Deep Koopman Operator with Control for Nonlinear Systems

Haojie Shi, Max Q.-H. Meng†, *Fellow, IEEE*

***Abstract*—Recently Koopman operator has become a promising data-driven tool to facilitate real-time control for unknown nonlinear systems. It maps nonlinear systems into equivalent linear systems in embedding space, ready for real-time linear control methods. However, designing an appropriate Koopman embedding function remains a challenging task. Furthermore, most Koopman-based algorithms only consider nonlinear systems with linear control input, resulting in lousy prediction and control performance when the system is fully nonlinear with the control input. In this work, we propose an end-to-end deep learning framework to learn the Koopman embedding function and Koopman Operator together to alleviate such difficulties. We first parameterize the embedding function and Koopman Operator with the neural network and train them end-to-end with the K-steps loss function. We then design an auxiliary control network to encode the nonlinear state-dependent control term to model the nonlinearity in control input. For linear control, this encoded term is considered the new control variable instead, ensuring the linearity of the embedding space. Then we deploy Linear Quadratic Regulator (LQR) on the linear embedding space to derive the optimal control policy and decode the actual control input from the control net. Experimental results demonstrate that our approach outperforms other existing methods, reducing the prediction error by order-of-magnitude and achieving superior control performance in several nonlinear dynamic systems like damping pendulum, CartPole, and 7 Dof robotic manipulator.**

***Index Terms*—Deep Learning Methods, Model Learning for Control, Machine Learning for Robot Control.**

I. INTRODUCTION

THE prediction and control of nonlinear systems remain a challenging task. Classical methods first require full knowledge of the system model to predict the evolution of dynamical systems and then design state feedback control laws [22] or optimization-based control policies [1], [16]. However, the hidden dynamics of chaotic nonlinear systems are usually unknown or too complicated to be modeled, prohibiting such controllers from taking effect.

† Corresponding author

This work is partially supported by National Key R&D program of China with Grant No. 2019YFB1312400 and in part by the Hong Kong RGC GRF grants # 14200618 awarded to Max Q.-H. Meng.

Haojie Shi is from the Chinese University of Hong Kong, Hong Kong, h.shi@link.cuhk.edu.hk.

Max Q.-H. Meng is with Shenzhen Key Laboratory of Robotics Perception and Intelligence, and the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China, on leave from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute of The Chinese University of Hong Kong, Shenzhen 518057, China, max.meng@ieee.org

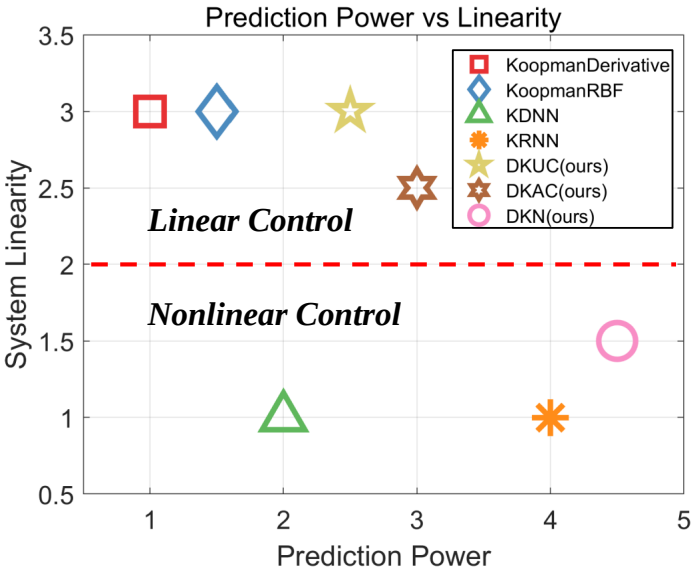

Fig. 1: Prediction Power vs. System Linearity in embedding space. Previous Koopman-based methods can be easily controlled with linear control methods while lacking in prediction power. And deep learning-based approaches take the opposite. Our proposed approach makes a good trade-off between prediction power and system linearity for control.

For the prediction of nonlinear systems, great research interest has been put into data-driven methods like SINDy [7]. The deep neural network also plays an important role in nonlinear system modeling with its strong expressive power [14]. Both of these methods discover hidden dynamics in a nonlinear form, demanding nonlinear control methods such as iLQR [16] and NMPC [1]. But these optimization-based methods require tremendous computational time with the increasing dimensions of the system state, prohibitive for real-time control of high dimensional nonlinear systems. Besides, model-based RL approaches illustrate great success in controlling nonlinear systems. They first train the neural network to approximate the dynamics and then apply policy search methods for trajectory optimization via backpropagation [26], [28]. And model-free RL approaches also achieve strong performance on continuous control of nonlinear systems [11], [25], [25]. However, these methods suffer from sample efficiency issues and lack of generalization.

Recently, the Koopman operator shows great potential to alleviate such difficulties for nonlinear control under unknown dynamics. As a data-driven approach, the Koopman operator can learn the system model automatically with the least square method. Furthermore, the Koopman operator is capable of mapping nonlinear dynamics to linear systems in the embedding space. And the embedded linear system is ready to be controlled via linear methods like LQR [3], [20] and MPC [8], [13]. But the selection of embedding function to maintain

prediction quality remains a tough task. Recent approaches focus on learning the embedding functions with deep neural networks [2], [18] and then apply linear control methods [12], [17]. But after lifting the state to embedded space, the linear-quadratic form of the cost function can be distorted in the hidden space, increasing the subsequent control difficulty. Moreover, previous Koopman-based approaches only model nonlinear systems with linear control input, destroying the prediction quality for general nonlinear systems.

In this work, we propose an end-to-end deep learning framework to learn the Koopman embedding function and Koopman Operator together to alleviate such difficulties. To maintain the consistency of cost function in embedding space, our proposed embedding function simply concatenates the original state and the neural network encodings so that the original state can be preserved for further control. Furthermore, we propose an auxiliary control network that both encode the state and control to model the nonlinear state-dependent term in control. Then the encoded term is considered as the new control variable for linear control. After deriving the optimal control policy, the actual control input can be recovered from the encoded term. In this way, we maintain both the prediction power and system linearity for control.

