Gaussian-process-augmented projection-based model order reduction

Carlos González Hernández, Radek Tezaur, Charbel Farhat

APS DFD 2024

Stanford University

Motivation

- Many important engineering applications rely on many queries of high-dimensional computational models
  - Model predictive control
  - Uncertainty quantification
  - Design analysis & optimization
- Surrogate models are often required for these to be computationally tractable
  - Seek a more parsimonious description of the high-dimensional model through dimensionality reduction
- Dimensionality reduction approaches
  - External representations: linear regression, gaussian process regression, artificial neural network regression
  - **Internal representations**: operator-inference model, physics-informed neural networks, projection-based reduced-order models

Projection-based model order reduction (PMOR)

- Principled, **physics-based method** for **machine learning with model(s) and data**
- Semi-discrete or discrete, parametric, linear or nonlinear computational model

  $\mathbf{R}(\boldsymbol{q};\boldsymbol{\mu}) = \mathbf{0}$, $\boldsymbol{q} \in \mathbb{R}^N$, $\mathbf{R} \in \mathbb{R}^N$, $\boldsymbol{\mu} \in \boldsymbol{\mathcal{D}} \subset \mathbb{R}^{N_\mu}$, N very large, N_μ moderately large
- Hypothesis: best approximation of the solution in a lower-dimensional space

$$\boldsymbol{q}(\boldsymbol{\mu}) \approx \overline{\boldsymbol{q}}(\boldsymbol{\mu}) = \boldsymbol{f}(\boldsymbol{q_r}(\boldsymbol{\mu})) \qquad \boldsymbol{q_r} \in \mathbb{R}^n, \quad n \ll N$$

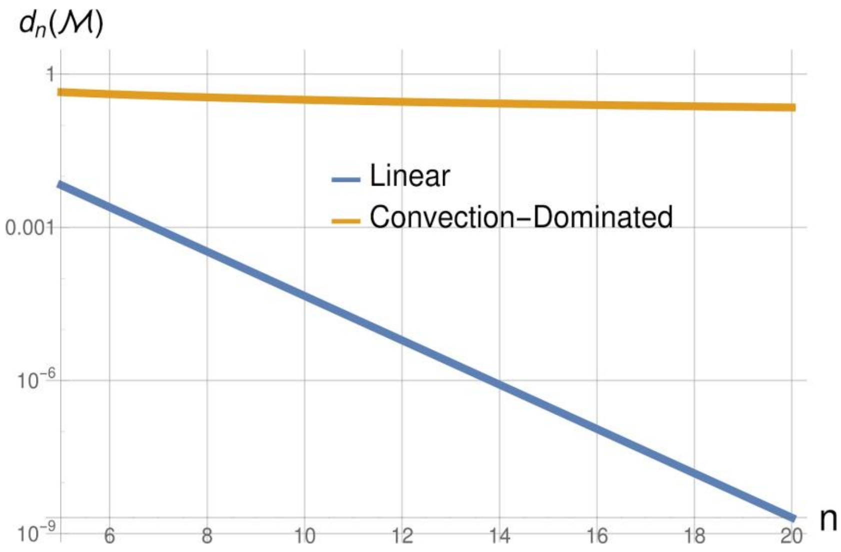
e.g. (traditional PROM) $\boldsymbol{f}\big(\boldsymbol{q_r}(\boldsymbol{\mu})\big) = \boldsymbol{q_{ref}} + \boldsymbol{V}\boldsymbol{q_r} \qquad \boldsymbol{V} \in \mathbb{R}^{N\times n}$

- Representation: reduced-order basis (ROB) $\boldsymbol{V}$
- Data-driven learning: ROB $\boldsymbol{V}$ is learned from solutions snapshots and their compression
- Minimization of a loss function

$$\boldsymbol{q_r} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\| \boldsymbol{R}\Big(\boldsymbol{f}\big(\boldsymbol{x}(\boldsymbol{\mu})\big);\boldsymbol{\mu}\Big) \right\|_2$$

Challenge in PMOR – breaking the Kolmogorov n-width

- PMOR based on the *affine subspace* approximation $\tilde{\boldsymbol{u}} = \boldsymbol{u_{ref}} + \boldsymbol{V}\boldsymbol{q}$, where $\boldsymbol{V} \in \mathbb{R}^{N\times n}$, $\boldsymbol{q} \in \mathbb{R}^{n}$, $\boldsymbol{u_{ref}} \in \mathbb{R}^{N}$
- For convection-dominated PDEs, the convergence of a subspace approximation is limited by the *slow decay of the Kolmogorov n-width* $d_n(\mathcal{M})$
- Recent strategies for mitigating this issue share the abandonment of the traditional affine approximation in favour of a nonlinear one (*nonlinear PMOR*): ***piecewise linear*** approximation; ***autoencoder-based*** approximation; ***nonlinear parametrization*** of affine approximation; ***quadratic*** approximation manifold

