drones

MDPI

Article

Computationally Efficient Learning-Augmented MPC for Quadrotor Trajectory Tracking via Control-Sensitivity Residual Injection

Jiahao Wei [1], Muyang Niu [1], Ziyi Wang [1], Yijia Huo [1], Yi Zhang [1], Tongyang Shi [2] and Pengyun Chen [1,*]

[1] School of Aerospace Engineering, North University of China, Taiyuan 030051, China; s202418025@st.nuc.edu.cn (J.W.); s202418024@st.nuc.edu.cn (M.N.); s202318027@st.nuc.edu.cn (Z.W.); s202418007@st.nuc.edu.cn (Y.H.); zyiii0720@sina.com (Y.Z.)

[2] School of Electrical and Control Engineering, North University of China, Taiyuan 030051, China; sty202502@163.com

* Correspondence: chenpengyun@nuc.edu.cn

Highlights

What are the main findings?

- CSRI-MPC reduces mean three-dimensional position RMSE by 12.69–17.65% compared with zero-order residual compensation across three test trajectories with and without wind.
- CSRI-MPC achieves a mean control computation time of 4.99 ms on Jetson Xavier NX, compared with 27.84 ms for full embedding of the same CSRM.

What are the implications of the main findings?

- Analytical control sensitivity can be incorporated into MPC without computing the residual control Jacobian through online automatic differentiation or embedding a full neural network in the solver.
- Structured residual injection provides an effective balance between trajectory-tracking accuracy and online computational cost for quadrotor control.

Abstract

This paper investigates computationally efficient quadrotor trajectory tracking under nominal model mismatch. A control-sensitivity residual model (CSRM) and a control-sensitivity residual injection MPC method (CSRI-MPC) are proposed. CSRM separates the learned residual into a nonlinear state component and a linear control component, allowing the control sensitivity to be obtained analytically after training. In each control cycle, the residual bias and control sensitivity are passed to the optimal control problem as fixed parameters, avoiding online automatic differentiation and full neural network embedding. A local prediction-error bound characterizes the approximation caused by freezing the state-dependent residual. Simulation experiments on three trajectories with and without wind show that CSRI-MPC improves tracking accuracy over zero-order residual compensation. Timing results show that its mean control computation time is lower than that of full embedding of the same CSRM. The method achieves an effective balance between closed-loop performance and computational efficiency under the tested conditions.

Keywords: control sensitivity; learning-augmented control; nonlinear model predictive control; quadrotor; residual learning; trajectory tracking

Check for updates

Academic Editor: Zhi Feng

Received: 25 October 2026
Revised: 19 November 2026
Accepted: 24 November 2026
Published: 26 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Model predictive control (MPC) jointly accounts for system dynamics, control input constraints, and trajectory tracking objectives over a finite prediction horizon, and serves as an important method for trajectory tracking and agile flight control of quadrotor unmanned aerial vehicles. For quadrotor systems, MPC typically relies on a nominal rigid-body dynamics model to predict future states and obtains control inputs by solving an optimal control problem (OCP) online. However, the system behavior in real flight or high-fidelity simulation is often difficult to describe fully with a simplified nominal model. As flight speed, acceleration, and trajectory curvature increase, unmodeled effects such as aerodynamic drag, rotor interaction, motor dynamics, gyroscopic coupling, and parameter deviations gradually become the dominant error sources, which degrades prediction accuracy and closed-loop tracking performance [1–3].

To compensate for nominal model errors, learning-augmented MPC typically learns residual dynamics from flight data and uses the residual information to correct the prediction model. Gaussian process (GP) residual models have been adopted to improve the modeling of aerodynamic errors in high-speed quadrotor flight [1,3]. Neural networks, neural ordinary differential equations, and physics-inspired hybrid dynamics models have also been used to enhance the expressiveness of quadrotor dynamics models [2,4,5]. These studies show that data-driven residual models can capture dynamics errors that are difficult to cover with analytical models. For real-time MPC, however, the prediction accuracy of the residual model is not the only concern; how the learned model enters the online optimization process and whether it introduces additional forward inference, derivative computation, and solver overhead also determine whether the model is suitable for high-frequency control deployment.

Representative GP-MPC studies typically introduce the learned residual into the prediction model as a correction to the nominal model [1,3]. When only residual function values or residual biases are updated outside the solver, such zero-order injection keeps the computational cost low but does not explicitly provide the optimizer with local derivatives of the residual with respect to candidate control inputs. Other approaches use full-model embedding or externally evaluated local approximations, with different costs for model evaluation and sensitivity computation [6–8]. Real-time residual-augmented MPC therefore requires a trade-off between online computational cost and local control-direction information.

To address this problem, this paper proposes CSRM, a control-sensitivity residual model designed for real-time MPC deployment, and constructs CSRI-MPC based on it. CSRI-MPC injects the residual bias and the control-direction sensitivity at the current operating point into the OCP as fixed-dimension parameters, so that the solver can use local control-direction residual information without embedding a full neural network or performing online backpropagation in every control cycle. Rather than simply increasing the capacity of the residual network, this paper focuses on how residual information enters real-time MPC and how this choice affects closed-loop performance and the full control computation time.

The experimental evaluation is carried out in a unified AirSim/PX4-SITL software-in-the-loop environment [9,10], supplemented by hardware-in-the-loop computation tests. The experiments compare nominal nonlinear MPC (NMPC), GP-MPC, zero-order CSRM residual compensation, Jacobian injection based on automatic differentiation or finite differences, CSRI-MPC, and full embedding of CSRM or MLP in the solver symbolic graph. The comparison covers residual prediction error, full control computation time, and closed-loop trajectory tracking accuracy on three trajectories with and without wind, together with a sensitivity study of the MPC prediction horizon and control penalty.

https://doi.org/10.3390/drones10120890

The main contributions of this paper are threefold. First, it proposes CSRM, a control-sensitivity residual model for real-time MPC deployment, which preserves residual modeling capability while providing MPC with control-direction residual information that can be obtained at low cost. Second, it develops CSRI-MPC, a control-sensitivity residual injection MPC method that injects the zero-order residual bias and the control-direction sensitivity into the OCP as parameters, and thus avoids embedding the full neural network into the symbolic graph or computing the Jacobian by online automatic differentiation in every control cycle. A local prediction-error bound is derived for freezing the state-dependent residual. Third, it compares residual modeling and injection schemes through AirSim/PX4-SITL experiments and hardware-in-the-loop computation tests, including full embedding of the same CSRM and MLP-based finite-difference injection. Residual prediction error, full control computation time, and closed-loop tracking accuracy are evaluated to analyze the trade-off achieved by CSRI-MPC.

2. Related Work and Research Motivation

MPC for quadrotor unmanned aerial vehicles is typically based on nonlinear rigid-body dynamics models and handles control input constraints, state constraints, and trajectory tracking objectives through online optimization. With the development of real-time optimization solvers and onboard computing capability, model predictive control has been applied to high-speed flight and complex trajectory tracking [11–13]. In particular, Romero et al. combine online time-optimal replanning with model predictive contouring control, demonstrating flights above 60 km/h through moving gates and, in wind-disturbance experiments, operation under winds up to 68 km/h [11]. These methods explicitly exploit system dynamics and constraint information, but their performance still depends on the accuracy of the prediction model. For high-speed or high-acceleration flight, aerodynamic effects and rotor coupling affect closed-loop performance [1,2]. Actuator dynamics and parameter uncertainty also cause deviations between the nominal model and the real system [14]. Improving prediction model accuracy while preserving real-time capability is therefore one of the core problems in quadrotor MPC research.

Residual learning is an important direction for compensating nominal model errors. Torrente et al. introduce Gaussian process residual models into quadrotor MPC to model aerodynamic errors in high-speed flight [1], and Choo and Kayacan examine how GP residual models can be combined with MPC from a computational efficiency perspective [3]. Beyond GP, NeuroBEM combines blade element momentum theory with a learned model to improve the accuracy of complex aerodynamic force modeling [2]. KNODE-MPC combines a first-principles model with a knowledge-based neural ordinary differential equation and integrates the resulting hybrid dynamics model into MPC [4]. PI-TCN learns quadrotor dynamics with a physics-inspired temporal convolutional network and applies it to model predictive trajectory tracking [5]. More recently, PI-WAN combines a temporal convolutional network with a physics-informed loss and incorporates the learned quadrotor dynamics into MPC to improve tracking under previously unseen conditions [15]. Manzoor et al. integrate a physics-informed sparse control-affine model with adaptive tube MPC, linking learned residual uncertainty to robust constraint handling [16]. These studies show that learned models can capture errors that are difficult to cover with analytical dynamics models, but model expressiveness, generalization, and online computational cost still need to be considered together.

Within the more general learning-based MPC framework, safe learning and predictive safety filters focus mainly on constraint satisfaction in learning-based control [17,18]. Predictive stability filters address the closed-loop behavior of learned controllers from a stability perspective [19]. In parallel, online dynamics learning and deep ensemble methods

are used to improve model prediction capability and uncertainty representation [20,21]. Later studies also account for the influence of learned model uncertainty on predictive control performance [22]. Ning et al. learn a switched Koopman model through neural state lifting and formulate predictive control for constrained switched nonlinear systems subject to stochastic disturbances [23]. Wu et al. combine process data, mechanistic knowledge, and physical state constraints in a physics-informed recurrent neural network for predictive control of batch crystallization [24]. These studies provide an important foundation for learning-augmented control, but their focus is usually not the low-cost injection of residual models into real-time MPC solvers.

The way learned models enter real-time optimization solvers has also received attention. Real-time Neural-MPC targets the deployment of large-capacity neural network dynamics models in MPC and highlights the model evaluation and sensitivity computation issues that arise when complex learned dynamics are combined with fast real-time iteration [6]. L4CasADi provides an interface between PyTorch learned models and CasADi numerical optimization [7]. L4acados supports the integration of non-CasADi learned models, external sensitivities, and custom Jacobian approximations for acados [8]. These tools offer general interfaces for embedding learned models into numerical optimization, but when a full neural network or a black-box residual model directly participates in online optimization, forward propagation, derivative computation, and solver-internal computation can still increase the deployment cost of high-frequency MPC.

