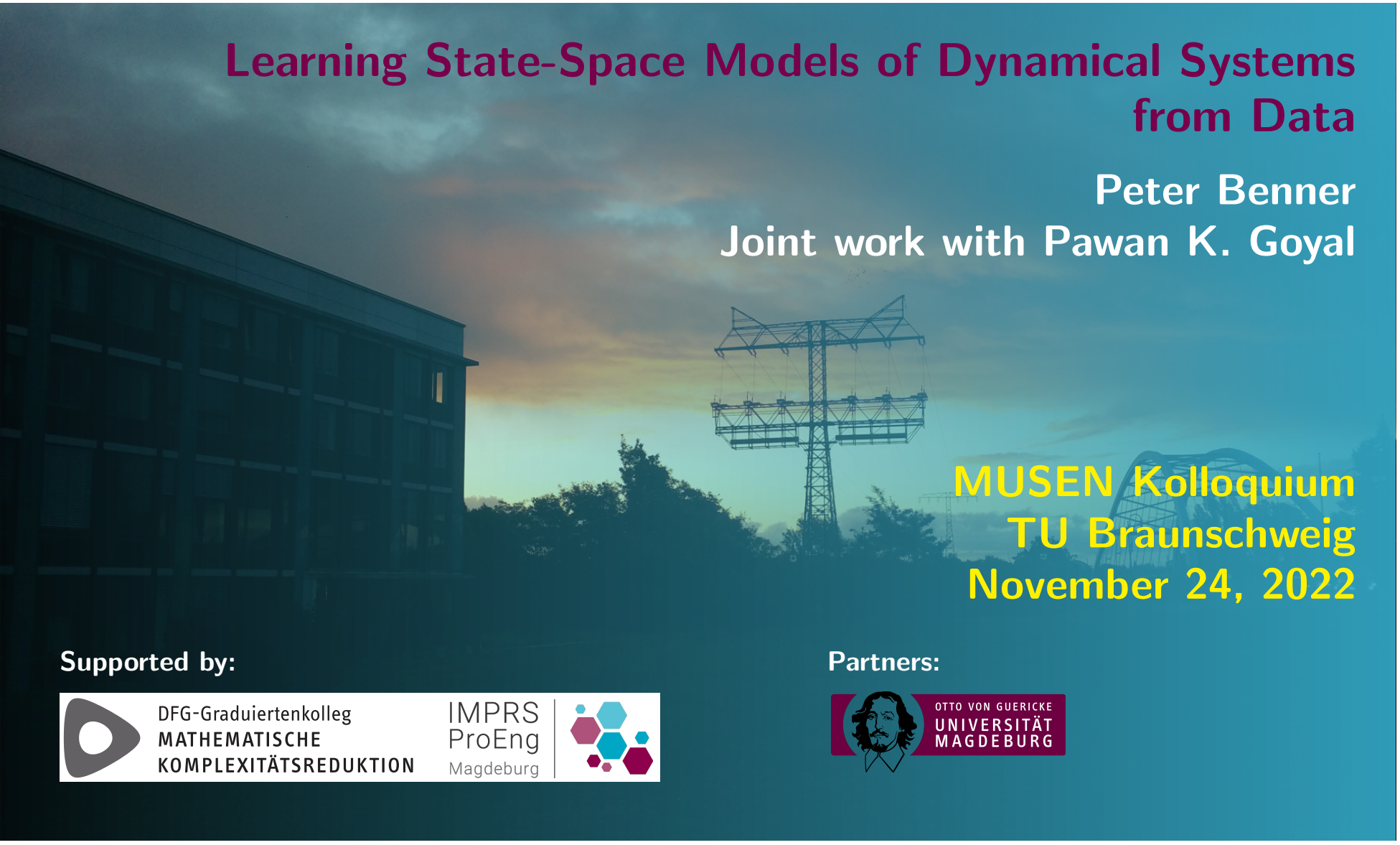


MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Learning State-Space Models of Dynamical Systems from Data

Peter Benner

Joint work with Pawan K. Goyal

MUSEN Kolloquium
TU Braunschweig
November 24, 2022

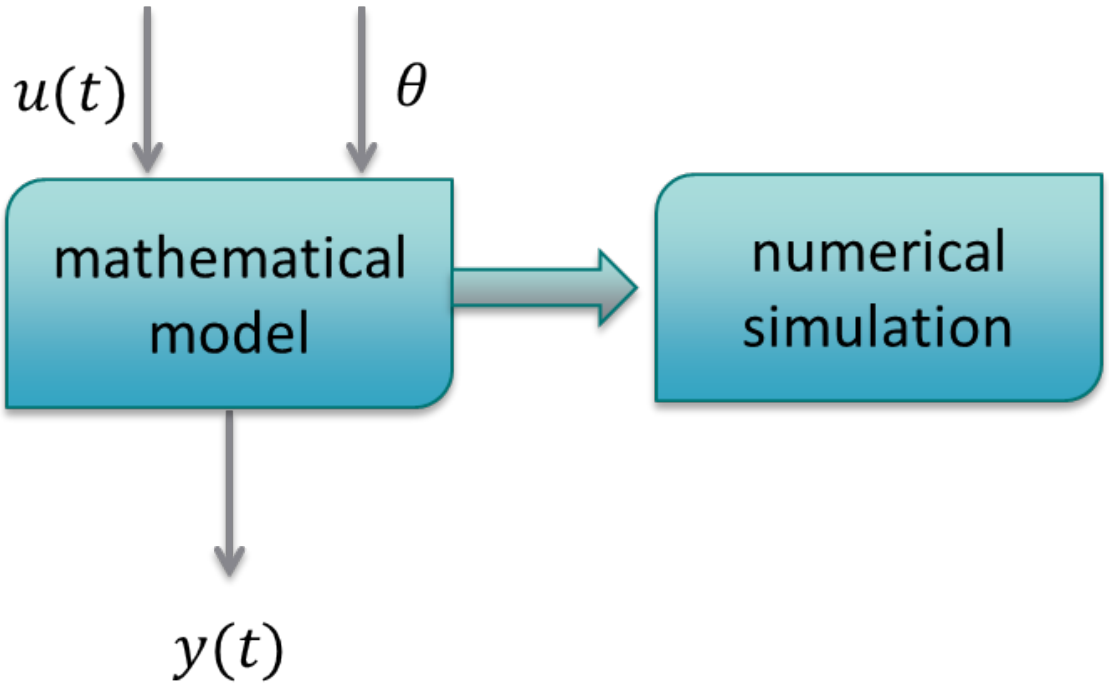
Supported by:

DFG-Graduiertenkolleg
MATHEMATISCHE
KOMPLEXITÄTSREDUKTION

IMPRS
ProEng
Magdeburg

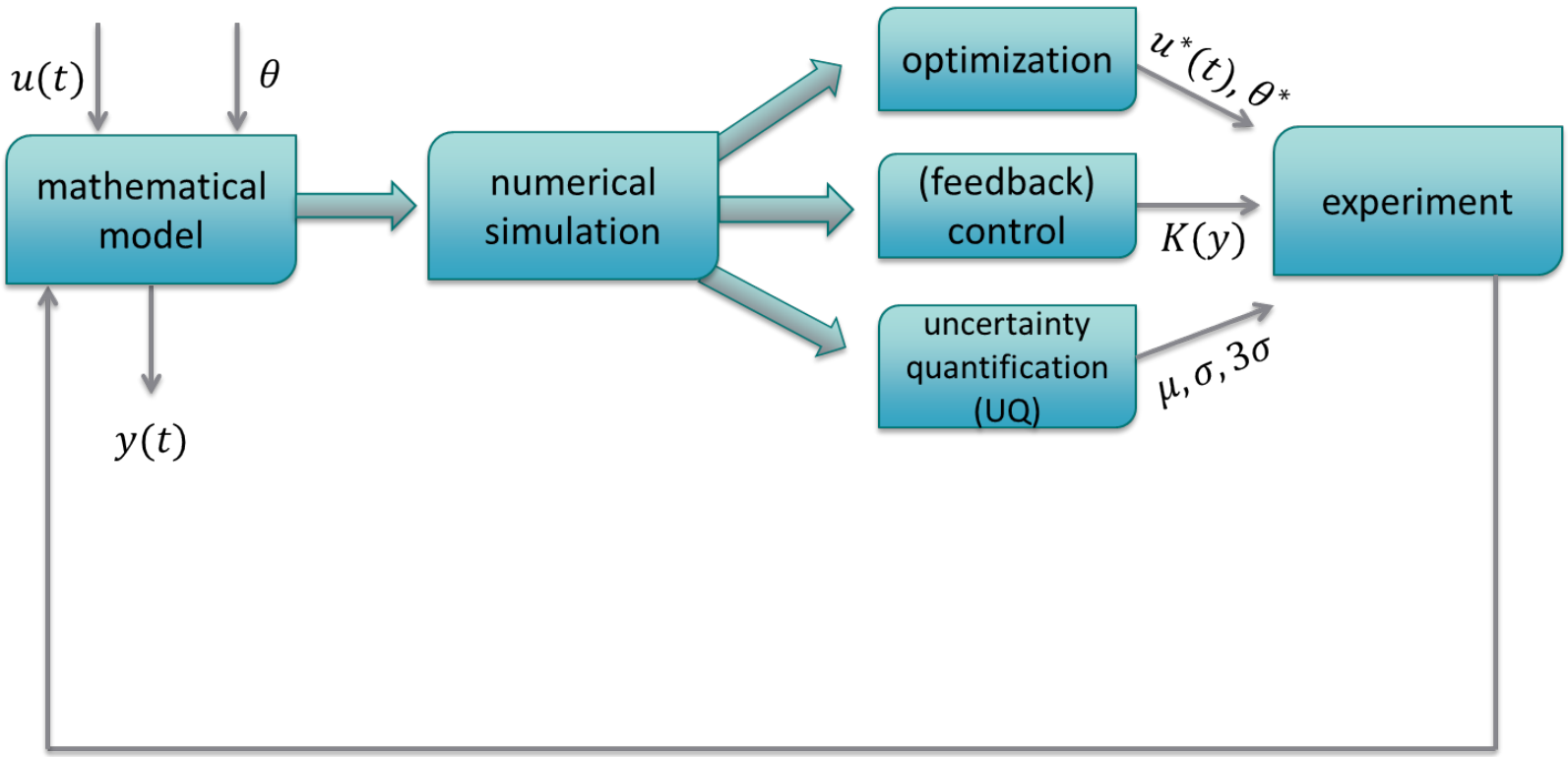
Partners:

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG


$u(t)$
θ
mathematical model
numerical simulation
$y(t)$

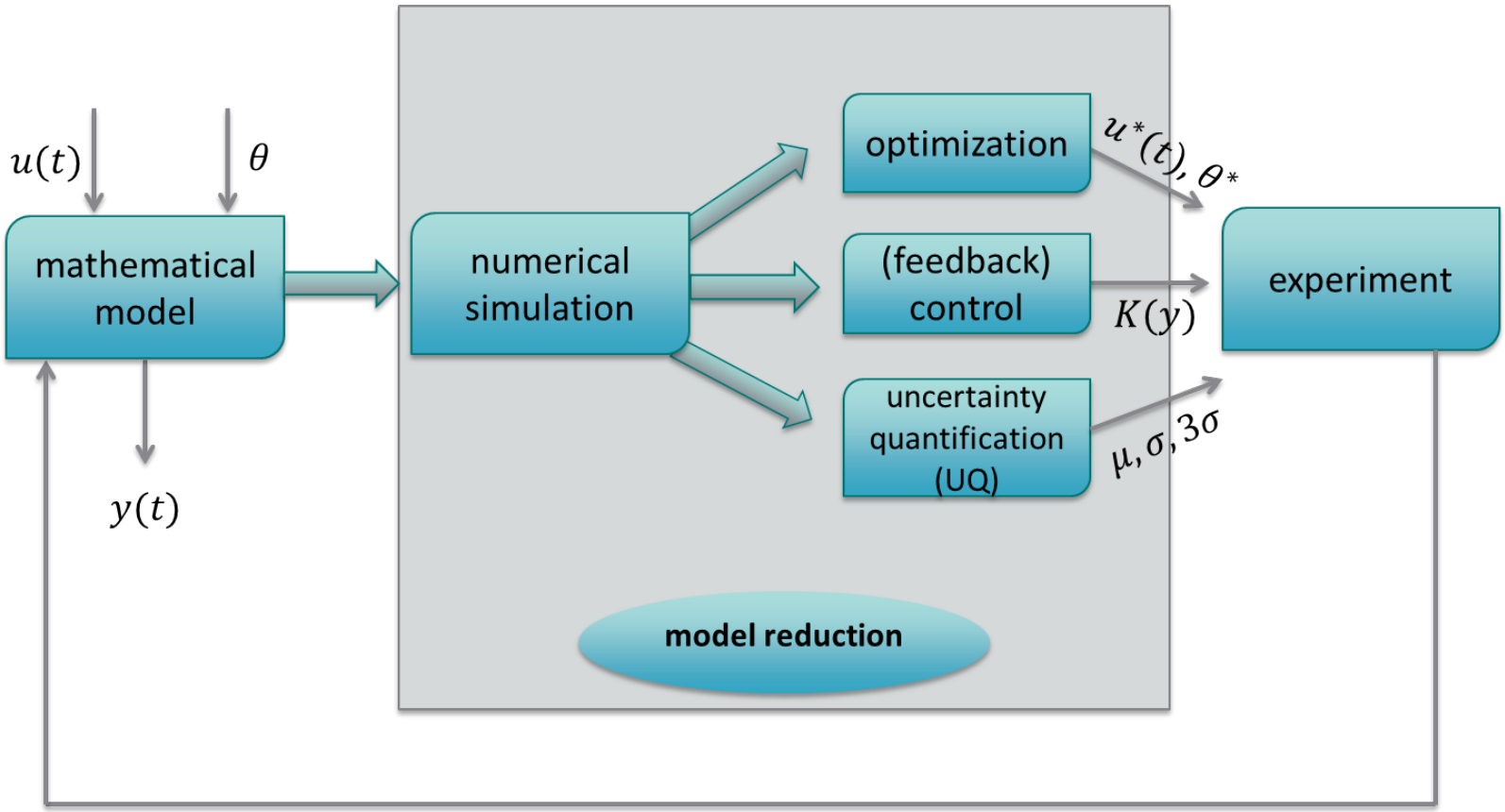
A Process Chain in Computational Sciences and Engineering (CSE)

Optimization, Control, and Uncertainty Quantification



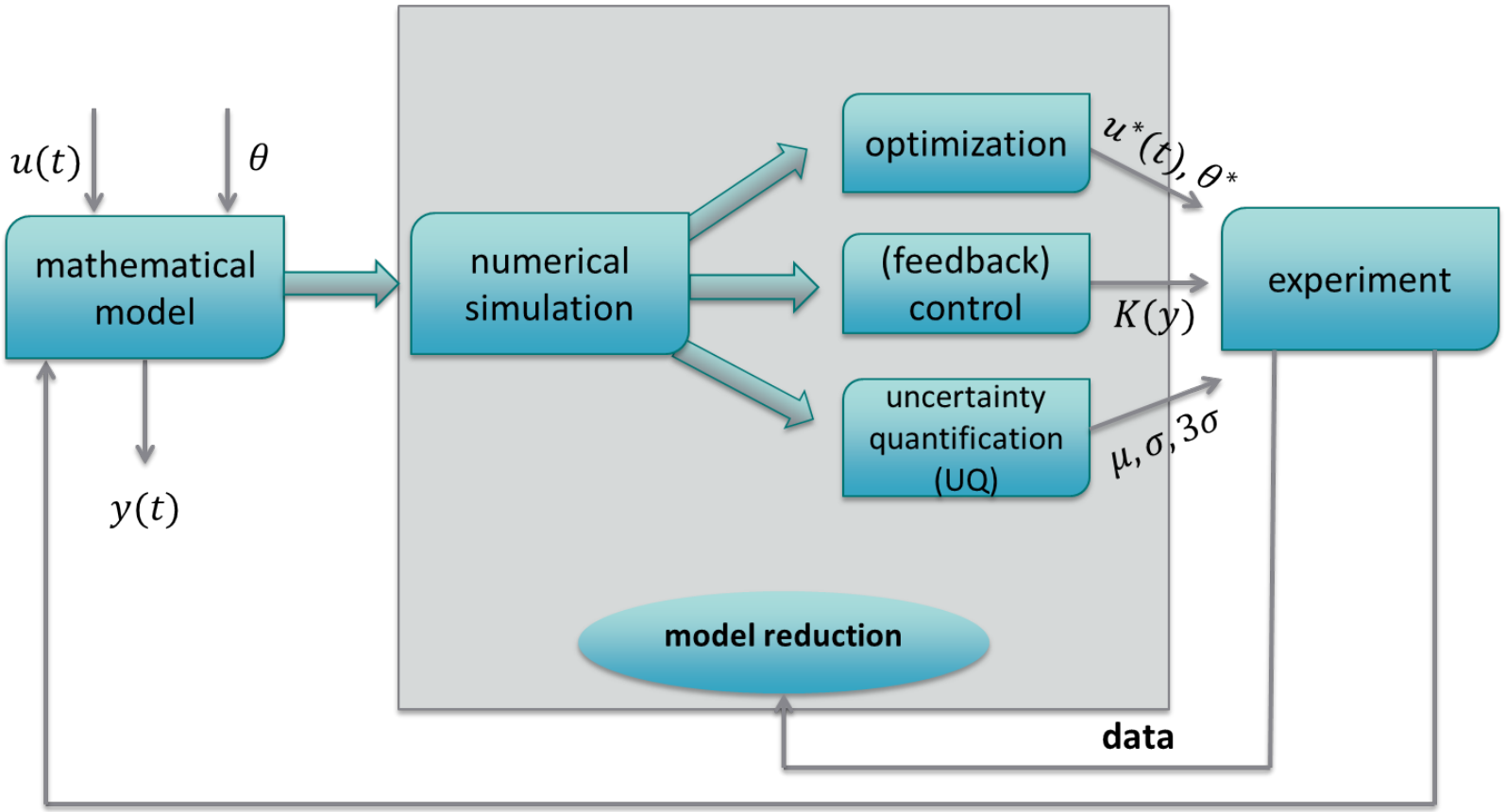
A Process Chain in Computational Sciences and Engineering (CSE)

Model Order Reduction (MOR)



