# Optimal Control of Quadrotor Attitude System Using Data-driven Approximation of Koopman Operator ⋆

**Ketong Zheng** * **Peng Huang** *,** **Gerhard P. Fettweis** *,**

* *Vodafone Chair Mobile Communications Systems, Technische Universität Dresden, 01069 Dresden, Germany*
** *Barkhausen Institut, 01062 Dresden, Germany*
*(E-mail: {ketong.zheng, peng.huang}@ifn.et.tu-dresden.de, gerhard.fettweis@tu-dresden.de)*

**Abstract:** The nonlinear dynamics has posed a great challenge in the optimal control of quadcoptors. This paper applies data-driven methods to approximate the Koopman operator of the quadrotor attitude control system. The Koopman operator is a linear operator with infinite dimensions that can advance the nonlinear state dynamics linearly in a lifted space without compromising the operation range. The function basis and the propagation matrices to approximate the Koopman operator are learned interactively through Deep Neural Network (DNN). Simulations with the quadrotor attitude model in $SO(3)$ are carried out to verify the model precision and the tracking performance using Linear Quadratic Regulator (LQR) with the lifted linear system. We compared the results with first-order approximation and Extended Dynamic Mode Decomposition (eDMD), which is another data-driven method that selects the function basis from a fixed library. Both data-driven approaches have notable advantages over the first-order approximation, while the system learned through the DNN has better precision and control performance than the eDMD.

*Keywords:* Nonlinear System, Optimal Control, Data-driven Control, Koopman Operator, Deep Learning, Dynamic Mode decomposition, Unmanned Aerial Vehicle

## 1. INTRODUCTION

The rapid growth of communications technologies and the realization of the Tactile Internet (Fettweis, 2014) have brought forward many Unmanned Aerial Vehicle (UAV) applications in Cyber-physical Systems (CPS). The control and collision avoidance of UAVs are the main problems in most scenarios, such as transportation, search and rescue, which require multiple UAVs to cooperate and fly in a complex environment.

Many model-based approaches assume linearized dynamics around a fixed point to avoid difficulties carried by the nonlinearity. This simplified model can readily achieve performance like formation control and multi-UAV dynamic encirclement (Schmitt et al., 2022; Hafez et al., 2015). Since the linear model only adds affine constraints in the optimal control, it can even be applied to swarm-sized systems without adding too much computational complexity (Soria et al., 2021b,a). However, with a linearized model, the system states are limited to a very tight range in order to guarantee the model precision. Thus, we cannot fully explore the UAV dynamics to make agile maneuvers. A straightforward solution is to design the controller directly using the nonlinear model. In the work of Romero et al. (2022), the control problem and the time allocation problem are solved concurrently with the full nonlinear dynamics to guide the quadrotor flying through multiple waypoints. Nevertheless, the brutal use of nonlinear dynamics in optimal control will make the problem nonconvex, so that real-time applications will become challenging. Although differential flatness is another effective method to directly control the quadrotor with full nonlinear dynamics, a well planned trajectory is required and it is difficult to cope with input and state limits (Mellinger and Kumar, 2011; Faessler et al., 2017; Morrell et al., 2018; Sun et al., 2022).

Another major drawback of the model-based approach is the requirement for exact knowledge of system dynamics and parameters, which is unreliable in many conditions. This brings us to data-driven strategies, where the controller is designed using state measurements only. This paper mainly focuses on the data-driven approximation of the Koopman operator, which is an infinite-dimensional linear operator introduced by Koopman (1931). The system can evolve linearly through the Koopman operator by lifting the original nonlinear system into a Hilbert space formed by infinite basis functions. Since the infinite dimension operator cannot be directly used in linear

⋆ The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK001K; and the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

control, data-driven methods can be applied to find a finite number of dominant eigenfunctions and eigenvalues to approximate the Koopman operator. There are two commonly used data-driven approaches for finding the eigenfunction basis. The first one is the Extended Dynamic Mode Decomposition (eDMD), where the basis functions are selected from a fixed library. Williams et al. (2015) first used eDMD to approximate nonlinear autonomous systems, and the effectiveness in controlled systems with low dimensional embedded nonlinearity was shown later on by Brunton et al. (2016); Korda and Mezić (2018); Igarashi et al. (2020). The function basis can also be learned through Deep Learning, in which each forward path represents one basis function. The propagation matrices that advance the dynamics can be either learned together with the function basis (Lusch et al., 2018), or updated separately during the training process (Han et al., 2020; Wang et al., 2021). However, the performance was only tested under low dimensional nonlinear systems in these works, and the feasibility of quadrotor dynamics, where the nonlinearity preserves in a higher dimensional space, is not validated. Although Chen et al. (2023) and Zinage and Bakolas (2021) approximated the Koopman operator of quadrotors within $SO(3)$ and $SE(3)$, the methods are not data-driven, and the derived propagation matrices are state-dependent, which will cause a problem for controller design.

