From model reduction to scientific machine learning and back: Learning reduced models from data and their applications in uncertainty quantification and beyond

---

Benjamin Peherstorfer
Courant Institute of Mathematical Sciences, New York University

July 2020

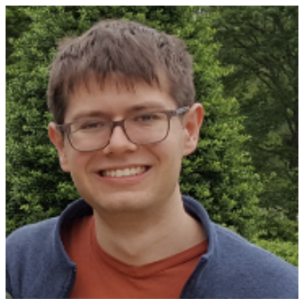
My group

Benjamin Peherstorfer
Courant Institute of Mathematical Sciences, New York University

Research computational mathematics, machine learning, computational statistics, numerical analysis, and scientific computing

Terrence Alsup
PhD student

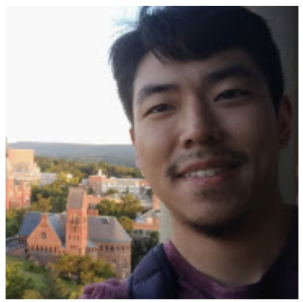
Frederick Law
PhD student
(co-advised)



Karl Otness
PhD student
(co-advised)

Donsub Rim
PostDoc

Nihar Sawant
PhD student



Wayne Uy
Courant instructor

Group website:
https://cims.nyu.edu/∼pehersto

Scientific machine learning

Training phase

- Collect training data from system
- Define a hypothesis space
- Minimize loss on training data

Testing/evaluation phase

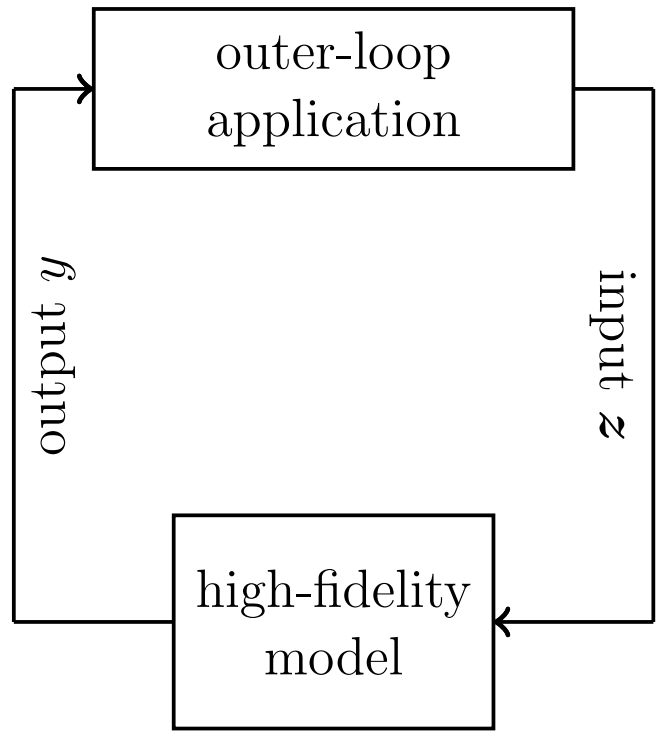
- Evaluate model to predict at new input
- Generalization to unseen inputs?

Scientific machine learning

- Encoding physics in model
- Interpretability of predictions
- Increasing robustness, ...

Model reduction

Outer-loop application "Computational applications that form outer loops around a model where in each iteration an input z is received and the corresponding model output $y = f(z)$ is computed, and an overall outer loop result is obtained at the termination of the outer loop" [P., Willcox, Gunzburger, SIAM Review, 2018]

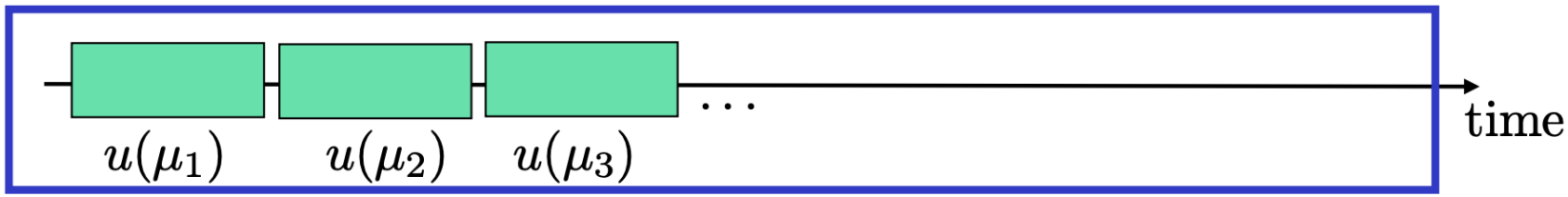


Examples

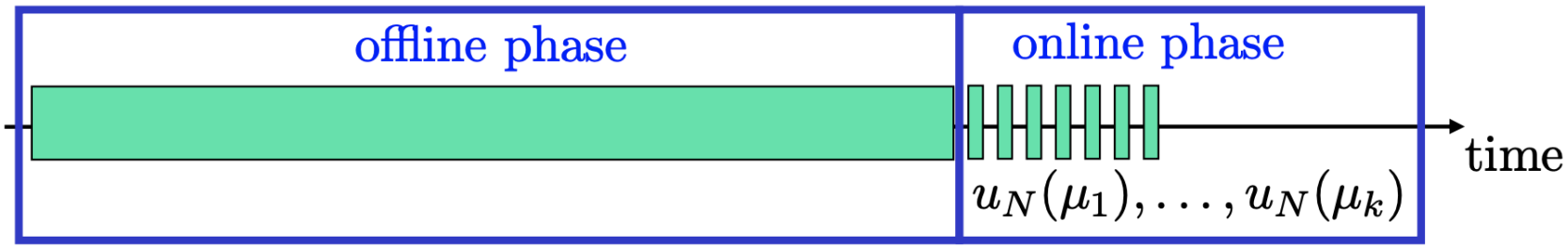
- Optimization
  outer-loop result = optimal design
- Uncertainty quantification
  outer-loop result = estimate of statistics
- Inverse problems
- Data assimilation
- Control problems
- Sensitivity analysis

Model reduction reduces expensive models

Multi-query with high dimensional model:



Multi-query with reduced model:



Offline

- Generate snapshots/library (data), using high-fidelity models
- Generate reduced models

Online

- Select appropriate library records and/or reduced models
- Rapid prediction, control, optimization, UQ using (multi-fidelity) methods

[Figure: Bernard Haasdonk]

Machine learning

"The scientific study of algorithms & statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns & inference instead." [Wikipedia]

Reduced-order modeling

"Model order reduction (MOR) is a technique for reducing the computational complexity of mathematical models in numerical simulations." [Wikipedia]

What is the connection between reduced-order modeling and machine learning?

Model reduction methods have grown from Computational Science & Engineering, with focus on ***reducing*** **high-dimensional models** that arise from physics-based modeling, whereas machine learning has grown from Computer Science, with a focus on ***creating*** **low-dimensional models** from black-box data streams. [Swischuk et al., *Computers & Fluids*, 2019]

[Slide: Karen Willcox]

Machine learning

"The scientific study of algorithms & statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns & inference instead." [Wikipedia]

Reduced-order modeling

"Model order reduction (MOR) is a technique for reducing the computational complexity of mathematical models in numerical simulations." [Wikipedia]

Reduced-order modeling & machine learning: **Can we get the best of both worlds?**

Discover hidden structure	Embed governing equations
Non-intrusive implementation	Structure-preserving
Black-box & flexible	Predictive (error estimators)
Accessible & available	Stability-preserving

[Slide: Karen Willcox]

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Format

Talk about concepts, try them in hands-on sessions, learn about details in references

Lectures/discussions

- Takes about 45min per meeting
- Slides/board
- Strongly encouraged to ask questions
- This should be interactive

Hands-on session

- Download handout and Matlab code (ownCloud)
- Will start breakout rooms and randomly assign you to rooms
- There will be 5-8 people per room
- I will go through the rooms and will be available for questions/discussion

White board notes and slides will be available for download (ownCloud)

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Outline

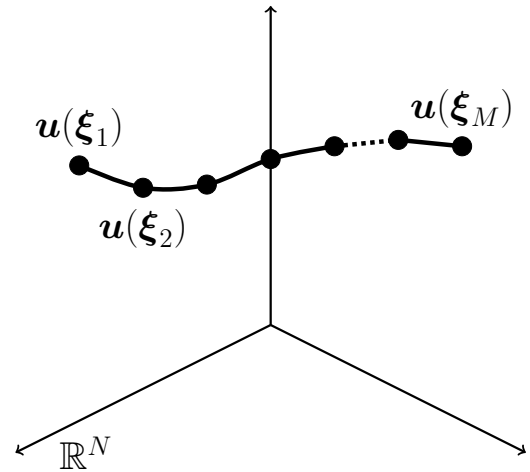
- **Introduction to (intrusive) model reduction**
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Model reduction

white board

MOR: Classical (intrusive) model reduction

Given full model f, construct reduced $\tilde{f}$ via projection



1. Construct n-dim. basis $V = [v_1, \ldots, v_n] \in \mathbb{R}^{N \times n}$

- Proper orthogonal decomposition (POD)
- Interpolatory model reduction
- Reduced basis method (RBM), ...

2. Project full-model operators $A_1, \ldots, A_\ell, B$ onto reduced space, e.g.,

$$\tilde{A}_i = \underbrace{V^T \overbrace{A_i}^{N \times N^i} (V \otimes \cdots \otimes V)}_{n \times n^i}, \qquad \tilde{B} = \underbrace{V^T \overbrace{B}^{N \times p}}_{n \times p}$$

3. Construct reduced model

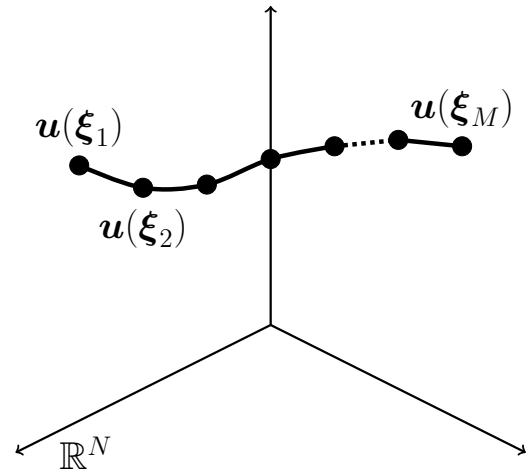
$$\tilde{x}_{k+1} = \tilde{f}(\tilde{x}_k, u_k) = \sum_{i=1}^{\ell} \tilde{A}_i \tilde{x}_k^i + \tilde{B} u_k\,, \qquad k = 0, \ldots, K-1$$

with $n \ll N$ and $\|V\tilde{x}_k - x_k\|$ small in appropriate norm

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]

MOR: Classical (intrusive) model reduction

Given full model f, construct reduced $\tilde{f}$ via projection



1. **Construct n-dim. basis $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$**
   - Proper orthogonal decomposition (POD)
   - Interpolatory model reduction
   - Reduced basis method (RBM), ...

2. **Project full-model operators $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell, \boldsymbol{B}$ onto reduced space, e.g.,**

$$\tilde{\boldsymbol{A}}_i = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{A}_i}^{N \times N^i} (\boldsymbol{V} \otimes \cdots \otimes \boldsymbol{V})}_{n \times n^i}, \qquad \tilde{\boldsymbol{B}} = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{B}}^{N \times p}}_{n \times p}$$

3. **Construct reduced model**

$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k^i + \tilde{\boldsymbol{B}} \boldsymbol{u}_k\,, \qquad k = 0, \ldots, K-1$$

with $n \ll N$ and $\|\boldsymbol{V}\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_k\|$ small in appropriate norm

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]

MOR: References

Model reduction

- G Rozza, DBP Huynh, AT Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, Archives of Computational Methods in Engineering 15 (3), 1
- AC Antoulas, Approximation of large-scale dynamical systems, Society for Industrial and Applied Mathematics, 2004
- P Benner, S Gugercin, K Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM review 57 (4), 483-531
- JS Hesthaven, G Rozza, B Stamm, Certified reduced basis methods for parametrized partial differential equations, Springer, 2016

Interpolating reduced operators

- D Amsallem, C Farhat, Interpolation method for adapting reduced-order models and application to aeroelasticity, AIAA Journal 46 (7), 1803-1813
- J Degroote, J Vierendeels, K Willcox, Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis, International Journal for Numerical Methods in Fluids 63 (2), 207-230

Working with quadratic and polynomial systems

- Peter Benner and Tobias Breiten, Two-Sided Projection Methods for Nonlinear Model Order Reduction, SIAM Journal on Scientific Computing 2015 37:2, B239-B260, 2015
- B Kramer, K Willcox, Nonlinear Model Order Reduction via Lifting Transformations and Proper Orthogonal Decomposition, AIAA Journal 57 (6), 2297-2307, 2019
- B Peherstorfer, K Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Computer Methods in Applied Mechanics and Engineering 306, 196-215, 2016

Introduction to model reduction[1]

1 Problem setup

We will apply model reduction to the Burgers' equation given by

$$\frac{\partial}{\partial t}x(\xi,t;\mu) + x(\xi,t;\mu)\frac{\partial}{\partial \xi}x(\xi,t;\mu) - \mu\frac{\partial^2}{\partial \xi^2}x(\xi,t;\mu) = 0$$

subject to the Dirichlet boundary conditions

$$x(-1,t;\mu) = u(t), \quad x(1,t;\mu) = -u(t)$$

where $\xi \in (-1,1)$ is the spatial variable, $t \in [0,T]$ is time, μ is some parameter, and $u(t)$ is some input signal. We discretize the spatial domain with finite difference on an equidistant grid with mesh size δx and apply the forward Euler method in time with step size δt to obtain the discrete system

$$\boldsymbol{x}_{k+1}(\mu) = \boldsymbol{A}(\mu)\boldsymbol{x}_k(\mu) + \boldsymbol{F}(\mu)\boldsymbol{x}_k^2(\mu) + \boldsymbol{B}(\mu)u_k \tag{1}$$

where $\boldsymbol{x}_k(\mu) \in \mathbb{R}^N, \boldsymbol{x}_k^2(\mu) \in \mathbb{R}^{N(N+1)/2}, u_k \in \mathbb{R}, \boldsymbol{A}(\mu) \in \mathbb{R}^{N\times N}, \boldsymbol{F}(\mu) \in \mathbb{R}^{N\times N(N+1)/2}, \boldsymbol{B}(\mu) \in \mathbb{R}^{N\times 1}$ and $k = 0,1,2,\ldots$. The components of $\boldsymbol{x}_k(\mu)$ represent approximations to $x(\xi,t;\mu)$ at various spatial locations for a specified time. The vector $\boldsymbol{x}_k^2$ is a vector representing products formed by components of $\boldsymbol{x}_k$ without duplicates. More formally, if $\boldsymbol{y} = [y_1,\ldots,y_N]^T$ then

$$\boldsymbol{y}^2 = \begin{bmatrix} \boldsymbol{y}^{(1)} \\ \vdots \\ \boldsymbol{y}^{(N)} \end{bmatrix} \in \mathbb{R}^{N(N+1)/2} \tag{2}$$

where $\boldsymbol{y}^{(i)}$ is defined as

$$\boldsymbol{y}^{(i)} = y_i \begin{bmatrix} y_1 \\ \vdots \\ y_i \end{bmatrix} \in \mathbb{R}^i.$$

The goal of this exercise is to formulate the reduced system for the full system (1), construct it numerically, and use it for prediction.

To assist you with this exercise, we have provided Matlab functions that assemble the matrices in the full system (1). The functions `get_x_sq.m` and `getBurgersMatrices.m` are extracted from the Github repository of Elizabeth Qian. The function `genDiscSys.m` then utilizes these functions to produce $\boldsymbol{A}(\mu), \boldsymbol{F}(\mu), \boldsymbol{B}(\mu)$ for specified values of $N, \delta x, \delta t, \mu$.

You may want to investigate how `get_x_sq.m` operates by applying the function to `v = [1 2 3]` in the command line. Investigate what happens instead when the input is `v = [1 2 3; 4 5 6]`. Observe that the function `get_x_sq` applies (2) to each row in the input argument.

[1]Prepared by: Wayne Uy, PhD

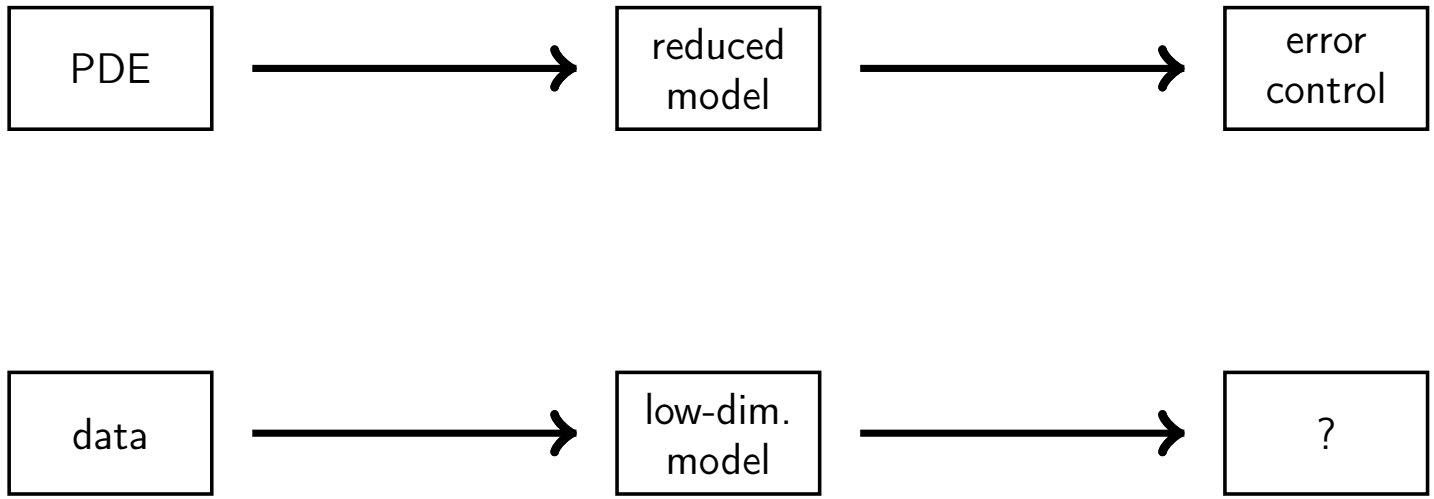
Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Outline

- Introduction to (intrusive) model reduction
- **Learning reduced models from data**
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

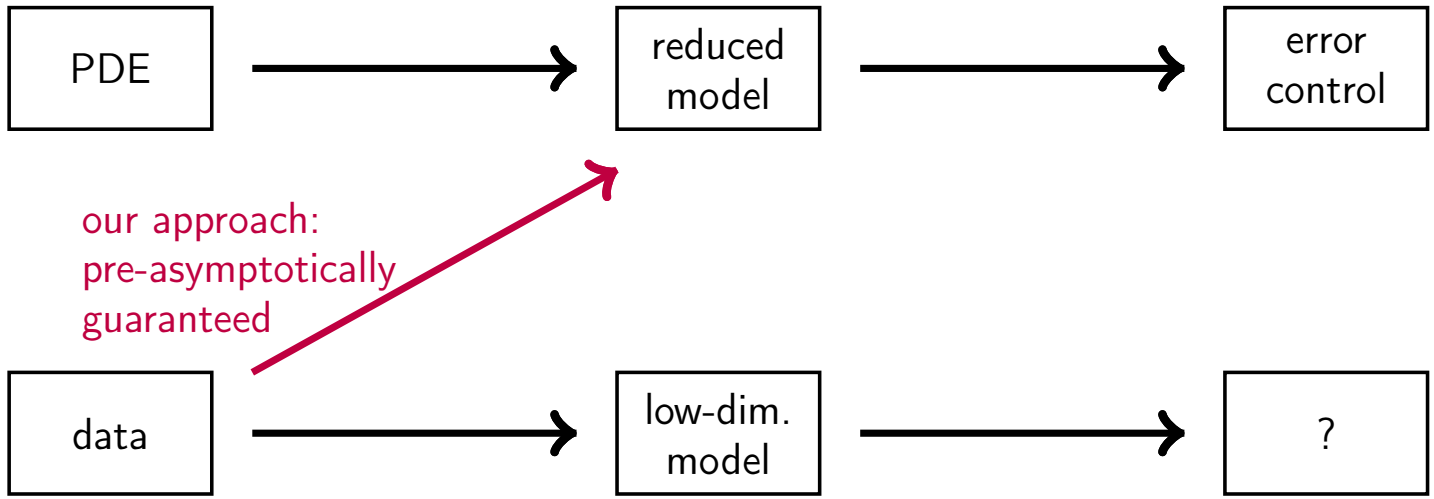
Learning dynamical-system models from data



Learn low-dimensional model from data of dynamical system

- Interpretable
- System & control theory
- Fast predictions
- Guarantees for finite data

Recovering reduced models from data



Learn low-dimensional model from data of dynamical system

- Interpretable
- System & control theory
- Fast predictions
- Guarantees for finite data

Learn reduced model from trajectories of high-dim. system

- Recover *exactly* and *pre-asymptotically* reduced models from data
- Then build on rich theory of model reduction to establish error control

Intro: Polynomial nonlinear terms

Models with polynomial nonlinear terms

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t;\boldsymbol{\mu}) &= \boldsymbol{f}(\boldsymbol{x}(t;\boldsymbol{\mu}),\boldsymbol{u}(t);\boldsymbol{\mu}) \\ &= \sum_{i=1}^{\ell} \boldsymbol{A}_i(\boldsymbol{\mu})\boldsymbol{x}^i(t;\boldsymbol{\mu}) + \boldsymbol{B}(\boldsymbol{\mu})\boldsymbol{u}(t)\end{aligned}$$

- Polynomial degree $\ell \in \mathbb{N}$
- Kronecker product $\boldsymbol{x}^i(t;\boldsymbol{\mu}) = \bigotimes_{j=1}^{i} \boldsymbol{x}(t;\boldsymbol{\mu})$
- Operators $\boldsymbol{A}_i(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N^i}$ for $i = 1,\ldots,\ell$
- Input operator $\boldsymbol{B}(\boldsymbol{\mu}) \in \mathbb{R}^{N\times p}$

Lifting and transformations

- Lift general nonlinear systems to quadratic-bilinear ones [Gu, 2011], [Benner, Breiten, 2015], [Benner, Goyal, Gugercin, 2018], [Kramer, Willcox, 2019], [Swischuk, Kramer, Huang, Willcox, 2019], [Qian, Kramer, P., Willcox, 2019]
- Koopman lifts nonlinear systems to infinite linear systems [Rowley et al, 2009], [Schmid, 2010]

Intro: Beyond polynomial terms (nonintrusive)

Mathematics > Numerical Analysis

Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems

Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, Karen Willcox

(Submitted on 17 Dec 2019 (v1), last revised 23 Dec 2019 (this version, v2))

We present Lift & Learn, a physics-informed method for learning low-dimensional models for large-scale dynamical systems. The method exploits knowledge of a system's governing equations to identify a coordinate transformation in which the system dynamics have quadratic structure. This transformation is called a lifting map because it often adds auxiliary variables to the system state. The lifting map is applied to data obtained by evaluating a model for the original nonlinear system. This lifted data is projected onto its leading principal components, and low-dimensional linear and quadratic matrix operators are fit to the lifted reduced data using a least-squares operator inference procedure. Analysis of our method shows that the Lift & Learn models are able to capture the system physics in the lifted coordinates at least as accurately as traditional intrusive model reduction approaches. This preservation of system physics makes the Lift & Learn models robust to changes in inputs. Numerical experiments on the FitzHugh-Nagumo neuron activation model and the compressible Euler equations demonstrate the generalizability of our model.

Intro: Beyond polynomial terms (nonintrusive)

arXiv.org > math > arXiv:1912.08177

Search... All fields Search

Help | Advanced Search

Mathematics > Numerical Analysis

Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems

Elizabeth Qian, Boris

(Submitted on 17 Dec 201

We present Lift & Lea
exploits knowledge o
structure. This transfo
data obtained by eval
low-dimensional linea
Analysis of our metho
accurately as traditio
changes in inputs. Nu
demonstrate the gene

Download:

- PDF
- Other formats

ontext:

e by:

ations

Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems

Elizabeth Qian (elizqian@mit.edu)
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology

Boris Kramer
Department of Mechanical & Aerospace Engineering
University of California - San Diego

Benjamin Peherstorfer
Courant Institute of Mathematical Sciences
New York University

Karen Willcox
Oden Institute for Computational Engineering & Sciences
University of Texas at Austin

Deriving low-dimensional models

Traditional solvers for nonlinear PDEs are expensive: need inexpensive surrogate models for practical computations

- Projection-based reduced models rely on full knowledge of physics and their construction traditionally requires intrusive access to codes
- Data-fit models in machine learning treat solvers as black boxes and ignore physics

We propose **Lift & Learn**, a physics-informed method for learning reduced models that can recover the generalization accuracy of traditional intrusive reduced models

- Knowledge of the governing PDE is exploited to identify a lifting map (variable transformation + auxiliary variables) that exposes quadratic structure in the PDE
- Lifting lets us reformulate nonlinear model reduction as a non-intrusive polynomial operator inference

Lifting PDEs to quadratic form

Consider the general nonlinear governing PDE with state s:

$$\frac{\partial s}{\partial t} = f(s)$$

A quadratic lifting map $\mathcal{T}$ transforms and augments the system state so that the PDE in the lifted state, $w = \mathcal{T}(s)$, contains only quadratic nonlinearities, e.g.:

$$\frac{\partial w}{\partial t} = a_0 w + a_1 \frac{\partial w}{\partial x} + a_2 w^2 + a_3 w \frac{\partial w}{\partial x}$$

This structure allows us to reformulate the learning task as a polynomial operator inference problem.

Example

Original PDE $\frac{\partial s}{\partial t} = -e^s$ *Lifting map* $\mathcal{T}: s \mapsto \begin{pmatrix} s \\ -e^s \end{pmatrix} \equiv \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ *Lifted PDE* $\frac{\partial}{\partial t}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} w_2 \\ (w_2)^2 \end{pmatrix}$

How general is the lifting approach?

Many nonlinear terms in engineering applications can be lifted to quadratic form.[1] In some cases, quadratic transformations are known, e.g., the specific volume variables for the Euler and Navier-Stokes equations underlying many fluids applications.

How is the lifting derived?

Our current strategy is problem-specific: we introduce auxiliary variables for non-quadratic terms of the PDE and augment the system with evolution equations for these new variables. Automated discovery of a lifting is a direction for future work.

Acknowledgments

The authors acknowledge the NSF GRFP, the Hertz Foundation, the US Air Force (FA9550-17-1-0195, FA9550-15-1-0038, FA9550-18-1-0023), the US Department of Energy Applied Mathematics Program (DE-FG02-08ER2585, DE-SC0009297, DESC0019334), and the SUTD-MIT International Design Centre.

Lift & Learn

1. Solve N-dimensional spatial discretization of original nonlinear PDE to generate K snapshots. Apply **lifting** map to snapshot data to obtain K lifted state and time derivative pairs $(\mathbf{W}, \dot{\mathbf{W}} \in \mathbb{R}^{N \times K})$
2. Compute a d-dimensional global basis, $\mathbf{V}_d$, for the lifted data, e.g. via Proper Orthogonal Decomposition (POD) and project data:
$$\hat{\mathbf{W}} = \mathbf{V}_d^T \mathbf{W}, \qquad \dot{\hat{\mathbf{W}}} = \mathbf{V}_d^T \dot{\mathbf{W}}$$
where $d \ll N$. The reduced model for projected data can be parameterized by small, dense matrix operators:
$$\frac{\mathrm{d}\hat{\mathbf{w}}}{\mathrm{d}t} = \hat{\mathbf{A}}\hat{\mathbf{w}} + \hat{\mathbf{H}}(\hat{\mathbf{w}} \otimes \hat{\mathbf{w}})$$
3. Use least-squares operator inference[2] procedure to **learn** $\hat{\mathbf{A}} \in \mathbb{R}^{d \times d}, \hat{\mathbf{H}} \in \mathbb{R}^{d \times d^2}$ from data:
$$\min_{\hat{\mathbf{A}} \in \mathbb{R}^{d \times d}, \hat{\mathbf{H}} \in \mathbb{R}^{d \times d^2}} \frac{1}{K} \left\| \dot{\hat{\mathbf{W}}}^T \hat{\mathbf{A}}^T + \left(\hat{\mathbf{W}} \otimes \hat{\mathbf{W}}\right)^T \hat{\mathbf{H}}^T - \dot{\hat{\mathbf{W}}}^T \right\|_F^2$$

Bounding the residual of Lift & Learn models[3]:

If the original nonlinear PDE solver uses a spatial discretization scheme with order of accuracy p, and the map $\mathcal{T}$ is continuous with Lipschitz derivative, then the residual of the Lift & Learn model on the training data is bounded:

$$\min_{\hat{\mathbf{A}} \in \mathbb{R}^{d \times d}, \hat{\mathbf{H}} \in \mathbb{R}^{d \times d^2}} \frac{1}{K} \left\| \dot{\hat{\mathbf{W}}}^T \hat{\mathbf{A}}^T + \left(\hat{\mathbf{W}} \otimes \hat{\mathbf{W}}\right)^T \hat{\mathbf{H}}^T - \dot{\hat{\mathbf{W}}}^T \right\|_F^2 \leq (c_0 N^{0.5-p} + c_1 \varepsilon)^2$$

where ε is the projection error of $\mathbf{W}$ onto $\mathbf{V}_d$ and c_0, c_1 are constants.

Implications:

1. By respecting the problem physics in the lifted coordinates we can put an upper bound on the residual of the Lift & Learn model.
2. The Lift & Learn model residual is at least as good as the residual of an intrusive lifted POD reduced model.

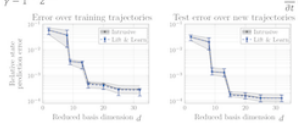
Generalization & accuracy: Euler equations

Original PDE $\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{pmatrix} = 0$, $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2$

Lifting map $\mathcal{T}: \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix} \mapsto \begin{pmatrix} 1/\rho \equiv \zeta \\ u \\ p \end{pmatrix}$

Lifted PDE

$$\frac{\partial \zeta}{\partial t} = -u\frac{\partial \zeta}{\partial x} + \zeta\frac{\partial u}{\partial x}$$
$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \zeta\frac{\partial p}{\partial x}$$
$$\frac{\partial p}{\partial t} = -\gamma p\frac{\partial u}{\partial x} - u\frac{\partial p}{\partial x}$$



Generalization & accuracy: FitzHugh-Nagumo neuron activation model[4]

A benchmark problem in nonlinear model reduction:

Original PDE:

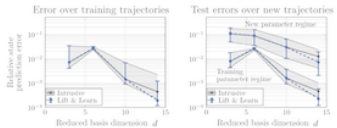
$$\gamma\frac{\partial s_1}{\partial t} = \gamma^2\frac{\partial^2 s_1}{\partial x^2} - s_1^3 + 1.1s_1^2 - 0.1s_1 + s_2 + 0.05$$
$$\frac{\partial s_2}{\partial t} = 0.5s_1 - 2s_2 + 0.05$$

Lifting map:

$$\mathcal{T}: \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \mapsto \begin{pmatrix} s_1 \\ s_2 \\ (s_1)^2 \end{pmatrix} \equiv \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

Lifted quadratic PDE:

$$\gamma\frac{\partial w_1}{\partial t} = \gamma^2\frac{\partial^2 w_1}{\partial x^2} - w_1 w_3 + 1.1w_1^2 - 0.1w_1 + w_2 + 0.05$$
$$\frac{\partial w_2}{\partial t} = 0.5w_1 - 2w_2 + 0.05$$
$$\frac{\gamma}{2}\frac{\partial w_3}{\partial t} = \gamma^2 w_1\frac{\partial^2 w_1}{\partial x^2} - w_2^2 + 1.1w_1 w_3 - 0.1w_3 + w_1 w_2 + 0.05w_1$$



Conclusions

Our Lift & Learn approach infers low-dimensional quadratic models for nonlinear PDEs:

- Lifting maps expose quadratic structure in the nonlinear PDE so that a low-dimensional model can be explicitly parametrized by polynomial matrix operators
- We fit polynomial operators to lifted data obtained non-intrusively from the original nonlinear model
- Numerical experiments show that Lift & Learn models recover the generalization accuracy of intrusive projection-based reduced models

References

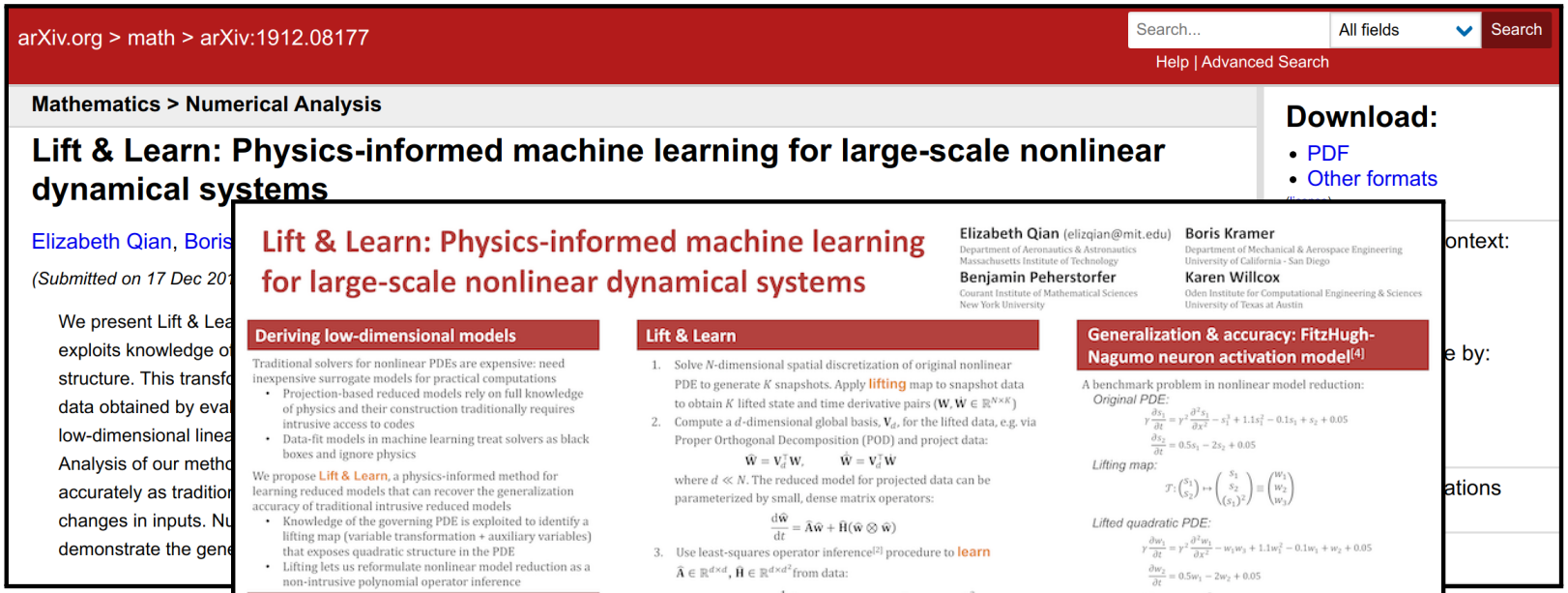
[1] C. Gu, QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 30 (2011) 1307–1320.

[2] B. Peherstorfer and K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Comput. Method Appl. M., 306 (2016), pp. 196–215.

[3] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox, "Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems." To appear, 2020, in Physica D.

[4] S. Chaturantabut, D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. on Sci. Comp. 32 (2010) 2737–2764.

Intro: Beyond polynomial terms (nonintrusive)



Operator inference for non-intrusive model reduction of non-polynomial nonlinear systems

Boris Kramer, University of California San Diego

We present a data-driven non-intrusive model reduction method that learns low-dimensional models of dynamical systems with non-polynomial nonlinear terms that are spatially local and that are given in analytic form. The proposed approach requires only the non-polynomial terms in analytic form and learns the rest of the dynamics from snapshots computed with a potentially black-box full-model solver. The linear and polynomially nonlinear dynamics are learned by solving a linear least-squares problem where the analytically given non-polynomial terms are incorporated in the right-hand side of the least-squares problem. The resulting ROM thus contains learned polynomial operators together with the analytic form of the non-polynomial nonlinearity. The proposed method is demonstrated on several test problems which provides evidence that the proposed approach learns reduced models that achieve comparable accuracy as state-of-the-art intrusive model reduction methods that require full knowledge of the governing equations.

Intro: Parametrized systems

Consider time-invariant system with polynomial nonlinear terms

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t;\boldsymbol{\mu}) &= \boldsymbol{f}(\boldsymbol{x}(t;\boldsymbol{\mu}),\boldsymbol{u}(t);\boldsymbol{\mu}) \\ &= \sum_{i=1}^{\ell} \boldsymbol{A}_i(\boldsymbol{\mu})\boldsymbol{x}^i(t;\boldsymbol{\mu}) + \boldsymbol{B}(\boldsymbol{\mu})\boldsymbol{u}(t)\end{aligned}$$

Parameters

- Infer models $\hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_1),\ldots,\hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_M)$ at parameters

$$\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_M \in \mathcal{D}$$

- For new $\boldsymbol{\mu} \in \mathcal{D}$, interpolate operators of [Amsallem et al., 2008], [Degroote et al., 2010]

$$\hat{\boldsymbol{f}}(\boldsymbol{\mu}_1),\ldots,\hat{\boldsymbol{f}}(\boldsymbol{\mu}_M)$$

Trajectories

$$\boldsymbol{X} = [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K] \in \mathbb{R}^{N\times K}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1,\ldots,\boldsymbol{u}_K] \in \mathbb{R}^{p\times K}$$

Intro: Parametrized systems

Consider time-invariant system with polynomial nonlinear terms

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) =& \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ &= \sum_{i=1}^{\ell} \boldsymbol{A}_i \boldsymbol{x}^i(t) + \boldsymbol{B}\boldsymbol{u}(t)\end{aligned}$$

Parameters

- Infer models $\hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_1), \ldots, \hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_M)$ at parameters

$$\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M \in \mathcal{D}$$

- For new $\boldsymbol{\mu} \in \mathcal{D}$, interpolate operators of [Amsallem et al., 2008], [Degroote et al., 2010]

$$\hat{\boldsymbol{f}}(\boldsymbol{\mu}_1), \ldots, \hat{\boldsymbol{f}}(\boldsymbol{\mu}_M)$$

Trajectories

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K] \in \mathbb{R}^{p \times K}$$

Intro: Parametrized systems

Consider time-invariant system with polynomial nonlinear terms

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \sum_{i=1}^{\ell} \boldsymbol{A}_i \boldsymbol{x}_k^i + \boldsymbol{B}\boldsymbol{u}_k\,, \qquad k = 0, \ldots, K-1 \end{aligned}$$

Parameters

- Infer models $\hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_1), \ldots, \hat{\boldsymbol{f}}(\cdot,\cdot;\boldsymbol{\mu}_M)$ at parameters

$$\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M \in \mathcal{D}$$

- For new $\boldsymbol{\mu} \in \mathcal{D}$, interpolate operators of [Amsallem et al., 2008], [Degroote et al., 2010]

$$\hat{\boldsymbol{f}}(\boldsymbol{\mu}_1), \ldots, \hat{\boldsymbol{f}}(\boldsymbol{\mu}_M)$$

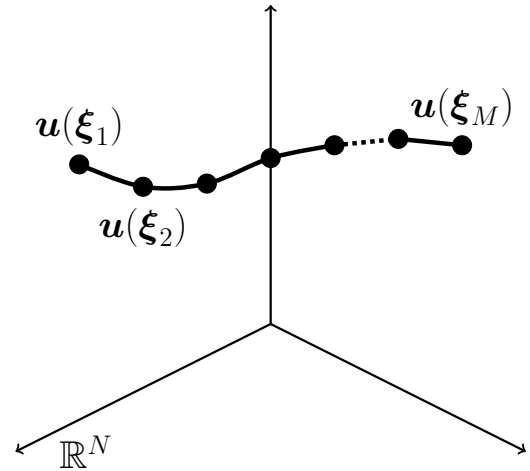
Trajectories

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K] \in \mathbb{R}^{p \times K}$$

Intro: Classical (intrusive) model reduction

Given full model f, construct reduced $\tilde{f}$ via projection



1. Construct n-dim. basis $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$

- Proper orthogonal decomposition (POD)
- Interpolatory model reduction
- Reduced basis method (RBM), ...

2. Project full-model operators $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell, \boldsymbol{B}$ onto reduced space, e.g.,

$$\tilde{\boldsymbol{A}}_i = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{A}_i}^{N \times N^i} (\boldsymbol{V} \otimes \cdots \otimes \boldsymbol{V})}_{n \times n^i}, \qquad \tilde{\boldsymbol{B}} = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{B}}^{N \times p}}_{n \times p}$$

3. Construct reduced model

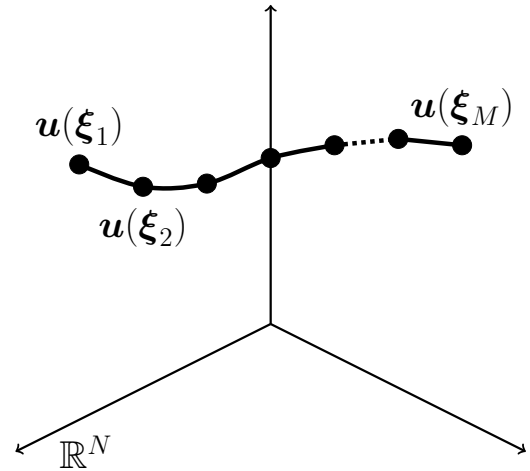
$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k^i + \tilde{\boldsymbol{B}} \boldsymbol{u}_k \,, \qquad k = 0, \ldots, K-1$$

with $n \ll N$ and $\| \boldsymbol{V} \tilde{\boldsymbol{x}}_k - \boldsymbol{x}_k \|$ small in appropriate norm

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]

Intro: Classical (intrusive) model reduction

Given full model f, construct reduced $\tilde{f}$ via projection



1. **Construct n-dim. basis $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$**
   - Proper orthogonal decomposition (POD)
   - Interpolatory model reduction
   - Reduced basis method (RBM), ...

2. **Project full-model operators $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell, \boldsymbol{B}$ onto reduced space, e.g.,**

$$\tilde{\boldsymbol{A}}_i = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{A}_i}^{N \times N^i} (\boldsymbol{V} \otimes \cdots \otimes \boldsymbol{V})}_{n \times n^i}, \qquad \tilde{\boldsymbol{B}} = \underbrace{\boldsymbol{V}^T \overbrace{\boldsymbol{B}}^{N \times p}}_{n \times p}$$

3. **Construct reduced model**

$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k^i + \tilde{\boldsymbol{B}} \boldsymbol{u}_k\,, \qquad k = 0, \ldots, K-1$$

with $n \ll N$ and $\|\boldsymbol{V}\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_k\|$ small in appropriate norm

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]

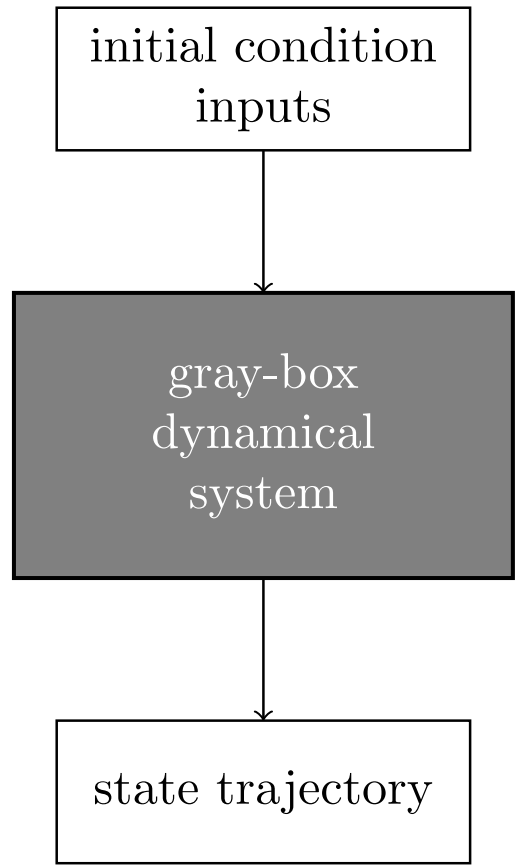
Our approach: Learn reduced models from data

Sample (gray-box) high-dimensional system with inputs

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_0 & \cdots & \boldsymbol{u}_{K-1} \end{bmatrix}$$

to obtain trajectory

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_0 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_K \\ | & | & & | \end{bmatrix}$$



Learn model $\hat{f}$ from data $\boldsymbol{U}$ and $\boldsymbol{X}$

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1} &= \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) \\ &= \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i \boldsymbol{x}_k^i + \hat{\boldsymbol{B}} \boldsymbol{u}_k \,, \qquad k = 0, \ldots, K-1 \end{aligned}$$

Intro: Literature overview

System identification [Ljung, 1987], [Viberg, 1995], [Kramer, Gugercin, 2016], ...

Learning in frequency domain [Antoulas, Anderson, 1986], [Lefteriu, Antoulas, 2010], [Antoulas, 2016], [Gustavsen, Semlyen, 1999], [Drmac, Gugercin, Beattie, 2015], [Antoulas, Gosea, Ionita, 2016], [Gosea, Antoulas, 2018], [Benner, Goyal, Van Dooren, 2019], ...

Learning from time-domain data (output and state trajectories)

- Time series analysis (V)AR models, [Box et al., 2015], [Aicher et al., 2018, 2019], ...
- Learning models with dynamic mode decomposition [Schmid et al., 2008], [Rowley et al., 2009], [Proctor, Brunton, Kutz, 2016], [Benner, Himpe, Mitchell, 2018], ...
- Sparse identification [Brunton, Proctor, Kutz, 2016], [Schaeffer et al, 2017, 2018], ...
- Deep networks [Raissi, Perdikaris, Karniadakis, 2017ab], [Qin, Wu, Xiu, 2019], ...
- Bounds for LTI systems [Campi et al, 2002], [Vidyasagar et al, 2008], ...

Correction and data-driven closure modeling

- Closure modeling [Chorin, Stinis, 2006], [Oliver, Moser, 2011], [Parish, Duraisamy, 2015], [Iliescu et al, 2018, 2019], ...
- Higher order dynamic mode decomposition [Le Clainche and Vega, 2017], [Champion et al., 2018]

OpInf: Fitting low-dim model to trajectories

1. Construct POD (PCA) basis of dimension $n \ll N$

$$\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$$

2. Project state trajectory onto the reduced space

$$\breve{\boldsymbol{X}} = \boldsymbol{V}^T \boldsymbol{X} = [\breve{\boldsymbol{x}}_1, \cdots, \breve{\boldsymbol{x}}_K] \in \mathbb{R}^{n \times K}$$

3. Find operators $\hat{\boldsymbol{A}}_1, \ldots, \hat{\boldsymbol{A}}_\ell, \hat{\boldsymbol{B}}$ such that

$$\breve{\boldsymbol{x}}_{k+1} \approx \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i \breve{\boldsymbol{x}}_k^i + \hat{\boldsymbol{B}} \boldsymbol{u}_k, \qquad k = 0, \cdots, K-1$$

by minimizing the residual in Euclidean norm

$$\min_{\hat{\boldsymbol{A}}_1, \ldots, \hat{\boldsymbol{A}}_\ell, \hat{\boldsymbol{B}}} \sum_{k=0}^{K-1} \left\| \breve{\boldsymbol{x}}_{k+1} - \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i \breve{\boldsymbol{x}}_k^i - \hat{\boldsymbol{B}} \boldsymbol{u}_k \right\|_2^2$$

[P., Willcox, *Data driven operator inference for nonintrusive projection-based model reduction*; Computer Methods in Applied Mechanics and Engineering, 306:196-215, 2016]

OpInf: Inferred operators

Fit operators $\hat{\boldsymbol{A}}_1, \ldots, \hat{\boldsymbol{A}}_\ell, \hat{\boldsymbol{B}}$ by solving least-squares problem

$$\min_{\hat{\boldsymbol{A}}_1, \ldots, \hat{\boldsymbol{A}}_\ell, \hat{\boldsymbol{B}}} \sum_{k=0}^{K-1} \left\| \breve{\boldsymbol{x}}_{k+1} - \sum_{i=1}^{\ell} \boldsymbol{A}_i \breve{\boldsymbol{x}}_k^i - \hat{\boldsymbol{B}} \boldsymbol{u}_k \right\|_2^2$$

- Transform into n independent least-squares problem
- Can be solved efficiently with standard solvers

Recover "intrusive operators" [P., Willcox, 2016]

- Need sufficient data
- Need that $\boldsymbol{V}$ spans $\mathbb{R}^N$ for $n \to N$
- Of little practical value because no rate of convergence

If $\ell = 1$ (linear), then the inferred operators are the DMD operators

[Schmid et al., 2008], [Rowley et al., 2009], [Tu et al., 2013], [Chung, Chung, 2014], [Proctor et al., 2016], [Xie, Mohebujjaman, Rebholz, Iliescu, 2017]

An experiment

- Generate a 10×10 matrix $\mathcal{A}$ with eigenvalues logarithmically spaced between -10^{-1} and -10^{-2}
- Gives rise to the time-continuous autonomous system $\dot{x}(t) = \mathcal{A}x(t)$
- Discretize in time with Runge-Kutta 4th order scheme and $\delta t = 1$ and $K = 100$ time steps
- Obtain time-discrete system $\boldsymbol{x}_{k+1} = \boldsymbol{A}_1 \boldsymbol{x}_k\,, \qquad k = 1, \ldots, K$
- Set $\boldsymbol{x}_0 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{10}$ and generate trajectory

$$\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]$$

- Set basis matrix

$$\boldsymbol{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{10 \times 2}$$

- Intrusive model reduction with Galerkin projection gives $\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{A}}_1 \tilde{\boldsymbol{x}}_k$ with $\tilde{\boldsymbol{A}}_1 = \boldsymbol{V}^T \boldsymbol{A}_1 \boldsymbol{V}$

An experiment (cont'd)

- Project trajectory $\check{\boldsymbol{X}} = \boldsymbol{V}^T \boldsymbol{X}$
- Fit OpInf model $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{A}}_1 \hat{\boldsymbol{x}}_k$
- Test models with initial condition $\boldsymbol{x}_0^{\text{test}} = [1, 1, 0, \ldots, 0]^T \in \mathbb{R}^{10}$

Matlab experiment

https://github.com/pehersto/reproj

OpInf: Learning from projected trajectory

Fitting model to projected states

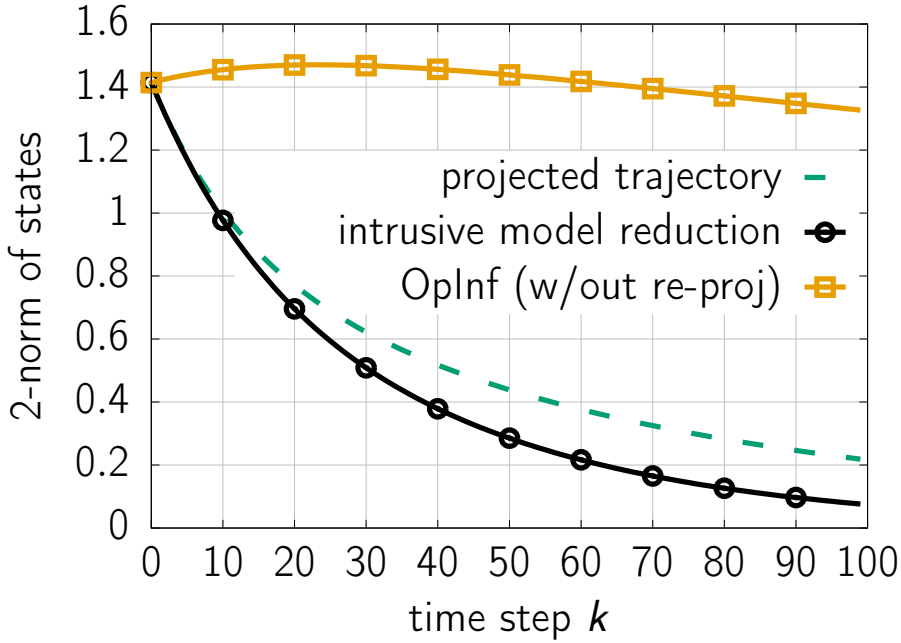
- We fit model to projected trajectory

$$\check{\boldsymbol{X}} = \boldsymbol{V}^T \boldsymbol{X}$$

- Would need $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_K]$ because

$$\sum_{k=0}^{K-1} \left\| \tilde{\boldsymbol{x}}_{k+1} - \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k^i - \tilde{\boldsymbol{B}} \boldsymbol{u}_k \right\|_2^2 = 0$$

- However, trajectory $\tilde{\boldsymbol{X}}$ unavailable



Thus, $\|\hat{\boldsymbol{f}} - \tilde{\boldsymbol{f}}\|$ small critically depends on $\|\check{\boldsymbol{X}} - \tilde{\boldsymbol{X}}\|$ being small

- Increase dimension n of reduced space to decrease $\|\check{\boldsymbol{X}} - \tilde{\boldsymbol{X}}\|$

  $\Rightarrow$ increases degrees of freedom in OpInf $\Rightarrow$ ill-conditioned
- Decrease dimension n to keep number of degrees of freedom low

  $\Rightarrow$ difference $\|\check{\boldsymbol{X}} - \tilde{\boldsymbol{X}}\|$ increases

OpInf: Closure of linear system

Consider autonomous linear system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k\,, \qquad \boldsymbol{x}_0 \in \mathbb{R}^N\,, \quad k = 0, \ldots, K-1$$

- Split $\mathbb{R}^N$ into $\mathcal{V} = \text{span}(\boldsymbol{V})$ and $\mathcal{V}_\perp = \text{span}(\boldsymbol{V}_\perp)$

$$\mathbb{R}^N = \mathcal{V} \oplus \mathcal{V}_\perp$$

- Split state

$$\boldsymbol{x}_k = \boldsymbol{V}\underbrace{\boldsymbol{V}^T\boldsymbol{x}_k}_{\boldsymbol{x}_k^{\parallel}} + \boldsymbol{V}_\perp\underbrace{\boldsymbol{V}_\perp^T\boldsymbol{x}_k}_{\boldsymbol{x}_k^{\perp}}$$

Represent system as

$$\begin{aligned}\boldsymbol{x}_{k+1}^{\parallel} &= \boldsymbol{A}_{11}\boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_{12}\boldsymbol{x}_k^{\perp}\\ \boldsymbol{x}_{k+1}^{\perp} &= \boldsymbol{A}_{21}\boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_{22}\boldsymbol{x}_k^{\perp}\end{aligned}$$

with operators

$$\boldsymbol{A}_{11} = \underbrace{\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}}_{=\tilde{\boldsymbol{A}}}, \quad \boldsymbol{A}_{12} = \boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}_\perp\,, \boldsymbol{A}_{21} = \boldsymbol{V}_\perp^T\boldsymbol{A}\boldsymbol{V}\,, \quad \boldsymbol{A}_{22} = \boldsymbol{V}_\perp^T\boldsymbol{A}\boldsymbol{V}_\perp$$

[Givon, Kupferman, Stuart, 2004], [Chorin, Stinis, 2006] [Parish, Duraisamy, 2017]

OpInf: Closure term as a non-Markovian term

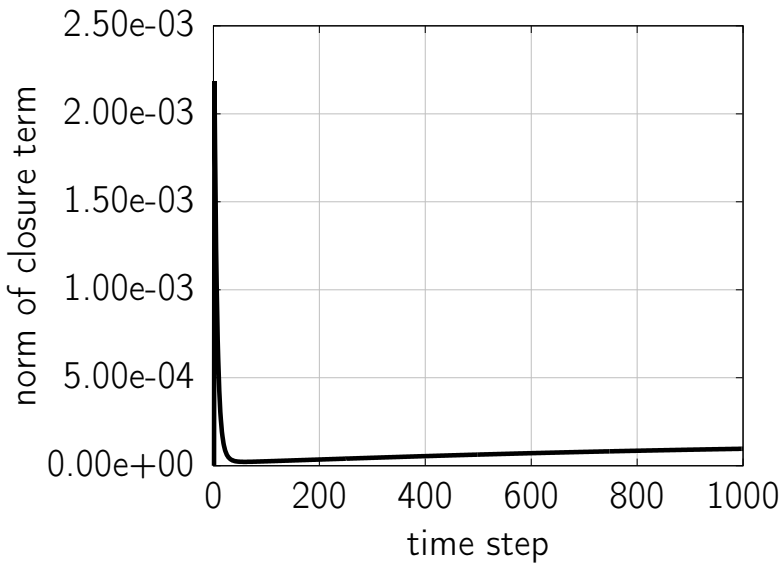
Projected trajectory $\check{\boldsymbol{X}}$ mixes dynamics in $\mathcal{V}$ and $\mathcal{V}_\perp$

$$\boldsymbol{V}^T \boldsymbol{x}_{k+1} = \check{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel} = \boldsymbol{A}_{11}\boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_{12}\boldsymbol{x}_k^{\perp}$$

Mori-Zwanzig formalism gives [Givon, Kupferman, Stuart, 2004], [Chorin, Stinis, 2006]

$$\begin{aligned}\boldsymbol{V}^T \boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel} &= \boldsymbol{A}_{11}\boldsymbol{x}_k^{\parallel} + \boldsymbol{A}_{12}\boldsymbol{x}_k^{\perp} \\ &= \boldsymbol{A}_{11}\boldsymbol{x}_k^{\parallel} + \sum_{j=1}^{k-1} \boldsymbol{A}_{22}^{k-j-1}\boldsymbol{A}_{21}\boldsymbol{x}_j^{\parallel} + \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{k-1}\boldsymbol{x}_0^{\perp}\end{aligned}$$

Non-Markovian (memory) term models *unobserved* dynamics



ReProj: Handling non-Markovian dynamics

Ignore non-Markovian dynamics

- Have significant impact on model accuracy (much more than in classical model reduction?)
- Guarantees on models?

