IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Neural Predictor for Flight Control With Payload

Ao Jin, Chenhao Li, Qinyi Wang, Ya Liu, Panfeng Huang, *Senior Member, IEEE*, and Fan Zhang

***Abstract*—Aerial robotics for transporting suspended payloads as the form of freely-floating manipulator are growinggreat interest in recent years. However, the force/torque caused by payload and residual dynamics will introduce unmodeled perturbations to the aerial robotics, which negatively affects the closed-loop performance. Different from estimation-like methods, this letter proposes Neural Predictor, a learning-based approach to model force/torque induced by payload and residual dynamics as a dynamical system. It yields a hybrid model that combines the first-principles dynamics with the learned dynamics. The hybrid model is then integrated into a MPC framework to improve closed-loop performance. Effectiveness of proposed framework is verified extensively in both numerical simulations and real-world flight experiments. The results indicate that our approach can capture force/torque caused by suspended payload and residual dynamics accurately, respond quickly to the changes of them and improve the closed-loop performance significantly. In particular, Neural Predictor outperforms a state-of-the-art learning-based estimator and has reduced the force and torque estimation errors by up to 66.15% and 33.33% while requiring less samples.**

***Index Terms*—Learning Based Control, Model Learning for Control, Model Predictive Control.**

I. Introduction

OWING to the advantages in terms of strong flexibility, safe manipulation, low cost and transportability, tethered-UAV systems have been widely leveraged in various aerial transportation tasks. By attaching the payload to UAV through flexible tethers, there is no need to consider the shape and size of payload, making it highly adaptable to different transportation scenarios.

While attaching payloads to UAVs using flexible tethers offers significant advantages over rigid attachment methods such as robotic arms or clamps, several challenges remain to be addressed for real-world high-precision transportation tasks. The primary challenge in tethered-UAV systems lies in accurately estimating or predicting the external force/torque vector induced by the tether and payload. Recently, machine learning methods have demonstrated remarkable success in estimating external uncertainties and designing controllers across diverse applications. Additionally, the data-driven Koopman operator (DDKO) theory, which enables the discovery of unknown system dynamics from data in an analytical framework, has garnered great attentions in control community.

In this work, we introduce a learning-based framework called **Neural Predictor** to model the force/torque induced by the payload and residual dynamics of a tethered-UAV system. Specifically, leveraging insights from DDKO theory, we derive the formulation of lifted linear system (LLS), which is subsequently learned from data to capture the external force/torque. By integrating the learned dynamics with the nominal system dynamics, the hybrid model of the tethered-UAV system is constructed. This hybrid model is then embedding within a model predictive control framework, referred to as **NP-MPC**. An overview of the proposed framework is illustrated in Fig. 1. Through extensive simulations and real-world flight experiments, we demonstrate that our approach not only outperforms state-of-the-art learning-based estimators in predicting external forces and torques but also achieves significant improvements in closed-loop control performance.

II. Related Work

The modeling of tethered-UAV system has been studied a lot in the literature. The studies [1], [2], [3], [4] focused on modeling the UAV, tether and payload as a whole system. There is a transition of established dynamics when tether tension becomes zero, which posed great challenges for controller design. In addition, the tether is assumed to be massless and straight in [1]. However, some of them modeled tethered-UAV system upon assumptions that hardly satisfy in practice, some of them require the prior knowledge of payload and tether for modeling. The studies [5], [6] estimated external force caused by unknown payload by adaptive law within the framework of backstepping, but the external torque did not consider.

Recently, machine learning schemes have emerged as a promising approach for modeling complex aerodynamics and external force/torque of quadrotor. NeuroBEM [7] utilized deep neural networks and BEM theory to model aerodynamic effect during high speeds or agile maneuvers. Neural-Fly [8] adopts a deep neural network (DNN) to predict the effect of strong wind. Gaussian processes (GP) regression [9] is also used to model residual aerodynamics but it suffers from computational complexity problem. The studies [10], [11], [12] use neural ordinary differential equations (NODE) to model uncertainties and integrate them into controller. However, the learned dynamics in these works are black-box and implicit. This leads to

Received 11 February 2025; accepted 10 May 2025. Date of publication 26 May 2025; date of current version 4 June 2025. This article was recommended for publication by Associate Editor X. LI and Editor C. Gosselin upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant 62222313, Grant 62327809, Grant 62303312, and Grant 62173275. *(Ao Jin and Chenhao Li are contributed equally to this work.) (Corresponding author: Fan Zhang.)*

The authors are with the Research Center for Intelligent Robotics, Shaanxi Province Innovation Team of Intelligent Robotic Technology, School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: jinao@mail.nwpu.edu.cn; fzhang@nwpu.edu.cn).

The code of proposed Neural Predictor can be found at https://github.com/NPU-RCIR/Neural-Predictor.git.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3573624, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3573624

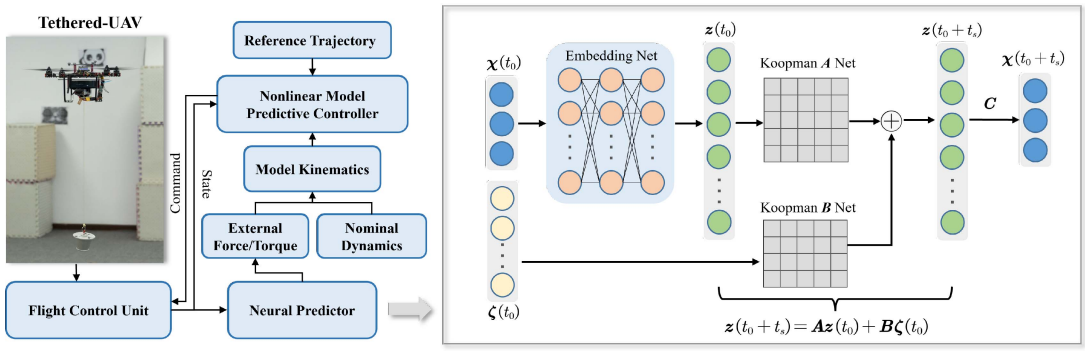

Fig. 1. Illustration of proposed framework: Neural Predictor cooperated with NMPC scheme.

hindrances in analyzing the dynamical characteristics of external uncertainty and residual dynamics. Moreover, the learning-based models proposed in these works do not provide theoretical guarantees of the boundedness of prediction error either. The work [13] proposed NeuroMHE framework to estimate residual aerodynamics in a moving horizon manner. But it needs to solve an optimization problem online and the selection of moving horizon is thorny in practice.

The **main contributions** of our work are summarized as follows. First, inspired by DDKO theory, the external force/torque induced by payload and residual dynamics during flight are explicitly modeled via a linear system, with the model parameters derived through a data-driven learning methodology. It results our proposed **Neural Predictor**, which offers greater interpretability than traditional black-box models. Furthermore, a theoretical guarantee is given for the boundedness of prediction error. Second, a learning-based control framework **NP-MPC** shown in Fig. 1 is proposed for flight control with payload. Our framework eliminates the necessity for modeling of payload and does not rely on any force/torque sensors. The results in both simulations and real-world experiments indicate that our proposed framework not only owns better ability of fast response and accurate estimation of external force/torque against state-the-of-art estimator, but also improves the closed-loop performance significantly.

