Generalized Stability Guaranteed Quadratic Embeddings for Nonlinear Dynamics

Pawan Goyal, appliedAI Initiative

Joint work with
Peter Benner (Max Planck Institute, Magdeburg)
Igor Pontes Duff (Max Planck Institute, Magdeburg)

TransferLab Seminar,
Online

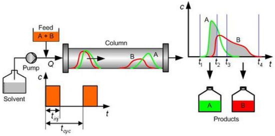
September, 12, 2024

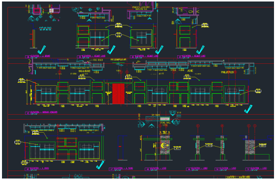
initiative for applied artificial intelligence

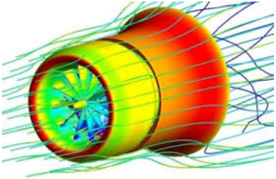
MAX-PLANCK-INSTITUT FÜR DYNAMIK KOMPLEXER TECHNISCHER SYSTEME MAGDEBURG

UNTER NEHMER TUM

IPAI







Dynamic models are important to

- analyze transient behavior under operating conditions
- control design
- parameter optimization
- long-time horizon prediction

Objective

Problem set-up

- Construct a mathematical model

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

describing dynamics of the process.

Objective

Problem set-up

- Construct a mathematical model

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

  describing dynamics of the process.
- Neural network-based approaches: recurrent neural networks and long short time memory networks

Objective

Problem set-up

- Construct a mathematical model

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

  describing dynamics of the process.
- Neural network-based approaches: recurrent neural networks and long short time memory networks
- The more information about the process is known, the more we can make learning efficient.

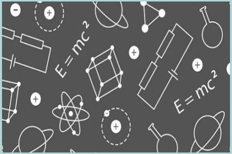
Problem set-up

- Construct a mathematical model

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

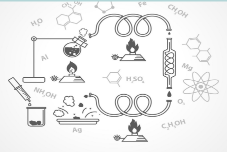
  describing dynamics of the process.
- Neural network-based approaches: recurrent neural networks and long short time memory networks
- The more information about the process is known, the more we can make learning efficient.

Key sources of information



Physical laws

Domain knowledge



Collected data

Objective

Problem set-up

- Construct a mathematical model

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

  describing dynamics of the process.
- Neural network-based approaches: recurrent neural networks and long short time memory networks
- The more information about the process is known, the more we can make learning efficient.
- For efficient engineering study, e.g., parameter optimization, control, we want the model to be as simple as possible.

Koopman Operator and DMD

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

Koopman Operator and DMD

- The simplest model, one can think of, is Linear Models
  - ⇝ 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 model, one can think of, is Linear Models
  - ⇝ 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 as Dynamic Mode Decomposition, or Operator Inference

Koopman Operator and DMD

- The simplest model, one can think of, is Linear Models
  - ⇝ 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 as Dynamic Mode Decomposition, or Operator Inference
- Once linear models are learned and verified, we can deploy for engineering studies

Koopman Operator and DMD

- The simplest model, one can think of, is Linear Models
  - ⇝ 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

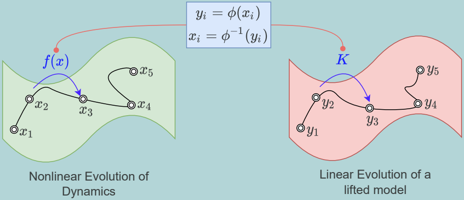
$$\min_{\mathbf{A}} \|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\|,$$

  where $\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \dots, \dot{\mathbf{x}}(t_n)]$ and $\mathbf{X} = [\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)]$
- Often referred to as Dynamic Mode Decomposition, or Operator Inference
- Once linear models are learned and verified, we can deploy for engineering studies
- However, challenges are:
  - Cannot measure the full state $\mathbf{x} \rightsquigarrow$ partial measurements
  - The world is nonlinear, thus learning a linear model may not be sufficient to characterize complex dynamic behavior

Koopman Operator and DMD

Koopman Operator in Nutshell [Koopman 1931]

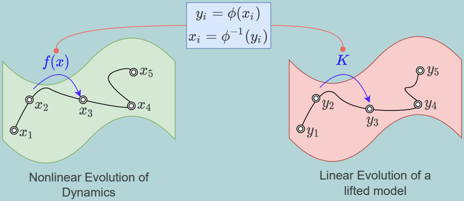
A nonlinear dynamical system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$ can be written as a linear system in a infinite dimensional Hilbert space.



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.



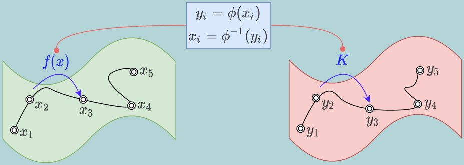
Extended DMD [Williams et al. 2015]

- An 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.



Nonlinear Evolution of Dynamics

Linear Evolution of a lifted model

Extended DMD [Williams et al. 2015]

- An aim is to approximate infinite dimensional Koopman linear operator via a finite dimensional one.
- For this, often hand-design observables are needed, which are
    - challenging, and gives an approximation.

Koopman Operator and DMD

Example 1 [Lusch et al. '18]

- Consider nonlinear system.

