

Article

An Efficient Differentiable Model Predictive Control for UAV Attitude Regulation

Yue Qu ¹, Jian Huang ², Tianyi Wang ³, Rui Zhang ⁴ and Wenjun Yi ^{2,*}¹ Industrial Perception and Intelligent Manufacturing Equipment Engineering Research Center of Jiangsu Province, Nanjing University of Industry Technology, Nanjing 210023, China; quyue@njust.edu.cn² National Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China³ Jiangnan Industrial Group Co., Ltd., Changsha 410114, China⁴ Inner Mongolia North Heavy Industries Group Limited Company Nanjing Institute, Nanjing 210049, China

* Correspondence: wjy@mail.njust.edu.cn

Abstract

Data-driven learning optimization, which considers optimization as a means to perform end-to-end learning, is an emerging methodology used to solve large-scale learning and continuous control tasks. These methods provide mathematically tractable solutions with inherent interpretability, but their training can be inefficient due to the need to differentiate through the optimization problem and to solve gradient information in every training iteration. To address this issue, we propose a novel learning algorithm called Efficient Differentiable Model Predictive Control (EDMPC), which significantly improves computational efficiency by integrating an optimization-driven inference step within the backpropagation process. More concretely, we derive an analytical gradient of the trajectory from the optimal problem relative to the tunable parameter, enabling significant acceleration of the end-to-end learning procedure. This gradient can be solved by a standard optimizer in a recursive form, which then provides the direction for parameter updates. An unmanned aerial vehicle control task was conducted to demonstrate the effectiveness of EDMPC in controlling complex dynamic systems. The results indicate that the proposed EDMPC method exhibits superior efficiency in the learning process and accuracy in the control process.

Keywords: differentiable model predictive control; computational efficiency; end-to-end learning; unmanned aerial vehicle



Academic Editor: Gokhan Inalhan

Received: 20 February 2026

Revised: 8 April 2026

Accepted: 21 April 2026

Published: 31 July 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 \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly deployed in a range of demanding tasks, such as swarm flight [1] and large-scale, high-efficiency delivery missions [2]. To meet the core requirements of low cost, low energy consumption, and high flexibility, under-actuated UAVs have become dominant in the market due to their inherent compatibility with these objectives. The primary control challenge for such under-actuated systems stems from the fact that the number of control inputs is fewer than the system's degrees of freedom [3]. This results in uncontrollable subspaces within the state space, making direct and precise attitude regulation via conventional linear control strategies inherently infeasible [4].

Over the past few decades, various methods have been explored to achieve attitude regulation for under-actuated UAVs. One class of approaches utilizes feedback lineariza-

tion, applying a nonlinear coordinate transformation to linearize the controllable dynamics of the under-actuated system [5]. For the remaining uncontrollable subspace, techniques like passive stabilization or constrained optimization are implemented, ensuring the primary stability of key states (such as attitude) while indirectly attenuating perturbations. Another class of methods includes backstepping [6] and dynamic surface control [7]. These approaches decompose the system into cascaded subsystems and design control laws recursively from the inner to the outer loops. Stability is established via a Lyapunov-based inductive framework, which indirectly manages coupling in the uncontrollable subspace [8]. A notable limitation of the above methods is their high dependence on precise model knowledge and their generally complex design process. Sliding mode control exhibits strong robustness to model inaccuracies [9]. The core idea is to design a sliding surface that restricts the complex dynamics of an under-actuated system to a lower-dimensional manifold [10]. A discontinuous control law forces the system state to converge along this surface, thereby rejecting disturbances from the uncontrollable subspace and maintaining robustness. However, such methods inherently suffer from the chattering phenomenon, where high-frequency switching in the control signal may damage actuators. Optimal Control (OC), powered by modern optimization algorithms such as Dynamic Programming (DP) [11], Differential Dynamic Programming (DDP) [12], and iterative Linear Quadratic Regulation (iLQR) [13], serves as the cornerstone for numerous challenging control and decision-making tasks. However, OC methods are often vulnerable to performance degradation caused by model inaccuracies. Model Predictive Control (MPC) is a modern control methodology that systematically handles constraints, performance specifications, and nonlinear dynamics. At its core lies a receding-horizon strategy: at each time step, an OC problem is solved over a finite prediction horizon; the first element of the optimal input sequence is then applied, and the procedure repeats with updated measurement [14,15]. Although classical MPC requires an accurate model, learning-based [16–19] approaches offer effective ways to improve its performance using data.

More recently, machine learning has been employed to train agents for under-actuated control by a reward function including attitude and position errors, enabling them to learn “attitude regulation and position tracking” policies, which directly map current states to actions. While such data-driven methods are adaptable to complex nonlinear systems, they produce black-box policies and are often plagued by high sample complexity and poor generalization. There is growing interest in integrating control-theoretic policies with machine learning techniques. One class of such methods integrates optimization problems as individual layers within larger end-to-end trainable deep networks, as exemplified by methods like OptNet [20], a groundbreaking framework that embeds quadratic programs (QPs) as differentiable layers in end-to-end networks. In a similar vein, Tamar et al. differentiate through an iterative LQR (iLQR) solver to learn a cost-shaping term offline [21]. Williams et al. presented a way to differentiate through path integral optimal control and learn a planning policy end-to-end [22]. Previous work with similar aims has either simply unrolled or differentiated through a simple optimization procedure. Another strategy is to differentiate through the MPC algorithm itself [23,24]. Amos et al. proposed differentiable MPC, which differentiates through MPC solvers using the KKT conditions of convex approximations at fixed points [25]. To further unify learning and control tasks, Jin et al. developed Pontryagin Differentiable Programming (PDP) [26] and safe PDP [27], a framework that differentiates through Pontryagin’s Maximum/Minimal Principle (PMP) to obtain a gradient with respect to the decision variables and enable back propagation through it. To relax the restriction that the optimization problem must be convex or easily relaxable, non-convex, Kong et al. proposed SO-EBM [28], which parameterizes explicit differentiable layers using energy functions. Collectively, these works advance a paradigm

shift from isolated optimization and learning to integrated end-to-end frameworks. However, they can be inefficient in training due to the requirement of solving and differentiating through the optimization problem in every training iteration.

Motivated by the above, we propose a novel learning framework called Efficient Differentiable Model Predictive Control (EDMPC), which is both capable enough for diverse control tasks and efficient enough for handling high-dimensional, continuous-space problems. A key advantage of EDMPC is its computational efficiency. As EDMPC differentiates through MPC solvers using Karush–Kuhn–Tucker (KKT) conditions, EDMPC eliminates the need to unroll the optimization procedure to obtain derivatives in every training iteration. Furthermore, we derive an analytical gradient by reconstructing the MPC problem; the solution of the reconstructed problem is exactly the gradient of the trajectory of the original MPC problem with respect to the parameter. Our contributions are summarized as follows:

- (1) We propose an end-to-end learning framework to learn the control policy for controlling under-actuated systems. To validate the performance of EDMPC for complex dynamic systems, we applied the proposed method in the task of UAV attitude regulation.
- (2) We integrate an optimization-driven inference step into the learning process. This enables an efficient method that computes the analytical gradient of the trajectory with respect to the parameters using standard optimization solvers.

2. Problem Statement

2.1. Optimization Objective

The prediction model of EDMPC is formulated as a general discrete-time time-invariant dynamical system:

$$x_{t+1} = f(x_t, u_t) \quad (1)$$

where $x_t \in \mathbb{R}^n$ is the system state, and $u_t \in \mathbb{R}^m$ is the control input. The dynamic model, defined by the function $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, is nonlinear and input-non-affine, and it is assumed to be twice differentiable.

At each time step t , EDMPC solves the following finite prediction horizon OC problem:

$$\begin{aligned} \min_{\xi} L_t^{\xi}(x_t) &:= T_N(x_N) + \sum_{t=0}^{N-1} l_t(\xi_t) \\ \text{s.t. } &(1) \end{aligned} \quad (2)$$

where the trajectory $\xi = \{x_{0:N}, u_{0:N-1}\}$ is a sequence of state–control pairs serving as the optimization variable, N is the prediction horizon, and $L_t^{\xi}: \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ represents the objective function, which includes the cost function $l_t(\xi_t): \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ and the terminal cost function $T_N(x_N): \mathbb{R}^n \mapsto \mathbb{R}$.

2.2. Parameterization of the EDMPC Problem

To integrate the OC problem into an end-to-end learning framework, we parameterize the OC problem with the parameter $\theta \in \mathbb{R}^p$:

$$\begin{aligned} \xi_{\theta}^* &\in \argmin_{\xi} L_t^{\xi}(x_t, \theta) := T_N(x_N, \theta) + \sum_{t=0}^{N-1} l_t(\xi_t, \theta) \\ \text{s.t. } &(1) \end{aligned} \quad (3)$$

For a determined parameter θ , one can find an optimal trajectory $\xi_{\theta}^* = \{x_{0:N}^{\theta}, u_{0:N-1}^{\theta}\}$ by solving (3). $u^{\theta} = u(t, x_t, \theta)$ is a parameterized policy, where the subscript θ indicates that the solution of the policy depends on the parameter θ . The framework of EDMPC is

designed to train the optimal parameter θ that minimizes the objective function associated with the policy learning problem.

