Data-driven control with Koopman Operators

Hakan Girgin

L2C course

10.06.2021

Koopman Operators: Theory

Dynamical systems can be expressed in discrete-time as $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ where $\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \in \mathbb{R}^{d_x}$ are respectively the current and next states, $\boldsymbol{u}_t \in \mathbb{R}^{d_u}$ is the current control command and the function $f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$ describes the evolution of the dynamical system

Koopman theory asserts that any nonlinear dynamical system can be expressed linearly in infinite dimensional Hilbert space. We define a discrete-time Koopman operator $\mathcal{K}$ as the linear operator which advances observation functions $\boldsymbol{\phi} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_l}$ of the state in time as in

$$\boldsymbol{\phi}(\boldsymbol{x}_{t+1}) = \mathcal{K} \circ \boldsymbol{\phi}(\boldsymbol{x}_t)$$

where d_l is the dimension of the output of the observation function.

Finite approximation:

$$\boldsymbol{K} \in \mathbb{R}^{d_l \times d_l}$$

Koopman Operators: Example of slow manifold dynamics

An example in continuous time dynamics:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \mu x_1 \\ \lambda(x_2 - x_1^2) \end{bmatrix}}_{\mathbf{f}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\tilde{\boldsymbol{B}}} u \Longrightarrow \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix} = \underbrace{\begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}}_{\boldsymbol{\phi}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{\boldsymbol{B}} u$$

Koopman Operators: Related Work I

- Some methods assume linear observables $\phi(x) = x$, while others use predefined observation function dictionary for such as polynomial, radial, Fourier basis functions.
- Considering an autonomous nonlinear dynamical system:

$$x_{t+1} = f(x_t)$$

we can find such an approximation of the Koopman operator by setting a least-squares problem as

$$\min_{K} \sum_{t=1}^{T} \left\| \phi(x_{t+1}) - K\phi(x_t) \right\|^2$$

- Some methods parameterize the observation functions with an unknown parameter θ which is learned together with the Koopman operator at the same time.

$$\min_{K,\theta} \sum_{t=1}^{T} \left\| \phi_\theta(x_{t+1}) - K\phi_\theta(x_t) \right\|^2$$

Challenges

1. **What are the observables?**
2. **What are the A and B matrices?**
3. **How to learn the dynamics model?**
4. **How to learn the dynamics model for control?**

Koopman Operators: Related Work II

- Many of them include also an exogenous control input to transform the problem to a more general version as in

$$\min_{\boldsymbol{K},\boldsymbol{\theta}} \sum_{t=1}^{T} \left\| \boldsymbol{\phi_\theta}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_{t+1}) - \boldsymbol{K}\boldsymbol{\phi_\theta}(\boldsymbol{x}_t, \boldsymbol{u}_t) \right\|^2$$

- Although this formulation in helps to determine a better approximation for more complicated dynamics models, it does not admit a linear form that the linear control theory works on. So many of methods investigated:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{\theta}} \sum_{t=1}^{T} \left\| \boldsymbol{\phi_\theta}(\boldsymbol{x}_{t+1}) - \boldsymbol{A}\boldsymbol{\phi_\theta}(\boldsymbol{x}_t) - \boldsymbol{B}\boldsymbol{u}_t \right\|^2$$

Challenges

1. **What are the observables?**
2. **What are the A and B matrices?**
3. **How to learn the dynamics model?**
4. **How to learn the dynamics model for control?**

- Some work investigate the multi-step prediction errors as in the following as opposed to the previous single-step prediction errors minimization

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{\theta}} \sum_{t=1}^{T} \left\| \boldsymbol{\phi_\theta}(\boldsymbol{x}_{t+1}) - \boldsymbol{A}^{t+1}\boldsymbol{\phi_\theta}(\boldsymbol{x}_0) - \sum_{\tau=0}^{t} \boldsymbol{A}^{\tau}\boldsymbol{B}\boldsymbol{u}_{t-\tau} \right\|^2 .$$

Koopman Operators: Related Work III

- The linear approximation of the dynamics is appealing as it allows us to use well-established model-based linear control techniques such as LQR.