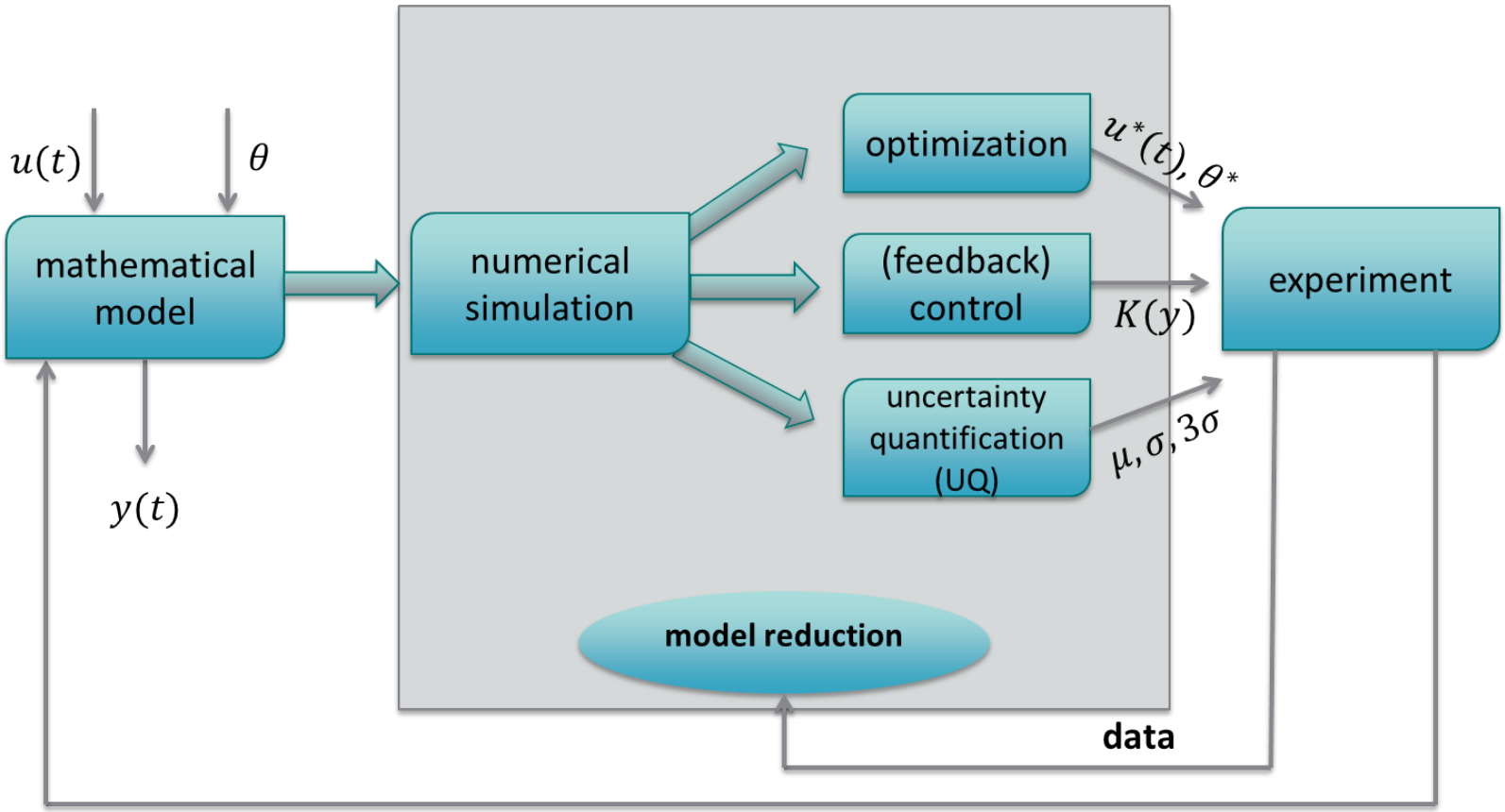
A Process Chain in Computational Sciences and Engineering (CSE)

Data-Driven Sciences



A Process Chain in Computational Sciences and Engineering (CSE)

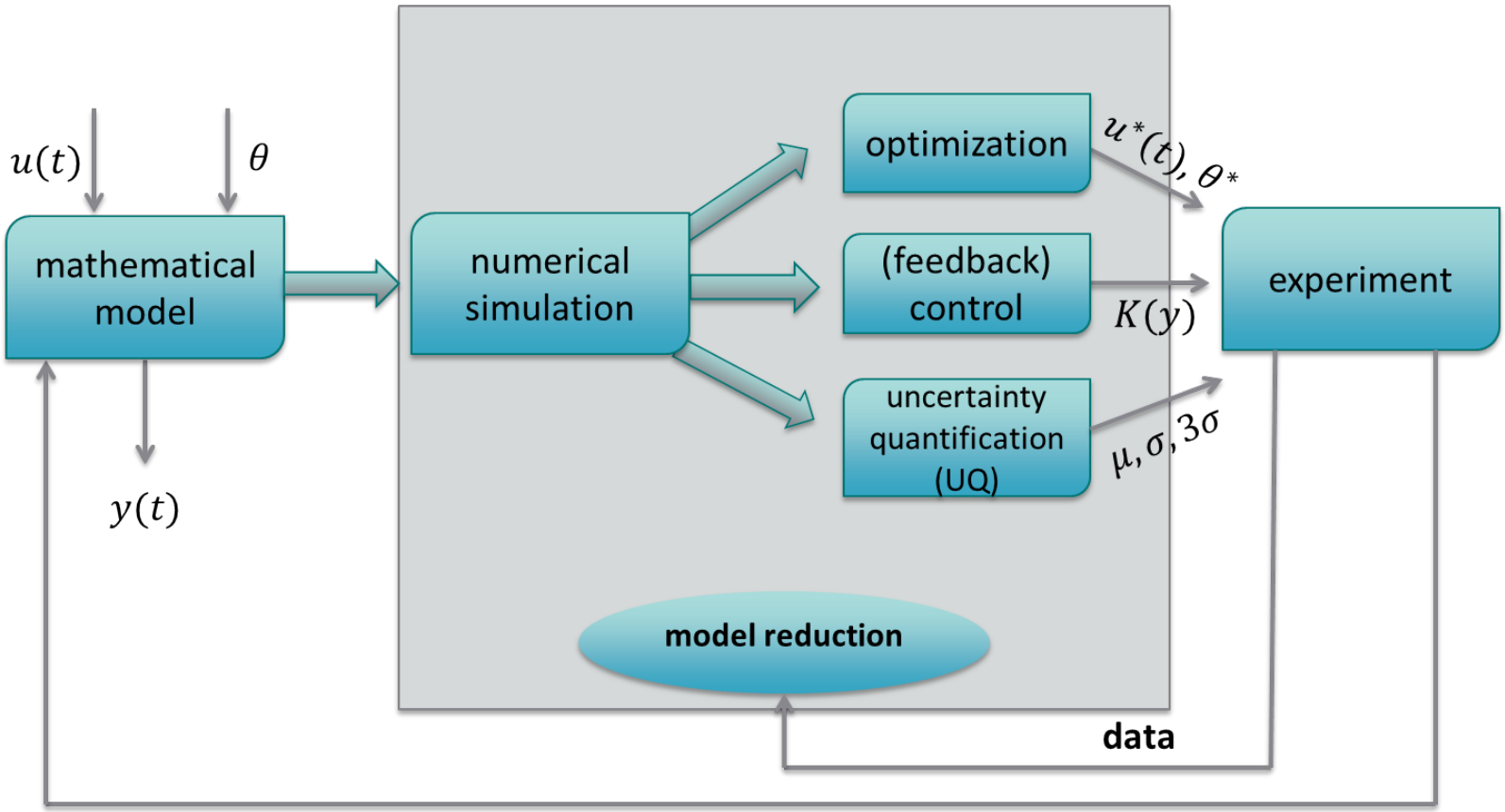
Data-Driven Sciences



Goal: *Use all acquired knowledge about the model during the CSE process chain in the design of the reduced-order model, including experimental data.*

A Process Chain in Computational Sciences and Engineering (CSE)

Data-Driven Sciences

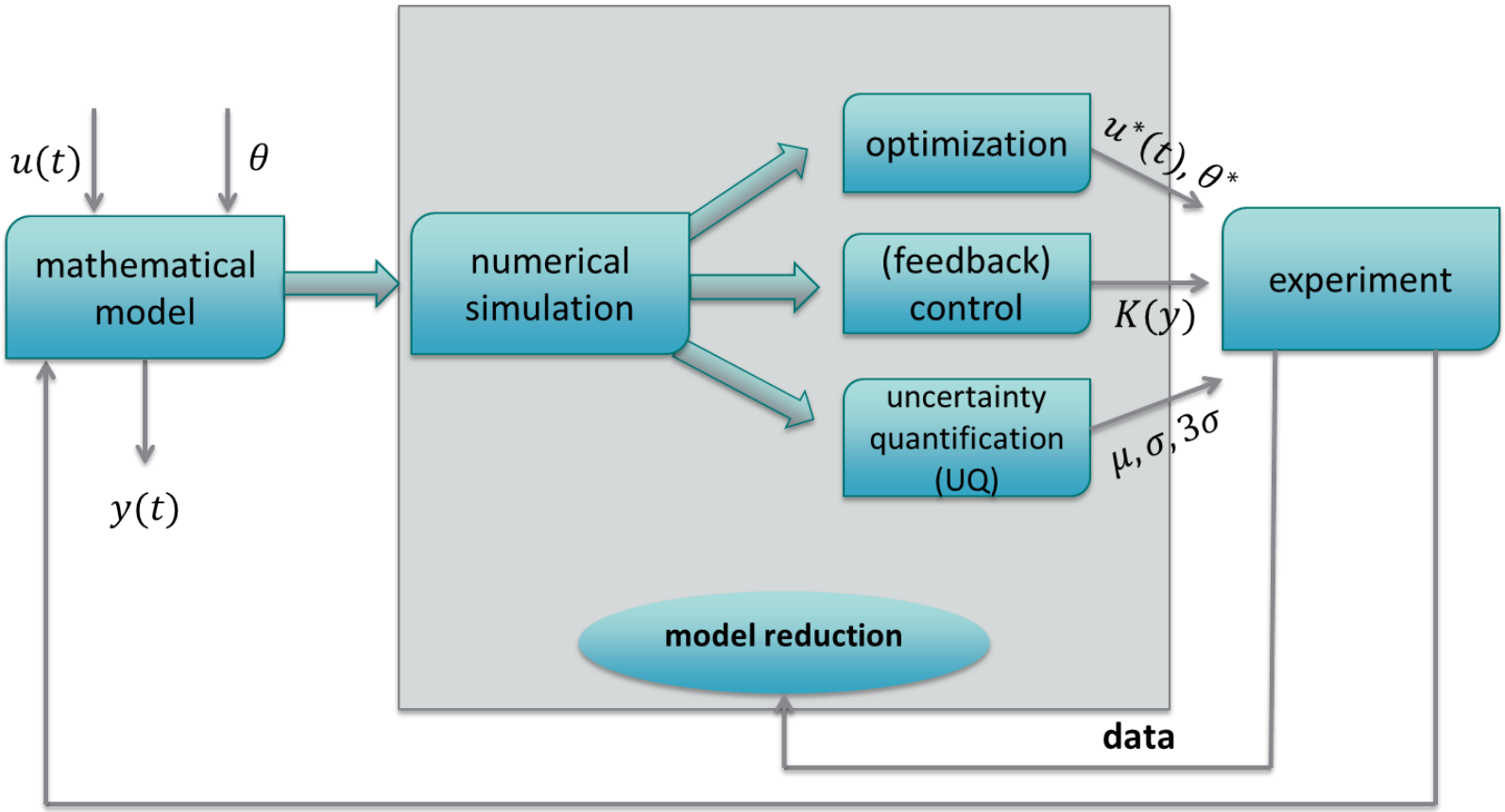


Goal: *Use all acquired knowledge about the model during the CSE process chain in the design of the reduced-order model, including experimental data.*

$\rightsquigarrow$ *Data-driven reduced-order modeling*

A Process Chain in Computational Sciences and Engineering (CSE)

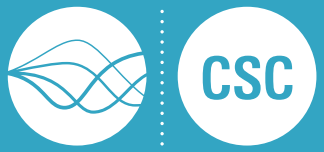
Data-Driven Sciences



Goal: *Use all acquired knowledge about the model during the CSE process chain in the design of the reduced-order model, including experimental data.*

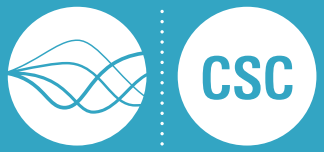
$\rightsquigarrow$ *Data-driven reduced-order modeling*

or in AI/ML idiom: Learning a dynamical (compact) model from data.



Outline

1. Model Order Reduction of Dynamical Systems
2. Data-driven/-enhanced Model Reduction
3. Koopman Operator and DMD
4. Lifting Principle for Dynamical Systems



Outline

Original System

$$\Sigma : \begin{cases} \dot{x}(t) &= f(t, x(t), u(t)), \\ y(t) &= g(t, x(t), u(t)), \end{cases}$$

- states $x(t) \in \mathbb{R}^n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $y(t) \in \mathbb{R}^p$.

Original System

$$\Sigma : \begin{cases} \dot{x}(t) &= f(t, x(t), u(t)), \\ y(t) &= g(t, x(t), u(t)), \end{cases}$$

- states $x(t) \in \mathbb{R}^n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $y(t) \in \mathbb{R}^p$.

Reduced-Order Model (ROM)

$$\widehat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) &= \widehat{f}(t, \hat{x}(t), u(t)), \\ \hat{y}(t) &= \widehat{g}(t, \hat{x}(t), u(t)), \end{cases}$$

- states $\hat{x}(t) \in \mathbb{R}^r$, $r \ll n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $\hat{y}(t) \in \mathbb{R}^p$.

Model Order Reduction of Dynamical Systems

Original System

$$\Sigma : \begin{cases} \dot{x}(t) &= f(t, x(t), u(t)), \\ y(t) &= g(t, x(t), u(t)), \end{cases}$$

- states $x(t) \in \mathbb{R}^n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $y(t) \in \mathbb{R}^p$.

Reduced-Order Model (ROM)

$$\widehat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) &= \widehat{f}(t, \hat{x}(t), u(t)), \\ \hat{y}(t) &= \widehat{g}(t, \hat{x}(t), u(t)), \end{cases}$$

- states $\hat{x}(t) \in \mathbb{R}^r$, $r \ll n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $\hat{y}(t) \in \mathbb{R}^p$.

Goals:

$\|y - \hat{y}\| < \text{tolerance} \cdot \|u\|$ for all admissible input signals.

Model Order Reduction of Dynamical Systems

Original System

$$\Sigma : \begin{cases} \dot{x}(t) &= f(t, x(t), u(t)), \\ y(t) &= g(t, x(t), u(t)), \end{cases}$$

- states $x(t) \in \mathbb{R}^n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $y(t) \in \mathbb{R}^p$.

Reduced-Order Model (ROM)

$$\widehat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) &= \widehat{f}(t, \hat{x}(t), u(t)), \\ \hat{y}(t) &= \widehat{g}(t, \hat{x}(t), u(t)), \end{cases}$$

- states $\hat{x}(t) \in \mathbb{R}^r$, $r \ll n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $\hat{y}(t) \in \mathbb{R}^p$.

Goals:

$\|y - \hat{y}\| < \text{tolerance} \cdot \|u\|$ for all admissible input signals.

Secondary goal: reconstruct approximation of x from $\hat{x}$.

Original System

$$\Sigma : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t) + Du(t). \end{cases}$$

- states $x(t) \in \mathbb{R}^n$,
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $y(t) \in \mathbb{R}^p$.

Reduced-Order Model (ROM)

$$\widehat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) = \widehat{A}\hat{x}(t) + \widehat{B}u(t), \\ \hat{y}(t) = \widehat{C}\hat{x}(t) + \widehat{D}u(t). \end{cases}$$

- states $\hat{x}(t) \in \mathbb{R}^r$, $r \ll n$
- inputs $u(t) \in \mathbb{R}^m$,
- outputs $\hat{y}(t) \in \mathbb{R}^p$.

Goals:

$\|y - \hat{y}\| <$ tolerance $\cdot \|u\|$ for all admissible input signals.

Secondary goal: reconstruct approximation of x from $\hat{x}$.

Model Order Reduction of Linear Systems

Model Reduction Schematically

$$E\,\dot{x}(t) = A\,x(t) + B\,u(t)$$

$$y(t) = C\,x(t) + D\,u(t)$$

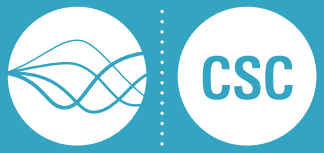
MOR

$$\hat{E}\,\dot{\hat{x}}(t) = \hat{A}\,\hat{x}(t) + \hat{B}\,u(t)$$

$$\hat{y}(t) = \hat{C}\,\hat{x}(t) + \hat{D}\,u(t)$$

- $E, A \in \mathbb{R}^{n\times n}$
- $B \in \mathbb{R}^{n\times m}$
- $C \in \mathbb{R}^{p\times n}$
- $D \in \mathbb{R}^{p\times m}$

- $\hat{E}, \hat{A} \in \mathbb{R}^{r\times r}$
- $\hat{B} \in \mathbb{R}^{r\times m}$
- $\hat{C} \in \mathbb{R}^{p\times r}$
- $\hat{D} \in \mathbb{R}^{p\times m}$



MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

Then, with $\hat{x} = W^T x$, we obtain $x \approx V\hat{x} = VW^T x =: \tilde{x}$ so that

$$||x - \tilde{x}|| = ||x - V\hat{x}||.$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

Then, with $\hat{x} = W^T x$, we obtain $x \approx V\hat{x} = VW^T x =: \tilde{x}$ so that

$$||x - \tilde{x}|| = ||x - V\hat{x}||.$$

For linear systems ($E = I_n$ for simplicity), the reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

For linear systems ($E = I_n$ for simplicity), the reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

Important observation:

- The state equation residual satisfies $\dot{\tilde{x}} - A\tilde{x} - Bu \perp \mathcal{W}$, since

$$W^T\left(\dot{\tilde{x}} - A\tilde{x} - Bu\right) \quad = \quad W^T\left(VW^T\dot{x} - AVW^T x - Bu\right)$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

For linear systems ($E = I_n$ for simplicity), the reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

Important observation:

- The state equation residual satisfies $\dot{\tilde{x}} - A\tilde{x} - Bu \perp \mathcal{W}$, since

$$\begin{aligned} W^T\left(\dot{\tilde{x}} - A\tilde{x} - Bu\right) &= W^T\left(VW^T\dot{x} - AVW^T x - Bu\right) \\ &= \underbrace{W^T\dot{x}}_{\dot{\hat{x}}} - \underbrace{W^T A V}_{=\hat{A}}\underbrace{W^T x}_{=\hat{x}} - \underbrace{W^T B}_{=\hat{B}}u \end{aligned}$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

For linear systems ($E = I_n$ for simplicity), the reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

Important observation:

- The state equation residual satisfies $\dot{\tilde{x}} - A\tilde{x} - Bu \perp \mathcal{W}$, since

$$\begin{aligned} W^T\left(\dot{\tilde{x}} - A\tilde{x} - Bu\right) &= W^T\left(VW^T\dot{x} - AVW^T x - Bu\right) \\ &= \underbrace{W^T\dot{x}}_{\dot{\hat{x}}} - \underbrace{W^T A V}_{=\hat{A}}\underbrace{W^T x}_{=\hat{x}} - \underbrace{W^T B}_{=\hat{B}}u \\ &= \dot{\hat{x}} - \hat{A}\hat{x} - \hat{B}u = 0. \end{aligned}$$

MOR Methods Based on Projection

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r \qquad (\text{or } W^T E V = I_r).$$

For linear systems ($E = I_n$ for simplicity), the reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

Extends to nonlinear systems with some effort:

$$\begin{aligned} \dot{\hat{x}} &= W^T f(t, V\hat{x}, u), \\ \hat{y} &= g(t, V\hat{x}, u). \end{aligned}$$

Needs hyperreduction if the cost for evaluation of the functions $W^T f, g$ is not reduced!