3. Efficient Differentiable Model Predictive Control

3.1. End-to-End Learning Framework

From the perspective of policy learning, (3) can be viewed as a learnable module characterized by the unknown parameter θ . Such a module can be seamlessly integrated within an end-to-end learning framework, which is illustrated in Figure 1. The core procedure of the training process entails computing the derivatives of a predefined objective function L with respect to θ , which are subsequently used to update the parameters:

$$\theta_{J+1} = \theta_J - \alpha \frac{dL(\xi_\theta, \theta)}{d\theta} \quad (4)$$

where $J = 0, 1, \dots$ represents the iteration number, α is the learning rate, and $\frac{dL(\xi_\theta, \theta)}{d\theta}$ is the derivative of the function L with respect to the parameter vector θ . This derivative can be computed using the rule specified below:

$$\frac{dL}{d\theta} = \frac{\partial L}{\partial \xi_\theta^*} \frac{\partial \xi_\theta^*}{\partial \theta} + \frac{\partial L}{\partial \theta} \quad (5)$$

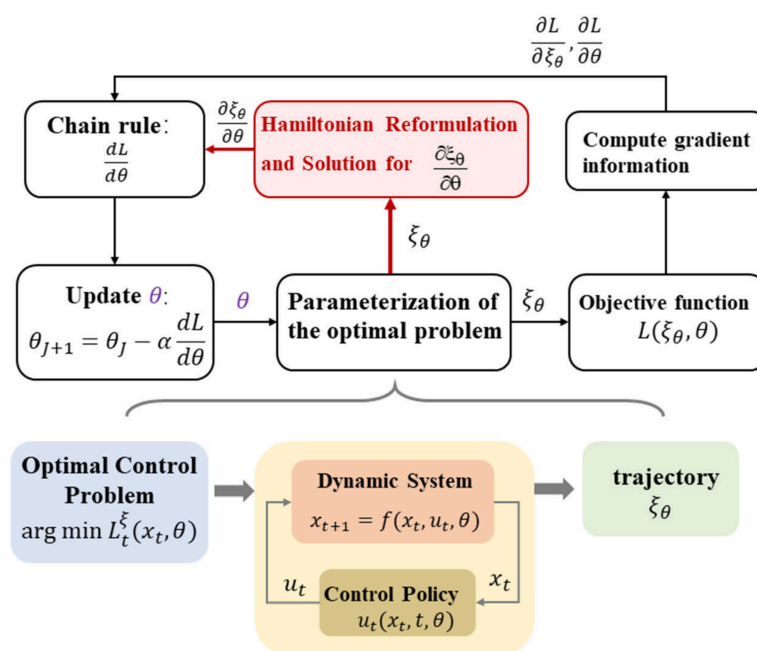


Figure 1. Schematic diagram of EDMPC, where an optimal control system is parameterized by a tunable parameter θ in control objective function (marked with a blue block), dynamic (marked with an orange block) and policy (marked with a brown block). The framework of EDMPC involves a forward pass given θ , and a backward pass where $\frac{\partial L}{\partial \xi_\theta^*}$, $\frac{\partial L}{\partial \theta}$ and $\frac{\partial \xi_\theta^*}{\partial \theta}$ are computed. Hamiltonian reformulation and optimal control solution for $\frac{\partial \xi_\theta^*}{\partial \theta}$ is the key point, which is marked with a red block.

It can be observed that each update of the parameter θ encompasses two distinct phases: a forward pass and a backward pass. During the forward pass at the J -th iteration, EDMPC can employ any OC solver—such as iLQR [13] or CasADi [29]—to solve the optimization problem specified in (3) and obtain the optimal trajectory $\xi_{\theta_J}^*$. The backward pass comprises the following steps:

- compute the gradient of the objective function with respect to the trajectory $\frac{\partial L}{\partial \xi_\theta^*}$;

- compute the gradient of the trajectory ζ_θ^* with respect to the parameter $\frac{\partial \zeta_\theta^*}{\partial \theta}$;
- compute the gradient of the objective function with respect to the parameter $\frac{\partial L}{\partial \theta}$.

Finally, based on the chain rule, the total gradient of the objective function with respect to θ_j is obtained using (5), and the parameter θ_{j+1} is subsequently updated via (4).

In the backward pass, obtaining $\frac{\partial L}{\partial \zeta^*}$ and $\frac{\partial L}{\partial \theta}$ is easy; the key challenge lies in how to efficiently and accurately compute the gradient of the trajectory with respect to the parameter θ , i.e., $\frac{\partial \zeta_\theta^*}{\partial \theta}$. To this end, we propose a systematic analytical approach to derive the corresponding gradient update law for efficiently computing $\frac{\partial \zeta_\theta^*}{\partial \theta}$.

3.2. Hamiltonian Reformulation and OC Solution for $\frac{\partial \xi_\theta}{\partial \theta}$

Following the preceding analysis, the policy learning problem can be formulated as solving the constrained optimization problem below:

$$\min_{\theta} L(\xi_\theta, \theta) = \sum_{t=0}^{N-1} l_t(x_t, u_t, \theta) + T_N(x_N, \theta) \quad (6)$$

s.t. (1).

For the constrained OC problem (6), the standard approach involves introducing Lagrange multipliers $\lambda \in \mathbb{R}^n$ to transform the original problem into an unconstrained OC problem:

$$\min_{\theta} \hat{L}(\xi_\theta, \theta) = T_N(x_N, \theta) + \sum_{t=0}^{N-1} \left[l_t(x_t, u_t, \theta) + \lambda_t^{\theta \top} (f(x_t, u_t) - x_{t+1}) \right] \quad (7)$$

To solve the above unconstrained OC problem (7), the Hamiltonian function [30] is introduced:

$$H_t = l_t(x_t, u_t, \theta) + \lambda_t^{\theta \top} f(x_t, u_t) \quad (8)$$

Then, (7) can be rewritten as:

$$\hat{L}(\xi_\theta, \theta) = T_N(x_N, \theta) + \sum_{t=0}^{N-1} \left[H_t(x_t, u_t, \theta) - \lambda_t^{\theta \top} x_{t+1} \right] \quad (9)$$

According to Pontryagin's Maximum Principle, the optimal solution of (9) must maintain the following conditions:

$$\begin{aligned} \text{Costate equation :} \quad & \lambda_{t-1}^{\theta} = \frac{\partial H_t}{\partial x_t^{\theta}} = \frac{\partial l_t}{\partial x_t^{\theta}} + \frac{\partial f}{\partial x_t^{\theta}} \lambda_t^{\theta} \\ \text{Dynamics equation :} \quad & x_{t+1}^{\theta} = \frac{\partial H_t}{\partial \lambda_t^{\theta}} = f(x_t^{\theta}, u_t^{\theta}) \\ \text{Input equation :} \quad & 0 = \frac{\partial H_t}{\partial u_t^{\theta}} = \frac{\partial l_t}{\partial u_t^{\theta}} + \frac{\partial f}{\partial u_t^{\theta}} \lambda_t^{\theta} \\ \text{Boundarycondition :} \quad & \lambda_N^{\theta} = \frac{\partial T_N}{\partial x_N^{\theta}}, \quad x_0^{\theta} = x_0 \end{aligned} \quad (10)$$

Note that ξ_θ^* can be obtained through a forward pass and $\lambda_{1:N}^{\theta}$ can be calculated by solving (10). Differentiating (10) with respect to θ gives:

$$\begin{aligned} \frac{\partial \lambda_{t-1}^{\theta}}{\partial \theta} &= \frac{\partial^2 H_t}{\partial x_t^{\theta} \partial x_t^{\theta}} \frac{\partial x_t^{\theta}}{\partial \theta} + \frac{\partial^2 H_t}{\partial x_t^{\theta} \partial u_t^{\theta}} \frac{\partial u_t^{\theta}}{\partial \theta} + \frac{\partial f}{\partial x_t^{\theta}} \frac{\partial \lambda_t^{\theta}}{\partial \theta} + \frac{\partial^2 H_t}{\partial x_t^{\theta} \partial \theta} \\ \frac{\partial x_{t+1}^{\theta}}{\partial \theta} &= \frac{\partial f}{\partial x_t^{\theta}} \frac{\partial x_t^{\theta}}{\partial \theta} + \frac{\partial f}{\partial u_t^{\theta}} \frac{\partial u_t^{\theta}}{\partial \theta} + \frac{\partial f}{\partial \theta} \\ 0 &= \frac{\partial^2 H_t}{\partial u_t^{\theta} \partial x_t^{\theta}} \frac{\partial x_t^{\theta}}{\partial \theta} + \frac{\partial^2 H_t}{\partial u_t^{\theta} \partial u_t^{\theta}} \frac{\partial u_t^{\theta}}{\partial \theta} + \frac{\partial f}{\partial u_t^{\theta}} \frac{\partial \lambda_t^{\theta}}{\partial \theta} + \frac{\partial^2 H_t}{\partial u_t^{\theta} \partial \theta} \\ \frac{\partial \lambda_N^{\theta}}{\partial \theta} &= \frac{\partial^2 T_N}{\partial x_N^{\theta} \partial x_N^{\theta}} \frac{\partial x_N^{\theta}}{\partial \theta} + \frac{\partial^2 T_N}{\partial x_N^{\theta} \partial \theta}, \quad \frac{\partial x_0^{\theta}}{\partial \theta} = \frac{\partial x_0}{\partial \theta} = 0. \end{aligned} \quad (11)$$

