Flatness-Preserving Residual Learning for Real-Time Tight Quadrotor Formation Flight *

Pei-An Hsieh[†1], Fengjun Yang[†1], Nikolai Matni[1], and M. Ani Hsieh[1]

[†]Equal contribution

[1]GRASP Laboratory, University of Pennsylvania, PA, USA

August 20, 2026

Abstract

Quadrotors flying in tight formations are severely affected by turbulent aerodynamic interactions, such as downwash, that can cause catastrophic collisions if left unmodeled. To compensate for these effects, we propose a physics-informed residual dynamics learning framework that captures complex aerodynamic interactions while ensuring the joint multi-quadrotor system remains differentially flat. We leverage this preserved flatness to design a computationally efficient feedback linearization controller that is easily tunable with linear control techniques and cancels aerodynamic disturbances via feedforward compensation. Hardware experiments demonstrate our framework reduces average tracking errors by 31% compared to nominal baselines. Crucially, our lightweight approach matches the tracking performance of state-of-the-art nonlinear model predictive control (NMPC) while requiring an order of magnitude less computation. We are the first to show that stable, tight formation flight can be achieved with under 30 seconds of training data and a 5ms loop rate, unlocking high-fidelity aerodynamic compensation for compute-constrained flight stacks.

The video of our physical experiments can be found at https://www.youtube.com/watch?v=uF26IkRFQMk.

1 Introduction

Flying multiple quadrotors in close proximity poses significant challenges for controller design, as turbulent airflow generated by neighboring vehicles can degrade performance or even cause catastrophic collisions if left unaccounted for. Learning residual dynamics from flight data offers a powerful way to model these aerodynamic effects. To integrate general, unstructured residual models into the downstream control pipeline, state-of-the-art frameworks rely on NMPC (Chee et al., 2024). However, NMPC demands substantial computational resources and scales poorly with the complexity of the learned model.

To overcome this computational bottleneck, structures, such as smoothness (Shi et al., 2019, 2021) or specific dependencies (Spitzer and Michael, 2021), can be imposed on the learned residual

*This work was supported in part by NSF Awards SLES-2331880, ECCS-2045834, ECCS-2231349, AFOSR Award FA9550-24-1-0102, DARPA TRS program under contract HR00112590145, ONR Award N000142512070, and ONR Award N000142512171.

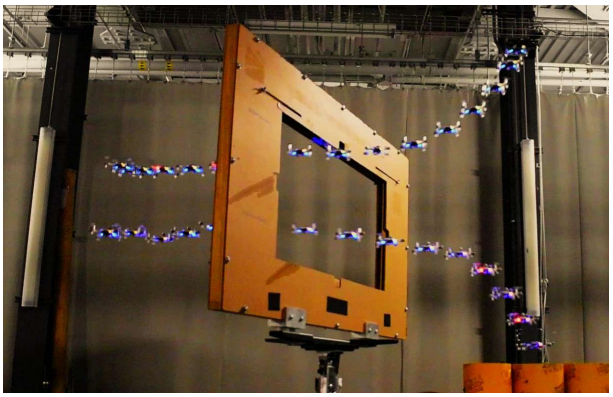

Figure 1: Time-lapse of two Crazyflie quadrotors using our proposed method merging from a 1.1 m *stacked* formation into a tight 0.1 m separation to fly through a 0.4 m high window.

to enable efficient feedback control. Building on this, we propose a physics-informed parameterization of the residual dynamics that strikes a balance between model fidelity and control efficiency by restricting our learned residual model to depend strictly on the joint positions and velocities of the quadrotor team. This ensures that the dynamics augmented by the learned residual are *differentially flat*, a powerful property for quadrotor planning and control (Mellinger and Kumar, 2011). Leveraging differential flatness, we design a computationally efficient controller that is easily tunable with standard linear control design techniques. Through simulation and hardware experiments, we show that our proposed controller achieves tracking performance on par with NMPC baselines (Chee et al., 2024) while requiring an order of magnitude less computation.

1.1 Related Works

1.1.1 Residual Dynamics Learning for Quadrotors

Prior works have successfully integrated learned residual dynamics to improve quadrotor tracking under aerodynamic disturbances. NeuralLander (Shi et al., 2019) and NeuralSwarm (Shi et al., 2021) enforce smoothness of the learned residual to establish stability guarantees for a cascaded controller. Unlike their approach, we utilize a more data-efficient, physics-informed parameterization and explicitly differentiates the learned residual to provide anticipatory feedforward attitude compensation to reduce phase lag during disturbances. Spitzer and Michael (2021) similarly restrict their residual dependency for feedback linearization, but only consider single-vehicle acceleration disturbances. In contrast, we consider multi-quadrotor settings with a physics-informed residual parameterization and implement a different controller. Alternatively, the KNODE-MPC framework (Chee et al., 2024, 2022; Hsieh et al., 2025) learns unconstrained, highly expressive residual dynamics that can be integrated into a downstream NMPC. While they achieve impressive performance, the optimization-based NMPC poses significant computational challenges for resource-constrained flight stacks. We directly compare our approach with this baseline in an identical hardware setup and show that our proposed controller matches their tracking performance while using 20× less compute.

1.1.2 Addressing Unmodeled Dynamics in Flatness-Based Control

Flatness-preserving aerodynamic disturbances have been explored analytically in Faessler et al. (2017) for quadrotors, who show that a specific, hand-derived aerodynamic drag model preserves differential flatness for a single quadrotor. Recently, Yang et al. (2025a) proposed a theoretical framework for learning flatness-preserving residual dynamics using structured parameterizations. We leverage this framework and demonstrate its first successful hardware application for tight formation flight. Finally, rather than explicitly modeling residuals from data offline, an alternative is to actively estimate and reject the unmodeled dynamics online (Hsieh et al., 2025; Join et al., 2024). Integrating online estimation with our framework remains a promising direction for future work.

1.2 Contributions

Our contributions are fourfold.

- First, we show that the joint differential flatness and flat outputs of a multi-quadrotor team are preserved when the nominal rigid-body dynamics are augmented with a structured parameterization of the residual dynamics. We then explicitly derive the mappings to reconstruct states and inputs from flat outputs and their derivatives (Sec. 3.2).
- Second, we leverage the flatness-preserving structure to design a residual dynamics learning framework that utilizes known physical structures to enable sample-efficient learning. (Sec. 4).
- Third, we design a computationally efficient controller that leverages the learned residual dynamics to compensate for predicted aerodynamic disturbances (Sec. 3.3).
- Finally, we evaluate the proposed method in both simulation and hardware experiments and show that our proposed controller achieves similar tracking performance to an NMPC baseline while requiring an order of magnitude less computation (Sec. 6).

