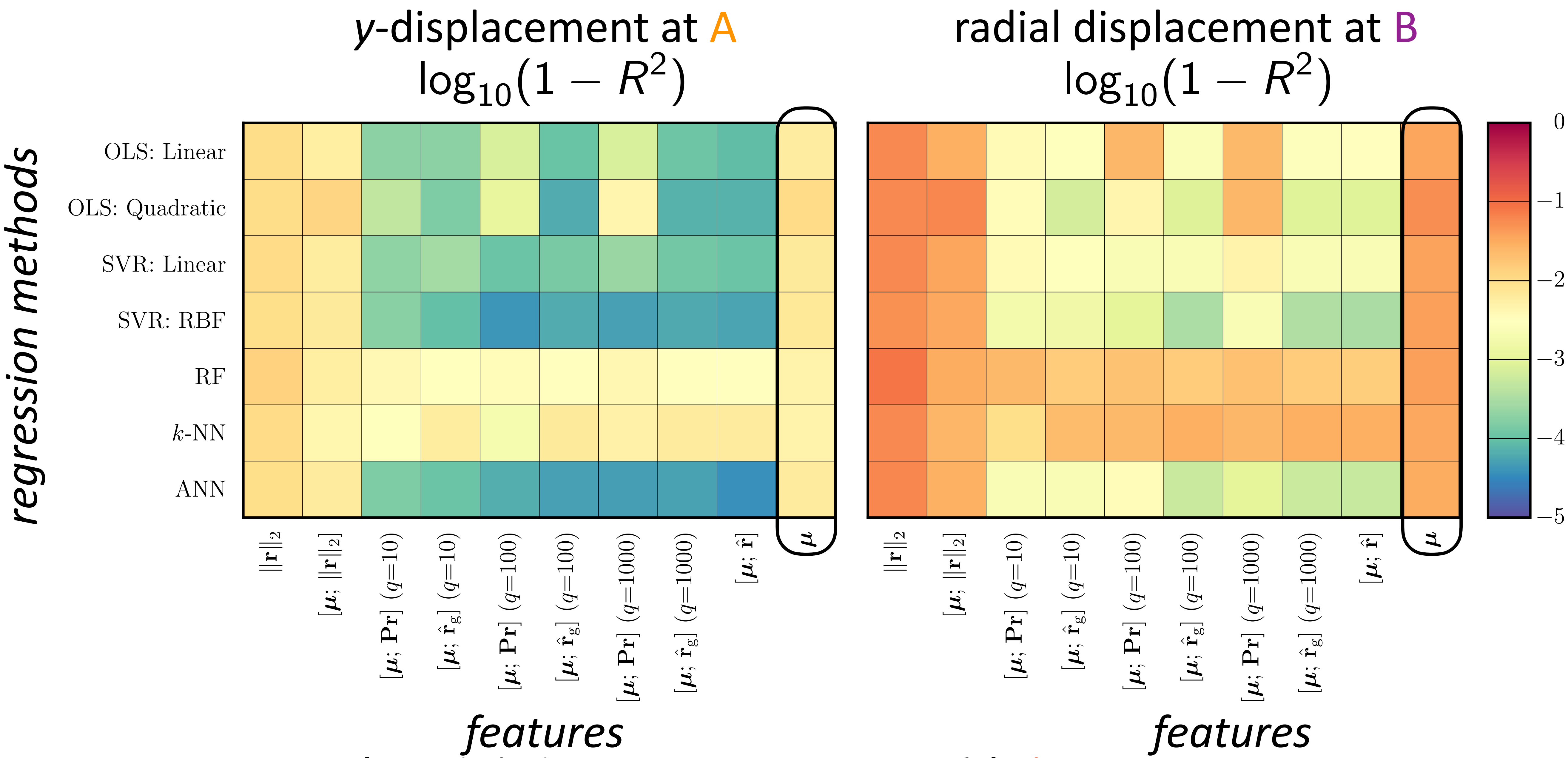


- *high-fidelity model dimension:* 2.8×10^5
- *reduced-order model dimensions:* $1, \ldots, 5$
- *inputs* $\boldsymbol{\mu}$*:* elastic modulus, Poisson ratio, applied pressure
- *quantities of interest: y*-displacement at A, radial displacement at B
- *training data*: 150 training examples, 150 testing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

