Model-Based Design for Legged Robots: Predictive Control and Reinforcement Learning

by

Ayush Agrawal

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Koushil Sreenath, Chair
Professor Francesco Borrelli
Professor Claire Tomlin

Fall 2022

Model-Based Design for Legged Robots: Predictive Control and Reinforcement Learning

Abstract

Model-Based Design for Legged Robots: Predictive Control and Reinforcement Learning

by

Ayush Agrawal

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Associate Professor Koushil Sreenath, Chair

Legged robots have the potential to provide extreme mobility in highly rugged terrain. Developing such locomotion capabilities within these robots is challenging for several reasons, including their inherent hybrid and nonlinear dynamics. Moreover, uncertainties in the robot's model arising due to external payloads, joint friction, wear and tear, and uncertainties introduced by environmental factors such as changing contact conditions or force perturbations can significantly hinder the performance and reliability of these robots. Developing safe, reliable, and robust feedback controllers is crucial as we begin to deploy such robots into environments with humans. Moreover, the underlying feedback controllers must be able to quickly adapt to rapidly changing environmental factors that can significantly affect their performance.

Recent developments in Reinforcement Learning have shown tremendous success and impressive results on several legged robot platforms to navigate challenging terrain. While these methods can generate very complex behaviors, they are highly sample inefficient as they do not take into account any knowledge of the dynamical structure of the robot. Model-based methods such as Control Lyapunov Functions (CLFs), Control Barrier Functions (CBFs), and Model Predictive Control (MPC), on the other hand, provide us with a set of tools to achieve desired control objectives while remaining within specified constraints for the closed-loop system. The performance of both RL-based control and model-based methods can significantly degrade if an accurate model of the underlying system is not known. In such cases, achieving the best performance will require learning from real-world data. This thesis develops planning algorithms and feedback controls through the lens of model-based techniques to achieve safe, stable, and robust legged locomotion on challenging terrain. In particular, we present a trajectory generation method to achieve aperiodic running with precise foot placement for a 2D bipedal robot model. We then develop a novel coordinate-free geometric MPC for the Cassie biped and validate our approach through several hardware experiments. We apply the developed trajectory generation method and geometric MPC

on a quadruped robot to navigate challenging terrain with visual feedback. Using tools from model-based methods such as CLFs and CBFs, we then develop novel reward function shaping methods to achieve safe and robust locomotion policies for bipedal and quadrupedal robots. We show that our method can significantly reduce the sample complexity to learn a stabilizing controller, which allows us to finetune policies directly on hardware using only a few seconds to a few minutes of data.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis could not have been completed without the love, support, and mentorship of the many talented and wonderful people I've met along my PhD journey.

First, I would like to thank my advisor, Professor Koushil Sreenath, with whom I've had the pleasure of working for over seven years. Thank you, Koushil, for giving me this unique opportunity to work on such challenging problems alongside the coolest robots on this planet, for spending numerous hours helping me with experiments, for your constant support and guidance throughout graduate school and for showing me what it takes to be great mentor.

Thank you, Professor Claire Tomlin and Professor Francesco Borrelli, for being on my committee and for the many wonderful discussions I have had over the years during office hours and group meetings.

Thank you to my group members: Quan, for showing me what it takes to be a great researcher and for inspiring me to pursue a PhD; Avinash, for our endless discussions on geometric control and RL; Bike and Shu, for the countless hours helping me with experiments; Zhongyu, for showing me what it takes to be a great PhD student mentor. Thank you Katie, Jun, Matthew, and Akshay for your support throughout my graduate studies.

Thank you to my many collaborators: Tyler and Fernando for our countless discussions on model-based control, CBFs, RL and for the many hours we spent on debugging hardware; Jason and Somil for teaching me so much more about reachability and for the many wonderful conversations around hybrid systems, locomotion and learning.

Thank you to Akshara Rai for providing me with the opportunity to work with her as an intern at FAIR and for her mentorship and support during my internship. Thank you to Dennis Da for taking the time to chat about controls and learning for legged robots.

