Model-based deep reinforcement learning for fixed-wing UAV attitude control with prior physics knowledge

Titouan Guerin, *Sorbonne Université, ONERA*, Olivier Sigaud, *Sorbonne Université*, Pierre Fournier, *ONERA* and Julien Marzat, *ONERA*

Abstract

Fixed-wing UAVs are efficient for long-range missions, but their attitude dynamics grow strongly nonlinear and coupled away from level flight, making them harder to control than the quadrotors that most of the literature targets. Reinforcement learning can learn controllers directly from data, but model-free RL requires a large number of costly simulator interactions. Model-based RL instead learns a dynamics model to train on, which can be made more reliable by embedding known aerodynamic physics as a prior. However, this assumes the prior is accurate, which is rarely the case. In this work we extend PhIHP, a physics-informed model-based RL method validated only on classic control benchmarks, to fixed-wing UAV attitude control. These test environments have dynamics that are low-dimensional and exactly known, and we test whether the approach holds on a system whose aerodynamics are neither. We build a physics prior for a simulated Skywalker X8 fixed-wing UAV, paired with a learned residual that corrects what the prior misses. We evaluate controllers on settled attitude tracking error: the steady-state deviation, in degrees, between commanded and achieved roll and pitch. First, we show that a policy trained purely on trajectories imagined through this dynamics model, with no further simulator interaction, comes close to the tracking accuracy of a policy trained directly on the simulator, and beats it when the environment is parallelized. Second, we show that when the prior is inaccurate, tracking error rises sharply in non-nominal flying conditions, but jointly optimizing the prior's parameters alongside the residual recovers 37-38% of this loss on the hardest targets. Third, combining the trained policy with a short-horizon planner into a Hybrid Controller reduces tracking errors by up to 43% over the policy alone. The modifications we introduce depend only on having a physics prior available, not on the specifics of our system.

I. Introduction

Fixed-wing unmanned aerial vehicles (UAVs) support tasks such as search and rescue, environmental monitoring, or aerial photography, many of which require operating through turbulence, or under aggressive attitude commands [1]. This demands accurate attitude control across the full flight envelope, not just the calm, near-level conditions autopilots are usually designed for. Near trim, the equilibrium condition in which aerodynamic and gravitational forces and moments balance, roll, pitch, and airspeed are only weakly coupled and close to linear. At larger angles of attack and sideslip, lift and drag turn nonlinear and the axes couple strongly, with wind and turbulence adding further disturbance.

Classical autopilots sidestep this rather than resolve it. Attitude control is typically implemented as cascaded, single-axis PID or gain-scheduled loops that assume decoupled, near-linear dynamics, valid only near the trim condition they were tuned around [2]. This keeps the controller simple, but forces flight to stay within conservative margins that avoid the aggressive, strongly coupled maneuvers these tasks would benefit from. Instead of working with a fixed physical model, an alternative is to learn a controller directly from data. Reinforcement learning (RL) has proven to be a solid approach to this problem, allowing a control policy to be optimized directly against a performance objective through interaction with the system [3].

Most learning-based control literature targets quadrotor UAVs rather than fixed-wing aircraft. Quadrotors are mechanically simpler, can hover, and are cheaper to prototype and test, which makes them a more

common and more forgiving research platform [4], [5]. Fixed-wing aircraft, in contrast, are more energy efficient and better suited to long-endurance and long-range missions, but their dynamics are more strongly coupled and harder to control, and the fixed-wing control literature is comparatively small [6]. This work focuses on fixed-wing UAV control.

Model-Free Reinforcement Learning (MFRL) learns control agents without requiring an accurate model of the system's dynamics, but requires a huge number of samples to learn a decent controller. In contrast, Model-Based Reinforcement Learning (MBRL) learns a model of the environment's dynamics and uses it to train a policy in a much more sample-efficient way. When some knowledge of the system's physics is already available, MBRL can exploit it directly, combining the flexibility of a learned model with the structure and reliability of known physical laws to produce more robust solutions than a purely data-driven approach alone [7].

This work builds directly on APHYNITY [8] and its extension to control, PhIHP [9]. APHYNITY targets autonomous physical systems with no external control input, such as reaction-diffusion processes and wave propagation. It decomposes the system's dynamics into a fixed physics-based prior and a minimal-norm residual learned on the evaluation environment, which only accounts for what the prior cannot explain. Models of this form, combining an interpretable physical component with a learned black-box correction, are commonly referred to as grey-box models. PhIHP extends this decomposition to controlled systems, that is, to a transition function that depends on a state and the action of an agent. It trains a policy entirely on trajectories imagined through the resulting dynamics model.

Using a physics prior in this way has been shown to increase robustness and generalization compared to purely data-driven dynamics models, while also requiring substantially less training data. Indeed, the prior already accounts for a large part of the system's behavior before any data is collected [10], [8]. However, this benefit assumes the prior itself is reasonably accurate. In practice, this is a strong assumption: physical models for a given vehicle are rarely known exactly, particularly for a new or lightweight airframe. Banerjee et al. identify the choice of physics prior as an open problem in physics-informed RL, noting that it is difficult, requires extensive system-specific study, and rarely generalizes across tasks [11].

In [9], PhIHP was applied to classic control benchmarks (Cartpole, Pendulum, Acrobot [12]) where the equations of the system are well known and the dynamics simple. In this paper, we investigate the capability of the PhIHP hybrid planning method, which we refer to as the Hybrid Controller, to address the more challenging context of fixed-wing UAVs. Our contributions are the following:

- We show that policies trained entirely in imagination (on trajectories generated by our dynamics model) come close to the asymptotic performance of policies trained directly on the simulator. Because the dynamics model is cheap to instantiate, the model environment can also be run in parallel, which the simulator does not easily allow. This makes policies converge faster and yields the best asymptotic performance, even compared to policies trained in the simulator. (Section VI-A).

- We show that a dynamics model learned from a biased or incomplete physics prior can still be used to train a policy of comparable quality, provided the parameters of the prior are optimized jointly with the residual. A degraded, frozen prior leads to substantially worse policies, but tuning it recovers most of this lost performance, with the largest gains on the hardest targets (by 37-38%, see Section VI-B).

- We show that the Hybrid Controller improves tracking overall, and most of all on targets that are hard to reach, away from nominal flight conditions (by up to 43% in the best conditions). We additionally show that this gain comes almost entirely from one of the two learned components the RL agent

contributes to the planner, rather than from both (Section VI-C).

II. Related Work

RL for fixed-wing UAV control: RL has emerged as a promising approach for control tasks that are difficult to specify or solve with classical methods, since it learns a control policy directly from interaction with the system rather than from a hand-derived model [13], [14]. This has led to RL being applied across a wide range of robotic control problems, including UAV flight control. Within fixed-wing UAV control specifically, RL has been used for attitude control and altitude hold, generally reporting performance competitive with or better than classical PID baselines under disturbances such as wind and turbulence [15], [16], [17]. Other fixed-wing RL studies target different control tasks, such as path following or hybrid fixed-wing/multicopter control, without necessarily evaluating performance under wind or turbulence disturbances [18], [19], [20], [21]. Richter et al. [6] review this body of work in detail. Overall, these studies show RL is a viable alternative to classical fixed-wing control, though most contributions are still evaluated only in simulation.

Control strategies: Control strategies within this literature range from purely Model-Free Reinforcement Learning (MFRL) to hybrid schemes that combine RL with classical control, to Model-Based Reinforcement Learning (MBRL). MFRL approaches learn a policy directly from environment interaction without an explicit dynamics model [15], [18]. Other work retains classical control in the loop, using RL to replace or compensate only part of the controller, for example an inner PID loop [16], [19], or, more broadly, a classical low-level controller alongside a learned high-level policy [5]. MBRL approaches instead learn a dynamics model and use it for training or planning, and have been shown to improve tracking accuracy and robustness over MFRL methods on fixed-wing attitude control [17]. The present work follows the MBRL direction.

Physics-informed dynamics modeling: A central challenge in MBRL is learning a dynamics model that is accurate enough for training or planning over long horizons. Some methods learn such dynamics models entirely in a latent space without considering physics at all, such as TD-MPC [22], [23]. Physics-informed approaches instead address this by combining an analytical or partial physical model with a data-driven residual that only corrects what the physical model does not capture, rather than learning the transition function purely from data. This grey-box formulation, sitting between fully interpretable white-box physical models and purely data-driven black-box ones, has been explored in several forms [8], [24], [25], [26]. A similar physics-plus-residual idea has been applied to quadrotor UAVs, though with a different modeling approach: Torrente et al. augment a nominal quadrotor dynamics model with a Gaussian-process residual within a Model Predictive Control (MPC) pipeline [4], which is methodologically distinct from the neural-network residual formulations above but highly relevant to the present work's physics-plus-residual philosophy. Also on quadrotor UAVs, Osaka et al. show that this physics-plus-residual structure is more robust to domain shift than a purely data-driven model, since the structure of the physics component remains valid even when its parameters change between source and target domains, allowing a hybrid model to adapt to new flight conditions with substantially less target-domain data [10].

PhIHP uses this grey-box modeling approach to learn a dynamics model which is then used to learn policies through trajectories generated from this model. However, training purely in imagination is nonetheless difficult, since prediction errors compound over a rollout. Most MBRL methods therefore restrict themselves to short rollouts of a few steps, often relying on model ensembles for stability [27], [28]. Overall, physics-informed models are known to generalize better than purely data-driven ones [29]. Banerjee et al. [11] survey this broader physics-informed RL landscape. This line of work has been validated mainly on classic control benchmarks and quadrotor flight, but not on fixed-wing UAVs, which

we are tackling in this work.

Regularizing the residual: A separate design question within this framework is how the data-driven residual should be regularized relative to the prior, so that it corrects the physics model rather than overwriting it. APHYNITY addresses this by selecting a minimal-norm residual, so that the data-driven component only accounts for what the prior cannot explain [8]. Osaka et al. compare norm-based and correlation-based regularizers for this purpose [25], and Kübel et al. instead propose an energy-based regularizer, penalizing changes in the predicted system's kinetic and potential energy relative to the physics prior alone; they show this improves both tracking accuracy and closed-loop flight stability [30]. In our work, we follow the regularization technique introduced in APHYNITY.

Physics prior quality and tunability:
A separate question is whether the prior should be fixed at all. APHYNITY optimizes a subset of the prior coefficients jointly with the residual, which improves forecasting accuracy across several levels of physical model incompleteness [8]. PhIHP does not take up this option: its prior stays fixed, and its contribution is instead to show that a policy trained purely in imagination on a physics-informed model matches training on real data, whereas the same policy trained on a fully data-driven model does not [9]. Neither work asks whether tuning an inaccurate prior, which APHYNITY shows improves forecasting, also yields a dynamics model good enough to train a high-performing policy, which is what PhIHP evaluates.

Planning at inference time: Several MBRL methods combine a learned policy with online planning through the learned model rather than relying purely on the trained policy, using MPC to optimize a sequence of actions over a short horizon. Different optimization strategies have been proposed for this MPC step, including Cross-Entropy Method-based MPC (CEM-MPC) [31], [27], [32] and Model Predictive Path Integral-based MPC (MPPI-MPC) [23]. We follow the CEM-MPC approach used by PhIHP [9] and RT-HCP [33], combining sampled action sequences with a learned policy and value function to guide the search and reduce planning cost. Physics-informed dynamics models combined with MPC-based planning and RL have proven effective when validated on simpler control problems. PhIHP is evaluated on Pendulum, Cartpole, and Acrobot, along with their swingup variants from the Gymnasium classic control suite [12]. We apply this method to a more difficult problem.

III. Background

A. RL and POMDPs

Reinforcement Learning (RL) addresses the problem of an agent learning a policy π to maximize the expected discounted return $\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$ through trial-and-error interactions with an environment [34], where $\gamma \in [0,1)$ is the discount factor. In RL, the environment is typically formalized as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, $T(s_{t+1} \mid s_t, a_t)$ is the transition dynamics, and $r(s_t, a_t)$ is the reward function. The Markov property holds as long as $P(s_{t+1} \mid s_{1\ldots t}, a_{1\ldots t}) = P(s_{t+1} \mid s_t, a_t)$.

When the environment dynamics (T and r) are unknown, the agent must rely on MFRL algorithms to learn directly from experience samples [34]. MFRL algorithms are broadly categorized into:

- Value-based methods: learn an action-value function $Q(s,a)$ estimating expected returns (e.g., Q-learning) and use it directly to act [35].
- Policy-based methods: directly optimize the parameters of a parameterized policy $\pi_\phi(a \mid s)$ using policy gradients [36], [13].
- Actor-Critic methods: combine both paradigms by using an actor to propose actions and a critic to evaluate them via a learned value function, significantly reducing gradient variance [37], [38].

In our work, we specifically work with TD3 [14], which is an off-policy actor-critic algorithm for continuous action spaces, extending DDPG [37] with three changes that reduce value overestimation and stabilize training. The actor is a deterministic policy $\pi_\phi(o)$ mapping an observation to an action, trained to maximize the return estimated by a critic. TD3 trains two independent critics, Q_{θ_1} and Q_{θ_2}, and takes their minimum to form the temporal-difference target,

$$y = r + \gamma \min_{j\in\{1,2\}} Q_{\theta'_j}\big(o', \pi_{\phi'}(o') + \epsilon\big), \qquad \epsilon \sim \mathrm{clip}(\mathcal{N}(0,\sigma), -c, c), \tag{1}$$

where θ'_j and ϕ' are target networks updated slowly toward θ_j and ϕ, and the clipped noise ϵ smooths the target policy so the critics cannot exploit sharp, incorrect peaks in the estimated value.