Application: Predictive capability assessment project

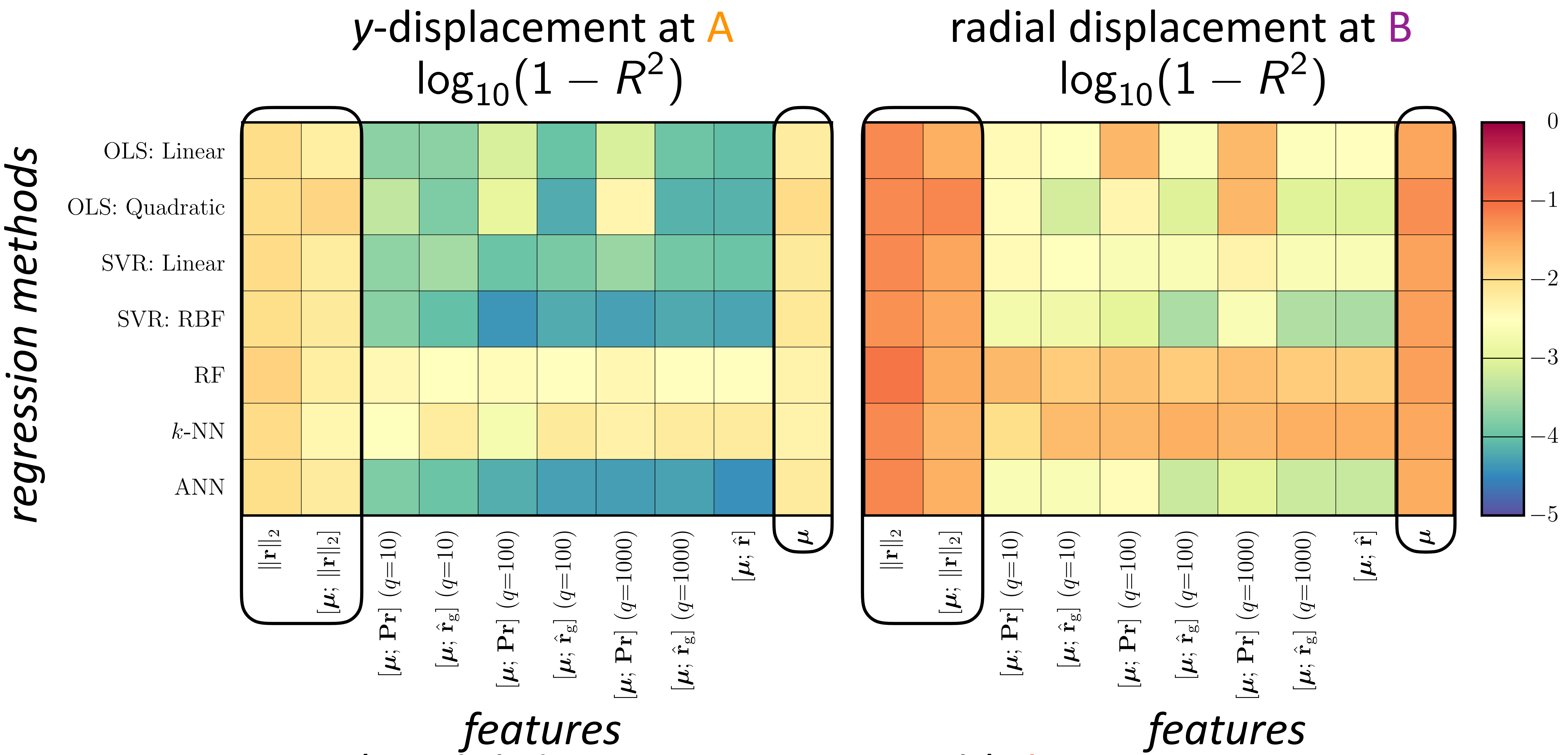


Application: Predictive