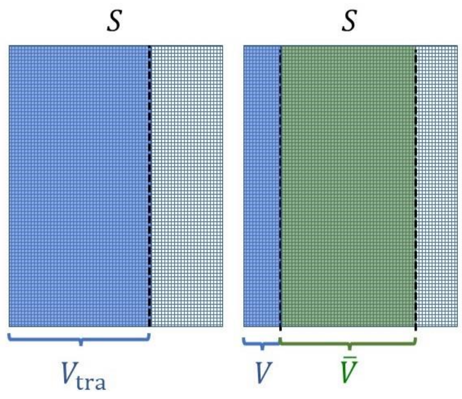


For most linear problems, $d_n(\mathcal{M})$ exhibits exponential decay; for convection-dominated flow problems, it exhibits a decay of $\mathcal{O}(n^{-1/2})$

Arbitrarily nonlinear approx. for mitigating Kolmogorov barrier

- Nonlinear approximation manifold generated by a ROB and an ANN: $\boldsymbol{u}(t;\mu) \approx \tilde{\boldsymbol{u}}(t;\mu) = \boldsymbol{u}_{ref} + \boldsymbol{V}\boldsymbol{q}(t;\mu) + \overline{\boldsymbol{V}}\mathcal{N}(\boldsymbol{q}(t;\mu))$
  - $\boldsymbol{V} \in \mathbb{R}^{N\times n}$ using the first $n \ll N$ columns of U_s, where $S = U_s\Sigma_S Y_S^T$. $n \ll n_{tra}$ (PROM)
  - $\overline{\boldsymbol{V}} \in \mathbb{R}^{N\times \bar{n}}$ using a subset of the next $\bar{n} \ll N$ columns of U_s
  - $\mathcal{N}$: $\mathbb{R}^n \to \mathbb{R}^{\bar{n}}$ is a map represented by an ANN
  - Projection of solution onto $\boldsymbol{V}, \overline{\boldsymbol{V}} \to \ \overline{\boldsymbol{q}}^l = \mathcal{N}(\boldsymbol{q}^l)$, $\ \mathrm{l} = 1, \dots, N_s$

  PROM-ANN (Barnett et al. 2023, JCP)
- Objectives
  - $n \ll n_{tra}$
  - Demonstrated on inviscid Burgers' problem and double cone hypersonic flow benchmark problem
- Challenges
  - Can't derive mathematical bounds for errors (i.e. black box)



Construction of a ROB for: a traditional PROM (left); and a PROM-ANN (right)

Application: Inviscid Burgers' problem

$$\frac{\partial u_x}{\partial t} + \frac{1}{2}\left(\frac{\partial u_x^2}{\partial x} + \frac{\partial (u_x u_y)}{\partial y}\right) = 0.02\exp(\mu_2 x)$$

$$\frac{\partial u_y}{\partial t} + \frac{1}{2}\left(\frac{\partial (u_y u_x)}{\partial x} + \frac{\partial u_y^2}{\partial y}\right) = 0$$

$$u_x(x = 0, y, t; \mu) = \mu_1$$

$$u_x(x, y, t = 0) = u_y(x, y, t = 0) = 1$$

- Godunov-type scheme on two uniform meshes: M1: $\mathrm{N} = 250 \times 250$, M2: $\mathrm{N} = 750 \times 750$
- Trapezoidal method and constant $\Delta t = 0.05$ ($N_t = 500$ time-steps)
- $u_{ref} = 0$ (in all cases)
- 2-norm based relative error

$$\mathbb{RE} = \frac{\sum_{m=0}^{N_t} \|u^m(\mu) - \tilde{u}^m(\mu)\|_2}{\sum_{m=0}^{N_t} \|u^m(\mu)\|_2}$$

- Computational domain: $(x, y) \in [0,100] \times [0,100]$
- Time interval: $t \in [0,25]$
- Parameter domain: $\mu = (\mu_1, \mu_2) \in \mathcal{D} = [4.25, 5.50] \times [0.015,0.03]$, uniform sampling by a 3 x 3 grid → 9 training parameter points characterized by $\Delta\mu_1 = 0.625$ and $\Delta\mu_2 = 0.0075$
- Computing system: 1 node with 24 cores, CPU: 2.3GHz, RAM (shared): 192 GB
- 4,501 solutions snapshots