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\mathbf{x}_1 \\ \mathbf{x}_2 - \mathbf{x}_1^2 \end{bmatrix}.$$

Example 1 [Lusch et al. '18]

- Consider nonlinear system.

$$\begin{bmatrix}\dot{\mathbf{x}}_1(t)\\ \dot{\mathbf{x}}_2(t)\end{bmatrix} = \begin{bmatrix}-\mathbf{x}_1\\ \mathbf{x}_2 - \mathbf{x}_1^2\end{bmatrix}.$$

- Introduce a variable $\mathbf{x}_3 := \mathbf{x}_1^2$. This gives $\dot{\mathbf{x}}_3 = 2\cdot\dot{\mathbf{x}}_1\mathbf{x}_1 = -2\cdot\mathbf{x}_1^2 = -2\cdot\mathbf{x}_3$.

Koopman Operator and DMD

Example 1 [Lusch et al. '18]

- Consider nonlinear system.
$$\begin{bmatrix}\dot{\mathbf{x}}_1(t)\\ \dot{\mathbf{x}}_2(t)\end{bmatrix} = \begin{bmatrix}-\mathbf{x}_1\\ \mathbf{x}_2 - \mathbf{x}_1^2\end{bmatrix}.$$
- Introduce a variable $\mathbf{x}_3 := \mathbf{x}_1^2$. This gives $\dot{\mathbf{x}}_3 = 2\cdot\dot{\mathbf{x}}_1\mathbf{x}_1 = -2\cdot\mathbf{x}_1^2 = -2\cdot\mathbf{x}_3$.
- Lifted dynamics is
$$\begin{bmatrix}\dot{\mathbf{x}}_1(t)\\ \dot{\mathbf{x}}_2(t)\\ \dot{\mathbf{x}}_3(t)\end{bmatrix} = \begin{bmatrix}-\mathbf{x}_1\\ \mathbf{x}_2 - \mathbf{x}_3\\ -2\cdot\mathbf{x}_3\end{bmatrix}.$$

Example 1 [Lusch et al. '18]

- Consider nonlinear system.

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\mathbf{x}_1 \\ \mathbf{x}_2 - \mathbf{x}_1^2 \end{bmatrix}.$$

- Introduce a variable $\mathbf{x}_3 := \mathbf{x}_1^2$. This gives $\dot{\mathbf{x}}_3 = 2 \cdot \dot{\mathbf{x}}_1 \mathbf{x}_1 = -2 \cdot \mathbf{x}_1^2 = -2 \cdot \mathbf{x}_3$.
- Lifted dynamics is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \\ \dot{\mathbf{x}}_3(t) \end{bmatrix} = \begin{bmatrix} -\mathbf{x}_1 \\ \mathbf{x}_2 - \mathbf{x}_3 \\ -2 \cdot \mathbf{x}_3 \end{bmatrix}.$$

Example 2

- Consider a simple pendulum model:

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

- For this example, we do not have an exact linear representation due to continuum spectrum.

Our Ideal Goal

- Seek to rewrite a nonlinear system to a common structure and in finite dimension
  - In Koopman theory, the structure is linear systems and it is infinite dimensional

Our Ideal Goal

- Seek to rewrite a nonlinear system to a common structure and in finite dimension
  - In Koopman theory, the structure is linear systems and it is infinite dimensional

Lifting-Principle

Our Ideal Goal

- Seek to rewrite a nonlinear system to a common structure and in finite dimension
  - In Koopman theory, the structure is linear systems and it is infinite dimensional

Lifting-Principle

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

Our Ideal Goal

- Seek to rewrite a nonlinear system to a common structure and in finite dimension
  - In Koopman theory, the structure is linear systems and it is infinite dimensional

Lifting-Principle

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

Our Ideal Goal

- Seek to rewrite a nonlinear system to a common structure and in finite dimension
  - In Koopman theory, the structure is linear systems and it is infinite dimensional

Lifting-Principle

- McCormick proposed a convex relaxation to solve nonlinear non-convex optimization. [McCormick 1976]
- Key ingredient is lifting; the optimization problem in a higher-dimensional using auxiliary variables (can think of observables in Koopman theory).
- Similar ideas have been developed for learning dynamical systems.

Quadratic Lifting-Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models to Quadratic Systems 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.

Quadratic Lifting-Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models to Quadratic Systems 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 mapping $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in

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

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

Quadratic Lifting-Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models to Quadratic Systems 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 mapping $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in

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

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 developed, e.g., in [Savageau/Voit 1987] for control purposes.
- For model reduction for nonlinear systems [Gu 2009/2011, Benner/Breiten 2015, Kramer/Willcox 2022].

Quadratic Lifting-Principle for Dynamical Systems

Lifted Nonlinear Dynamical Models to Quadratic Systems 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 mapping $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$, and its inverse mapping $\mathcal{L}^\sharp : \mathbb{R}^m \to \mathbb{R}^n$, resulting in

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

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 developed, e.g., in [Savageau/Voit 1987] for control purposes.
- For model reduction for nonlinear systems [Gu 2009/2011, Benner/Breiten 2015, Kramer/Willcox 2022].
- Lifting in data-driven setting [Qian et al. 2019].

