IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Monte Carlo Approach to Koopman Direct Encoding and Its Application to the Learning of Neural-Network Observables*

Itta Nozawa[1,2], Emily Kamienski[1], Cormac O'Neill[1], and H. Harry Asada[1], *Life Fellow, IEEE*

***Abstract*—This paper presents a computational method, called Bootstrapped Koopman Direct Encoding (B-KDE) that allows us to approximate the Koopman operator with high accuracy by combining Koopman Direct Encoding (KDE) with a deep neural network. Deep learning has been applied to the Koopman operator method for finding an effective set of observable functions. Training the network, however, inevitably faces difficulties such as local minima, unless enormous computational efforts are made. Incorporating KDE can solve or alleviate this problem, producing an order of magnitude more accurate prediction. KDE converts the state transition function of a nonlinear system to a linear model in the lifted space of observables that are generated by deep learning. The combined KDE-deep model achieves higher accuracy than that of the deep learning alone. In B-KDE, the combined model is further trained until it reaches a plateau, and this computation is alternated between the neural network learning and the KDE computation. The result of the MSE loss implies that the neural network may get rid of local minima or at least find a smaller local minimum, and further improve the prediction accuracy. The KDE computation however, entails an effective algorithm for computing the inner products of observables and the nonlinear functions of the governing dynamics. Here, a computational method based on the Quasi-Monte Carlo integration is presented. The method is applied to a three-cable suspension robot, which exhibits complex switched nonlinear dynamics due to slack in each cable. The prediction accuracy is compared against its traditional counterparts.**

***Index Terms*—Dynamics, Deep Learning Methods**

I. Introduction

THE Koopman operator allows us to linearize nonlinear dynamics while retaining the properties of the original nonlinear dynamical systems [1]. In the past decade, the Koopman operator theory has been extensively studied by diverse communities, including fluid mechanics, nonlinear dynamics, and system dynamics and control, and has been applied to broad areas. In robotics, the Koopman lifting linearization has made significant impacts in the modeling and control of soft robotic systems [2], [3], active learning of mobile robots [4], and dynamic modeling of switched dynamical systems [5], to name just a few. However, the prediction accuracy of Koopman lifting linearization is still limited, even for autonomous nonlinear systems. Finding an effective set of observable functions remains a challenge despite recent breakthroughs. Dynamic Mode Decomposition (DMD) and the related methods such as Extended DMD (EDMD) based on least squares estimate produce biased estimates in approximating the Koopman operator. The goal of the current work is to establish an alternative approach that can supplement the existing method and significantly improve prediction accuracy.

Neural networks have been used to find effective observable functions. To the authors' knowledge, Li, Dietrich, Bollt and Kevrekidis first introduced deep learning for finding observable functions [6]. Lusch, Kutz and Brunton used a fully-connected autoencoder for obtaining a linear dynamics and provided a procedure for determining observables based on deep neural networks [7]. Extending the network they proposed, several neural network architectures and/or algorithms have been proposed to obtain the linear dynamics and/or the control matrix [8]–[12]. The neural network-based approaches to finding observables are now widely used and are known to be effective to represent a nonlinear system with a smaller number of observables. However, the approximation accuracy is still not satisfactory in many applications, and various network architectures are being explored [8]–[12]. Additionally, neural network training, in general, faces a number of challenges, including local minima and over-fitting, although many techniques have been developed to alleviate the difficulties.

Koopman Direct Encoding (KDE) converts nonlinear dynamic equations governing a nonlinear system directly to a linear dynamic equation [13]. KDE can be combined with the deep learning method. Using the observables generated through the deep learning, we can construct a linear model based on KDE. The final layer of the deep neural network is a linear layer, the weights of which can be replaced by the ones obtained from KDE. In deep learning, the linear final layer is trained together with the hidden layers based on gradient descent, whereas the one derived from KDE is fundamentally different - computations of inner products. It can be expected that combining these two distinct methods may produce synergistic effects.

In the current work, we aim to establish a hybrid method where KDE and deep learning are integrated into simultaneous optimization of both observable functions and the linear state

Manuscript received: September 15, 2023; Revised: November 25, 2023; Accepted: December 21, 2023.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Sumitomo Heavy Industries, Ltd.

[1]The authors are with the Department of Mechanical Engineering, Massachusetts Innstitute of Technology, Cambrige, MA 02139, U.S. nozawa@mit.edu; croneill@mit.edu; emilyk@mit.edu; asada@mit.edu

[2]Itta Nozawa is with Sumitomo Heavy Industries, Ltd., Yokosuka-shi, Kanagawa 237-8555, Japan. itta.nozawa@shi-g.com

Digital Object Identifier (DOI): see top of this page.

transition matrix. While the learning based on the gradient descent and error backpropagation alone may get stuck at a local minimum, the proposed method may get rid of the local minima or at least find a smaller local minimum during the learning process, according to the significant reduction prediction error in simulation experiments. KDE and error backpropagation complement each other in seeking effective observables and the Koopman linear model.

KDE entails computations of inner products in the space of independent state variables. There are two approaches to the computation of the inner products. One is the case where explicit nonlinear dynamic equations are not available. Ng and Asada [13], [14] developed a data-driven algorithm for computing the inner products from data. Assuming the integrand of the inner product is Riemann integrable, the authors calculate multi-dimensional volumes for the Riemann sum based on Delaunay triangulation. This calculation allows us to compute KDE in the data-driven manner, but it is not applicable to high-dimensional nonlinear systems because the computational complexity $\mathcal{O}(N^{\lfloor d/2 \rfloor})$ exponentially increases depending on the system dimension d and the number of sampling points N. The other approach is to exploit a given set of nonlinear dynamic equations, which is the method used in the current work. The algorithm of Quasi-Monte Carlo (QMC) integration will be used for the inner product computation, which is applicable to higher-order nonlinear dynamical systems than that of Delaunay triangulation.