Thank you to my Bala and Prasanth for tolerating me as your housemate for over four years, for all the fun road trips and house parties, and for being the best friends I could ever ask for.

Most importantly, thank you to my parents for being so patient and supportive of my decision. I couldn't have done it without you by my side.

Finally, I would like to acknowledge the financial support from the National Science Foundation (Grants CMMI-1944722 and IIS-1834557), from META AI through BAIR Open Research Commons, from InnoHK of the Government of Hong Kong via the Hong Kong Centre for Logistics Robotics and from UC Berkely through the scholarships: 1) William S. Floyd, Jr. Graduate Student Fellowship, 2)The Gordon M. and Merle I. Steck Cal Club Endowed Scholarship, 3) The William C. Webster Graduate Fellowship and 4) Graduate Division Block Grant Award.

Chapter 1

Introduction

1.1 Motivation

Animals with legs exhibit highly dynamic locomotion capabilities in nature. They are able to locomote with extreme agility on challenging terrain using visual feedback, adapt to changing environmental conditions, and leverage contacts to stabilize themselves. Animals can also change their gait type depending on the locomotion speed to minimize energy consumption. Taking inspiration from the versatile locomotion capabilites of these animals, roboticists have sought to build legged robots that have the ability to navigate on rugged and unstructured terrain. These capabilities make them ideal candidates for various applications, such as search and rescue operations, space exploration, building maintenance, and offshore platform inspection. Moreover, the human-like form factor of robots such as Agility Robotics' Digit [2], and Boston Dynamics' Atlas [21] are particularly well suited for deployment in spaces designed for humans, such as warehouses and inside homes with narrow corridors and doorways. Robotic legged locomotion can also inform the design and control of interactive and assistive devices such as lower-limb exoskeletons and prostheses [6, 59].

1.2 Challenges

Enabling such autonomous locomotion capabilities in legged robots is, however, challenging due to their increased complexity compared to other robots such as wheeled platforms and fixed-base manipulators. Additionally, several components, including low-level balance control, high-level motion planning, state estimation, and perception, must work together reliably. Moreover, as we begin to integrate these robots into spaces around humans, it is important for the underlying feedback controllers to be robust against uncertainties in the environment and the robot model, both of which can significantly change over its operational life cycle. There are several challenges associated with designing feedback controllers for legged robots.

1. *Nonlinear and Hybrid Dynamics:* Legged locomotion such as walking, running, and hopping can be described by a set of nonlinear continuous phases followed by discrete impact events [53]. This nonlinear and hybrid nature of locomotion, coupled with the high-dimensionality of legged robots, makes it particularly challenging to synthesize stabilizing feedback controllers and planning algorithms that can be realized in real-time.

2. *Constrained Environments:* Legged robots need to navigate in constrained environments, such as across narrow passageways and over flights of stairs. This places tight constraints on the foot positions and base pose of the robot. Additionally, these robots need to operate within torque limits and friction cone limits imposed by the environment.

3. *State Estimation:* Model-based control, especially that rely on estimates of centroidal quantities, requires accurate state estimation, which can be particularly challenging for legged robots that impact with the ground.

4. *High-dimensional visual inputs:* For legged robots to interact with their environment, they must have the capability to perceive and reason about the surrounding terrain.

1.3 Legged Locomotion Research over the Years

Legged robotics research has a rich history dating back to the 1960's, when one of the first walking machines, the *Walking Truck*, was developed by General Electric in 1965. The machine was mechanically driven through hydraulic valves coupled to the human operator's hand and foot motions and without any computer control.