An illustration

- Consider a 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 Dynamicalf Systems

An illustration

- Consider a simple pendulum model:

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

- Let us define lifted coordinates (observables) and an inverse transformation:

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

An illustration

- Consider a simple pendulum model:

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

- Let us define lifted coordinates (observables) and an 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:

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

An illustration

- Consider a simple pendulum model:

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

- Let us define lifted coordinates (observables) and an inverse transformation:

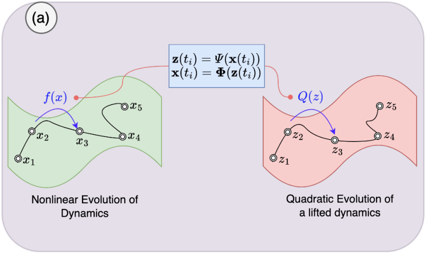
$$\mathcal{L} : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ \sin(x_2) \\ \cos(x_2) \end{bmatrix} =: \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \qquad \mathcal{L}^\sharp : \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \mapsto \begin{bmatrix} y_1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

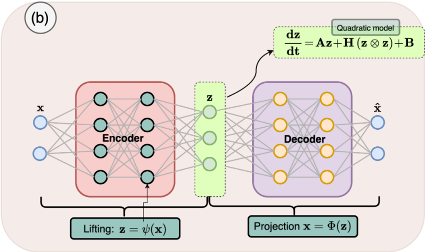
- Consequently, we can write the dynamics in the variables y_i as a quadratic system:

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

- Note that the inverse mapping is linear, and even continuous spectrum models can be easily written using appropriate observables.

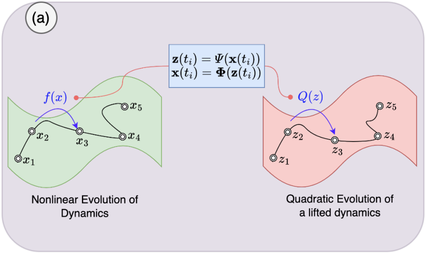
- Using observables—inspired by *Lifting-principle*—we can write nonlinear systems as quadratic systems which are
  - finite dimensional
  - given nonlinear systems, lifted observables are easy to determine

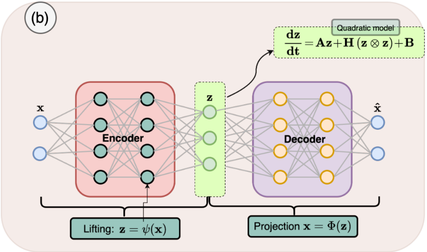




Lifting-Principle for Dynamical Systems

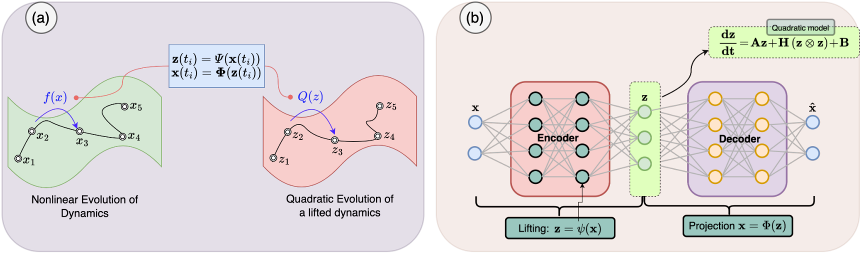
- Using observables—inspired by *Lifting-principle*—we can write nonlinear systems as quadratic systems which are
  - finite dimensional
  - given nonlinear systems, lifted observables are easy to determine





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

- Using observables—inspired by *Lifting-principle*—we can write nonlinear systems as quadratic systems which are
  - finite dimensional
  - given nonlinear systems, 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 **(G./Benner 2024)**

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 **(G./Benner 2024)**

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

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

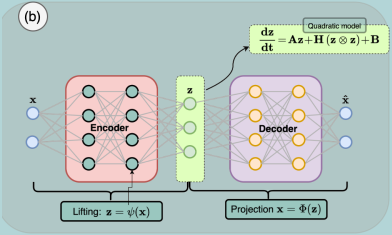
Problem Statement **(G./Benner 2024)**

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

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

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



Lifting-Principle for Dynamical Systems

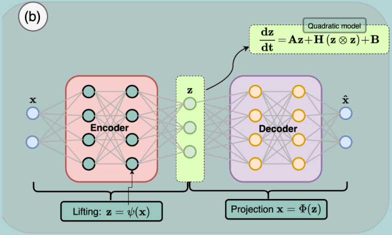
Problem Statement **(G./Benner 2024)**

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

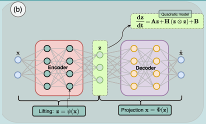
$$\dot{\mathbf{z}}(t) = \mathbf{A}\mathbf{z}(t) + \mathbf{H}\left(\mathbf{z}(t) \otimes \mathbf{z}(t)\right) + \mathbf{B} =: \mathcal{Q}(\mathbf{z}),$$
$$\mathbf{x}(t) = \phi(\mathbf{z}(t)).$$

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



Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:



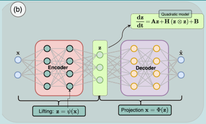
$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

Lifting-Principle for Dynamical Systems

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$



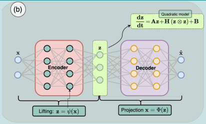
- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right).$$

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$



- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right).$$

- Derivative loss for $\dot{\mathbf{z}}$:

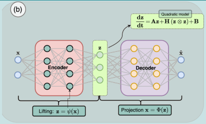
$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\nabla_{\mathbf{x}}\Psi(\mathbf{x}(t_i))\dot{\mathbf{x}}(t_i) - (\mathbf{A}\mathbf{z}(t_i) + \mathbf{H}\left(\mathbf{z}(t_i)\otimes\mathbf{z}(t_i)\right) + \mathbf{B})\|$$

with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$



- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right).$$

- Derivative loss for $\dot{\mathbf{z}}$:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\nabla_{\mathbf{x}}\Psi(\mathbf{x}(t_i))\dot{\mathbf{x}}(t_i) - (\mathbf{A}\mathbf{z}(t_i) + \mathbf{H}\left(\mathbf{z}(t_i) \otimes \mathbf{z}(t_i)\right) + \mathbf{B})\|$$

