2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

Enhancing Sample Efficiency and Uncertainty Compensation in Learning-based Model Predictive Control for Aerial Robots

Kong Yao Chee, Thales C. Silva, M. Ani Hsieh, George J. Pappas

***Abstract*— The recent increase in data availability and reliability has led to a surge in the development of learning-based model predictive control (MPC) frameworks for robot systems. Despite attaining substantial performance improvements over their non-learning counterparts, many of these frameworks rely on an offline learning procedure to synthesize a dynamics model. This implies that uncertainties encountered by the robot during deployment are not accounted for in the learning process. On the other hand, learning-based MPC methods that learn dynamics models online are computationally expensive and often require a significant amount of data. To alleviate these shortcomings, we propose a novel learning-enhanced MPC framework that incorporates components from $\mathcal{L}_1$ adaptive control into learning-based MPC. This integration enables the accurate compensation of both matched and unmatched uncertainties in a sample-efficient way, enhancing the control performance during deployment. In our proposed framework, we present two variants and apply them to the control of a quadrotor system. Through simulations and physical experiments, we demonstrate that the proposed framework not only allows the synthesis of an accurate dynamics model on-the-fly, but also significantly improves the closed-loop control performance under a wide range of spatio-temporal uncertainties.**

I. INTRODUCTION

Model predictive control (MPC) is a versatile control framework that generates control actions through the consideration of a possibly nonlinear dynamics model, as well as state and control input constraints. Due to its flexibility, MPC has been applied to a variety of robot systems such as ground vehicles [1], quadruped robots [2] and aerial robots [3], [4]. With an increase in data accessibility, there is an upcoming trend of integrating machine learning methods into MPC, in an attempt to improve model accuracy and control performance [5].

One prominent direction in this domain of learning-based MPC is to utilize learning tools for the construction of dynamics models. In [6] and [7], the authors use Gaussian processes (GPs) to model the residual dynamics of an autonomous vehicle and a quadcopter respectively, before applying the learned models within an MPC framework. While there are sample-efficient variants, such as sparse GPs [8], it is often challenging for GPs to handle large amounts of data without any additional data selection strategies.

On the other hand, there are a number of works that use neural networks (NNs) to model robot dynamics for MPC. The authors in [10] use a feedforward NN to model vehicle dynamics to account for friction. In [11], a temporal convolutional NN is used to model the dynamics of a quadcopter. Within the context of model-based reinforcement learning (MBRL), NN ensembles are employed to create uncertainty-aware dynamics models [12], before using them for the control of robots within the MuJoCo [13] environment. In [14], a NN is used to learn the dynamics of a ground vehicle before applying it in a sampling-based MBRL framework. An overarching theme in these works is that while these learned models have been shown to be accurate in representing complex dynamics, they often require relatively large architectures, with a high number of hidden layers, neurons or models within the ensemble. Hence, it is challenging to use these methods within a conventional nonlinear MPC formulation [15], in which a constrained nonlinear optimization problem is solved at every time step during deployment.

One possible solution is to utilize the KNODE-MPC framework proposed in [16], where a neural ordinary differential equation (NODE) model is used to characterize the residual dynamics of a quadrotor system. The NODE model is combined with a first principles model to form a knowledge-based NODE (KNODE) model, which is then employed within an MPC framework. Because of its

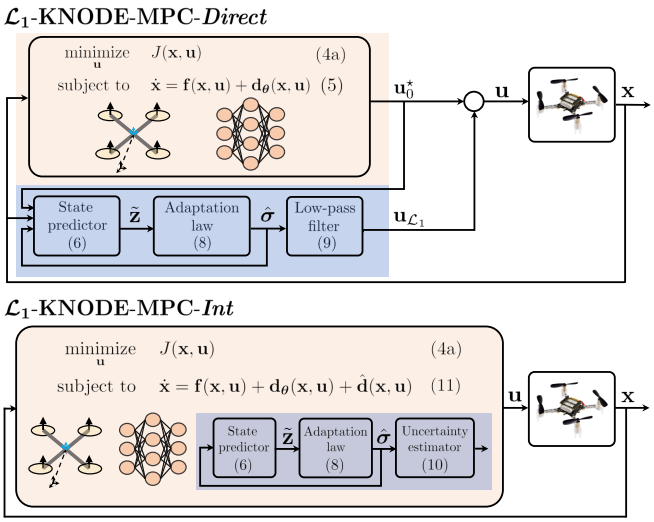


Fig. 1. Schematic of the proposed $\mathcal{L}_1$-KNODE-MPC framework, for the control of a quadrotor system. **Top**: The first variant, $\mathcal{L}_1$-KNODE-MPC-*Direct*, combines the control signals from KNODE-MPC (highlighted in orange) and the $\mathcal{L}_1$ adaptive module (blue) in a direct way. **Bottom**: The second variant, $\mathcal{L}_1$-KNODE-MPC-*Int*, integrates the uncertainties estimated from a modified adaptive module (blue) into the dynamics model within the KNODE-MPC framework (orange). *Image source for quadrotor*: [9].

This work was supported by NSF IIS 1910308, ONR Award N00014-22-1-2157, and DSO National Laboratories, 12 Science Park Drive, Singapore 118225.

The authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. {ckongyao, scthales, m.hsieh, pappasg}@seas.upenn.edu

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341774

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

lightweight architecture and sample efficiency, the KNODE-MPC framework is more flexible and allows for practical extensions [17], [18].

Another drawback of these dynamics learning methods is that the training of the model typically happens offline. As a result, the disturbances experienced by the robot during operation are not accounted for in the training process. Although there are recent studies that attempt to bridge this gap by introducing online and active learning methods, *e.g.*, [17], [19], they tend to be computationally heavy and require a relatively large amount of data for training. This potentially prohibits the robot from adapting to fast-changing disturbances within dynamic environments.