Studies on high-speed quadrotor flight, autonomous racing, and reinforcement-learning-based flight show that model fidelity, control latency, and policy execution capability become increasingly important when aerial robots approach their performance limits [25–27]. Research on time-optimal online replanning and learning-based minimum-time flight emphasizes the coupling among model accuracy, real-time optimization, and trajectory execution capability [11,28]. Related studies on robust time-optimal trajectory planning and waypoint-flight planning further demonstrate the importance of computationally efficient optimization for agile quadrotor flight [29,30]. The acados and CasADi frameworks provide software foundations for fast embedded optimal control and nonlinear optimization modeling, respectively [31,32]. In addition, learning-based MPC and neural-network-based MPC methods explore the integration of learned prediction models with online optimization from different perspectives [33,34]. These works mainly address trajectory planning, optimization modeling and solver tools, prediction model expressiveness, task performance, or the handling of safety constraints, whereas this paper focuses on how residual information enters the real-time MPC optimization process in a low-cost form.

This paper neither embeds a full neural network residual model directly into the OCP nor relies on computing the Jacobian of a black-box residual model with respect to the control input by automatic differentiation in every control cycle. Instead, it provides a low-cost parameterized injection form that lies between zero-order residual compensation and full neural network embedding. The goal is to supply the optimizer with richer local control-direction information than the zero-order residual bias while keeping the online computational cost low.

3. Methodology

3.1. CSRM-Enhanced Quadrotor Residual Dynamics Model

3.1.1. Nominal Dynamics and Residual Channels

The world frame is denoted by $\mathcal{W}$ and the body frame by $\mathcal{B}$. The full attitude vector is $\eta = [\phi, \theta, \psi]^\top$, where ϕ, θ, and ψ are the roll, pitch, and yaw angles, respectively. Let $e_3 = [0, 0, 1]^\top$. The rotation matrix $R(\eta)$ from the body frame to the world frame is

$$R(\eta) = R_z(\psi)R_y(\theta)R_x(\phi). \tag{1}$$

The quadrotor state is defined as

$$x = \begin{bmatrix} p^\top & v^\top & \eta^\top & \omega^\top \end{bmatrix}^\top \in \mathbb{R}^{12}, \tag{2}$$

where $p \in \mathbb{R}^3$ is the position in the world frame, $v \in \mathbb{R}^3$ is the linear velocity in the world frame, $\eta = [\phi, \theta, \psi]^\top$ contains the roll, pitch, and yaw Euler angles, and $\omega = [\omega_x, \omega_y, \omega_z]^\top$ is the angular velocity in the body frame. The control input $u = [u_0, u_1, u_2, u_3]^\top \in [0, 1]^4$ consists of four normalized motor signals. In the nominal model, the thrust of the i-th rotor is obtained from the normalized motor signal through a static thrust mapping, $T_i = h_T(u_i)$, $i = 0, 1, 2, 3$, where $h_T(\cdot)$ is the nominal static thrust mapping. In the subsequent MPC formulation, the four normalized motor signals u serve directly as the control variables. The nominal dynamics are given by

$$f_{\text{nom}}(x, u) = \begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{\eta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ \frac{T_\Sigma}{m} R(\eta)e_3 - ge_3 \\ W(\phi, \theta)\omega \\ J^{-1}\tau \end{bmatrix}, \tag{3}$$

where T_i is the thrust produced by the i-th rotor, $T_\Sigma = \sum_{i=0}^{3} T_i$ is the total thrust of the four rotors, and $J = \text{diag}(J_x, J_y, J_z) \in \mathbb{R}^{3\times3}$ is the nominal body-frame moment-of-inertia matrix. The term $R(\eta)e_3$ represents the direction of the body-frame thrust axis in the world frame, and $W(\phi, \theta)$ is the Euler angle kinematics matrix. The torque vector τ is given by the nominal torque allocation matrix:

$$\tau = M_\tau [T_0, T_1, T_2, T_3]^\top, \quad M_\tau = \begin{bmatrix} -d & d & d & -d \\ d & -d & d & -d \\ k_\tau & k_\tau & -k_\tau & -k_\tau \end{bmatrix}, \tag{4}$$

where M_τ is the nominal torque allocation matrix, d is the effective moment arm projected onto the body x and y axes in the X configuration, and k_τ is the yaw torque-to-thrust ratio (m). The signals u_0, u_1, u_2, and u_3 are arranged in the same order as the four normalized motor commands in the data logs and the MPC output, and the control feature matrix G introduced later uses the same motor channel ordering.

To retain the nominal position and Euler angle kinematics, a six-dimensional learned residual is introduced only in the velocity derivative and body-frame angular acceleration channels. The learned residual is written as

$$r_\Theta(x, u) \in \mathbb{R}^6, \tag{5}$$

where Θ denotes the trained CSRM parameters. The six output channels correspond to the nominal model errors in the three velocity derivatives and the three body-frame angular accelerations, that is,

$$r_\Theta(x, u) = [r_{\dot{v}_x}, r_{\dot{v}_y}, r_{\dot{v}_z}, r_{\dot{\omega}_x}, r_{\dot{\omega}_y}, r_{\dot{\omega}_z}]^\top. \tag{6}$$

The continuous-time dynamics with residual compensation can therefore be written as

$$\dot{x} = f_{\text{nom}}(x, u) + E r_\Theta(x, u), \tag{7}$$

where $E \in \mathbb{R}^{12\times 6}$ is the channel selection matrix that embeds the six-dimensional residual into the corresponding velocity derivative and body-frame angular acceleration channels of the full state derivative.

3.1.2. Structured Residual Model

Based on the six-dimensional residual channels defined above, CSRM represents the learned residual $r_\Theta(x, u)$ in a structured form. To balance residual expressiveness against real-time MPC deployment cost, the learned residual is decomposed into a state-dependent path and a control-feature path:

$$r_\Theta(x, u) = r_s(\phi_s(x)) + r_c(\phi_c(u)), \tag{8}$$

where $\phi_s(x)$ is the state feature constructed from the state, $r_s(\cdot)$ captures the state-dependent residual related to velocity, attitude, and angular velocity, $\phi_c(u)$ is the control feature constructed from the control input, and $r_c(\cdot)$ represents the linear residual component of the learned residual model that can be explained by the current control features.

This decomposition does not represent a strict physical decoupling of the true quadrotor residual; it is a structured approximation oriented toward real-time MPC deployment. The state path preserves nonlinear residual expressiveness, and the linear control path provides a control-direction sensitivity that can be obtained analytically after training. General state-control coupling is not represented explicitly, and the model provides no predictive uncertainty estimate.

3.1.3. Two-Path Network and Control-Direction Sensitivity

Figure 1 shows the two-path network implementation of CSRM, which includes the state-dependent path and the control-feature path. The state path of CSRM adopts a nonlinear multilayer perceptron (MLP) to fit the state-dependent component of the six-dimensional residual. The input of the state path is a 12-dimensional feature:

$$\begin{aligned}\phi_s(x) = [&v_x, v_y, v_z, \sin\phi, \cos\phi, \sin\theta, \cos\theta,\\ &\sin\psi, \cos\psi, \omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^{12},\end{aligned} \tag{9}$$

where v_x, v_y, and v_z are the velocity components in the world frame, ϕ, θ, and ψ are the roll, pitch, and yaw angles, respectively, and ω_x, ω_y, and ω_z are the body-frame angular velocity components. The state path is defined as

$$r_s(\phi_s(x)) = h_s(\phi_s(x)), \tag{10}$$

where $h_s(\cdot)$ is an MLP with two hidden layers, SiLU activation functions, and an output dimension of 6. The control path of CSRM adopts a linear layer without a bias term. The four normalized motor inputs are first mapped to the control features:

$$\phi_c(u) = Gu, \quad G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}, \tag{11}$$

where G is a fixed control feature matrix whose input channel order is consistent with $u = [u_0, u_1, u_2, u_3]^\top$, the nominal torque allocation matrix M_τ, and the order of the four normalized motor commands in the data logs. Its output components correspond to the collective, roll differential, pitch differential, and yaw differential control features constructed from the normalized motor signals. Here, G serves only as a fixed control

feature mapping and does not represent an exact mapping from normalized motor signals to true thrusts or torques. The control path is defined as

$$r_c(\phi_c(u)) = W_c\phi_c(u), W_c \in \mathbb{R}^{6\times 4}, \tag{12}$$

where W_c is the bias-free linear weight matrix obtained from training. The control path outputs the control-feature-related component of the six-dimensional residual. Since $\phi_c(u) = Gu$, the derivative of the control path with respect to the MPC control input is

$$J_c = \frac{\partial r_c(\phi_c(u))}{\partial u} = W_c G, \quad J_c \in \mathbb{R}^{6\times 4}. \tag{13}$$

The control-direction sensitivity J_c of CSRM can therefore be computed directly from the trained linear weights and the fixed control feature matrix. For the defined CSRM, J_c is the exact derivative of $r_\Theta(x, u)$ with respect to u and is independent of x and u. The residual value at the current operating point and J_c enter the OCP through the subsequent parameterized residual injection.