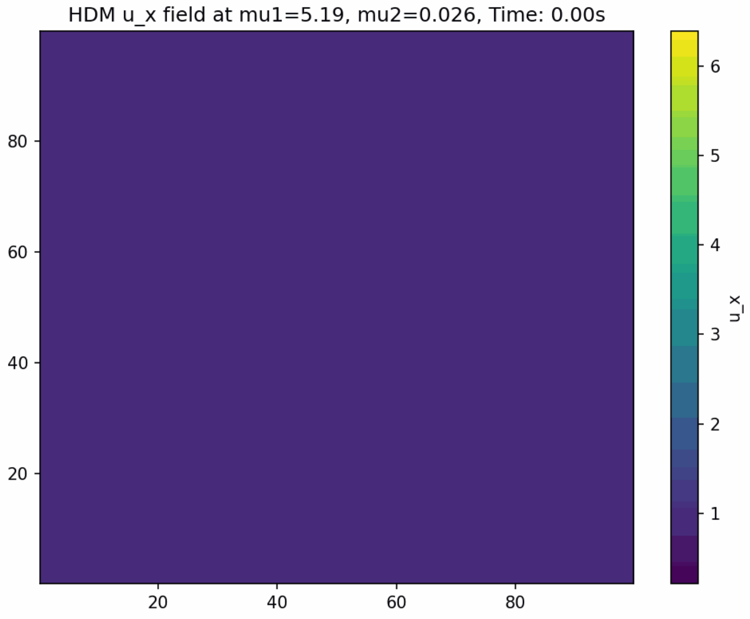
Application: Inviscid Burgers' problem

$$\frac{\partial u_x}{\partial t} + \frac{1}{2}\left(\frac{\partial u_x^2}{\partial x} + \frac{\partial (u_x u_y)}{\partial y}\right) = 0.02\exp(\mu_2 x)$$

$$\frac{\partial u_y}{\partial t} + \frac{1}{2}\left(\frac{\partial (u_y u_x)}{\partial x} + \frac{\partial u_y^2}{\partial y}\right) = 0$$

$$u_x(x = 0, y, t; \mu) = \mu_1$$

$$u_x(x, y, t = 0) = u_y(x, y, t = 0) = 1$$

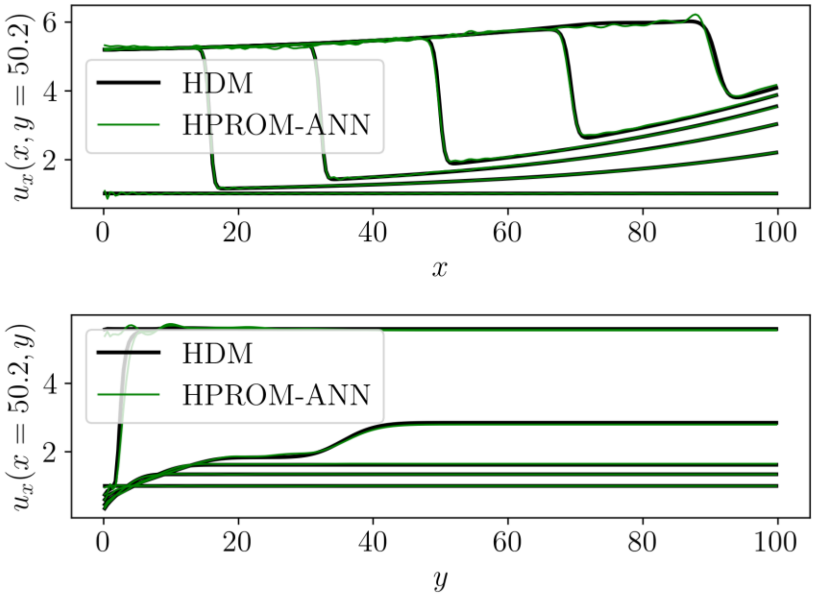


Alternative approach for building the map in the latent space: gaussian process

- Original ANN: 4 layers (Barnett et al. 2023, JCP)
- Test with 1 layer: relative errors of 1.5% vs 0.8%
- Deep learning is therefore *not needed* → replace ANN by GP regression which is analyzable
- A GP is a collection of random variables, any finite number of which have Gaussian distributions
- A GP is fully specified by a mean function $m(x)$ and covariance function $k(x_i, x_j)$
- The stochasticity might be used for uncertainty quantification; beyond scope of this talk
- Our problem: $\bar{\boldsymbol{q}}^l = GP(\boldsymbol{q}^l)$, $\mathrm{l} = 1, \dots, N_s$
- `scikit-learn` for constructing the map $\bar{\boldsymbol{q}}^l = GP(\boldsymbol{q}^l)$ and its gradient $\partial GP/\partial q$
- All tests are performed on out-of-sample μ values, reported only most unfavorable case

