

# Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions

Jason Choi<sup>\*1</sup>, Fernando Castañeda<sup>\*1</sup>, Claire J. Tomlin<sup>2</sup>, Koushil Sreenath<sup>1</sup>

<sup>1</sup>Department of Mechanical Engineering, <sup>2</sup>Department of Electrical Engineering and Computer Sciences, UC Berkeley  
Email: {jason.choi, fcastaneda, tomlin, koushils}@berkeley.edu

**Abstract**—In this paper, the issue of model uncertainty in safety-critical control is addressed with a data-driven approach. For this purpose, we utilize the structure of an input-output linearization controller based on a nominal model along with a Control Barrier Function and Control Lyapunov Function based Quadratic Program (CBF-CLF-QP). Specifically, we propose a novel reinforcement learning framework which learns the model uncertainty present in the CBF and CLF constraints, as well as other control-affine dynamic constraints in the quadratic program. The trained policy is combined with the nominal model-based CBF-CLF-QP, resulting in the *Reinforcement Learning-based CBF-CLF-QP (RL-CBF-CLF-QP)*, which addresses the problem of model uncertainty in the safety constraints. The performance of the proposed method is validated by testing it on an underactuated nonlinear bipedal robot walking on randomly spaced stepping stones with one step preview, obtaining stable and safe walking under model uncertainty.

## I. INTRODUCTION

In this work, we address the issue of model uncertainty in safety-critical control using a data-driven machine learning approach. Our goal is to benefit from the recent successes of learning-based control in highly uncertain dynamical systems, such as in Hwangbo et al. [11] and Levine et al. [12], yet to also account for safety in a formal way. We seek to combine the benefits of these data-driven approaches with the benefits of classical model-based control methods which have theoretical guarantees on stability and safety. Towards this end, we use Control Lyapunov Function- and Control Barrier Function-based controllers designed on nominal systems that are then trained through reinforcement learning (RL) to work on systems with uncertainty.

### A. Related Work

In the field of controls, Control Lyapunov Function (CLF)-based and Control Barrier Function (CBF)-based control methods have been shown to be successful for safety-critical control. Galloway et al. [9] and Ames and Powell [1] have shown that CLF-based quadratic programs (CLF-QP) with constraints

can be solved online in order to perform locomotion and manipulation tasks. In Ames et al. [3], CBFs are incorporated with the CLF-QP, namely CBF-CLF-QP, to handle safety constraints effectively in real time.

These CLF-based and CBF-based methods heavily rely on accurate knowledge of the system model. When the model is uncertain, we must consider adaptive or robust versions. In Nguyen and Sreenath [13], an  $L_1$  adaptive controller is incorporated with the CLF-QP in order to adapt to model uncertainty, and is shown to work effectively for bipedal walking. In Nguyen and Sreenath [15], a robust version of the CBF-CLF-QP is proposed, that solves the quadratic program for the worst case effect of model uncertainty. While these methods can tackle model uncertainty to some degree, they may often fail to account for the correct magnitudes of adaptation and uncertainty.

Recently, several methods addressing the issue of model uncertainty in the control problem using a data-driven approach have been proposed. Westenbroek et al. [22] proposes an RL-based method to learn the model uncertainty compensation for input-output linearization control. In Castañeda et al. [6] the former method is extended to underactuated bipedal walking on a flat terrain. Taylor et al. [19] and Taylor et al. [20] each addresses how to learn the uncertainty in CLF and CBF constraints respectively, using empirical risk minimization. Our methodologies most closely align with these works in that we are also using learning methods to reduce model uncertainty explicitly in input-output linearization, CLF, and CBF-based control. However, the main novelty in our approach is that we have devised a unified RL-based framework for learning model uncertainty in CLF, CBF, and other dynamic control-affine constraints altogether in a single learning process. In addition to the aforementioned papers, there are also a few approaches [4, 5, 8] that learn model uncertainty through probabilistic models such as Gaussian Processes. Although these approaches allow for an insightful analysis of the learned model or policy, they can scale poorly with state dimension.

### B. Contributions

In this paper, we present a novel RL-based framework which combines two key components: 1) an RL agent which learns model uncertainty in multiple general dynamic constraints

\* Indicates equal contribution.

The work of Jason Choi received the support of a fellowship from Kwanjeong Educational Foundation, Korea. The work of Fernando Castañeda received the support of a fellowship (code LCF/BQ/AA17/11610009) from "la Caixa" Foundation (ID 100010434). This work was partially supported through National Science Foundation Grant CMMI-1931853.

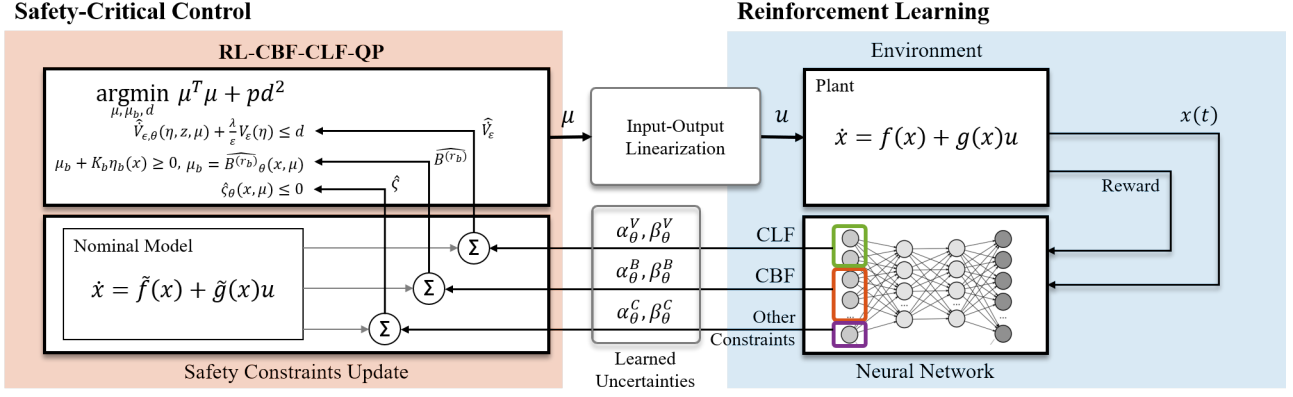


Fig. 1: Our method (RL-CBF-CLF-QP): We propose a control barrier function (CBF) and control Lyapunov function (CLF) based parametrized quadratic program, where the parameter  $\theta$  corresponds to weights of a neural network that estimates the uncertainty in the CLF and CBF dynamics through reinforcement learning. An RL agent is used to learn uncertainties in the CLF, CBF and other constraint dynamics. The quadratic program uses learned uncertainties in combination with safety and stability constraints from a nominal model to solve for the control input point-wise in time.

