2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Learning Koopman Operators with Control Using Bi-level Optimization

Daning Huang[1], Muhammad Bayu Prasetyo[2], Yin Yu[1], Junyi Geng[1]

Abstract— **The accurate modeling and control of nonlinear dynamical effects are crucial for numerous robotic systems. The Koopman formalism emerges as a valuable tool for linear control design in nonlinear systems within unknown environments. However, it still remains a challenging task to learn the Koopman operator with control from data, and in particular, the simultaneous identification of the Koopman linear dynamics and the mapping between the physical and Koopman states. Conventionally, the simultaneous learning of the dynamics and mapping is achieved via single-level optimization based on one-step or multi-step discrete-time predictions, but the learned model may lack model robustness, training efficiency, and/or long-term predictive accuracy. This paper presents a bi-level optimization framework that jointly learns the Koopman embedding mapping and Koopman dynamics with exact long-term dynamical constraints. Our formulation allows back-propagation in standard learning framework and the use of state-of-the-art optimizers, yielding more accurate and stable system prediction in long-time horizon over various applications compared to conventional methods.**

I. Introduction

Accurately modeling and controlling nonlinear dynamical effects is critical for robots, especially in challenging scenarios, e.g., aerial robotics [1], aerial manipulation tasks [2], offroad driving [3]. These scenarios often exhibit nonlinear effects, such as the coupling between translation and rotational motion, the self-motion and the manipulated objects, or the complex dynamics due to the environment, making control design difficult. Traditional methods, such as state feedback [4] or optimization-based control [5], require full knowledge of the system model to predict dynamics and design controllers. However, real-world effects such as wind gusts, boundary layer effects, rough terrain for mobile robots, and hidden dynamics of chaotic nonlinear effects are too complex to be fully captured, leading to poor control performance under these scenarios. As a result, new approaches are needed to model and control these systems accurately and efficiently, especially when faced with complex, uncertain, or rapidly changing environments.

Data-driven approaches have been successful in capturing unknown dynamics and patterns in complex systems [6], [7], allowing for accurate dynamics prediction. However, in many cases, these methods produce nonlinear models and hence require nonlinear control methods such as iterative Linear Quadratic Regulator (iLQR) [8] or Nonlinear Model Predictive Control (NMPC) [9] to achieve effective system control. These control methods can be computationally expensive as the system states increases, making them infeasible for real-time applications where fast and accurate control is essential. Although some Reinforcement Learning (RL) approaches [10], either model-based or model-free, can also achieve good performance on nonlinear control, they often suffer from sampling inefficiency and lack of generalizability. Therefore, there is a need for more efficient control methods that can be used in conjunction with data-driven techniques to enable real-time control of nonlinear systems.

Koopman operator has recently attracted growing interest and shown great potential to provide an elegant way of addressing the control problem under unknown dynamics [11]–[16]. It embeds the nonlinear system dynamics in a lifted, higher-dimensional space where the dynamics is governed by a linear but possibly infinite dimensional operator. Data-driven methods for identifying the Koopman models have gained considerable attention due to the strong expressive power and the rigorous operator-theoretic guarantees [17]–[19]. The learned linear system on the embedded space is readily amenable for linear or bilinear control techniques. Conventionally, the possible representation of the embedding is given as a predefined dictionary [20]–[22]. However, finding the mapping between the physical and Koopman states and selecting the embedding representation remains a challenging task, especially in terms of maintaining the predictive accuracy and generalizability.

There are some existing approaches focusing on learning the mapping expressed by deep neural networks [23], [24], and then apply linear control methods [25], [26]; these work show an improvement in the model compactness as well as predictive accuracy. However, due to the lack of capability to handle constraints in standard learning frameworks, the existing Koopman learning approaches often rely on a single-level unconstrained optimization formulation that attempts to minimize either only one step prediction errors, or multi-step prediction errors, typically with hand-tuned weights for penalty terms. Such approaches not only require significant amount of effort to tune the penalty term coefficients and loss components during practical implementation, but also suffer from increased computational overhead in backpropagation especially when multi-step prediction errors are optimized. As a result, the existing methods suffer from poor training efficiency, and the learned models may lack robustness to data noise and long-term predictive accuracy.

To overcome the above limitations, this paper proposes a bi-level optimization framework to learn the Koopman operator with control by jointly learning the embedding and the Koopman dynamics. Specifically, in the inner optimization,

[1] Department of Aerospace Engineering, Pennsylvania State University, University Park, PA, 16802, USA. {daning, yzy5368, jgeng}@psu.edu
[2] Department of Engineering Science and Mechanics, Pennsylvania State University, University Park, PA, 16802, USA. mbp5652@psu.edu

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10383799

we minimize the loss in the Koopman embedding space with exact constraints of long-horizon Koopman dynamics; in the outer optimization, we minimize the reconstruction loss in the original space with the inner optimization serving as constraints. This formulation removes the need to hand-tune weight parameters and exactly enforces Koopman dynamics during the learning process. Furthermore, our framework reformulates the Koopman dynamics using an integral form to eliminate the nested backpropagation calculations in the conventional formulations to boost up training efficiency while maintaining the compatibility with standard learning frameworks. Overall, the framework enforces the reproduction of dynamics over entire trajectory and thus mitigates the issues in data noise and the long-term prediction instability, and holds promise for a more accurate and numerically stable predictive model for control applications.