- If the dynamics model is given as:

$$\boldsymbol{\phi_\theta}(\boldsymbol{x}_{t+1}) = \boldsymbol{A}\boldsymbol{\phi_\theta}(\boldsymbol{x}_t) + \boldsymbol{B}\boldsymbol{u}_t$$

and if the cost is:

$$c = \boldsymbol{\phi_\theta}(\boldsymbol{x}_t)^\top \boldsymbol{Q}\boldsymbol{\phi_\theta}(\boldsymbol{x}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}\boldsymbol{u}_t$$

then this is an LQR problem.

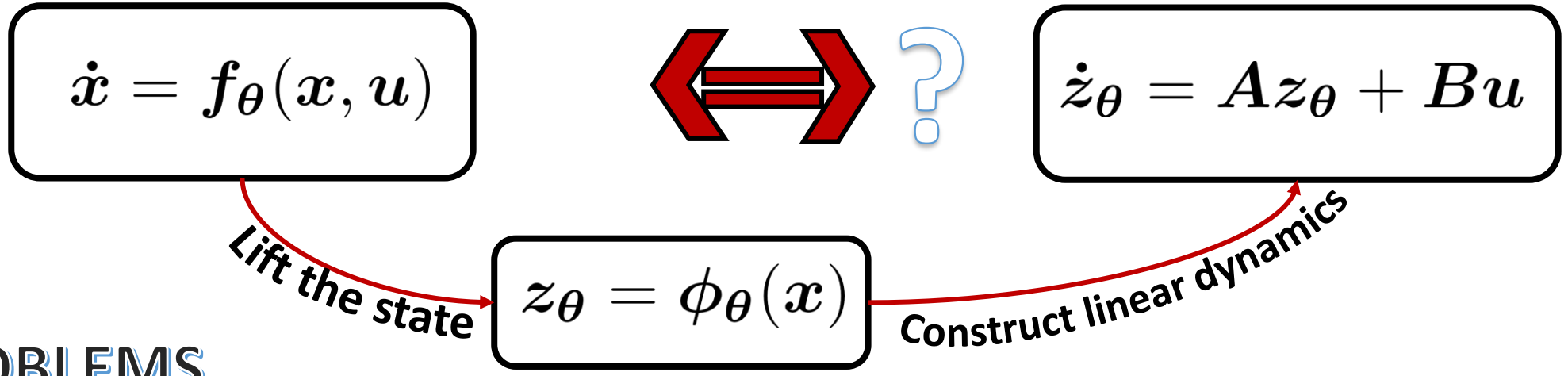
Challenges

1. **What are the observables?**
2. **What are the A and B matrices?**
3. **How to learn the dynamics model?**
4. **How to learn the dynamics model for control?**

However: how to design a cost function with respect to observables instead of original variables (state)?

Answer: Include the state itself in the new observables: $\boldsymbol{\psi}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\phi}(\boldsymbol{x}) \end{bmatrix}$

Koopman Operators: Approximation of the dynamics



PROBLEMS

Control affine dynamics: (e.g. manipulator dynamics) $\dot{x} = f(x) + g(x)u$

Multiplicative terms of **x** and **u** are most of the time ignored.

$z_\theta = \phi_\theta(x)$

Basis functions are chosen **either** manually requiring a large amount of choices or prior knowledge, **or** using deep networks requiring a large amount of data and a good regression cost.

If the regression cost is **one-step ahead prediction squared error,** the resulting model may diverge as we perform trajectory rollouts, if it is **multi-step ahead prediction squared error,** the problem is insanely nonconvex.

Example: nD robot arm dynamics

Rigid body dynamics of a robot arm is written as:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{u}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the mass matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the centrifugal and coriolis matrix, $\boldsymbol{g}(\boldsymbol{q})$ is the gravity vector and $\boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}})$ are the other non-modeled nonlinearities of the system.

The state space representation is given as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\begin{bmatrix}\boldsymbol{q}\\ \dot{\boldsymbol{q}}\end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\dot{\boldsymbol{q}}\\ -\boldsymbol{M}^{-1}(\boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g} + \boldsymbol{f})\end{bmatrix}}_{\mathbf{f}(\boldsymbol{x})} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \boldsymbol{M}^{-1}\end{bmatrix}}_{\tilde{\boldsymbol{B}}(\boldsymbol{x})}\boldsymbol{u}$$