To improve the adaptability and control performance of the quadrotor system against uncertainties, there exist techniques that combine adaptive control with MPC. The authors in [20] combine $\mathcal{L}_1$ adaptive control with MPC to improve the closed-loop performance of a quadrotor system, albeit in a linear setting. In [21], the authors apply $\mathcal{L}_1$ adaptive control on top of a nonlinear MPC scheme. Along the same vein, the authors in [22], [23] and [24] apply a similar $\mathcal{L}_1$ adaptive module to geometric control, MBRL, and to a set of RL algorithms. It is important to note that in these works, the $\mathcal{L}_1$ adaptive module is structured as an additive component where the $\mathcal{L}_1$ control signal is directly added to a signal of an existing controller. Moreover, the $\mathcal{L}_1$ control signal only accounts for matched uncertainties, a subset of all possible uncertainties. Matched uncertainties refer to uncertainties that enter the system through the same channel as the control input. For the quadrotor system, matched uncertainties are the unmodeled forces along the body z axis, in the same direction as the thrust generated by the motors, as well as unmodeled moments about all three axes. On the other hand, unmatched uncertainties are the unmodeled forces along the body x and y axes [22].

Contributions: Our contributions in this work are four-fold. First, we propose a novel learning-enhanced MPC framework, $\mathcal{L}_1$-KNODE-MPC, which compensates for both matched and unmatched uncertainties in a sample-efficient manner during deployment. Second, to the best of the authors' knowledge, this is the first work that combines both Neural ODEs and $\mathcal{L}_1$ adaptive control methods within a nonlinear constrained MPC framework and demonstrates its application on quadrotor control. A schematic of the framework is depicted in Fig. 1. Third, we provide an important insight on how both matched and unmatched uncertainties can be incorporated into the dynamics model within KNODE-MPC. We leverage this insight in the formulation of our second variant, $\mathcal{L}_1$-KNODE-MPC-*Int*. Lastly, through simulations and physical experiments, we demonstrate that $\mathcal{L}_1$-KNODE-MPC not only allows the uncertainties to be estimated accurately, but also enables the quadrotor system to achieve significant performance improvements in terms of trajectory tracking.

II. PRELIMINARIES

We first describe the dynamics of a quadrotor system and a learning-based nonlinear MPC framework, KNODE-MPC.

A. Quadrotor Dynamics

We consider a quadrotor system with the following dynamics [25],

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}, & m\dot{\mathbf{v}} &= -m\mathbf{g} + \mathbf{R}\mathbf{e}_3\,\eta, \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= \boldsymbol{\tau} - \boldsymbol{\omega}\times\mathbf{J}\boldsymbol{\omega}, & \dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\Omega}\mathbf{q}, \end{aligned} \tag{1}$$

where $\mathbf{r}$, $\mathbf{v}$, $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\mathbf{q} \in \mathbb{R}^4$ are the position, velocity, angular rate and quaternions representing the dynamics and kinematics of the quadrotor. The parameters $\eta \in \mathbb{R}$ and $\boldsymbol{\tau} \in \mathbb{R}^3$ are the thrust and moments generated by the motors of the quadrotor. The vector $\mathbf{g}$ denotes gravity in the inertial reference frame and $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the transformation matrix from the body to the inertial reference frame. The inertial frame follows the East-North-Up convention where the z axis is pointing upwards. The unit vector $\mathbf{e}_i \in \mathbb{R}^3$ has value one in the i^{th} element and zero elsewhere. The matrix $\boldsymbol{\Omega} \in \mathbb{R}^{4\times4}$ is defined as

$$\boldsymbol{\Omega} := \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}, \tag{2}$$

where ω_x, ω_y, ω_z are components of the angular rate $\boldsymbol{\omega}$. The mass and inertia matrix of the quadrotor are denoted by m and $\mathbf{J} \in \mathbb{R}^{3\times3}$ respectively. By defining the state and control inputs as $\mathbf{x} := [\mathbf{r}^\top\ \mathbf{v}^\top\ \mathbf{q}^\top\ \boldsymbol{\omega}^\top]^\top$ and $\mathbf{u} := [\eta\ \boldsymbol{\tau}^\top]^\top$, the equations of motion (1) can be written in a compact form,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{3}$$

which we refer to as the nominal dynamics model.

B. KNODE-MPC

For applications that do not require a high-fidelity model, the nominal dynamics model in (3) may be of sufficient accuracy. However, when it is applied within a model-based control framework such as nonlinear MPC, it is challenging to ascertain if the nominal model is accurate enough, especially in the presence of unmodeled dynamics such as actuator dynamics and structural vibrations [26]. On the other hand, when the quadrotor is deployed, we can collect data that provide information about the true system dynamics. The KNODE framework [27] leverages the knowledge of the nominal model, as well as the collected data, to construct an accurate dynamics representation, known as the KNODE model. This model is then applied within an MPC framework, collectively known as KNODE-MPC, to improve the closed-loop tracking performance of the quadrotor system [16], [17]. Within KNODE-MPC, we consider the following constrained optimization problem,

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} ||\mathbf{x}_i - \mathbf{x}_{r,i}||_Q^2 + ||\mathbf{u}_i||_R^2 && \text{(4a)} \\ & + ||\mathbf{x}_N - \mathbf{x}_{r,N}||_P^2 && \text{(4b)} \\ \text{subject to} \quad & \mathbf{x}_{i+1} = \mathbf{f}_{\boldsymbol{\theta},d}(\mathbf{x}_i, \mathbf{u}_i), \quad \forall\, i \in [0,\, N-1] && \text{(4c)} \\ & \mathbf{x}_i \in \mathcal{X}, \quad \mathbf{u}_i \in \mathcal{U}, \quad \forall\, i \in [0,\, N-1] && \text{(4d)} \\ & \mathbf{x}_N \in \mathcal{X}_f, \quad \mathbf{x}_0 = \mathbf{x}(k), && \text{(4e)} \end{aligned}$$

where N is the prediction horizon, $\mathcal{X}$, $\mathcal{X}_f$, $\mathcal{U}$ are sets in which state and control input constraints are defined. The matrices Q, P and R are weighting matrices for the stage,

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

terminal and control input costs respectively. For a vector $\mathbf{s}$ and matrix $\mathbf{A}$, $||\mathbf{s}||^2_{\mathbf{A}}$ denotes $\mathbf{s}^\top \mathbf{A}\mathbf{s}$. The vector $\mathbf{x}(k)$ is the state measurement obtained at each time step $k \in \mathbb{N}$ and $\{\mathbf{x}_{r,0}, \ldots, \mathbf{x}_{r,N}\}$ is a sequence of reference states. At each time step k, we solve (4) and obtain a sequence of optimal control inputs, $\mathbf{u}^\star := \{\mathbf{u}^\star_0, \ldots, \mathbf{u}^\star_{N-1}\}$. The first vector in this sequence $\mathbf{u}^\star_0(\mathbf{x}(k))$ is then applied to the system as the control action.