Over the last five decades, researchers around the world have worked to address several challenges associated with robotic legged locomotion. The last decade in particular, has seen breakthroughs in trajectory generation, optimal control, and numerical optimization techniques, as well as reinforcement learning that has led to impressive results in agile locomotion behaviors on legged systems. One of the earliest and most notable works on dynamic legged locomotion was by Marc Raibert's group at the MIT LegLab in the early 1980s, where they developed a series of highly dynamic hopping and running robots [115]. The proposed feedback controllers relied on simple heuristics to maintain the velocity, body height and posture. Later that decade, Tad McGeer showed it was possible to achieve stable *passive dynamic walking* without the need for any feedback control [90]. As mentioned earlier, a primary challenge with designing feedback controllers for legged robots arises due to their high dimensionality and having to coordinate several joints together, in addition to their nonlinear and hybrid dynamics. This inspired several works on utilizing *reduced-order* models for planning foot-steps and ground reaction forces to stabilize the robot while reducing the computational burden for online planning and control. Taking a departure from reduced-order models, Grizzle et al. proposed the *Hybrid Zero Dynamics* (HZD) framework that leveraged

nonlinear feedback control, hybrid dynamics of the robot, and numerical computation of limit cycles for walking and running.

With the advances in numerical optimization, legged robotics research saw a paradigm shift in control methods. While reduced-order models are still popular today, recent model-based feedback controllers leverage optimization programs such as Quadratic Programs (QPs) that can be evaluated in real-time. An advantage of such methods is that physical constraints such as torque limits and friction cone constraints can be incorporated as constraints in the optimization. Advances in numerical optimization have also led to efficient ways to compute trajectories for the full nonlinear and hybrid model of the robot.

More recently, another breakthrough in legged locomotion control took place with the advances in robotic *Reinforcement Learning*. Several recent works have shown impressive results on achieving robust and agile locomotion on a variety of legged platforms. RL is not only able to synthesize and stabilize highly complex behaviors [122, 143], but also learn robust and adaptive policies in the presence of model and environmental uncertainties [121, 79]. RL has also proven to be effective in the presence of high dimensional visual inputs [1, 91].

1.4 Contributions

This thesis explores model-based control design and learning techniques for developing agile and robust legged locomotion control for bipedal and quadrupedal robots. In particular, we take a first-principles approach and use tools from trajectory optimization, Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs), geometric mechanics, Model Predictive Control (MPC), and Reinforcement Learning (RL). The key contributions of this thesis are summarized below and outlined in Figure 1.1.

1. *Planning and Controls for Robotic Running with Precise Foot Placement:* Due to their high dimensionality and nonlinear and hybrid nature of the dynamics, planning trajectories for legged robots in real-time is challenging, especially for agile maneuvers such as running with multiple modes. This thesis explores the use of trajectory libraries generated offline for dynamic locomotion for bipedal robotic running.

2. *Geometric Model Predictive Control:* We develop a novel coordinate-free Model Predictive Control algorithm on a rigid-body reduced order model for legged robots evolving on the $SO(3)$ manifold. We show that our method is versatile and applicable to a different legged robotic platform, including the bipedal robot Cassie, the humanoid robot Digit and the Unitree A1, a small-scale quadrupedal robot. We perform several indoor and outdoor experiments with the A1 and Cassie robots on a variety of terrain to demonstrate the robustness of our approach.

3. *Vision-Aided Locomotion on Challenging Terrain:* Utilizing our method to generate trajectory libraries offline and the geometric MPC algorithm, we develop a planning and controls framework for a quadruped robot to navigate a field of stepping stones using visual inputs from a depth camera.

4. *Robust and Efficient Robotic Reinforcement Learning:* We explore the combination of model-based control with model-free policy optimization methods to develop safe and robust feedback controllers for legged robots.

1.5 Thesis Organization

The thesis is divided into two parts. In Part I, we develop model-based feedback controllers for quadruped and bipedal robots to navigate challenging terrain. We begin our presentation in Chapter 3, where we extend our framework for walking to running with precise foot placement on stepping stones for a 2D bipedal robot. Our initial hypothesis suggested a straightforward extension to running. However, a primary challenge with running is the loss of control authority during flight phases. This places tight constraints on the forward take-off velocity to achieve precise foot placement. Through extensive numerical simulations, we illustrate that the choice of outputs is crucial to obtain accurate foot placement. We propose a set of output variables that enables the controller to achieve precise foot placement.