III. Problem Statement

It is assumed that the quadrotor is a 6 degree-of-freedom (DoF) rigid body with mass m and moment of inertia $\boldsymbol{J} \in \mathbb{R}^{3\times3}$. Then the dynamics of quadrotor of tethered UAV system could be written as

$$\begin{aligned}\dot{\boldsymbol{p}}_w &= \boldsymbol{v}_w,\ \dot{\boldsymbol{v}}_w = m^{-1}\left(-mg\boldsymbol{z} + \boldsymbol{R}f_u\boldsymbol{z} + \boldsymbol{f}_p + \boldsymbol{f}_{\text{res}}\right)\\ \dot{\boldsymbol{R}} &= \boldsymbol{R}\boldsymbol{\omega}_b^{\times},\ \dot{\boldsymbol{\omega}}_b = \boldsymbol{J}^{-1}\left(-\boldsymbol{\omega}_b^{\times}\boldsymbol{J}\boldsymbol{\omega}_b + \boldsymbol{\tau}_m + \boldsymbol{\tau}_p + \boldsymbol{\tau}_{\text{res}}\right)\end{aligned} \quad (1)$$

where $\boldsymbol{f}_p$ and $\boldsymbol{\tau}_p$ are the force and torque subjected to the quadrotor caused by payload, $\boldsymbol{f}_{\text{res}}$ and $\boldsymbol{\tau}_{\text{res}}$ are residual forces and torques, respectively. And $\boldsymbol{p}_w = [x, y, z]^{\mathrm{T}}$ and $\boldsymbol{v}_w = [v_x, v_y, v_z]^T$ are the position and velocity of the quadrotor's center-of-mass in the world frame $\mathcal{I}$, $\boldsymbol{\omega}_b = [\omega_x, \omega_y, \omega_z]^{\mathrm{T}}$ is the angular velocity in the quadrotor body frame $\mathcal{B}$. The $\boldsymbol{R} \in SO^3$ is the rotation matrix from $\mathcal{B}$ to $\mathcal{I}$. The gravitational acceleration is g and $\boldsymbol{z} = [0, 0, 1]^{\mathrm{T}}$. The f_u and $\boldsymbol{\tau}_m = [\tau_{mx}, \tau_{my}, \tau_{mz}]^{\mathrm{T}}$ denote the total thrust and torque produced by the quadrotor's four motors. We define $\boldsymbol{x} = [\boldsymbol{p}_w^{\mathrm{T}}, \boldsymbol{v}_w^{\mathrm{T}}, vec(\boldsymbol{R}), \boldsymbol{\omega}^{\mathrm{T}}]^{\mathrm{T}}$ as the quadrotor state, where $vec(\cdot)$ represents the vectorization of matrix and $\boldsymbol{u} = [f_u, \tau_{mx}, \tau_{my}, \tau_{mz}]^{\mathrm{T}}$ as the control input, respectively. For the sake of completeness, we define the external force/torque as the sum of force/torque caused by payload and residual force/torque, i.e., $\boldsymbol{f}_e = \boldsymbol{f}_p + \boldsymbol{f}_{\text{res}}$ and $\boldsymbol{\tau}_e = \boldsymbol{\tau}_p + \boldsymbol{\tau}_{\text{res}}$.

Problem Formulation: Considering the tethered-UAV system dynamics (1), the objective of this work is to learn the external force $\boldsymbol{f}_e$ and $\boldsymbol{\tau}_e$ induced by suspended payload and residual dynamics from data such that the closed-loop performance can be improved.

IV. Learning External Force and Torque Using Lifted Linear System

This section elaborates the details of proposed framework. The main idea is modeling the external force $\boldsymbol{f}_e$ and torque $\boldsymbol{\tau}_e$ as a dynamical system and learning them from data. Specifically, the proposed framework utilizes lifted linear system derived from Koopman operator theory to capture the external force/torque in an explicit way.

A. Lifted Linear System

The Koopman operator embeds nonlinear dynamical systems into equivalent infinite-dimensional linear representations from data [14]. This provides a data-driven pattern to capture unknown nonlinear dynamics such as external force/torque of (1) in an explicit way. Thus, our work adopts this property to learn $\boldsymbol{f}_e$ and $\boldsymbol{\tau}_e$ in a data-driven manner. The basic idea of lifted linear system is introduced briefly as follows.

Consider an unknown nonlinear dynamics with control input

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \quad (2)$$

where $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^p$. We define a set of scalar valued functions $\boldsymbol{g}$, which are called embedding (lifting) functions. The set of embedding functions defines an infinite dimensional Hilbert space $\mathcal{H}$. We assume that the embedding functions take the form

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{L}\boldsymbol{u} \quad (3)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

where $\boldsymbol{L}$ is a constant matrix. Under the assumption that control input $\boldsymbol{u}$ does not evolve in the Hilbert space $\mathcal{H}$, the dynamics of (2) can be reformulated as [15], [16]

$$\begin{aligned}\mathcal{K}\boldsymbol{g}(\boldsymbol{x}(t_0),\boldsymbol{u}(t_0)) &= \boldsymbol{g}(\boldsymbol{x}(t_0+t_s),0) = \boldsymbol{\Phi}(\boldsymbol{x}(t_0+t_s)) \\ &= \begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Phi}(\boldsymbol{x}) \\ \boldsymbol{u}\end{bmatrix} \end{aligned} \tag{4}$$

where $\boldsymbol{x}(t_0+t_s) = \boldsymbol{x}(t_0) + \int_{t_0}^{t_0+t_s} \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\mathrm{d}t$ with sampling time t_s. The above equation figures the forward-time evolution of the embedding functions $\boldsymbol{g}(\boldsymbol{x})$. Taking the notion $\boldsymbol{z}(t) \triangleq \boldsymbol{\Phi}(\boldsymbol{x}(t))$, the above equation can be rewritten as

$$\boldsymbol{z}(t_0+t_s) = \boldsymbol{A}\boldsymbol{z}(t_0) + \boldsymbol{B}\boldsymbol{u}(t_0) \tag{5}$$

which is called **lifted linear system (LLS)** of unknown nonlinear dynamics (2). If there is a sequence of data $\{(\boldsymbol{x}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{x}_{T-2},\boldsymbol{u}_{T-2}),(\boldsymbol{x}_{T-1})\}$ of unknown nonlinear dynamics (2), the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be approximated by solving the following least-squares minimization problem

$$\boldsymbol{A},\boldsymbol{B} = \arg\min_{\boldsymbol{A},\boldsymbol{B}} \|\boldsymbol{z}_{1:T-1} - (\boldsymbol{A}\boldsymbol{z}_{0:T-2} + \boldsymbol{B}\boldsymbol{u}_{0:T-2})\| \tag{6}$$

where $\boldsymbol{z}_{0:T-2} = [\boldsymbol{z}_0,\boldsymbol{z}_1,\boldsymbol{z}_i,\ldots,\boldsymbol{z}_{T-2}]^{\mathrm{T}}$, $\boldsymbol{z}_{1:T-1} = [\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_i,\ldots,\boldsymbol{z}_{T-1}]^{\mathrm{T}}$, $\boldsymbol{u}_{0:T-2} = [\boldsymbol{u}_0,\boldsymbol{u}_1,\boldsymbol{u}_i,\ldots,\boldsymbol{u}_{T-2}]^{\mathrm{T}}$, $\boldsymbol{z}_i$ represents the embedding state corresponding to the i-th sample $\boldsymbol{x}_i$ $(i=0,1,\ldots,T-1)$. When the sequence length T of $\boldsymbol{z}_{0:T-2},\boldsymbol{u}_{0:T-2}$ is finite, the above least-squares minimization problem leads to a finite dimensional approximation of Koopman operator [17]. The equation (5) is in discrete form as it is derived in a data-driven manner. Thus, according to (5), the continuous finite-dimensional approximation of unknown nonlinear system (2) could be written as

