Proceedings of Machine Learning Research vol vvv:1–22, 2024

NLBAC: A Neural ODEs-based Framework for Stable and Safe Reinforcement Learning

Liqun Zhao LIQUN.ZHAO@ENG.OX.AC.UK
Keyan Miao KEYAN.MIAO@ENG.OX.AC.UK
Konstantinos Gatsis KONSTANTINOS.GATSIS@ENG.OX.AC.UK
Antonis Papachristodoulou ANTONIS@ENG.OX.AC.UK
Department of Engineering Science, University of Oxford

Abstract

Reinforcement learning (RL) excels in applications such as video games and robotics, but ensuring safety and stability remains challenging when using RL to control real-world systems where using model-free algorithms suffering from low sample efficiency might be prohibitive. This paper first provides safety and stability definitions for the RL system, and then introduces a Neural ordinary differential equations-based Lyapunov-Barrier Actor-Critic (NLBAC) framework that leverages Neural Ordinary Differential Equations (NODEs) to approximate system dynamics and integrates the Control Barrier Function (CBF) and Control Lyapunov Function (CLF) frameworks with the actor-critic method to assist in maintaining the safety and stability for the system. Within this framework, we employ the augmented Lagrangian method to update the RL-based controller parameters. Additionally, we introduce an extra backup controller in situations where CBF constraints for safety and the CLF constraint for stability cannot be satisfied simultaneously. Simulation results[1] demonstrate that the framework leads the system to approach the desired state and allows fewer violations of safety constraints with better sample efficiency compared to other methods.

Keywords: Safe reinforcement learning, control barrier function, control Lyapunov function, neural ordinary differential equations, model-based reinforcement learning

1. Introduction

The notable achievements of reinforcement learning (RL) in solving sophisticated sequential decision-making problems, including Atari games, have prompted researchers to investigate its applicability in real-world situations, such as robotics (Wang and Boyle 2023; Kober et al. 2013; Wang et al. 2023b; He et al. 2022). However, random explorations involved in most of the RL algorithms may lead to actions with unsafe or detrimental consequences. Moreover, maintaining stability is paramount in these scenarios. Stability, which means the state will remain close or converge to the equilibrium, represents a fundamental requirement for a control system (Han et al. 2020, 2023).
Safety in RL has become a focal point in current research activity. Existing safe RL approaches primarily use the constrained Markov decision process (CMDP) framework (Altman 1999), and various techniques have been proposed, including primal-dual update (Tessler et al. 2018; Paternain et al. 2019), and trust region policy optimization (Achiam et al. 2017; Yang et al. 2020). However, prior investigations often define safety constraints based on the cumulative cost of an entire trajectory, rather than on the cost at each timestep. As a result, safety violations at specific timesteps are not prevented (Ma et al. 2021a,b, 2022).

1. Code available in the GitHub repository: https://github.com/LiqunZhao/Neural-ordinary-differential-equations-based-Lyapunov-Barrier-Actor-Critic-NLBAC

Recently, an increasing focus has emerged on integrating control-theoretic approaches with RL to help ensure the safety for the system (Hewing et al. 2020; Koller et al. 2018). Concepts like safe set algorithm (SSA) (Ma et al. 2021b; Wei and Liu 2019; Zhao et al. 2023b), implicit safe set algorithm (ISSA) (Zhao et al. 2021), and control barrier function (CBF) (Cheng et al. 2019; Emam et al. 2021; Wang et al. 2022; Tan et al. 2023; do Nascimento et al. 2023; Ames et al. 2019; Tan and Dimarogonas 2021; Dawson et al. 2022a) have been used as safety constraints to help RL training maintain safety. However, in many previous studies such as (Cheng et al. 2019), a physics-based control-affine nominal model of the controlled system is required, which could potentially restrict the practical applicability of the algorithms in real-world scenarios since in numerous instances, the exact system dynamics or even a nominal model is unavailable.

The Lyapunov framework is a widely employed tool in control and dynamical systems, and has recently found application in the field of learning (Cao et al. 2023a,b; Zhang et al. 2023; Miao and Gatsis 2023a). Certain previous investigations have devised approaches for constructing Lyapunov functions in the context of RL training (Chow et al. 2018), whereas (Berkenkamp et al. 2017) suggested a method utilizing Lyapunov functions to guide safe exploration. In the work of (Han et al. 2020) and (Chang and Gao 2021), Lyapunov functions are employed in model-free RL to help guarantee the stability for a range of systems. However, the successes of these model-free algorithms in terms of performance often come with the drawback of being data-intensive. This is due to their requirement for a significant number of interactions with the environment.

Through learning a model of the environment and subsequently employing it as a surrogate for the real system, model-based reinforcement learning algorithms hold the promise of achieving considerably higher sample efficiency compared to model-free algorithms. In recent explorations within the domain of model-based RL, researchers have incorporated methods like probabilistic models (Chua et al. 2018) and ensembles (Kurutach et al. 2018) to mitigate model bias. However, when employing neural networks to directly approximate the transition, there is an implicit assumption that all data is collected with the same discretization step, and all predictions can only be made with that discretization step as well. Inspired by the fact that many physical systems are modeled by differential equations, (Alvarez et al. 2020) and (Du et al. 2020) leverage neural ordinary differential equations (NODEs) (Chen et al. 2018) to approximate the real dynamics. They then utilize the learned model for predictions, subsequently enhancing value estimation and optimizing the RL-based controller. However, earlier studies that integrate NODEs with RL do not take the safety and stability of the controlled system into consideration, and the tendency for approximation errors to accumulate and propagate remains, particularly in scenarios with long planning horizons.

Our contributions. Our primary contributions include: 1. We introduce a primary controller that combines the CBF and CLF frameworks with the Soft Actor-Critic (SAC) algorithm (Haarnoja et al. 2018) for a system whose dynamics is approximated by NODEs. Predictions from NODEs are used to formulate CBF and CLF constraints for safety and stability, respectively, and therefore the planning horizon length is equal to the relative degree of the CBF, which alleviates the problem of large error propagation that is challenging in the field of model-based RL since the relative degree is typically not high. Moreover, we efficiently update the controller parameters through the augmented Lagrangian method that simplifies the hyperparameter tuning for different learning rates and can provide stable training convergence, which might be challenging with the primal-dual update commonly used in previous studies. 2. We propose an RL-based backup controller for scenarios where satisfying both safety and stability constraints simultaneously is not feasible. The backup controller's constraints are formulated using CBFs to help the system achieve and maintain safety. 3.

We introduce a framework called Neural ordinary differential equations-based Lyapunov-Barrier Actor-Critic (NLBAC) and assess its performance on two simulation scenarios. Our experiments indicate that the framework assists in guaranteeing the safety and stability of the system with higher sample efficiency.

2. Background

2.1. Preliminaries

2.1.1. MARKOV DECISION PROCESS

Consider a Markov decision process (MDP) with a control affine dynamics

$$\dot{x}_t = f(x_t) + g(x_t)u_t, \tag{1}$$

defined by the tuple $\mathcal{M}$ which is $(\mathcal{X}, \mathcal{U}, \mathcal{T}, r, c, \gamma, \gamma_c)$. $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ are state and control signal spaces, and $x_{t_k} \in \mathcal{X}$ is the state at timestep t_k, $u_{t_k} \in \mathcal{U}$ is the control signal at timestep t_k. $\mathcal{T}$ denotes the dynamics, and by continually applying the control signal u_{t_k} between consecutive timesteps t_k and t_{k+1}, it can be described as

$$x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} \big(f(x_\chi) + g(x_\chi)u_{t_k}\big)d\chi, \tag{2}$$

following (1). The reward and cost are denoted as r and c, and γ and γ_c are the discount factors. The transition probability is defined as $P(x_{t_{k+1}}|x_{t_k}, u_{t_k}) \triangleq I_{\{x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} \left(f(x_\chi)+g(x_\chi)u_{t_k}\right)d\chi\}}$, where $I_{\{x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} \left(f(x_\chi)+g(x_\chi)u_{t_k}\right)d\chi\}}$ is an indicator function that equals 1 if $x_{t_{k+1}}$ satisfies (2) given x_{t_k} and u_{t_k}, and 0 otherwise. Following (Han et al. 2020), the closed-loop transition probability is denoted as $P_\pi(x_{t_{k+1}}|x_{t_k}) \triangleq \int_{\mathcal{U}} \pi(u_{t_k}|x_{t_k})P(x_{t_{k+1}}|x_{t_k}, u_{t_k})du_{t_k}$. Additionally, the closed-loop state distribution at timestep t_k is denoted by $\upsilon(x_{t_k}|\rho, \pi, t_k)$, which is calculated as $\upsilon(x_{t_{k+1}}|\rho, \pi, t_{k+1}) = \int_{\mathcal{X}} P_\pi(x_{t_{k+1}}|x_{t_k})\upsilon(x_{t_k}|\rho, \pi, t_k)dx_{t_k}$, $\forall t_k \in \mathbb{N}$ by using the closed-loop transition probability. Here, $\upsilon(x_{t_0}|\rho, \pi, t_0) = \rho$ represents the initial state distribution.