The specific form of the derivative $\frac{\partial \lambda_{1:N}^\theta}{\partial \theta}$ and $\frac{\partial x_{0:N}^\theta}{\partial \theta}$ arises from the linearization of the Pontryagin's Maximum Principle conditions, and the matrix-valued structure of the derivative follows from the need to handle high-dimensional parameter spaces, and its recursive solution aligns with efficient numerical methods for constrained optimal control problems [30]. In (11), the unknown variables are: $\frac{\partial x_{0:N}^\theta}{\partial \theta}$, $\frac{\partial u_{0:N-1}^\theta}{\partial \theta}$ and $\frac{\partial \lambda_{1:N}^\theta}{\partial \theta}$. Now, an OC problem can be reformulated based on (11). Before that, the reconstructed OC problem includes $\hat{x}_t$ and $\hat{u}_t$ as its new state and control variables, respectively:

$$\hat{x}_t = \frac{\partial x_t}{\partial \theta} \in \mathbb{R}^{n \times p}, \quad \hat{u}_t = \frac{\partial u_t}{\partial \theta} \in \mathbb{R}^{m \times p} \quad (12)$$

Then, by following the approach presented in [31], we reconstructed an OC problem satisfying the optimality conditions (11), which is rigorously formulated as follows:

$$\begin{aligned} \min_{\hat{x}_{0:N}, \hat{u}_{0:N-1}} \hat{L} = & \text{Tr} \sum_{t=0}^{N-1} \left(\frac{1}{2} \begin{bmatrix} \hat{x}_t \\ \hat{u}_t \end{bmatrix}^\top \begin{bmatrix} A_t & B_t \\ B_t^\top & E_t \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{u}_t \end{bmatrix} + \begin{bmatrix} D_t \\ C_t \end{bmatrix}^\top \begin{bmatrix} \hat{x}_t \\ \hat{u}_t \end{bmatrix} \right) \\ & + \frac{1}{2} \text{Tr}(\hat{x}_N^\top A_N \hat{x}_N) + \text{Tr}(D_N x_N) \\ \text{s.t. } & \hat{x}_t = 0 \\ & \hat{x}_{t+1} = F_t^x \hat{x}_t + F_t^u \hat{u}_t + F_t, \end{aligned} \quad (13)$$

where $\text{Tr}(\cdot)$ denotes the trace of the matrix. To simplify notation, several known quantities are defined as:

$$\begin{aligned} F_t^x &= \frac{\partial f}{\partial x_t^\theta}, \quad F_t^u = \frac{\partial f}{\partial u_t^\theta}, \quad F_t = \frac{\partial f}{\partial \theta}, \\ A_t &= \frac{\partial^2 H_t}{\partial x_t^\theta \partial x_t^\theta}, \quad B_t = \frac{\partial^2 H_t}{\partial x_t^\theta \partial u_t^\theta} = \left(\frac{\partial^2 H_t}{\partial u_t^\theta \partial x_t^\theta} \right)^\top \\ D_t &= \frac{\partial^2 H_t}{\partial x_t^\theta \partial \theta}, \quad E_t = \frac{\partial^2 H_t}{\partial u_t^\theta \partial u_t^\theta}, \quad C_t = \frac{\partial^2 H_t}{\partial u_t^\theta \partial \theta}, \\ A_N &= \frac{\partial^2 T_N}{\partial x_N^\theta \partial x_N^\theta}, \quad D_N = \frac{\partial^2 T_N}{\partial x_N^\theta \partial \theta}. \end{aligned} \quad (14)$$

Note that quantities defined in (14) can be obtained in the forward propagation process.

Here, we can see that the reconstructed optimization problem in (13) is a Linear Quadratic (LQ) problem, and both the dynamic and its solution are determined by ξ_θ of the original problem in (3). Hence, we obtain the following important results.

Lemma 1. Let $\{\hat{x}_{0:N}^\theta, \hat{u}_{0:N-1}^\theta\}$ be the output of the reconstructed optimization problem in (13). It satisfies the optimal conditions in (11), and:

$$\left\{ \hat{x}_{0:N}^\theta, \hat{u}_{0:N-1}^\theta \right\} = \left\{ \frac{\partial x_{0:N}^\theta}{\partial \theta}, \frac{\partial u_{0:N-1}^\theta}{\partial \theta} \right\} = \frac{\partial \xi_\theta}{\partial \theta}. \quad (15)$$

A proof of Lemma 1 is detailed in Appendix A, and it explains the assertion that $\{\hat{x}_{0:N}^\theta, \hat{u}_{0:N-1}^\theta\}$ is exactly the gradient of the trajectory of the original optimal problem with respect to θ . Owing to Lemma 1, the solution $\frac{\partial \xi_\theta}{\partial \theta}$ can be efficiently derived by appealing to Lemma 2.

Lemma 2. The analytical solution of (13) can be obtained by leveraging dynamic programming as below. First, we can define backward recursions as:

$$\begin{aligned} P_t &= Q_t + \Lambda_t^\top (I + P_{t+1} R_t)^{-1} P_{t+1} \Lambda_t \\ W_t &= \Lambda_t^\top (I + P_{t+1} R_t)^{-1} (W_{t+1} + P_{t+1} M_t) + N_t \end{aligned} \quad (16)$$

with $P_N = A_N, W_N = D_N$. I is the identity matrix, Q_t, Λ_t, R_t, M_t and N_t

$$\begin{aligned} Q_t &= A_t - B_t E_t^{-1} B_t^\top, \Lambda_t = F_t^x - F_t^u E_t^{-1} B_t^\top, R_t = F_t^u E_t^{-1} (F_t^u)^\top, \\ M_t &= F_t - F_t^u E_t^{-1} C_t, N_t = D_t - B_t E_t^{-1} C_t \end{aligned}$$

are known. Then, we can forward update the new control input and the new state:

$$\begin{aligned} \hat{u}_t^\theta &= -E_t^{-1} \left(B_t^\top \hat{x}_t^\theta + C_t + (F_t^u)^\top (I + P_{t+1} R_t)^{-1} (P_{t+1} \Lambda_t \hat{x}_t^\theta + P_{t+1} M_t + W_t) \right), \\ \hat{x}_{t+1}^\theta &= F_t^x \hat{x}_t^\theta + F_t^u \hat{u}_t^\theta + F_t, \end{aligned} \quad (17)$$

The proof of Lemma 2 is presented in Appendix B. We summarize the above procedures in Algorithm 1.

Algorithm 1: Solving the analytic solution of $\frac{\partial \xi_\theta}{\partial \theta}$

Input: The trajectory ξ_θ

1. initialize $P_N \leftarrow A_N, W_N \leftarrow D_N$;
2. **for** $t \leftarrow N$ **to** 0 **do**
3. | update P_t and W_t backward in time using (16);
4. **end for**
5. initialize $\hat{x}_0^\theta, \hat{u}_0^\theta \leftarrow 0$;
6. **for** $t \leftarrow 0$ **to** N **do**
7. | update $\hat{x}_t^\theta, \hat{u}_t^\theta$ forward in time using (17);
8. **end for**

Return: $\frac{\partial \xi_\theta}{\partial \theta} = \{\hat{x}_{0:N}^\theta, \hat{u}_{0:N-1}^\theta\}$

The approach is computationally efficient. The key lies in its recursive formulation derived from explicitly solving the Riccati equation, which enables an iterative solution procedure.

3.3. Model-Based Policy Gradient

Thanks to the above results, we show that the analytical solution of the system (13) is exactly the gradient of the trajectory of the original system with respect to θ . Given the analytical gradient in (5), we construct a model-based policy gradient algorithm that learns the parameter directly from the task-related objective function:

$$L(\xi_\theta, \theta) = \sum_{t=0}^{N-1} \|x_t - x_t^{ref}\|_{\Xi}^2 + T_N(x_N, \theta) \quad (18)$$

where x_t^{ref} is the reference state and Ξ is a weight matrix that adjusts the importance of different state components in the objective function. Here, the objective function for the training parameter θ has been constructed as the weighted summation of the trajectory tracking errors over the prediction horizon N . We summarize the algorithm in Algorithm 2.