Therefore, the main contributions of this paper are stated below: we set up a DNN architecture to learn the Koopman approximation of the quadrotor attitude control system inside Lie group $SO(3)$ from data. The propagation matrices in the lifted space are updated interactively with the function basis in each training epoch, which is shown to have a faster convergence speed and lower converged value than training the whole system together. First, the model precision using the DNN is compared with the eDMD and the first-order approximation around the equilibrium. Then, we further compare the tracking performance of these models with linear controllers designed through Linear Quadratic Regulator (LQR). The results show the benefit of using the DNN approximated Koopman operator to design the linear controller of the quadrotor attitude system.

The remainder of this paper is organized as follows: the Koopman operator basics and the quadrotor attitude dynamics are briefly introduced in Section 2. In Section 3, the eDMD and DNN architectures are specified. Section 4 shows the simulation parameters and results. Finally, the conclusion and possible future work are discussed in Section 5.

## 2. PRELIMINARIES

### *2.1 Notations*

$L^p(X)$ denotes the Lebesgue spaces that all elements are integrable with the order of $p$ inside $X$. $\mathcal{H}$ represents the Hilbert space. The overlined function $\overline{f}$ implies its complex conjugate. The braces $<,>$ stands for the inner product, and $\circ$ is the function composition, such that $[f \circ g](x) = f[g(x)]$. The mapping $[\cdot]_\times : \mathbb{R}^3 \mapsto \mathfrak{so}(3)$ is a skew-symmetric matrix, where $\mathfrak{so}(3)$ is the Lie algebra of the Lie group $SO(3)$. The trigonometric functions are simplified as $s(\cdot), c(\cdot), t(\cdot)$ which correspond to $\sin(\cdot), \cos(\cdot), \tan(\cdot)$, respectively. $D = diag(a_1, ..., a_n) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $D_{ii} = a_i, i \in \mathbb{N}_+$. The set of integers between $a$ and $b$ are denoted as $[a, b]_d$. Finally, $\|\cdot\|_F$ represents the Frobenius norm of the corresponding matrix, and $A^\dagger$ indicates the Moore–Penrose inverse of matrix $A$.

### *2.2 Koopman Operator Review*

We first consider the nonlinear autonomous system in discrete-time

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k), \tag{1}$$

where $\mathbf{x}_k \in X \subset \mathbb{R}^n$. The function $f$ is the state transition function. We also assume function $g \in L^2(X) \subset \mathcal{H}$ to be the observation function inside the Hilbert space spanned by a set of square-integrable orthonormal basis $\varphi = \{\varphi_1, \varphi_2, ...\}$. Therefore, the function $g(\mathbf{x})$ can be decomposed as

$$\begin{aligned} g(\mathbf{x}) &= \sum_{j=1}^{\infty} < g(\mathbf{x}), \overline{\varphi_j}(\mathbf{x}) > \varphi_j(\mathbf{x}) \\ &= \sum_{j=1}^{\infty} \left[ \varphi_j(\mathbf{x}) \int_X g(\mathbf{x}) \cdot \overline{\varphi_j}(\mathbf{x}) d\mathbf{x} \right]. \end{aligned} \tag{2}$$

According to (2), the function composition can be written as

$$\begin{aligned} g \circ f(\mathbf{x}) &= \sum_{j=1}^{\infty} \left[ \varphi_j(\mathbf{x}) \circ f(\mathbf{x}) \int_X g(\mathbf{x}) \overline{\varphi_j}(\mathbf{x}) d\mathbf{x} \right] \\ &= [\mathcal{K} g](\mathbf{x}), \end{aligned} \tag{3}$$

where $\mathcal{K}$ is an infinite dimensional Koopman operator that encodes the state transition function within the function basis (Brunton et al., 2016). Since the Koopman operator is a linear operator, it can be characterized by eigenfunctions $\varphi_j(\boldsymbol{x})$ and eigenvalues $\lambda_j$ as follows:

$$[\mathcal{K}\varphi_j](\mathbf{x}) = \lambda_j \cdot \varphi_j(\mathbf{x}), \quad j \in \mathbb{N}_+. \tag{4}$$

It is clear that the observations can be made as the function basis as long as they are linearly independent. Thus, we aim to find a finite number of observations $G(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_N(\mathbf{x})]^T$ and state transition matrix $A \in \mathbb{R}^{N\times N}$ that can well approximate the Koopman operator from data. Therefore, the state transition in the lifted space $\mathbb{R}^N$ is given by

$$G \circ f(\mathbf{x}_k) = G(\mathbf{x}_{k+1}) = [\mathcal{K}G](\mathbf{x}_k) \approx A \cdot G(\mathbf{x}_k). \tag{5}$$

Generalizing the Koopman operator into the autonomous systems with control input is trivial. According to Korda and Mezić (2018), we need to extend the original state space into $\boldsymbol{\chi}_k = [\mathbf{x}_k; \mathbf{U}_k] \in \mathbb{R}^n \times l(\mathcal{U})$, where $\mathbf{U}_k = (\mathbf{u}_{k+i})_{i=0}^{\infty}$ is the control sequence with $\mathbf{u}_{k+i} \in \mathcal{U} \subset \mathbb{R}^m$, while $l(\mathcal{U})$ is the space of all control sequences. Then, the extended dynamics can be expressed by

$$\boldsymbol{\chi}_{k+1} = \tilde{f}(\boldsymbol{\chi}_k) = \begin{bmatrix} f(\mathbf{x}_k, \mathbf{U}_k(0)) \\ \mathcal{S}\mathbf{U}_k \end{bmatrix}, \tag{6}$$

where $\mathcal{S} : l(\mathcal{U}) \mapsto l(\mathcal{U})$ is a left-shift operator such that $\mathcal{S}\mathbf{U}_k = \mathbf{U}_{k+1}$, and $\mathbf{U}_k(0)$ represents the first element in $\mathbf{U}_k$. In this case, the same idea approximating the Koopman operator in an autonomous system without control input can be applied.