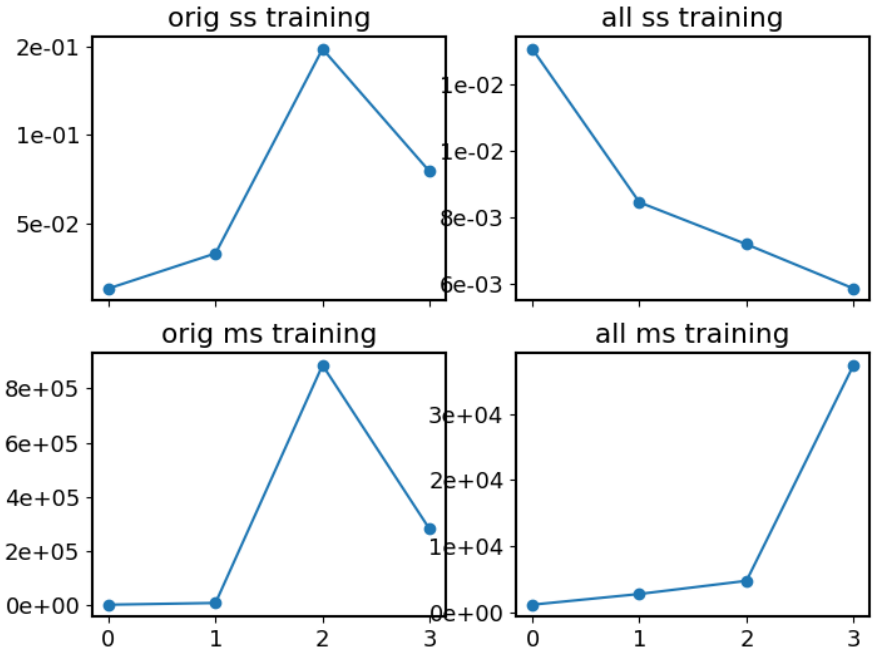
For a simple pendulum **M** is a constant → we can learn a good Koopman representation!

Example: 3D robot arm dynamics

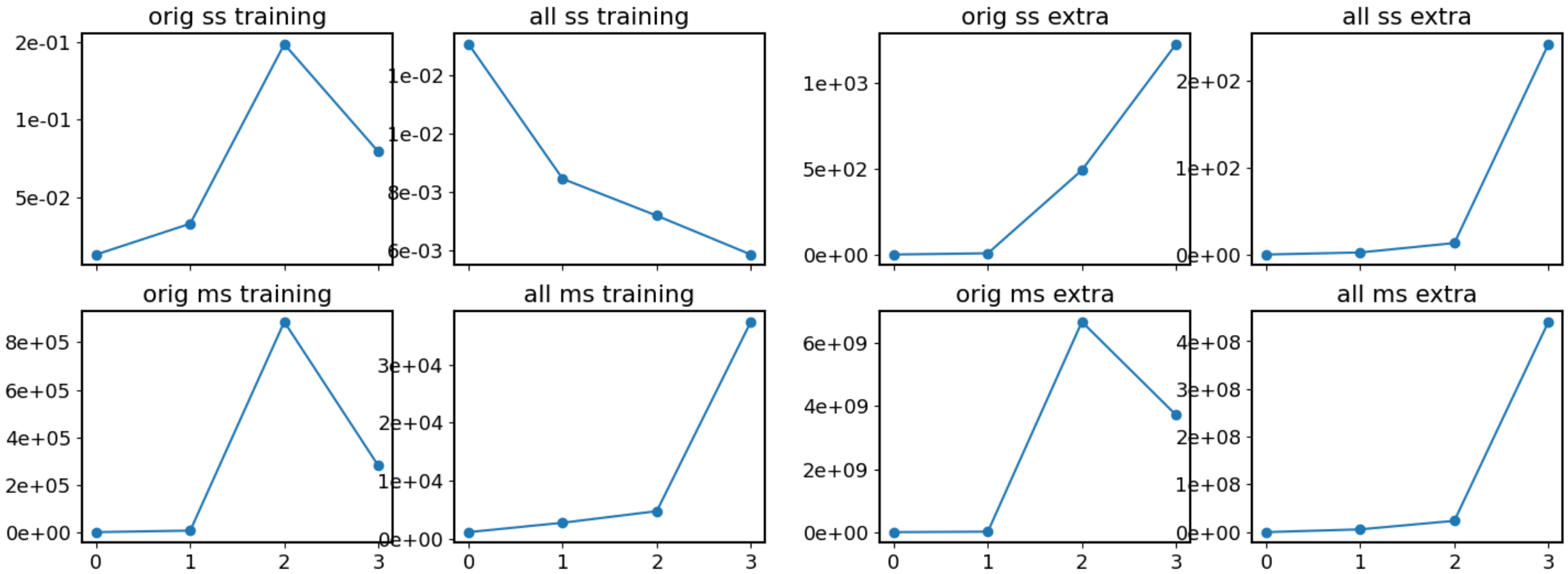
- Experiments done with known lifting functions (observables) chosen arbitrarily with increasing complexity.
- Since the observables are known, this is just a least-squares problem.
- The error is computed as first the single-step prediction errors on the training data (called *training*) and on the testing data (called *extra*), and then as the multi-step prediction errors over the whole horizon.
- *Orig* refers to the error only on the original state variables and *all* refers to the error on the overall lifted vector