Fit models with different forms to capture non-Markovian dynamics

- Length of memory (support of kernel) typically unknown
- Time-delay embedding increase dimension of reduced states, which is what we want to reduce
- Model reduction (theory) mostly considers Markovian reduced models

Our approach: Control length of memory when sampling trajectories

- Set length of memory to 0 for sampling Markovian dynamics
- Increase length of memory in a controlled way (lag is known)
- Modify the sampling scheme, instead of learning step
- Emphasizes importance of generating the "right" data

ReProj: Avoiding closure

Mori-Zwanzig formalism explains projected trajectory as

$$\boldsymbol{V}^T \boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1}^{\parallel} = \underbrace{\boldsymbol{A}_{11} \boldsymbol{x}_k^{\parallel}}_{\text{reduced model}} + \underbrace{\sum_{j=1}^{k-1} \boldsymbol{A}_{22}^{k-j-1} \boldsymbol{A}_{21} \boldsymbol{x}_j^{\parallel}}_{\text{memory}} + \underbrace{\boldsymbol{A}_{12} \boldsymbol{A}_{22}^{k-1} \boldsymbol{x}_0^{\perp}}_{\text{noise}}$$

Sample Markovian dynamics by setting memory and noise to 0

- Set $\boldsymbol{x}_0 \in \mathcal{V}$, then noise is 0
- Take a single time step, then memory term is 0

Sample trajectory by re-projecting state of previous time step onto $\mathcal{V}$

Establishes "independence"

ReProj: Sampling with re-projection

Data sampling: Cancel non-Markovian terms via re-projection

1. Project initial condition $\boldsymbol{x}_0$ onto $\mathcal{V}$

$$\bar{\boldsymbol{x}}_0 = \boldsymbol{V}^T \boldsymbol{x}_0$$

2. Query high-dim. system for **a single time step** with $\boldsymbol{V}\bar{\boldsymbol{x}}_0$

$$\boldsymbol{x}_1 = \boldsymbol{f}(\boldsymbol{V}\bar{\boldsymbol{x}}_0, \boldsymbol{u}_0)$$

3. Re-project to obtain $\bar{\boldsymbol{x}}_1 = \boldsymbol{V}^T \boldsymbol{x}_1$
4. Query high-dim. system with re-projected initial condition $\boldsymbol{V}\bar{\boldsymbol{x}}_1$

$$\boldsymbol{x}_2 = \boldsymbol{f}(\boldsymbol{V}\bar{\boldsymbol{x}}_1, \boldsymbol{u}_1)$$

5. Repeat until end of time-stepping loop

Obtain trajectories

$$\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}}_0, \ldots, \bar{\boldsymbol{x}}_{K-1}], \qquad \bar{\boldsymbol{Y}} = [\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_K], \qquad \boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{K-1}]$$

[P., Sampling low-dimensional Markovian dynamics for pre-asymptotically recovering reduced models from data with operator inference. arXiv:1908.11233, 2019.]

ReProj: Operator inference with re-projection

Operator inference with re-projected trajectories

$$\min_{\hat{\boldsymbol{A}}_1,\ldots,\hat{\boldsymbol{A}}_\ell,\hat{\boldsymbol{B}}} \left\| \bar{\boldsymbol{Y}} - \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i \bar{\boldsymbol{X}}^i - \hat{\boldsymbol{B}}\boldsymbol{U} \right\|_F^2$$

Theorem (*Simplified*) Consider time-discrete system with polynomial nonlinear terms of maximal degree ℓ and linear input. If $K \geq \sum_{i=1}^{\ell} n^i + 2$ and matrix $[\bar{\boldsymbol{X}}, \boldsymbol{U}, \bar{\boldsymbol{X}}^2, \ldots, \bar{\boldsymbol{X}}^\ell]$ has full rank, then $\|\bar{\boldsymbol{X}} - \tilde{\boldsymbol{X}}\| = 0$ and thus $\hat{\boldsymbol{f}} = \tilde{\boldsymbol{f}}$ in the sense

$$\|\hat{\boldsymbol{A}}_1 - \tilde{\boldsymbol{A}}_1\|_F = \cdots = \|\hat{\boldsymbol{A}}_\ell - \tilde{\boldsymbol{A}}_\ell\|_F = \|\tilde{\boldsymbol{B}} - \hat{\boldsymbol{B}}\|_F = 0$$

- Pre-asymptotic guarantees, in contrast to learning from projected data
- Re-projection is a nonintrusive operation
- Requires querying high-dim. system twice
- Initial conditions remain "physically meaningful"

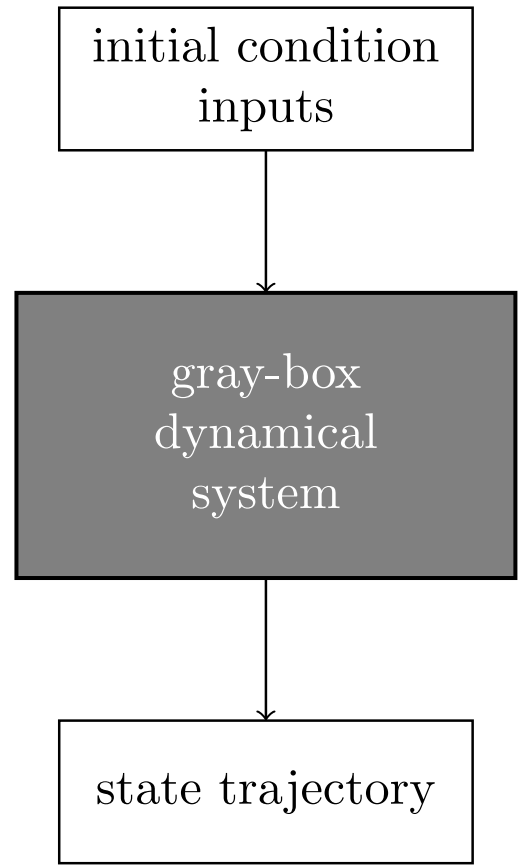
Provides a means to find model form

[P., Sampling low-dimensional Markovian dynamics for pre-asymptotically recovering reduced models from data with operator inference. arXiv:1908.11233, 2019.]

RePROJ: Queryable systems

Definition: Queryable systems [Uy, P., 2020]

A dynamical system is queryable, if the trajectory $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]$ *with* $K \geq 1$ *can be computed for initial condition* $\boldsymbol{x}_0 \in \mathcal{V}$ *and feasible input trajectory* $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K]$.



Details about *how* trajectories computed unnecessary

- Discretization (FEM, FD, FV, etc)
- Time-stepping scheme
- Time-step size
- In particular, neither explicit nor implicit access to operators required

Insufficient to have only data available

- Need to query system at re-projected states
- Similar requirement as for active learning

An example (cont'd)

Matlab experiment

https://github.com/pehersto/reproj

RePROJ: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

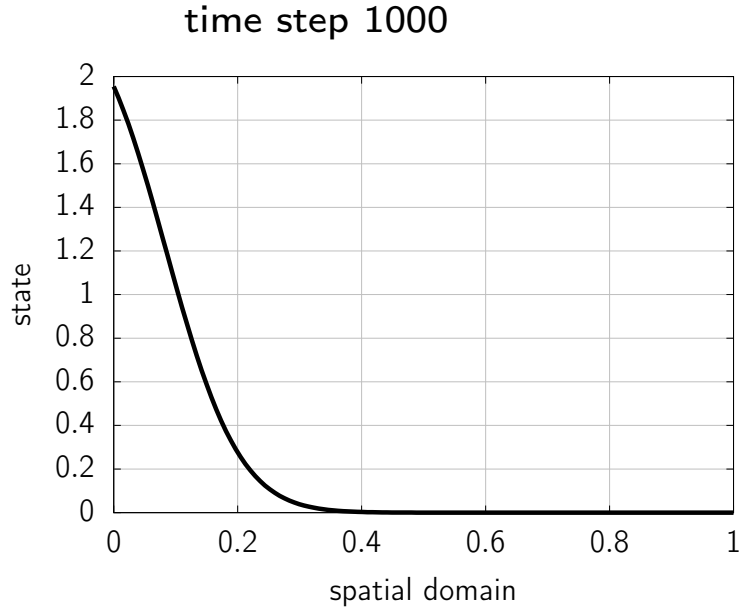
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

ReProj: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

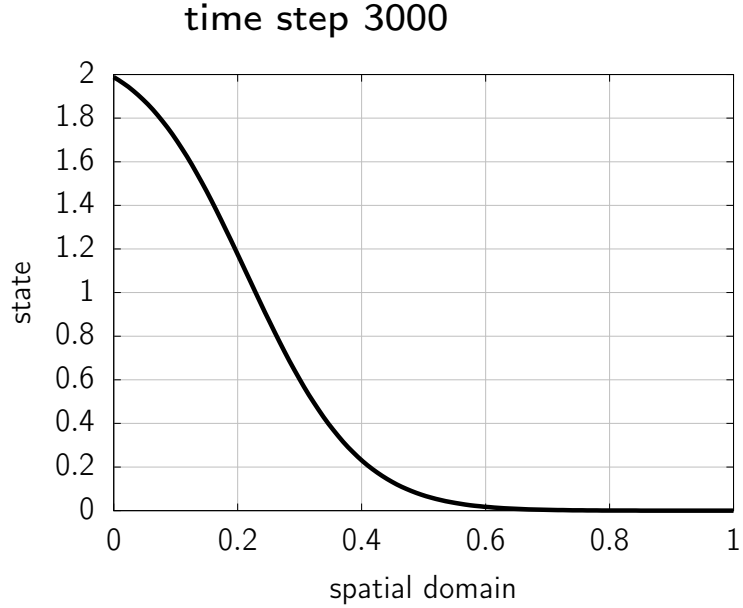
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

ReProj: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

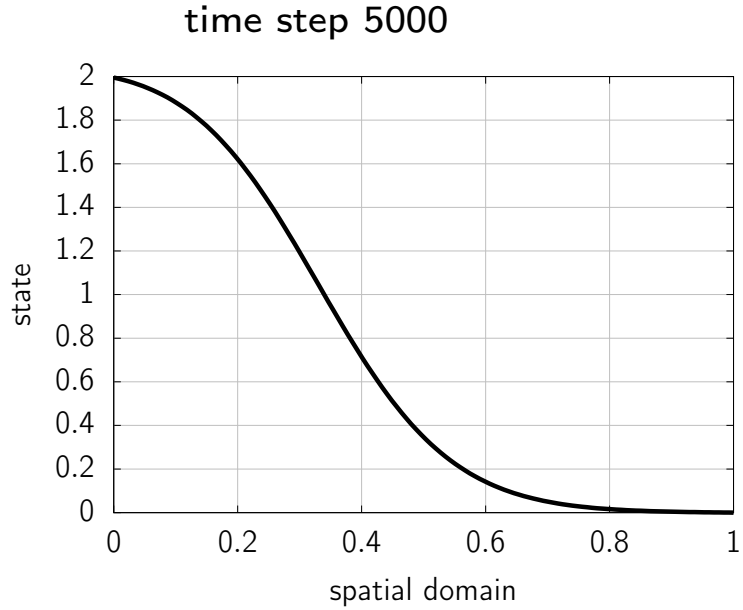
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

ReProj: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

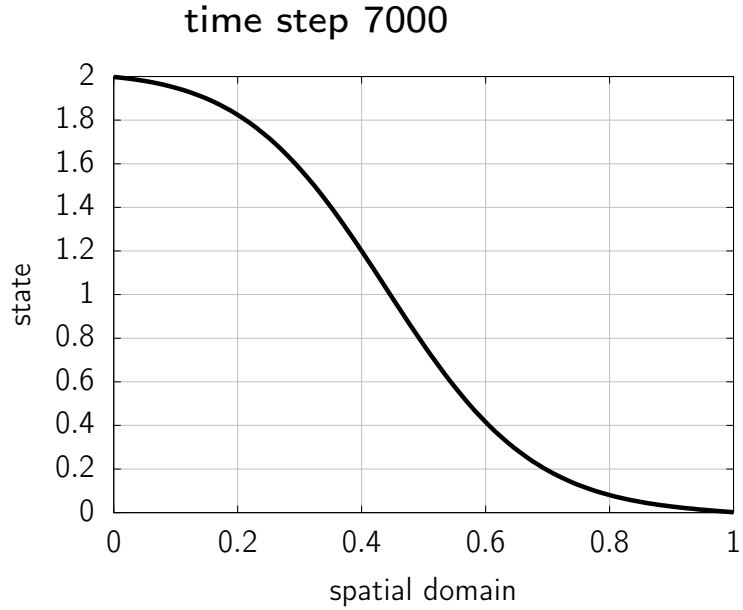
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

ReProj: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

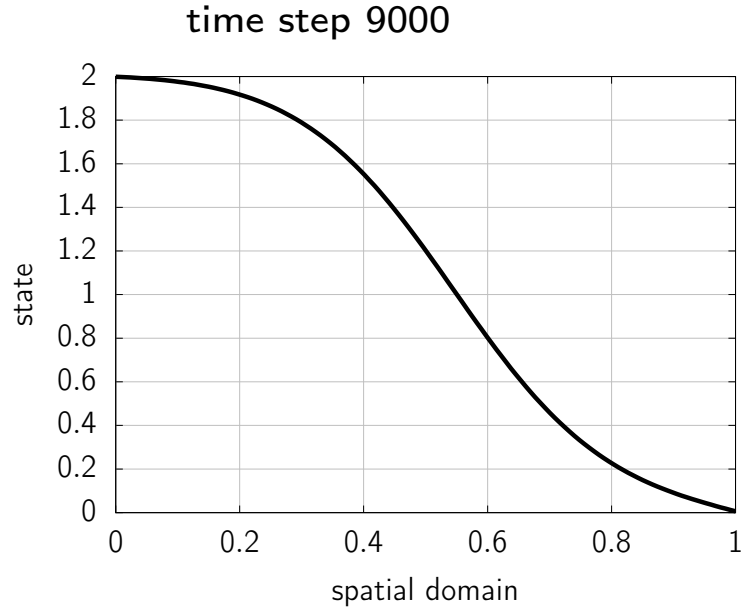
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

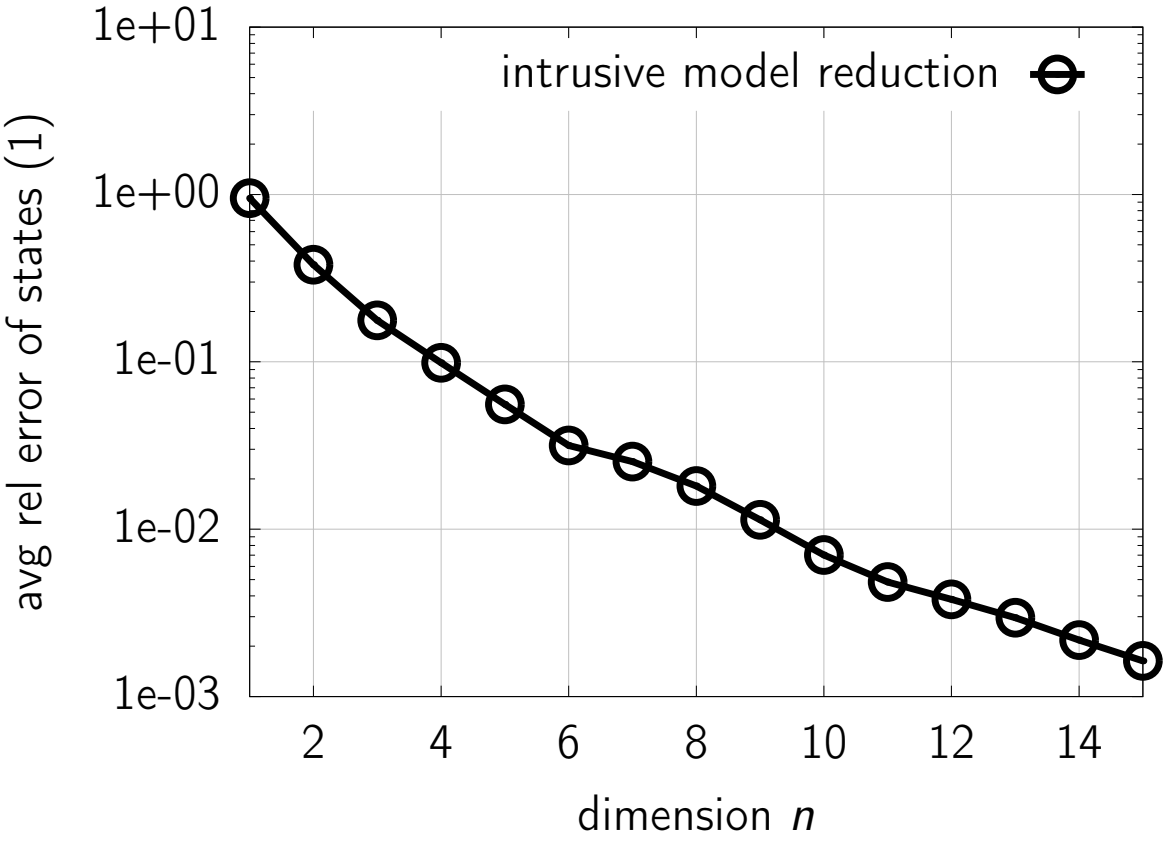
- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

ReProj: Burgers': Operator inference

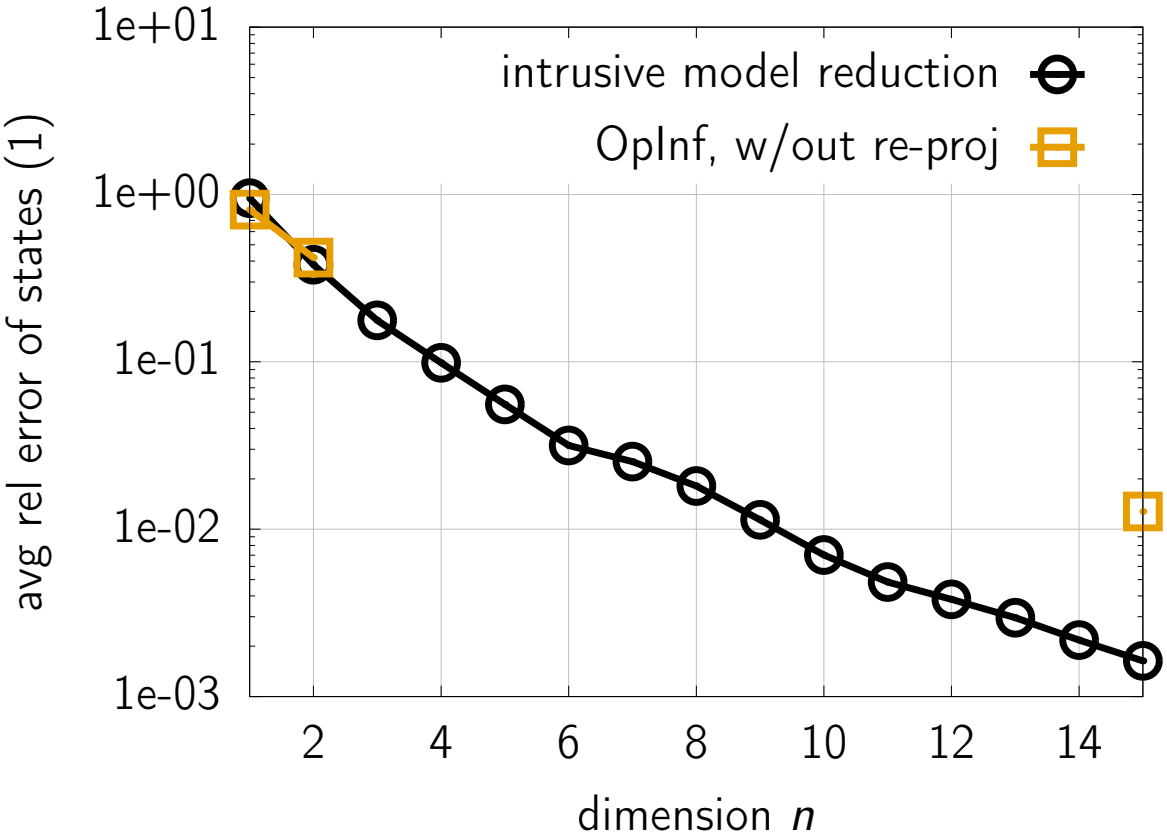


Error of reduced models at test data

- Inferring operators from projected data fails in this example
- Recover reduced model from re-projected data

(1)

ReProj: Burgers': Operator inference

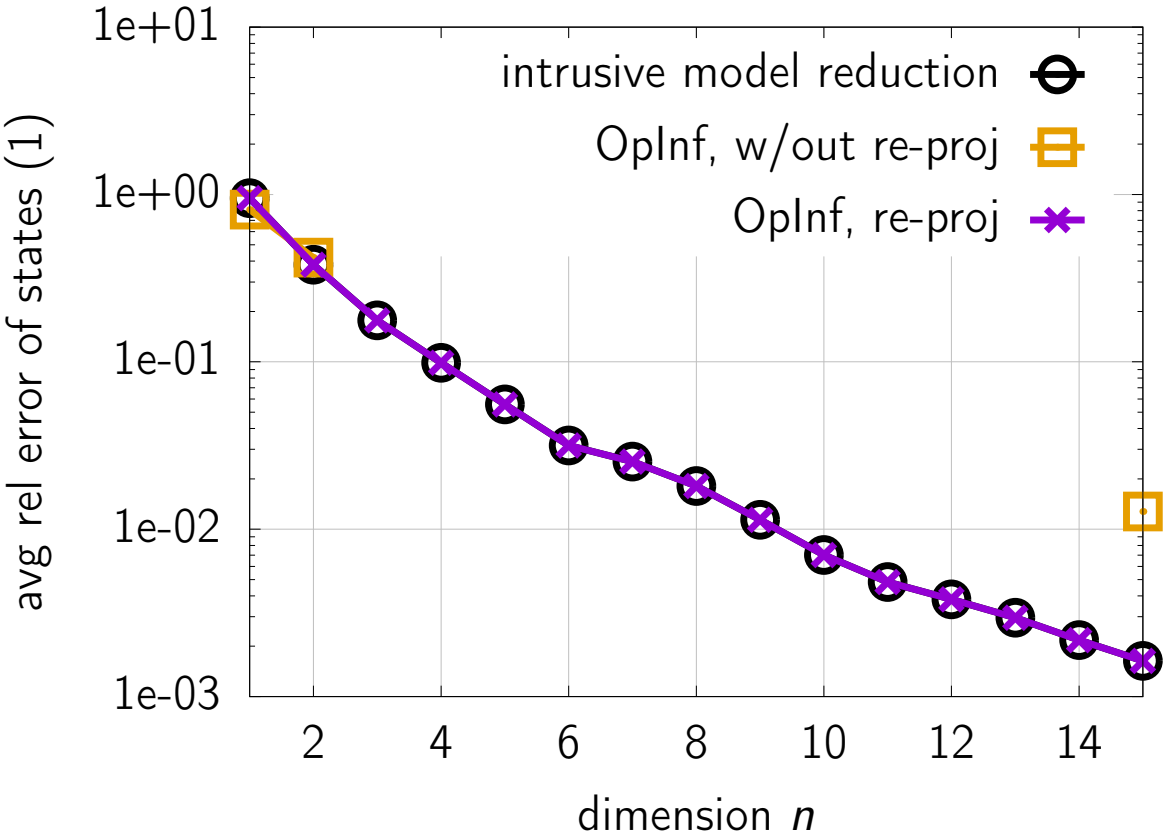


Error of reduced models at test data

- Inferring operators from projected data fails in this example
- Recover reduced model from re-projected data

(1)

ReProj: Burgers': Operator inference

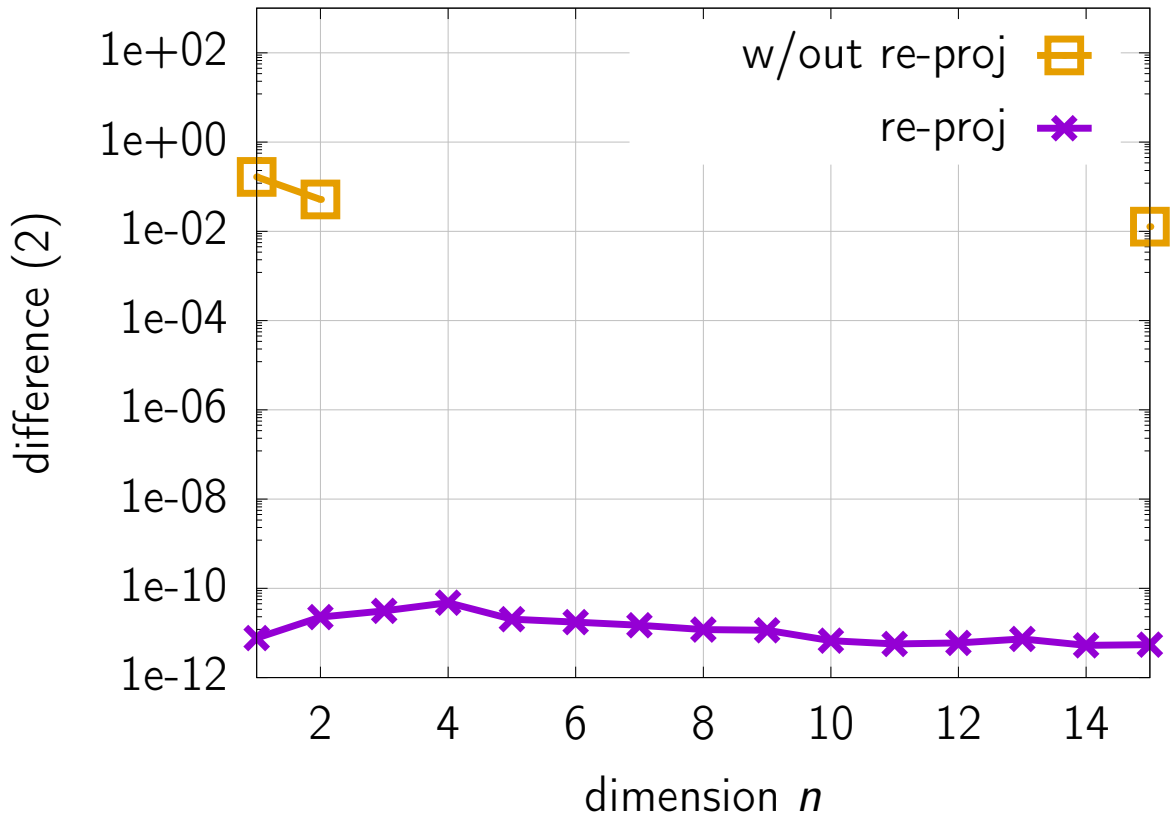


Error of reduced models at test data

- Inferring operators from projected data fails in this example
- Recover reduced model from re-projected data

(1)

ReProj: Burgers': Recovery



The difference between state trajectories

- Model from intrusive model reduction same as OpInf with re-proj.
- Model learned from state trajectories without re-projection differs

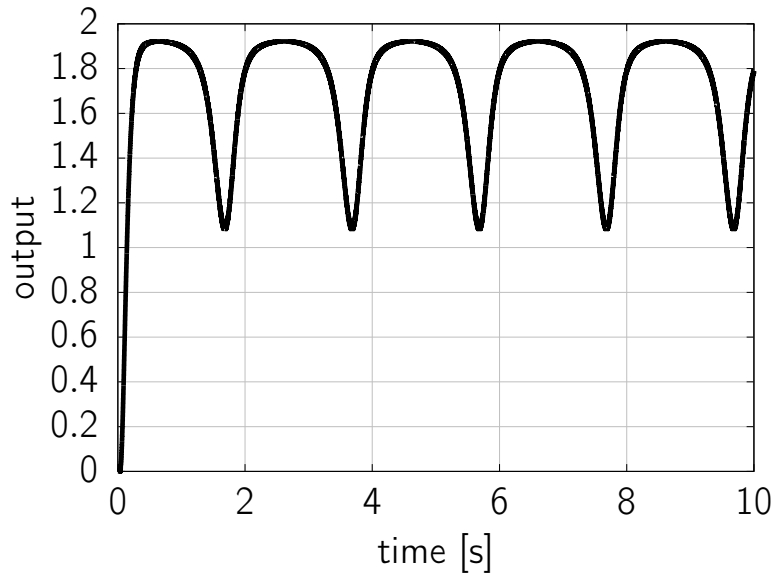
(2)

ReProj: Chafee: Chafee-Infante example

Chafee-Infante equation

$$\frac{\partial}{\partial t}x(\omega,t) + x^3(\omega,t) - \frac{\partial^2}{\partial \omega^2}x(\omega,t) - x(\omega,t) = 0$$

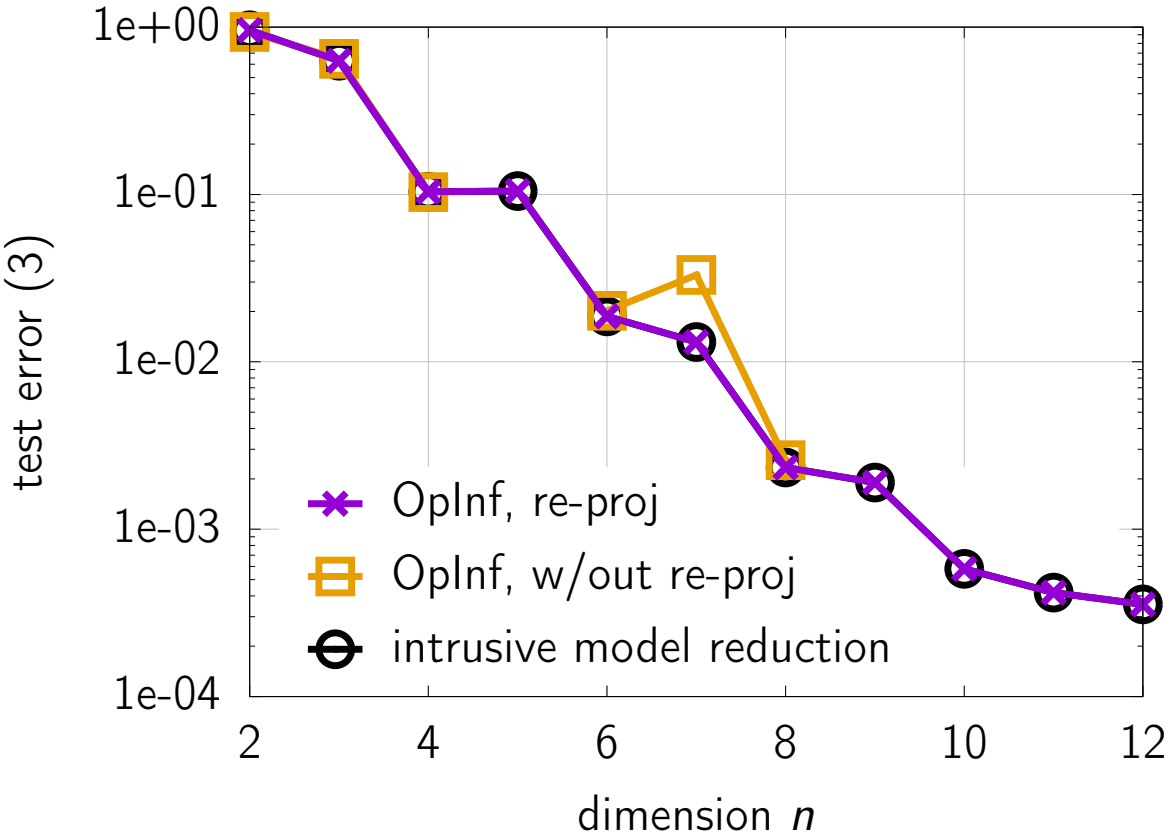
- Boundary conditions as in [Benner et al., 2018]
- Spatial domain $\omega \in [0,1]$
- Time domain $t \in [0,10]$
- Forward Euler with $\delta t = 10^{-4}$
- Cubic nonlinear term



Operator inference

- Infer operators from single trajectory corresponding to random inputs
- Test inferred model on oscillatory input

ReProj: Chafee: Recovery



Error of reduced models on test parameters

- Projected data leads to unstable inferred model
- Inference from data with re-projection shows stabler behavior

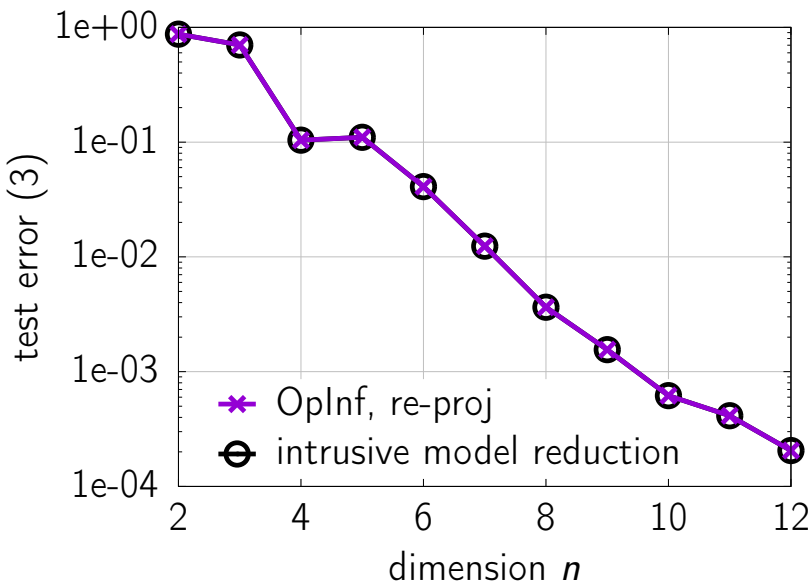
(3)

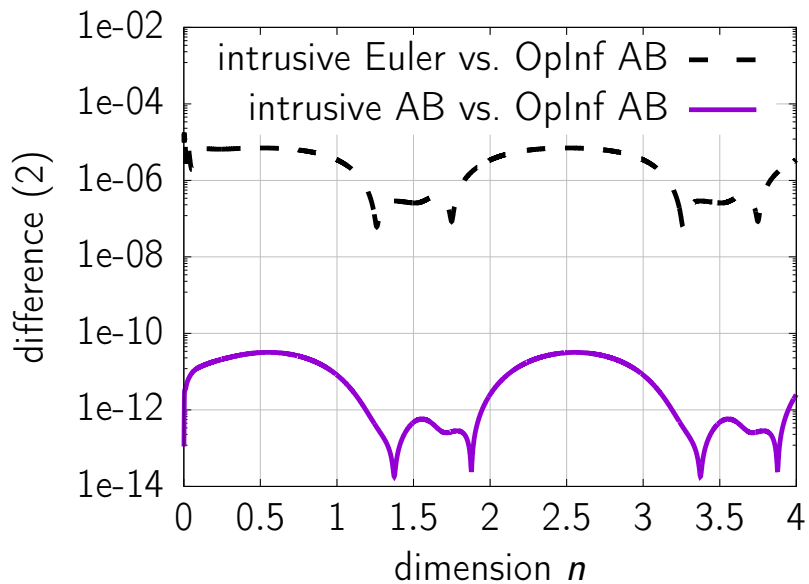
ReProj: AB2 time-stepping

Two-step Adams-Bashforth time stepping for cubic system

$$\boldsymbol{x}_{k+2} = \boldsymbol{x}_{k+1} + \frac{3}{2}\left((\boldsymbol{A}_1 - \boldsymbol{I})\boldsymbol{x}_{k+1} + \boldsymbol{A}_2\boldsymbol{x}_{k+1}^2 + \boldsymbol{A}_3\boldsymbol{x}_{k+1}^3 + \boldsymbol{B}\boldsymbol{u}_{k+1}\right) - \frac{1}{2}\left((\boldsymbol{A}_1 - \boldsymbol{I})\boldsymbol{x}_k + \boldsymbol{A}_2\boldsymbol{x}_k^2 + \boldsymbol{A}_3\boldsymbol{x}_k^3 + \boldsymbol{B}\boldsymbol{u}_k\right)$$

- The first time step is with explicit Euler
- Re-projection is applicable in the first time step





(a) Adams-Bashforth time stepping (b) time-stepping with Euler vs. Adams-Bashforth (AB)

Different point of view: Run ROM and learn ROM

Time-discrete dynamical system

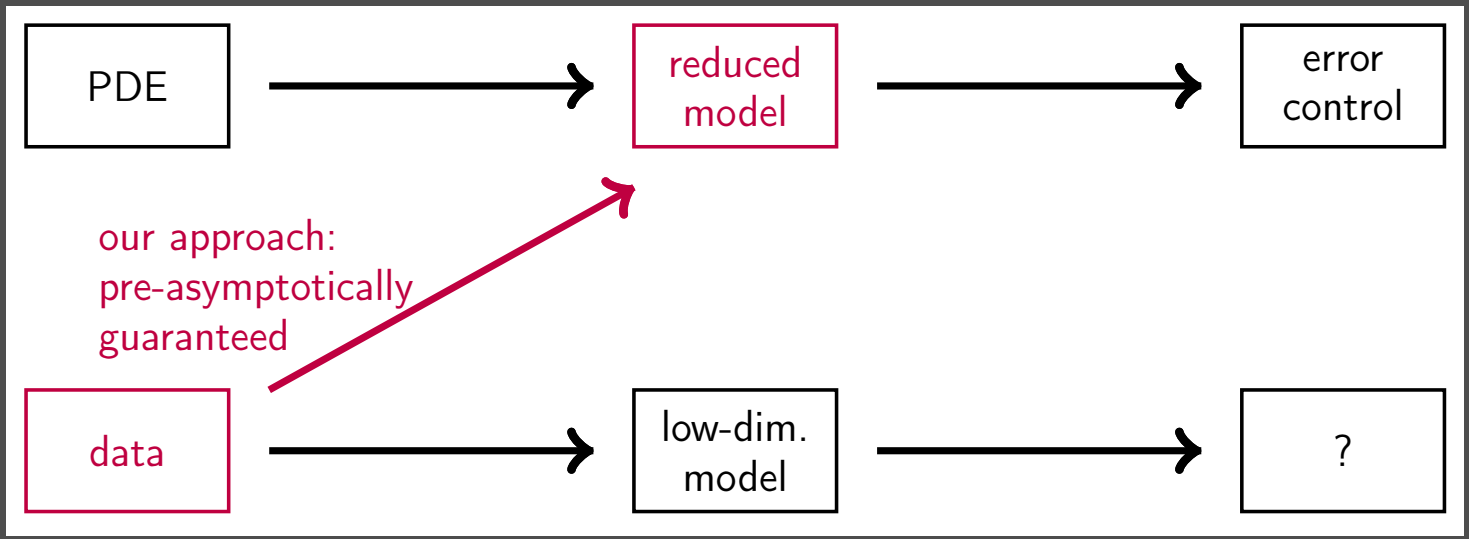
$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

Galerkin-reduced model with basis matrix $\boldsymbol{V}$

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{V}^T f(\boldsymbol{V}\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k)$$

Interpretation

- Data generation with re-projection solves the Galerkin-reduced model without explicitly assembling it
- Paves the way for extensions for other time-stepping schemes and Petrov-Galerkin projection


PDE
reduced model
error control
our approach:
pre-asymptotically
guaranteed
data
low-dim. model
?

Exercise in breakout rooms

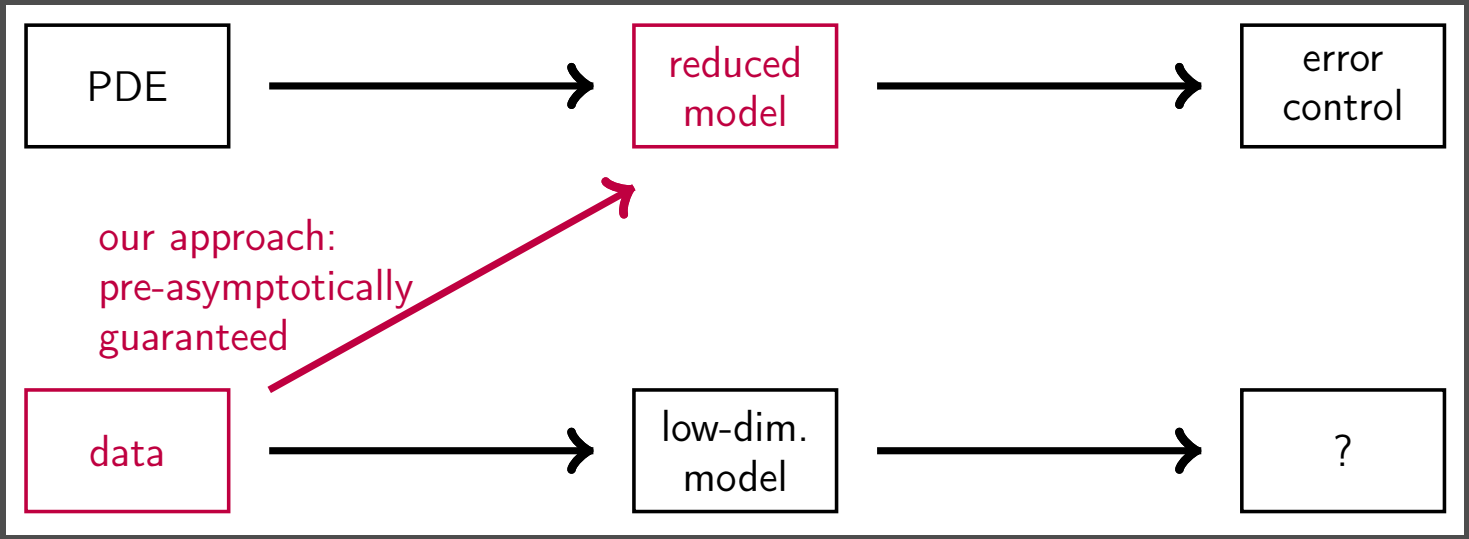
Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- **Error estimation of learned reduced models**
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Recap from last time



Error estimation in intrusive model reduction

Error estimation is major topic in intrusive model reduction

- Rigorously upper bound the error of reduced to full prediction
- Efficient in pre-asymptotic regime, i.e., for dimension n small
- Error estimators are key building blocks for constructing reduced models

Requirements

- Offline/online splitting to compute error estimators with costs independent of full-model dimension
- Estimators should not overestimate error by too much
- All constants and quantities need to be computable

Error estimation in model reduction

- Martin A. Grepl and Anthony T. Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, ESAIM: M2AN, 39 1 (2005) 157-181
- Veroy, K. and Patera, A.T. (2005), Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds. Int. J. Numer. Meth. Fluids, 47: 773-788
- G Rozza, DBP Huynh, AT Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, Archives of Computational Methods in Engineering 15 (3), 1
- JS Hesthaven, G Rozza, B Stamm, Certified reduced basis methods for parametrized partial differential equations, Springer, 2016

ErrEst: Error of linear time-invariant systems

Consider LTI system [Grepl, Patera, 2005], [Haasdonk, Ohlberger, 2009]

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_{k+1}$$

Reduced model (w/out intrusive model reduction)

$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}_k + \tilde{\boldsymbol{B}}\boldsymbol{u}_{k+1}$$

Residual

$$\boldsymbol{r}_{k+1} = \boldsymbol{A}\boldsymbol{V}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_{k+1} - \boldsymbol{V}\tilde{\boldsymbol{x}}_{k+1}$$

State error is

$$\boldsymbol{x}_k - \boldsymbol{V}\tilde{\boldsymbol{x}}_k = \boldsymbol{A}^k(\boldsymbol{x}_0 - \boldsymbol{V}\tilde{\boldsymbol{x}}_0) + \sum_{i=0}^{k-1} \boldsymbol{A}^{k-l-1}\boldsymbol{r}_{l+1}$$

[Haasdonk, Ohlberger, Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition, 2009], [Grepl, Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, 2005]

ErrEst: Error of linear time-invariant systems (cont'd)

Define the quantity

$$\Delta_k^x(c_0, \ldots, c_k; \boldsymbol{x}_0, \boldsymbol{U}) = c_0 \|\boldsymbol{x}_0 - \boldsymbol{V}\tilde{\boldsymbol{x}}_0\|_2 + \sum_{i=0}^{k-1} c_{l+1} \|\boldsymbol{r}_{l+1}\|_2$$

- Initial condition $\boldsymbol{x}_0$
- Input trajectory $\boldsymbol{U} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$
- Constants $c_0, \ldots, c_k \in \mathbb{R}$

Bounding the norm of the state error

$$\begin{aligned}\|\boldsymbol{x}_k - \boldsymbol{V}\tilde{\boldsymbol{x}}_k\|_2 \leq & \Delta_k^x(\|\boldsymbol{A}^k\|_2, \ldots, \|\boldsymbol{A}^0\|_2; \boldsymbol{x}_0, \boldsymbol{U}) \\ = & \|\boldsymbol{A}^k\|_2 \|\boldsymbol{x}_0 - \boldsymbol{V}\tilde{\boldsymbol{x}}_0\|_2 + \sum_{i=0}^{k-1} \|\boldsymbol{A}^{k-l-1}\|_2 \|\boldsymbol{r}_{l+1}\|_2\end{aligned}$$

[Haasdonk, Ohlberger, Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition, 2009], [Grepl, Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, 2005]

ErrEst: Quantities

Bounding the norm of the state error

$$\begin{aligned}\|\boldsymbol{x}_k - \boldsymbol{V}\tilde{\boldsymbol{x}}_k\|_2 \leq & \Delta_k^x(\|\boldsymbol{A}^k\|_2, \ldots, \|\boldsymbol{A}^0\|_2; \boldsymbol{x}_0, \boldsymbol{U}) \\ = & \|\boldsymbol{A}^k\|_2\|\boldsymbol{x}_0 - \boldsymbol{V}\tilde{\boldsymbol{x}}_0\|_2 + \sum_{i=0}^{k-1} \|\boldsymbol{A}^{k-l-1}\|_2\|\boldsymbol{r}_{l+1}\|_2\end{aligned}$$

- Need either $\|\boldsymbol{A}^i\|_2$ or at least $\|\boldsymbol{A}^i\|_2 \leq C$ for $i = 0, \ldots, k$
- Need residual norm $\|\boldsymbol{r}_i\|_2$ for $i = 1, \ldots, k$
- We know the initial condition $\boldsymbol{x}_0$ and $\boldsymbol{V}$ and so can compute $\|\boldsymbol{x}_0 - \boldsymbol{V}\tilde{\boldsymbol{x}}_0\|$

Residual norm [Haasdonk, Ohlberger, 2009]

$$\begin{aligned}\|\boldsymbol{r}_k\|_2^2 = \boldsymbol{r}_k^T\boldsymbol{r}_k = & \hat{\boldsymbol{x}}_k^T \underbrace{\boldsymbol{V}^T\boldsymbol{A}_1^T\boldsymbol{A}_1\boldsymbol{V}}_{\boldsymbol{M_1}} \hat{\boldsymbol{x}}_k + u_k \underbrace{\boldsymbol{B}^T\boldsymbol{B}}_{\boldsymbol{M_2}} u_k + \hat{\boldsymbol{x}}_{k+1}\boldsymbol{V}^T\boldsymbol{V}\hat{\boldsymbol{x}}_{k+1} \\ & + 2u_k^T \underbrace{\boldsymbol{B}^T\boldsymbol{A}_1\boldsymbol{V}}_{\boldsymbol{M_3}} \hat{\boldsymbol{x}}_k - 2\hat{\boldsymbol{x}}_{k+1}^T\hat{\boldsymbol{A}}_1\hat{\boldsymbol{x}}_{k+1} - 2\hat{\boldsymbol{x}}_{k+1}\hat{\boldsymbol{B}}u_k\end{aligned}$$

ErrEst: Error estimation for learned models

Assumptions*: Linear time-invariant system

- Derive reduced model with operator inference and re-projection
- Requires full residual of reduced-model states in training phase

Error estimation based on [Haasdonk, Ohlberger, 2009]

- Residual at time step k

$$\boldsymbol{r}_k = \boldsymbol{A}_1 \boldsymbol{V} \hat{\boldsymbol{x}}_k + \boldsymbol{B} u_k - \boldsymbol{V} \hat{\boldsymbol{x}}_{k+1}$$

- Bound on state error if initial condition in $\operatorname{span}\{\boldsymbol{V}\}$

$$\|\boldsymbol{x}_k - \boldsymbol{V}\hat{\boldsymbol{x}}_k\|_2 \leq C_1 \left(\sum_{i=1}^{k-1} \|\boldsymbol{r}_k\|_2 \right)$$

- Offline/online splitting of computing residual norm $\|\boldsymbol{r}_k\|_2$

$$\begin{aligned}\|\boldsymbol{r}_k\|_2^2 =& \hat{\boldsymbol{x}}_k^T \underbrace{\boldsymbol{V}^T \boldsymbol{A}_1^T \boldsymbol{A}_1 \boldsymbol{V}}_{\boldsymbol{M_1}} \hat{\boldsymbol{x}}_k + u_k \underbrace{\boldsymbol{B}^T \boldsymbol{B}}_{\boldsymbol{M_2}} u_k + \hat{\boldsymbol{x}}_{k+1} \boldsymbol{V}^T \boldsymbol{V} \hat{\boldsymbol{x}}_{k+1} \\ &+ 2 u_k^T \underbrace{\boldsymbol{B}^T \boldsymbol{A}_1 \boldsymbol{V}}_{\boldsymbol{M_3}} \hat{\boldsymbol{x}}_k - 2 \hat{\boldsymbol{x}}_{k+1}^T \hat{\boldsymbol{A}}_1 \hat{\boldsymbol{x}}_{k+1} - 2 \hat{\boldsymbol{x}}_{k+1} \hat{\boldsymbol{B}} u_k\end{aligned}$$

ErrEst: Learning error operators from data

From [Haasdonk, Ohlberger, 2009] have

$$\|\boldsymbol{r}_k\|_2^2 = \hat{\boldsymbol{x}}_k^T \underbrace{\boldsymbol{V}^T \boldsymbol{A}_1^T \boldsymbol{A}_1 \boldsymbol{V}}_{\boldsymbol{M}_1} \hat{\boldsymbol{x}}_k + u_k \underbrace{\boldsymbol{B}^T \boldsymbol{B}}_{\boldsymbol{M}_2} u_k + \hat{\boldsymbol{x}}_{k+1} \boldsymbol{V}^T \boldsymbol{V} \hat{\boldsymbol{x}}_{k+1}$$
$$+ 2u_k^T \underbrace{\boldsymbol{B}^T \boldsymbol{A}_1 \boldsymbol{V}}_{\boldsymbol{M}_3} \hat{\boldsymbol{x}}_k - 2\hat{\boldsymbol{x}}_{k+1}^T \hat{\boldsymbol{A}}_1 \hat{\boldsymbol{x}}_{k+1} - 2\hat{\boldsymbol{x}}_{k+1} \hat{\boldsymbol{B}} u_k$$

Query system at training inputs to compute residual trajectories

$$\boldsymbol{R} = \begin{bmatrix} | & | & & | \\ \boldsymbol{r}_1 & \boldsymbol{r}_2 & \dots & \boldsymbol{r}_K \\ | & | & & | \end{bmatrix}$$

Learn quantities $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ via operator inference

- Fit error operators $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ to residual trajectories
- Least-squares problem with unique solution that is $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$

Obtain *certified* reduced models from data alone

[Uy, P., *Pre-asymptotic error bounds for low-dimensional models learned from systems governed by linear parabolic partial differential equations with control inputs, in preparation*, 2020]

ErrEst: Probabilistic bounds of constants

For $l \in \mathbb{N}$, let $\boldsymbol{\Theta}^{(l)} = \boldsymbol{A}^l \boldsymbol{Z}_1$ where $\boldsymbol{Z}_1 \sim N(\boldsymbol{0}_{N\times 1}, \boldsymbol{I}_N)$ so that $\boldsymbol{\Theta}^{(l)}$ is an N-dimensional Gaussian random vector with mean zero and covariance $\boldsymbol{A}^l(\boldsymbol{A}^l)^T$.

Suppose that $\{\boldsymbol{\Theta}_i^{(l)}\}_{i=1}^M$ are $M \in \mathbb{N}$ independent and identically distributed N-dimensional random vectors with the same law as $\boldsymbol{\Theta}^{(l)}$. Then, for $\gamma_l > 0$,

$$P\left(\gamma_l \max_{i=1,\ldots,M} \|\boldsymbol{\Theta}_i^{(l)}\|_2^2 \geq \|\boldsymbol{A}^l\|_2^2\right) \geq 1 - \left[F_{\chi^2_1}\left(\frac{1}{\gamma_l}\right)\right]^M \tag{4}$$

where $F_{\chi^2_1}$ is the cumulative distribution function of the chi-squared distribution with 1 degree of freedom.