### 2.3 Quadrotor Attitude Dynamics in SO(3)

According to (Sabatino, 2015; Chen et al., 2023), the attitude control dynamics is given by the following equation.

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & s(\phi)t(\theta) & c(\phi)t(\theta)\\ 0 & c(\phi) & -s(\phi)\\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)}\end{bmatrix} \cdot \begin{bmatrix}{}_B\omega_x\\ {}_B\omega_y\\ {}_B\omega_z\end{bmatrix},$$

$$\mathcal{J} \cdot {}_B\dot{\boldsymbol{\omega}} = \boldsymbol{M} - [{}_B\boldsymbol{\omega}]_\times \cdot \mathcal{J} \cdot {}_B\boldsymbol{\omega}, \tag{7}$$

with $\mathbf{M}$ corresponding to the total torque in the body frame, which can be expressed as

$$\boldsymbol{M} = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix} bl(\Omega_3^2 - \Omega_1^2)\\ bl(\Omega_4^2 - \Omega_2^2)\\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2)\end{bmatrix}. \tag{8}$$

The Euler angles (roll, pitch, yaw) are represented by $[\phi, \theta, \psi]^T$, ${}_B\boldsymbol{\omega} = [{}_B\omega_x, {}_B\omega_y, {}_B\omega_z]^T$ stands for the angular velocity within the body frame, $b, l, d \in \mathbb{R}_+$ are constant model parameters, and $\Omega_i$ indicates the angular speed of each rotor. $\mathcal{J} = diag(I_x, I_y, I_z)$ represents the diagonal inertia matrix. Here, we assume that we have the direct control over the torque on each axis, which means $\boldsymbol{u} = [\tau_x, \tau_y, \tau_z]^T$.

## 3. DATA-DRIVEN APPROXIMATION OF THE KOOPMAN OPERATOR

In this section, two data-driven approaches named eDMD and DNN to approximate the Koopman operator with control inputs are shown in detail.

### 3.1 Extended Dynamic Mode Decomposition

After extending the original state space with the space of all control sequences, the nonlinear dynamics with control can be advanced using (6). Since there is no need to infer future inputs from the current state, which also betrays the causality, the extended state $\boldsymbol{\chi}_k$ can be simplified into $\boldsymbol{\chi}_k = [\boldsymbol{x}_k; \boldsymbol{U}_k(0)] \in \mathbb{R}^{n+m}$, such that we only focus on the relation between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{\chi}_k$. Assume a controlled trajectory with $p+1$ samples, where the input dataset is shown in the form $\Gamma_{(n+m)\times p} = [\Gamma_x; \Gamma_u] = [\boldsymbol{\chi}_1, \boldsymbol{\chi}_2, ..., \boldsymbol{\chi}_p]$, where $\Gamma_x = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_p]$ is the lumped original states, $\Gamma_u = [\boldsymbol{U}_1(0), \boldsymbol{U}_2(0), ..., \boldsymbol{U}_p(0)]$ is the lumped inputs. The label dataset that only contains the original states are expressed as $Y_{n\times p} = [\boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_{p+1}]$. With the Dynamic Mode Decomposition (DMD), the projection matrix $\tilde{A}$ between $\Gamma$ and $Y$ can be directly found by solving the following least-square problem:

$$\tilde{A} = [A, B] = \underset{A,B}{\operatorname{argmin}} \ \|Y - [A, B] \cdot \Gamma\|_F. \tag{9}$$

Nevertheless, the DMD only observes the original states, meaning that $G(\boldsymbol{x}) = I_{n\times n}\boldsymbol{x}$, which is far from enough to approximate the Koopman function basis. Therefore, with eDMD, we can select additional nonlinear observation functions from a fixed function library or just by intuition. In this paper, we choose observations from the TPS-RBF (Thin Plate Spline Radial Basis Function), which is one of the most commonly used function libraries given as follows (Korda and Mezić, 2018):

$$g_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{p}_i\|^2 \cdot \log(\|\boldsymbol{x} - \boldsymbol{p}_i\|), \tag{10}$$

where $\boldsymbol{p}_i \in \boldsymbol{X}$ are fixed points which selected randomly from the state space. With the additional observations, the linear relation can be approximated better in the lifted space $\boldsymbol{z} = G(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}^N$. In many researches (Igarashi et al., 2020; Lusch et al., 2018; Han et al., 2020), the original state $\boldsymbol{x}$ is be retrieved from $\boldsymbol{z}$ by solving another least-square problem:

$$\min_{C_{n\times N}} \ \sum_{i=1}^{p} \|\boldsymbol{x}_i - C \cdot \boldsymbol{z}_i\|_F. \tag{11}$$

However, by forcing the first $n$ observations to be the original state $g_i(\boldsymbol{x}) = \boldsymbol{x}, i \in [1, n]_d$ while others remain the same as (10), the $C$ matrix can be directly written as $C = [\boldsymbol{I}_{n\times n} | \boldsymbol{0}_{n\times(N-n)}]$.

