Sampling low-dimensional Markovian dynamics for learning certified reduced models from data

Wayne Isaac Tan Uy and Benjamin Peherstorfer
Courant Institute of Mathematical Sciences, New York University

February 2020
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 \textit{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

\[
\frac{d}{dt} x(t; \mu) = f(x(t; \mu), u(t); \mu) \\
= \sum_{i=1}^{\ell} A_i(\mu)x^i(t; \mu) + B(\mu)u(t)
\]

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

Lifting and transformations

- Koopman lifts nonlinear systems to infinite linear systems [Rowley et al, 2009], [Schmid, 2010]
Intro: Beyond polynomial terms (nonintrusive)

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)

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

Elizabeth Qian, Boris Kramer

We present Lift & Learn, a physics-informed method for learning reduced models that can recover the generalization accuracy of traditional intrusive reduced models. We exploit knowledge of the governing PDE to identify a lifting map (variable transformation + auxiliary variables) that exposes quadratic structure in the PDE.

Lifting maps as approximations to nonlinear model reduction and a non-intrusive polynomial operator inference.

Deriving low-dimensional models

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

- Solution-based reduced models rely on full knowledge of physics and their construction traditionally requires intrusive access to codes.
- Data-driven 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.
- Lift maps as approximations to nonlinear model reduction and a non-intrusive polynomial operator inference.

Lifting PDEs to quadratic form

Consider the general nonlinear governing PDE with state $\frac{d}{dt} x = f(t)$.

A quadratic lifting map $T$ transforms and augments the system state so that the PDE in the lifted state, $\dot{y} = T_1(y)$, contains only quadratic nonlinearities, e.g.,

\[ \dot{y} = a_{ij} y_i y_j + b_i y_i + c_i y_j + d_i y_i y_j, \]

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

Example

Original PDE

\[ \frac{d}{dt} x = A x + b. \]

Lifted PDE

\[ \frac{d}{dt} y = A T(x) + b T(x). \]

How general is the lifting derived?

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

Is the lifting derived?

Our current strategy is difficult to perform auxillary variables are not quadratic terms of the PDE and augment the system with evolution equations for these new variables.

Acknowledgments

The authors acknowledge the NSF, the DOE, the Air Force, the DARPA, and the USAF for financial support. The authors also acknowledge the support of the Department of Energy High Performance Computing Applications Program (DE-SC0001250), the DOE Office of Science (BER), the Defense Advanced Research Projects Agency (DARPA), and the Simons Foundation.

References

[5] ...
Intro: Beyond polynomial terms (nonintrusive)

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

Elizabeth Qian, Boris Kramer

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

Lift & Learn

1. Solve N-dimensional spatial discretization of original nonlinear PDE to generate K snapshots. Apply lifting map to snapshot data to obtain a lifted state and time derivative pairs (\( \mathbf{x}, \dot{\mathbf{x}} \)).
2. Compute a d-dimensional global basis, \( \mathbf{V} \), for the lifted data, e.g., via Proper Orthogonal Decomposition (POD) and project data:
   \( \mathbf{x} = \mathbf{V} \mathbf{y} \)
   where \( \mathbf{d} \) or \( \mathbf{N} \). The reduced model for projected data can be parameterized by small, dense matrix operators:
   \( \frac{d\mathbf{y}}{dt} = \mathbf{A} \mathbf{y} + \mathbf{B} \mathbf{w} \mathbf{y} \mathbf{w} \)
3. Use least-squares operator inference procedure to learn \( \mathbf{A} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}} \) and \( \mathbf{B} \in \mathbb{R}^{\mathbf{d} \times \mathbf{n}} \)

Generalization & accuracy: FitzHugh-Nagumo neuron activation model

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

\[
\frac{d}{dt} x(t; \mu) = f(x(t; \mu), u(t); \mu) = \sum_{i=1}^{\ell} A_i(\mu)x^i(t; \mu) + B(\mu)u(t)
\]