[Uy, P., *Pre-asymptotic error bounds for low-dimensional models learned from systems governed by linear parabolic partial differential equations with control inputs*, *in preparation*, 2020]

ErrEst: Probabilistic bounds on constants (cont'd)

To generate a random variable $\boldsymbol{\Theta}^{(l)} \sim N(\mathbf{0}_{N\times 1}, \boldsymbol{A}^l(\boldsymbol{A}^l)^T)$, simulate the system at $\boldsymbol{x}_0 \sim N(\mathbf{0}, \boldsymbol{I}_N)$ with input $\boldsymbol{u} = [\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0}]$

$$
\begin{aligned}
\boldsymbol{x}_1 &= \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\boldsymbol{u}_1 = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\mathbf{0} = \boldsymbol{A}\boldsymbol{x}_0 \\
\boldsymbol{x}_2 &= \boldsymbol{A}\boldsymbol{x}_1 = \boldsymbol{A}^2\boldsymbol{x}_0 \\
&\vdots \\
\boldsymbol{x}_l &= \boldsymbol{A}^l\boldsymbol{x}_0 \sim N(\mathbf{0}_{N\times 1}, \boldsymbol{A}^l(\boldsymbol{A}^l)^T)
\end{aligned}
$$

- Exploits that system is LTI
- Exploits that the system is queryable

[Uy, P., *Pre-asymptotic error bounds for low-dimensional models learned from systems governed by linear parabolic partial differential equations with control inputs, in preparation*, 2020]

ErrEst: Algorithm

Offline phase

1: Construct a low-dimensional basis $\boldsymbol{V}_n$ from the snapshot matrix
2: Generate $\{\bar{\boldsymbol{x}}_k\}_{k=0}^{K}$ via re-projection and its residual $\{\bar{\boldsymbol{r}}_k\}_{k=0}^{K-1}$ using $\boldsymbol{U}^{\text{train}}$
3: Perform operator inference to obtain $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}$
4: Infer $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$ for computing residual norms
5: Simulate M realizations $\{\boldsymbol{z}_i\}_{i=1}^{M}$ of $\boldsymbol{Z} \sim N(\boldsymbol{0}_{N\times 1}, \boldsymbol{I}_N)$
6: Produce M realizations $\{\boldsymbol{\theta}_i^{(l)}\}_{i=1}^{M}$ of $\boldsymbol{\Theta}^{(l)}$ for $l = 1, \ldots, J$ by querying full system for J time steps with $\boldsymbol{x}_0 = \boldsymbol{z}_i$, $i = 1, \ldots, M$ and input $\boldsymbol{0}$
7: Compute $\xi_l = \sqrt{\gamma_l \max_{i=1,\ldots,M} \|\boldsymbol{\theta}_i^{(l)}\|_2^2}$ for $l = 1, \ldots, J$

Online phase

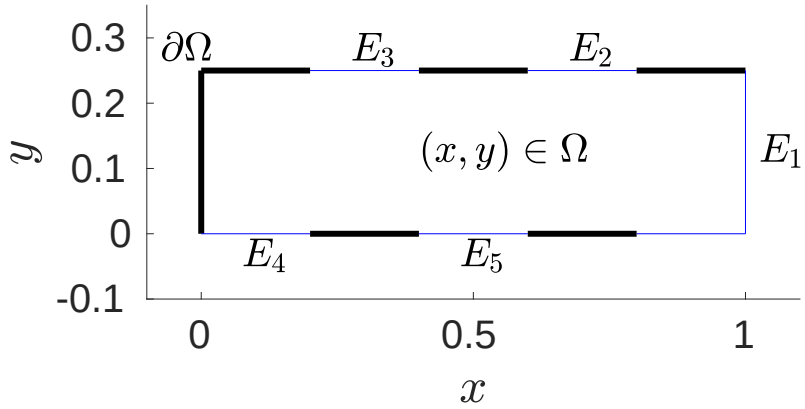
8: Calculate the low-dimensional solution $\{\tilde{\boldsymbol{x}}_k^{\text{test}}\}_{k=1}^{J}$ using the inferred $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}$ and input $\boldsymbol{U}^{\text{test}}$
9: Evaluate $\|\boldsymbol{r}_k^{\text{test}}\|_2^2$ for $k = 1, \ldots, J$ utilizing the inferred $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3$
10: Estimate the *a posteriori* error with probabilistic bound for $k = 1, \ldots, J$

[Uy, P., ***Pre-asymptotic error bounds for low-dimensional models learned from systems governed by linear parabolic partial differential equations with control inputs, in preparation***, 2020]

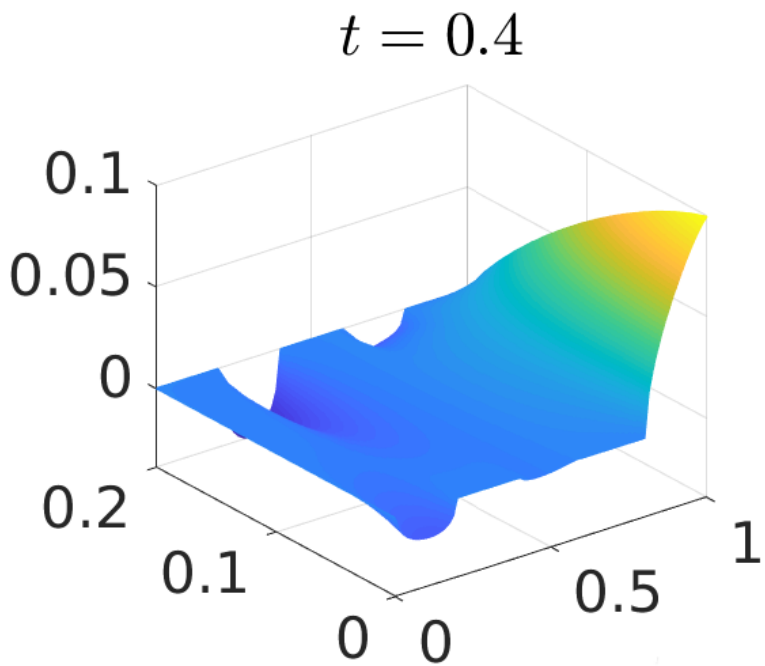
ErrEst: Convection-diffusion in a pipe

Governed by parabolic PDE

$$\frac{\partial x}{\partial t} = \Delta x - (1,1)\cdot\nabla x, \qquad \text{in } \Omega$$
$$x = 0, \qquad \Gamma\backslash\{E_i\}_{i=1}^{5}$$
$$\nabla x \cdot \mathbf{n} = g_i(t), \qquad \text{in } E_i$$



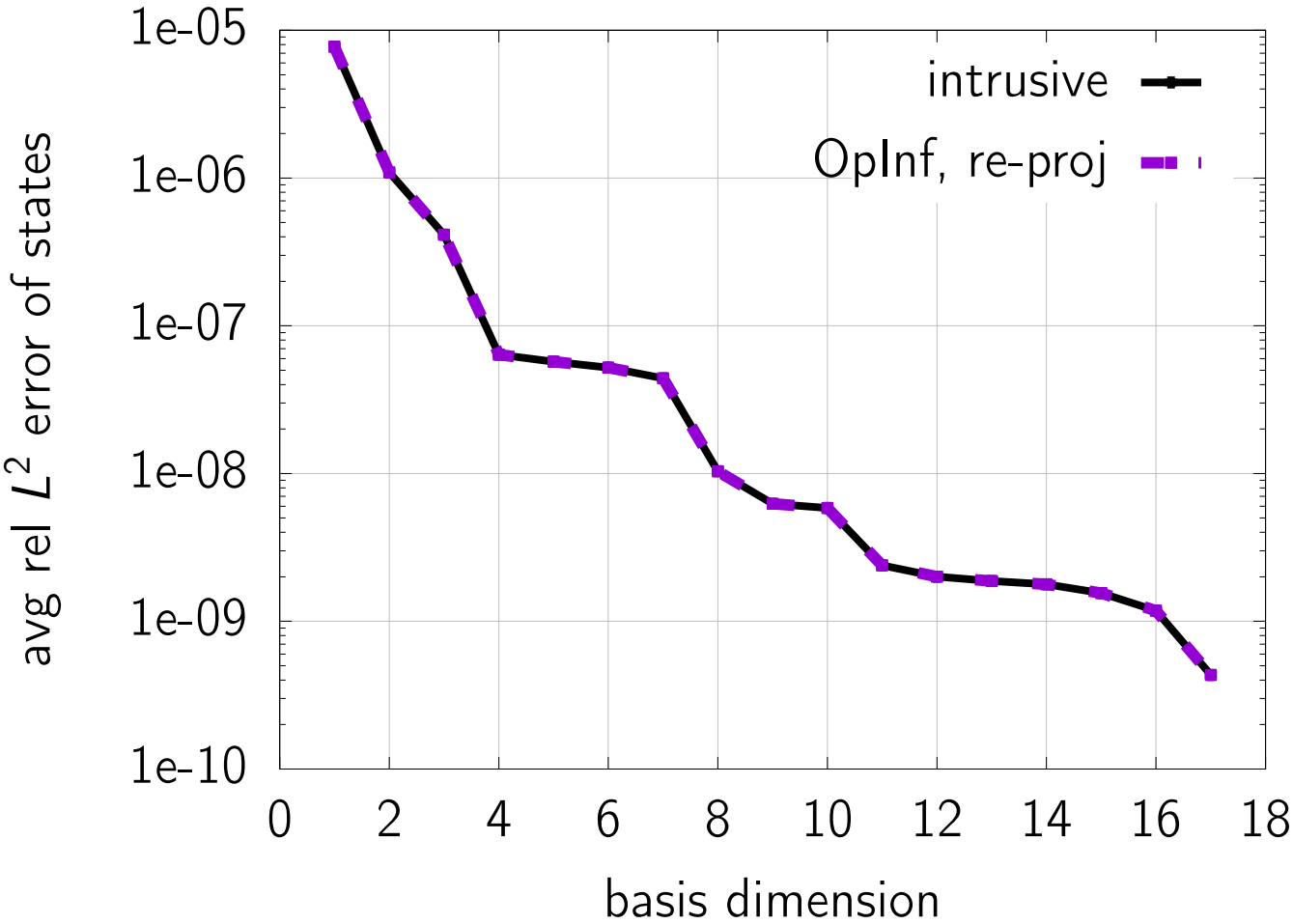
- Discretize with finite elements
- Degrees of freedom $N = 1121$
- Forward Euler method $\delta t = 10^{-5}$
- End time is $T = 0.5$



Input signals

- Training signal is sinusoidal
- Test signal is exponentially decaying sinusoidal with different frequency than training

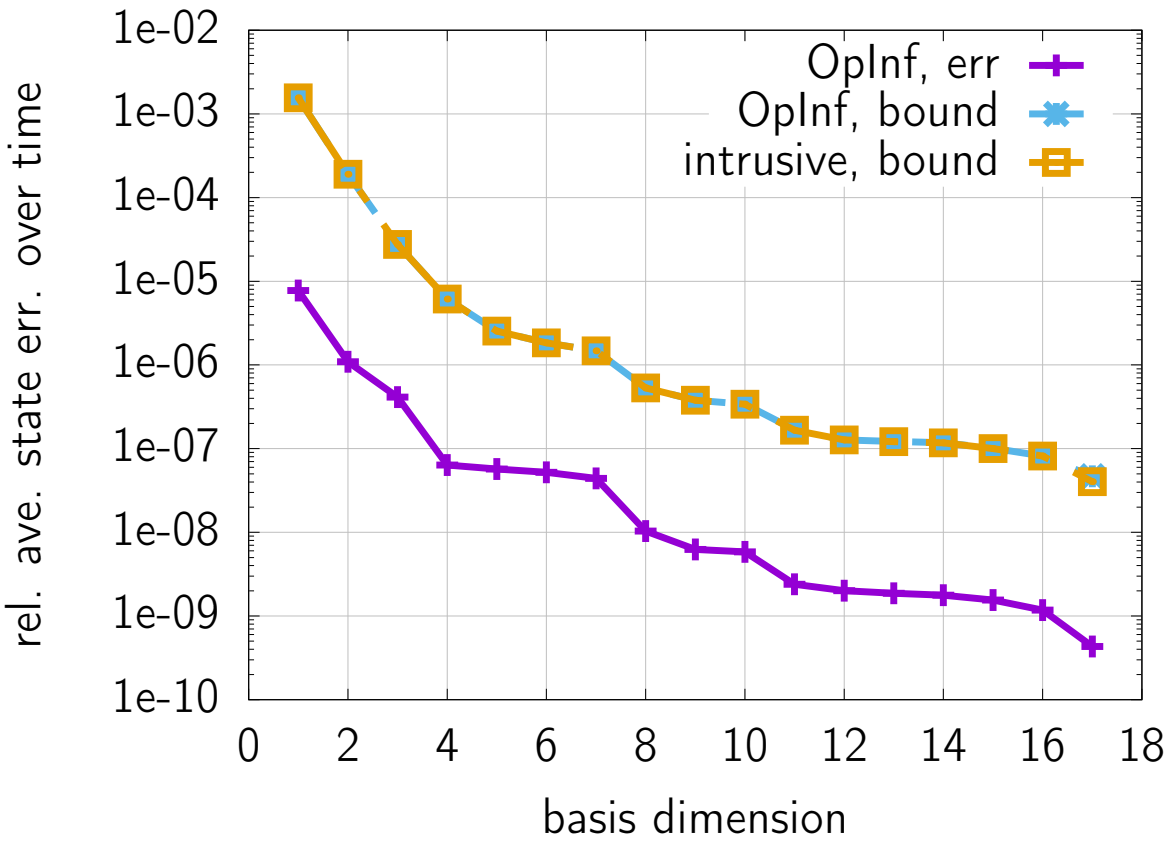
ErrEst: Recovering reduced models from data



Recover reduced models from data

- Error averaged over time
- Recover reduced model up to numerical errors

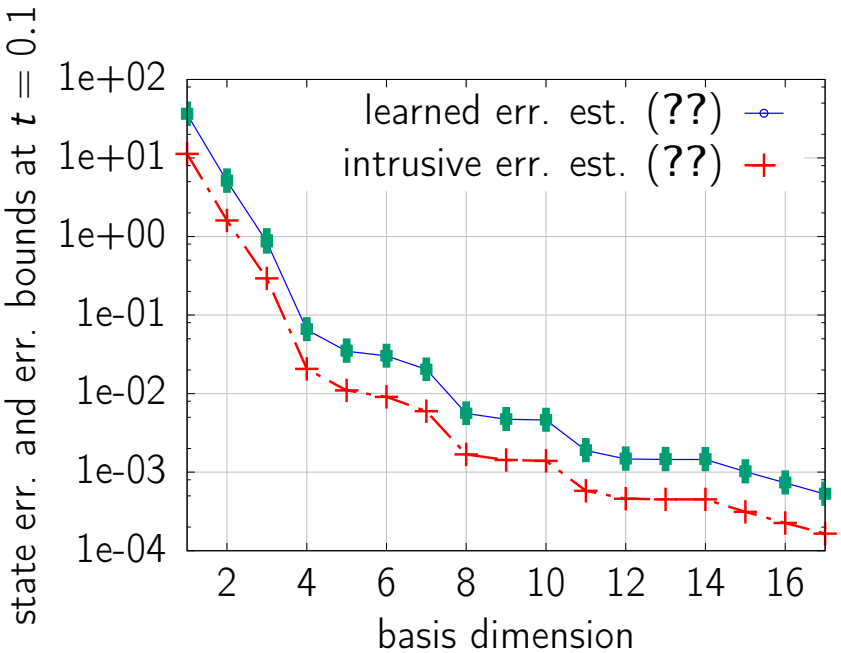
ErrEst: Error bounds



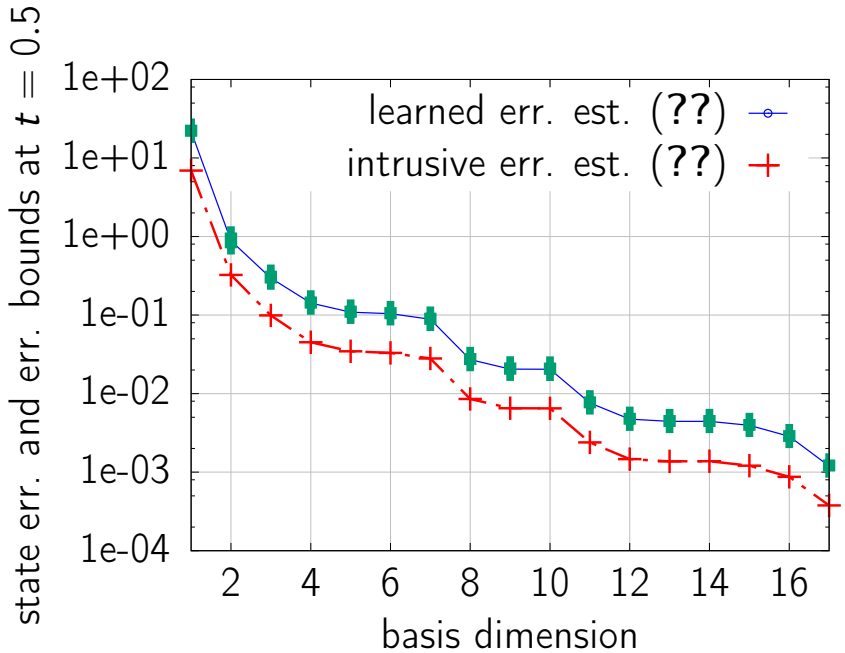
Learn certified reduced model from data alone

- Train with sinusoidal and test with exponential input
- Infer quantities from residual of full model (offline/training)
- Estimate error for test inputs

ErrEst: Variance of estimators



(a) error estimates at $t = 0.1$

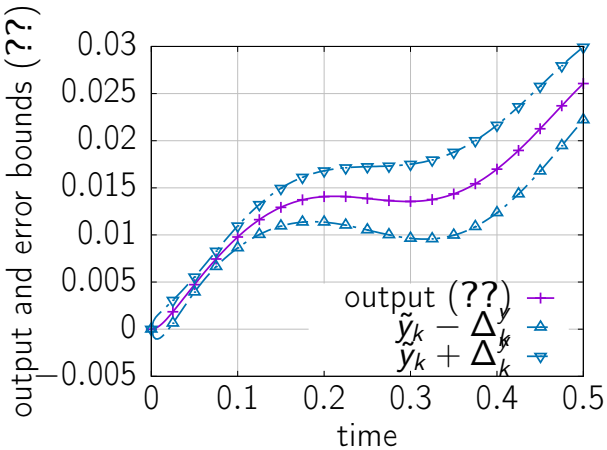


(b) error estimates at $t = 0.5$

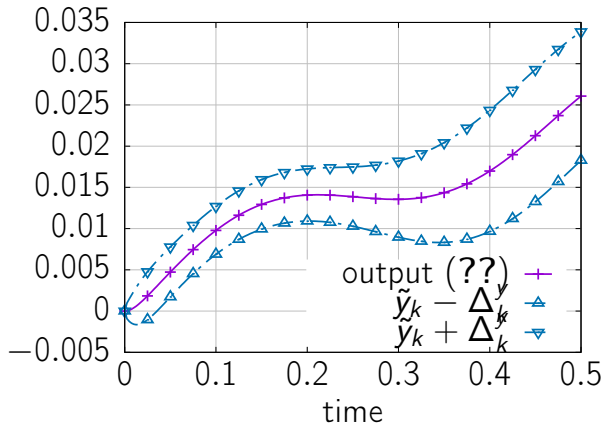
Variation of error estimator is rather low in this example

- Mean, minimum, and maximum over 50 realizations
- Fixed $\gamma = 1$ and number of samples $M = 35$

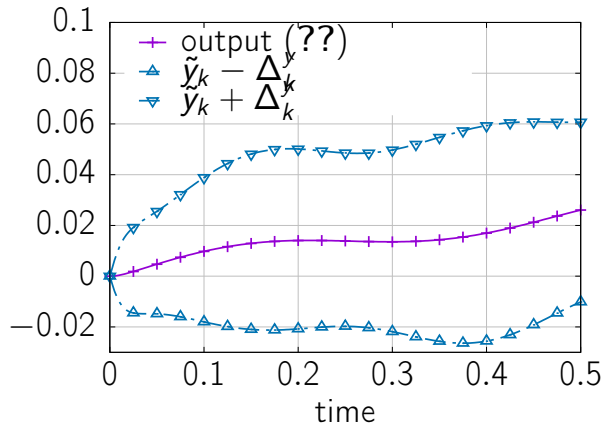
ErrEst: Output bounds



(a) $n = 17$ basis



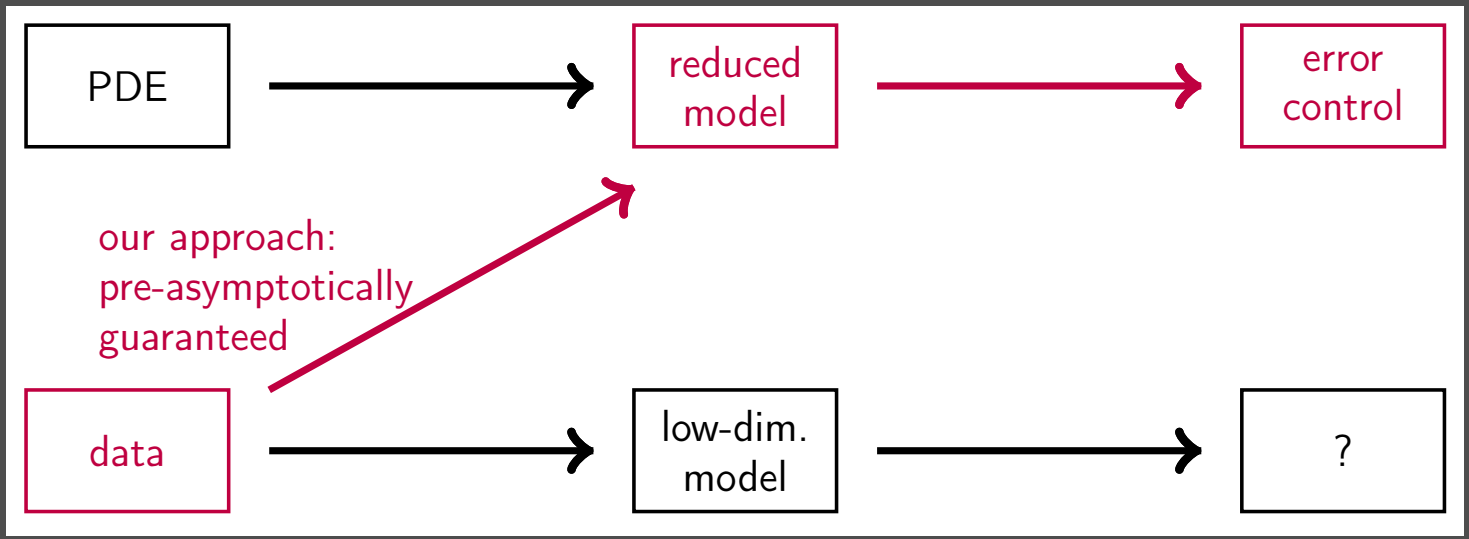
(b) $n = 12$ basis

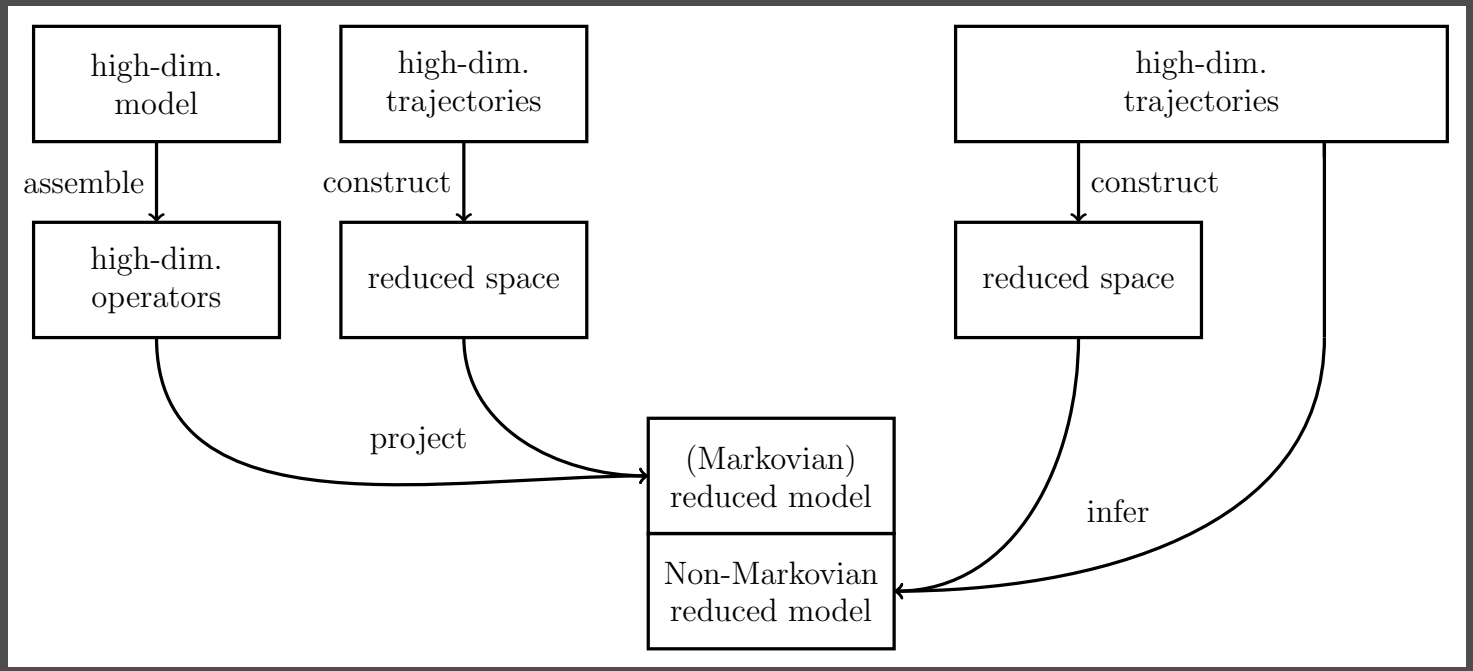


(c) $n = 7$ basis

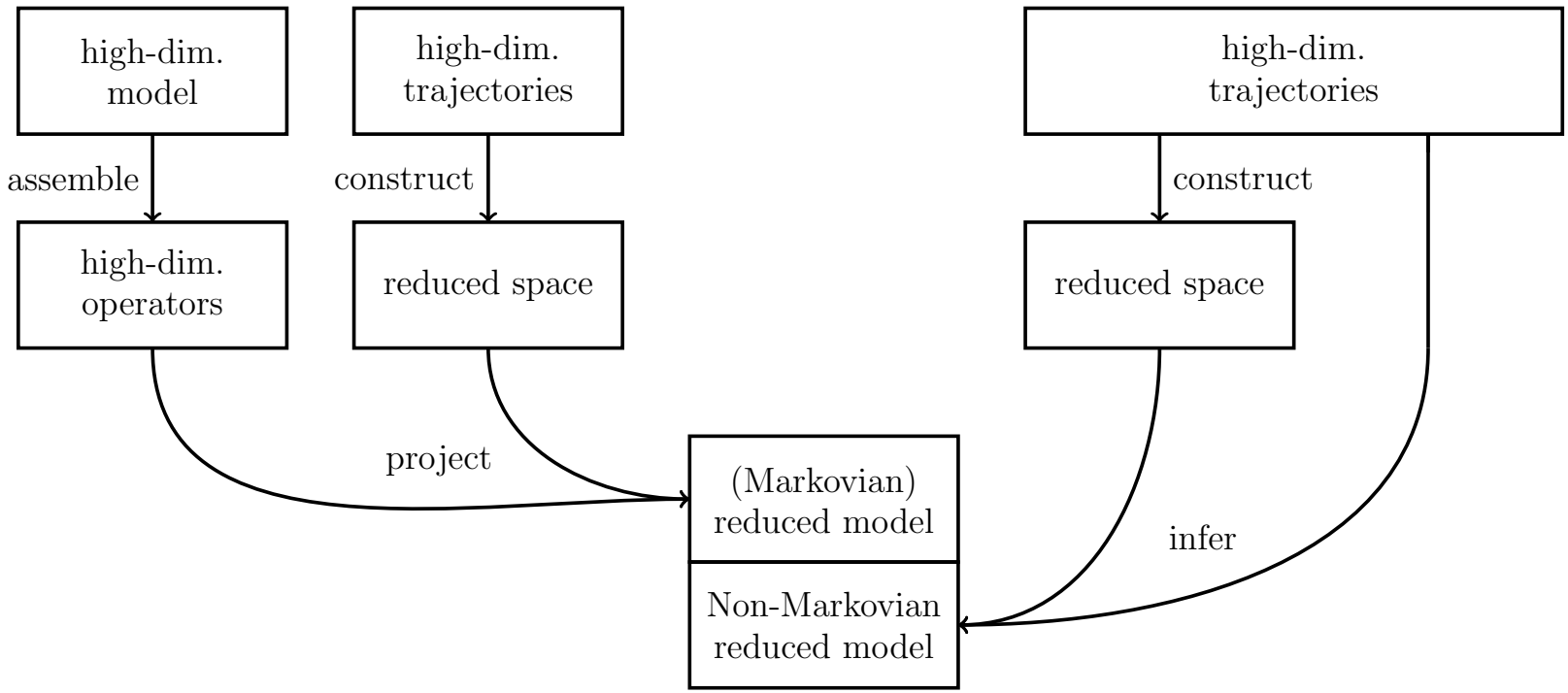
If output is linear $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k$ in state with operator $\boldsymbol{C}$ and known norm $\|\boldsymbol{C}\|_2$

- Probabilistic bound of the error $\|\boldsymbol{y}_k - \tilde{\boldsymbol{y}}_k\|_2$
- Error bound indicates that error is reduced when dimension n of reduced spaces is increased


PDE
reduced model
error control
our approach:
pre-asymptotically
guaranteed
data
low-dim. model
?


high-dim.
model
high-dim.
trajectories
high-dim.
trajectories
assemble
construct
construct
high-dim.
operators
reduced space
reduced space
project
(Markovian)
reduced model
infer
Non-Markovian
reduced model

NonM: Non-Markovian reduced models



Learning non-Markovian low-dim. models in model reduction

- (Full model is non-Markovian [Schulze, Unger, Beattie, Gugercin, 2018])
- Closure error is high and needs to be corrected (steep gradients, shocks)
- Only partially observed state trajectory available

NonM: Learning non-Markovian reduced models

With re-projection, exactly learn Markovian reduced model

$$\tilde{\boldsymbol{x}}_{k+1} = \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k^i + \tilde{\boldsymbol{B}} \boldsymbol{u}_k$$

However, loose dynamics modeled by non-Markovian terms

$$\breve{\boldsymbol{x}}_{k+1} = \sum_{i=1}^{\ell} \tilde{\boldsymbol{A}}_i \breve{\boldsymbol{x}}_k^i + \tilde{\boldsymbol{B}} \boldsymbol{u}_k + \sum_{i=1}^{k-1} \boldsymbol{\Delta}_i(\breve{\boldsymbol{x}}_{k-1}, \ldots, \breve{\boldsymbol{x}}_{k-i+1}, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k-i+1}) + 0$$

Learn unresolved dynamics via approximate non-Markovian terms

$$\hat{\boldsymbol{x}}_{k+1} = \sum_{i=1}^{\ell} \hat{\boldsymbol{A}}_i \hat{\boldsymbol{x}}_k^i + \hat{\boldsymbol{B}} \boldsymbol{u}_k + \sum_{i=1}^{k-1} \hat{\boldsymbol{\Delta}}_i^{\boldsymbol{\theta}_i}(\hat{\boldsymbol{x}}_{k-1}, \ldots, \hat{\boldsymbol{x}}_{k-i+1}, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k-i+1})$$

- Parametrization $\boldsymbol{\theta}_i \in \Theta$ for $i = 0, \ldots, K-1$
- Non-Markovian models extensively used in statistics but less so in MOR

NonM: Sampling with stage-wise re-projection

Learning model operators and non-Markovian terms at the same
$\Rightarrow$ Dynamics mixed, same issues as learning from projected states

Build on re-projection to learn non-Markovian terms stage-wise

- Sample trajectories of length $r+1$ with re-projection

$$\bar{\boldsymbol{X}}^{(0)}, \ldots, \bar{\boldsymbol{X}}^{(K-1)} \in \mathbb{R}^{n \times r+1}$$

- Infer Markovian reduced model $\hat{\boldsymbol{f}}_1$ from one-step trajectories

$$\bar{\boldsymbol{X}}_1^{(i)} = [\bar{\boldsymbol{x}}_0^{(i)}, \bar{\boldsymbol{x}}_1^{(i)}], \qquad i = 0, \ldots, K-1$$

- Simulate $\hat{\boldsymbol{f}}_1$ to obtain

$$\hat{\boldsymbol{X}}_2^{(i)} = [\hat{\boldsymbol{x}}_0^{(i)}, \hat{\boldsymbol{x}}_1^{(i)}, \hat{\boldsymbol{x}}_2^{(i)}], \qquad i = 0, \ldots, K-1$$

- Fit parameter $\boldsymbol{\theta}_1$ of non-Markovian term $\hat{\boldsymbol{\Delta}}_1^{\boldsymbol{\theta_1}}$ to difference

$$\min_{\boldsymbol{\theta_1} \in \Theta} \sum_{i=0}^{K-1} \| \bar{\boldsymbol{x}}_2^{(i)} - \hat{\boldsymbol{x}}_2^{(i)} - \hat{\boldsymbol{\Delta}}_1^{(\boldsymbol{\theta_1})}(\bar{\boldsymbol{x}}_0^{(i)}, \boldsymbol{u}_i) \|_2^2$$

- Repeat this r times to learn $\hat{\boldsymbol{f}}_r$ with lag r

NonM: Learning non-Markovian terms

Parametrization of non-Markovian terms

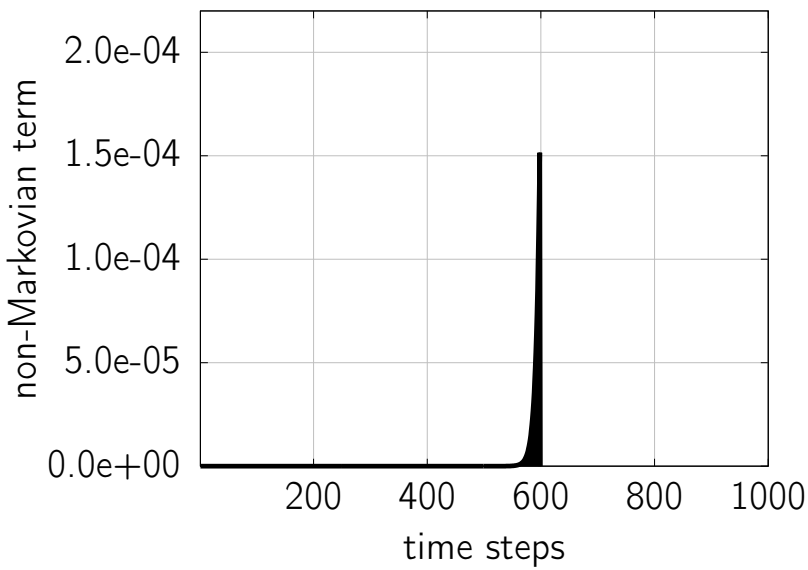
- Set $\theta_i = [\boldsymbol{D}_i, \boldsymbol{E}_i]$ with $\boldsymbol{D}_i \in \mathbb{R}^{n \times n}$ and $\boldsymbol{E}_i \in \mathbb{R}^{n \times p}$
- Non-Markovian term is

$$\hat{\boldsymbol{\Delta}}_i^{(\theta_i)}(\hat{\boldsymbol{x}}_{k-1}, \ldots, \hat{\boldsymbol{x}}_{k-i+1}, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k-i+1}) = \boldsymbol{D}_i \hat{\boldsymbol{x}}_{k-i+1} + \boldsymbol{E}_i \boldsymbol{u}_{k-i+1}$$

- Other parametrizations with higher-order terms and neural networks

Choosing maximal lag

- Assumption (observation) is that non-Markovian term of system has small support
- Need to go back in time only a few steps
- Lag r can be chosen small

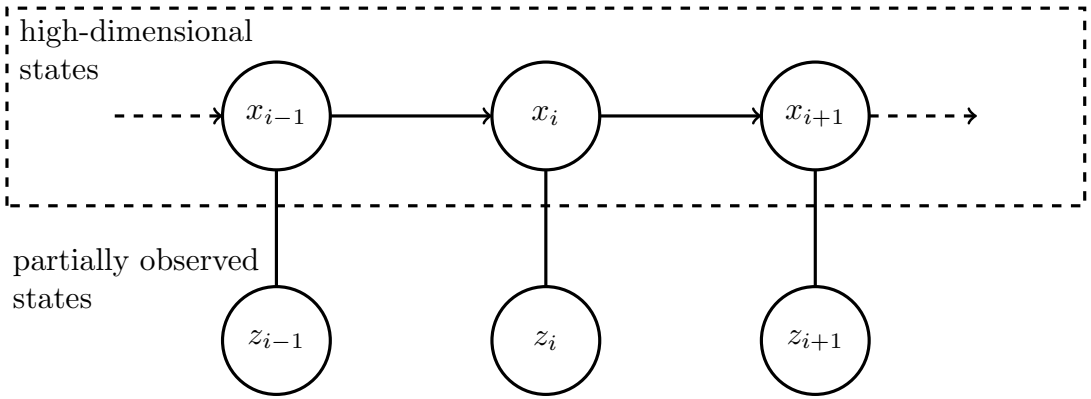


NonM: Learning from partially observed states

Partially observed state trajectories

- Unknown selection operator $\boldsymbol{S} \in \{0,1\}^{N_s \times N}$ with $N_s < N$ and

$$\boldsymbol{z}_k = \boldsymbol{S}\boldsymbol{x}_k$$



- Learn models from trajectory $\boldsymbol{Z} = [\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{K-1}]$ instead of $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{K-1}]$

- Apply POD (PCA) to $\boldsymbol{Z}$ to find basis matrix $\boldsymbol{V}$ of subspace $\mathcal{V}$ of $\mathbb{R}^{N_s}$

Non-Markovian terms to compensate unobserved state components

- Mori-Zwanzig formalism applies
- Non-Markovian terms compensate unobserved components

NonM: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

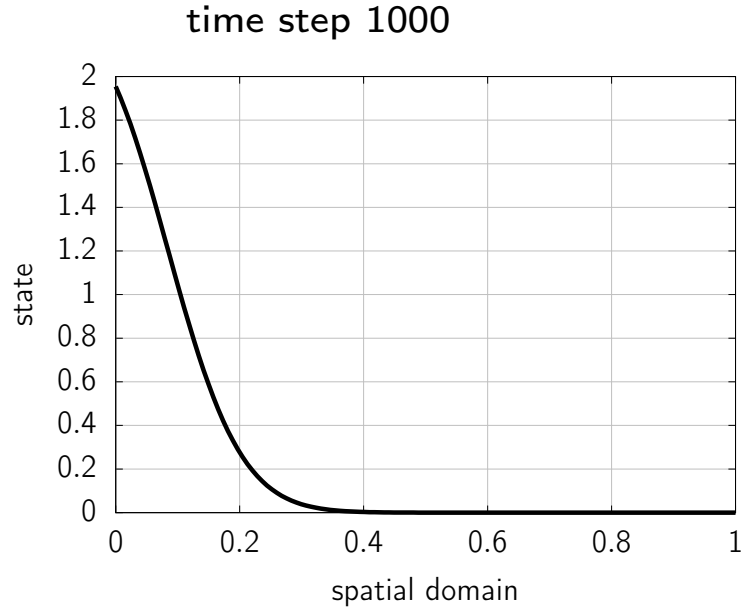
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

NonM: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

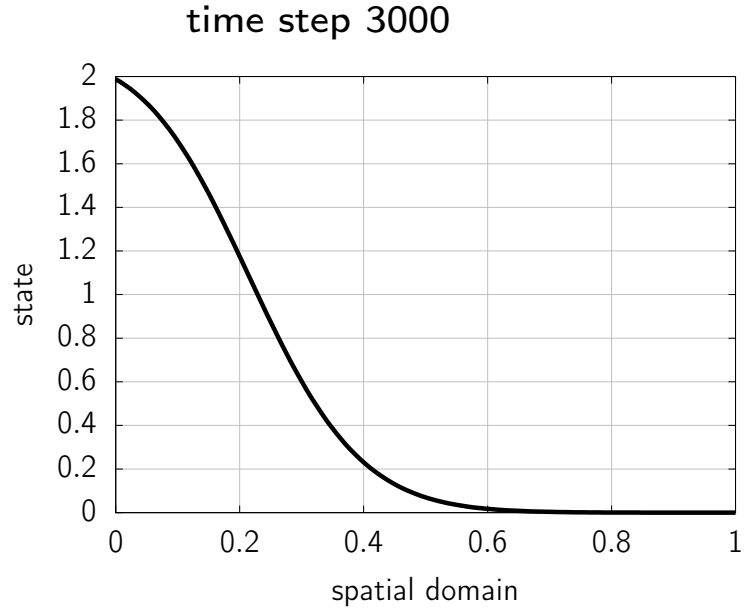
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

NonM: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

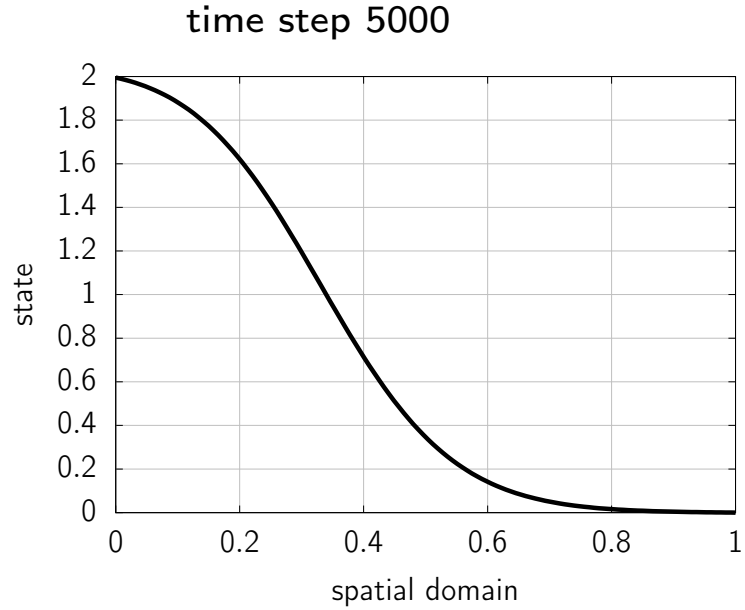
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

NonM: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

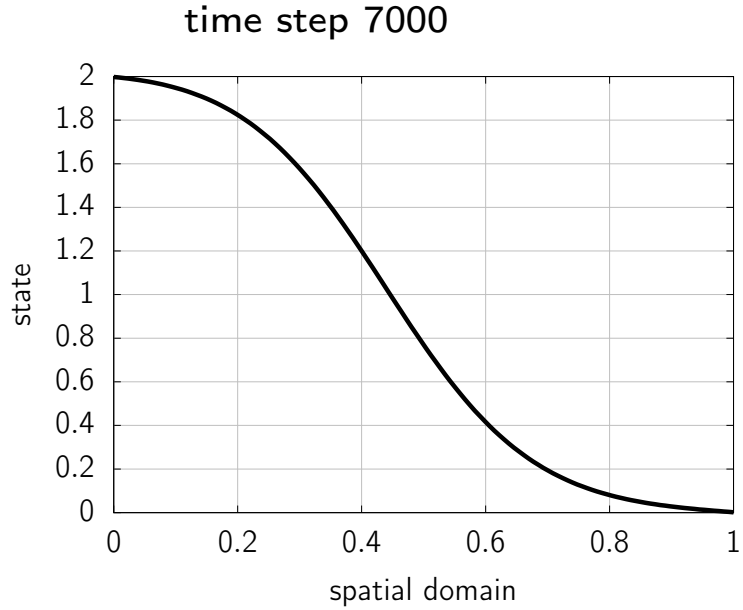
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

NonM: Burgers': Burgers' example

Viscous Burgers' equation

$$\frac{\partial}{\partial t}x(\omega, t; \mu) + x(\omega, t; \mu)\frac{\partial}{\partial \omega}x(\omega, t; \mu) - \mu\frac{\partial^2}{\partial \omega^2}x(\omega, t; \mu) = 0$$

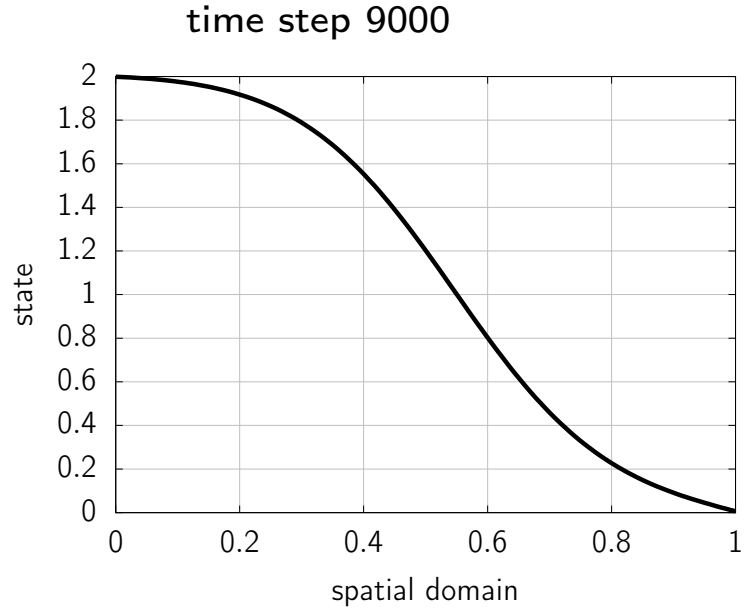
- Spatial, time, and parameter domain

$$\omega \in [0, 1], \quad t \in [0, 1], \quad \mu \in [0.1, 1]$$

- Dirichlet boundary conditions

$$x(0, t; \mu) = -x(1, t; \mu) = u(t)$$

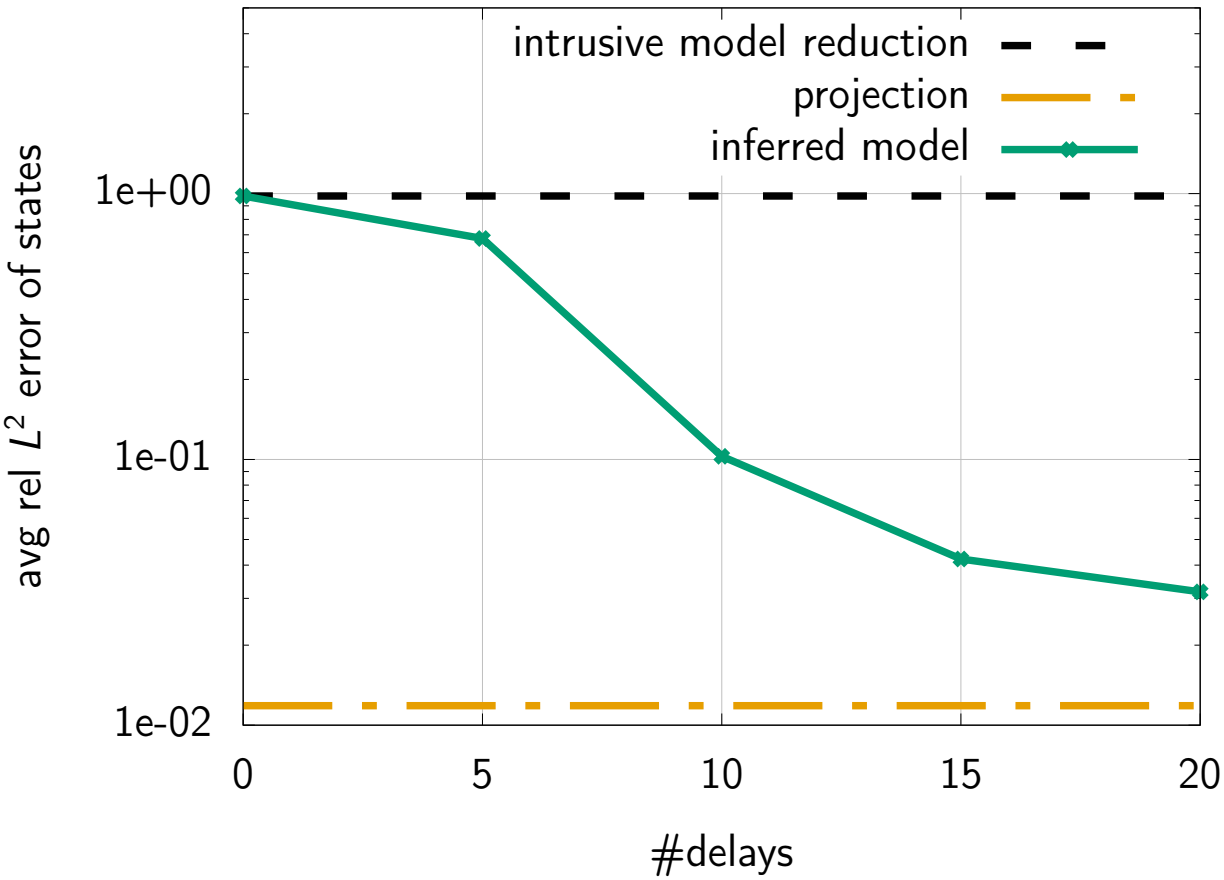
- Discretize with forward Euler
- Time step size is $\delta t = 10^{-4}$



Operator inference

- Training data are 2 trajectories with random inputs
- Infer operators for 10 equidistant parameters in $[0.1, 1]$
- Interpolate inferred operators at 7 test parameters and predict

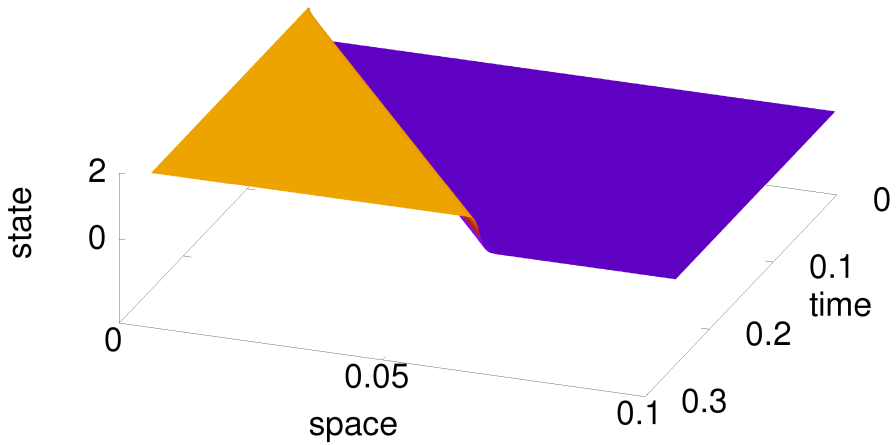
NonM: Burgers': Partial observations



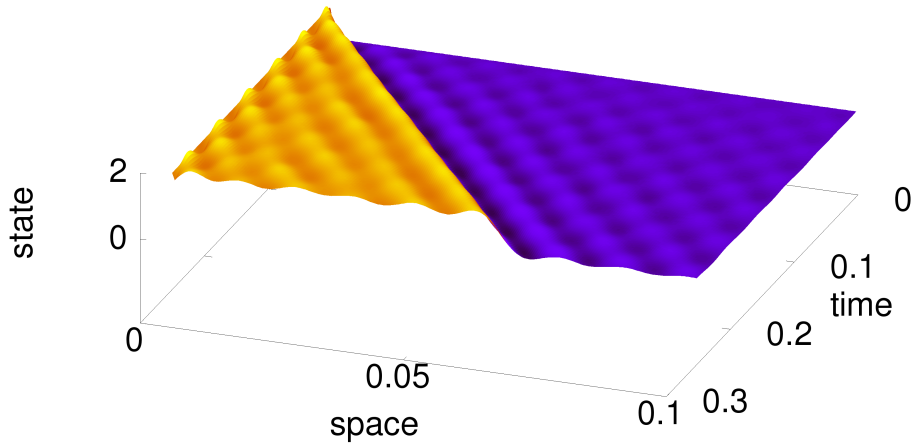
Observe only about 50% of all state components

- Linear time-delay terms with stage-wise re-projection
- Reduces error of inferred model by more than one order of magnitude

NonM: Burgers': Shock formation



(a) ground truth (full model)

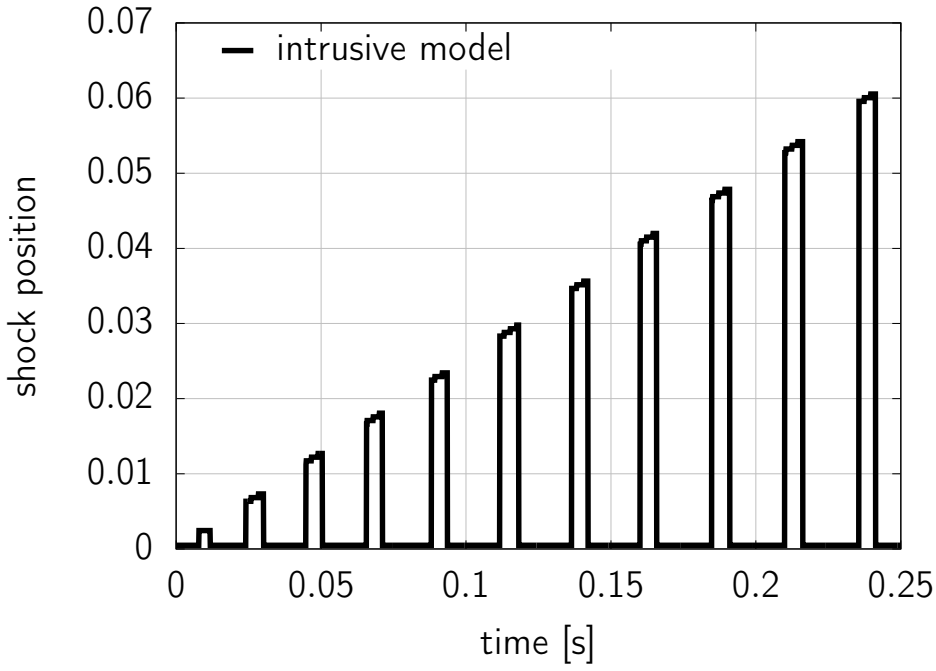


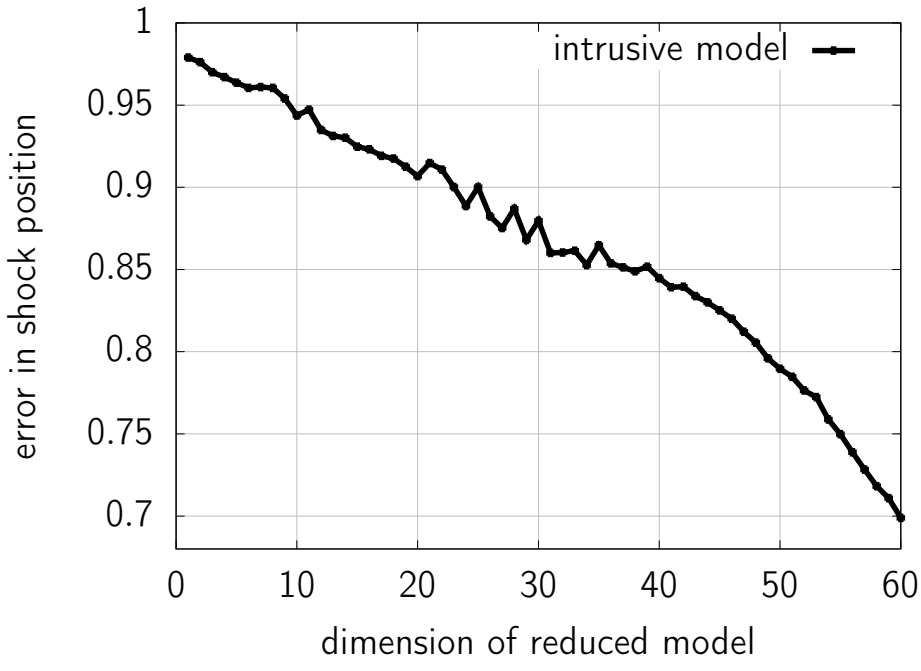
(b) intrusive model reduction

Modify coefficients of Burgers' equation to obtain solution with shock

- Solutions with shocks are challenging to reduce with model reduction
- Here, reduced model from intrusive model reduction has oscillatory error

NonM: Burgers': Capturing shock position

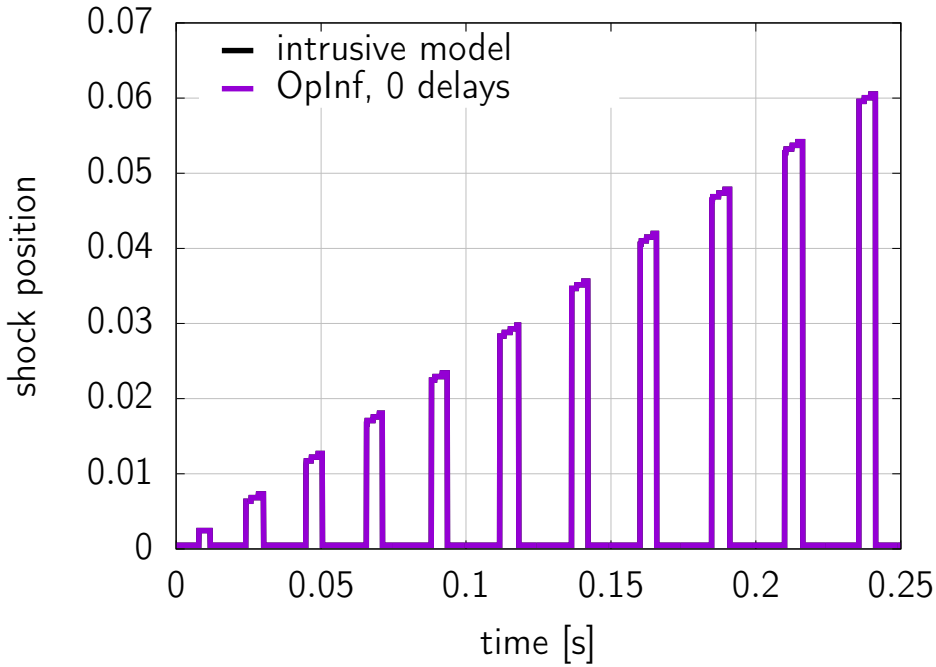


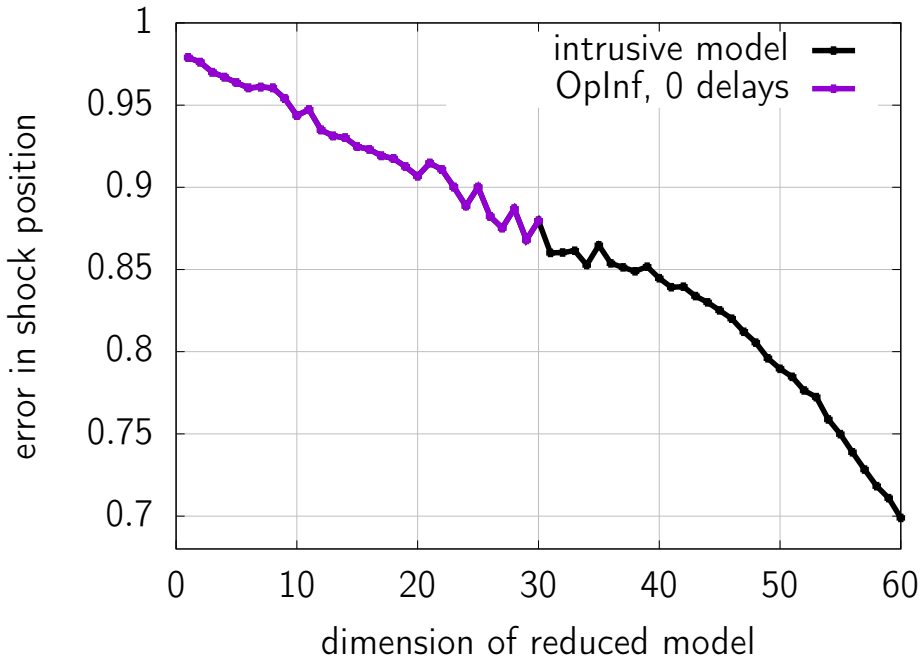


Learn time-delay terms stage-wise with (re-)re-projection

- Learn linear time-delay corrections
- In this example, time delay of order 4 sufficient to capture shock
- Higher-order time-delay terms learned in, e.g., [Pan, Duraisamy, 2018]