$$\phi_0(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \end{bmatrix} \qquad \phi_2(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x}^2 \end{bmatrix}$$

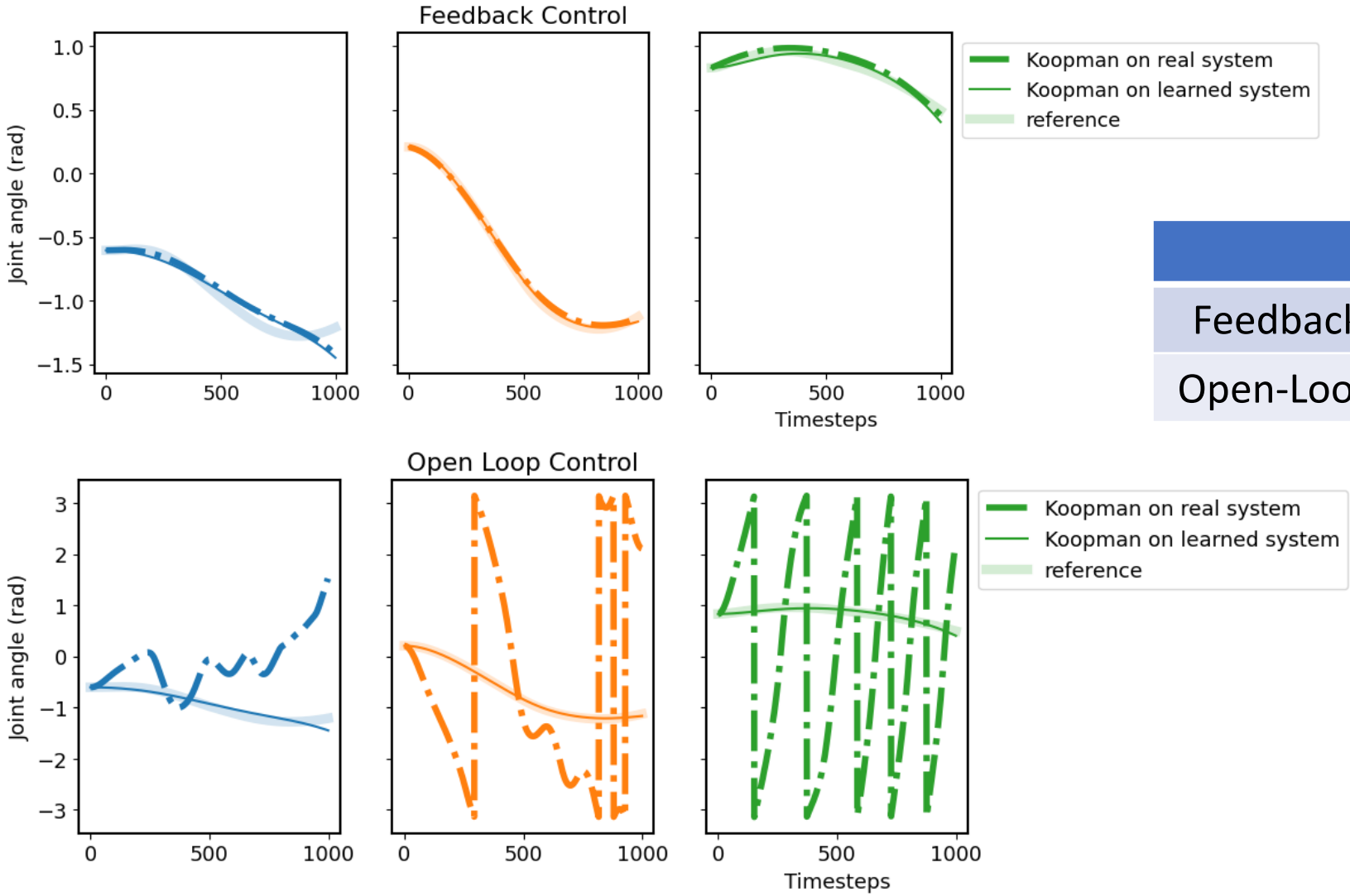
$$\phi_1(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \\ \cos(\boldsymbol{x}) \\ \sin(\boldsymbol{x}) \end{bmatrix} \qquad \phi_3(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \\ \cos(\boldsymbol{x}) \\ \sin(\boldsymbol{x}) \\ \boldsymbol{x}^2 \end{bmatrix}$$