To further evaluate our results, we conduct experiments on several nonlinear systems. For prediction, experimental results demonstrate that our approach outperforms other Koopman-based approaches and achieves comparable or even better performance than the deep recurrent neural network. Consequently, better prediction performance ensures our approach to achieve better control results in nonlinear dynamic environments like damping pendulum, CartPole, and 7 Dof robotic manipulator. [1]

In summary, the main contributions of these papers are:

1) We propose an end-to-end deep learning framework that trains the embedding function and Koopman Operator together, maintaining both the prediction quality and the consistency of cost function in embedded space.
2) We design an auxiliary control network that models the state-dependent nonlinearity in control input, improving the prediction performance while retaining the system linearity for control in embedding space.
3) We evaluate our approach in several nonlinear dynamic systems and achieve better prediction and control performance than existing Koopman-based methods and deep learning-based approaches.

II. RELATED WORK

Prediction of nonlinear systems: For unknown dynamics, data-driven methods have been considered as a leading tool to learn the hidden nonlinear dynamics model. Dynamic model decomposition algorithms (DMD,EDMD) 23], [27], [29 seeks an optimal linear operator matrix to fit the evolution of nonlinear systems with spectral decomposition. Sparse identification of nonlinear dynamics algorithms [6], [7] aim to discover the governing equations from data of nonlinear dynamics via sparse regression of the dynamics matrix. Deep neural networks (DNNs) also play an important role in directly learning the hidden nonlinear dynamics from data [14], especially the deep recurrent neural networks that take advantage of the temporal information of data [10]. Koopman-based approaches first select embedding function to lift the original state into embedding space and then learn the linear operator with the least square method to fit the linear dynamics in the lifted space. To choose the embedding functions, standard methods utilize the radial basis functions (RBFs) as the representation function [13], but RBFs lack expressive power and cause significant prediction error in high dimensional systems. Derivative-based Koopman [20] uses higher-order derivatives of nonlinear dynamics as lift function, which achieves promising prediction performance, but it requires the knowledge of the derivatives of unknown dynamics. In contrast, deep learning approaches promote the learning of embedding functions with deep neural networks [2], [13], improving the prediction quality for a long time horizon.

Direct Nonlinear Control: For nonlinear control, two kinds of methods exist: direct control approaches and linearization approaches. As for direct control approaches, they directly control the nonlinear model with backstepping feedback control laws [9] or optimal control such as iLQR [16] and NMPC [1]. Backstepping methods require the analytical form of dynamic equations, which is inaccessible for complex nonlinear systems. And direct optimal control methods demand tremendous computational time for high dimensional systems, computationally prohibitive for real-time control. Besides, RL-based methods train the neural network to approximate the optimal policy via policy gradient or value iteration [11], [15], [24]. They can compute optimal policy in real-time with feedforward networks but still suffer from sample efficiency and generalization issues.

Nonlinear Control Linearization: Local linearization serves as the basic approach for nonlinear control. It first linearizes the nonlinear function at the current state and then controls the system with LQR or MPC [22], but the stability is only guaranteed in a local region where the linearization approximates the nonlinear dynamics well. Koopman-based approaches [12], [13], [17], [20] highly enlarge the stable area by globally mapping the nonlinear dynamics to linear dynamics in embedding space. After lifting the state space to embedding space, liner control methods like LQR and MPC are ready to be deployed for real-time control.

III. BACKGROUND

A. Basics of Koopman Operator

Given the nonlinear systems with discrete dynamics:

$$s_{k+1} = F(s_k) \tag{1}$$

Koopman Operator $\mathcal{K}$ is an infinite linear operator that evolves the embedding functions g of the state:

$$\mathcal{K}g(s) = g \circ F(s) \tag{2}$$

Considering the control input with the assumption that $u_{k+1} = u_k$, the evolution flows follows as:

$$g(x_{k+1}, u_{k+1}) = \mathcal{K}g(x_k, u_k) \tag{3}$$

[1]Source Code: https://github.com/HaojieSHI98/DeepKoopmanWithControl

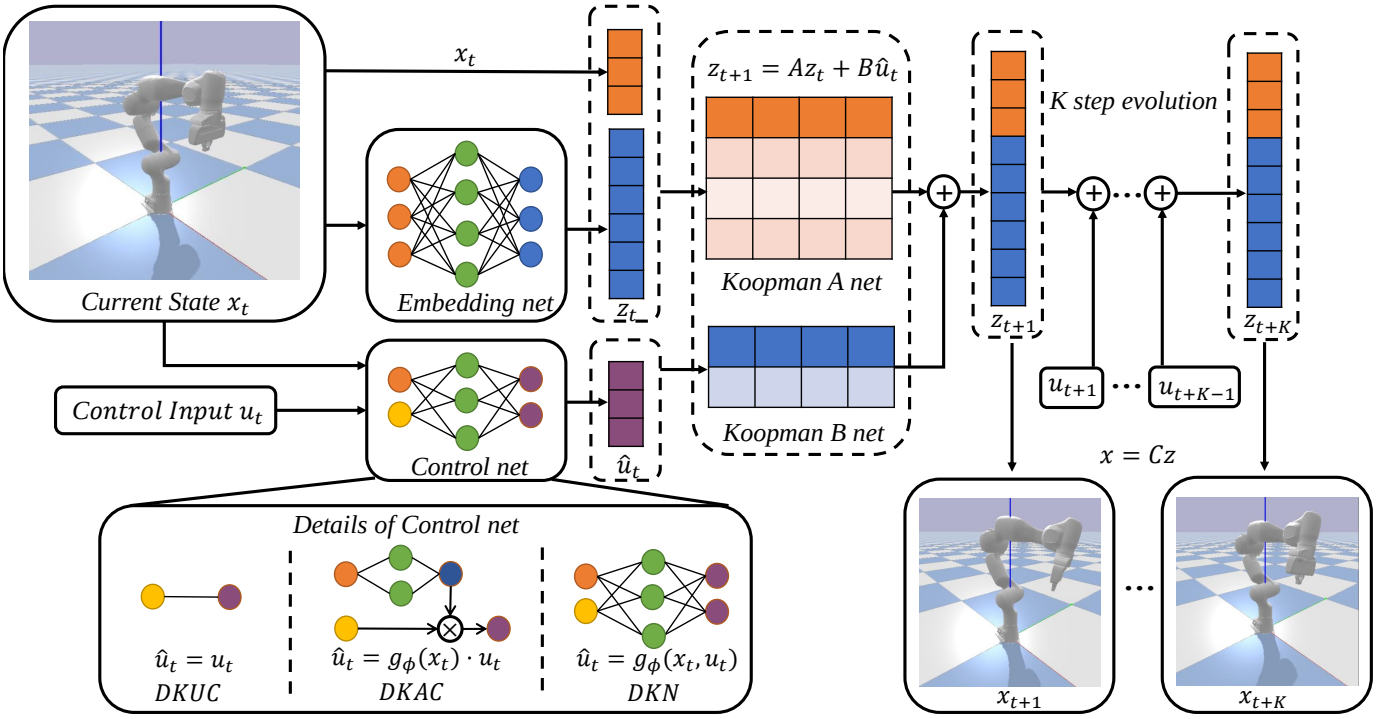

Fig. 2: The overview of neural network framework for K-steps predictions

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m$, and $g : \mathbb{R}^{n+m} \to \mathbb{R}^d$. Therefore, with the embedding function g lifting the state space to embedding space, Koopman Operator $\mathcal{K}$ maps the nonlinear dynamics to linear dynamics. Readers can refer to [5] for more details of Koopman Operator theory.

Previous methods simplify (3) by linearizing the control term. They first seperate the embedding function into two parts that $g(x,u) = [g_x(x,u); g_u(x,u)]$, and assume that $g_x(x,u) = g_x(x), g_u(x,u) = u$. Then the Eq.(3) is simplified as:

$$\begin{bmatrix} g_x(x_{k+1}) \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} K_{xx} & K_{xu} \\ K_{ux} & K_{uu} \end{bmatrix} \begin{bmatrix} g_x(x_k) \\ u_k \end{bmatrix} \tag{4}$$

which can further be written as:

$$g_x(x_{k+1}) = K_{xx} g_x(x_k) + K_{xu} u_k \tag{5}$$

But such simplification inevitably damages the prediction performance of the nonlinear system and can further affect the control results.

B. Learning Koopman Operator

Data-driven method is deployed to learn the Koopman Operator $\mathcal{K}$. Given the dataset $g_k = g(x_k, u_k), k = [1, 2, ..., N]$, the Koopman Operator K_d is computed by optimizing the mean square loss:

$$K_d = \arg\min_{K_d} \Sigma_{i=1}^{N-1} ||g_{i+1} - K_d g_i|| \tag{6}$$

The least square problem has a closed-form solution as:

$$K_d^* = PG^\dagger \tag{7}$$

where $P = \frac{1}{N-1}\Sigma_{i=1}^{N-1} g_{i+1} g_i^T$, $G = \frac{1}{N-1}\Sigma_{i=1}^{N-1} g_i g_i^T$, and $\dagger$ is the Moore–Penrose pseudoinverse.

IV. METHODOLOGY

A. Framework

We first seperate the embedding function into two parts: $g(x,u) = [g_x(x,u); g_u(x,u))]$. Without loss of generality, we assume that the first term is only related to the system state, so we get $g_x(x,u) = g_x(x)$. Based on Eq.(3), it is derived that:

$$g_x(x_{k+1}) = K_{xx} g_x(x_k) + K_{xu} g_u(x_k, u_k) \tag{8}$$

For the first part of embedding function, we use a deep neural network θ to parameterize $g_x(x,k)$. Instead of directly encoding the state like [18], which requires an extra decode network, we concatenate the origin state and network encoding together:

$$z_k = g_x(x_k) = \begin{bmatrix} x_k \\ g_\theta(x_k) \end{bmatrix} \tag{9}$$

where z_k is the embedded state, $g_\theta : \mathbb{R}^n \to \mathbb{R}^d$ is the parameterized neural network. The advantage of this design is that the original state can be easily recovered by:

$$x_k = Cz_k \tag{10}$$

where the matrix $C \in \mathbb{R}^{n \times (n+d)}$ holds the simple form as:

$$C = \begin{bmatrix} I_n & 0 \end{bmatrix} \tag{11}$$

In this way, we can retain the form of the cost function for further control.

For the second part of embedding function, we have three versions to represent $g_u(x_k, u_k)$. The first version is called Deep KoopmanU with Control (DKUC) algorithm. It just simplifies the function as $g_u(x_k, u_k) = u_k$, then the evolution equation degrades to Eq.(5). The sceond version is called Deep Koopman Affine with control (DKAC) algorithm, it

considers the evolution function as control affine form, so that $g_u(x_k, u_k) = g_u(x_k)u$, and the Eq.(3) is modified as:

$$z_{k+1} = K_{xx}z_k + K_{xu}g_u(x_k)u \tag{12}$$

And we parameterize the function $g_u(x_k)$ by designing an auxlliary control network ϕ, so that $g_u(x_k) = g_\phi(x_k)$. For the final version, we use the control network ϕ to approximate the function $g_u(x_k, u_k)$ directly as $g_u(x_k, u_k) = g_\phi(x_k, u_k)$, and the evolution function remains the same form as Eq.(8). It is called Deep Koopman Nonlinear (DKN) algorithm.

It can be easily seen that the DKN algorithm directly parameterizes the second part of the embedding function, raising the prediction power, but the increasing nonlinearity leaves further difficulty for control. And the DKUC algorithm assumes that the evolution function is linear with control input, which is favorable for linear control. But it is not the case of a general nonlinear system, degrading the prediction power of the Koopman operator. And DKAC algorithm kindly takes a satisfactory trade-off between the prediction power and system linearity for control. The whole framework is shown in Fig. 2.