II. Preliminaries

In this subsection, we briefly introduce the Koopman Direct Encoding method for representing a nonlinear, autonomous system as a linear state equation in discrete time. First, consider a discrete-time dynamical system given by

$$x_{t+1} = F(x_t) \tag{1}$$

where $x \in \mathbb{X} \subset \mathbb{R}^d$ denotes a d-dimensional state vector, which is involved in a dynamic range of the considered system, $\mathbb{X}$. The index t means the discrete time, $t = 0, 1, \cdots$. $F : \mathbb{X} \to \mathbb{X}$ is a nonlinear self-map, called a state transition function.

Let $\mathbb{H}$ be a Hilbert space and let us introduce a function $g(x) \in \mathbb{H} : \mathbb{X} \to \mathbb{R}$, that maps the state variable x to a real number, called an observable. Let $g_1, g_2, \cdots$ be an independent and complete set of observable functions spanning $\mathbb{H}$, and define an infinite dimensional column vectors $z(x_t)$ as follows.

$$z_t = z(x_t) = [g_1(x_t), g_2(x_t), \cdots]^T \tag{2}$$

From Eq.(1), the infinite dimensional vector at time $t+1$, $z(x_{t+1})$ can be written as

$$z_{t+1} = z(x_{t+1}) = [g_1(F(x_t)), g_2(F(x_t)), \cdots]^T \tag{3}$$

Underpinned by the Koopman Operator theory, the time evolution of the infinite dimensional vector $z(x_t)$ is expressed as a linear state transition:

$$z_{t+1} = A z_t \tag{4}$$

where z_t is treated as a new state vector in the lifted space.

In this paper, we adopt the inner-product formulation of the above state transition matrix A, referred to as Koopman Direct Encoding (KDE) [13]. Post-multiplying $z(x_t)^T$ to both sides of Eq. (4) and integrating it over the dynamic range of the system $\mathbb{X}$, we can obtain the following relations:

$$\int_{\mathbb{X}} z(x_{t+1}) z(x_t)^T dx = A \int_{\mathbb{X}} z(x_t) z(x_t)^T dx \tag{5}$$

Note that each element involved in the integrals represents the inner product of two functions. Denoting these by infinite-dimensional matrices, R and Q,

$$Q = AR \tag{6}$$

$$R = \begin{pmatrix} \langle g_1, g_1 \rangle & \langle g_1, g_2 \rangle & \cdots \\ \langle g_2, g_1 \rangle & \langle g_2, g_2 \rangle & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{7}$$

$$Q = \begin{pmatrix} \langle g_1 \circ F, g_1 \rangle & \langle g_1 \circ F, g_2 \rangle & \cdots \\ \langle g_2 \circ F, g_1 \rangle & \langle g_2 \circ F, g_2 \rangle & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{8}$$

where $\langle , \rangle : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ is an inner product expressed as:

$$\langle f_1, f_2 \rangle = \int_{\mathbb{X}} f_1(x) f_2(x) dx \tag{9}$$

where $f_1, f_2 \in \mathbb{H}$. The important point is that we can compute $A = QR^{-1}$ directly because R is a non-singular matrix. We can confirm R's non-singularity by noting that $g_1, g_2, \cdots$ are independent in the inner-product space, by their definition.

Practically, we also need to truncate the number of observables to a finite number that is sufficient to linearize the given nonlinear dynamics. This may cause the matrix R to be ill-conditioned. As such, the pseudo-inverse can be used in lieu of R^{-1}, or the matrix A can be determined by solving a least squares estimate problem associated with $Q = AR$.

III. Koopman Direct Encoding Using Quasi-Monte Carlo Integration

Although the KDE has been analytically formulated in the previous section, the practical problem of KDE is how to numerically compute the inner products of observables and how to use it to linearize high-dimensional systems. To relax this practical limitation for the computational complexity, we introduce the Quasi-Monte Carlo (QMC) integration for the inner-product computation in this section.

Monte Carlo (MC) integration is the numerical method for stochastic-integral computation based on a uniform distribution and is one of the common methods for solving multivariate integration while avoiding the curse of dimensionality. Let $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ be an integrand and $x = [x_n^{(1)} x_n^{(2)} \cdots x_n^{(d)}] \in \mathbb{R}^d$ be a n-th order random-variable vector sampled from a uniform distribution. Then the MC integration I can be described as

$$I \approx \left(\prod_{p=1}^{d} \left(x_{max}^{(p)} - x_{min}^{(p)} \right) \right) \left(\frac{1}{N} \sum_{n=1}^{N} f\left(x_n^{(1)}, x_n^{(2)}, \cdots, x_n^{(d)} \right) \right) \tag{10}$$

where $x_{max}^{(p)} - x_{min}^{(p)}$ denotes the integration interval for the p-th element. Note that MC integration makes it possible

to perform multivariable integration without computing d-dimensional volumes, which are required for the Riemann sum, so that it can avoid the heavy computation. Because the error convergence of the MC integration is proportional to $1/\sqrt{N}$, the integration asymptotically approaches the true value given that there are a sufficient number of sampling points. It is possible to improve upon this convergence rate by using pseudo-random variables which are generated based on low-discrepancy sequences [15]. This numerical integration is called QMC integration and the error convergence is known to be proportional to $1/N$, because the sequence gives sampling points that distribute more uniformly over the range than the one obtained by random sampling from a uniform distribution. In other words, it can inhibit a temporary bias or sampling to nearly identical points. There are several low-discrepancy sequences used for QMC integration, and we adopted the Halton sequence [16] in this paper.