NonM: Burgers': Capturing shock position

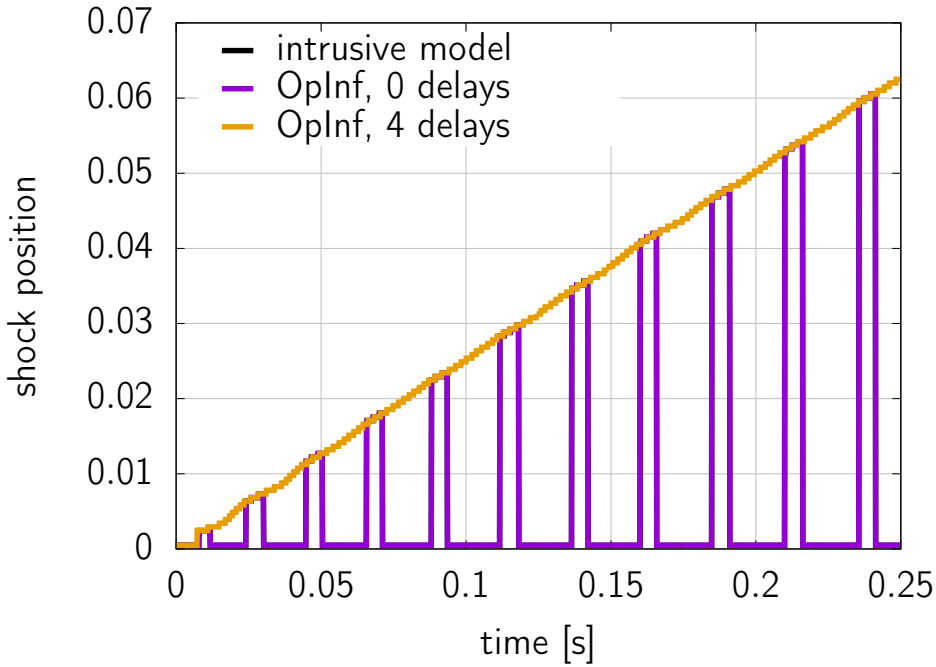


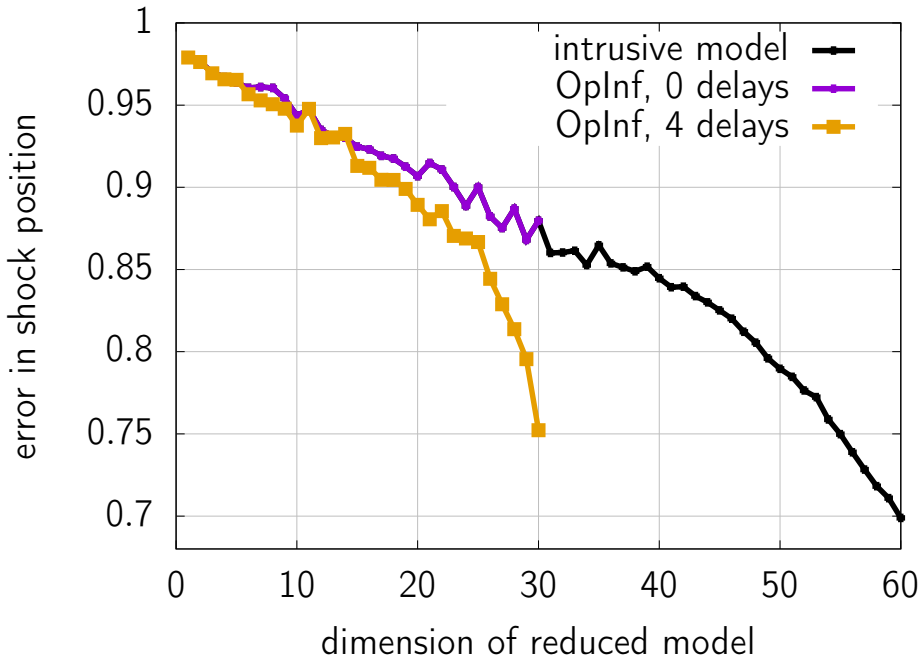


Learn time-delay terms stage-wise with (re-)re-projection

- Learn linear time-delay corrections
- In this example, time delay of order 4 sufficient to capture shock
- Higher-order time-delay terms learned in, e.g., [Pan, Duraisamy, 2018]

NonM: Burgers': Capturing shock position

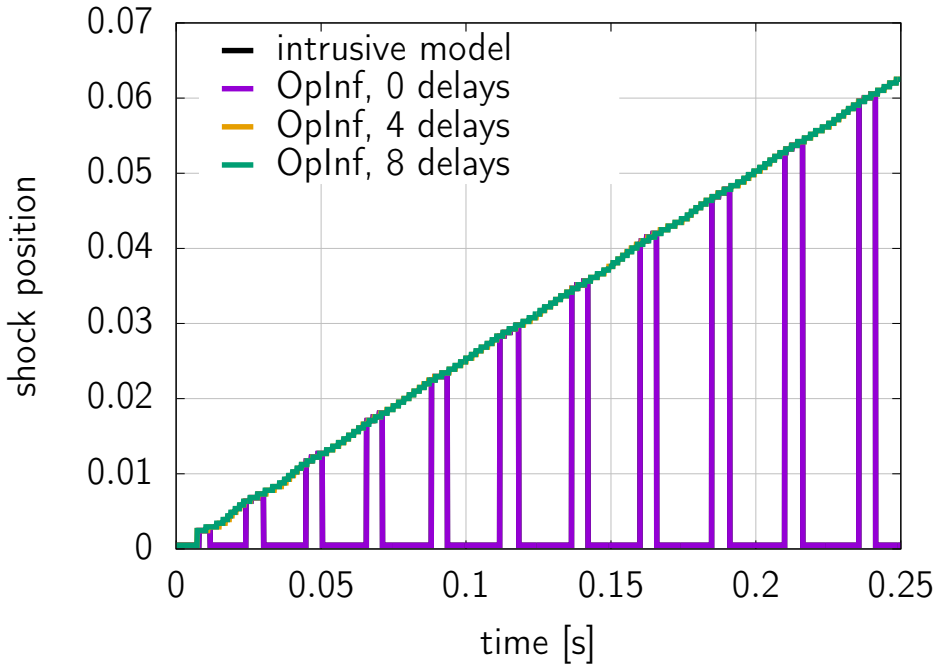


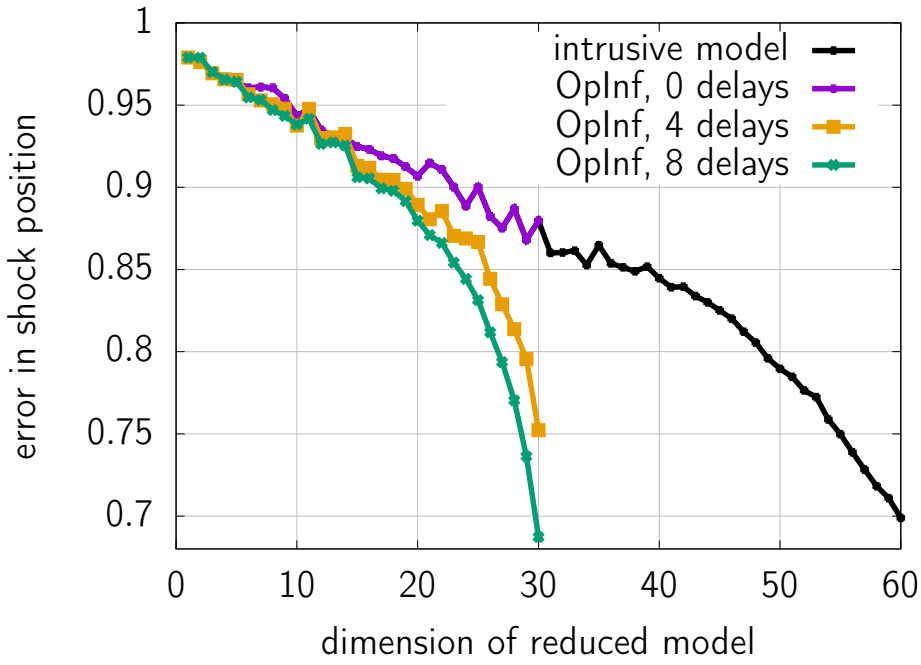


Learn time-delay terms stage-wise with (re-)re-projection

- Learn linear time-delay corrections
- In this example, time delay of order 4 sufficient to capture shock
- Higher-order time-delay terms learned in, e.g., [Pan, Duraisamy, 2018]

NonM: Burgers': Capturing shock position

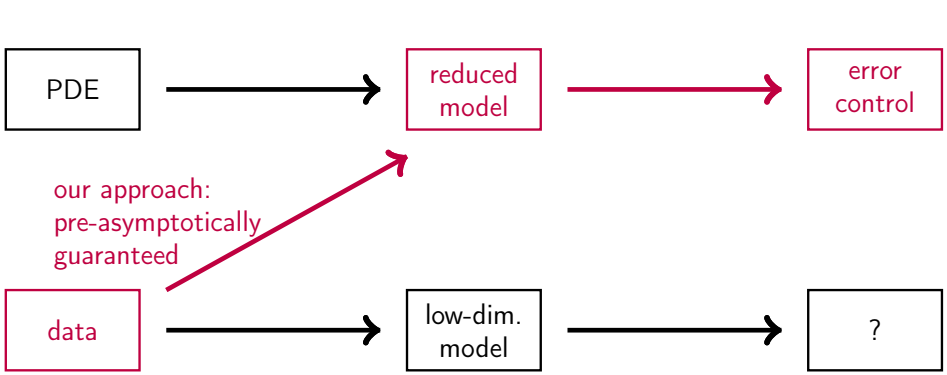


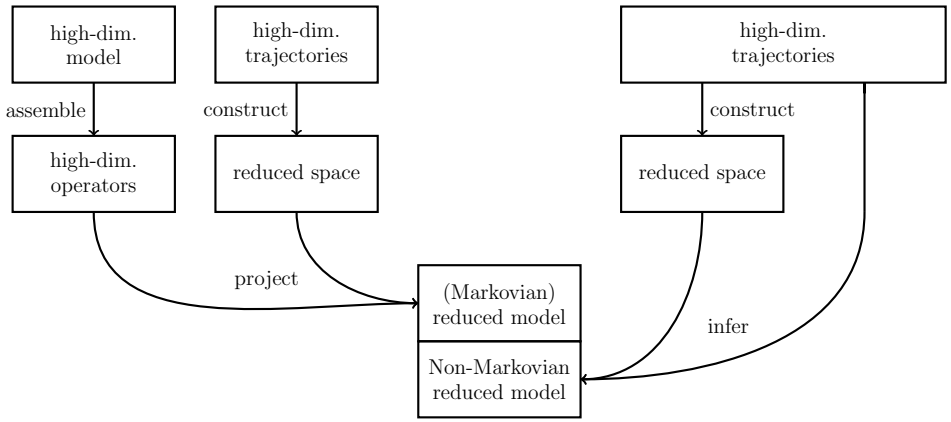


Learn time-delay terms stage-wise with (re-)re-projection

- Learn linear time-delay corrections
- In this example, time delay of order 4 sufficient to capture shock
- Higher-order time-delay terms learned in, e.g., [Pan, Duraisamy, 2018]

Conclusions





Learning dynamical-system models from data with error guarantees

- Operator inference *exactly* recovers reduced models from data
- Generating the right data is key to learning reduced models in our case
- Pre-asymptotic guarantees (finite data) under certain conditions
- Going beyond reduced models by learning non-Markovian corrections

References: https://cims.nyu.edu/∼pehersto

- Uy, P., *Pre-asymptotic error bounds for low-dimensional models learned from systems governed by linear parabolic partial differential equations with control inputs*, *in preparation*, 2020.
- P., Sampling low-dimensional Markovian dynamics for pre-asymptotically recovering reduced models from data with operator inference. arXiv:1908.11233, 2019.
- P., Willcox, Data-driven operator inference for nonintrusive projection-based model reduction. Computer Methods in Applied Mechanics and Engineering, 306:196-215, 2016.

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- **Learning from frequency-response data**
- Multi-fidelity uncertainty quantification

Loewner: Dynamical systems

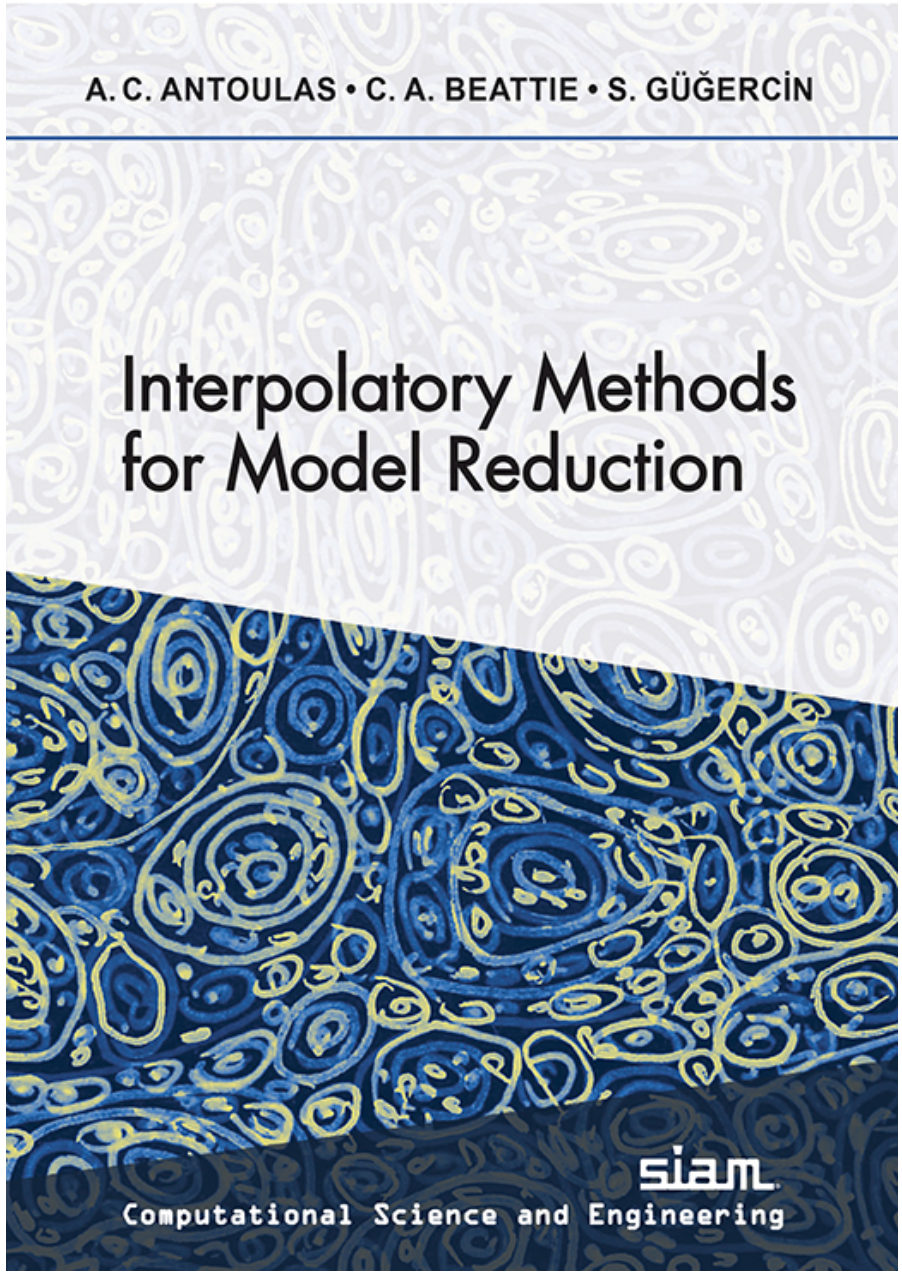
Consider linear time-invariant (LTI) system

$$\boldsymbol{\Sigma}: \begin{cases} \boldsymbol{E}\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k\,, & k \in \mathbb{N}\,, \\ y_k = \boldsymbol{C}\boldsymbol{x}_k \end{cases}$$

- Time-discrete single-input-single-output (SISO) LTI system
- System matrices $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{N\times 1}$, $\boldsymbol{C} \in \mathbb{R}^{1\times N}$
- Input u_k and output y_k at time step $t_k, k \in \mathbb{N}$
- State $\boldsymbol{x}_k$ at time step $t_k, k \in \mathbb{N}$
- Asymptotically stable

Important is the mapping $u \mapsto y$, not the complete state

- Want to find a reduced model that accurately approximates the input-output map $u \mapsto y$
- Consider y to be the quantity of interest


A. C. ANTOULAS • C. A. BEATTIE • S. GÜĞERCİN
Interpolatory Methods for Model Reduction
siam
Computational Science and Engineering

Loewner: Impulse response

The output y_k is the convolution of the impulse response of the system $\boldsymbol{\Sigma}$ with the inputs $u_0, \ldots, u_k$

$$y_k = \sum_{i=0}^{k} h_i u_{k-i} \,,$$

with impulse response

$$h_k = \begin{cases} \boldsymbol{C}(\boldsymbol{E}^{-1}\boldsymbol{A})^{k-1}(\boldsymbol{E}^{-1}\boldsymbol{B}) \,, & k > 0 \\ 0 \,, & k \leq 0 \end{cases} .$$

- Impulse response h_k defines the input-output map $u \mapsto y$
- In the continuous-time setting, the convolution-sum becomes a convolution-integral

Loewner: Transfer function

Transform the time-domain output $\{y_k\}_{k=0}^{\infty}$ into frequency domain with Z-transform (Laplace transform)

$$Y(z) = \sum_{k=0}^{\infty} y_k z^{-k}$$

Transform the impulse response $\{h_k\}_{k=0}^{\infty}$ to obtain the *transfer function*

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k}$$

Convolution becomes multiplication in frequency domain

$$Y(z) = H(z)U(z)$$

The transfer function defines the map $u \mapsto y$

Loewner: Transfer function (cont'd)

Different representation of transfer function of LTI system $\boldsymbol{\Sigma}$

$$H(z) = \boldsymbol{C}(z\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}\,, \qquad z \in \mathbb{C}$$

Consider reduced model $\tilde{\boldsymbol{\Sigma}}$ with

$$\tilde{H}(z) = \tilde{\boldsymbol{C}}(z\tilde{\boldsymbol{E}} - \tilde{\boldsymbol{A}})^{-1}\tilde{\boldsymbol{B}}\,, \qquad z \in \mathbb{C}$$

Measure error of reduced transfer function $\tilde{H}$ as

$$\|H - \tilde{H}\|_{\mathcal{H}_\infty} = \sup_{|z|=1} |H(z) - \tilde{H}(z)|$$

Relate to error in quantity of interest

$$\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|_2 \leq \|H - \tilde{H}\|_{\mathcal{H}_\infty} \|\boldsymbol{u}\|_2$$

If $\tilde{H}$ approximates H well, then know that $\tilde{\boldsymbol{y}}$ approximates $\boldsymbol{y}$ well

Loewner: Model reduction via projection

- Choose trial subspace spanned by columns of $\boldsymbol{V} \in \mathbb{R}^{N \times n}$
- Choose test subspace spanned by columns of $\boldsymbol{W} \in \mathbb{R}^{N \times n}$
- Approximate $\boldsymbol{x}_k \approx \boldsymbol{V}\tilde{\boldsymbol{x}}_k$ by forcing $\tilde{\boldsymbol{x}}_k$ to satisfy

$$\boldsymbol{W}^T(\boldsymbol{E}\boldsymbol{V}\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{A}\boldsymbol{V}\tilde{\boldsymbol{x}}_k - \boldsymbol{B}u_k) = 0$$

- Petrov-Galerkin projection because trial and test subspaces can be different
- Leads to reduced model

$$\tilde{\boldsymbol{E}} = \boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}, \quad \tilde{\boldsymbol{A}} = \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}, \quad \tilde{\boldsymbol{B}} = \boldsymbol{W}^T\boldsymbol{B}, \quad \tilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{V}$$

Loewner: Interpolatory model reduction

Select $m = 2n$ interpolation points

$$z_1, \ldots, z_m \in \mathbb{C}$$

Construct bases as

$$\boldsymbol{V} = \begin{bmatrix} (z_1 \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} & \ldots & (z_n \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{N \times n}$$
$$\boldsymbol{W} = \begin{bmatrix} (z_{n+1} \boldsymbol{E}^T - \boldsymbol{A}^T)^{-1} \boldsymbol{C}^T & \ldots & (z_{n+n} \boldsymbol{E}^T - \boldsymbol{A}^T)^{-1} \boldsymbol{C}^T \end{bmatrix} \in \mathbb{R}^{N \times n}$$

Project (Petrov-Galerkin) to obtain operators

$$\tilde{\boldsymbol{E}} = \boldsymbol{W}^T \boldsymbol{E} \boldsymbol{V}, \quad \tilde{\boldsymbol{A}} = \boldsymbol{W}^T \boldsymbol{A} \boldsymbol{V}, \quad \tilde{\boldsymbol{B}} = \boldsymbol{W}^T \boldsymbol{B}, \quad \tilde{\boldsymbol{C}} = \boldsymbol{C} \boldsymbol{V}$$

Then obtain reduced model $\tilde{\boldsymbol{\Sigma}}$ with $\tilde{H}$

$$H(z_i) = \tilde{H}(z_i), \qquad i = 1, \ldots, m$$

Loewner: Interpolatory model reduction

Select $m = 2n$ interpolation points

$$z_1, \ldots, z_m \in \mathbb{C}$$

Construct bases as

$$\boldsymbol{V} = \begin{bmatrix} (z_1 \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} & \ldots & (z_n \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{N \times n}$$
$$\boldsymbol{W} = \begin{bmatrix} (z_{n+1} \boldsymbol{E}^T - \boldsymbol{A}^T)^{-1} \boldsymbol{C}^T & \ldots & (z_{n+n} \boldsymbol{E}^T - \boldsymbol{A}^T)^{-1} \boldsymbol{C}^T \end{bmatrix} \in \mathbb{R}^{N \times n}$$

Project (Petrov-Galerkin) to obtain operators

$$\tilde{\boldsymbol{E}} = \boldsymbol{W}^T \boldsymbol{E} \boldsymbol{V}, \quad \tilde{\boldsymbol{A}} = \boldsymbol{W}^T \boldsymbol{A} \boldsymbol{V}, \quad \tilde{\boldsymbol{B}} = \boldsymbol{W}^T \boldsymbol{B}, \quad \tilde{\boldsymbol{C}} = \boldsymbol{C} \boldsymbol{V}$$

Then obtain reduced model $\tilde{\boldsymbol{\Sigma}}$ with $\tilde{H}$

$$H(z_i) = \tilde{H}(z_i), \qquad i = 1, \ldots, m$$

Requires full operators $\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ either in assembled or implicit form

Loewner: Interpolatory model reduction (cont'd)

Loewner framework derives $\tilde{\boldsymbol{\Sigma}}$ directly from $H(z_1), \ldots, H(z_m)$ with

$$\mathbb{L}_{ij} = \frac{H(z_i) - H(z_{n+j})}{z_i - z_{n+j}}, \quad \mathbb{L}_{ij}^{(s)} = \frac{z_i H(z_i) - z_{n+j} H(z_{n+j})}{z_i - z_{n+j}}, \qquad i, j = 1, \ldots, n$$

Reduced operators of $\tilde{\boldsymbol{\Sigma}}$ are

$$\tilde{\boldsymbol{E}} = -\mathbb{L}, \quad \tilde{\boldsymbol{A}} = -\mathbb{M}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} H(z_1) & \ldots & H(z_n) \end{bmatrix}^T,$$
$$\text{and} \quad \tilde{\boldsymbol{C}} = \begin{bmatrix} H(z_{n+1}) & \ldots & H(z_{n+n}) \end{bmatrix}$$

Data-driven (nonintrusive) construction of $\tilde{\boldsymbol{\Sigma}}$

- No access to $\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ required (explicit or implicit)
- Requires transfer function values (frequency-response data)

[Antoulas, Anderson, 1986], [Lefteriu, Antoulas, 2010], [Mayo, Antoulas, 2007]

Loewner: Interpolatory model reduction (cont'd)

Loewner framework derives $\tilde{\Sigma}$ directly from $H(z_1), \ldots, H(z_m)$ with

$$\mathbb{L}_{ij} = \frac{H(z_i) - H(z_{n+j})}{z_i - z_{n+j}}, \quad \mathbb{L}_{ij}^{(s)} = \frac{z_i H(z_i) - z_{n+j} H(z_{n+j})}{z_i - z_{n+j}}, \qquad i, j = 1, \ldots, n$$

Reduced operators of $\tilde{\Sigma}$ are

$$\tilde{\boldsymbol{E}} = -\mathbb{L}, \quad \tilde{\boldsymbol{A}} = -\mathbb{M}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} H(z_1) & \ldots & H(z_n) \end{bmatrix}^T,$$
$$\text{and} \quad \tilde{\boldsymbol{C}} = \begin{bmatrix} H(z_{n+1}) & \ldots & H(z_{n+n}) \end{bmatrix}$$

Data-driven (nonintrusive) construction of $\tilde{\Sigma}$

- No access to $\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ required (explicit or implicit)
- Requires transfer function values (frequency-response data)

[Antoulas, Anderson, 1986], [Lefteriu, Antoulas, 2010], [Mayo, Antoulas, 2007], [Antoulas, 2016], [Gustavsen, Semlyen, 1999], [Drmac, Gugercin, Beattie, 2015], [Antoulas, Gosea, Ionita, 2016], [Gosea, Antoulas, 2018], [Schulze, Unger, Beattie, Gugercin, 2018], [Benner, Goyal, Van Dooren, 2019], ...

Loewner: (Classical) Loewner reduced model

Given $m = 2n$ interpolation points on unit disc in $\mathbb{C}$

[Antoulas, Anderson, 1986]
[Lefteriu, Antoulas, 2010]
[Mayo, Antoulas, 2007]

$$\{z_1, \ldots, z_m\} = \{\mu_1, \ldots, \mu_n\} \uplus \{\gamma_1, \ldots, \gamma_n\}$$

Evaluate transfer function

$$H(z) = \boldsymbol{C}(z\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}$$

of $\boldsymbol{\Sigma}$ at $z_1, \ldots, z_m$

$$H(z_1), \ldots, H(z_m)$$

Derive Loewner matrices $\mathbb{L} \in \mathbb{C}^{n\times n}$ and $\mathbb{M} \in \mathbb{C}^{n\times n}$

$$\mathbb{L}_{ij} = \frac{H(\mu_i) - H(\gamma_j)}{\mu_i - \gamma_j}, \quad \mathbb{L}^{(s)}_{ij} = \frac{\mu_i H(\mu_i) - \gamma_j H(\gamma_j)}{\mu_i - \gamma_j}, \qquad i, j = 1, \ldots, n$$

Construct reduced system $\tilde{\boldsymbol{\Sigma}}$ with $H(z_i) = \tilde{H}(z_i), i = 1, \ldots, m$

$$\tilde{\boldsymbol{E}} = -\mathbb{L}, \quad \tilde{\boldsymbol{A}} = -\mathbb{L}^{(s)}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} H(\mu_1) & \ldots & H(\mu_n) \end{bmatrix}^T, \quad \text{and} \quad \tilde{\boldsymbol{C}} = \begin{bmatrix} H(\gamma_1) & \ldots & H(\gamma_n) \end{bmatrix}$$

} nonintrusive but requires values of transfer function

Matlab demo

Loewner: Noisy transfer-function values

- Let $\mu \in \mathbb{C}$ and $0 < \sigma \in \mathbb{R}$ to define

$$\epsilon \sim \mathcal{CN}(\mu, \sigma)\,,$$

  where real part $R(\epsilon)$ and imaginary part $I(\epsilon)$ are independent normal with mean $R(\mu)$ and $I(\mu)$, respectively
- Consider $\epsilon_1, \ldots, \epsilon_n \sim \mathcal{CN}(0,1)$ and $\eta_1, \ldots, \eta_n \sim \mathcal{CN}(0,1)$
- Noisy transfer-function values

$$H_\sigma(\mu_i) = H(\mu_i)(1+\sigma\epsilon_i)\,, \qquad H_\sigma(\gamma_i) = H(\gamma_i)(1+\sigma\eta_i)$$

- Noise pollutes transfer-function values in a relative sense (measurement error relative to value)
- Define noisy Loewner matrices

$$\hat{L}_{ij} = \frac{H_\sigma(\mu_i) - H_\sigma(\gamma_j)}{\mu_i - \gamma_j} \qquad \hat{L}^{(s)}_{ij} = \frac{\mu_i H_\sigma(\mu_i) - \gamma_j H_\sigma(\gamma_j)}{\mu_i - \gamma_j}$$

[Drmac, P. Learning low-dimensional dynamical-system models from noisy frequency-response data with Loewner rational interpolation. arXiv:1910.00110, 2019.]

Loewner: Structure in noise

- Key observation is that

$$\hat{L} = L + \sigma \delta L \,,$$

with

$$\delta L_{i,j} = \frac{H(\mu_i)\epsilon_i - H(\gamma_i)\eta_j}{\mu_i - \gamma_j}$$

- Similar decompositions possible for $\hat{L}_\sigma^{(s)}$ and $\hat{B}$ and $\hat{C}$

- Building on this structure in the noise, the following holds:

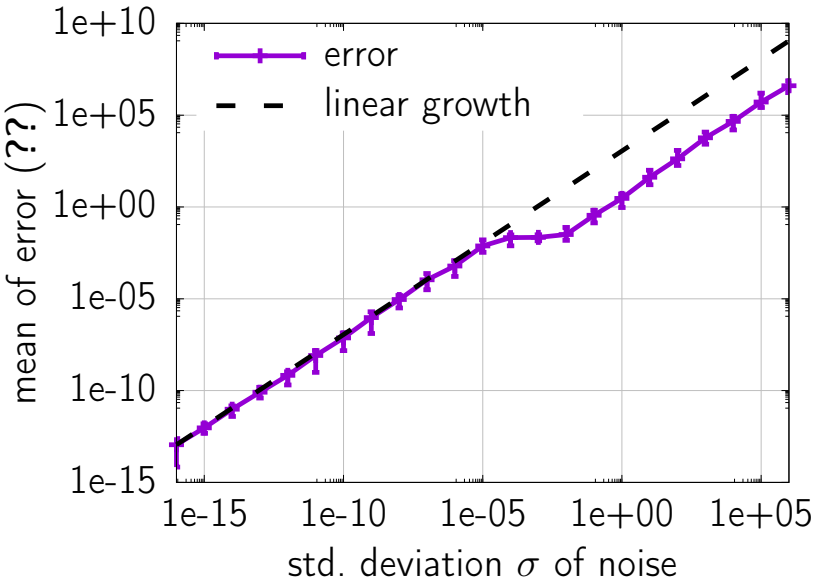
Under certain assumptions on the size of the standard deviation σ*, there exists a constant* $C_s > 0$ *such that*

$$|\tilde{H}(s) - \hat{H}(s)| \leq C_s \sigma \,,$$

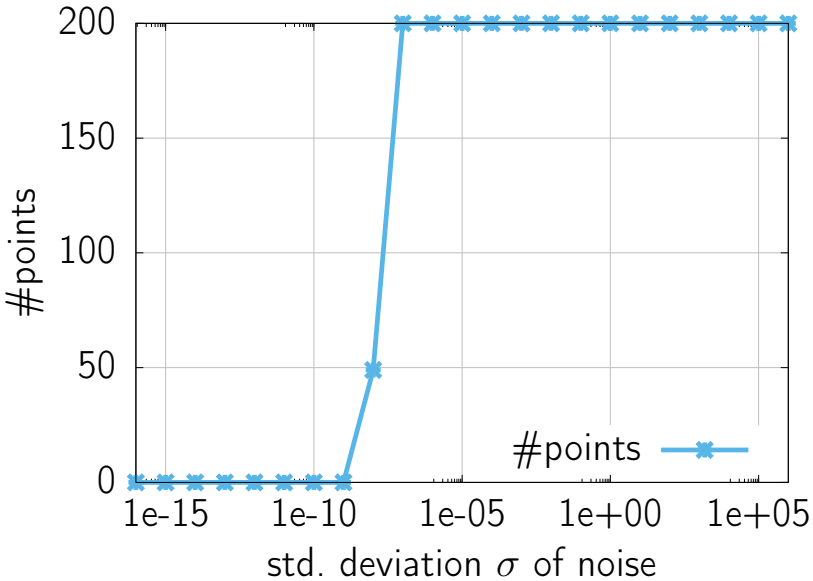
with probability at least $1 - 4\exp(-n/2)$

[Drmac, P. Learning low-dimensional dynamical-system models from noisy frequency-response data with Loewner rational interpolation. arXiv:1910.00110, 2019.]

Loewner: Numerical example with noisy data



(a) error, dimension $n = 20$



(b) assumption violated, dimension $n = 20$

- CD player example (same as in Matlab demo)
- Linear growth of error with standard deviation σ until assumptions are violated
- Results indicate that bound is conservative

[Drmac, P. Learning low-dimensional dynamical-system models from noisy frequency-response data with Loewner rational interpolation. arXiv:1910.00110, 2019.]

Loewner: Interpolatory model reduction (cont'd)

Loewner framework derives $\tilde{\Sigma}$ directly from $H(z_1), \ldots, H(z_m)$ with

$$\mathbb{L}_{ij} = \frac{H(z_i) - H(z_{n+j})}{z_i - z_{n+j}}, \quad \mathbb{L}_{ij}^{(s)} = \frac{z_i H(z_i) - z_{n+j} H(z_{n+j})}{z_i - z_{n+j}}, \qquad i, j = 1, \ldots, n$$

Reduced operators of $\tilde{\Sigma}$ are

$$\tilde{\boldsymbol{E}} = -\mathbb{L}, \quad \tilde{\boldsymbol{A}} = -\mathbb{M}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} H(z_1) & \ldots & H(z_n) \end{bmatrix}^T,$$
$$\text{and} \quad \tilde{\boldsymbol{C}} = \begin{bmatrix} H(z_{n+1}) & \ldots & H(z_{n+n}) \end{bmatrix}$$

Data-driven (nonintrusive) construction of $\tilde{\Sigma}$

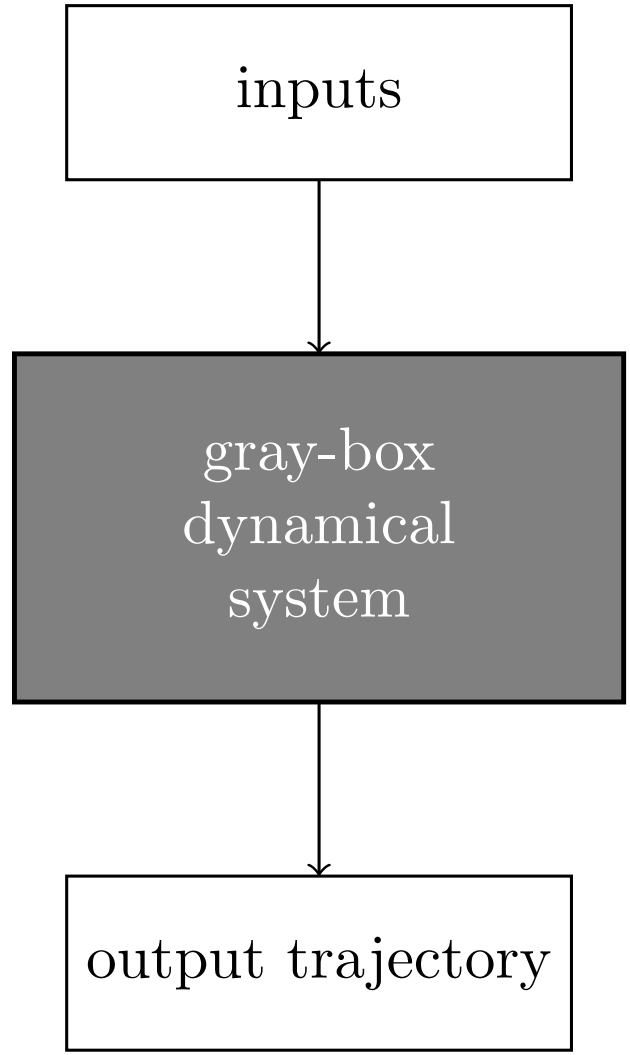
- No access to $\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ required (explicit or implicit)
- Requires transfer function values (frequency-response data)

[Antoulas, Anderson, 1986], [Lefteriu, Antoulas, 2010], [Mayo, Antoulas, 2007], [Antoulas, 2016], [Gustavsen, Semlyen, 1999], [Drmac, Gugercin, Beattie, 2015], [Antoulas, Gosea, Ionita, 2016], [Gosea, Antoulas, 2018], [Schulze, Unger, Beattie, Gugercin, 2018], [Benner, Goyal, Van Dooren, 2019], ...

Loewner: Problem formulation

Can time-step LTI model $\boldsymbol{\Sigma}$ for $K \in \mathbb{N}$ time steps

- Given inputs $\boldsymbol{u} = [u_0, u_1, \ldots, u_{K-1}]^T \in \mathbb{C}^K$
- Compute outputs $\boldsymbol{y} = [y_0, y_1, \ldots, y_{K-1}]^T \in \mathbb{C}^K$ via time stepping
- Transfer function H *un*available ($\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ unavailable as well); no states



Goal: Approximate transfer function values from $\boldsymbol{y}$

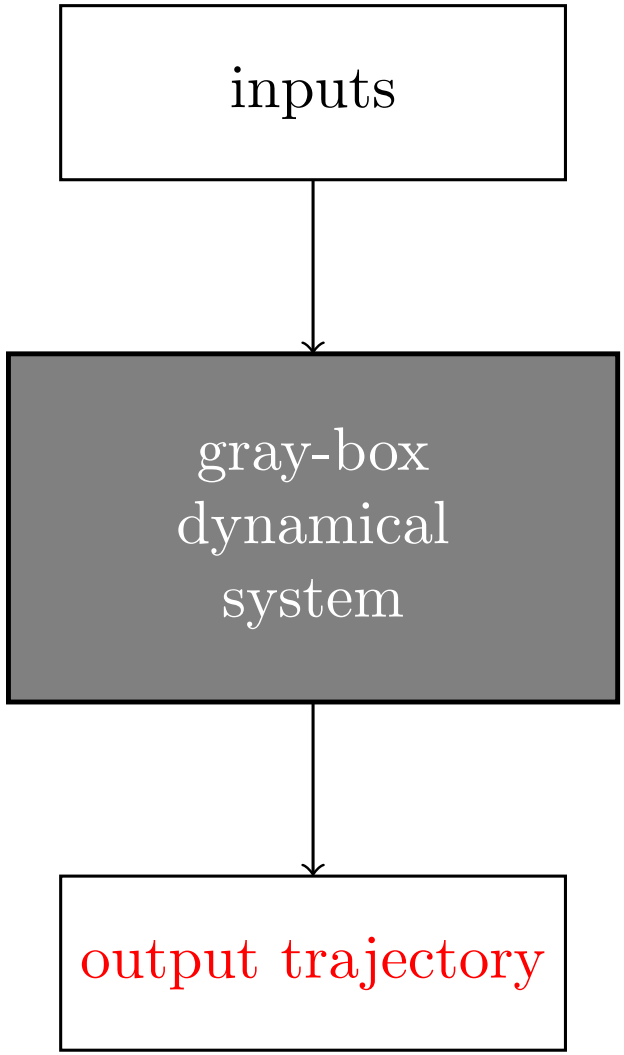
- Given are interpolation points $z_1, \ldots, z_m$
- Perform *single* time-domain simulation of $\boldsymbol{\Sigma}$ until steady state is reached
- Derive approximate $\hat{H}(z_1), \ldots, \hat{H}(z_m)$ values from output trajectory $\boldsymbol{y}$
- Construct $\hat{\boldsymbol{\Sigma}}$ to approximate (classical) Loewner $\tilde{\boldsymbol{\Sigma}}$

$$\underbrace{H(z_i)}_{\text{full model}} = \underbrace{\tilde{H}(z_i)}_{\substack{\text{classical} \\ \text{Loewner model}}} \approx \underbrace{\hat{H}(z_i)}_{\substack{\text{time-domain} \\ \text{Loewner model}}}, \qquad i = 1, \ldots, m$$

Loewner: Problem formulation

Can time-step LTI model $\boldsymbol{\Sigma}$ for $K \in \mathbb{N}$ time steps

- Given inputs $\boldsymbol{u} = [u_0, u_1, \ldots, u_{K-1}]^T \in \mathbb{C}^K$
- Compute outputs $\boldsymbol{y} = [y_0, y_1, \ldots, y_{K-1}]^T \in \mathbb{C}^K$ via time stepping
- Transfer function H *un*available ($\boldsymbol{E}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ unavailable as well); no states



Goal: Approximate transfer function values from $\boldsymbol{y}$

- Given are interpolation points $z_1, \ldots, z_m$
- Perform *single* time-domain simulation of $\boldsymbol{\Sigma}$ until steady state is reached
- Derive approximate $\hat{H}(z_1), \ldots, \hat{H}(z_m)$ values from output trajectory $\boldsymbol{y}$
- Construct $\hat{\boldsymbol{\Sigma}}$ to approximate (classical) Loewner $\tilde{\boldsymbol{\Sigma}}$

$$\underbrace{H(z_i)}_{\text{full model}} = \underbrace{\tilde{H}(z_i)}_{\substack{\text{classical} \\ \text{Loewner model}}} \approx \underbrace{\hat{H}(z_i)}_{\substack{\text{time-domain} \\ \text{Loewner model}}}, \qquad i = 1, \ldots, m$$

Loewner: Laplace (or Z-) transform

Input/output relationship in time domain (convolution)

$$y_k = \sum_{l=0}^{k} h_l u_{k-l}$$

with impulse/response

$$h_k = \boldsymbol{C}(\boldsymbol{E}^{-1}\boldsymbol{A})^{k-1}(\boldsymbol{E}^{-1}\boldsymbol{B}), \quad k > 0, h_0 = 0$$

Z-Transform of outputs $\{y_k\}_{k=1}^{\infty}$

$$Y(z) = \sum_{k=0}^{\infty} y_k z^{-k}$$

Z-Transform of impulse/response $\{h_k\}_{k=1}^{\infty}$

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k}$$

Input/output relationship in frequency domain

$$Y(z) = H(z)U(z)$$

Loewner: Output of LTI system

Define points on the unit circle

$$q_i = \mathrm{e}^{\frac{2\pi j}{K} i}, \qquad i = 0, \ldots, K-1$$

Represent input in discrete Fourier coefficients $\boldsymbol{U} = [U_0, \ldots, U_{K-1}]^T$

$$u_k = \sum_{i=0}^{K-1} U_i q_i^k, \qquad k = 0, \ldots, K-1$$

W.l.o.g. have set $\mathcal{I}_r = \{1, \ldots, r\}$ of non-zero Fourier coefficients

$$u_k = \sum_{i=1}^{r} U_i q_i^k, \qquad k = 0, \ldots, K-1$$

Output is convolution of impulse response h_k and input u_k

$$y_k = \sum_{l=0}^{k} h_l u_{k-l} = \sum_{l=0}^{k} h_l \underbrace{\sum_{i=1}^{r} U_i q_i^{k-l}}_{u_{k-l}} = \sum_{i=1}^{r} U_i q_i^k \underbrace{\sum_{l=0}^{k} h_l q_i^{-l}}_{=H_k(q_i)}, \qquad k = 0, \ldots, K-1$$

Loewner: Asymptotic properties of $H_k(z)$

Relationship between output y_k and $H_k(q_i)$

$$y_k = \sum_{i=1}^{r} U_i q_i^k \underbrace{\sum_{l=0}^{k} h_l q_i^{-l}}_{=H_k(q_i)} = \sum_{i=1}^{r} H_k(q_i) U_i q_i^k \,, \qquad k = 0, \ldots, K-1$$

Transfer function H is z-transform of impulse response

$$H(z) = \sum_{l=0}^{\infty} h_l z^{-l} \,, \qquad z \in \mathbb{D}$$

Sequence $(H_k(z))$ converges to $H(z)$ for $z \in \mathbb{D}$

$$|H(z) - H_k(z)| \leq c\rho^k$$

Decay of error $|H(z) - H_k(z)|$ depends on spectral radius ρ of $\boldsymbol{E}^{-1}\boldsymbol{A}$

- Problem-dependent rate of decay of error $|H(z) - H_k(z)|$
- Slow decay of error if many time steps to reach steady state

Loewner: Regression problem

Relationship between output y_k and $H_k(q_i)$

$$y_k = \sum_{i=1}^{r} H_k(q_i) U_i q_i^k \,, \qquad k = 0, \ldots, K-1$$

Solve for approximate transfer function values $\hat{H}_1, \ldots, \hat{H}_r \in \mathbb{C}$

$$\underset{\hat{H}_1, \ldots, \hat{H}_r \in \mathbb{C}}{\arg\min} \sum_{k=k_{\min}}^{K-1} \left(\underbrace{y_k}_{\text{output}} - \sum_{i=1}^{r} \hat{H}_i \underbrace{U_i q_i^k}_{\substack{\text{non-zero} \\ \text{Fourier} \\ \text{component}}} \right)^2$$

$\Rightarrow$ Note that dim of optimization problem grows with r
For tolerance $\epsilon > 0$, select value $k_{\min} \in \mathbb{N}$ such that

$$|H(q_i) - H_{k_{\min}}(q_i)| < \epsilon \,, \qquad i = 1, \ldots, r$$

- Controls the time step from which on $H_k(q_i)$ sufficiently accurate
- Asymptotic analysis confirms that $k_{\min}$ is problem-dependent
- If $k_{\min} \leq K - r$, then system overdetermined

Loewner: Time-domain Loewner algorithm

Time-domain Loewner approach

1. Time-step full model $\boldsymbol{\Sigma}$ for input $\boldsymbol{u}$ to obtain output $\boldsymbol{y}$
2. Select value $k_{\min}$
3. Determine indices $\{i_1, \ldots, i_r\}$ of non-zero Fourier coefficients of $\boldsymbol{u}$
4. Solve for approximate transfer function values $\hat{H}_1, \ldots, \hat{H}_r$
5. Select interpolation points $z_1, \ldots, z_m \subset \{q_{i_1}, \ldots, q_{i_r}\}$
6. Use Loewner with $\hat{H}_1, \ldots, \hat{H}_m$ to derive $\hat{\boldsymbol{\Sigma}}$ with

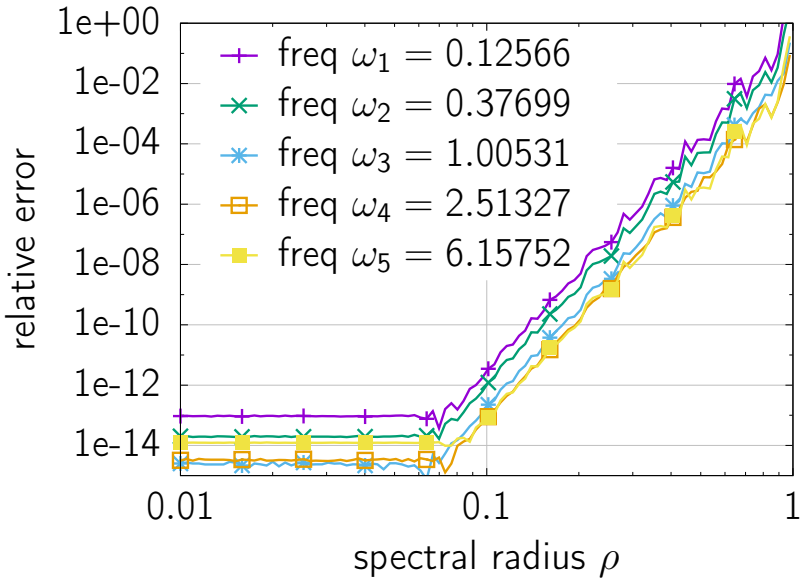
$$H(z_i) \approx \hat{H}(z_i)\,, \qquad i = 1, \ldots, m$$

Choice of interpolation points

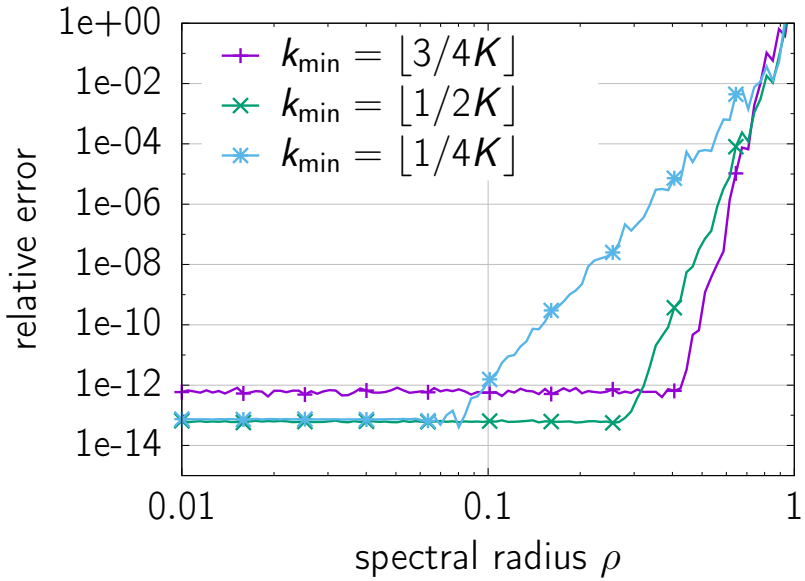
- Restricted by non-zero Fourier coefficients of input
- Number of time steps K determines frequency range

$$\left[\frac{2\pi}{K}, \frac{2\pi(K-1)}{K}\right] \subset \mathbb{R}$$

Loewner: Numerical results



(a) dependence on ρ, $K = 50$



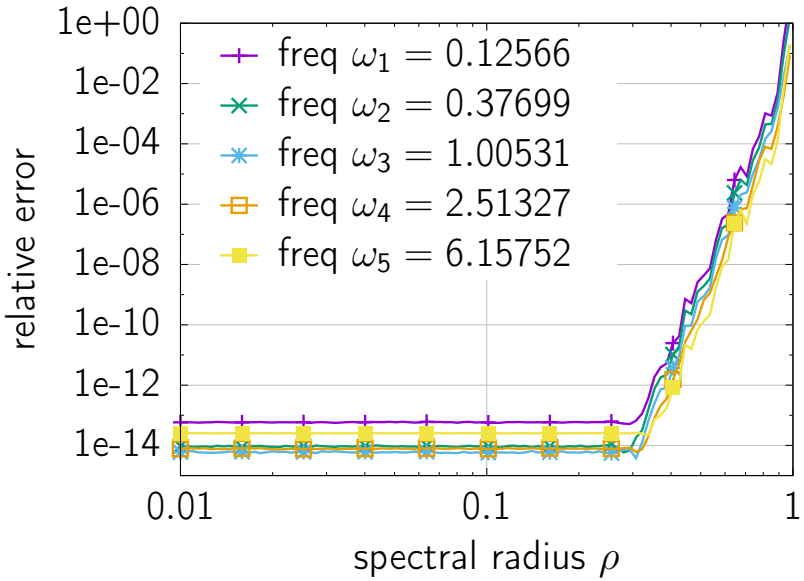
(b) dependence on $k_{\min}$, $K = 50$

- Synthetic example where we can control ρ
- Relative error of approximate transfer function values

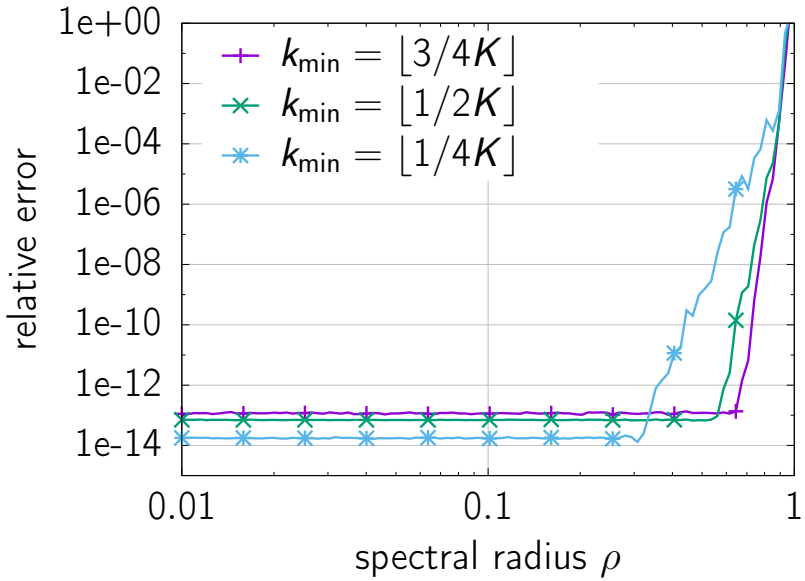
$$\mathrm{err}_{\mathrm{rel}}(\hat{H}_l) = \frac{|H(q_{i_l}) - \hat{H}_l|}{|H(q_{i_l})|}, \qquad l = 1, \ldots, m$$

- A large spectral radius leads to larger error for fixed K
- Large $k_{\min}$ avoids early, inaccurate transfer function approximations
- Setting $k_{\min}$ too large, leads to ill-conditioned least-squares problem

Loewner: Numerical results



(a) dependence on ρ, $K = 100$



(b) dependence on $k_{\min}$, $K = 100$

- Synthetic example where we can control ρ
- Relative error of approximate transfer function values

$$\mathrm{err}_{\mathrm{rel}}(\hat{H}_l) = \frac{|H(q_{i_l}) - \hat{H}_l|}{|H(q_{i_l})|}, \qquad l = 1, \ldots, m$$

- A large spectral radius leads to larger error for fixed K
- Large $k_{\min}$ avoids early, inaccurate transfer function approximations
- Setting $k_{\min}$ too large, leads to ill-conditioned least-squares problem

Loewner: Eady example

Eady LTI system

- Order of system is $N = 598$
- Discretize with 4th-order scheme
- Time step size $\delta t = 10^{-1}$ and $K = 10^3$