2.1.2. DEFINITION OF SAFETY

In this subsection, we outline the conditions for maintaining system safety in closed loop. With the assumption of m distinct safety constraints, the system is considered safe if the following condition holds for all $i = 1, \ldots, m$:

$$h_i(x_{t_k}) \geq 0 \quad \forall t_k \geq t_0. \tag{3}$$

Here $h_i : \mathbb{R}^n \to \mathbb{R}$ is a function corresponding to the i^{th} safety constraint, and a safe set $\mathcal{C}_{i,0} \subset \mathbb{R}^n$ is defined by the super-level set of h_i as follows:

$$\mathcal{C}_{i,0} = \{x_{t_k} \in \mathcal{X} | h_i(x_{t_k}) \geq 0\}. \tag{4}$$

We require the system state to remain within this set $\mathcal{C}_{i,0}$, meaning the safe set $\mathcal{C}_{i,0}$ should be forward invariant. The control barrier function framework can be used to ensure the forward invariance of the safe set, and here we first introduce the definition of the relative degree:

Definition 1 (Relative Degree (Chakrabarty et al. 2016)) *The output $y_{t_k} = o(x_{t_k})$ of the system (2) is said to have relative degree r iff*

$$\begin{aligned} y_{t_{k+i}} &= o_i(x_{t_k}) \qquad\qquad \forall i \in [0, r-1] \\ y_{t_{k+r}} &= o_r(x_{t_k}, u_{t_k}) \end{aligned} \tag{5}$$

which means r is the steps of delay for the control signal u_{t_k} to appear in the output y_{t_k}. Similarly, the constraint $h_i(x_{t_k}) \geq 0$ is also said to be of relative degree r in this paper, and it is said to have a high relative degree when $r > 1$.
The discrete-time CBF can be developed for cases with relative degree r (Xiong et al. 2022), and a list of functions can be defined based on h_i:

$$\begin{aligned} \Phi_{i,0}(x_{t_k}) &:= h_i(x_{t_k}) \\ \Phi_{i,1}(x_{t_k}) &:= \Delta\Phi_{i,0}(x_{t_k}, u_{t_k}) + \kappa_{i,1}(\Phi_{i,0}(x_{t_k})) \\ &\vdots \\ \Phi_{i,r}(x_{t_k}) &:= \Delta\Phi_{i,r-1}(x_{t_k}, u_{t_k}) + \kappa_{i,r}(\Phi_{i,r-1}(x_{t_k})) \end{aligned} \tag{6}$$

where $\Delta\Phi_{i,j}(x_{t_k}, u_{t_k}) := \Phi_{i,j}(x_{t_{k+1}}) - \Phi_{i,j}(x_{t_k})$, $j = 0, 1, \ldots, r-1$, and $\kappa_{i,j}(\cdot)$ are class $\mathcal{K}$ functions satisfying $\kappa_{i,j}(x) < x$, where $j = 1, \ldots, r$, for the i^{th} safety constraints. Furthermore, with these functions, a list of sets can be defined:

$$\begin{aligned} \mathcal{C}_{i,0} &:= \{x_{t_k} \in \mathcal{X} | \Phi_{i,0}(x_{t_k}) \geq 0\} \\ \mathcal{C}_{i,1} &:= \{x_{t_k} \in \mathcal{X} | \Phi_{i,1}(x_{t_k}) \geq 0\} \\ &\vdots \\ \mathcal{C}_{i,r-1} &:= \{x_{t_k} \in \mathcal{X} | \Phi_{i,r-1}(x_{t_k}) \geq 0\}, \end{aligned} \tag{7}$$

and then the definition of the discrete-time CBF can be given as follows:

Definition 2 (Discrete-time Control Barrier Function (Xiong et al. 2022)) *For the system (2), the function $h_i : \mathbb{R}^n \to \mathbb{R}$ is called a discrete-time CBF of relative degree r if there exists $\Phi_{i,j}(x_{t_k})$, $j = 0, 1, \ldots, r$ defined by (6) and $\mathcal{C}_{i,j}$, $j = 0, 1, \ldots, r-1$ defined by (7) such that*

$$\Phi_{i,r}(x_{t_k}) \geq 0, \tag{8}$$

holds for all $x_{t_k} \in \bigcap_{j=0}^{r-1} \mathcal{C}_{i,j}$.

Given the existence of a CBF, a controller satisfying (8) will make the set $\bigcap_{j=0}^{r-1} \mathcal{C}_{i,j}$ forward invariant. Consequently, safety is maintained if there exists a controller such that $\forall i \in [1, m]$, (8) holds for all $x_{t_k} \in \bigcap_{i=1}^{m} \bigcap_{j=0}^{r-1} \mathcal{C}_{i,j}$.

2.1.3. DEFINITION OF STABILITY

In a stabilization task, our objective is to find a controller capable of driving the system state finally to the equilibrium, which is also called the desired state. In pursuit of this objective, we define the instantaneous cost signal as $c(x_{t_k}, u_{t_k}) = \left\| x_{t_{k+1}} - x_{\text{desired}} \right\|$ where $x_{t_{k+1}}$ is the next state following (2) given the state x_{t_k} and control signal u_{t_k}, and x_{desired} denotes the desired state (i.e., equilibrium). Similarly, for a tracking task, the control signal can be denoted as $c(x_{t_k}, u_{t_k}) = \left\| x_{t_{k+1}} - r_s \right\|$

where r_s is the reference signal needs to be tracked.
Because we explore the stability of a closed-loop system under a nondeterministic RL-based controller π, we combine $\pi(u_{t_k}|x_{t_k})$, which is a Gaussian distribution in this paper, with the cost signal $c(x_{t_k}, u_{t_k})$ to formulate the cost function under the controller π as follows:

$$c_\pi(x_{t_k}) = \mathbb{E}_{u_{t_k}\sim\pi} c(x_{t_k}, u_{t_k}) = \mathbb{E}_{u_{t_k}\sim\pi}[\left\|x_{t_{k+1}} - x_{\text{desired}}\right\|]. \tag{9}$$

The cost function $c_\pi(x_{t_k})$ denotes the expected value of the norm representing the difference between the next state $x_{t_{k+1}}$ following (2), and the desired state x_{desired}. This expectation is taken over u_{t_k} sampled from the distribution $\pi(u_{t_k}|x_{t_k})$. Naturally, we expect that the value of $c_\pi(x_{t_k})$ decreases as t_k increases to drive the system state towards the equilibrium, and achieve $c_\pi(x_{t_k}) = 0$ finally, which means the state reaches the equilibrium. However, given that the control signal is drawn from a Gaussian distribution, the state at timestep t_k is also distributed. Consequently, we adopt the stability definition outlined in (Han et al. 2020) which employs the concept of "expected value" for $c_\pi(x_{t_k})$ within the proposed framework.

Definition 3 (Stability in Mean Cost) *Let $\upsilon(x_{t_k}|\rho, \pi, t_k)$ denote the closed-loop state distribution at timestep t_k. The equilibrium of a system is said to be stable in mean cost if there exists a positive constant b such that for any initial state $x_{t_0} \in \{x_{t_0}|c_\pi(x_{t_0}) < b\}$, the condition*

$$\lim_{t_k\to\infty} \mathbb{E}_{x_{t_k}\sim\upsilon}\left[c_\pi(x_{t_k})\right] = 0 \tag{10}$$

holds. If b is arbitrarily large, the equilibrium is globally stable in mean cost.

2.1.4. INTRODUCTION TO NEURAL ORDINARY EQUATIONS (NODES)

According to (6), it is evident that applying CBFs as safety constraints mandates an understanding of the system dynamics to obtain $\{x_{t_{k+1}}, x_{t_{k+2}}, \ldots, x_{t_{k+r}}\}$, where r is the relative degree, given x_{t_k} and u_{t_k}. However, in numerous instances, the precise system dynamics or even a necessary nominal model remain undisclosed, compelling the approximation of actual dynamics. Neural networks have emerged as crucial tools for estimating the dynamics.
Residual Networks (ResNets), proposed by (He et al. 2016), revolutionized neural networks by overcoming depth limitations using skip connections, significantly boosting their performance. This period also saw the rise in considering neural network hidden layers as states within a dynamical system. (Chen et al. 2018) expanded on this notion by introducing an ODE to model continuous dynamics within neural networks.

Definition 4 *A Neural ODE is a system of the form*

$$\begin{cases} \dot{\hat{x}}_{t_k} = \mathcal{F}_\psi\left(t_k, \hat{x}_{t_k}, u_{t_k}\right) \\ \hat{x}_{t_0} = x_{t_0} \end{cases} \tag{11}$$

where $\mathcal{S} := [t_0, t_f]$ is the depth domain and $\mathcal{F}$ is a neural network called ODENet which is chosen as a part of the machine learning model with parameter ψ.