including CLF and CBF constraints through training, and 2) a quadratic program that solves for the control that satisfies the safety constraints under the learned model uncertainty. We name this framework **Reinforcement Learning-based Control Barrier Function and Control Lyapunov Function Quadratic Program (RL-CBF-CLF-QP)**. After training, the RL-CBF-CLF-QP can be executed online with fast computation. The overall diagram of our framework is presented in Fig. 1. Here is the summary of the contribution of our work:

- 1) We present an RL framework that learns model uncertainty for CLF, CBF and other control-affine dynamic constraints in a single learning process.
- 2) We generalize our method to high relative-degree outputs and Control Barrier Functions.
- 3) Our method can learn the uncertainty in the dynamics of parameterized CBFs that are not only state-dependent but also dependent on other parameters.
- 4) We numerically validate our method on an underactuated nonlinear hybrid system: a bipedal robot walking on stepping stones with significant model uncertainty.

### C. Organization

In Section II, we briefly explain the necessary background for the paper. In Section III, we discuss how we can learn model uncertainty in the CLF constraint for CLF-QP through RL. In Section IV, we expand this method to learn uncertainties in the CBF and general control-affine dynamic constraints, and propose the RL-CBF-CLF-QP. In Section V, we discuss how the RL agent can learn aforementioned uncertainties. In Sections VI and VII, we explain the results of the demonstration of RL-CBF-CLF-QP for a bipedal robot. Finally, we discuss limitations of our method in Section VIII and give concluding remarks in Section IX.

## II. BACKGROUND

### A. Input-Output Linearization

Consider a control affine nonlinear system

$$\begin{aligned}\dot{x} &= f(x) + g(x)u, \\ y &= h(x),\end{aligned}\tag{1}$$

where  $x \in \mathbb{R}^n$  is the system state,  $u \in \mathbb{R}^m$  the control input and  $y \in \mathbb{R}^m$  the output of the system, assuming there are the same number of input and output variables. We also make the standard assumption that  $f$  and  $g$  are Lipschitz continuous. Then, if the vector relative degree of the outputs is  $r$ , we have

$$y^{(r)} = L_f^r h(x) + L_g L_f^{r-1} h(x)u,\tag{2}$$

where the functions  $L_f^r h$  and  $L_g L_f^{r-1} h$  are known as  $r^{th}$  order Lie derivatives [17]. Here,  $y^{(r)}$  is the vector of  $r^{th}$  derivatives of each output in  $y$ , and (2) indicates that no input in  $u$  appears at lower than the  $r^{th}$  derivative of each output. If the  $m \times m$  matrix  $L_g L_f^{r-1} h(x)$  is nonsingular  $\forall x \in D$ , with  $D \subset \mathbb{R}^n$  being a compact subset containing the origin, then we can apply a control input which renders the input-output dynamics of the system linear:

$$u(x, \mu) = u^*(x) + \left( L_g L_f^{r-1} h(x) \right)^{-1} \mu,\tag{3}$$

where  $u^*$  is the feedforward term:

$$u^*(x) = - \left( L_g L_f^{r-1} h(x) \right)^{-1} L_f^r h(x),\tag{4}$$

and  $\mu \in \mathbb{R}^m$  is the *auxiliary input*.

Using this control law yields the input-output linearized system  $y^{(r)} = \mu$ , and we can define a state transformation  $\Phi : x \rightarrow (\eta, z)$ , with

$$\eta = [h(x)^\top, L_f h(x)^\top, \dots, L_f^{r-1} h(x)^\top]^\top\tag{5}$$

and  $z \in Z$ , where  $Z = \{x \in \mathbb{R}^n \mid \eta \equiv 0\}$  is the zero-dynamics manifold. The closed-loop dynamics of the system can then be

represented as a linear time-invariant system on the transverse coordinates  $\eta$ , and the zero-dynamics:

$$\begin{cases} \dot{\eta} = F\eta + G\mu, \\ \dot{z} = p(\eta, z), \end{cases} \quad (6)$$

where

$$F = \begin{bmatrix} 0 & I_m & \cdot & \cdot & 0 \\ 0 & 0 & I_m & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & I_m \\ 0 & \cdot & \cdot & \cdot & 0 \end{bmatrix} \text{ and } G = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \\ I_m \end{bmatrix}, \quad (7)$$

with  $F \in \mathbb{R}^{mr \times mr}$  and  $G \in \mathbb{R}^{mr \times m}$ .

### B. Control Lyapunov Function Based Quadratic Programs

In Ames et al. [2] a control method that guarantees exponential stability of the transverse dynamics  $\eta$  with a rapid enough convergence rate is presented. It introduces the concept of a *rapidly exponentially stabilizing control Lyapunov function (RES-CLF)*. Specifically, a one-parameter family of continuously differentiable functions  $V_\varepsilon : \mathbb{R}^{mr} \rightarrow \mathbb{R}$  is said to be an RES-CLF for system (1) if  $\exists \gamma, c_1, c_2 > 0$  such that  $\forall 0 < \varepsilon < 1$  and  $\forall \eta \in \mathbb{R}^{mr}$ , the following holds:

$$c_1 \|\eta\|^2 \leq V_\varepsilon(\eta) \leq \frac{c_2}{\varepsilon^2} \|\eta\|^2, \quad (8)$$

$$\dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq 0. \quad (9)$$

If we define a control input  $\mu$  that makes  $\eta$  exponentially stable, of the form

$$\mu = \left[ -\frac{1}{\varepsilon^r} K_r, \dots, -\frac{1}{\varepsilon^2} K_2, -\frac{1}{\varepsilon} K_1 \right] \eta = K\eta, \quad (10)$$

where  $K \in \mathbb{R}^{m \times mr}$ , then we can choose a quadratic CLF candidate  $V_\varepsilon(\eta) = \eta^T P_\varepsilon \eta$ , where  $P_\varepsilon$  is the solution of the Lyapunov equation  $A^T P_\varepsilon + P_\varepsilon A = -Q$ , with  $A$  being the closed-loop dynamics matrix  $A = F + GK$  and  $Q$  any symmetric positive-definite matrix. Defining  $\bar{f} = F\eta$ ,  $\bar{g} = G$ , we can write the derivative of the RES-CLF as:

$$\dot{V}_\varepsilon(\eta, \mu) = L_{\bar{f}} V_\varepsilon(\eta) + L_{\bar{g}} V_\varepsilon(\eta) \mu, \quad (11)$$

with

$$L_{\bar{f}} V_\varepsilon(\eta) = \eta^T (F^T P_\varepsilon + P_\varepsilon F) \eta, \quad L_{\bar{g}} V_\varepsilon(\eta) = 2\eta^T P_\varepsilon G. \quad (12)$$