To the best of our knowledge, this is the first time feedback linearization controllers with learned residuals have been applied to tight quadrotor formation flight.

2 Problem Formulation

We consider a system of N interacting quadrotors. Let $i \in \{1, \dots, N\}$ denote the index of an individual vehicle. The dynamics for agent i are given by:

$$\dot{\mathbf{p}}^i = \mathbf{v}^i, \tag{1a}$$
$$m^i \dot{\mathbf{v}}^i = u^i \mathbf{R}^i \mathbf{e}_3 + m^i \mathbf{g} + \mathbf{d}^{a,i}, \tag{1b}$$
$$\dot{\mathbf{R}}^i = \mathbf{R}^i \left[\boldsymbol{\omega}^i\right]_\times, \tag{1c}$$
$$J^i \dot{\boldsymbol{\omega}}^i = J^i \boldsymbol{\omega}^i \times \boldsymbol{\omega}^i + \boldsymbol{\tau}^i + \mathbf{d}^{\tau,i}. \tag{1d}$$

where $\mathbf{p}^i, \mathbf{v}^i \in \mathbb{R}^3$ are the position and velocity of vehicle i in the world frame $\mathcal{W}$, and $\mathbf{R}^i \in \mathrm{SO}(3)$ its orientation. Its angular velocity $\boldsymbol{\omega}^i$ is expressed in the body-fixed frame $\mathcal{B}$. The world frame follows an East-North-Up convention, with the three axes denoted as $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. The inputs are

thrust $u^i \in \mathbb{R}$ and torque $\boldsymbol{\tau}^i \in \mathbb{R}^3$. The disturbances $\mathbf{d}^{a,i}$ and $\mathbf{d}^{\tau,i} \in \mathbb{R}^3$ arise from complex multi-agent aerodynamic effects. When the vehicles are sufficiently separated, these effects are small, and controllers can be designed for just the *nominal* rigid-body dynamics that ignore such effects, i.e., treating $\mathbf{d}^{a,i} = \mathbf{d}^{\tau,i} = \mathbf{0}$.

However, when the quadrotors are flying in tight formations, complex aerodynamic effects like downwash inject significant disturbances ($\mathbf{d}^{a,i}, \mathbf{d}^{\tau,i} \gg \mathbf{0}$) that can degrade tracking performance if left unaccounted for. Residual dynamics learning addresses this by using past trajectory data to learn approximate residual models $\hat{\mathbf{f}}^{a,i}$ and $\hat{\mathbf{f}}^{\tau,i}$ that predict these perturbations from a state-dependent feature vector $\boldsymbol{\xi}$, yielding the *augmented* acceleration dynamics for agent i[1] as:

$$m^i \dot{\mathbf{v}}^i = u^i \mathbf{R}^i \mathbf{e}_3 + m^i \mathbf{g} + \hat{\mathbf{f}}^{a,i}(\boldsymbol{\xi}), \tag{2a}$$

$$J^i \dot{\boldsymbol{\omega}}^i = J^i \boldsymbol{\omega}^i \times \boldsymbol{\omega}^i + \boldsymbol{\tau}^i + (\mathbf{R}^i)^\top \hat{\mathbf{f}}^{\tau,i}(\boldsymbol{\xi}). \tag{2b}$$

While the most general parameterization of $\hat{\mathbf{f}}^{a,i}$ and $\hat{\mathbf{f}}^{\tau,i}$ using the full multi-agent state maximizes expressivity, incorporating it into downstream planning and control is non-trivial and often requires computationally expensive NMPC (Chee et al., 2024). To capture and reject unmodeled aerodynamic effects while minimizing the computational burden, we parameterize $\boldsymbol{\xi}$ to depend only on the vehicles' positions and velocities. Crucially, this parameterization renders the joint augmented dynamics of the quadrotor team *differentially flat*. This enables the use of flatness-based controllers that not only enjoy computational benefits but are also tunable with standard linear techniques (Sec. 3). Further, to mitigate potential generalization loss from this restricted expressivity, we parameterize the residual as the sum of a reduced-order model (ROM) and a learned term (Sec. 4). This formulation ultimately enables an efficient control pipeline, which we demonstrate experimentally in Sec. 5.

3 Flatness-Preserving Residuals

Informally, a system is differentially flat if its states and control inputs can be entirely reconstructed from a set of *flat outputs* and their time derivatives (Fliess et al., 1995). This mapping between the flat outputs and system states establishes an equivalence between the nonlinear system and a linear one, thereby simplifying planning and control. In what follows, we first briefly review the differential flatness of a single quadrotor. We then introduce our parameterization of the residual dynamics and show that it renders the joint system differentially flat, allowing us to design controllers with simple linear design techniques. To reduce the notational burden, we drop the agent superscript in the derivations and present the formulation from the perspective of an ego-quadrotor, treating the states of all other agents as an exogenous parameter.

3.1 Background on Quadrotor Flatness

The nominal rigid-body quadrotor dynamics (i.e. Eq. (1) with $\mathbf{d}^{a,i} = \mathbf{d}^{\tau,i} = \mathbf{0}$) are known to be differentially flat with respect to the flat outputs $\mathbf{y} = [\mathbf{p}, \psi]$, where ψ is the yaw angle of the vehicle (Mellinger and Kumar, 2011). We briefly recall how the state $[\mathbf{p}, \mathbf{v}, \mathbf{R}, \boldsymbol{\omega}]$ and control inputs $[u, \boldsymbol{\tau}]$ can be reconstructed from $\mathbf{y}$ and its time derivatives.

[1] We let $\hat{\mathbf{f}}^{\tau,i}$ represent the residual torque in the world frame. It is thus mapped to the body frame via $(\mathbf{R}^i)^\top = \mathbf{R}^{\mathcal{B}}_{\mathcal{W}}$ in (2b).