capability assessment project



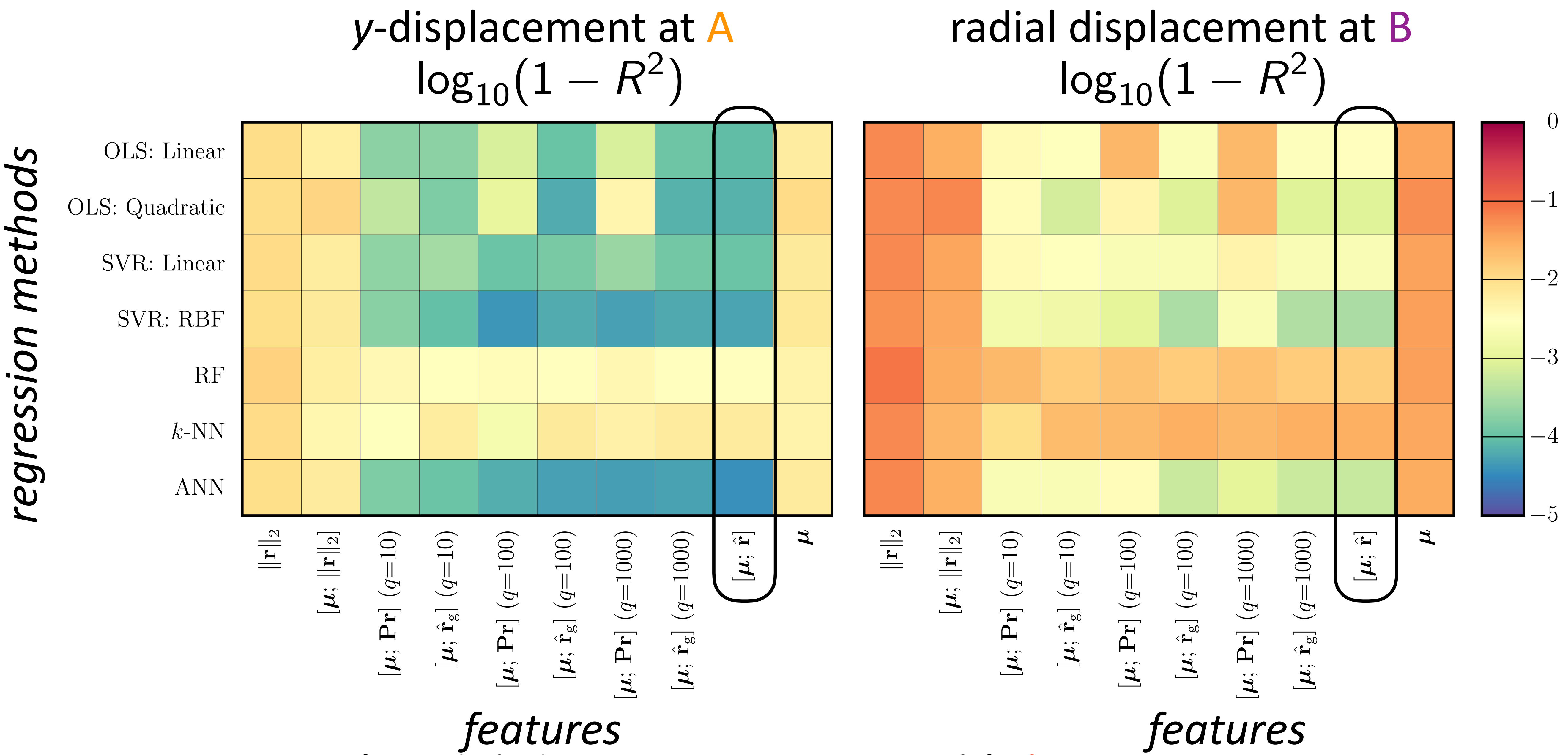
- parameters (model-discrepancy approach): large variance