Let $\tilde{\boldsymbol{\chi}}_k = [\boldsymbol{z}_k; \boldsymbol{U}_k(0)]$, $\tilde{\Gamma} = [\tilde{\Gamma}_x; \Gamma_u] = [\tilde{\boldsymbol{\chi}}_1, \tilde{\boldsymbol{\chi}}_2, ..., \tilde{\boldsymbol{\chi}}_p]$, where $\tilde{\Gamma}_x$ and $\Gamma_u$ are the lumped lifted states and the lumped inputs, respectively. $\tilde{Y} = [\boldsymbol{z}_2, \boldsymbol{z}_3, ..., \boldsymbol{z}_{p+1}]$. For a large data set ($p \gg N$), deriving the analytical solution of (9) in the lifted space $\tilde{A} = \tilde{Y} \cdot \tilde{\Gamma}^\dagger$ would be computationally complex. As long as $\tilde{\Gamma}$ has independent rows, the solution can be simplified into

$$\tilde{A} = \tilde{Y} \cdot \tilde{\Gamma}^T \cdot (\tilde{\Gamma} \cdot \tilde{\Gamma}^T)^{-1}, \tag{12}$$

so that the singular value decomposition only happens on a $N$ by $N$ matrix (Korda and Mezić, 2018).

### 3.2 Deep Neural Network

It is obvious that sometimes the selected functions from a fixed library are not good enough to approximate the Koopman operator; an alternative approach is to use DNN to identify proper basis functions and the propagation matrix in the lifted space. This can be done either together in one backpropagation or by an interactive update. Although the gradient calculation within each backpropagation depends on the propagation matrix $\tilde{A}$, which changes every epoch, it is easy to show that this update will contribute to the convergence speed. Now assume the following loss function

$$J_k = h(\tilde{\Gamma}(\boldsymbol{w}_k, \boldsymbol{\beta}_k), \tilde{Y}(\boldsymbol{w}_k, \boldsymbol{\beta}_k), \tilde{A}_k) \tag{13}$$

that is monotonically decreasing with the prediction error in the lifted space

$$e_k = \left\|\tilde{Y}(\boldsymbol{w}_k, \boldsymbol{\beta}_k) - \tilde{A}_k \cdot \tilde{\Gamma}(\boldsymbol{w}_k, \boldsymbol{\beta}_k)\right\|_F, \tag{14}$$

with $k \in \mathbb{N}_+$ indicates the learning time step and $\boldsymbol{w}, \boldsymbol{\beta}$ represent the weight and bias parameters in the DNN. Let $\tilde{A}_k^+$ denotes the updated propagation matrix using $\tilde{\Gamma}_k$ and $\tilde{Y}_k$ through (12). Since (12) always minimizes (9) in the lifted space, it is easy to generalize

$$e_k^+ = \left\|\tilde{Y}(\boldsymbol{w}_k, \boldsymbol{\beta}_k) - \tilde{A}_k^+ \cdot \tilde{\Gamma}(\boldsymbol{w}_k, \boldsymbol{\beta}_k)\right\|_F \le e_k, \tag{15}$$

such that

$$J_{k+1}^+ - J_k \le J_{k+1} - J_k, \tag{16}$$

where $J_{k+1}^+$ is the loss at time-step $k+1$ with the updated propagation matrix $\tilde{A}_k^+$. In fact, we found that updating the propagation matrix after each backpropagation will not only converge the learning problem faster but also lead to a lower value. Finally, the following loss function is applied for the training of the DNN:

$$J = \alpha_1 \cdot \left\|\tilde{Y} - \tilde{A} \cdot \tilde{\Gamma}\right\|_F + \alpha_2 \cdot \left\|Y - C \cdot \tilde{Y}\right\|_F + \alpha_3 \cdot \|\boldsymbol{W}\|_2^2, \tag{17}$$