We can then define for every time step an optimization problem in which condition (9) becomes a linear constraint on the auxiliary input  $\mu$ . The objective function can be set to minimize the norm of the control inputs, in which case the optimization problem is a quadratic program (QP):

---

#### CLF-QP:

$$\mu^*(x) = \underset{\mu}{\operatorname{argmin}} \mu^T \mu \quad (13)$$

$$\text{s.t. } \dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq 0 \quad (\text{CLF})$$

### C. Control Barrier Function and Control Lyapunov Function Based Quadratic Programs

In Nguyen and Sreenath [14] the concept of an Exponential Control Barrier Function (ECBF) is defined. Specifically, a function  $B : \mathbb{R}^m \rightarrow \mathbb{R}$  is an ECBF of relative degree  $r_b$  for the system (1) if there exists  $K_b \in \mathbb{R}^{1 \times r_b}$  such that

$$\sup_u \left[ L_f^{r_b} B(x) + L_g L_f^{r_b-1} B(x) u + K_b \eta_b(x) \right] \geq 0 \quad (14)$$

for  $\forall x \in \{x \in \mathbb{R}^n \mid B(x) \geq 0\}$  with

$$\eta_b(x) = \begin{bmatrix} B(x) \\ \dot{B}(x) \\ \ddot{B}(x) \\ \vdots \\ B^{(r_b-1)}(x) \end{bmatrix} = \begin{bmatrix} B(x) \\ L_f B(x) \\ L_f^2 B(x) \\ \vdots \\ L_f^{r_b-1} B(x) \end{bmatrix}, \quad (15)$$

that guarantees  $B(x_0) \geq 0 \implies B(x(t)) \geq 0, \forall t \geq 0$ .

We can then choose a *virtual input*  $\mu_b$  that input-output linearizes the ECBF dynamics:

$$B^{(r_b)}(x, \mu) = L_f^{r_b} B(x) + L_g L_f^{r_b-1} B(x) u(x, \mu) =: \mu_b, \quad (16)$$

with  $u$  defined in (3). We refer readers to Nguyen and Sreenath [14] for more details. The condition in (14) then translates to choosing a  $\mu_b$  such that

$$\mu_b + K_b \eta_b \geq 0, \quad (17)$$

which is added to the following QP, where safety is prioritized over stability by relaxing the CLF constraint:

---

#### CBF-CLF-QP:

$$\mu^*(x) = \underset{\mu, \mu_b, d}{\operatorname{argmin}} \mu^T \mu + p d^2 \quad (18)$$

$$\text{s.t. } \dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq d \quad (\text{CLF})$$

$$\mu_b + K_b \eta_b \geq 0 \quad (\text{CBF})$$

$$\begin{aligned} \mu_b &= B^{(r_b)}(x, \mu) \\ A_c(x) \mu + b_c(x) &\leq 0 \quad (\text{Constraints}) \end{aligned}$$

Formulating a QP allows us to incorporate additional control-affine constraints (last line in (18)). These could be input saturation constraints or other state-dependent constraints such as contact-force constraints.

### III. REINFORCEMENT LEARNING FOR CLF-QP BASED CONTROLLERS UNDER UNCERTAIN DYNAMICS

In this section, we address the issue of having a mismatch between the model and the plant dynamics when the true plant vector fields  $f, g$  are not precisely known. Specifically, between this and the next sections we analytically examine the effects of model uncertainty on the dynamics of the CLF, CBF and other control-affine dynamic constraints. For each of these cases we will define the goal of the RL agent and the policy to be learned.

### A. Reinforcement Learning for CLF-QP Based Controllers: First Approach

Let the *nominal model* used in the controller be

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u. \quad (19)$$

We assume: 1) the vector fields  $\tilde{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n, \tilde{g} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$  are Lipschitz continuous and 2) the vector relative degrees of the model and plant dynamics are the same ( $r$ ). These are the standard assumptions that have been made in most of the literature [15, 19, 20, 22] to tackle the mismatch terms analytically.

The pre-control law (3) of input-output linearization computed based on the nominal model  $\tilde{f}, \tilde{g}$  has the following form

$$\tilde{u}(x, \mu) = \tilde{u}^*(x) + \left( L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x) \right)^{-1} \mu, \quad (20)$$

with a feedforward term

$$\tilde{u}^*(x) := - \left( L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x) \right)^{-1} L_{\tilde{f}}^r h(x). \quad (21)$$

Using this  $\tilde{u}$  in (2) yields

$$y^{(r)} = \mu + \Delta_1(x) + \Delta_2(x)\mu, \quad (22)$$

where

$$\begin{aligned} \Delta_1(x) &:= L_f^r h(x) - L_g L_f^{r-1} h(x) \left( L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x) \right)^{-1} L_{\tilde{f}}^r h(x), \\ \Delta_2(x) &:= L_g L_f^{r-1} h(x) \left( L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x) \right)^{-1} - I_m. \end{aligned} \quad (23)$$

The dynamics of  $\eta$  from (6) now take the form:

$$\dot{\eta} = (F\eta + G\Delta_1(\eta, z)) + G(I_m + \Delta_2(\eta, z))\mu. \quad (24)$$

Note that this equation is the same as (6) if the uncertainty terms are zero, i.e.  $\Delta_1 = \Delta_2 = 0$ . Thus, (6) can be considered a nominal model for the true transverse dynamics (24).

For this first approach we use RL to define an additional input whose goal is to cancel out the uncertainty terms present in the transverse dynamics (24), and therefore manipulate the transverse dynamics to behave like (6), as done in Castañeda et al. [6] and Westebroek et al. [22]. If this is achieved exactly, there will not be any uncertain terms in the CLF dynamics, since  $\dot{V}_\varepsilon$  only depends on the matrices  $F$  and  $G$  of the input-output linearized dynamics.