Each critic minimizes the squared error to this target,

$$\mathcal{L}(\theta_j) = \mathbb{E}\left[\big(Q_{\theta_j}(o,a) - y\big)^2\right], \qquad j \in \{1,2\}, \tag{2}$$

and the actor is updated less often, once every d critic updates, to follow the gradient of actions with respect to the first critic,

$$\mathcal{L}(\phi) = -\mathbb{E}\left[Q_{\theta_1}\big(o, \pi_\phi(o)\big)\right]. \tag{3}$$

When modeling an agent acting in a real world environment as a MDP, the agent often does not perceive the full state or which the Markov Property holds, but only an observation. This setting is formalized as a Partially Observable MDP (POMDP) [39]. The MDP tuple is extended with an observation space $\mathcal{O}$ and an observation function $\Omega(o_t \mid s_t, a_t)$, giving $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, r, \gamma)$. At each timestep the agent receives the observation $o_t \in \mathcal{O}$ instead of the true state s_t.

B. FW-UAV environment

Reinforcement learning for aircraft control is generally carried out in flight simulators, which numerically integrate an aircraft's equations of motion under configurable flight conditions. We use JSBSim, an open-source flight dynamics simulator, made compatible with the Gymnasium interface through the FW-JSBGym library [17]. Simulators of this kind support a wide range of aircraft, fixed-wing among them, and let the user configure the surrounding flight conditions: the atmosphere model, steady wind, turbulence, and wind gusts. In this work we control a fixed-wing UAV, a Skywalker X8 [40], in windless, turbulence-free air with constant airflow, so that the difficulty of the control problem comes from the aircraft's own dynamics rather than from external disturbances.

Several control tasks are studied on fixed-wing UAVs, most commonly attitude control and waypoint tracking. This work focuses on attitude tracking: driving the aircraft's roll and pitch to a commanded reference and holding it there.

A simulator of this kind maintains a much richer internal state than the agent is given access to, including position, altitude, and propulsion variables that the task does not expose. The agent therefore acts on a partial view of the simulator state, and the environment is formally a POMDP, as defined in Section III-A. The full technical description of the environment is available in Section V-A.

C. Physics Prior of a FW-UAV

In this work, we aim to learn the underlying dynamics of the FW-UAV environment within our model-based framework. Rather than relying entirely on a black-box data-driven model to predict system transitions, we construct an analytical physics prior $F_p^{\theta_p}$ that models the primary flight mechanics and

attitude dynamics of the Skywalker X8.

Let ϕ be the roll angle, θ the pitch angle, V_a the airspeed, $\boldsymbol{\omega}^b = [p, q, r]^T$ the vector of angular rates, α the angle of attack, and β the sideslip angle. Together these form the physical state

$$\boldsymbol{s}^p = [\phi, \theta, V_a, \boldsymbol{\omega}^b, \alpha, \beta]^T \in \mathbb{R}^8, \tag{4}$$

the eight variables the prior operates on.

The prior takes as input this physical state together with the actuator state z, which describes the current position and rate of the control surfaces and throttle. The commanded action is not used directly: the control surfaces take time to reach a commanded deflection, so what acts on the airframe at a given instant is the actual deflection described by z, not the command that produced it. The dynamics of the environment are then given by

$$\dot{\boldsymbol{s}}^p = F_p^{\theta_p}(\boldsymbol{s}^p, z), \tag{5}$$

where θ_p denotes the constants the prior depends on: the aerodynamic coefficients of the airframe together with the inertia terms, listed in Table VI of Appendix B. These are measured or estimated once and are therefore only approximate. They can be tuned, and they form a different set from the parameters θ_a of the residual network learned alongside the prior in Section IV-A. The dynamics expand component-wise as

$$\dot{\boldsymbol{s}}^p = \left[\dot{\phi}, \dot{\theta}, \dot{V}_a, \dot{\boldsymbol{\omega}}^b, \dot{\alpha}, \dot{\beta}\right]^T. \tag{6}$$

Each component of $\boldsymbol{s}^p$ is defined below.

$$\dot{\phi} = p + \sin(\phi)\tan(\theta)q + \cos(\phi)\tan(\theta)r \tag{7}$$

$$\dot{\theta} = \cos(\phi)q - \sin(\phi)r \tag{8}$$

$$\dot{V}_a = \frac{dV_a}{dt} = \frac{d}{dt}\sqrt{u_r^2 + v_r^2 + w_r^2} = \frac{u_r\dot{u}_r + v_r\dot{v}_r + w_r\dot{w}_r}{V_a} \tag{9}$$

$$\dot{\boldsymbol{\omega}}^b = \begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n\\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{1}{J_y}m\\ \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_8 n\end{bmatrix} \tag{10}$$

$$\dot{\alpha} = \frac{d\alpha}{dt} = \frac{d}{dt}\tan^{-1}\left(\frac{w_r}{u_r}\right) = \frac{u_r\dot{w}_r - w_r\dot{u}_r}{u_r^2 + w_r^2} \tag{11}$$

$$\dot{\beta} = \frac{d\beta}{dt} = \frac{d}{dt}\sin^{-1}\left(\frac{v_r}{V_a}\right) = \frac{V_a\dot{v}_r - v_r\dot{V}_a}{V_a\sqrt{V_a^2 - v_r^2}} \tag{12}$$

These eight variables constitute the full physics model. The full detailed version of the calculations is available in Appendix A.

This prior is an approximation of the simulator (evaluation environment) it is meant to describe. Lift and drag are modelled as linear in the angle of attack, so the prior does not capture stall or the nonlinear behaviour that appears at large aerodynamic angles, and higher-order and coupling effects present in the simulator's own aerodynamic tables are absent. These are precisely the effects the residual is expected to absorb.

D. CEM-MPC Method

MPC is a model-based control framework that optimizes action sequences over a finite prediction horizon using an internal dynamics model. Unlike purely reactive model-free approaches, MPC explicitly evaluates future trajectories to select immediate actions that optimize performance. To find the optimal action sequence, various optimization techniques can be employed depending on the differentiability of the dynamics model, ranging from gradient-based methods to derivative-free stochastic optimizers such as the Cross-Entropy Method (CEM).

At each control step t, given an estimated environment dynamics model $\hat{\mathcal{T}}_\theta$, current state s_t, and planning horizon H, MPC solves a finite-horizon optimal control problem to find the optimal sequence of actions $A^\star = (a_t^\star, a_{t+1}^\star, \ldots, a_{t+H-1}^\star)$ that maximizes expected predicted return:

$$A^\star = \underset{a_{t:t+H-1}}{\arg\max} \sum_{k=0}^{H-1} \gamma^k r(s_{t+k}, a_{t+k}), \quad \text{s.t. } s_{t+k+1} = \hat{\mathcal{T}}_\theta(s_{t+k}, a_{t+k}). \tag{13}$$

Upon solving for $A^\star$, MPC executes only the first action $a_t^\star$ in the environment, discards the rest, and replans at the next timestep.

In our work, we employ the derivative-free Cross-Entropy Method (CEM), referred to as CEM-MPC in this case, to search for $A^\star$. CEM-MPC maintains a sampling distribution over candidate action sequences, initialized as a Gaussian $\mathcal{N}(\mu, \sigma^2)$. At each optimization iteration, CEM-MPC samples N candidate action sequences, rolls each sequence out through the dynamics model to evaluate its return, and selects the K top-performing candidates, which we call elite sequences. The Gaussian parameters μ and σ^2 are then updated to fit these elites, and this process repeats for I iterations. The first action of the final converged mean sequence μ is executed in the environment. We refer to the CEM-MPC budget as the population size N times the number of iterations I, the main cost during inference time for CEM-based planners.

E. APHYNITY and PhIHP

Given a physical system whose dynamics follow an autonomous ODE/PDE with no external control term,

$$\dot{X}_t = F(X_t), \tag{14}$$

APHYNITY decomposes the unknown transition function as $F = F_p + F_a$, where F_p is an analytic prior encoding partial physical knowledge and F_a is a data-driven residual correcting what F_p does not capture. Each component has learnable parameters.

$$\frac{dX_t}{dt} = F(X_t) = (F_p^{\theta_p} + F_a^{\theta_a})(X_t). \tag{15}$$

Since this decomposition is not unique in general (because infinitely many residuals can reproduce the observed dynamics for a given prior), APHYNITY selects the pair $(F_p^{\theta_p}, F_a^{\theta_a})$ with minimal-norm residual, formulated as the constrained optimization problem

$$\min_{F_p^{\theta_p} \in \mathcal{F}_p,\, F_a^{\theta_a} \in \mathcal{F}} \|F_a\| \quad \text{s.t.} \quad \forall X \in \mathcal{D},\ \forall t,\ \frac{dX_t}{dt} = (F_p^{\theta_p} + F_a^{\theta_a})(X_t), \tag{16}$$

where $\mathcal{D}$ is the set of observed trajectories from the system being modeled (a physical process or a simulated environment). This ensures F_p explains as much of the dynamics as possible while F_a only accounts for what F_p cannot, which is solved in practice via an adaptive augmented-Lagrangian scheme.

PhIHP extends APHYNITY to MBRL by decomposing the transition function of a controlled system as $\hat{\mathcal{T}}_\theta = F_p^{\theta_p} + F_a^{\theta_a}$, where F_p is a known analytic prior and F_a a data-driven residual. We specify in equation

(17) that F_p has a set of trainable parameters (it is not specified in their work). Unlike APHYNITY, the dynamics are conditioned on an action a_t, and the system is described by

$$\left.\frac{ds_t}{dt}\right|_{t=t_0} = \hat{\mathcal{T}}_\theta(s_{t_0}, a_{t_0}) = (F_p^{\theta_p} + F_a^{\theta_a})(s_{t_0}, a_{t_0}), \tag{17}$$

with the next state obtained via $s_{t+1} \simeq \text{ODESolve}(s_t, a_t, \hat{\mathcal{T}}_\theta, t, t+\Delta t)$. Each predicted state is fed back as input to produce the next, integrated with fixed-step RK4, chained over n steps to give a rollout $\hat{s}_1^p, \ldots, \hat{s}_n^p$

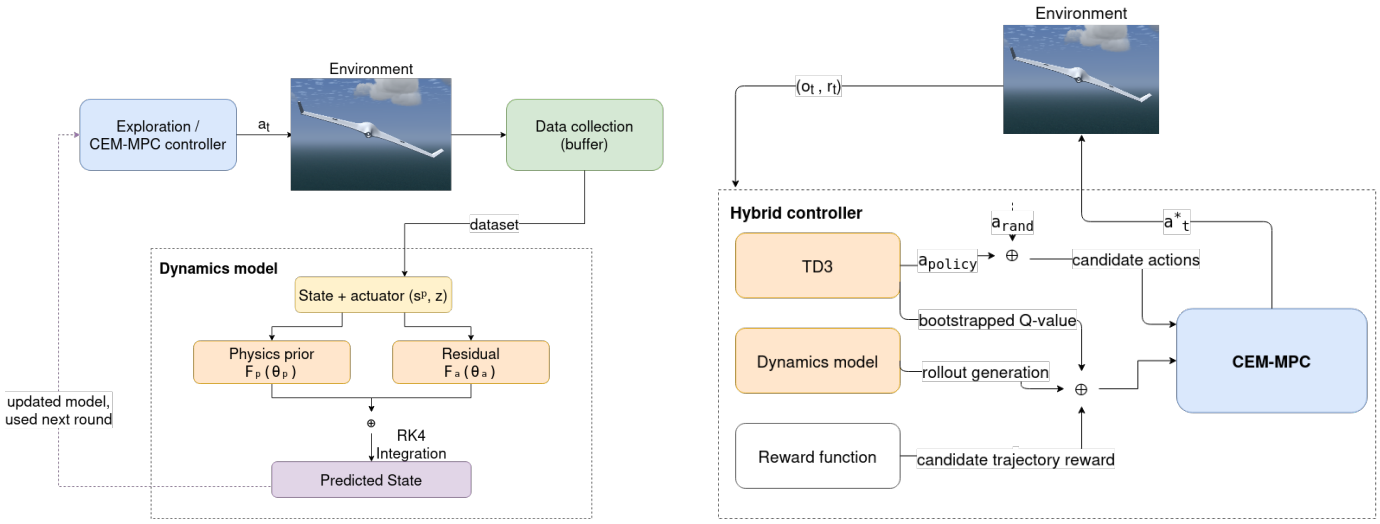
The prediction loss $\mathcal{L}$ used by these methods is:

$$\mathcal{L} = \frac{1}{n}\sum_{t=1}^{n} \left\|\hat{s}_t^p - s_t^p\right\|^2, \tag{18}$$

where $\hat{s}_t^p$ is a predicted state at a time t, and s_t^p is the ground truth state at time t.

IV. Methods

Our method proceeds in three stages, two of which are illustrated in Figure 1. We first learn a physics-informed dynamics model from data collected in the simulator (Section IV-A). This model then replaces the simulator entirely, and a TD3 policy is trained on trajectories imagined in the model (Section IV-B). Finally, at inference time, the trained policy and critic are combined with the dynamics model inside a CEM-MPC planner to make the Hybrid Controller (Section IV-C). Overall, we follow the training procedure of the dynamics model and inference method as in PhIHP, but we adapt it to work better for our setting and specify when that is the case.



(a) Learning the dynamics model.

(b) Control at inference time.

Fig. 1: **Overview of the method.** (a) The dynamics model is trained over several rounds: a controller explores the environment, the collected trajectories are added to a growing buffer, and the model, a physics prior combined with a learned residual, is retrained from scratch on the full buffer each round. The updated model is then used to explore in the following round. (b) At inference time, the hybrid controller scores candidate action sequences, drawn both randomly and from the trained policy, by rolling them through the dynamics model and evaluating them with the reward function and the bootstrapped critic.