In the following, we explain how QMC integration is used for inner-product calculations for KDE. First, the pseudo-random variables from the Halton sequence are generated and scaled to fit the dynamic range of the system $\mathbb{D}$. While the defined domain for $\mathbb{D}$ is not necessary in the mathematical description in the original theory [13], which can handle infinite spaces, the practical numerical treatment requires limiting the state space to a user-defined range. This is because the dataset needs to satisfy the condition that $x_{t+1} = F(x_t) \in \mathbb{D}$ if $x_t \in \mathbb{D}$ for KDE computation, and this precondition is necessary to generate pseudo-random variables that are enclosed in $\mathbb{D}$.

To ensure that the variables are closed in $\mathbb{D}$, we adopt a maximum total energy constraint. Suppose that we have a priori knowledge of the dynamical system we consider, then we can easily obtain the Hamiltonian H of the system. If we set the maximum value of H, we will be able to collect the sample points from the subspace where $H \leq H_{max}$ is satisfied. Note that H_{max} is a hyperparameter that can be determined according to the region of the subspace we would like to consider.

Based on the above, we will explain an example to illustrate the sampling method using the Halton sequence with the energy constraint. Suppose we consider the example of a single rigid-rod pendulum. The equation of motion is described as follows:

$$mL\ddot{\theta} = -mg\sin\theta \tag{11}$$

where θ, m, g and L denote angle of the pendulum, mass, acceleration of gravity, and a rigid cable length, respectively. The Hamiltonian of the system in the case of $m = 1$ and $L = 1$ is described as:

$$H_{pend} = \frac{\dot{\theta}^2}{2} + g(1 - \cos\theta) \tag{12}$$

If we set the maximum total energy of the system we consider to be obtained at $\theta = 3.1$ and $\dot{\theta} = 0$, then $H_{pend,max} \approx 19.6$. Then, the quasi-random points generated from the Halton sequence are sampled from the region where $H_{pend}(\theta, \dot{\theta}) \leq H_{pend,max}$ is satisfied. Fig. 1 shows the 2D histogram of the system mentioned above. As can be seen, the quasi-random points are distributed through the region inside the isoenergetic surface of $H_{pend,max}$. Note that this sampling method can only be applied to stable or marginally stable systems because the precondition no longer holds if a current state x_t is mapped outside of $\mathbb{D}$ by $F(\cdot)$.

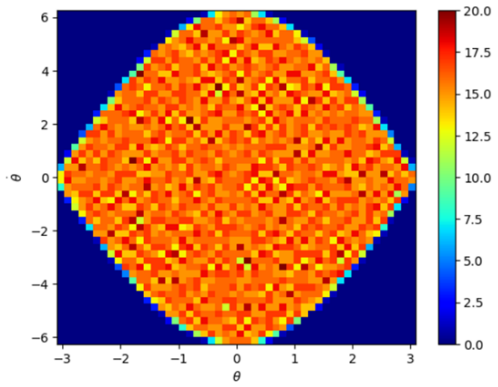

Fig. 1. The example of the 2D histogram of the sampling points. (N=25,788, bin=50×50) The color bar denotes the frequency.

On the basis of the sampling method mentioned above, we can rewrite the inner-product calculation for KDE as follows:

$$\langle g_j, g_i \rangle \approx \left(\prod_{p=1}^{d} \left(x_{max}^{(p)} - x_{min}^{(p)} \right) \right) \left(\frac{1}{N} \sum_{n=1}^{N} g_j(x_n) g_i(x_n) \right) \tag{13}$$

$$\langle g_j \circ F, g_i \rangle \approx \left(\prod_{p=1}^{d} \left(x_{max}^{(p)} - x_{min}^{(p)} \right) \right) \left(\frac{1}{N} \sum_{n=1}^{N} g_j\left(F(x_n)\right) g_i(x_n) \right) \tag{14}$$

where x_n is a pseudo-random variable generated from the Halton sequence. Here, the key point is that the limitation on the number of system dimensions is relaxed by computing KDE using QMC integration.

IV. Hybrid KDE-Deep Learning Method

This section presents the architecture of neural networks to compute a Koopman operator called Bootstrapped Koopman Direct Encoding (B-KDE). The framework of this computation is composed of two processes: (a) Pretraining and (b) Training. The details of the computation are shown in Algorithm 1. In the pretraining process, the deep encoder network for obtaining the Koopman operator is trained as shown in Fig. 2. This network is a fully-connected encoder network with an additional linear layer. This network also corresponds to the encoder and the linear layer in [7]. The weight matrix of this final layer at the end of the pretraining process corresponds to the initial estimate of the Koopman matrix: A_{pre}.

The input dataset for this network are the state variables $x_t \in \mathbb{R}^d$. In this study, a self regularized non-monotonic activation function (Mish) [17] is employed as the activation function for the hidden layers (HL). The intermediate outputs that pass through the first HL to the k-th HL are observables, $g(x_t) \in \mathbb{R}^m$, and $z_t = [x_t, g(x_t)]^T$ are input to the linear layer. The linear layer's outputs are linearly combined and fed into the output layer without a bias term. The activation function of the output layer is simply an identity function. With respect to the outputs, the following Mean Squared Error (MSE) loss

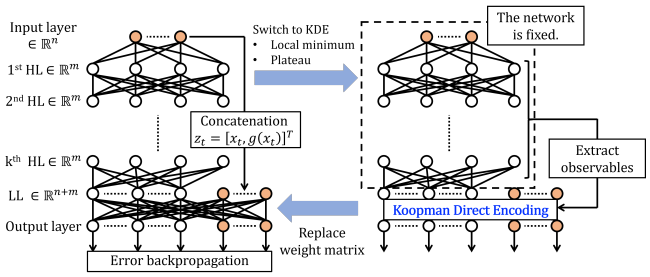

Fig. 2. The neural network architecture of the Bootstrapped Koopman Direct Encoding.