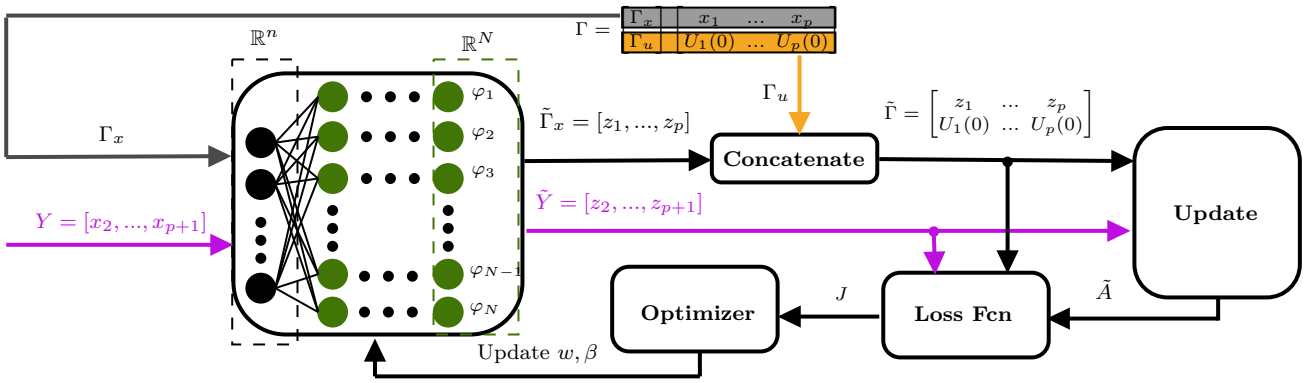


Fig. 1. DNN training loop.

**Algorithm 1** DNN training loop

**Preparation:**
Get training data: $\Gamma = [\Gamma_x; \Gamma_u], Y$
Initialize DNN model($\boldsymbol{w}, \boldsymbol{\beta}$)
Initialize $\tilde{A}$
Set $C = [I_{n\times n}|0_{n\times(N-n)}]$
**Training:**
**for** $k$ in range(epoch) **do**
  Derive $\tilde{\Gamma}_x = \text{DNN}(\Gamma_x)$
  $\tilde{\Gamma} = [\tilde{\Gamma}_x; \Gamma_u]$
  Derive $\tilde{Y} = \text{DNN}(Y)$
  Without grad():
    **update** $\tilde{A} = \tilde{Y} \cdot \tilde{\Gamma}^T \cdot (\tilde{\Gamma} \cdot \tilde{\Gamma}^T)^{-1}$
  Calculate loss function:
    $J = \alpha_1 \|\tilde{Y} - \tilde{A} \cdot \tilde{\Gamma}\|_F + \alpha_2 \|Y - C \cdot \tilde{Y}\|_F$
      $+\alpha_3 \|\boldsymbol{W}(\boldsymbol{w}, \boldsymbol{\beta})\|_2^2$
  **update** $\boldsymbol{w}, \boldsymbol{\beta}$ ▷ backpropagation

where $\boldsymbol{W}$ is the vector of all training parameters and $\alpha_1, \alpha_2, \alpha_3$ are coefficients that weighting the prediction cost, reconstruction cost, and overfitting cost, respectively. Algorithm 1 and Fig. 1 show the detail of the training process.

## 4. SIMULATION RESULTS

The training data is first generated using FORCESPRO, which is a very efficient and fast solver for the mathematical optimization problem, especially for Nonlinear Model Predictive Control (NMPC) (Zanelli et al., 2020). Then the Koopman approximation is carried on through both the eDMD and the DNN with the training data set, and the model precision is verified on another test data set. Finally, the tracking performance with the LQR controller designed using the learned propagation matrix is compared.

### *4.1 Training Data Generation*

The training data (quadrotor state trajectories) are derived by tracking the reference trajectories using the nonlinear model in (7) and (8). The optimal control inputs are calculated by solving a NMPC problem using FORCESPRO with parameters indicated by Table 1. We set two types of reference trajectories; each type contains 400 trajectory segments with 501 time steps per segment. The training data set contains 75% of the overall segments, while the rest 25% are selected as the test data set. For each segment, the first 500 steps are taken as the training inputs, while the corresponding labels are formed by applying a left shift operator on this segment (therefore, 400,000 input-label pairs in total). In *Type A*, the initial state is randomly initialized for each segment. Every 50 time-steps, we also change the reference points randomly within the state limitations. The weight matrices for the NMPC tracking problem also vary from different segments. In *Type B*, the initial state always starts from the origin. Each trajectory segment only has one randomized reference point. The agent is controlled back to the origin after it stabilizes around the reference point. We generate the training data in this way because it can explore the state space with not only a broader span (*Type A*), but also a common operational range (*Type B*).

### *4.2 Model Training Precision*

The model is first trained with the eDMD and the DNN using 600 training trajectory segments. The precision is then checked by calculating the step-wise Root Mean Square Error (RMSE) over the 200 testing trajectories.

*Extended Dynamic Mode Decomposition* The fixed points for the eDMD in (10) are randomly generated inside $\mathbb{R}^6$ space following the state limits in Table 1. After augmenting the original state space with (10), the matrix $\tilde{A}_{edmd}$ can be derived using (12). The additional dimensions are first set to 150, which means the lifted space has 156 states in total, whereas the first six states represent the original Euler angles and body frame angular velocities. Fig. 2 compares the eDMD model precision with first-order approximation around the origin of two types of test data sets. It is obvious that the eDMD augmented model has almost precision one order of magnitude higher than the first-order approximation making a continuous prediction for 300 steps. With *Type A* data set where the states start and end randomly within limits, the eDMD surpasses the first-order approximation initially because the operating points are not always close to the equilibrium. With *Type B* data set where the initial and final states are always located at the origin, the eDMD has a slightly lower precision than the first-order approximation at the initial stage ($\leq$ 30 steps) since the higher-order terms of the nonlinearity almost fade away.