MOR Methods Based on Projection

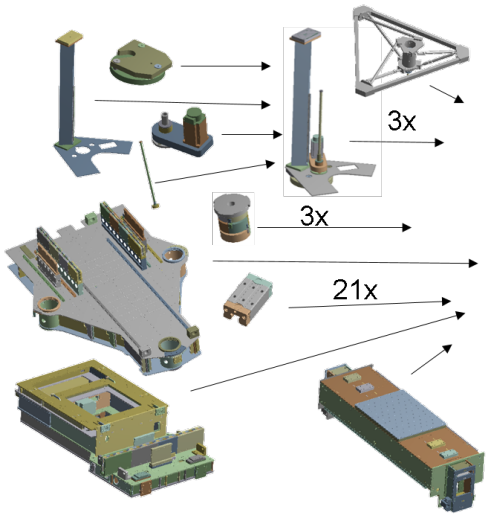
Classes of Projection-based MOR Methods

1. Modal Truncation
2. Rational Interpolation / Moment Matching
   (Padé-Approximation and (rational) Krylov Subspace Methods)
3. Balanced Truncation
4. Proper Orthogonal Decomposition (POD) / Principal Component Analysis (PCA)
5. Reduced Basis Method
6. ...

MOR Methods Based on Projection

Example: Thermal model of experimental machine tool MAX

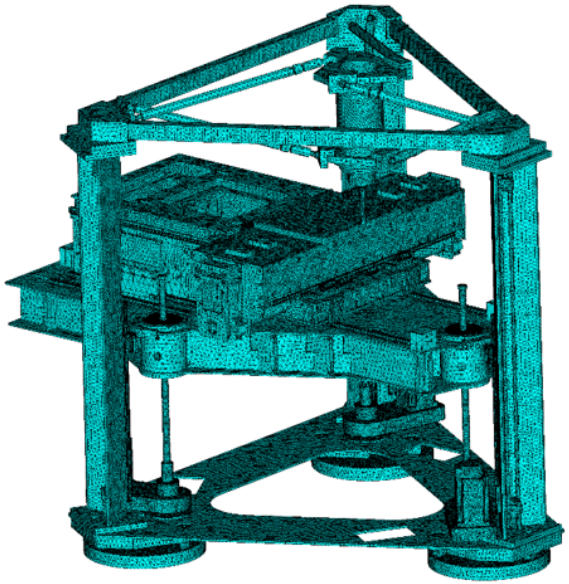
50 subassemblies



CAD model

$\overset{FEM}{\rightsquigarrow}$

FE-Model: 1.2M DOFs



MAX: Results considering an inhomogeneous initial condition $T_0 \neq 0$

Results by Julia Vettermann (MilT/TU Chemnitz)

FE-coupled

method	red. order tol 10^{-3}	t_{red}
2phase	196	6.5h
BTX0	174	4.5h

output-coupled

method	red. order tol 10^{-3}	t_{red}
2phase	3005	2h
BTX0	2515	1.8h

MAX: Results considering an inhomogeneous initial condition $T_0 \neq 0$

Results by Julia Vettermann (MilT/TU Chemnitz)

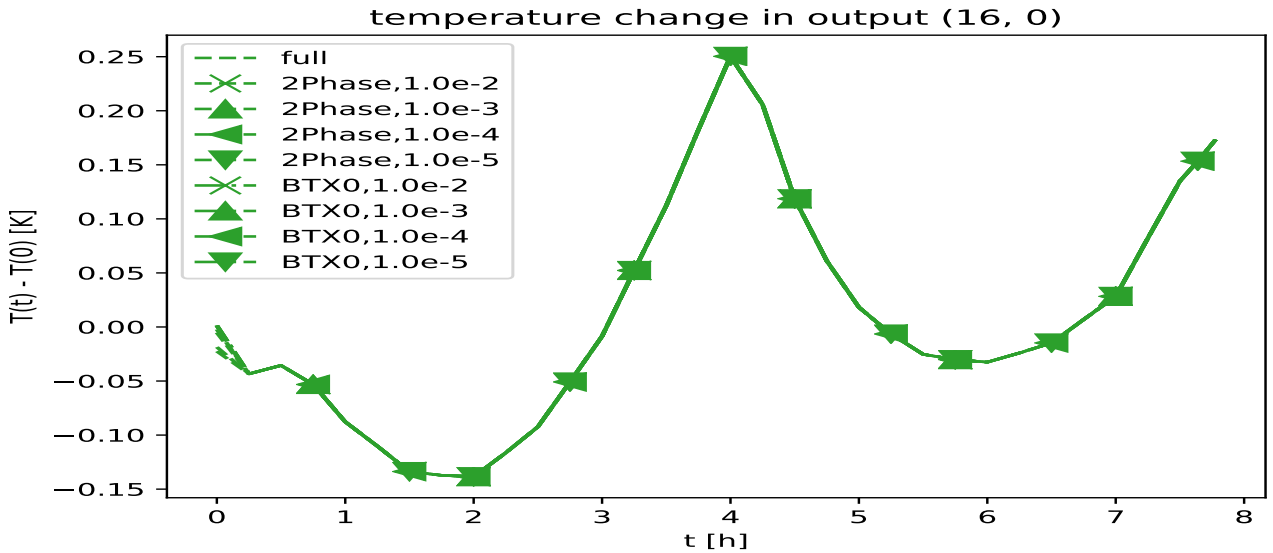
FE-coupled

method	red. order tol 10^{-3}	t_{red}
2phase	196	6.5h
BTX0	174	4.5h

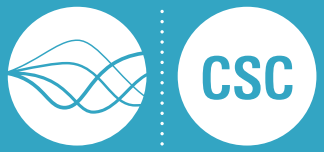
output-coupled

method	red. order tol 10^{-3}	t_{red}
2phase	3005	2h
BTX0	2515	1.8h

→ *Required storage for reduced matrices just 1MB!*
→ *Simulation speed-up factors range from* ≈ 8–$2,000$.



Vettermann, J., Sauerzapf, S., Naumann, A., Beitelschmidt, M., Herzog, R., Benner, P., Saak, J. (2021): Model order reduction methods for coupled machine tool models. MM Science Journal, pp. 4652–4659.



What about the Data?

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r.$$

The reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

What about the Data?

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r.$$

The reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

We need the matrices A, B, C, D to compute the reduced-order model!

What about the Data?

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r.$$

The reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

We need the matrices A, B, C, D to compute the reduced-order model!

Using proprietary simulation software, we would need to **intrude** the software to get the matrices

$\rightsquigarrow$ **intrusive MOR**

$=$ *learning (compact, surrogate) models from (full, detailed) models.*

This is often impossible!

What about the Data?

Assumption: trajectory $x(t;u)$ is contained in low-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$.
Thus, use Galerkin or Petrov-Galerkin-type projection of state-space onto $\mathcal{V}$ (trial space) along complementary subspace $\mathcal{W}$ (test space), where

$$\operatorname{range}(V) = \mathcal{V}, \quad \operatorname{range}(W) = \mathcal{W}, \quad W^T V = I_r.$$

The reduced-order model is

$$\hat{x} = W^T x, \quad \hat{A} := W^T A V, \quad \hat{B} := W^T B, \quad \hat{C} := CV, \quad (\hat{D} := D).$$

We need the matrices A, B, C, D to compute the reduced-order model!

Using proprietary simulation software, we would need to **intrude** the software to get the matrices

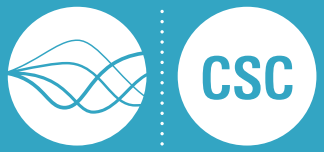
$\rightsquigarrow$ **intrusive MOR**

$=$ *learning (compact, surrogate) models from (full, detailed) models.*

This is often impossible!

$\rightsquigarrow$ **non-intrusive MOR**

$=$ *LEARNING (compact, surrogate) MODELS FROM DATA!*



Outline

1. Model Order Reduction of Dynamical Systems

2. Data-driven/-enhanced Model Reduction
   - A few Remarks on System Identification and DNNs
   - DMD in a Nutshell
   - Operator Inference

3. Koopman Operator and DMD

4. Lifting Principle for Dynamical Systems

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Some methods:

- System identification (incl. ERA, N4SID, MOESP): frequency and time domain
  [Ho/Kalman 1966; Ljung 1987/1999; Van Overschee/De Moor 1994; Verhaegen 1994; De Wilde, Eykhoff, Moonen, Sima, ...]

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Some methods:

- System identification (incl. ERA, N4SID, MOESP): frequency and time domain
  [Ho/Kalman 1966; Ljung 1987/1999; Van Overschee/De Moor 1994; Verhaegen 1994; De Wilde, Eykhoff, Moonen, Sima, ...]
- Neural networks: time domain [Narendra/Parthasarathy 1990; Lee/Carlberg 2019; ...]

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Some methods:

- System identification (incl. ERA, N4SID, MOESP): frequency and time domain
  [Ho/Kalman 1966; Ljung 1987/1999; Van Overschee/De Moor 1994; Verhaegen 1994; De Wilde, Eykhoff, Moonen, Sima, ...]
- Neural networks: time domain [Narendra/Parthasarathy 1990; Lee/Carlberg 2019; ...]
- Loewner interpolation: frequency and time domain
  [Antoulas/Anderson 1986; Mayo/Antoulas 2007; Gosea, Gugercin, Ionita, Lefteriu, Peherstorfer, ...]

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Some methods:

- System identification (incl. ERA, N4SID, MOESP): frequency and time domain
  [Ho/Kalman 1966; Ljung 1987/1999; Van Overschee/De Moor 1994; Verhaegen 1994; De Wilde, Eykhoff, Moonen, Sima, ...]
- Neural networks: time domain [Narendra/Parthasarathy 1990; Lee/Carlberg 2019; ...]
- Loewner interpolation: frequency and time domain
  [Antoulas/Anderson 1986; Mayo/Antoulas 2007; Gosea, Gugercin, Ionita, Lefteriu, Peherstorfer, ...]
- Koopman/Dynamic Mode Decomposition (DMD): time domain
  [Mezič 2005; Schmid 2008; Brunton, Kevrekidis, Kutz, Rowley, Noé, Nüske, Schütte, Peitz, ...],
  for control systems [Kaiser/Kutz/Brunton 2017, B./Himpe/Mitchell 2018]

Now assume we are only given an oracle, allowing us to compute y (including cases with $y \equiv x$), given $u(t)$ or $U(s)$:

Black box Σ: the only information we can get is either

- time domain data / times series: $u_k \approx u(t_k)$ and $x_k \approx x(t_k)$ or $y_k \approx y(t_k)$, or
- frequency domain data / measurements: $U_k \approx U(\jmath\omega_k)$ and $X_k \approx X(\jmath\omega_k)$ or $Y_k \approx Y(\jmath\omega_k)$.

Some methods:

- System identification (incl. ERA, N4SID, MOESP): frequency and time domain
  [Ho/Kalman 1966; Ljung 1987/1999; Van Overschee/De Moor 1994; Verhaegen 1994; De Wilde, Eykhoff, Moonen, Sima, ...]
- Neural networks: time domain [Narendra/Parthasarathy 1990; Lee/Carlberg 2019; ...]
- Loewner interpolation: frequency and time domain
  [Antoulas/Anderson 1986; Mayo/Antoulas 2007; Gosea, Gugercin, Ionita, Lefteriu, Peherstorfer, ...]
- Koopman/Dynamic Mode Decomposition (DMD): time domain
  [Mezič 2005; Schmid 2008; Brunton, Kevrekidis, Kutz, Rowley, Noé, Nüske, Schütte, Peitz, ...],
  for control systems [Kaiser/Kutz/Brunton 2017, B./Himpe/Mitchell 2018]
- Operator inference (OpInf): time domain [Peherstorfer/Willcox 2016; Kramer, Qian, B., Goyal,...]

A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.

A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

  from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.
- Early survey already 1971: Aström/Eykhoff, AUTOMATICA 7(2):123–162.

A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

  from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.

- Early survey already 1971: Aström/Eykhoff, Automatica 7(2):123–162.
- Popular methods are
  1. MOESP — Multivariable Output Error State-SPace [Verhaegen/Dewilde 1992],
  2. N4SID — Numerical algorithm for Subspace State Space System IDentification [Van Overschee/De Moor 1994].

A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

  from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.

- Early survey already 1971: Aström/Eykhoff, Automatica 7(2):123–162.
- Popular methods are
  1. MOESP — Multivariable Output Error State-SPace [Verhaegen/Dewilde 1992],
  2. N4SID — Numerical algorithm for Subspace State Space System IDentification [Van Overschee/De Moor 1994].
- Both are based on decompositions of certain block-Hankel matrices built from the input-output data and are available in standard software packages like the MATLAB System Identification Toolbox and SLICOT.

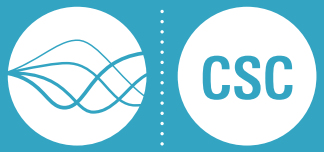
A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

  from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.

- Early survey already 1971: Aström/Eykhoff, Automatica 7(2):123–162.
- Popular methods are
  1. MOESP — Multivariable Output Error State-SPace [Verhaegen/Dewilde 1992],
  2. N4SID — Numerical algorithm for Subspace State Space System IDentification [Van Overschee/De Moor 1994].
- Both are based on decompositions of certain block-Hankel matrices built from the input-output data and are available in standard software packages like the MATLAB System Identification Toolbox and SLICOT.
- Continuous-time system can be identified, e.g., by "inverse" Euler method.



A few Remarks on System Identification and DNNs

- System identification tries to infer discrete linear time-invariant (LTI) systems

$$x_{k+1} = Ax_k + Bu_k + Kw_k,$$
$$y_k = Cx_k + Du_k + v_k.$$

  from input-output data, given as time series $(u_0, y_0), (u_1, y_1), \ldots, (u_K, y_K)$, where v_k, w_k are uncorrelated Gaussian white noise processes.

- Early survey already 1971: Aström/Eykhoff, Automatica 7(2):123–162.
- Popular methods are
  1. MOESP — Multivariable Output Error State-SPace [Verhaegen/Dewilde 1992],
  2. N4SID — Numerical algorithm for Subspace State Space System IDentification [Van Overschee/De Moor 1994].
- Both are based on decompositions of certain block-Hankel matrices built from the input-output data and are available in standard software packages like the MATLAB System Identification Toolbox and SLICOT.
- Continuous-time system can be identified, e.g., by "inverse" Euler method.
- Many extensions to nonlinear systems, imposing certain structural assumptions, including artificial neural networks...

A few Remarks on System Identification and DNNs

A paper from 1990...

4 IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 1, NO. 1, MARCH 1990

Identification and Control of Dynamical Systems Using Neural Networks

KUMPATI S. NARENDRA FELLOW, IEEE, AND KANNAN PARTHASARATHY

Abstract—**The paper demonstrates that neural networks can be used effectively for the identification and control of nonlinear dynamical systems. The emphasis of the paper is on models for both identification and control. Static and dynamic back-propagation methods for the adjustment of parameters are discussed. In the models that are introduced, multilayer and recurrent networks are interconnected in novel configurations and hence there is a real need to study them in a unified fashion. Simulation results reveal that the identification and adaptive control schemes suggested are practically feasible. Basic concepts and definitions are introduced throughout the paper, and theoretical questions which have to be addressed are also described.**

are well known for such systems [1]. In this paper our interest is in the identification and control of nonlinear dynamic plants using neural networks. Since very few results exist in nonlinear systems theory which can be directly applied, considerable care has to be exercised in the statement of the problems, the choice of the identifier and controller structures, as well as the generation of adaptive laws for the adjustment of the parameters.

Two classes of neural networks which have received considerable attention in the area of artificial neural net-

Narendra, K.S., Parthasarathy, K. (1990): Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1):4–27.

A few Remarks on System Identification and DNNs

A paper from 1990...

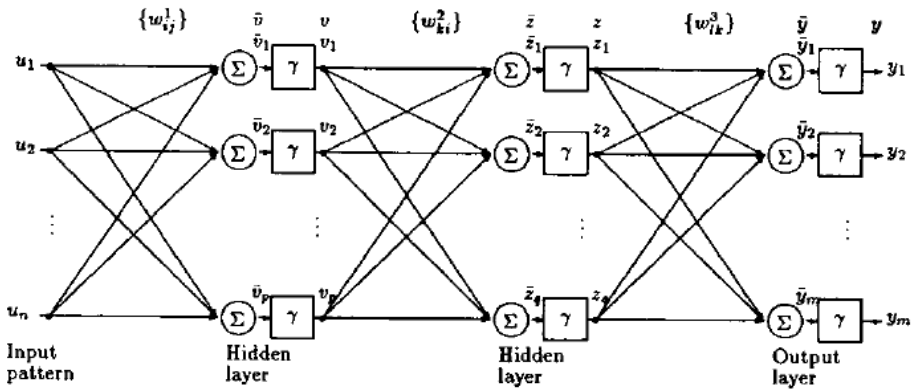


Fig. 2. A three layer neural network.

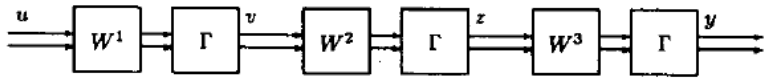
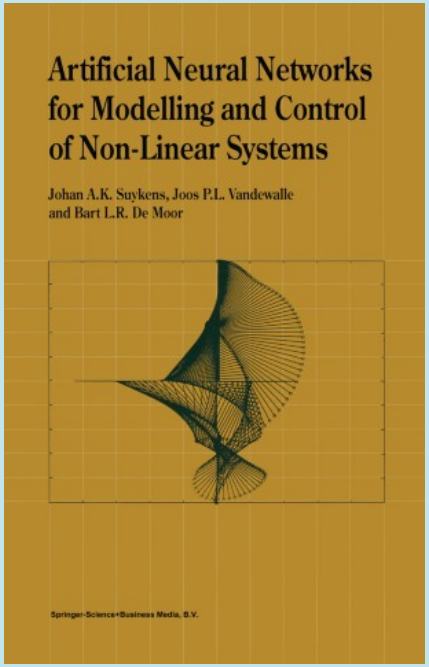


Fig. 3. A block diagram representation of a three layer network.

Narendra, K.S., Parthasarathy, K. (1990): Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1):4–27.

A few Remarks on System Identification and DNNs

A book from 1996...



Narendra, K.S., Parthasarathy, K. (1990): Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1):4–27.

Suykens, J.A.K., Vandewalle, J.P.L., de Moor, B.L. (1996): Artificial Neural Networks for Modelling and Control of Non-Linear Systems. Springer US.