function is minimized based on error backpropagation, and it allows the network to estimate $\hat{z}_{t+1} \in \mathbb{R}^{d+m}$ from $z_t \in \mathbb{R}^{d+m}$.

$$L = ||\hat{z}_{t+1} - z_{t+1}||_2^2 = ||A_{pre} z_t - z_{t+1}||_2^2 \tag{15}$$

where z_{t+1} can be found by feeding x_{t+1} into the network and extracting the output from the linear layer. Note that we only know the ground truths of the state variables, and there is no exact information of ground truths for the observables. In other words, $g(x_t)$ is only a function estimated based on the training of the network, and there is no a priori information about the best set of functions for the given system. However, this loss function enforces the observables to enforce linearity by penalizing deviation from the linear relationship in predicting the one-step-ahead transition of the observables as shown in Eq. (15). This treatment is practically useful when using deep learning to find appropriate observables to linearize a given nonlinear dynamics. However, this loss function enforces the observables to guarantee linearity to some extent by considering the linearity of the observables at one step ahead as shown in Eq. (15). This treatment is practically useful when using deep learning to find appropriate observables to linearize a given nonlinear dynamics. As pointed out in [7], it is also possible to extract x_t from $g(x_t)$ by implementing a decoder network, but in the case of this network, the result from the output layer contains the state variables, so the decoder architecture is unnecessary.

In the training phase, the weight matrix of the linear layer is replaced with the Koopman matrix calculated based on KDE with the QMC integration, A_{KDE} as shown in Fig. 2. The reason why we performed the pretraining is that it is necessary to obtain a suitable set of observables to linearize the dynamics to some extent for computing KDE. Using a network that has not been pre-trained at all may cause the computation of Eq. (6) to diverge. The KDE computation is shown to allow us to obtain a more accurate prediction of $\hat{z}_{t+1}$ than the one computed by EDMD with the same observables [14]. However, there is room to finely optimize observables for KDE because it only calculates a Koopman matrix once. B-KDE makes it possible to tune both a Koopman matrix and observables based on KDE. As shown in Fig. 2, B-KDE can iteratively compute the Koopman matrix by KDE with the QMC integration and optimize the observable selection in order to obtain a Koopman matrix with better estimation accuracy. Let this optimized Koopman matrix be $A_{B\text{-}KDE}$ for convenience. In the training process, the loss function is the same as Eq. (15) except for using A_{KDE} instead of A_{pre}. In the next section, we will report the details about the result by showing two illustrative examples.

Algorithm 1 Bootstrapped Koopman Direct Encoding.

Input: x_t, x_{t+1}
Output: $A_{B\text{-}KDE}$, $g(\cdot)$

Initialization
set the neural network shown in Fig. 2
PRETRAINING PROCESS
for each pretraining epoch **do**
  calculate $g_t = g(x_t)$ & $g_{t+1} = g(x_{t+1})$
  calculate $z_t = [x_t, g(x_t)]^T$ & $z_{t+1} = [x_{t+1}, g(x_{t+1})]^T$
  calculate $\hat{z}_{t+1} = A_{pre} z_t$
  calculate $MSE = ||\hat{z}_{t+1} - z_{t+1}||_2^2$
  optimize weight matrices based on backpropagation
  if epoch == end of pretraining **then**
    compute A_{KDE} based on Eq. (13) and Eq. (14)
    replace a weight matrix of LL with A_{KDE}
    Break
  end if
end for
TRAINING PROCESS
for each training epoch **do**
  calculate $g_t = g(x_t)$ & $g_{t+1} = g(x_{t+1})$
  calculate $z_t = [x_t, g(x_t)]^T$ & $z_{t+1} = [x_{t+1}, g(x_{t+1})]^T$
  calculate $\hat{z}_{t+1} = A_{KDE} z_t$
  calculate $MSE = ||\hat{z}_{t+1} - z_{t+1}||_2^2$
  optimize weight matrices based on backpropagation
  if epoch == user-defined epoch **then**
    recalculate A_{KDE}
    replace A_{KDE} with new A_{KDE}
    Break
  end if
end for
extract $A_{B\text{-}KDE}$ from the weight matrix of LL
return $A_{B\text{-}KDE}$, $g(\cdot)$

V. Results and Discussion

A. Pendulum with a Damping Component

First, we checked the effectiveness of B-KDE by demonstrating lifting linearization for a single pendulum with a damping component. The equation of motion is described as follows:

$$mL\ddot{\theta} = -mg\sin\theta - b\dot{\theta} \tag{16}$$

where θ denotes angle of the pendulum. In this example, $m = 1$, $L = 1$ and $b = 0.3$ are used as the model parameters. As seen from Eq. (16), this system can be described by θ and $\dot{\theta}$, and all the trajectories finally converged to an equilibrium point at $\theta = 0$ and $\dot{\theta} = 0$ because of the damping component. The training data was collected based on the sampling method we previously explained. In this case, the maximum total energy constraint was the same as the example of the simple pendulum because the difference between the equations are only a dissipation function with respect to the total energy. Under the above condition, x_t and $x_{t+1} = F(x_t)$

with N=51,558 different initial conditions were computed based on Eq. (16). Additionally, 259 trajectories with different initial conditions were also simulated for the validation dataset, and each trajectory has 102 time-steps ($\Delta t = 0.1$), including the initial conditions, in order to check the future prediction accuracy based on the Koopman matrix.

Table I shows parameters used for training the B-KDE model to linearize the pendulum system. As explained earlier, the training data contained 51,558 points for each x_t and x_{t+1}, and x_{t+1} was used as the ground truth for the loss calculation and the KDE computation. The learning rates for the pretraining and training were set to be different because B-KDE worked as a final adjustment of the observables and therefore the lower learning rate was considered to be more appropriate following pretraining.