Using the shorthand $\mathbf{z} := \mathbf{R}\mathbf{e}_3$ for the unit vector in the direction of the body z-axis and $\mathbf{a} := \ddot{\mathbf{p}}$ as the acceleration, recall the translational dynamics (1b)

$$m\mathbf{a} = u\mathbf{z} + m\mathbf{g}. \tag{3}$$

Since $\mathbf{z}$ is a unit vector, the commanded thrust can be directly recovered from (3) as:

$$u = m\left\|\mathbf{a} - \mathbf{g}\right\|. \tag{4}$$

Rearranging this equation gives the body z-axis

$$\mathbf{z} = \frac{m}{u}(\mathbf{a} - \mathbf{g}). \tag{5}$$

Crucially, because $\mathbf{a} = \ddot{\mathbf{p}}$, both u and $\mathbf{z}$ are entirely determined by the second derivatives of the flat outputs. The full orientation is then determined by the yaw angle ψ. Defining the heading vector $\bar{\mathbf{x}} = [\cos\psi, \sin\psi, 0]^\top$, the remaining body axes can be found as:

$$\mathbf{y} = \frac{\mathbf{z} \times \bar{\mathbf{x}}}{\|\mathbf{z} \times \bar{\mathbf{x}}\|}, \quad \mathbf{x} = \mathbf{y} \times \mathbf{z}. \tag{6}$$

To find the angular velocity, we must differentiate $\mathbf{z}$, which naturally introduces a dependence on the jerk $\mathbf{j} := \dddot{\mathbf{p}}$. We define the world-frame angular velocity excluding the z-axis rotation as $\boldsymbol{\omega}_{xy}^{\mathcal{W}} := \mathbf{z} \times \dot{\mathbf{z}}$. Differentiating the acceleration equation (3) yields

$$m\mathbf{j} = \dot{u}\mathbf{z} + u\dot{\mathbf{z}}. \tag{7}$$

Taking the cross product of $\mathbf{z}$ with (7) isolates the x and y-axes angular velocities as

$$\boldsymbol{\omega}_{xy}^{\mathcal{W}} = \frac{m}{u}\mathbf{z} \times \mathbf{j}. \tag{8}$$

Projecting this onto the body frame, we get that

$$\boldsymbol{\omega}_{xy}^{\mathcal{B}} = \mathbf{R}^\top \boldsymbol{\omega}_{xy}^{\mathcal{W}} = \frac{m}{u}\mathbf{e}_3 \times \mathbf{R}^\top \mathbf{j}. \tag{9}$$

The z-axis angular velocity ω_z is recovered from the yaw trajectory using standard quadrotor kinematics (Mellinger and Kumar, 2011):

$$\omega_z = \frac{\omega_x(\mathbf{z}^\top \bar{\mathbf{x}}) + \left(\mathbf{y}^\top \dot{\bar{\mathbf{x}}}\right)}{\mathbf{x}^\top \bar{\mathbf{x}}}, \tag{10}$$

giving the full angular velocity $\boldsymbol{\omega} = \boldsymbol{\omega}_{xy}^{\mathcal{B}} + \omega_z \mathbf{e}_3$. Finally, differentiating the angular velocity yields $\dot{\boldsymbol{\omega}}$, which can be substituted into (1d) to recover the torque:

$$\boldsymbol{\tau}^{\mathcal{B}} = J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times J\boldsymbol{\omega}. \tag{11}$$

As a result, any sufficiently smooth trajectory of the flat outputs can be mapped to a dynamically feasible state and input trajectory via (3)-(11). Thus, certain planning problems reduce to finding smooth flat output trajectories (Mellinger and Kumar, 2011), and feedback controllers can be designed directly for the equivalent linear system (See e.g. Fliess et al. (1995)). However, introducing a residual can easily break this chain of algebraic substitutions. Next, we demonstrate that flatness is preserved provided the residual terms $\hat{\mathbf{f}}^a$ and $\hat{\mathbf{f}}^\tau$ depend strictly on the position and velocity of the quadrotor team.

3.2 Flatness under Augmented Dynamics

We now consider the augmented dynamics (2), which incorporate the learned residual terms. The acceleration and jerk equations become:

$$m\mathbf{a} = u\mathbf{z} + m\mathbf{g} + \hat{\mathbf{f}}^a(\boldsymbol{\xi}), \tag{12a}$$

$$m\mathbf{j} = \dot{u}\mathbf{z} + u\dot{\mathbf{z}} + \dot{\hat{\mathbf{f}}}^a(\boldsymbol{\xi}). \tag{12b}$$

As a result, if the residual feature vector $\boldsymbol{\xi}$ included the thrust u, for example, solving for u would become highly nontrivial. To preserve and reuse the chain of algebraic substitutions–and hence the flatness–of the nominal dynamics, we restrict our residual models $\hat{\mathbf{f}}^a(\boldsymbol{\xi})$ and $\hat{\mathbf{f}}^\tau(\boldsymbol{\xi})$ to depend exclusively on the joint kinematic state of the swarm:

$$\boldsymbol{\xi} = [\mathbf{p}^1, \mathbf{v}^1, \ldots, \mathbf{p}^N, \mathbf{v}^N]^\top. \tag{13}$$

We now show that in this case, we can reconstruct the state and control inputs from the same flat outputs $\mathbf{y} = [\mathbf{p}, \psi]$ via small adjustments to the nominal procedure.

First, we adapt (4) and (5) to recover the thrust and body z-axis by subtracting the acceleration residual:

$$u = \left\| m(\mathbf{a} - \mathbf{g}) - \hat{\mathbf{f}}^a(\boldsymbol{\xi}) \right\|, \tag{14a}$$

$$\mathbf{z} = \frac{m(\mathbf{a} - \mathbf{g}) - \hat{\mathbf{f}}^a(\boldsymbol{\xi})}{u}. \tag{14b}$$

The remaining body axes $\mathbf{x}$ and $\mathbf{y}$ are computed exactly as in (6) using the yaw angle. To find the angular velocity, we again isolate the x and y angular velocities from (12b) and project them into the body frame, yielding

$$\boldsymbol{\omega}_{xy}^{\mathcal{B}} = \frac{1}{u}\mathbf{e}_3 \times \mathbf{R}^\top \left(m\mathbf{j} - \dot{\hat{\mathbf{f}}}^a(\boldsymbol{\xi}) \right), \tag{15}$$

where the residual derivative $\dot{\hat{\mathbf{f}}}^a$ can be computed via the chain rule, $\dot{\hat{\mathbf{f}}}^a(\boldsymbol{\xi}) = \left[D_{\boldsymbol{\xi}} \hat{\mathbf{f}}^a(\boldsymbol{\xi}) \right] \dot{\boldsymbol{\xi}}$. The Jacobian $D_{\boldsymbol{\xi}} \hat{\mathbf{f}}^a$ can be obtained via automatic differentiation, and $\dot{\boldsymbol{\xi}}$ consists of the swarm's concatenated velocities and accelerations. The z-axis rate ω_z is recovered identically to the nominal case as in (10). Finally, incorporating the torque residual, the torque is found by rearranging (2b):

$$\boldsymbol{\tau} = J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times J\boldsymbol{\omega} - \mathbf{R}^\top \hat{\mathbf{f}}^\tau(\boldsymbol{\xi}). \tag{16}$$

Finally, because this algebraic mapping holds identically for every vehicle, the multi-quadrotor system is jointly flat with respect to the concatenated flat outputs of the individual vehicles. While we focus on two-quadrotor teams for this paper, this joint flatness provides an amenable structure for distributed flatness-based control in larger swarms (Yang et al., 2025b).