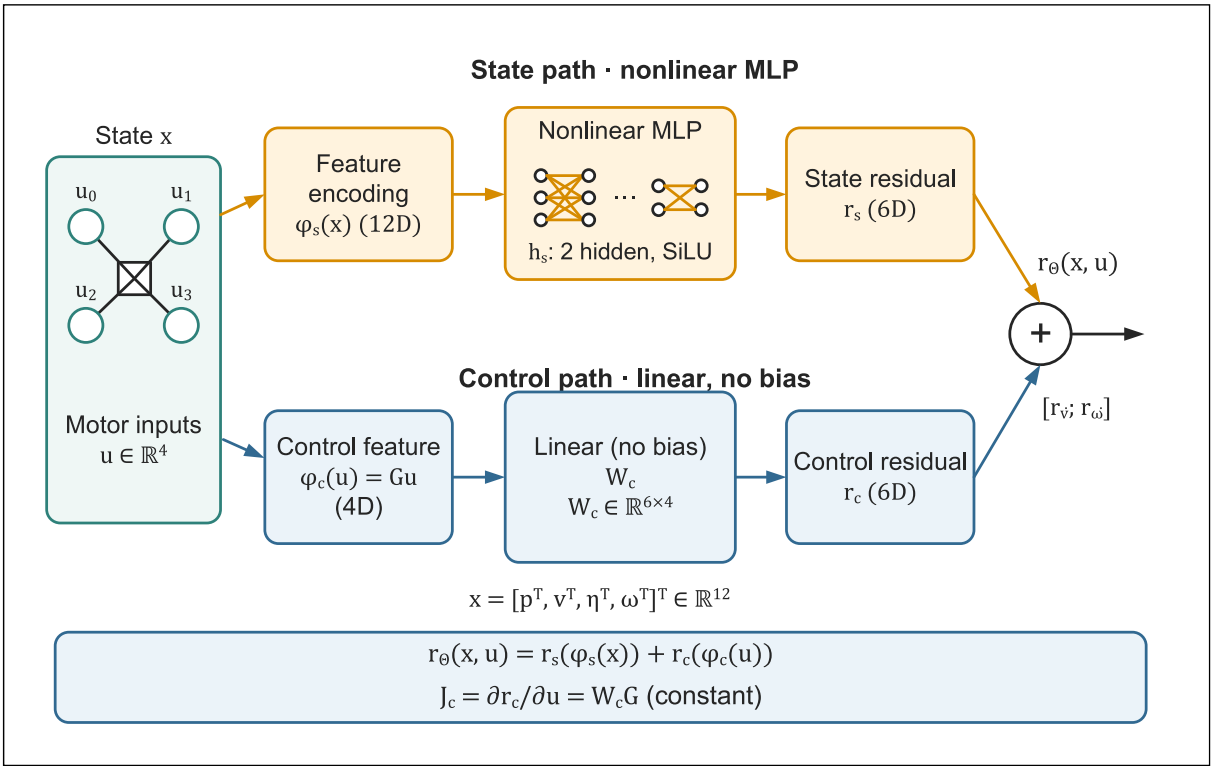


Figure 1. Two-path residual network architecture of CSRM.

3.2. CSRI-MPC

3.2.1. Parameterized Residual Injection

CSRI-MPC adopts a parameterized residual injection form, in which the residual bias evaluated at the current operating point, the control-direction sensitivity, and the baseline control input are passed to the solver as OCP parameters.

At the k-th control cycle, let x_k denote the current measured state and let $\overline{u}_k$ denote the second input in the previous cycle's optimal control sequence. This input is used to compute the zero-order residual bias at the current operating point and remains fixed within the current control cycle. The residual bias is first computed as

$$b_k = r_\Theta(x_k, \overline{u}_k), \quad b_k \in \mathbb{R}^6. \tag{14}$$

The sensitivity of the residual with respect to the MPC control input is the control-direction sensitivity $J_c = W_c G$ obtained from the CSRM control path in (13). The residual approximation injected into the OCP in the current control cycle is therefore written as

$$\hat{r}_k(u) = b_k + J_c(u - \overline{u}_k), \quad \hat{r}_k(u) \in \mathbb{R}^6, \tag{15}$$

where b_k, J_c, and $\overline{u}_k$ are passed to the solver as parameters. The augmented continuous-time dynamics are

$$\dot{x} = f_{\text{nom}}(x, u) + E\hat{r}_k(u), \quad E \in \mathbb{R}^{12\times 6}. \tag{16}$$

The above injection form fixes the state-dependent residual component at the current measured state x_k and, through the linear control path, preserves the control-related residual variation caused by changes in the candidate control input. The CSRM structure yields

$$\begin{aligned} \hat{r}_k(u) &= r_s(\phi_s(x_k)) + W_c G u, \\ r_\Theta(x, u) - \hat{r}_k(u) &= r_s(\phi_s(x)) - r_s(\phi_s(x_k)). \end{aligned}$$

Thus, relative to the same full CSRM, the injection error arises solely from freezing the state-dependent residual.

Figure 2 summarizes the parameterized residual injection and the closed-loop solution procedure of CSRI-MPC. In the offline stage, flight data are used to train CSRM, and the control-direction sensitivity $J_c = W_c G$ is obtained from the trained linear control path. In the online stage, the controller computes $b_k = r_\Theta(x_k, \overline{u}_k)$ from x_k and $\overline{u}_k$, and then writes b_k, J_c, and $\overline{u}_k$ into the OCP as fixed-dimension parameters. The solver performs the predictive optimization based on the parameterized residual dynamics and applies the first control input to the quadrotor system. In the next control cycle, the controller updates the residual bias, the baseline control input, and the initial state of the OCP according to the new measured state and the warm-start control sequence.

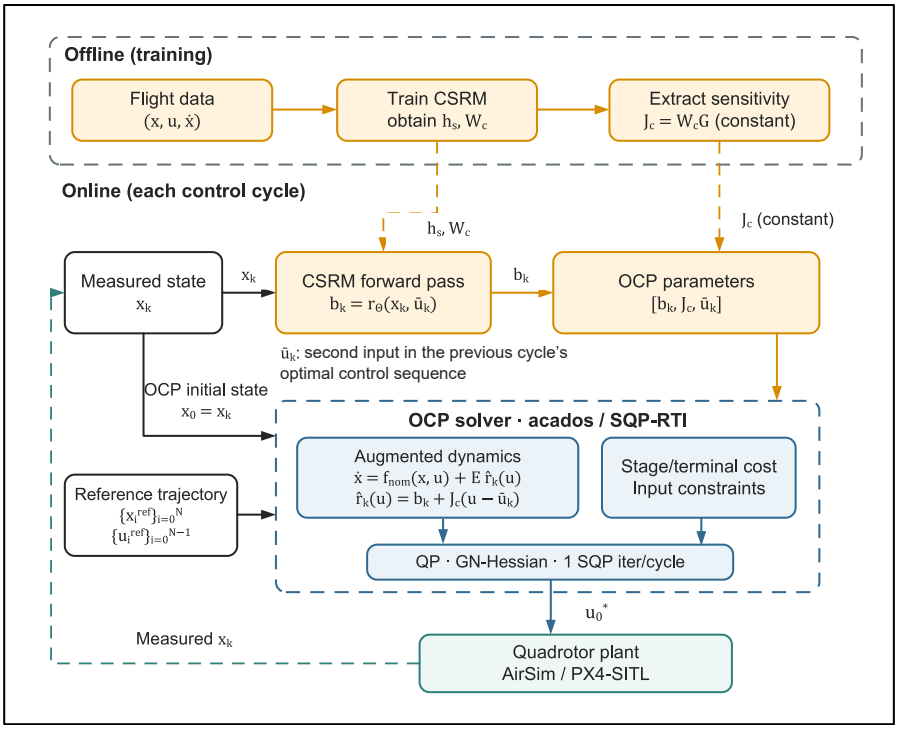


Figure 2. CSRI-MPC residual injection and control flow. u_0^* is the first optimized input applied to the quadrotor. Dashed orange transfers trained parameters, solid orange shows training and residual processing, blue shows solver flow, and dashed green shows state feedback.

3.2.2. Nonlinear Model Predictive Control

In every control cycle, CSRI-MPC performs prediction with the currently injected parameterized residual dynamics, that is, the augmented dynamics in (16) with the residual approximation $\widehat{r}_k(u)$ in (15). Fourth-order Runge–Kutta (RK4) integration yields the discrete-time map

$$x_{i+1} = F_k(x_i, u_i; b_k, J_c, \overline{u}_k), \tag{17}$$

where the subscript k indicates that the discrete dynamics depend on the parameters b_k, J_c, and $\overline{u}_k$ injected in the current control cycle. Within one control cycle, b_k, J_c, and $\overline{u}_k$ remain fixed over the prediction horizon, and in the next control cycle, the initial state of the OCP, the residual bias b_k, and the baseline control input $\overline{u}_k$ are updated according to the new measured state and the warm-start control sequence. J_c is the control-direction sensitivity parameter in the continuous-time parameterized residual dynamics, and the full control Jacobian of the discrete map is computed by the solver from the augmented dynamics and the numerical integration process.

In the CSRI-MPC framework, the following finite-horizon OCP is solved in every control cycle, where the optimization variables are the predicted state sequence $\mathbf{x} = \{x_i\}_{i=0}^{N}$ and the control sequence $\mathbf{u} = \{u_i\}_{i=0}^{N-1}$:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i, x_i^{\mathrm{ref}}, u_i^{\mathrm{ref}}) + \ell_f(x_N, x_N^{\mathrm{ref}}) \\ \text{s.t.} \quad & x_0 = x_k, \\ & x_{i+1} = F_k(x_i, u_i; b_k, J_c, \overline{u}_k), i = 0, \ldots, N-1, \\ & u_i \in \mathcal{U}, i = 0, \ldots, N-1, \end{aligned} \tag{18}$$

where $\ell_i(\cdot)$ and $\ell_f(\cdot)$ are the stage cost and the terminal cost, respectively, and $\mathcal{U} = [0,1]^4$ is the admissible set of normalized motor inputs.

The cost function takes a quadratic form. The stage cost is

$$\ell_i = \|x_i - x_i^{\mathrm{ref}}\|_Q^2 + \|u_i - u_i^{\mathrm{ref}}\|_R^2, \tag{19}$$

and the terminal cost is

$$\ell_f = \|x_N - x_N^{\mathrm{ref}}\|_{Q_f}^2, \tag{20}$$

where Q, R, and Q_f are weighting matrices. The OCP is implemented in acados and solved using SQP-RTI with one SQP iteration per control cycle.

3.3. Approximation Error and Operating Conditions

Consider a compact operating domain away from Euler angle singularities. Suppose that the state-residual branch is L_r-Lipschitz on the corresponding feature domain. The additional residual error introduced by freezing satisfies

$$\|r_\Theta(x,u) - \widehat{r}_k(u)\| \le L_r \|\phi_s(x) - \phi_s(x_k)\|.$$

Thus, the injection error depends on the change in state features, while the control-dependent contribution is retained exactly within the defined CSRM.

Let $x_F(\tau)$ and $x_I(\tau)$ denote the continuous-time predictions of the full CSRM and the injected model, respectively, starting from x_k under the same piecewise-constant input sequence. Assume that both predictions remain in the operating domain, that the full vector field is L_f-Lipschitz in the state uniformly over admissible inputs, with $L_f > 0$,

and that $\|d\phi_s(x_I)/d\tau\| \leq M_\phi$ almost everywhere. With $A = \|E\|L_r M_\phi$, the prediction difference satisfies

$$\|x_F(\tau) - x_I(\tau)\| \leq \frac{A}{L_f^2}\left(e^{L_f\tau} - 1 - L_f\tau\right) = O(\tau^2), \tau \to 0.$$

Indeed, setting $d(\tau) = \|x_F(\tau) - x_I(\tau)\|$ gives $d(\tau) \leq \int_0^\tau [L_f d(s) + As]ds$, with $d(0) = 0$; the integral inequality yields the bound. Here, τ is measured from the prediction origin of control cycle k.