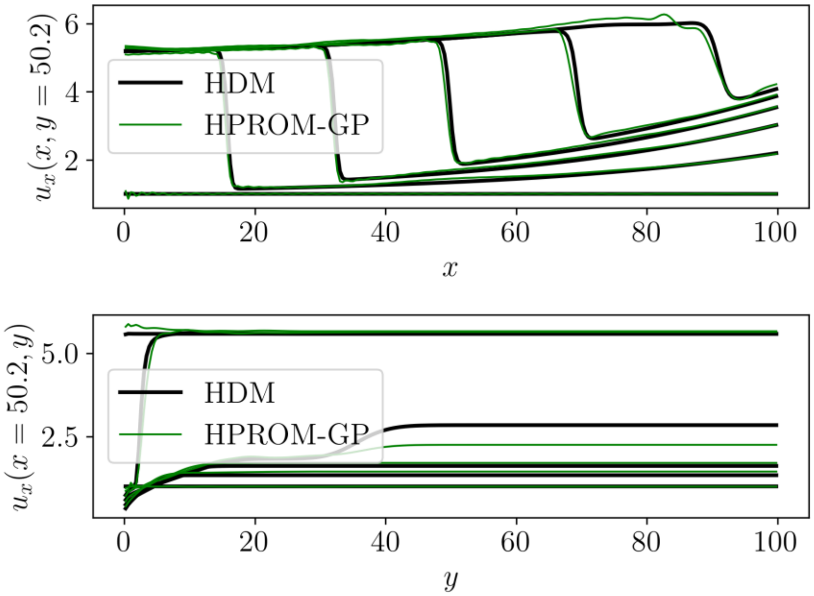
M1, $\mu_3 = (5.19, 0.026)$

RE= 1.51%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(n,\bar{n}) = (10{,}140)$

RE= 3.44%

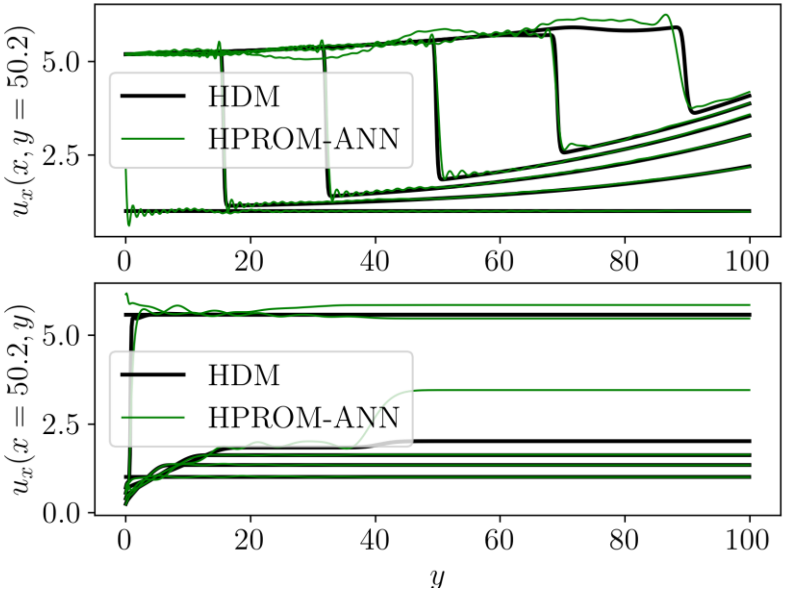


Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(n,\bar{n}) = (10{,}140)$

Stanford University

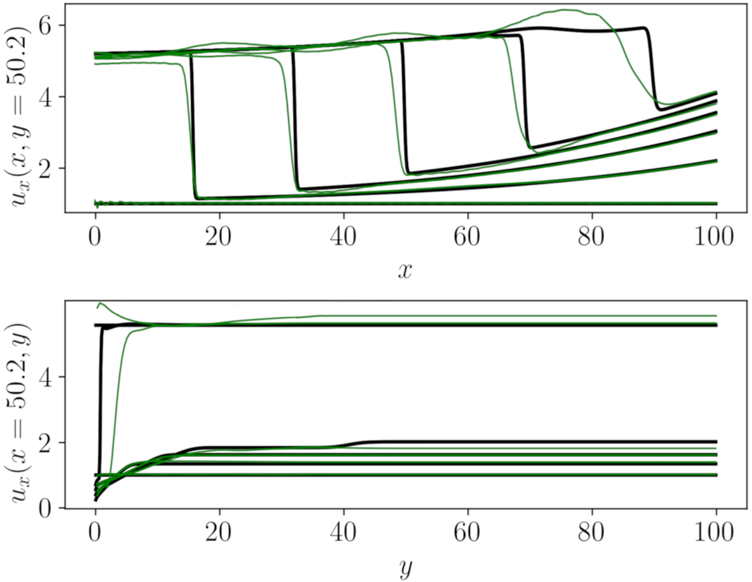
M2, $\mu_3 = (5.19, 0.026)$

RE=3.42%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(n, \bar{n}) = (10{,}140)$

RE=4.68%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(n, \bar{n}) = (10{,}140)$

Offline performance (wall clock time)

	Computational Model (offline)	n (N for HDM)	n_e (N_e for HDM)	Time (minutes)
Mesh M1	HDM	125,000	62,500	106.92
	PROM	95	4,390	85.16
	PROM-ANN	10	1,496	**22.24**
	PROM-GP	10	1,496	**17.53**
Mesh M2	HDM	1,125,000	562,500	404.88
	PROM	95	63,106	45.25*
	PROM-ANN	10	3,496	**24.54**
	PROM-GP	10	3,496	**24.25**

Table: Model parameters and offline performance results for both meshes

*Domain decomposition was used for M2

Online performance (wall clock time)