$$\begin{cases}\dot{\boldsymbol{z}} = \boldsymbol{A}_{\boldsymbol{c}}\boldsymbol{z} + \boldsymbol{B}_{\boldsymbol{c}}\boldsymbol{u} \\ \boldsymbol{x} = \boldsymbol{\Phi}^{-1}(\boldsymbol{\Phi}(\boldsymbol{x}))\end{cases} \tag{7}$$

where $\boldsymbol{A}_{\boldsymbol{c}} = \log(\boldsymbol{A})/t_s$, $\boldsymbol{B}_{\boldsymbol{c}} = \boldsymbol{B}/(\int_0^{t_s} e^{\boldsymbol{A}_{\boldsymbol{c}}t}\mathrm{d}t)$, $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^K$, $K \gg n$, $\boldsymbol{u} \in \mathbb{R}^p$, $\boldsymbol{A}_{\boldsymbol{c}} \in \mathbb{R}^{K\times K}$, $\boldsymbol{B}_{\boldsymbol{c}} \in \mathbb{R}^{K\times p}$, t_s is the sampling interval.

B. A Learning-Based Scheme to Capture External Force and Torque

In this work, the unknown dynamics of external force and torque are assumed to be described by [8]

$$\dot{\boldsymbol{\chi}} = \boldsymbol{\xi}(\boldsymbol{\chi},\boldsymbol{\zeta}) \tag{8}$$

where $\boldsymbol{\chi} = [\boldsymbol{f}_e,\boldsymbol{\tau}_e]^{\mathrm{T}}$, $\boldsymbol{\xi}$ represents the unknown dynamics of the external force and torque, and $\boldsymbol{\zeta}$ denotes the *control input* of (8) and it is usually the state or substate (part of full state) of quadrotor. The selection of $\boldsymbol{\zeta}$ will be discussed in the implementation section later.

As discussed above, the unknown nonlinear dynamics like (8) can be captured by a finite-dimensional LLS (7) from data. While it is not an easy task to find the embedding functions $\boldsymbol{\Phi}$. Therefore, a deep neural network parameterized by weights $\boldsymbol{\theta} = W^1,\ldots,W^{L+1}$ is raised to approximate the finite dimensional embedding functions

$$\boldsymbol{\Phi}(\boldsymbol{\chi};\boldsymbol{\theta}) = W^{L+1}\phi(W^L(\ldots\phi(W^1\boldsymbol{\chi}\ldots)) \tag{9}$$

Algorithm 1: Offline Learning LLS.

Input A dataset $\{\boldsymbol{x}^i_{0:m-1},\boldsymbol{u}^i_{0:m-1}\}$ $(i=1,\ldots,s)$.
1: Calculate labels $\boldsymbol{\chi}^i_{0:m-1}$ for training.
2: **while** L not decrease to tolerance **do**
3: Calculate embedding states $\boldsymbol{z}^i_{0:m-1}$, $\boldsymbol{z}^i_{0:m-2}$, $\boldsymbol{z}^i_{1:m-1}$ with $\boldsymbol{\Phi}(\boldsymbol{\chi};\boldsymbol{\theta})$.
4: Estimate matrices $\boldsymbol{A}^i$, $\boldsymbol{B}^i$ by solving $\min_{\boldsymbol{A}^i,\boldsymbol{B}^i} \|\boldsymbol{z}^i_{1:m-1} - (\boldsymbol{A}^i\boldsymbol{z}^i_{0:m-2} + \boldsymbol{B}^i\boldsymbol{\zeta}^i_{0:m-2})\|$.
5: Predict embedding states by iterating LLS forward and backward in time, ${}^{\mathrm{f}}\hat{\boldsymbol{z}}^i_{0:m-1}$, ${}^{\mathrm{b}}\hat{\boldsymbol{z}}^i_{0:m-1}$.
6: Update parameters of $\boldsymbol{\Phi}(\boldsymbol{\chi};\boldsymbol{\theta})$ and matrix $\boldsymbol{C}$ by minimizing loss function L (10).
7: **end while**
Output Embedding functions $\boldsymbol{\Phi}(\boldsymbol{\chi};\boldsymbol{\theta})$ and matrix $\boldsymbol{C}$.

where ϕ is the ReLU activation function. Sample data from tethered-UAV system flying with a baseline controller to construct a dataset $\{\boldsymbol{x}^i_{0:m-1},\boldsymbol{u}^i_{0:m-1}\}$ $(i=1,\ldots,s)$ which contains s trajectories. Every trajectory in the dataset has m steps data, $\boldsymbol{x}^i_{0:m-1} = [\boldsymbol{x}^i_0,\ldots,\boldsymbol{x}^i_{m-1}]^{\mathrm{T}}$, $\boldsymbol{u}^i_{0:m-1} = [\boldsymbol{u}^i_0,\ldots,\boldsymbol{u}^i_{m-1}]^{\mathrm{T}}$. The labels for offline training $\boldsymbol{\chi}^i_{0:m-1} = [\boldsymbol{\chi}^i_0,\ldots,\boldsymbol{\chi}^i_{m-1}]^{\mathrm{T}}$ are calculated by using (1).

To find the embedding functions associated with Koopman operator, we minimize the following loss function

$$L = \beta_1 L_{\text{forward}} + \beta_2 L_{\text{backward}} + \beta_3 L_{\text{recons}} \tag{10}$$

where β_1, β_2 and β_3 are positive hyperparameters. The detailed implementation of learning LLS is depicted in Algorithm 1. Three loss functions L_{forward}, L_{backward}, and L_{recons} are developed as follows.

i) Multi-Steps Prediction Loss L_{forward} and L_{backward}: The embedding state at each sampling instant can be represented as $\boldsymbol{z}^i_{0:m-1} = [\boldsymbol{z}^i_0,\ldots,\boldsymbol{z}^i_{m-1}]^{\mathrm{T}} = [\boldsymbol{\Phi}(\boldsymbol{\chi}\mathbf{o}^i),\ldots,\boldsymbol{\Phi}(\boldsymbol{\chi}^i_{m-1})]^{\mathrm{T}}$. The state predicted by iterating LLS (7) with the initial state $\boldsymbol{z}^i_0$ forward in time and backward in time [1] are denoted as ${}^{\mathrm{f}}\hat{\boldsymbol{z}}^i_{0:m-1} = [{}^{\mathrm{f}}\hat{\boldsymbol{z}}^i_0,\ldots,{}^{\mathrm{f}}\hat{\boldsymbol{z}}^i_{m-1}]^{\mathrm{T}}$ and ${}^{\mathrm{b}}\hat{\boldsymbol{z}}^i_{0:m-1} = [{}^{\mathrm{b}}\hat{\boldsymbol{z}}^i_0,\ldots,{}^{\mathrm{b}}\hat{\boldsymbol{z}}^i_{m-1}]^{\mathrm{T}}$, respectively. The loss function of the forward and backward prediction error is raised as

$$\begin{aligned}L_{\text{forward}} &= \sum_{i=1}^{s} \mu_1^i \mathrm{MSE}(\boldsymbol{z}^i_{0:m-1}, {}^{\mathrm{f}}\hat{\boldsymbol{z}}^i_{0:m-1}) \\ L_{\text{backward}} &= \sum_{i=1}^{s} \mu_2^i \mathrm{MSE}(\boldsymbol{z}^i_{0:m-1}, {}^{\mathrm{b}}\hat{\boldsymbol{z}}^i_{0:m-1})\end{aligned} \tag{11}$$

where the $\mu_1,\mu_2 \in (0,1)$ are hyperparameters. Loss function (11) focuses on minimizing the multi-steps prediction error forward and backward in time, which is beneficial to predict state more precisely over a longer horizon.

ii) Reconstruction Loss L_{recons}: The state of nonlinear system should be constructed from the invariant Hilbert space $\mathcal{H}$. The second equation in (7) needs to obtain the inverse expression of embedding functions (9), which is relatively challenging.