Algorithm 2: model-based policy gradient**Reference**

state: x_t^{ref} , learning rate α , duration for each episode T_{end} , convergence tolerance ε

Initialization: $x_0, u_0, \theta_0 \leftarrow \mathbf{0}, L_{prev} \leftarrow \infty$

1. **while** $|L_{prev} - L(\xi_\theta, \theta)| > \varepsilon$ **do**
2. $L_{prev} = L(\xi_\theta, \theta);$
3. **for** $J \leftarrow 0$ **to** T_{end}
4. Solve (3) to obtain ξ_θ ;
5. Obtain $\frac{\partial \xi_\theta}{\partial \theta}$ using Algorithm 1;
6. Compute current $L(\xi_\theta, \theta)$ using (18);
7. Apply the chain rule (5) to obtain $\frac{dL}{d\theta}$;
8. Update $\theta_{J+1} = \theta_J - \alpha \frac{dL(\xi_\theta, \theta)}{d\theta}$;
9. **end for**
10. **end while**

4. EDMPC Solving for UAV Attitude Regulation

To demonstrate the performance of DMPC for UAV attitude regulation, this section presents its application to the task of UAV trajectory tracking.

A continuous-time 6-DoF UAV model $f_c(x, u)$ is given by (for a more detailed explanation, please refer to Reference [32]):

$$\begin{aligned} \dot{p}^n &= R_b^n v^b \\ m\dot{v}^b &= R_n^b f_g^n + f_{thrust}^b \\ \dot{q}_{b,n} &= \frac{1}{2}\Omega(q_{b,n}) \begin{bmatrix} 0 \\ w^b \end{bmatrix} \\ J\dot{w}^b &= -Y(w^b)Jw^b + M^b \end{aligned} \quad (19)$$

where the superscripts b, n indicate the body frame and inertial frame, respectively, which are simplified in the notation of state and action for clarity (see Table 1). $p^n \in \mathbb{R}^3$ is the position in the inertial frame, $v^b, w^b \in \mathbb{R}^3$ represents the velocity and angular velocity in the body frame. $f_g^n = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T$ represents the gravitational force vector. $f_{thrust}^b = \begin{bmatrix} T_1 & 0 & 0 \end{bmatrix}^T$ and $M^b = \begin{bmatrix} T_2 & T_3 & T_4 \end{bmatrix}$ are control inputs representing thrust and torque in the UAV's body. R_b^n denotes the transformation matrix from body frame to inertial frame, and J is the inertia matrix. $q_{b,n} = \begin{bmatrix} \eta_{b,n} & \mu_{b,n}^T \end{bmatrix}^T \in \mathbb{R}^4$ is the quaternion that describes the rotation of the body frame relative to the inertial frame, where $\eta_{b,n}$ is the scalar part and $\mu_{b,n}$ is the vector part. The quaternion multiplication operator $\Omega(q_{b,n})$ is defined as:

$$\Omega(q_{b,n}) := \begin{bmatrix} \eta_{b,n} & -\mu_{b,n}^T \\ \mu_{b,n}^T & \eta_{b,n}I + Y(\mu_{b,n}) \end{bmatrix}, \quad (20)$$

where $Y(\cdot)$ is an operator, and $Y(w^b)$ for $w^b = [w_x, w_y, w_z]^T$ is defined as:

$$Y(w^b) := \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}. \quad (21)$$

Table 1. Details of the system state and control input.

	Symbol	Unit	Explanation
state vector x_t	$p_t \in \mathbb{R}^3$	[m]	position
	$v \in \mathbb{R}^3$	[m/s]	velocity
	$w \in \mathbb{R}^3$	[deg/s]	angular velocity
	$q \in \mathbb{R}^3$		attitude
Input u_t	T_1	[N]	thrust
	T_2, T_3, T_4	[Nm]	torque

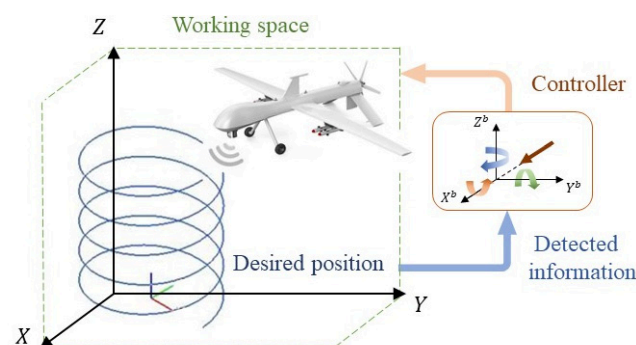
The continuous-time UAV model $f_c(x_t, u_t)$ can be discretized using the forward Euler method [33]:

$$x_{t+1} = x_t + f_c(x_t, u_t)\Delta t, \quad (22)$$

where Δt is the time interval, which is set as 0.05 s in this task. The state variables and control inputs of the UAV dynamic system are presented below:

4.1. Policy Optimization for Attitude Regulation

The first control objective is to drive the 6-DoF UAV to track a desired trajectory with the shortest path while realizing a minimum flight time, as depicted in Figure 2. Real-time states are assumed to be detected by the onboard sensor.

**Figure 2.** The waypoint approach task.

The control policy is parameterized as $u^\theta = u(t, \theta)$, where $\theta = [u_0, \dots, u_N]^T$. Task loss l_t is defined as:

$$l_t = \|p_t - p_d\|^2 + \|v_t - v_d\|^2 + \|w_t\|^2 + \frac{1}{2}\text{Tr}(I - R'(q_t)R(q_d)) + \|u_t\|^2, \quad (23)$$

where subscript d denotes the desired vector, for instance, the desired trajectory is defined as:

Desired position trajectory: The 3D position reference p_d is defined as a helical trajectory: $p_d(t) = [10^3 \cos(0.05t), 10^3 \sin(0.05t), t]^T$ m;

Desired velocity: The desired velocity magnitude $v_d = 40$ m/s is set to maintain a constant forward speed for the fixed-wing UAV, ensuring stable flight along the helical path.

Desired attitude: The desired attitude q_d is designed using the method presented in [32], which aligns the UAV's body frame with the tangent direction of the helical trajectory.

Direction cosine matrix: The direction cosine matrix $R(q) \in \mathbb{R}^{3 \times 3}$ is derived from the attitude quaternion q_d , which transforms vectors from the body frame to the inertial frame and is used to compute the attitude tracking error in the cost function.

In (23), the first term imposes a penalty for deviation of the UAV's current position from its desired position, the second and third terms penalize high velocity and angular velocity to encourage smoother control policies. The final term addresses attitude error, promoting the UAV's maintenance of the desired attitude.

We set the learning rate $\alpha = 10^{-4}$ and randomly selected the initial state x_0 . We implemented the simulation in Python 3.13, employing Ipopt via CasADi [34] as the optimization solver. The computational environment is a PC equipped with an AMD Ryzen 95950X 16-core processor.

Here, we conducted a comparison between EDMPC and DMPC [20]; the main difference between the above two methods lies in the approach to gradient computation. In the backward pass, DMPC computes gradients by differentiating a local LQR approximation of the original optimal control system. This process involves inverting a coefficient matrix of size $(2n + m)T \times (2n + m)T$.

Figure 3 plots the objective function value against the number of iteration steps for four experiments. The above training results show that both DMPC and EDMPC ($N = 50$) converge to a minimum after 150 iterations. This suggests that EDMPC successfully finds the optimal control policy to successfully perform the UAV tracking the desired trajectory. Compared to DMPC, EDMPC achieves a lower loss. This is because EDMPC performs differentiation in the backward pass directly on the parameterized optimal control system, rather than on the LQR approximation. Conversely, EDMPC and DMPC with a shorter control horizon ($N = 5$) fail to converge effectively, as evidenced by the persistently high objective function value, which indicates that the performance of EDMPC is quite sensitive to the value of N .

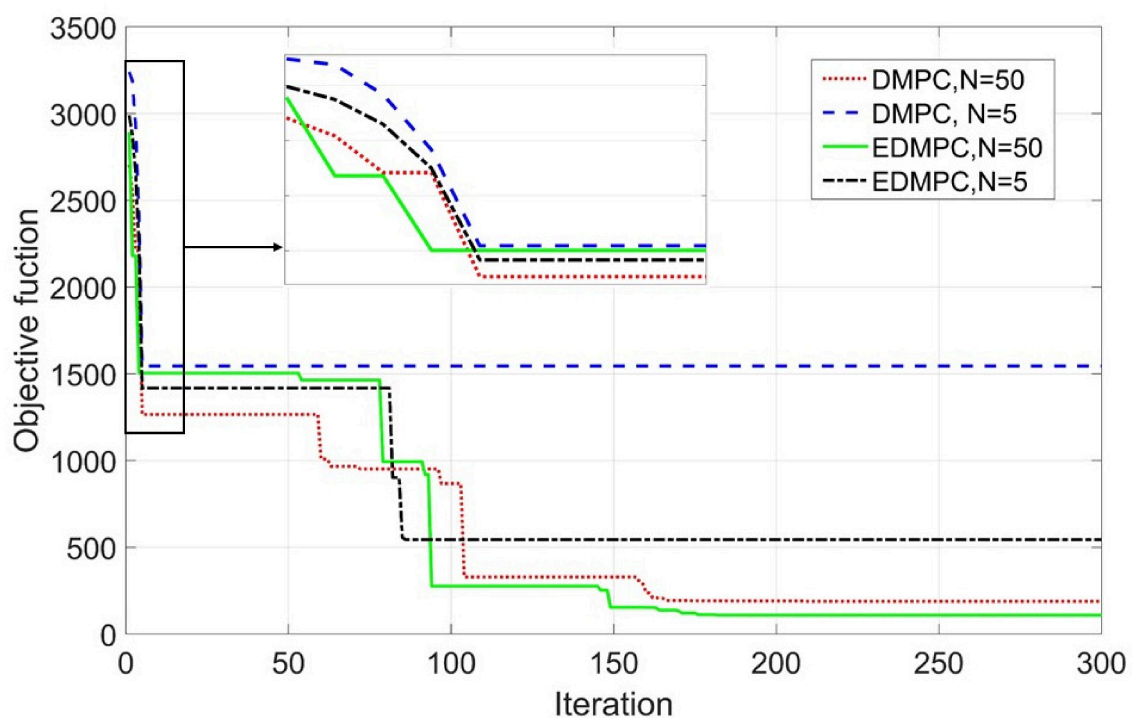


Figure 3. Objective function value versus iteration.