B. Feedforward Prediction

Given the current state x_t, we can easily predict future K-steps states by the feedforward network. Besides the embedding function, we also parameterize the Koopman Operator matrix $K_x x$ and $K_x u$ by one layer linear network that $K_{xx} = A, K_{xu} = B$. Then the K-steps prediction follows:

$$\begin{aligned} z_{t+k+1} &= Az_{t+k} + Bg_\phi(x_{t+k}, u_{t+k}), k = 0, ..., K-1 \\ z_t = g_x(x_t) &= \begin{bmatrix} x_t \\ g_\theta(x_t) \end{bmatrix} \\ x_{t+k} &= Cz_{t+k} \end{aligned} \tag{13}$$

where $g_\phi(x, u) = u$ for KPUC algorithm, $g_\phi(x, u) = g_\phi(x)u$ for KPAC algorithm, and $g_\phi(x, u) = g_\phi(x, u)$ for KPN algorithm.

C. K-steps Loss Function

Standard Koopman-based algorithm first selects the embedding function and then learns the Koopman Operator matrix with linear regression for only step prediction. In this work, we learn the embedding function and Koopman Operator end-to-end instead, and we design K-steps prediction loss function for long-term prediction. Given the dataset $\left[X_i \in \mathbb{R}^{N\times n}, U_i \in \mathbb{R}^{N\times m}, i = 0, 1, 2, ..., K\right]$, we can compute the real embedded state $Z_i = g_x(X)$, and predict the K-steps sates $[\hat{Z}_i, i = 1, 2, ..., K]$ in embedding space from the initial state X_0 via Eq.(13). The loss function is designed as:

$$L(\theta, \phi) = \frac{1-\gamma^K}{1-\gamma}\Sigma_{i=1}^{K}\gamma^{i-1}MSE(Z_i, \hat{Z}_i) \tag{14}$$

where γ is the weight decay hyper-parameter, and MSE is the mean square loss function. Instead of only considering the following one-step prediction error, the K-steps loss focuses on the weighted sum of K-steps prediction error, conducive to prediction in a long time horizon.

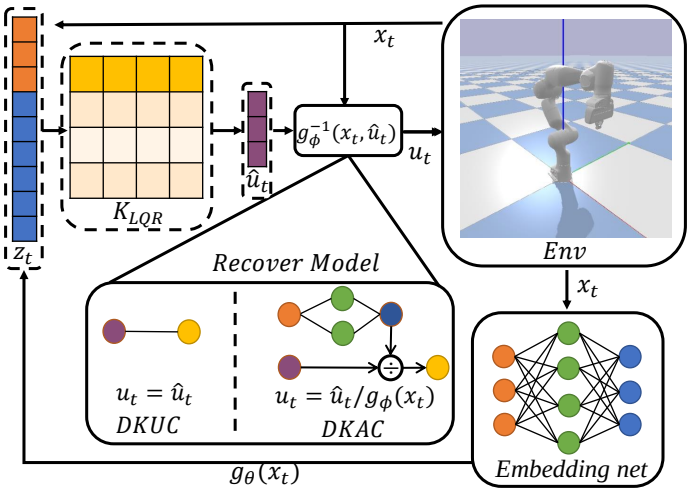

Fig. 3: Framework of LQR Control with Deep Koopman Operator

D. LQR Control

Without loss of generality, we consider quadratic loss function and optimal control with infinite horizons and ignore constraints in this work. For general nonlinear systems, given the desired state x_t^{des}, the standard optimal control problem takes the form that:

$$\begin{aligned} \min_{u_{t=1,...,\infty}} \ & \Sigma_{t=1}^{\infty}(x_t - x_t^{des})^T Q(x_t - x_t^{des}) + u_t^T R u_t \\ \text{s.t. } & x_t = F(x_{t-1}, u_{t-1}) \end{aligned} \tag{15}$$

where Q, R is the cost matrix related to state and control input. Since the evolution function is nonlinear, the optimal control problem is tough to be solved directly.

On the contrary, the Koopman Operator maps the nonlinear systems to linear dynamics in embedding space, ensuring the feasibility of linear control methods. Therefore, we apply optimal control in embedding space and consider $g_\phi(x, u)$ as the new control variable instead, in which case it degrades to the LQR control problem. Based on Eq.(13),(10), the LQR problem can be rewritten as:

$$\begin{aligned} \min_{\hat{u}_{t=1,...,\infty}} \ & \Sigma_{t=1}^{\infty}(z_t - z_t^{des})^T \hat{Q}(z_t - z_t^{des}) + \hat{u}_t^T \hat{R}\hat{u}_t \\ \text{s.t. } & z_{t+1} = Az_t + B\hat{u}_t, \quad z_0 = g_x(x_0) \\ & \hat{u}_t = g_\phi(Cz_t, u_t) \end{aligned} \tag{16}$$

where $z_t^{des} = g_\theta(x_t^{des})$, $\hat{Q} = C^T Q C$, $\hat{R} = R$ for KPUC algorithm, and $\hat{R}_t = g_\phi(x_t)^{-T} R g_\phi(x_t)^{-1}$ for KPAC algorithm. Since R is the pre-defined matrix and can be modified for different algorithms, we just assume $\hat{R} = R$ for simplification.

Eq.(16) has the closed-form solution via the LQR algorithm. After computing the LQR gain matrix K_{LQR}, the optimal control takes the form of:

$$\hat{u}_t^* = K_{LQR}(z_t - z_t^{des}) \tag{17}$$

And we can recover the actual optimal control by:

$$u_t^* = g_\phi^{-1}(x_t, \hat{u}_t^*) \tag{18}$$

where $u_t^* = \hat{u}_t^*$ for KPUC algorithm, and $u_t^* = g_\phi(x_t)^{-1}\hat{u}_t^*$ for KPAC algorithm. The detailed algorithm is illustrated in Algorithm 1.