TABLE I
NEURAL NETWORK PARAMETERS AND CHARACTERISTICS FOR B-KDE TO LINEARIZE THE PENDULUM MOTION.

Parameters and Characteristics	Value
The Number of Hidden Layers (HL)	3
Activation Function for HL	Mish
Width of 1st HL	100
Width of 2nd HL	100
Width of 3rd HL	100
Width of Linear Layer	102
Learning Rate for Pretraining	5.0×10^{-4}
Learning Rate for B-KDE	5.0×10^{-10}
Pretraining Epochs	400
Training Epochs	200

Fig. 3 shows the transition of MSE of the state variables during training process of B-KDE. In this case, the training process was switched from the pretraining to the training of B-KDE at 400 epochs. As can be seen in Fig. 3, each MSE drastically decreased when the Koopman matrix was calculated based on KDE. This gap at 400 epochs indicates the effectiveness of KDE. Moreover, the MSE gradually decreased as training of B-KDE progressed. The weight matrix of the linear layer was updated for a second time at 500 epochs based on KDE using the tuned observables in order to reduce the loss. As shown in the inset of Fig. 3, the MSE clearly decreased by performing the second KDE. Fig. 4 shows the comparison of the typical estimates among B-KDE, KDE and pretraining. It can be seen that the estimation accuracy improved in the order of pretraining, KDE and B-KDE. This result implies that we can obtain a more accurate Koopman matrix by tuning observables via B-KDE. It may also be possible to improve the accuracy of pretraining by improving the loss function, but this is outside the scope of this paper.

Fig. 5 depicts a comparison between the ground truth and the estimate computed with $A_{B\text{-}KDE}$ in the $\theta - \dot{\theta}$ phase plane and time evolution of θ and $\dot{\theta}$. The time evolution with 259 different initial θ_0 and $\dot{\theta}_0$ from the evaluation dataset was displayed in superposition. As can be seen from Fig. 5, it is possible to estimate the 101 steps-ahead time evolution for any initial conditions, indicating that the ground truths and the estimated values were qualitatively in good agreement over a long time horizon. In Koopman operator theory, the

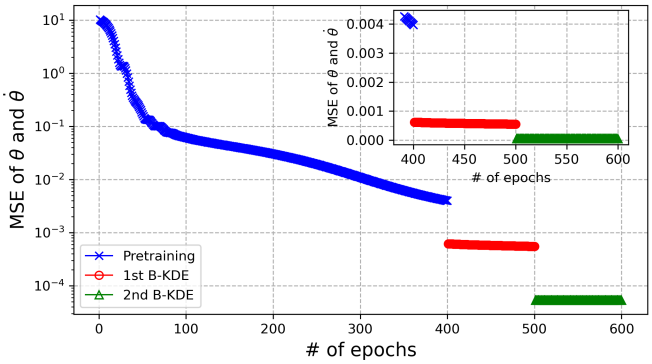

Fig. 3. The comparison among the MSEs of the state variables during training process of B-KDE. The inset shows an enlarged view in the linear scale after 390 epoch.

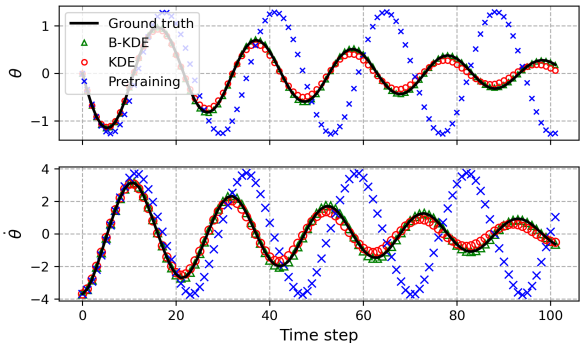

Fig. 4. The comparison of typical estimated time evolution among B-KDE, KDE, and pretraining. The black line denotes the ground truth.

T-step-ahead estimate, $\hat{z}_T$, is simply expressed as $\hat{z}_T = A^T z_0$, but the observables were recalculated at each step in our case. The reason why we performed this calculation is due to the following three points: (a) The estimation errors of the observables $||\hat{g}_T - g(x_T)||_2^2$ are considered to be larger than those of the state variables $||\hat{x}_T - (x_T)||_2^2$ because there are no exact ground truths for the observables. (b) What we would like to predict is not the time evolution of observables, but that of state variables. (c) Recalculating observables at each time step can suppress the effect of these unstable poles due to the numerical artifacts in the computation of Eq. (6) on the prediction. The same calculation flow was used for the latter case study in order to evaluate estimation errors.

Fig. 6 shows the result of the quantitative evaluation of prediction accuracy based on Mean Absolute Error (MAE) for each time step. For comparison, the EDMD results are also shown when 100 radial basis functions were used as the observables for EDMD. The solid lines denote MAE of θ and $\dot{\theta}$, and the areas filled with colors mean the ranges of standard deviation of $\pm 1\sigma$. It can be seen from Fig. 6 that a sufficiently small estimation MAE with respect to the maximum and minimum values of θ and $\dot{\theta}$ in the phase plane can be achieved even 101 steps ahead. The MAE appeared to slightly oscillate because the maximum state values varied with the period of the pendulum motions.