The model $\mathbf{f}_{\boldsymbol{\theta},d}(\mathbf{x}, \mathbf{u})$ in (4c) is a discrete-time version of the KNODE model, which, with a slight abuse of notation, is given as

$$\dot{\mathbf{x}} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) := \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathbf{d}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}), \tag{5}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is the nominal model in (3) and $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$ is a neural network with parameters $\boldsymbol{\theta}$ representing the learned residual dynamics. Further details on the data collection process and training of the KNODE model can be found in [16]. With a suitable choice of the cost matrix P and constraint set $\mathcal{X}_f$, sufficient conditions for asymptotic stability of the closed-loop system can be attained [18].

While the neural network $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$ is able to characterize the residual dynamics that are not accounted for by the nominal model, it is important to note that training of the KNODE model happens offline. This implies that the KNODE model is only able to account for uncertainties that manifest within the collected data, and not those in subsequent deployments. This motivates us to incorporate the components of $\mathcal{L}_1$ adaptive control to account for uncertainties that are not compensated by the KNODE model.

III. INCORPORATING $\mathcal{L}_1$ ADAPTATION

A. $\mathcal{L}_1$ Adaptive Control

For the control of a quadrotor system, existing state-of-the-art frameworks use $\mathcal{L}_1$ adaptive control as a separate, additive module to a baseline control scheme, *e.g.*, [21], [22]. The $\mathcal{L}_1$ adaptive control structure consists of three submodules; a state predictor, an adaptation law and a low-pass filter. The state predictor estimates the mismatch between a model and the true dynamics in the form of matched and unmatched uncertainties. By defining a partial state vector $\mathbf{z} := [\mathbf{v}^\top \boldsymbol{\omega}^\top]^\top$ and considering the residual dynamics in the KNODE model, $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$, the state predictor is given as

$$\dot{\hat{\mathbf{z}}} = \begin{bmatrix} -\mathbf{g} \\ -\mathbf{J}^{-1}(\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \end{bmatrix} + \mathbf{d}_{\boldsymbol{\theta},\mathbf{z}}(\mathbf{x}, \bar{\mathbf{u}}) + \mathbf{G}_1\bar{\mathbf{u}} + \mathbf{G}\hat{\boldsymbol{\sigma}} + \mathbf{A}(\hat{\mathbf{z}} - \mathbf{z}), \tag{6}$$

where $\bar{\mathbf{u}} := \mathbf{u}_b + \mathbf{u}_{\mathcal{L}_1}$, $\mathbf{u}_b$ and $\mathbf{u}_{\mathcal{L}_1}$ are the control inputs from the baseline control scheme and $\mathcal{L}_1$ control law respectively. The vector $\hat{\boldsymbol{\sigma}} := [\hat{\boldsymbol{\sigma}}_m^\top \ \hat{\boldsymbol{\sigma}}_{um}^\top]^\top \in \mathbb{R}^6$ consists of the estimates of the matched uncertainties $\hat{\boldsymbol{\sigma}}_m \in \mathbb{R}^4$ and the unmatched uncertainties $\hat{\boldsymbol{\sigma}}_{um} \in \mathbb{R}^2$. The state-dependent matrix $\mathbf{G} := [\mathbf{G}_1 \, \mathbf{G}_2] \in \mathbb{R}^{6\times 6}$ describes the mapping between the uncertainties and the predicted state, with the matrices $\mathbf{G}_1 \in \mathbb{R}^{6\times 4}$ and $\mathbf{G}_2 \in \mathbb{R}^{6\times 2}$ defined as

$$\mathbf{G}_1 := \begin{bmatrix} (1/m)\mathbf{R}\mathbf{e}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 1} & \mathbf{J}^{-1} \end{bmatrix}, \ \mathbf{G}_2 := \begin{bmatrix} (1/m)\mathbf{R}\mathbf{e}_1 & (1/m)\mathbf{R}\mathbf{e}_2 \\ \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} \end{bmatrix}. \tag{7}$$

The matrix $\mathbf{A}$ is a pre-specified diagonal Hurwitz matrix. This formulation is similar to the one in [22], except for the vector $\mathbf{d}_{\boldsymbol{\theta},\mathbf{z}}(\mathbf{x}, \bar{\mathbf{u}}) \in \mathbb{R}^6$. This vector is a subset of the residual dynamics in the KNODE model (5) that corresponds to the partial state $\mathbf{z}$. The inclusion of $\mathbf{d}_{\boldsymbol{\theta},\mathbf{z}}(\mathbf{x}, \bar{\mathbf{u}})$ ensures that only uncertainties that have not been considered within the KNODE model are estimated by the adaptation law. Next, the adaptation law is formulated in a piecewise constant manner [22] such that for $t \in [kT, \, (k+1)T]$,

$$\hat{\boldsymbol{\sigma}}(t) := \hat{\boldsymbol{\sigma}}(kT) := \mathbf{G}^{-1}\left(e^{\mathbf{A}T} - \mathbf{I}\right)^{-1} \mathbf{A}e^{\mathbf{A}T}(\mathbf{z}(k) - \hat{\mathbf{z}}(k)), \tag{8}$$

where $k \in \mathbb{N}$, T is the sampling time step and $\mathbf{I}$ is the identity matrix with appropriate dimensions. Finally, the $\mathcal{L}_1$ control law is constructed by accounting for matched uncertainties that are within the bandwidth of the specified low-pass filter [22]. In the Laplace domain, the control signal $\mathbf{u}_{\mathcal{L}_1}$ is written as

$$\mathbf{u}_{\mathcal{L}_1}(s) = -\mathbf{F}(s)\hat{\boldsymbol{\sigma}}_m(s), \tag{9}$$

where $\mathbf{F}(s)$ is the transfer function of the low-pass filter.