[1] Backward dynamics: $\boldsymbol{z}_{\boldsymbol{k}} = \boldsymbol{A}^{-1}(\boldsymbol{z}_{k+1} - \boldsymbol{B}\boldsymbol{\zeta}_k)$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Therefore, this work utilizes $\boldsymbol{\chi} = \boldsymbol{C}\boldsymbol{z}$ for the reconstruction. The loss functions L_{recons} is given by

$$L_{\text{recons}} = \sum_{i=1}^{s} \|\boldsymbol{\chi}^i_{0:m-1} - \boldsymbol{C}\boldsymbol{z}^i_{0:m-1}\| \tag{12}$$

where matrix $\boldsymbol{C}$ is parameterized by a linear layer without bias.

C. Theoretical Guarantee for the Learned Dynamics

As the learned dynamics may have a divergent prediction, which will engender an inaccurate prediction of external force/torque. Thus, a constraint for bounding the prediction error of the learned dynamics must be considered.

Theorem 1: Assuming that $\alpha_{\boldsymbol{\chi}} > 0$ and $\alpha_{\boldsymbol{\zeta}} > 0$ such that $\|\boldsymbol{\chi}(t_0 + t_s) - \boldsymbol{\chi}(t_0)\| \leq \alpha_{\boldsymbol{\chi}}$ and $\|\boldsymbol{\zeta}(t_0 + t_s) - \boldsymbol{\zeta}(t_0)\| \leq \alpha_{\boldsymbol{\zeta}}$ are satisfied, the approximated Koopman operator $\hat{\boldsymbol{\mathcal{K}}}$ is stable and the embedding functions associated it have a Lipschitz constant $L_{\boldsymbol{\Phi}}$, then the prediction error is globally bounded.

Proof: The global prediction error [14] induced by the approximated Koopman operator $\hat{\boldsymbol{\mathcal{K}}}$ after n sampling instants is given by

$$\boldsymbol{E_n} = \boldsymbol{z}(t_0 + nt_s) - \hat{\boldsymbol{\mathcal{K}}}^n \boldsymbol{z}(t_0) \tag{13}$$

By recursively iterating (5), the global prediction error can be represented as

$$\begin{aligned} \|\boldsymbol{E_n}\| &= \|\boldsymbol{\Phi}(t_0 + nt_s) - \hat{\boldsymbol{\mathcal{K}}}^n \boldsymbol{\Phi}(t_0)\| \\ &= \|\sum_{i=0}^{n-1} \hat{\boldsymbol{\mathcal{K}}}^i \boldsymbol{e}(t_0 + (n-i)t_s)\| \\ &\leq \sum_{i=0}^{n-1} \|\hat{\boldsymbol{\mathcal{K}}}^i\| \cdot \|\boldsymbol{e}(t_0 + (n-i)t_s)\| \end{aligned} \tag{14}$$

where the $\boldsymbol{e}(t) = \hat{\boldsymbol{\chi}}(t) - \boldsymbol{\chi}(t)$ is the local prediction error and $\hat{\boldsymbol{\chi}}$ denotes the prediction of external force and torque.

Since the $\boldsymbol{\chi}, \boldsymbol{\zeta}$ and the embedding functions $\boldsymbol{\Phi}$ are Lipschitz continuous, the local prediction error $\boldsymbol{e}(t)$ is bounded by (Theorem 1 in [17])

$$\begin{aligned} \lim_{n_h \to \infty} \sup \|\boldsymbol{e}(t)\| &= (\|\boldsymbol{C}\boldsymbol{A}\| L_{\boldsymbol{\Phi}} + 1)\alpha_{\boldsymbol{\chi}} + \|\boldsymbol{C}\boldsymbol{B}\|\alpha_{\boldsymbol{\zeta}} \\ &\quad + \max_{\overline{\boldsymbol{\chi}} \in \mathbb{B}} \|\overline{\boldsymbol{\chi}} - \boldsymbol{C}\boldsymbol{\Phi}(\overline{\boldsymbol{\chi}})\| \triangleq c \end{aligned} \tag{15}$$

where n_h is number of the last hidden layer of the embedding functions and $\mathbb{B}$ denotes a set that contains $\boldsymbol{\chi}$ at every sampling instant. As the prediction local error $\boldsymbol{e}(t)$ is bounded and the approximated Koopman operator $\hat{\boldsymbol{\mathcal{K}}}$ is stable, then

$$\|\boldsymbol{E_n}\| \leq \sum_{i=0}^{n-1} \|\hat{\boldsymbol{\mathcal{K}}}^i\| \cdot \|\boldsymbol{e}(t_0 + (n-i)t_s)\| \leq \|c\| \sum_{i=0}^{n-1} \|\boldsymbol{\mathcal{K}}^i\| \tag{16}$$

is bounded. Therefore the prediction error is bounded. ■

D. Constrain Lipschitz Constant of Embedding Functions

As stated in Theorem 1, the prediction error is bounded if embedding functions $\boldsymbol{\Phi}$ have a Lipschitz constant. To achieve this, the spectral normalization (SN) is utilized to constrain the Lipschitz constant of embedding functions $\boldsymbol{\Phi}$ in this work. By definition, the Lipschitz constant of a function ρ is equal to the maximum spectral norm of its gradient i.e. $\|\rho\|_{\text{Lip}} = \sup \sigma(\nabla \rho)$. Therefore, for embedding functions $\boldsymbol{\Phi}$ that parameterized by the deep neural network (9), the Lipschitz constant can be bounded by constraining the spectral norm of each layer.

The Lipschitz constant of a linear layer $g(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$ is given by $\|g\|_{\text{Lip}} = \sup \sigma(\nabla(g)) = \sup \sigma(\boldsymbol{W}) = \sigma(\boldsymbol{W})$. Thus, the Lipschitz constant of the embedding functions (9) can be computed as [2]

$$\begin{aligned} \|\boldsymbol{\Phi}(\boldsymbol{\chi})\|_{\text{Lip}} &\leq \|g^{L+1}\|_{\text{Lip}} \cdot \|\phi_L\|_{\text{Lip}} \cdots \|\phi_1\|_{\text{Lip}} \cdot \|g_1\|_{\text{Lip}} \\ &= \prod_{l=1}^{L+1} \sigma(\boldsymbol{W}^l) \end{aligned} \tag{17}$$

where the Lipschitz constant of ReLU activation function is equal to 1, i.e., $\|\phi_l\|_{\text{Lip}} = 1$. The spectral normalization applied to weights of each layer during training is given by

$$\hat{\boldsymbol{W}} = \boldsymbol{W}/\sigma(\boldsymbol{W}) \cdot \gamma^{\frac{1}{L+1}} \tag{18}$$

Lemma 1: With the realization of spectral normalization (18) to the embedding functions, the Lipschitz constant of the embedding functions will satisfy:

$$\|\boldsymbol{\Phi}(\boldsymbol{\chi})\|_{\text{Lip}} \leq \gamma \tag{19}$$

where γ is the expected Lipschitz constant of $\boldsymbol{\Phi}$.