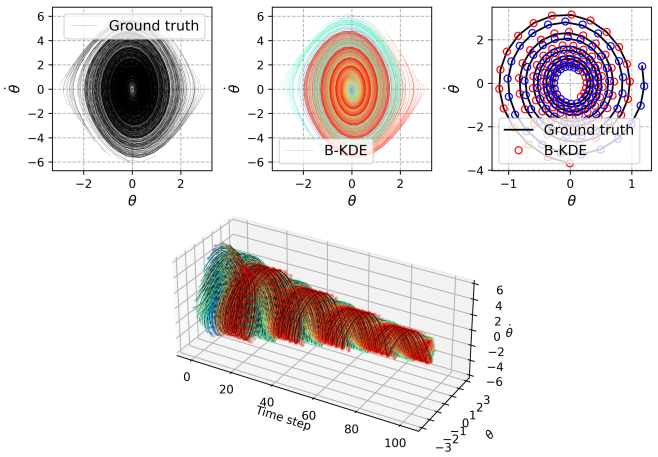

Fig. 5. The comparison between the ground truths (upper left), all the estimates based on B-KDE (upper middle), and the typical estimates based on B-KDE (upper right) in $\theta - \dot{\theta}$ plane, and time evolution of state variables (bottom). The color denotes the different initial conditions.

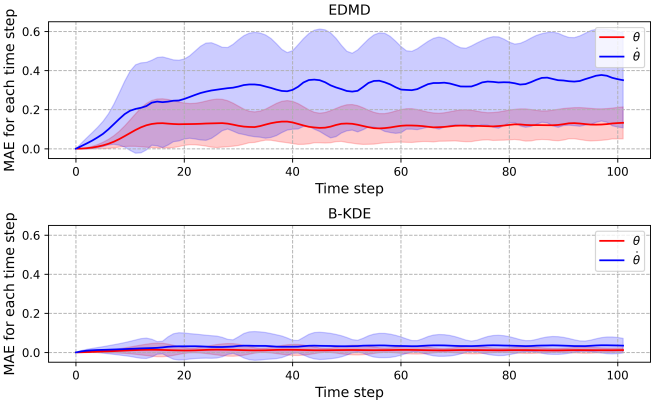

Fig. 6. The Mean Absolute Error (MAE) of θ and $\dot{\theta}$ for each time step based on EDMD (top) and B-KDE (bottom). The filled areas denote the ranges of standard deviation of $\pm 1\sigma$.

B. Three-cable pendulum

The results of the pendulum example indicated that B-KDE allows us to linearize the nonlinear dynamical system, but this system has a relatively benign nonlinearity. In order to check whether this method can linearize more complex nonlinear dynamical systems with higher dimensions, we attempted to use B-KDE to linearize a three-cable pendulum, which is a 6-dimensional switched nonlinear system.

Fig. 7 depicts the schematic diagram of the three-cable pendulum. The pendulum consists of a point-mass bob with mass $m = 1$ hanging from three cables with unstretched length L_0. The coordinate of the bob is $P = (x, y, z)$. The pivot points of each cable, A_0, B_0 and C_0 are on the circle with a radius of 0.5 m on the z = 0 plane. A_0, B_0 and C_0 form an equilateral triangle and the centroid matches with O, which is the origin of the coordinate system. The unstretched length of each cable is set to be $L_0 = 1$ and the elongation of each cable dL_i is defined as follows:

$$dL_{id} = \sqrt{(x - x_{id})^2 + (y - y_{id})^2 + (z - z_{id})^2} - L_0 \qquad (17)$$

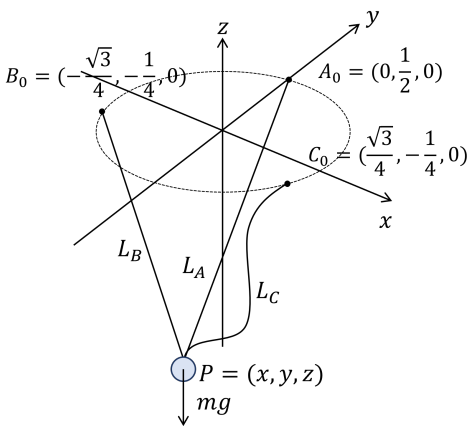

Fig. 7. Three-dimensional three-cable pendulum.

where *id* corresponds to the cable identifier A, B or C. In this model, assuming that the tension of the cables follow Hooke's law, the product of the spring constant $k = 2000$ and dL_{id} for each cable gives the tension applied to the cable. Additionally, the taut-slack conditions of the cables are switched based on Eq. (17). If $dL_{id} \geq 0$, then the cable is taut, and tension of the cable *id* is generated, whereas if $dL_{id} < 0$, then the tension of the cable is set to be zero. Therefore, the $2^3 = 8$ different taut-slack conditions vary depending on the condition of each of the three cables. The damping component is assumed to be proportional to be the velocity of the bob, $v = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$. The kinematic energy, K and potential energy, U are described as follows:

$$K = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) \qquad (18)$$

$$U = mgz + \frac{1}{2}(k_A dL_A^2 + k_B dL_B^2 + k_C dL_C^2). \qquad (19)$$

The Lagrangian, $\mathscr{L}$ of the system can be written as $\mathscr{L} = K - U$. Because the system is assumed to lose its energy due to friction, the dissipation function, E_0 was also considered as follows:

$$E_0 = \frac{1}{2} b(\dot{x}^2 + \dot{y}^2 + \dot{z}^2). \qquad (20)$$

The Euler-Lagrangian equations can be obtained with respect to $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ and were solved as an initial value problem of the system of ordinary differential equations.

The training data were collected based on the pseudo-random sampling method with the total energy constraint. In this example, H_{max} is set based on the potential energy of a free-falling mass point from a starting position of $P = (0, 0, 0)$. Under the above condition, x_t and $x_{t+1} = F(x_t)$ with N=36,074 different initial conditions were computed based on the governing equations. Additionally, 191 trajectories with different initial conditions were computed as for the validation dataset. The time step and the number of time steps for the simulation were set to be 0.02 and 200, respectively.