Applying the following input to (2)

$$u(x, \mu) = \tilde{u}(x, \mu) + u_\theta(x, \mu), \quad (25)$$

with  $\tilde{u}$  as defined in (20) and with

$$u_\theta(x, \mu) := \left( L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x) \right)^{-1} (\alpha_\theta(x)\mu + \beta_\theta(x)), \quad (26)$$

yields

$$y^{(r)} = \mu + (\Delta_1(x) + \Delta_3(x)\beta_\theta(x)) + (\Delta_2(x) + \Delta_3(x)\alpha_\theta(x))\mu, \quad (27)$$

where  $\Delta_3(x) := \Delta_2(x) + I_m$ , and  $\theta \in \Theta \subset \mathbb{R}^N$  are parameters of a neural network to be learned. We can now clearly see the goal of the RL agent for this approach: design

a policy  $\alpha_\theta, \beta_\theta$  such that  $y^{(r)}$  is as close as possible to  $\mu$ . Thus, the time-wise reward function can be defined as

$$R(x, \mu) = -\|y^{(r)} - \mu\|_2^2 \quad (28)$$

where  $y^{(r)}$  is numerically estimated. After training, the  $\mu$  present in the final control input (25) is obtained by solving the CLF-QP of (13) in real time. We call this first approach *IO-RL + CLF-QP*.

### B. Reinforcement Learning for CLF-QP Based Controllers: Second Approach

In the second approach, we do not directly correct the uncertain terms of the transverse dynamics (24) as we did in the first approach. Instead, we directly analyze the impact of this uncertainty on the dynamics of the CLF.

For this approach, we assume that the CLF designed for the nominal model's transverse dynamics is also a CLF for the true plant's transverse dynamics (24).

In the presence of uncertainty,  $\dot{V}_\varepsilon$  becomes

$$\dot{V}_\varepsilon(\eta, z, \mu) = L_{\tilde{f}} V_\varepsilon(\eta, z) + L_{\tilde{g}} V_\varepsilon(\eta, z)\mu, \quad (29)$$

where

$$\begin{aligned} L_{\tilde{f}} V_\varepsilon(\eta, z) &= L_{\tilde{f}} V_\varepsilon(\eta) + \underbrace{2\eta^\top P_\varepsilon G \Delta_1(\eta, z)}_{=: \Delta_1^v(\eta, z)}, \\ L_{\tilde{g}} V_\varepsilon(\eta, z) &= L_{\tilde{g}} V_\varepsilon(\eta) + \underbrace{2\eta^\top P_\varepsilon G \Delta_2(\eta, z)}_{=: \Delta_2^v(\eta, z)}. \end{aligned} \quad (30)$$

Here,  $\tilde{f}$  and  $\tilde{g}$  are the nominal model input-output linearized dynamics: namely,  $\dot{V}_\varepsilon(\eta, \mu) = L_{\tilde{f}} V_\varepsilon(\eta) + L_{\tilde{g}} V_\varepsilon(\eta)\mu$ . Therefore, under uncertainty:

$$\dot{V}_\varepsilon(\eta, z, \mu) = \tilde{V}_\varepsilon(\eta, \mu) + \Delta_1^v(\eta, z) + \Delta_2^v(\eta, z)\mu. \quad (31)$$

In this second approach we use RL to estimate the uncertainty terms in  $\dot{V}_\varepsilon$ :  $\Delta_1^v$  and  $\Delta_2^v$ . For this purpose, we construct an estimate

$$\hat{V}_{\varepsilon, \theta}(\eta, z, \mu) = \tilde{V}_\varepsilon(\eta, \mu) + \beta_\theta^V(\eta, z) + \alpha_\theta^V(\eta, z)\mu, \quad (32)$$

where  $\theta \in \Theta \subset \mathbb{R}^N$  are again the neural network parameters to be learned. The goal of RL is then obvious: learn a policy  $\alpha_\theta^V, \beta_\theta^V$  such that  $\hat{V}_{\varepsilon, \theta}$  is as close as possible to  $\dot{V}_\varepsilon$ . Any reward function that penalizes the absolute value of the difference between the two terms can be used. More details on the specific RL implementation are discussed in Section V.

**Remark 1:** For convenience, it is assumed here that  $\alpha_\theta^V, \beta_\theta^V$  share the same network parameters  $\theta$ , but this does not need to be the case. In this paper, we will assume that all the policy functions to be learned are sharing the same parameters.

The estimate  $\hat{V}_{\varepsilon, \theta}$  in (32) is then used as our best guess of  $\dot{V}_\varepsilon$  for the optimization problem:

#### RL-CLF-QP:

$$\begin{aligned} \mu_\theta^*(x) &= \underset{\mu}{\operatorname{argmin}} \quad \mu^T \mu \\ \text{s.t.} \quad & \hat{V}_{\varepsilon, \theta}(\eta, z, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq 0 \quad (\text{RL-CLF}) \end{aligned} \quad (33)$$

**Remark 2:** We have illustrated the case in which the CLF is applied to the input-output linearized dynamics. The reason why we use a CLF on the input-output linearized dynamics instead of the full dynamics is that in this way we have a systematic way of computing a CLF candidate, whereas on the original nonlinear system this process could be challenging. However, this approach is not confined to the input-output linearization structure and is also applicable to any general nonlinear control-affine system.

#### IV. REINFORCEMENT LEARNING FOR CBF-CLF-QP BASED CONTROLLERS UNDER UNCERTAIN DYNAMICS

Having studied how to compensate for the effects of model uncertainty on CLF-based min-norm controllers, we will now extend our framework to the safety-critical CBF-CLF-QP by following a similar approach.

##### A. Reinforcement Learning for CBFs

In the presence of uncertainty, (16) becomes

$$\widetilde{B^{(r_b)}}(x, \mu) = L_{\tilde{f}}^{r_b} B(x) + L_{\tilde{g}} L_{\tilde{f}}^{r_b-1} B(x) \tilde{u}(x, \mu), \quad (34)$$

and the actual CBF's  $r_b^{th}$  derivative can be written as:

$$B^{(r_b)}(x, \mu) = \widetilde{B^{(r_b)}}(x, \mu) + \Delta_1^b(x) + \Delta_2^b(x)\mu, \quad (35)$$

where  $\Delta_1^b$  and  $\Delta_2^b$  are the uncertain terms that arise from the model-plant mismatch. We omit analytic expressions of  $\Delta_1^b, \Delta_2^b$  for conciseness, but they can be derived similarly to (23).

**Remark 3:** When the state of the system can be represented as  $x = [q, \dot{q}]^T$ , as in most robotic systems, even for high relative degree CBFs model uncertainty only affects the  $r_b^{th}$  time derivative of  $B$ , since  $B^{(r_b)}$  is the only term that depends on the plant dynamics through the vector fields  $f$  and  $g$ .