Time-domain Loewner reduced model

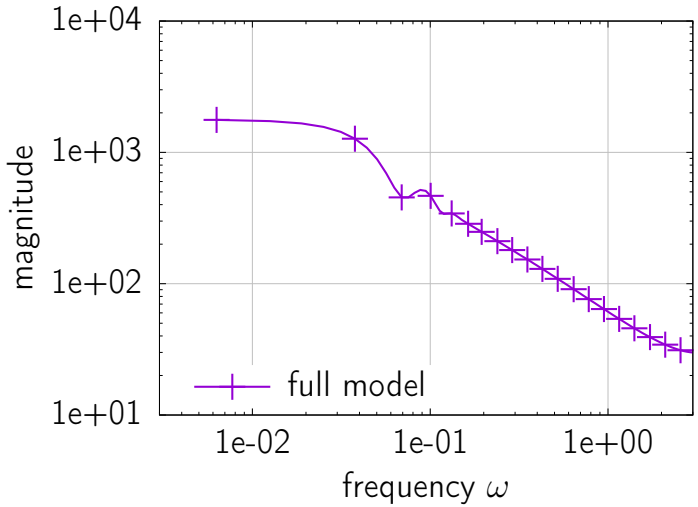
- Dimension of reduced model $n = 5$
- Set $k_{\min} = \lfloor 1/4K \rfloor$
- Select m logarithmic interpolation pts

$$q_{i_1}, \ldots, q_{i_m} \subset \{q_0, \ldots, q_{K-1}\}$$

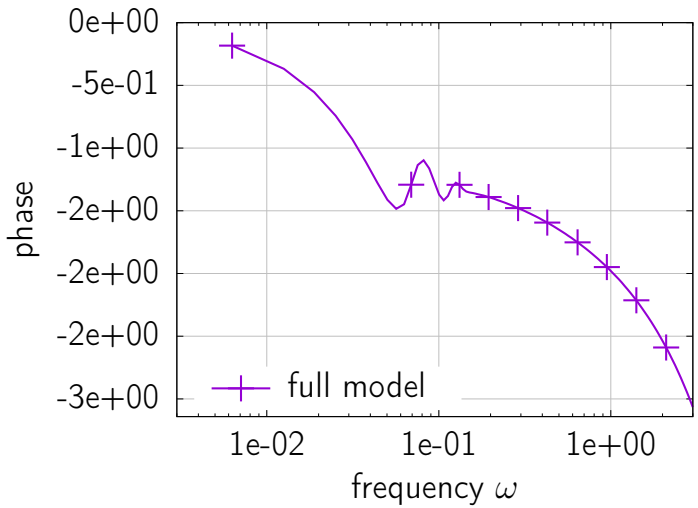
- Input u_k at time $t_k, k = 0, \ldots, K-1$

$$u_k = \frac{1}{K} \sum_{l=1}^{m} (1+j)\, q_{i_l}^k$$

- Simulate full model $\boldsymbol{\Sigma}$ *once*



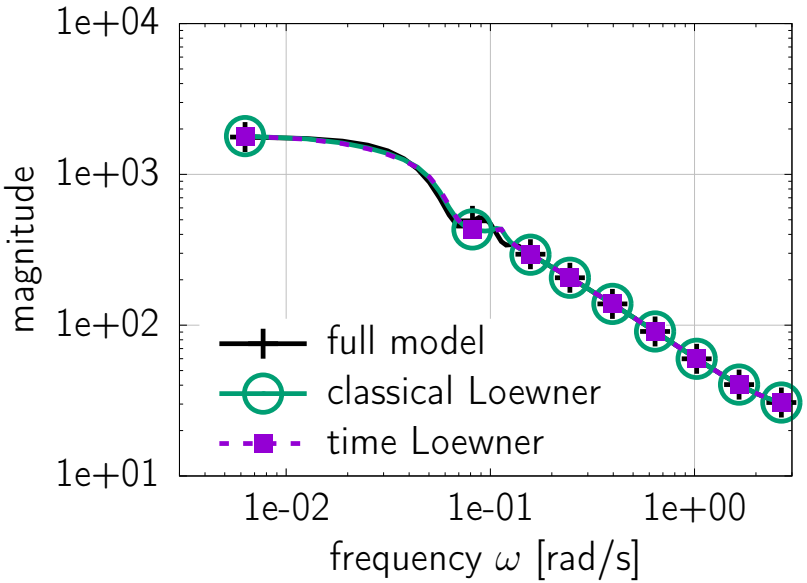
(a) magnitude



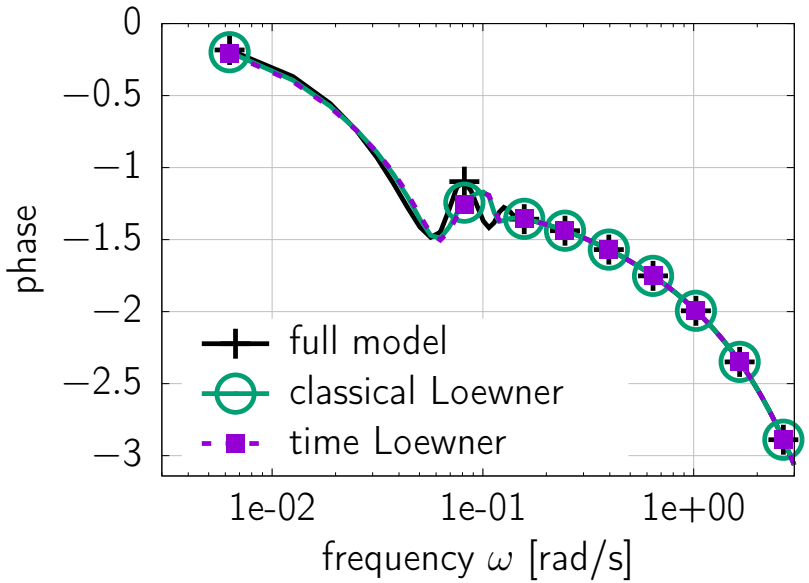
(b) phase

http://slicot.org/20-site/126-benchmark-examples-for-model-reduction

Loewner: Eady example: Transfer function



(a) magnitude



(b) phase

$\frac{\|H-\tilde{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\tilde{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$
1.41×10^{-1}	1.11×10^{-1}	5.42×10^{-2}	3.63×10^{-1}	2.42×10^{-1}	1.85×10^{-1}

- Construct time-domain Loewner from single trajectory
- Magnitude of transfer function matched well; slight difference in phase
- Time-domain (& classical) Loewner model are asymptotically stable

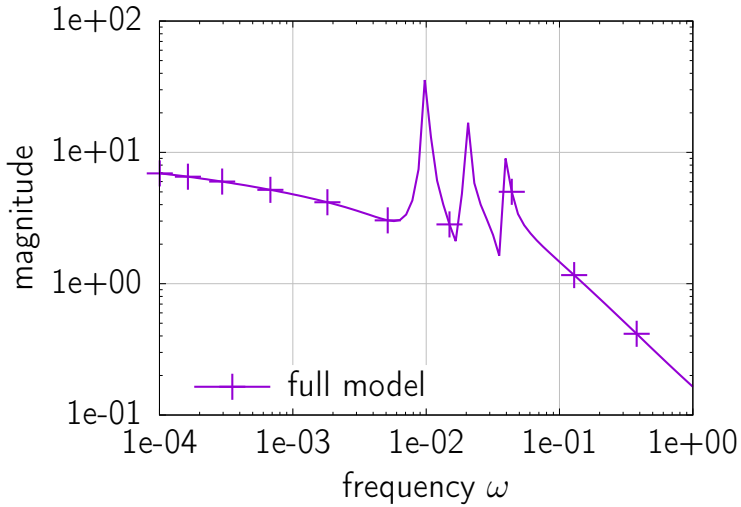
Loewner: Penzl example

Penzl LTI system

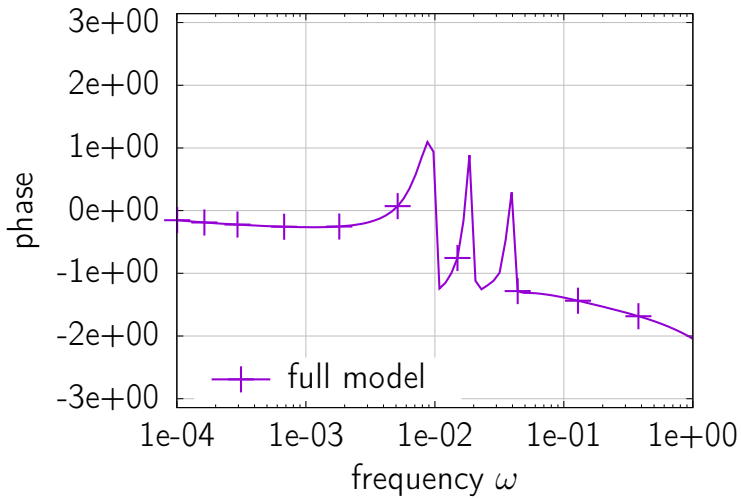
- Order of system is $N = 1006$
- Discretize in time with implicit Euler
- Time step size $\delta t = 10^{-4}$
- Number of time steps $K = 10^6$

Time-domain Loewner reduced model

- Dimension of reduced model $n = 10$
- Set $k_{\min} = \lfloor 1/4K \rfloor$
- Select m logarithmic interpolation pts
- Construct input as in Eady example
- Simulate full model $\boldsymbol{\Sigma}$ *once*



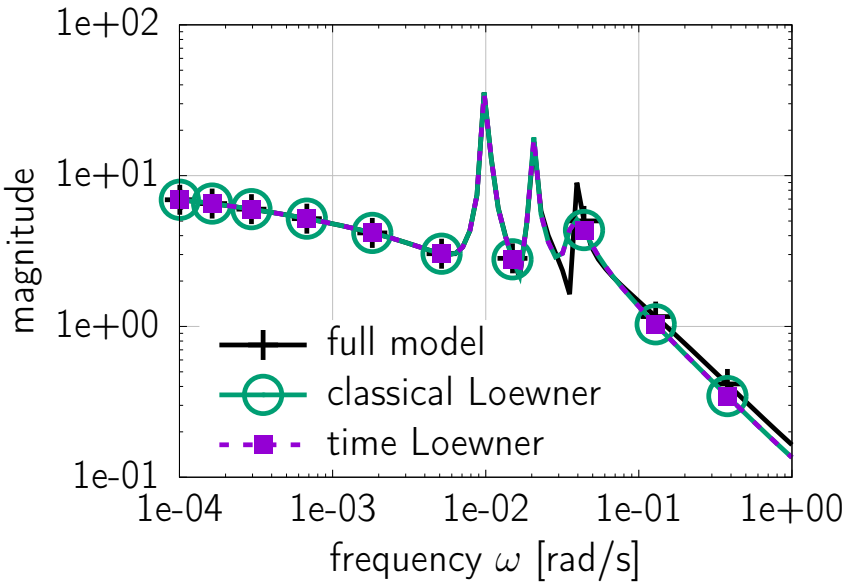
(a) magnitude



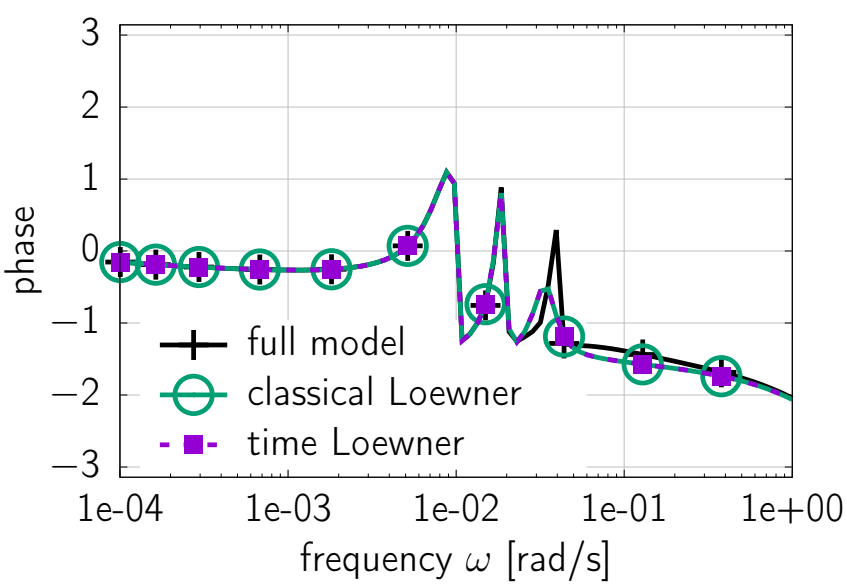
(b) phase

[Penzl, 2006], [Ionita, 2013]

Loewner: Penzl example: Transfer function



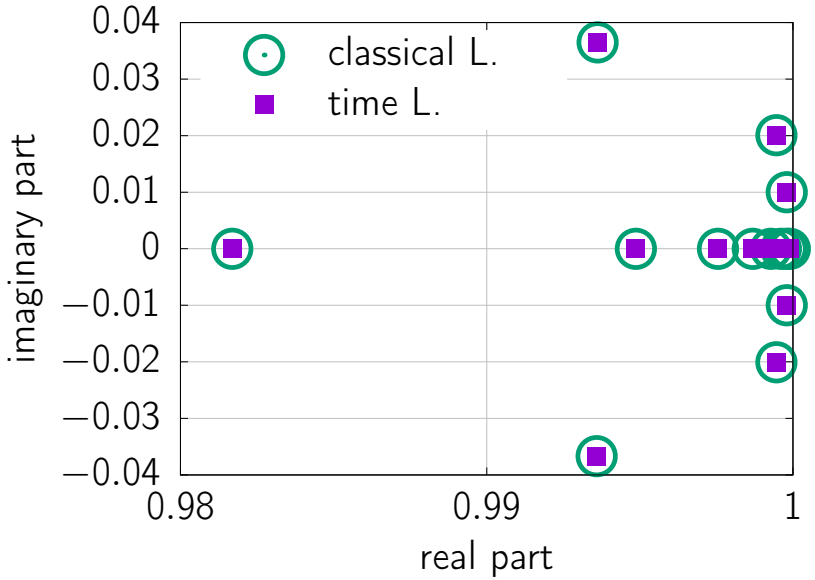
(a) magnitude



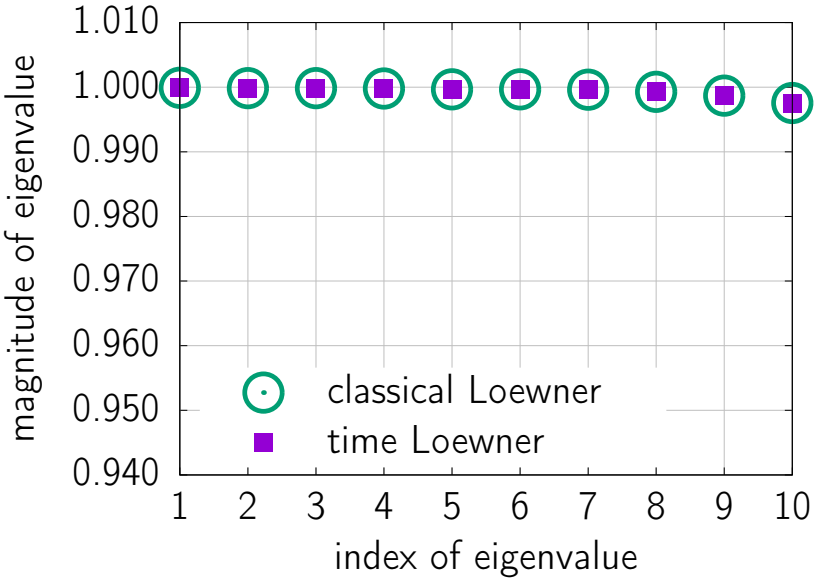
(b) phase

- Number of interpolation points $m = 64$
- Test points logarithmically distributed in range $[10^{-4}, 1]$
- Time-domain Loewner matches classical Loewner model well

Loewner: Penzl example: Poles



(a) eigenvalues



(b) magnitude of eigenvalues

$\frac{\|H-\tilde{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\tilde{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$
5.88×10^{-1}	5.88×10^{-1}	1.07×10^{-4}	2.67×10^{-3}	2.67×10^{-3}	9.97×10^{-6}

- Time-domain Loewner model matches poles of classical Loewner
- Time-domain (& classical) Loewner model are asymptotically stable

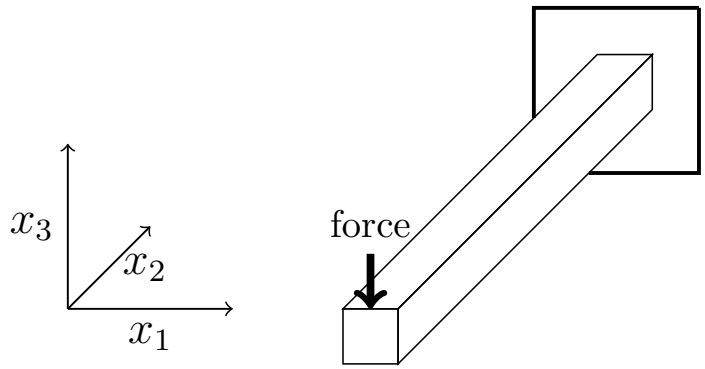
Loewner: Beam example

Cantilever beam

- Full 3D finite-element model of beam
- Force applied at tip of beam
- Implicit Euler, $\delta t = 10^{-4}, K = 10^6$

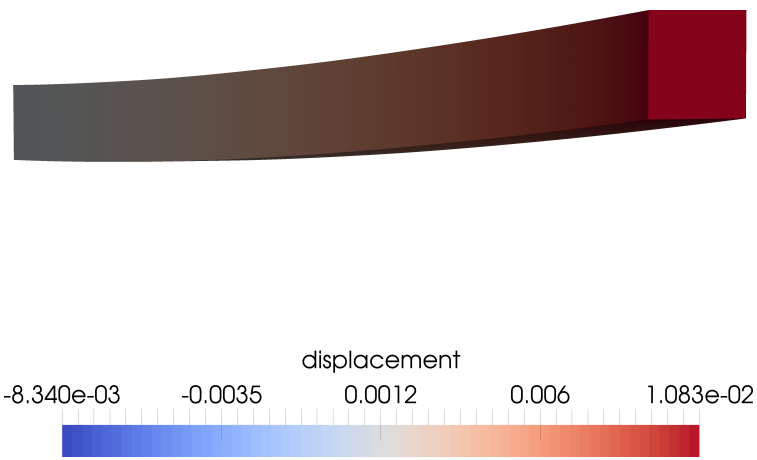
Time-domain Loewner

- Dimension of reduced model $n = 8$
- Select $m = 150$ interpolation points
- Same $k_{\min}$ and input as in Eady
- Simulate full model $\mathbf{\Sigma}$ once

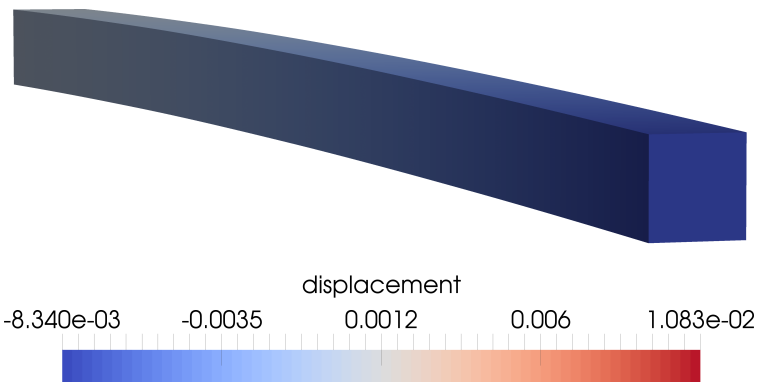


(a) geometry of beam

[Panzer et al., 2009]

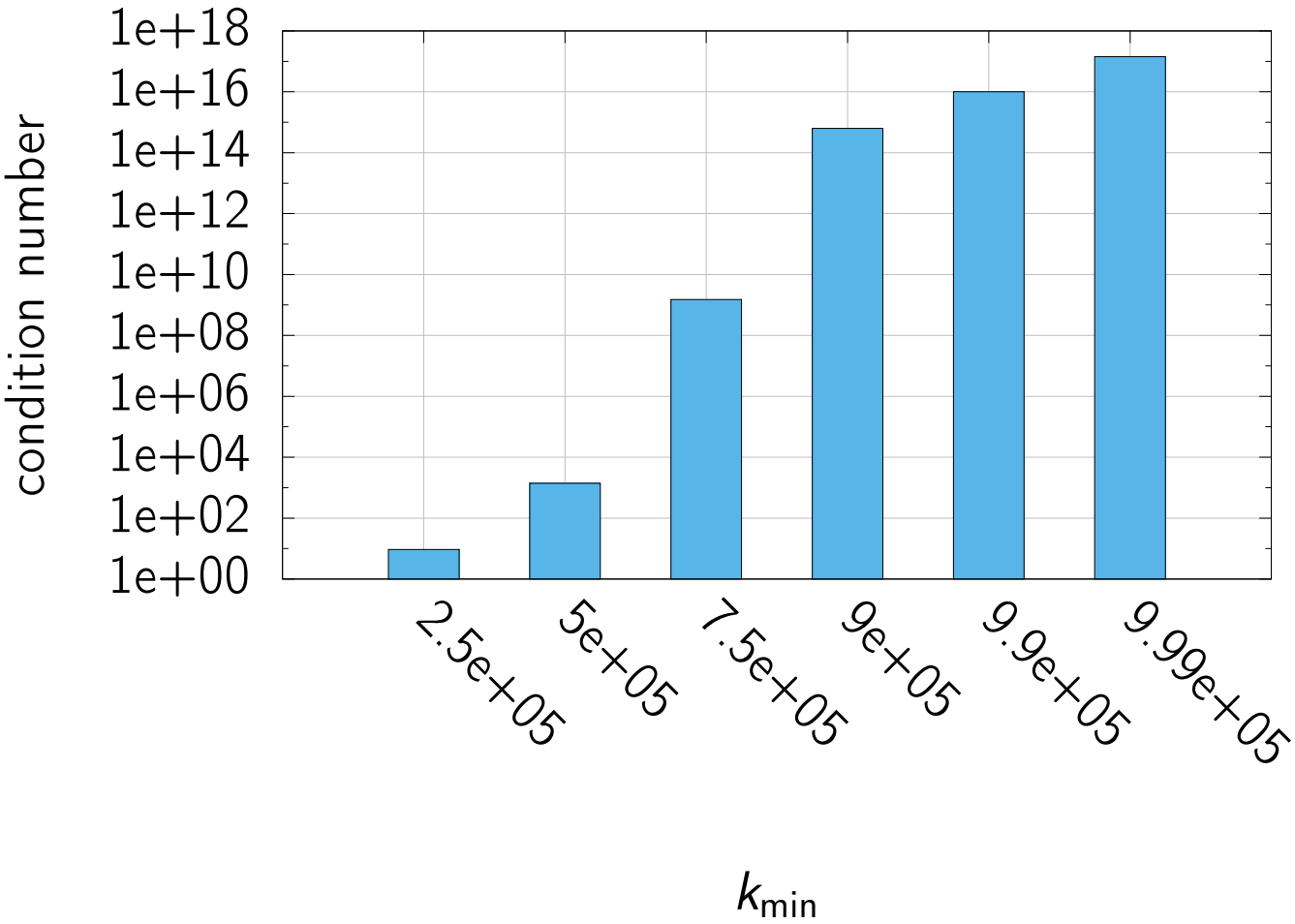


(b) beam at time step 4452



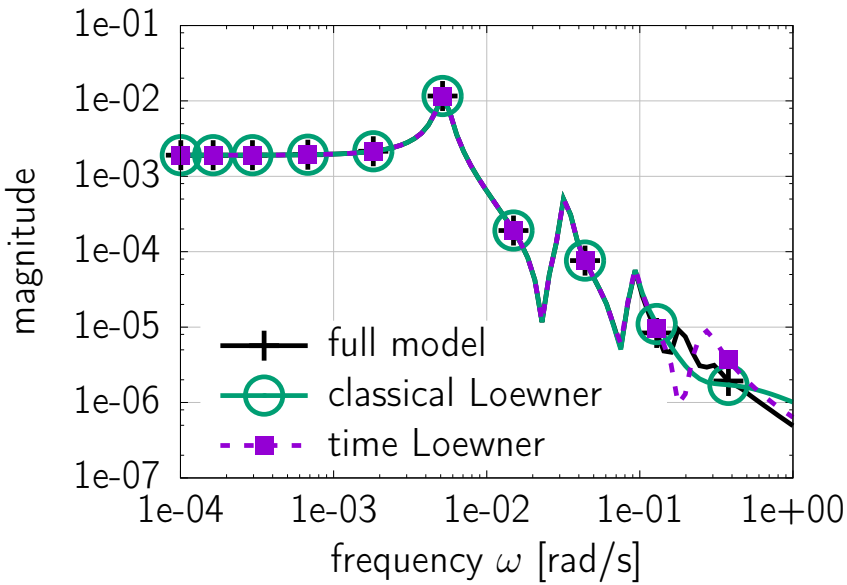
(c) beam at time step 5061

Loewner: Beam example: The $k_{\min}$ value

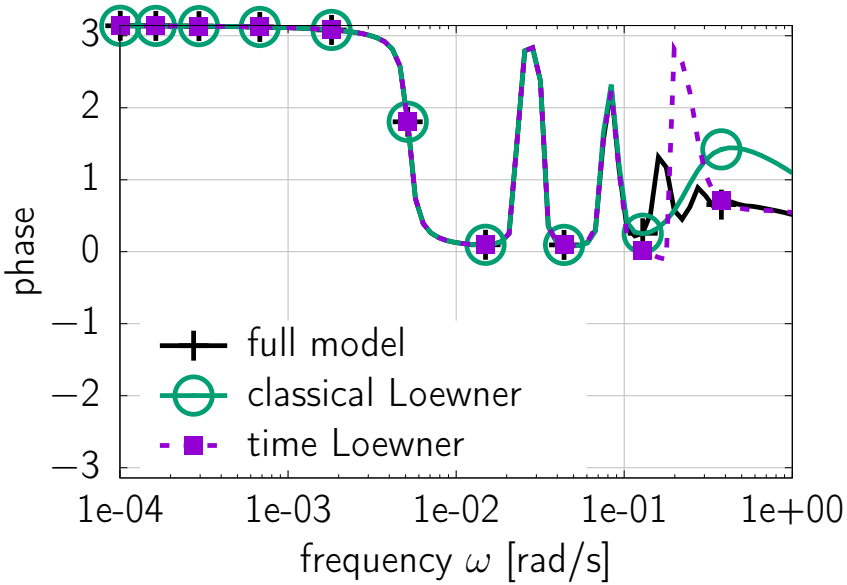


- The $k_{\min}$ significantly influences the condition number
- Conservative choice seems sufficient in practice

Loewner: Beam example: Transfer function



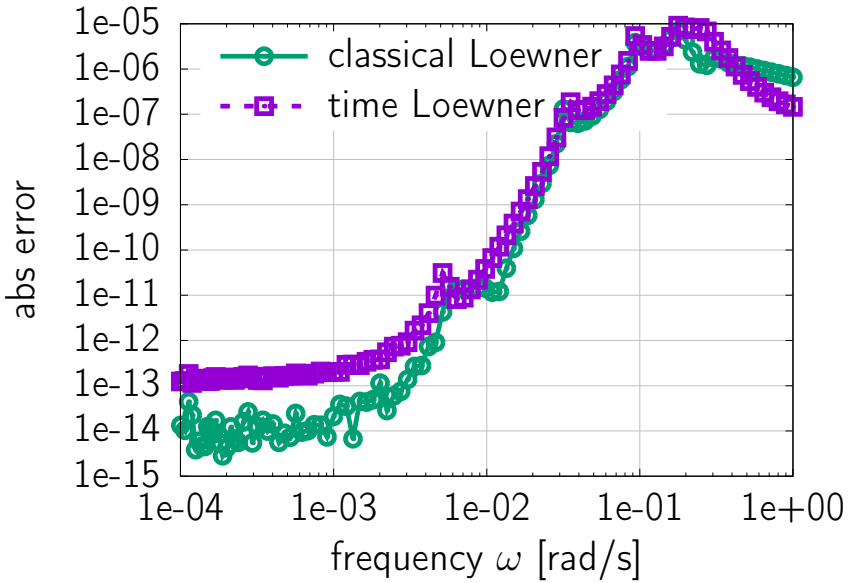
(a) magnitude



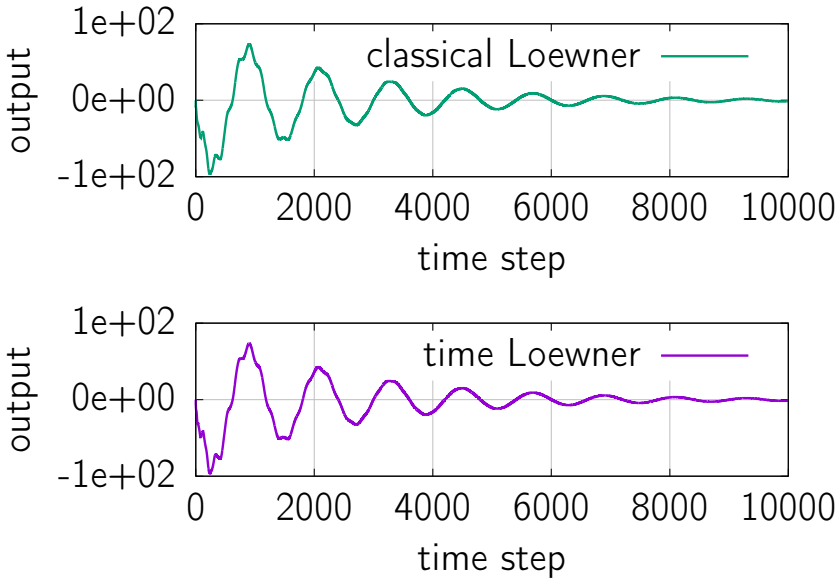
(b) phase

- Time-domain Loewner model matches transfer function well
- Differences can be seen for high frequencies

Loewner: Beam example: Error



(a) absolute error

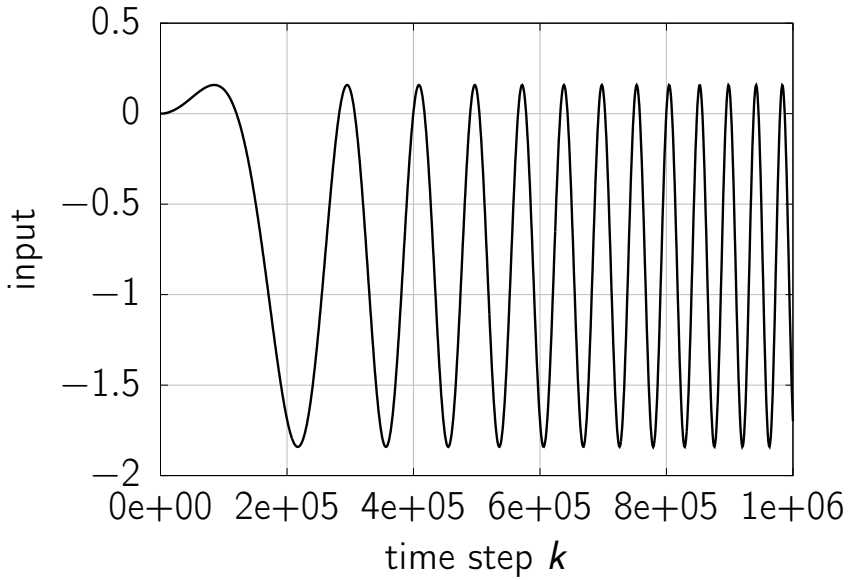


(b) output

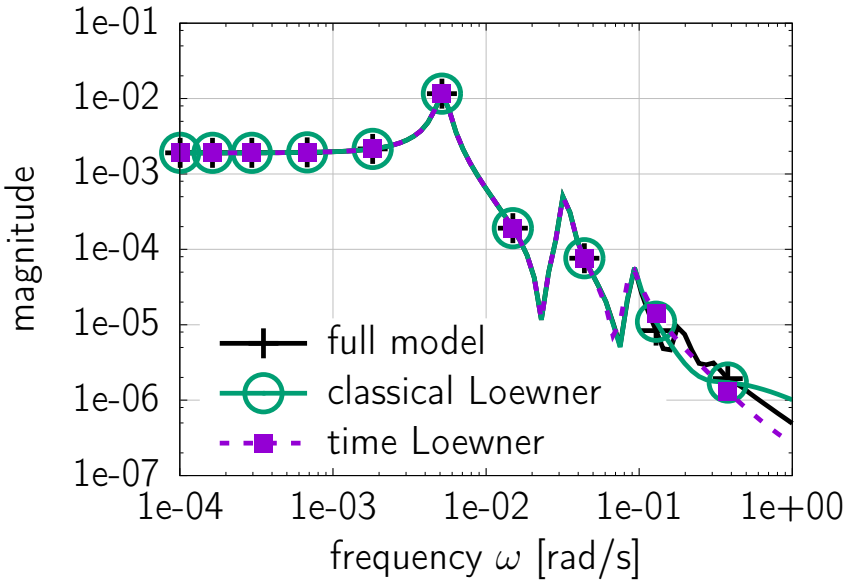
$\frac{\|H-\tilde{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_2}}{\|H\|_{\mathcal{H}_2}}$	$\frac{\|H-\tilde{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|H-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$	$\frac{\|\tilde{H}-\hat{H}\|_{\mathcal{H}_\infty}}{\|H\|_{\mathcal{H}_\infty}}$
2.51×10^{-2}	1.28×10^{-2}	2.12×10^{-2}	2.26×10^{-4}	2.16×10^{-4}	1.21×10^{-4}

- Absolute error is low for low frequencies
- Perform time-domain simulation of reduced model
- Output of time-domain Loewner matches output of classical Loewner

Loewner: Beam example: Input signals



(a) chirp signal



(b) magnitude

- Extract input $\boldsymbol{u}$ from "chirp" signal (non-zero Fourier coefficients)
- Simulate $\boldsymbol{\Sigma}$ at $\boldsymbol{u}$ and construct time-domain Loewner model
- Time-domain Loewner shows similar behavior as for synthetic input

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Generalization error of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification

Outline

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Generalization error of learned reduced models
- Learning from frequency-response data
- **Multi-fidelity uncertainty quantification**

Intro: What is uncertainty quantification?

There are known knowns;
there are things we know we know.

We also know there are known unknowns; that is to say, we know there are some things we do not know.

But there are also unknown unknowns the ones we do not know we do not know.

U.S.Secretary of Defense, Donald Rumsfeld, DoD News Briefing; Feb. 12, 2002

Intro: What is uncertainty quantification?

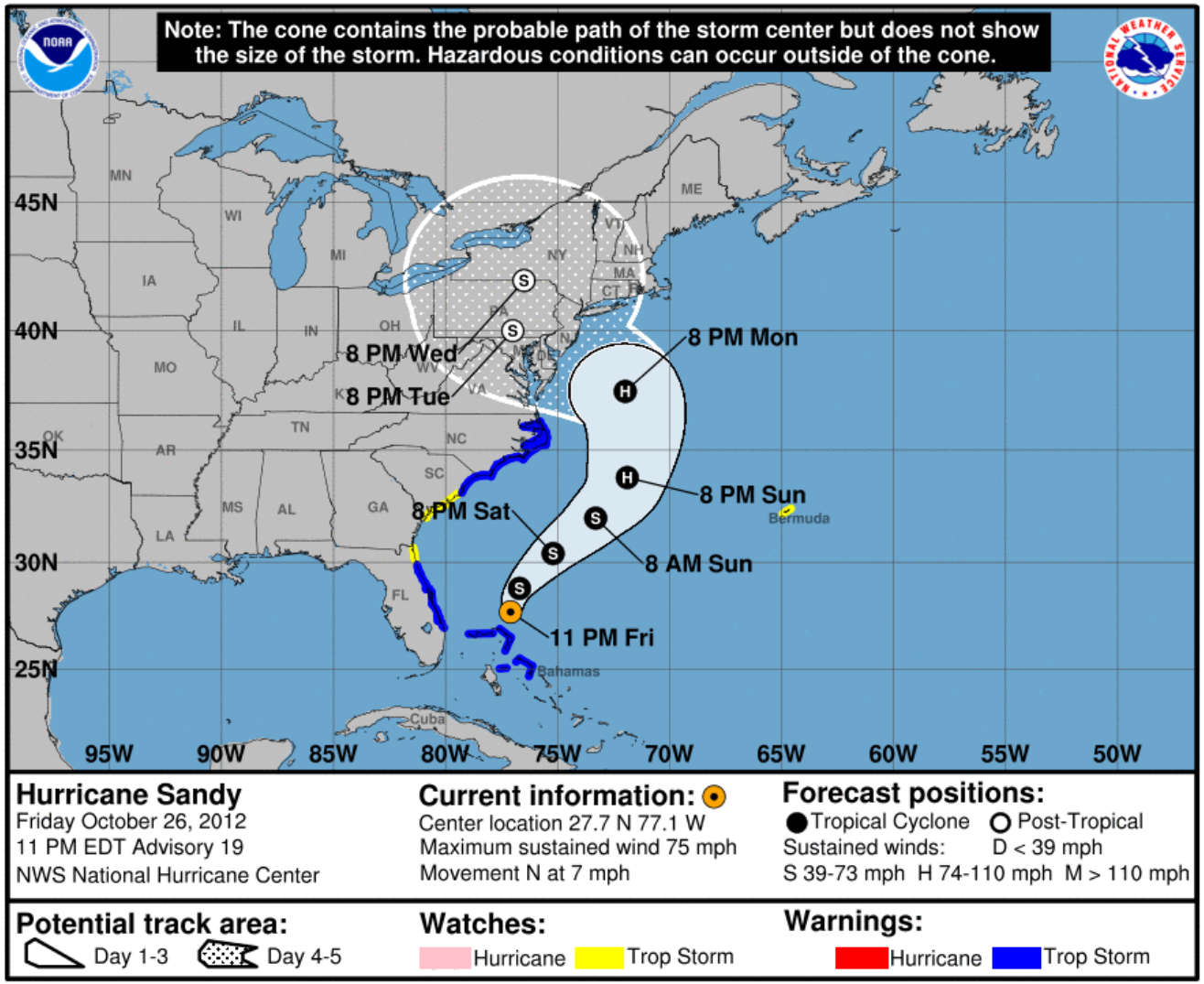
There are known knowns;
there are things we know we know.

We also know there are known unknowns; that is to say, we know there are some things we do not know.

But there are also unknown unknowns the ones we do not know we do not know.

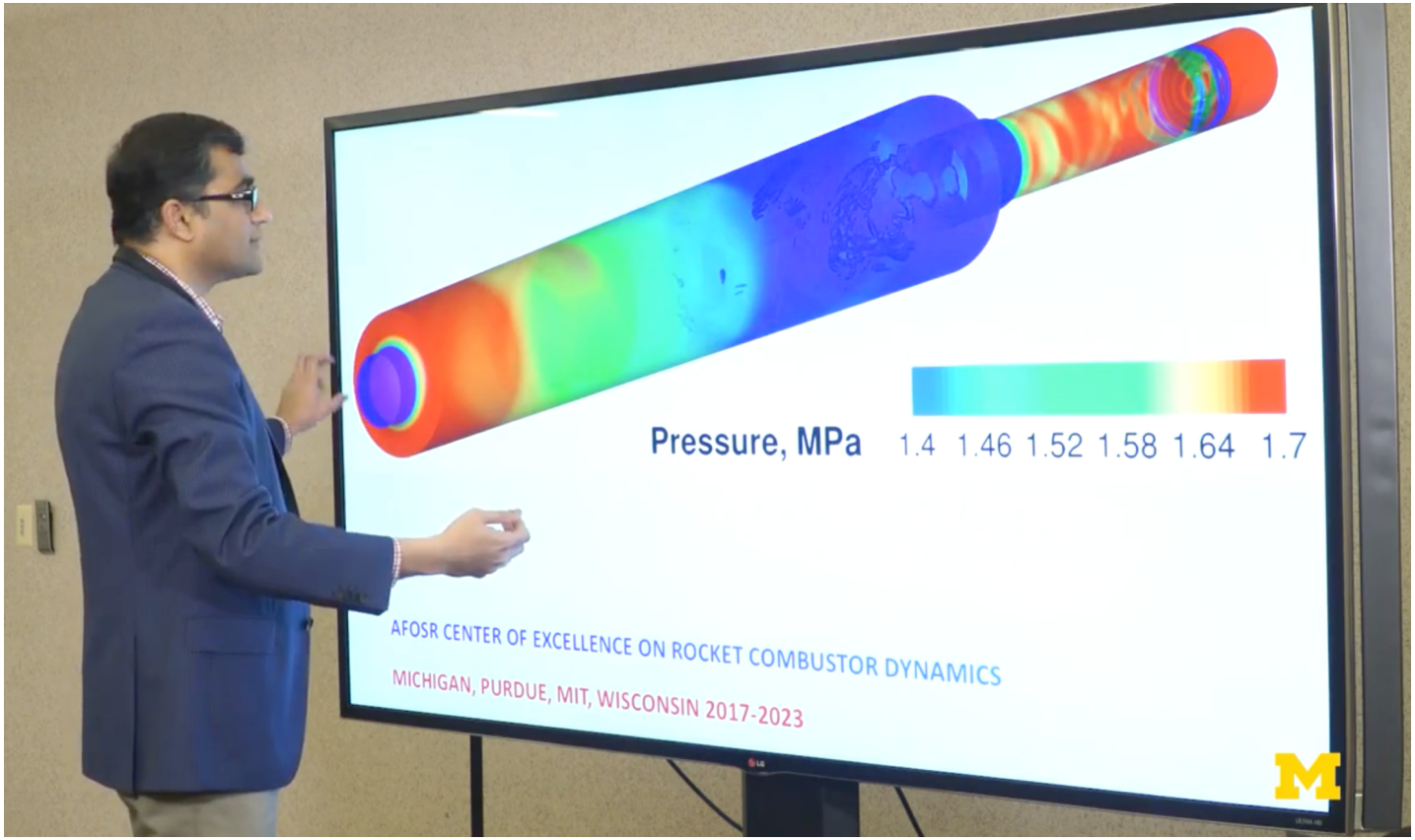
U.S.Secretary of Defense, Donald Rumsfeld, DoD News Briefing; Feb. 12, 2002

Intro: Known unknowns



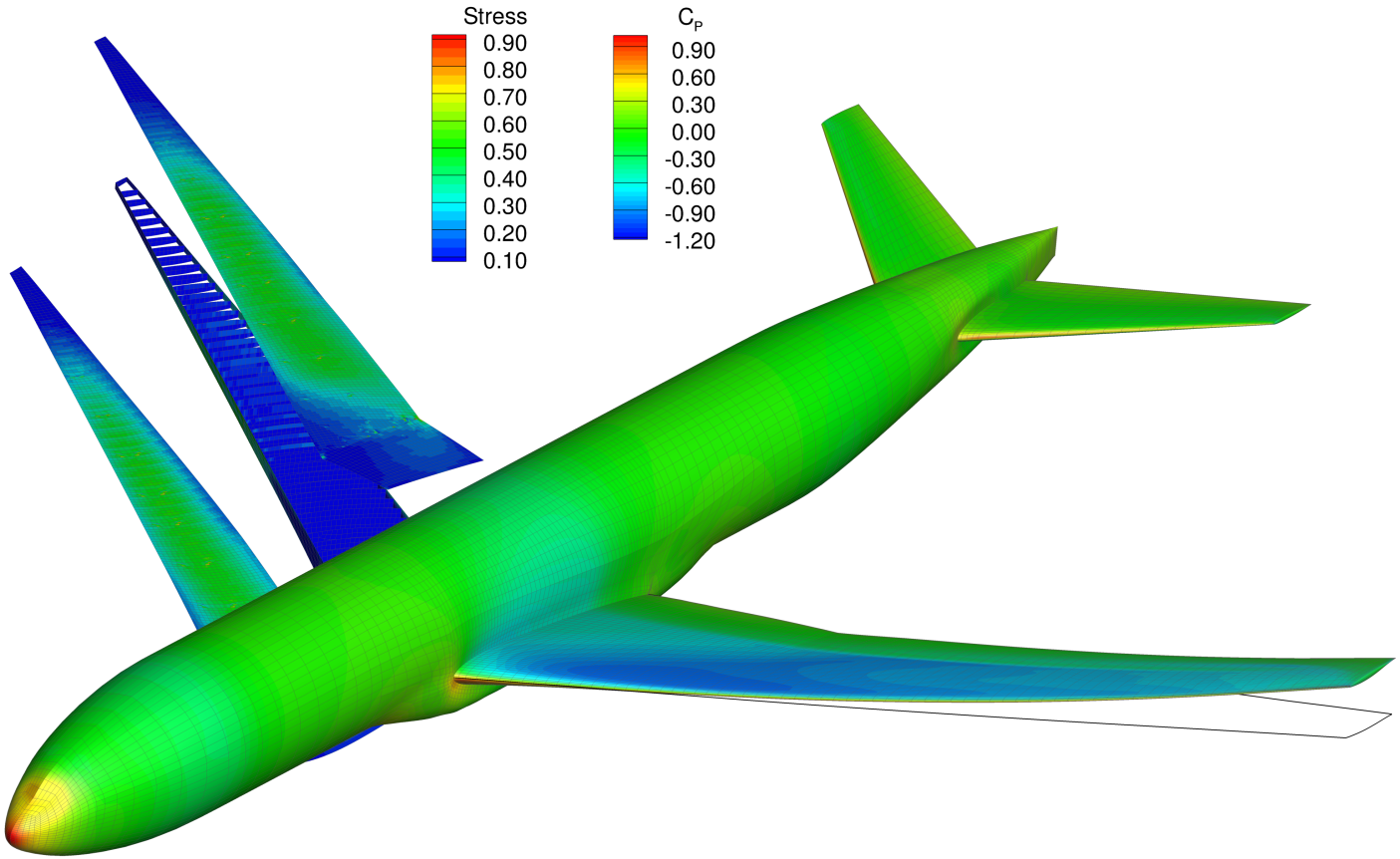
[Figure: NOAA]

Intro: No hope to exhaustively model physics



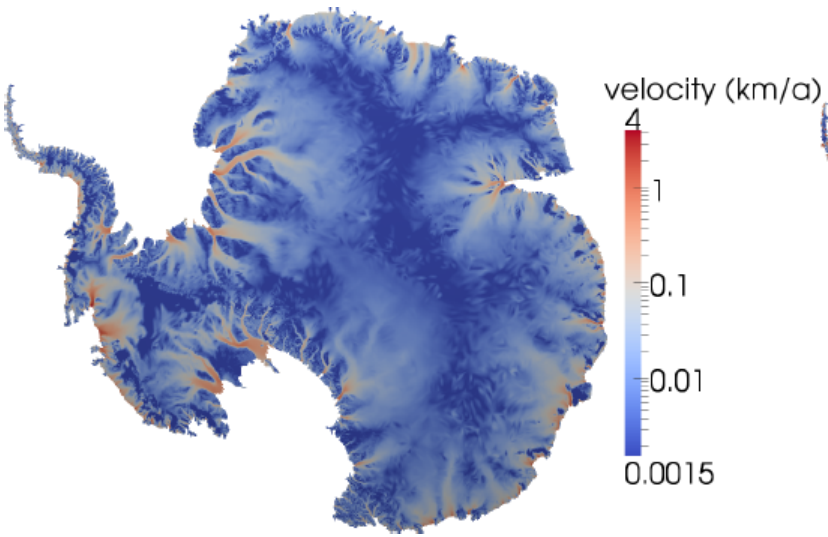
[Figure: University of Michigan]

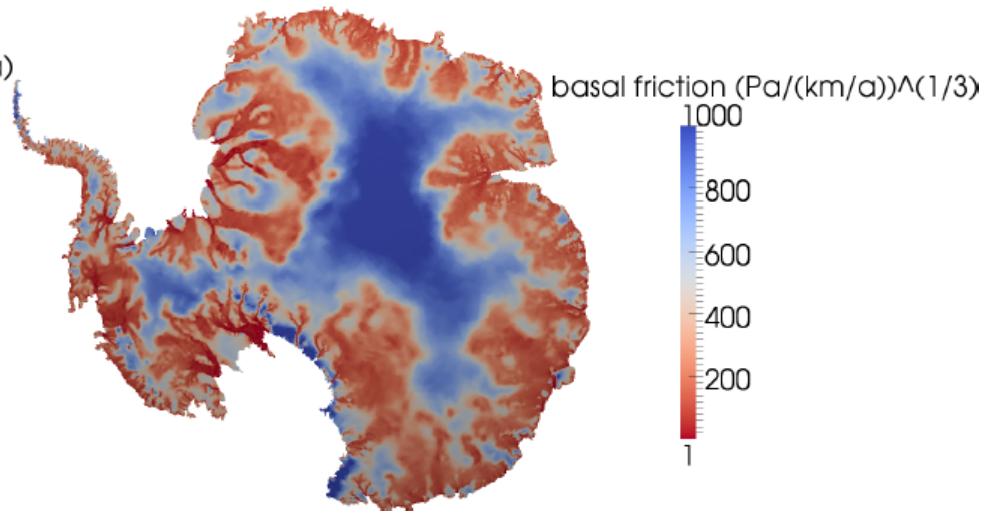
Intro: Rapidly changing dynamics



[Kenway, G. K., Martins, J. R., & Kennedy, G. J. (2014). Aerostructural optimization of the Common Research Model configuration. Group (ADODG), 6(7), 8-9.]

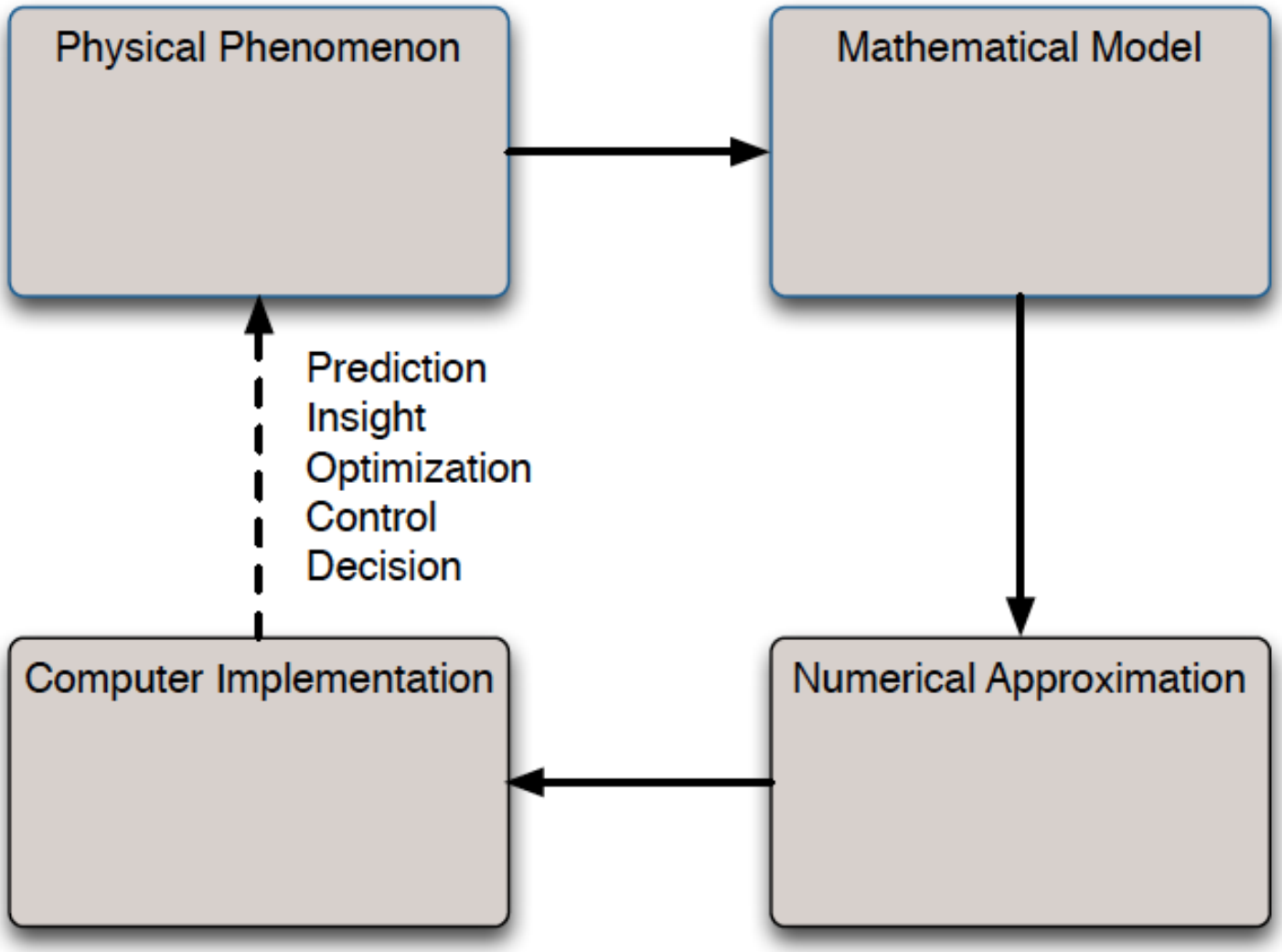
Intro: Uncertainties due to data





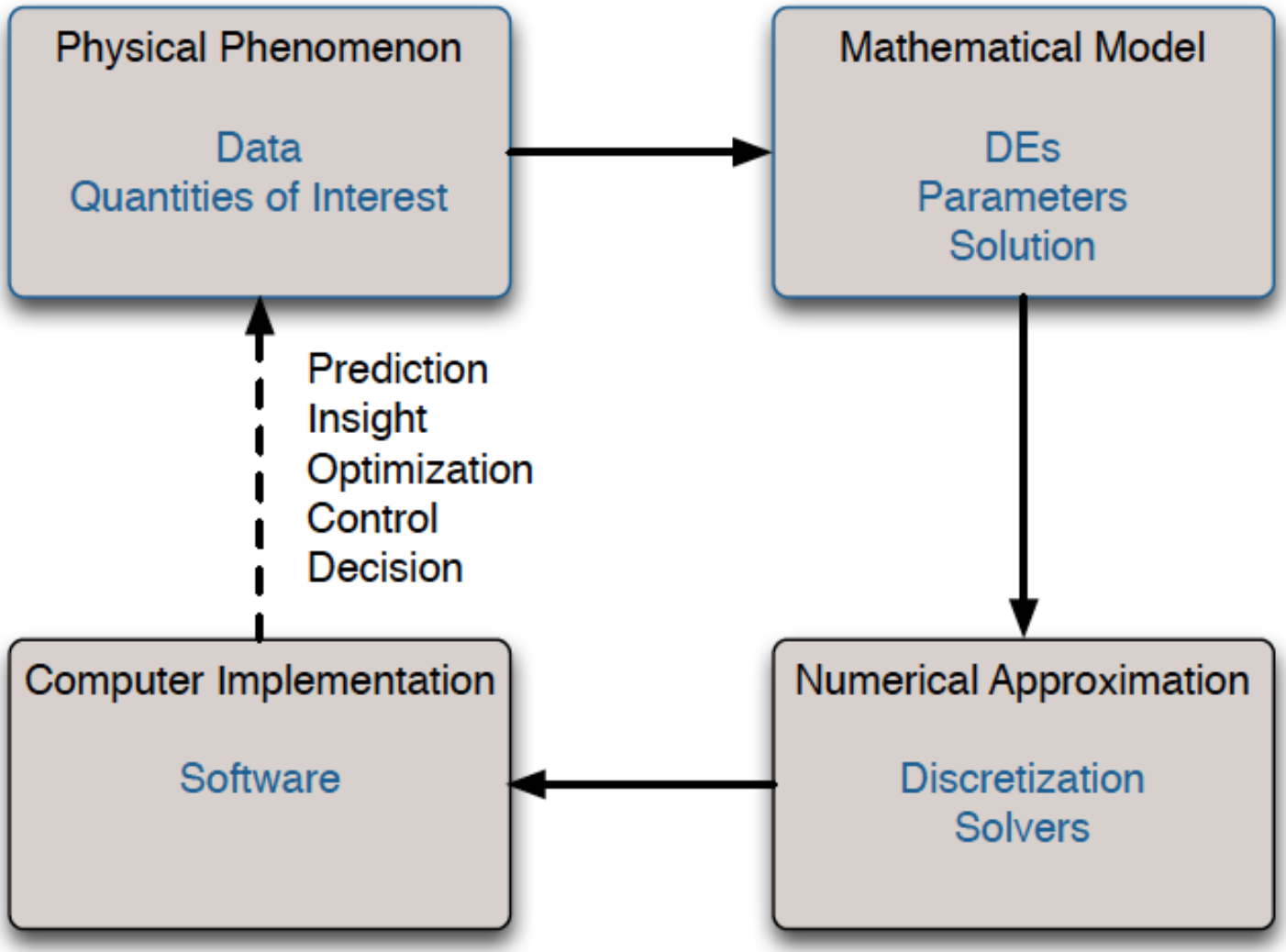
[Figures: Petra, Ghattas, Isaac, Martin, Stadler, et al.]