V. Implementation of Controller With Learned Dynamics

As a well-known controller, robust MPC takes uncertainty into consideration [18]. However, these methods rely on some hypotheses about uncertainty, which is hard to satisfy in practice. In addition, the closed-loop performance of MPC and accuracy of model are inextricably linked. The Neural Predictor can capture dynamics that is absent in the nominal dynamics from data. Therefore, the Neural Predictor integrated into MPC framework leads our proposed NP-MPC framework for tethered-UAV system, which is shown in Fig. 1.

The MPC adopts an optimization-based approach to generate a sequence of optimal control inputs in a receding horizon manner. Thus, the optimization problem of NP-MPC considering the learned dynamics can be written as follows

$$\begin{aligned} \text{minimize}_{\bar{\boldsymbol{u}}} \quad & \sum_{i=0}^{N-1} \left(\boldsymbol{e}^{\mathrm{T}}_{\boldsymbol{x}_i} \boldsymbol{Q} \boldsymbol{e}_{\boldsymbol{x}_i} + \boldsymbol{e}^{\mathrm{T}}_{\boldsymbol{u}_i} \boldsymbol{R} \boldsymbol{e}_{\boldsymbol{u}_i}\right) + \boldsymbol{e}^{\mathrm{T}}_{\boldsymbol{x}_N} \boldsymbol{P} \boldsymbol{e}_{\boldsymbol{x}_N} \\ & \bar{\boldsymbol{x}}_0 = \boldsymbol{x}_k, \quad \bar{\boldsymbol{x}}_N \in \mathcal{X}_f, \\ \text{subject to} \quad & \bar{\boldsymbol{x}}_{i+1} = \boldsymbol{f}_{\text{nominal}}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i) + \boldsymbol{\Xi}\hat{\boldsymbol{\chi}}_i, \\ & \hat{\boldsymbol{\chi}}_i = \boldsymbol{C}\boldsymbol{z}_i = \boldsymbol{C}(\boldsymbol{A}^k \boldsymbol{z}_{i-1} + \boldsymbol{B}^k \boldsymbol{\zeta}_{i-1}), \\ & \bar{\boldsymbol{x}}_i \in \mathcal{X}, \quad \bar{\boldsymbol{u}}_i \in \mathcal{U}, \quad \forall i = 0, \ldots, N-1 \end{aligned} \tag{20}$$

where $\boldsymbol{x}_k$ denotes the state of quadrotor at time instant k; $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{u}}_i$ are the predicted state and control input at time

[2] Using the inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
COMPARISON RESULTS OF ESTIMATION ERRORS (RMSES) ON 6 UNSEEN AGILE FLIGHT TRAJECTORIES

Flight Trajectory	Method	F_x[N]	F_y[N]	F_z[N]	τ_x[Nm]	τ_y[Nm]	τ_z[Nm]	F_{xy}[N]	τ_{xy}[Nm]	F[N]	τ[Nm]
Linear oscillation	NeuroBEM	0.164	0.185	0.456	0.013	0.011	0.006	0.247	0.017	0.518	0.018
	NeuroMHE	0.119	0.105	0.186	0.011	0.007	0.005	0.159	0.013	0.244	0.014
	NP	**0.041**	**0.024**	**0.101**	**0.007**	**0.004**	**0.004**	**0.048**	**0.008**	**0.112**	**0.009**
Race track_1	NeuroBEM	0.169	0.158	0.463	0.009	0.009	0.004	0.231	0.013	0.517	0.013
	NeuroMHE	0.141	0.092	0.115	0.007	0.004	0.004	0.168	0.009	0.204	0.009
	NP	**0.025**	**0.018**	**0.048**	**0.005**	**0.003**	**0.002**	**0.031**	**0.006**	**0.057**	**0.006**
3D circle_2	NeuroBEM	0.110	0.129	0.470	0.006	0.009	0.004	0.170	0.011	0.499	0.011
	NeuroMHE	0.140	0.135	0.075	0.003	0.002	0.004	0.194	0.004	0.208	0.006
	NP	**0.055**	**0.021**	0.139	0.007	0.003	0.004	**0.058**	0.007	**0.151**	0.008
Figure-8_3	NeuroBEM	0.145	0.168	0.584	0.010	0.012	0.006	0.221	0.015	0.624	0.017
	NeuroMHE	0.118	0.133	0.151	0.010	0.006	0.005	0.178	0.012	0.233	0.013
	NP	**0.058**	**0.029**	**0.127**	**0.007**	**0.004**	**0.004**	**0.065**	**0.008**	**0.143**	**0.009**
Melon_2	NeuroBEM	0.244	0.198	0.921	0.009	0.006	0.003	0.314	0.015	0.974	0.016
	NeuroMHE	0.254	0.213	0.094	0.005	0.003	0.004	0.331	0.005	0.344	0.007
	NP	**0.069**	**0.025**	0.182	0.009	0.004	0.005	**0.073**	0.010	**0.197**	0.011
Random points	NeuroBEM	0.161	0.183	0.471	0.008	0.008	0.005	0.244	0.012	0.530	0.013
	NeuroMHE	0.115	0.114	0.204	0.010	0.006	0.005	0.162	0.012	0.260	0.012
	NP	**0.032**	**0.028**	**0.077**	**0.007**	**0.004**	**0.003**	**0.042**	**0.008**	**0.088**	**0.008**

instant $k+i$; $\boldsymbol{e}_{\boldsymbol{x}_i} = \bar{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}^*_{k+i}$ and $\boldsymbol{e}_{\boldsymbol{u}_i} = \bar{\boldsymbol{u}}_{\boldsymbol{i}} - \boldsymbol{u}^*_{k+i}$ denote the state and control tracking errors at time time instant $k+i$; $\hat{\boldsymbol{\chi}}_i$ represents predicted external force/torque at time instant $k+i$; $\boldsymbol{\Xi}$ is a matrix mapping the learned external force/torque into state space; $\boldsymbol{z}_{-1} = \boldsymbol{\Phi}(\boldsymbol{\chi}_{k-1})$, $\boldsymbol{\zeta}_{-1} = \boldsymbol{\zeta}_{k-1}$, $\boldsymbol{\zeta}_i$ $(i = 0, \ldots, N-1)$ is constructed with the predicted state $\bar{\boldsymbol{x}}_i$; the reference states and control inputs are generated by a trajectory generator; N is the prediction horizon; $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$ are the state, control input and terminal state constraint sets; $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the state and control input weighting matrices, $\boldsymbol{P}$ is the terminal state weighting matrix; $\boldsymbol{f}_{\text{nominal}}$ is the nominal discrete-time dynamics of quadrotor. At time instant k, utilizing the dataset $\{\boldsymbol{x}^k_{0:T-1}, \boldsymbol{u}^k_{0:T-1}\}$, as formalized in Algorithm 1, the matrices $\boldsymbol{A}^k$ and $\boldsymbol{B}^k$ are estimated by solving the problem $\min_{\boldsymbol{A}^k, \boldsymbol{B}^k} \|\boldsymbol{z}^k_{1:T-1} - (\boldsymbol{A}^k \boldsymbol{z}^k_{0:T-2} + \boldsymbol{B}^k \boldsymbol{\zeta}^k_{0:T-2})\|$, where $\boldsymbol{x}^k_{0:T-1} = [\boldsymbol{x}_{k-T}, \ldots, \boldsymbol{x}_{k-1}]^{\mathrm{T}}$, $\boldsymbol{u}^k_{0:T-1} = [\boldsymbol{u}_{k-T}, \ldots, \boldsymbol{u}_{k-1}]^{\mathrm{T}}$.