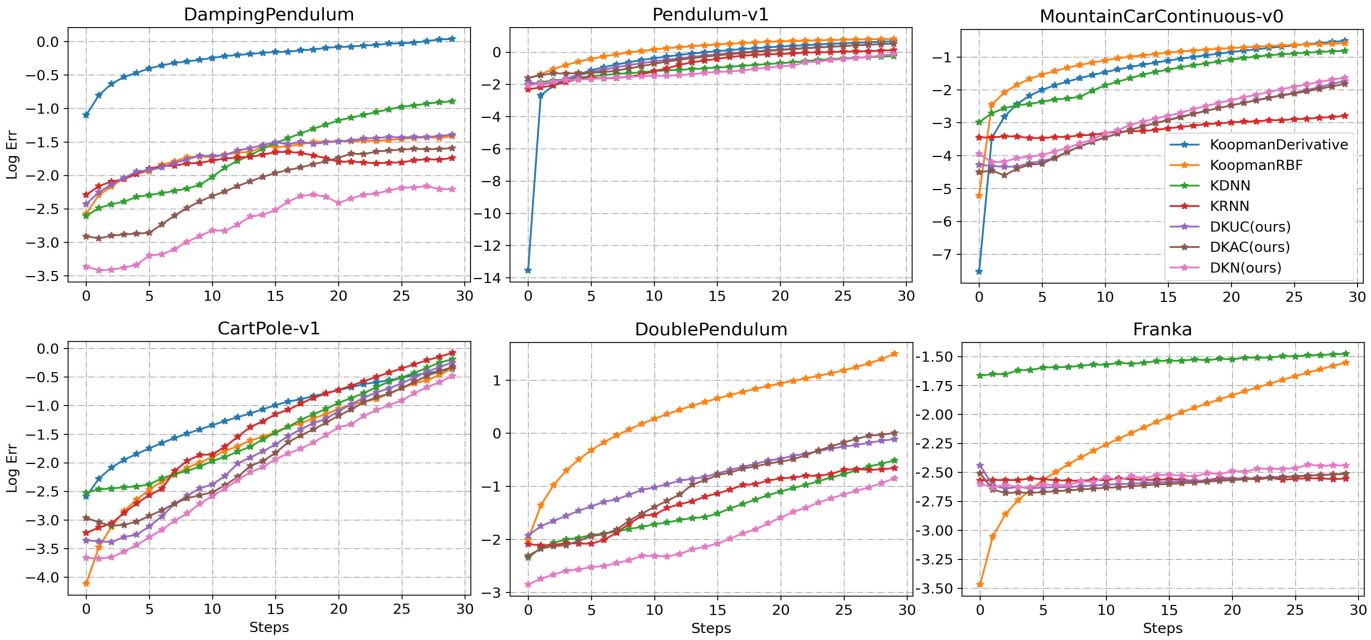

Fig. 4: Prediction Results

Algorithm 1 Deep Koopman with Control

Require: g_θ, embedding network
g_ϕ, auxlliary control network
A, B, C, Koopman Operator matrix and recover matrix
Q, R, cost matrix
x_0, x_t^{des}, initial state and desired trajectory
T, control steps
F, nonlinear evolution equation

initialize LQR gain matrix: $K_{LQR} \leftarrow LQR(A, B, Q, R)$
embed the desired state $z_t^{des} \leftarrow g_x(x_t^{des})$
reset the *env* with initial state x_0
for each t in {0...T-1} **do**
  embed current state $z_t \leftarrow g_x(x_t)$
  compute optimal control $\hat{u}_t^* \leftarrow K_{LQR}(z_t - z_t^{des})$
  recover actual optimal control $u_t^* \leftarrow g_\phi^{-1}(x_t, \hat{u}_t^*)$
  simulate the system forward $x_{t+1} \leftarrow F(x_t, u_t^*)$
end for

V. EXPERIMENTS

In this section, we conduct complete experiments to answer the following questions:

(1). How is the prediction performance of our approaches compared with previous Koopman-based approaches and deep learning-based approaches?

(2). How is the control performance of our approaches compared with previous Koopman-based methods with linear control?

(3). Is our proposed control net conducive to nonlinear system prediction and control? Specifically, do our DPAC and DKN algorithms outperform the DKUC algorithm in both prediction and control?

A. Environments

To evaluate our approach, we conduct experiments on six nonlinear dynamics simulation envoriments, including (a) DampingPendulum, (b) Pendulum, (c) MountainCarContious, (d) CartPole, (e) DoublePendulum, and (f) Franka. Among above six envrionments, b)-d) are environments modified from OpenAI gym [4], and f) Franka is the same as [21].

Besides, the dynamics of (a) DampingPendulum environment has the following form:

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta}\\ -glsin\theta + \frac{1}{ml}b\dot{\theta} + \frac{1}{ml}cos\theta u\end{bmatrix} \tag{19}$$

where m,l is the mass and length of the pendulum, b is the damping ratio, and u is the external force put horizontally (the control input).

The dynamics of (e) holds the following partial derivative equation [19]:

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) = u \tag{20}$$

where $M(\theta)$ is the mass matrix, $c(\theta, \dot{\theta})$ is the velocity-product term, $g(\theta)$ is the gravity term and $u \in \mathbb{R}^2$ is the input force.

B. K-steps Prediction

We evaluate the prediction performance of our approaches on the above six environments and compare them with previous methods: Derivative-based Koopman (Koopman-Derivative) [20], RBF-based Koopman (Koopman-RBF) [13], Deep neural networks (KDNN) and Deep recurrent networks (KRNN). For KoopmanDerivative and Koopman-RBF, we train them with linear regression as mentioned in Eq.(6). For KDNN and KRNN, we generally train a neural network to approximate the nonlinear evolution function $F(x, u)$ with K-steps Loss as in Eq.(14). We use three hidden layers and 128 hidden units, the same learning rate, and 15-steps prediction loss for all deep learning-based approaches.