The paper is organized as follows. Section II presents a brief summary of Koopman operator with control, and the standard learning methods. In Section III, we provide the details in the formulation, analysis, and numerical algorithms of the proposed bi-level optimization framework. In Section IV, we present numerical examples to show the effectiveness of the proposed methodology in terms of training efficiency, predictive accuracy and generalizability. Finally, we conclude the work and point out possible future directions for further investigation in Section V.

II. Koopman Theory Preliminary

A. Basic Formulation

Koopman Bilinear Form (KBF) [16], [27] provides a means to globally bilinearize a control-affine system of the following form,

$$\dot{\mathbf{x}} = \mathbf{f}_0(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{f}_i(\mathbf{x}) u_i, \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{1}$$

where $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^r$ is the state vector, $\mathbf{u} = [u_1 \dots u_m]^\top \in \mathbb{R}^m$ is the input vector, $\mathbf{f}_0 : \mathbb{X} \to \mathbb{R}^r$ is the system dynamics, and $\mathbf{f}_i : \mathbb{X} \to \mathbb{R}^r$ are the control input coupling terms.

In the autonomous case [12], i.e., when $\mathbf{u} = 0$, the system generates a flow $\mathbf{F}_t(\mathbf{x}_0) = \mathbf{x}(t)$ from an initial condition $\mathbf{x}_0$. The continuous time Koopman operator $\mathcal{K}_t : \mathcal{F} \to \mathcal{F}$ is an infinite-dimensional linear operator such that $\mathcal{K}_t g = g \circ \mathbf{F}_t$ for all $g \in \mathcal{F}$, where $g : \mathbb{X} \to \mathbb{C}$ is a complex-valued observable function of the state vector $\mathbf{x}$, $\mathcal{F}$ is the function space of all possible observables, and $\circ$ denotes function composition. As a linear operator, $\mathcal{K}_t$ admits eigenpairs (λ, φ) such that $\mathcal{K}_t \varphi = \varphi \circ \mathbf{F}_t = e^{\lambda t}\varphi$ where $\lambda \in \mathbb{C}$ and $\varphi \in \mathcal{F}$ are the Koopman eigenvalue and Koopman eigenfunction, respectively.

The infinitesimal generator of $\mathcal{K}_t$ associated with $\mathbf{f}_0$, referred to as the Koopman generator, is defined as $L_{\mathbf{f}_0} = \lim_{t\to 0} \frac{\mathcal{K}_t - I}{t}$, where I is the identity operator, and turns out to be the Lie derivative $L_{\mathbf{f}_0} = \mathbf{f}_0 \cdot \nabla$, with eigenpair (λ, φ), $\dot{\varphi} = L_{\mathbf{f}_0}\varphi = \lambda\varphi$ Given a set of eigenpairs $\{(\lambda_i, \varphi_i)\}_{i=1}^n$, the *Koopman Canonical Transform* (KCT) [13] of the control-affine system (1) is $\dot{\boldsymbol{\varphi}} = \boldsymbol{\Lambda}\boldsymbol{\varphi} + \sum_{i=1}^m L_{\mathbf{f}_i}\boldsymbol{\varphi} u_i$, where $\boldsymbol{\Lambda} = \mathrm{diag}([\lambda_1, \cdots, \lambda_n])$, $\boldsymbol{\varphi} = [\varphi_1, \cdots, \varphi_n]$, and Lie derivatives for the control terms are $L_{\mathbf{f}_i} = \mathbf{f}_i \cdot \nabla$.

Suppose the set of eigenfunctions is sufficiently large, such that $\boldsymbol{\varphi}$ span an invariant space for $L_{\mathbf{f}_i}$, i.e., each of $L_{\mathbf{f}_i}$ can be represented using a $l \times l$ matrix $\mathbf{D}_i$, $L_{\mathbf{f}_i}\boldsymbol{\varphi} = \mathbf{D}_i\boldsymbol{\varphi}$, then the KCT can be brought to a bilinear form [16], $\dot{\boldsymbol{\varphi}} = \boldsymbol{\Lambda}\boldsymbol{\varphi} + \sum_{i=1}^m \mathbf{D}_i\boldsymbol{\varphi} u_i$. Often it is difficult to directly obtain the eigenfunctions of $\mathcal{K}_t$, and instead it is more convenient to learn the bilinear dynamics in a lifted coordinates via a mapping $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^n$, e.g., parametrized by a neural network, leading to the commonly used Koopman Bilinear Form (KBF) [16], [27],

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \sum_{i=1}^{m} \mathbf{B}_i \mathbf{z} u_i. \tag{2}$$

The eigendecomposition $\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^H$ reproduces the Koopman eigenvalues $\boldsymbol{\Lambda}$ and the Koopman eigenfunctions $\boldsymbol{\varphi} = \boldsymbol{\Psi}^H\mathbf{z}$, where $\square^H$ denotes conjugate transpose. The original states are recovered from an inverse mapping $\mathbf{x} = \boldsymbol{\phi}^{-1}(\mathbf{z}) \equiv \boldsymbol{\psi}(\mathbf{z})$. In standard KBF [13], $\boldsymbol{\phi}^{-1}$ is a linear mapping, but nonlinear versions of $\boldsymbol{\phi}^{-1}$ have also been proposed, e.g., in [19], [28].