	Computational Model (online)	n (N for HDM)	n_e (N_e for HDM)	$\mathbb{RE}_{max}$	Time (minutes)	Speedup factor
Mesh M1	HDM	125,000	62,500	—	106.92	—
	PROM	95	4,390	1.43%	1.11	96.3
	PROM-ANN	10	1,496	**1.51%**	**0.30**	**356.4**
	PROM-GP	10	1,496	**3.44%**	**0.72**	**148.5**
Mesh M2	HDM	1,125,000	562,500	—	404.88	—
	PROM	95	63,106	7.98%	35.79	11.31
	PROM-ANN	10	3,496	**3.45%**	**0.70**	**578.4**
	PROM-GP	10	3,496	**4.68%**	**1.72**	**235.4**

Table: Model parameters and online performance results for both meshes

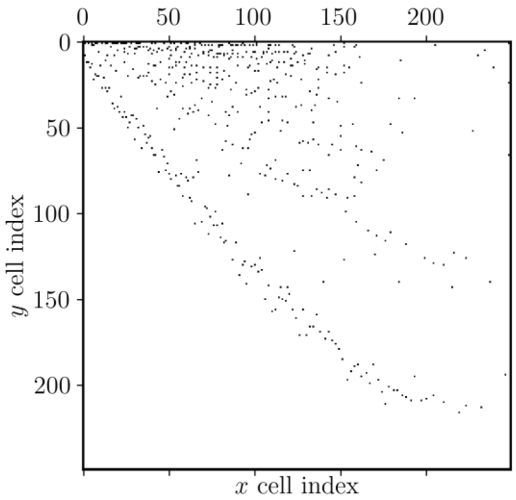
Conclusion & Future work

- Proposed concept of a PROM-GP approach based on the arbitrarily nonlinear approximation $\tilde{\boldsymbol{u}}(t;\mu) = \boldsymbol{u}_{ref} + \boldsymbol{V}\boldsymbol{q}(t;\mu) + \overline{\boldsymbol{V}}GP(\boldsymbol{q}(t;\mu))$, $\boldsymbol{V} \in \mathbb{R}^{N\times n}$, $\overline{\boldsymbol{V}} \in \mathbb{R}^{N\times \bar{n}}$, $GP\colon \mathbb{R}^n \to \mathbb{R}^{\bar{n}}$, $n \ll \bar{n} \ll N$
- GP's training is performed in the latent space and thus does not involve data whose dimension scales with N
- PROM-GP is hyperreducible using any well-established hyperreduction method
- For a parametric, 2D, inviscid Burgers' problem, PROM-GP delivers the same desired level of accuracy as the traditional PROM
- The online phase of PROM-GP gives a similar level of accuracy as PROM-ANN, in a time ranging from same to double, although it is implementation-specific
- Main objective: deriving mathematical error bounds; future: showcase on CFD benchmark

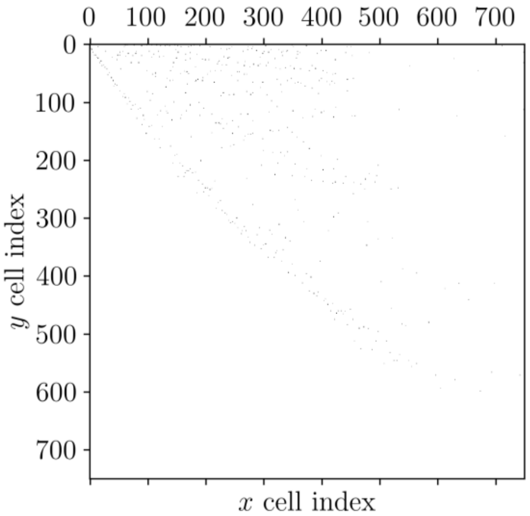
Back-up slides

Hyperreduction using ECSW

- ECSW training:
  - at only one of the sample parameter points – namely, (4.25, 0.0225)
  - at every 10-th solution snapshot



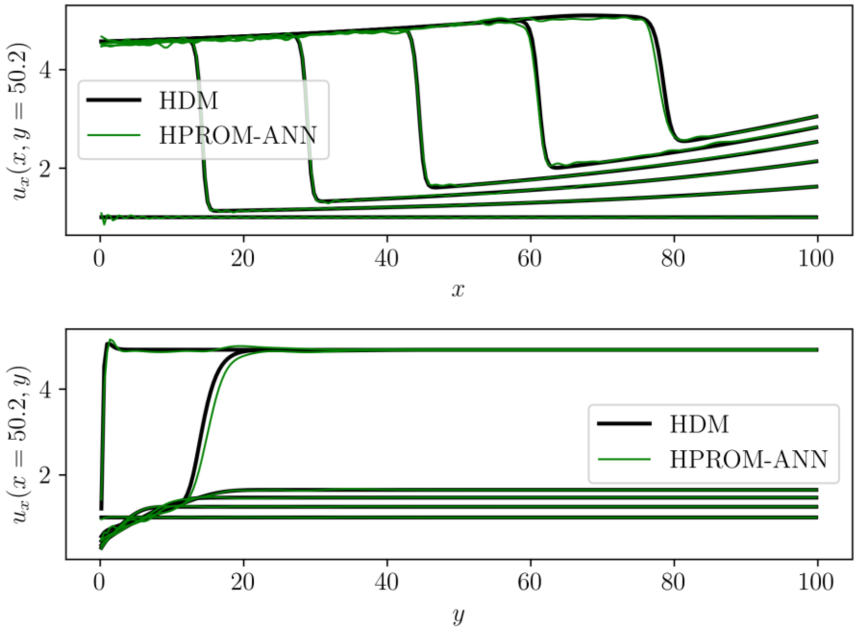
Reduced mesh M1: $n_e = 1496$ elements (2.4% of $\mathrm{N_e} = 62500$ elements)