Table 1. System and NMPC settings

| Param | | Param | |
|---|---|---|---|
| Sampling period ($T_s$) | 0.01 s | Roll & pitch angle ($\phi, \theta$) | $(-\frac{\pi}{2}, \frac{\pi}{2})$ rad |
| Prediction steps ($N_p$) | 50 | Yaw angle ($\psi$) | $[-\pi, \pi]$ rad |
| Inertia ($I_x, I_y$) | $7.5 \times 10^{-2}$ kg m$^2$ | Body frame angular velocity ($_B\omega_x$, $_B\omega_y$, $_B\omega_z$) | $[-1.25\pi, 1.25\pi]$ rad s$^{-1}$ |
| Inertia($I_z$) | $7.5 \times 10^{-2}$ kg m$^2$ | Input torque ($\tau_x, \tau_y$) | $[-0.6, 0.6]$ N m |
| | | Input torque ($\tau_z$) | $[-1, 1]$ N m |

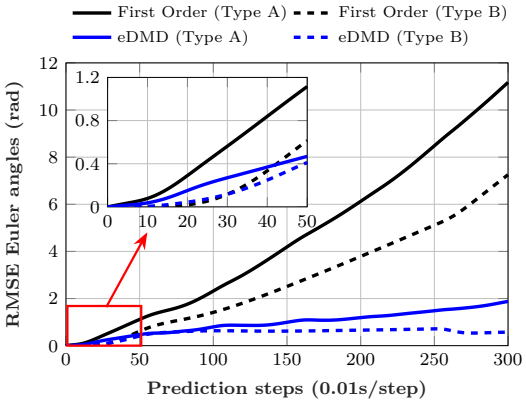


Fig. 2. Comparison of model precision between the first order approximation and the eDMD within $\mathbb{R}^{156}$ lifted space over two test data sets (*Type A* and *B*). Trajectories in *Type A* start and end at random points in the state space, while trajectories in *Type B* always start and end at the origin.

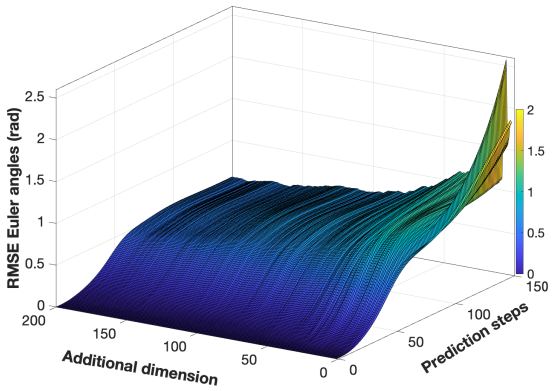


Fig. 3. The RMSE over *Type B* test set (the initial state always starts from the origin) using eDMD with regard to prediction steps and additional dimensions.

The benefits of adding additional states are also explored in this paper. The decrease of RMSE with the number of augmented states is shown in Fig. 3. It can be seen that the prediction error decreases exponentially with the number of additional dimensions. However, when the additional dimensions reach 100, the model precision almost stops increasing due to the limitation of the training data and the selected function library.

*Deep Neural Network* The following DNN is defined to train a better function basis: the neural network has four fully connected layers, where the width of the input layer is 6, and the widths for other layers are set to 150. Each layer has the form of $f_l(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ enters into a tanh

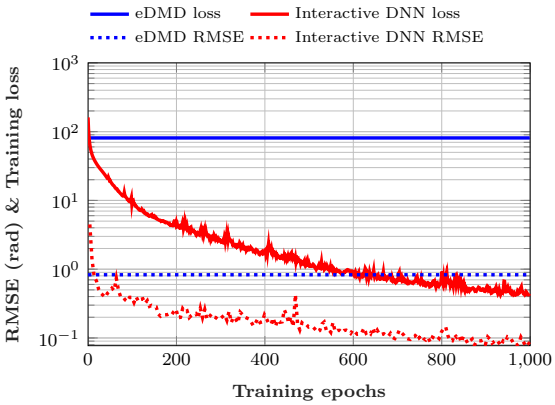


Fig. 4. This figure demonstrates the training loss and RMSE during the training process. The solid lines indicate the training loss with different models over the training set. The dotted lines represent the RMSE over the *Type B* test set, in which trajectories always start and end at the origin. Since the eDMD model is only derived once at the beginning, the loss and RMSE are fixed over the training process.

activation function $f_a(\boldsymbol{x}) = \sinh(\boldsymbol{x})/\cosh(\boldsymbol{x})$. The weight and bias matrices are updated through backpropagation using *AMSGrad* optimizer with a learning rate $lr = 1e-3$. The model is trained for 3 hours on a 24GB NVIDIA GeForces 3090Ti GPU with Pytorch framework. To show the benefits of using the interactive DNN in Algorithm 1, another normal DNN is generated with an additional linear layer without bias patched behind, such that the function basis and the propagation matrix are updated together through backpropagation. The RMSE over the test data set using the DNN is calculated at every epoch to track the performance of the model during the training process.