Example: 3D robot arm dynamics - results



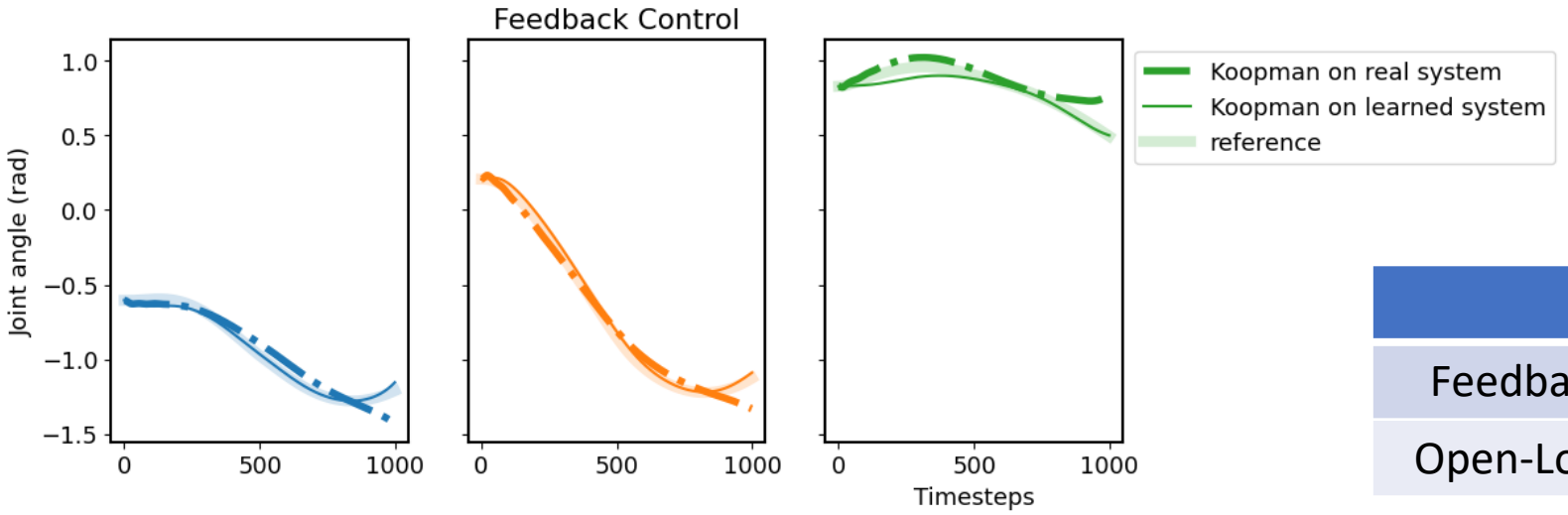
LQT applied to the learned model - 1



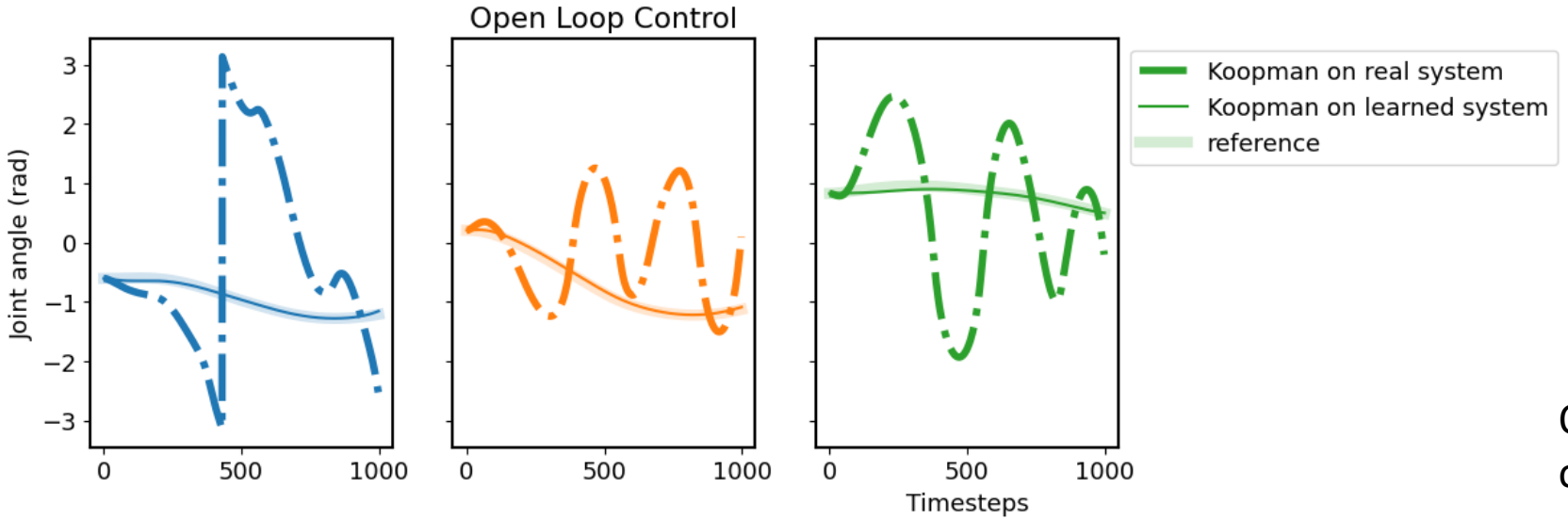
	Tracking Cost	Control Cost
Feedback	6.40E+00	8.34E+05
Open-Loop	9.90E+03	1.00E+06