3.3 Flatness-based Controller Design

Assuming noisy measurements of $\mathbf{p}, \mathbf{v}$, and $\mathbf{R}$, we leverage flatness to design a cascaded feedback linearization controller that tracks a reference trajectory $\mathbf{p}_r$. We first compute a commanded acceleration via PID feedback on the position error:

$$\mathbf{a} = \mathbf{a}_r - k_p(\mathbf{p} - \mathbf{p}_r) - k_i \int (\mathbf{p} - \mathbf{p}_r)dt - k_d(\mathbf{v} - \mathbf{v}_r), \tag{17}$$

where $\mathbf{v}_r$ and $\mathbf{a}_r$ are the reference velocity and acceleration. This commanded acceleration maps directly to the required thrust u via (14a).

To compute the desired angular velocity $\boldsymbol{\omega}_r$, we differentiate (17) to find the corresponding jerk:

$$\mathbf{j} = \mathbf{j}_r - k_p(\mathbf{v} - \mathbf{v}_r) - k_i(\mathbf{p} - \mathbf{p}_r) - k_d(\mathbf{a} - \mathbf{a}_r). \tag{18}$$

The desired angular velocity $\boldsymbol{\omega}_r$ is then recovered using (15) and (10). Because direct IMU acceleration measurements are often noisy and biased, we estimate the current vehicle acceleration $\mathbf{a}$ from measured positions and velocities using an $\alpha - \beta - \gamma$ filter (Brookner, 1998).

While the exact feedforward torque could be found via (16), evaluating the term $\dot{\boldsymbol{\omega}}_r$ involves computing the second-order time derivative of the residual, $\ddot{\hat{\mathbf{f}}}^a(\boldsymbol{\xi})$. This exact evaluation adds computational overhead and is highly susceptible to noise amplification from high-order state derivatives. Therefore, we instead rely on a high-gain proportional controller on the angular velocity error to find the torque input

$$\boldsymbol{\tau} = -K_\omega(\omega - \omega_r) - \mathbf{R}^\top \hat{\mathbf{f}}^\tau(\boldsymbol{\xi}), \tag{19}$$

where the torque residual $\hat{\mathbf{f}}^\tau(\boldsymbol{\xi})$ is incoporated as a feedforward compensation.

Finally, we note that although exact feedback linearization is possible via dynamic extension (Spitzer and Michael, 2021), it requires differentiating the acceleration residual $\hat{\mathbf{f}}^a$ to find $\ddot{u}$, which can filter out important low-frequency signals. We empirically found our approach to be significantly more robust for hardware deployment.

4 Physics-Informed Residual Learning

To compensate for the potential generalization loss from preserving flatness, we inject strong inductive biases into our residual parameterization by explicitly encoding structural symmetries and introducing a ROM of the downwash dynamics.

Consider a pair of quadrotors with relative position $\Delta\mathbf{p} := \mathbf{p}^2 - \mathbf{p}^1 = [\Delta p_x, \Delta p_y, \Delta p_z]^\top$. We define the radial separation as $\Delta r := \sqrt{\Delta p_x^2 + \Delta p_y^2}$ and the vertical separation as $\Delta z := \Delta p_z$. Assuming near-hover flight for the upper quadrotor ($\mathbf{z}^1 \approx \mathbf{e}_3$), we construct a local flow-field frame $\mathcal{F}$ centered at the upper quadrotor. Its orthonormal axes $\{\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$ are defined as:

$$\mathbf{e}_z = -\mathbf{e}_3,\ \mathbf{e}_x = \mathbf{e}_y \times \mathbf{e}_z,\ \mathbf{e}_y = \frac{\mathbf{e}_z \times \Delta\mathbf{p}}{\|\mathbf{e}_z \times \Delta\mathbf{p}\|_2}. \tag{20}$$

We denote the rotation matrix from $\mathcal{F}$ to the world frame $\mathcal{W}$ as $R_{\mathcal{F}}^{\mathcal{W}} \in SO(3)$. By projecting the relative kinematics into this frame, we form a rotationally equivariant feature vector $\mathbf{h}_0 = [\Delta z, \Delta r]^\top$, while satisfying our flatness-preserving conditions.

Leveraging this invariant feature space, we parameterize the total residual dynamics $\hat{\mathbf{f}}(\boldsymbol{\xi})$ as the sum of a physics-based ROM $\hat{\mathbf{f}}_d$ and a learned neural network $\hat{\mathbf{f}}_\Theta$:

$$\hat{\mathbf{f}}(\boldsymbol{\xi}) = \hat{\mathbf{f}}_d(\mathbf{h}_0) + \hat{\mathbf{f}}_\Theta(\mathbf{h}_0), \tag{21}$$

where each term can be broken into an acceleration component (e.g. $\hat{\mathbf{f}}_d^a$) and a torque component (e.g. $\hat{\mathbf{f}}_d^\tau$).

The ROM term $\hat{\mathbf{f}}_d(\mathbf{h}_0)$ is adapted from the pairwise downwash model in Chee et al. (2024) and serves as a physically grounded foundation. It uses Δz and Δr to estimate the induced airflow

velocity from the upper quadrotor (Bauersfeld et al., 2024) and then applies standard drag equations to compute the resulting forces and torques on the lower vehicle (Jain et al., 2019) (See details in Appendix A).