A. Learning the Dynamics Model

The dynamics model is split into two components sharing the same input, the physical state s_t^p and the actuator state z_t, and both predicting the rate of change of the physical state: a physics prior $F_p^{\theta_p}$ and a residual $F_a^{\theta_a}$, combined as

$$\dot{\hat{s}}_t^p = \hat{\mathcal{T}}_\theta(s_t^p, z_t) = F_p^{\theta_p}(s_t^p, z_t) + F_a^{\theta_a}(s_t^p, z_t). \tag{19}$$

Data collection: As in PhIHP, we collect training data iteratively over multiple rounds: the dataset grows during training, rather than training on a fixed one from the start. The first round consists of purely random exploration. Every following round instead selects actions by planning with CEM-MPC (using the last trained dynamics model), with a decaying probability of taking a random action instead, to preserve some exploration during the first few rounds. Each round consists of 25 episodes of 1000 steps, each with its own attitude target.

Our setting departs from PhIHP here, since our task is goal-conditioned: the model must be accurate across the whole range of attitude targets it will later be asked to reach, not just along the trajectories the planner happens to visit. Drawing targets independently at random leaves this to chance, and with only a handful of rounds, a round can easily contain no target in the hard corners of the flight envelope. We therefore stratify the sampling: the roll-pitch envelope is divided into 5 bins each, and every bin is visited once before any is repeated, with the target drawn uniformly within the selected bin. This leaves the target distribution unchanged and only reduces its variance, while guaranteeing that every round covers the full envelope.

After each round, the model is retrained from scratch on the full cumulative dataset collected so far, rather than fine-tuned on the new data alone. Fine-tuning would let the model keep whatever bias it picked up while the dataset was still small and unrepresentative, therefore retraining from scratch on the growing dataset each round avoids compounding that bias round after round.

Autoregressive prediction and rollout supervision: Within each round, the model is trained on n-step rollouts rather than single-step transitions. Predictions are obtained by applying the dynamics model autoregressively, like explained in Section III-E. Supervising the loss over the full rollout, rather than one step at a time, penalizes the compounding integration error that accumulates over a long rollout, an argument made explicitly by APHYNITY [8]. This matters in our setting since training a policy purely in imagination requires the model to stay accurate for the length of a full episode, which can take hundreds of steps to reach a target attitude from a neutral starting position.

Averaged over the rollout, our prediction loss varies slightly compared to equation (18):

$$\mathcal{L} = \frac{1}{n}\sum_{t=1}^{n} \left\| \frac{\hat{s}_t^p - s_t^p}{x_c} \right\|^2, \tag{20}$$

we add a normalization term x_c. It is a fixed, per-dimension reference scale.

Physics-derived normalization: Each dimension of s^p has a different physical unit and scale: airspeed in m/s, angular rates in rad/s, angles of order one. Under a plain squared error, whichever dimension happens to have the largest raw magnitude would dominate the loss, regardless of how much it actually matters for control. We correct for this with a fixed reference

$$x_c = \left[1,\ 1,\ V_0,\ \frac{2V_0}{b},\ \frac{2V_0}{\bar{c}},\ \frac{2V_0}{b},\ 1,\ 1\right], \tag{21}$$

$V_0, \bar{c}$ and b are respectively the trim airspeed, wingspan and mean chord. Rather than computing this scale from data, for instance from the variance of the states visited so far, which would depend on what the exploration policy happened to visit and would shift between rounds, we reuse the same reference scales

already used to define the aerodynamic coefficients of the prior (see Appendix A). Since our approach is already grounded in known physics, we lean on that same physics to normalize the loss, rather than on the data. Setting $x_c = 1$ everywhere recovers the non-normalized APHYNITY/PhIHP loss.

This is not specific to fixed-wing aerodynamics. The entries of x_c above are particular to this airframe, but the rule that produces them is not. Any physics prior is written in terms of characteristic scales for the quantities it models, and those scales are what make its coefficients dimensionless in the first place. Whenever a prior is available, it therefore already carries the reference needed to normalize a loss over the same state variables, and x_c can be read off it directly. We designed the normalization this way deliberately, so that moving to a different vehicle, task, or simulator requires only reading the new prior rather than retuning anything. The construction requires no data and no tuning.

Residual constraint: As specified in Section III-E, the decomposition $\hat{\mathcal{T}}_\theta = F_p^{\theta_p} + F_a^{\theta_a}$ is not unique, so we enforce F_a minimal via an augmented-Lagrangian penalty. For a batch of trajectories, each consisting of (s^p, z) pairs at every timestep, we define the norm of a component $\Phi \in \{F_p^{\theta_p}, F_a^{\theta_a}\}$ as its root-mean-square (scaled) magnitude over the N resulting points in the batch,

$$\|\Phi\| = \left(\frac{1}{N} \sum_{i=1}^{N} \left\| \frac{\Phi(s_i^p, z_i)}{x_c} \right\|_2^2 \right)^{1/2}, \tag{22}$$

which is an empirical norm over the current training batch. One new addition from the PhIHP formula is that rather than penalizing $\|F_a\|$ alone, we penalize the ratio $\|F_a\|/\|F_p\|$. The magnitude of F_p is not constant across the flight envelope, being larger in aggressive maneuvers and smaller near trim, so a fixed absolute penalty would tolerate a larger residual in one regime than another for no principled reason. This ratio instead keeps the penalty meaningful relative to the magnitude of the prior itself at each point in the flight envelope, and, unlike the normalization introduced next, is recomputed every step.

We introduce a second, independent normalization for the prediction loss itself. First, define $\mathcal{L}_{\text{pred}}$ to be the prediction computed from the full dynamics model on the data. We divide $\mathcal{L}_{\text{pred}}$ by $\mathcal{L}_{\text{prior}}$, a fixed constant computed once per round as the value of $\mathcal{L}$ obtained with the residual disabled ($F_a \equiv 0$), i.e. the error of the bare physics prior alone:

$$\widetilde{\mathcal{L}}_{\text{pred}} = \frac{\mathcal{L}_{\text{pred}}}{\mathcal{L}_{\text{prior}}}. \tag{23}$$

This re-expresses the loss as a scale-free ratio to the bare-prior error: $\widetilde{\mathcal{L}}_{\text{pred}} \approx 1$ whenever the model performs no better than the prior alone, and drops below 1 once the residual helps. Since the residual is zero-initialized, $\widetilde{\mathcal{L}}_{\text{pred}}$ starts at exactly 1 at the beginning of every round. The full training objective is then

$$\mathcal{L} = \widetilde{\mathcal{L}}_{\text{pred}} + \frac{1}{\lambda_t} \frac{\|F_a\|}{\|F_p\|}, \qquad \lambda_{t+1} = \lambda_t + \tau\, \widetilde{\mathcal{L}}_{\text{pred}}, \tag{24}$$

The Lagragian formulation is identical to the one found in PhIHP. Since $\widetilde{\mathcal{L}}_{\text{pred}} \geqq 0$, λ_t grows monotonically, so the penalty weight $1/\lambda_t$ shrinks over training. The growth rate tracks $\widetilde{\mathcal{L}}_{\text{pred}}$: λ_t grows fastest while the model is still far from beating the bare prior, relaxing the minimality constraint early to let F_a fit freely, then grows more slowly as $\widetilde{\mathcal{L}}_{\text{pred}}$ falls, letting the constraint reassert itself once the residual has done its job. The schedule is thus self-paced by prediction quality, prioritizing fit before minimality: λ_0 and τ are fixed once and reused across all rounds and datasets, without retuning.

Trainable Physics-Prior Coefficients: PhIHP keeps θ_p fixed by default: only θ_a is optimized against $\mathcal{L}$ above. This assumes the physics prior is already reasonably accurate. When it is not, which is a realistic

case since aerodynamic coefficients are rarely known exactly for a new airframe, we instead allow a chosen subset of θ_p to be optimized jointly with θ_a, following the APHYNITY formulation. No change to $\mathcal{L}$ is required: $\|F_p\|$ is recomputed fresh on every forward pass, so the penalty above remains well-defined as θ_p moves. Section V studies the effect of prior quality and coefficient trainability on the learned dynamics model, the resulting policy, and hybrid planning.

Takeaways

We extend the PhIHP dynamics model in four ways:

- custom exploration strategy for better environment coverage,
- a physics-derived loss normalization x_c term,
- two scale-free penalty terms (the residual-to-prior ratio and the bare-prior loss normalization),
- the option to optimize the prior coefficients θ_p jointly with the residual.

None of these depend on the specifics of fixed-wing aerodynamics: each follows from having a physics prior available, and transfers to any grey-box dynamics model of this form.

B. Learning a Policy in Imagination

RL is known to be sample inefficient, and the interaction with a real system is often costly. As in PhIHP, we collect data in the simulator once to train the dynamics model and from then on, the model stands in for the simulator. Any number of policies can be trained against it without a single additional simulated interactions. Decoupling dynamics model learning from policy learning in this way amortizes the cost of interactions across every policy trained afterward, rather than paying it again for each one. This is the biggest advantage of this method, since we only need to interact with the simulator once, aside for evaluation.

Like mentioned in Section II, physics-informed models generalize better than data-driven ones, therefore we do not set out to demonstrate it again here. It is, however, implicitly confirmed by the fact that our model supports rollouts spanning hundreds of steps, from a single model with no ensembling. This is what makes training a policy entirely in imagination viable in the first place.

Each imagined transition is generated by rolling the current policy through the dynamics model. Starting from a state x_t, the agent selects an action, the model advances the state, and the reward is evaluated on the resulting observation:

$$a_t = \pi_\phi(o_t) + \epsilon, \qquad x_{t+1} = \hat{\mathcal{T}}_\theta(x_t, a_t), \qquad r_{t+1} = r(o_{t+1}), \tag{25}$$

with ϵ the usual TD3 exploration noise and o_t formed from x_t and the commanded attitude reference, as defined in Section III-B. The resulting transition $(o_t, a_t, r_{t+1}, o_{t+1})$ is stored in a replay buffer exactly as a real transition would be. Since the reward function is known, it is evaluated directly on the predicted observation rather than learned alongside the dynamics.

We train TD3 agents on these imagined transitions. Both the policy and the critics are trained purely on transitions generated by the dynamics model, with no additional real-environment interaction, except for periodic evaluation.

C. Hybrid Controller

We reuse the CEM-MPC planner introduced in Section III-D, with two modifications. First, the candidate set $\mathcal{C}$ combines random samples from the CEM Gaussian with rollouts of the trained policy π with

exploration noise, seeding the search with an already reasonable candidate rather than relying on random sampling alone. $\mathcal{C}$ is refined over the CEM iterations as defined previously. Second, CEM-MPC on a task like this typically requires a long planning horizon to perform well, which is computationally expensive. We instead plan over a short horizon H and bootstrap the remaining return with the trained critic, evaluated at the policy's action at the final predicted state, so that a short, cheap horizon still accounts for the long-term value of the state-action pair it ends on. Since TD3 trains twin critics, the bootstrap uses their clipped minimum. The resulting optimization problem is

$$A^\star = \arg\max_{A \in \mathcal{C}} \left(\sum_{k=0}^{H-1} \gamma^k \, r\big(o_{t_0+k+1}\big) + \alpha \, \gamma^H \min_{j \in \{1,2\}} Q_j\big(o_{t_0+H},\, \pi(o_{t_0+H})\big) \right), \tag{26}$$

where $A^\star$ is the optimal sequence of actions. In practice, we only select the first action a_t^*, as mentioned in Section III-D. The coefficient α weights the bootstrapped term relative to the planned return. Each observation along the rollout is formed as $o_{t+1} = (x_{t+1}, \phi^\star, \theta^\star)$, with x_{t+1} predicted by the dynamics model $\hat{\mathcal{T}}_\theta(x_t, a_t)$ and the attitude reference held fixed over the horizon, as defined in Section III-B. In section V-D, we split this controller into different controllers to study the effect of each component of this method.

V. Experiments

This section defines the environment, the training procedures, and the evaluation protocol used to answer the three research questions raised by the contributions introduced in Section I. We pose three questions here, and answer each in Section VI:

Q1: Can a policy trained purely in imagination, on a physics-informed dynamics model, match the performance of a policy trained directly on the simulator? We compare policies trained purely in imagination against a baseline of TD3 trained directly in the simulator. The point is not to measure attitude tracking quality in absolute terms, but to verify that substituting a learned dynamics model for real-simulator interaction does not cost final policy quality. This confirms a claim already made in the MBRL literature rather than contributing a new one. We additionally show that convergence can be sped up by parallelizing the dynamics model, which is not as easily done with the simulator.

Q2: Does tuning the physics prior's own parameters alongside the learned residual recover the policy performance lost to an inaccurate prior? We train dynamics models from six different starting points: varying the quality of the physics prior and whether the optimizer is allowed to tune it. Keeping the prior frozen corresponds to PhIHP's original formulation, while allowing it to be tuned is the modification we propose. Comparing the policies trained on each of the resulting models measures whether this modification helps, and under which prior conditions. This is where attitude tracking performance is evaluated directly.

Q3: How much does the Hybrid Controller improve attitude tracking at inference time, and which of its components drives that improvement? We run an ablation over the hybrid controller's components at inference time, to measure how much it improves attitude tracking and which of its parts drives that improvement. Since these parts differ in computational cost, inference time is reported alongside tracking accuracy.