Application: Predictive capability assessment project



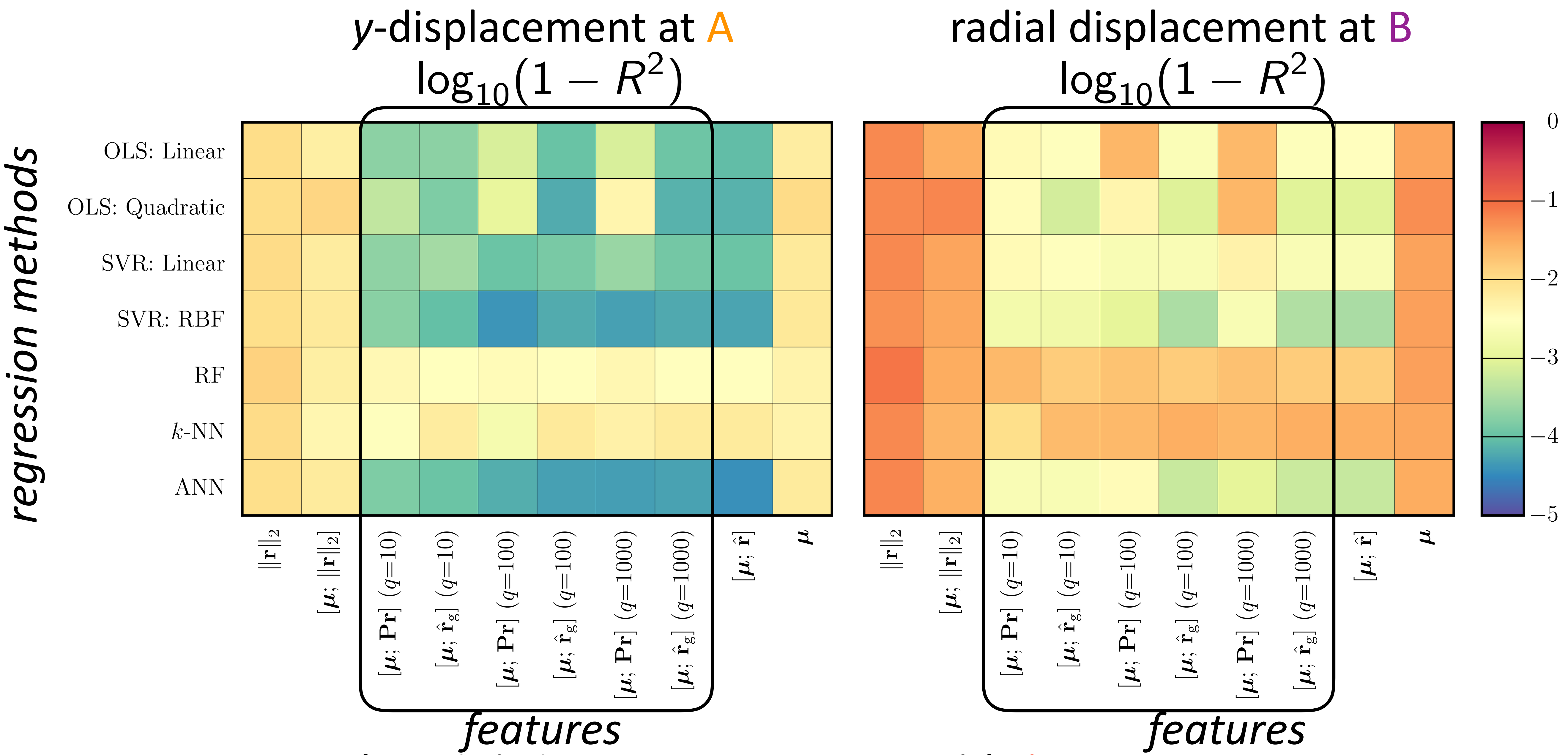
- parameters (model-discrepancy approach): large variance
- small number of low-quality features: large variance