Intro: UQ and the scientific computing paradigm



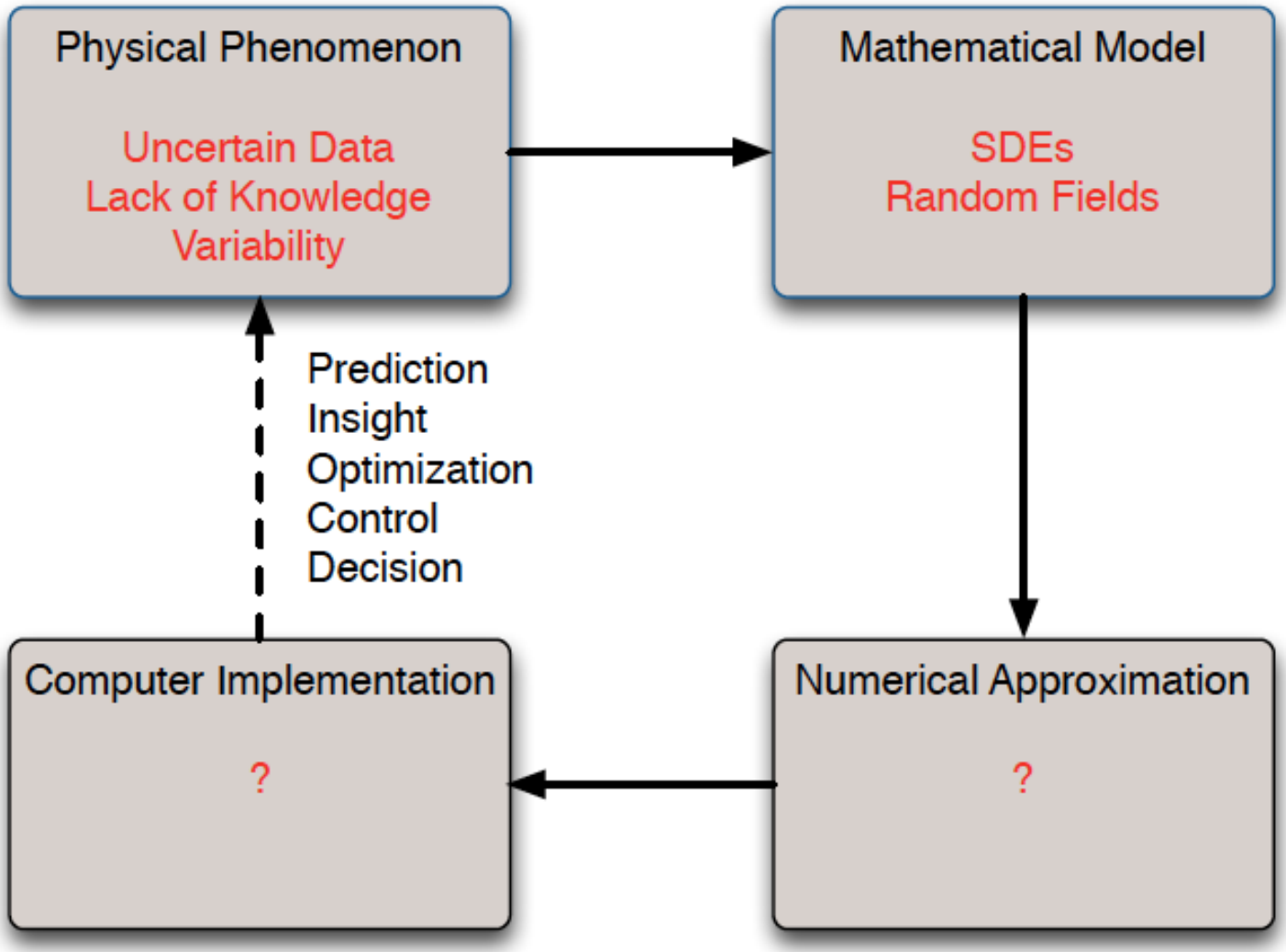
[Figure: Oliver Ernst]

Intro: UQ and the scientific computing paradigm



[Figure: Oliver Ernst]

Intro: UQ and the scientific computing paradigm



[Figure: Oliver Ernst]

Intro: Confidence in computer predictions

Validation: Are we solving the right problem?

Determine if a mathematical model adequately represents physical/engineering phenomena under study

Verification: Are we solving the problem correctly?

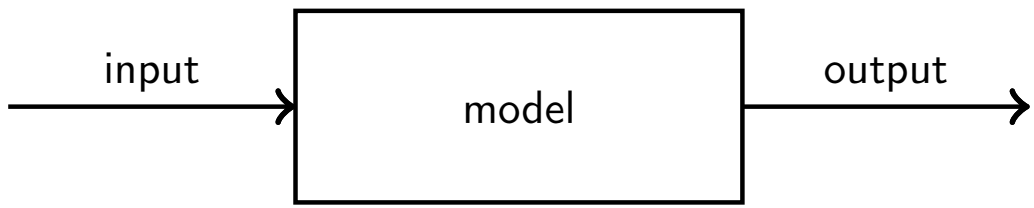
Determine if an algorithm and/or computer code correctly implements a given mathematical model

- Code verification (software engineering)
- Solution verification (a posterior error estimation)

Intro: Model

Model of system of interest

- Model describes response of system to inputs, parameters, configurations
- Response typically is a quantity of interest
- Evaluating a model means numerically simulating the model
- Many models given in form of partial differential equations



Mathematical formulation

$$f : \mathcal{D} \to \mathcal{Y}$$

- Input domain $\mathcal{D}$ and output domain $\mathcal{Y}$
- Maps $\boldsymbol{z} \in \mathcal{D}$ input onto $\boldsymbol{y} \in \mathcal{Y}$ output (quantity of interest)

Intro: Model - Navier-Stokes equations

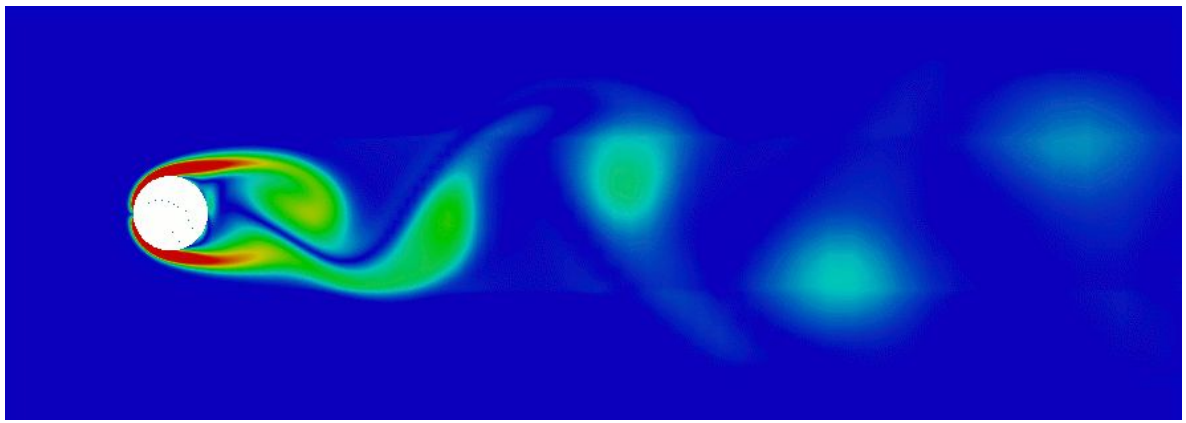
$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) = -\nabla p + \mu \Delta u + g$$

Examples of inputs

- Density ρ
- Dynamic viscosity μ

Examples of outputs (quantities of interest)

- Velocity at monitoring point
- Average pressure



[Figure: MFIX, NETL, DOE]

Intro: Model - Diffusion-convection-reaction flow

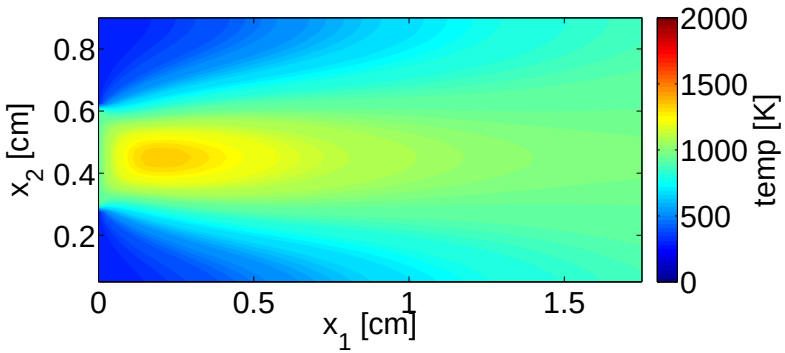
$$\frac{\partial u}{\partial t} = \Delta u - v\nabla u + g(u, \boldsymbol{\mu})$$

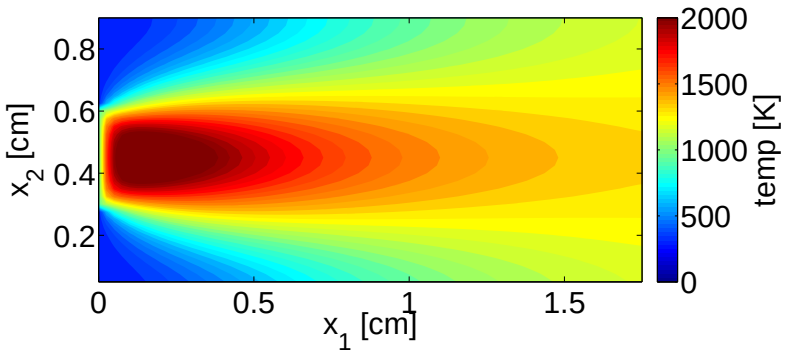
Examples of inputs

- Activation energy and pre-exponential factor (Arrhenius-type reaction)
- Temperature at boundary
- Ratio of fuel and oxidizer

Examples of outputs

- Average temperature in chamber





Intro: Uncertain inputs

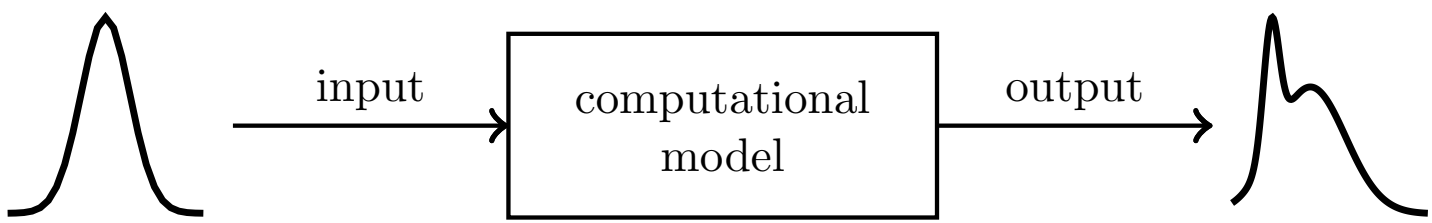
Inputs are uncertain

- Measurement errors in boundary conditions
- Manufacturing variations
- Model parameters determined by engineering judgment, etc.

Mathematically formulate uncertain inputs as random variables

$$Z : \Omega \to \mathcal{D}$$

Quantify effect of uncertainties in inputs on model outputs



Intro: General sampling-based approach to UQ

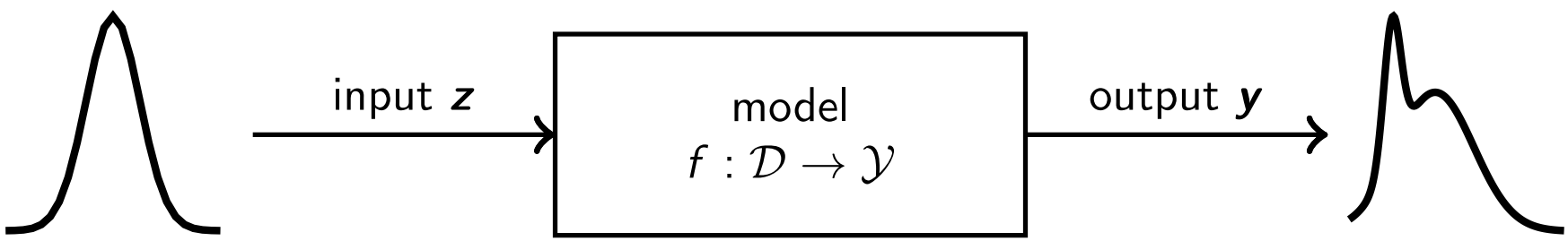
- Take many realizations of input random variable Z

$$\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n \in \mathcal{D}$$

- Evaluate model f at all $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ realizations

$$\boldsymbol{y}_1 = f(\boldsymbol{z}_1), \ldots, \boldsymbol{y}_n = f(\boldsymbol{z}_n)$$

- Estimate statistics from outputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$



Intro: General sampling-based approach to UQ

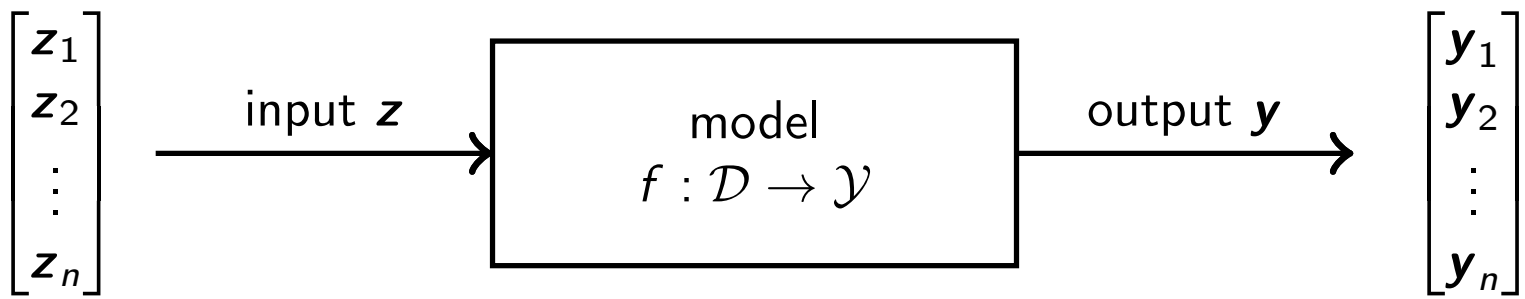
- Take many realizations of input random variable Z

$$\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n \in \mathcal{D}$$

- Evaluate model f at all $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ realizations

$$\boldsymbol{y}_1 = f(\boldsymbol{z}_1), \ldots, \boldsymbol{y}_n = f(\boldsymbol{z}_n)$$

- Estimate statistics from outputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$



Intro: General sampling-based approach to UQ

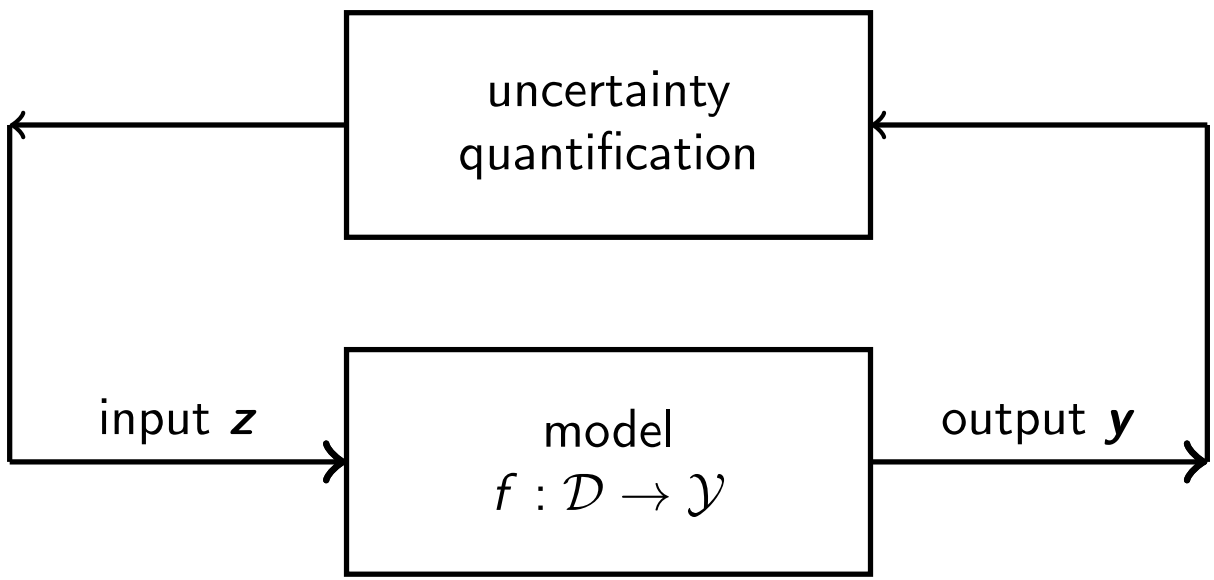
- Take many realizations of input random variable Z

$$\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n \in \mathcal{D}$$

- Evaluate model f at all $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ realizations

$$\boldsymbol{y}_1 = f(\boldsymbol{z}_1), \ldots, \boldsymbol{y}_n = f(\boldsymbol{z}_n)$$

- Estimate statistics from outputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$



Monte Carlo

[Figure: Martinp1, Wikipedia]

Monte Carlo

- Models treated as black box

[Figure: Martinp1, Wikipedia]

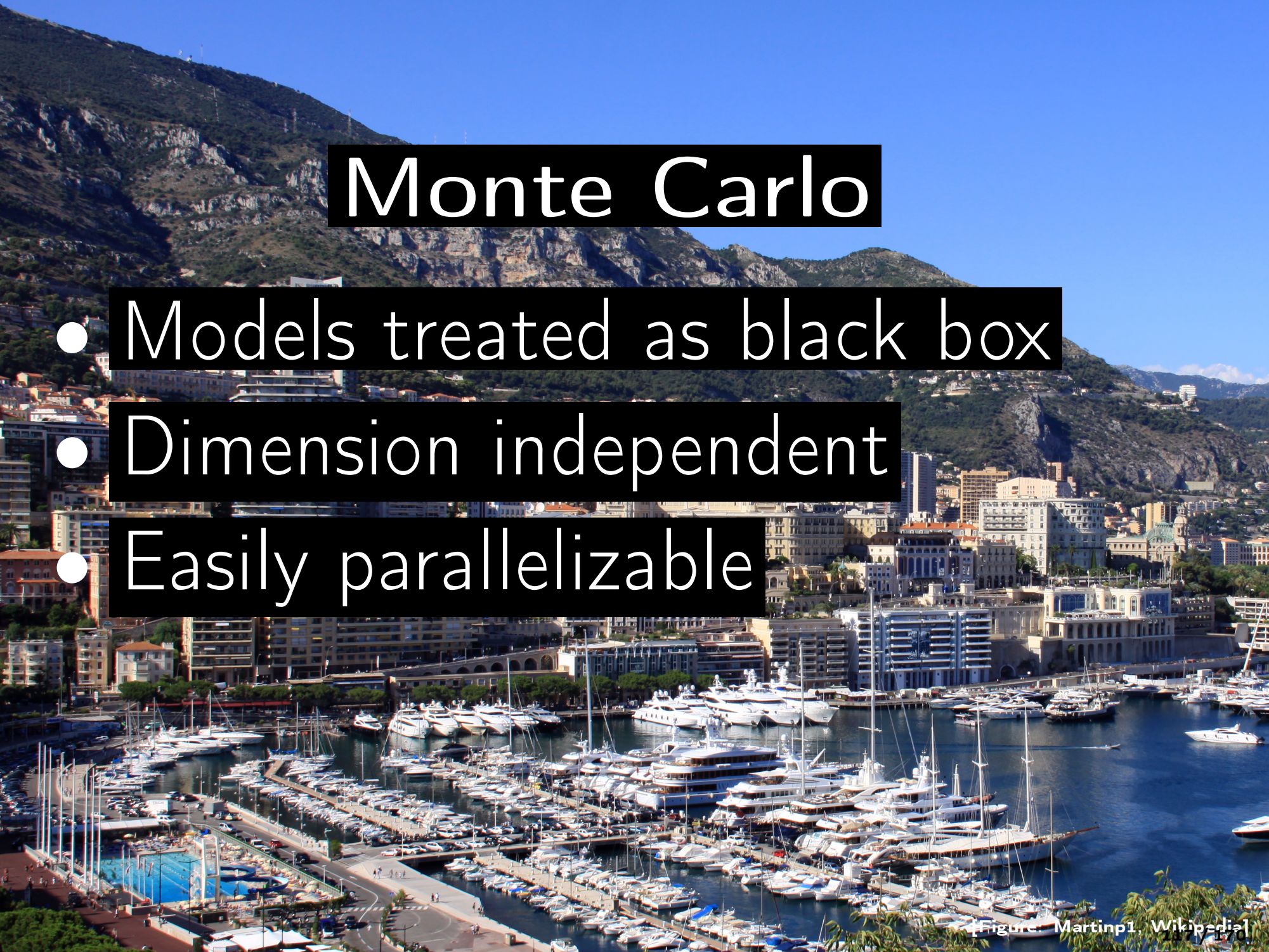
Monte Carlo

- Models treated as black box
- Dimension independent

[Figure: Martinp1, Wikipedia]

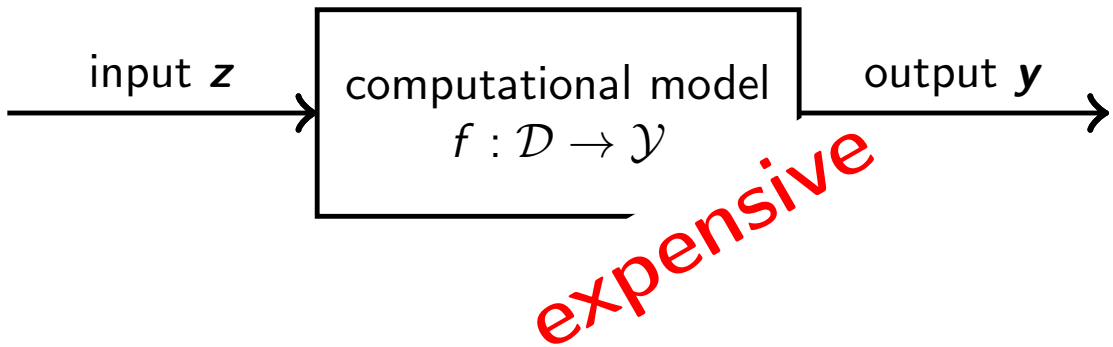
Monte Carlo

- Models treated as black box
- Dimension independent
- Easily parallelizable



[Figure: Martinp1, Wikipedia]

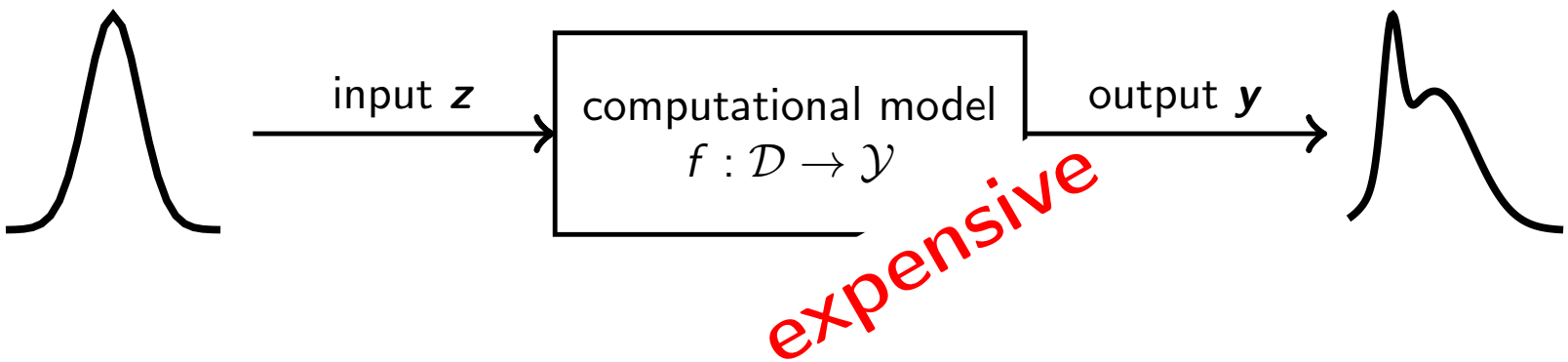
Intro: Challenges of sampling-based UQ



Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension

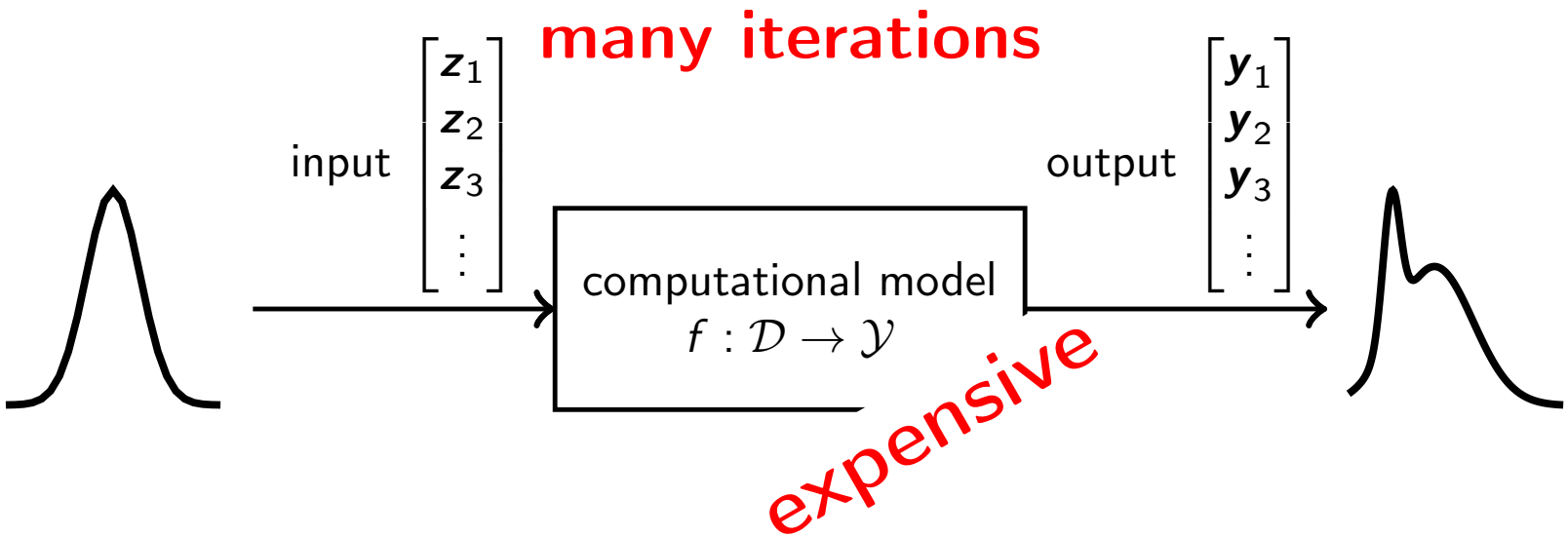
Intro: Challenges of sampling-based UQ



Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension

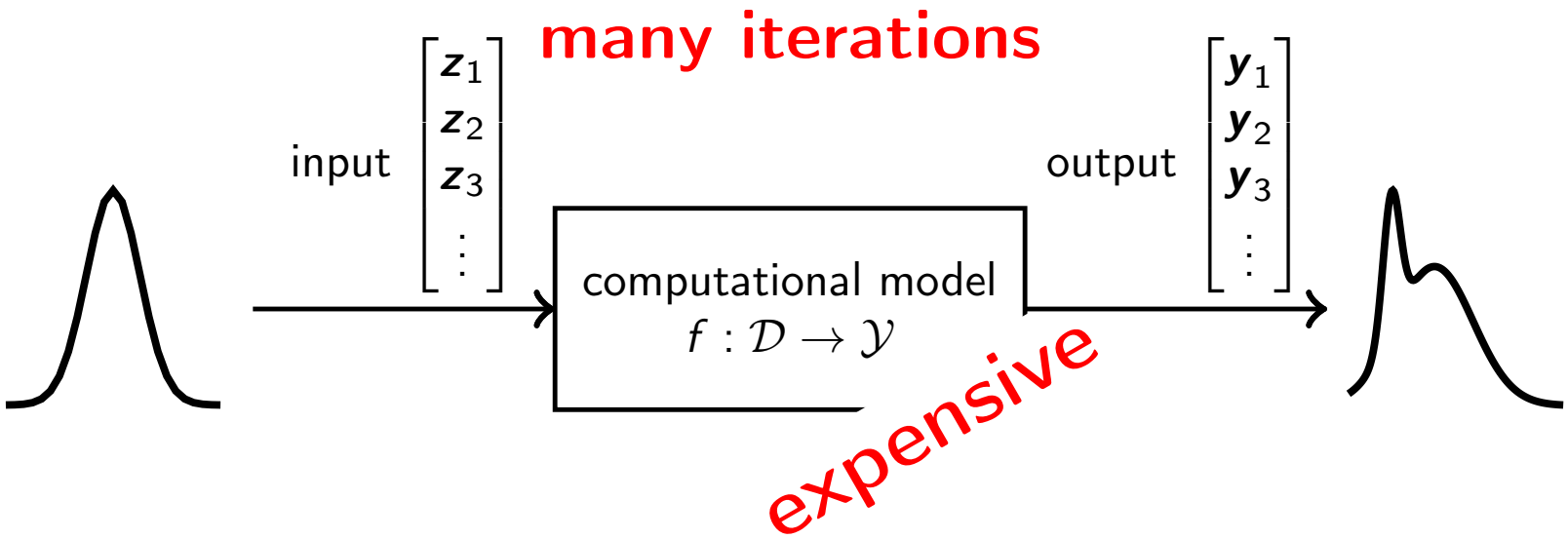
Intro: Challenges of sampling-based UQ



Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension

Intro: Challenges of sampling-based UQ



Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension

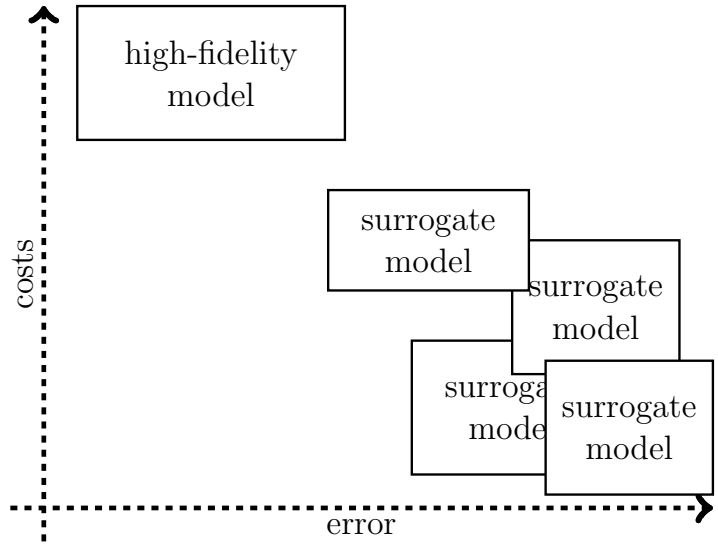
Intro: Opportunity of low-fidelity models

Given is typically a high-fidelity model

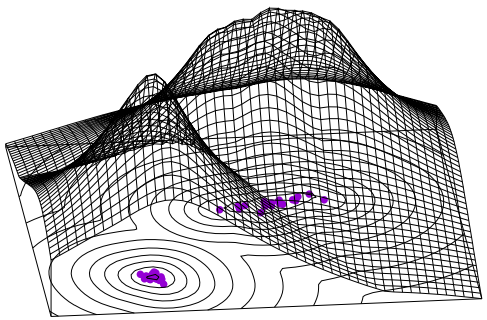
- Large-scale numerical simulation
- Achieves required accuracy
- Computationally expensive

Additionally, often have low-fidelity models

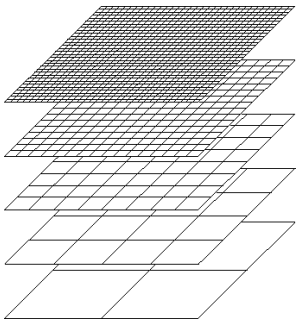
- Approximate same quantity of interest
- Often orders of magnitudes cheaper
- Less accurate



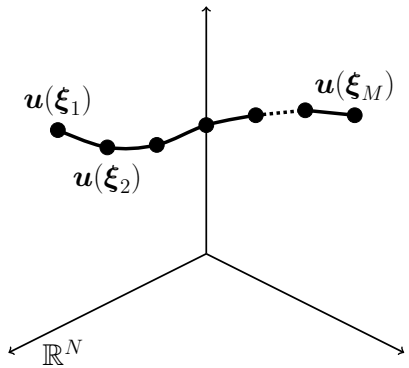
Examples of low-fidelity models



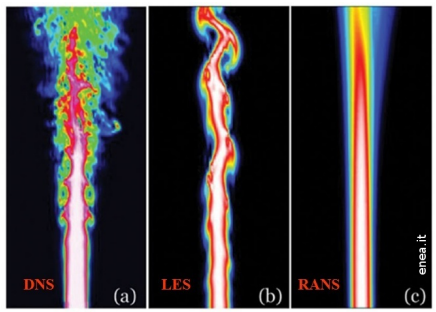
data-fit models, response surfaces, machine learning



coarse-grid approximations



reduced basis, proper orthogonal decomposition



simplified models, linearized models

Intro: Low-fidelity models

Replace high- with low-fidelity model

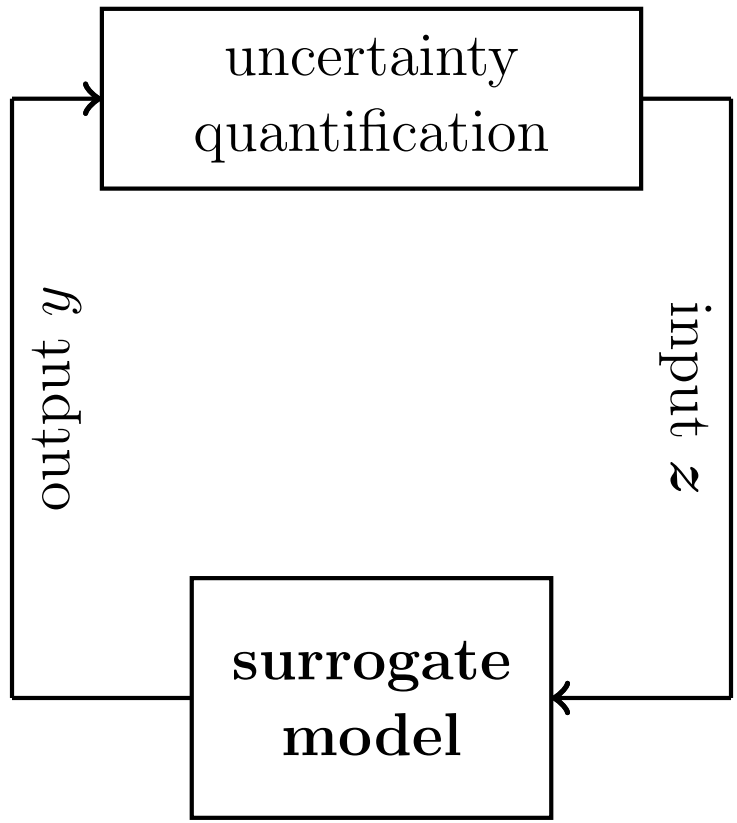
- Costs of outer loop application reduced
- Often orders of magnitude speedups

Low-fidelity model introduces error

- Control with error bounds/estimators*
- Rebuild if accuracy too low
- No guarantees without bounds/estimators

Issues

- Propagation of output error on estimate
- Applications without error control
- Costs of rebuilding a low-fidelity model



Multifidelity: Combine multiple models

Combine high-fidelity and low-fidelity models

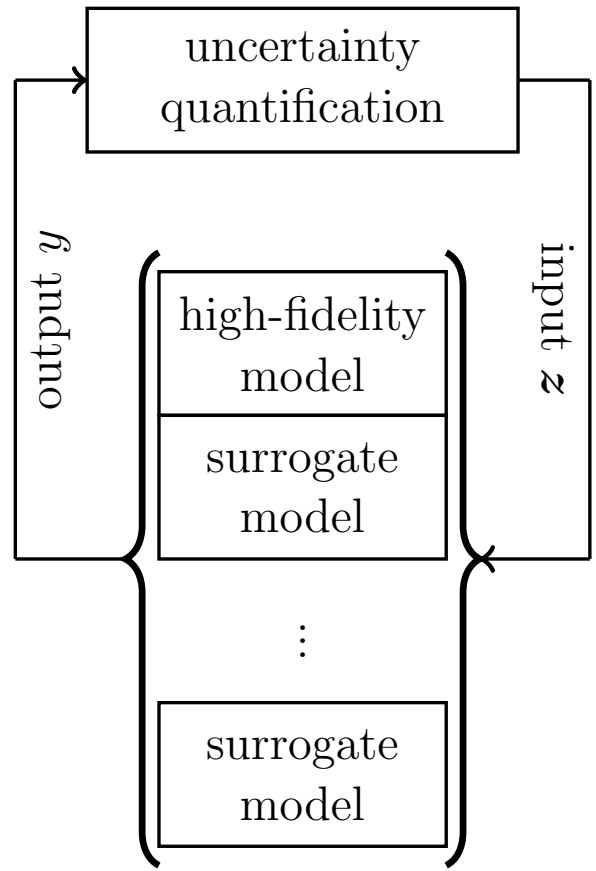
- Leverage low-fidelity models for speedup
- Recourse to high-fidelity for accuracy

Multifidelity speeds up computations

- Balance #solves among models
- Adapt, fuse, filter with low-/high-fidelity models

Multifidelity guarantees accuracy of high-fidelity

- Occasional recourse to high-fidelity model
- High-fidelity model is kept in the loop
- Independent of error control for low-fidelity



[P., Willcox, Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Review, 60(3):550-591, 2018]

Intro: Survey with *many* references

SIAM Review
Vol. 60, No. 3, pp. 550–591

Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*

Benjamin Peherstorfer†
Karen Willcox‡
Max Gunzburger§

Abstract. In many situations across computational science and engineering, multiple computational models are available that describe a system of interest. These different models have varying evaluation costs and varying fidelities. Typically, a computationally expensive high-fidelity model describes the system with the accuracy required by the current application at hand, while lower-fidelity models are less accurate but computationally cheaper than the high-fidelity model. Outer-loop applications, such as optimization, inference, and uncertainty quantification, require multiple model evaluations at many different inputs, which often leads to computational demands that exceed available resources if only the high-fidelity model is used. This work surveys multifidelity methods that accelerate the solution of outer-loop applications by combining high-fidelity and low-fidelity model evaluations, where the low-fidelity evaluations arise from an explicit low-fidelity model (e.g., a simplified physics approximation, a reduced model, a data-fit surrogate) that approximates the same output quantity as the high-fidelity model. The overall premise of these multifidelity methods is that low-fidelity models are leveraged for speedup while the high-fidelity model is kept in the loop to establish accuracy and/or convergence guarantees. We categorize multifidelity methods according to three classes of strategies: adaptation, fusion, and filtering. The paper reviews multifidelity methods in the outer-loop contexts of uncertainty propagation, inference, and optimization.

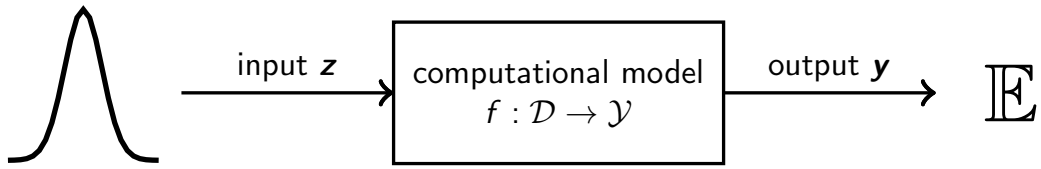
Key words. multifidelity, surrogate models, model reduction, multifidelity uncertainty quantification, multifidelity uncertainty propagation, multifidelity statistical inference, multifidelity optimization

AMS subject classifications. 65-02, 62-02, 49-02

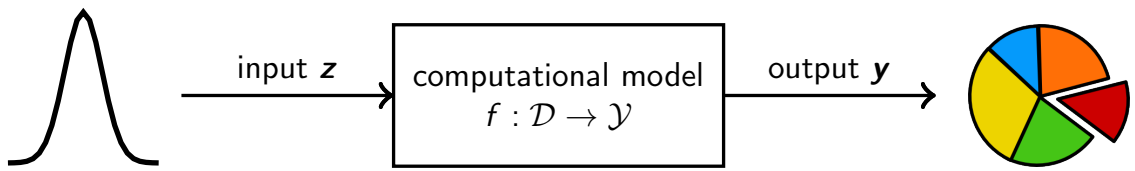
DOI. 10.1137/16M1082469

Uncertainty quantification tasks

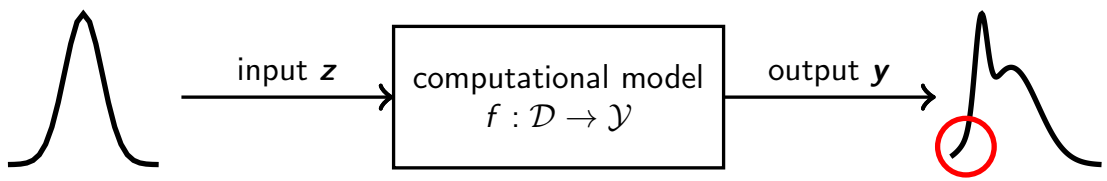
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



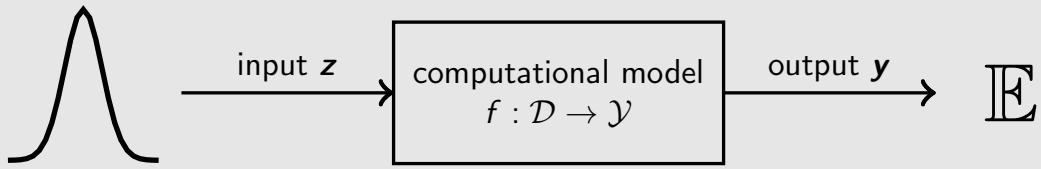
3. Multifidelity failure probability estimation



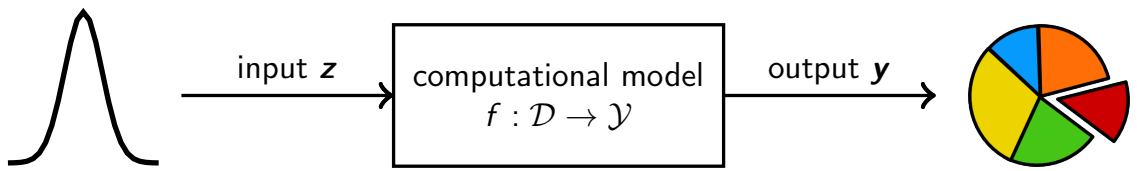
4. Other multifidelity uncertainty quantification tasks

Uncertainty quantification tasks

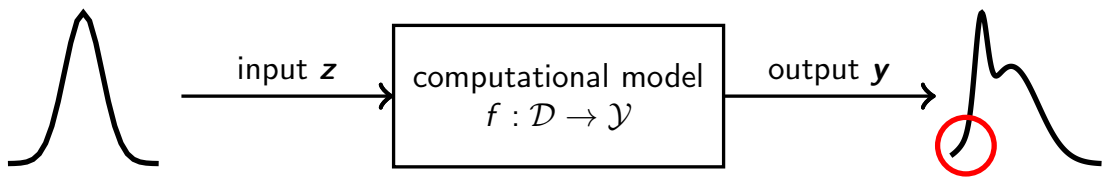
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



3. Multifidelity failure probability estimation



4. Other multifidelity uncertainty quantification tasks

MFMC: Monte Carlo estimation

High-fidelity ("truth") model, costs $w_1 > 0$

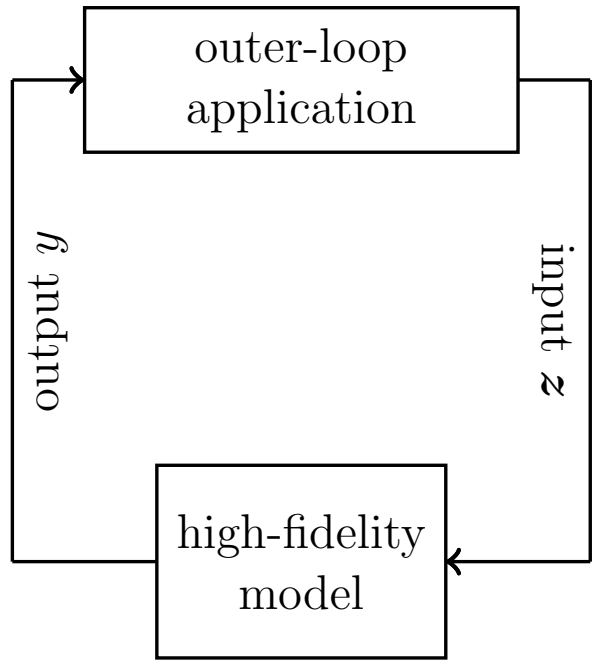
$$f^{(1)} : \mathcal{D} \to \mathcal{Y}$$

Random variable Z, estimate

$$s = \mathbb{E}[f^{(1)}(Z)]$$

Monte Carlo estimate of s with real. $z_1, \ldots, z_n$

$$\overline{y}_n^{(1)} = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i)$$



Computational costs

- Many evaluations of high-fidelity model
- Typically $10^3 - 10^6$ evaluations
- Intractable if $f^{(1)}$ expensive

MFMC: Control variates

Given is a random variable A with *unknown* statistics

$$s_A = \mathbb{E}[A]$$

MFMC: Control variates

Given is a random variable A with *unknown* statistics

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$

MFMC: Control variates

Given is a random variable A with *unknown* statistics

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$

Regular Monte Carlo estimator of s_A

$$\bar{a}_n = \frac{1}{n} \sum_{i=1}^{n} a_i$$

MFMC: Control variates

Given is a random variable A with *unknown* statistics

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$

Regular Monte Carlo estimator of s_A

$$\bar{a}_n = \frac{1}{n} \sum_{i=1}^{n} a_i$$

Unbiased estimator $\mathbb{E}[\bar{a}_n] = s_A$ with mean-squared error (MSE)

$$\mathrm{Var}[\bar{a}_n] = \frac{1}{n^2} \mathrm{Var}\left[\sum_{i=1}^{n} a_i\right] = \frac{\mathrm{Var}[A]}{n}$$

MFMC: Control variates (cont'd)

Additional random variable B with $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

MFMC: Control variates (cont'd)

Additional random variable B with $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

Regular Monte Carlo estimator of s_B

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$

MFMC: Control variates (cont'd)

Additional random variable B with $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

Regular Monte Carlo estimator of s_B

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$

Control variate estimator of s_A that uses samples from A and B

$$\hat{s}_A = \bar{a}_n + \left(s_B - \bar{b}_n\right)$$

MFMC: Control variates (cont'd)

Additional random variable B with $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

Regular Monte Carlo estimator of s_B

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$

Control variate estimator of s_A that uses samples from A and B

$$\hat{s}_A = \bar{a}_n + \left(s_B - \bar{b}_n\right)$$

Introduce coefficient $\alpha \in \mathbb{R}$ to balance A and B

$$\hat{s}_A = \bar{a}_n + \alpha \left(s_B - \bar{b}_n\right)$$

Combines n samples of A and n samples of B

[Nelson, 87]

MFMC: Control variates (cont'd)

Control variate estimator

$$\hat{s}_A = \bar{a}_n + \alpha \left(s_B - \bar{b}_n \right)$$

MFMC: Control variates (cont'd)

Control variate estimator

$$\hat{s}_A = \bar{a}_n + \alpha \left(s_B - \bar{b}_n \right)$$

Unbiased estimator of s_A because

$$\mathbb{E}[\hat{s}_A] = \underbrace{\mathbb{E}[\bar{a}_n]}_{=s_A} + \alpha \underbrace{\mathbb{E}[s_B - \bar{b}_n]}_{=0} = s_A$$

MFMC: Control variates (cont'd)

Control variate estimator

$$\hat{s}_A = \bar{a}_n + \alpha \left(s_B - \bar{b}_n \right)$$

Unbiased estimator of s_A because

$$\mathbb{E}[\hat{s}_A] = \underbrace{\mathbb{E}[\bar{a}_n]}_{=s_A} + \alpha \underbrace{\mathbb{E}[s_B - \bar{b}_n]}_{=0} = s_A$$

Variance of control variate estimator for optimal* $\alpha \in \mathbb{R}$

$$\mathrm{Var}[\hat{s}_A] = (1 - \rho^2) \frac{\mathrm{Var}[A]}{n} = (1 - \rho^2) \, \mathrm{Var}[\bar{a}_n]$$

- Correlation coefficient $-1 \leq \rho \leq 1$ of A and B
- If $\rho = 0$, same variance as regular Monte Carlo
- If $|\rho| > 0$, lower variance
- The higher correlated, the lower variance of $\hat{s}_A$

MFMC: Multifidelity Monte Carlo Estimation

Estimate expected value

$$s = \mathbb{E}[f^{(1)}(Z)]$$

Low-fidelity models

$$f^{(2)}, \ldots, f^{(k)} : \mathcal{D} \to \mathcal{Y}$$

Correlation coefficients

$$\rho_2 = \mathrm{Corr}[f^{(1)}, f^{(2)}], \rho_3 = \mathrm{Corr}[f^{(1)}, f^{(3)}], \ldots, \rho_k = \mathrm{Corr}[f^{(1)}, f^{(k)}]$$

Costs

$$w_1, \ldots, w_k > 0$$

MFMC: Multifidelity Monte Carlo

Reminder: Monte Carlo estimator

$$\overline{y}_n^{(1)} = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i)$$

Multifidelity Monte Carlo (MFMC) estimator

$$\hat{s} = \underbrace{\overline{y}_{m_1}^{(1)}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \underbrace{\left(\overline{y}_{m_i}^{(i)} - \overline{y}_{m_{i-1}}^{(i)} \right)}_{\text{from low-fid. models}}$$

- Monte Carlo estimator

$$\bar{y}_{m_i}^{(i)} = \frac{1}{m_i} \sum_{i=1}^{m_i} f^{(i)}(\boldsymbol{z}_i)$$

- Number of model evaluations $\boldsymbol{m} = [m_1, \ldots, m_k]^T$
- Control variate coefficients $\boldsymbol{\alpha} = [\alpha_2, \ldots, \alpha_k]^T$
- Optimal selection of $\boldsymbol{m}$ and $\boldsymbol{\alpha} \rightarrow$ our code

MFMC: Multifidelity Monte Carlo

Reminder: Monte Carlo estimator

$$\overline{y}_n^{(1)} = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i)$$

Multifidelity Monte Carlo (MFMC) estimator

$$\hat{s} = \underbrace{\overline{y}_{m_1}^{(1)}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \underbrace{\left(\overline{y}_{m_i}^{(i)} - \overline{y}_{m_{i-1}}^{(i)} \right)}_{\text{from low-fid. models}}$$

- Monte Carlo estimator

$$\overline{y}_{m_i}^{(i)} = \frac{1}{m_i} \sum_{i=1}^{m_i} f^{(i)}(\boldsymbol{z}_i)$$

- Number of model evaluations $\boldsymbol{m} = [m_1, \ldots, m_k]^T$
- Control variate coefficients $\boldsymbol{\alpha} = [\alpha_2, \ldots, \alpha_k]^T$
- Optimal selection of $\boldsymbol{m}$ and $\boldsymbol{\alpha} \rightarrow$ our code

MFMC: Recipe 1

Download

`https://github.com/pehersto/mfmc`

Given

- Models $f^{(1)}, \ldots, f^{(k)}$
- Computational budget b

Pilot run

- Draw m_0 (≈ 50) realizations of Z
- Evaluate each model $f^{(1)}, \ldots, f^{(k)}$ at the m_0 realizations

$$\boldsymbol{Y} = \begin{bmatrix} f^{(1)}(\boldsymbol{z}_1) & f^{(2)}(\boldsymbol{z}_1) & \ldots & f^{(k)}(\boldsymbol{z}_1) \\ \vdots & \vdots & & \vdots \\ f^{(1)}(\boldsymbol{z}_{m_0}) & f^{(2)}(\boldsymbol{z}_{m_0}) & \ldots & f^{(k)}(\boldsymbol{z}_{m_0}) \end{bmatrix}$$

- Estimate computational costs of model evaluations $\boldsymbol{w} = [w_1, \ldots, w_k]^T$

MFMC: Recipe 1 (cont'd)

Determine number of model evaluations

```
[ m, a ] = optiMlevelCorr( Y, w, b )
```

- Number of model evaluations $\boldsymbol{m} = [m_1, \ldots, m_k]^T$
- Coefficients $\boldsymbol{a} = [\alpha_2, \ldots, \alpha_k]^T$

Draw realizations

$$\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{m_k}$$

Evaluate models

$$f^{(i)}(\boldsymbol{z}_1), \ldots, f^{(i)}(\boldsymbol{z}_{m_i}), \qquad i = 1, \ldots, k$$

Estimate

$$\hat{s} = \underbrace{\overline{y}^{(1)}_{m_1}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \underbrace{\left(\overline{y}^{(i)}_{m_i} - \overline{y}^{(i)}_{m_{i-1}}\right)}_{\text{from low.-fid. models}}$$

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

```

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
  Y(:, i) = modelList{i}(mu);
end
```

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
  Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optiMlevelCorr(Y, w, budget); % MFMC
```

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
  Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optiMlevelCorr(Y, w, budget); % MFMC

z = drawSamples(m(end)); % draw realizations
```

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
  Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optiMlevelCorr(Y, w, budget); % MFMC

z = drawSamples(m(end)); % draw realizations

y = modelList{1}(z(1:m(1), :)); % evaluate HFM
sHat = alpha(1)*mean(y);
```

MFMC: Matlab code for Recipe 1

```
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
  Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optiMlevelCorr(Y, w, budget); % MFMC

z = drawSamples(m(end)); % draw realizations

y = modelList{1}(z(1:m(1), :)); % evaluate HFM
sHat = alpha(1)*mean(y);

% evaluate low-fidelity models
for i=2:length(modelList)
  y = modelList{i}(z(1:m(i), :));
  sHat = sHat+alpha(i)*(mean(y)-mean(y(1:m(i-1))));
end
```

MFMC: Recipe 2 (MFMC as post-processing)

Given

- Model evaluations

$$f^{(i)}(\boldsymbol{z}_1), \ldots, f^{(i)}(\boldsymbol{z}_{m_i}), \qquad i = 1, \ldots, k$$

- Model evaluation costs $w_1, \ldots, w_k$

Pilot samples

- Use the first $m_0 \ll m_1$ samples to form

$$\boldsymbol{Y} = \begin{bmatrix} f^{(1)}(\boldsymbol{z}_1) & f^{(2)}(\boldsymbol{z}_1) & \ldots & f^{(k)}(\boldsymbol{z}_1) \\ \vdots & \vdots & & \vdots \\ f^{(1)}(\boldsymbol{z}_{m_0}) & f^{(2)}(\boldsymbol{z}_{m_0}) & \ldots & f^{(k)}(\boldsymbol{z}_{m_0}) \end{bmatrix}$$

- Derive coefficients

```
[ ∼, a ] = optiMlevelCorr( Y, w, b )
```

Estimate

$$s = \underbrace{\overline{y}^{(1)}_{m_1}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \underbrace{\left(\overline{y}^{(i)}_{m_i} - \overline{y}^{(i)}_{m_{i-1}} \right)}_{\text{from low.-fid. models}}$$

There are theoretic subtleties that are typically negligible in practice

MFMC: Recipe 2 (MFMC as post-processing)

Given

- Model evaluations

$$f^{(i)}(z_1), \ldots, f^{(i)}(z_{m_i}), \qquad i = 1, \ldots, k$$

- Model evaluation costs $w_1, \ldots, w_k$

Pilot samples

- Use the first $m_0 \ll m_1$ samples to form

$$\boldsymbol{Y} = \begin{bmatrix} f^{(1)}(z_1) & f^{(2)}(z_1) & \ldots & f^{(k)}(z_1) \\ \vdots & \vdots & & \vdots \\ f^{(1)}(z_{m_0}) & f^{(2)}(z_{m_0}) & \ldots & f^{(k)}(z_{m_0}) \end{bmatrix}$$

- Derive coefficients