To capture the remaining unmodeled dynamics, the learned component $\hat{\mathbf{f}}_\Theta(\mathbf{h}_0) := [\hat{\mathbf{f}}^a_{\Theta^a}(\mathbf{h}_0), \hat{\mathbf{f}}^\tau_{\Theta^\tau}(\mathbf{h}_0)]^\top$ is parameterized as an L-layer feedforward neural network. For the acceleration residual:

$$\begin{aligned} \mathbf{h}_{i+1} &= \sigma(W^a_i \mathbf{h}_i + \mathbf{b}^a_i), \quad i = 0, \dots, L-2 \\ \hat{\mathbf{f}}^a_{\Theta^a}(\mathbf{h}_0) &= M_a [W^a_{L-1}\mathbf{h}_{L-1} + \mathbf{b}^a_{L-1}]^\top, \end{aligned} \tag{22}$$

where $\Theta^a := \{W^a_l, \mathbf{b}^a_l\}_{l=0}^{L-1}$ are the learnable weights and biases, and $\sigma(\cdot)$ is a continuously differentiable activation function (e.g., tanh) ensuring smoothness. The matrix $M_a = R^{\mathcal{W}}_{\mathcal{F}}/m$ transforms the network output from $\mathcal{F}$ to the world frame with the appropriate mass scaling. The torque residual $\hat{\mathbf{f}}^\tau_{\Theta^\tau}(\mathbf{h}_0)$ is constructed identically with parameters Θ^τ and scaling matrix $M_\tau = J^{-1}R^{\mathcal{W}}_{\mathcal{F}}/\|J^{-1}\|_F$, where $\|\cdot\|_F$ is the Frobenius norm.

Finally, to train the network parameters $\Theta = \{\Theta^a, \Theta^\tau\}$, we employ Knowledge-based Neural Ordinary Differential Equations (KNODE) (Chee et al., 2022; Jiahao et al., 2021), which avoids explicit finite differencing of noisy IMU measurements and the resulting biases. Given a dataset of K trajectory samples $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{u}_k)\}_{k=1}^K$, we optimize the weights using the Adam optimizer (Kingma and Ba, 2014) to minimize the weighted mean squared error between the integrated predictions $\hat{\mathbf{x}}_k(\Theta)$ and the ground-truth states:

$$\mathcal{L}(\Theta) := \frac{1}{K-1}\sum_{k=2}^{K} \|\hat{\mathbf{x}}_k(\Theta) - \mathbf{x}_k\|^2_{W_x}. \tag{23}$$

5 Experiment

We validate our residual dynamics learning framework and flatness-based controller through simulation and hardware experiments. Our evaluations demonstrate three key results: (1) residuals learned with our flatness-preserving parameterization accurately capture complex downwash effects; (2) our proposed feedback linearization (FBL) controller (Section 3.3) significantly reduces tracking errors by leveraging this learned residual; and (3) our approach achieves tracking accuracy comparable to a state-of-the-art nonlinear MPC (Chee et al., 2024) while reducing computation by an order of magnitude. Additionally, we show that our learned residual can provide feedforward compensation for standard SE(3) geometric controllers (Lee et al., 2010), highlighting its broad applicability.

To systematically validate these claims, we conduct simulation and hardware experiments that evaluate both the residual learning framework and the proposed FBL controller in a two-quadrotor *stacked* formation flight, where the *bottom* quadrotor flies directly within the turbulent wake of a *top* quadrotor at a fixed vertical separation. Both vehicles execute straight, minimum-snap trajectories in the same direction, forcing the bottom quadrotor to constantly compensate for significant aerodynamic disturbances.

Further, to isolate the contributions from the reduced-order downwash model and the learned neural network component (21), we conduct ablation studies over four model fidelities:

- **Nominal** ($\hat{\mathbf{f}} = 0$): ignores aerodynamic effects.

- **ROM** ($\hat{\mathbf{f}} = \hat{\mathbf{f}}_d$): includes the ROM of the downwash dynamics but does not use learning.
- **Learned w/o ROM** ($\hat{\mathbf{f}} = \hat{\mathbf{f}}_\Theta$): learns the residual dynamics directly on top of the rigid-body quadrotor dynamics.
- **Learned + ROM** ($\hat{\mathbf{f}} = \hat{\mathbf{f}}_d + \hat{\mathbf{f}}_\Theta$): learns the residual on top of both the rigid-body quadrotor dynamics and the ROM of downwash dynamics.

Across the various experiments below, we demonstrate that both the ROM and the learned neural network are crucial to the overall performance gains.

5.1 Simulation Experiments

Setup: We simulate the dynamics (1) using the mass and moment of inertia of the Crazyflie quadrotors, subject to the downwash model from Chee et al. (2024). To emulate the discrepancy between our ROM and true aerodynamic disturbances, we set the drag coefficient $C_D = 0.7$ for the simulated environment and $C_D = 0.3$ for the ROM. Crucially, the simulated downwash model depends on variables outside our flatness-preserving parameterization (13) (e.g., vehicle orientations). Thus, this setup rigorously tests whether our strictly kinematic features can approximate complex, highly coupled aerodynamic interactions in the given scenario.

To capture residual forces and torques, we parameterize $\hat{\mathbf{f}}^a_{\Theta^a}$ and $\hat{\mathbf{f}}^\tau_{\Theta^\tau}$ as two separate neural networks with a single hidden layer of nine neurons each. For data collection, we utilize both the aforementioned *stacked* formation, as well as a *static top* formation, where the top quadrotor hovers while the bottom quadrotor executes a straight, constant-speed trajectory directly beneath it. The training dataset $\mathcal{D}$ comprises 18 seconds of flight data collected using the FBL controller augmented with the ROM and consists of four scenarios: *static top* and *stacked* formations at vertical separations of 0.3 m and 0.4 m. The loss function (23) utilizes a diagonal weight matrix $W_x \in \mathbb{R}^{13\times 13}$, with the final three elements set to 0.1 and all others to 1.

To test the controllers, we assume full, noise-free state observability. This isolates controller performance from the state estimator, a separation that is difficult to achieve in physical experiments. The FBL controller generates thrust and torque commands at 100 Hz, while the simulator integrates the dynamics via an RK45 method with a 10 ms time step. For the SE(3) controllers (Lee et al., 2010), the learned residual modifies the desired orientation via the exact same mapping defined in (14b).

Results: We first evaluate the prediction error of our learned residual model both within and slightly outside its training distribution. We gather evaluation trajectories by flying two quadrotors in a stacked formation at vertical separations of $0.2, 0.3, 0.4$, and 0.5m, sampling the bottom quadrotor's state at 100 Hz. Because we have access to the ground-truth downwash forces $\mathbf{d}^a$ and torques $\mathbf{d}^\tau$ in simulation, we define the acceleration and angular acceleration prediction errors for a given feature vector $\boldsymbol{\xi}$ as:

$$\mathbf{e}^a(\boldsymbol{\xi}) = \frac{1}{m}\left\|\mathbf{d}^a(\boldsymbol{\xi}) - \hat{\mathbf{f}}^a_\Theta(\boldsymbol{\xi})\right\|,$$
$$\mathbf{e}^\tau(\boldsymbol{\xi}) = \left\|J^{-1}\left(\mathbf{d}^\tau(\boldsymbol{\xi}) - \hat{\mathbf{f}}^\tau_\Theta(\boldsymbol{\xi})\right)\right\|.$$