We highlight that the control signal in (9) only accounts for matched uncertainties. In Section III-B, we present an important insight and show that it is possible and beneficial to incorporate *both* the matched and unmatched uncertainties into the KNODE-MPC framework.

B. $\mathcal{L}_1$-KNODE-MPC

In our proposed framework, we present two variants. In the first variant, $\mathcal{L}_1$-KNODE-MPC-*Direct*, we consider KNODE-MPC as the baseline control scheme and incorporate $\mathcal{L}_1$ adaptive control *directly* as an add-on module. Specifically, we add the control signal obtained from KNODE-MPC, *i.e.*, $\mathbf{u}^\star_0(\mathbf{x}(k))$, with the discretized adaptive control signal $\mathbf{u}_{\mathcal{L}_1}(k)$ from (9). In this variant, the combination of $\mathcal{L}_1$ adaptive control and KNODE-MPC occurs at the level of the control inputs. This coupling can be seen as an intuitive, albeit weaker integration of the two components, and does not account for unmatched uncertainties. Nonetheless, it shows the feasibility of such a combination. This is depicted in the top panel of Fig. 1. In our second variant, $\mathcal{L}_1$-KNODE-MPC-*Int*, shown in the bottom panel of Fig. 1, we *integrate* the uncertainties estimated from the adaptation law into the KNODE model (5). We observe from (6) and (8) that the product of the matrix $\mathbf{G}$ and $\hat{\boldsymbol{\sigma}}$ can be interpreted as part of the system dynamics, and in particular, part of the residual translational and rotational accelerations. This observation allows us to formulate the piecewise-constant uncertainties as residual accelerations acting on the quadrotor such that for $t \in [kT, \, (k+1)T]$,

$$\begin{bmatrix} \mathbf{f}_{\boldsymbol{\sigma}}(t) \\ \mathbf{m}_{\boldsymbol{\sigma}}(t) \end{bmatrix} := \begin{bmatrix} \mathbf{f}_{\boldsymbol{\sigma}}(kT) \\ \mathbf{m}_{\boldsymbol{\sigma}}(kT) \end{bmatrix} := \mathbf{G}\hat{\boldsymbol{\sigma}}(kT), \tag{10}$$

where $\mathbf{f}_{\boldsymbol{\sigma}}(t)$, $\mathbf{m}_{\boldsymbol{\sigma}}(t) \in \mathbb{R}^3$ are the translational and rotational uncertainties. We then combine these uncertainties with the KNODE model (5) to form the following $\mathcal{L}_1$-KNODE model,

$$\dot{\mathbf{x}} := \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) + \hat{\mathbf{d}}(\mathbf{x}, \mathbf{u}) := \mathbf{f}_{\boldsymbol{\theta},\mathcal{L}_1}(\mathbf{x}, \mathbf{u}), \tag{11}$$

where $\hat{\mathbf{d}}(\mathbf{x}, \mathbf{u}) := [\mathbf{0}_{1\times 3} \ \mathbf{f}_{\boldsymbol{\sigma}}(t)^\top \ \mathbf{0}_{1\times 4} \ \mathbf{m}_{\boldsymbol{\sigma}}(t)^\top]^\top$. With this combination, we augment the data-driven KNODE model to include the estimated uncertainties from the adaptation law. This augmented model is then used as the dynamics model (4c) within KNODE-MPC. We highlight that these uncertainties are estimated using the current state, and are not computed in conjunction with the predicted states in

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: $\mathcal{L}_1$-KNODE-MPC-*Direct*

Input: KNODE model $\mathbf{f}_{\boldsymbol{\theta}}$, state measurements $\mathbf{x}(k)$,
Output: Control action $\bar{\mathbf{u}}(k)$

Initialize: $\bar{\mathbf{u}} \leftarrow \mathbf{0}$, $\hat{\mathbf{z}} \leftarrow \mathbf{z}$
At each time step $k \in \mathbb{N}$*:*
  Extract $\mathbf{d}_{\boldsymbol{\theta},\mathbf{z}}$ from $\mathbf{f}_{\boldsymbol{\theta}}$ in (5)
  Discretize and propagate (6) to get $\hat{\mathbf{z}}(k)$
  Compute uncertainties $\hat{\boldsymbol{\sigma}}(k)$ using (8)
  Compute $\mathbf{u}_{\mathcal{L}_1}(k)$ with (9) or (12)
  Discretize KNODE model $\mathbf{f}_{\boldsymbol{\theta}}$ in (5) and set it as $\mathbf{f}_{\boldsymbol{\theta},d}$ in (4c)
  Apply $\mathbf{x}(k)$ to solve (4) and get $\mathbf{u}_0^\star(\mathbf{x}(k))$
  Compute $\bar{\mathbf{u}}(k) \leftarrow \mathbf{u}_0^\star(\mathbf{x}(k)) + \mathbf{u}_{\mathcal{L}_1}(k)$

Algorithm 2: $\mathcal{L}_1$-KNODE-MPC-*Int*

Input: KNODE model $\mathbf{f}_{\boldsymbol{\theta}}$, state measurements $\mathbf{x}(k)$,
Output: Control action $\mathbf{u}_0^\star(\mathbf{x}(k))$

Initialize: $\mathbf{u}_0^\star \leftarrow \mathbf{0}$, $\hat{\mathbf{z}} \leftarrow \mathbf{z}$
At each time step $k \in \mathbb{N}$*:*
  Extract $\mathbf{d}_{\boldsymbol{\theta},\mathbf{z}}$ from $\mathbf{f}_{\boldsymbol{\theta},\mathcal{L}_1}$ in (11)
  Discretize and propagate (6) to get $\hat{\mathbf{z}}(k)$
  Compute uncertainties $\hat{\boldsymbol{\sigma}}(k)$ using (8)
  Compute $\hat{\mathbf{d}}(\mathbf{x},\mathbf{u})$ with (10) and (11)
  Discretize $\mathcal{L}_1$-KNODE model $\mathbf{f}_{\boldsymbol{\theta},\mathcal{L}_1}$ in (11) and set it as $\mathbf{f}_{\boldsymbol{\theta},d}$ in (4c)
  Apply $\mathbf{x}(k)$ to solve (4) and get $\mathbf{u}_0^\star(\mathbf{x}(k))$

(4). They are integrated into the dynamics constraints (4c) after the estimation procedure, and act as parameters to (4). This allows the MPC scheme to account for these estimated uncertainties during the optimization procedure. We summarize the two proposed variants of $\mathcal{L}_1$-KNODE-MPC in Algorithms 1 and 2.