B. Learning of KBF Model

The KBF model is usually identified from data, that is typically obtained at finite sampling rate. Suppose there are K sampled trajectories with inputs, each having $N+1$ steps; denote the dataset $\mathcal{D} = \{(\hat{\mathbf{x}}_i^{(k)}, \mathbf{u}_i^{(k)})_{i=0}^N\}_{k=1}^K$. The parameters to be learned from $\mathcal{D}$ include the system matrices

$$\boldsymbol{\Gamma} = [\mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_m]$$

and the encoder $\boldsymbol{\phi}$ and decoder $\boldsymbol{\psi}$.

A brutal force formulation to learn KBF is an ODE-constrained optimization problem. Without loss of generalizability, we show the case for one trajectory data.

$$\min_{\boldsymbol{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}} \mathcal{L}_e(\mathbf{z}; \boldsymbol{\phi}, \mathcal{D}) + \mathcal{L}_d(\mathbf{z}; \boldsymbol{\psi}, \mathcal{D}) + \mathcal{L}_r(\boldsymbol{\phi}, \boldsymbol{\psi}; \mathcal{D}), \text{ s.t. (2)} \tag{3}$$

where three loss terms are introduced: (1) Encoder loss: $\mathcal{L}_e = \frac{1}{(N+1)n}\sum_{i=0}^N \|\boldsymbol{\phi}(\hat{\mathbf{x}}_i) - \mathbf{z}_i\|^2$, (2) Decoder loss: $\mathcal{L}_d = \frac{1}{(N+1)r}\sum_{i=0}^N \|\hat{\mathbf{x}}_i - \boldsymbol{\psi}(\mathbf{z}_i)\|^2$, (3) Reconstruction loss: $\mathcal{L}_r = \frac{1}{(N+1)r}\sum_{i=0}^N \|\hat{\mathbf{x}}_i - \boldsymbol{\psi}(\boldsymbol{\phi}(\hat{\mathbf{x}}_i))\|^2$. The term $\mathcal{L}_e = 0$ is a necessary condition for satisfying the invariant space assumption in KBF formulation, $\mathcal{L}_d$ penalizes on the N-step prediction error at each of N steps, and $\mathcal{L}_r$ penalizes on the reconstruction error. In addition, sometimes a regularization term $\mathcal{R}(\mathcal{D})$ is included to improve the model generalizability.

However, due to the ODE constraint, optimizing problem (3) requires an adjoint ODE solver, that is expensive to evaluate; also a generic adjoint solver does not leverage the bilinear structure of KBF for training efficiency.

A more widely-used approach is to discretize the KBF model (2) with a time step size Δt. Assuming a zeroth-order hold of input $\mathbf{u}_k$ at time t_k and a sufficiently small

Δt, discrete-time KBF takes the following form of up to first-order time accuracy [29]

$$\mathbf{z}_{k+1} = \mathbf{A}_d \mathbf{z}_k + \sum_{i=1}^{m} \mathbf{B}_{d,i} \mathbf{z}_k u_{k,i} \equiv \mathbf{A}_k \mathbf{z}_k \tag{4}$$

where $\mathbf{A}_d = \mathbf{I} + \mathbf{A}\Delta t$, $\mathbf{B}_{d,i} = \mathbf{B}_i \Delta t$, and $\mathbf{A}_k = \mathbf{A}_d + \sum_{i=1}^{m} \mathbf{B}_{d,i} u_{k,i}$ is effectively a family of matrices parametrized by $\mathbf{\Gamma}$. Subsequently, the optimization is reduced to an equality-constrained formulation,

$$\underset{\mathbf{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}}{\operatorname{argmin}} \ \mathcal{L}_e + \mathcal{L}_d + \mathcal{L}_r \tag{5a}$$

$$\text{s.t. } \mathbf{z}_k = \mathbf{A}_k \mathbf{z}_{k-1}, \quad k = 1, \cdots, N \tag{5b}$$

$$\mathbf{z}_0 = \boldsymbol{\phi}(\hat{\mathbf{x}}_0) \tag{5c}$$

which will be referred to as *single-level optimization* (SLO).