This bound characterizes only the continuous-time prediction difference caused by freezing the state-dependent residual; it excludes learning and numerical integration errors. Faster changes in state features or longer prediction intervals increase the bound. Closed-loop stability analysis additionally involves the ideal feedback law, optimizer errors, and sampling conditions [35]; the prediction bound alone provides no guarantee of closed-loop stability or robustness.

4. Experimental Setup and Comparison Schemes

4.1. Experimental Platform

The experiments are conducted on an AirSim/PX4 simulation platform. The PC-based software-in-the-loop architecture is shown in Figure 3. AirSim provides the quadrotor dynamics, sensor, and environment simulation, PX4-SITL runs the flight control stack, and the ROS 2 control node receives state estimates and sends normalized motor control commands. The host machine is equipped with an Intel Core i7-14700HX CPU and an NVIDIA GeForce RTX 4060 GPU. PX4-SITL, the ROS 2 control node, and the MPC control program run in a VMware virtual machine, while AirSim runs on the host system. Hardware-in-the-loop tests use an NVIDIA Jetson Xavier NX (16 GB) and a Pixhawk Mini 6C flight controller. The MPC controller runs on the Jetson Xavier NX, PX4 runs on the Pixhawk Mini 6C, and AirSim runs on the PC. All MPC schemes use the same nominal physical parameters, as listed in Table 1.

Table 1. Nominal Dynamics Parameters of the Quadrotor.

Parameter	Value	Unit
m	1.0	kg
L	0.2275	m
d	0.1609	m
g	9.81	m/s^2
J_x	0.0075	$kg \cdot m^2$
J_y	0.0075	$kg \cdot m^2$
J_z	0.013	$kg \cdot m^2$
T_{max}	4.179	N
k_τ	0.01329	m

k_τ denotes the rotor reaction-torque-to-thrust ratio.

Model mismatch exists between the AirSim/PX4 closed-loop system and the nominal rigid-body model adopted in this paper, mainly including effects that the nominal model does not explicitly describe, such as inertia deviations, aerodynamic drag, gyroscopic coupling, and first-order motor response. This setup is used to evaluate how well the residual model and its injection scheme compensate for nominal model errors.

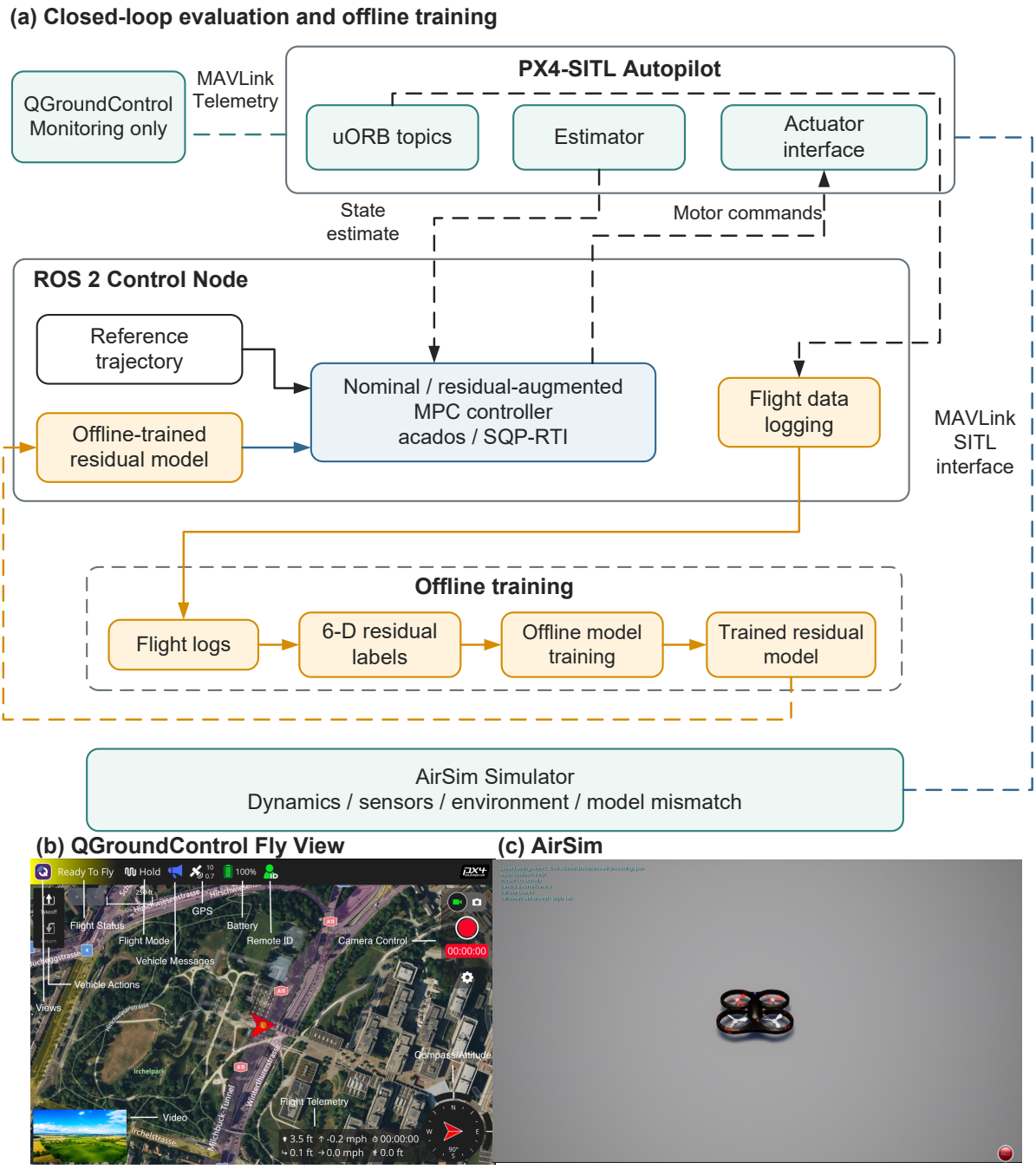


Figure 3. PC-based simulation platform. (**a**) Closed-loop evaluation and offline training workflow. The dashed orange arrow shows deployment of the trained residual model. (**b**) Illustrative QGroundControl Fly View from the QGroundControl User Guide (CC BY 4.0). (**c**) AirSim simulation scene. The MPC block is used for closed-loop evaluation. Training data are collected with the native PX4 controllers.

4.2. Data Collection and Network Training

The training data are collected in the AirSim/PX4-SITL environment using the native PX4 controllers during an irregular three-dimensional maneuver. The maneuver includes horizontal motion in multiple directions, ascent, descent, and speed variations. Figure 4 shows the horizontal projection of the recorded trajectory, colored by horizontal speed. Data are recorded for 500 s at 50 Hz, including the system state, the normalized motor control commands, and the state evolution information.

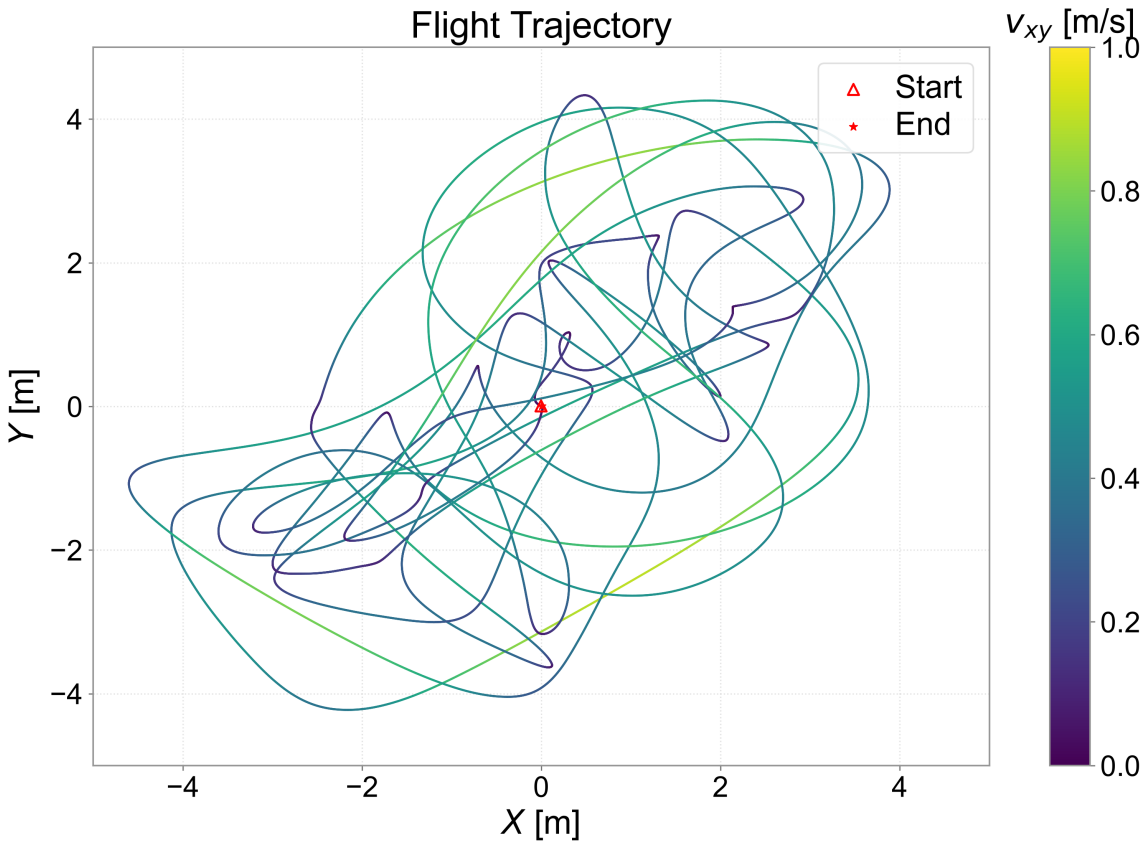


Figure 4. Horizontal projection of the training trajectory, colored by horizontal speed.