Table II shows parameters used for training the B-KDE model to linearize the three-cable pendulum system. Because this system has a more challenging nonlinearity and higher dimensions than the pendulum system, we increased the number of HLs and the width of the linear layer.

TABLE II
NEURAL NETWORK PARAMETERS AND CHARACTERISTICS FOR B-KDE TO LINEARIZE THE THREE-CABLE PENDULUM SYSTEM.

Parameters and Characteristics	Value
The Number of Hidden Layers (HL)	4
Activation Function	Mish
Width of 1st HL	100
Width of 2nd HL	100
Width of 3rd HL	100
Width of 4th HL	500
Width of Linear Layer	506
Learning Rate for Pretraining	1.0×10^{-2}
Learning Rate for B-KDE	1.0×10^{-8}
Decay Rate of Scheduler for Pretraining	0.999
Pretraining Epochs	2500
Training Epochs	1000

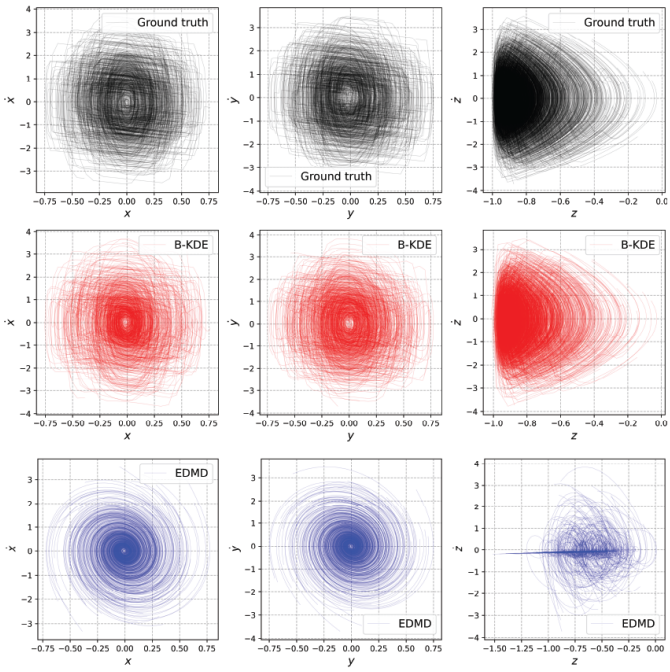

Fig. 8. The comparison between the ground truths (black) and the estimates based on B-KDE (red) and EDMD with radial basis functions (blue) from the viewpoints of $x-\dot{x}$, $y-\dot{y}$, $z-\dot{z}$ phase planes.

Fig. 8 shows the result of lifting linearization for the three-cable pendulum system. Using 506 lifted state variables, including both state variables and observables, we were able to obtain prediction results that were in qualitatively good agreement with the ground truth data, even if the system dimension is high. For comparison, the results of EDMD with 500 radial basis functions are also shown. Additionally, Fig. (9) shows the comparison of typical estimates among ground truths, B-KDE, and EDMD to make it easy to compare the results intuitively.

Fig. 10 shows the quantitative evaluation of prediction accuracy based on MAE for each time step. As can be seen from the Fig. 10, MAE increased as the prediction were repeated several tens of time steps ahead because of the accumulation of prediction errors. However, the MAE values are smaller than the maximum and minimum values of the

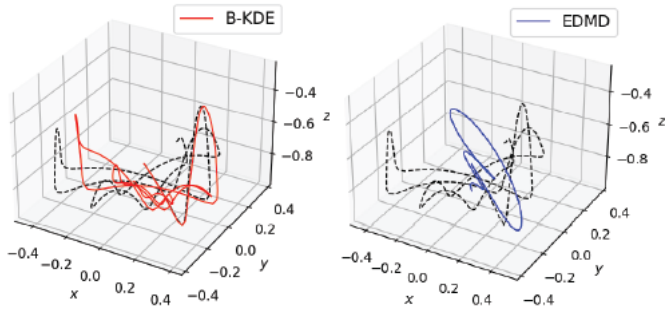

Fig. 9. The comparison of typical estimated time evolution between B-KDE and EDMD. The black line denotes the ground truth.

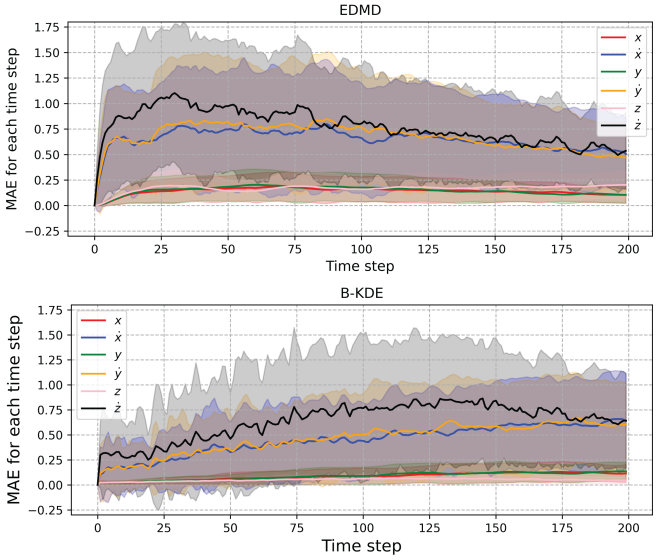

Fig. 10. The quantitative evaluation of prediction accuracy based on MAE for each state variable at each time step based on EDMD (top) and B-KDE (bottom). The filled areas denote the ranges of standard deviation of $\pm1\sigma$.

dynamic range seen in the Fig. 8, and this result indicated that complex nonlinear systems such as this switched system can be linearly approximated based on B-KDE.