DMD in a Nutshell

Basic Framework

Given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!), and find "best possible" A_* such that

$$x_{k+1} \approx A_* x_k.$$

Given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!), and find "best possible" A_* such that

$$x_{k+1} \approx A_* x_k.$$

Motivation: Koopman theory

- $\exists$ a linear, infinite-dimensional operator describing the evolution of $f(x(\cdot))$ in an appropriate function space setting.
- Can be considered as lifting of a finite-dimensional, nonlinear problem to a infinite-dimensional, linear problem.

DMD in a Nutshell

Basic Framework

Given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!), and find "best possible" A_* such that

$$x_{k+1} \approx A_* x_k.$$

Motivation: Koopman theory

- $\exists$ a linear, infinite-dimensional operator describing the evolution of $f(x(\cdot))$ in an appropriate function space setting.
- Can be considered as lifting of a finite-dimensional, nonlinear problem to a infinite-dimensional, linear problem.

Basic DMD Algorithm

Set $X_0 := [\, x_0, x_1, \ldots, x_{K-1} \,] \in \mathbb{R}^{n \times K}$, $X_1 := [\, x_1, x_2, \ldots, x_K \,] \in \mathbb{R}^{n \times K}$ and note that $X_1 = AX_0$ is desired $\rightsquigarrow$ over-/underdetermined linear system, solved by linear least-squares problem (regression):

$$A_* := \arg\min_{A \in \mathbb{R}^{n \times n}} \|X_1 - AX_0\|_F + \beta \|A\|_q$$

with a potential regularization term choosing $\beta > 0$, $q = 0, 1, 2$.

Computation usually via singular value decomposition (SVD), many variants.

DMD in a Nutshell

DMD with Inputs and Outputs

Given a smooth control system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n, \qquad y(t) = g(x(t), u(t)),$$

with control $u(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}^p$.

DMD in a Nutshell

DMD with Inputs and Outputs

Given a smooth control system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n, \qquad\qquad y(t) = g(x(t), u(t)),$$

with control $u(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}^p$.

Take state, control, and output snapshots

$$x_k := x(t_k), \quad u_k := u(t_k), \quad y_k := y(t_k), \qquad k = 0, 1, \ldots, K$$

(using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!), and find "best possible" discrete-time LTI system such that

$$x_{k+1} \approx A_* x_k + B_* u_k, \qquad y_k \approx C_* x_k + D_* u_k.$$

Given a smooth control system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n, \qquad y(t) = g(x(t), u(t)),$$

with control $u(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}^p$.

Take state, control, and output snapshots

$$x_k := x(t_k), \quad u_k := u(t_k), \quad y_k := y(t_k), \qquad k = 0, 1, \ldots, K$$

(using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!), and find "best possible" discrete-time LTI system such that

$$x_{k+1} \approx A_* x_k + B_* u_k, \qquad y_k \approx C_* x_k + D_* u_k.$$

Basic ioDMD Algorithm ($\equiv$ N4SID)

Let $\mathbb{S} := \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times m} \times \mathbb{R}^{p\times n} \times \mathbb{R}^{p\times m}$. Set X_0, X_1 as before and

$$U_0 := [\, u_0, u_1, \ldots, u_{K-1} \,] \in \mathbb{R}^{m\times K}, \qquad Y_0 := [\, y_0, y_1, \ldots, y_{K-1} \,] \in \mathbb{R}^{p\times K}.$$

Solve the linear least-squares problem (regression):

$$(A_*, B_*, C_*, D_*) := \arg\min_{(A,B,C,D)\in\mathbb{S}} \left\| \begin{bmatrix} X_1 \\ Y_0 \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} \right\|_F + \beta \| [\, A\, B\, C\, D \,] \|_q$$

with a potential regularization term choosing $\beta > 0$, $q = 0, 1, 2$.

DMD in a Nutshell

Selected References (Chronological)

Koopman, B.O. (1931): Hamiltonian systems and transformation in Hilbert space. Proc. Natl. Acad. Sci. 17(5):315—381.

Mezić, I. (2005): Spectral properties of dynamical systems, model reduction and decompositions. Nonlinear Dyn. 41(1):309—325. 10.1007/s11071-005-2824-x

Schmid, P.J. (2010): Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech. 656:5—28. 10.1017/S0022112010001217

Kutz, J.N., Brunton, S.L., Brunton, B.W., Proctor, J.L. (2016): Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems. SIAM, Philadelphia.

Proctor, J.L., Brunton, S.L., Kutz, J.N. (2016): Dynamic mode decomposition with control. SIAM J. Appl. Dyn. Syst. 15(1):142—161. 10.1137/15M1013857

Benner, P., Himpe, C., Mitchell, T. (2018): On reduced input-output dynamic mode decomposition. Adv. Comp. Math. 44(6):1751–1768. 10.1007/s10444-018-9592-x

Gosea, I.V., Pontes Duff, I. (2020): Toward fitting structured nonlinear systems by means of dynamic mode decomposition. arXiv:2003.06484.

Mauroy, A., Mezić, I., Susuki, Y., eds., (2020): The Koopman Operator in Systems and Control. Concepts, Methodologies, and Applications. LNCIS 484, Springer, Cham.

Operator Inference

Reduced-order / compressive DMD

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

Operator Inference

Reduced-order / compressive DMD

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Idea: compress trajectories using POD / PCA:

1. Let $X := [\, x_0, x_1, \ldots, x_{K-1}, x_K \,] \in \mathbb{R}^{n \times K+1}$ be the matrix of all snapshots.

Operator Inference

Reduced-order / compressive DMD

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Idea: compress trajectories using POD / PCA:

1. Let $X := [\, x_0, x_1, \ldots, x_{K-1}, x_K \,] \in \mathbb{R}^{n \times K+1}$ be the matrix of all snapshots.
2. Compute principal / dominant singular vectors via SVD $X = U\Sigma V^T$ and set $W := U(:, 1 : r)$ such that $\sum_{k=r+1}^{K+1} \sigma_k < \varepsilon$ (potentially, use centered data).

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Idea: compress trajectories using POD / PCA:

1. Let $X := [\, x_0, x_1, \ldots, x_{K-1}, x_K \,] \in \mathbb{R}^{n \times K+1}$ be the matrix of all snapshots.
2. Compute principal / dominant singular vectors via SVD $X = U\Sigma V^T$ and set $W := U(:, 1 : r)$ such that $\sum_{k=r+1}^{K+1} \sigma_k < \varepsilon$ (potentially, use centered data).
3. Compute compressed snapshot matrix $\hat{X} := W^T X$.

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Idea: compress trajectories using POD / PCA:

1. Let $X := [\, x_0, x_1, \ldots, x_{K-1}, x_K \,] \in \mathbb{R}^{n \times K+1}$ be the matrix of all snapshots.
2. Compute principal / dominant singular vectors via SVD $X = U\Sigma V^T$ and set $W := U(:, 1:r)$ such that $\sum_{k=r+1}^{K+1} \sigma_k < \varepsilon$ (potentially, use centered data).
3. Compute compressed snapshot matrix $\hat{X} := W^T X$.
4. Apply DMD using $\hat{X}_0, \hat{X}_1$ and compute reduced-order $\hat{A}$ via

$$\hat{A}_* := \arg\min_{\hat{A} \in \mathbb{R}^{r \times r}} \|\hat{X}_1 - \hat{A}\hat{X}_0\|_F + \beta \|\hat{A}\|_q.$$

Operator Inference

Reduced-order / compressive DMD

Same setting as before: given a smooth dynamical system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n.$$

Take snapshots $x_k := x(t_k)$ on grid $t_k := kh$ for $k = 0, 1, \ldots, K$ and fixed $h > 0$ (using simulation software, or measurements from real life experiment $\rightsquigarrow$ nonintrusive!).

By construction, DMD yields a linear system of order n — this may be too large!

Idea: compress trajectories using POD / PCA:

1. Let $X := [\, x_0, x_1, \ldots, x_{K-1}, x_K \,] \in \mathbb{R}^{n \times K+1}$ be the matrix of all snapshots.
2. Compute principal / dominant singular vectors via SVD $X = U\Sigma V^T$ and set $W := U(:, 1:r)$ such that $\sum_{k=r+1}^{K+1} \sigma_k < \varepsilon$ (potentially, use centered data).
3. Compute compressed snapshot matrix $\hat{X} := W^T X$.
4. Apply DMD using $\hat{X}_0, \hat{X}_1$ and compute reduced-order $\hat{A}$ via

$$\hat{A}_* := \arg\min_{\hat{A} \in \mathbb{R}^{r \times r}} \|\hat{X}_1 - \hat{A}\hat{X}_0\|_F + \beta \|\hat{A}\|_q.$$

Can be combined with ioDMD to obtain reduced-order LTI system.

Operator Inference

Basic Algorithm

Basic idea: apply compressive ioDMD in continuous-time setting,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

and impose a nonlinear structure.

Basic idea: apply compressive ioDMD in continuous-time setting,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

and impose a nonlinear structure.

Here: try to infer quadratic system

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{H}\left(\hat{x}(t) \otimes \hat{x}(t)\right) + \hat{B}u(t),$$

where $P \otimes Q := [p_{ij}Q]_{ij}$ denotes the Kronecker (tensor) product, from data

$$X := [\, x_0, x_1, \ldots, x_K \,] \in \mathbb{R}^{n \times (K+1)}, \quad U := [\, u_0, u_1, \ldots, u_K \,] \in \mathbb{R}^{m \times (K+1)}.$$

Basic idea: apply compressive ioDMD in continuous-time setting,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

and impose a nonlinear structure.

Here: try to infer quadratic system

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{H}\left(\hat{x}(t) \otimes \hat{x}(t)\right) + \hat{B}u(t),$$

where $P \otimes Q := [p_{ij}Q]_{ij}$ denotes the Kronecker (tensor) product, from data

$$X := [\, x_0, x_1, \ldots, x_K \,] \in \mathbb{R}^{n \times (K+1)}, \quad U := [\, u_0, u_1, \ldots, u_K \,] \in \mathbb{R}^{m \times (K+1)}.$$

- Use compressed trajectories (via POD / PCA) $\rightsquigarrow \hat{X}$.

Basic idea: apply compressive ioDMD in continuous-time setting,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

and impose a nonlinear structure.

Here: try to infer quadratic system

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{H}\left(\hat{x}(t) \otimes \hat{x}(t)\right) + \hat{B}u(t),$$

where $P \otimes Q := [p_{ij}Q]_{ij}$ denotes the Kronecker (tensor) product, from data

$$X := [\, x_0, x_1, \ldots, x_K \,] \in \mathbb{R}^{n\times(K+1)}, \quad U := [\, u_0, u_1, \ldots, u_K \,] \in \mathbb{R}^{m\times(K+1)}.$$

- Use compressed trajectories (via POD / PCA) $\rightsquigarrow \hat{X}$.
- Compress snapshot matrix of time derivatives: if residuals $f(t_j, u_j)$ are available

  $$\dot{\hat{X}} := [\, \dot{x}(0), \dot{x}(t_1), \ldots, \dot{x}(t_K) \,] \approx [\, f(x_0, u_0), f(x_1, u_1), \ldots, f(x_K, u_K) \,] \in \mathbb{R}^{n\times(K+1)},$$

  otherwise, approximate time-derivatives by finite differences $\rightsquigarrow \dot{\hat{X}}$.

Basic idea: apply compressive ioDMD in continuous-time setting,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

and impose a nonlinear structure.

Here: try to infer quadratic system

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{H}\left(\hat{x}(t) \otimes \hat{x}(t)\right) + \hat{B}u(t),$$

where $P \otimes Q := [p_{ij}Q]_{ij}$ denotes the Kronecker (tensor) product, from data

$$X := [\, x_0, x_1, \ldots, x_K \,] \in \mathbb{R}^{n\times(K+1)}, \quad U := [\, u_0, u_1, \ldots, u_K \,] \in \mathbb{R}^{m\times(K+1)}.$$

- Use compressed trajectories (via POD / PCA) $\rightsquigarrow \hat{X}$.
- Compress snapshot matrix of time derivatives: if residuals $f(t_j, u_j)$ are available

  $$\dot{\hat{X}} := [\, \dot{x}(0), \dot{x}(t_1), \ldots, \dot{x}(t_K) \,] \approx [\, f(x_0, u_0), f(x_1, u_1), \ldots, f(x_K, u_K) \,] \in \mathbb{R}^{n\times(K+1)},$$

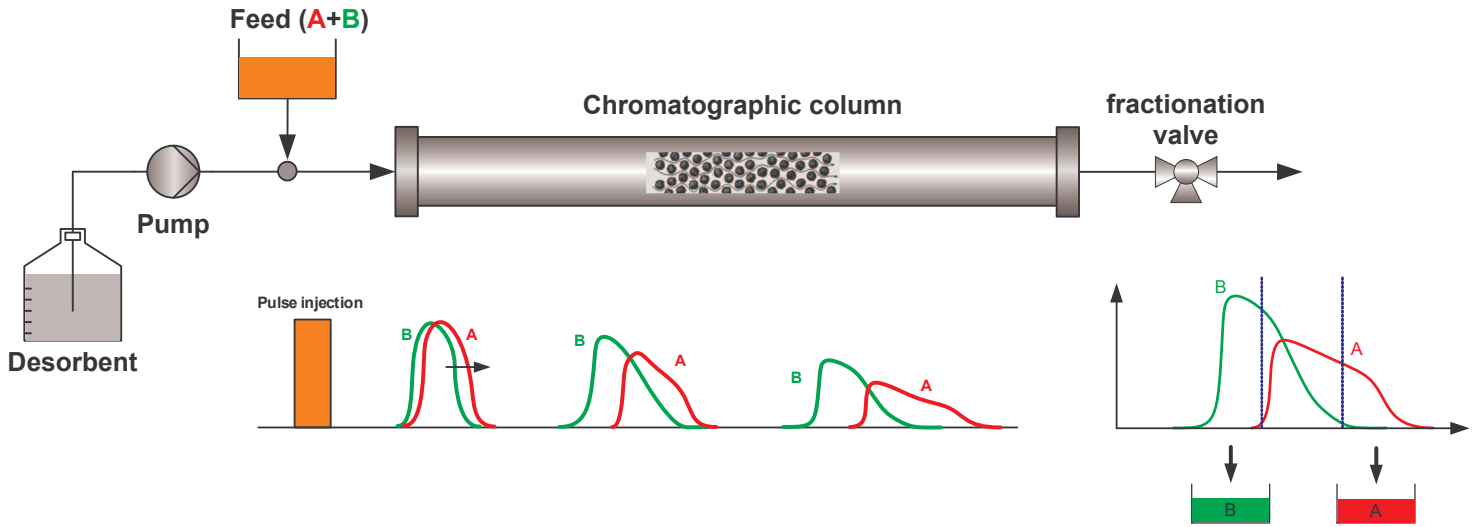
  otherwise, approximate time-derivatives by finite differences $\rightsquigarrow \dot{\hat{X}}$.
- Solve the linear least-squares problem (regression):

  $$(\hat{A}_*, \hat{H}_*, \hat{B}_*) := \arg\min_{(\hat{A},\hat{H},\hat{B})} \left\| \dot{\hat{X}} - \begin{bmatrix} \hat{A} & \hat{H} & \hat{B} \end{bmatrix} \begin{bmatrix} \hat{X} \\ \widehat{X^2} \\ U \end{bmatrix} \right\|_F + \beta \left\| \begin{bmatrix} \hat{A}\,\hat{H}\,\hat{B} \end{bmatrix} \right\|_q$$

  with potential regularization as before and $\widehat{X^2} := [x_0 \otimes x_0, \ldots, x_K \otimes x_K]$.

Operator Inference: Numerical Examples

Batch Chromatography: A Chemical Separation Process



- The dynamics of a batch chromatography column can be described by the coupled PDE system of advection-diffusion type:

$$\frac{\partial c_i}{\partial t} + \frac{1-\epsilon}{\epsilon}\frac{\partial q_i}{\partial t} + \frac{\partial c_i}{\partial x} - \frac{1}{\mathrm{Pe}}\frac{\partial^2 c_i}{\partial x^2} = 0,$$
$$\frac{\partial q_i}{\partial t} = \kappa_i \left(q_i^{Eq} - q_i\right).$$

- It is a coupled PDE; thus, the coupling structure is desired to be preserved in learned ROM
- This is achieved by block diagonal projection, thereby not mixing separate physical quantities.

Operator Inference: Numerical Examples

Batch Chromatography: A Chemical Separation Process

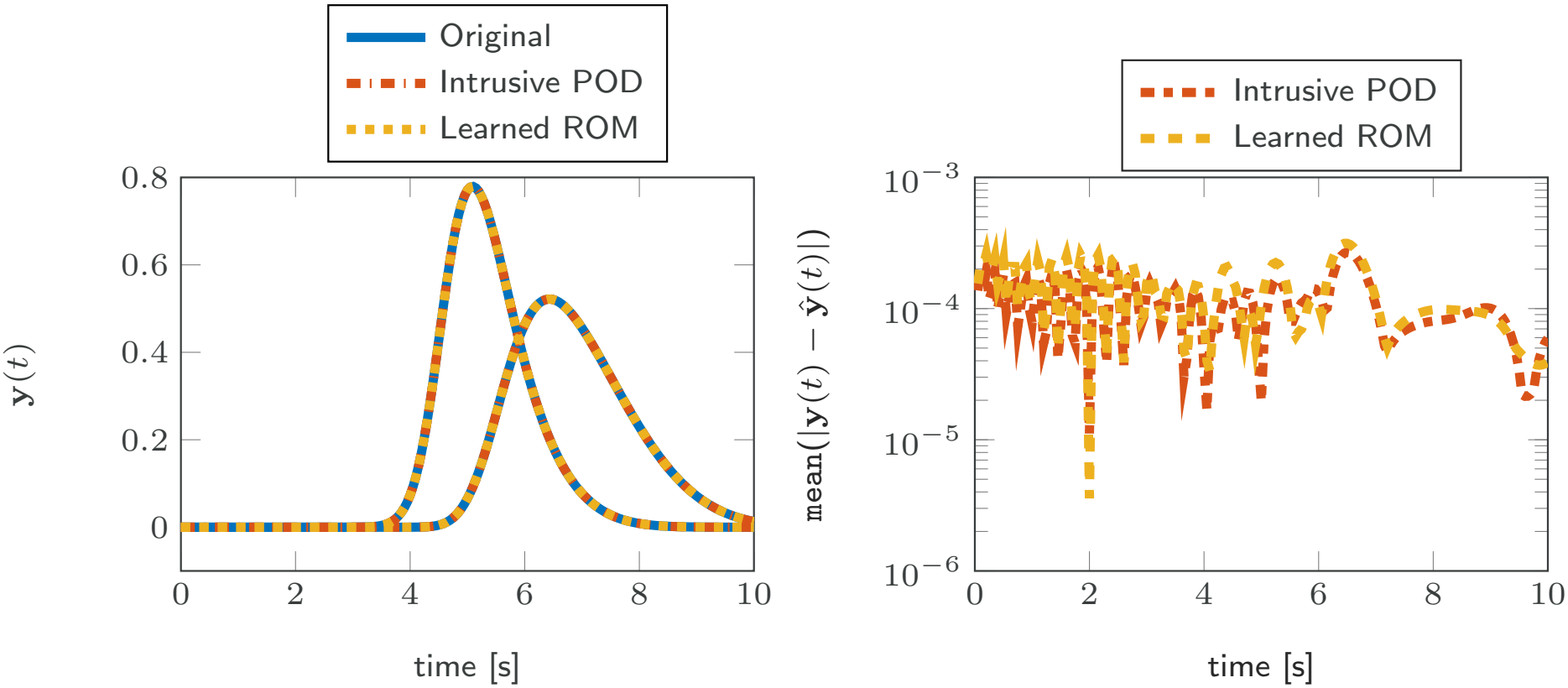


Figure: Batch chromatography example: A comparison of the POD intrusive model with the learned model of order $r = 4 \times 22$, where $n = 1600$ and $\mathrm{Pe} = 2000$.