Application: Predictive capability assessment project



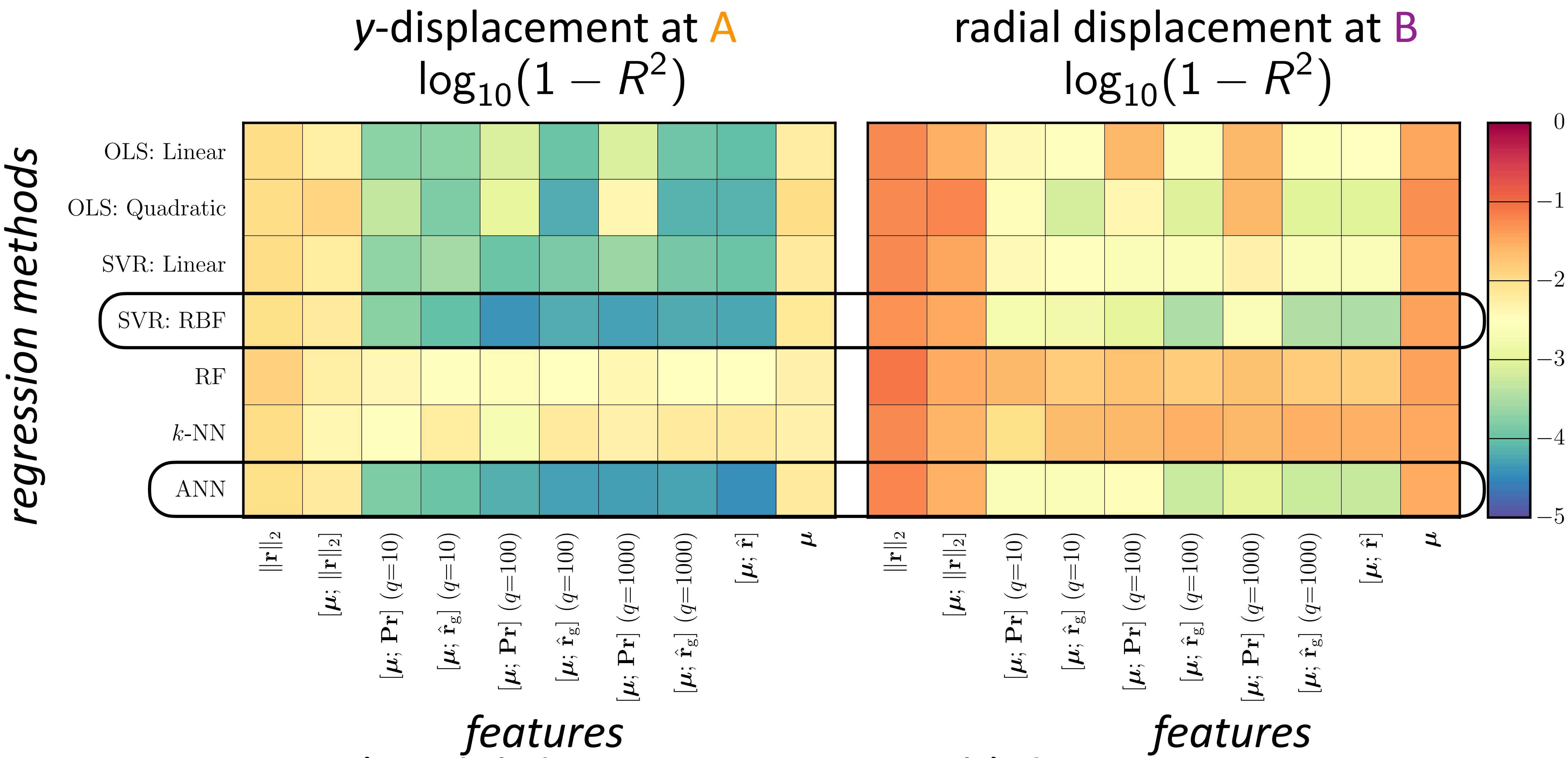
- parameters (model-discrepancy approach): large variance
- small number of low-quality features: large variance
- PCA of the residual: lowest variance overall but costly