C. Limitations and Open Questions

Here we would like to discuss limitations and open questions about B-KDE.

What is the upper bound of the system dimensions?

The degree to which B-KDE can handle high-dimensional systems is governed by the system dimensions where the inner product calculation can be performed. To the authors' knowledge, the QMC performance regarding the system dimensions highly depends on the integrand and the type of low discrepancy sequences used for sampling points. In addition, it is also important to consider at what dimensions the low discrepancy sequences can uniformly generate sample points. Depending on the dimension of the system being considered, sequences other than the Halton sequence may be appropriate in order to maintain uniformity of the sampling points in high dimensional systems.

How many sampling points are required for B-KDE?

Because B-KDE generates observable functions based on the deep encoder architecture, it is unknown that what the ground truths of the inner products for each pair of observables are. This indicates that we cannot estimate the integration errors, and hence, the number of sampling points cannot be determined based on the errors. However, it is possible to use our proposed method for calculating KDE based on QMC integration for the naive KDE computation. In this case, we can adopt functions whose inner products we can analytically compute, such as elementary functions, as observables. We believe that this treatment allows us to estimate the suitable number of sampling points based on the integration error analysis.

How many times B-KDE must be repeated?

In the computational framework of B-KDE, tuning of the observables and A_{KDE} was iteratively repeated, and the calculation was completed when the loss function reached a plateau. In this study, the indicators of how often A_{KDE} should be updated and how much of the same A_{KDE} should be used to tune the observations are determined by the user. It would be desirable to be able to automatically determine these values from the gradients and/or absolute values of the loss function.

How to tune the network hyperparameters

The neural network for B-KDE has some hyperparameters, including the widths of the hidden layers, activation function, loss function, and learning rates. These hyperparameters need to be tuned for a given nonlinear system for which linearization is being performed. It is desirable to devise a method to adjust hyperparameters so that a given accuracy can be achieved with as small a number of observables as possible. A systemic way to accomplish this hyperparameter tuning remains elusive.

How to remove unstable poles due to the numerical artifacts?

As mentioned before, A_{B-KDE} sometimes contains unstable poles and it requires recalculating observables in order to avoid the divergence of the predicted results. From the viewpoint of control applications, we believe that it is better for us to linearly compute multi-step predictions based on A_{B-KDE} instead of recalculating them. Although we can apply model predictive control (MPC) with a short time horizon or nonlinear MPC for the B-KDE model. One solution is to apply singular value decomposition to A_{B-KDE} and remove unstable poles based on the threshold for the singular values. In addition, since prior works, such as in the reference [18], provide methods to suppress unstable poles, combining B-KDE with such a method can probably solve this problem.

VI. Conclusions

In this paper, we presented a novel neural network structure called Bootstrapped Direct Encoding (B-KDE), which allows us to compute a Koopman matrix with high accuracy. This method consists of combining Koopman Direct Encoding (KDE) with a deep encoder network, which makes it possible to finely tune both observables and a Koopman matrix by iterating the optimization process. Additionally, by introducing Quasi-Monte Carlo integration for the inner-product computation of KDE, we relaxed the practical restriction in our prior works which limited the number of system dimensions. Through the examples of a single pendulum and a three-cable pendulum, it was shown that it is possible to linearize nonlinear dynamical systems with B-KDE even in the case of a high-dimensional switched system.

References

[1] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[2] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2021.

[3] D. A. Haggerty, M. J. Banks, E. Kamenar, A. B. Cao, P. C. Curtis, I. Mezić, and E. W. Hawkes, "Control of soft robots with inertial dynamics," *Science Robotics*, vol. 8, no. 81, p. eadd6864, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.add6864

[4] 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.

[5] J. Ng and H. H. Asada, "Model predictive control and transfer learning of hybrid systems using lifting linearization applied to cable suspension systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 682–689, 2022.

[6] 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, p. 103111, 10 2017. [Online]. Available: https://doi.org/10.1063/1.4993854

[7] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, nov 2018. [Online]. Available: https://doi.org/10.1038%2Fs41467-018-07210-0

[8] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*, 2019, pp. 4832–4839.

[9] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1890–1895.

[10] N. S. Selby and H. H. Asada, "Learning of causal observable functions for koopman-dfl lifting linearization of nonlinear controlled systems and its application to excavation automation," *IEEE Robotics and Automation Letters*, vol. 6, pp. 6297–6304, 2021.

[11] 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.

[12] S. P. Nandanoori, S. Guan, S. Kundu, S. Pal, K. Agarwal, Y. Wu, and S. Choudhury, "Graph neural network and koopman models for learning networked dynamics: A comparative study on power grid transients prediction," *IEEE Access*, vol. 10, pp. 32 337–32 349, 2022.

[13] H. H. Asada, "Global, unified representation of heterogenous robot dynamics using composition operators: A koopman direct encoding method," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 5, pp. 2633–2644, 2023.

[14] J. Ng and H. H. Asada, "Data-driven encoding: A new numerical method for computation of the koopman operator," *IEEE Robotics and Automation Letters*, vol. 8, pp. 3940–3947, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:255941823

[15] E. Braaten and G. Weller, "An improved low-discrepancy sequence for multidimensional quasi-monte carlo integration," *Journal of Computational Physics*, vol. 33, no. 2, pp. 249–258, 1979. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999179900196

[16] M. Berblinger, C. Schlier, and T. Weiss, "Monte carlo integration with quasi-random numbers: experience with discontinuous integrands," *Computer Physics Communications*, vol. 99, no. 2, pp. 151–162, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465596001312

[17] D. Misra, "Mish: A self regularized non-monotonic activation function," in *British Machine Vision Conference*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221113156

[18] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning stable models for prediction and control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2255–2275, 2023.