Note that the equality-constrained form of SLO is chosen for the sake of clarity, and it is equivalent to the more commonly used unconstrained formulation (e.g., [23]–[25], [28]), since the $N+1$ intermediate variables $\{\mathbf{z}_k\}_{k=0}^{N}$ can be solved exactly from the $N+1$ constraints, starting from (5c). Particularly, when $N = 1$, the SLO reduces to a single-step formulation [23]–[25], with encoder loss $\mathcal{L}_e = \frac{1}{Nn} \sum_{k=1}^{N} \|\boldsymbol{\phi}(\hat{\mathbf{x}}_k) - \mathbf{A}_{k-1} \boldsymbol{\phi}(\hat{\mathbf{x}}_{k-1})\|^2$, and decoder loss $\mathcal{L}_d = \frac{1}{Nr} \sum_{k=1}^{N} \|\hat{\mathbf{x}}_k - \boldsymbol{\psi}(\mathbf{A}_{k-1} \boldsymbol{\phi}(\hat{\mathbf{x}}_{k-1}))\|^2$.

For the learning of KBF, the SLO formulation poses three potential concerns. *First*, the time discretization of the learned KBF model is first-order time accurate; this may result in the error accumulation that impairs predictive accuracy over long time horizon, and may also pose a challenge when learning dynamics sampled at low frequency. *Second*, for mathematical rigor, the validity of KBF hinges on the satisfaction of $\mathcal{L}_e = 0$ for the invariance of Koopman subspace. However, in the sum of losses, upon convergence of the model training, each loss term would typically reach a small but nonzero value, and a nonzero $\mathcal{L}_e$ indicates an inaccurate KBF operator that produces error in the time horizon of N considered in the training. During the prediction, the error may start to accumulate within a short time horizon, and limit the long-term predictive capability. *Third*, while the SLO can be written in an unconstrained form and compatible with common deep learning frameworks, the dynamics losses have a recursive formulation that involves nested evaluation of the intermediate variables $\mathbf{z}$, leading to a high computational cost in the backpropagation during training on the order of $O(N^2)$, i.e., a quadratic (or at least superlinear) growth with respect to the length of time horizon; this renders the learning process inefficient.

III. Learning Koopman Operator Using Bi-level Optimization

We present a new *bi-level optimization* (BLO) to resolve the potential issues of SLO.

A. Integral formulation

First, to minimize the model error due to time discretization, we employ a general formulation based on numerical integration. Given a control input $\mathbf{u}(t)$, integrate on both sides of (2) over $[t_0, t_N]$,

$$\int_{t_0}^{t_N} \dot{\mathbf{z}} dt = \int_{t_0}^{t_N} \mathbf{A}\mathbf{z} + \sum_{i=1}^{m} \mathbf{B}_i \mathbf{z} u_i(t) dt$$

$$\Rightarrow \quad \mathbf{z}_N = \mathbf{z}_0 + \sum_{i=0}^{N} w_i \mathbf{A}_i \mathbf{z}_i, \tag{6}$$

where w_i are the weights for numerical integration using evenly spaced data points, which are obtained using the standard composite Newton-Cotes formulae, e.g., trapezoid rule for first-order accuracy and Simpson's 3/8 rule for third-order accuracy. When $k = 1$ and $[w_0, w_1] = [\Delta t, 0]$, (6) reduces to the zeroth-order hold formulation.

Next, using the integral form, SLO simplifies to

$$\underset{\mathbf{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}}{\operatorname{argmin}} \ \mathcal{L}_e^b + \mathcal{L}_d^b + \mathcal{L}_r \tag{7a}$$

$$\text{s.t. (6)}$$

$$\mathbf{z}_k = \boldsymbol{\phi}(\hat{\mathbf{x}}_k), \quad k = 0, \cdots, N-1 \tag{7b}$$

with new encoder and decoder losses: $\mathcal{L}_e^b = \frac{1}{Nn} \|\boldsymbol{\phi}(\hat{\mathbf{x}}_N) - \mathbf{z}_N\|^2$ and $\mathcal{L}_d^b = \frac{1}{Nr} \|\hat{\mathbf{x}}_N - \boldsymbol{\psi}(\mathbf{z}_N)\|^2$. The new single dynamics constraint (6) may appear "weaker" when compared to the explicit multi-step dynamics constraints in the standard SLO formulation, but it more accurately represents the long-horizon KBF dynamics at the order of numerical integration.

B. Bi-level optimization

Next, to mitigate the second issue of SLO, (7) is reformulated in a BLO form,

$$\underset{\boldsymbol{\phi}, \boldsymbol{\psi}}{\operatorname{argmin}} \ \mathcal{L}_e^b + \mathcal{L}_r \tag{8a}$$

$$\text{s.t. } \min_{\mathbf{\Gamma}} \mathcal{L}_e^b \tag{8b}$$

$$\text{s.t. (6), (7b)} \tag{8c}$$

The inner optimization solely minimizes the encoder loss with respect to the system matrices $\mathbf{\Gamma}$, and would achieve at least the approximate satisfaction of the invariance of Koopman subspace, meaning an accurate enforcement of Koopman dynamics over time horizon of length N. The outer optimization minimizes the losses with respect to the autoencoder $\{\boldsymbol{\phi}, \boldsymbol{\psi}\}$, while the decoder loss $\mathcal{L}_d^b$ is removed; the argument is that, when the reconstruction loss $\mathcal{L}_r$ is minimized, the fact that the encoder loss $\mathcal{L}_e^b$ is always minimized would imply a sufficiently low decoder loss.