Operator Inference: Numerical Examples

Parameterized Shallow Water Equations

- Parameterized shallow water equations are given by [Yildiz et al '21]

$$\begin{aligned}
\frac{\partial}{\partial t}\tilde{u} &= -h_x + \sin\theta\ \tilde{v} - \tilde{u}\tilde{u}_x - \tilde{v}\tilde{u}_y + \delta\cos\theta(h\tilde{u})_x - \frac{3}{8}\left(\delta\cos\theta\right)^2(h^2)_x,\\
\frac{\partial}{\partial t}\tilde{v} &= -h_y + \sin\theta\ \tilde{u} + \frac{1}{2}\delta\sin\theta\cos\theta\ h - \tilde{u}\tilde{v}_x - \tilde{v}\tilde{v}_y\\
&\qquad + \delta\cos\theta\left((h\tilde{u})_y + \frac{1}{2}h\left(\tilde{v}_x - \tilde{u}_y\right)\right) - \frac{3}{8}\left(\delta\cos\theta\right)^2(h^2)_y,\\
\frac{\partial}{\partial t}h &= -(h\tilde{u})_x - (h\tilde{v})_y + \frac{1}{2}\delta\cos\theta(h^2)_x.
\end{aligned}$$

- Parameterized by the latitude θ.
- $\tilde{\mathbf{u}} =: (\tilde{u}; \tilde{v})$ is the canonical velocity.
- h is the height field.
- We collect the training data for 5 different parameter realizations θ in $\left[\frac{\pi}{6}, \frac{\pi}{3}\right]$.
- Infer a reduced parametric model directly from data of order $r = 75$.

Operator Inference: Numerical Examples

Parameterized Shallow Water Equations

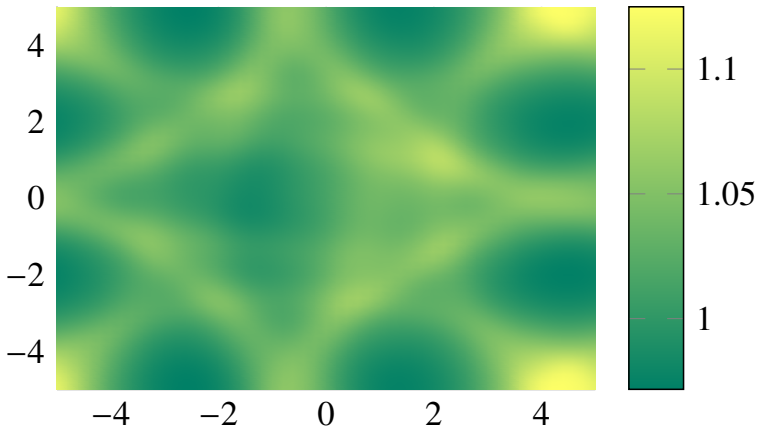
- Parameterized shallow water equations are given by [Yildiz et al '21]

$$\frac{\partial}{\partial t}\tilde{u} = -h_x + \sin\theta\ \tilde{v} - \tilde{u}\tilde{u}_x - \tilde{v}\tilde{u}_y + \delta\cos\theta(h\tilde{u})_x - \frac{3}{8}\left(\delta\cos\theta\right)^2(h^2)_x,$$

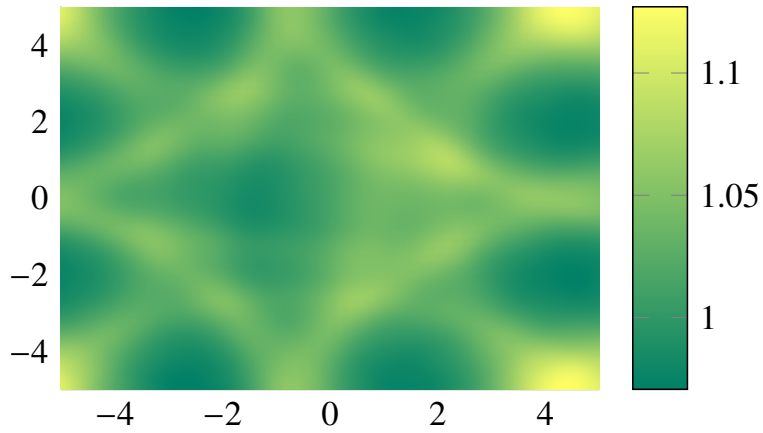
$$\frac{\partial}{\partial t}\tilde{v} = -h_y + \sin\theta\ \tilde{u} + \frac{1}{2}\delta\sin\theta\cos\theta\ h - \tilde{u}\tilde{v}_x - \tilde{v}\tilde{v}_y$$
$$+ \delta\cos\theta\left((h\tilde{u})_y + \frac{1}{2}h\left(\tilde{v}_x - \tilde{u}_y\right)\right) - \frac{3}{8}\left(\delta\cos\theta\right)^2(h^2)_y,$$

$$\frac{\partial}{\partial t}h = -(h\tilde{u})_x - (h\tilde{v})_y + \frac{1}{2}\delta\cos\theta(h^2)_x.$$

- Comparison of the height field for the parameter $\theta = \frac{5\pi}{24}$:



(a) FOM

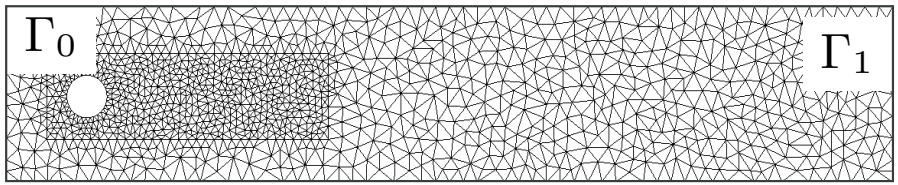


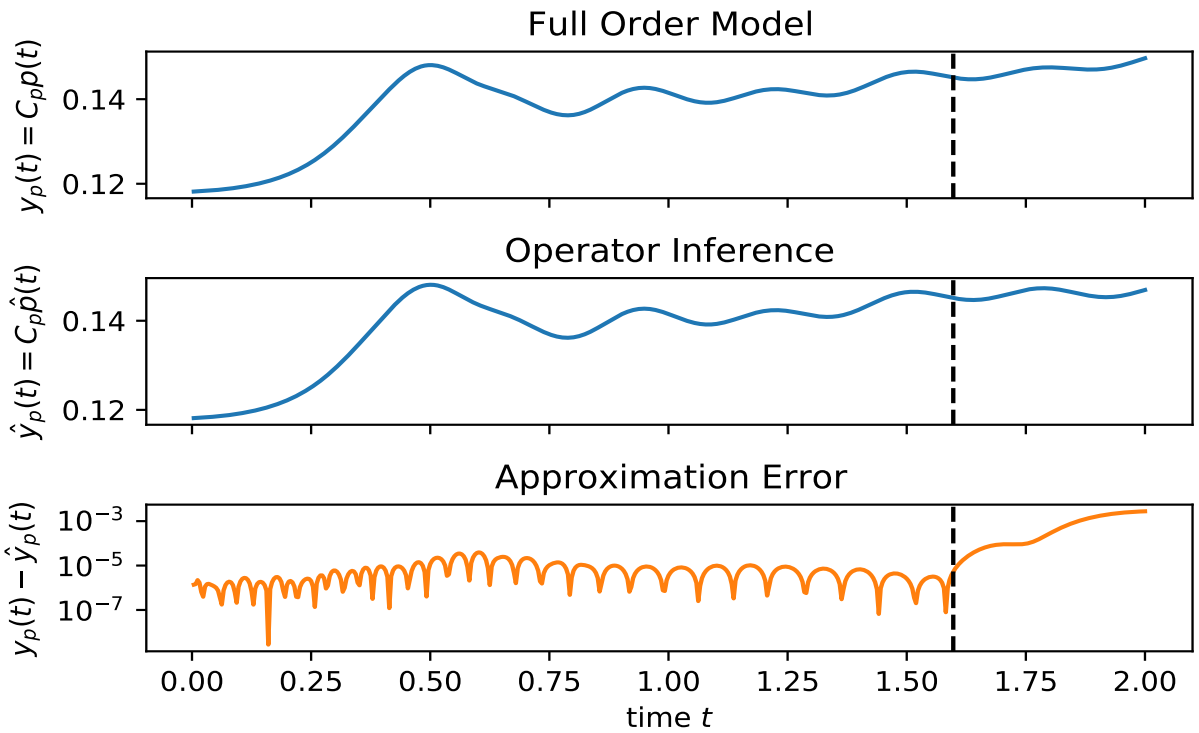
(b) Learned parametric model

Operator Inference: Numerical Examples

Navier-Stokes Equations

Tailored operator inference for incompressible Navier-Stokes equations, by heeding incompressibility condition. [B./Goyal/Heiland/Pontes '22]





Operator Inference

Selected References (Chronological)

Peherstorfer, B., Willcox, K. (2016): Data-driven operator inference for nonintrusive projection-based model reduction. Comput. Methods Appl. Mech. Eng. 306:196–215.

Brunton, B.W., Johnson, L.A., Ojemann, J.G., Kutz, J.N. (2016): Extracting spatial-temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. J. Neurosci. Methods 258:1–15.

Annoni, J., Seiler, P. (2017): A method to construct reduced-order parameter-varying models. Int. J. Robust Nonlinear Control 27(4):582–597.

Qian, E., Kramer, B., Peherstorfer, B., Willcox, K. (2020): Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. Physica D: Nonlinear Phenomena 406:132401.

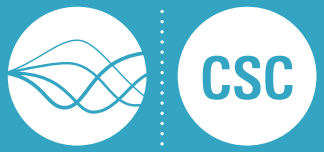
Benner, P., Goyal, P., Kramer, B., Peherstorfer, B., Willcox, K. (2020): Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. Comp. Meth. Appl. Mech. Eng., 372:113433.

Yıldız, S., Goyal, P., Benner, P., Karasozen, B. (2021): Learning reduced-order dynamics for parametrized shallow water equations from data. Int. J. Numer. Meth. Eng., 93(8):2803–2821.

Benner, P., Goyal, P., Heiland, J., Pontes Duff, I. (2022): Operator inference and physics-informed learning of low-dimensional models for incompressible flows. Elec. Trans. Numer. Anal., 56:28–51.

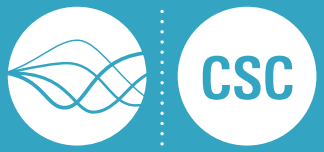
Conclusions — Part I

- DMD and operator inference (OpInf) are regression-based powerful methods to infer linear and certain nonlinear dynamical systems from data.
- Both look simple, but the devil is in the details.
- Choice of good observables? (Learning to learn?)
- Statistical aspects are not too well understood: impact of noise in the data on inferred models?
- Recent work combines OpInf with neural networks to solve nonlinear identification problems **($\rightsquigarrow$ Part II).**
- Error bounds for non-intrusive MOR not well developed yet, but theoretic results indicate that the OpInf model asymptotically (when increasing the number of snapshots) yields the POD model. Then, intrusive MOR error bounds can be applied.



Outline

1. Model Order Reduction of Dynamical Systems
2. Data-driven/-enhanced Model Reduction
3. Koopman Operator and DMD
4. Lifting Principle for Dynamical Systems



Koopman Operator and DMD

- The simplest type of model one can think of is a Linear Model
  - ⇝ Many tools for optimal/feedback control, optimization, and prediction

Koopman Operator and DMD

- The simplest type of model one can think of is a Linear Model
  - ⇝ Many tools for optimal/feedback control, optimization, and prediction

- Given data $\mathbf{x}(t_i)$ and its derivative $\dot{\mathbf{x}}(t_i)$, a linear model can be determined by solving

$$\min_{\mathbf{A}} \|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\|,$$

where $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_n)]$ and $\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)]$.

Koopman Operator and DMD

- The simplest type of model one can think of is a Linear Model
  - ⇝ Many tools for optimal/feedback control, optimization, and prediction
- Given data $\mathbf{x}(t_i)$ and its derivative $\dot{\mathbf{x}}(t_i)$, a linear model can be determined by solving

$$\min_{\mathbf{A}} \|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\|,$$

  where $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_n)]$ and $\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)]$.
- Often referred to (in a simplistic view) as Dynamic Mode Decomposition or Operator Inference.

Koopman Operator and DMD

- The simplest type of model one can think of is a Linear Model
  - ⇝ Many tools for optimal/feedback control, optimization, and prediction
- Given data $\mathbf{x}(t_i)$ and its derivative $\dot{\mathbf{x}}(t_i)$, a linear model can be determined by solving

$$\min_{\mathbf{A}} \|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\|,$$

  where $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_n)]$ and $\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)]$.
- Often referred to (in a simplistic view) as Dynamic Mode Decomposition or Operator Inference.
- Once a linear model is learned and verified, it can be deployed for control and design tasks.

Koopman Operator and DMD

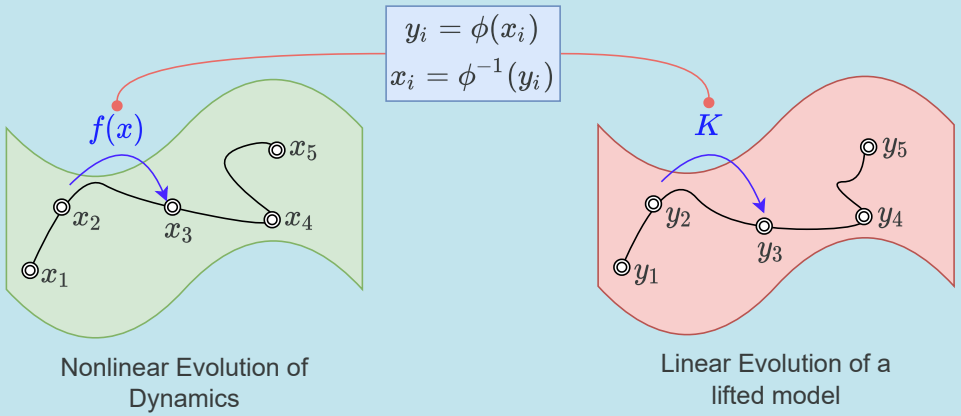
- The simplest type of model one can think of is a Linear Model
  - ⇝ Many tools for optimal/feedback control, optimization, and prediction
- Given data $\mathbf{x}(t_i)$ and its derivative $\dot{\mathbf{x}}(t_i)$, a linear model can be determined by solving

$$\min_{\mathbf{A}} \|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\|,$$

  where $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_n)]$ and $\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)]$.
- Often referred to (in a simplistic view) as Dynamic Mode Decomposition or Operator Inference.
- Once a linear model is learned and verified, it can be deployed for control and design tasks.
- However, several challenges remain:
  - Often, one cannot measure the full state $\mathbf{x}$ ⇝ partial measurements!
  - Real-world processes are often nonlinear, thus learning a linear model may not be sufficient to characterize complex dynamic behavior.

Koopman Operator in Nutshell [Koopman 1931]

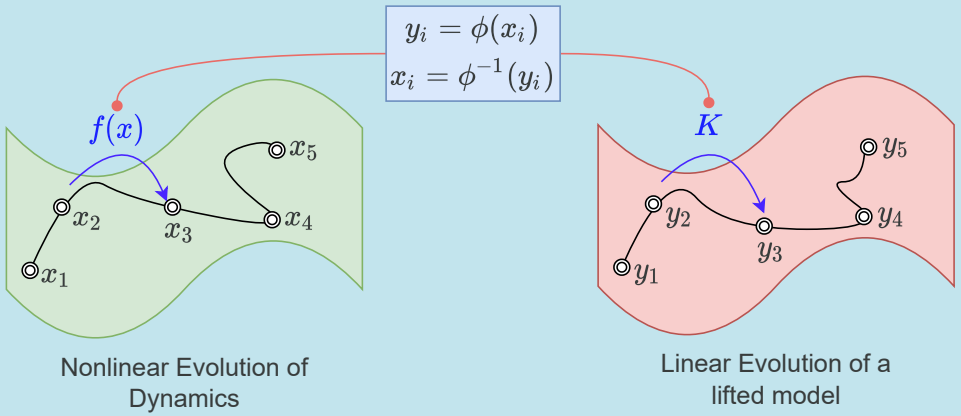
A nonlinear dynamical system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$ can be written as a linear system in a infinite dimensional Hilbert space.



Koopman Operator and DMD

Koopman Operator in Nutshell [Koopman 1931]

A nonlinear dynamical system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$ can be written as a linear system in a infinite dimensional Hilbert space.



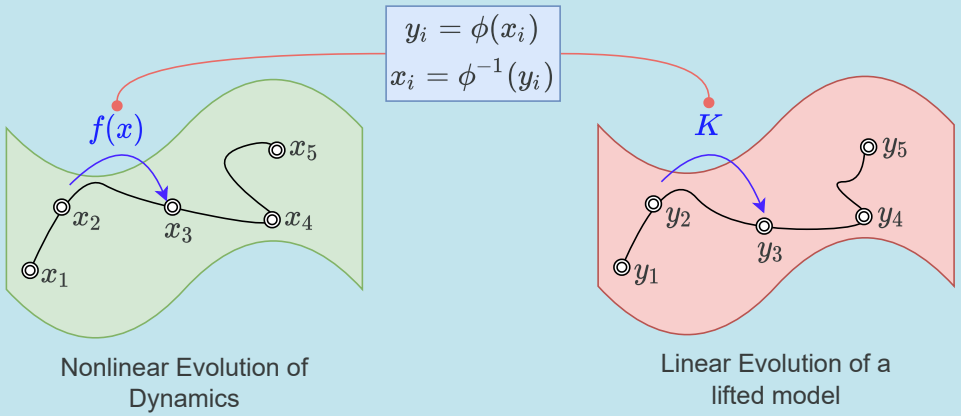
Extended DMD [Williams et al. 2015]

- The aim is to approximate infinite dimensional Koopman linear operator via a finite dimensional one.

Koopman Operator and DMD

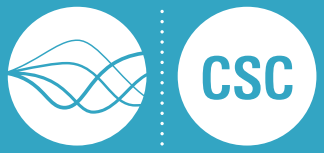
Koopman Operator in Nutshell [Koopman 1931]

A nonlinear dynamical system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$ can be written as a linear system in a infinite dimensional Hilbert space.