Reduced mesh M2: $n_e = 3496$ elements (5.6% of $\mathrm{N_e} = 62500$ elements)

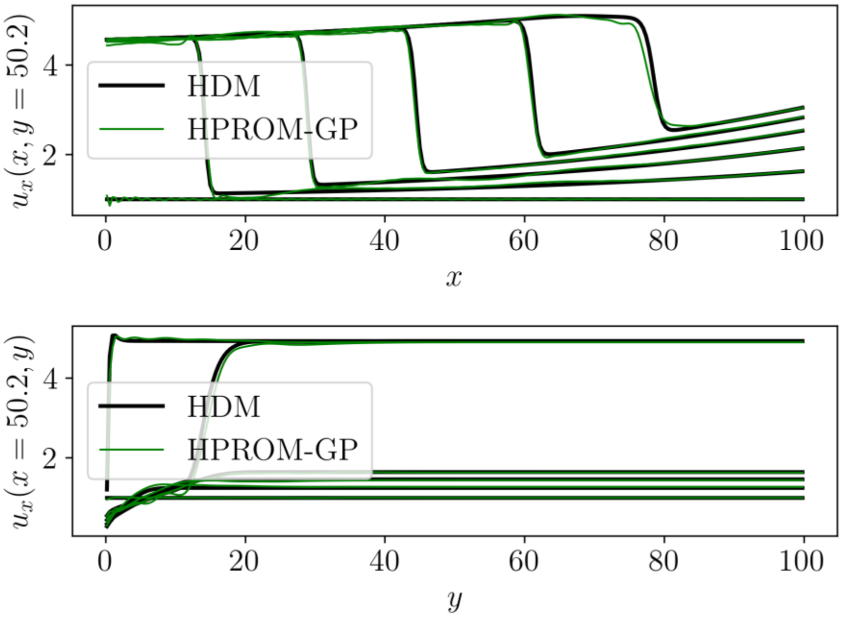
M1, $\mu_1 = (4.56, 0.019)$

RE= 1.01%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(\boldsymbol{n},\overline{\boldsymbol{n}}) = (10,140)$

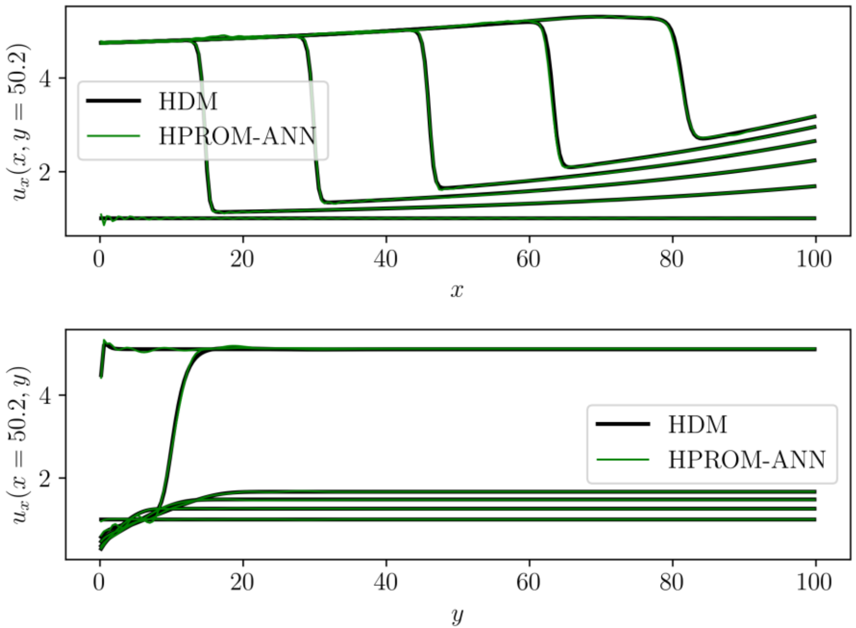
RE= 2.08%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(\boldsymbol{n},\overline{\boldsymbol{n}}) = (10,140)$