The solution of this ODE at timestep t_f from an initial value $\hat{x}_{t_0}$ is $\hat{x}_{t_f}$, obtained with a differential equation solver by means of a specific solution scheme according to desired accuracy.
Vanilla Neural ODEs do not include the term u_{t_k} since their focus lies in the application of Neural

ODEs to classic machine learning problems, such as classification, considering it as an autonomous system. However, given Neural ODEs' origin in dynamical systems, they naturally lend themselves well to related applications, including non-autonomous dynamical systems where control variable u is incorporated. For instance, they can be effectively utilized in tasks like trajectory planning and system identification (Liang et al. (2021); Miao and Gatsis (2023b); Djeumou et al. (2022)).
In this study, we focus on using Neural ODEs to approximate control-affine dynamics (1) where the dynamics remain unknown. Moreover, the framework of NODEs allows for simple incorporation of prior information (such as control-affine property). Therefore, here, $\mathcal{F}_\psi = f_{\psi_1}(t_k, \hat{x}_{t_k}) + g_{\psi_2}(t_k, \hat{x}_{t_k})u_{t_k}$. A crucial consideration in our numerical assessment is assuming a constant control signal throughout the state-control trajectory, i.e., $u_\chi = u_{t_k}, \chi \in [t_k, t_{k+1})$. This assumption holds true as the action is governed by an external policy before the system evolves for a single timestep. The inference of Neural ODEs is carried out by solving the initial value problem (IVP):

$$\hat{x}_{t_{k+1}} = \hat{x}_{t_k} + \int_{t_k}^{t_{k+1}} \mathcal{F}_\psi\left(\chi, \hat{x}_\chi, u_\chi\right) d\chi = \hat{x}_{t_k} + \int_{t_k}^{t_{k+1}} \left(\hat{f}_{\psi_1}(\chi, \hat{x}_\chi) + \hat{g}_{\psi_2}(\chi, \hat{x}_\chi)u_{t_k}\right) d\chi \quad (12)$$

which is used to estimate Equation (2). Then we can calculate $\{\hat{x}_{t_{k+1}}, \hat{x}_{t_{k+2}}, \ldots, \hat{x}_{t_{k+r}}\}$ by iteratively using (12) r times to approximate $\{x_{t_{k+1}}, x_{t_{k+2}}, \ldots, x_{t_{k+r}}\}$ required for constructing (8).
We note that, due to the fundamental nature of NODEs in learning the continuous form of system dynamics, they readily adapt to varying discretization steps required for different problems. Essentially, NODEs can compute states at any time point by merely altering the integration's time span. In contrast, conventional neural networks lack this adaptability, as they learn directly in a discrete manner and necessitate retraining with varying discretization steps.

2.2. Problem Statement Definition of the Safe and Stable Control Problem

Building upon the preceding subsections, similar to (Dawson et al. 2022b), we define the safe and stable control problem as follows:

Problem 1 (Safe and Stable Control Problem) *For system (2), given a unique desired state (equilibrium) $x_{desired}$, a set $\mathcal{X}_b = \{x_{t_0} | c_\pi(x_{t_0}) < b\}$ where b is an arbitrarily large positive number and x_{t_0} denotes the initial state, a set of unsafe states $\mathcal{X}_{unsafe} \subseteq \mathcal{X}$, and a set of safe states $\mathcal{X}_{safe} \subseteq \mathcal{X}$ such that $x_{desired} \in \mathcal{X}_{safe}$ and $\mathcal{X}_{safe} \cap \mathcal{X}_b \neq \emptyset$, find a controller π generating control signal sequence $\{u_{t_k}\}_{t_k \geq t_0}$ such that the state sequence $\{x_{t_k}\}_{t_k \geq t_0}$ satisfying (2) and $x_{t_0} \in \mathcal{X}_{safe} \cap \mathcal{X}_b$ has the following properties:*

- ***Safety**: $x_{t_k} \in \mathcal{X}_{safe} \quad \forall t_k \geq t_0$.*
- ***Stability in Mean Cost at the Equilibrium:** $\lim_{t_k \to \infty} \mathbb{E}_{x_{t_k} \sim \upsilon}\left[c_\pi(x_{t_k})\right] = 0$ where $\upsilon(x_{t_k} | \rho, \pi, t_k)$ is the closed-loop state distribution at timestep t_k.*

In essence, the controller is employed to guide the system to the desired state x_{desired} while avoiding unsafe states. However, in many cases, an exact expression of the real dynamics (2) is not available. Therefore, we utilize NODEs to approximate the real dynamics and use $\{\hat{x}_{t_{k+1}}, \hat{x}_{t_{k+2}}, \ldots, \hat{x}_{t_{k+r}}\}$ following (12) to construct constraints, facilitating the learning of the controller. The detailed description of the proposed framework is provided in Section 3.

Algorithm 1: NODEs-based Dynamics Learning

Input : Sequence length h, learning rate η_1

Gather dataset of random trajectories $X = \{x_{t_k}, x_{t_{k+1}}, \ldots, x_{t_{k+h}}\}$ and associated control variable sequences $U = \{u_{t_k}, u_{t_{k+1}}, \ldots, u_{t_{k+h-1}}\}$; Initialize approximated trajectories $\hat{X}$ = List() with $\hat{x}_{t_k} = x_{t_k}$; Initialize dynamics model $\mathcal{F}_\psi (i.e. \hat{f}_{\psi_1}, \hat{g}_{\psi_2})$

for $i = 1$ **to** h **do**

  $\hat{x}_{t_{k+i}} = \hat{x}_{t_{k+i-1}} + \int_{t_{k+i-1}}^{t_{k+i}} \left(\hat{f}_{\psi_1}(\chi, \hat{x}_\chi) + \hat{g}_{\psi_2}(\chi, \hat{x}_\chi) u_{t_{k+i-1}} \right) d\chi$

  Append $\hat{x}_{t_{k+i}}$ to $\hat{X}$

end

$\psi_1 \leftarrow \psi_1 - \eta_1 \nabla_{\psi_1} \ell$ `// Update` $\hat{f}$ `parameters`

$\psi_2 \leftarrow \psi_2 - \eta_1 \nabla_{\psi_2} \ell$ `// Update` $\hat{g}$ `parameters`

3. Framework Design

In this section, we start by outlining the process of approximating the system dynamics using NODEs. Following this, we proceed to define the cost's value function, accompanied by the establishment of a stability condition. Subsequently, we use the augmented Lagrangian method for controller parameter updates. This endeavor is directed towards satisfying safety and stability conditions, and thus contributes to guaranteeing both properties for the system. Lastly, considering the potential infeasibility of the constrained optimization problem, we introduce a backup controller that serves as a substitute for the primary controller in situations where no feasible solution exists to simultaneously satisfy both safety and stability constraints. Additionally, the pseudocode outlining the complete NODEs-based Lyapunov-Barrier Actor-Critic (NLBAC) framework is presented.

3.1. Learning the Dynamics via NODEs

A better initialization for approximated system dynamics is beneficial to the overall training. Hence, before the controller learning phase, we leverage a limited offline dataset to pre-train the approximation of the system's dynamics through NODEs.

Initially, we sample control signal sequences and their corresponding state sequences from real dynamics. We sample rollouts $X = \{x_{t_k}, x_{t_{k+1}}, \ldots, x_{t_{k+h}}\}$ from the initial state x_{t_k} based on $U = \{u_{t_k}, u_{t_{k+1}}, \ldots, u_{t_{k+h-1}}\}$ which is sampled randomly from the action space. Here, h represents the horizon length of the collected sequence. The model then computes the approximated state sequence $\hat{X}$ based on (12). Model loss is designed as (13) and the training process is illustrated as Algorithm 1.

$$\ell = \frac{1}{h} \sum_{i=1}^{h} |x_{t_{k+i}} - \hat{x}_{t_{k+i}}| \tag{13}$$

After this pre-training stage preceding the controller learning phase, the NODEs model can still be updated by using the data collected during the RL-based controller learning process. This update is incorporated into Algorithm 2.