VI. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of Neural Predictor both in numerical simulations and real-world flights to demonstrate superior effectiveness of our proposed scheme. The lifting functions $\boldsymbol{\Phi}$ are parameterized by a two-layer deep neural network with 128 neurons of each layer. The activation function is selected as ReLU. The hyperparameters are set to $\beta_1 = \beta_2 = \beta_3 = 1.0$, $\mu_1 = \mu_2 = 0.999$. The dimension K of LLS is set to 24in all numerical simulations and physical experiments. The length of data sequence for matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ estimation during training and online execution is set to 40.

A. Numerical Evaluation

In this part, Neural Predictor is compared with state-of-the-art learning-based estimator NeuroMHE [13] and NeuroBEM [7] for demonstrating superiority of our proposed approach for capturing external force/torque. The dataset for training and validating in this part is the same as flight dataset[3] presented in [7]. This dataset was collected from a variety of agile and high-intensity flight maneuvers. The ground-truth of quadrotor states and time-varying external force/torque can be obtained from the dataset.

The Neural Predictor is trained using supervised learning as the NeuroMHE in numerical experiments (See Section VII-A in [13]) does. The inputs of Neural Predictor are linear and angular velocities of quadrotor. To ensure a fair comparison, the training dataset is selected as a 10-s long trajectory segment from a wobbly circle flight, covering a velocity range of 0.19 m/s to 5.18 m/s, which is consistent with training data used for NeuroMHE in [13]. And for test datasets, we also use the same 13 unseen agile flight trajectories in [13] to compare performance of Neural Predictor against NeuroMHE and NeuroBEM.

The estimation performance of external force/torque on a part of aggressive trajectory (from 45 s to 55 s, labeled as "Random points" in Table I) of NeuroMHE, NeuroBEM and Neural Predictor (NP) is shown in Fig. 2. It is clear that all three methods can estimate the external force/torque accurately. However, during the aggressive maneuver of quadrotor (layout enlargement in Fig. 2), both NeuroMHE and NeuroBEM fail to capture external force/torque well, resulting poor estimation performance. In contrast, the estimation of Neural Predictor remains closer to the ground truth, indicating that our proposed Neural Predictor maintains accurate force/torque estimation even in challenging scenarios. This can be attributed to the fact that Neural Predictor models the external force/torque as a linear dynamical system, with the state of quadrotor states serving as inputs. When quadrotor executes aggressive maneuvers, the changes of states drive Neural Predictor to respond quickly the changes of force/torque, which enables fast response. Furthermore, Neural Predictor, compared to state-of-the-art estimator NeuroMHE, has reduced the force and torque estimation error significantly by 66.15% and 33.33%, respectively.

[3]https://rpg.ifi.uzh.ch/NeuroBEM.html

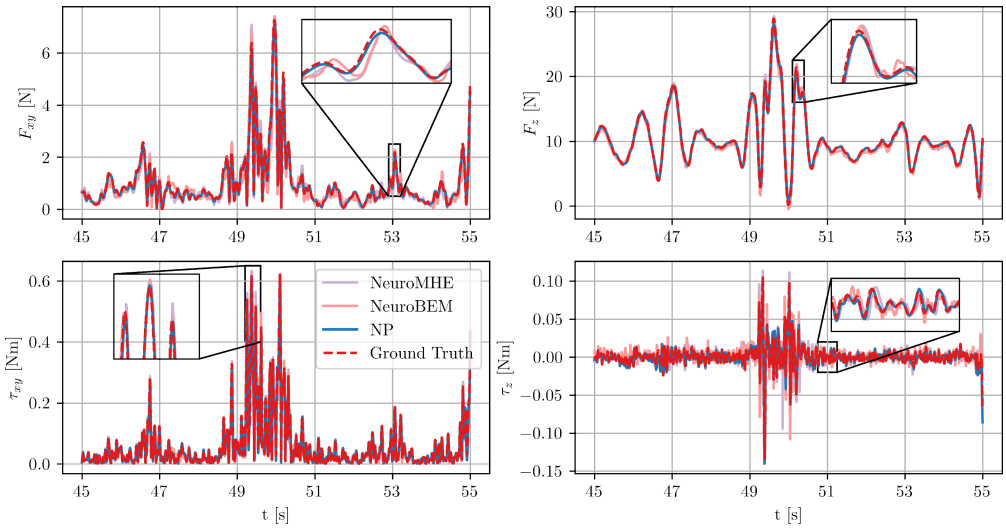

Fig. 2. Force and torque estimation performance of Neural Predictor, NeuroBEM and NeuroMHE on a part of "Random points" test dataset (from 45 s to 55 s). The estimation errors on the whole "Random points" trajectory are $\text{RMSE}_F^{\text{NeuroMHE}} = 0.260$, $\text{RMSE}_\tau^{\text{NeuroMHE}} = 0.012$, $\text{RMSE}_F^{\text{NeuroBEM}} = 0.530$, $\text{RMSE}_\tau^{\text{NeuroBEM}} = 0.013$, $\text{RMSE}_F^{\text{NP}} = \mathbf{0.088}$, $\text{RMSE}_\tau^{\text{NP}} = \mathbf{0.008}$, where $F_{xy} = \sqrt{{F_x}^2 + {F_y}^2}$, $\tau_{xy} = \sqrt{{\tau_x}^2 + {\tau_y}^2}$, $F = \sqrt{{F_x}^2 + {F_y}^2 + {F_z}^2}$ and $\tau = \sqrt{{\tau_x}^2 + {\tau_y}^2 + {\tau_z}^2}$.

To further demonstrate the effectiveness of our approach, we evaluate the performance of Neural Predictor on 6 unseen agile flight trajectories. The detailed results are provided in Table I, where NP denotes our proposed Neural Predictor. The RMSEs of the estimation errors are computed in the same manner as NeuroMHE in [13]. It is clear that Neural Predictor achieves much smaller RMSEs for force/torque estimation and outperforms than NeuroMHE and NeuroBEM across all cases. Notably, our method has reduced the force estimation error by up to 72.06% (for the"Race track_1" trajectory in Table I). Regarding torque estimation, Neural Predictor outperforms NeuroMHE and NeuroBEM in four cases, with comparable performance in the remaining two cases. These results demonstrate that Neural Predictor owns good generalization ability across a variety of unseen aggressive trajectories and delivers superior performance for estimating force/torque compared to state-of-the-art learning-based estimator.