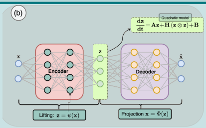
with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

- Combining all these elements, we can have weighted total loss for training:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{encdec}} + \lambda_2 \mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \lambda_3 \mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}},$$

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:



$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right).$$

– **Things do not work, when we try to integrate and make prediction.**

- Derivative loss for $\dot{\mathbf{z}}$:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\nabla_{\mathbf{x}}\Psi(\mathbf{x}(t_i))\dot{\mathbf{x}}(t_i) - (\mathbf{A}\mathbf{z}(t_i) + \mathbf{H}\left(\mathbf{z}(t_i)\otimes\mathbf{z}(t_i)\right) + \mathbf{B})\|$$

with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

- Combining all these elements, we can have weighted total loss for training:

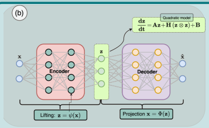
$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{encdec}} + \lambda_2\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \lambda_3\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}},$$

Lifting-Principle for Dynamical Systems

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$



- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{Az} + \mathbf{H}\left(\mathbf{z} \otimes \mathbf{z}\right) + \mathbf{B}\right).$$

- Derivative loss for $\dot{\mathbf{z}}$:

with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

– Things do not work, when we try to integrate and make prediction.
– These are continuous dynamical systems; thus, in the formulation none of the properties are included.

- Combining all these elements, we can have weighted total loss for training:

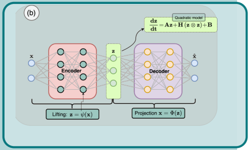
$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{encdec}} + \lambda_2 \mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \lambda_3 \mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}},$$

Lifting-Principle for Dynamical Systems

Naive Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$



- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}(\mathbf{z}\otimes\mathbf{z}) + \mathbf{B}\right).$$

- Derivative loss for $\dot{\mathbf{z}}$:

– Things do not work, when we try to integrate and make prediction.

– These are continuous dynamical systems; thus, in the formulation none of the properties are included.

– We are interested in stability properties of dynamical systems.

- Combining all these elements, we can have weighted total loss for training:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{encdec}} + \lambda_2 \mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \lambda_3 \mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}},$$

Stability of linear systems

A linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is asymptotically stable if and only if all the eigenvalues of $\mathbf{A}$ lie in left-half plane strictly. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$.

[1] $\min_{\mathbf{A}} \|\mathbf{x}(t_{i+1}) - \int_{t_i}^{t_{i+1}} \mathbf{A}\mathbf{x}(t)dt\|$

Stability of linear systems

A linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is asymptotically stable if and only if all the eigenvalues of $\mathbf{A}$ lie in left-half plane strictly. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$.

Inference of linear systems

- Given data $\dot{\mathbf{X}}$ and $\mathbf{X}$, a linear model can be inferred by solving

$$\min_{\mathbf{A}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} \right\| \quad \text{such that} \quad \Lambda\left(\mathbf{A}\right) \subset \mathbb{C}^{-}.$$

[1] $\min_{\mathbf{A}} \|\mathbf{x}(t_{i+1}) - \int_{t_i}^{t_{i+1}} \mathbf{A}\mathbf{x}(t)dt\|$

Stability of linear systems

A linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is asymptotically stable if and only if all the eigenvalues of $\mathbf{A}$ lie in left-half plane strictly. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$.

Inference of linear systems

- Given data $\dot{\mathbf{X}}$ and $\mathbf{X}$, a linear model can be inferred by solving

$$\min_{\mathbf{A}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} \right\| \quad \text{such that} \quad \Lambda\left(\mathbf{A}\right) \subset \mathbb{C}^{-}.$$

- Non-smooth optimization problem, which can be expensive; moreover, it is not straightforward to incorporate in integral forms of inference[1].

[1] $\min_{\mathbf{A}} \|\mathbf{x}(t_{i+1}) - \int_{t_i}^{t_{i+1}} \mathbf{A}\mathbf{x}(t)dt\|$

Stability of linear systems

A linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is asymptotically stable if and only if all the eigenvalues of $\mathbf{A}$ lie in left-half plane strictly. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$.

Inference of linear systems

- Given data $\dot{\mathbf{X}}$ and $\mathbf{X}$, a linear model can be inferred by solving

$$\min_{\mathbf{A}} \left\|\dot{\mathbf{X}} - \mathbf{A}\mathbf{X}\right\| \quad \text{such that} \quad \Lambda\left(\mathbf{A}\right) \subset \mathbb{C}^{-}.$$

- Non-smooth optimization problem, which can be expensive; moreover, it is not straightforward to incorporate in integral forms of inference[1].

Stable matrix parameterization [Gillis/Sharma '17]

Any stable matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be parameterized as follows:

$$\mathbf{A} = (\mathbf{J} - \mathbf{R})\mathbf{Q},$$

where $\mathbf{J} = -\mathbf{J}^\top$, $\mathbf{R} = \mathbf{R}^\top \succ 0$, and $\mathbf{Q} = \mathbf{Q}^\top \succ 0$.