C. Algorithm for solving bi-level optimization

Subsequently, we present an algorithm for solving BLO at a cost of $O(N)$ to resolve the third issue of SLO.

The inner optimization (8b)-(8c) is converted to an unconstrained one, $\min_{\mathbf{\Gamma}} \|\Delta \mathbf{z} - \mathbf{\Gamma} \boldsymbol{\xi}\|^2$, where $\mathbf{z}_i = \boldsymbol{\phi}(\hat{\mathbf{x}}_i)$, $\Delta \mathbf{z} = \mathbf{z}_N - \mathbf{z}_0$, $\boldsymbol{\xi} = \sum_{i=0}^{N} w_i \mathbf{y}_i$, and $\mathbf{y} = [\mathbf{z}^\top, u_1 \mathbf{z}^\top, u_2 \mathbf{z}^\top, \ldots, u_m \mathbf{z}^\top]^\top$.

For a dataset of K trajectories, define $\mathbf{Z} = [\Delta\mathbf{z}^{(1)}, \Delta\mathbf{z}^{(2)}, \cdots, \Delta\mathbf{z}^{(K)}]$ and $\boldsymbol{\Xi} = [\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \cdots, \boldsymbol{\xi}^{(K)}]$, and the unconstrained BLO formulation is

$$\underset{\boldsymbol{\phi},\boldsymbol{\psi}}{\operatorname{argmin}} \ \mathcal{L}_e^K(\boldsymbol{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}) + \mathcal{L}_r^K(\boldsymbol{\phi}, \boldsymbol{\psi}) \tag{9a}$$

$$\text{s.t.} \ \min_{\boldsymbol{\Gamma}} \mathcal{L}_e^K(\boldsymbol{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{9b}$$

where

$$\mathcal{L}_e^K = \frac{1}{K(N+1)n} \left\| \mathbf{Z} - \boldsymbol{\Gamma}\boldsymbol{\Xi} \right\|^2$$

$$\mathcal{L}_r^K = \frac{1}{K(N+1)r} \sum_{i,j} \left\| \hat{\mathbf{x}}_i^{(j)} - \boldsymbol{\psi}(\boldsymbol{\phi}(\hat{\mathbf{x}}_i^{(j)})) \right\|^2$$

Despite its complexity of two levels, the BLO problem may be solved at relative ease leveraging the coordinate descent strategy. This is because the system matrices $\boldsymbol{\Gamma}$ are only solved in the inner optimization (9b), and kept fixed in the outer optimization (9a); vice versa for the autoencoder parameters $\boldsymbol{\phi}, \boldsymbol{\psi}$. At the inner level, due to its simple quadratic structure, (9b) has a closed-form solution $\boldsymbol{\Gamma} = \boldsymbol{\Xi}\mathbf{Y}^+$, where $\square^+$ denotes pseudo-inverse. At the outer level, one may employ a typical learning framework based on stochastic gradient descent (SGD) methods, e.g., RMSProp or Adam, to minimize the loss. A similar strategy is employed in [23], [24], where a one-step discrete-time formulation was used.

Algorithm 1 Learning algorithm for Koopman with control based on bi-level optimization

Input: Dataset of K trajectories, length of time horizon N, number of epoches N_{ep}, number of batches N_b for SGD, order of numerical integration P.

Output: Model parameters $\{\boldsymbol{\Gamma}, \boldsymbol{\phi}, \boldsymbol{\psi}\}$.

```
Initialize {Γ^(1), φ^(1), ψ^(1)}
for n = 1, 2, ··· , N_ep do
    Form matrices Z and Ξ from all data using order P. ▷ O(νNK)
    Fix {φ^(n), ψ^(n)}, solve (9b) for Γ^(n+1). ▷ O(ν^2 K)
    Shuffle and split the data into N_b batches.
    for i = 1, 2, ··· , N_b do
        Form matrices Z^i and Ξ^i from the ith data batch using order P. ▷ O(νNK/N_b)
        Fix Γ^(n+1), and update {φ, ψ} using Z^i and Ξ^i. ▷ O(NK/N_b)
    end for
end for
```

D. Computational complexity analysis

The complete Koopman learning algorithm based on BLO is listed in Alg. 1 and the computational cost of each major step is labelled. The details are discussed further as follows.

Let $\dim(\boldsymbol{\xi}) = \nu = n(m+1)$, and typically the number of trajectories $K \gg \nu$. At the inner level, forming matrices $\mathbf{Z}$ and $\boldsymbol{\Xi}$ costs $O(\nu NK)$, and solving the least squares problem using SVD costs $O(\nu^2 K)$. At the outer level, the cost in each batch is dominated by forming the matrices, and in total the cost is $O(\nu NK)$. Therefore, the computational complexity of BLO is $O(\nu(\nu+N)KN_{ep})$ and scales *linearly* with the length of time horizon; this is in sharp contrast with the quadratic (or at least superlinear) growth in SLO.

IV. Numerical Simulation

To demonstrate the effectiveness of the proposed approach, we investigate two example nonlinear systems: a well-studied two-dimensional nonlinear system in nonlinear control; and a lightly-damped double pendulum system with dimension 4, which shows that the proposed algorithm can generalize to higher-dimensional systems.