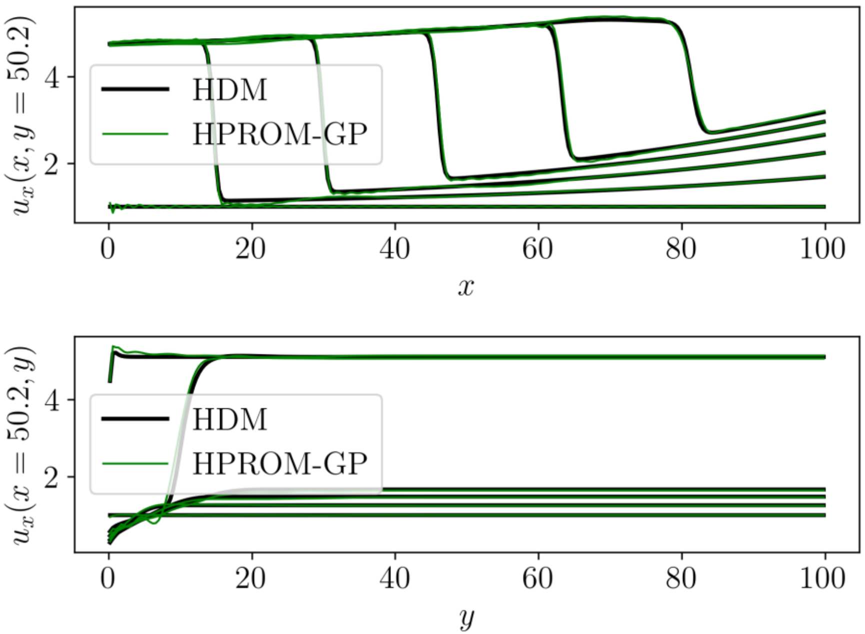
M1, $\mu_2 = (4.75, 0.02)$

RE= 1.21%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(\boldsymbol{n},\overline{\boldsymbol{n}}) = (10,140)$

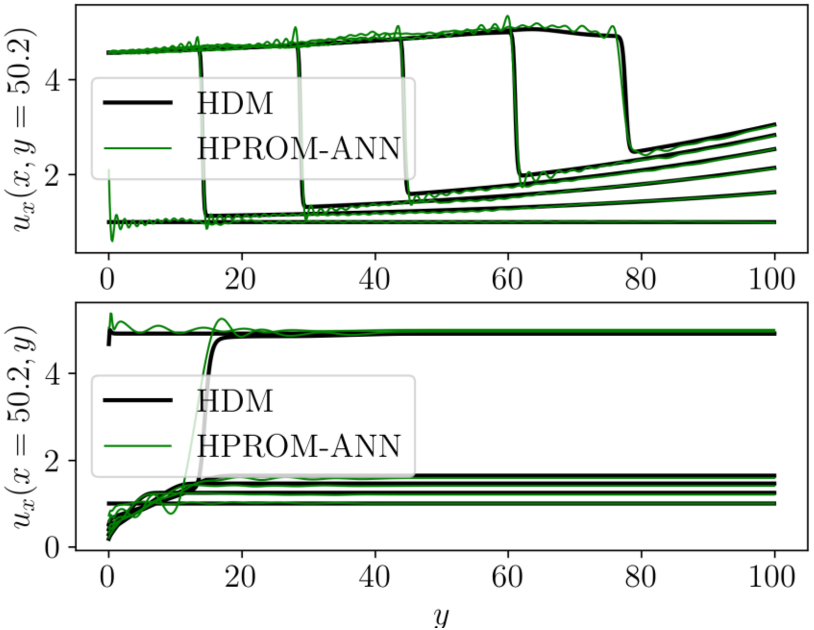
RE= 1.62%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(\boldsymbol{n},\overline{\boldsymbol{n}}) = (10,140)$

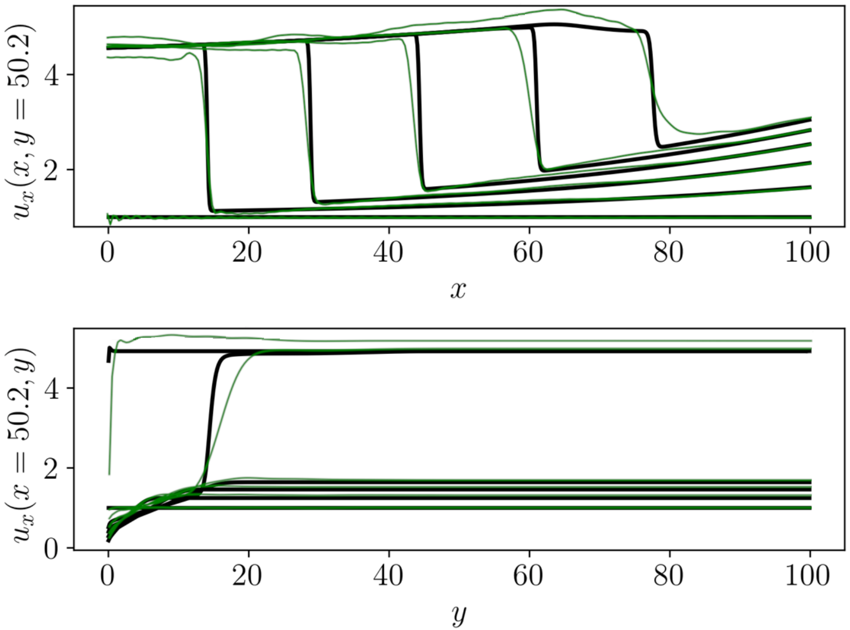
M2, $\mu_1 = (4.56, 0.019)$

RE=3.45%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(\boldsymbol{n},\bar{\boldsymbol{n}}) = (10,140)$