[1] $\min_{\mathbf{A}} \|\mathbf{x}(t_{i+1}) - \int_{t_i}^{t_{i+1}} \mathbf{A}\mathbf{x}(t)dt\|$

Inference of Linear Stable Systems

Linear stable inference [G./Pontes/Benner '22]

- An inference problem for linear systems, ensuring stability

$$\min_{\mathbf{J},\mathbf{R},\mathbf{Q}} \left\|\dot{\mathbf{X}} - (\mathbf{J}-\mathbf{R})\mathbf{Q}\mathbf{X}\right\| \quad \text{such that} \quad \mathbf{J} = -\mathbf{J}^\top, \mathbf{R} = \mathbf{R}^\top \succ 0, \mathbf{Q} = \mathbf{Q}^\top \succ 0.$$

Inference of Linear Stable Systems

Linear stable inference [G./Pontes/Benner '22]

- An inference problem for linear systems, ensuring stability

$$\min_{\mathbf{J},\mathbf{R},\mathbf{Q}} \left\|\dot{\mathbf{X}} - (\mathbf{J}-\mathbf{R})\mathbf{Q}\mathbf{X}\right\| \quad \text{such that} \quad \mathbf{J} = -\mathbf{J}^\top, \mathbf{R} = \mathbf{R}^\top \succ 0, \mathbf{Q} = \mathbf{Q}^\top \succ 0.$$

- Alternatively, by relaxing strict positive constraints on $\mathbf{R}$ and $\mathbf{Q}$, we can write

$$\min_{\tilde{\mathbf{J}},\tilde{\mathbf{R}},\tilde{\mathbf{Q}}} \left\|\dot{\mathbf{X}} - \left((\tilde{\mathbf{J}}-\tilde{\mathbf{J}}^\top) - \tilde{\mathbf{R}}\tilde{\mathbf{R}}^\top\right)\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^\top\mathbf{X}\right\|.$$

Inference of Linear Stable Systems

Linear stable inference [G./Pontes/Benner '22]

- An inference problem for linear systems, ensuring stability

$$\min_{\mathbf{J},\mathbf{R},\mathbf{Q}} \left\| \dot{\mathbf{X}} - (\mathbf{J} - \mathbf{R})\mathbf{Q}\mathbf{X} \right\| \quad \text{such that} \quad \mathbf{J} = -\mathbf{J}^\top, \mathbf{R} = \mathbf{R}^\top \succ 0, \mathbf{Q} = \mathbf{Q}^\top \succ 0.$$

- Alternatively, by relaxing strict positive constraints on $\mathbf{R}$ and $\mathbf{Q}$, we can write

$$\min_{\tilde{\mathbf{J}},\tilde{\mathbf{R}},\tilde{\mathbf{Q}}} \left\| \dot{\mathbf{X}} - \left((\tilde{\mathbf{J}} - \tilde{\mathbf{J}}^\top) - \tilde{\mathbf{R}}\tilde{\mathbf{R}}^\top \right) \tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^\top \mathbf{X} \right\|.$$

- We solve these optimization problems using gradient descent.

- Recall, for inference of quadratic systems, we require to solve

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\|,$$

resulting $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) + \mathbf{B}$.

- Recall, for inference of quadratic systems, we require to solve

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\|,$$

resulting $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) + \mathbf{B}$.

Local stability of quadratic systems

- Consider a quadratic system, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{H}(\mathbf{x} \otimes \mathbf{x})$. If $\Lambda(\mathbf{A}) \subset \mathbb{C}^{-}$, then the system is locally asymptotically stable. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$ if $\mathbf{x}(0) \in \mathcal{B}(0, r)$.
- Moreover, $\mathbf{V}(\mathbf{x}) = \mathbf{x}^{\top}\mathbf{Q}\mathbf{x}$ is a Lyapunov function.

- Recall, for inference of quadratic systems, we require to solve

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\|,$$

resulting $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) + \mathbf{B}$.

Local stability of quadratic systems

- Consider a quadratic system, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{H}(\mathbf{x} \otimes \mathbf{x})$. If $\Lambda(\mathbf{A}) \subset \mathbb{C}^{-}$, then the system is locally asymptotically stable. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$ if $\mathbf{x}(0) \in \mathcal{B}(0, r)$.
- Moreover, $\mathbf{V}(\mathbf{x}) = \mathbf{x}^{\top}\mathbf{Q}\mathbf{x}$ is a Lyapunov function.

Inference of locally stable quadratic systems [Sawant et al. '23, G./Pontes/Benner '23]

- We, thus, require to solve the following constraints:

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\| \quad \text{such that} \quad \Lambda(\mathbf{A}) \in \mathbb{C}^{-}.$$

- Recall, for inference of quadratic systems, we require to solve

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\|,$$

resulting $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) + \mathbf{B}$.

Local stability of quadratic systems

- Consider a quadratic system, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{H}(\mathbf{x} \otimes \mathbf{x})$. If $\Lambda(\mathbf{A}) \subset \mathbb{C}^-$, then the system is locally asymptotically stable. This implies, $\lim_{t\to\infty} \mathbf{x}(t) = 0$ if $\mathbf{x}(0) \in \mathcal{B}(0, r)$.
- Moreover, $\mathbf{V}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$ is a Lyapunov function.

Inference of locally stable quadratic systems [Sawant et al. '23, G./Pontes/Benner '23]

- We, thus, require to solve the following constraints:

$$\min_{\mathbf{A},\mathbf{H},\mathbf{B}} \left\| \dot{\mathbf{X}} - \mathbf{A}\mathbf{X} - \mathbf{H}\mathbf{X}^{\otimes} - \mathbf{B} \right\| \quad \text{such that} \quad \Lambda(\mathbf{A}) \in \mathbb{C}^-.$$

- We already know how to parameterize stable matrices, i.e., $\mathbf{A} = (\mathbf{J} - \mathbf{R})\mathbf{Q}$.