A. A two-dimension nonlinear system

1) Problem setup: We consider a variant of a well-known nonlinear system [11]:

$$\dot{x}_1 = \mu x_1 + u_1 + u_3 x_1 \tag{10a}$$

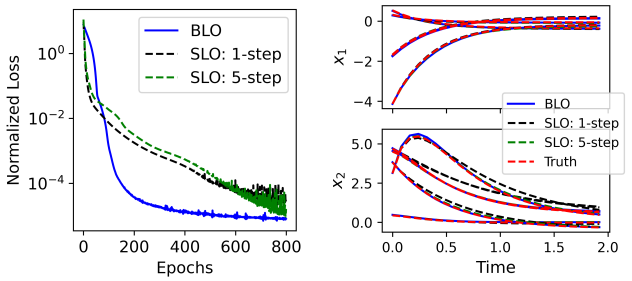
$$\dot{x}_2 = \lambda(x_2 - x_1^2) + u_2 \tag{10b}$$

where $\mu = -3$ and $\lambda = -2$ are pre-defined system parameters controlling characteristic time scales, and (u_1, u_2, u_3) are time-varying controls to the system. The system has an isolated equilibrium point at $\mathbf{x}_e = (\frac{-u_1}{\mu+u_3}, \frac{u_1^2}{(\mu+u_3)^2} - \frac{u_2}{\lambda})$, and increasing u_3 slows down the convergence to $x_{e,1}$.

The trajectories for model training and validation were generated by uniformly sampling initial conditions $\mathbf{x}_0 \in [-5, 5] \times [-5, 5]$ with 32 points in both directions, with step inputs as control that are randomly generated with $u_i \in [-1.8, 1.8]$. Another 100 trajectories are randomly sampled from the same range of $\mathbf{x}_0$ and u_i. All trajectories were generated using by 4th order Runge Kutta with a time step size of 0.08s for 25 steps. All trajectories were normalized to $[0, 1]$. During training, when a time horizon of N is used, using a sliding window of $N+1$, a L-step trajectory can produce $K = L - N$ trajectories for training.

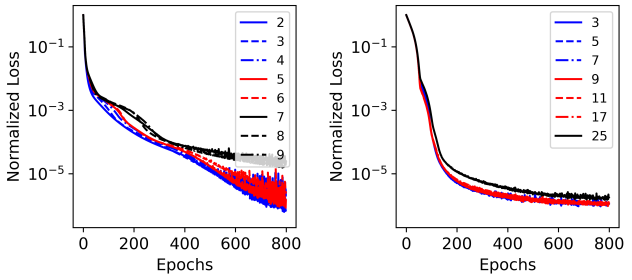
A 4-dimensional embedding space is selected based on empirical observation, where three dimensions are learned using a neural network for encoding/decoding, and the remaining dimension is set to be 1 for the completeness of the basis. Based on a cross-validation study, both the encoder and decoder have 2 hidden layers with Swish activation and have sizes (16, 16). For all benchmark cases, the model is trained for 800 epochs with 16 batches using the Adam optimizer, and learning rates of 0.001 and 0.0001 are used for SLO and BLO, respectively. For consistency in comparison, all algorithms are implemented using the JAX package.

2) Results: First, we compare the BLO against a single-step SLO and a 5-step SLO case in Fig. 1 in terms of convergence characteristics and prediction accuracy. Due to the differences in the implementation, only the prediction losses in the original state space are reported, and the losses are normalized by their respective initial values, so that the relative decreases in the loss are compared. The SLO cases have a similar initial convergence rate and start to show difficulty to reduce the predictive loss further, presumably due to the competing effects with the other losses. The



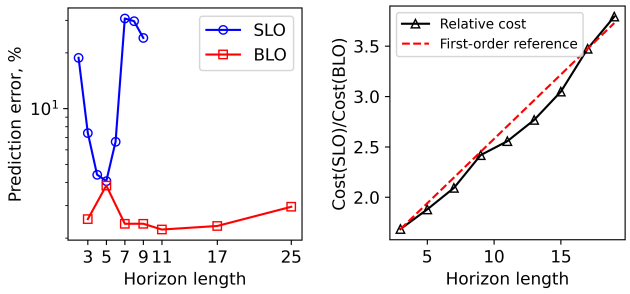
(a) Convergence history (b) Samples of predicted trajectory

Fig. 1: Baseline performances of BLO and SLO.



(a) SLO (b) BLO

Fig. 2: Effect of horizon length.



(a) Error in prediction (b) Cost in training

Fig. 3: Comparison between BLO and SLO.

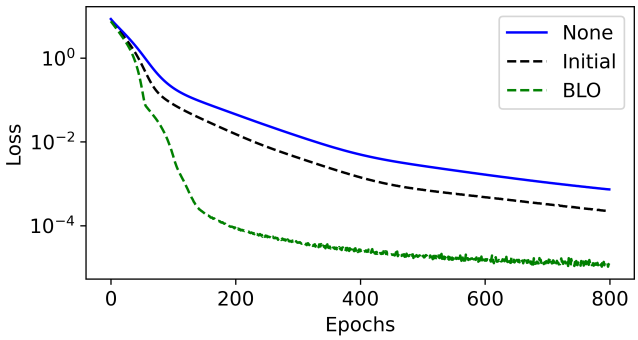


Fig. 4: Effect of BLO on convergence.