3.2. Value Function of the Cost

To help maintain stability for the system, drawing inspiration from commonly-used value functions in RL, we define the value function of the cost at the state x_{t_k} as

$$L_\pi(x_{t_k}) = \mathbb{E}_{\tau\sim\pi}\big[\sum_{i=0}^{\infty}\gamma_c^i c_\pi(x_{t_{k+i}})\big]. \tag{14}$$

Here $\tau = \{x_{t_k}, x_{t_{k+1}}, x_{t_{k+2}}, \dots\}$ is a trajectory under controller π starting from the initial state x_{t_k}. Based on this definition, $L_\pi(x_{t_k})$ can also be approximated by a neural network.
According to (14), a natural strategy to achieve stability is to introduce a condition that drives the value of $L_\pi(x_{t_k})$ to decrease along the trajectory τ. Inspired by the concept of exponentially stabilizing CLF and insights from (Han et al. 2020), we first make two assumptions for the MDP:

Assumption 1 *The state and control signals are sampled from compact sets, and the reward and cost signals received at any timestep are bounded by r_{max} and c_{max}, respectively.*

Assumption 2 (Ergodicity) *The Markov chain induced by controller π is ergodic with a unique stationary distribution $q_\pi(x) = \lim_{t_k\to\infty}\upsilon(x_{t_k} = x|\rho,\pi,t_k)$, where $\upsilon(x_{t_k}|\rho,\pi,t_k)$ is the closed-loop state distribution.*

Subsequently, we introduce Lemma 5 proposed in (Zhao et al. 2023a) invoked by (Han et al. 2020):

Lemma 5 *Under Assumptions 1, 2, the system (2) is stable in mean cost if there exist positive constants α_1, α_2, β, and a controller π, such that*

$$\alpha_1 c_\pi(x_{t_k}) \le L_\pi(x_{t_k}) \le \alpha_2 c_\pi(x_{t_k}) \tag{15}$$

$$\mathbb{E}_{x_{t_k}\sim\mu_\pi, x_{t_{k+1}}\sim P_\pi}\big[L_\pi(x_{t_{k+1}}) - L_\pi(x_{t_k})\big] \le -\beta\mathbb{E}_{x_{t_k}\sim\mu_\pi}\big[L_\pi(x_{t_k})\big]. \tag{16}$$

Here L_π defined in (14) is the value function of the cost under the controller π, and

$$\mu_\pi(x) \triangleq \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N}\upsilon(x_{t_k} = x|\rho,\pi,t_k) \tag{17}$$

is the sampling distribution, where $\upsilon(x_{t_k}|\rho,\pi,t_k)$ is the closed-loop state distribution at timestep t_k.

The proof provided in (Wang et al. 2023a) establishes that L_π naturally satisfies the constraints (15) under Assumption 1. In the subsequent subsection, we introduce a method to obtain a controller leading to an L_π that satisfies (16).

3.3. Augmented Lagrangian Method for Parameter Updating

In the preceding sections, we established conditions (8) and (16) that a controller should satisfy in order to help maintain safety and stability. Our focus now shifts to the task of updating an RL-based primary controller to meet these conditions. This task can be framed as a constrained optimization problem, where the above conditions serve as constraints. In previous research, the primal-dual

method was used to update the parameters of the RL-based controller; however, this method requires sophisticated hyperparameter tuning, which could be challenging and time-consuming, to adjust the learning rates used in updating parameters of different networks and Lagrangian multipliers (Zhao et al. 2023a). Here we use the augmented Lagrangian method to update the parameters of the RL-based primary controller inspired by its efficacy in solving constrained optimization problems and (Li et al. 2021).

According to the previous sections, by calculating expected values, the CBF and CLF constraints for safety and stability can be given as follows:

$$\begin{aligned}&\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]\leq 0 \qquad \forall i\in[1,m],\\&\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[L_{\pi_p}(\hat{x}_{t_{k+1}})-L_{\pi_p}(x_{t_k})+\beta L_{\pi_p}(x_{t_k})\big]\leq 0\end{aligned} \tag{18}$$

where $\hat{x}_{t_{k+1}}$ is the predicted next state following the primary controller π_p according to (12), and $\{\hat{x}_{t_{k+1}},\hat{x}_{t_{k+2}},\ldots,\hat{x}_{t_{k+r}}\}$ is also calculated according to (12) with the relative degree r as the planning horizon for constructing CBF constraints. This alleviates the problem of large error propagation since the value of the relative degree is usually not large. Here $\mu_{\pi_p}(x)$ is the sampling distribution under the controller π_p.

The actor-critic method is now employed for training the RL-based controller. We denote the parameters of the RL-based primary controller and two action-value networks Q^{π_p} by θ_p and ϕ_i, where $i=1,2$, respectively. Additionally, we employ L_ν, referred to as the Lyapunov network, to approximate L_{π_p} with parameters ν. This results in the following constrained optimization problem:

$$\begin{aligned}&\min_{\theta_p}\ -V^{\theta_p}\\&s.t.\ \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]\leq 0 \qquad \forall i\in[1,m],\\&\qquad\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}})-L_\nu(x_{t_k})+\beta L_\nu(x_{t_k})\big]\leq 0.\end{aligned} \tag{19}$$

The objective function $-V^{\theta_p}$ is:

$$-V^{\theta_p}=-\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},\xi\sim\mathcal{N}}\big[\min_{j=1,2}Q^{\pi_p}_{\phi_j}(x_{t_k},\tilde{u}_{\theta_p}(x_{t_k},\xi))-\alpha_p\log\pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_k},\xi)|x_{t_k})\big]. \tag{20}$$

Also, $\tilde{u}_{\theta_p}(x_{t_k},\xi)=\tanh(\mu_{\theta_p}(x_{t_k})+\sigma_{\theta_p}(x_{t_k})\odot\xi),\xi\sim\mathcal{N}(0,I)$, where μ_{θ_p} and σ_{θ_p} represent the mean and standard deviation of the primary controller π_p modeled as a Gaussian distribution, and the $\odot$ denotes element-wise multiplication. Loss functions for updating the action-value networks $Q^{\pi_p}_{\phi_i},i=1,2$, coefficient α_p, and Lyapunov network L_ν are:

$$\begin{aligned}J_{Q^{\pi_p}}(Q^{\pi_p}_{\phi_i})=&\mathbb{E}_{(x_{t_k},u_{t_k},r_{t_k},x_{t_{k+1}})\sim\mathcal{D},\xi\sim\mathcal{N}}\Big[\big[r_{t_k}+\gamma\big(\min_{j=1,2}Q^{\pi_p}_{targ,\phi_j}(x_{t_{k+1}},\\&\tilde{u}_{\theta_p}(x_{t_{k+1}},\xi))-\alpha_p\log\pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_{k+1}},\xi)|x_{t_{k+1}})\big)-Q^{\pi_p}_{\phi_i}(x_{t_k},u_{t_k})\big]^2\Big],\end{aligned} \tag{21}$$

$$J_{\alpha_p}(\alpha_p)=-\alpha_p\times\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},\xi\sim\mathcal{N}}\big[\log\pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_k},\xi)|x_{t_k})+\mathcal{H}\big], \tag{22}$$

$$J_L(L_\nu)=\mathbb{E}_{(x_{t_k},c_{t_k},x_{t_{k+1}})\sim\mathcal{D}}\Big[\big[c_{t_k}+\gamma_c L_{targ,\nu}(x_{t_{k+1}})-L_\nu(x_{t_k})\big]^2\Big], \tag{23}$$

where $Q_{targ,\phi_i},i=1,2$ are the target action-value networks, $\mathcal{H}$ is a predefined threshold set as the lower bound of the controller's entropy, and $\mathcal{D}$ denotes the batch of transitions following π_p. By

introducing a vector of additional variables $\boldsymbol{z_p} = (z_{1,p}^2, z_{2,p}^2, \cdots, z_{m,p}^2, z_{m+1,p}^2)$, we can convert the problem (19) into an equality constrained problem problem given by:

$$\begin{aligned} \min_{\theta_p, \boldsymbol{z_p}} \ & -V^{\theta_p} \\ s.t. \ & \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big] + z_{i,p}^2 = 0 \quad \forall i \in [1,m], \\ & \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\big] + z_{m+1,p}^2 = 0 \end{aligned} \tag{24}$$

This problem can be solved using the augmented Lagrangian method. Denote the Lagrangian multipliers for CBF and CLF constraints used for this primary controller as $\lambda_{i,p}$ and ζ, respectively, and the penalty parameter as c_p whose value increases with the number of its updates increasing, according to (Bertsekas 1982), the values of the additional variables are explicitly given as:

$$\begin{aligned} z_{i,p}^2 &= \max\{0, -\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big] - \tfrac{\lambda_{i,p}}{c_p}\} \qquad \forall i \in [1,m] \\ z_{m+1,p}^2 &= \max\{0, -\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\big] - \tfrac{\zeta}{c_p}\} \end{aligned} \tag{25}$$

For simplicity, we define $f_{i,p}(\theta_p) \triangleq \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[-\Phi_{i,r}(x_{t_k})\big]$, $\forall i \in [1,m]$, and $g(\theta_p) \triangleq \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p}, u_{t_k}\sim\pi_p}\big[L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\big]$. Then, the corresponding augmented Lagrangian function is given by:

$$\begin{aligned} \mathcal{L}_{c_p}^p(\theta_p, \lambda_{i,p}, \zeta) = & -V^{\theta_p} + \sum_{i=1}^m \lambda_{i,p} \times (f_{i,p}(\theta_p) + z_{i,p}^2) + \sum_{i=1}^m \frac{c_p}{2} \times (f_{i,p}(\theta_p) + z_{i,p}^2)^2 \\ & + \zeta \times (g(\theta_p) + z_{m+1,p}^2) + \frac{c_p}{2} \times (g(\theta_p) + z_{m+1,p}^2)^2. \end{aligned} \tag{26}$$

The complete procedure for updating the parameters is included as a part of Algorithm 2. Additionally, we introduce some tricks used in practical implementations. Firstly, a different timescale method is applied similar to (Yu et al. 2022; Chow et al. 2017), and furthermore, we set n_m, n_L, and n_b to update the NODEs model, Lagrangian multipliers and the backup controller with delayed update tricks (Ma et al. 2021a). Also, in practical implementation, we sample transition pairs from the replay buffer $\mathcal{B}$ for updating parameters, and before calculating the expected values in (18), we first apply the ReLU function to CBF and CLF constraints at each sampled x_{t_k}. This adjustment aligns with our objective that the constraints $-\Phi_{i,r}(x_{t_k}) \leq 0$ and $L_{\pi_p}(\hat{x}_{t_{k+1}}) - L_{\pi_p}(x_{t_k}) + \beta L_{\pi_p}(x_{t_k}) \leq 0$ should be satisfied at each x_{t_k} where the constraints are currently not met, and therefore these constraints are state-wise for each step. Additionally, with the ReLU function, the variables $z_{i,p}^2 = 0, \forall i \in [1, m+1]$ according to (25), obviating the need for their computation during the training phase and thereby reducing the computational burden.