The training labels are defined as the difference between the observed state derivatives and the derivatives predicted by the nominal model at the same state-control point. The supervised channels are the six dynamics channels corresponding to the velocity derivatives and the body-frame angular accelerations, while the position channels and the Euler angle kinematics channels are given by the nominal model and do not participate in residual learning. The samples are randomly split into training and validation sets at a ratio of 80:20 using a fixed seed of 42. All models use the same split, and the validation set is used only for model selection and residual prediction error evaluation.

Table 2 lists the models and ablation configurations used in the residual prediction experiments. Nominal denotes the analytical nominal model without any learned residual correction; in the residual prediction error evaluation, its additional residual output is set to a six-dimensional zero vector. This zero vector only indicates that no learned residual correction is applied, and the full nominal dynamics are still given by f_{nom}. MLP-compact and MLP-S both take the state x and the control input u as inputs, concatenate them, and feed the result into a two-hidden-layer MLP that outputs the six-dimensional dynamics residual; MLP-compact is a compact black-box MLP baseline, and MLP-S is a higher-capacity MLP reference model. The residual module of CSRM w/o ctrl takes only the state as input and keeps the nonlinear state path driven by $\phi_s(x)$, which evaluates the role of the linear control path. The full CSRM takes the state and the control input as model inputs, where x and u enter the nonlinear state path and the bias-free linear control path through ϕ_s and ϕ_c, respectively. In Table 2, the MLP Widths column denotes the widths of the two hidden layers, and $4 \rightarrow 6$ in the Linear Path column denotes the bias-free linear mapping from the four-dimensional control features to the six-dimensional residual output. MLP-S serves only for residual prediction accuracy comparison and does not participate in the closed-loop trajectory tracking and computational efficiency experiments.

Table 2. Models and Ablation Configurations for Residual Prediction.

Model	Input	Output	MLP Widths	Linear Path	Params
Nominal	—	$\mathbf{0}_6$	—	—	0
MLP-compact	$[x, u]$	r_Θ	48-48	—	3462
CSRM w/o ctrl	$[x]$	r_Θ	48-48	—	3270
CSRM	$[x, u]$	r_Θ	48-48	$4 \to 6$	3294
MLP-S	$[x, u]$	r_Θ	64-64	—	5638

Except for Nominal, all neural network residual models follow the same training procedure. The optimizer is AdamW with an initial learning rate of 10^{-3} and a weight decay of 10^{-5}, and the learning rate is scheduled by ReduceLROnPlateau with a patience of 20 and a decay factor of 0.5. The batch size is 2048, the gradient clipping norm is 1.0, and the maximum number of training epochs is 1200; training stops early if the validation loss does not improve for 100 consecutive epochs. The control path weights of CSRM are initialized to zero. The training loss is the weighted mean squared error (MSE) over the six supervised channels corresponding to the velocity derivatives and the body-frame angular accelerations. Each neural network residual model is trained with three random seeds (42, 123, and 456), and the mean and standard deviation of the weighted MSE on the same validation set are reported.

4.3. Comparison Schemes and MPC Configuration

Table 3 presents the control schemes used in the closed-loop trajectory tracking and computational efficiency experiments. The nominal NMPC, zero-order CSRM-MPC, and CSRI-MPC are denoted by C1, C2, and C3, respectively. The remaining schemes are GP-MPC, AD-inject (Backprop-inject), MLP-in-solver (NN-in-solver), CSRM-in-solver, and FD-inject. The closed-loop tracking and NX computational efficiency experiments compare all eight schemes, while the PC timing comparison includes C1, C2, C3, AD-inject, and MLP-in-solver. Except for the learned residual model, the residual injection scheme, and the sensitivity evaluation method, all schemes use the same control frequency, prediction horizon, discretization step, cost function weights, constraints, and solver configuration in the baseline comparisons.

All MPC controllers are implemented with acados [31] and configured for a target control frequency of 50 Hz. The OCP is solved in the sequential quadratic programming real-time iteration (SQP-RTI) mode, with one SQP iteration per control cycle, and the QP subproblems use a Gauss–Newton Hessian approximation. The continuous-time dynamics are discretized by the fourth-order Runge–Kutta method. The baseline prediction horizon is $N = 5$ with a discretization step of $\Delta t = 0.02$ s, corresponding to a 0.10 s prediction window.

The baseline weights follow the state ordering $x = [p^\top, v^\top, \eta^\top, \omega^\top]^\top$ and the input ordering $u = [u_0, u_1, u_2, u_3]^\top$:

$$Q = \mathrm{diag}(80, 80, 800, 5, 5, 600, 150, 150, 20, 15, 15, 3),$$
$$R = R_0 = 3I_4, \quad Q_f = 3Q.$$

C1 uses only the nominal dynamics without any learned residual compensation. GP-MPC consists of three mutually independent single-input single-output Gaussian processes, which predict the corresponding velocity derivative residuals from the world-frame linear velocity components v_x, v_y, and v_z, respectively. Each GP uses 20 inducing points selected at regular intervals within the corresponding training input range. During prediction, the GP posterior mean is updated at each node of the prediction horizon with the world-frame

linear velocity of the predicted state and added to the velocity derivative channels, while the body-frame angular acceleration channels still use the nominal model.

Table 3. Closed-Loop Controller Configurations.

Scheme	Residual Model	Dim.	OCP Residual	Residual Control Sensitivity
C1	None	—	None	—
GP-MPC	GP-20	3	GP mean	—
C2	CSRM	6	Fixed bias	—
C3	CSRM	6	Affine form	Closed form
AD-inject	MLP-compact	6	Affine form	Autograd
MLP-in-solver	MLP-compact	6	Embedded MLP	Symbolic AD
CSRM-in-solver	CSRM	6	Embedded CSRM	Symbolic AD
FD-inject	MLP-compact	6	Affine form	Forward difference

C2 uses the full CSRM to compute the six-dimensional zero-order residual bias at the current operating point but does not provide the optimizer with the control-direction sensitivity. C3 additionally injects the control-direction sensitivity obtained from the linear control path of CSRM. CSRM-in-solver embeds the full CSRM into the solver and evaluates its state and control paths at the predicted states and inputs, including the internal RK4 stages. C2, C3, and CSRM-in-solver share the same trained CSRM weights, allowing the effect of the injection strategy to be compared with the residual model fixed.

AD-inject, FD-inject, and MLP-in-solver share the same trained MLP-compact weights. AD-inject uses the same parameterized injection interface as C3 and computes the control Jacobian using autograd in PyTorch 2.2.0 (CUDA 11.8 build). FD-inject uses the same affine injection interface as AD-inject but approximates the control Jacobian by first-order forward differences, perturbing each normalized motor input with a fixed step $h = 10^{-3}$ while holding the state fixed. MLP-in-solver embeds MLP-compact into the CasADi/acados symbolic graph, with the network forward computation and its derivatives handled inside the solver.

C1, C2, and C3 form the residual injection ablation, corresponding to no residual compensation, zero-order residual compensation, and control-sensitivity residual injection. AD-inject and FD-inject isolate the sensitivity evaluation method with the MLP and injection interface fixed. GP-MPC and the MLP-based schemes differ from C3 in residual-model structure, so their comparisons with C3 are not treated as single-variable ablations.

4.4. Test Conditions and Evaluation Protocol

The eight control schemes are evaluated on a horizontal circle, a three-dimensional circle with periodic altitude variation, and a planar Gerono figure eight, each with and without wind. These reference trajectories are separate from the irregular training maneuver. Using horizontal coordinates relative to the trajectory center, the references are

$$p_{\text{circle}}(\theta) = [\rho\cos\theta, \rho\sin\theta, z_0]^\top,$$
$$p_{\text{3D}}(\theta) = [\rho\cos\theta, \rho\sin\theta, z_0 + (1\,\text{m})\sin(2\theta)]^\top,$$
$$p_{\text{eight}}(\theta) = [\rho\sin\theta, (\rho/2)\sin(2\theta), z_0]^\top.$$

Here, $\rho = 5$ m, $z_0 = 5$ m, and $\dot{\theta} = v/\rho$. The horizontal circle uses $v = 4$ m/s; the three-dimensional circle and figure eight use phase-scale parameters of $v = 2.5$ and 2 m/s, respectively. Only the horizontal circle has constant speed equal to v. The three-

dimensional reference varies in both altitude and spatial curvature, while the figure eight includes changes in turning direction and a central crossing.

For all three trajectories, the wind-disturbed tests use the same world-frame wind from $t = 0$ to the end of each run:

$$w(t) = \begin{bmatrix} 2 + 0.45\sin(0.6t) \\ 0.6 + 0.30\sin(0.4t + \pi/3) \\ 0.10\sin(0.3t + \pi/5) \end{bmatrix} \text{m/s}.$$

Here, t is in seconds. Aerodynamic drag is computed componentwise from the body-frame relative air velocity $R_b^\top (v - w)$, with linear and quadratic terms opposing the corresponding velocity component. The coefficient vectors are $[0.06, 0.06, 0.08]^\top$ N s/m and $[0.035, 0.035, 0.045]^\top$ N s^2/m^2, respectively; R_b rotates body-frame vectors into the world frame.

Tracking accuracy is measured by the root mean square error of the three-dimensional position:

$$\text{RMSE} = \sqrt{\frac{1}{N_s}\sum_{j=1}^{N_s}\left\|p_j - p_j^{\text{ref}}\right\|_2^2}. \tag{21}$$

Here, N_s is the number of evaluation samples, and p_j and p_j^{ref} are the actual and reference positions at sample j. Each method-condition combination is repeated five times. All methods use the same evaluation interval within a condition; the RMSE is computed separately for each run. The mean and standard deviation of the five run-level RMSE values are reported.

The sensitivity of C3 to the prediction horizon and control penalty is evaluated on the wind-free horizontal circle of radius 5 m at 4 m/s. The prediction-step counts $N = 3, 5, 10$ are tested at $R = R_0$, and the control weights $R = \alpha R_0$ with $\alpha = 0.5, 1, 2$ are tested at $N = 5$. The prediction step $\Delta t = 0.02$ s, Q, Q_f, residual-model weights, and other settings are fixed. Each configuration is run five times using the same RMSE evaluation rule as the baseline comparison.

5. Experimental Results and Analysis

5.1. Residual Prediction Performance

Table 4 presents the residual prediction errors of different residual models on the validation set. The error metric is the weighted MSE on the validation set (Val weighted MSE) over the six supervised channels, which include the velocity derivatives and the body-frame angular accelerations, while the position channels and the Euler angle kinematics channels do not participate in residual learning.