Next, we present how to estimate the uncertainty terms for the CBF and for other dynamic constraints using RL. The approach presented in Section III-A cannot be used here since the CBF functions depend on the full dynamics of the system, and not the transverse dynamics.

We build an estimator of  $B^{(r_b)}$ :

$$\widetilde{B^{(r_b)}}_{\theta}(x, \mu) = \widetilde{B^{(r_b)}}(x, \mu) + \beta_{\theta}^B(x) + \alpha_{\theta}^B(x)\mu, \quad (36)$$

and the goal of RL is to learn a policy  $\alpha_{\theta}^B, \beta_{\theta}^B$  such that  $\widetilde{B^{(r_b)}}_{\theta}$  is as close as possible to  $B^{(r_b)}$ .

In order to integrate everything in a new QP we define the new virtual input of the CBF dynamics as

$$\mu_b := \widetilde{B^{(r_b)}}_{\theta}. \quad (37)$$

In cases where the CBF also depends on a set of parameters  $\psi \in \mathbb{R}^q$ , then we need to define the CBF as  $B : \mathbb{R}^{n \times q} \rightarrow \mathbb{R}$ . The neural-network policy will now need to take  $\psi$  as additional inputs  $\alpha_{\theta}^B : \mathbb{R}^{n \times q} \rightarrow \mathbb{R}^m$ ,  $\beta_{\theta}^B : \mathbb{R}^{n \times q} \rightarrow \mathbb{R}$  and the proposed estimate of the  $r_b^{th}$  time derivative of  $B$  becomes:

$$\widetilde{B^{(r_b)}}_{\theta}(x, \mu, \psi) = \widetilde{B^{(r_b)}}(x, \mu, \psi) + \beta_{\theta}^B(x, \psi) + \alpha_{\theta}^B(x, \psi)\mu. \quad (38)$$

##### B. Reinforcement Learning for Additional Control-Affine Dynamic Constraints

Now we study the effects of uncertainty on other linear constraints that depend on the dynamics of the system:

$$\underbrace{A_c(x, f, g)\mu + b_c(x, f, g)}_{=: \zeta(x, \mu)} \leq 0. \quad (39)$$

In the presence of model mismatch we have

$$\begin{aligned} b_c(x, f, g) &= b_c(x, \tilde{f}, \tilde{g}) + \Delta_1^c(x), \\ A_c(x, f, g) &= A_c(x, \tilde{f}, \tilde{g}) + \Delta_2^c(x), \end{aligned} \quad (40)$$

where  $\Delta_1^c$  and  $\Delta_2^c$  represent the uncertainty terms. We can then define the nominal constraint

$$\tilde{\zeta}(x, \mu) = b_c(x, \tilde{f}, \tilde{g}) + A_c(x, \tilde{f}, \tilde{g})\mu. \quad (41)$$

And the real value of the constraint can be expressed as

$$\zeta(x, \mu) = \tilde{\zeta}(x, \mu) + \Delta_1^c(x) + \Delta_2^c(x)\mu. \quad (42)$$

We can build an estimator of the form

$$\hat{\zeta}_{\theta}(x, \mu) = \tilde{\zeta}(x, \mu) + \beta_{\theta}^C(x) + \alpha_{\theta}^C(x)\mu, \quad (43)$$

with a learned policy  $\alpha_{\theta}^C, \beta_{\theta}^C$ . The goal of the RL agent is again in this case to make the estimator  $\hat{\zeta}_{\theta}$  as close as possible to  $\zeta$ . Expanding  $\tilde{\zeta}$  we can rewrite the estimator as

$$\hat{\zeta}_{\theta}(x, \mu) = \underbrace{(b_c(x, \tilde{f}, \tilde{g}) + \beta_{\theta}^C(x))}_{=: b_{\theta}^C(x)} + \underbrace{(A_c(x, \tilde{f}, \tilde{g}) + \alpha_{\theta}^C(x))}_{=: A_{\theta}^C(x)} \mu. \quad (44)$$

So far, we have explained our method of constructing an estimator of a single  $B^{(r_b)}$  and a single  $\zeta(x, \mu)$ . This can be applied to  $n_b$  multiple CBFs and  $n_c$  multiple control-affine constraints. The final optimization problem, which includes all the learned estimates of the uncertain terms is:

##### RL-CBF-CLF-QP:

$$\mu_{\theta}^*(x) = \underset{\mu, \mu_b, d}{\operatorname{argmin}} \quad \mu^T \mu + p \, d^2 \quad (45)$$

$$\text{s.t.} \quad \hat{V}_{\varepsilon, \theta}(\eta, z, \mu) + \frac{\lambda}{\varepsilon} V_{\varepsilon}(\eta) \leq d \quad (\text{RL-CLF})$$

$$\text{for } i = 1 \cdots n_b \quad \mu_{b,i} + K_{b,i} \eta_{b,i} \geq 0 \quad (\text{RL-CBF})$$

$$\mu_{b,i} = \widetilde{B^{(r_b)}}_{i, \theta}(x, \mu)$$

$$\text{for } j = 1 \cdots n_c \quad A_{j, \theta}^c(x)\mu + b_{j, \theta}^c(x) \leq 0 \quad (\text{RL-Constraints})$$

#### V. REINFORCEMENT LEARNING-BASED FRAMEWORK

In this section, we present a unified RL framework that can learn the uncertainty terms in the CLF, CBF, and other dynamic constraints by building the terms specified in the earlier sections as  $\alpha_{\theta}^V, \alpha_{\theta}^B, \alpha_{\theta}^C, \beta_{\theta}^V, \beta_{\theta}^B, \beta_{\theta}^C$ .

A diagram of this framework is illustrated in Fig. 1. The RL agent learns a policy, which is a combination of uncertainty terms in CLF, CBF and other dynamic constraints. These terms are then added to the QP constraints derived from the nominal model, resulting in the estimates of the true plant constraints. Using these estimates, the RL-CBF-CLF-QP optimization

problem, in which model uncertainty is addressed, is solved point-wise in time to obtain the control input.

The reward function of the learning problem is designed such that it minimizes each of the estimation errors. Thus, the time-wise loss functions are defined as:

$$\begin{aligned} l_{V,\theta} &:= \|\dot{V}_\varepsilon - \hat{V}_{\varepsilon,\theta}(x, \mu)\|^2 \\ l_{B,\theta} &:= \|B^{(r_b)} - \widehat{B}^{(r_b)}_\theta(x, \mu)\|^2 \\ l_{C,\theta} &:= \|\zeta - \hat{\zeta}_\theta(x, \mu)\|^2 \end{aligned} \quad (46)$$

It is important to note that the true plant's dynamics information is not used for computing the values of these loss functions. We use explicit expressions for  $V_\varepsilon$ ,  $B$  and  $\zeta$  and compute the time-derivatives  $\dot{V}_\varepsilon$ ,  $B^{(r_b)}$  using numerical differentiation. For the CBF, it is important to note that regardless of the value of  $r_b$  we only need to do numerical differentiation once, as follows from Remark 3.