While our previous works were focused on 2D bipedal locomotion in constrained settings, in Chapter 4, we extend our MPC approach to the bipedal robot platform Cassie. We illustrate the efficacy of our controller in handling a wide variety of outdoor terrain, including large slopes, grass, mulch, stepping down curbs, and external disturbances in the form of pushes.

In Chapter 5, we illustrate the effectiveness of offline motion libraries and online model predictive control for a quadruped robot to autonomously navigate a field of stepping stones using visual feedback.

For robots to reliably work in the real-world, the underlying locomotion controller must be robust to unmodeled disturbances such as added payloads and uncertainty in the operating environment. A primary disadvantage of model-based techniques is the requirement of an accurate whole-body or reduced-order model of the robot. An inaccurate model can significantly degrade the control performance and lead to undesired behavior. Additionally, model-based techniques like MPC can require a significant amount of parameter tuning (such as cost function weights and feedback gains) on hardware to achieve the desired performance.

In Part II, we aim to address some of these challenges through data-driven approaches.

In particular, Chapter 6 develops a framework inspired by CLFs and CBFs to learn a safe and stabilizing controller when the true dynamics of the underlying system are unknown. We validate our approach through numerical simulations of a walking robot on randomly spaced stepping stones.

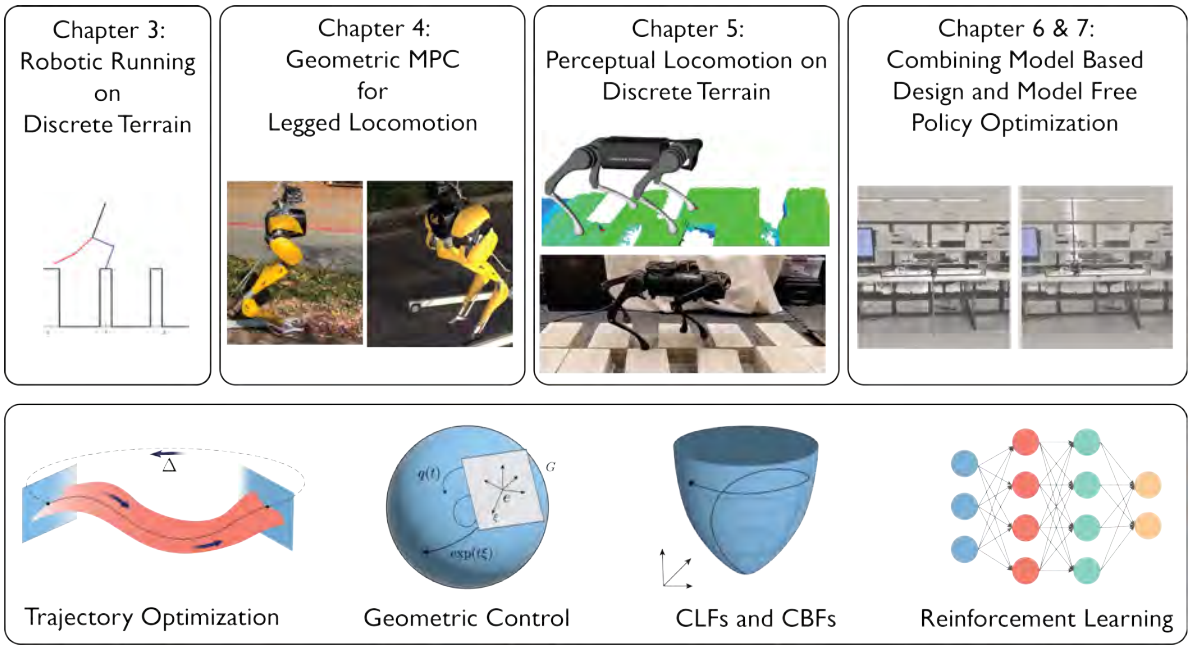


Figure 1.1: An overview of this thesis.