Table 4. Weighted Validation MSE of the Residual Models.

Model	Weighted Validation MSE
Nominal	7.5
MLP-compact	1.179 ± 0.071
CSRM w/o ctrl	1.162 ± 0.067
CSRM	1.141 ± 0.069
MLP-S	1.213 ± 0.056

Neural models: mean ± standard deviation over three training seeds (42, 123, and 456) on the same validation set. Nominal is deterministic and has no training-seed variation.

Table 4 shows that the validation weighted MSE of the Nominal model over the six evaluation channels is markedly higher than those of the learned residual models.

This result indicates that the training data contain residuals unexplained by the nominal dynamics and that these residuals can be effectively fitted by supervised learning models.

Among the compact models of similar size, the validation errors of MLP-compact, CSRM w/o ctrl, and CSRM are of the same order, and the full CSRM still attains the lowest mean validation error, although its margin over the other two compact models is relatively limited. This shows that the structured decomposition into a state path and a linear control path does not noticeably weaken the residual prediction capability, while keeping a modeling accuracy comparable to that of the joint black-box MLP.

The comparison between CSRM w/o ctrl and the full CSRM evaluates the role of the linear control path. With only the state path retained, the model still achieves a validation error close to that of the full CSRM, and adding the linear control path further reduces the mean validation error, which shows that the control path provides complementary information for the six-dimensional residual modeling. The significance of this control path is not limited to improving residual prediction accuracy; more importantly, it provides CSRI-MPC with an analytically available control-direction sensitivity. MLP-S has a larger network capacity, yet its validation error is not better than those of the compact models, indicating that under the current dataset and residual learning task, simply increasing the capacity of a standard MLP does not guarantee a lower validation error.

The additive state-control structure is a modeling approximation adopted to retain analytical control sensitivity and does not provide uncertainty quantification. With insufficient training, errors in the residual or its control sensitivity may lead to inappropriate compensation and degrade tracking performance.

The subsequent closed-loop experiments therefore use the same trained full CSRM for C2, C3, and CSRM-in-solver to compare the effects of zero-order residual compensation, control-direction sensitivity injection, and full-model embedding on closed-loop performance.

5.2. Computational Efficiency Analysis

Integrating a learned residual model into MPC affects both the computation outside the solver and the computation inside the solver. The full control computation time is therefore used as the efficiency metric. It comprises four stages: system processing, residual/Jacobian computation, parameter setup, and OCP solution. System processing includes the ROS 2 callback, state reading, reference generation, and motor command publishing. Residual computation includes network inference and control-sensitivity evaluation; setup includes writing the initial state, constraints, references, and parameters; OCP solution is the time for one SQP-RTI iteration.

Table 5 and Figure 5 compare the control computation times on the PC and the Jetson Xavier NX 16 GB/Pixhawk Mini 6C hardware-in-the-loop platform. The NX uses the same Ubuntu version and acados solver as the PC experiment. The target control frequency is 50 Hz, with a 20 ms computation budget, $N = 5$, and $\Delta t = 0.02$ s.

The PC results are mean wall-clock times over control cycles in steady operation. Each NX test follows the wind-free horizontal circle of radius 5 m at 4 m/s for ten laps. The four timing components and their total are averaged over recorded control cycles in a contiguous 7.854 s window centered in the ten-lap record.

On the PC, C2 increases the mean computation time from the 2.3 ms of C1 to 2.8 ms, mainly through residual computation and parameter setup. C3 also takes 2.8 ms because its control sensitivity is available from the trained linear path without per-cycle backpropagation. AD-inject takes 7.8 ms, including 5.1 ms for residual/Jacobian computation, while MLP-in-solver takes 17.0 ms, with most of the cost inside the solver.

Table 5. Mean Control Computation Time and Its Components (ms).

Platform	Scheme	System	Residual/ Jacobian	Setup	OCP	Total
PC	C1	0.5	—	1.5	0.3	2.3
PC	C2	0.5	0.1	1.8	0.4	2.8
PC	C3	0.5	0.1	1.8	0.4	2.8
PC	AD-inject	0.5	5.1	1.8	0.4	7.8
PC	MLP-in-solver	0.5	—	1.5	15.0	17.0
NX	C1	0.50	0.00	2.70	0.75	3.95
NX	GP-MPC	0.50	0.00	2.70	2.50	5.70
NX	C2	0.50	0.25	3.24	1.00	4.99
NX	C3	0.50	0.25	3.24	1.00	4.99
NX	AD-inject	0.50	12.75	3.24	1.00	17.49
NX	FD-inject	0.50	1.10	3.24	1.00	5.84
NX	CSRM-in-solver	0.50	0.00	2.70	24.64	27.84
NX	MLP-in-solver	0.50	0.00	2.70	37.50	40.70

PC: five-method measurements. NX: eight-method hardware-in-the-loop measurements. Evaluation of solver-embedded models is included in the OCP solution time.

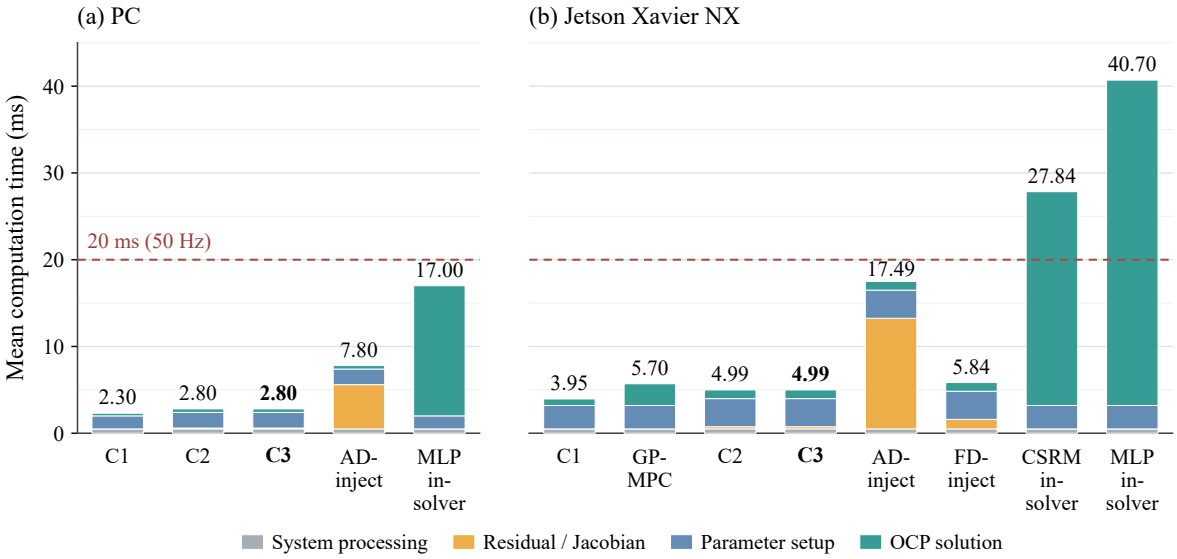


Figure 5. Mean control computation time on (**a**) the PC and (**b**) Jetson Xavier NX. Stacked bars show the timing components, labels indicate total mean times, and the dashed line marks the 20 ms cycle budget.

On the NX, C2 and C3 both take 4.99 ms at the reported precision. Full embedding of the same CSRM takes 27.84 ms, so C3 reduces the mean computation time by 82.08%. AD-inject and FD-inject take 17.49 and 5.84 ms, respectively; their residual/Jacobian stages take 12.75 and 1.10 ms. With the MLP and injection interface fixed, forward differences provide a lower-cost sensitivity approximation than autograd in this implementation. C3 is a further 0.85 ms faster than FD-inject on average, although this comparison also involves different residual-model structures.

In the NX hardware-in-the-loop tests, the mean computation time of C3 is below the 20 ms cycle budget corresponding to 50 Hz control, while those of CSRM-in-solver and MLP-in-solver, at 27.84 and 40.70 ms, exceed it. These results demonstrate the reduction in online computation cost achieved by CSRI-MPC under the tested configuration.

5.3. Closed-Loop Trajectory Tracking

Table 6 and Figure 6 present the closed-loop tracking results for the three trajectories with and without wind.