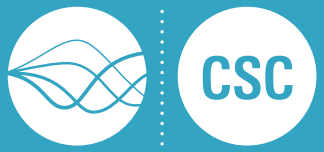
Extended DMD [Williams et al. 2015]

- The aim is to approximate infinite dimensional Koopman linear operator via a finite dimensional one.
- For this, often hand-design observables are needed,
  - but challenging design decisions need to be taken, and it still is an approximation.



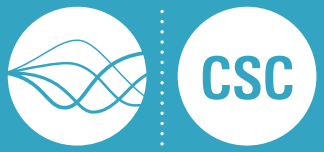
Our "Holy Grail"

- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.



Our "Holy Grail"

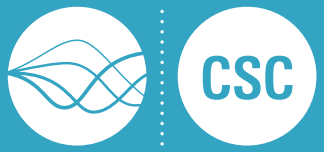
- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.
- Mapping from the observables to the state is linear (at least for good low-dimensional linear representation).



Our "Holy Grail"

- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.
- Mapping from the observables to the state is linear (at least for good low-dimensional linear representation).

Lifting Principle

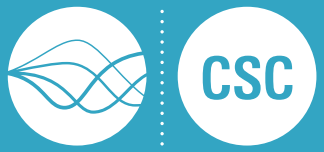


Our "Holy Grail"

- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.
- Mapping from the observables to the state is linear (at least for good low-dimensional linear representation).

Lifting Principle

- McCormick proposed a convex relaxation to solve nonlinear non-convex optimization problems. [McCormick 1976]

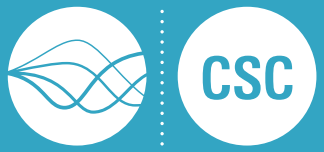


Our "Holy Grail"

- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.
- Mapping from the observables to the state is linear (at least for good low-dimensional linear representation).

Lifting Principle

- McCormick proposed a convex relaxation to solve nonlinear non-convex optimization problems. [McCormick 1976]
- Key ingredient is lifting the optimization problem to a higher dimensional space using auxiliary variables (similar to observables in Koopman theory).

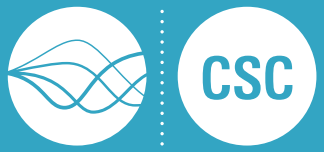


Our "Holy Grail"

- Try to re-write a nonlinear system using a simple structure in finite dimensions.
  - In Koopman theory, the structure is that of a linear system and it is infinite dimensional.
- Mapping from the observables to the state is linear (at least for good low-dimensional linear representation).

Lifting Principle

- McCormick proposed a convex relaxation to solve nonlinear non-convex optimization problems. [McCormick 1976]
- Key ingredient is lifting the optimization problem to a higher dimensional space using auxiliary variables (similar to observables in Koopman theory).
- This ideas has been further developed for learning dynamical systems.



Outline

1. Model Order Reduction of Dynamical Systems
2. Data-driven/-enhanced Model Reduction
3. Koopman Operator and DMD
4. Lifting Principle for Dynamical Systems

Lifting Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models using Lifting principle

- Consider a nonlinear system of the generic form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^n$, and the function $\mathbf{f}(\cdot)$ is assumed to be smooth enough.

Lifting Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models using Lifting principle

- Consider a nonlinear system of the generic form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}),$$

  where $\mathbf{x} \in \mathbb{R}^n$, and the function $\mathbf{f}(\cdot)$ is assumed to be smooth enough.

- Then, there exists a lifting $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its "left" inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in a quadratic model

$$\dot{\mathbf{y}}(t) = \mathbf{A}\mathbf{y} + \mathbf{H}\left(\mathbf{y}(t) \otimes \mathbf{y}(t)\right) + \mathbf{B},$$

  where $\mathbf{y}(t) = \mathcal{L}(\mathbf{x}(t))$ and $\mathcal{L}^\sharp\left(\mathbf{y}(t)\right) = \mathbf{x}(t)$.

Lifting Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models using Lifting principle

- Consider a nonlinear system of the generic form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}),$$

  where $\mathbf{x} \in \mathbb{R}^n$, and the function $\mathbf{f}(\cdot)$ is assumed to be smooth enough.

- Then, there exists a lifting $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its "left" inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in a quadratic model

$$\dot{\mathbf{y}}(t) = \mathbf{A}\mathbf{y} + \mathbf{H}\left(\mathbf{y}(t) \otimes \mathbf{y}(t)\right) + \mathbf{B},$$

  where $\mathbf{y}(t) = \mathcal{L}(\mathbf{x}(t))$ and $\mathcal{L}^\sharp\left(\mathbf{y}(t)\right) = \mathbf{x}(t)$.

- Such a lifting concept was first developed by [Savageau/Voit 1987] for control purposes.
- Also used for model reduction for nonlinear systems [Gu 2009, B./Breiten 2015].

Lifting Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models using Lifting principle

- Consider a nonlinear system of the generic form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}),$$

  where $\mathbf{x} \in \mathbb{R}^n$, and the function $\mathbf{f}(\cdot)$ is assumed to be smooth enough.

- Then, there exists a lifting $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its "left" inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in a quadratic model

$$\dot{\mathbf{y}}(t) = \mathbf{A}\mathbf{y} + \mathbf{H}\left(\mathbf{y}(t) \otimes \mathbf{y}(t)\right) + \mathbf{B},$$

  where $\mathbf{y}(t) = \mathcal{L}(\mathbf{x}(t))$ and $\mathcal{L}^\sharp\left(\mathbf{y}(t)\right) = \mathbf{x}(t)$.

- Such a lifting concept was first developed by [Savageau/Voit 1987] for control purposes.
- Also used for model reduction for nonlinear systems [Gu 2009, B./Breiten 2015].
- Recently, it has become popular using terminology Lift and Learn by Willcox, Peherstorfer, Qian, Krämer, ... [Qian et al. 2019].

Lifting Principle for Dynamical Systems

An illustration

- Consider the simple pendulum model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\sin(x_2) \\ x_1 \end{bmatrix}.$$

Lifting Principle for Dynamical Systems

An illustration

- Consider the simple pendulum model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\sin(x_2) \\ x_1 \end{bmatrix}.$$

- Lifted coordinates (observables) and the corresponding inverse transformation:

$$\mathcal{L} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ \sin(x_2) \\ \cos(x_2) \end{bmatrix} =: \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \qquad \mathcal{L}^\sharp : \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \mapsto \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Lifting Principle for Dynamical Systems

An illustration

- Consider the simple pendulum model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\sin(x_2) \\ x_1 \end{bmatrix}.$$

- Lifted coordinates (observables) and the corresponding inverse transformation:

$$\mathcal{L} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ \sin(x_2) \\ \cos(x_2) \end{bmatrix} =: \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \qquad \mathcal{L}^\sharp : \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \mapsto \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

- Consequently, we can write the dynamics in the variables y_i as a quadratic system:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} -y_3 \\ y_1 \\ y_1 y_4 \\ -y_1 y_3 \end{bmatrix}.$$

Lifting Principle for Dynamical Systems

An illustration

- Consider the simple pendulum model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\sin(x_2) \\ x_1 \end{bmatrix}.$$

- Lifted coordinates (observables) and the corresponding inverse transformation:

$$\mathcal{L} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ \sin(x_2) \\ \cos(x_2) \end{bmatrix} =: \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \qquad \mathcal{L}^\sharp : \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \mapsto \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

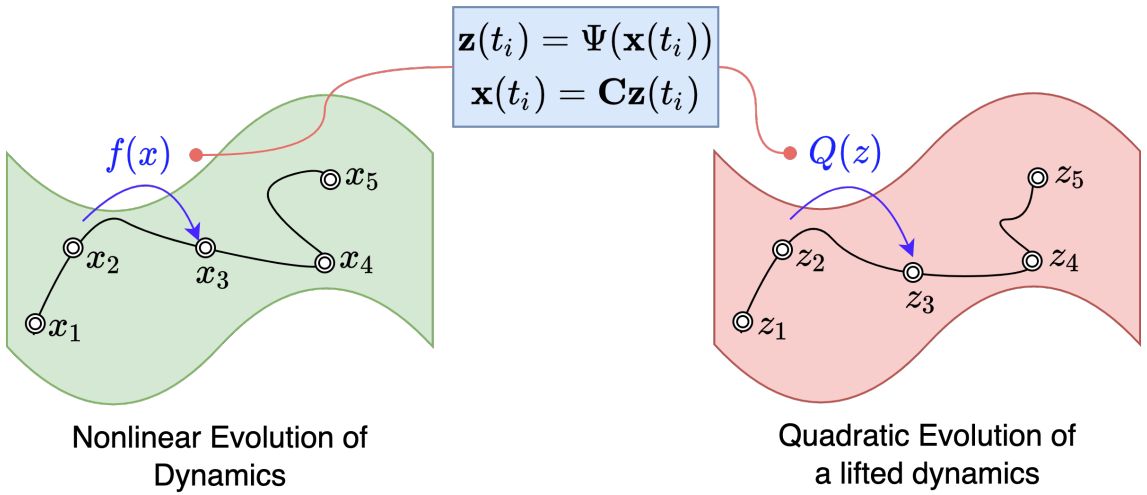
- Consequently, we can write the dynamics in the variables y_i as a quadratic system:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} -y_3 \\ y_1 \\ y_1 y_4 \\ -y_1 y_3 \end{bmatrix}.$$

- Note that the inverse mapping is indeed linear.

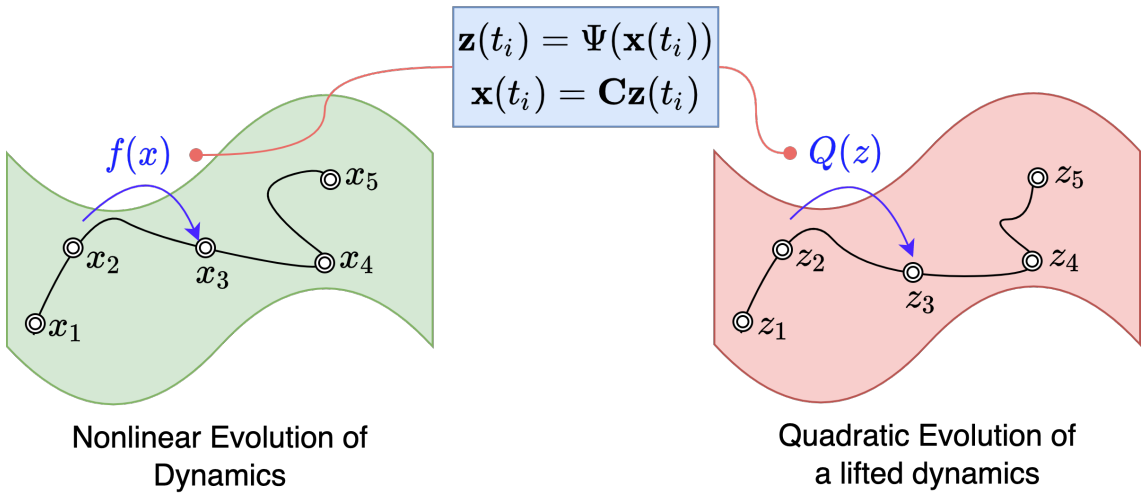
Lifting Principle for Dynamical Systems

- Using observables—inspired by *lifting principle*—we can write nonlinear systems as quadratic systems
  - which are finite dimensional
  - for which we can reconstruct full-state using a linear projection (restriction) of observables.
  - so that for a given nonlinear system, lifted observables are easy to determine.



Lifting Principle for Dynamical Systems

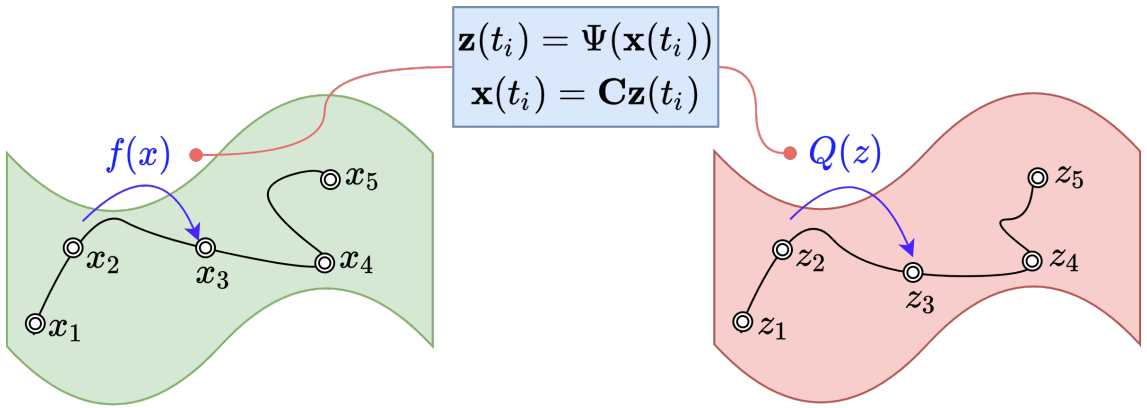
- Using observables—inspired by *lifting principle*—we can write nonlinear systems as quadratic systems
  - which are finite dimensional
  - for which we can reconstruct full-state using a linear projection (restriction) of observables.
  - so that for a given nonlinear system, lifted observables are easy to determine.



- For given nonlinear dynamical models, we can determine suitable observables.

Lifting Principle for Dynamical Systems

- Using observables—inspired by *lifting principle*—we can write nonlinear systems as quadratic systems
  - which are finite dimensional
  - for which we can reconstruct full-state using a linear projection (restriction) of observables.
  - so that for a given nonlinear system, lifted observables are easy to determine.



- For given nonlinear dynamical models, we can determine suitable observables.
- However, our goal itself is to learn dynamical models from data.

Lifting Principle for Dynamical Systems

Problem Statement (for fast decay of Kolmogorov n-width) **(Goyal/Benner 2022)**

Given data $\{\mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$ and derivative information $\{\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_{\mathcal{N}})\}$, we seek to identify

Lifting Principle for Dynamical Systems

Problem Statement (for fast decay of Kolmogorov n-width) **(Goyal/Benner 2022)**

Given data $\{\mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$ and derivative information $\{\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_{\mathcal{N}})\}$, we seek to identify

- observables $\mathbf{z} := \psi(\mathbf{x})$ such that

$$\begin{aligned}\dot{\mathbf{z}}(t) &= \mathbf{A}\mathbf{z}(t) + \mathbf{H}\left(\mathbf{z}(t) \otimes \mathbf{z}(t)\right) + \mathbf{B} =: \mathcal{Q}(\mathbf{z}),\\ \mathbf{x}(t) &= \mathbf{C}\mathbf{z}(t).\end{aligned}$$

Lifting Principle for Dynamical Systems

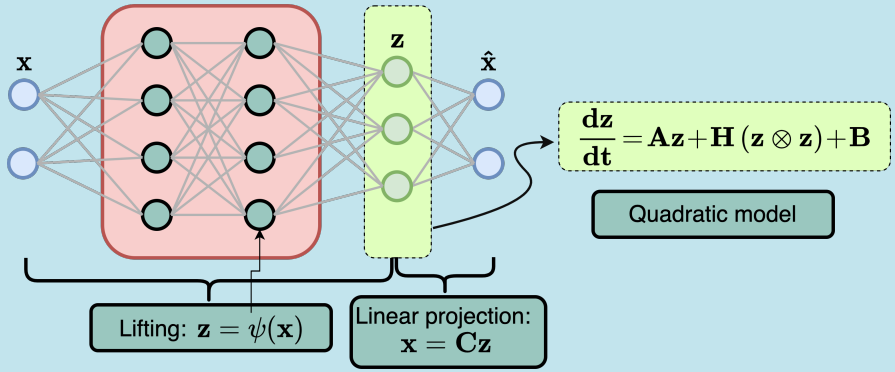
Problem Statement (for fast decay of Kolmogorov n-width) **(Goyal/Benner 2022)**

Given data $\{\mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$ and derivative information $\{\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_{\mathcal{N}})\}$, we seek to identify

- observables $\mathbf{z} := \psi(\mathbf{x})$ such that

$$\begin{aligned}\dot{\mathbf{z}}(t) &= \mathbf{A}\mathbf{z}(t) + \mathbf{H}\left(\mathbf{z}(t) \otimes \mathbf{z}(t)\right) + \mathbf{B} =: \mathcal{Q}(\mathbf{z}),\\ \mathbf{x}(t) &= \mathbf{C}\mathbf{z}(t).\end{aligned}$$

- Since we do not have any prior information, we learn $\psi(\cdot)$ using a neural network.



Lifting Principle for Dynamical Systems

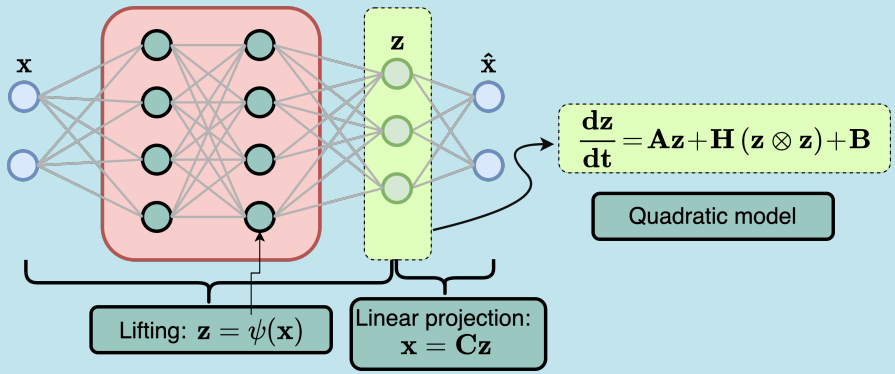
Problem Statement (for fast decay of Kolmogorov n-width) **(Goyal/Benner 2022)**

Given data $\{\mathbf{x}(t_1), \ldots, \mathbf{x}(t_{\mathcal{N}})\}$ and derivative information $\{\dot{\mathbf{x}}(t_1), \ldots, \dot{\mathbf{x}}(t_{\mathcal{N}})\}$, we seek to identify

- observables $\mathbf{z} := \psi(\mathbf{x})$ such that

$$\begin{aligned}\dot{\mathbf{z}}(t) &= \mathbf{A}\mathbf{z}(t) + \mathbf{H}\left(\mathbf{z}(t) \otimes \mathbf{z}(t)\right) + \mathbf{B} =: \mathcal{Q}(\mathbf{z}),\\ \mathbf{x}(t) &= \mathbf{C}\mathbf{z}(t).\end{aligned}$$

- Since we do not have any prior information, we learn $\psi(\cdot)$ using a neural network.
- We learn parameters of neural network $\psi(\cdot)$ and the system matrices $\{\mathbf{A}, \mathbf{H}, \mathbf{B}, \mathbf{C}\}$ simultaneously.