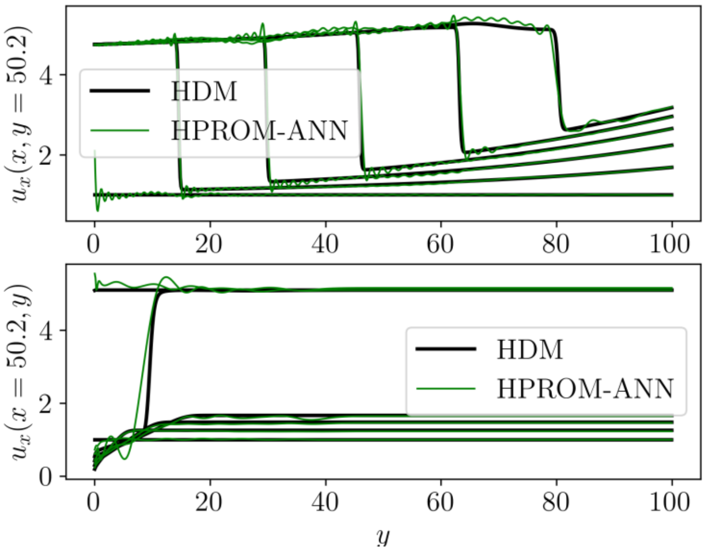
RE=4.10%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(\boldsymbol{n},\bar{\boldsymbol{n}}) = (10,140)$

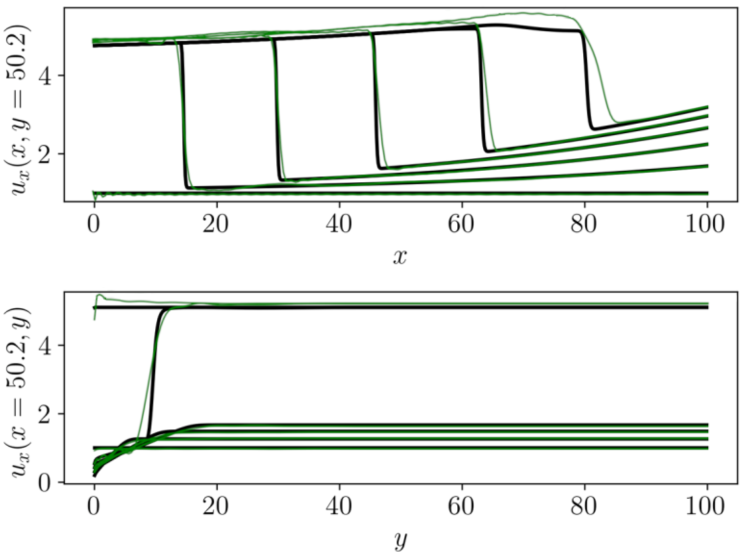
M2, $\mu_2 = (4.75, 0.02)$

RE=3.28%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-ANN with $(\boldsymbol{n}, \bar{\boldsymbol{n}}) = (10, 140)$

RE=4.47%



Numerical predictions performed using the LSPG- and ECSW-based HPROM-GP with $(\boldsymbol{n}, \bar{\boldsymbol{n}}) = (10, 140)$

Alternative approach: gaussian processes

- Original ANN: 6 layers (4 hidden layers)

  Barnett et al. 2023, JCP
- Test with 1 hidden layer and various sizes

  $$(q,512) \xrightarrow{\text{ELU}} (512,1024) \xrightarrow{\text{ELU}} (1024,\bar{q})$$
- We propose to replace ANN by a GP
- A GP is a collection of random variables, any finite number of which have Gaussian distributions
- A GP is fully specified by a mean function $m(x)$ and covariance function $k(x_i, x_j)$
- Our problem: $\bar{\boldsymbol{q}}^l = GP(\boldsymbol{q}^l)$, $l = 1, \dots, N_s$
- Matérn kernel:

$$k(x_i, x_j) = \left(1 + \frac{\sqrt{3}}{l} d(x_i, x_j)\right) \exp\left(-\frac{\sqrt{3}}{l} d(x_i, x_j)\right)$$

ANN Type	Elapsed Time (s)	RE
4HL, original	16.01	0.79%
1HL,(32,64)	15.63	8.38%
1HL,(256,256)	19.65	2.19%
1HL,(512,512)	20.19	1.55%
1HL,(512,1024)	22.80	1.74%

- `scikit-learn` for constructing the $GP(\boldsymbol{q}^l)$ and its gradient $\partial GP / \partial q$
- All tests are performed on out-of-sample