In $\mathcal{L}_1$-KNODE-MPC-*Int*, there is no segregation between matched and unmatched uncertainties. Both types of uncertainties are considered, which is in stark contrast to existing $\mathcal{L}_1$ adaptive control frameworks. This allows the MPC scheme to generate control inputs that account for both sources of uncertainties. For instance, if the quadrotor experiences an uncertain force in the body x axis, the inclusion of an estimate of this force into the dynamics model, across the prediction horizon, allows the MPC scheme to orientate the quadrotor with a non-zero pitch angle and adjust the motor thrust to counteract this uncertain force, while compensating for gravitational forces. Hence, even though the quadrotor is not able to actuate in the body x axis directly, it is still able to account for disturbances in that direction.

Another advantage of this second variant is that it resolves the inconsistency between the dynamics model used in MPC and that of the $\mathcal{L}_1$ state predictor, as pointed out in [19]. Both dynamics models consider the residual dynamics estimated by the NODE model $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u})$, as well as the dynamics estimated by the adaptive module, $\hat{\mathbf{d}}(\mathbf{x},\mathbf{u})$.

From a broader perspective, the $\mathcal{L}_1$-KNODE-MPC framework utilizes the expressiveness of the deep learning component, *i.e.*, the NODE model, to primarily account for systematic uncertainties, whose effects are likely to be present in the training data. The adaptive module in the framework is leveraged upon to account for time-varying external uncertainties experienced by the robot during deployment in an online fashion.

C. Implementation Details

We use the Optistack class within the CasADi package [28] to implement the optimization problem in (4). A solver based on the interior-point method, IPOPT [29], is used to solve (4). Discretization of the KNODE model (5), $\mathcal{L}_1$-KNODE model (11) is performed using the explicit fourth-order Runge-Kutta (RK) method, while the state predictor in (6) is discretized with the fifth-order RK method [30]. The filter (9) is implemented as

$$\mathbf{u}_{\mathcal{L}_1}(k) = \big(\mathbf{u}_{\mathcal{L}_1}(k-1) + \hat{\boldsymbol{\sigma}}_m(kT)\big)e^{-\gamma T} - \hat{\boldsymbol{\sigma}}_m(kT), \quad (12)$$

for all $k \in \mathbb{N}$ and γ is the cut-off frequency. Training of the KNODE model is done in PyTorch using *torchdiffeq* [31].

IV. EXPERIMENTS AND RESULTS

In our experiments and evaluation procedure, we set out to answer the following questions about our proposed framework: (i) How accurate are the uncertainties estimated by the $\mathcal{L}_1$ adaptive module? (ii) How much performance improvement do the variants of the $\mathcal{L}_1$-KNODE-MPC provide, in the presence of either matched, unmatched or both types of uncertainties? (iii) Can such performance improvements observed in simulations be carried forward and validated in physical experiments? To address the first question, we apply time-varying uncertainties to the quadrotor in simulation and compare the uncertainty estimates against the ground truth. For the other two questions, we conduct tests in both simulations and physical experiments and compare the proposed variants against four benchmarks.

The first benchmark is a nonlinear MPC framework. It uses the nominal model in (3) and does not account for any uncertainties. The second benchmark is KNODE-MPC [16]. Comparison with this benchmark allows us to quantify the improvement brought forth by $\mathcal{L}_1$ adaptation. As a third benchmark, we consider a non-learning MPC framework that directly includes the $\mathcal{L}_1$ adaptive module, which we refer to as $\mathcal{L}_1$-MPC. A similar framework is presented in [21]. This framework combines the $\mathcal{L}_1$ adaptation at the level of the control input signals, similar to $\mathcal{L}_1$-KNODE-MPC-*Direct*, but does not consider any learning or data-driven components. The last benchmark that we consider is KNODE-MPC-Online [17]. This framework uses online learning to update the dynamics model using data collected during robot deployment.

A. Simulations

We first construct a quadrotor model using the equations given in (3). An explicit 5$^{\text{th}}$ order RK method (RK45) with a time step of 2 milliseconds is used for numerical integration to simulate dynamic responses for the quadrotor system. It is assumed that the model predictive controller and the $\mathcal{L}_1$ adaptation module have access to the states of the quadrotor. The control commands are generated using the benchmarks and the proposed control algorithms. These act as inputs to

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

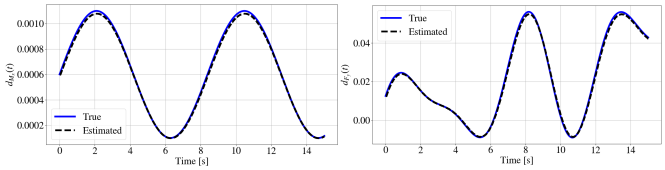


Fig. 2. **Uncertainty estimates:** Time histories of the uncertainties estimated by the adaptive module, against the true uncertainties. Left panel: Roll moment disturbance $d_{M_x}(t)$. Right panel: Side force disturbance $d_{F_y}(t)$.

the quadrotor model. The KNODE model is trained using data collected from flying on a circular trajectory of radius 3m and speed 1m/s. The NN in the KNODE model has one hidden layer with 32 neurons and uses the hyperbolic tangent activation function. It is important to note that the NN is only required to characterize the residual dynamics and not the full dynamics, and this enables the lightweight structure. Being lightweight is beneficial because it not only eases the training process in terms of sample efficiency, but also allows the optimization problem (4) to be solved efficiently. To test the efficacy of the KNODE model, we consider a nominal uncertainty in the form of a mass difference. The quadrotor has a true mass of 0.04kg, while the mass within the nominal model is set to be 0.03kg. The KNODE model is expected to account for this mass difference after training, but not other disturbances that occur during subsequent deployments. Additionally, we include time and state-dependent uncertainties during flight under the following test cases:

- Case 1: Roll moment disturbance
$$d_{M_x}(t) := 5 \times 10^{-4} \sin(0.75t) + 6 \times 10^{-4}, \tag{13}$$
- Case 2: Roll moment and side force disturbances
$$d_{M_x}(t),\, d_{F_y}(t) := 0.025(\sin(t) + 0.5\cos(1.5t) + 0.1t), \tag{14}$$
- Case 3: Case 2 with state-dependent disturbances
$$d_{M_x}(t), d_{F_y}(t),\, d_F(\mathbf{x}) := -\left(\mathbf{C}_d/m\right) \operatorname{sign}(\mathbf{v}_b)\, \mathbf{v}_b^2, \tag{15}$$

where $\mathbf{v}_b \in \mathbb{R}^3$ are the velocities in the body reference frame, $\mathbf{C}_d \in \mathbb{R}^3$ are linear coefficients, with products computed element-wise. Case 1 considers only matched uncertainties, while Cases 2 and 3 include both matched and unmatched uncertainties. For the $\mathcal{L}_1$ adaptive module, we set $\mathbf{A} := -\mathbf{I}$ and the low-pass filter to be the identity function in our simulations.

To ascertain the accuracy of the uncertainty estimates, we apply the disturbances in Case 2 to the quadrotor separately, and compare the true and estimated uncertainties. As depicted in Fig. 2, the adaptive module is able to estimate the disturbances accurately. Next, closed-loop simulations are conducted for each of the benchmarks and the proposed $\mathcal{L}_1$-KNODE-MPC variants. For each of the test cases, we simulate the quadrotor with circular and lemniscate flight profiles of radii 3m and 6m, and at reference speeds ranging from 0.5 to 2 m/s. Each simulation has a flight duration of 15 seconds. To evaluate the trajectory tracking performance of the proposed framework, we compute the root mean squared errors (RMSE) between the true and reference position vectors. For the KNODE-MPC-Online approach, the RMSEs are obtained by taking the average across 10 runs. Results are shown in Fig. 3. In Case 1 where there are

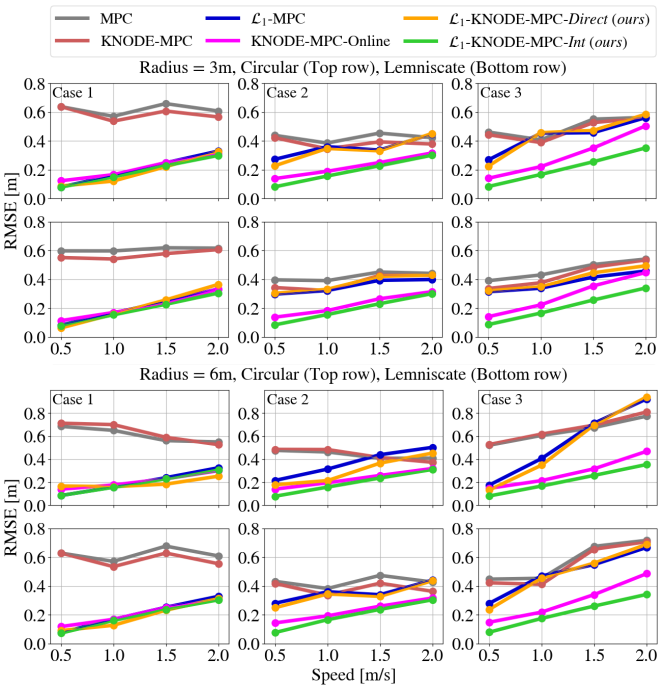


Fig. 3. **Simulation results**: Root mean squared errors (RMSEs) between the true and reference position vectors under the 3 test cases described in Section IV-A. Simulations are conducted for both circular and lemniscate trajectories, across different radii and speeds.

only matched uncertainties, the first proposed variant, $\mathcal{L}_1$-KNODE-MPC-*Direct* outperforms MPC, KNODE-MPC and $\mathcal{L}_1$-MPC on average. In Cases 2 and 3, due to the presence of unmatched uncertainties, there is a noticeable degradation in performance for $\mathcal{L}_1$-MPC and $\mathcal{L}_1$-KNODE-MPC-*Direct*. On the other hand, since $\mathcal{L}_1$-KNODE-MPC-*Int* accounts for *both* matched and unmatched uncertainties, we observe consistently lower RMSEs across all test cases, at different speeds and radii, and for circular and lemniscate trajectories. This demonstrates the efficacy of the tighter integration between the $\mathcal{L}_1$ adaptive module and KNODE-MPC within $\mathcal{L}_1$-KNODE-MPC-*Int*. Overall, $\mathcal{L}_1$-KNODE-MPC-*Int* achieves the best performance, with an average RMSE of 0.2m across all runs, outperforming the benchmarks by at least 18.1%. We also observe that $\mathcal{L}_1$-KNODE-MPC-*Int* incurs a lower computational cost than KNODE-MPC-Online. The latter involves training a NN online, which is more computationally expensive than the computations required in the $\mathcal{L}_1$ adaptive module.

B. Physical Experiments

The setup for our physical experiments is depicted in Fig. 4. The Crazyflie quadrotor has a size of 9 cm^2 and weighs approximately 34g. Linear velocities are estimated from the Vicon measurements through a combination of high and low pass filters, while accelerations and angular velocities are measured from the sensors onboard the quadrotor. For the training of the KNODE model, data is collected by flying the Crazyflie along a circular trajectory of radius 1m at a speed of 0.4m/s, for a duration of 25 seconds. The NODE model has 1 hidden layer of 4 neurons and uses the hyperbolic tangent activation function. The control frameworks run on the base station. These frameworks generate three-

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

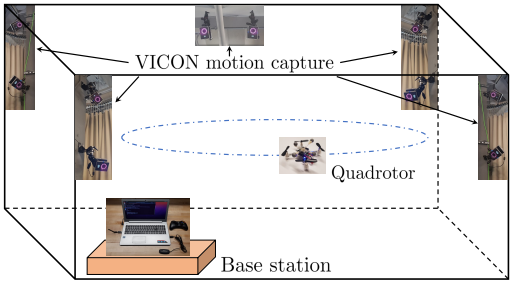


Fig. 4. **Experimental setup**: The open-source Crazyflie 2.1 quadrotor [9] is used as the experimental platform. A laptop running on Intel i7 CPU acts as the base station and communication with the Crazyflie is established via Crazyradio PA at an average rate of 300 Hz. A Vicon motion capture system is deployed to obtain pose information and communicates with the base station at an approximate rate of 100Hz. The CrazyROS wrapper [32] is used as part of the software architecture.