3.4. Backup Controller Design

Because of the existence of multiple constraints, the feasibility of the constrained optimization problem (19) becomes a crucial problem during the learning process. The invalid control signals generated by the RL-based controller due to the infeasibility could impede the system's swift approaching towards its desired state, or lead to severe violations of safety constraints. Priority is given to safety constraints when safety and stability constraints cannot be satisfied simultaneously,

Algorithm 2: Neural Ordinary Differential Equations-based Lyapunov-Barrier Actor-Critic

Input : Number of steps N, pre-trained NODEs model parameterized by φ_1 and φ_2, action-value networks $Q_{\phi_i}^{\pi_p}, i = 1, 2$, Lyapunov network L_ν, RL-based primary controller network π_{θ_p}, coefficient α_p and Lagrange multipliers $\lambda_{i,p}$ and ζ for the primary controller, RL-based backup controller network π_{θ_b}, coefficient α_b and Lagrange multipliers $\lambda_{i,b}$ for the backup controller, replay buffer $\mathcal{B}$, coefficients of quadratic terms c_p and c_b, quadratic term coefficient factor $\rho_c \in (1, \infty)$, learning rates η_1, η_2, η_3, n_m, n_L and n_b which are delay steps for NODEs model, Lagrangian multipliers and backup controller updates, respectively.

$N \leftarrow 0$
for *each episode* **do**
  for *each step* **do**
    $N \leftarrow N + 1$
    if $N \bmod n_m = 0$ **then** `// Update the NODEs model`
      Calculate (13) with data collected during the RL-based controller learning process
      $\psi_1 \leftarrow \psi_1 - \eta_1 \nabla_{\psi_1} \ell$; $\psi_2 \leftarrow \psi_2 - \eta_1 \nabla_{\psi_2} \ell$
    end
    Construct CBF and CLF constraints with the NODEs model and transition pairs from $\mathcal{B}$
    Update the Lyapunov network and action-value networks by using (23) and (21)
    $\nu \leftarrow \nu - \eta_2 \nabla_\nu J_L(L_\nu)$; $\phi_i \leftarrow \phi_i - \eta_2 \nabla_{\phi_i} J_{Q^{\pi_p}}(Q_{\phi_i}^{\pi_p})$ for $i \in \{1, 2\}$
    Update the primary controller network and its coefficient α_p by using (26) and (22)
    $\theta_p \leftarrow \theta_p - \eta_3 \nabla_{\theta_p} \mathcal{L}_{c_p}^p(\theta_p, \lambda_{i,p}, \zeta)$; $\alpha_p \leftarrow \alpha_p - \eta_3 \nabla_{\alpha_p} J_{\alpha_p}(\alpha_p)$
    $c_p \leftarrow \rho_c \times c_p$ `// Update` c_p
    if $N \bmod n_L = 0$ **then**
      Update the Lagrangian multipliers $\lambda_{i,p}$ and ζ according to (Bertsekas 1982)
      $\lambda_{i,p} \leftarrow \max\{0, \lambda_{i,p} + c_p f_{i,p}(\theta_p)\}$; $\zeta \leftarrow \max\{0, \zeta + c_p g(\theta_p)\}$
    end
    if $N \bmod n_b = 0$ **then**
      Update the backup controller network and its coefficient α_b by using (30) and (29)
      $\theta_b \leftarrow \theta_b - \eta_3 \nabla_{\theta_b} \mathcal{L}_{c_b}^b(\theta_b, \lambda_{i,b})$; $\alpha_b \leftarrow \alpha_b - \eta_3 \nabla_{\alpha_b} J_{\alpha_b}(\alpha_b)$
      $c_b \leftarrow \rho_c \times c_b$ `// Update` c_b
      if $N \bmod (n_b \times n_L) = 0$ **then**
        Update the Lagrangian multipliers $\lambda_{i,b}$
        $\lambda_{i,b} \leftarrow \max\{0, \lambda_{i,b} + c_b f_{i,b}(\theta_b)\}$
      end
    end
    if *Backup controller should be used according to the condition specific to the task* **then**
      $u_{t_k} \sim \pi_{\theta_b}(u_{t_k}|x_{t_k})$ and apply control signal u_{t_k}
    else
      $u_{t_k} \sim \pi_{\theta_p}(u_{t_k}|x_{t_k})$ and apply control signal u_{t_k}
      Store the transition pair $(x_{t_k}, u_{t_k}, r_{t_k}, c_{t_k}, x_{t_{k+1}})$ in $\mathcal{B}$
    end
  end
end

Output: π_{θ_p}, π_{θ_b}, $Q_{\phi_i}^{\pi_p}, i = 1, 2$, and L_ν

therefore, for a specific primary controller π_p, we design an RL-based backup controller π_b parameterized by θ_b by formulating an additional constrained optimization problem that only leverages CBFs as constraints, as follows:

$$\begin{aligned} &\min_{\theta_b} \ -V^{\theta_b} \\ &s.t. \ \mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_b}\big[-\Phi_{i,r}(x_{t_k})\big]\leq 0 \qquad \forall i\in[1,m]. \end{aligned} \tag{27}$$

and $\{\hat{x}_{t_{k+1}},\hat{x}_{t_{k+2}},\dots,\hat{x}_{t_{k+r}}\}$ is now the predicted future states following the backup controller π_b instead of π_p. The CBF constraints used here are to help satisfy the safety constraints for the system after following the primary controller π_p. The objective function is designed to maximize the cumulative discounted reward if we use the backup controller π_b for one step and then switch back to the primary controller π_p, and therefore is given as:

$$-V^{\theta_b} = -\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},\xi\sim\mathcal{N}}\big[\min_{j=1,2} Q^{\pi_p}_{\phi_j}(x_{t_k},\tilde{u}_{\theta_b}(x_{t_k},\xi)) - \alpha_b\log\pi_{\theta_b}(\tilde{u}_{\theta_b}(x_{t_k},\xi)|x_{t_k})\big], \tag{28}$$

and the loss function of the coefficient α_b is

$$J_{\alpha_b}(\alpha_b) = -\alpha_b\times\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},\xi\sim\mathcal{N}}\big[\log\pi_{\theta_b}(\tilde{u}_{\theta_b}(x_{t_k},\xi)|x_{t_k})+\mathcal{H}\big]. \tag{29}$$

For simplicity, we define $f_{i,b}(\theta_b)\triangleq\mathbb{E}_{x_{t_k}\sim\mu_{\pi_p},u_{t_k}\sim\pi_b}\big[-\Phi_{i,r}(x_{t_k})\big]$, where $\{\hat{x}_{t_{k+1}},\hat{x}_{t_{k+2}},\dots,\hat{x}_{t_{k+r}}\}$ follows π_b, $\forall i\in[1,m]$. Therefore, the augmented Lagrangian function for updating the backup controller is given as:

$$\mathcal{L}^b_{c_b}(\theta_b,\lambda_{i,b}) = -V^{\theta_b} + \sum_{i=1}^{m}\lambda_{i,b}\times(f_{i,b}(\theta_b)+z^2_{i,b}) + \sum_{i=1}^{m}\frac{c_b}{2}\times(f_{i,b}(\theta_b)+z^2_{i,b})^2, \tag{30}$$

where $\lambda_{i,b}$, c_b and $\boldsymbol{z_b}=(z^2_{1,b},z^2_{2,b},\cdots,z^2_{m,b})$ are Lagrangian multipliers for CBF constraints, the penalty parameter, and additional variables for the update of the backup controller, and the values of $\boldsymbol{z_b}$ are calculated similarly to (25). Also, in real implementation, we sample the data from the replay buffer $\mathcal{B}$, and apply the ReLU function to CBF constraints at each sampled x_{t_k}.
In summary, the framework combining the primary and backup controllers can be summarized as Algorithm 2.