We further tested the algorithm efficiency by measuring the running time needed for both DMPC and EDMPC one per iteration. The result (see Figure 4) indicates that the running time of both DMPC and EDMPC increases with the growth of N . Table 2 summarizes the running time for different horizons ($N = 5, 20, 50, 100$). It is demonstrated that EDMPC significantly surpasses DMPC in computational efficiency, especially during

backpropagation, achieving a computational speed improvement of 55% to 81%. This indicates that EDMPC can be executed at a high frequency. The reason for this significant improvement is that EDMPC avoids inverting a large coefficient matrix; instead, it solves a forward-backward Riccati recursion, which is far more efficient and numerically stable. Consequently, EDMPC is particularly well-suited for long-horizon control and learning tasks.

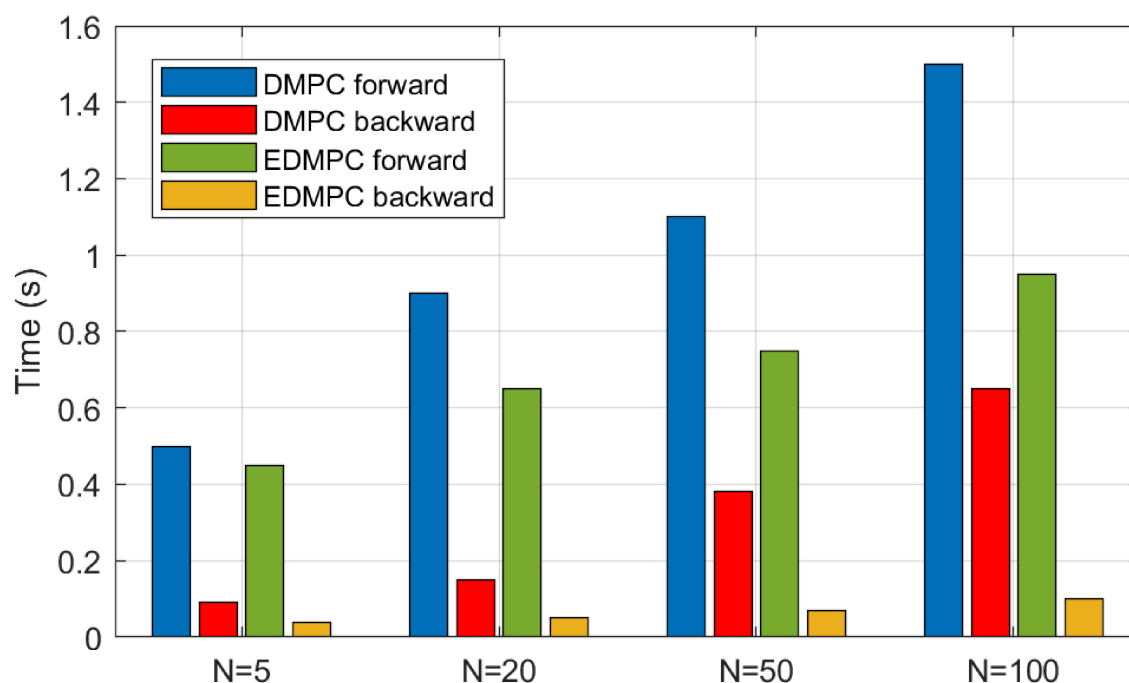
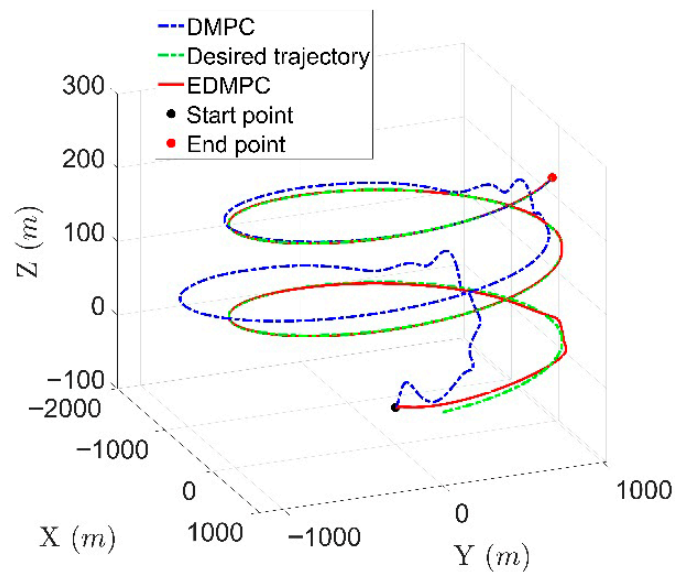


Figure 4. Running time (per iteration).

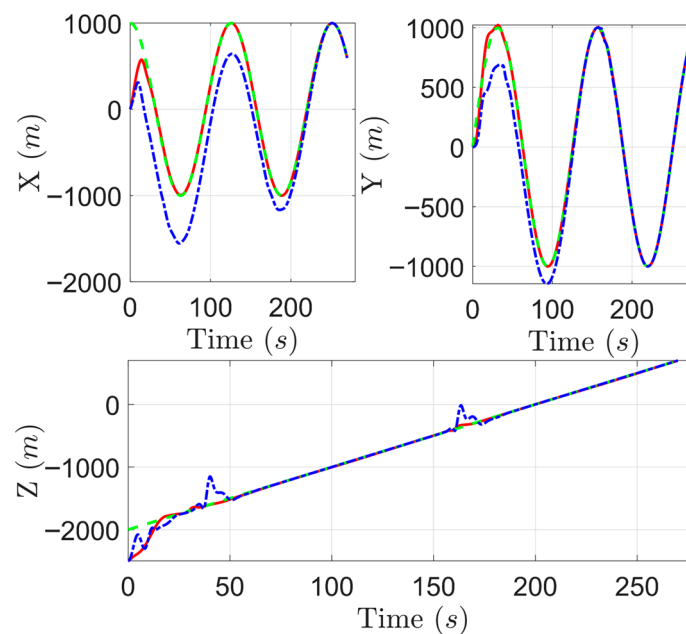
Table 2. Runtime statistics of EDMPC and DMPC.

	Control Horizon	Forward	Backward	Total
EDMPC	N = 5	0.453 s	0.041 s	0.494 s
	N = 20	0.652 s	0.053 s	0.705 s
	N = 50	0.754 s	0.076 s	0.830 s
	N = 100	0.951 s	0.120 s	1.071 s
DMPC	N = 5	0.523 s	0.091 s	0.614 s
	N = 20	0.912 s	0.151 s	1.063 s
	N = 50	1.121 s	0.383 s	1.504 s
	N = 100	1.54 s	0.653 s	2.193 s

We next demonstrate the learned controller's ability to execute control tasks. The UAV's tracking trajectories are illustrated in Figure 5. In comparison, EDMPC effectively regulates the UAV's attitude, enabling the position to gradually converge to the desired trajectory, which results in higher trajectory convergence accuracy and a faster convergence speed (see Figure 5b).



(a) Trajectory tracking diagram in three-dimensional space.



(b) Trajectory tracking plots for the X, Y, and Z dimensions separately.

Figure 5. UAV tracking trajectory in (a) three-dimensional space, and in (b) X, Y, and Z dimensions.

Figure 6 demonstrates that EDMPC maintains the UAV at the prescribed reference velocity. Figure 7 illustrates the attitude regulation performance, where the attitude quaternions converge to steady values—specifically, $q(3)$ and $q(4)$ stabilize at nonzero constants. This behavior arises because the UAV must sustain a persistent body tilt to follow the designed spiral-ascent reference trajectory. Figure 8 further confirms the stable convergence of the angular velocity. In summary, the controller trained via the EDMPC algorithm achieves effective UAV attitude regulation and successfully executes the predefined trajectory-tracking mission.

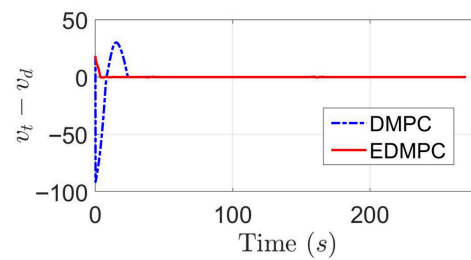


Figure 6. Velocity tracking error.