dimensional acceleration commands and run in tandem with a geometric controller [25], as well as attitude and thrust controllers within the Crazyflie firmware. The $\mathcal{L}_1$ adaptive module is formulated by considering the velocities as states, *i.e.*, $\mathbf{z} := \mathbf{v}$. We set $\mathbf{A} := -2\mathbf{I}$ and the bandwidths of the low-pass filters to be 0.125, 0.125 and 0.75 rad/s for the three axes respectively. These filters are implemented for both proposed schemes in the physical experiments. Based on the experimental results, the estimated uncertainties improve closed-loop performance, even with the addition of the low-pass filter. To verify the adaptiveness of the framework, we attach a slung payload of mass 2g during flight at approximately 18.5 seconds after takeoff. We highlight that the addition of this slung payload manifests as both matched and unmatched uncertainties for the quadrotor system. The slung payload is not attached to the quadrotor rigidly and is allowed to move in the x and y axes. This effect is compounded in the second set of experiments where five pieces of payload are attached to one another at individual contact points. To get a better understanding of the physical experiments, we refer the reader to the accompanying video.

For each of the proposed methods and benchmarks, we conduct 10 runs along circular trajectories of radius 1m and speed of 0.4m/s to evaluate closed-loop trajectory tracking performance (a total of 60 runs). First, we investigate the effect of adaptation by plotting the time histories of the quadrotor height under the various control methods. As shown in Fig. 5, the quadrotor experiences a drop in height when the slung payload is attached at t=18.5s. For KNODE-MPC, since the effects of this payload are not present in the training data, it does not account for those effects and the height does not recover to the original height. For KNODE-MPC-Online, because the data collection and training procedure takes time, it took approximately 11 seconds for the height to recover. For the methods with $\mathcal{L}_1$ adaptation, the height recovers within approximately 4 seconds.

Next, to evaluate the overall performance, we compute the RMSEs between the measured and reference position vectors in all 3 axes. Fig. 6 depicts the statistics of the RMSEs. The proposed variants, $\mathcal{L}_1$-KNODE-MPC-*Direct* and *Int*, achieve significant improvements over the benchmarks. Specifically, the median RMSEs for the two variants are 14.1% and 18.4%

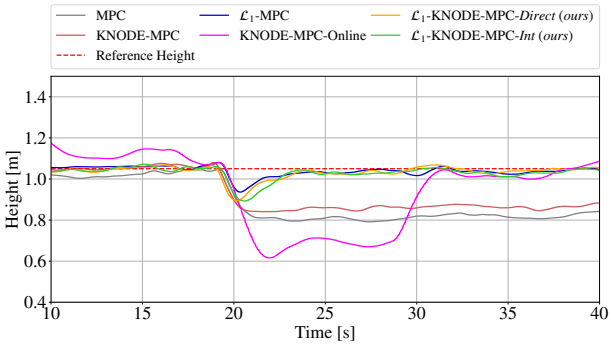


Fig. 5. **Height-time plots (i):** Time histories of the quadrotor height for the benchmarks and proposed variants during flight. A slung payload is attached to the quadrotor system at approximately t=18.5s.

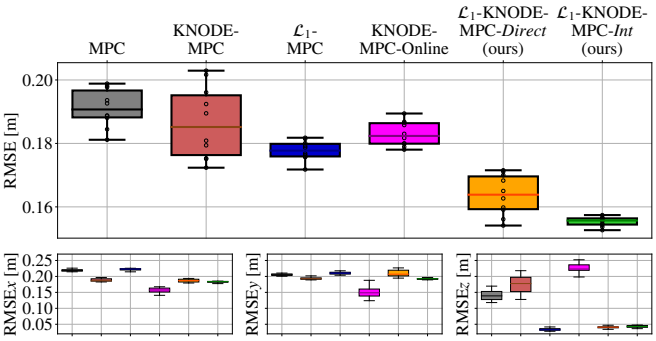


Fig. 6. **Experimental results:** Statistics of the RMSEs for the various MPC frameworks, with a slung payload attached during flight. For each of the box plots, the center line denotes the median and the height of the boxes denote the inter-quantile range. The RMSEs for the individual runs are marked with black circles.

lower than that of MPC. Furthermore, the variance for $\mathcal{L}_1$-KNODE-MPC-*Int* is 76.9% smaller than that of nominal MPC, in terms of inter-quantile ranges. This implies that $\mathcal{L}_1$-KNODE-MPC-*Int* provides more accurate and consistent performance in terms of trajectory tracking. More generally, we observe a reduction in variance when adaptation is incorporated into the control scheme. This implies that the quadrotor is able to achieve more consistent trajectory tracking with adaptation. The performance improvement obtained from the second variant over the first variant validates the tighter integration between the adaptive module and KNODE-MPC, and in particular, with the formulation of the $\mathcal{L}_1$-KNODE model.

To further investigate the adaptiveness of the framework, we consider a more challenging test case in which we attach slung payloads to the quadrotor periodically. Starting from t=18.5s, a piece of payload is added every 15 seconds, until a total of 5 pieces are attached to the quadrotor system. These pieces have a total mass of approximately 4.9g. A depiction of this set of experiments can be found in the accompanying video. Fig. 7 depicts the height-time plots of the quadrotor for this test case. Notably, the $\mathcal{L}_1$ adaptation in the proposed framework allows the quadrotor to track the reference height closely. Although the KNODE-MPC-Online method is able to alleviate the disturbances to some extent, the training of the NN takes time and hence, the height does not recover as quickly as compared to the proposed variants. While the $\mathcal{L}_1$-MPC benchmark provides similar improvements in

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.