From the training results shown in Fig. 4, we can easily tell that with the interactive DNN, the training loss in the logarithmic scale decreases exponentially as the training proceeds, and it has a positive correlation with the continuous prediction RMSE over the test set, which indicates a proper setting of the neural network and the target function. By comparing the RMSE between the interactive DNN and the eDMD, we can conclude that the precision of the learned DNN model surpasses the eDMD shortly after the training starts, and it is again one order of magnitude higher than the eDMD when the training is completed. The training loss of the normal DNN settings indicated by the pink line in Fig. 5 has a much higher converged loss than the interactive approach, which means the separate update of $\tilde{A}_{dnn}$ through (12) provides a more suitable gradient descent direction and step length. Fig. 6 demonstrates the detailed performance using one

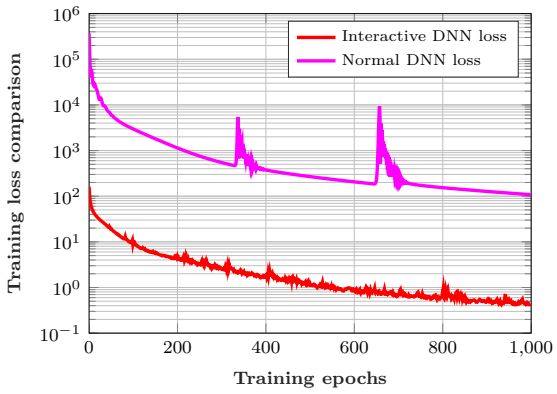


Fig. 5. Comparison of the training process with the normal DNN, which trains the propagation matrices together with the function basis.

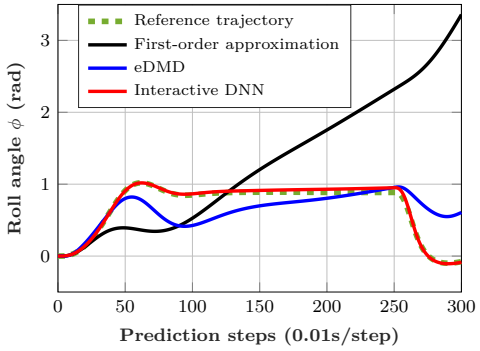


Fig. 6. Model precision comparison using one randomly selected segment from *Type B* test set. The green line denotes the reference trajectory controlled by NMPC, which is the ground truth output of the original nonlinear system. For each model, the trajectory is predicted by feeding in the initial condition and the input sequences from the reference trajectory.

trajectory segment randomly selected from the *Type B* test set. The roll angle $\phi$ is controlled to 0.9 rad ($\approx 51.5°$), which deviates far from the equilibrium. The predicted trajectory using the interactive DNN almost overlaps with the reference over the entire prediction horizon, while the prediction through the first-order model diverges fast after $\phi$ reaches 0.5 rad ($\approx 28.6°$).

### *4.3 Tracking Performance with Optimal Control*

To check the performance in the control application, we first derive the optimal controller $K_{linear}$, $K_{edmd}$, $K_{dnn}$ for each model through LQR, which solves a discrete algebraic Riccati equation with the corresponding linear propagation matrices and weight matrices $Q_{lqr}$, $R_{lqr}$. Each model has the same weights for the original states and zeros weights across additional augmented states if they exist. Saturation is added for both inputs and states following the limits shown in Table 1. The calculated optimal input $\boldsymbol{u}_k^*$ is then fed into the nonlinear system described in (7) and (8). Here, we select the step signal as the reference for the Euler angles. The tracking performance is shown in Fig. 7, where the step reference of Euler angles changes every 300 steps from 0.6 rad to 1 rad and 1.35 rad. Within the first 300 steps where the reference angle is 0.6 rad, all three models show an excellent tracking performance, while the first-order model has an obvious faster settling time in $\phi$ and $\theta$. When the reference angle changes to 1 rad, the system with the first-order model shows a high overshoot and starts to oscillate due to the deviation from the equilibrium. The increase in deviation does not impact the performance of the eDMD and the DNN systems. As the step reference increases to 1.35 rad ($\approx 77°$) which is close to the critical angle, the error of small angle approximation is too large such that the first-order model cannot be stabilized. The eDMD model exhibits an oscillation with a very small amplitude and high frequency. There are mainly two reasons that incur this problem: first of all, the selected basis functions, together with the training data, are insufficient to approximate the Koopman operator within this operating point. Secondly, the weight for the input in LQR is too small compared to the state. Therefore, the input $\boldsymbol{u}_{edmd}^*$ is bouncing back and forth between $\boldsymbol{u}_{max}$ and $\boldsymbol{u}_{min}$ with LQR saturation. However, the DNN model still maintains an outstanding performance, demonstrating that the neural network learned a more appropriate function basis than eDMD, which matches the model precision results. It also shows the effectiveness of using DNN to approximate the Koopman operator for high dimensional nonlinear systems and the applicability to control applications.