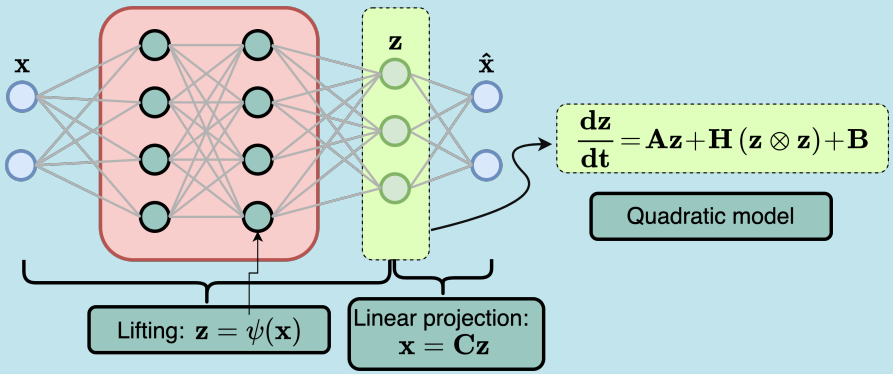


Loss function (Goyal/B. 2022)

- Compute $\dot{\mathbf{z}}$ using $\dot{\mathbf{x}}$ by the chain rule:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \| \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \dot{\mathbf{x}} - \mathcal{Q}(\mathbf{z}) \|$$

where $\mathcal{Q}(\mathbf{z}) := (\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B})$ and $\mathbf{z} = \Psi(\mathbf{x})$.



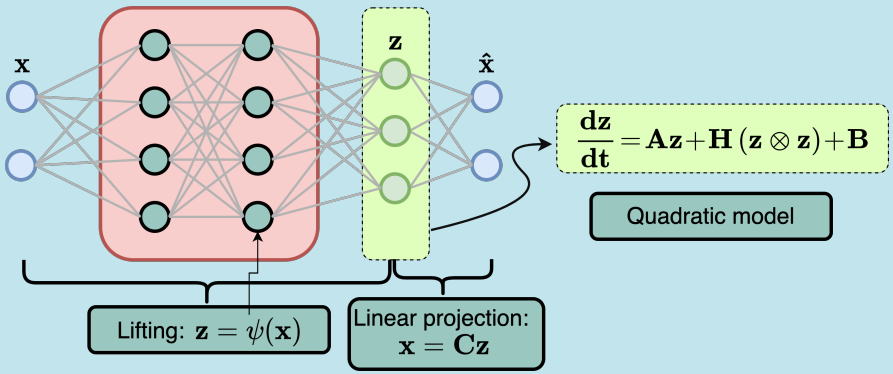
Lifting Principle for Dynamical Systems

Loss function (Goyal/B. 2022)

- Compute $\dot{\mathbf{z}}$ using $\dot{\mathbf{x}}$ by the chain rule:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \| \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \dot{\mathbf{x}} - \mathcal{Q}(\mathbf{z}) \|$$

  where $\mathcal{Q}(\mathbf{z}) := (\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B})$ and $\mathbf{z} = \Psi(\mathbf{x})$.



- Compute $\dot{\mathbf{x}}$ using $\dot{\mathbf{z}}$:

$$\dot{\mathbf{x}} = \mathbf{C}\dot{\mathbf{z}} = \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right), \text{ yielding}$$

$$\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} = \|\dot{\mathbf{x}} - \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right)\|.$$

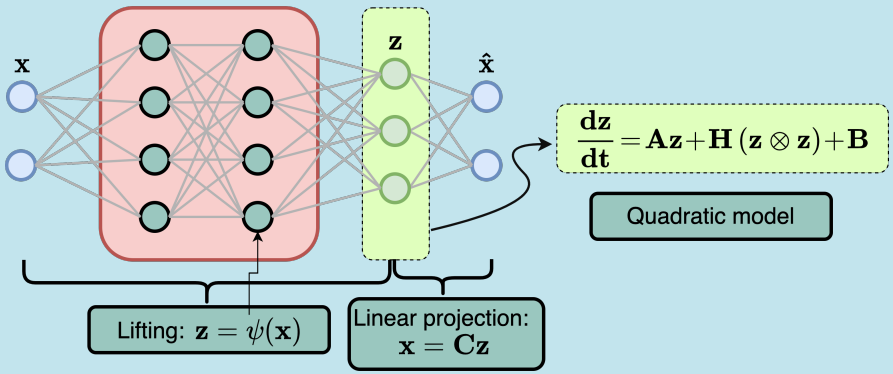
Lifting Principle for Dynamical Systems

Loss function (Goyal/B. 2022)

- Compute $\dot{\mathbf{z}}$ using $\dot{\mathbf{x}}$ by the chain rule:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \| \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \dot{\mathbf{x}} - \mathcal{Q}(\mathbf{z}) \|$$

  where $\mathcal{Q}(\mathbf{z}) := (\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B})$ and $\mathbf{z} = \Psi(\mathbf{x})$.



- Compute $\dot{\mathbf{x}}$ using $\dot{\mathbf{z}}$:

$$\dot{\mathbf{x}} = \mathbf{C}\dot{\mathbf{z}} = \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right), \text{ yielding}$$

$$\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} = \|\dot{\mathbf{x}} - \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right)\|.$$

- Autoencoder loss: $\mathcal{L}_{\text{encdec}} = \|\mathbf{x} - \mathbf{C}\Psi(\mathbf{x})\|$.

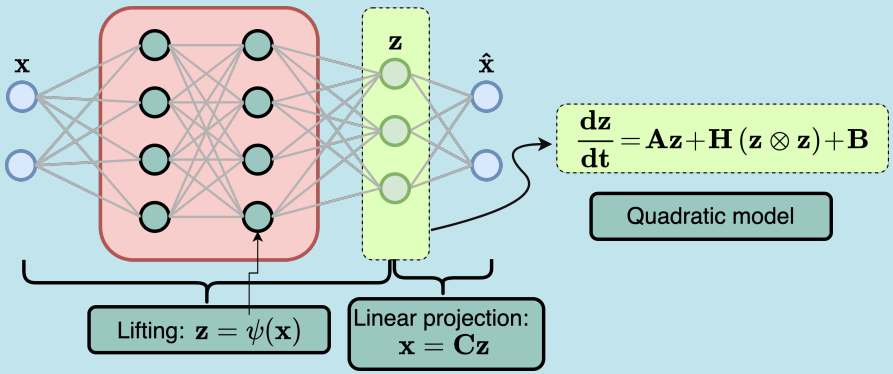
Lifting Principle for Dynamical Systems

Loss function (Goyal/B. 2022)

- Compute $\dot{\mathbf{z}}$ using $\dot{\mathbf{x}}$ by the chain rule:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \| \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \dot{\mathbf{x}} - \mathcal{Q}(\mathbf{z}) \|$$

  where $\mathcal{Q}(\mathbf{z}) := (\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B})$ and $\mathbf{z} = \Psi(\mathbf{x})$.



- Compute $\dot{\mathbf{x}}$ using $\dot{\mathbf{z}}$:

$$\dot{\mathbf{x}} = \mathbf{C}\dot{\mathbf{z}} = \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right), \text{ yielding}$$

$$\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} = \|\dot{\mathbf{x}} - \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right)\|.$$

- Autoencoder loss: $\mathcal{L}_{\text{encdec}} = \|\mathbf{x} - \mathbf{C}\Psi(\mathbf{x})\|$.
- Total loss is $\mathcal{L} = \mathcal{L}_{\text{encdec}} + \mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}}$.

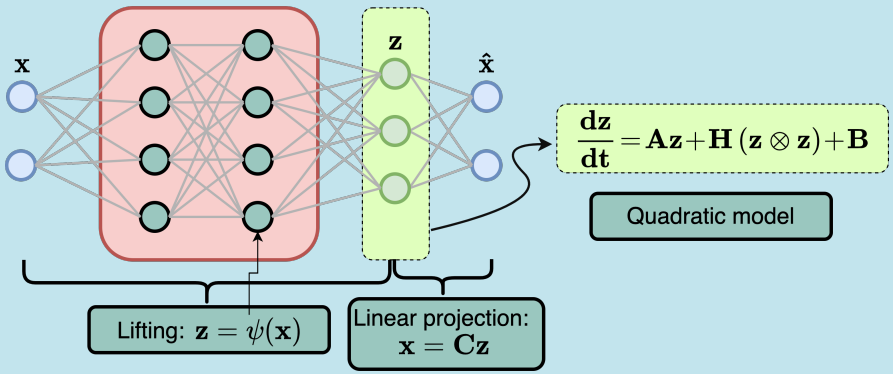
Lifting Principle for Dynamical Systems

Loss function (Goyal/B. 2022)

- Compute $\dot{\mathbf{z}}$ using $\dot{\mathbf{x}}$ by the chain rule:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \| \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \dot{\mathbf{x}} - \mathcal{Q}(\mathbf{z}) \|$$

  where $\mathcal{Q}(\mathbf{z}) := (\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B})$ and $\mathbf{z} = \Psi(\mathbf{x})$.



- Compute $\dot{\mathbf{x}}$ using $\dot{\mathbf{z}}$:

$$\dot{\mathbf{x}} = \mathbf{C}\dot{\mathbf{z}} = \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right), \text{ yielding}$$

$$\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} = \|\dot{\mathbf{x}} - \mathbf{C}\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right)\|.$$

- Autoencoder loss: $\mathcal{L}_{\text{encdec}} = \|\mathbf{x} - \mathbf{C}\Psi(\mathbf{x})\|$.
- Total loss is $\mathcal{L} = \mathcal{L}_{\text{encdec}} + \mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}}$.

- Note that once we have all these parameters, we need encoder (neural network) only to get initial condition for $\mathbf{z}$.
- The rest of the model is very classical state-space quadratic model
  $\rightsquigarrow$ can be used for engineering design.

Numerical Examples

Lambda–Omega reaction–diffusion example

- The governing equations are

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}).$$

- We take 100×100 grid and collect 100 data points in time $t = [0, 5]$.

Numerical Examples

Lambda–Omega reaction–diffusion example

- The governing equations are

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}).$$

- We take 100×100 grid and collect 100 data points in time $t = [0, 5]$.
- We consider the first 75 points for training and the last 25 for testing.

Numerical Examples

Lambda–Omega reaction–diffusion example

- The governing equations are

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}).$$

- We take 100×100 grid and collect 100 data points in time $t = [0, 5]$.
- We consider the first 75 points for training and the last 25 for testing.
- The data are high-dimensional $(2 \cdot 10^4)$ and exhibit an exponential decay of singular values, so we compress the data using projection onto the first two POD models.

Numerical Examples

Lambda–Omega reaction–diffusion example

- The governing equations are

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}).$$

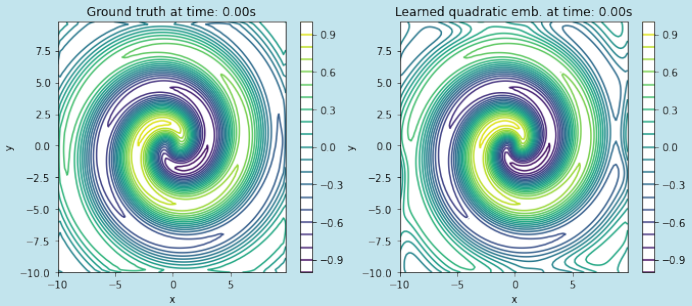
- We take 100×100 grid and collect 100 data points in time $t = [0, 5]$.
- We consider the first 75 points for training and the last 25 for testing.
- The data are high-dimensional $(2 \cdot 10^4)$ and exhibit an exponential decay of singular values, so we compress the data using projection onto the first two POD models.
- We learn a quadratic model of $\dim = 2$ using the projected data as input.

Lambda–Omega reaction–diffusion example

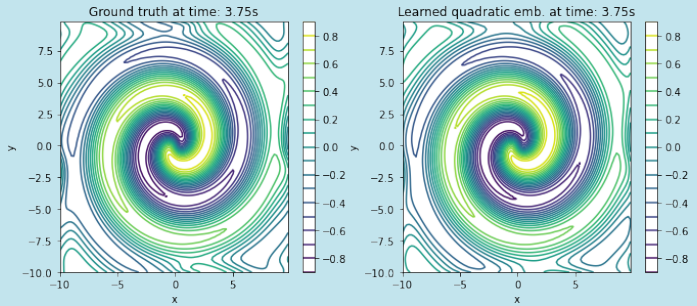
- The governing equations are

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d_1(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d_2(v_{xx} + v_{yy}).$$

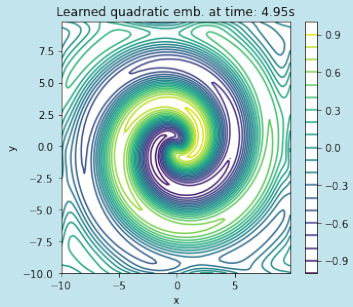
- We take 100×100 grid and collect 100 data points in time $t = [0, 5]$.
- We consider the first 75 points for training and the last 25 for testing.
- The data are high-dimensional $(2 \cdot 10^4)$ and exhibit an exponential decay of singular values, so we compress the data using projection onto the first two POD models.
- We learn a quadratic model of $\dim = 2$ using the projected data as input.



training



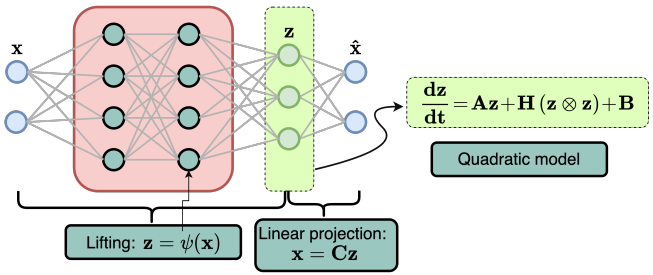
testing truth known



testing truth unknown

Challenges and Possible Solutions

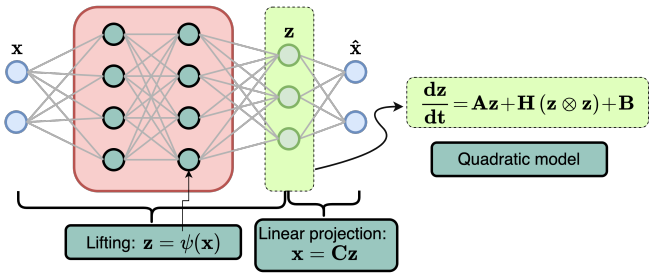
1. Recall that we have a linear projection from $\mathbf{z} \mapsto \mathbf{x}$. It works good only if we have a fast decay of singular values of our high-dimensional data
   - However, there is no suitable low-dimensional linear subspace for advection-dominant problems.
     - ⇝ Slow decay of Komologov n-width.
     - ⇝ Need a large-dimensional $\mathbf{z}$, meaning engineering studies can still be intractable.

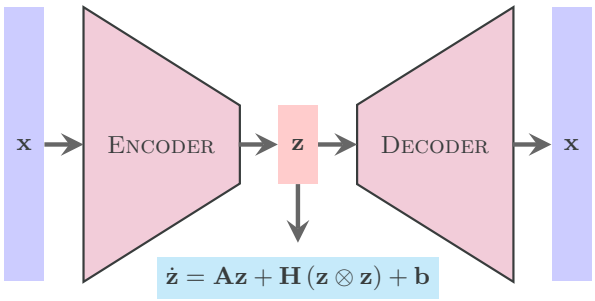


Challenges and Possible Solutions

1. Recall that we have a linear projection from $\mathbf{z} \mapsto \mathbf{x}$. It works good only if we have a fast decay of singular values of our high-dimensional data
   - However, there is no suitable low-dimensional linear subspace for advection-dominant problems.
     - ⇝ Slow decay of Komologov n-width.
     - ⇝ Need a large-dimensional $\mathbf{z}$, meaning engineering studies can still be intractable.

Remedy: Use a nonlinear decoder using neural networks (e.g., convolutional NNs for structured data)



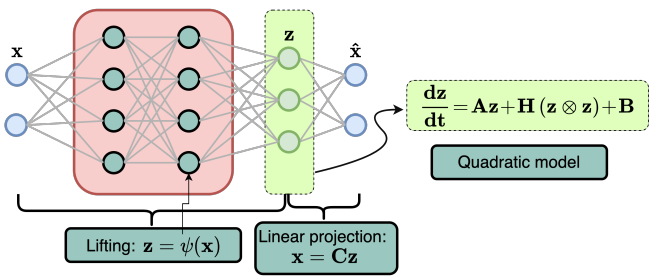


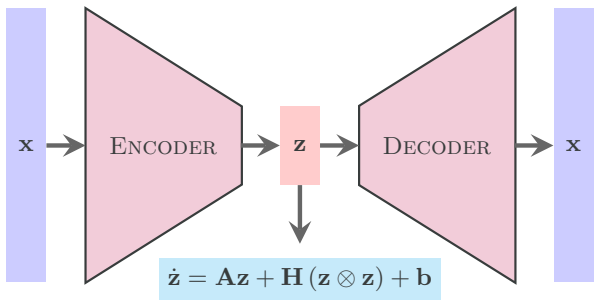
Challenges and Possible Solutions

1. Recall that we have a linear projection from $\mathbf{z} \mapsto \mathbf{x}$. It works good only if we have a fast decay of singular values of our high-dimensional data
   - However, there is no suitable low-dimensional linear subspace for advection-dominant problems.
     - ⇝ Slow decay of Komologov n-width.
     - ⇝ Need a large-dimensional $\mathbf{z}$, meaning engineering studies can still be intractable.

   Remedy: Use a nonlinear decoder using neural networks (e.g., convolutional NNs for structured data)

2. Moreover, to train networks, we need to determine derivative of output w.r.t. inputs.
   - If $\dim(\mathbf{x})$ is large, then derivative computations using, e.g., `autograd` become computationally very expensive.





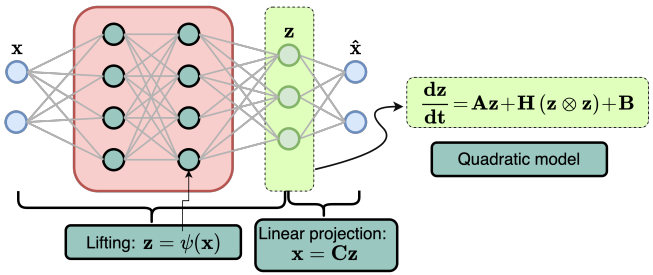
Challenges and Possible Solutions

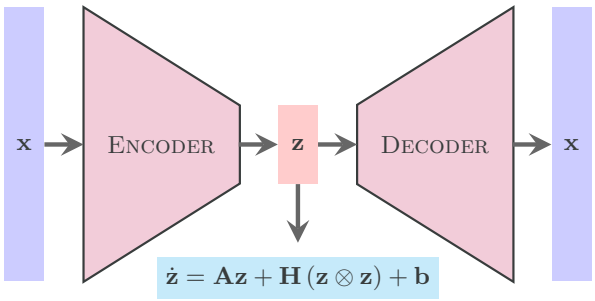
1. Recall that we have a linear projection from $\mathbf{z} \mapsto \mathbf{x}$. It works good only if we have a fast decay of singular values of our high-dimensional data
   - However, there is no suitable low-dimensional linear subspace for advection-dominant problems.
     - ⇝ Slow decay of Komologov n-width.
     - ⇝ Need a large-dimensional $\mathbf{z}$, meaning engineering studies can still be intractable.