While our previous work focused on learning a policy entirely in simulation, it did not allow for adapting to disturbances that were out-of-distribution on hardware. In Chapter 7, we develop a novel cost-shaping method inspired by CLFs to rapidly learn stabilizing policies using data collected from hardware. We perform several experiments on a cartpole system and on the A1 robot, along with simulations on a 2D model of the RABBIT biped to test the efficacy of our approach.

1.6 Chapter Summary

In this chapter, we presented the main motivation to develop legged robots and the primary challenges associated with designing feedback controllers for such robots. We presented a summary of legged locomotion control literature ranging from reduced-order pendulum models to reinforcement learning methods. The next chapter presents dynamical models of legged robots and a method to obtain periodic trajectories for legged locomotion.

Chapter 2

Robot Dynamics and Trajectory Optimization

In this chapter, we first present the dynamical models of the various robots considered in this dissertation. We will use these models to develop model-based feedback controllers in Part I as well as design cost functions for learning safe and robust locomotion policies in Part II. Next, we present a method to obtain periodic orbits for legged locomotion using developed the hybrid models.

2.1 Dynamical Models for Legged Locomotion

Legged locomotion can be characterized by of alternating continuous phases depending on the nature of interaction with the environment. For example, walking in a bipedal robot can be described by alternating modes of *single-support* and *double-support* phases when one or two feet are in contact with the ground, respectively. Similarly, *trotting* in a quadrupedal robot can be described by alternating phases of double-support, when two diagonally opposing feet are in contact with the ground, and quadruple-support when all four feet are in contact with the ground. This interaction of the feet with the ground can be described by either a compliant or rigid model. As is common practice in designing feedback controllers for legged robots, we will consider this interaction to be non-compliant. Under this assumption, the ground contact is modeled as a *holonomic constraint*, meaning that the foot is rigidly attached to the ground through constraint forces. These forces are typically known as the *Ground Reaction Forces* (GRFs). The dynamical equations describing the motion of each phase are obtained as

$$D\ddot{q} + C\dot{q} + G = B\tau + J_c^T F_c + J_{st}^T F_{st}, \tag{2.1}$$

where $q \in \mathbb{R}^n$ denotes the generalized coordinates and $\tau \in \mathbb{R}^m$ of the robot with n *degrees*

of freedom and m *degrees of actuation*. The variable $D \in \mathbb{R}^{n \times n}$ denotes the inertia matrix, $C \in \mathbb{R}^{n \times n}$ denotes the Coriolis matrix, $G \in \mathbb{R}^n$ denotes the generalized gravity vector and $B \in \mathbb{R}^{n \times m}$ denotes the actuation distribution matrix. The term $F_{st} \in \mathbb{R}^{n_{st}}$ denotes the GRFs acting on the robot's contacting feet and $J_{st} \in \mathbb{R}^{n_{st} \times n}$ denotes the Jacobian of the associated contact frame. Additionally, robots can have constraints between their joints, such as through *five-bar* linkages, or have compliant elements between joints, such as in *Series Elastic Actuators* (SEAs). These can be captured by additional forces $F_c \in \mathbb{R}^{n_c}$ acting on the robot, with $J_c \in \mathbb{R}^{n_c \times n}$ denoting the associated Jacobian. The general form of the equation in (2.1) are typically known as the *manipulator equations*. The configuration variables q typically include the pose ($p_b \in \mathbb{R}^3, R_b \in SO(3)$) of the robot, the actuated joints $q_a \in \mathbb{R}^m$ as well as unactuated joints.