Globally Asymptotically Stable Quadratic Systems

- Local stability is not sufficient in many practical cases.

Globally Asymptotically Stable Quadratic Systems

- Local stability is not sufficient in many practical cases.
- Hence, our interest are in globally asymptotically stability.

Globally Asymptotically Stable Quadratic Systems

- Local stability is not sufficient in many practical cases.
- Hence, our interest are in globally asymptotically stability.

Energy preserving quadratic nonlinearity **e.g., [Lorenz '63, Schlegel/Noack '15]**

- A quadratic non-linearity $\mathbf{H}(\mathbf{x} \otimes \mathbf{x})$ is said to be energy preserving if $\mathbf{x}^\top \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) = 0$ for all $\mathbf{x}$.
- Such non-linearity naturally appears, e.g., in flow problems (Navier-Stokes equations).

Globally Asymptotically Stable Quadratic Systems

- Local stability is not sufficient in many practical cases.
- Hence, our interest are in globally asymptotically stability.

Energy preserving quadratic nonlinearity **e.g., [Lorenz '63, Schlegel/Noack '15]**

- A quadratic non-linearity $\mathbf{H}(\mathbf{x} \otimes \mathbf{x})$ is said to be energy preserving if $\mathbf{x}^\top \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) = 0$ for all $\mathbf{x}$.
- Such non-linearity naturally appears, e.g., in flow problems (Navier-Stokes equations).

Globally asymptotically stable quadratic systems

- Consider a quadratic system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{H}(\mathbf{x} \otimes \mathbf{x})$ with energy-preserving non-linearity. If the matrix $\mathbf{A}$ is stable, then it is globally asymptotically stable.
- Moreover, the function $\mathbf{V}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$ is a Lyapunov function.

Globally Asymptotically Stable Quadratic Systems

- Local stability is not sufficient in many practical cases.
- Hence, our interest are in globally asymptotically stability.

Energy preserving quadratic nonlinearity e.g., [Lorenz '63, Schlegel/Noack '15]

- A quadratic non-linearity $\mathbf{H}(\mathbf{x} \otimes \mathbf{x})$ is said to be energy preserving if $\mathbf{x}^\top \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) = 0$ for all $\mathbf{x}$.
- Such non-linearity naturally appears, e.g., in flow problems (Navier-Stokes equations).

Globally asymptotically stable quadratic systems

- Consider a quadratic system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{H}(\mathbf{x} \otimes \mathbf{x})$ with energy-preserving non-linearity. If the matrix $\mathbf{A}$ is stable, then it is globally asymptotically stable.
- Moreover, the function $\mathbf{V}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$ is a Lyapunov function.

How to parameterize energy-preserving nonlinearity [G./Pontes/Benner '23]

If the matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ is of the following form:

$$\mathbf{H} = \left[\mathbf{H}_1, \dots, \mathbf{H}_n\right],$$

where $\mathbf{H}_i = -\mathbf{H}_i^\top$, then $\mathbf{x}^\top \mathbf{H}(\mathbf{x} \otimes \mathbf{x}) = 0$, or the non-linearity is energy preserving.

Lifting-Principle for Dynamical Systems

Stability-Enforced Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

Stability-Enforced Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\mathsf{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right),$$

where $\mathbf{A} = (\mathbf{J} - \mathbf{R})$ with $\mathbf{J} = -\mathbf{J}^\top$, $\mathbf{R} = \mathbf{R}^\top \succ 0$, and $\mathbf{H} = \left[\mathbf{H}_1, \ldots, \mathbf{H}_n\right]$ with $\mathbf{H}_i = -\mathbf{H}_i^\top$.

Lifting-Principle for Dynamical Systems

Stability-Enforced Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right),$$

where $\mathbf{A} = (\mathbf{J} - \mathbf{R})$ with $\mathbf{J} = -\mathbf{J}^\top$, $\mathbf{R} = \mathbf{R}^\top \succ 0$, and $\mathbf{H} = \left[\mathbf{H}_1, \ldots, \mathbf{H}_n\right]$ with $\mathbf{H}_i = -\mathbf{H}_i^\top$.

- Derivative loss for $\dot{\mathbf{z}}$:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\nabla_{\mathbf{x}}\Psi(\mathbf{x}(t_i))\dot{\mathbf{x}}(t_i) - (\mathbf{A}\mathbf{z}(t_i) + \mathbf{H}\left(\mathbf{z}(t_i)\otimes\mathbf{z}(t_i)\right) + \mathbf{B})\|$$

with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

Stability-Enforced Loss for Training (G./Benner 2024)

- Auto-encoder type loss:

$$\mathcal{L}_{\text{encdec}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\mathbf{x}(t_i) - \Phi(\Psi(\mathbf{x}(t_i)))\|.$$

- Derivative loss for $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\dot{\mathbf{z}} = \nabla_{\mathbf{z}}\Phi(\mathbf{z})\left(\mathbf{A}\mathbf{z} + \mathbf{H}\left(\mathbf{z}\otimes\mathbf{z}\right) + \mathbf{B}\right),$$

where $\mathbf{A} = (\mathbf{J} - \mathbf{R})$ with $\mathbf{J} = -\mathbf{J}^\top$, $\mathbf{R} = \mathbf{R}^\top \succ 0$, and $\mathbf{H} = \left[\mathbf{H}_1, \ldots, \mathbf{H}_n\right]$ with $\mathbf{H}_i = -\mathbf{H}_i^\top$.