$$\phi_0(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} \end{bmatrix}$$

LQT applied to the learned model - 2



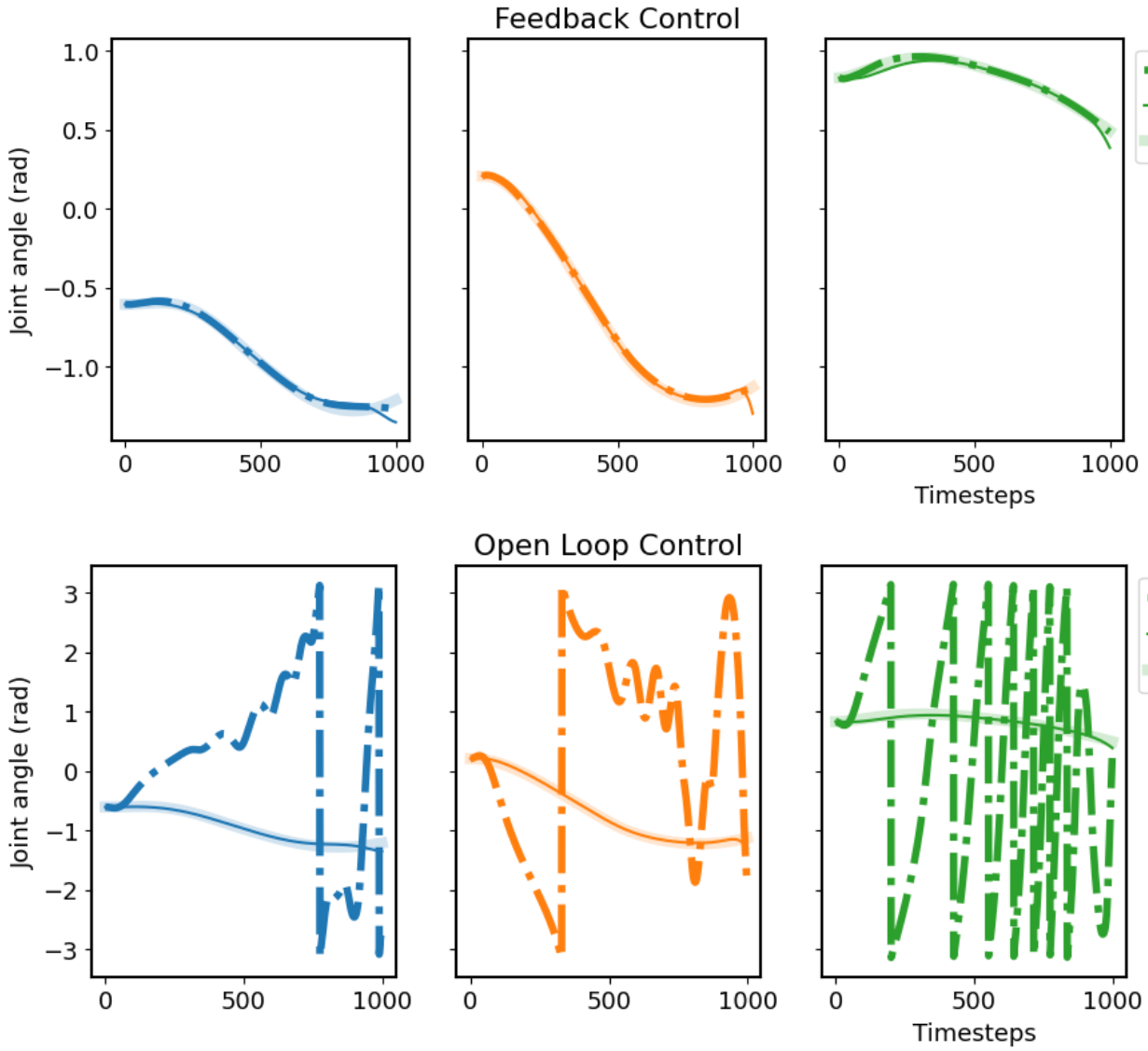
	Tracking Cost	Control Cost
Feedback	1.35E+01	1.06E+06
Open-Loop	5.99E+03	2.77E+06



$$\phi_1(x) = \begin{bmatrix} x \\ \cos(x) \\ \sin(x) \end{bmatrix}$$

Others → divergence or oscillations ..

Learning the lifting functions with MLP and apply LQT



	Tracking Cost	Control Cost
Feedback	4.11E-01	1.66E+06
Open-Loop	1.22E+04	1.13E+07

$$\phi_{\theta}(x) = \begin{bmatrix} x \\ \mathrm{MLP}(x) \end{bmatrix}$$

Comparison of these costs for each of three cases

	Tracking Cost	Control Cost
Feedback	6.40E+00	8.34E+05
Open-Loop	9.90E+03	1.00E+06

$$\phi_0(x) = [x]$$

	Tracking Cost	Control Cost
Feedback	1.35E+01	1.06E+06
Open-Loop	5.99E+03	2.77E+06

$$\phi_1(x) = \begin{bmatrix} x \\ \cos(x) \\ \sin(x) \end{bmatrix}$$

	Tracking Cost	Control Cost
Feedback	4.11E-01	1.66E+06
Open-Loop	1.22E+04	1.13E+07

$$\phi_\theta(x) = \begin{bmatrix} x \\ \mathrm{MLP}(x) \end{bmatrix}$$

Dynamical consistency

Control affine dynamics:
(e.g. manipulator dynamics)

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{g}(\boldsymbol{x}_t)\boldsymbol{u}_t$$

Problem was: Multiplicative terms of **x** and **u** are most of the time ignored.

Consider the following Koopman structure:

$$\dot{\boldsymbol{\psi}}_t = \boldsymbol{A}\boldsymbol{\psi}_t + \boldsymbol{B}(\boldsymbol{\psi}_t)\boldsymbol{u}_t$$

$$\begin{aligned}\dot{\boldsymbol{\psi}}(\boldsymbol{x}_t) &= \frac{\partial \boldsymbol{\psi}_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}\dot{\boldsymbol{x}}_t \\ &= \frac{\partial \boldsymbol{\psi}_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} f(\boldsymbol{x}_t) + \frac{\partial \boldsymbol{\psi}_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} g(\boldsymbol{x}_t)\boldsymbol{u}_t \\ &= \boldsymbol{A}\boldsymbol{\psi}_t + \boldsymbol{B}(\boldsymbol{\psi}_t)\boldsymbol{u}_t\end{aligned}$$