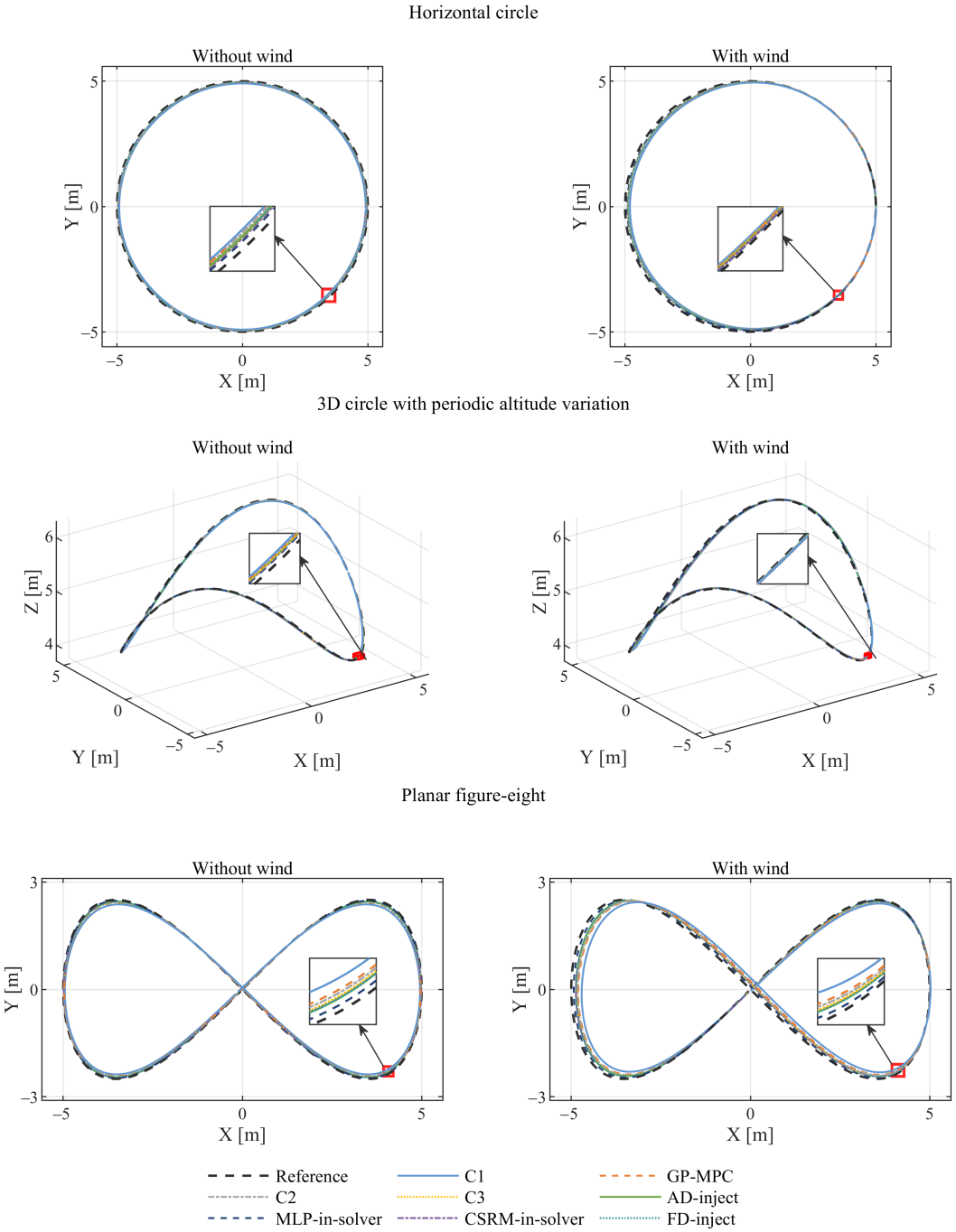


Figure 6. Closed-loop trajectory tracking with and without wind. From top to bottom: horizontal circle, 3D circle with periodic altitude variation, and planar figure eight. (**Left**): without wind. (**Right**): with wind. Insets show local magnifications.

All learned-residual schemes reduce the mean RMSE relative to C1 in the six tested conditions. C2 improves on C1 by injecting the zero-order CSRM residual, and C3 further reduces the mean RMSE relative to C2 by 12.69–17.65%. This shows that the control-direction sensitivity provides useful additional information for the optimizer on the horizontal, three-dimensional, and figure-eight references. The reported standard deviations are 2.1–2.3 mm for C3 and 3.0–3.2 mm for C2.

Table 6. 3D Position RMSE for the Three Trajectories with and without Wind (mm).

Condition	C1	GP-MPC	C2	C3	AD-Inject	MLP-in-Solver	CSRM-in-Solver	FD-Inject
Horizontal circle Without wind	175.176 ± 4.9	154.740 ± 4.1	144.549 ± 3.2	119.237 ± 2.3	122.257 ± 2.2	79.907 ± 1.2	110.042 ± 2.2	120.349 ± 2.3
Horizontal circle With wind	216.640 ± 4.8	191.299 ± 4.0	179.811 ± 3.1	148.075 ± 2.2	152.145 ± 2.1	99.769 ± 1.1	136.459 ± 2.1	149.896 ± 2.2
3D circle Without wind	116.839 ± 4.9	106.337 ± 4.1	99.220 ± 3.2	86.632 ± 2.3	82.399 ± 2.2	61.772 ± 1.2	79.100 ± 2.2	82.637 ± 2.3
3D circle With wind	189.512 ± 4.7	172.356 ± 3.9	161.808 ± 3.0	141.025 ± 2.1	134.518 ± 2.0	101.230 ± 1.0	128.613 ± 2.0	134.996 ± 2.1
Figure eight Without wind	155.321 ± 4.9	109.435 ± 4.1	94.055 ± 3.2	81.105 ± 2.3	70.982 ± 2.2	39.099 ± 1.2	74.432 ± 2.2	71.093 ± 2.3
Figure eight With wind	268.956 ± 4.8	191.231 ± 4.0	167.373 ± 3.1	143.554 ± 2.2	126.653 ± 2.1	70.083 ± 1.1	131.266 ± 2.1	127.054 ± 2.2

Results are mean ± standard deviation over five runs. C3 denotes CSRI-MPC.

With the CSRM weights fixed, CSRM-in-solver gives lower mean RMSE than C3 in all six conditions, with differences of 6.673–12.412 mm. This comparison shows the tracking cost of the reduced injection formulation relative to evaluating the full state-dependent residual throughout prediction. Its mean NX computation time is 27.84 ms, compared with 4.99 ms for C3. These results quantify the trade-off between tracking accuracy and computation cost.

AD-inject and FD-inject give close mean RMSE values, differing by 0.111–2.249 mm across the six conditions. Their ordering relative to C3 depends on the trajectory: C3 gives lower mean RMSE on the horizontal circle, while AD-inject and FD-inject give lower values on the other two references. MLP-in-solver has the lowest mean RMSE in all six conditions, at the highest measured NX computation cost. Offline weighted residual MSE and closed-loop position RMSE evaluate different quantities and need not yield the same model ranking.

Wind increases the mean RMSE of every method on each trajectory. For C3, the horizontal-circle RMSE increases from 119.237 to 148.075 mm, the 3D-circle RMSE from 86.632 to 141.025 mm, and the figure-eight RMSE from 81.105 to 143.554 mm. C3 retains its improvement over C2 in all three wind-disturbed tests. Figure 6 illustrates the corresponding tracking differences. These results show that CSRI-MPC maintains improved tracking relative to zero-order residual compensation under the tested three-dimensional trajectories and wind conditions.

5.4. MPC Parameter Sensitivity

Table 7 and Figure 7 show the tracking results of C3 under the five MPC parameter configurations.

Table 7. 3D Position RMSE of C3 under Different MPC Parameter Settings (mm).

Configuration	N	α	RMSE (mm)
Shorter horizon	3	1	134.000 ± 2.9
Baseline	5	1	119.237 ± 2.3
Longer horizon	10	1	114.000 ± 2.9
Lower control penalty	5	0.5	110.000 ± 2.9
Higher control penalty	5	2	128.000 ± 2.8

Relative to the baseline RMSE of 119.237 mm, shortening the prediction horizon increases the mean error to 134.000 mm, while extending it reduces the error to 114.000 mm. Reducing the control penalty gives the lowest mean error in this test, 110.000 mm, whereas increasing it gives 128.000 mm. Figure 7 shows the corresponding trajectories. The baseline has the smallest reported standard deviation, 2.3 mm, compared with 2.8–2.9 mm for the other settings. These results indicate sensitivity to both parameters; they do not establish a universally optimal configuration.

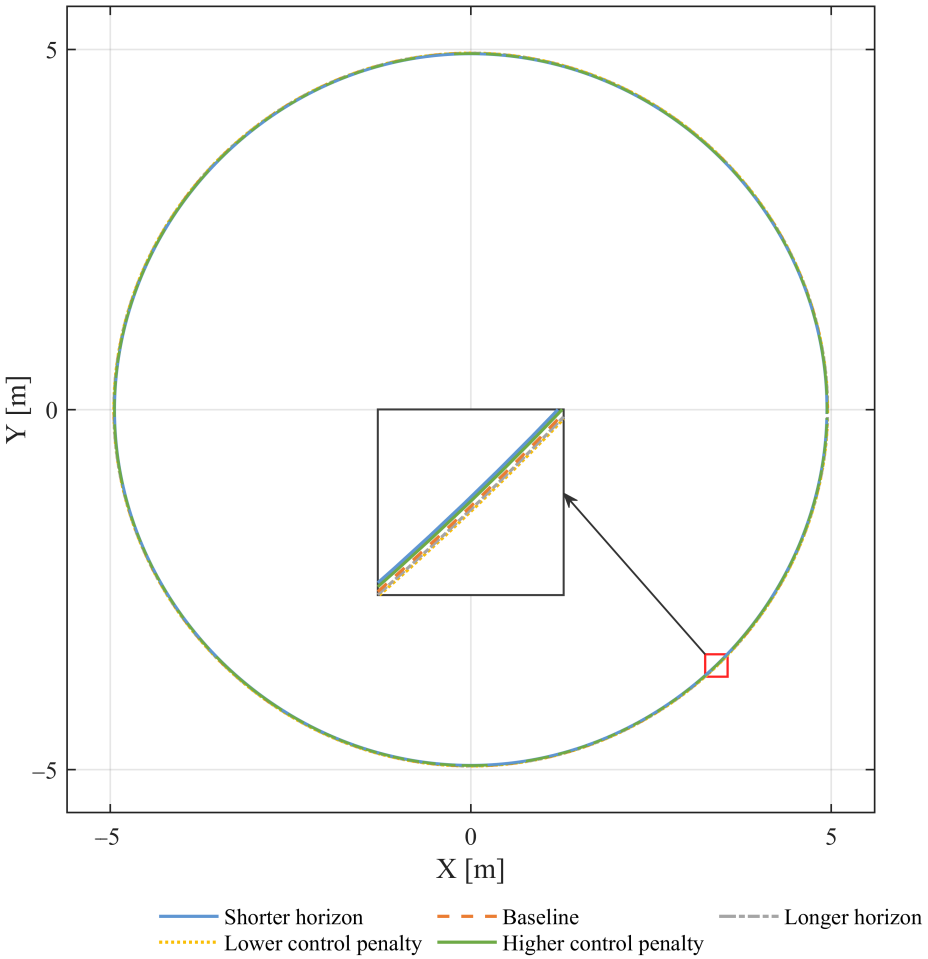


Figure 7. C3 trajectory tracking on the wind-free horizontal circle under different prediction horizons and control penalties. The inset shows a local magnification.

6. Conclusions

This paper proposes the control-sensitivity residual model CSRM and its parameterized injection method CSRI-MPC for compensating nominal dynamics model errors of quadrotors. CSRI-MPC injects the zero-order residual bias at the current operating point and the control-direction sensitivity provided by the linear control path into the OCP as parameters, which avoids embedding a full neural network into the solver symbolic graph and computing the Jacobian of a black-box residual model with respect to the control input in every cycle. Software-in-the-loop experiments in AirSim/PX4-SITL show that the mean 3D position RMSE of CSRI-MPC is lower than that of zero-order CSRM-MPC on all three trajectories with and without wind. Hardware-in-the-loop tests on the Jetson Xavier NX/Pixhawk Mini 6C platform show that its mean control computation time is close to that of the zero-order injection scheme and lower than that of full embedding of the same CSRM. The results show that the method achieves an effective trade-off between closed-loop tracking accuracy and online computational cost under the tested conditions.