Constrained Dynamics

The dynamical equations in (2.1) represent the *unconstrained* equations of motion of the robot. The terms F_{st} and F_c denoting the holonomic constraint forces can be eliminated by finding their closed-form expressions. In this section, we illustrate how to obtain these closed-form expressions. For simplicity, we will drop the term $J_c^T F_c$ in the discussion to follow and consider specific cases in later chapters to obtain F_c in later chapters. Consider the equation of the form

$$D\ddot{q} + C\dot{q} + G = B\tau + J_{st}^T F_{st}. \tag{2.2}$$

Next, we can obtain the expressions for holonomic constraint $h_{st} \in \mathbb{R}^{n_{st}}$ describing the pose of the contacting foot with the ground, which can be obtained through the forward kinematics of the robot. The equation describing the holonomic constrained is obtained as

$$h_{st}(q) = 0. \tag{2.3}$$

We can then find the first and second-time derivatives of (2.3), and we obtain the following,

$$\dot{h}_{st} = J_{st}\dot{q} = 0, \tag{2.4}$$
$$\ddot{h}_{st} = J_{st}\ddot{q} + \dot{J}\dot{q} = 0. \tag{2.5}$$

We can now solve for $\ddot{q}$ and F_{st} from (2.2) and (2.5),

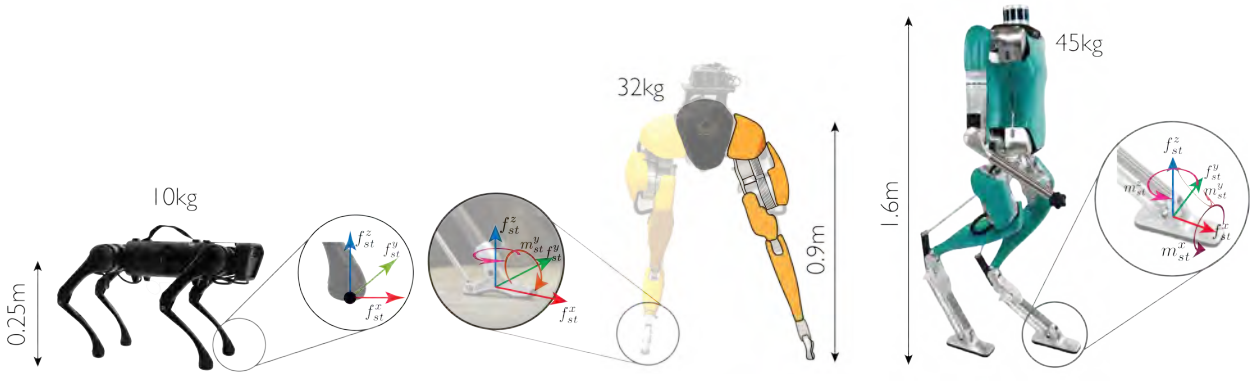


Figure 2.1: Various robots considered in this dissertation, along with their model scales and associated contact models depicting the contact wrench. (Left) `Unitree A1` quadruped robot with a point contact foot, (Center) `Cassie` biped with a line contact foot, and (Right) `Digit` Humanoid robot with a planar contact foot.

$$\underbrace{\begin{bmatrix} D & -J_{st}^T \\ J_{st} & \mathbf{0}_{n_c \times n_c} \end{bmatrix}}_{(n_{st}+n)\times(n_{st}+n)} \begin{bmatrix} \ddot{q} \\ F_{st} \end{bmatrix} = \begin{bmatrix} -C\dot{q} - G + B\tau \\ -\dot{J}_{st}\dot{q} \end{bmatrix}. \tag{2.6}$$

The equation in (2.6) represents a set of $(n_{st}+n)$ simultaneous linear equations in $(\ddot{q}, F_{st})$ and $(n_{st}+n)$ unknowns. From (2.6), a closed form expression for F_{st} can be obtained as

$$F_{st} = -\left(J_{st} D J_{st}^T\right)^{\dagger} \left(J_{st} D^{-1} \tau + \dot{J}_{st}\dot{q}\right). \tag{2.7}$$

Remark 2.1. *Having obtained a closed form expression for F_{st}, we can now substitute this in (2.2) to obtain the* constrained *dynamical equations. Additionally, note that the solution for F_{st} is affine w.r.t. the inputs τ. Substituting the expression for F_{st} in (2.2) maintains the affine structure of the dynamics with respect to the inputs τ.*

Defining the state of the robot to be $x := (q, \dot{q})$, the dynamics of the continuous phase can be obtained as

$$\dot{x} = f(x) + g(x)\tau, \tag{2.8}$$

where the vector fields f and g are obtained from (2.1) and (2.7).