- Derivative loss for $\dot{\mathbf{z}}$:

$$\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \|\nabla_{\mathbf{x}}\Psi(\mathbf{x}(t_i))\dot{\mathbf{x}}(t_i) - \left(\mathbf{A}\mathbf{z}(t_i) + \mathbf{H}\left(\mathbf{z}(t_i)\otimes\mathbf{z}(t_i)\right) + \mathbf{B}\right)\|$$

with $\mathbf{z}(t_i) = \Psi(\mathbf{x}(t_i))$.

- Combining all these elements, we can have weighted total loss for training:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{encdec}} + \lambda_2\mathcal{L}_{\dot{\mathbf{x}}\dot{\mathbf{z}}} + \lambda_3\mathcal{L}_{\dot{\mathbf{z}}\dot{\mathbf{x}}},$$

Illustrative Examples

Pendulum example

- Governing equation is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\mathbf{x}_2(t)) - 0.025\mathbf{x}_1(t) \\ \mathbf{x}_1(t) \end{bmatrix},$$

Illustrative Examples

Pendulum example

- Governing equation is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\mathbf{x}_2(t)) - 0.025\mathbf{x}_1(t) \\ \mathbf{x}_1(t) \end{bmatrix},$$

- The data are collected for time interval $[0, 25]s$ for 50 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-3, 3]$.

Pendulum example

- Governing equation is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\mathbf{x}_2(t)) - 0.025\mathbf{x}_1(t) \\ \mathbf{x}_1(t) \end{bmatrix},$$

- The data are collected for time interval $[0, 25]s$ for 50 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-3, 3]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 75]s$.

Pendulum example

- Governing equation is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\mathbf{x}_2(t)) - 0.025\mathbf{x}_1(t) \\ \mathbf{x}_1(t) \end{bmatrix},$$

- The data are collected for time interval $[0, 25]s$ for 50 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-3, 3]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 75]s$.

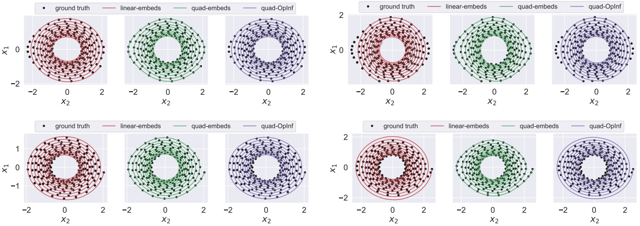


Figure: Nonlinear pendulum example: A comparison of the trajectories.

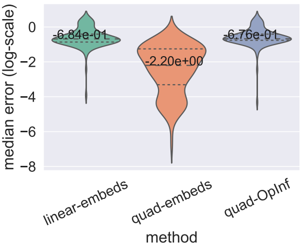
Illustrative Examples

Pendulum example

- Governing equation is

$$\begin{bmatrix}\dot{\mathbf{x}}_1(t)\\ \dot{\mathbf{x}}_2(t)\end{bmatrix} = \begin{bmatrix}-\sin(\mathbf{x}_2(t)) - 0.025\mathbf{x}_1(t)\\ \mathbf{x}_1(t)\end{bmatrix},$$

- The data are collected for time interval $[0, 25]s$ for 50 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-3, 3]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 75]s$.



Dissipative Lotka-Volterra example

- Governing equation is

$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}-e^{\mathbf{p}} - 0.05 \cdot \mathbf{q} + 1\\ e^{\mathbf{q}} - 0.05 \cdot \mathbf{p} - 2\end{bmatrix},$$

Dissipative Lotka-Volterra example

- Governing equation is
$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}-e^{\mathbf{p}} - 0.05 \cdot \mathbf{q} + 1\\ e^{\mathbf{q}} - 0.05 \cdot \mathbf{p} - 2\end{bmatrix},$$
- The data are collected for time interval $[0, 10]s$ for 10 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-1.5, 1.5]$.

Illustrative Examples

Dissipative Lotka-Volterra example

- Governing equation is

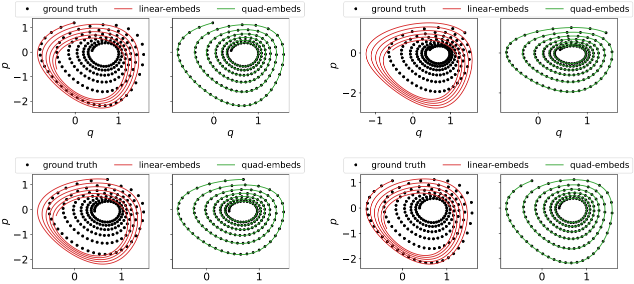
$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}-e^{\mathbf{p}} - 0.05 \cdot \mathbf{q} + 1\\ e^{\mathbf{q}} - 0.05 \cdot \mathbf{p} - 2\end{bmatrix},$$

- The data are collected for time interval $[0, 10]s$ for 10 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-1.5, 1.5]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 30]s$.

Illustrative Examples

Dissipative Lotka-Volterra example

- Governing equation is
$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}-e^{\mathbf{p}} - 0.05\cdot\mathbf{q} + 1\\ e^{\mathbf{q}} - 0.05\cdot\mathbf{p} - 2\end{bmatrix},$$
- The data are collected for time interval $[0, 10]s$ for 10 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-1.5, 1.5]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 30]s$.