Author Contributions: Conceptualization, J.W. and P.C.; methodology, J.W. and P.C.; software, J.W.; validation, J.W., M.N., Z.W., Y.H., Y.Z. and T.S.; formal analysis, J.W. and Y.Z.; investigation, J.W., M.N., Z.W., Y.H., Y.Z. and T.S.; data curation, J.W., M.N. and Z.W.; writing—original draft preparation, J.W.; writing—review and editing, J.W., M.N., Z.W., Y.H., Y.Z., T.S. and P.C.; visualization, J.W. and Y.H.; supervision, P.C.; project administration, P.C.; and funding acquisition, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Research and Development Program of Shanxi Province, grant number 202202020101001, and the National Natural Science Foundation of China, grant numbers 51909245 and 62003314.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (OpenAI, GPT-6.1 Sol) to assist with English translation and improve language and readability. The authors reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Torrente, G.; Kaufmann, E.; Föhn, P.; Scaramuzza, D. Data-Driven MPC for Quadrotors. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3769–3776. https://doi.org/10.1109/lra.2021.3061307.
2. Bauersfeld, L.; Kaufmann, E.; Foehn, P.; Sun, S.H.; Scaramuzza, D. NeuroBEM: Hybrid Aerodynamic Quadrotor Model. *arXiv* **2021**, arXiv:2106.08015. https://doi.org/10.15607/RSS.2021.XVII.042.
3. Choo, W.; Kayacan, E. Computationally Efficient Data-Driven MPC for Agile Quadrotor Flight. In Proceedings of the American Control Conference (ACC), San Diego, CA, USA, 31 May–2 June 2023; pp. 2627–2632. https://doi.org/10.23919/acc55779.2023.10156053.
4. Chee, K.Y.; Jiahao, T.Z.; Hsieh, M.A. KNODE-MPC: A Knowledge-Based Data-Driven Predictive Control Framework for Aerial Robots. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2819–2826. https://doi.org/10.1109/lra.2022.3144787.
5. Saviolo, A.; Li, G.R.; Loianno, G. Physics-Inspired Temporal Learning of Quadrotor Dynamics for Accurate Model Predictive Trajectory Tracking. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10256–10263. https://doi.org/10.1109/lra.2022.3192609.
6. Salzmann, T.; Kaufmann, E.; Arrizabalaga, J.; Pavone, M.; Scaramuzza, D.; Ryll, M. Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2397–2404. https://doi.org/10.1109/lra.2023.3246839.
7. Salzmann, T.; Arrizabalaga, J.; Andersson, J.; Pavone, M.; Ryll, M. Learning for CasADi: Data-driven Models in Numerical Optimization. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*; PMLR: Cambridge, MA, USA, 2024; Volume 242, pp. 541–553.
8. Lahr, A.; Näf, J.; Wabersich, K.P.; Frey, J.; Siehl, P.; Carron, A.; Diehl, M.; Zeilinger, M.N. L4acados: Learning-Based Models for acados, Applied to Gaussian Process-Based Predictive Control. *IEEE Trans. Control Syst. Technol.* **2026**, *34*, 1091–1105. https://doi.org/10.1109/tcst.2025.3638421.
9. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*; Springer Proceedings in Advanced Robotics; Hutter, M., Siegwart, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 5, pp. 621–635. https://doi.org/10.1007/978-3-319-67361-5_40.
10. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 6235–6240. https://doi.org/10.1109/ICRA.2015.7140074.
11. Romero, A.; Penicka, R.; Scaramuzza, D. Time-Optimal Online Replanning for Agile Quadrotor Flight. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7730–7737. https://doi.org/10.1109/lra.2022.3185772.
12. Hanover, D.; Foehn, P.; Sun, S.H.; Kaufmann, E.; Scaramuzza, D. Performance, Precision, and Payloads: Adaptive Nonlinear MPC for Quadrotors. *IEEE Robot. Autom. Lett.* **2022**, *7*, 690–697. https://doi.org/10.1109/lra.2021.3131690.

13. Foehn, P.; Kaufmann, E.; Romero, A.; Penicka, R.; Sun, S.H.; Bauersfeld, L.; Laengle, T.; Cioffi, G.; Song, Y.; Loquercio, A.; et al. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Sci. Robot.* **2022**, *7*, eabl6259. https://doi.org/10.1126/scirobotics.abl6259.
14. Sun, S.H.; de Visser, C.C.; Chu, Q.P. Quadrotor Gray-Box Model Identification from High-Speed Flight Data. *J. Aircr.* **2019**, *56*, 645–661. https://doi.org/10.2514/1.C035135.
15. Wang, M.; Wang, B.; Niu, Y.; Wang, C. PI-WAN: A Physics-Informed Wind-Adaptive Network for Quadrotor Dynamics Prediction in Unknown Environments. In Proceedings of the 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hangzhou, China, 19–25 October 2025; pp. 5413–5419. https://doi.org/10.1109/IROS60139.2025.11247234.
16. Manzoor, T.; Ali, Y.; Xia, Y.; You, L.; Wang, Y. Physics-informed sparse identification-based tube model predictive control for aerial vehicles. *Eng. Appl. Artif. Intell.* **2026**, *181*, 115494. https://doi.org/10.1016/j.engappai.2026.115494.
17. Hewing, L.; Wabersich, K.P.; Menner, M.; Zeilinger, M.N. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annu. Rev. Control Robot. Auton. Syst.* **2020**, *3*, 269–296. https://doi.org/10.1146/annurev-control-090419-075625.
18. Wabersich, K.P.; Zeilinger, M.N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica* **2021**, *129*, 109597. https://doi.org/10.1016/j.automatica.2021.109597.
19. Didier, A.; Zanelli, A.; Wabersich, K.P.; Zeilinger, M.N. Predictive stability filters for nonlinear dynamical systems affected by disturbances. *IFAC-PapersOnLine* **2024**, *58*, 200–207. https://doi.org/10.1016/j.ifacol.2024.09.031.
20. Chee, K.Y.; Hsieh, M.A.; Matni, N. Learning-enhanced Nonlinear Model Predictive Control using Knowledge-based Neural Ordinary Differential Equations and Deep Ensembles. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*; PMLR: Cambridge, MA, USA, 2023; Volume 211, pp. 1125–1137.
21. Jiahao, T.Z.; Chee, K.Y.; Hsieh, M.A. Online Dynamics Learning for Predictive Control with an Application to Aerial Robots. In *Proceedings of The 6th Conference on Robot Learning*; PMLR: Cambridge, MA, USA, 2023; Volume 205, pp. 2251–2261.
22. Chee, K.Y.; Silva, T.C.; Hsieh, M.A.; Pappas, G.J. Enhancing Sample Efficiency and Uncertainty Compensation in Learning-based Model Predictive Control for Aerial Robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023; pp. 9435–9441. https://doi.org/10.1109/iros55552.2023.10341774.
23. Ning, Z.; Fang, X.; Li, Y.; Xie, L. Learning-Based Predictive Control of Switched Nonlinear Systems via Koopman Modeling Approach. *J. Syst. Sci. Complex.* **2025**, *38*, 691–716. https://doi.org/10.1007/s11424-025-4546-0.
24. Wu, G.; Yion, W.T.G.; Dang, K.L.N.Q.; Wu, Z. Physics-informed machine learning for MPC: Application to a batch crystallization process. *Chem. Eng. Res. Des.* **2023**, *192*, 556–569. https://doi.org/10.1016/j.cherd.2023.02.048.
25. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Müller, M.; Koltun, V.; Scaramuzza, D. Learning high-speed flight in the wild. *Sci. Robot.* **2021**, *6*, eabg5810. https://doi.org/10.1126/scirobotics.abg5810.
26. Song, Y.L.; Romero, A.; Müller, M.; Koltun, V.; Scaramuzza, D. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Sci. Robot.* **2023**, *8*, eadg1462. https://doi.org/10.1126/scirobotics.adg1462.
27. Kaufmann, E.; Bauersfeld, L.; Loquercio, A.; Mueller, M.; Koltun, V.; Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature* **2023**, *620*, 982–987. https://doi.org/10.1038/s41586-023-06419-4.
28. Penicka, R.; Song, Y.L.; Kaufmann, E.; Scaramuzza, D. Learning Minimum-Time Flight in Cluttered Environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7209–7216. https://doi.org/10.1109/lra.2022.3181755.
29. Zhou, Z.Y.; Wang, G.; Sun, J.; Wang, J.K.; Chen, J. Efficient and Robust Time-Optimal Trajectory Planning and Control for Agile Quadrotor Flight. *IEEE Robot. Autom. Lett.* **2023**, *8*, 7913–7920. https://doi.org/10.1109/lra.2023.3322075.
30. Foehn, P.; Romero, A.; Scaramuzza, D. Time-optimal planning for quadrotor waypoint flight. *Sci. Robot.* **2021**, *6*, eabh1221. https://doi.org/10.1126/scirobotics.abh1221.
31. Verschueren, R.; Frison, G.; Kouzoupis, D.; Frey, J.; van Duijkeren, N.; Zanelli, A.; Novoselnik, B.; Albin, T.; Quirynen, R.; Diehl, M. acados—A modular open-source framework for fast embedded optimal control. *Math. Program. Comput.* **2022**, *14*, 147–183. https://doi.org/10.1007/s12532-021-00208-8.
32. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi: A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2019**, *11*, 1–36. https://doi.org/10.1007/s12532-018-0139-4.
33. Li, G.R.; Tunchez, A.; Loianno, G. Learning Model Predictive Control for Quadrotors. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 5872–5878. https://doi.org/10.1109/icra46639.2022.9812077.

34. Jiang, B.L.; Li, B.Y.; Zhou, W.F.; Lo, L.Y.; Chen, C.K.; Wen, C.Y. Neural Network Based Model Predictive Control for a Quadrotor UAV. *Aerospace* **2022**, *9*, 460. https://doi.org/10.3390/aerospace9080460.
35. Zanelli, A.; Tran-Dinh, Q.; Diehl, M. A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control. *Automatica* **2021**, *134*, 109901. https://doi.org/10.1016/j.automatica.2021.109901.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.