Contact Types

Continuing our development of the dynamical model of legged robots, in this section, we introduce the specific forms for the contact wrenches F_{st} depending on the design of the robot's feet. Below, we describe the most common contact types in legged robots.

1. **Point Contact:** For quadruped robots like the `Unitree A1` and bipedal robots like `RABBIT`, the interaction between the ground and the foot can be modeled as a point contact. In this case, the contact wrench $F_{st} \in \mathbb{R}^3$ consists of only linear forces as illustrated in Figure 2.1,

$$F_{st} = \begin{bmatrix} f_{st,x} \\ f_{st,y} \\ f_{st,z} \end{bmatrix}. \tag{2.9}$$

2. **Line Contact:** For robots with narrow feet like that of the biped `Cassie`, the robot is unactuated about the length of its feet. In this case, the contact wrench is five-dimensional and defined as,

$$F_{st} = \begin{bmatrix} f_{st,x} \\ f_{st,y} \\ f_{st,z} \\ m_{st,y} \\ m_{st,z} \end{bmatrix}, \tag{2.10}$$

   where $m_{st,y}$ is the moment in the pitch direction and $m_{st,z}$ is the moment in the yaw direction as depicted in Figure 2.1.

3. **Planar Contact**: Humanoid robots like `Digit`, on the other hand, have a significantly larger footprint, and the contact between the ground and the feet are typically modeled as a *planar* surface contact. In this case, the contact wrench is illustrated in Figure 2.1 and defined as,

$$F_{st} = \begin{bmatrix} f_{st,x} \\ f_{st,y} \\ f_{st,z} \\ m_{st,x} \\ m_{st,y} \\ m_{st,z} \end{bmatrix}. \tag{2.11}$$

In the next section, we describe additional constraints on these wrenches.

Contact Constraints

Note that we have not placed any additional constraints on the forces F_{st} yet. Due to physical constraints such as friction, the wrench produced by the ground is limited. Additional constraints imposed on the desired locomotion behavior (such as *flat-footed walking*), will further limit the ground reaction wrench. In this section, we briefly describe the constraints imposed on different contact types. These will be used in later chapters of this dissertation to obtain dynamically feasible trajectories and develop optimization-based locomotion controllers.

1. **Point Contact**: For a point contact, the contact wrench is limited by friction constraints, and due to the fact that the ground cannot pull the robot (typically known as the *unilateral* vertical ground reaction force constraint). These are expressed as the following inequalities,

$$\sqrt{{f_{st}^z}^2 + {f_{st}^y}^2} < \mu f_{st}^z, \tag{2.12}$$

$$f_{st}^z \geq 0. \tag{2.13}$$

where the inequality in (2.12) is based on the *Amontons–Coulomb* model and μ denotes the coefficient of static friction. Since the inequality in 2.12 is nonlinear, a more conservative but linear friction model is typically used for feedback control,

$$-\frac{\mu}{\sqrt{2}} f_{st}^z < f_{st}^x < \frac{\mu}{\sqrt{2}} f_{st}^z, \tag{2.14}$$

$$-\frac{\mu}{\sqrt{2}} f_{st}^z < f_{st}^y < \frac{\mu}{\sqrt{2}} f_{st}^z. \tag{2.15}$$

The constraints in (2.13), (2.14) and (2.15) can be written in matrix form,

$$\underbrace{\begin{bmatrix} -1 & 0 & -\frac{\mu}{\sqrt{2}} \\ 1 & 0 & -\frac{\mu}{\sqrt{2}} \\ 0 & -1 & -\frac{\mu}{\sqrt{2}} \\ 0 & 1 & -\frac{\mu}{\sqrt{2}} \\ 0 & 0 & -1 \end{bmatrix}}_{A_{st}^{point}} F_{st} \leq 0. \tag{2.16}$$

We define the set $\mathcal{K}_{st}^{point}$

$$\mathcal{K}_{st}^{point} := \{F_{st} \in \mathbb{R}^3 \mid A_{st}^{point} F_{st} \leq 0\}. \tag{2.17}$$

to be the set of contact wrenches that satisfies the inequalities in (2.16).