To demonstrate the superior sample efficiency of the proposed method, we trained Neural Predictor with 1 K, 2 K, 3 K, 4 K, 5 K and 6 K samples[4] of training set. The estimation results on test dataset are presented in Fig. 3. As observed, the RMSE of the trained Neural Predictor decreases as the sample size increases. However, the performance of Neural Predictor trained with 4 K, 5 K, and 6 K samples is nearly identical on the test dataset, indicating that 4 K samples are sufficient to effectively capture the dynamics between force/torque and linear/angular velocity. Moreover, compared to NeuroMHE trained with 4 K samples, Neural Predictor exhibits comparable performance even with fewer than 4 K samples. Specifically, for force estimation, Neural Predictor achieves performance similar to that of NeuroMHE trained with 4 K samples using only approximately 1.8 K samples, significantly reducing the required sample size for training.

[4]These samples represent 2.5-s, 5-s, 7.5-s, 10-s, 12.5-s and 15-s long segment of trajectory, respectively.

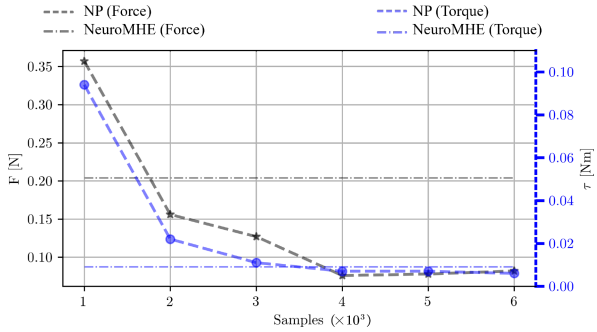

Fig. 3. Force and torque estimation RMSEs of Neural Predictor that is trained with different sample size on "Race track_1" test dataset. The gray and blue dash-dotted lines denote force and torque estimation RMSEs of NeuroMHE which is trained with 4 K samples on "Race track_1" test dataset, respectively.

Overall, the numerical simulation results indicate that: 1) the proposed Neural Predictor owns capability of fast response and accurate estimation of external force/torque even there is an aggressive maneuvers, 2) the trained Neural Predictor demonstrates better generalization performance and sampling efficiency on unseen agile flight scenarios.

B. *Real-World Flight Experiments*

The setup for real-world flight experiments is shown in Fig. 4. The custom quadrotor weighs 2.0 kg, with a motor wheelbase of 360 mm. It is equipped with a Pixhawk4 for attitude estimation and low-level control. The position and linear velocity of quadrotor are estimated from measurements of a motion capture system (VICON). The acceleration and angular velocity are measured from the accelerometers and gyroscope sensors of Pixhawk4. All implementations for solving the optimization problem (20) are deployed with CasADi [19] and acados [20]. The data for training is collected at 50 Hz with a baseline MPC controller

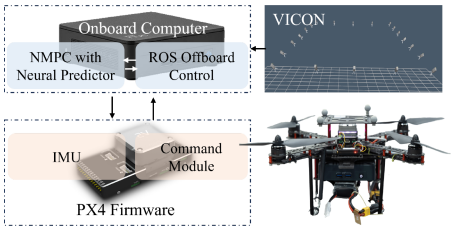

Fig. 4. Schematic of real-world flight experiments setup.

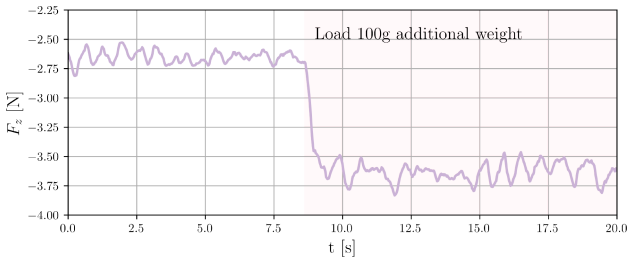

Fig. 5. Estimation of the external force in Z-axis when additional payload is attached.

from a circular flight trajectory[5] of tethered-UAV system. The training set contains 60-s data (3000 data points), and the labels (external force/torque) for learning are calculated using equation (1). In the data collection experiment, the tether length is 0.8 m and the payload mass is 260 g, which are unknown to the pipeline of the proposed framework.

We first demonstrate the ability of Neural Predictor for accurately predicting the external force. During hovering with a payload weighted 260 g at a height of 1.6 m, an additional payload weighted 100 g is attached to the tethered-UAV. The prediction results of external force are shown in Fig. 5. It is seen that the dynamical response of Neural Predictor to changes in external force is almost instant. And when the weight of payload increased by 100 g, the output of Neural Predictor increased by 0.96 N with a standard deviation of 0.004 N. Note that as there is residual dynamics, the changes of output of Neural Predictor does not strictly equal to the force of gravity on additional payload. These results indicate that Neural Predictor achieves high-fidelity real-time estimation of external force. Furthermore, it can be adaptive to different unknown payload.

Subsequently, real-world flight experiments involving trajectory tracking are conducted to verify the superiority of NP-MPC against nominal MPC and other two similar frameworks, GP-MPC [9] and KNODE-MPC [10]. For fair comparison,all models of three learning-based frameworks (NP-MPC, GP-MPC, and KNODE-MPC) are trained on identical 60-s circular trajectories obtained from the aforementioned data collection experiment. The KNODE network architecture maintains an identical configuration to the lifting function, while the GP model employs a radial basis function (RBF) kernel with 100 uniformly sampled data points for training. The inputs of NP, GP and KNODE models are all linear and angular velocity of

[5]The radius and flying speed are 1 m and 1.5 m/s, respectively.

TABLE II
COMPARISON RESULTS OF TRACKING ERRORS (RMSES) ON TWO REAL-WORLD FLIGHT TRAJECTORIES

Trajectory	Method	E_{xy} [m]	E_z [m]
Circle	Nominal MPC	0.1797	0.2329
	GP-MPC	0.1414	0.1692
	KNODE-MPC	0.0919	0.1307
	NP-MPC	**0.0837**	**0.0758**
Lemniscate	Nominal MPC	0.2029	0.2539
	GP-MPC	0.1878	0.1966
	KNODE-MPC	0.1503	0.1406
	NP-MPC	**0.0849**	**0.0591**

quadrotor like we did in the numerical evaluation. The reference signals are produced by a polynomial trajectory generator and the reference height is 1.6 m, while the suspended payload mass is maintained at 260 g throughout the experiments.

The performance results of four frameworks for tracking two trajectories lasting 50-s are summarized in Table II and visualized in Fig. 6. Notably, due to the modeling mismatch induced by payload, nominal MPC has significant tracking errors in X-Y plane and Z-axis. In contrast, all three learning-based frameworks, GP-MPC, KNODE-MPC and NP-MPC, captured the external force induced by payload from data, which results smaller tracking error against nominal MPC. In the circle trajectory tracking experiment, GP-MPC, KNODE-MPC and NP-MPC own better performance than nominal MPC since they learned modeling mismatch from data and compensated it to the MPC framework. However, our proposed NP-MPC outperforms the other two frameworks and reduces the tracking error by 53.45% in X-Y plane, 67.45% in Z-axis compared with nominal MPC. Similar performance trends persist in the lemniscate trajectory tracking experiments, where all learning-based frameworks maintain advantages over nominal MPC. However, GP-MPC and KNODE-MPC exhibit substantial tracking degradation in high-curvature regions (characterized by small radius of curvature). Despite the lemniscate trajectory's geometric dissimilarity to the circular training data, NP-MPC maintains precise tracking even under these geometrically distinct conditions, achieving error reductions of 58.16% (X-Y plane) and 76.72% (Z-axis) compared to nominal MPC. These experimental findings collectively demonstrate that the Neural Predictor not only accurately estimates the external force induced by payload but also exhibits great generalization capabilities to out-of-distribution trajectory patterns unseen during training.