Parameters

- Infer models \( \hat{f}(\cdot, \cdot; \mu_1), \ldots, \hat{f}(\cdot, \cdot; \mu_M) \) at parameters \( \mu_1, \ldots, \mu_M \in \mathcal{D} \)
- For new \( \mu \in \mathcal{D} \), interpolate operators of [Amsallem et al., 2008], [Degroote et al., 2010]

\[ \hat{f}(\mu_1), \ldots, \hat{f}(\mu_M) \]

Trajectories

\[
X = [x_1, \ldots, x_K] \in \mathbb{R}^{N \times K} \\
U = [u_1, \ldots, u_K] \in \mathbb{R}^{p \times K}
\]
Intro: Parametrized systems

Consider time-invariant system with polynomial nonlinear terms

\[
\frac{d}{dt} x(t) = f(x(t), u(t)) = \sum_{i=1}^{\ell} A_i x^i(t) + B u(t)
\]

Parameters

- Infer models \( \hat{f}(\cdot, \cdot; \mu_1), \ldots, \hat{f}(\cdot, \cdot; \mu_M) \) at parameters \( \mu_1, \ldots, \mu_M \in D \)

- For new \( \mu \in D \), interpolate operators of [Amsallem et al., 2008], [Degroote et al., 2010]

\( \hat{f}(\mu_1), \ldots, \hat{f}(\mu_M) \)

Trajectories

\[
X = [x_1, \ldots, x_K] \in \mathbb{R}^{N \times K}
\]
\[
U = [u_1, \ldots, u_K] \in \mathbb{R}^{p \times K}
\]
Intro: Parametrized systems

Consider time-invariant system with polynomial nonlinear terms

\[ x_{k+1} = f(x_k, u_k) \]

\[ = \sum_{i=1}^{\ell} A_i x_k^i + B u_k, \quad k = 0, \ldots, K - 1 \]

Parameters

- Infer models \( \hat{f}(\cdot, \cdot; \mu_1), \ldots, \hat{f}(\cdot, \cdot; \mu_M) \) at parameters

\[ \mu_1, \ldots, \mu_M \in \mathcal{D} \]

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

\[ \hat{f}(\mu_1), \ldots, \hat{f}(\mu_M) \]

Trajectories

\[ X = [x_1, \ldots, x_K] \in \mathbb{R}^{N \times K} \]

\[ U = [u_1, \ldots, u_K] \in \mathbb{R}^{p \times K} \]
Intro: Classical (intrusive) model reduction

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

1. Construct $n$-dim. basis $V = [v_1, \ldots, v_n] \in \mathbb{R}^{N \times n}$
   - 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.,
   \[
   \tilde{A}_i = V^T \left( \underbrace{A_i (V \otimes \cdots \otimes V)}_{n \times n^i} \right), \quad \tilde{B} = V^T B
   \]

3. Construct reduced model
   \[
   \tilde{x}_{k+1} = \tilde{f}(\tilde{x}_k, u_k) = \sum_{i=1}^{\ell} \tilde{A}_i \tilde{x}_k^i + \tilde{B}u_k, \quad k = 0, \ldots, K - 1
   \]

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

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]
Intro: 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.,

   \[
   \tilde{A}_i = V^T \left( A_i (V \otimes \cdots \otimes V) \right), \quad \tilde{B} = V^T B
   \]

3. Construct reduced model

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

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

[Rozza, Huynh, Patera, 2007], [Benner, Gugercin, Willcox, 2015]
Our approach: Learn reduced models from data

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

\[
U = \begin{bmatrix} u_0 & \cdots & u_{K-1} \end{bmatrix}
\]

to obtain trajectory

\[
X = \begin{bmatrix} x_0 & x_1 & \cdots & x_K \end{bmatrix}
\]

Learn model \( \hat{f} \) from data \( U \) and \( X \)

\[
\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k) = \sum_{i=1}^{\ell} \hat{A}_i x_k^i + \hat{B} u_k, \quad k = 0, \ldots, K - 1
\]
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]
Outline