BLO shows an initially slower convergence rate but achieves convergence within 400 epochs. In model prediction, the BLO model achieves the lowest error of 3.5%, which is attributed to the more accurate model representation. Among the SLO formulation, multi-step case (3.7%) is relatively better than the single-step case (19%), as the former accounts for longer horizon in the training to achieve better accuracy.

Next, a parametric study is performed on the length of horizon, as shown in Fig. 2. The SLO starts to show difficulty in convergence when $N > 6$, while the BLO consistently achieves high convergence rate up to $N = 25$. The SLO-based training with $N > 9$ is not performed due to the high computational cost. Figure 3a shows the model prediction error, where both SLO and BLO show a decrease in model error when N is small, however, the error in SLO model quickly increases when $N > 6$ while the BLO model error remains consistently low. Figure 3b compares the growth in computational cost with increasing horizon length. For each horizon length, each of the SLO and BLO cases are run for 5 epochs, the time costs are recorded, and the ratio is reported. It is clear that the complexity of BLO is $O(N)$ less than that of SLO, which is attributed to the removal of the nested formulation from SLO.

Lastly, we also briefly show the effect of bi-level formulation, as shown in Fig. 4, where three cases are considered: (1) "None": The BLO loss (8a) is treated as if SLO and $\mathbf{\Gamma}, \phi, \psi$ are optimized together with random initial guesses; (2) "Initial": $\mathbf{\Gamma}$ is computed only once at the start of training and used as an initial guess, and then the "None" strategy is used; (3) "BLO": The proposed algorithm. The only case that achieves sufficient convergence is BLO; this is attributed to the optimality of the system matrix maintained by the inner optimization.

B. Double Pendulum

Next, we consider a damped and controlled double pendulum problem to show the feasibility of the proposed algorithm for high-dimensional systems.

$$\ddot{\theta}_1 = (M_2 L_1 \dot{\theta}_1^2 \sin\delta\cos\delta + M_2 g \sin\theta_2 \cos\delta + M_2 L_2 \dot{\theta}_2^2 \sin\delta - \bar{M} g \sin\theta_1 + u_1)/(L_1\rho) - \dot{\theta}_1 \quad (11)$$

$$\ddot{\theta}_2 = (-M_2 L_2 \dot{\theta}_2^2 \sin\delta\cos\delta + \bar{M} g \sin\theta_1 \cos\delta - \bar{M} L_1 \dot{\theta}_1^2 \sin\delta - \bar{M} g \sin\theta_1 + u_2)/(L_2\rho) - \dot{\theta}_2 \quad (12)$$

where $\delta = \theta_2 - \theta_1$, $\bar{M} = M_1 + M_2$, and $\rho = \bar{M} - M_2 \cos^2\delta$. The masses and lengths of the two pendulums are $M_1 =$ 1kg (upper), $M_2 =$ 1kg (lower), $L_1 =$ 1m, $L_2 =$ 1m. The damping terms are added to both pendulums, so as to create a stable isolated equilibrium point in the system. 320 trajectories were generated with initial conditions randomly generated for $\theta_i \in [-10°, 10°]$ and $\dot{\theta}_i \in [-10°/s, 10°/s]$ and control $u_i \in [-0.25, 0.25]$N. Another 100 trajectories are generated for test. The embedded space is of dimension 9, with 8 being learned through an autoencoder, and one added to be the constant 1. The encoder and decoder hidden state sizes are (32, 32, 32). all the rest of the details are the same as those for the first example.

Figure 5 shows the prediction performance of the learned Koopman model and proposed method performs well for this higher dimensional system. While the training data is sampled at 12.5 Hz for 2s, the prediction is performed at 50 Hz for 4s, thanks to the new continuous-time formulation. The model matches with the truth well with an error of 4.5%. The SLO produces models that have over 50% error and the predicted trajectories are not shown; the high error is likely due to the low data sampling rate, for which the discrete-time

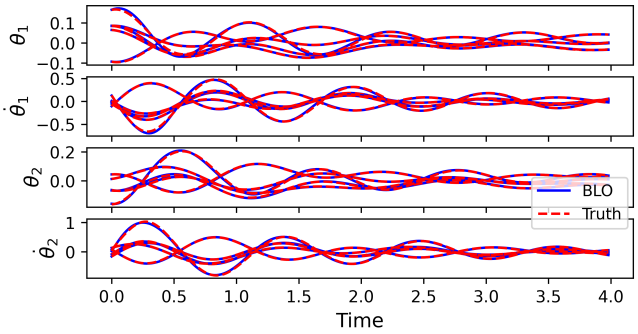


Fig. 5: Predicted double pendulum dynamics.

formulation is inaccurate.