To demonstrate the effective force/torque estimation of NP during payload oscillations, a hovering experiment under oscillatory condition is conducted. The detailed results can be found in [21]. The computational time required for evaluating the neural network and solving the optimization problem (6) on the companion computer[6] of Pixhawk4 is less than 5 ms. For solving the MPC problem (20), the computational time is less than 10 ms. This indicates that our framework is computationally efficient and feasible for practical applications.

[6]The companion computer of Pixhawk4 is equipped with an Intel Core i5-1135G7 and 16 GB RAM

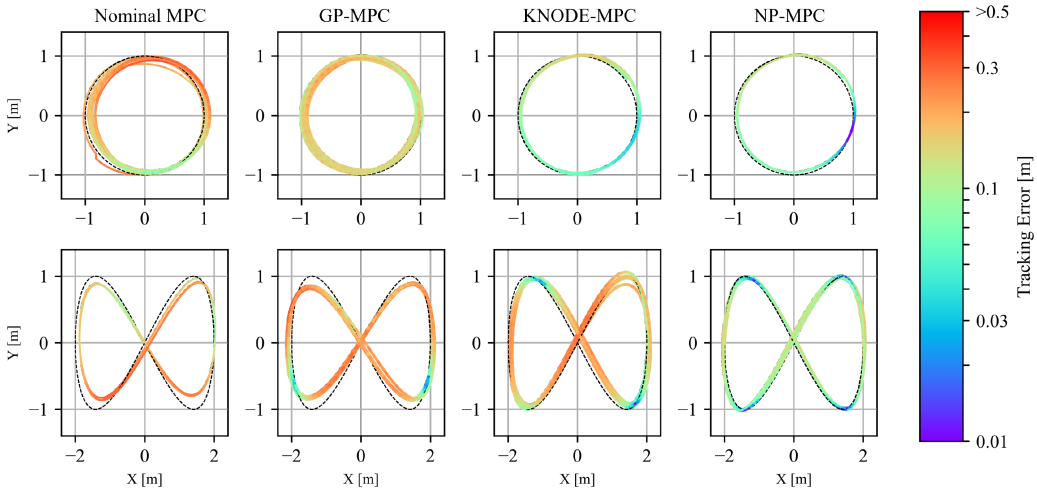

Fig. 6. Plane trajectory tracking performance of nominal MPC, GP-MPC, KNODE-MPC and NP-MPC with flying speed 1.5 m/s. The reference trajectory is shown in dot line. The tracking error refers the RSMEs between real flight trajectory and the reference trajectory.

Overall, the real-world flight results indicate that: 1) the fast response, accurate estimation and great generalization ability of proposed Neural Predictor have been further validated. 2) Neural Predictor cooperating with MPC can significantly improve the tracking performance and outperform other similar frameworks.

VII. Conclusion

In this work, we present the Neural Predictor, a learning-based framework designed to accurately capture the external force/torque induced by suspended payload. Our critical insight is to model external force/torque as a dynamical system by utilizing DDKO theory. A theoretical guarantee is also given to bound prediction error. This makes Neural Predictor owns capability of fast response, good generalization, better sample efficiency and more accurate estimation of external force/torque over state-of-the-art estimator. Furthermore, combing Neural Predictor with MPC framework known as NP-MPC, significantly improves the closed-loop tracking performance and outperforms other similar frameworks in real-world experiments. The payload oscillation mitigation of the proposed method can be achieved through predictive compensation, which is a passive approach. Our future work will focus on extending the proposed method to address challenges in active payload oscillation mitigation and multi-UAVs cooperative transportation under unknown disturbances.

References

[1] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system," in *Proc. 2012 IEEE Int. Conf. Robot. Automat.*, IEEE, 2013, pp. 4888–4895.

[2] P. J. Cruz, M. Oishi, and R. Fierro, "Lift of a cable-suspended load by a quadrotor: A hybrid system approach," in *Proc. 2015 Amer. Control Conf. (ACC)*, Jul. 2015, pp. 1887–1892.

[3] S. Tang, V. Wüest, and V. Kumar, "Aggressive flight with suspended payloads using vision-based control," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1152–1159, Apr. 2018.

[4] H. Li, H. Wang, C. Feng, F. Gao, B. Zhou, and S. Shen, "Autotrans: A complete planning and control framework for autonomous UAV payload transportation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 10, pp. 6859–6866, Oct. 2023.

[5] G. Yu, W. Xie, D. Cabecinhas, R. Cunha, and C. Silvestre, "Adaptive control with unknown mass estimation for a quadrotor-slung-load system," *ISA Trans.*, vol. 133, pp. 412–423, Feb. 2023.

[6] L. Kong, J. Reis, W. He, X. Yu, and C. Silvestre, "On dynamic performance control for a quadrotor-slung-load system with unknown load mass," *Automatica*, vol. 162, Apr. 2024, Art. no. 111516.

[7] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid aerodynamic quadrotor model," in *Proc. Robot. Sci. Syst. Found.*, 2021.

[8] M. O'Connell et al., "Neural-fly enables rapid learning for agile flight in strong winds," *Sci. Robot.*, vol. 7, no. 66, May 2022, Art. no. eabm 6597.

[9] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3769–3776, Apr. 2021.

[10] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2819–2826, Apr. 2022.

[11] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online dynamics learning for predictive control with an application to aerial robots," in *Proc. Conf. Robot Learn.*, PMLR, 2023, pp. 2251–2261.

[12] T. Duong, A. Altawaitan, J. Stanley, and N. Atanasov, "Port-hamiltonian neural ODE networks on lie groups for robot dynamics learning and control," *IEEE Trans. Robot.*, vol. 40, pp. 3695–3715, 2024.

[13] B. Wang, Z. Ma, S. Lai, and L. Zhao, "Neural moving horizon estimation for robust flight control," *IEEE Trans. Robot.*, vol. 40, pp. 639–659, 2024.

[14] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning stable models for prediction and control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 2255–2275, Jun. 2023.

[15] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM J. Appl. Dynamical Syst.*, vol. 17, no. 1, pp. 909–930, Jan. 2018.

[16] J. Morton, F. D. Witherden, and M. J. Kochenderfer, "Deep variational koopman models: Inferring koopman observations for uncertainty-aware dynamics modeling and control," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3173–3179.

[17] W. Hao, B. Huang, W. Pan, D. Wu, and S. Mou, "Deep Koopman learning of nonlinear time-varying systems," *Automatica*, vol. 159, Jan. 2024, Art. no. 111372.

[18] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, WI, USA: Nob Hill Publishing, 2017.

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[20] R. Verschueren et al., "Acados—a modular open-source framework for fast embedded optimal control," *Math. Program. Computation*, vol. 14, no. 1, pp. 147–183, Mar. 2022.

[21] A. Jin, C. Li, Q. Wang, Y. Liu, P. Huang, and F. Zhang, "Neural predictor for flight control with payload," 2024, *arXiv:2410.15946*.