A canonical RL problem can be formulated, with the reward for a given state  $x$  defined as the weighted sum of the negative loss functions in (46), in addition to a user-specific failure-case penalty  $-l_e : \mathbb{R}^n \rightarrow \mathbb{R}$ :

$$R(x, \theta) = -w_v l_{V,\theta} - \sum_{i=1}^{n_b} w_{b,i} l_{B_i,\theta} - \sum_{j=1}^{n_c} w_{c,j} l_{C_j,\theta} - l_e(x). \quad (47)$$

The learning problem is then defined as:

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{x_0 \sim X_0, w \sim \mathcal{N}(0, \sigma^2)} \int_0^T R(x(\tau), \theta) d\tau, \\ \text{s.t.} \quad & \dot{x} = f(x) + g(x)\tilde{u}(x, \mu_\theta^*(x) + \omega), \end{aligned} \quad (48)$$

where  $\mu_\theta^*(x)$  is the solution of (45),  $X_0$  is the initial state distribution, and  $w \sim \mathcal{N}(0, \sigma^2)$  is white noise added to encourage exploration. A discretized version of this problem can be solved using conventional RL algorithms.

**Remark 4:** While running training experiments or simulations, it is assumed that the robot operates under the true plant dynamics. We will later show in Section VII that the trained policy works well even when the true plant in the evaluation differs from the plant of the training environment.

## VI. APPLICATION TO BIPEDAL ROBOTS

The goal of this section is to validate that the RL-CBF-CLF-QP framework enables safety-critical control when model uncertainty is present. We test our method on RABBIT [7], a planar five-link bipedal robot, walking on a discrete terrain of stepping stones with one step preview.

### A. Simulation Settings

We run two simulation scenarios with our method and offer comparisons with the previous methods. The first simulation consists of RABBIT simply walking on a flat terrain. We evaluate the CLF based methods in Section III in this scenario. This is to verify only the stabilizing capacity of our proposed method under model uncertainty. In the second simulation, we put the robot on a discrete terrain of randomly spaced stepping stones (Fig. 4). The robot's task here is to always place the

foot on the next stepping stone, while managing the stability and not violating the contact-force constraint. The full RL-CBF-CLF-QP is tested in this simulation scenario.

The main model uncertainty in both demonstrations is introduced by scaling all mass and inertia parameters of each link by a constant scale factor = 2, i.e. the nominal model's mass and inertia terms are half of those of the actual plant.

A single periodic walking gait trajectory is generated offline by the Fast Robot Optimization and Simulation Toolkit (FROST) [10]. The output function  $h(x)$  is defined as the difference between the actuated joint angles and the desired trajectory's joint angles from the obtained periodic orbit. The gait's nominal step length is 0.35m. Finally, a torque saturation of 200Nm is applied to the control inputs of all simulations, including training and evaluation.

### B. Reinforcement Learning Settings

We train our agent using a standard Deep Deterministic Policy Gradient Algorithm (DDPG) [18]. The input for the actor neural network is 14 observations, which is RABBIT's full state  $x$ , in addition to the CBF parameter  $\psi = l_{min,k}$  corresponding to the minimum step length of the  $k$ th stepping stone (Fig. 4) in the second simulation. We use two CBFs  $B_1$  and  $B_2$  to constrain the position of the swing foot so that it lands on the stepping stone, as shown in Fig. 4. We use two dynamic constraints  $C_1$  and  $C_2$  which correspond to the unilateral normal force and friction cone constraints respectively. The output dimension is 25, corresponding to the  $4 \times 1$   $\alpha_\theta^V, \alpha_\theta^{B_1}, \alpha_\theta^{B_2}, \alpha_\theta^{C_1}, \alpha_\theta^{C_2}$  and the  $1 \times 1$   $\beta_\theta^V, \beta_\theta^{B_1}, \beta_\theta^{B_2}, \beta_\theta^{C_1}, \beta_\theta^{C_2}$ .

Both actor and critic neural networks have two hidden layers of widths 400 and 300. This agent is trained on the simulation of ten walking steps per episode, and a discrete time step  $T_s = 0.01$ sec is used. The failure cases are determined by the robot's pose. Training on six multiple cores of Intel(R) Core(TM) i5-9400F CPU (2.90GHz) without the use of GPU took about 34 seconds per episode. The final agent in use is obtained after 110, 79 and 133 episodes for IO-RL + CLF-QP, RL-CLF-QP and RL-CBF-CLF-QP respectively.

## VII. RESULTS

During the evaluation, the robot is tested not only on the uncertainty that is introduced in the training, but in addition to it, two other kinds of uncertainty are also introduced. First, the robot's motor dynamics that restricts the rate of change of joint torques is applied in every evaluation. The time constant of motors used in the simulation is 0.004 seconds. Second, the robot is also tested on an alternative kind of uncertainty, which consists of an added weight to the torso of the robot, instead of scaling the links masses and inertias. This weight can represent the robot carrying a payload, and it is deliberately introduced to evaluate the trained policy's robustness to an unfamiliar kind of uncertainty that it was not trained on.

### A. Simulation 1: Bipedal Walking on Flat Ground

For the first simulation, we evaluate the two RL approaches for CLF explained in Section III, and compare them with

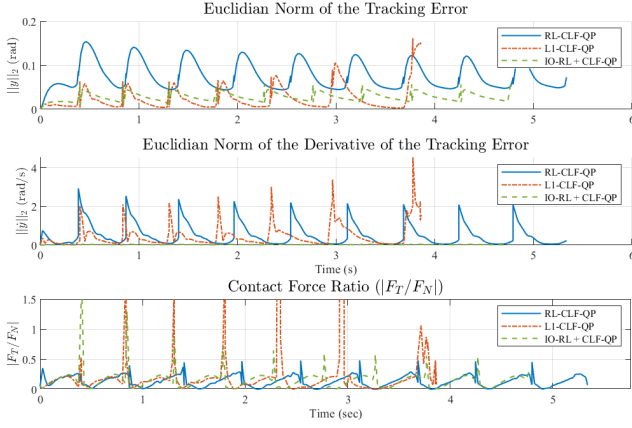


Fig. 2: Tracking error (top), its derivative (middle), and tangential-normal contact force ratio (bottom) of IO-RL + CLF-QP (Sec. III-A), RL-CLF-QP (Sec. III-B), and L1-CLF-QP [13] controllers, simulated for ten walking steps, where the robot’s mass and inertia values are scaled by a factor of 2. Both IO-RL + CLF-QP and RL-CLF-QP maintain the stability while L1-CLF-QP fails. Only the RL-CLF-QP satisfies the friction cone constraint  $|F_T/F_N| \leq k_f = 0.8$ .

the standard L1 Adaptive CLF-QP method of Nguyen and Sreenath [13], which guarantees the CLF to be bounded to a small value under model uncertainty if using a sufficiently large adaptation gain.