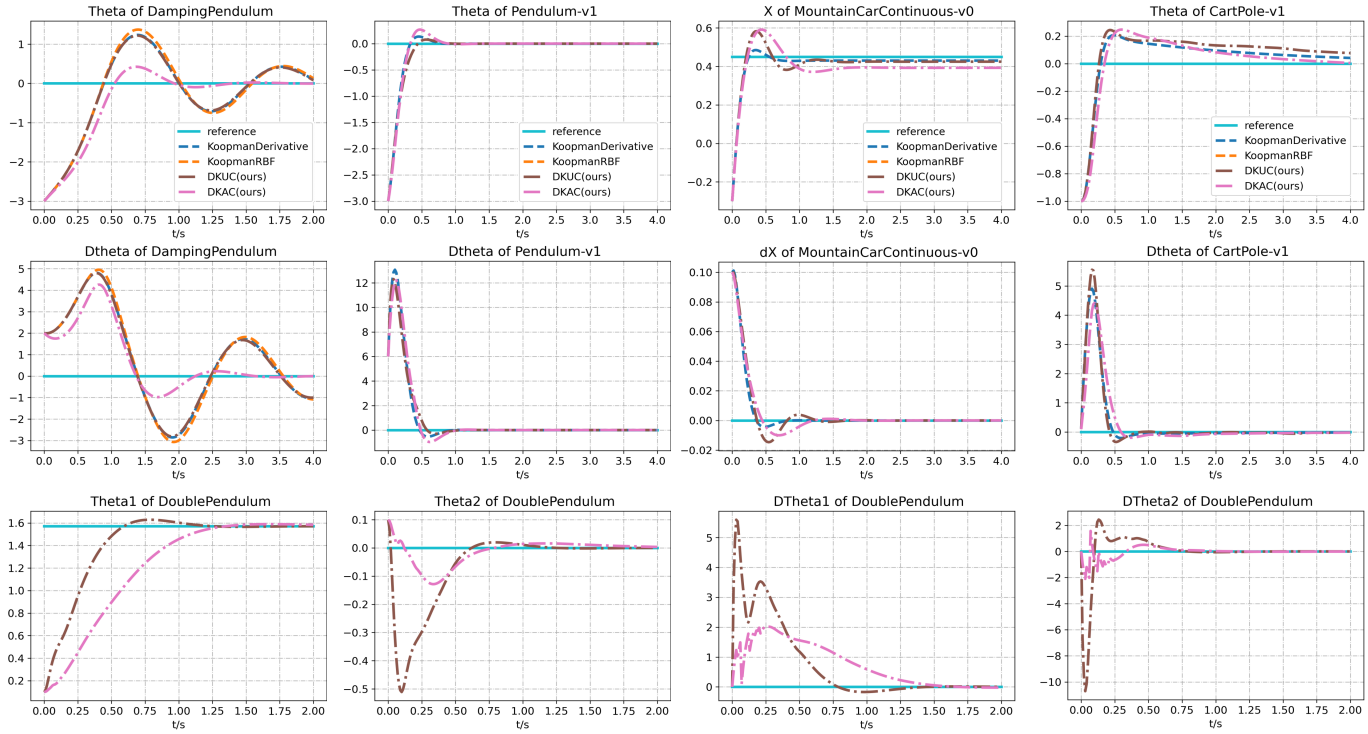

Fig. 5: Classical Control Results. Control results out of range are not plotted. For DoublePendulum, the derivative is not available, so KoopmanDerivative is not applicable. It can be seen that DKUC and DKAC achieve the best performance in all of the environments.

The final 30-steps prediction results is plotted in Fig. 4. We run four times for every environment and algorithm and plot the mean of log10 of the maximum error in 30-time horizons. And the errors at the 15th step are shown in Table I. KoopmanDerivative requires the Nth derivative of the dynamics, which is too complex or inaccessible for DoublePendulum and Franka, so we fail to conduct these two experiments for KoopmanDerivative. From the above figure and tables, we can find that: (1). KoopmanDerivative and KoopmanRBF work better at the first several steps since the least square methods only optimize the first step prediction. On the contrary, deep learning-based approaches achieve better performance in a long time horizon. (2). Our proposed algorithm DKN performs best and even better than KRNN in most cases, although KRNN takes advantage of the extra temporal information of data. (3). Our algorithms DKUC and DKAC reduce the prediction error by the degree of magnitudes than previous Koopman-based approaches (KoopmanDerivative & KoopmanRBF) and achieve comparable or even better results than fully deep learning-based approaches (KDNN & KRNN). (4). Our algorithm DKAC outperforms DKUC in prediction, especially on DampingPendulum, where the 15th step prediction error of DKAC is almost one-third of that of DKUC. It illustrates that our designed control net works for better prediction.

C. LQR Control

As for control performance, we compare our approaches with KoopmanRBF and KoopmanDerivative, which are suitable for liner control methods like LQR, while KDNN and KRNN methods are fully nonlinear, and linear control methods are inapplicable to them. Besides, since we haven't found an invertible $g_\phi(x,u)$ of the DKN algorithm, we only deploy our proposed DKUC and DKAC algorithms for control.

1) Classical Control: Environments from (a) to (e) are classical control problems with force as input. The control results are plotted in Fig. 5. Table II shows the total cost defined in Eq. (15). Table III illustrates the final state errors $\Sigma_{t=t_f-5}^{tf}|x_t - x_t^{des}|$, where NaN means that the control result is not stable and results in out-of-range cost and error.

From the above plots and tables, it can be seen that our approaches DKUC and DKAC achieve the best control performance in most of the environments, and KoopmanRBF is quite unstable since it's highly dependent on the selection of centers of RBF functions. Furthermore, we can find that DKAC outperforms DKUC and KoopmanDerivative a lot in an environment like DampingPendulum, where the control term is highly nonlinear and related to the state. Based on Fig5, only DKAC succeeds in controlling the damping pendulum to achieve the zero point, and the final state error of DKAC is reduced by order-of-magnitude than other methods, from $e-1$ to $e-3$.