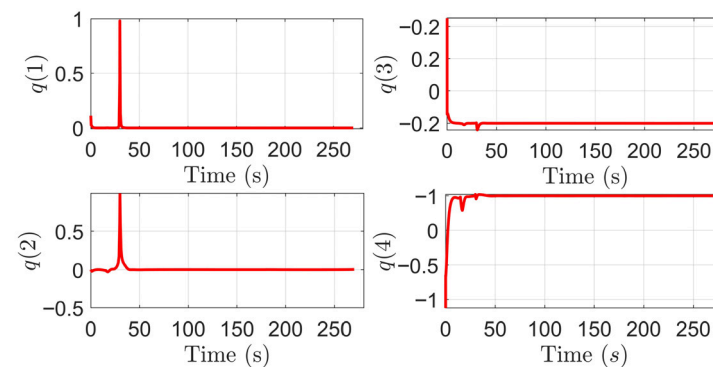


Figure 7. Attitude of UAV.

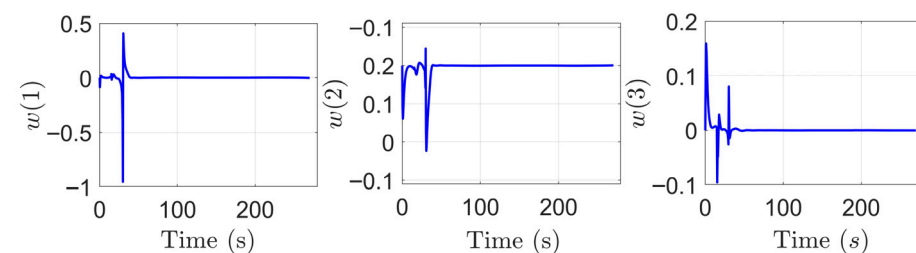


Figure 8. Angular velocity of UAV.

We also use Euler angles to describe the attitude of the UAV, as shown in Figure 9, where the yaw angle exhibits a linear increase with time, indicating a steady turning rate during the helical motion. The abrupt jumps at approximately $t = 30$ s and $t = 150$ s are due to angle wrapping (normalization to the $[-\pi, \pi]$ interval) rather than actual attitude discontinuities. The roll angle and pitch angle, which remain near zero with small fluctuations, demonstrate that the UAV maintains a nearly level attitude while executing the helical trajectory.

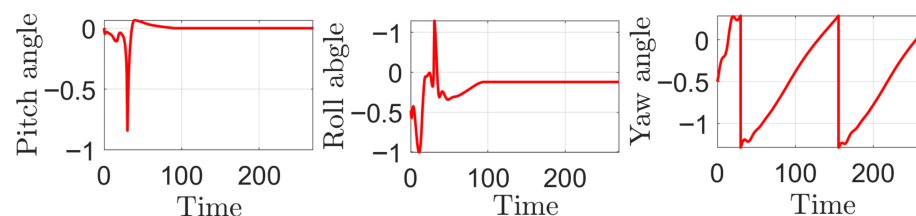


Figure 9. Euler angles of UAV.

4.2. Safe Motion Planning

We also demonstrated the scalability of EDMPC in addressing safety-critical control problems, such as obstacle avoidance tasks. The procedure is as follows:

- Step 1: Define safety constraints;

- Step 2: Incorporate the safety constraints into the objective function through barrier functions;
- Step 3: Use EDMPC to solve the safety-critical control task in an end-to-end learning pipelines.

The second control objective is to drive the 6-DOF UAV to a desired waypoint with the shortest path while avoiding a moving obstacle. A safety constraint can be established as inequalities based on the relative distance between the UAV and the obstacle:

$$C_t(x_t) = \tau_{safe} - \|p_t - p_{obs}\| \leq 0 \quad \forall t, \quad (24)$$

where τ_{safe} is a constant value, which is set as 50; p_{obs} is the position of the obstacle. Safety constraints (24) ensure that the distance between the UAV and obstacles always remains greater than a specified safety threshold. Then, in the spirit of interior-point methods, we incorporate a logarithmic barrier function $\ln(-C_t)$ into the control objective function:

$$L(\xi_\theta, \theta) = \sum_{t=0}^{N-1} [l_t - \gamma \ln(-C_t)] \quad (25)$$

where γ is a barrier parameter, which controls the objective (25) to converge to the optimal $L^*(\xi_\theta, \theta)$. For the obstacle-avoidance task, we aim to find a trajectory, ξ_θ^* , that minimizes the given cost function l_t in (23), and satisfies the given safe constraint $C_t(x_t) \leq 0, t = 0, 1, 2, \dots, N$. Furthermore, if safety constraints are equality constraints, a quadratic barrier function can be employed. Thus, by incorporating barrier functions, the constrained optimization problem is transformed into an unconstrained optimization problem.

We set the position of a desired waypoint as $p^d = [2 \quad 1 \quad 1]^T \times 10^3$, the initial position of the UAV and obstacle as $p(t_0) = [0 \quad 0 \quad -100]^T$ and $p_{obs}(t_0) = [984.4 \quad 1396.43 \quad 390.1]^T$, and the velocity of the obstacle as $v_{obs} = [0 \quad -40 \quad 0]^T$. If obstacle avoidance measures are not taken, the conditions set above will cause the UAV to collide with obstacles at the collision point $p_c = [1001 \quad 452.6 \quad 388.2]^T$. We also compared the effects of two different settings of the barrier parameter $\gamma_1 = 0.1, \gamma_2 = 0.001$ on both constraint satisfaction and convergence speed.

As shown in the training results in Table 3, a smaller γ causes the constrained-optimization problem to approach the true solution of the original constrained system, meaning that the constraints become stricter, and constraint satisfaction accuracy improves as γ decreases. However, under the same learning rate, compared to a larger γ , a smaller γ significantly slows down the algorithm's convergence speed.

Table 3. Comparison of minimum value of (25) and convergence iterations with different γ .

Parameter Value	$\gamma_1 = 0.1$	$\gamma_2 = 0.001$
Minimum value of (25)	218	151
Convergence iterations	445	920

In Figure 10, the red line (UAV Trajectory 1) represents the trajectory generated by the policy trained with $\gamma = 0.001$, the yellow line (UAV Trajectory 2) represents the trajectory generated by the policy trained with $\gamma = 0.1$, the blue dashed line indicates the moving path of the obstacle, and the gray line shows the original trajectory of the UAV without obstacle avoidance maneuvers. It can be observed that both Trajectory 1 and Trajectory 2 successfully reach the way-point while avoiding the obstacle. The minimum distance between Trajectory 1 and the obstacle's path is 101.6 m, whereas the minimum distance

between Trajectory 2 and the obstacle's path is 55.4 m. This verifies that a smaller γ is guaranteed to satisfy more strict inequality constraints in (24).

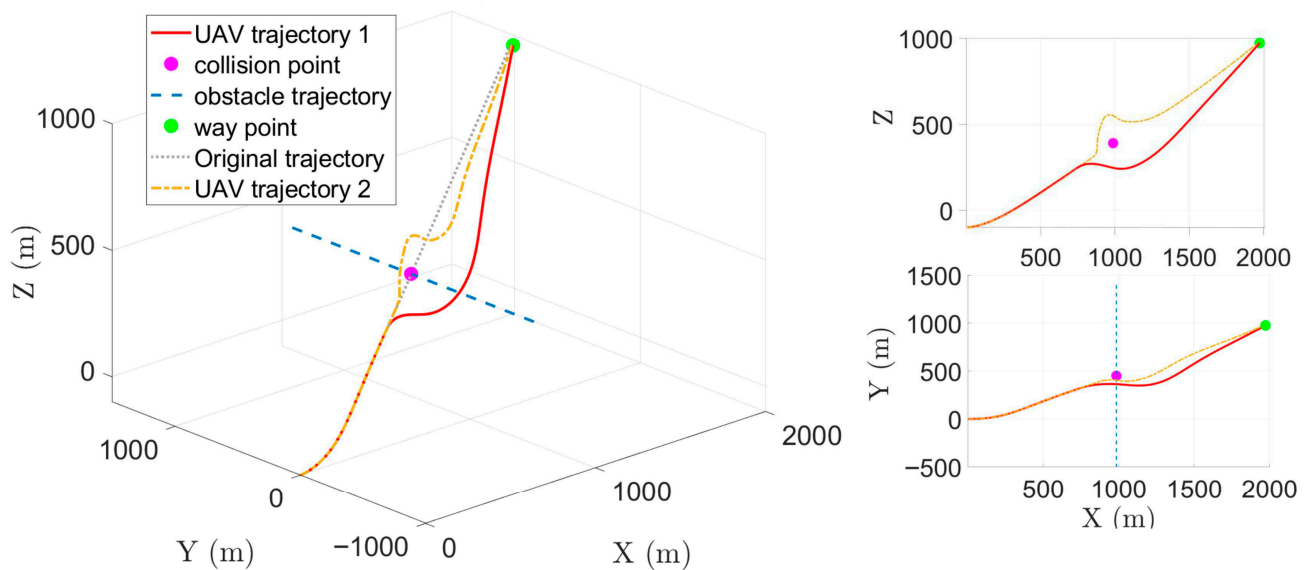


Figure 10. UAV obstacle avoidance trajectory.