We report the root mean square error (RMSE) of these predictions in Table 1. Overall, incorporating the reduced-order downwash model and the neural network residual yields significant performance gains. Our proposed learned model consistently achieves low acceleration prediction error,

including on out-of-distribution (OOD) trajectories at 0.2m and 0.5m vertical separation. For angular acceleration, the learned model performs well in-distribution and at closer proximity (0.2m); however, at the larger 0.5m separation, just using the ROM slightly outperforms it. This suggests that the neural network may mildly overfit to the highly turbulent close-proximity data, thereby marginally weakening its generalization to angular dynamics at larger, less perturbed distances. On the other hand, learning the residual directly from data without leveraging the structured prior can achieve competitive translational acceleration error in some cases, but leads to substantially larger angular acceleration errors. Moreover, the ROM provides the largest gains in regimes with stronger aerodynamic coupling, particularly at smaller vertical separations where extrapolation from limited data is more challenging. These results highlight the value of physics-based structure for improving generalization under limited data.

Regime	z **[m]**	**Nominal**	**ROM**	**Learned w/o ROM**	**Learned + ROM**
In-Dist.	0.3	0.57 / 13.75	0.29 / 7.86	0.13 / 9.99	**0.12** / **6.01**
	0.4	0.52 / 11.55	0.27 / 6.80	**0.09** / 10.09	0.10 / **6.02**
OOD	0.2	0.62 / 15.98	0.31 / 8.90	0.18 / 10.84	**0.15** / **6.52**
	0.5	0.47 / 9.84	0.25 / **6.05**	**0.08** / 10.78	**0.08** / 6.36

Table 1: Residual Prediction Error RMSE ($\mathbf{e}^a$ / $\mathbf{e}^\tau$).

To systematically evaluate the closed-loop tracking performance of the FBL and SE(3) controllers, we simulate an array of reference trajectories with varying vertical separations (0.2 to 0.6m) and reference speeds (0.2 to 0.5m/s). We measure performance using the 3D position RMSE and the maximum vertical tracking error (z_{max}), as downwash predominantly disrupts the vertical axis. Figure 2 visualizes the performance of the FBL controller across this sweep. Incorporating the physics-based flat downwash model immediately improves tracking errors compared to the Nominal baseline, particularly in the challenging close-proximity regime ($z = 0.2$m). Adding the learned residual further reduces both the mean 3D position error and the maximum vertical tracking error, with the largest gains appearing in z_{max}. We report quantitative results in Table 2, where the learned model consistently achieves the lowest or near-lowest tracking errors across both in-distribution and OOD separations. The ablation without the ROM shows that directly learning the residual can also improve closed-loop tracking, indicating that the FBL controller is relatively robust to approximation error in the learned model. However, incorporating the physics-structured prior in the learned model provides more reliable improvements at close separation, where aerodynamic coupling is strongest, and achieves the best overall 3D position tracking, which supports the prediction-error results in Table 1.

In addition to the FBL controller, we use the learned residual to adjust the desired acceleration and computed torque in an SE(3) geometric controller (Lee et al., 2010), with representative results reported in Table 3. The learned residual also improves SE(3) tracking performance over the Nominal and ROM baselines, indicating that the learned correction is not tied to the FBL controller and can be incorporated into alternative control architectures. In particular, the gains persist across both the in-distribution and OOD separations evaluated here, supporting the broader applicability of the learned residual model.

Finally, we evaluate our proposed controller against NMPC whose dynamics are augmented with the same learned residual model. Both controllers were tested head-to-head using the previous

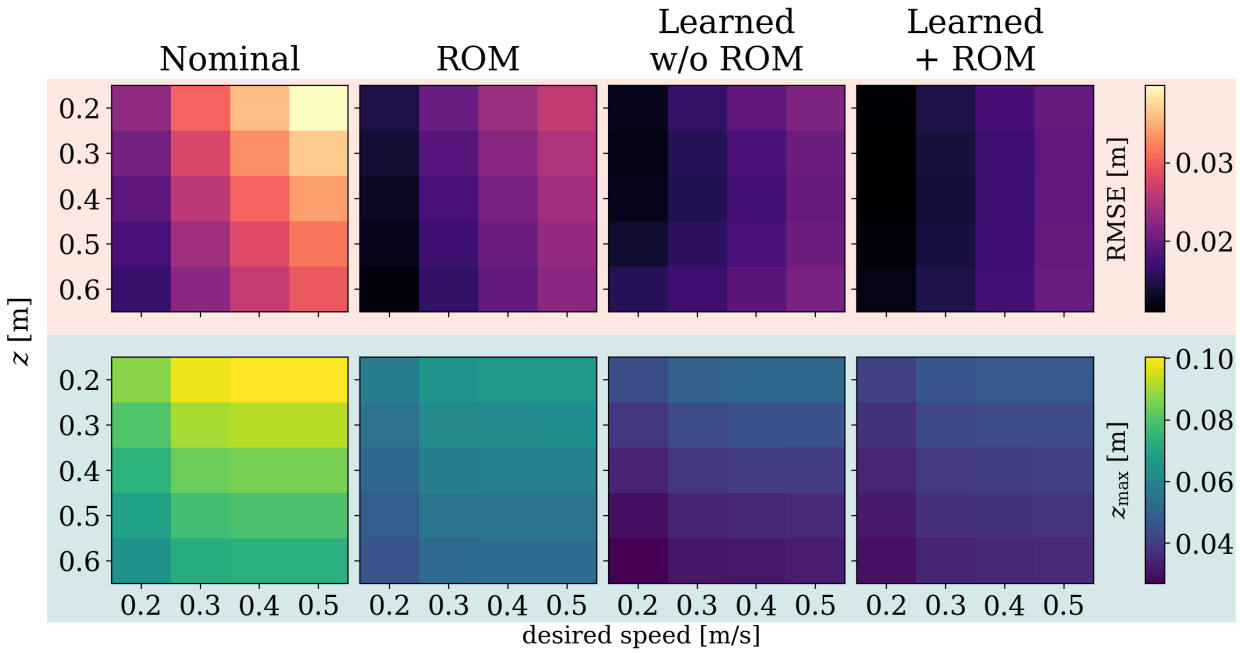

Figure 2: Closed-loop tracking performance (RMSE and z_{max}) of the FBL controller over varying reference speeds and vertical separations (z). The heatmaps illustrate the progressive reduction in tracking error when upgrading from the Nominal model (left) to the ROM (center left), learned w/o ROM (center right) and finally to the residual learned with ROM (right).

experimental setups, with results for mean 3D position error and maximum z error reported in Table 4. Our flatness-based controller achieves tracking performance on par with the NMPC. Moreover, while the NMPC was implemented in Acados (Verschueren et al., 2022), compiled into C code, and utilized real-time iteration, our JIT-compiled PyTorch-based controller still achieved a 20× reduction in solve time. This efficiency suggests broader applicability for computationally constrained hardware.