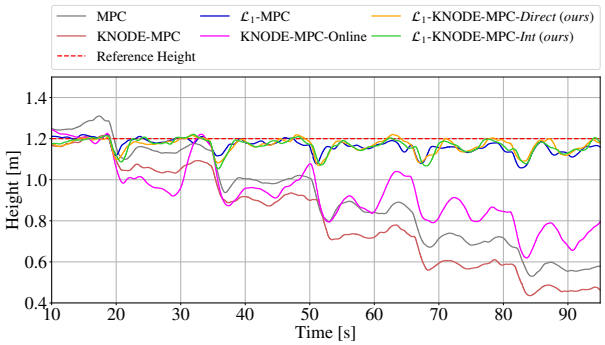


Fig. 7. **Height-time plots (ii):** Time histories of the quadrotor height for the benchmarks and proposed variants, under a more challenging test case in which payloads are added periodically.

terms of height recovery, it has a larger RMSE in the x-y plane, as compared to the proposed variants. Specifically, the x-y RMSE for $\mathcal{L}_1$-MPC is 22.0% and 24.6% higher than those of the two proposed variants. Overall, the average RMSE improvements brought forth by $\mathcal{L}_1$-KNODE-MPC-*Direct* and *Int* are 44.9% and 45.9% above the benchmarks, demonstrating the effectiveness of the framework.

V. CONCLUSION

In this work, we propose $\mathcal{L}_1$-KNODE-MPC, an adaptive learning-enhanced MPC framework that incorporates components from $\mathcal{L}_1$ adaptive control into KNODE-MPC. We present two variants in our framework. The first variant fuses the $\mathcal{L}_1$ adaptive module directly with KNODE-MPC at the level of control inputs, while the second variant integrates the uncertainty estimates from the $\mathcal{L}_1$ adaptive module into the KNODE model, before applying it within KNODE-MPC. Through simulations and physical experiments, we show that the proposed framework achieves significant performance improvements over a number of benchmarks. As future work, we plan to apply $\mathcal{L}_1$-KNODE-MPC to other robotic applications to verify its generalizability.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of T. Z. Jiahao for providing the setup of the physical experiments.

REFERENCES

[1] H. Guo, D. Cao, H. Chen, Z. Sun, and Y. Hu, "Model predictive path following control for autonomous cars considering a measurable disturbance," *Mech. Syst. Signal Process.*, vol. 118, pp. 41–60, 2019.

[2] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *Int. Conf. on Robot. and Autom. (ICRA)*, 2019, pp. 8484–8490.

[3] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*. IEEE, 2018, pp. 6740–6745.

[4] D. Wang, Q. Pan, Y. Shi, J. Hu *et al.*, "Efficient nonlinear model predictive control for quadrotor trajectory tracking," *IEEE Trans. on Cybern.*, vol. 51, no. 10, pp. 5057–5068, 2021.

[5] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robot., and Auto. Syst.*, vol. 3, pp. 269–296, 2020.

[6] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based mpc for autonomous racing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3363–3370, 2019.

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

[8] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in Neural Info. Process. Syst.*, vol. 18, 2005.

[9] Bitcraze, "Crazyflie 2.1." [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/

[10] N. A. Spielberg, M. Brown, and J. C. Gerdes, "Neural network model predictive motion control applied to automated driving with unknown friction," *IEEE Trans. on Control Syst. Tech.*, vol. 30, no. 5, pp. 1934–1945, 2022.

[11] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE RA-L*, vol. 7, no. 4, pp. 10 256–10 263, 2022.

[12] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in Neural Info. Process. Syst.*, vol. 31, 2018.

[13] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ Int. Conf. on intelligent robots and Syst.* IEEE, 2012, pp. 5026–5033.

[14] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE Int. Conf. on Robot. and Automa. (ICRA)*, 2017, pp. 1714–1721.

[15] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid Syst.* Cambridge University Press, 2017.

[16] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A Knowledge-Based Data-Driven Predictive Control Framework for Aerial Robots," *IEEE RA-L*, vol. 7, no. 2, pp. 2819–2826, 2022.

[17] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online dynamics learning for predictive control with an application to aerial robots," *Conf. On Robot Learning (CoRL)*, 2022.

[18] K. Y. Chee, M. A. Hsieh, and N. Matni, "Learning-enhanced nonlinear MPC using KNODEs and deep ensembles," in *Proceedings of The 5th Annual Learning for Dynamics and Controls*, 2023, pp. 1125–1137.

[19] A. Saviolo, J. Frey, A. Rathod, M. Diehl, and G. Loianno, "Active learning of discrete-time dynamics for uncertainty-aware model predictive control," *arXiv preprint arXiv:2210.12583*, 2022.

[20] K. Pereida and A. P. Schoellig, "Adaptive MPC for high-accuracy trajectory tracking in changing conditions," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*, 2018, pp. 7831–7837.

[21] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors," *IEEE Robot. and Autom. Letters*, vol. 7, no. 2, pp. 690–697, 2021.

[22] Z. Wu, S. Cheng, K. A. Ackerman, A. Gahlawat, A. Lakshmanan, P. Zhao, and N. Hovakimyan, "L1 adaptive augmentation for geometric tracking control of quadrotors," in *Int. Conf. on Robot. and Autom. (ICRA)*, 2022, pp. 1329–1336.

[23] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "L1-adaptive mppi architecture for robust and agile control of multirotors," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*, 2020, pp. 7661–7666.

[24] Y. Cheng, P. Zhao, F. Wang, D. J. Block, and N. Hovakimyan, "Improving the robustness of reinforcement learning policies with L1 adaptive control," *IEEE Robot. and Autom. Letters*, vol. 7, no. 3, pp. 6574–6581, 2022.

[25] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE Int. Conf. on Robot. and Autom.* IEEE, 2011, pp. 2520–2525.

[26] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer Science & Business Media, 2010.

[27] T. Z. Jiahao, M. A. Hsieh, and E. Forgoston, "Knowledge-based learning of nonlinear dynamics and chaos," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 11, p. 111101, 2021.

[28] 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, 2019.

[29] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[30] L. F. Shampine, "Some practical runge-kutta formulas," *Mathematics of computation*, vol. 46, no. 173, pp. 135–150, 1986.

[31] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Info. Process. Syst.*, 2018.

[32] W. Hönig and N. Ayanian, *Flying Multiple UAVs Using ROS*. Springer Int. Publishing, 2017, pp. 83–118.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 14:29:43 UTC from IEEE Xplore. Restrictions apply.