A. Environment

We instantiate the attitude tracking task introduced in Section III-B as follows. The agent operates at 100 Hz ($\Delta t = 0.01$ s), and its objective is to drive roll and pitch to a commanded reference $(\phi^\star, \theta^\star)$ and

hold it, while regulating airspeed as a secondary, more loosely penalised objective.

State and observation: The physical state $s_t^p \in \mathbb{R}^8$ matches the eight variables used by the physics prior (Section III-C). This physical state alone does not determine the next state, since it excludes the current position and rate of the control surfaces and throttle: the aircraft does not respond to commands instantaneously, so the same state and the same commanded action can lead to different outcomes depending on recent command history. We therefore extend the state with an actuator state $z_t \in \mathbb{R}^5$, giving the full environment state

$$x_t = (s_t^p,\, z_t), \qquad s_t^p \in \mathbb{R}^8, \quad z_t \in \mathbb{R}^5. \tag{27}$$

Providing the agent with this actuator information, rather than the raw commanded action alone, has been shown sufficient to learn effective policies on this task [41]. The agent observes

$$o_t = \big(x_t,\, (\phi_t^\star, \theta_t^\star)\big) \in \mathbb{R}^{15}, \tag{28}$$

combining the physical state and actuator state with the attitude reference. The agent does not need to explicitly see the previous action, since the actuator state implicitly carries this information.

Action: The action is

$$a_t = (a_t^{\text{ail}},\, a_t^{\text{ele}},\, a_t^{\text{thr}}) \in [-1, 1]^3, \tag{29}$$

the commanded aileron and elevator deflections and the commanded throttle. The deflection that actually acts on the airframe at a given step is given by the actuator state z_t.

Reward: The reward penalises deviation from the commanded attitude and from a nominal airspeed $V_a^\star = 16.67$ m/s (60 km/h). The attitude part is the sum of two sub-rewards, one per tracked angle, each penalising how far the current angle is from its commanded value. Roll and pitch are scaled separately rather than by a shared constant, since the two angles have different typical error ranges over the course of an episode. The coefficients c_ϕ and c_θ are set to the approximate size of that range for each angle, so that a given fraction of c_ϕ or c_θ corresponds to a comparably large error on either axis:

$$r_t = -\left(\text{clip}\left(\tfrac{|\phi_t - \phi^\star|}{c_\phi}, 0, \kappa_\phi\right) + \text{clip}\left(\tfrac{|\theta_t - \theta^\star|}{c_\theta}, 0, \kappa_\theta\right) + \text{clip}\left(\tfrac{|V_{a,t} - V_a^\star|}{c_v}, 0, \kappa_v\right)\right), \tag{30}$$

Each term is normalised and capped, so that no single axis can dominate once its error is large. The reward is bounded and maximised by holding the commanded attitude at the nominal airspeed. We use $c_\phi = 3.3$, $\kappa_\phi = 0.5$, $c_\theta = 2.25$, $\kappa_\theta = 0.5$, and a deliberately loose airspeed term, $c_v = 48$, $\kappa_v = 0.1$. The task is attitude tracking, and a tight airspeed penalty would prevent the agent from trading speed for attitude authority, which is required to track steep nose-down references.

Real and imagination environments: Two distinct kinds of environment appear throughout our experiments. The simulator is FW-JSBGym itself, used to collect the data the dynamics models are trained on and, in every experiment, to evaluate the final policies. The imagination environments are the learned dynamics models, used only to generate the trajectories that policies train on. Since we train several dynamics models from physics priors of differing quality (Section V-B), each one defines its own imagination environment, and a policy trained in one is effectively trained in a different, imperfect approximation of the simulator. Evaluation always takes place in the simulator, so all reported performance figures are directly comparable regardless of which imagination environment a policy was trained in.

B. Dynamics Models and Imagination Environments

We train six dynamics models, and therefore six imagination environments, in a 3×2 design. The first factor is the quality of the physics prior the model starts from: the true aerodynamic coefficients, a mild degradation with ten coefficients perturbed by 25-50%, and a severe degradation with the same ten coefficients perturbed by 75-100%. The second factor is whether those prior coefficients are frozen during training, as in PhIHP's original formulation, or made trainable, that is, optimized jointly with the learned residual F_a, as we propose in Section IV-A.

We consider the true prior to be the one coming from the Gryte et al. paper [40]. These coefficients were measured using a wind tunnel or through XFLR5 simulations (an analysis tool for planes), and are considered reliable. The degraded priors stand in for the more realistic case where a vehicle's aerodynamic coefficients are only approximately known. The full list of coefficients is available in Table VI in Appendix B for reproducibility, including the justification for their choice.

All six models follow the same training recipe, so the prior is the only variable that changes between them. Each model is trained for one round of 175 random-exploration episodes followed by five iterative CEM data-collection rounds of 25 episodes each, for 300k transitions in total, with each episode lasting 1000 steps.

C. Policy Training

From each of the six dynamics models, we train ten TD3 policies from independent seeds, entirely in imagination, for 700k steps with no additional real-environment interaction.

Because the dynamics model is cheap to run compared to the simulator, it can be instantiated many times in parallel. We use 64 parallel imagination environments. The update-to-data (UTD) ratio is preserved across all training regimes, so every policy receives the same number of gradient updates for the same number of environment steps, and the comparison against the real-environment baseline remains fair. Further implementation details for TD3 are given in Section C.

D. Hybrid Controller Ablation

The Hybrid Controller adds two components to a short-horizon CEM-MPC planner: policy action proposals, which seed the candidate pool with actions the trained policy considers good, and the critic bootstrap, which estimates the return beyond the planning horizon. We evaluate (see next section) each component on its own and both together, giving the four planner variants listed in Table I. The trained policy acting alone is included as a reference point, since it is what the planner is meant to improve on.

TABLE I: Description of the different controllers compared in the ablation study in Section VI-C.

Controller	Description
Policy alone	No planning, the trained policy acts directly
CEM-MPC	Short-horizon planning through the dynamics model
CEM-MPC $+\,\pi$	Policy proposals seeded into the candidate action pool
CEM-MPC $+\,Q$	Critic estimate of the return beyond the horizon per trajectory
Hybrid ($\pi + Q$)	Both policy proposals and critic estimate

E. Evaluation Protocol

We evaluate policies on a large sample of attitude targets, split between easy and hard targets. Easy targets are considered to be nominal flying conditions. The target difficulty classification is defined in Table II, exactly as defined in [17].

TABLE II: Reference classification.

Difficulty	Roll (°)	Pitch (°)
Easy	$(-45, 45)$	$(-25, 25)$
Hard	$[-60, -45) \cup (45, 60]$	$[-30, -25) \cup (25, 30]$

Each evaluation episode runs for 1000 steps in the simulator, starts from the aircraft's neutral, trim attitude, and commands a fixed target for the full episode, so the tracking error reflects the policy's ability to reach and hold that target. Two metrics are reported:

- **Settled attitude tracking error:** the asymptotic performance is measured with the RMSE between the achieved and commanded roll and pitch over the final 20% of an episode, once the agent has supposedly settled. Roll and pitch RMSE are averaged into a single attitude error per target, then averaged over targets to give one score per seed. We report tracking error in degrees rather than radians, since a one-degree error is easier to relate to physically meaningful attitude deviations than the equivalent fraction of a radian. We consider a settled attitude error below 1° to indicate near-perfect tracking, and an error between 1° and 2° to be acceptable.

- **Inference time:** additionally reported for the hybrid controller experiments, since planning adds a computational cost at every control step that a policy-only controller does not have. The inference time is reported in ms/step. The real-time budget is 10 ms/step, which we use as the reference point when interpreting these numbers.

Following the evaluation protocol of Agarwal et al. [42], we report the median, interquartile mean (IQM), and mean across seeds, each with a 95% percentile-bootstrap Confidence Interval (CI) computed over the seeds.

VI. Results

In this section, we conduct thorough experiments to answer the questions defined in Section V. Results are available as tables to see numerical results and also as interval plots for better readability.

A. Validating Training in Imagination

To answer Q1 (Section V), we compare three training regimes on the true-prior dynamics model: TD3 trained directly in the simulator (Model Free TD3), TD3 trained in imagination on a single instance of the learned dynamics model (Model Environment TD3), and TD3 trained in imagination with multiple parallelized instances of the dynamics model (Parallelized Model Environment TD3).

Is the dynamics model accurate enough over long horizons to replace the simulator? Reaching a commanded attitude from a neutral start takes on the order of a hundred steps. A policy trained in imagination must therefore be exposed to rollouts of at least that length, generated from a single model with no ensembling. This is well beyond the few-step rollouts most MBRL methods restrict themselves to [27], [28]. State prediction error alone does not settle whether the model holds up over such horizons: a drifting model still produces plausible trajectories, but the policies trained on them fail once deployed.

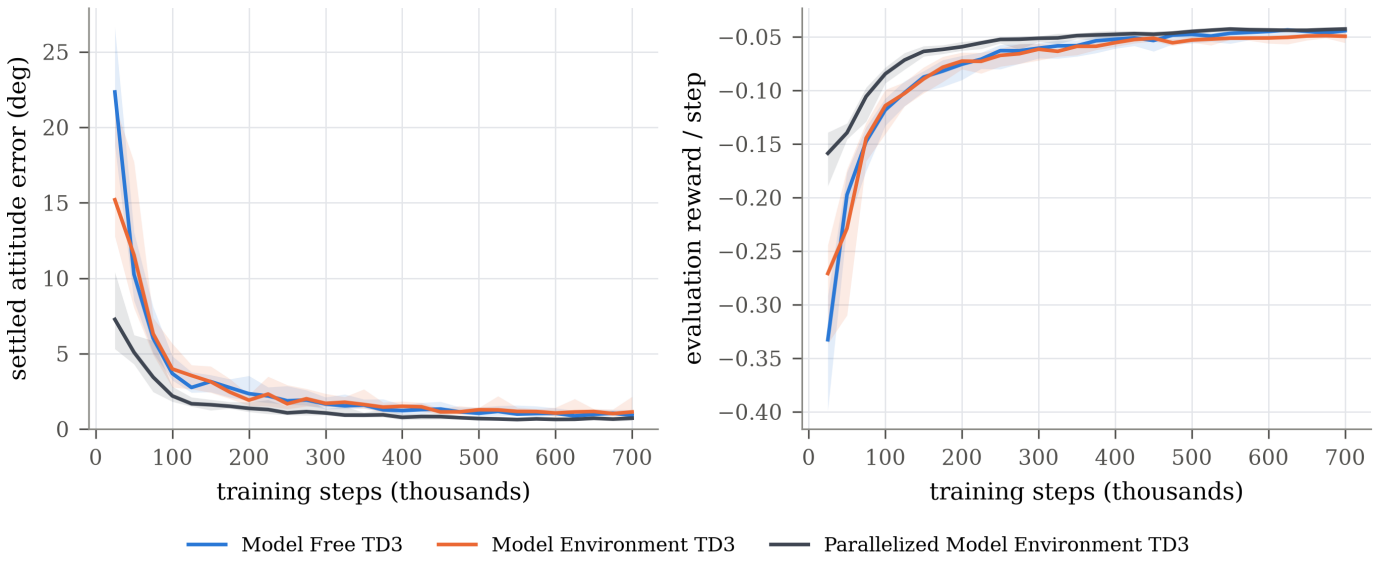


Fig. 2: **Performance and reward curves comparing TD3 trained in the simulator, in imagination on a single dynamics-model instance, and in imagination with parallelized dynamics-model instances.** Left: settled attitude tracking error. Right: average evaluation reward per step. Center lines are the IQM across 10 seeds, shaded bands are 95% CI. We see the parallelized TD3 converges faster than the other to with a slightly better asymptotic performance overall, while the other converge at the same speed, with the Model Free TD3 slightly beating the Imagination TD3 in performance.

We test this by training policies on long imagined rollouts and evaluating them on the simulator.

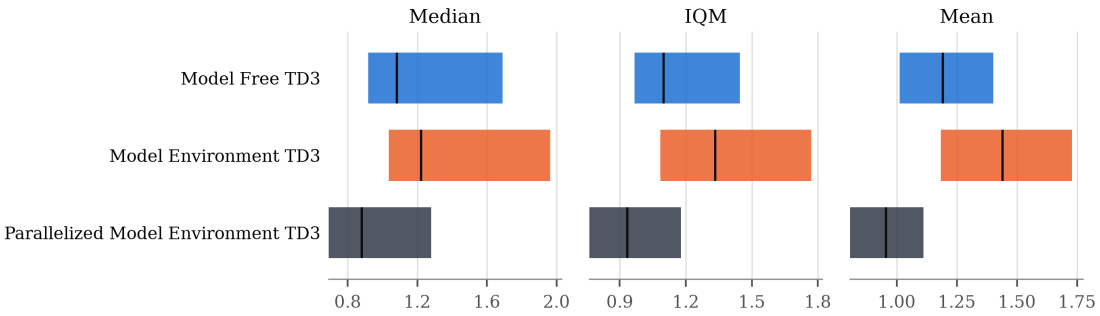


Fig. 3: **Final settled attitude error over the full evaluation grid, for the same three training regimes.** Median, IQM, and mean across 10 seeds, with 95% CI. Lower is better.

Figure 2 shows that Model Free TD3 and Model Environment TD3 follow the same learning behavior, on both settled attitude error and evaluation reward, with largely overlapping confidence bands. Figure 3 evaluates the final policies across the full target grid, rather than the smaller set used during training. At this scale, a small gap appears: Model Free TD3 is slightly better across median, IQM, and mean alike. Training in imagination therefore costs a little asymptotic performance, but the two regimes remain close and their intervals overlap. The model stays faithful enough over a full episode for the resulting policies to transfer. Table IX shows that the mean performance difference between the two is 20% but with partly overlapping CIs.