4. Simulations

We conduct experiments on two tasks in this section to test:

- Whether the NLBAC framework helps guarantee the stability for the system? As these tasks reward the system when it approaches and arrives at the desired state (equilibrium), we employ cumulative reward as a metric, and a higher cumulative reward indicates better convergence to the equilibrium.
- Whether the NLBAC framework results in fewer safety constraint violations compared to baseline algorithms?
- Whether the NLBAC framework has a better sample efficiency, namely achieves a good result steadily with fewer interactions with the environment and less collected data compared to baseline algorithms?

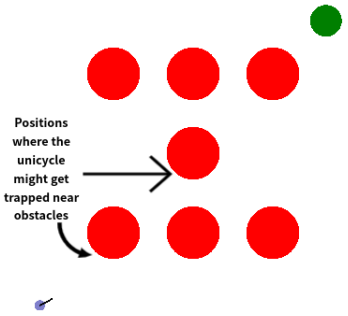

Figure 1: The unicycle environment (Zhao et al. 2023a). The blue point represents the unicycle, the green circle indicates the destination (desired state), and the red circles are obstacles to avoid.

We use MBPPO-Lagrangian (Jayant and Bhatnagar 2022), LAC (Han et al. 2020), CPO (Achiam et al. 2017), PPO-Lagrangian and TRPO-Lagrangian (Ray et al. 2019) as baselines. These baselines encompass diverse approaches, such as the model-based method, Lyapunov-based method, trust region policy optimization, and primal-dual method, respectively. They represent significant contributions in the field of constrained reinforcement learning.

4.1. Unicycle

Currently, there are many studies applying machine learning algorithms to solve problems in the field of transportation (Ma 2022; Ma et al. 2023; Miao et al. 2023; Su et al. 2023), and the environment called Unicycle which is modified from (Emam et al. 2021; Zhao et al. 2023a) is used here to test whether our framework can help achieve a better performance in a transportation scenario. In this experiment setup, a unicycle is tasked with reaching the designated location, i.e., the destination, while avoiding collisions with obstacles. The real dynamics of the system is:

$$\dot{x}_{t_k} = \begin{bmatrix} \cos(\theta_{t_k}) & 0 \\ \sin(\theta_{t_k}) & 0 \\ 0 & 1.0 \end{bmatrix} (u_{t_k} + u_{d,t_k}).$$

Here $x_{t_k} = [x_{1t_k}, x_{2t_k}, \theta_{t_k}]^T$ is the state where x_{1t_k} and x_{2t_k} represent the X-coordinate and Y-coordinate of the unicycle and θ_{t_k} is the orientation at the timestep t_k. The control signal $u_{t_k} = [v_{t_k}, \omega_{t_k}]^T$ comprises linear velocity v_{t_k} and angular velocity ω_{t_k}. The time interval used in this experiment is 0.02s. $u_{d,t_k} = -0.1[\cos(\theta_{t_k}), 0]^T$. As to NODEs-based models, the input of network $\hat{f}$ is the state with the dimension of 3, and the output dimension is 3; the input of network $\hat{g}$ is the state with the dimension of 3, and the output dimension is 6 (which is then be resized as $[3, 2]$). Then, a point at a distance l_p ahead of the unicycle is considered to establish safety constraints for collision-free navigation, and the function $p : \mathbb{R}^3 \to \mathbb{R}^2$ is defined as:

$$p(x_{t_k}) = \begin{bmatrix} x_{1t_k} \\ x_{2t_k} \end{bmatrix} + l_p \begin{bmatrix} \cos(\theta_{t_k}) \\ \sin(\theta_{t_k}) \end{bmatrix}.$$

The reward signal is formulated as $-K_1(v_t - v_s)^2 + K_2\Delta d$, where v_s represents the predefined velocity, Δd denotes the reduction in the distance between the unicycle and the destination in two

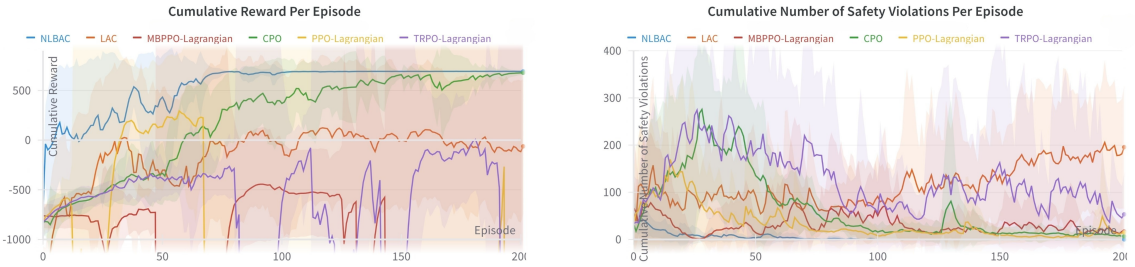

Figure 2: The cumulative reward and cumulative number of safety violations of each episode in the unicycle environment setting are compared between the proposed NLBAC (drawn in blue) and baselines. Each curve illustrates the average across ten experiments employing different random seeds, with the shaded area denoting the standard deviation.

consecutive timesteps, and K_1 and K_2 are coefficients set to 0.1 and 30, respectively. The cost signal is given by $\left\|p(x_{t_{k+1}}) - p(x_{\text{desired}})\right\|$ where $p(x_{\text{desired}}) = [x_{1\text{desired}}, x_{2\text{desired}}]^T$ denotes the position of the desired location. CBFs are defined as $h_i(x_{t_k}) = \frac{1}{2}\left((p(x_{t_k}) - p_{\text{obs}_i})^2 - \delta^2\right)$ where p_{obs_i} represents the position of the i^{th} obstacle, and δ is the minimum required distance between the unicycle and obstacles. Hence, the relative degree is 1 and the planning horizon for NODEs predictions is 1. If the stability constraint is violated due to the inability to satisfy safety and stability constraints simultaneously, the unicycle can become trapped close to obstacles. In such cases, the backup controller takes over from the primary controller. The primary controller is reinstated when the unicycle moves a long distance away from the trapped position, or when the predefined time threshold for using the backup controller is exceeded.

Figure 2 presents the outcomes of our simulations. Our framework demonstrates superior performance, yielding higher cumulative rewards compared to baselines, which suggests that the unicycle, under our framework, can more efficiently approach and reach its destination (equilibrium). Moreover, the fluctuations observed in the cumulative reward are notably smaller than those exhibited by any other baseline algorithm. This suggests that the system's ability to achieve high rewards can quickly recover to its original high level after deterioration. Concerning safety, as indicated by the cumulative number of safety violations per episode, our framework exhibits much fewer violations compared to other baselines. This highlights the substantial role CBFs play in helping maintain safety, rendering our framework suitable for real-world applications with safety-critical considerations. Furthermore, it is evident that our algorithm requires the fewest interactions with the environment to achieve a good performance, which means it has higher sample efficiency.

4.2. Simulated Car Following

This environment, simulating a chain of five cars following each other on a straight road, is modified from (Cheng et al. 2019; Emam et al. 2021). The objective is to control the acceleration of the 4^{th} car to maintain a desired distance from the 3^{rd} car while avoiding collisions with other cars. The real dynamics of all cars except the 4^{th} one is given by:

$$\dot{x}_{t_k,i} = \begin{bmatrix} v_{t_k,i} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 + d_i \end{bmatrix} a_{t_k,i} \qquad \forall i \in \{1, 2, 3, 5\}.$$

Each state of the system is denoted as $x_{t_k,i} = [p_{t_k,i}, v_{t_k,i}]^T$, indicating the position $p_{t_k,i}$ and velocity $v_{t_k,i}$ of the i^{th} car at the timestep t_k, $d_i = 0.1$. The time interval used in this experiment is 0.02s. The predefined velocity of the 1^{st} car is $v_s - 4\sin(t_k)$, where $v_s = 3.0$. Its acceleration is given as $a_{t_k,1} = k_v(v_s - 4\sin(t_k) - v_{t_k,1})$ where $k_v = 4.0$. Accelerations of Car 2 and 3 are given by:

$$a_{t_k,i} = \begin{cases} k_v(v_s - v_{t_k,i}) - k_b(p_{t_k,i-1} - p_{t_k,i}) & if\ |p_{t_k,i-1} - p_{t_k,i}| < 6.5 \\ k_v(v_s - v_{t_k,i}) & otherwise, \end{cases}$$

where $k_b = 20.0$ and $i = 2, 3$. The acceleration of the 5^{th} car is:

$$a_{t_k,5} = \begin{cases} k_v(v_s - v_{t_k,5}) - k_b(p_{t_k,3} - p_{t_k,5}) & if\ |p_{t_k,3} - p_{t_k,5}| < 13.0 \\ k_v(v_s - v_{t_k,5}) & otherwise. \end{cases}$$

The model of the 4^{th} car is as follows:

$$\dot{x}_{t_k,4} = \begin{bmatrix} v_{t_k,4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1.0 \end{bmatrix} u_{t_k},$$

where u_{t_k} is the acceleration of the 4^{th} car, and also the control signal generated by the controller at the timestep t_k.