Figures 11 and 12 show the detailed results of attitude regulation of the learned policy (the barrier parameter is set as $\gamma = 0.001$). At the 22 s mark of the simulation, the UAV initiates obstacle avoidance maneuvers, causing a brief divergence in attitude quaternions and angular velocity before subsequently converging back to zero. This indicates that after avoiding the obstacle, the UAV continues flying toward the waypoint with the desired attitude and angular velocity.

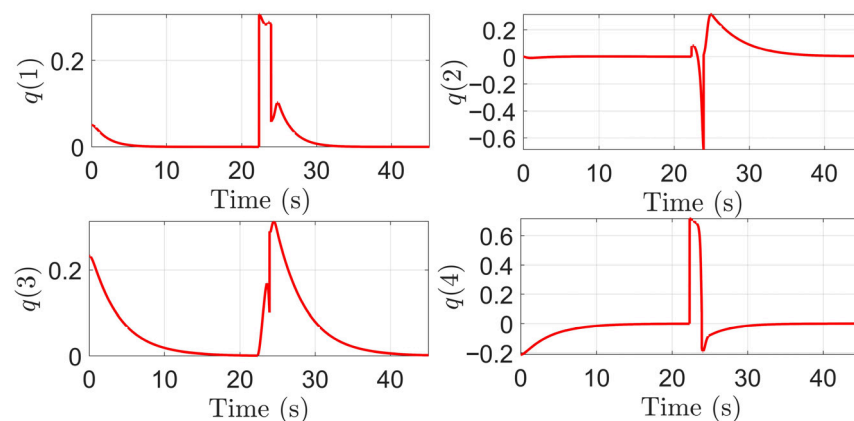


Figure 11. Attitude of UAV.

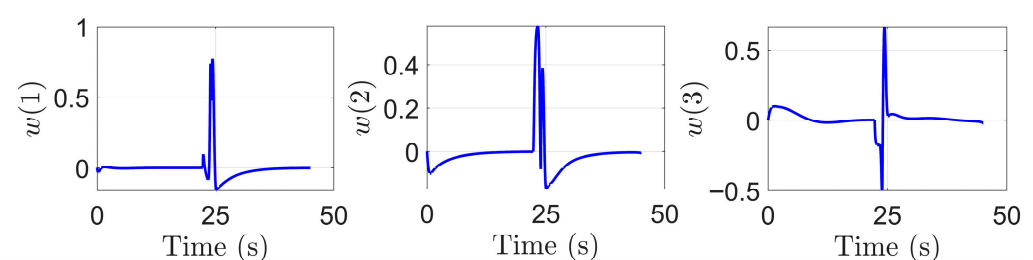


Figure 12. Angular velocity of UAV.

5. Conclusions

This paper proposes an end-to-end learning framework named EDMPC, whose innovation lies in integrating an optimization-driven inference step within the backward pass to compute gradient information. This approach offers notable advantages, including enhanced computational efficiency and improved accuracy in gradient estimation. To validate the method, we implemented it on an under-actuated UAV system with the objective of performing trajectory tracking or obstacle avoidance. The simulation results demonstrate the promising performance and highlight the framework's significant potential for incorporating safety constraints.

At the current stage, our work focuses on learning the control policy for the OC problem. However, this end-to-end learning framework can be easily extended to learn a dynamics model, or even an objective function. In addition, this work has only considered equality constraints (system dynamics) and inequality constraints (safety constraints). In the future, we will develop safety control algorithms that consider a wider variety of constraints in the OC problem. Methods for addressing safety constraints include, but are not limited to, control barrier functions. Subsequently, our main focus will be on developing safety control algorithms capable of practical deployment in real-world scenarios.

Author Contributions: Conceptualization, Y.Q. and J.H.; methodology, Y.Q. and J.H.; software, Y.Q. and J.H.; validation, Y.Q. and J.H.; formal analysis, Y.Q. and J.H.; investigation, R.Z.; resources, W.Y.; data curation, Y.Q. and J.H.; writing—original draft preparation, Y.Q.; writing—review and editing, J.H.; visualization, Y.Q. and J.H.; supervision, T.W.; funding acquisition, W.Y., Y.Q. and J.H. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Start-up Fund for New Talented Researchers of Nanjing Vocational University of Industry Technology (Grant No. YK24-01-04).

Data Availability Statement: The data are available from the corresponding author upon request.

Conflicts of Interest: Author Tianyi Wang was employed by the company Jiangnan Industrial Group Co., Ltd. Author Rui Zhang was employed by the company Inner Mongolia North Heavy Industries Group Limited Company Nanjing Institute. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DMPC	Differentiable Model Predictive Control
DOP	Differentiable Optimization Problem
EDMPC	Efficient Differentiable Model Predictive Control
UAV	Unmanned Aerial Vehicle
OC	Optimal Control
DP	Dynamic Programming
DDP	Differential Dynamic Programming
ILQR	iterative Linear Quadratic Regulation
QP	Quadratic Programs
KKT	Karush–Kuhn–Tucker

Appendix A

To prove Lemma 1, we need to show that the optimal condition of the reconstructed optimization problem (13) is exactly the differential optimal conditions in (11). To this end, we defined the following Hamiltonian for (13)

$$\bar{L} = \hat{L} + \text{Tr} \sum_{t=0}^{N-1} \Theta_t^\top (F_t^x \hat{x}_t + F_t^u \hat{u}_t + F_t) \quad (\text{A1})$$

where Θ_t is the dual variable for the reconstructed optimization problem (13). The optimal solution of (13) must satisfy the following conditions:

$$\begin{aligned} \frac{\partial \bar{L}_t}{\partial \Theta_t} &= F_t^x \hat{x}_t^\theta + F_t^u \hat{u}_t^\theta + F_t \\ \frac{\partial \bar{L}_t}{\partial \hat{x}_t^\theta} &= A_t \hat{x}_t^\theta + B_t \hat{u}_t^\theta + D_t + (F_t^x)^\top \Theta_{t+1} = \Theta_t \\ \frac{\partial \bar{L}_t}{\partial \hat{u}_t^\theta} &= E_t \hat{u}_t^\theta + (B_t)^\top \hat{x}_t^\theta + C_t + (F_t^u)^\top \Theta_{t+1} = \mathbf{0} \\ \Theta_N &= A_N \hat{x}_N^\theta + D_N, \hat{x}_0^\theta = \mathbf{0} \end{aligned} \quad (\text{A2})$$

The above optimal conditions are identical to the differential optimal conditions of (10), i.e., (11). This completes the proof.

Appendix B

From Lemma 1 and its proof, we get that the KKT conditions of (13) are exactly the differential KKT equations of (10). Now, we are to derive the recursive Riccati equations and the optimal control law given in Lemma 2. From (A2), we have

$$\hat{u}_t^\theta = -(E_t)^{-1} \left((B_t)^\top \hat{x}_t^\theta + C_t + (F_t^u)^\top \Theta_{t+1} \right). \quad (\text{A3})$$

Given the definition of $Q_t, \Lambda_t, R_t, M_t, N_t$ and (A3), we obtain the state evolution equation and costate evolution equation:

$$\begin{aligned} \hat{x}_{t+1}^\theta &= \Lambda_t^\theta \hat{x}_t^\theta - R_t \Theta_{t+1}^\theta + M_t, \quad t = 0, 1, \dots, N-1 \\ \Theta_t^\theta &= Q_t \hat{x}_t^\theta + \left(\Lambda_t^\theta \right)^\top \Theta_{t+1}^\theta + N_t, \quad t = 0, 1, \dots, N-1 \\ \Theta_N^\theta &= A_N \hat{x}_N^\theta + D_N, \end{aligned} \quad (\text{A4})$$

Next, based on the known structure of LQ problems (where the objective function is quadratic in the state, making its gradient linear), we conjecture the existence of P_t, W_t that means Θ_t^θ can be expressed as a linear function of the state $\hat{x}_t^\theta$, such that

$$\Theta_t^\theta = P_t \hat{x}_t^\theta + W_t \quad (\text{A5})$$

The relation (A5) is the form we aim to prove. When $t = N$, we have

$$\Theta_N^\theta = A_N \hat{x}_N^\theta + D_N \quad (\text{A6})$$

The relation (A5) holds if $P_N = A_N, W_N = D_N$. This matches the terminal conditions given in Lemma 2. Next, we prove that the relation (A5) also holds at time $t + 1$.

$$\Theta_{t+1}^\theta = P_{t+1} \hat{x}_{t+1}^\theta + W_{t+1} \quad (\text{A7})$$

Now, substitute (A7) into the expression of $\hat{x}_{t+1}^\theta$ in (A4)

$$\hat{x}_{t+1}^\theta = (I + R_t P_{t+1})^{-1} \left(\Lambda_t^\theta \hat{x}_t^\theta - R_t W_{t+1} + M_t \right) \quad (\text{A8})$$

And then, substitute (A7) into the expression of Θ_t^θ in (A4)

$$\Theta_t^\theta = Q_t \hat{x}_t^\theta + \left(\Lambda_t^\theta\right)^\top P_t \hat{x}_{t+1}^\theta + \left(\Lambda_t^\theta\right)^\top W_{t+1} + N_t \quad (\text{A9})$$

By substituting (A8) into (A9), we have

$$\begin{aligned} \Theta_t^\theta = & \left[Q_t + \left(\Lambda_t^\theta\right)^\top P_t (I + R_t P_{t+1})^{-1} \Lambda_t^\theta \right] \hat{x}_t^\theta + \\ & \left(\Lambda_t^\theta\right)^\top (I + R_t P_{t+1})^{-1} (W_{t+1} + P_{t+1} M_t) + N_t \end{aligned} \quad (\text{A10})$$

By mathematical induction, we have proven that the linear relationship of (A5) holds if P_t, W_t satisfy the expression given in (16). This completes the proof.