Does parallelizing the Model Environment help? The dynamics model is cheap to instantiate many times over, whereas the simulator is not, so we take advantage of this. Figure 2 shows the effect on convergence speed: the parallelized variant reaches its asymptotic region in substantially fewer training

steps than either other regime. Figure 3 shows the benefit is not limited to speed. Parallelizing also yields the lowest final settled attitude error and the tightest intervals of the three regimes, with a clear margin over single-instance imagination training. For the same training budget, parallelism gives the agent proportionally more environment coverage per step, which recovers the small asymptotic gap left above and improves on the simulator baseline. In Table IX in Appendix F, we see that the performance gain is of about 20%, the same order of difference observed between the other TD3 training regimes.

That parallelism improves both convergence speed and final performance is not a new claim in the RL literature. What matters here is that it holds in our setting, and that it is possible because the environment being parallelized is a learned model rather than a flight simulator. Every policy used in the following experiments was trained this way. Implementation details are given in Appendix C.

Takeaway

Our dynamics model stays accurate over rollouts spanning hundreds of steps, from a single model with no ensembling: policies trained entirely on these rollouts transfer to the simulator with only a small loss in tracking accuracy. Parallelizing the Model Environment, which the simulator does not permit easily, recovers that gap and converges faster.

B. Impact of Prior Quality and Tuning on Policy Performance

In this section, we answer Q2 (Section V). Table III reports the performance of policies trained from dynamics models with different prior conditions, each score aggregating ten independently seeded TD3 policies (Section V-B). All values referenced below can be found in that table. Figure 4 shows the same results as an interval plot.

TABLE III: Policy performance across the six physics-prior conditions (easy targets reported in Appendix F). Each entry aggregates $n = 10$ independently seeded TD3 policies, the per-seed score is the mean settled attitude error. Brackets give 95% percentile-bootstrap CI over the seeds. Lower is better, best results in bold and second best underlined.

	Settled attitude error (deg) [95% CI]		
Prior condition	Median	IQM	Mean
All targets			
True (frozen)	**0.91** [0.75, 1.22]	<u>0.97</u> [0.79, 1.17]	<u>0.97</u> [0.83, 1.12]
True (trainable)	<u>1.00</u> [0.74, 1.11]	**0.95** [0.79, 1.09]	**0.94** [0.82, 1.06]
Mild degr. (frozen)	1.65 [1.13, 2.30]	1.69 [1.21, 2.18]	1.67 [1.31, 2.03]
Mild degr. (trainable)	1.18 [0.81, 1.52]	1.15 [0.86, 1.49]	1.20 [0.95, 1.46]
Severe degr. (frozen)	1.87 [1.31, 2.20]	1.79 [1.40, 2.14]	1.77 [1.48, 2.06]
Severe degr. (trainable)	1.10 [0.83, 1.37]	1.09 [0.91, 1.36]	1.15 [0.97, 1.35]
Hard targets			
True (frozen)	<u>1.52</u> [0.97, 2.15]	<u>1.55</u> [1.07, 2.04]	<u>1.56</u> [1.21, 1.90]
True (trainable)	**1.39** [1.13, 1.73]	**1.41** [1.16, 1.65]	**1.40** [1.19, 1.60]
Mild degr. (frozen)	3.26 [1.88, 4.75]	3.34 [2.19, 4.48]	3.31 [2.46, 4.15]
Mild degr. (trainable)	1.99 [1.16, 2.70]	1.89 [1.32, 2.67]	2.04 [1.49, 2.67]
Severe degr. (frozen)	3.25 [1.77, 4.19]	3.05 [2.04, 4.03]	3.08 [2.34, 3.81]
Severe degr. (trainable)	1.90 [1.25, 2.39]	1.83 [1.41, 2.40]	1.94 [1.53, 2.40]

How much does a wrong prior cost? With the prior frozen, prior accuracy strongly determines policy quality. We can see from Table III that the overall mean error rises from $0.97°$ for the true prior to $1.67°$ (mild) and $1.77°$ (severe). The cost is not spread evenly across the flight envelope: on the easy targets it

is negligible. On the hard targets it more than doubles ($1.56°$ true, against $3.31°$ mild and $3.08°$ severe). A wrong prior is therefore cheap near trim and expensive at the edge of the envelope, where the residual must extrapolate furthest from its training data. The full results, including easy targets results, are available in Table X in Appendix F.

Does tuning the prior recover this loss? Tuning the prior does recover most of this loss, and this is the key result of this study. Overall error drops by 28% for the mild degradation ($1.67°$ to $1.20°$) and 35% for the severe one ($1.77°$ to $1.15°$), with comparable reductions on the hard targets (37-38%). The spread across prior conditions shrinks accordingly: the three trainable conditions land within $0.94°$-$1.20°$ overall, against $0.97°$-$1.77°$ when frozen. The benefit is not limited to degraded priors either. On the easy targets, tuning changes little, except for the severely degraded prior where it still helps clearly ($0.84°$ to $0.58°$). Figure 4 makes this shift visible directly: for every degraded prior, the orange (trainable) band sits to the left of the corresponding blue (frozen) band, while the two true-prior rows remain close together.

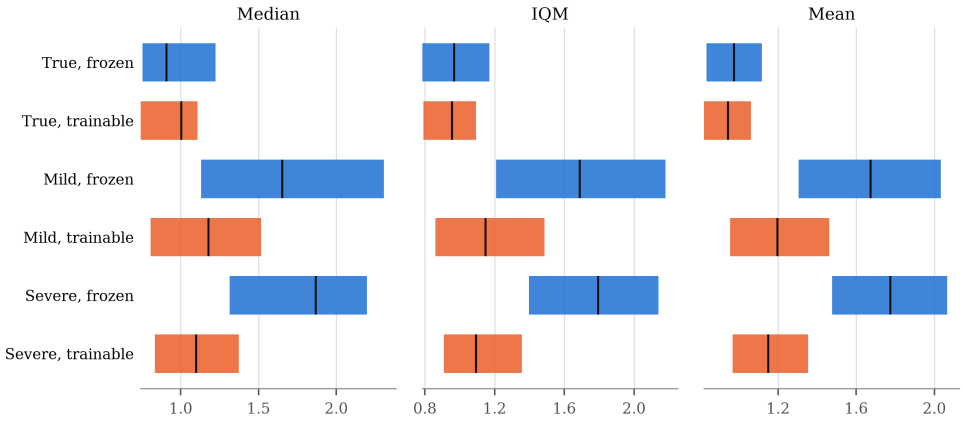


Fig. 4: **TD3 policy performance across the six physics-prior conditions, all targets.** Each row shows the point estimate (bold rule) and 95% percentile-bootstrap CI (bar) over $n = 10$ seeded policies, for the median, IQM, and mean settled attitude error (deg). Blue = frozen prior, orange = trainable prior. Lower is better.

Takeaway

Allowing the optimizer to tune the prior alongside the residual improves downstream policy performance, specifically when a vehicle's aerodynamic coefficients are only approximately known. This makes our method robust to inaccurate physics priors.

C. Hybrid Controller Improvements

This section answers Q3 (Section V). Table IV evaluates the hybrid controller over all six policy-dynamics model combinations from Section VI-B, and Table XI in Appendix F restricts the same comparison to the true frozen prior. In this section, we report mostly these results as interval plots for better interpretability.

Near-trim attitude tracking is already handled well by classical controllers such as PID [17], so improving on it is not where a planner earns its cost. The easy targets do reveal one thing: vanilla CEM-MPC and CEM-MPC+π fail to converge to accurate tracking, reaching $4.0°$-$4.1°$. This is not a general loss of accuracy but a sharp failure mode. Both track near-trim targets very well, then fail outright once a target approaches the edge of the nominal envelope, a roll of $40°$ for instance, while such a maneuver is

still being classified as easy. Since the easy bucket includes some of these more aggressive targets, this alone inflates the mean. We analyze this failure mode in table VIII in Appendix D. We therefore focus on overall performance across the full flight regime, and on hard targets specifically.

Figures 7, 8, 9 in Appendix E illustrate the quality of the tracking between the Hybrid Controller and the TD3 agent. They show overall that the Hybrid Controller shows superior tracking ability, specifically for difficult targets.

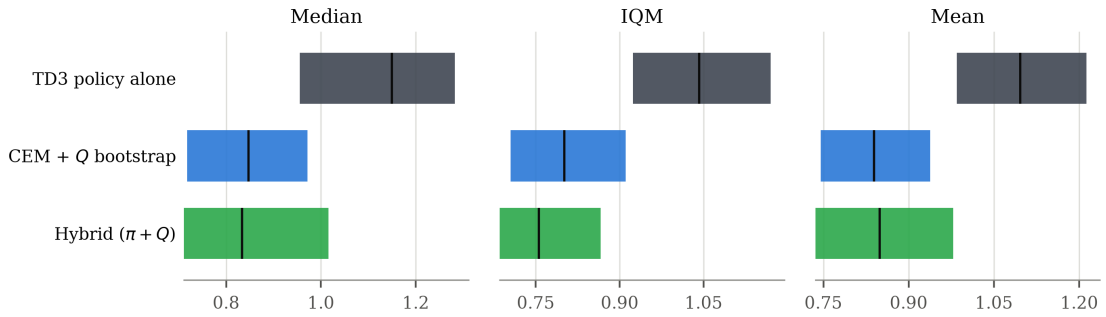


Fig. 5: **Controller performance aggregated over all six physics-prior conditions, all targets.** Point estimate (bold rule) and 95% percentile-bootstrap CI (bar) for the median, IQM, and mean settled attitude error (deg). Lower is better.

What makes CEM-MPC viable? In Table IV, on the hard targets, every target lies past the edge of a short-horizon CEM-MPC's working range, so the failure mode above applies uniformly rather than to a few targets. Both vanilla CEM-MPC and CEM-MPC+π are non-functional, at $39.30°$ and $42.04°$ mean error. Adding policy proposals changes nothing. Adding the critic bootstrap fixes this almost entirely: CEM-MPC+Q drops to $1.86°$ and the full Hybrid Controller to $1.74°$, a 22% reduction over the TD3-alone baseline of $2.22°$. Across all targets the gain is a 23% improvement ($1.29°$ to $0.99°$). Bootstrapping the value beyond the planning horizon is therefore what makes CEM-MPC usable on this task, not seeding the candidate pool. Policy proposals help only once the bootstrap is already present, and then only slightly.

Takeaway

The key element of the Hybrid Controller is the critic bootstrap: instead of having to plan far into the future to estimate a trajectory's return, the critic estimates it directly from a short horizon, making long-horizon planning possible.

Where is the Hybrid Controller most impactful? Table XI in Appendix F restricts the comparison to our base case with the true frozen prior, our most faithful dynamics model (relevant to the simulated physics). On the hard targets, the hybrid controller improves the TD3-alone reference from $1.56°$ to $0.89°$, a 43% reduction, with CEM-MPC+Q being indistinguishable at $0.87°$. Across all targets the reduction reaches 35% ($0.97°$ to $0.63°$), against 23% when aggregated over all six priors like mentioned above. Planning through the dynamics model is only as good as the model itself, so the Hybrid Controller is most impactful when the dynamics model is accurate.

TABLE IV: Controller performance aggregated across the six physics-prior conditions (6 conditions $\times$ 10 seeds $=$ 60 runs). Lower is better, best results in bold and second best underlined.

	Settled attitude error (deg) [95% CI]		
Controller	Median	IQM	Mean
All targets			
TD3 policy alone	1.17 [1.04, 1.33]	1.17 [1.07, 1.29]	1.29 [1.19, 1.38]
CEM (H=5)	18.04 [17.55, 18.52]	18.17 [17.80, 18.57]	18.74 [18.36, 19.12]
CEM $+\pi$	18.71 [18.20, 19.54]	19.57 [19.04, 20.13]	19.85 [19.28, 20.44]
CEM $+Q$	**0.88** [0.77, 1.00]	<u>0.82</u> [0.74, 0.93]	<u>1.06</u> [0.92, 1.21]
Hybrid Controller	<u>0.88</u> [0.77, 1.06]	**0.77** [0.71, 0.89]	**0.99** [0.87, 1.11]
Easy targets			
TD3 policy alone	0.59 [0.56, 0.62]	0.59 [0.56, 0.62]	0.62 [0.59, 0.65]
CEM (H=5)	3.64 [3.41, 3.91]	3.72 [3.57, 3.89]	4.05 [3.94, 4.16]
CEM $+\pi$	3.63 [3.43, 3.94]	3.74 [3.59, 3.92]	4.00 [3.88, 4.14]
CEM $+Q$	<u>0.48</u> [0.46, 0.51]	<u>0.48</u> [0.46, 0.50]	<u>0.49</u> [0.47, 0.51]
Hybrid Controller	**0.44** [0.42, 0.47]	**0.44** [0.42, 0.46]	**0.45** [0.44, 0.47]
Hard targets			
TD3 policy alone	1.99 [1.68, 2.36]	1.91 [1.70, 2.19]	2.22 [1.99, 2.45]
CEM (H=5)	37.55 [36.29, 38.82]	38.33 [37.36, 39.31]	39.30 [38.41, 40.20]
CEM $+\pi$	39.04 [37.90, 41.13]	41.44 [40.09, 42.82]	42.04 [40.65, 43.47]
CEM $+Q$	**1.46** [1.22, 1.75]	<u>1.30</u> [1.11, 1.56]	<u>1.86</u> [1.53, 2.21]
Hybrid Controller	<u>1.50</u> [1.21, 1.90]	**1.21** [1.06, 1.49]	**1.74** [1.46, 2.04]

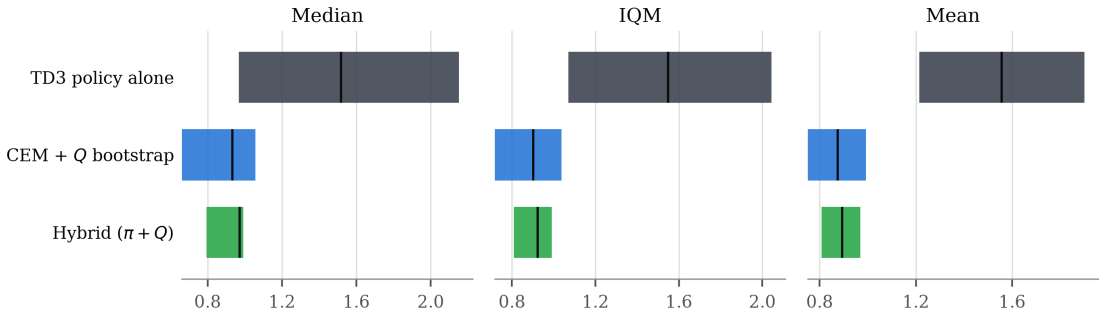


Fig. 6: **Controller performance on the hard targets, true frozen prior only.** Point estimate (bold rule) and 95% percentile-bootstrap CI (bar) over $n = 10$ seeded policies. Lower is better.