The reward signal is defined to minimize the overall control effort, and an additional reward of 2.0 is granted during timesteps when $d_{t_k} = p_{t_k,3} - p_{t_k,4}$, which is the distance between the 3^{rd} and 4^{th} car, falls within $[9.0, 10.0]$. This range is defined as the desired region for d_{t_k}. Thus, the cost signal is determined as $\left\| d_{t_{k+1}} - d_{\text{desired}} \right\|$, where $d_{\text{desired}} = 9.5$. CBFs are defined as $h_1(x_{t_k}) = p_{t_k,3} - p_{t_k,4} - \delta$ and $h_2(x_{t_k}) = p_{t_k,4} - p_{t_k,5} - \delta$, with δ being the minimum required distance between the cars. Hence, the relative degree is 2 and the planning horizon for making predictions using NODEs is 2. When a safety constraint is violated if two types of constraints cannot be satisfied simultaneously, the 4^{th} car might be in close proximity to the 5^{th} car in order to make d_{t_k} be within $[9.0, 10.0]$. In such cases, the backup controller is activated. The primary controller is reactivated when the 4^{th} car moves beyond the dangerous area, namely out of the vicinity of the 5^{th} car, or when the predetermined time threshold for utilizing the backup controller is exceeded.

Noted that when we use NODEs to model this system, we assume that we do not have the priori information that this system is control-affine which means model (11) is used here. The input of network $\mathcal{F}$ is $(t_k, \hat{x}_{t_k}, u_{t_k})$ with the dimension of 12, and the output dimension is 10. As illustrated in Figure 3, the proposed framework consistently achieves the highest cumulative reward. This outcome indicates its exceptional capability in effectively regulating the distance d_{t_k} within the desired range $[9.0, 10.0]$. Moreover, although larger than others for a short period in the beginning, the cumulative number of safety violations caused by the proposed NLBAC framework is considerably smaller than those of others after some episodes and finally decreases to almost 0. This shows the efficacy of the proposed framework in guiding the system toward safety through the incorporation of CBFs. Also, the number of data this proposed framework NLBAC requires to update the parameters of the RL-based controllers to achieve a higher reward and small number of safety violations is the smallest, which means this framework is still the most sample-efficient one in the algorithms tested in this environment.

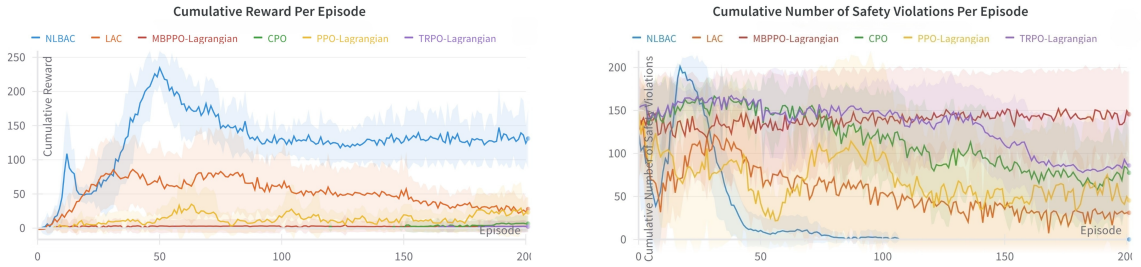

Figure 3: The cumulative reward and cumulative number of safety violations of each episode in the simulated car following environment setting are compared between the proposed NLBAC (drawn in blue) and baselines. Each curve illustrates the average across ten experiments employing different random seeds, with the shaded area denoting the standard deviation.

5. Conclusions

This paper introduces the NLBAC framework, an approach combining CBF and CLF constraints with the actor-critic method to assist in guaranteeing both the safety and stability of the controlled system whose dynamics is approximated by neural ordinary differential equations. This framework, with the application of the ReLU function, imposes safety constraints for each trajectory step, which is pivotal in real-world applications with critical safety requirements. Moreover, our framework helps to guarantee the stability of the system, enabling the system to approach the desired state (equilibrium) more effectively and achieve higher cumulative rewards in tasks where high rewards are provided upon the system's approaching or arrival at the desired state, as seen in navigation scenarios. In summary, the experiments demonstrate higher cumulative rewards and fewer violations of safety constraints with better sample efficiency.

Nonetheless, the framework has certain limitations: 1. Predefined CBFs are required before the controller learning, posing a challenge in real-world applications where the construction of valid CBFs may be a complex task; 2. Although good performance is achieved when the constraints are constructed based on the predictions provided by NODEs-based models, the effect of the difference between the real dynamics and the NODEs-based models is still unknown and worth further investigation. We believe that exploring answers to these existing questions could be interesting future directions.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

Victor M Martinez Alvarez, Rareş Roşca, and Cristian G Fălcuţescu. Dynode: Neural ordinary differential equations for dynamics modeling in continuous control. *arXiv preprint arXiv:2009.04278*, 2020.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference*, pages 3420–3431. IEEE, 2019.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic Press, 1982.

Hongpeng Cao, Yanbing Mao, Lui Sha, and Marco Caccamo. Physical deep reinforcement learning towards safety guarantee. *arXiv preprint arXiv:2303.16860*, 2023a.

Hongpeng Cao, Yanbing Mao, Lui Sha, and Marco Caccamo. Physical deep reinforcement learning: Safety and unknown unknowns. *arXiv preprint arXiv:2305.16614*, 2023b.

Sohom Chakrabarty, Bijnan Bandyopadhyay, Jaime A Moreno, and Leonid Fridman. Discrete sliding mode control for systems with arbitrary relative degree output. In *2016 14th International Workshop on Variable Structure Systems (VSS)*, pages 160–165. IEEE, 2016.

Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809. IEEE, 2021.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Charles Dawson, Bethany Lowenkamp, Dylan Goff, and Chuchu Fan. Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robotics and Automation Letters*, 7(2): 1904–1911, 2022a.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022b.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022.

Allan Andre do Nascimento, Antonis Papachristodoulou, and Kostas Margellos. A game theoretic approach for safe and distributed control of unmanned aerial vehicles. 2023.

Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. *Advances in Neural Information Processing Systems*, 33:19805–19816, 2020.

Yousef Emam, Paul Glotfelter, Zsolt Kira, and Magnus Egerstedt. Safe model-based reinforcement learning using robust control barrier functions. *arXiv preprint arXiv:2110.05415*, 2021.

Tuomas Haarnoja et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020.

Minghao Han, Zhaojian Li, Xiang Yin, and Xunyuan Yin. Robust learning and control of time-delay nonlinear systems with deep recurrent koopman operators. *IEEE Transactions on Industrial Informatics*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sihong He, Yue Wang, Shuo Han, Shaofeng Zou, and Fei Miao. A robust and constrained multi-agent reinforcement learning framework for electric vehicle amod systems. *arXiv preprint arXiv:2209.08230*, 2022.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Ashish K Jayant and Shalabh Bhatnagar. Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm. *Advances in Neural Information Processing Systems*, 35:24432–24445, 2022.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control*, pages 6059–6066. IEEE, 2018.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Jingqi Li, David Fridovich-Keil, Somayeh Sojoudi, and Claire J. Tomlin. Augmented lagrangian method for instantaneously constrained reinforcement learning problems. In *2021 60th IEEE Conference on Decision and Control*, pages 2982–2989, 2021. doi: 10.1109/CDC45484.2021.9683088.

Yuxuan Liang, Kun Ouyang, Hanshu Yan, Yiwei Wang, Zekun Tong, and Roger Zimmermann. Modeling trajectories with neural ordinary differential equations. In *IJCAI*, pages 1498–1504, 2021.

Haitong Ma, Yang Guan, Shegnbo Eben Li, Xiangteng Zhang, Sifa Zheng, and Jianyu Chen. Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety. *arXiv preprint arXiv:2105.10682*, 2021a.

Haitong Ma, Changliu Liu, Shengbo Eben Li, Sifa Zheng, Wenchao Sun, and Jianyu Chen. Learn zero-constraint-violation policy in model-free constrained reinforcement learning. *arXiv preprint arXiv:2111.12953*, 2021b.

Haitong Ma, Changliu Liu, Shengbo Eben Li, Sifa Zheng, and Jianyu Chen. Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning. In *Learning for Dynamics and Control Conference*, pages 97–109. PMLR, 2022.

Xiaobo Ma. *Traffic Performance Evaluation Using Statistical and Machine Learning Methods*. PhD thesis, The University of Arizona, 2022.

Xiaobo Ma, Abolfazl Karimpour, and Yao-Jan Wu. On-ramp and off-ramp traffic flows estimation based on a data-driven transfer learning framework. *arXiv preprint arXiv:2308.03538*, 2023.

Keyan Miao and Konstantinos Gatsis. Towards optimal network depths: Control-inspired acceleration of training and inference in neural ODEs. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023a. URL https://openreview.net/forum?id=wErWEsPY8g.