- Introduction and motivation
- Operator inference for learning low-dimensional models
- Sampling Markovian data for recovering reduced models
- Rigorous and pre-asymptotic error estimators
- Learning time delays to go beyond Markovian models
- Conclusions
OpInf: Fitting low-dim model to trajectories

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

\[
V = [v_1, \cdots, v_n] \in \mathbb{R}^{N \times n}
\]

2. Project state trajectory onto the reduced space

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

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

\[
\hat{x}_{k+1} \approx \sum_{i=1}^{\ell} \hat{A}_i \hat{x}_k + \hat{B}u_k, \quad k = 0, \cdots, K - 1
\]

by minimizing the residual in Euclidean norm

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

OpInf: Learning from projected trajectory

Fitting model to projected states

- We fit model to projected trajectory
  \( \dot{\tilde{X}} = V^T X \)
- Would need \( \tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_K] \) because
  \[
  \sum_{k=0}^{K-1} \left\| \dot{\tilde{x}}_{k+1} - \sum_{i=1}^{\ell} \tilde{A}_i \dot{\tilde{x}}_k - \tilde{B}u_k \right\|_2^2 = 0
  \]
- However, trajectory \( \tilde{X} \) unavailable

Thus, \( \| \hat{f} - \tilde{f} \| \) small critically depends on \( \| \dot{\tilde{X}} - \tilde{X} \| \) being small

- Increase dimension \( n \) of reduced space to decrease \( \| \dot{\tilde{X}} - \tilde{X} \| \)
  \( \Rightarrow \) increases degrees of freedom in OpInf \( \Rightarrow \) ill-conditioned
- Decrease dimension \( n \) to keep number of degrees of freedom low
  \( \Rightarrow \) difference \( \| \dot{\tilde{X}} - \tilde{X} \| \) increases
OpInf: Closure of linear system

Consider autonomous linear system

\[ x_{k+1} = Ax_k, \quad x_0 \in \mathbb{R}^N, \quad k = 0, \ldots, K - 1 \]

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

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

- Split state

\[ x_k = \underbrace{\mathcal{V} \mathcal{V}^T x_k}_{x_k^\parallel} + \underbrace{\mathcal{V}_\perp \mathcal{V}_\perp^T x_k}_{x_k^\perp} \]

Represent system as

\[ x_{k+1}^\parallel = A_{11} x_k^\parallel + A_{12} x_k^\perp \]

\[ x_{k+1}^\perp = A_{21} x_k^\parallel + A_{22} x_k^\perp \]

with operators

\[ A_{11} = \overset{\rightarrow}{\mathcal{V}^T A \mathcal{V}} = \tilde{A}, \quad A_{12} = \mathcal{V}^T A \mathcal{V}_\perp, \quad A_{21} = \mathcal{V}_\perp^T A \mathcal{V}, \quad A_{22} = \mathcal{V}_\perp^T A \mathcal{V}_\perp \]

[Givon, Kupferman, Stuart, 2004], [Chorin, Stinis, 2006] [Parish, Duraisamy, 2017]
OpInf: Closure term as a non-Markovian term

Projected trajectory $\dot{X}$ mixes dynamics in $V$ and $V_\perp$

$$ V^T x_{k+1} = \dot{x}_{k+1} = x_{k+1}^{\parallel} = A_{11} x_k^{\parallel} + A_{12} x_k^{\perp} $$

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

$$ V^T x_{k+1} = x_{k+1}^{\parallel} = A_{11} x_k^{\parallel} + A_{12} x_k^{\perp} $$

$$ = A_{11} x_k^{\parallel} + \sum_{j=1}^{k-1} A_{22}^{k-j-1} A_{21} x_j^{\parallel} + A_{12} A_{22}^{k-1} x_0^{\perp} $$