Application: Predictive capability assessment project



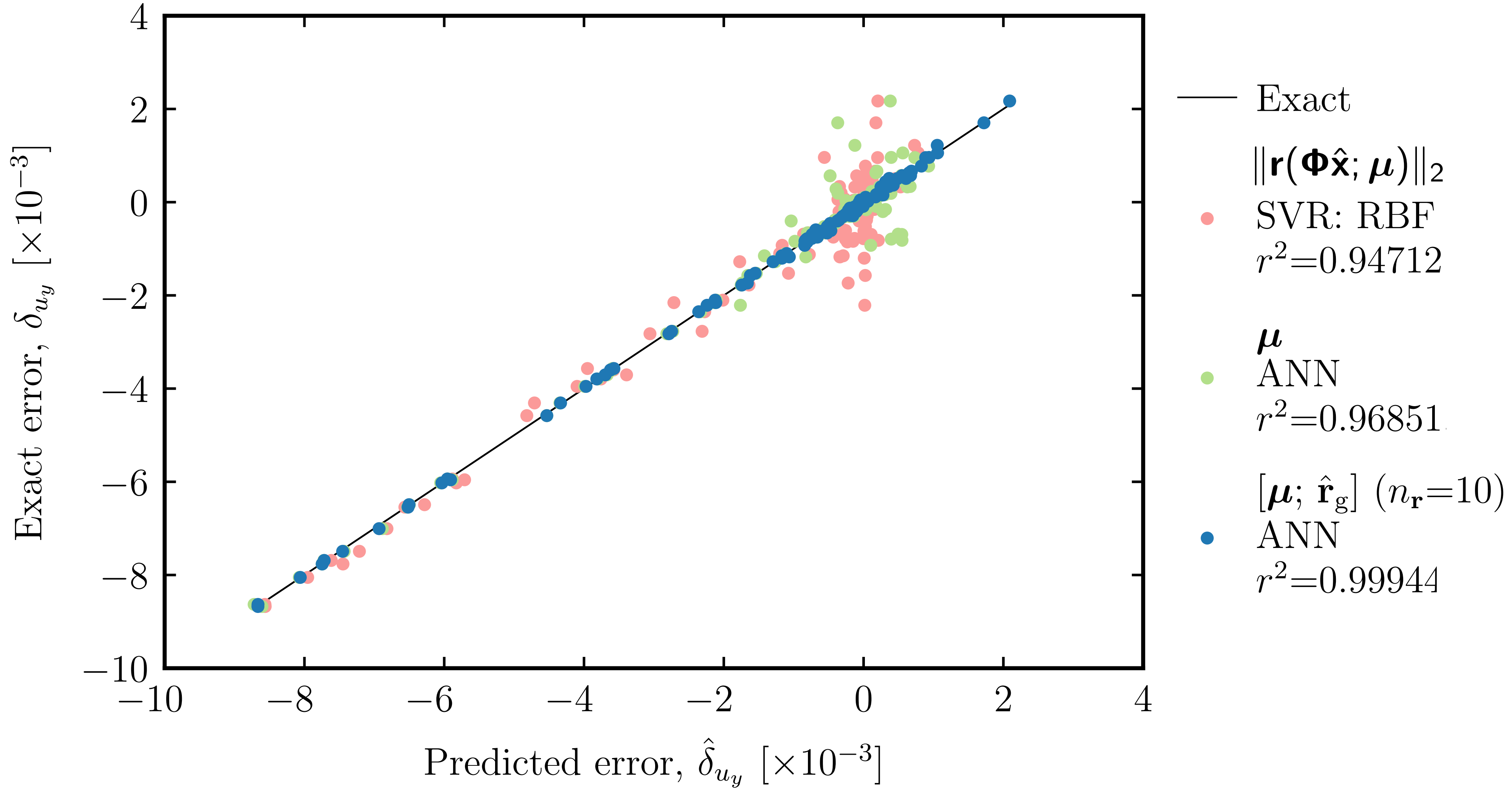
- parameters (model-discrepancy approach): large variance
- small number of low-quality features: large variance
- PCA of the residual: lowest variance overall but costly
- \+ gappy PCA of the residual: nearly as low variance, but much cheaper

Application: Predictive capability assessment project



- parameters (model-discrepancy approach): large variance
- small number of low-quality features: large variance
- PCA of the residual: lowest variance overall but costly
- \+ gappy PCA of the residual: nearly as low variance, but much cheaper
- \+ neural networks and SVR: RBF yield lowest-variance models

Application: Predictive capability assessment project



- Traditional features $\boldsymbol{\mu}$ and $\|\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})\|_2$:
  - high noise variance
  - expensive for $\|\mathbf{r}(\boldsymbol{\Phi}\hat{\mathbf{x}};\boldsymbol{\mu})\|_2$: compute entire residual
- Proposed features $[\boldsymbol{\mu};\hat{\mathbf{r}}_g]$:
  - \+ low noise variance
  - \+ extremely cheap: only compute 10 elements of the residual

Summary

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

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

Questions?

LSPG reduced-order model:

- **C, Barone, and Antil. "Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction," Journal of Computational Physics, Vol. 330, p. 693–734 (2017).**
- C, Farhat, Cortial, and Amsallem. "The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows," Journal of Computational Physics, Vol. 242, p. 623–647 (2013).
- C, Bou-Mosleh, and Farhat. "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," International Journal for Numerical Methods in Engineering, Vol. 86, No. 2, p. 155–181 (2011).

Machine-learning error models:

- **Freno, C. "Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations," arXiv e-Print, 1808.02097 (2018).**
- Trehan, C, and Durlofsky. "Error modeling for surrogates of dynamical systems using machine learning," International Journal for Numerical Methods in Engineering, Vol. 112, No. 12, p. 1801–1827 (2017).
- Drohmann and C. "The ROMES method for statistical modeling of reduced-order-model error," SIAM/ASA Journal on Uncertainty Quantification, Vol. 3, No. 1, p.116–145 (2015).