What is the inference cost of the Hybrid Controller? Like all planning-based methods, our controller has an inference cost. Table V shows the cost of these performance gains. The deterministic policy alone is fast, at 0.187 ms per step, well within the 10 ms budget of our 100 Hz control loop (Section V-A). Every CEM-MPC based controller exceeds this budget: CEM-MPC alone already takes 12.05 ms, about 20% over budget, and cost grows further as components are added, reaching 16.63 ms for CEM-MPC+π, 16.34 ms for CEM-MPC+Q, and 21.25 ms for the full hybrid controller, slightly more than double the maximum time available for a step.

Takeaway

Compared to a pure RL agent controller, the Hybrid Controller improves attitude control by up to 43% on hard targets when leveraging a dynamics model trained from an accurate prior, and 22% across the full flight-regime.

TABLE V: Inference time per control step across the five controllers

Mode	Inference time (ms/step)
Policy only	0.187
CEM (H=5)	12.047
CEM $+\pi$	16.625
CEM $+Q$	16.341
Hybrid Controller	21.252

VII. Conclusion

This work asked whether physics-informed hybrid planning, previously validated only on classic control benchmarks and quadrotors, extends to the more strongly coupled and less-studied setting of fixed-wing UAV attitude control, and whether it remains useful once the physics prior it relies on is imperfect.

We find that it does on both counts. A dynamics model built from an aerodynamic physics prior of the Skywalker X8 stays accurate over long horizons. This lets us train a policy purely in imagination. Through environment parallelization, resulting policies exceed the ones trained through direct interaction with the simulated environment, without further simulator interaction beyond the initial data collection. When the prior itself is inaccurate, tracking degrades sharply away from trim. This is a realistic case for a new or lightweight airframe. Tuning the parameters of the physics prior jointly with the learned residual largely closes this gap. Prior work had shown that tuning the coefficients of a physics prior improves forecasting accuracy, but had not tested whether this translates into better downstream policies. We close this gap: tuning the prior does yield better policies, and the benefit is largest exactly when expert knowledge is unavailable or inaccurate. Finally, combining the trained policy and critic into a short-horizon Hybrid Controller improves tracking substantially on the hardest targets. Our ablation shows this gain comes almost entirely from bootstrapping the critic beyond the planning horizon, not from seeding the search with policy proposals. This distinction has not been isolated before in this line of work.

The main cost of this improvement is computational: the planning time of the Hybrid Controller exceeds our real-time control budget. We return to this limitation in Section VIII. More broadly, these results suggest physics-informed model-based RL is a viable path to sample-efficient, robust control on fixed-wing airframes. The modifications we introduce to reach that result are not tied to this setting: the loss normalization, the residual penalty, and the tunable prior all follow from having a physics prior at hand, and apply to any grey-box dynamics model of this form. What remains system-specific is the prior itself.

To recap, our main takeways are as follows:

- Policies trained purely in imagination perform slightly worse than policies trained through direct simulator interaction, but parallelizing the model environment closes this gap and yields the best performance overall.
- Jointly tuning the physics prior with the learned residual recovers most of the downstream tracking accuracy lost to an inaccurate prior.
- The Hybrid Controller cuts tracking error by up to 43%, driven almost entirely by the critic bootstrap rather than policy-seeded proposals.

VIII. Future Works

Waypoint tracking: This work focused on attitude tracking, holding a fixed commanded roll and pitch. A natural extension is waypoint tracking, a harder task requiring the controller to sequence and reach a series of spatial targets rather than a single static attitude reference. This would test whether the accuracy gains from a tuned physics prior and the Hybrid Controller persist under a longer-horizon, more

compositional task.

Reducing inference time: The Hybrid Controller's main limitation is computational: at 21.25 ms/step it exceeds our 10 ms real-time budget by more than a factor of two. Rather than shortening the planning horizon outright, which would reduce the effective lookahead of the controller, this cost could be reduced by introducing action delay or executing short action sequences instead of replanning at every single step, reducing the frequency at which CEM-MPC needs to run without proportionally reducing planning quality [33]. Other state-of-the-art methods use different planning strategies. TD-MPC, for instance, relies on Model Predictive Path Integral (MPPI) control [23], which is more computationally efficient than CEM and could reduce inference time.

Joint learning of the dynamics model and the policy: Our method decouples dynamics model learning from policy learning: the model is trained first, then frozen while the TD3 policy trains on it. According to [33], learning both jointly should improve downstream performance.

Hierarchical control: Our controller operates on a single, fixed time constant, but a real mission profile combines dynamics at different timescales, like a fast attitude-control loop and a slower guidance loop for a full drone mission. A hierarchical extension of our method, introducing shorter-horizon objective sequences within the current planning loop rather than a single flat horizon, could extend it to tasks with varying horizon requirements or greater temporal complexity, such as waypoint tracking or mission-level guidance. This type of control is a promising direction for this extension.

Comparing our method to state-of-the-art methods: This work evaluates our method against its own ablations rather than against other RL algorithms. Olivares et al. [17] established baselines such as PPO and TD-MPC on this same fixed-wing attitude tracking task. A direct numerical comparison against these methods, on our evaluation protocol, would situate our results within the broader literature.

REFERENCES

[1] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, pp. 109–137, Mar. 2023.

[2] M. Abdullah, A. Zulfiqar, and J. H. Arman, "A Comprehensive Review on UAV Control Systems: From Linear to Intelligent Approaches," *ACM Comput. Surv.*, vol. 58, no. 13, pp. 337:1–337:35, 2026.

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[4] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, pp. 3769–3776, Apr. 2021.

[5] A. A. Habel, R. A. Khan, F. Mehboob, C. Fortin, and D. Tsetserukou, "GustPilot: A Hierarchical DRL-INDI Framework for Wind-Resilient Quadrotor Navigation," Mar. 2026. arXiv:2603.19966 [cs].

[6] D. J. Richter, R. A. Calix, and K. Kim, "A Review of Reinforcement Learning for Fixed-Wing Aircraft Control Tasks," *IEEE Access*, vol. 12, pp. 103026–103048, 2024.

[7] A. Ramesh and B. Ravindran, "Physics-Informed Model-Based Reinforcement Learning," in *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, pp. 26–37, PMLR, June 2023.

[8] Y. Yin, V. L. Guen, J. Dona, E. d. Bézenac, I. Ayed, N. Thome, and P. Gallinari, "Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, p. 124012, Dec. 2021. arXiv:2010.04456 [stat].

[9] Z. E. Asri, O. Sigaud, and N. Thome, "Physics-Informed Model and Hybrid Planning for Efficient Dyna-Style Reinforcement Learning," July 2024. arXiv:2407.02217 [cs].

[10] A. Osaka, N. Takeishi, and T. Yairi, "Domain adaptation with hybrid modeling for learning dynamical systems," in *The 17th Asian Conference on Machine Learning (Conference Track)*, 2025.

[11] C. Banerjee, K. Nguyen, C. Fookes, and M. Raissi, "A Survey on Physics Informed Reinforcement Learning: Review and Open Problems," *Expert Systems with Applications*, vol. 287, p. 128166, Aug. 2025. arXiv:2309.01909 [cs].

[12] A. Kwiatkowski, M. Towers, J. K. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, K. Andreas, M. Krimmel, A. Kg, R. D. L. Perez-Vicente, A. Pierré, S. V. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis, "Gymnasium: A Standard Interface for Reinforcement Learning Environments," Oct. 2024.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[14] S. Fujimoto, H. v. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," Oct. 2018. arXiv:1802.09477 [cs.AI].

[15] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 523–533, June 2019. arXiv:1911.05478 [cs.RO].

[16] H. R. Khanzada, A. Maqsood, and A. Basit, "Artificial Intelligence in UAV Flight Controls: Deep Reinforcement Learning Based Altitude-Hold Strategies for Fixed-Wing UAVs," *IEEE Access*, vol. 13, pp. 109670–109686, 2025.

[17] D. Olivares, P. Fournier, P. Vasishta, and J. Marzat, "Model-Free versus Model-Based Reinforcement Learning for Fixed-Wing UAV Attitude Control Under Varying Wind Conditions," Sept. 2024. arXiv:2409.17896 [cs.RO].

[18] Y. Zhang, Y. Zhang, and Z. Yu, "Path Following Control for UAV Using Deep Reinforcement Learning Approach," *Guidance, Navigation and Control*, vol. 01, p. 2150005, Mar. 2021.

[19] H. He, Z. Zhao, and F. Kong, "Longitudinal Control and Optimization of Fixed-Wing UAV Based on Deep Reinforcement Learning," in *Advances in Guidance, Navigation and Control* (L. Yan, H. Duan, and Y. Deng, eds.), (Singapore), pp. 1–11, Springer Nature, 2025.

[20] J. Choi, H. M. Kim, H. J. Hwang, Y.-D. Kim, and C. O. Kim, "Modular Reinforcement Learning for Autonomous UAV Flight Control," *Drones*, vol. 7, p. 418, July 2023.

[21] J. Xu, T. Du, M. Foshey, B. Li, B. Zhu, A. Schulz, and W. Matusik, "Learning to fly: computational controller design for hybrid UAVs with reinforcement learning," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 42:1–42:12, 2019.

[22] N. Hansen, X. Wang, and H. Su, "Temporal Difference Learning for Model Predictive Control," July 2022. arXiv:2203.04955 [cs].

[23] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, Robust World Models for Continuous Control," Mar. 2024. arXiv:2310.16828 [cs].

[24] X.-Y. Liu and J.-X. Wang, "Physics-informed Dyna-style model-based deep reinforcement learning for dynamic control," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 477, p. 20210618, Nov. 2021.

[25] A. Osaka, N. Takeishi, and T. Yairi, "Deterministic and Stochastic Hybrid Modeling with Regularization," in *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 5637–5642, Oct. 2025.

[26] P. Ghanem, A. Demirkaya, T. Imbiriba, A. Ramezani, Z. Danziger, and D. Erdogmus, "Learning Physics Informed Neural ODEs With Partial Measurements," Dec. 2024. arXiv:2412.08681 [cs].

[27] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[28] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to Trust Your Model: Model-Based Policy Optimization," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[29] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[30] J. Kübel, H. Krauss, J. Li, and M. Zhao, "Energy-based Regularization for Learning Residual Dynamics in Neural MPC for Omnidirectional Aerial Robots," Apr. 2026. arXiv:2604.14678 [eess.SY].

[31] R. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology And Computing In Applied Probability*, vol. 1, pp. 127–190, Sept. 1999.

[32] M. Walker, D. Frisch, and U. D. Hanebeck, "Sample-Efficient and Smooth Cross-Entropy Method Model Predictive Control Using Deterministic Samples," 2025. Version Number: 2.

[33] Z. E. Asri, I. Laiche, C. Rambour, O. Sigaud, and N. Thome, "RT-HCP: Dealing with Inference Delays and Sample Efficiency to Learn Directly on Robotic Platforms," Sept. 2025. arXiv:2509.06714 [cs].

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," July 2019. arXiv:1509.02971 [cs.LG].

[38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," Aug. 2018. arXiv:1801.01290 [cs.LG].

[39] M. T. Spaan, "Partially observable markov decision processes," in *Reinforcement learning: State-of-the-art*, pp. 387–414, Springer, 2012.

[40] K. Gryte, R. Hann, M. Alam, J. Roháč, T. A. Johansen, and T. I. Fossen, "Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 826–835, June 2018. ISSN: 2575-7296.

[41] D. Olivares, *Reinforcement learning based control of a fixed-wing UAV under wind disturbances*. Theses, Université Paris-Saclay, Dec. 2025. Issue: 2025UPASG112.

[42] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. Bellemare, "Deep Reinforcement Learning at the Edge of the Statistical Precipice," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 29304–29320, Curran Associates, Inc., 2021.

[43] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

APPENDIX A
DETAILED PHYSICS PRIOR CALCULATIONS

This appendix presents the complete derivation of the forces, moments, and translational velocity derivatives that form the Skywalker X8 physics prior $F_p^{\theta_p}(\boldsymbol{s}^p, z)$ introduced in Section III-C. Standard aircraft dynamic conventions follow Beard and McLain [43].