As illustrated in Fig. 2, both of the proposed methods manage to get RABBIT to stably walk for multiple steps, while the L1 Adaptive CLF-QP controller leads to failure. The original nominal CLF-QP, although not shown in the figure, also fails under this scaled model uncertainty. Note that all three methods do not have friction constraints in the QP and could potentially violate them. In particular, the RL-CLF-QP method succeeds in satisfying the friction constraint ( $|F_T/F_N| \leq k_f = 0.8$ ) for all steps, the IO-RL + CLF-QP exceeds the limit in the first two steps, and the L1-CLF-QP violates it for multiple steps. Therefore, IO-RL + CLF-QP needs the inclusion of friction constraints in the QP.

Displayed in Fig. 3 is the plot of tracking error and contact force ratio of the three controllers when, instead of the mass-inertia-scaling, an additional torso weight of 32kg (100% of the robot mass) is introduced. It is notable that both the RL-CLF-QP and IO-RL + CLF-QP manage to adapt to this uncertainty, which has not been faced during the training. Furthermore, the RL-CLF-QP manages to stabilize the walking gait with an additional torso weight of up to 72kg (225% of robot mass). On the other hand, IO-RL + CLF-QP manages to adapt to additional weights up to 53kg (166% of robot mass).

### B. Simulation 2: Bipedal Walking on Stepping Stones with One Step Preview

We now evaluate the full RL-CBF-CLF-QP method with the safety-critical constraint of walking on stepping stones and the inclusion of friction constraints, which are dependent on the dynamics. In this simulation scenario, for each step the robot

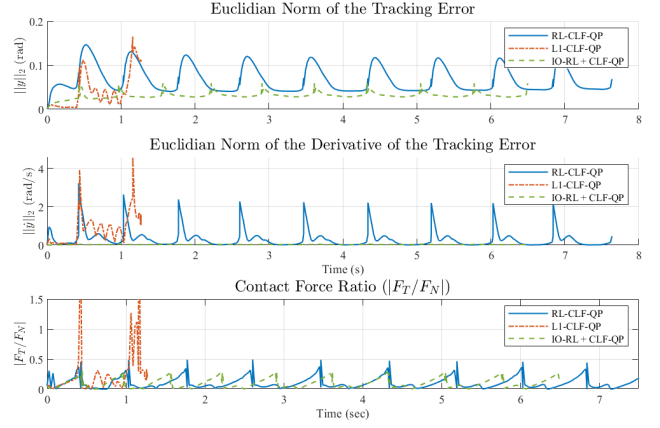


Fig. 3: Tracking error (top), its derivative (middle), and contact force ratio (bottom) of the three CLF-based controllers, simulated for ten walking steps with the additional torso weight 32kg (this amounts to the weight of RABBIT, i.e. 100% additional weight).

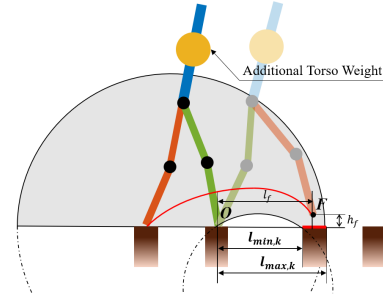


Fig. 4: Safety Constraint: In order to guarantee the swing foot lands on the stepping stone, we use two CBFs to ensure the swing foot position  $F$  is within the grey area during the entire walking step.

faces a random placement of a stepping stone. Therefore, when the swing foot hits the ground at the end of the step, we want the step length to be within a specific range:

$$l_{min,k} \leq l_k \leq l_{max,k}, \quad (49)$$

where  $k$  indicates the step index. Two position-constraints-based second order ECFBs parameterized by  $l_{min,k}$ ,  $l_{max,k}$  that are a sufficient condition for (49) are devised by Nguyen and Sreenath [15]. Basically these constraints imply that the swing foot position ( $F$  in Fig. 4) needs to stay within the grey area. Note that  $l_{min,k}$ ,  $l_{max,k}$  change for every step.

We also include contact force constraints in the RL-CBF-CLF-QP as control-affine dynamic constraints, following the procedure of Subsection IV-B. These are important since the original CBF-CLF-QP violates the friction cone and the unilateral normal force constraints repeatedly.

The robot is trained to walk on randomly spaced stepping stones, of which  $l_{min}$  is sampled from a normal distribution  $\mathcal{N}(0.35m, 0.02m)$ , truncated at  $2.5\sigma$ .  $l_{max}$  is set as  $l_{min} + 0.05m$ .

Fig. 5 shows the result of the evaluation, where the robot

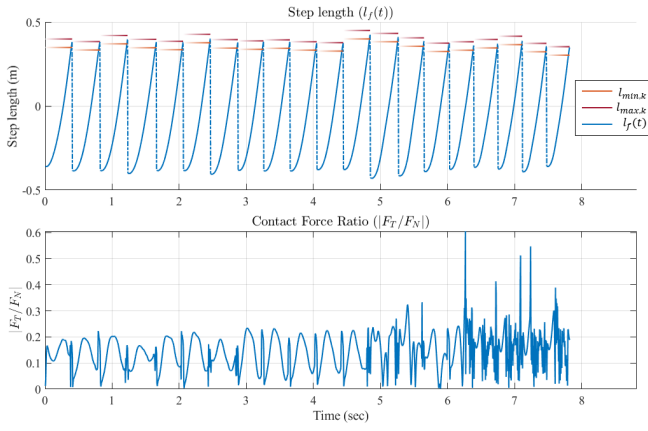


Fig. 5: Results of the simulation of 20 steps of walking on stepping stones, where the robot’s mass and inertia values are scaled by a factor of 2. (Top) History of swing foot position  $l_f$  for each step, with the stepping stone constraints  $l_{min}$ ,  $l_{max}$ . (Bottom) History of tangential-normal contact force ratio that satisfies to stay below  $|F_T/F_N| \leq k_f = 0.8$ .

walks on 20 randomly spaced stepping stones. We can check that the foot placement is always on the stepping stones. Also, it is verified that the contact force never exceeds the friction limit. Note that the sample distribution of  $l_{min}$  here is same as during training.