Keyan Miao and Konstantinos Gatsis. Learning robust state observers using neural odes. In *Learning for Dynamics and Control Conference*, pages 208–219. PMLR, 2023b.

Yang Miao, Iro Armeni, Marc Pollefeys, and Daniel Barath. Volumetric semantically consistent 3d panoptic mapping. *arXiv preprint arXiv:2309.14737*, 2023.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems*, 32, 2019.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

Sanbao Su, Yiming Li, Sihong He, Songyang Han, Chen Feng, Caiwen Ding, and Fei Miao. Uncertainty quantification of collaborative detection for self-driving. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5588–5594. IEEE, 2023.

Daniel CH Tan, Fernando Acero, Robert McCarthy, Dimitrios Kanoulas, and Zhibin Li. Value functions are control barrier functions: Verification of safe policies using control theory. *arXiv preprint arXiv:2306.04026*, 2023.

Xiao Tan and Dimos V Dimarogonas. On the undesired equilibria induced by control barrier function based quadratic programs. *arXiv preprint arXiv:2104.14895*, 2021.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Han Wang, Kostas Margellos, and Antonis Papachristodoulou. Safety verification and controller synthesis for systems with input constraints. *arXiv preprint arXiv:2204.09386*, 2022.

Shengjie Wang et al. A rl-based policy optimization method guided by adaptive stability certification. *arXiv preprint arXiv:2301.00521*, 2023a.

Yanran Wang and David Boyle. Trustworthy reinforcement learning for quadrotor uav tracking control systems. *arXiv preprint arXiv:2302.11694*, 2023.

Yanran Wang, James O'Keeffe, Qiuchen Qian, and David Boyle. Quadue-ccm: Interpretable distributional reinforcement learning using uncertain contraction metrics for precise quadrotor trajectory tracking. In *Conference on Robot Learning*, pages 2306–2316. PMLR, 2023b.

Tianhao Wei and Changliu Liu. Safe control algorithms using energy functions: A unified framework, benchmark, and new directions. In *2019 IEEE 58th Conference on Decision and Control*, pages 238–243. IEEE, 2019.

Yuhan Xiong, Di-Hua Zhai, Mahdi Tavakoli, and Yuanqing Xia. Discrete-time control barrier function: High-order case and adaptive case. *IEEE Transactions on Cybernetics*, 2022.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.

Songyuan Zhang, Yumeng Xiu, Guannan Qu, and Chuchu Fan. Compositional neural certificates for networked dynamical systems. In *Learning for Dynamics and Control Conference*, pages 272–285. PMLR, 2023.

Liqun Zhao, Konstantinos Gatsis, and Antonis Papachristodoulou. A barrier-lyapunov actor-critic reinforcement learning approach for safe and stable control. *arXiv preprint arXiv:2304.04066*, 2023a.

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. State-wise safe reinforcement learning: A survey. *arXiv preprint arXiv:2302.03122*, 2023b.

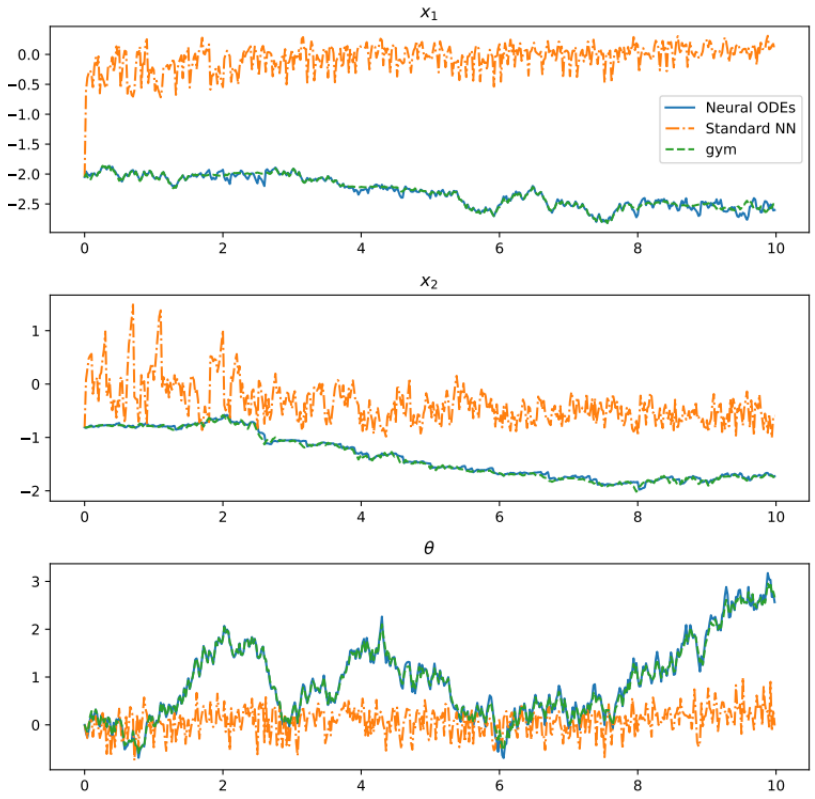

Figure 4: The comparison of modeling performance between NODEs (depicted by the curve labeled as "Neural ODEs") and the neural network baseline (indicated by the curve labeled as "Standard NN") for the Unicycle environment. The ground truth is the output of the gym environment and therefore is labeled as "gym".

6. Appendix

In this appendix, we present comparisons of modeling performance between Neural ODEs, which model system dynamics, and a common neural network baseline that directly outputs the predicted next state.

Unicycle: In this environment, with the prior knowledge that the system is control-affine, we utilize two separate networks to represent $\hat{f}$ and $\hat{g}$, respectively, to model the dynamics through Neural ODEs. For the common neural network baseline, we employ a multilayer perceptron (MLP) with 5 hidden layer, each with a hidden dimension of 100, to predict the next states given the current states and actions. The test result is depicted in Figure 4, where the mean squared error of the NODEs-based model, computed using the PyTorch `nn.MSELoss` function, is 0.0029, and the mean squared error of the common NN-based model computed in the same way is 2.5176.

Simulated Car Following: In this environment, lacking the prior knowledge about whether the system is control-affine or not, we directly use a single network to represent $\mathcal{F}$ proposed in (11) to model the dynamics through Neural ODEs. For the common neural network baseline, we employ an MLP with 4 hidden layers, each with a hidden dimension of 64, to predict the next states. The test result is illustrated in Figure 5, where the mean squared error of the NODEs-based model, computed

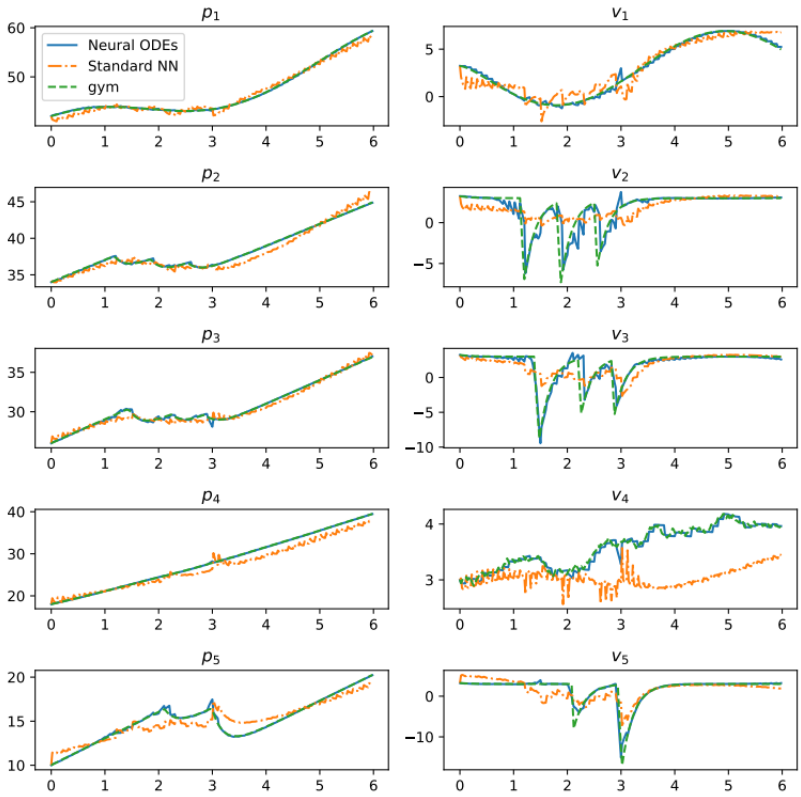

Figure 5: The comparison of modeling performance between NODEs (depicted by the curve labeled as "Neural ODEs") and the neural network baseline (indicated by the curve labeled as "Standard NN") for the Simulated Car Following environment. The ground truth is the output of the gym environment and therefore is labeled as "gym".

using the PyTorch `nn.MSELoss` function, is 0.3682, and the mean squared error of the common NN-based model computed in the same way is 1.5534.