Dissipative Lotka-Volterra example

- Governing equation is
$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \dot{\mathbf{p}}(t)\end{bmatrix} = \begin{bmatrix}-e^{\mathbf{p}} - 0.05\cdot\mathbf{q} + 1\\ e^{\mathbf{q}} - 0.05\cdot\mathbf{p} - 2\end{bmatrix},$$
- The data are collected for time interval $[0, 10]s$ for 10 random initial conditions with $\{\mathbf{x}_1, \mathbf{x}_2\} \in [-1.5, 1.5]$.
- We learn 3-dimensional quadratic model ($\equiv 3$ observables).
- We test using 100 random initial conditions for time interval $[0, 30]s$.

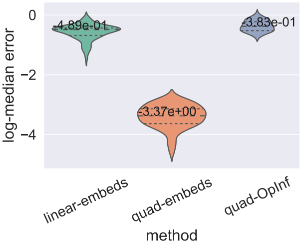


Figure: Dissipative Lotka-Volterra example: A comparison of the trajectories

A High-dimensional Example: Nonlinear Burgers' Equations

- Governing equations:

$$
\begin{aligned}
&u_t + uu_x + u^3 u_x = u_{xx}, \quad \text{with} \quad x \in (0,1) \text{ and } t \in (0,T),\\
&u(0,\cdot) = 0, \quad \text{and } u(1,\cdot) = 0,\\
&u(x,0) = 10 \cdot \sin(\pi x \cdot f) x(1-x),
\end{aligned}
$$

A High-dimensional Example: Nonlinear Burgers' Equations

- Governing equations:

$$u_t + uu_x + u^3 u_x = u_{xx}, \quad \text{with} \quad x \in (0,1) \text{ and } t \in (0,T),$$
$$u(0,\cdot) = 0, \quad \text{and } u(1,\cdot) = 0,$$
$$u(x,0) = 10 \cdot \sin(\pi x \cdot f)x(1-x),$$

- Since it is PDEs, thus the data are spatially coherent, we use convolutions autoencoder.

Numerical Examples

A High-dimensional Example: Nonlinear Burgers' Equations

- Governing equations:

$$\begin{aligned} &u_t + uu_x + u^3 u_x = u_{xx}, \quad \text{with} \quad x \in (0,1) \text{ and } t \in (0,T), \\ &u(0,\cdot) = 0, \quad \text{and } u(1,\cdot) = 0, \\ &u(x,0) = 10 \cdot \sin(\pi x \cdot f) x(1-x), \end{aligned}$$

- Since it is PDEs, thus the data are spatially coherent, we use convolutions autoencoder.
- We learn 4-dimensional quadratic model ($\equiv 4$ observables) in the latent space.

A High-dimensional Example: Nonlinear Burgers' Equations

- Governing equations:

$$\begin{aligned} &u_t + uu_x + u^3u_x = u_{xx}, \quad \text{with} \quad x \in (0,1) \text{ and } t \in (0,T), \\ &u(0,\cdot) = 0, \quad \text{and } u(1,\cdot) = 0, \\ &u(x,0) = 10 \cdot \sin(\pi x \cdot f)x(1-x), \end{aligned}$$

- Since it is PDEs, thus the data are spatially coherent, we use convolutions autoencoder.
- We learn 4-dimensional quadratic model ($\equiv 4$ observables) in the latent space.

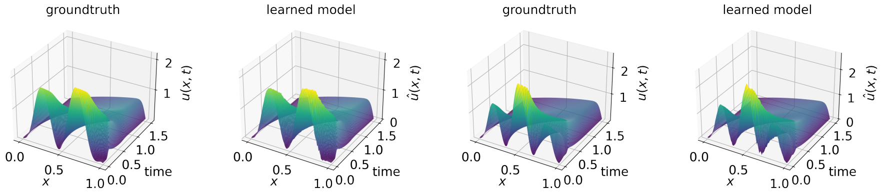


Figure: Burgers' equation: A comparison of the solutions.

Contribution

- Discussed Lifting-principle for nonlinear dynamical systems

Contribution

- Discussed Lifting-principle for nonlinear dynamical systems
- It allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).
    - $\rightsquigarrow$ Notion of quadratic-embeddings

Contribution

- Discussed Lifting-principle for nonlinear dynamical systems
- It allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).
  - $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust

Outlook

Contribution

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

  $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Outlook

Contribution

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

  $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extensions to parametric, and control cases

Outlook

Contribution

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

    $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extension to discrete systems
- Extensions to parametric, and control cases

Outlook

Contribution

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

  $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extension to discrete systems
- Extensions to parametric, and control cases

Outlook

Contribution

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

  $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extension to discrete systems
- Extensions to parametric, and control cases

Outlook

Contribution

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

  $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extension to discrete systems
- Extensions to parametric, and control cases

Outlook

Contribution

- Discussed Lifting-principle for nonlinear dynamical systems
- It allows us to write nonlinear systems as quadratic systems using observables (or lifted variables).
  - $\rightsquigarrow$ Notion of quadratic-embeddings
- Stability was a crucial factor to make things work with limited data and have robust
- Extension to Hamiltonian systems

Open work

- How to deal with unstable systems
- Extension to discrete systems
- Extensions to parametric, and control cases

Thank you for your attention!!

Selected References (alphabetical)

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. (2024).
Generalized quadratic-embeddings for nonlinear dynamics using deep learning.
Physica D: Nonlinear Phenomena, 463:134158.

Goyal, P., Duff, I. P., and Benner, P. (2023).
Guaranteed stable quadratic models and their applications in SINDy and operator inference.
arXiv preprint arXiv:2308.13819.

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.

Williams, M. O., Kevrekidis, I. G., and Rowley, C. W. (2015).
A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition.
J. Nonlinear Science, 25(6):1307–1346.