```
[ ∼, a ] = optiMlevelCorr( Y, w, b )
```

Estimate

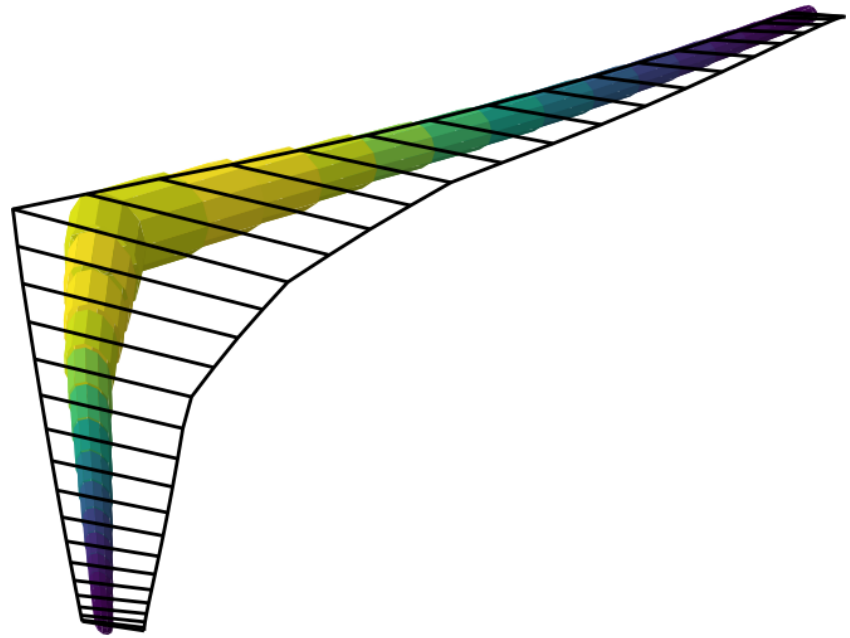
$$s = \underbrace{\overline{y}^{(1)}_{m_1}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \underbrace{\left(\overline{y}^{(i)}_{m_i} - \overline{y}^{(i)}_{m_{i-1}} \right)}_{\text{from low.-fid. models}}$$

There are theoretic subtleties that are typically negligible in practice

MFMC: AeroStruct: Problem setup

Coupled aero-structural wing analysis

- Uncertain are angle of attack, air density, Mach number
- Estimate expected fuel burn



High-fidelity model $f^{(1)}$

- OpenAeroStruct code
- Vortex-lattice method
- 6 DoF 3-dim spatial beam model
- Used with default configuration

Low-fidelity models

- Spline interpolants on equidistant grid
- Low-fidelity model $f^{(2)}$ from 343 points
- Low-fidelity model $f^{(3)}$ from 125 points

[Jasa, J. P., Hwang, J. T., and Martins, J. R. R. A., "Open-source coupled aerostructural optimization using Python," Structural and Multidisciplinary Optimization, 2018. (submitted)]

https://github.com/johnjasa/OpenAeroStruct/

MFMC: AeroStruct: Distribution of work

Model properties

model	evaluation costs [s]	offline costs [s]	correlation coefficient
high-fid. $f^{(1)}$	1.61×10^{-1}	-	-
low-fid. $f^{(2)}$	1.23×10^{-7}	55.382	9.9552×10^{-1}
low-fid. $f^{(3)}$	1.21×10^{-7}	20.183	9.9192×10^{-1}

Number of model evaluations

	Monte Carlo	MFMC with $f^{(1)}, f^{(2)}$		MFMC with $f^{(1)}, f^{(3)}$	
online costs [s]	#evals $f^{(1)}$	#evals $f^{(1)}$	#evals $f^{(2)}$	#evals $f^{(1)}$	#evals $f^{(3)}$
7.99×10^{0}	50	4.90×10^{1}	4.48×10^{5}	4.90×10^{1}	5.97×10^{5}
1.61×10^{1}	100	9.90×10^{1}	8.95×10^{5}	9.90×10^{1}	1.19×10^{6}
8.07×10^{1}	500	4.96×10^{2}	4.48×10^{6}	4.95×10^{2}	5.97×10^{6}
1.61×10^{2}	1000	9.93×10^{2}	8.95×10^{6}	9.90×10^{2}	1.19×10^{7}
8.07×10^{2}	5000	4.97×10^{3}	4.48×10^{7}	4.95×10^{3}	5.97×10^{7}

MFMC trades high-fidelity evaluations for low-fidelity evaluations

- The high-fidelity model evaluations guarantee unbiased
- The low-fidelity model evaluations help to reduce the variance
- The balance is optimal with respect to the mean-squared error

MFMC: AeroStruct: Distribution of work

Model properties

model	evaluation costs [s]	offline costs [s]	correlation coefficient
high-fid. $f^{(1)}$	1.61×10^{-1}	-	-
low-fid. $f^{(2)}$	1.23×10^{-7}	55.382	9.9552×10^{-1}
low-fid. $f^{(3)}$	1.21×10^{-7}	20.183	9.9192×10^{-1}

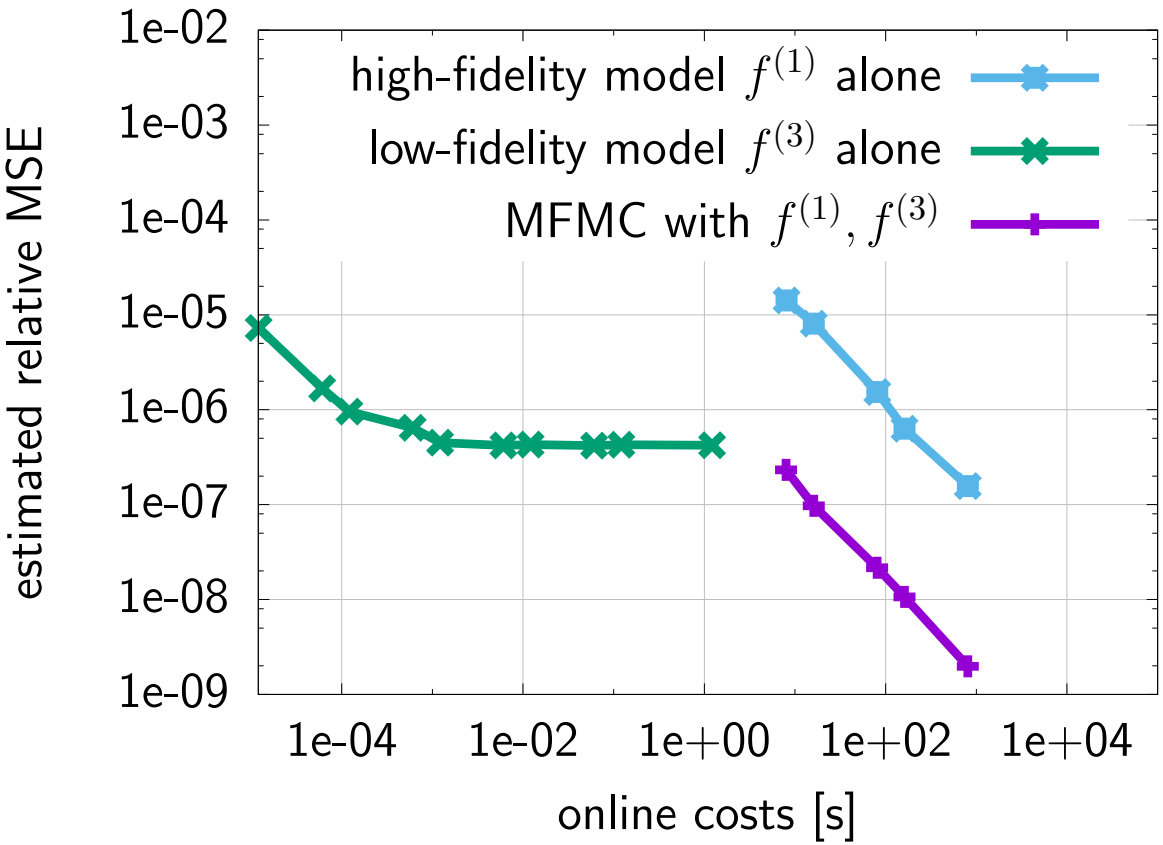
Number of model evaluations

	Monte Carlo	MFMC with $f^{(1)}, f^{(2)}$		MFMC with $f^{(1)}, f^{(3)}$	
online costs [s]	#evals $f^{(1)}$	#evals $f^{(1)}$	#evals $f^{(2)}$	#evals $f^{(1)}$	#evals $f^{(3)}$
7.99×10^{0}	50	4.90×10^{1}	4.48×10^{5}	4.90×10^{1}	5.97×10^{5}
1.61×10^{1}	100	9.90×10^{1}	8.95×10^{5}	9.90×10^{1}	1.19×10^{6}
8.07×10^{1}	500	4.96×10^{2}	4.48×10^{6}	4.95×10^{2}	5.97×10^{6}
1.61×10^{2}	1000	9.93×10^{2}	8.95×10^{6}	9.90×10^{2}	1.19×10^{7}
8.07×10^{2}	5000	4.97×10^{3}	4.48×10^{7}	4.95×10^{3}	5.97×10^{7}

MFMC trades high-fidelity evaluations for low-fidelity evaluations

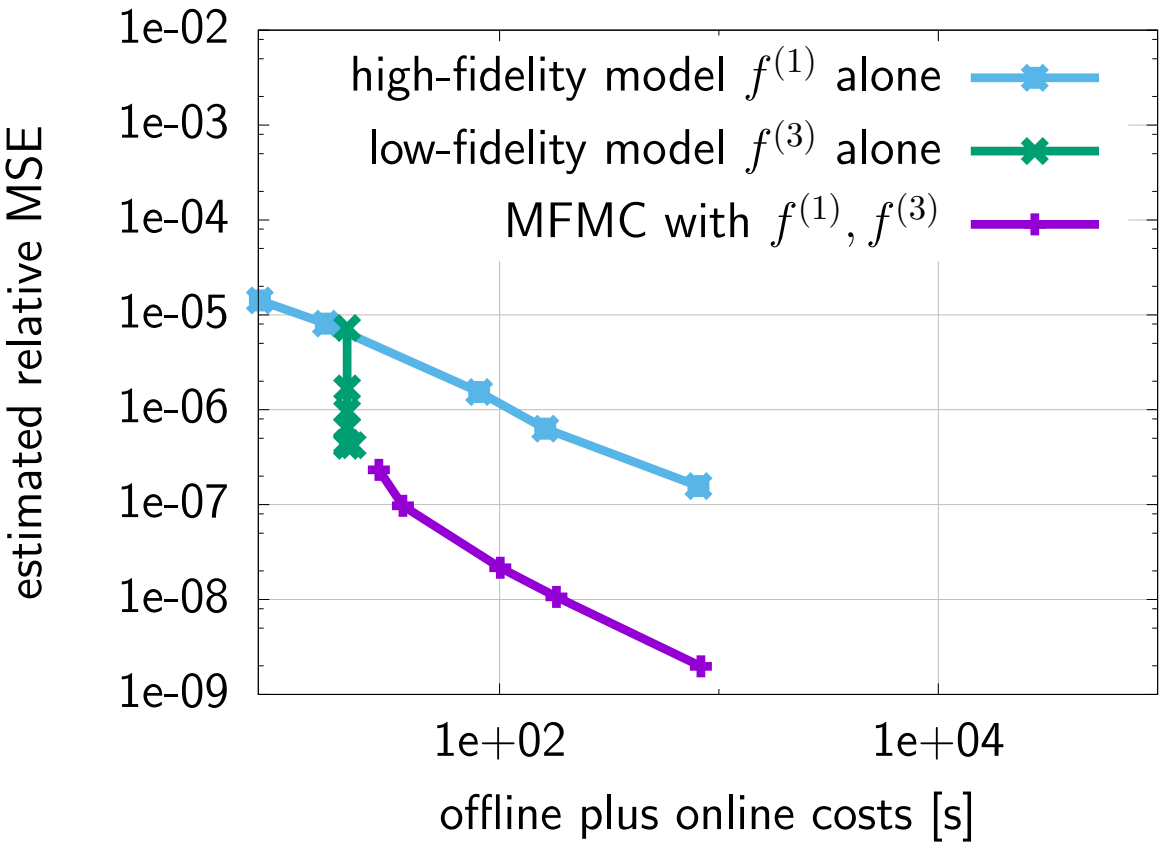
- The high-fidelity model evaluations guarantee unbiased
- The low-fidelity model evaluations help to reduce the variance
- The balance is optimal with respect to the mean-squared error

MFMC: AeroStruct: Speedup results



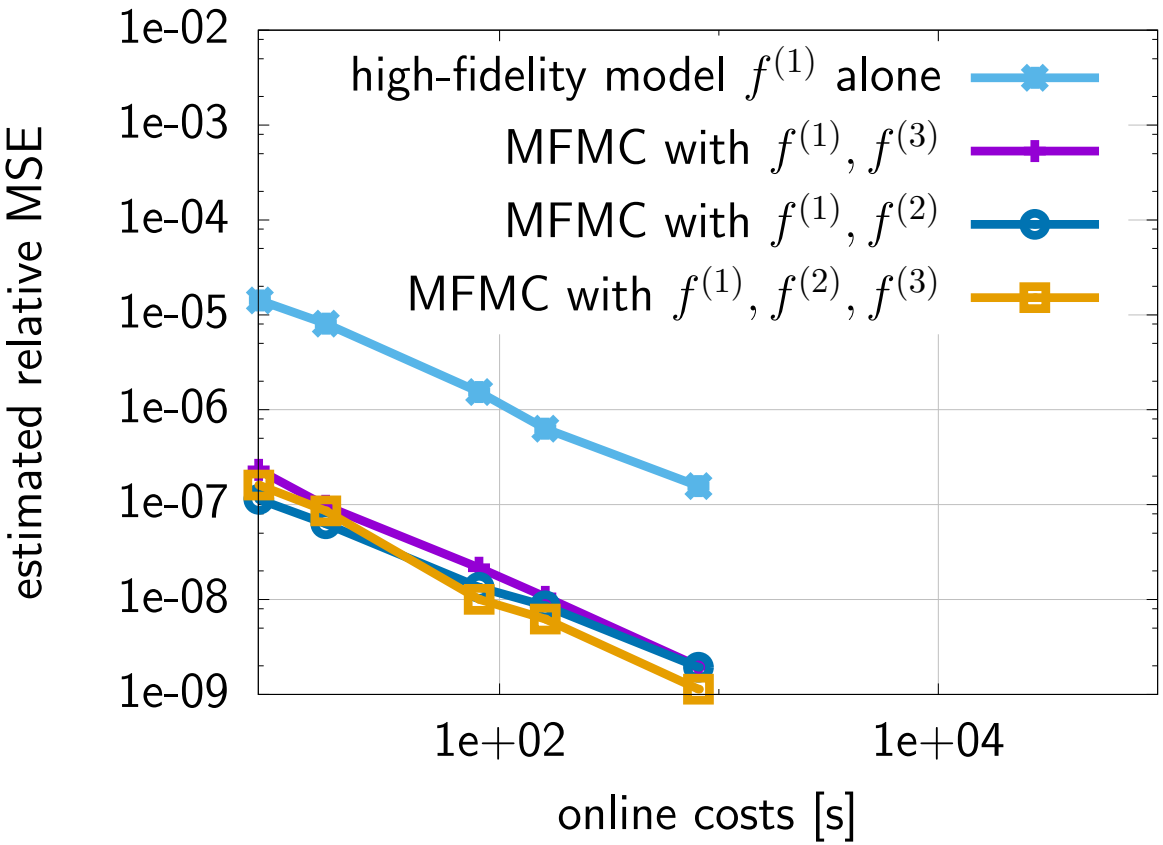
- Low-fidelity model alone leads to biased estimators
- MFMC achieves speedup of about one order of magnitude

MFMC: AeroStruct: Speedup with offline costs



- Constructing low-fidelity models incurs offline costs
- In this example, offline costs low compared to savings

MFMC: AeroStruct: Combining all three models



- Model $f^{(2)}$ and $f^{(3)}$ are similar with respect to costs/correlations
- Adding model $f^{(2)}$ as little effect

MFMC: Plate

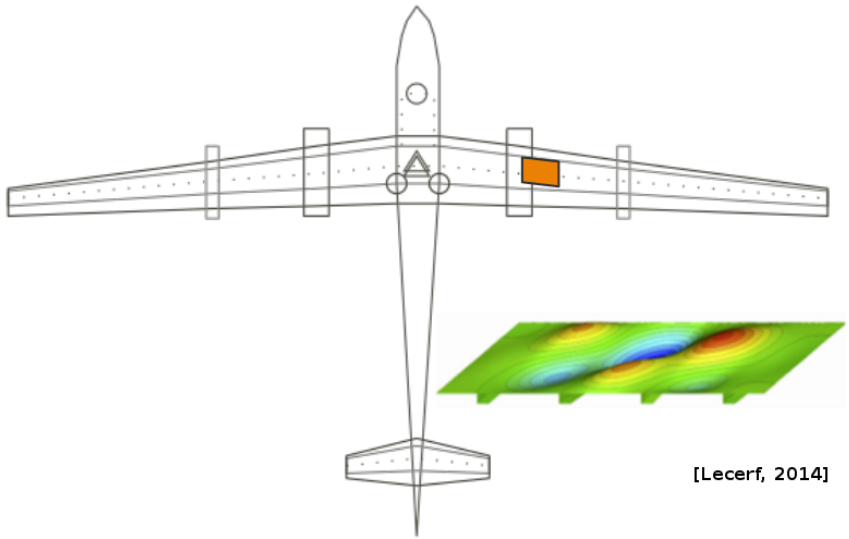
Locally damaged plate in bending

- Inputs: nominal thickness, load, damage
- Output: maximum deflection of plate
- **Only distribution of inputs known**
- Estimate **expected** deflection

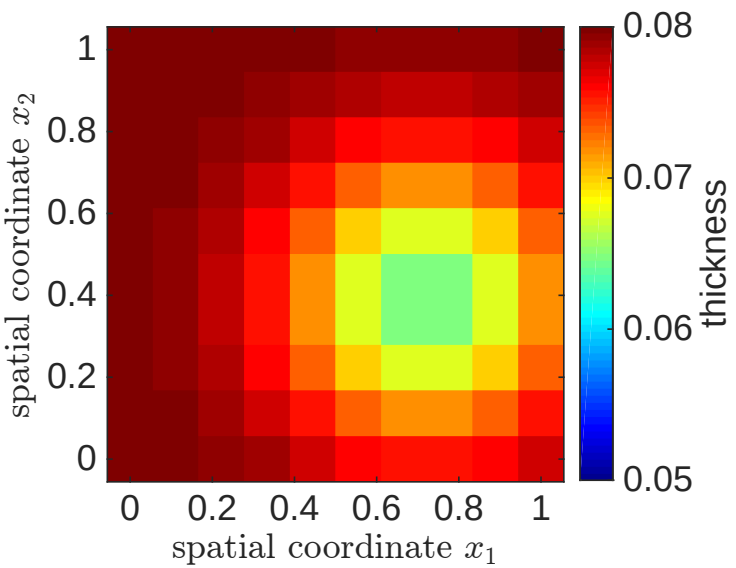
Six models

- High-fidelity model: FEM, 300 DoFs
- Reduced model: POD, 10 DoFs
- Reduced model: POD, 5 DoFs
- Reduced model: POD, 2 DoFs
- Data-fit model: linear interp., 256 pts
- Support vector machine: 256 pts

Var, corr, and costs est. from 100 samples

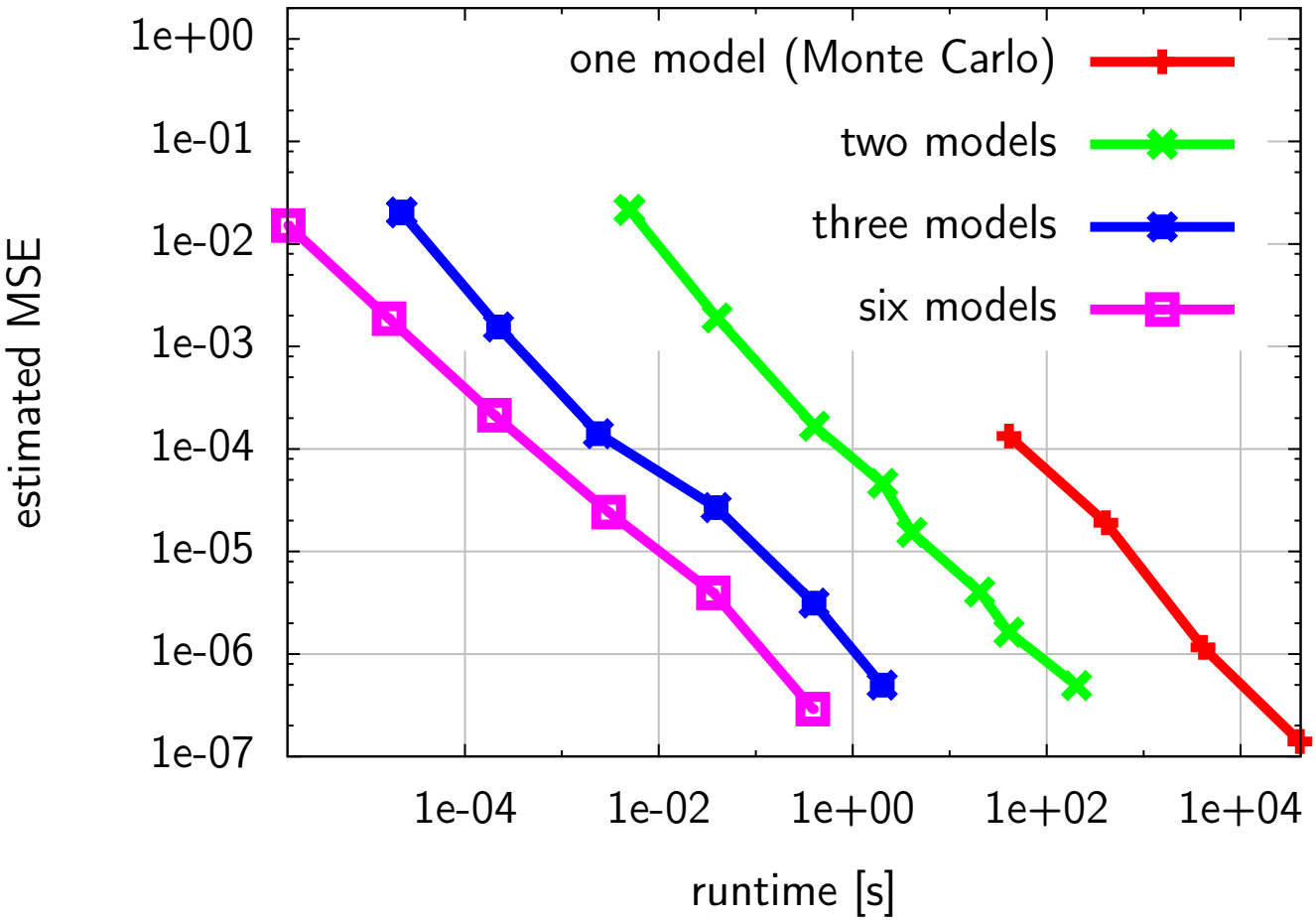


(a) wing panel



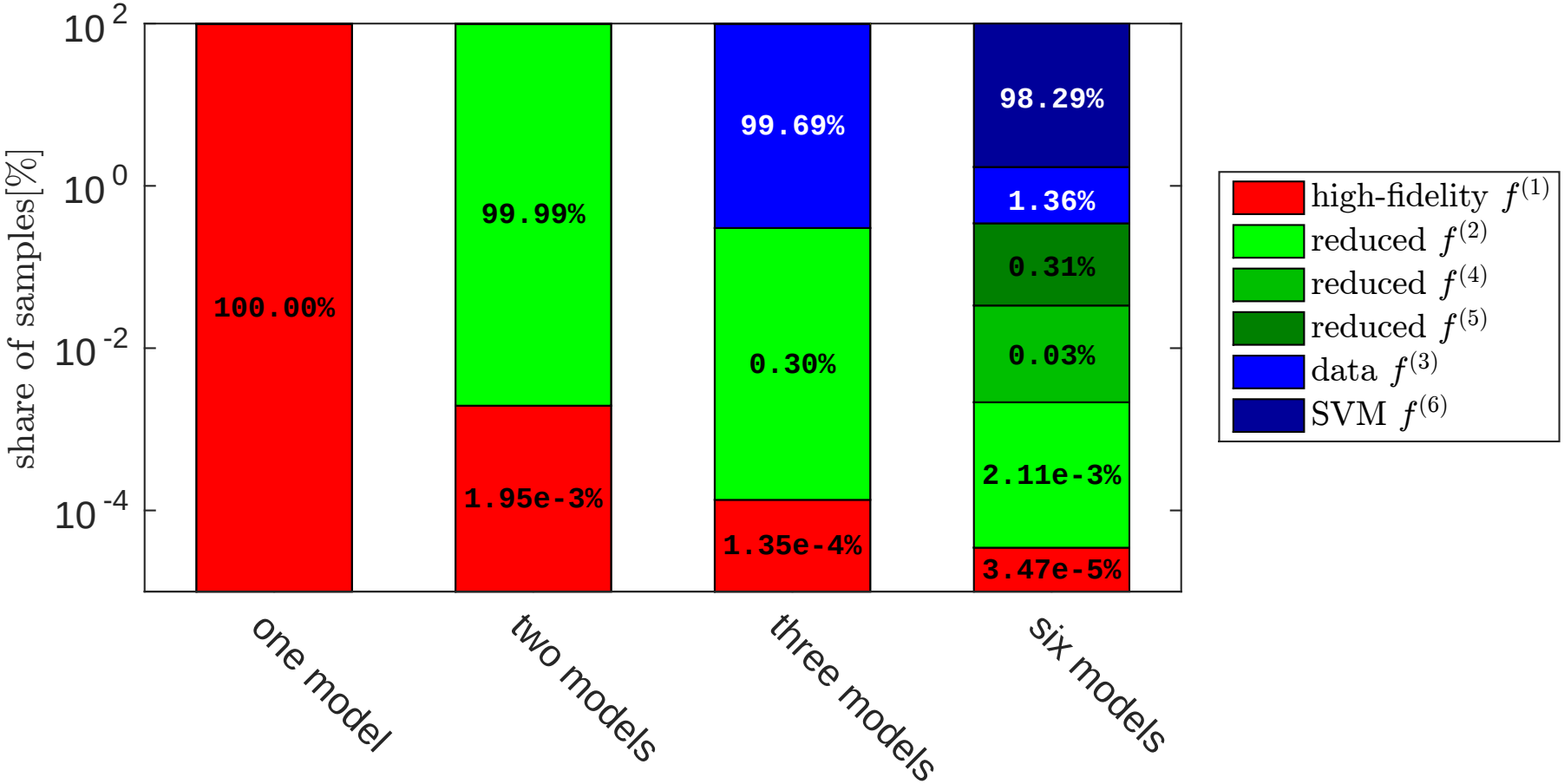
(b) damaged plate

MFMC: Plate: Combining many models



- Largest improvement from "single $\rightarrow$ two" and "two $\rightarrow$ three"
- Adding yet another reduced/SVM model reduces variance only slightly

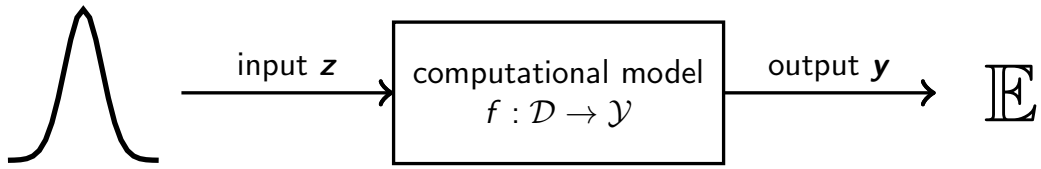
MFMC: Plate: #evals of models



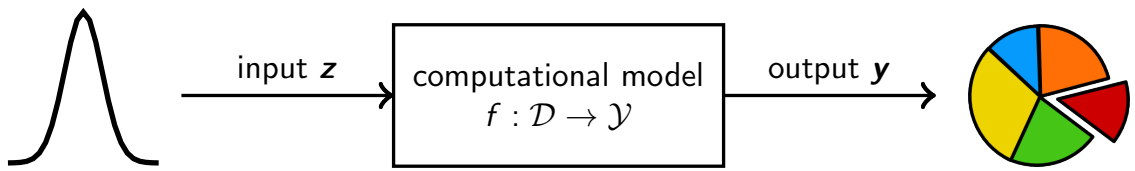
- MFMC distributes #evals among models depending on corr/costs
- Number of evaluation changes exponentially between models
- Highest #evals in data-fit models (cost ratio $w_1/w_6 \approx 10^6$)

Uncertainty quantification tasks

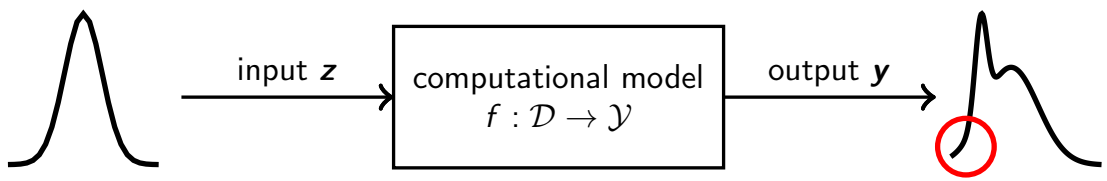
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



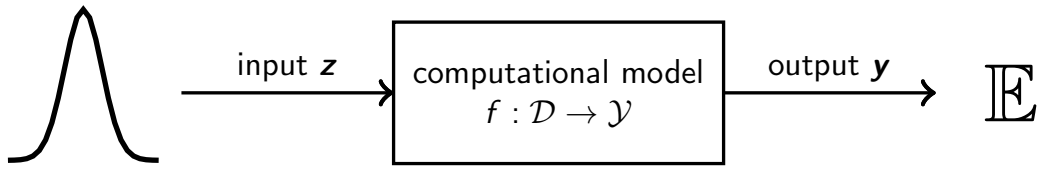
3. Multifidelity failure probability estimation



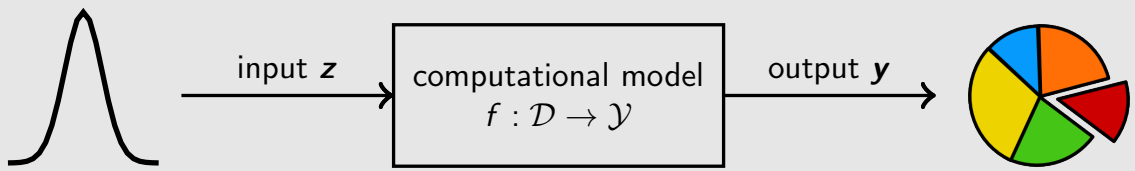
4. Other multifidelity uncertainty quantification tasks

Uncertainty quantification tasks

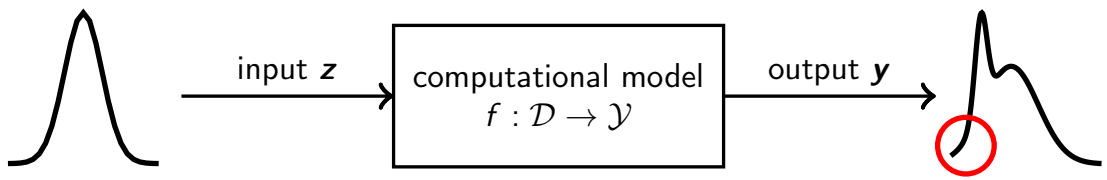
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis

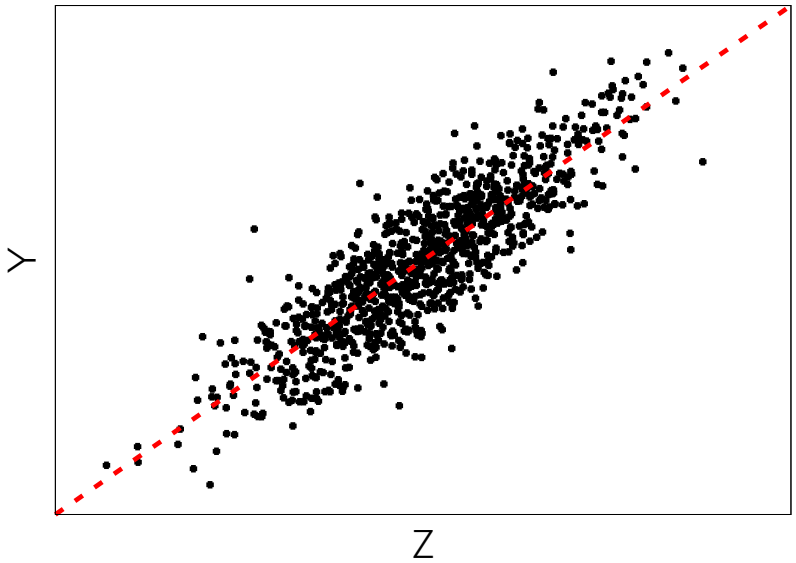


3. Multifidelity failure probability estimation

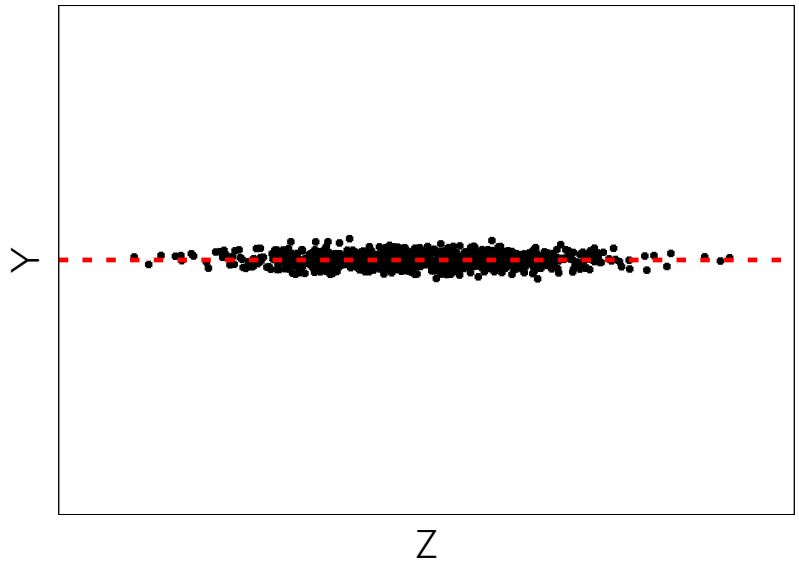


4. Other multifidelity uncertainty quantification tasks

MFGSA: Sensitivity analysis



Y is sensitive to Z



Y is *not* sensitive to Z

Sensitivity analysis

- Determine which inputs influence output most
- Can sample Y as a black box for inputs Z and need to determine what components of $Z = [Z_1, \ldots, Z_d]^T$ influence Y most

MFGSA: Sensitivity analysis in engineering

Risk communication for decision-making

- Determine if one can rely on model output or if "noise"
- Communicate to upstream decision-making which inputs are critical

Feedback to improve model

- Determine which inputs need to be sampled carefully
- Prioritize effort on reducing uncertainty
- Modify model with respect to sensitive inputs

Model reduction and dimensionality reduction

- Focus on important inputs and ignore ineffective inputs
- Derive surrogate models that depend on important inputs only

MFGSA: Variance-based global sensitivity analysis

- Input $Z = [Z_1, \ldots, Z_d]^T \in \mathcal{D}$ is a random *vector*
- Output of model $Y = f^{(1)}(Z_1, \ldots, Z_d)$ is sensitive to inputs
- Measure sensitivity with variance
- Main effect sensitivity

$$S_i = \frac{\mathrm{Var}[\mathbb{E}[Y|Z_i]]}{\mathrm{Var}[Y]}$$

- Main sensitivity indices are normalized

$$\sum_{i=1}^{d} S_i = 1\,, \qquad S_i \in [0, 1]$$

MFGSA: Multifidelity estimation

Estimation of sensitivity indices

- Estimate variance instead of expected value

$$S_i = \frac{\mathrm{Var}[\mathbb{E}[Y|Z_i]]}{\mathrm{Var}[Y]}$$

- Requires estimating variance for all d inputs $Z = [Z_1, \ldots, Z_d]$

Multifidelity estimation

- Given are low-fidelity models $f^{(2)}, \ldots, f^{(k)}$
- Similarly to MFMC, exploit correlations

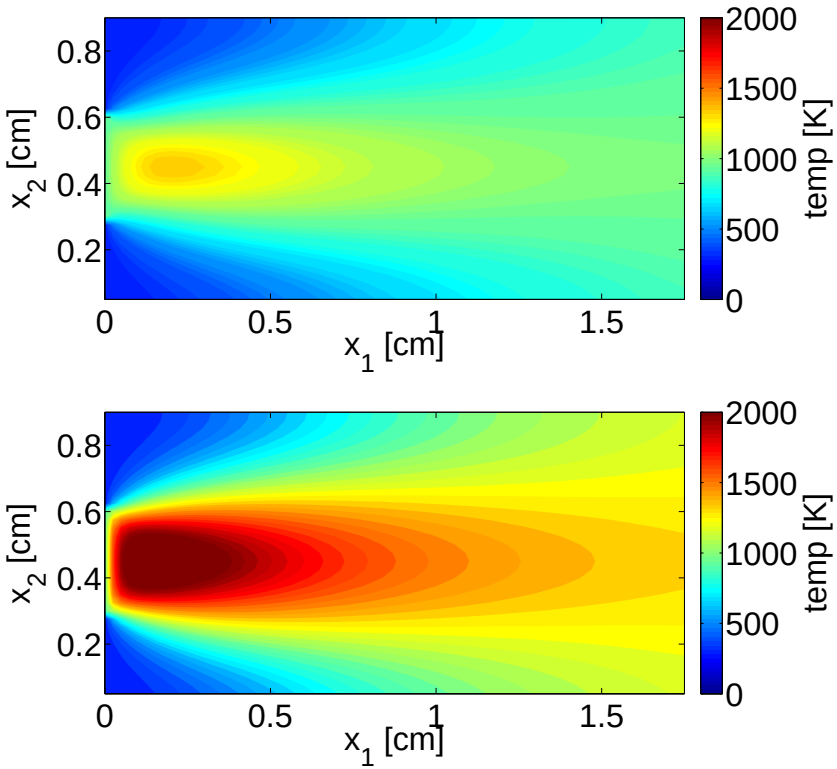
$$\rho_2 = \mathrm{Corr}[f^{(1)}, f^{(2)}], \rho_3 = \mathrm{Corr}[f^{(1)}, f^{(3)}], \ldots, \rho_k = \mathrm{Corr}[f^{(1)}, f^{(k)}]$$

- Estimator has similar structure as estimator for expected values

MFGSA: Premixed flame

Inputs to model are

- Parameters of Arrhenius reaction
- Temperatures at boundary
- Ratio of fuel and oxidizer
- Activation Energy

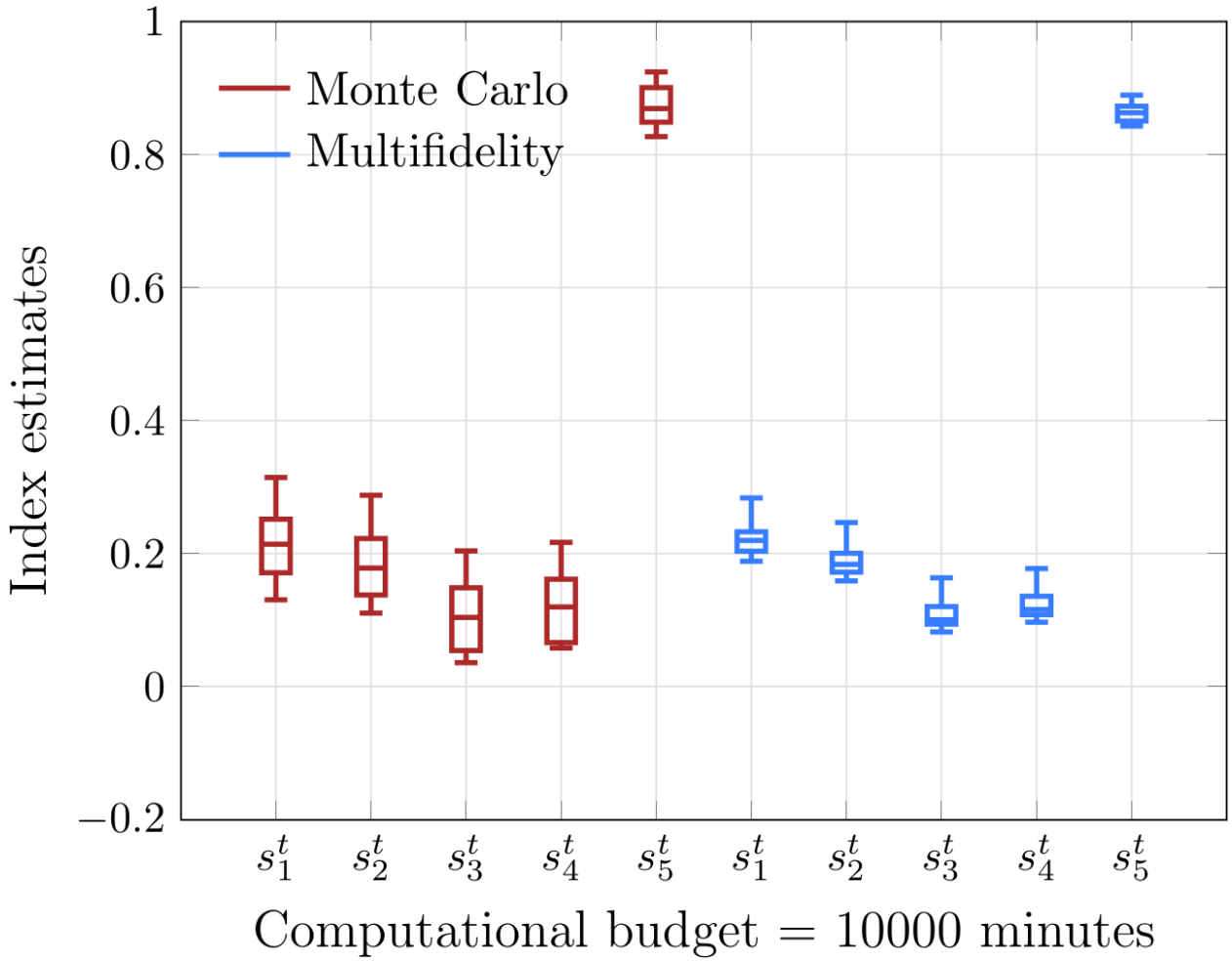


Output is maximum temperature in chamber

Models

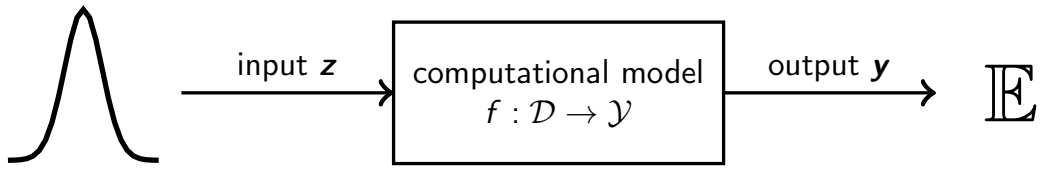
- Model based on finite differences serves as high-fidelity model
- Model with lower fidelity derived with proper orthogonal decomposition

MFGSA: Premixed flame: Results

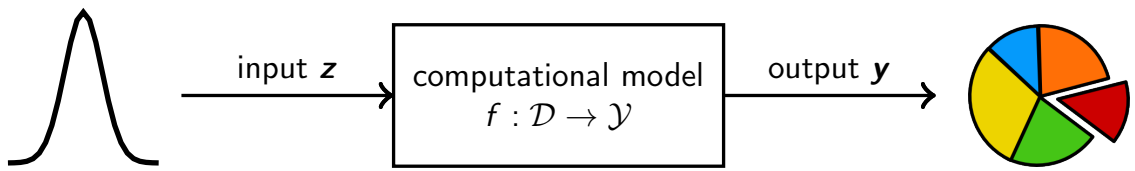


Uncertainty quantification tasks

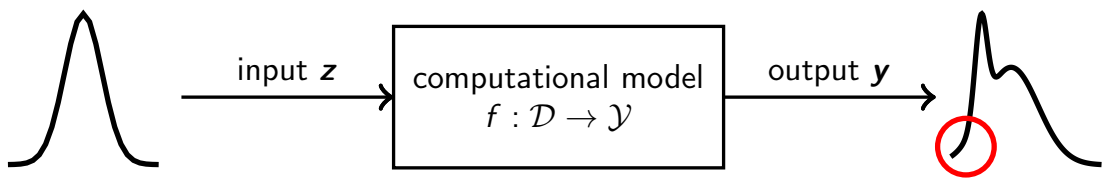
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



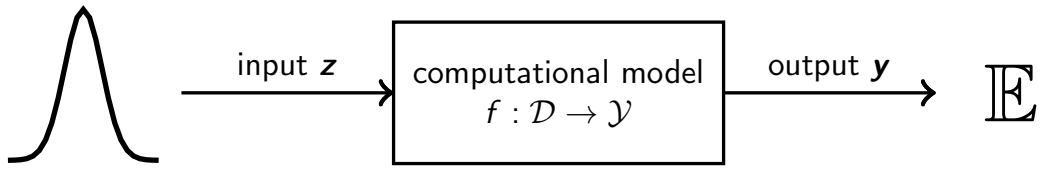
3. Multifidelity failure probability estimation



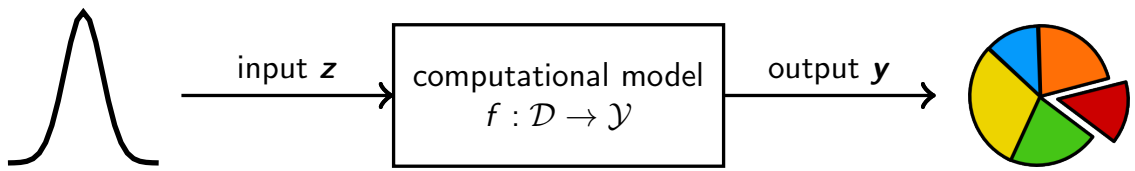
4. Other multifidelity uncertainty quantification tasks

Uncertainty quantification tasks

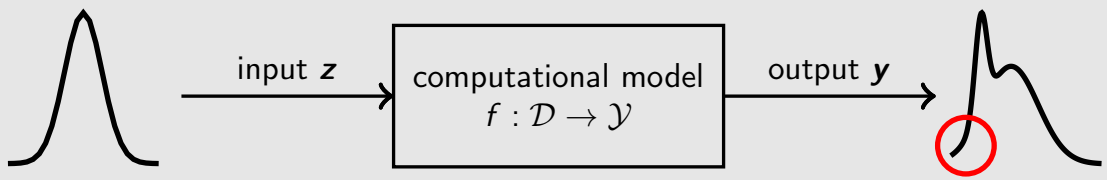
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



3. Multifidelity failure probability estimation



4. Other multifidelity uncertainty quantification tasks

MFIS: Failure probabilities

System described by high-fidelity model $f^{(1)} : \mathcal{D} \to \mathcal{Y}$

- Input $\boldsymbol{z} \in \mathcal{Z}$
- Output $\boldsymbol{y} \in \mathcal{Y}$
- Costs of one high-fidelity model evaluation $w_1 > 0$

Define indicator function

$$I^{(1)}(\boldsymbol{z}) = \begin{cases} 1\,, & f^{(1)}(\boldsymbol{z}) < 0 \\ 0\,, & \text{else}\,. \end{cases}$$

Indicator function $I^{(1)}(\boldsymbol{z}) = 1$ signals *failure* for input $\boldsymbol{z}$

Given random variable Z, estimate failure probability

$$P_f = \mathbb{E}_p[I^{(1)}(Z)]$$

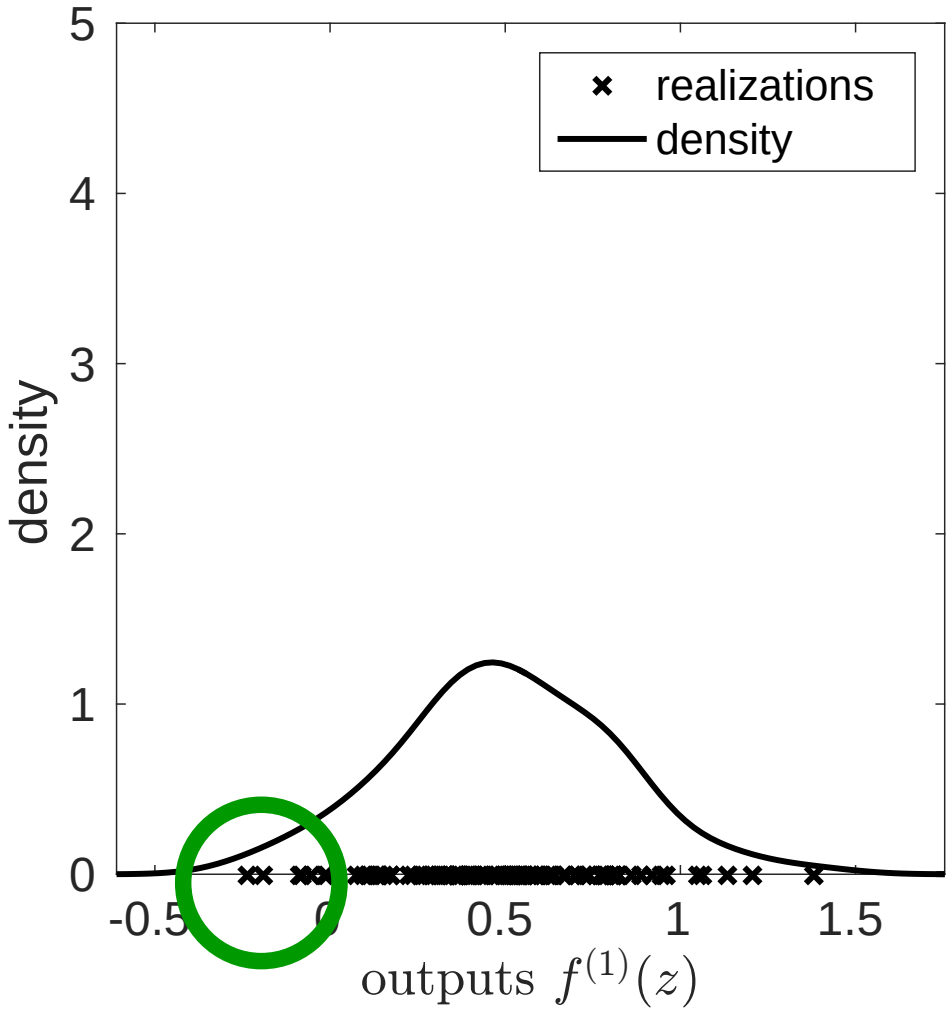
MFIS: Rare event simulation

- Monte Carlo estimator of P_f using $m \in \mathbb{N}$ realizations

$$P_f^{\mathrm{MC}} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)}(\boldsymbol{z}_i)$$

- If P_f small, then only few realizations with $f^{(1)}(\boldsymbol{z}) < 0$

- Require (very) large m to obtain Monte Carlo estimator with acceptable accuracy $\rightarrow$ expensive



MFIS: Rare event simulation is challenging

Costs of rare event simulation grow inverse proportional to P_f

- Monte Carlo estimation of P_f with m realizations

$$P_f^{\mathrm{MC}} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)}(\boldsymbol{z}_i)$$

- Relative mean-squared error (MSE) of P_f^{MC}

$$e(P_f^{\mathrm{MC}}) = \mathbb{E}_p \left[\left(\frac{P_f^{\mathrm{MC}} - P_f}{P_f} \right)^2 \right] = \frac{\mathrm{Var}_p \left[I^{(1)}(Z) \right]}{P_f^2 m} = \frac{1 - P_f}{P_f m}$$

- For constant m, the rel. MSE increases inverse proportional to P_f
- A small failure probability P_f needs to be compensated with a large number of samples m
- Example: For $P_f = 10^{-5}$ need $m \approx 10^7$ to achieve $e(P_f^{\mathrm{MC}}) \leq 10^{-2}$

Challenge

costs per sample + number of samples

MFIS: Rare events in aerospace engineering

Rare event simulation

- Failure probability estimation
- Reliability engineering

Risk assessment

- Communicate to upstream decision-making
- Mitigate catastrophic events

Risk-averse optimization

- Deliver baseline performance outside nominal operating conditions
- Take into account dynamics at limit states

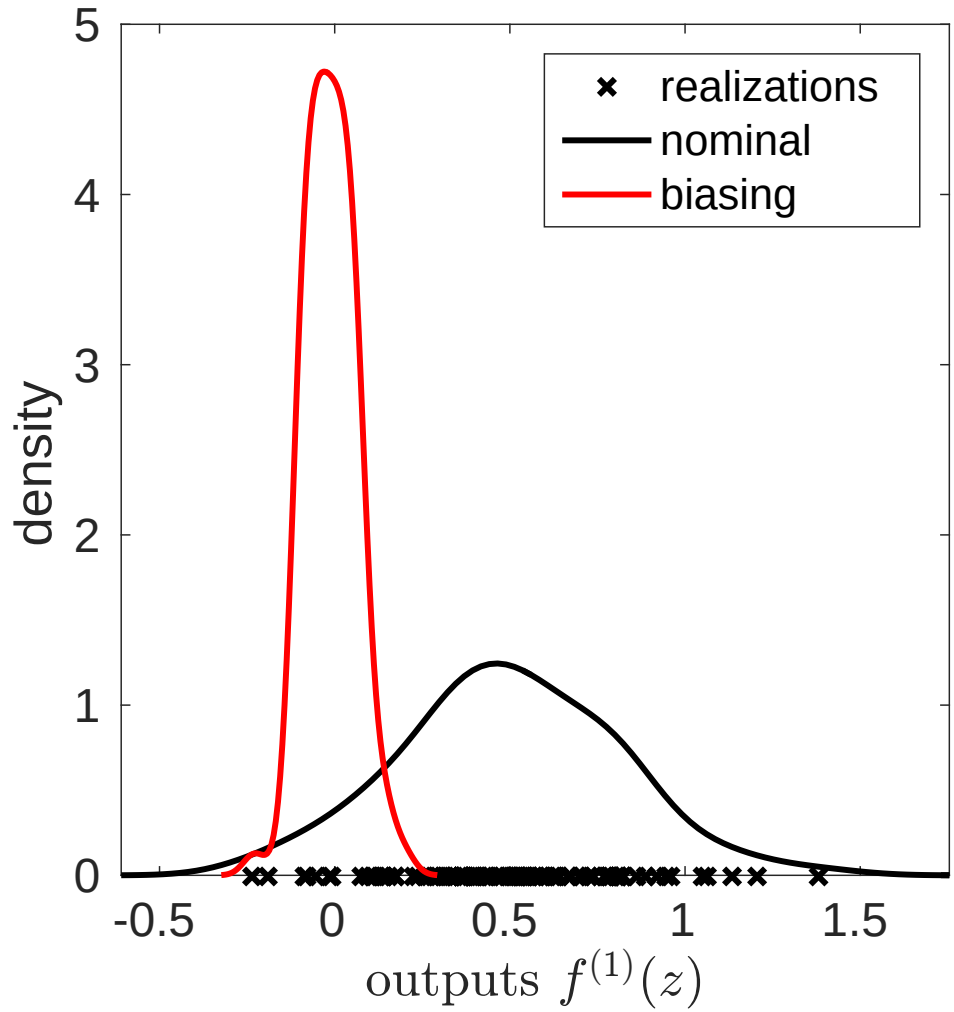
MFIS: Importance sampling

- Importance sampling (IS) creates biasing density q to put more weight on failure events
- Let $\hat{Z}$ be the corresponding random variable
- Introduce the weight function

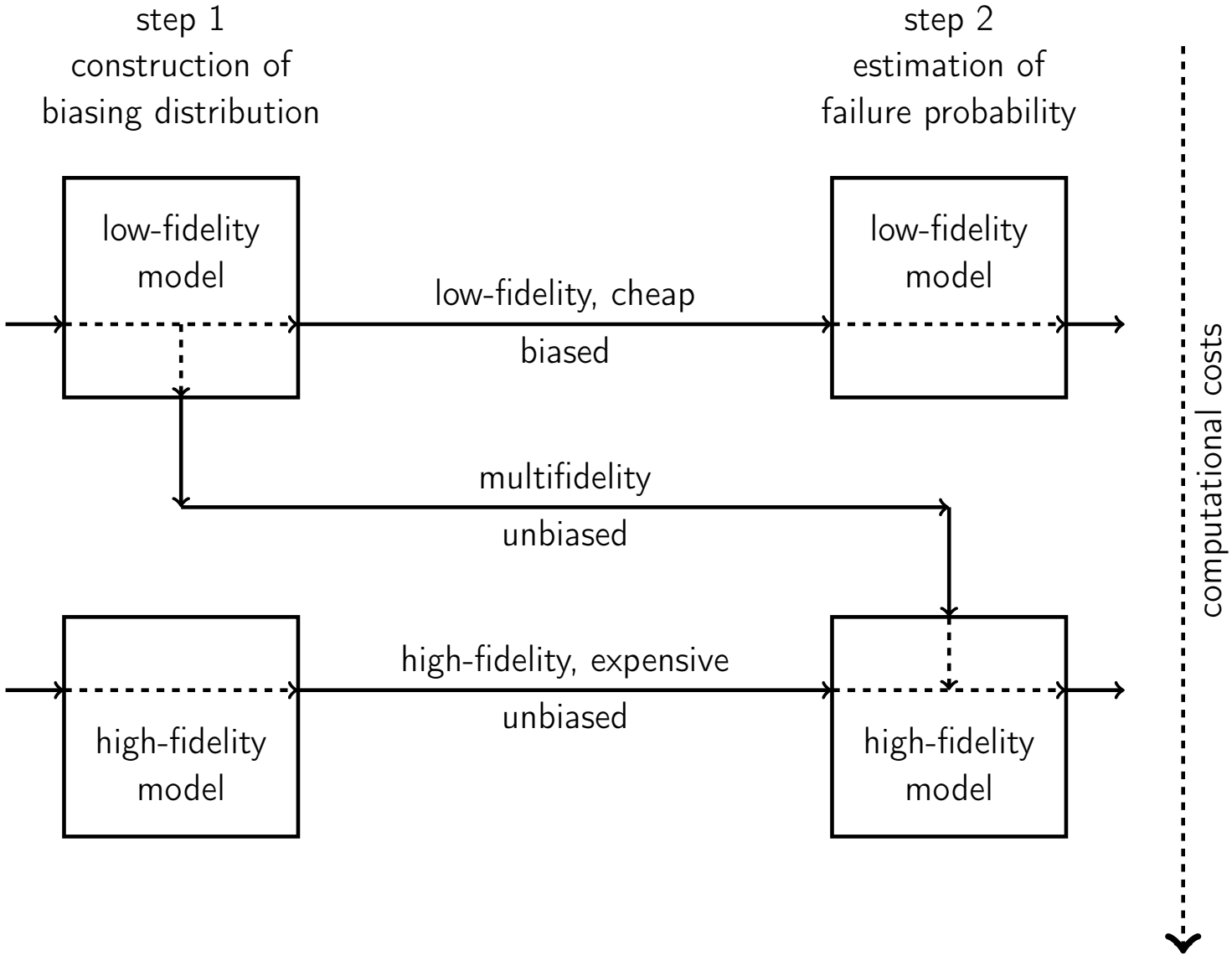
$$r(z') = \frac{p(\hat{z})}{p(\hat{z})}$$

- Reformulate failure probability

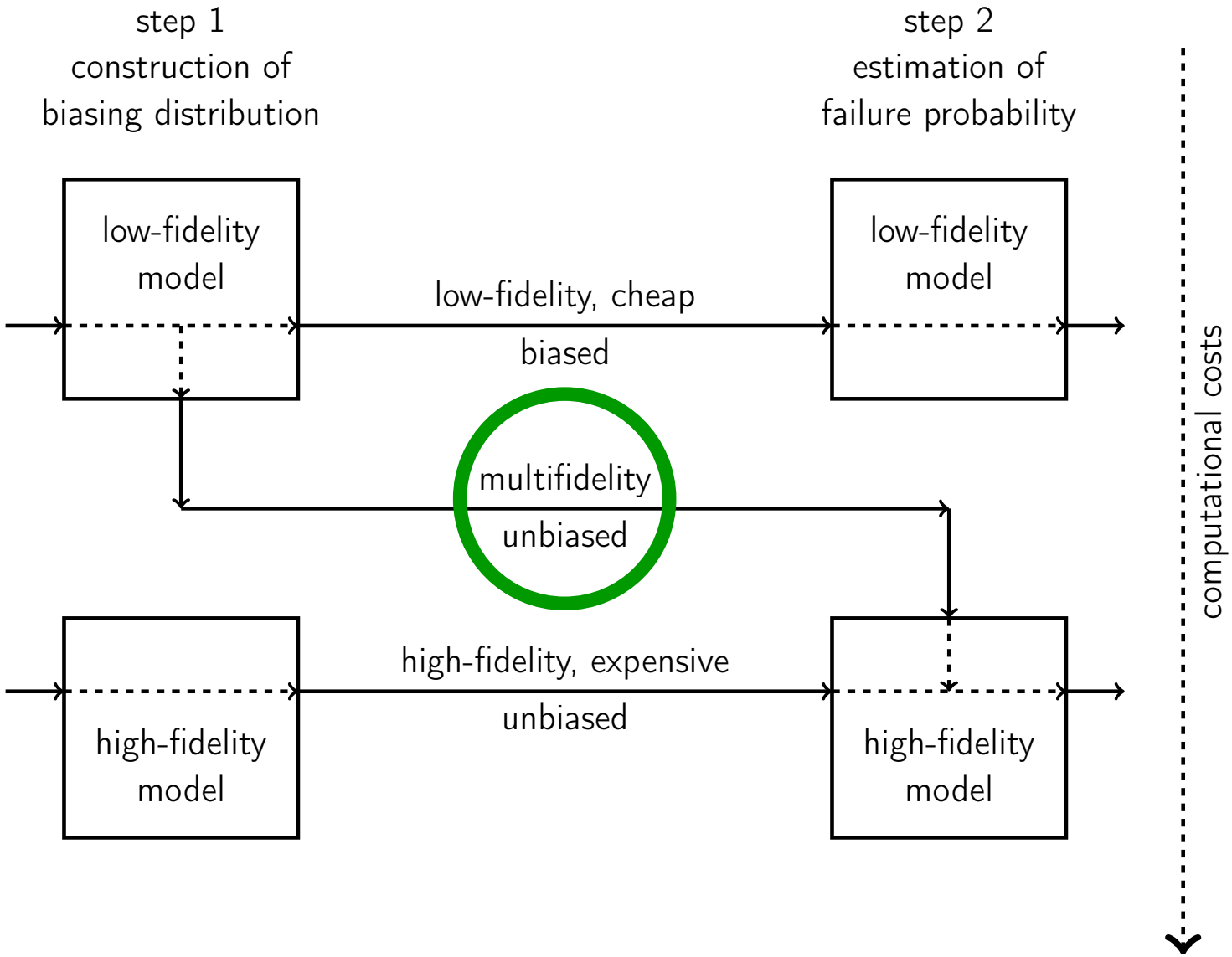
$$P_f = \mathbb{E}_p[I^{(1)}(Z)] = \mathbb{E}_q[I^{(1)}(\hat{Z})r(\hat{Z})]$$



MFIS: Multifidelity importance sampling



MFIS: Multifidelity importance sampling



MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ of Z
- Fit mixture model q (biasing) to realizations $\rightarrow$ `scikit-learn`, Matlab

$$\{\boldsymbol{z}_i \,|\, I^{(2)}(\boldsymbol{z}_i) = 1\,, i = 1, \ldots, n\}$$

- Derive random variable $\hat{Z}$ with density q

Step 2: Estimate P_f with high-fidelity model $f^{(1)}$

$$P_f^{\mathsf{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)}(\hat{\boldsymbol{z}}_i) \quad \frac{p(\hat{\boldsymbol{z}}_i)}{q(\hat{\boldsymbol{z}}_i)}$$

Multifidelity estimator P_f^{MFIS} is unbiased

$$P_{f^{(\mathbf{1})}} = \mathbb{E}_q[P_f^{\mathsf{MFIS}}]$$

MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ of Z
- Fit mixture model q (biasing) to realizations $\rightarrow$ `scikit-learn`, Matlab

$$\{\boldsymbol{z}_i \,|\, I^{(2)}(\boldsymbol{z}_i) = 1\,, i = 1, \ldots, n\}$$

- Derive random variable $\hat{Z}$ with density q

Step 2: Estimate P_f with high-fidelity model $f^{(1)}$

$$P_f^{\mathsf{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{I^{(1)}(\hat{\boldsymbol{z}}_i)}_{\substack{\text{uses} \\ \text{high-fidelity}}} \quad \frac{p(\hat{\boldsymbol{z}}_i)}{q(\hat{\boldsymbol{z}}_i)}$$

Multifidelity estimator P_f^{MFIS} is unbiased

$$P_{f^{(\mathbf{1})}} = \mathbb{E}_q[P_f^{\mathsf{MFIS}}]$$

MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ of Z
- Fit mixture model q (biasing) to realizations $\rightarrow$ `scikit-learn`, Matlab

$$\{\boldsymbol{z}_i \,|\, I^{(2)}(\boldsymbol{z}_i) = 1\,, i = 1, \ldots, n\}$$

- Derive random variable $\hat{Z}$ with density q