## 5. CONCLUSIONS AND FUTURE WORK

In short, this paper explores the performance of using the data-driven approximation of the Koopman operator on a controlled six-dimensional quadrotor attitude system in $SO(3)$. The basis functions are trained through the DNN with a collection of controlled trajectories using NMPC, while the propagation matrices in the lifted space are updated interactively at each training epoch. This interactive update approach also manifests a lower converged loss than training the propagation matrices altogether with basis functions. The model learned through DNN is then compared with the model derived through the eDMD and the first-order approximation. The results indicate that both data-driven models have significantly higher precision and better control performance than the first-order approximation, especially away from the equilibrium point. The results of the DNN always presents a robust control effect even close to the critical angle where the eDMD starts to oscillate with a high frequency, which also demonstrates the effectiveness of using the neural network to learn more appropriate function basis in high dimensional systems.

Future work first considers extending the system to $SE(3)$, which will add additional three dimensions. Since this will pose significant challenges to the data-driven approach due to insufficient training data, it would be better to combine differential flatness with the Koopman operator, such that the learned Koopman operator only applies to a low-level attitude controller.

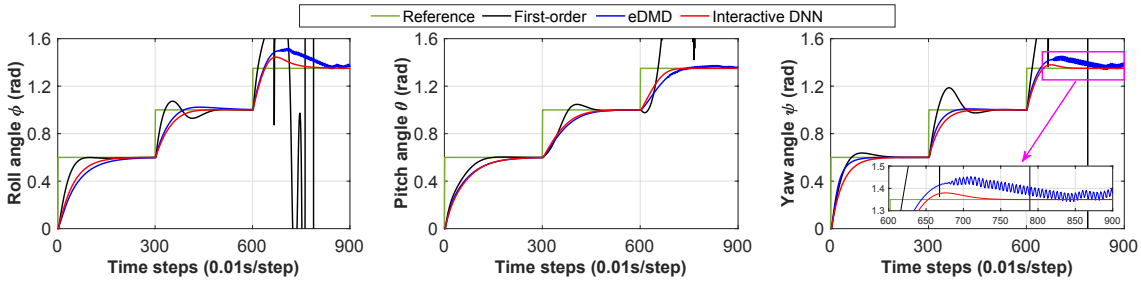


Fig. 7. Systems tracking performance with LQR using step reference of Euler angles.

## REFERENCES


Brunton, S.L., Brunton, B.W., Proctor, J.L., and Kutz, J.N. (2016). Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2), e0150171.

Chen, T., Shan, J., and Wen, H. (2023). Koopman-operator-based attitude dynamics and control on SO(3). In *Distributed Attitude Consensus of Multiple Flexible Spacecraft*, 177–210. Springer.

Faessler, M., Franchi, A., and Scaramuzza, D. (2017). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2), 620–626.

Fettweis, G.P. (2014). The tactile internet: Applications and challenges. *IEEE Vehicular Technology Magazine*, 9(1), 64–70.

Hafez, A.T., Marasco, A.J., Givigi, S.N., Iskandarani, M., Yousefi, S., and Rabbath, C.A. (2015). Solving multi-UAV dynamic encirclement via model predictive control. *IEEE Transactions on control systems technology*, 23(6), 2251–2265.

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

Igarashi, Y., Yamakita, M., Ng, J., and Asada, H.H. (2020). MPC performances for nonlinear systems using several linearization models. In *2020 American Control Conference (ACC)*, 2426–2431.

Koopman, B.O. (1931). Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5), 315–318.

Korda, M. and Mezić, I. (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93, 149–160.

Lusch, B., Kutz, J.N., and Brunton, S.L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1), 1–10.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, 2520–2525.

Morrell, B., Rigter, M., Merewether, G., Reid, R., Thakker, R., Tzanetos, T., Rajur, V., and Chamitoff, G. (2018). Differential flatness transformations for aggressive quadrotor flight. In *2018 IEEE international conference on robotics and automation (ICRA)*, 5204–5210.

Romero, A., Sun, S., Foehn, P., and Scaramuzza, D. (2022). Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*.

Sabatino, F. (2015). Quadrotor control: modeling, nonlinearcontrol design, and simulation.

Schmitt, E.J., González, A., and Fettweis, G.P. (2022). An event-based formation control architecture for UAVs using an estimator-based approach. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, 271–278.

Soria, E., Schiano, F., and Floreano, D. (2021a). Distributed predictive drone swarms in cluttered environments. *IEEE Robotics and Automation Letters*, 7(1), 73–80.

Soria, E., Schiano, F., and Floreano, D. (2021b). Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3(6), 545–554.

Sun, S., Romero, A., Foehn, P., Kaufmann, E., and Scaramuzza, D. (2022). A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight. *IEEE Transactions on Robotics*.

Wang, R., Han, Y., and Vaidya, U. (2021). Deep koopman data-driven control framework for autonomous racing. In *Proc. Int. Conf. Robot. Autom.(ICRA) Workshop Opportunities Challenges Auton. Racing*, 1–6.

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

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2020). Forces NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1), 13–29.

Zinage, V. and Bakolas, E. (2021). Koopman operator based modeling for quadrotor control on SE(3). *IEEE Control Systems Letters*, 6, 752–757.