   Remedy: Use a nonlinear decoder using neural networks (e.g., convolutional NNs for structured data)

2. Moreover, to train networks, we need to determine derivative of output w.r.t. inputs.
   - If $\dim(\mathbf{x})$ is large, then derivative computations using, e.g., `autograd` become computationally very expensive.

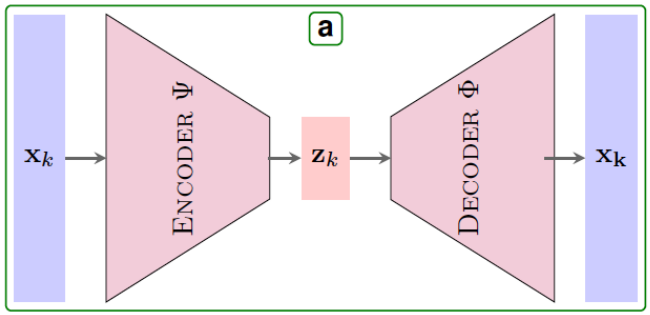
   Remedy: Embed a numerical integrator





Numerical Integrator and Low-dimensional Quadratic-embeddings

- Combining all, we have



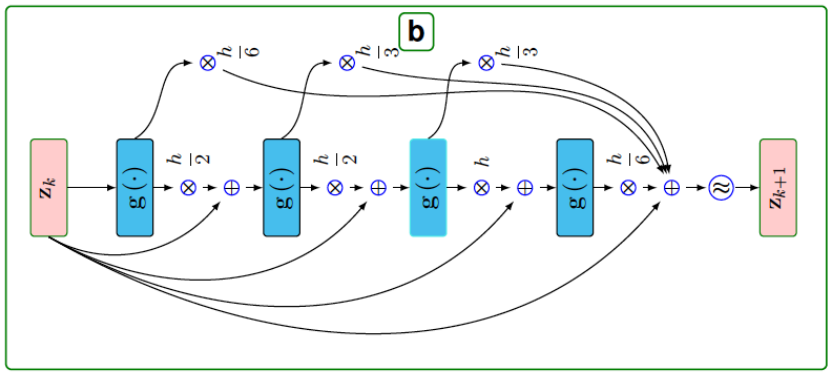
Encoding: $\mathbf{z}_k = \Psi(\mathbf{x}_k)$

Decoding: $\mathbf{x}_k = \Phi(\mathbf{z}_k)$

Runge-Kutta Constraint:

$\mathbf{z}_{k+1} \approx \Pi_{\mathsf{RK4}}(\mathbf{z}_k)$

$\mathbf{g}(\mathbf{z}) := \mathbf{A}\mathbf{z} + \mathbf{H}(\mathbf{z} \otimes \mathbf{z}) + \mathbf{b}$



- We focus on a Runge-Kutta scheme, but any integrator including adaptive ones can be utilized using Neural ODEs. [Chen et al. 2018]
- Once such an architecture is framed, we can learn encoder, decoder, and a quadratic model.

Numerical Example
Tubular Reactor Model

- One dimensional model with a single reaction, describing dynamics of the species concentration $\psi(x,t)$ and temperature $\theta(x,t)$ via

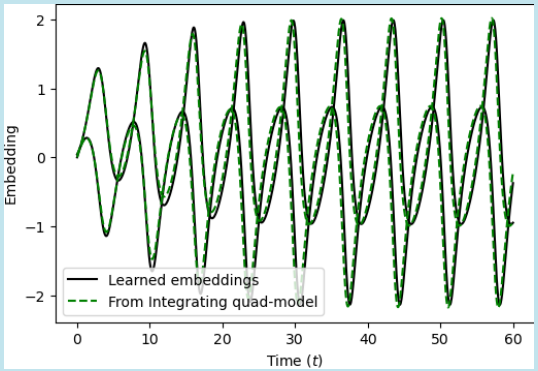
$$\begin{aligned}\frac{\partial \psi}{\partial t} &= \frac{1}{\mathrm{Pe}}\frac{\partial^2 \psi}{\partial x^2} - \frac{\partial \psi}{\partial x} - \mathcal{D}\mathcal{F}(\psi,\theta;\gamma),\\ \frac{\partial \theta}{\partial t} &= \frac{1}{\mathrm{Pe}}\frac{\partial^2 \theta}{\partial x^2} - \frac{\partial \theta}{\partial x} - \beta(\theta-\theta_{\mathsf{ref}}) + \mathcal{B}\mathcal{D}\mathcal{F}(\psi,\theta;\gamma),\end{aligned}$$

  with spatial variable $x \in (0,1)$, time $t > 0$ and Arrhenius reaction term

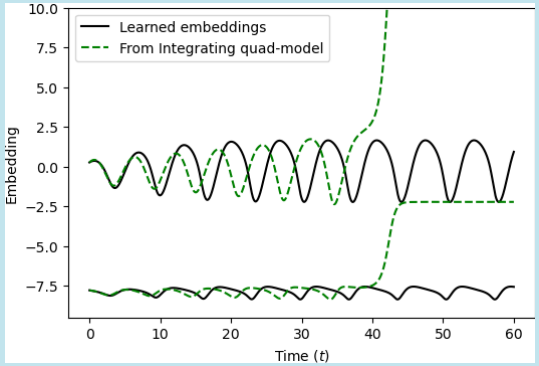
$$\mathcal{F}(\psi,\theta;\gamma) = \psi \exp\left(\gamma - \frac{\gamma}{\theta}\right).$$

- Collect snapshots in time $\mathbf{T} = [0,10]$.
- We learn 2-dimensional embeddings using convolutional autoencoder.
- For comparison, we also compute a 2-dimensional model using POD projection (classical OpInf).

Numerical Example
Tubular Reactor Model

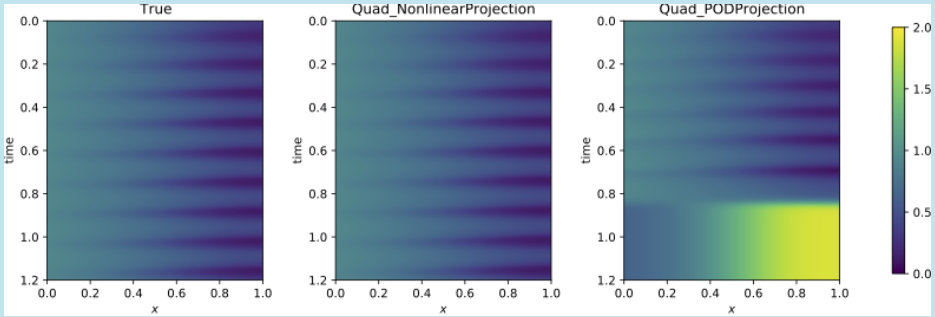


(a) convolutional autoencoder

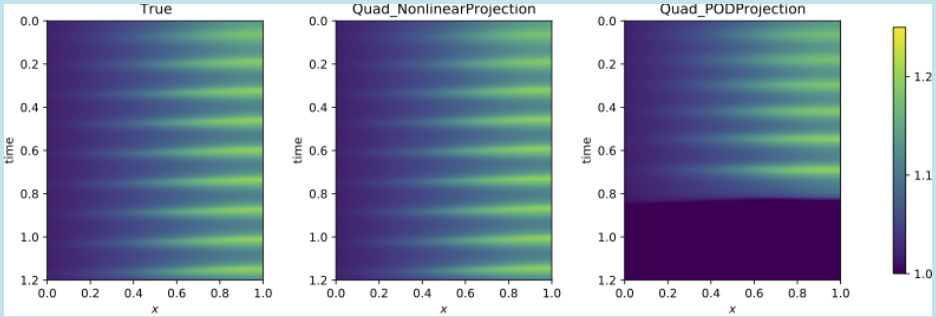


(b) POD

Figure: 2-dimensional embeddings.



(a) Concentration on the full-grid.



(b) Temperature on the full-grid.

Figure: Comparison of the convolutional autoencoders and POD-based approaches.

Numerical Example
2D Burgers equation

- Governing equation:

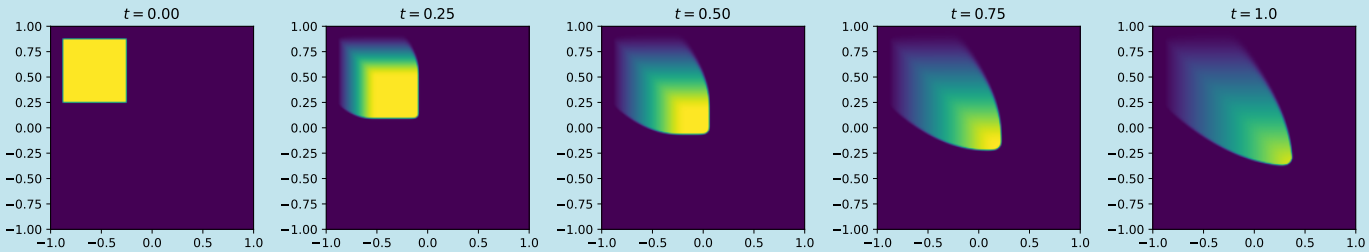
$$\frac{\partial u(x,t)}{\partial t} + \left(\frac{1}{2}, \frac{1}{2}\right)^\top \cdot \nabla u(x,t)^2 = 0 \quad \forall (x,t) \in \Omega \times [0,T].$$

Numerical Example
2D Burgers equation

- Governing equation:

$$\frac{\partial u(x,t)}{\partial t} + \left(\frac{1}{2}, \frac{1}{2}\right)^\top \cdot \nabla u(x,t)^2 = 0 \quad \forall (x,t) \in \Omega \times [0,T].$$

- We collect snapshots 100 snapshots in $[0,1]$ by taking 512 points in x and y directions $\rightsquigarrow$ full dimensional model with $262\,144$ DoFs.

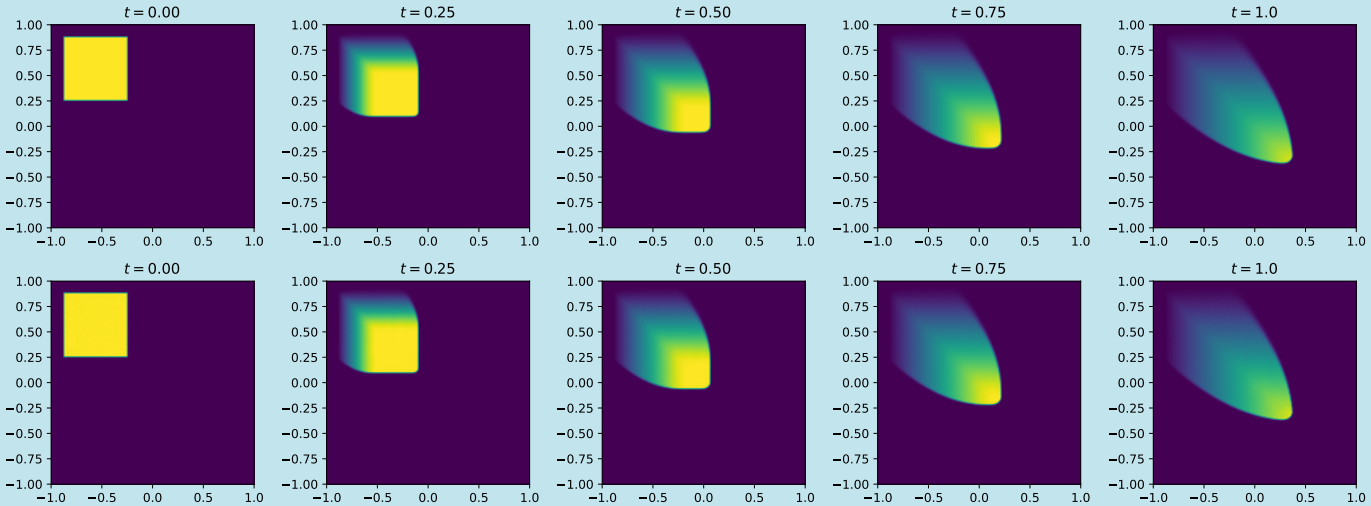


Numerical Example
2D Burgers equation

- Governing equation:

$$\frac{\partial u(x,t)}{\partial t} + \left(\frac{1}{2}, \frac{1}{2}\right)^\top \cdot \nabla u(x,t)^2 = 0 \quad \forall (x,t) \in \Omega \times [0,T].$$

- We collect snapshots 100 snapshots in $[0,1]$ by taking 512 points in x and y directions $\rightsquigarrow$ full dimensional model with $262\,144$ DoFs.
- We learn one-dimensional quadratic model, and encoder and decoder are convolutional neural networks. [Goyal/B. 2021]

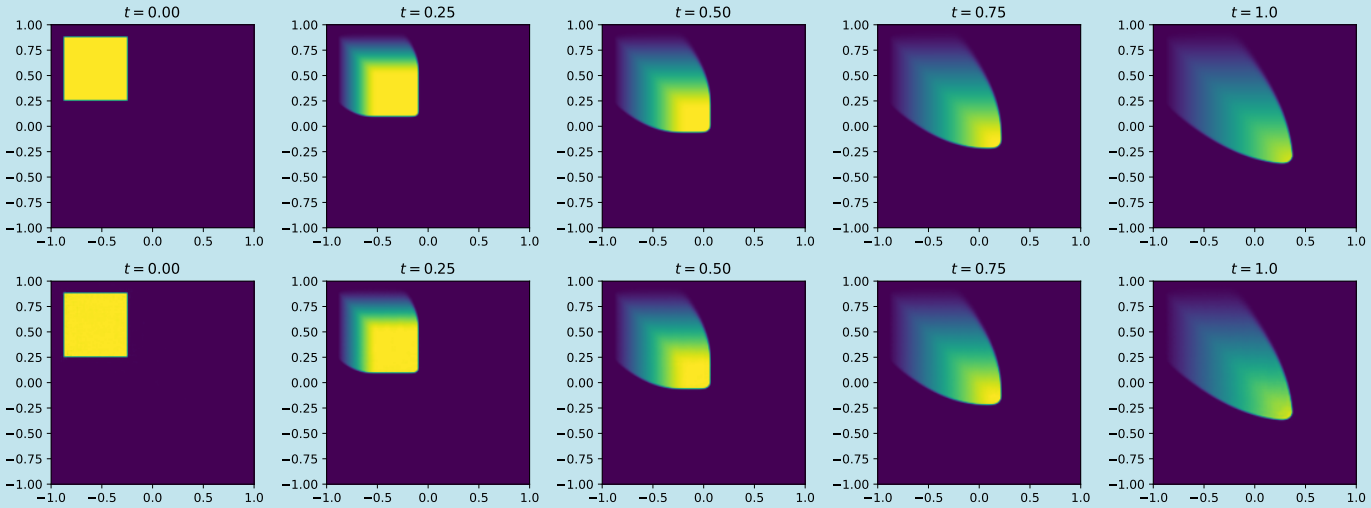


Numerical Example
2D Burgers equation

- Governing equation:

$$\frac{\partial u(x,t)}{\partial t} + \left(\frac{1}{2}, \frac{1}{2}\right)^\top \cdot \nabla u(x,t)^2 = 0 \quad \forall (x,t) \in \Omega \times [0,T].$$

- We collect snapshots 100 snapshots in $[0,1]$ by taking 512 points in x and y directions $\rightsquigarrow$ full dimensional model with $262\,144$ DoFs.
- We learn one-dimensional quadratic model, and encoder and decoder are convolutional neural networks. [Goyal/B. 2021]



- Note that for rich dynamics,we may need to increase the dimension of the quadratic embeddings.

Contributions

- Discussed lifting principle for nonlinear dynamical systems.

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).

  $\rightsquigarrow$ Notion of quadratic embeddings.

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).
  - $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).

  $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).
- For high-dimensional data with slow decay of singular values, we utilize nonlinear decoders, which allows identification of PDE models with slowly decaying Kolmogorov n-width.

Outlook

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).

  $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).
- For high-dimensional data with slow decay of singular values, we utilize nonlinear decoders, which allows identification of PDE models with slowly decaying Kolmogorov n-width.

Open work

- Extensions to Hamiltonian, parametric, and control systems.

Outlook

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).

  $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).
- For high-dimensional data with slow decay of singular values, we utilize nonlinear decoders, which allows identification of PDE models with slowly decaying Kolmogorov n-width.

Open work

- Extensions to Hamiltonian, parametric, and control systems.
- Stability guarantees of the quadratic model for the embeddings?

Outlook

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).

  $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).
- For high-dimensional data with slow decay of singular values, we utilize nonlinear decoders, which allows identification of PDE models with slowly decaying Kolmogorov n-width.

Open work

- Extensions to Hamiltonian, parametric, and control systems.
- Stability guarantees of the quadratic model for the embeddings?
- Work on real-engineering (reactor model) and investigate how to use more physics e.g., mass/energy conservation laws!

Outlook

Contributions

- Discussed lifting principle for nonlinear dynamical systems.
- Lifting allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).
  - $\rightsquigarrow$ Notion of quadratic embeddings.
- To determine embeddings, we make use of neural networks (e.g., CNNs).
- For high-dimensional data with slow decay of singular values, we utilize nonlinear decoders, which allows identification of PDE models with slowly decaying Kolmogorov n-width.

Open work

- Extensions to Hamiltonian, parametric, and control systems.
- Stability guarantees for learned models.
- Work on real-engineering (reactor model) and investigate how to use more physics e.g., mass/energy conservation laws!

Thank you for your attention!!

Benner, P. and Breiten, T. (2015).
Two-Sided Projection Methods for Nonlinear Model Order Reduction.
SIAM J. Sci. Comp., 37(2):B239—B260.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018).
Neural ordinary differential equations.
In *Advances Neural Inform. Processing Sys.*, pages 6571–6583.

Folkestad, C., Pastor, D., Mezic, I., Mohr, R., Fonoberova, M., and Burdick, J. (2020).
Extended dynamic mode decomposition with learned Koopman eigenfunctions for prediction and control.
In *American Control Conference (ACC)*, pages 3906–3913. IEEE.

Goyal, P. and Benner, P. (2021).
Learning low-dimensional quadratic-embeddings of high-fidelity nonlinear dynamics using deep learning.
e-print arXiv:2111.12995.

Goyal, P. and Benner, P. (2022).
Generalized quadratic-embeddings for nonlinear dynamics using deep learning.
e-print arXiv:2211.00357.

Gu, C. (2011).
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(9):1307–1320.

Koopman, B. O. (1931).
Hamiltonian systems and transformation in Hilbert space.
Proc. Nat. Acad. Sci. U.S.A., 17(5):315.

Lusch, B., Kutz, J. N., and Brunton, S. L. (2018).
Deep learning for universal linear embeddings of nonlinear dynamics.
Nature Commu., 9(1):1–10.