Step 2: Estimate P_f with high-fidelity model $f^{(1)}$

$$P_f^{\mathsf{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{I^{(1)}(\hat{\boldsymbol{z}}_i)}_{\substack{\text{uses} \\ \text{high-fidelity}}} \underbrace{\frac{p(\hat{\boldsymbol{z}}_i)}{q(\hat{\boldsymbol{z}}_i)}}_{\substack{\text{uses} \\ \text{low-fidelity}}}$$

Multifidelity estimator P_f^{MFIS} is unbiased

$$P_{f^{(\mathbf{1})}} = \mathbb{E}_q[P_f^{\mathsf{MFIS}}]$$

MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ of Z
- Fit mixture model q (biasing) to realizations $\rightarrow$ `scikit-learn`, Matlab

$$\{\boldsymbol{z}_i \,|\, I^{(2)}(\boldsymbol{z}_i) = 1\,, i = 1, \ldots, n\}$$

many realizations
but low-fidelity model

- Derive random variable $\hat{Z}$ with density q

Step 2: Estimate P_f with high-fidelity model $f^{(1)}$

$$P_f^{\text{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{I^{(1)}(\hat{\boldsymbol{z}}_i)}_{\substack{\text{uses} \\ \text{high-fidelity}}} \underbrace{\frac{p(\hat{\boldsymbol{z}}_i)}{q(\hat{\boldsymbol{z}}_i)}}_{\substack{\text{uses} \\ \text{low-fidelity}}}$$

Multifidelity estimator P_f^{MFIS} is unbiased

$$P_{f^{(1)}} = \mathbb{E}_q[P_f^{\text{MFIS}}]$$

MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ of Z
- Fit mixture model q (biasing) to realizations $\rightarrow$ `scikit-learn`, Matlab

$$\{\boldsymbol{z}_i \,|\, I^{(2)}(\boldsymbol{z}_i) = 1\,, i = 1, \ldots, n\}$$

many realizations
but low-fidelity model

- Derive random variable $\hat{Z}$ with density q

Step 2: Estimate P_f with high-fidelity model $f^{(1)}$

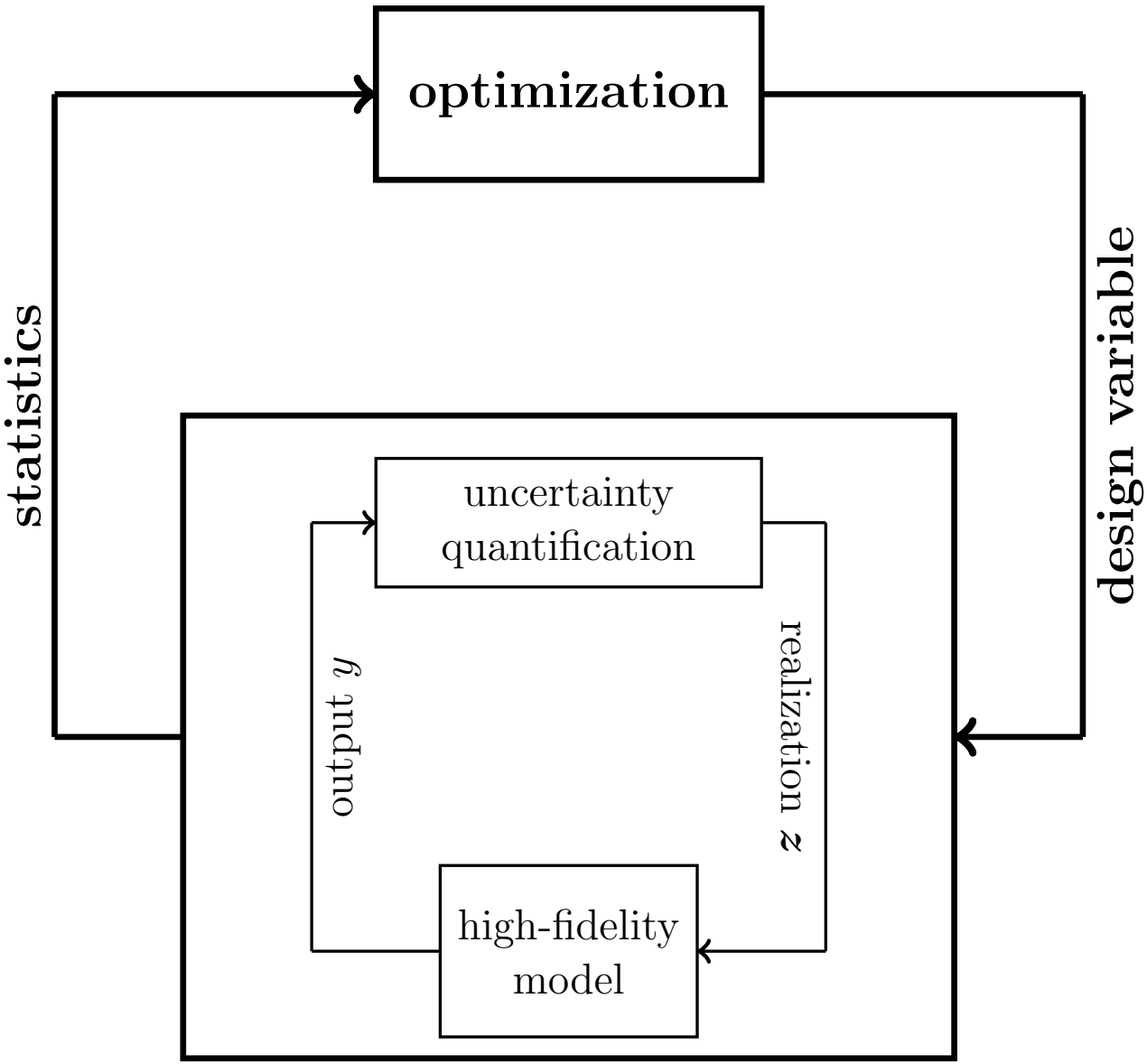
$$P_f^{\mathsf{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} \underbrace{I^{(1)}(\hat{\boldsymbol{z}}_i)}_{\substack{\text{uses} \\ \text{high-fidelity}}} \underbrace{\frac{p(\hat{\boldsymbol{z}}_i)}{q(\hat{\boldsymbol{z}}_i)}}_{\substack{\text{uses} \\ \text{low-fidelity}}}$$

high-fidelity model
but only few evals

Multifidelity estimator P_f^{MFIS} is unbiased

$$P_{f^{(1)}} = \mathbb{E}_q[P_f^{\mathsf{MFIS}}]$$

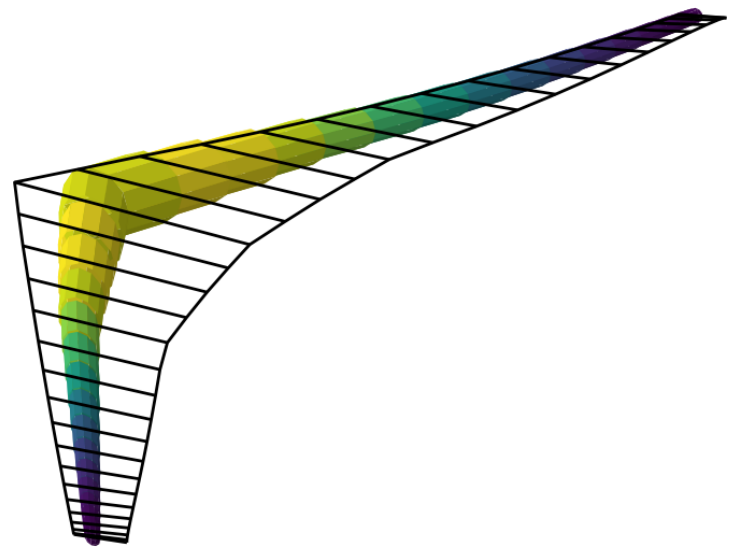
MFIS: Optimization for risk-averse designs



MFIS: Risk-averse design of wing

Consider baseline wing definition in OpenAeroStruct

- Design variables are thickness and position of control points
- Uncertain flight conditions (angle of attack, air density, Mach number)
- Output is fuel burn



Minimize fuel burn at limit states

$$\min_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[f^{(1)}(\boldsymbol{x}, Z) \mid f^{(1)}(\boldsymbol{x}, Z) \leq \beta]$$

Derive a data-fit surrogate at current design $\boldsymbol{x}$

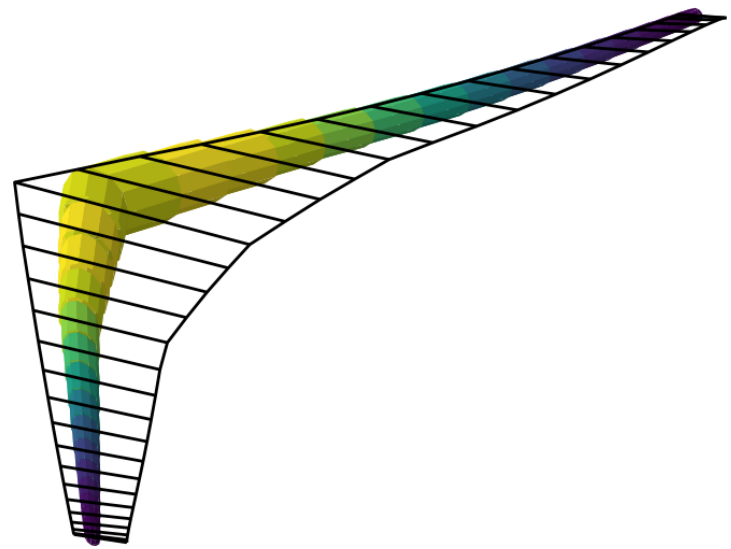
- Take a $3 \times 3 \times 3$ equidistant grid in stochastic domain
- Evaluate high-fidelity model at those 27 points at current design $\boldsymbol{x}$
- Derive linear interpolant of output

Apply multifidelity pre-conditioned cross-entropy method

MFIS: Risk-averse design of wing

Consider baseline wing definition in OpenAeroStruct

- Design variables are thickness and position of control points
- Uncertain flight conditions (angle of attack, air density, Mach number)
- Output is fuel burn



Minimize fuel burn at limit states

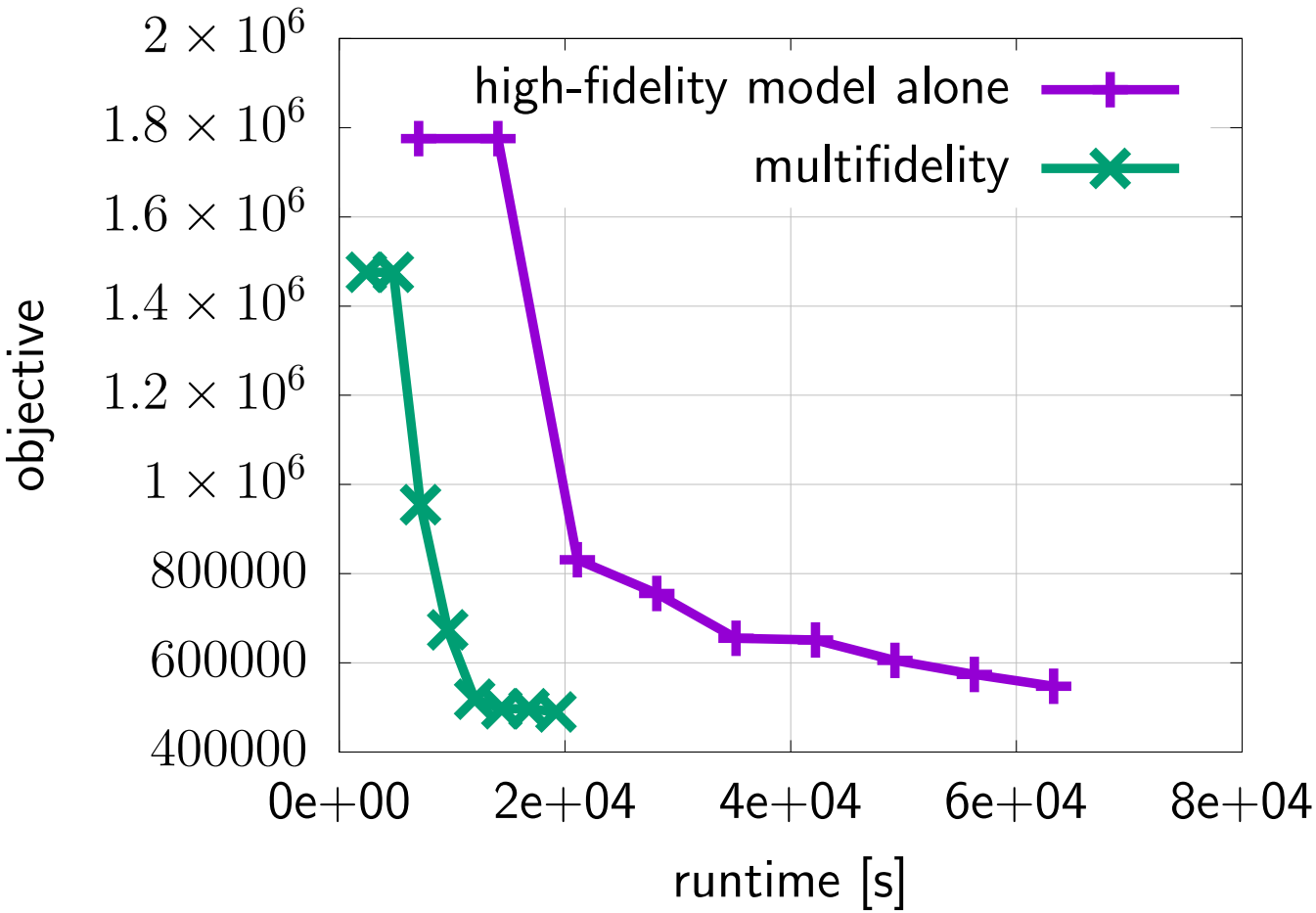
$$\min_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[f^{(1)}(\boldsymbol{x}, Z) \,|\, f^{(1)}(\boldsymbol{x}, Z) \leq \beta]$$

Derive a data-fit surrogate at current design $\boldsymbol{x}$

- Take a $3 \times 3 \times 3$ equidistant grid in stochastic domain
- Evaluate high-fidelity model at those 27 points at current design $\boldsymbol{x}$
- Derive linear interpolant of output

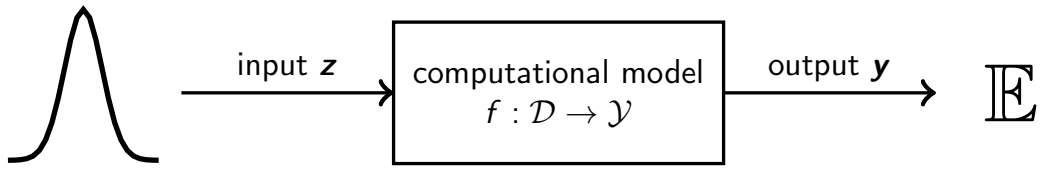
Apply multifidelity pre-conditioned cross-entropy method

MFIS: Risk-averse design of wing (cont'd)

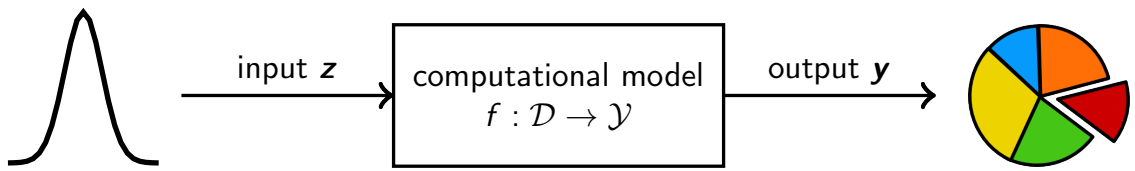


Uncertainty quantification tasks

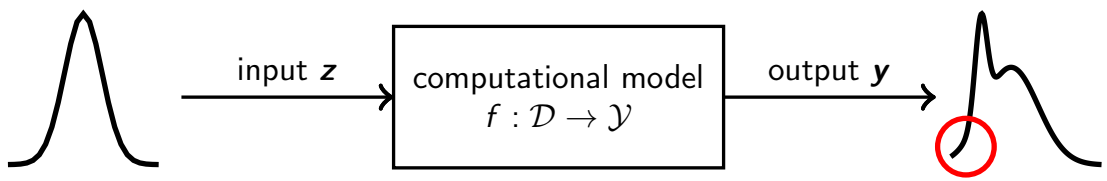
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



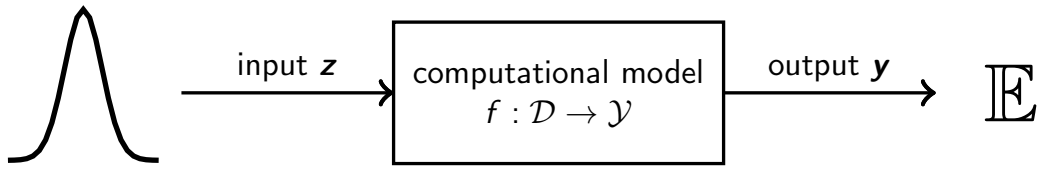
3. Multifidelity failure probability estimation



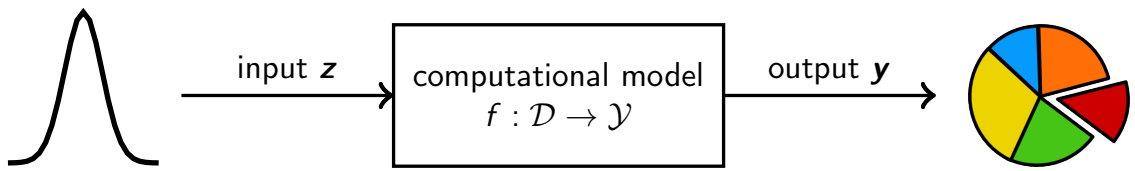
4. Other multifidelity uncertainty quantification tasks

Uncertainty quantification tasks

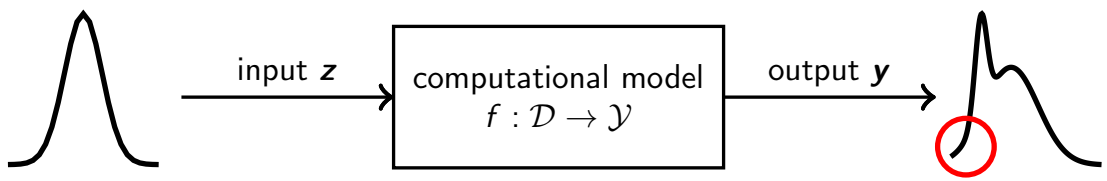
1. Multifidelity uncertainty propagation



2. Multifidelity sensitivity analysis



3. Multifidelity failure probability estimation



4. Other multifidelity uncertainty quantification tasks

Outlook: Inverse problems

Bayesian inference of parameters z from data y

- Parameters represented as random variable z with prior $p(z)$
- Define likelihood $p(y|z)$ of data y using model f
- Update distribution of z ("infer") with Bayes' rule

$$\underbrace{p(z|y)}_{\text{posterior}} \propto \underbrace{p(y|z)}_{\text{likelihood}} \underbrace{p(z)}_{\text{prior}}$$

Outlook: Inverse problems (cont'd)

$$\underbrace{p(\boldsymbol{z}|\boldsymbol{y})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{y}|\boldsymbol{z})}_{\text{likelihood}} \underbrace{p(\boldsymbol{z})}_{\text{prior}}$$

Posterior provides complete description of uncertainties in $\boldsymbol{z}$

- Input to future simulations for *predictions with quantified uncertainties*
- Explore posterior to *reduce uncertainties in future predictions*

Sampling posterior $p(\boldsymbol{z}|\boldsymbol{y})$

- Evaluate posterior expectation for function g

$$\mathbb{E}[g] = \int g(\boldsymbol{z}) p(\boldsymbol{z}|\boldsymbol{y}) \mathrm{d}\boldsymbol{z}$$

- Samples required as inputs in upstream simulations
- Explore posterior to decide where to take new data points
- Estimate quantiles

Making sampling tractable $\Rightarrow$ multifidelity

Outlook: Inverse problems (cont'd)

$$\underbrace{p(\boldsymbol{z}|\boldsymbol{y})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{y}|\boldsymbol{z})}_{\text{likelihood}} \underbrace{p(\boldsymbol{z})}_{\text{prior}}$$

Posterior provides complete description of uncertainties in $\boldsymbol{z}$

- Input to future simulations for *predictions with quantified uncertainties*
- Explore posterior to *reduce uncertainties in future predictions*

Sampling posterior $p(\boldsymbol{z}|\boldsymbol{y})$

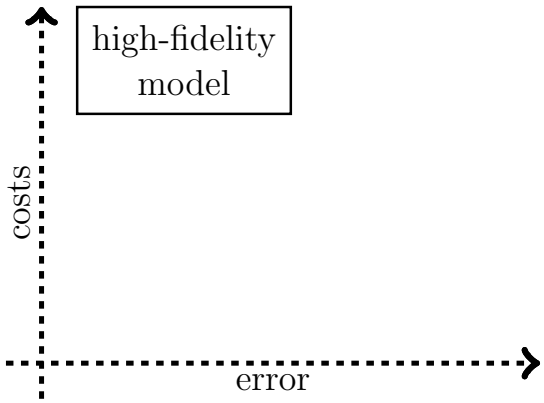
- Evaluate posterior expectation for function g

$$\mathbb{E}[g] = \int g(\boldsymbol{z}) p(\boldsymbol{z}|\boldsymbol{y}) \mathrm{d}\boldsymbol{z}$$

- Samples required as inputs in upstream simulations
- Explore posterior to decide where to take new data points
- Estimate quantiles

Making sampling tractable $\Rightarrow$ **multifidelity**

Outlook: Learning surrogates for multifidelity

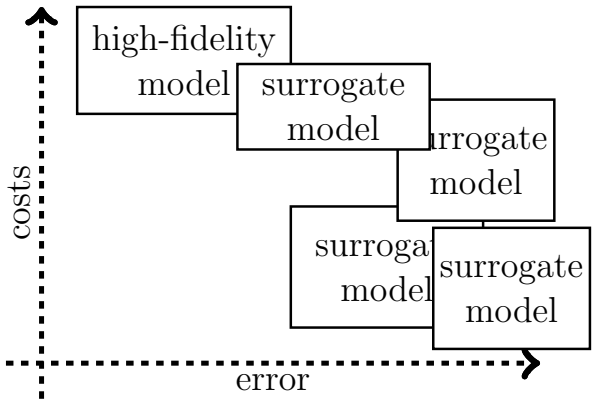


Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

$\Rightarrow$ **Need for model reduction that targets multifidelity**

Outlook: Learning surrogates for multifidelity

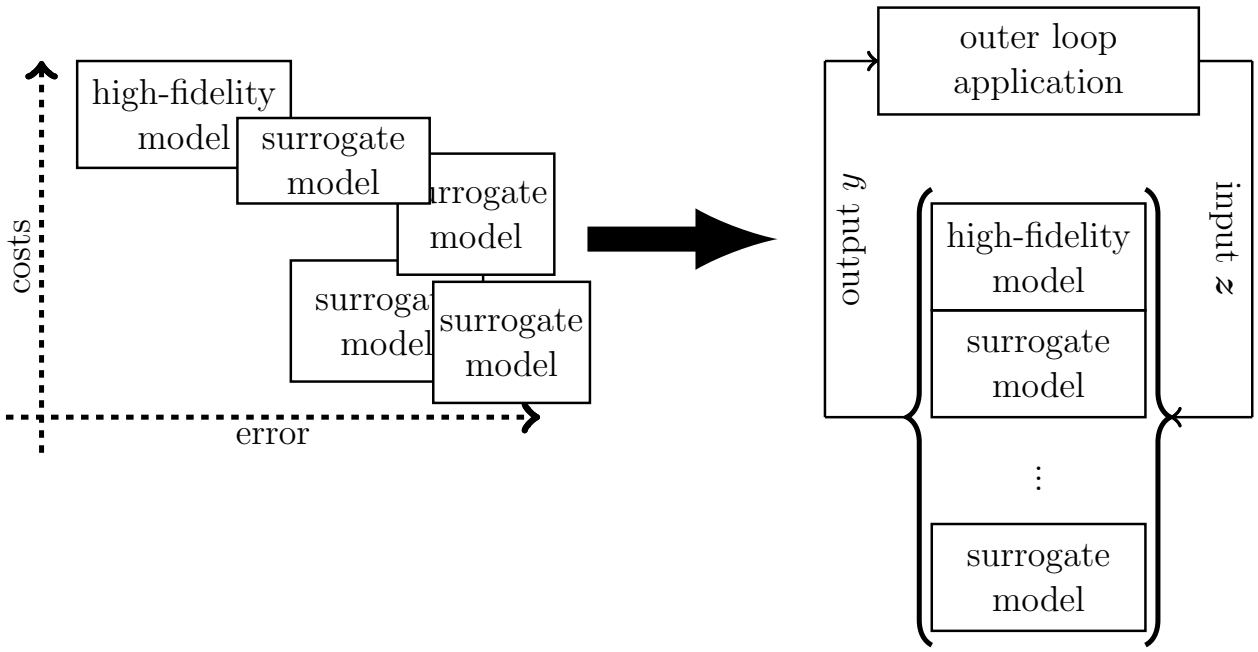


Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

$\Rightarrow$ **Need for model reduction that targets multifidelity**

Outlook: Learning surrogates for multifidelity

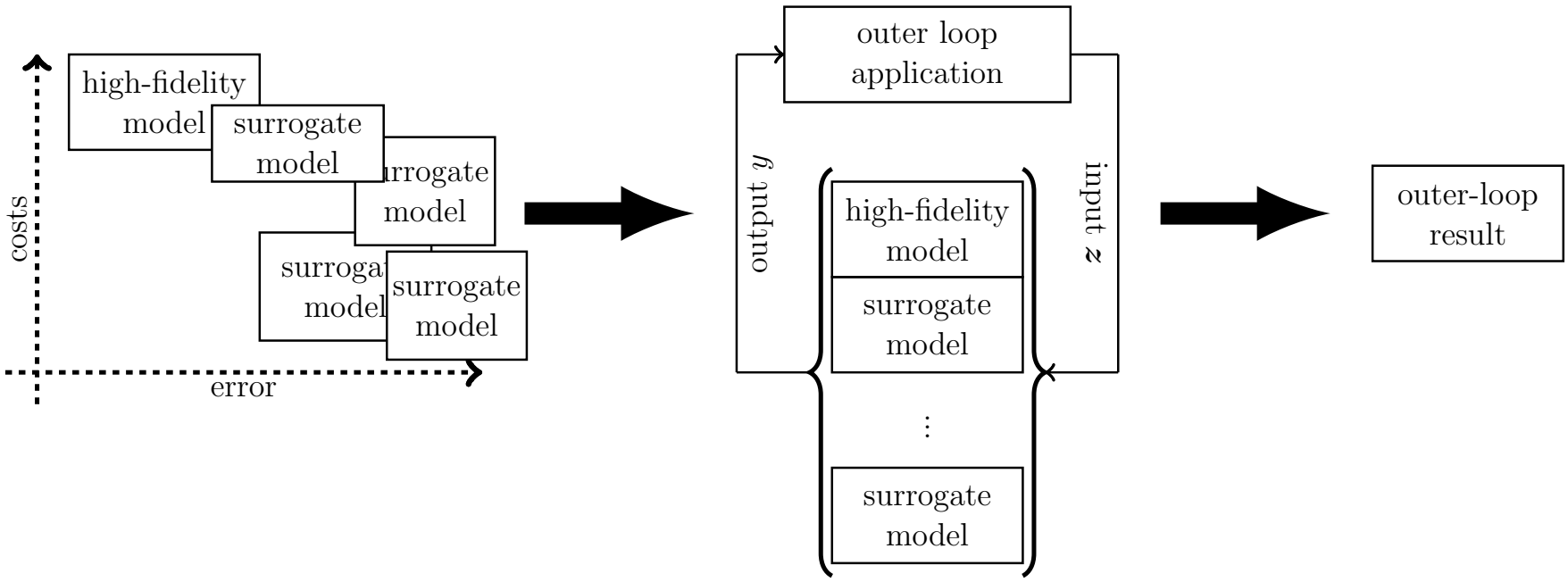


Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

$\Rightarrow$ **Need for model reduction that targets multifidelity**

Outlook: Learning surrogates for multifidelity

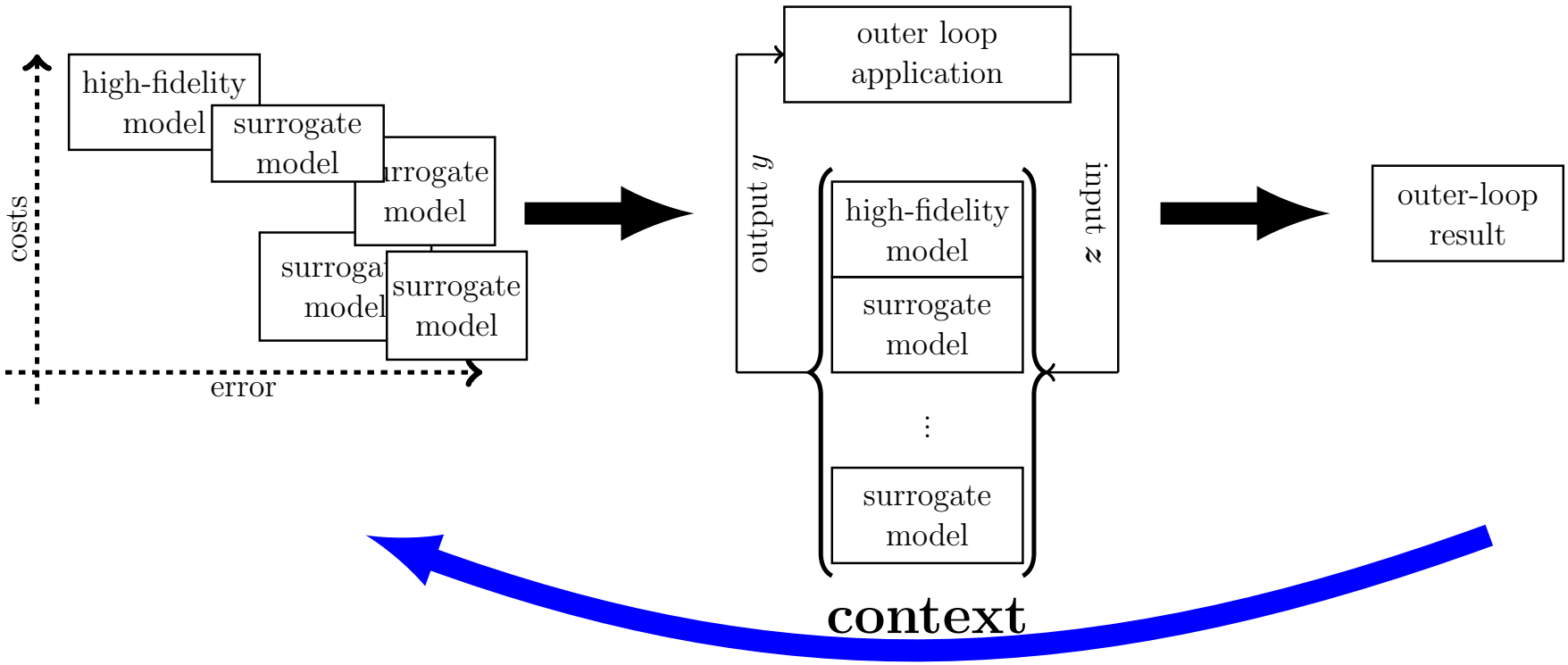


Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

$\Rightarrow$ **Need for model reduction that targets multifidelity**

Outlook: Learning surrogates for multifidelity

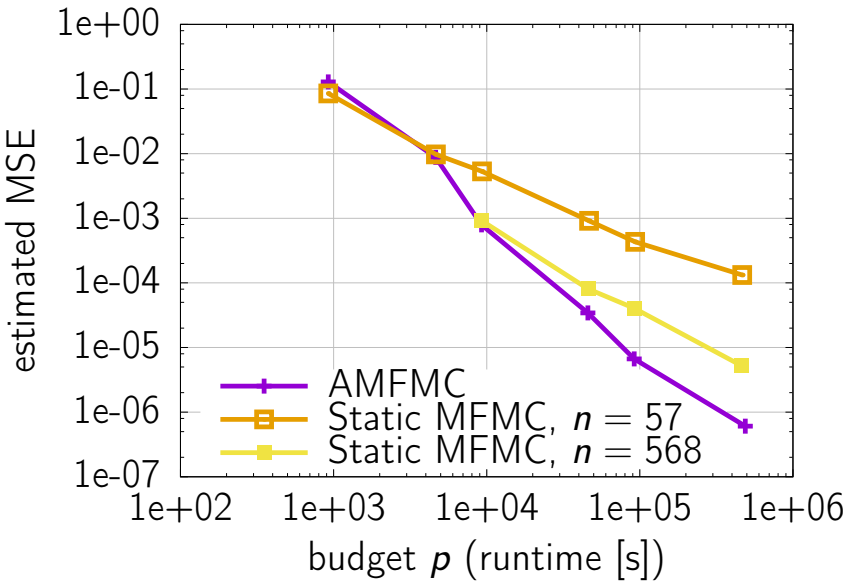


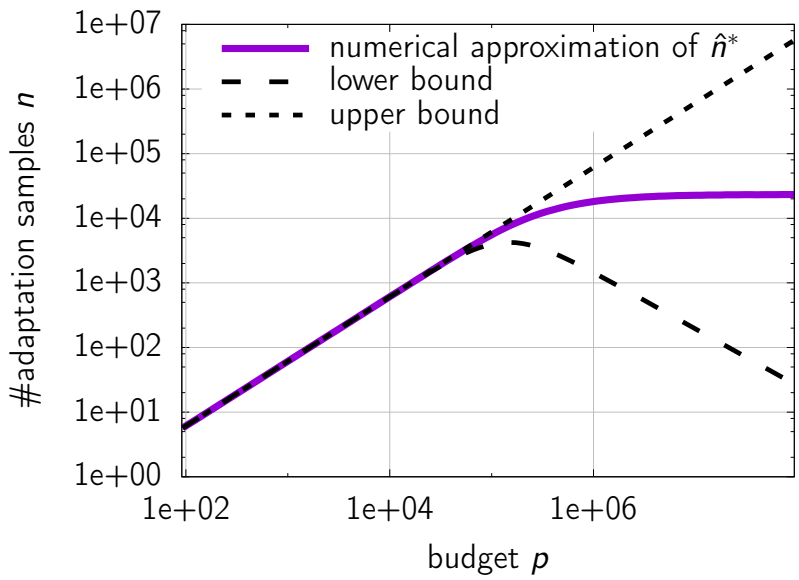
Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

$\Rightarrow$ **Need for learning surrogates that target multifidelity**

Outlook: Learning surrogates for multifidelity





Adapt surrogate models - but not too much

- Adapting surrogate models towards multifidelity is beneficial
- Crude, cheap surrogates can have better costs/benefit ratio
- Proved for MFMC that optimal amount to spend on learning surrogates is bounded

[P.: ***Multifidelity Monte Carlo estimation with adaptive low-fidelity models***. **SIAM/ASA Journal on Uncertainty Quantification, 2019.**]

Survey with *many* references

SIAM Review
Vol. 60, No. 3, pp. 550–591

Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*

Benjamin Peherstorfer†
Karen Willcox‡
Max Gunzburger§

Abstract. In many situations across computational science and engineering, multiple computational models are available that describe a system of interest. These different models have varying evaluation costs and varying fidelities. Typically, a computationally expensive high-fidelity model describes the system with the accuracy required by the current application at hand, while lower-fidelity models are less accurate but computationally cheaper than the high-fidelity model. Outer-loop applications, such as optimization, inference, and uncertainty quantification, require multiple model evaluations at many different inputs, which often leads to computational demands that exceed available resources if only the high-fidelity model is used. This work surveys multifidelity methods that accelerate the solution of outer-loop applications by combining high-fidelity and low-fidelity model evaluations, where the low-fidelity evaluations arise from an explicit low-fidelity model (e.g., a simplified physics approximation, a reduced model, a data-fit surrogate) that approximates the same output quantity as the high-fidelity model. The overall premise of these multifidelity methods is that low-fidelity models are leveraged for speedup while the high-fidelity model is kept in the loop to establish accuracy and/or convergence guarantees. We categorize multifidelity methods according to three classes of strategies: adaptation, fusion, and filtering. The paper reviews multifidelity methods in the outer-loop contexts of uncertainty propagation, inference, and optimization.

Key words. multifidelity, surrogate models, model reduction, multifidelity uncertainty quantification, multifidelity uncertainty propagation, multifidelity statistical inference, multifidelity optimization

AMS subject classifications. 65-02, 62-02, 49-02

DOI. 10.1137/16M1082469

Further reading on methods covered in this talk

[1] L. W. T. Ng and K. Willcox.
Multifidelity approaches for optimization under uncertainty.
International Journal for Numerical Methods in Engineering, 100(10):746–772, 2014.

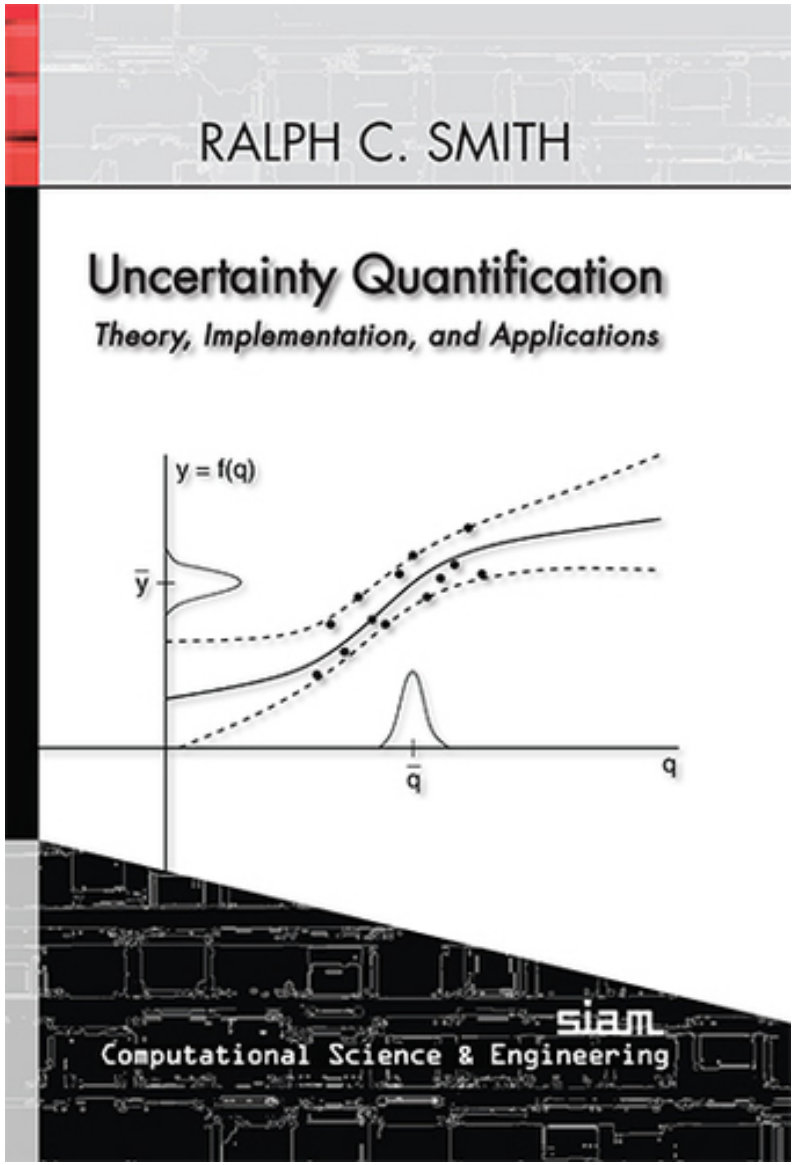
[2] B. Peherstorfer, T. Cui, Y. Marzouk, and K. Willcox.
Multifidelity importance sampling.
Computer Methods in Applied Mechanics and Engineering, 300:490–509, 2016.

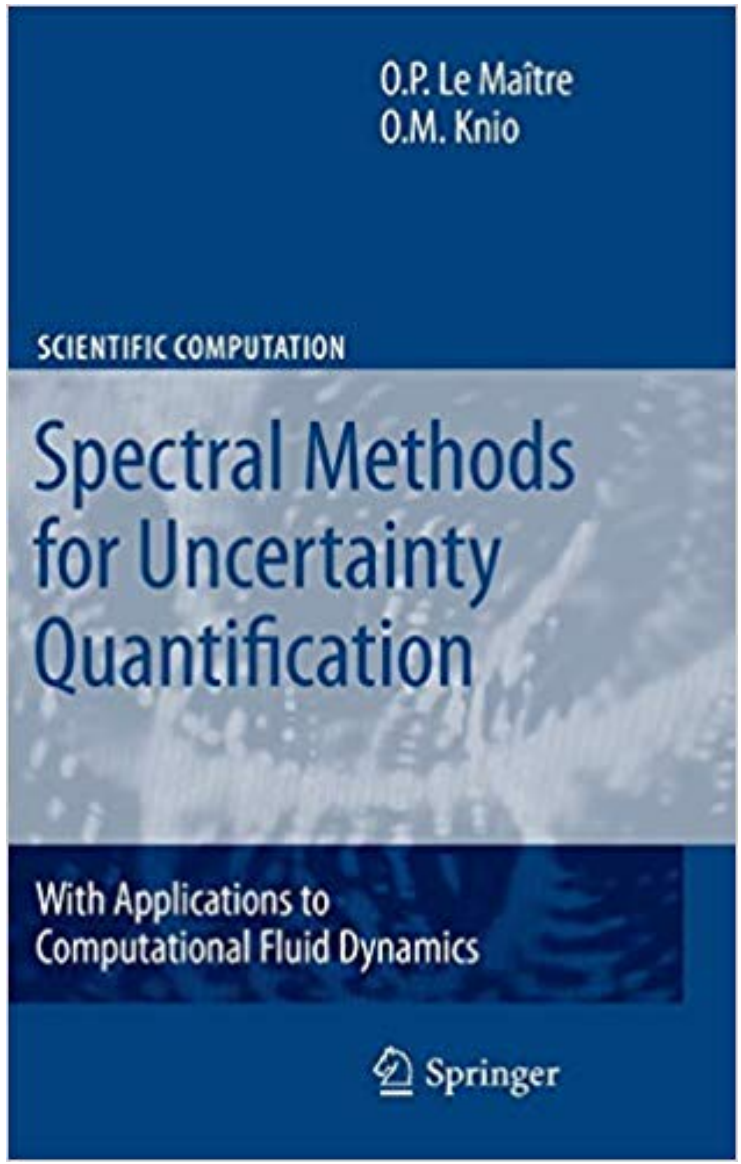
[3] B. Peherstorfer, B. Kramer, and K. Willcox.
Multifidelity preconditioning of the cross-entropy method for rare event simulation and failure probability estimation.
SIAM/ASA Journal on Uncertainty Quantification, 6(2):737–761, 2018.

[4] B. Peherstorfer, K. Willcox, and M. Gunzburger.
Optimal model management for multifidelity monte carlo estimation.
SIAM Journal on Scientific Computing, 38(5):A3163–A3194, 2016.

[5] E. Qian, B. Peherstorfer, D. O'Malley, V. Vesselinov, and K. Willcox.
Multifidelity monte carlo estimation of variance and sensitivity indices.
SIAM/ASA Journal on Uncertainty Quantification, 6(2):683–706, 2018.

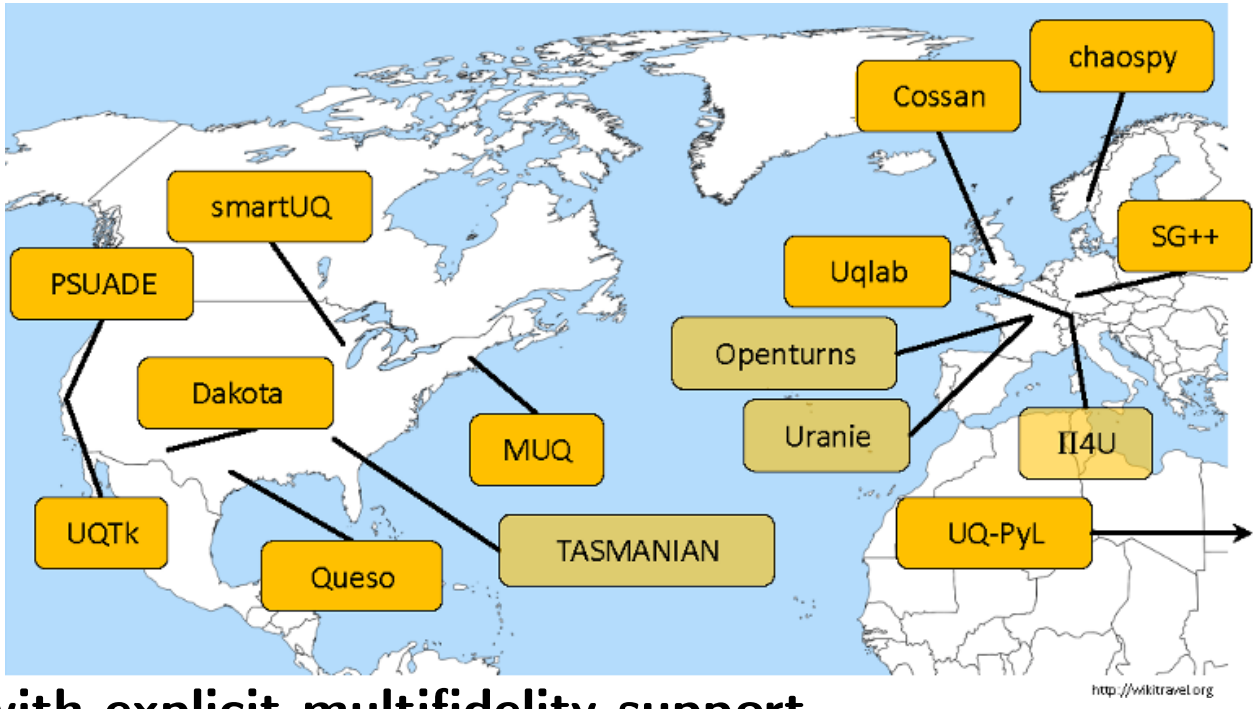
Books on uncertainty quantification





Software

Software for uncertainty quantification



[Figure: Pflüger et al., 2016]

Software with explicit multifidelity support

DAKOTA
Explore and predict with confidence.

https://dakota.sandia.gov/

MFMC

https://github.com/pehersto/mfmc

Further reading on methods covered in this talk

[1] L. W. T. Ng and K. Willcox.
Multifidelity approaches for optimization under uncertainty.
International Journal for Numerical Methods in Engineering, 100(10):746–772, 2014.

[2] B. Peherstorfer, T. Cui, Y. Marzouk, and K. Willcox.
Multifidelity importance sampling.
Computer Methods in Applied Mechanics and Engineering, 300:490–509, 2016.

[3] B. Peherstorfer, B. Kramer, and K. Willcox.
Multifidelity preconditioning of the cross-entropy method for rare event simulation and failure probability estimation.
SIAM/ASA Journal on Uncertainty Quantification, 6(2):737–761, 2018.

[4] B. Peherstorfer, K. Willcox, and M. Gunzburger.
Optimal model management for multifidelity monte carlo estimation.
SIAM Journal on Scientific Computing, 38(5):A3163–A3194, 2016.

[5] E. Qian, B. Peherstorfer, D. O'Malley, V. Vesselinov, and K. Willcox.
Multifidelity monte carlo estimation of variance and sensitivity indices.
SIAM/ASA Journal on Uncertainty Quantification, 6(2):683–706, 2018.

What we covered this week

- Introduction to (intrusive) model reduction
- Learning reduced models from data
- Error estimation of learned reduced models
- Learning from frequency-response data
- Multi-fidelity uncertainty quantification