5.2 Physical Experiments

Setup: We use two Crazyflie 2.1 quadrotors equipped with thrust upgrade bundles (Bitcraze). Each quadrotor has a body length of 10 cm and weighs 36 g with the battery and Vicon markers. The Vicon motion capture system streams pose estimates to the base station (a laptop with an Intel i5 CPU) at 120 Hz. This base station computes control commands that are transmitted to the quadrotors at 400 Hz via Crazyradio PA. The software for simultaneous control of the two Crazyflies was implemented using the CrazyROS wrapper (Hönig and Ayanian, 2017).

We implemented the FBL and SE(3) controllers using the same software architecture employed in simulation. However, to accommodate the default firmware interfaces, the FBL controller was configured to output thrust and angular velocity commands rather than thrust and torque. Therefore, we compensate exclusively for residual forces and rely on the high-frequency PID controller onboard to handle the residual torques. For the ROM downwash model, we set the drag coefficient to $C_D = 0.236$, identified via the system identification procedure in Chee et al. (2024) to ensure reliable baseline performance. Residual forces are captured using a 2-layer neural network with 4

Regime	z [m]	Nominal	ROM	Learned w/o ROM	Learned + ROM
In-Dist.	0.3	0.030 / 0.090	0.020 / 0.061	**0.016** / 0.044	**0.016** / **0.042**
	0.4	0.028 / 0.083	0.019 / 0.057	**0.016** / **0.038**	**0.016** / 0.039
OOD	0.2	0.032 / 0.097	0.021 / 0.065	0.018 / 0.050	**0.016** / **0.046**
	0.5	0.026 / 0.077	0.018 / 0.054	0.017 / **0.034**	**0.016** / 0.036

Table 2: Sim. Tracking Errors for the FBL Controller (RMSE / z_{max}) [m].

Regime	z [m]	Nominal	ROM	Learned w/o ROM	Learned + ROM
In-Dist.	0.3	0.088 / 0.157	0.054 / 0.096	0.034 / 0.058	**0.031** / **0.055**
	0.4	0.079 / 0.140	0.048 / 0.084	**0.026** / **0.044**	**0.026** / **0.044**
OOD	0.2	0.094 / 0.176	0.061 / 0.108	0.042 / 0.076	**0.038** / **0.067**
	0.5	0.070 / 0.124	0.042 / 0.074	**0.021** / **0.031**	0.022 / 0.035

Table 3: Sim. Tracking Errors for the SE(3) Controller (RMSE / z_{max}) [m].

neurons. The training dataset $\mathcal{D}$ spans 28 seconds (6,720 points) collected from trajectory segments where the quadrotors fly in *static* and *stacked* formations at a 0.4 m separation and 0.4 m/s velocity under an FBL controller unaugmented by learned residual dynamics.

Results: We study the closed-loop performance of both FBL and SE(3) controllers when augmented with residual models of varying fidelities, yielding six controller configurations: Learned-FBL, ROM-FBL, Nominal-FBL, and their respective SE(3) counterparts. We consider two commanded separations: 0.4m (training distribution) and 0.3m (out-of-distribution). A total of 60 experimental runs were conducted, with each test case repeated five times. As shown in Figure 3, our proposed Learned-FBL controller achieved the best overall performance. The ROM-FBL controller outperformed the Nominal-FBL baseline by an average of 24% in RMSE and 30% in maximum vertical error (z_{max}). By leveraging the combined physics-informed and learned framework, the Learned-FBL controller achieved additional improvements of 29% in RMSE and 43% in z_{max} relative to ROM-FBL. A similar trend was observed when integrating the residual models as feedforward terms into the SE(3) controllers: the Learned-SE(3) and ROM-SE(3) configurations reduced RMSE by 43.7% and 34%, respectively, compared to the nominal baseline. Notably, these performance gains were realized without any additional data collection or retraining of the neural network for the SE(3) variants. The superior performance of controllers utilizing high-fidelity models over their lower-fidelity counterparts indicates that our physics-informed parameterization successfully captures the complex aerodynamic interactions inherent in tight formation flight.

Overall, our Learned-FBL controller achieved an average RMSE of 5 cm and a z_{max} of 10 cm. These results match the reported tracking performance of KNODE-DW MPC (Chee et al., 2024), a state-of-the-art NMPC framework that also utilizes learned residual models. Crucially, our flatness-based approach achieves this state-of-the-art accuracy while avoiding the substantial computational overhead of real-time NMPC optimization. Finally, to demonstrate the agility unlocked by our computationally efficient method, we challenge our proposed controller to merge from a 1.1 m stacked formation into a tight 0.1 m vertical separation to fly through a 0.4 m high window (Figure 1).

z [m]	Ctrl.	Error (RMSE/z_{max})	Avg. Solve Time[ms]
0.3	Ours	**0.016** / 0.042	**0.99**
	NMPC	0.020 / **0.032**	27.6
0.5	Ours	**0.016** / 0.036	**1.3**
	NMPC	0.018 / **0.019**	26.2

Table 4: Benchmark against NMPC (Chee et al., 2024)

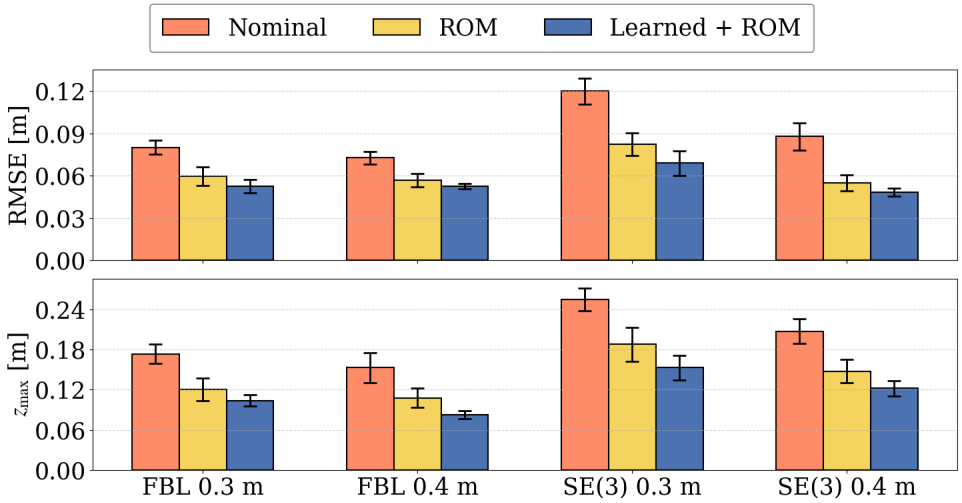

Figure 3: **Experiment statistics:** Statistics of the proposed tracking controller and the baselines over five runs. The RMSEs are shown in the top subplot, and the bottom subplot illustrates the maximum vertical separation error (z_{max}). The mean and standard deviation of the runs are depicted using the bars and error bars. The text and the value in the x-axis labels indicate the controller used and the commanded vertical separation.