A. Air-Relative Kinematics and Body Velocities

Assuming zero ambient wind ($[u_w, v_w, w_w]^T = \mathbf{0}$), the air-relative velocity components $[u_r, v_r, w_r]^T$ are identical to the body-frame inertial velocities $[u, v, w]^T$:

$$\begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{31}$$

These components are evaluated from airspeed V_a, angle of attack α, and sideslip angle β:

$$u = u_r = V_a \cos(\alpha)\cos(\beta), \tag{32}$$

$$v = v_r = V_a \sin(\beta), \tag{33}$$

$$w = w_r = V_a \sin(\alpha)\cos(\beta). \tag{34}$$

Under zero-wind conditions, relative body angular rates $[p_r, q_r, r_r]^T$ simplify directly to the body angular rates $\boldsymbol{\omega}^b = [p, q, r]^T$.

B. Translational Accelerations

The body-frame translational velocity derivatives $[\dot{u}, \dot{v}, \dot{w}]^T$ follow Newton's second law in a rotating frame:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \tag{35}$$

where m is the total UAV mass, and $\boldsymbol{f} = [f_x, f_y, f_z]^T$ represents the net external force vector along the body axes.

C. External Force Breakdown

The total force vector $\boldsymbol{f}$ combines gravitational ($\boldsymbol{f}^g$), propulsion ($\boldsymbol{f}^p$), and aerodynamic ($\boldsymbol{f}^a$) components:

$$\boldsymbol{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \boldsymbol{f}^g + \boldsymbol{f}^p + \boldsymbol{f}^a = \begin{bmatrix} f_x^g \\ f_y^g \\ f_z^g \end{bmatrix} + \begin{bmatrix} f_x^p \\ f_y^p \\ f_z^p \end{bmatrix} + \begin{bmatrix} f_x^a \\ f_y^a \\ f_z^a \end{bmatrix}. \tag{36}$$

a) Gravitational Forces ($\boldsymbol{f}^g$): Projected into the body frame using pitch (θ) and roll (ϕ) angles,

$$\boldsymbol{f}^g = \begin{bmatrix} f_x^g \\ f_y^g \\ f_z^g \end{bmatrix} = \begin{bmatrix} -mg\sin(\theta) \\ mg\cos(\theta)\sin(\phi) \\ mg\cos(\theta)\cos(\phi) \end{bmatrix}, \tag{37}$$

where g is the gravitational acceleration.

b) Propulsion Forces ($\boldsymbol{f}^p$): Assuming thrust acts exclusively along the body x-axis,

$$\boldsymbol{f}^p = \begin{bmatrix} f_x^p \\ f_y^p \\ f_z^p \end{bmatrix} = \begin{bmatrix} T_p \\ 0 \\ 0 \end{bmatrix}, \tag{38}$$

where thrust $T_p = C_p \delta_t$ is governed by the throttle command $\delta_t \in [0, 1]$ and constant motor gain $C_p = 5.9$.

c) Aerodynamic Forces ($\boldsymbol{f}^a$): The longitudinal aerodynamic forces f_x^a and f_z^a are transformed from wind-frame drag (F_{drag}) and lift (F_{lift}) forces,

$$\begin{bmatrix} f_x^a \\ f_z^a \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} -F_{\text{drag}} \\ -F_{\text{lift}} \end{bmatrix}. \tag{39}$$

The drag, lift, and lateral aerodynamic forces are given by:

$$F_{\text{drag}} = \frac{1}{2}\rho V_a^2 S \left[C_{D_0} + C_{D_\alpha}\alpha + C_{D_q}\frac{cq}{2V_a} + C_{D_{\delta_e}}\delta_e \right], \tag{40}$$

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S \left[C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}\frac{cq}{2V_a} + C_{L_{\delta_e}}\delta_e \right], \tag{41}$$

$$f_y^a = \frac{1}{2}\rho V_a^2 S \left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a \right], \tag{42}$$

yielding the complete aerodynamic force vector $\boldsymbol{f}^a$:

$$\boldsymbol{f}^a = \begin{bmatrix} f_x^a \\ f_y^a \\ f_z^a \end{bmatrix} = \begin{bmatrix} -\cos(\alpha)F_{\text{drag}} + \sin(\alpha)F_{\text{lift}} \\ \frac{1}{2}\rho V_a^2 S \left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a \right] \\ -\sin(\alpha)F_{\text{drag}} - \cos(\alpha)F_{\text{lift}} \end{bmatrix}, \tag{43}$$

where ρ is air density, S is wing planform area, b is wingspan, c is mean aerodynamic chord, and δ_a, δ_e are aileron and elevator deflections.

D. Aerodynamic Moments

The external body moments $\mathbf{M}^b = [l, m, n]^T$ driving angular acceleration vector $\dot{\boldsymbol{\omega}}^b = [\dot{p}, \dot{q}, \dot{r}]^T$ are computed as:

$$l = \frac{1}{2}\rho V_a^2 S b \left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{bp}{2V_a} + C_{l_r}\frac{br}{2V_a} + C_{l_{\delta_a}}\delta_a \right], \tag{44}$$

$$m = \frac{1}{2}\rho V_a^2 S c \left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta_e}}\delta_e \right], \tag{45}$$

$$n = \frac{1}{2}\rho V_a^2 S b \left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{bp}{2V_a} + C_{n_r}\frac{br}{2V_a} + C_{n_{\delta_a}}\delta_a \right]. \tag{46}$$

APPENDIX B
PRIOR DEGRADATION VALUES

Table VI contains the aerodynamics coefficients of the Skywalker-X8 taken from [40], [41]. The selected coefficients are the rotary derivatives (C_{l_p}, C_{n_r}, C_{n_p}), the sideslip couplings (C_{Y_β}, C_{l_β}, $C_{D_{\beta^2}}$), the drag-polar terms (C_{D_0}, C_{D_α}), and the primary control effectiveness derivatives ($C_{L_{\delta_e}}$, $C_{l_{\delta_a}}$). These are difficult to measure accurately, requiring dedicated dynamic manoeuvres and being sensitive to sensor noise and complex airflow behaviour. The untouched coefficients are dominated by the pitch-axis derivatives (C_{m_α}, C_{m_q}, C_{L_α}) and baseline trim terms, which standard flight-test identification recovers with high confidence.

TABLE VI: Aerodynamic coefficients with perturbed values shown where applicable. Percentages are the change in coefficient magnitude, the sign of each coefficient is preserved.

Coefficient	**Real**	**Mild**	**Severe**
C_{L0}	0.0867	–	–
C_{L_α}	4.0203	–	–
$C_{L_{\delta_e}}$	0.2781	0.18135 (35%)	0.06207 (78%)
C_{L_q}	3.8700	–	–
C_{D0}	0.0197	0.02774 (41%)	0.03903 (98%)
C_{D_α}	0.0791	0.04019 (49%)	0.01046 (87%)
$C_{D_{\alpha^2}}$	1.0555	–	–
C_{D_β}	−0.0058	–	–
$C_{D_{\beta^2}}$	0.1478	0.20999 (42%)	0.28428 (92%)
$C_{D_{\delta_e}}$	0.0633	–	–
C_{Y0}	0.0000	–	–
C_{Y_β}	−0.2239	−0.15744 (30%)	−0.05012 (78%)
C_{Y_p}	−0.1374	–	–
C_{Y_r}	0.0839	–	–
$C_{Y_{\delta_a}}$	0.0433	–	–
C_{l0}	0.0000	–	–
C_{l_β}	−0.0849	−0.11347 (34%)	−0.15286 (80%)
C_{l_p}	−0.4042	−0.59531 (47%)	−0.77604 (92%)
C_{l_r}	0.0555	–	–
$C_{l_{\delta_a}}$	0.1202	0.07479 (38%)	0.00347 (97%)
C_{m0}	0.0227	–	–
C_{m_α}	−0.4629	–	–
C_{m_q}	−1.3012	–	–
$C_{m_{\delta_e}}$	−0.2292	–	–
C_{n0}	0.0000	–	–
C_{n_β}	0.0283	–	–
C_{n_p}	0.0044	0.00635 (44%)	0.00863 (96%)
C_{n_r}	−0.0720	−0.09573 (33%)	−0.13760 (91%)
$C_{n_{\delta_a}}$	−0.00339	–	–
C_p	5.9	–	–

APPENDIX C
TD3 IMPLEMENTATION DETAILS

The training is conducted on a NVIDIA GeForce RTX 2060 with 8GB of memory, a consumer available card. This makes our results easily reproducible.

TABLE VII: TD3 hyperparameters (Fujimoto et al., 2018).

Parameter	Value
Actor network	MLP, 2 hidden layers (300, 300), ReLU, `tanh` output
Critic network (twin)	MLP, 3 hidden layers (200, 200, 200), ReLU
Discount factor γ	0.99
Target network update rate τ	0.05
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Policy update delay	2
Target policy smoothing noise (std)	0.2
Target policy smoothing noise clip	0.5
Exploration noise (std)	0.1
Batch size	256
Replay buffer size	10^6
Warmup (random-action) steps	10,000
Total training steps	700,000

APPENDIX D
OCCASIONAL FAILURES DURING CONTROL

TABLE VIII: Failure-mode characterization, across the six physics-prior conditions (6 conditions $\times$ 10 seeds). Min/Max are the single best/worst settled attitude error observed across all episodes. Divergence rate is the share of episodes with settled attitude error above 10°, with 95% stratified bootstrap CI over seeds.

Controller	Min (deg)	Max (deg)	Divergence rate (%) [95% CI]
All targets			
TD3 policy alone	0.03	39.35	1.7 [1.3, 2.2]
CEM (H=5)	0.01	115.09	31.5 [31.1, 31.8]
CEM $+\pi$	0.01	116.84	32.8 [32.3, 33.4]
CEM $+Q$	0.05	88.59	1.0 [0.7, 1.5]
Hybrid Controller	0.05	66.89	1.1 [0.8, 1.5]
Easy targets			
TD3 policy alone	0.03	3.68	0.0 [0.0, 0.0]
CEM (H=5)	0.01	102.07	7.1 [7.1, 7.1]
CEM $+\pi$	0.01	100.96	7.1 [7.1, 7.1]
CEM $+Q$	0.05	2.81	0.0 [0.0, 0.0]
Hybrid Controller	0.05	2.45	0.0 [0.0, 0.0]
Hard targets			
TD3 policy alone	0.05	39.35	4.1 [3.1, 5.2]
CEM (H=5)	0.06	115.09	65.6 [64.8, 66.4]
CEM $+\pi$	0.05	116.84	68.8 [67.6, 70.1]
CEM $+Q$	0.06	88.59	2.5 [1.6, 3.5]
Hybrid Controller	0.05	66.89	2.7 [1.8, 3.6]

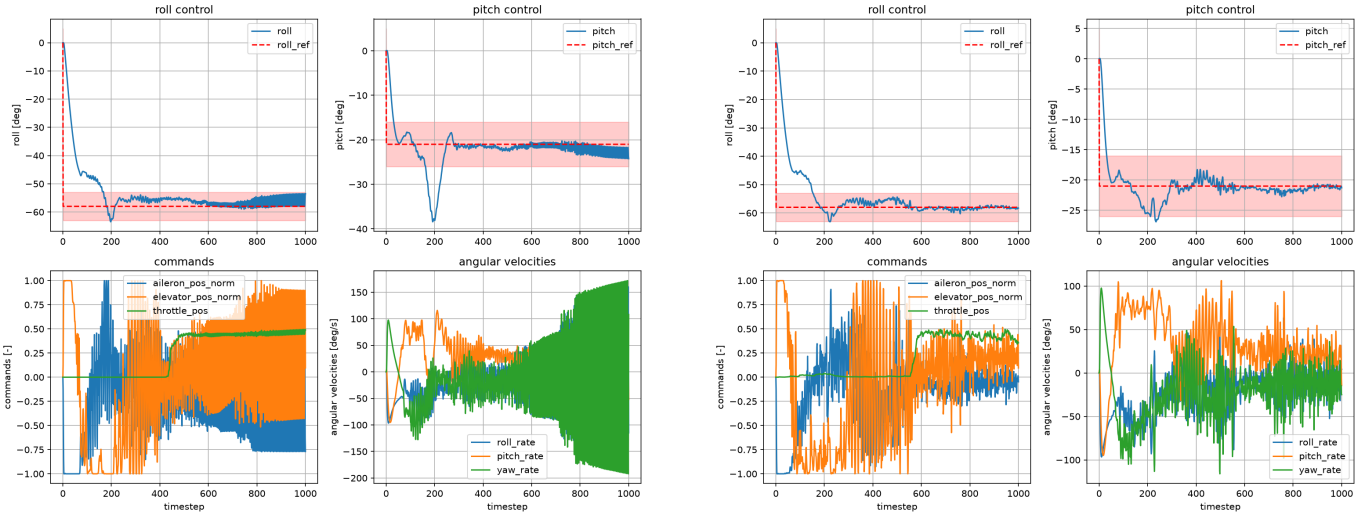
In Section VI-C, we show how well the hybrid controller and its variant work. However Table IV shows that the CEM-MPC and CEM-MPC+π has relatively high errors even for easy targets. We aimed to study the reason for this failure. The Table VIII isolates the catastrophic-failure behavior of the different control methods tested.

Interestingly, we observed that without the critic boostrap, the CEM-MPC controllers can completely shoot off-course, with errors reaching over 100° for easy and hard targets. However, we notice a difference in the occurrence rate of these events. Only 7% for easy targets, which explains why the mean tracking error for these two methods still remain relatively low in this case. But on hard targets, we can see a clear pattern: 65% to 68% of the episodes diverged. Therefore we can conclude that in this case, these variants are completely unable to solve the task.