Non-Markovian (memory) term models unobserved dynamics

![Graph showing the norm of the closure term over time steps]
Outline

- Introduction and motivation
- Operator inference for learning low-dimensional models
- **Sampling Markovian data for recovering reduced models**
- Rigorous and pre-asymptotic error estimators
- Learning time delays to go beyond Markovian models
- Conclusions
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

\[ \mathbf{V}^T \mathbf{x}_{k+1} = \mathbf{x}_{k+1}^\parallel = A_{11} \mathbf{x}_k^\parallel + \sum_{j=1}^{k-1} A_{22}^{k-j-1} A_{21} \mathbf{x}_j^\parallel + A_{12} A_{22}^{k-1} \mathbf{x}_0^\perp \]

Sample Markovian dynamics by setting memory and noise to 0

- Set \( \mathbf{x}_0 \in \mathcal{V} \), 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 $x_0$ onto $\mathcal{V}$
   \[ \bar{x}_0 = \mathbf{V}^T x_0 \]

2. Query high-dim. system for a single time step with $\mathbf{V} \bar{x}_0$
   \[ x_1 = f(\mathbf{V} \bar{x}_0, u_0) \]

3. Re-project to obtain $\bar{x}_1 = \mathbf{V}^T x_1$

4. Query high-dim. system with re-projected initial condition $\mathbf{V} \bar{x}_1$
   \[ x_2 = f(\mathbf{V} \bar{x}_1, u_1) \]

5. Repeat until end of time-stepping loop

Obtain trajectories

\[ \bar{X} = [\bar{x}_0, \ldots, \bar{x}_{K-1}], \quad \bar{Y} = [\bar{x}_1, \ldots, \bar{x}_K], \quad U = [u_0, \ldots, u_{K-1}] \]

ReProj: Operator inference with re-projection

Operator inference with re-projected trajectories

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

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

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

- 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

ReProj: Queryable systems

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

A dynamical system is queryable, if the trajectory \( X = [x_1, \ldots, x_K] \) with \( K \geq 1 \) can be computed for initial condition \( x_0 \in \mathcal{V} \) and feasible input trajectory \( U = [u_1, \ldots, u_K] \).

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
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
Error of reduced models at test data

- Inferring operators from projected data fails in this example
- Recover reduced model from re-projected data
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
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
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
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
Error of reduced models on test parameters

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

• Operator inference for learning low-dimensional models

• Sampling Markovian data for recovering reduced models

• Rigorous and pre-asymptotic error estimators

• Learning time delays to go beyond Markovian models

• Conclusions
Outline

• Introduction and motivation

• Operator inference for learning low-dimensional models

• Sampling Markovian data for recovering reduced models

• Rigorous and pre-asymptotic error estimators

• Learning time delays to go beyond Markovian models

• Conclusions
ErrEst: Error estimation for learned models

Assumptions*: Symmetric asymptotically stable linear system
- If not symmetric, then need to assume $\|A_1\| \leq 1$ (for now...)
- 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 $k$
  \[ r_k = A_1 V \hat{x}_k + B u_k - V \hat{x}_{k+1} \]
- Bound on state error if initial condition in $\text{span}\{V\}$
  \[ \|x_k - V \hat{x}_k\|_2 \leq C_1 \left( \sum_{i=1}^{k-1} \|r_i\|_2 \right) \]
- Offline/online splitting of computing residual norm $\|r_k\|_2$
  \[ \|r_k\|_2^2 = \hat{x}_k^T V^T A_1^T A_1 V \hat{x}_k + u_k B^T B u_k + \hat{x}_{k+1} V^T V \hat{x}_{k+1} \]
  \[ + 2u_k^T B^T A_1 V \hat{x}_k - 2\hat{x}_{k+1}^T A_1 \hat{x}_{k+1} - 2\hat{x}_{k+1}^T B u_k \]

*Note: Assumptions are based on system properties and may need to be modified based on specific application requirements.