References

1. Yang, X.; Hu, Y.; Gao, H.; Ding, K.; Li, Z.; Zhu, P.; Sun, Y.; Liu, C. Risk-Aware Non-Myopic Motion Planner for Large-Scale Robotic Swarm Using CVaR Constraints. In *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: New York, NY, USA, 2024; pp. 5784–5790.
2. Dai, S.; Lee, T.; Bernstein, D.S. Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass. In *Proceedings of the 53rd IEEE Conference on Decision and Control*; IEEE: New York, NY, USA, 2014; pp. 6149–6154.
3. Baldi, S.; Roy, S.; Yang, K.; Liu, D. An under-actuated control system design for adaptive autopilot of fixed-wing drones. *IEEE/ASME Trans. Mechatron.* **2022**, *27*, 4045–4056. [\[CrossRef\]](#)
4. Perk, B.E.; Inalhan, G. Safe motion planning and learning for unmanned aerial systems. *Aerospace* **2022**, *9*, 56. [\[CrossRef\]](#)
5. Kumar, G.M.; Gramuglia, M.; L’Afflito, A. Output-Feedback Linearization with Applications to Quadcopter UAVs. In *Adaptive Control Theory and Applications*; IntechOpen: London, UK, 2024; Volume 49.
6. Yu, K.; Li, Y.; Lv, M.; Tong, S. Adaptive Backstepping Control for Under-actuated 6-DoF USV with Uncertain Dynamics. *IEEE Trans. Intell. Veh.* **2024**, *9*, 7476–7484. [\[CrossRef\]](#)
7. Emran, B.J.; Najjaran, H. Global tracking control of quadrotor based on adaptive dynamic surface control. *Int. J. Dyn. Control* **2021**, *9*, 240–256. [\[CrossRef\]](#)
8. Singh, P.; Giri, D.K.; Ghosh, A.K. Dynamic surface neuro-backstepping based flight control with asymmetric output constraints. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *59*, 3859–3870. [\[CrossRef\]](#)
9. Hu, Z.; Xiao, L.; Yi, W. A Generalized Super-Twisting Extended State Observer for Angle-Constrained Terminal Sliding Mode Guidance Law. *Aerospace* **2025**, *12*, 252. [\[CrossRef\]](#)
10. Ye, Z.; Zhang, D.; Deng, C.; Yan, H.; Feng, G. Finite-time resilient sliding mode control of nonlinear UMV systems subject to DoS attacks. *Automatica* **2023**, *156*, 111170. [\[CrossRef\]](#)
11. Wang, D.; Gao, N.; Liu, D.; Li, J.; Lewis, F.L. Recent progress in reinforcement learning and adaptive dynamic programming for advanced control applications. *IEEE/CAA J. Autom. Sin.* **2023**, *11*, 18–36. [\[CrossRef\]](#)
12. Almubarak, H.; Stachowicz, K.; Sadegh, N.; Theodorou, E.A. Safety embedded differential dynamic programming using discrete barrier states. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2755–2762. [\[CrossRef\]](#)
13. Ma, J.; Cheng, Z.; Zhang, X.; Lin, Z.; Lewis, F.L.; Lee, T.H. Local learning enabled iterative linear quadratic regulator for constrained trajectory planning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 5354–5365. [\[CrossRef\]](#)
14. Abu-Ali, M.; Berkel, F.; Manderla, M.; Reimann, S.; Kennel, R.; Abdelrahman, M. Deep learning-based long-horizon MPC: Robust, high performing, and computationally efficient control for PMSM drives. *IEEE Trans. Power Electron.* **2022**, *37*, 12486–12501. [\[CrossRef\]](#)
15. Bjelonic, M.; Grandia, R.; Geilinger, M.; Harley, O.; Medeiros, V.S.; Pajovic, V.; Hutter, M. Offline motion libraries and online MPC for advanced mobility skills. *Int. J. Robot. Res.* **2022**, *41*, 903–924. [\[CrossRef\]](#)
16. Ren, H.; Li, Y.; Wang, Y.; Chen, C.-K.; Yang, L.; Zhao, Y. Learning-based model predictive control for safe path planning and control. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2025**, *239*, 3991–4004. [\[CrossRef\]](#)
17. Berberich, J.; Allgöwer, F. An overview of systems-theoretic guarantees in data-driven model predictive control. *Annu. Rev. Control Robot. Auton. Syst.* **2025**, *8*, 77–100. [\[CrossRef\]](#)
18. Yan, H.; Qu, Y.; Li, R.; Li, W.; Chu, H.; Zhao, J.; Yu, G.; Ma, F.; Li, S.E.; Duan, J. Hierarchical MPC-based safe shared control for forward collision avoidance in collaborative driving. *Control Eng. Pract.* **2025**, *165*, 106581. [\[CrossRef\]](#)
19. Reske, A.; Carius, J.; Ma, Y.; Farshidian, F.; Hutter, M. Imitation learning from mpc for quadrupedal multi-gait control. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2021; pp. 5014–5020.

20. Amos, B.; Kolter, J.Z. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*; PMLR: New York, NY, USA, 2017; pp. 136–145.
21. Tamar, A.; Thomas, G.; Zhang, T.; Levine, S.; Abbeel, P. Learning from the hindsight plan—Episodic MPC improvement. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2017; pp. 336–343.
22. Williams, G.; Aldrich, A.; Theodorou, E.A. Model predictive path integral control: From theory to parallel computation. *J. Guid. Control Dyn.* **2017**, *40*, 344–357. [[CrossRef](#)]
23. Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; Kolter, J.Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation Inc.: San Diego, CA, USA, 2019; Volume 32, pp. 1234–1245.
24. Böttcher, L.; Antulov-Fantulin, N.; Asikis, T. AI Pontryagin or how artificial neural networks learn to control dynamical systems. *Nat. Commun.* **2022**, *13*, 333. [[CrossRef](#)]
25. Amos, B.; Jimenez, I.; Sacks, J.; Boots, B.; Kolter, J.Z. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation Inc.: San Diego, CA, USA, 2018; Volume 31.
26. Jin, W.; Wang, Z.; Yang, Z.; Mou, S. Pontryagin differentiable programming: An end-to-end learning and control framework. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation Inc.: San Diego, CA, USA, 2020; Volume 33, pp. 7979–7992.
27. Jin, W.; Mou, S.; Pappas, G.J. Safe pontryagin differentiable programming. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation Inc.: San Diego, CA, USA, 2021; Volume 34, pp. 16034–16050.
28. Kong, L.; Cui, J.; Zhuang, Y.; Feng, R.; Prakash, B.A.; Zhang, C. End-to-end stochastic optimization with energy-based model. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation Inc.: San Diego, CA, USA, 2022; Volume 35, pp. 11341–11354.
29. Elhesasy, M.; Dief, T.N.; Atallah, M.; Okasha, M.; Kamra, M.M.; Yoshida, S.; Rushdi, M.A. Non-linear model predictive control using CasADi package for trajectory tracking of quadrotor. *Energies* **2023**, *16*, 2143. [[CrossRef](#)]
30. Pytlak, R. *Numerical Methods for Optimal Control Problems with State Constraints*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1999.
31. Athans, M. The matrix minimum principle. *Inf. Control* **1967**, *11*, 592–606. [[CrossRef](#)]
32. Oland, E.; Schlanbusch, R.; Kristiansen, R. Under-actuated waypoint tracking of a fixed-wing UAV. *IFAC Proc. Vol.* **2013**, *46*, 126–133.
33. Devalla, V.; Mondal, A.K.; Prakash, A.A.J.; Prateek, M.; Prakash, O. Guidance, navigation and control of a powered parafoil aerial vehicle. *Curr. Sci.* **2016**, *111*, 1045–1054. [[CrossRef](#)]
34. Biegler, L.T.; Zavala, V.M. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Comput. Chem. Eng.* **2009**, *33*, 575–582. [[CrossRef](#)]

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.