To further illustrate the advantage of DKAC, for damping-pendulum we conduct control experiments on all the initial conditions $\{(\theta,\dot\theta)|\theta\in[-4,4],\dot\theta\in[-4,4]\}$, and we plot the total cost in Fig.8 and log10 of final state error in Fig.7 given the corresponding initial states, where each point in the plane is the initial state $(\theta,\dot\theta)$, and the color means the corresponding value. And blank regions in the plots mean the controller fails to lead the system to the desired state given the initial state.

TABLE I: Prediction Results at the 15th Step

	DampingPendulum	Pendulum	MountainCar	CartPole	DoublePendulum	Franka
KoopmanDerivative	6.760e-1±3.284e-2	9.634e-1±1.936e-2	6.725e-2±3.524e-4	8.626e-2±2.031e-3	-	-
KoopmanRBF	2.700e-2±9.814e-4	2.624e+0±9.173e-2	1.233e-1±9.268e-3	2.854e-2±2.031e-3	3.879e+0±7.232e-2	8.639e-3±8.657e-5
KDNN	2.517e-2±4.462e-3	9.714e-2±2.020e-2	3.509e-2±5.250e-3	2.582e-2±2.923e-3	2.636e-2±1.486e-3	2.888e-2±8.885e-4
KRNN	2.032e-2±4.695e-3	2.990e-1±8.550e-2	**6.027e-4±2.128e-5**	5.314e-2±7.360e-3	6.392e-2±9.584e-3	2.726e-3±1.187e-4
DKUC(ours)	2.835e-2±5.166e-3	5.982e-1±3.138e-2	9.714e-4±7.404e-5	1.595e-2±1.042e-3	1.510e-1±1.005e-2	2.586e-3±3.630e-4
DKAC(ours)	9.511e-3±6.465e-4	5.153e-1±1.809e-2	9.663e-4±4.724e-5	1.075e-2±7.291e-4	1.304e-1±2.537e-2	**2.485e-3±6.110e-4**
DKN(ours)	**2.575e-3±8.424e-4**	**5.026e-2±2.677e-3**	1.341e-3±4.530e-4	**8.423e-3±8.117e-4**	**7.247e-3±4.099e-3**	2.995e-3±1.705e-4