Whereas our RL-CBF-CLF-QP method performs well, we have also tested the nominal model-based CBF-CLF-QP method on this simulation for comparison. The CBF-CLF-QP is also solved together with the friction constraints. However, it violates the step length safety constraints after an average of  $5.6 \pm 4.64$  steps. This value is obtained from 10 random executions of 20 steps simulation.

Finally, for the case of having an additional torso weight applied to the original unscaled plant, RL-CBF-CLF-QP still manages to stay within the safety and friction constraints when the weight is in the range of [43kg, 72kg] (134-225% of robot mass).

## VIII. DISCUSSION

In Sections VI and VII, we have demonstrated that our method can compensate well for the trained model uncertainty and that it shows some robustness to the introduction of additional uncertainty during evaluation. It is important to note that our method is not restricted to mass and inertia scaling uncertainties, rather they have been used as illustrative examples for this paper. We have additionally tested our framework for other uncertainties: a simplified model of joint friction (assuming that joint friction reduces motor power by a 15%, value taken from Chevallereau et al. [7]), joint damping (up to  $1 \text{ (rad/s}^2\text{)}/(\text{rad/s})$ ) and bending of links (up to 5% of their length) obtaining successful results.

However, a primary drawback of our approach is that we need the designed nominal controller to not rapidly fail on the uncertain system before RL can learn the uncertainty. This may not always be possible depending on the level of uncertainty.

Following this same reasoning, for high levels of uncertainty the CLF designed for the nominal model may not be a CLF for the true plant, in which case our assumption would not hold and the method would fail. There is ongoing research on designing CLFs for systems with uncertain dynamics [16, 21] that could be used to solve this issue, since our method is not restricted to any specific CLF.

An illustration of the aforementioned limitation is that we have also tested our framework for mass-inertia uncertainty scales of 0.7 and 0.5. For the case of scale=0.7, our framework produces a stabilizing controller that respects safety and friction constraints for indefinitely long periods of walking, whereas the nominal model-based controller fails after just one step. In contrast, for the scale of 0.5, the nominal controller fails after just 0.06 seconds, which makes the training a lot more challenging and our framework fails.

Another limitation is that the measurements of  $\dot{V}_e$  and  $B^{(r_b)}$  obtained from numerical differentiation could be noisy in experiments. However, a similar method is proved to be effective in real experiments in Westenbroek et al. [22], where an estimate of the output acceleration is computed by numerical differentiation, which is used to train the RL agent.

In this paper we specifically use RL to learn the uncertainty terms since in this way we can gradually enhance the quadratic program’s feasibility and performance while learning the safety constraints, increasingly exploring the state space of our interest. Moreover, RL allows us to unify the learning processes of uncertainty terms in multiple safety constraints to a single process. However, there are several works tackling similar problems with other learning methods, such as supervised learning [19, 20], and deciding which is the best approach is still an open question that might depend on the specific properties of the platform used for testing. We plan to address this in the future, adapting our safety-critical control framework to other learning methods.

Finally, it must be noted that feasibility of a CBF-CLF-QP with additional constraints, such as friction, is not guaranteed in general. However, using the trained RL-CBF-CLF-QP model, we observe that the feasibility drastically improves compared to the nominal CBF-CLF-QP.

## IX. CONCLUSION

We have addressed the issue of model uncertainty in safety-critical control with an RL-based data-driven approach. We have presented a formal analysis of uncertainty terms in CBF and CLF constraints, in addition to other dynamic constraints. Our framework includes two core components: 1) an RL agent which learns to minimize the effect of model uncertainty in the aforementioned safety constraints, and 2) the formulation of the RL-CBF-CLF-QP problem that solves online for the safety-critical control input. The proposed framework is tested on RABBIT, an underactuated nonlinear bipedal robot. We demonstrate walking on randomly spaced stepping stones with one step preview under high model uncertainty.

# REFERENCES

- [1] A. D. Ames and M. Powell. Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs. In *Control of Cyber-Physical Systems*, pages 219–240. Springer, 2013.
- [2] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, Apr 2014.
- [3] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278, Dec 2014.
- [4] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomlin. Goal-driven dynamics learning via bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173, Dec 2017.
- [5] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pages 908–918, 2017.
- [6] F. Castañeda, M. Wulfman, A. Agrawal, T. Westbroek, S. S. Sastry, C. J. Tomlin, and K. Sreenath. Improving input-output linearizing controllers for bipedal robots via reinforcement learning. *arXiv preprint arXiv:2004.07276*, 2020.
- [7] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. Canudas-De-Wit, and J. W. Grizzle. Rabbit: a testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.
- [8] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7): 2737–2752, July 2019.
- [9] K. Galloway, K. Sreenath, A. D. Ames, and J. W. Grizzle. Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs. *IEEE Access*, 3:323–332, 2015.
- [10] A. Hereid and A. D. Ames. Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726. IEEE, 2017.
- [11] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.
- [12] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, Jan 2016.
- [13] Q. Nguyen and K. Sreenath. L1 adaptive control for bipedal robots with control lyapunov function based quadratic programs. In *2015 American Control Conference (ACC)*, pages 862–867, July 2015.
- [14] Q. Nguyen and K. Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328, July 2016.
- [15] Q. Nguyen and K. Sreenath. Optimal robust time-varying safety-critical control with application to dynamic walking on moving stepping stones. In *ASME Dynamic Systems and Control Conference*, 2016.
- [16] S. M. Richards, F. Berkenkamp, and A. Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 466–476, Oct 2018.
- [17] S. Sastry. *Nonlinear systems: analysis, stability, and control*. Vol. 10. Springer Science and Business Media, 1999.
- [18] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML) - Volume 32*, pages I–387–395, 2014.
- [19] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. *arXiv preprint arXiv:1903.01577*, 2019.
- [20] A. J. Taylor, A. Singletary, Y. Yue, and A. D. Ames. Learning for safety-critical control with control barrier functions. *arXiv preprint arXiv:1912.10099*, 2019.
- [21] J. Umlauft, L. Pöhler, and S. Hirche. An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process. *IEEE Control Systems Letters*, 2(3):483–488, 2018.
- [22] T. Westbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin. Feedback linearization for unknown systems via reinforcement learning. *arXiv preprint arXiv:1910.13272*, 2019.