V. Conclusion

This paper presents a bi-level optimization framework to learn the Koopman Bilinear Form by jointly optimizing the Koopman embedding and dynamics with explicit and exact constraints of continuous-time Koopman dynamics. Our approach produces a continuous-time KBF model that is more accurate than the commonly used zeroth-order hold model by construction. Using an integral formulation, a long-horizon KBF dynamic constraint can be imposed during the learning process without needing to resort to a multi-step discrete-time constraint, that is time consuming to evaluate. Furthermore, using a bi-level optimization strategy, the KBF dynamics and the nonlinear mapping in a staggered format, and a high convergence rate in training is achieved.

We validate the proposed approach on two example nonlinear systems with control. Results show that our method successfully learns the nonlinear dynamics. Comparing to the single-level optimization method, our method achieves more accurate prediction with lower prediction error, faster convergence, and higher computational efficiency. Future work will investigate online model prediction for more complex physical scenarios, including the aerial vehicle flying in the wind gust environment. The developed bi-level optimization framework will be released via our open-source robotic learning library PyPose [30].

References

[1] M. Mousaei, J. Geng, A. Keipour, D. Bai, and S. Scherer, "Design, modeling and control for a tilt-rotor vtol uav in the presence of actuator failure," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4310–4317.

[2] J. Geng and J. W. Langelaan, "Cooperative transport of a slung load using load-leading control," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 7, pp. 1313–1331, 2020.

[3] S. Triest, M. G. Castro, P. Maheshwari, M. Sivaprakasam, W. Wang, and S. Scherer, "Learning risk-aware costmaps via inverse reinforcement learning for off-road navigation," in *2023 International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[5] T. Ji, J. Geng, and K. Driggs-Campbell, "Robust model predictive control with state estimation under set-membership uncertainty," in *2022 IEEE Conference on Decision and Control (CDC)*. IEEE, 2022.

[6] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[7] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. Ben Amor, "Estimation of perturbations in robotic behavior using dynamic mode decomposition," *Advanced Robotics*, vol. 29, no. 5, pp. 331–343, 2015.

[8] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[9] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[10] Y. Hu, J. Geng, C. Wang, J. Keller, and S. Scherer, "Off-policy evaluation with online adaptation for robot exploration in challenging environments," *IEEE Robotics and Automation Letters (Under Review)*, 2023.

[11] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLOS ONE*, vol. 11, no. 2, pp. 1–19, 02 2016.

[12] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[13] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using Koopman operator framework," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016, 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[14] A. Mauroy, Y. Susuki, and I. Mezić, *Koopman operator in systems and control*. Springer, 2020.

[15] P. Bevanda, S. Sosnowski, and S. Hirche, "Koopman operator dynamical models: Learning, analysis and control," *Annual Reviews in Control*, vol. 52, pp. 197–212, 2021.

[16] D. Goswami and D. A. Paley, "Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A Koopman spectral approach," *IEEE Transactions on Automatic Control*, vol. 67, pp. 2715–2728, 6 2022.

[17] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using Koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[18] J. A. Rosenfeld and R. Kamalapurkar, "Dynamic mode decomposition with control Liouville operators," *IFAC-PapersOnLine*, vol. 54, no. 9, pp. 707–712, 2021.

[19] J. C. Schulze and A. Mitsos, "Data-driven nonlinear model reduction using Koopman theory: Integrated control form and NMPC case study," *IEEE Control Systems Letters*, vol. 6, pp. 2978–2983, 2022.

[20] A. Goldschmidt, E. Kaiser, J. L. Dubois, S. L. Brunton, and J. N. Kutz, "Bilinear dynamic mode decomposition for quantum control," *New Journal of Physics*, vol. 23, no. 3, p. 033035, 2021.

[21] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of koopman eigenfunctions for control," *Machine Learning: Science and Technology*, vol. 2, no. 3, p. 035023, 2021.

[22] S. Klus, F. Nüske, and S. Peitz, "Koopman analysis of quantum systems," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 31, p. 314002, 2022.

[23] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 10, 2017.

[24] R. Wang, Y. Han, and U. Vaidya, "Deep koopman data-driven optimal control framework for autonomous racing," *Early Access*, vol. 5, 2021.

[25] C. Folkestad, S. X. Wei, and J. W. Burdick, "KoopNet: Joint learning of Koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1344–1350.

[26] H. Shi and M. Q.-H. Meng, "Deep Koopman operator with control for nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7700–7707, 2022.

[27] X. Jiang, Y. Li, and D. Huang, "Modularized bilinear Koopman operator for modeling and predicting transients of microgrids," *arXiv preprint arXiv:2205.03214*, 2022.

[28] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, p. 4950, 2018.

[29] S. Peitz, S. E. Otto, and C. W. Rowley, "Data-driven model predictive control using interpolated Koopman generators," *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 3, pp. 2162–2193, 2020.

[30] C. Wang, D. Gao, K. Xu, J. Geng, Y. Hu, Y. Qiu, B. Li, F. Yang, B. Moon, A. Pandey, *et al.*, "Pypose: A library for robot learning with physics-based optimization," in *2023 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.