Concerning the controller variants that do have the critic bootstrap, the divergence never occurs for easy targets, and only 2.5% to 2.7% of hard-target episodes diverge, which is not alarming.

APPENDIX E
ATTITUDE TRACKING EXAMPLES

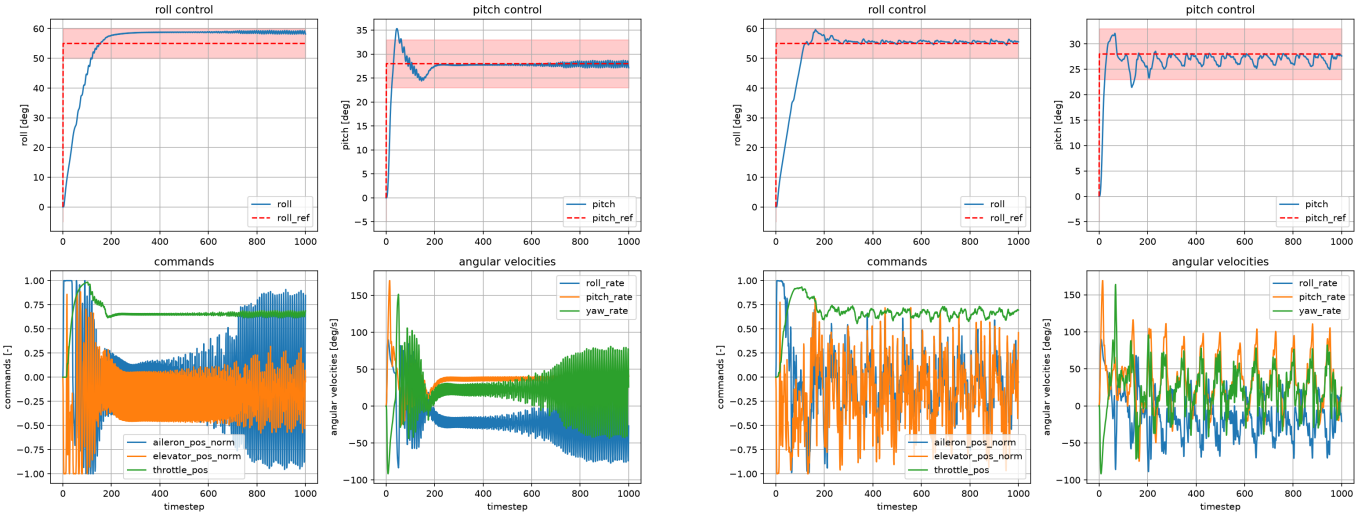
This section contains the plots that illustrates the tracking quality of various target references, comparing the TD3 agent to the Hybrid Controller. Figure 7 shows tracking quality for a difficult dive (high negative pitch) and hard roll objective. Figure 8 shows the same comparison but for a difficult altitude climb (high positive pitch) and hard roll objective. Finally, Figure 9 shows tracking quality of nominal targets, we relatively low roll and pitch objectives.



(a) Policy only

(b) Hybrid Controller

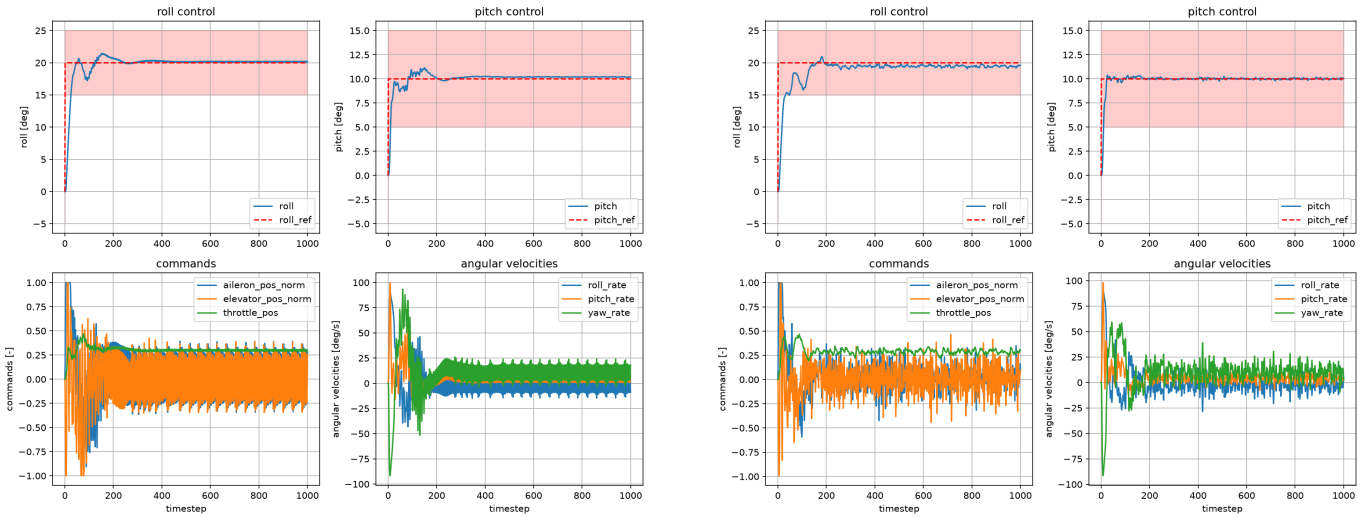
Fig. 7: **Hybrid Controller vs Policy only for hard dive attitude control.** Red dashed lines are the references, roll=-58° and pitch=-21°. The red area around the reference line corresponds to a ± 5° error bound. The plot shows that the Hybrid Controller is more stable and manages to converge better around the target for both roll and pitch, specifically for a very difficult dive and aggressive roll maneuver.



(a) Policy only

(b) Hybrid Controller

Fig. 8: **Hybrid Controller vs Policy only for hard climb attitude control.** Red dashed lines are the references, roll=55° and pitch=28°. The red area around the reference line corresponds to a ± 5° error bound. The plot shows that the policy is quite stable for pitch, but fails to converge on the roll objective. The Hybrid Controller manages to converge on roll, but oscillates around the target for pitch. The settled attitude error is slightly lower for the Hybrid Controller.



(a) Policy only

(b) Hybrid Controller

Fig. 9: **Hybrid Controller vs Policy only for easy attitude control.** Red dashed lines are the references, roll=20° and pitch=10°. The red area around the reference line corresponds to a ± 5° error bound. The plot shows that both controllers converge exactly at the target, which confirms that on easy targets, both controllers meet the objective with no issues.

APPENDIX F
TABLES WITH NUMERICAL RESULTS

This section contains all the tables with the numerical results that are presented in the Section VI. Table X's results are explained in Section VI-B, and Tables IV and XI are explained in Section VI-C.

TABLE IX: TD3 training-regime comparison on the full target grid. Brackets give 95% percentile-bootstrap CI over the seeds. Lower is better, best results in bold.

	Settled attitude error (deg) [95% CI]		
Training regime	Median	IQM	Mean
All targets			
Model Free TD3	1.08 [0.92, 1.69]	1.10 [0.97, 1.45]	1.19 [1.01, 1.40]
Model Environment TD3	1.22 [1.04, 1.96]	1.33 [1.08, 1.77]	1.44 [1.18, 1.73]
Parallelized Model Environment TD3	**0.88** [0.69, 1.28]	**0.93** [0.76, 1.18]	**0.95** [0.81, 1.11]
Easy targets			
Model Free TD3	0.74 [0.66, 0.95]	0.74 [0.68, 0.89]	0.79 [0.70, 0.92]
Model Environment TD3	0.94 [0.65, 1.13]	0.95 [0.76, 1.12]	0.96 [0.81, 1.12]
Parallelized Model Environment TD3	**0.54** [0.48, 0.62]	**0.54** [0.50, 0.60]	**0.55** [0.50, 0.59]
Hard targets			
Model Free TD3	1.37 [1.17, 2.16]	1.42 [1.21, 1.94]	1.59 [1.29, 1.95]
Model TD3	1.47 [1.31, 2.87]	1.65 [1.36, 2.50]	1.92 [1.43, 2.51]
Parallelized Model Environement TD3	**1.33** [0.84, 1.97]	**1.32** [0.94, 1.81]	**1.36** [1.06, 1.68]

TABLE X: Policy performance across the six physics-prior conditions. Each entry aggregates $n = 10$ independently seeded TD3 policies, the per-seed score is the mean settled attitude error. Brackets give 95% percentile-bootstrap CI over the seeds. Lower is better, best results in bold and second best underlined.

	Settled attitude error (deg) [95% CI]		
Prior condition	Median	IQM	Mean
All targets			
True (frozen)	**0.91** [0.75, 1.22]	<u>0.97</u> [0.79, 1.17]	<u>0.97</u> [0.83, 1.12]
True (trainable)	<u>1.00</u> [0.74, 1.11]	**0.95** [0.79, 1.09]	**0.94** [0.82, 1.06]
Mild degr. (frozen)	1.65 [1.13, 2.30]	1.69 [1.21, 2.18]	1.67 [1.31, 2.03]
Mild degr. (trainable)	1.18 [0.81, 1.52]	1.15 [0.86, 1.49]	1.20 [0.95, 1.46]
Severe degr. (frozen)	1.87 [1.31, 2.20]	1.79 [1.40, 2.14]	1.77 [1.48, 2.06]
Severe degr. (trainable)	1.10 [0.83, 1.37]	1.09 [0.91, 1.36]	1.15 [0.97, 1.35]
Easy targets			
True (frozen)	<u>0.56</u> [0.51, 0.61]	<u>0.55</u> [0.52, 0.60]	<u>0.56</u> [0.52, 0.59]
True (trainable)	0.62 [0.50, 0.74]	0.62 [0.51, 0.72]	0.62 [0.54, 0.70]
Mild degr. (frozen)	**0.49** [0.43, 0.57]	**0.50** [0.43, 0.57]	**0.50** [0.44, 0.57]
Mild degr. (trainable)	0.56 [0.52, 0.63]	0.58 [0.52, 0.65]	0.59 [0.53, 0.67]
Severe degr. (frozen)	0.81 [0.73, 1.00]	0.82 [0.73, 0.96]	0.84 [0.75, 0.95]
Severe degr. (trainable)	0.58 [0.53, 0.64]	0.58 [0.53, 0.63]	0.58 [0.55, 0.62]
Hard targets			
True (frozen)	<u>1.52</u> [0.97, 2.15]	<u>1.55</u> [1.07, 2.04]	<u>1.56</u> [1.21, 1.90]
True (trainable)	**1.39** [1.13, 1.73]	**1.41** [1.16, 1.65]	**1.40** [1.19, 1.60]
Mild degr. (frozen)	3.26 [1.88, 4.75]	3.34 [2.19, 4.48]	3.31 [2.46, 4.15]
Mild degr. (trainable)	1.99 [1.16, 2.70]	1.89 [1.32, 2.67]	2.04 [1.49, 2.67]
Severe degr. (frozen)	3.25 [1.77, 4.19]	3.05 [2.04, 4.03]	3.08 [2.34, 3.81]
Severe degr. (trainable)	1.90 [1.25, 2.39]	1.83 [1.41, 2.40]	1.94 [1.53, 2.40]

TABLE XI: Controller performance for the true frozen prior dynamics model, $n = 10$ seeds. Lower is better, best results in bold and second best underlined.

	Settled attitude error (deg) [95% CI]		
Controller	Median	IQM	Mean
All targets			
TD3 policy alone	0.91 [0.75, 1.22]	0.97 [0.79, 1.17]	0.97 [0.83, 1.12]
CEM-MPC (H=5)	19.50 [18.94, 20.60]	19.61 [18.89, 20.44]	19.60 [18.86, 20.29]
CEM-MPC $+\pi$	19.63 [18.87, 20.78]	19.76 [18.97, 20.87]	20.05 [19.24, 21.04]
CEM-MPC $+Q$	<u>0.65</u> [0.57, 0.75]	<u>0.67</u> [0.58, 0.74]	<u>0.66</u> [0.59, 0.73]
Hybrid Controller	**0.64** [0.56, 0.69]	**0.64** [0.57, 0.68]	**0.63** [0.58, 0.68]
Easy targets			
TD3 policy alone	0.56 [0.51, 0.61]	0.55 [0.52, 0.60]	0.56 [0.52, 0.59]
CEM-MPC (H=5)	4.79 [4.77, 4.81]	4.79 [4.77, 4.81]	4.79 [4.78, 4.80]
CEM-MPC $+\pi$	4.79 [4.76, 4.80]	4.79 [4.76, 5.08]	4.95 [4.77, 5.29]
CEM-MPC $+Q$	<u>0.50</u> [0.43, 0.56]	<u>0.50</u> [0.44, 0.57]	<u>0.51</u> [0.46, 0.57]
Hybrid Controller	**0.43** [0.40, 0.47]	**0.43** [0.40, 0.48]	**0.44** [0.41, 0.49]
Hard targets			
TD3 policy alone	1.52 [0.97, 2.15]	1.55 [1.07, 2.04]	1.56 [1.21, 1.90]
CEM-MPC (H=5)	40.13 [38.73, 42.78]	40.38 [38.64, 42.36]	40.34 [38.55, 41.99]
CEM-MPC $+\pi$	40.43 [37.84, 43.16]	40.49 [38.52, 43.37]	41.19 [39.09, 43.69]
CEM-MPC $+Q$	**0.93** [0.66, 1.06]	**0.90** [0.72, 1.04]	**0.87** [0.75, 0.99]
Hybrid Controller	<u>0.97</u> [0.79, 0.99]	<u>0.92</u> [0.81, 0.99]	<u>0.89</u> [0.81, 0.97]