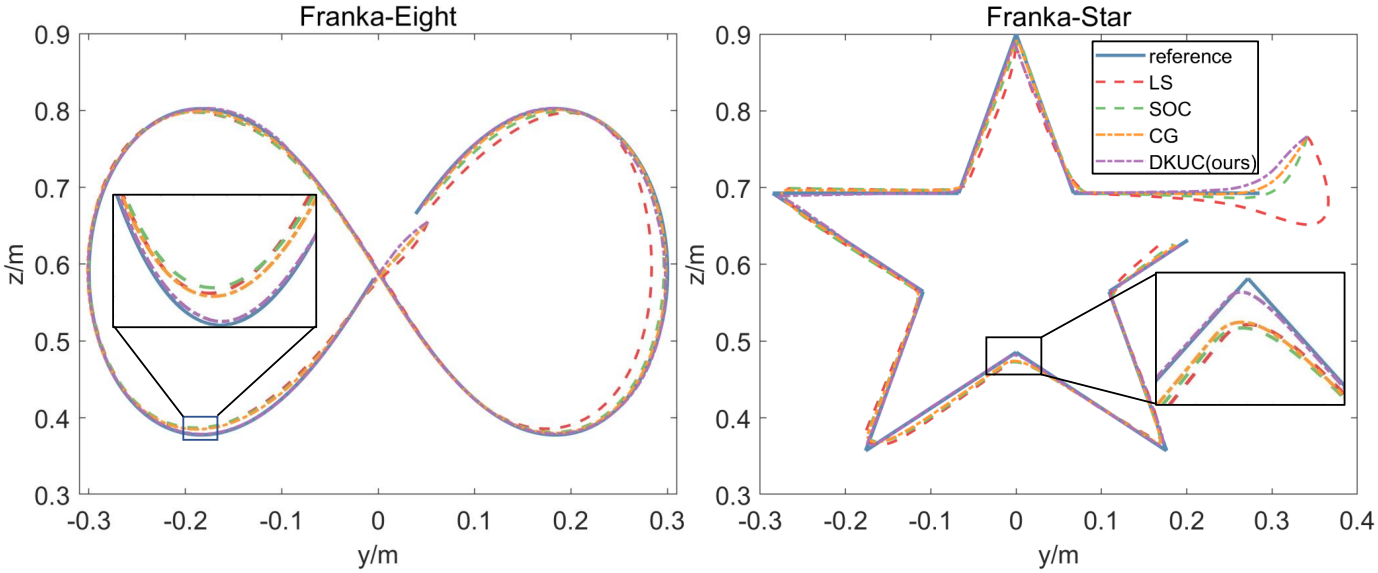

Fig. 6: Franka Control Results

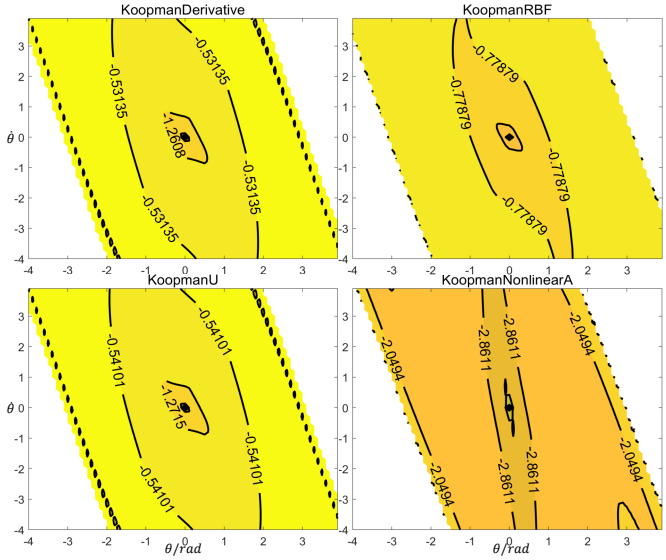

Fig. 7: Log10 Error of all initial states in DampingPendulum

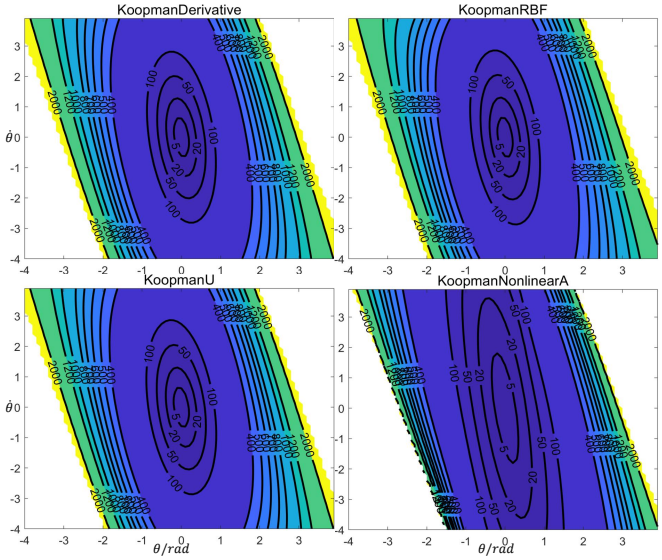

Fig. 8: Total cost of all initial states in DampingPendulum

TABLE II: The Total Costs of Classical Control

	KoopmanDerivative	KoopmanRBF	DKUC	DKAC
DampingPendulum	1196.692	1265.189	1183.103	**1053.759**
Pendulum	307.728	NaN	**305.387**	309.232
MountainCar	15.933	NaN	**13.873**	16.372
CartPole	90.827	NaN	100.781	**89.878**
DoublePendulum	-	NaN	2919.976	**2905.536**

TABLE III: The Final States Error of Classical Control

	KoopmanDerivative	KoopmanRBF	DKUC	DKAC
DampingPendulum	5.914e-1	6.211e-1	5.620e-1	**4.955e-3**
Pendulum	**1.179e-10**	NaN	1.589e-10	2.736e-9
MountainCar	**1.961e-2**	NaN	2.523e-2	5.715e-2
CartPole	2.997e-2	NaN	4.959e-2	**1.456e-2**
DoublePendulum	-	NaN	**1.222e-3**	1.318e-2

From the above plots, we can see that our DKAC algorithm succeeds in directing the system to the desired state with the final state error less than $e-2$ and less cost in most initial conditions, while other methods fail when the initial state is far away from the desired state.

TABLE IV: The Tracking Error in Franka Control

	Franka-Eight	Franka-Star
LS	2.968±90.051	3.395±90.289
SOC	2.588±90.060	2.918±90.124
WLS	233.834±914.601	262.825±916.344
CG	2.179±90.661	2.496±90.118
KoopmanRBF	359.744±9276.283	8165±922782
DKUC	**1.450±90.074**	**1.775±90.161**

2) Franka Control: For Franka Environment, our control input is the desired velocity in 100Hz, and the actual force is computed via PID in 1000Hz. Our control goal is to make the robot track the desired trajectory $x_t^{des}, t = 1, 2, ..., N$. The control term has 7 Dof, so it's hard for us to design the cost matrix R for DKAC. And since the derivative is not available for KoopmanDerivative, we only compare our DKUC algorithm with KoopmanRBF and methods in previous work [21], including least square (LS), weighted least square (WLS), constraint generation (CG), and SOC. The Franka control results are plotted in Fig.6 and we compare the tracking error $\Sigma_{t=1}^{N} ||x_t - x_t^{des}||_2$ in 10 random initial states in Table IV. We can see that our DKUC outperforms other approaches and almost reduce the tracking error by one-half.

VI. CONCLUSION

In this work, we propose an end-to-end deep learning framework to learn the Koopman embedding function and Koopman Operator together, improving the prediction power of nonlinear systems and the control performance with LQR. Furthermore, we design a control network to model the nonlinear state-dependent term related to control, enhancing the prediction performance by strengthening the expressive power, gaining even better prediction quality than the recurrent neural network. And experimental results prove that this approach is also conducive to the control of general systems with high nonlinearity on control terms. We hope that our work could facilitate the further study of real-time control on complex nonlinear systems.

REFERENCES

[1] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[2] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020.

[3] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[6] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[7] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[8] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[9] Thor I Fossen and Aslaug Grovlen. Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping. *IEEE transactions on control systems technology*, 6(1):121–128, 1998.

[10] Ramazan Gencay and Tung Liu. Nonlinear modelling and prediction with feedforward and recurrent networks. *Physica D: Nonlinear Phenomena*, 108(1-2):119–134, 1997.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[12] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[13] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[14] Rajesh Kumar, Smriti Srivastava, JRP Gupta, and Amit Mohindru. Comparative study of neural networks for dynamic nonlinear systems identification. *Soft Computing*, 23(1):101–114, 2019.

[15] Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.

[16] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229. Citeseer, 2004.

[17] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[18] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[19] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[20] Giorgos Mamakoukas, Maria L Castano, Xiaobo Tan, and Todd D Murphey. Derivative-based koopman operators for real-time control of robotic systems. *IEEE Transactions on Robotics*, 37(6):2173–2192, 2021.

[21] Giorgos Mamakoukas, Orest Xherija, and Todd Murphey. Memory-efficient learning of stable linear dynamical systems for prediction and control. *Advances in Neural Information Processing Systems*, 33:13527–13538, 2020.

[22] Shankar Sastry. *Nonlinear systems: analysis, stability, and control*, volume 10. Springer Science & Business Media, 2013.

[23] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[26] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[27] Jonathan H Tu. *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.

[28] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[29] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.