Then the constraints for dynamical consistency are:

$$\boldsymbol{A}\boldsymbol{\psi}_t = \frac{\partial \boldsymbol{\psi}_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} f(\boldsymbol{x}_t),$$

$$\boldsymbol{B}(\boldsymbol{\psi}_t)\boldsymbol{u}_t = \frac{\partial \boldsymbol{\psi}_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} g(\boldsymbol{x}_t)\boldsymbol{u}_t$$

$$\boldsymbol{\psi}_t = \boldsymbol{\psi}(\boldsymbol{x}_t) = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{\phi}(\boldsymbol{x}_t) \end{bmatrix}$$

Dynamical consistency –a proposed solution

Control affine dynamics:
(e.g. manipulator dynamics)

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{g}(\boldsymbol{x}_t)\boldsymbol{u}_t$$

Problem was: Multiplicative terms of **x** and **u** are most of the time ignored.

$$\dot{\boldsymbol{\psi}}_t = \boldsymbol{A}\boldsymbol{\psi}_t + \boldsymbol{B}(\boldsymbol{\psi}_t)\boldsymbol{u}_t$$

$$\boldsymbol{\psi}_t = \boldsymbol{\psi}(\boldsymbol{x}_t) = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{\phi}(\boldsymbol{x}_t) \end{bmatrix}$$

For the sake of simplicity, we choose the control matrix to be a linear function of the lifted state as

$$\boldsymbol{B}(\boldsymbol{\psi}_t) = \boldsymbol{W}\boldsymbol{\psi}_t\boldsymbol{p}^\top + \boldsymbol{C}, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{p} \in \mathbb{R}^{d_u}$ and $\boldsymbol{C} \in \mathbb{R}^{d\times d_u}$ are the unknown parameters of the control matrix to be learned.

Dynamical consistency - control

For the sake of simplicity, we choose the control matrix to be a linear function of the lifted state as

$$\boldsymbol{B}(\boldsymbol{\psi}_t) = \boldsymbol{W}\boldsymbol{\psi}_t\boldsymbol{p}^\top + \boldsymbol{C}, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{p} \in \mathbb{R}^{d_u}$ and $\boldsymbol{C} \in \mathbb{R}^{d\times d_u}$ are the unknown parameters of the control matrix to be learned.

Our Koopman dynamics model assumes the following final form:

$$\dot{\boldsymbol{\psi}}_t = f(\boldsymbol{\psi}_t, \boldsymbol{u}_t) = \boldsymbol{A}\boldsymbol{\psi}_t + (\boldsymbol{W}\boldsymbol{\psi}_t\boldsymbol{p}^\top + \boldsymbol{C})\boldsymbol{u}_t.$$

$$\frac{\partial f(\boldsymbol{\psi}_t, \boldsymbol{u}_t)}{\partial \boldsymbol{\psi}_t} = \boldsymbol{A} + \nabla\boldsymbol{B}\boldsymbol{u}_t,$$

$$\frac{\partial f(\boldsymbol{\psi}_t, \boldsymbol{u}_t)}{\partial \boldsymbol{u}_t} = (\boldsymbol{W}\boldsymbol{\psi}_t\boldsymbol{p}^\top + \boldsymbol{C}),$$

where $\nabla\boldsymbol{B}_{ijk} = \boldsymbol{W}_{ik}\boldsymbol{p}_j$. The derivative of the forward dynamics with respect to the lifted state is therefore a linear function of the control, whereas with respect to the control is a linear function of the lifted state.

Open problems

- Gathering meaningful experimental data so that the system can learn as fast as possible. → ideas around active learning, and iterative data gathering to learn better and faster with each collected trajectory.

- What should be done so that we extrapolate well even though we have small amount of data → ideas around a new structure of MLP that can help us, can also discuss this.

- What should be the training cost? Should there be equations related to dynamic consistency as constraints? Some papers deal with finding stable A and B matrices, but we need stabilizable A and B matrices, right?

- When we learn the Koopman dynamics model, we add another layer of complexity to predict also the evolution of the observables. However, they are functions of the state, therefore their evolution is related to the evolution of the state itself. Do we do something redundant?