6 Conclusion

We introduced a computationally efficient framework for parameterizing and learning residual dynamics by leveraging differential flatness and physics-based ROM. We presented a flatness-based controller that utilizes these learned residuals to achieve tracking performance on par with NMPC baselines, while requiring an order of magnitude less computation. Through both simulation and hardware experiments, we demonstrated that our method effectively handles complex multi-quadrotor proximity flight scenarios with significant performance gains over baseline controllers. Future work includes relaxing the structural assumptions of the flatness-preserving residuals and scaling up the experiments.

References

Kong Yao Chee, Pei-An Hsieh, George J Pappas, and M Ani Hsieh. Flying quadrotors in tight formations using learning-based model predictive control. *arXiv preprint arXiv:2410.09727*, 2024.

Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pages 9784–9790. IEEE, 2019.

Guanya Shi, Wolfgang Hönig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, 2021.

Alexander Spitzer and Nathan Michael. Feedback linearization for quadrotors with a learned acceleration error model. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6042–6048. IEEE, 2021.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

Kong Yao Chee, Tom Z Jiahao, and M Ani Hsieh. Knode-mpc: A knowledge-based data-driven predictive control framework for aerial robots. *IEEE Robotics and Automation Letters*, 7(2): 2819–2826, 2022.

Pei-An Hsieh, Kong Yao Chee, and M Ani Hsieh. Online adaptation for flying quadrotors in tight formations. *arXiv preprint arXiv:2506.17488*, 2025.

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

Fengjun Yang, Jake Welde, and Nikolai Matni. Learning flatness-preserving residuals for pure-feedback systems. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pages 3035–3042. IEEE, 2025a.

Cédric Join, Emmanuel Delaleau, and Michel Fliess. Flatness-based control revisited: The heol setting. *Comptes Rendus. Mathématique*, 362(G12):1693–1706, 2024.

Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.

Fengjun Yang, Jake Welde, and Nikolai Matni. Scalable distributed nonlinear control under flatness-preserving coupling. *arXiv preprint arXiv:2512.02138*, 2025b.

Eli Brookner. *Tracking and Kalman Filtering Made Easy*. Wiley New York, 1998.

Leonard Bauersfeld, Koen Muller, Dominic Ziegler, Filippo Coletti, and Davide Scaramuzza. Robotics meets fluid dynamics: A characterization of the induced airflow below a quadrotor as a turbulent jet. *IEEE Robotics and Automation Letters*, 10(2):1241–1248, 2024.

Karan P Jain, Trey Fortmuller, Jaeseung Byun, Simo A Mäkiharju, and Mark W Mueller. Modeling of aerodynamic disturbances for proximity flight of multirotors. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1261–1269. IEEE, 2019.

Tom Z Jiahao, M Ani Hsieh, and Eric Forgoston. Knowledge-based learning of nonlinear dynamics and chaos. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(11), 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.

Bitcraze. Crazyflie 2.1. URL https://www.bitcraze.io/products/crazyflie-2-1/. Accessed: 2026-02-25.

Wolfgang Hönig and Nora Ayanian. Flying multiple uavs using ros. In *Robot Operating System (ROS) The Complete Reference (Volume 2)*, pages 83–118. Springer, 2017.

Anoop Kiran, Narek Harutyunyan, Nora Ayanian, and Kenneth Breuer. Downwash dynamics: Impact of quadrotor separation on forces, moments, and velocities for dense formation flight. In *AIAA AVIATION FORUM AND ASCEND 2024*, page 4145, 2024.

A Details on the Physics Prior

Approximating the downwash with turbulent jet theory (Bauersfeld et al., 2024; Kiran et al., 2024) yields induced airflow velocity:

$$V\left(\Delta z, \Delta r\right) = \frac{U_H \frac{Bd}{\Delta\tilde{z}-z_0}}{\left(1+\left(\sqrt{2}-1\right)\left(\frac{\Delta\tilde{r}}{S(\Delta\tilde{z}-z_0)}\right)^2\right)^2}\mathbf{e}_z, \tag{24}$$

where $\Delta\tilde{z} := \Delta z/\lambda$ and $\Delta\tilde{r} := \Delta r/\lambda$ denote the vertical and horizontal separation normalized over the half-body length λ of the quadrotor. The constants B_d, S, U_H, and z_0 characterize the turbulent jet and are obtained through empirical airflow velocity measurements (Bauersfeld et al., 2024). By modeling the *bottom* quadrotor as a disk, the downwash-induced forces and torques are derived by integrating the drag equations over the disk's surface area A_b as follows (Jain et al., 2019):

$$\hat{f}_d^a\left(\mathbf{p}^1, \mathbf{p}^2\right) = R_{\mathcal{F}}^{\mathcal{W}}\left(\int_{A_b} C_D\,\zeta\,dA\right)\mathbf{e}_z, \tag{25a}$$

$$\hat{f}_d^\tau\left(\mathbf{p}^1, \mathbf{p}^2\right) = R_{\mathcal{F}}^{\mathcal{W}}\int_{A_b}\left(\mathbf{p}_{A_b} - \mathbf{p}'\right)\times\mathbf{e}_z\,C_D\,\zeta\,dA, \tag{25b}$$

where $\mathbf{p}' = R_{\mathcal{W}}^{\mathcal{F}}\Delta\mathbf{p}$, and $\mathbf{p}_{A_b} \in \mathbb{R}^3$ denotes the position of the points on A_b in $\mathcal{F}$. The constant $C_D \in \mathbb{R}$ denotes the drag coefficient, and $\zeta := \frac{1}{2}\rho||V(\Delta z, \Delta r)||_2^2 \in \mathbb{R}$ represents the dynamic pressure.