M1: $M_1 = A_1 V^T A_1$
M2: $M_2 = B^T B$
M3: $M_3 = B^T A_1 V$
ErrEst: Learning error operators from data

From [Haasdonk, Ohlberger, 2009] have
\[
\|r_k\|^2_2 = \hat{x}_k^T V^T A_1^T A_1 V \hat{x}_k + u_k B^T B u_k + \hat{x}_{k+1} V^T V \hat{x}_{k+1}
\]
\[
+ 2u_k^T B^T A_1 V \hat{x}_k - 2\hat{x}_{k+1}^T A_1 \hat{x}_{k+1} - 2\hat{x}_{k+1} B u_k
\]

Query system at training inputs to compute residual trajectories

\[
R = \begin{bmatrix}
  r_1 & r_2 & \cdots & r_K
\end{bmatrix}
\]

Learn quantities \( M_1, M_2, M_3 \) via operator inference

- Fit error operators \( M_1, M_2, M_3 \) to residual trajectories
- Bound constant \( C_1 \) and constants for output error

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: Convection-diffusion in a pipe

Governed by parabolic PDE

$$\frac{\partial x}{\partial t} = \Delta x - (1, 1) \cdot \nabla x,$$

in $\Omega$

$$x = 0, \quad \Gamma \backslash \{E_i\}_{i=1}^{5}$$

$$\nabla x \cdot n = g_i(t),$$

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
• Introduction and motivation

• Operator inference for learning low-dimensional models

• Sampling Markovian data for recovering reduced models

• Rigorous and pre-asymptotic error estimators

• Learning time delays to go beyond Markovian models

• Conclusions
Outline

- Introduction and motivation
- Operator inference for learning low-dimensional models
- Sampling Markovian data for recovering reduced models
- Rigorous and pre-asymptotic error estimators
- Learning time delays to go beyond Markovian models
- Conclusions
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

\[ \tilde{x}_{k+1} = \sum_{i=1}^{\ell} \tilde{A}_i \tilde{x}_k^i + \tilde{B} u_k \]

However, loose dynamics modeled by non-Markovian terms

\[ \tilde{x}_{k+1} = \sum_{i=1}^{\ell} \tilde{A}_i \tilde{x}_k^i + \tilde{B} u_k + \sum_{i=1}^{k-1} \Delta_i(\tilde{x}_{k-1}, \ldots, \tilde{x}_{k-i+1}, u_k, \ldots, u_{k-i+1}) + 0 \]

Learn unresolved dynamics via approximate non-Markovian terms

\[ \hat{x}_{k+1} = \sum_{i=1}^{\ell} \hat{A}_i \hat{x}_k^i + \hat{B} u_k + \sum_{i=1}^{k-1} \hat{\Delta}_i(\hat{x}_{k-1}, \ldots, \hat{x}_{k-i+1}, u_k, \ldots, u_{k-i+1}) \]

- Parametrization \( \theta_i \in \Theta \) 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
⇒ 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

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

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

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

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

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

- Fit parameter $\theta_1$ of non-Markovian term $\Delta_1^{\theta_1}$ to difference

  $\min_{\theta_1 \in \Theta} \sum_{i=0}^{K-1} \| \tilde{X}_2^{(i)} - \hat{X}_2^{(i)} - \Delta_1^{(\theta_1)}(\tilde{x}_0^{(i)}, u_i) \|_2^2$

- Repeat this $r$ times to learn $\hat{f}_r$ with lag $r$
NonM: Learning non-Markovian terms

Parametrization of non-Markovian terms

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

$$\hat{\Delta}_{i}^{(\theta_i)}(\hat{x}_{k-1}, \ldots, \hat{x}_{k-i+1}, u_k, \ldots, u_{k-i+1}) = D_i \hat{x}_{k-i+1} + E_i u_{k-i+1}$$

- 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
  \[ S \in \{0, 1\}^{N_s \times N} \text{ with } N_s < N \text{ and } \]
  \[ z_k = S x_k \]

- Learn models from trajectory \( Z = [z_0, \ldots, z_{K-1}] \) instead of \( X = [x_0, \ldots, x_{K-1}] \)

- Apply POD (PCA) to \( Z \) to find basis matrix \( V \) 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
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]
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]
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