2. **Line Contact**: In addition to the friction constraints in (2.14) and (2.15), we place additional constraints to avoid rotation of the foot about the heel and the toe. These are described by the following inequalities,

$$-\frac{l_f}{2} f_{st}^z < m_{st}^y < \frac{l_f}{2} f_{st}^z, , \tag{2.18}$$

where l_f is the length of the contacting foot. These are known as the *Zero Moment Point (ZMP)* constraints. Additionally, we place friction constraints on the yaw moment,

$$-\gamma f_{st}^z < m_{st}^z < \gamma f_{st}^z, , \tag{2.19}$$

where γ is the coefficient of torsional friction. Combining the inequalities in (2.14), (2.15), (2.18) and (2.19), we obtain the following linear inequality constraint in F_{st},

$$\underbrace{\begin{bmatrix} -1 & 0 & -\frac{\mu}{\sqrt{2}} & 0 & 0 \\ 1 & 0 & -\frac{\mu}{\sqrt{2}} & 0 & 0 \\ 0 & -1 & -\frac{\mu}{\sqrt{2}} & 0 & 0 \\ 0 & 1 & -\frac{\mu}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -\frac{l_f}{2} & -1 & 0 \\ 0 & 0 & -\frac{l_f}{2} & 1 & 0 \\ 0 & 0 & -\gamma & 0 & -1 \\ 0 & 0 & -\gamma & 0 & 1 \end{bmatrix}}_{A_{st}^{line}} F_{st} \leq 0. \tag{2.20}$$

Similar to (2.17), we define the set $\mathcal{K}_{st}^{line}$ to be

$$\mathcal{K}_{st}^{line} := \{F_{st} \in \mathbb{R}^5 \mid A_{st}^{line} F_{st} \leq 0\}. \tag{2.21}$$

3. **Planar Contact:** The planar contact model includes an additional constraint on the roll contact moment, similar to the ZMP constraints introduced for the line contact,

$$-\frac{w_f}{2} f_{st}^z < m_{st}^x < \frac{w_f}{2} f_{st}^z, , \tag{2.22}$$

where w_f is the width of the foot. We define the set $\mathcal{K}_{st}^{planar}$ to be

$$\mathcal{K}_{st}^{line} := \{F_{st} \in \mathbb{R}^6 \mid A_{st}^{planar} F_{st} \leq 0\}, \tag{2.23}$$

where A_{st}^{planar} is defined as

$$\underbrace{\begin{bmatrix} -1 & 0 & -\frac{\mu}{\sqrt{2}} & 0 & 0 & 0 \\ 1 & 0 & -\frac{\mu}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & -1 & -\frac{\mu}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 1 & -\frac{\mu}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{l_f}{2} & 0 & -1 & 0 \\ 0 & 0 & -\frac{l_f}{2} & 0 & 1 & 0 \\ 0 & 0 & -\frac{w_f}{2} & -1 & 0 & 0 \\ 0 & 0 & -\frac{w_f}{2} & 1 & 0 & 0 \\ 0 & 0 & -\gamma & 0 & 0 & -1 \\ 0 & 0 & -\gamma & 0 & 0 & 1 \end{bmatrix}}_{A_{st}^{planar}} F_{st} \leq 0. \tag{2.24}$$

Impact Model

The equations of motion in (2.1) describe the dynamics of the continuous phases of locomotion, such as when the robot is in single-support or flight phases. Legged robots continuously make and break contact with the environment, such as when the swing leg of the robot makes contact with the ground. The set of states (and inputs) where impact occurs is known as the *switching surface* $\mathcal{S}$. Such interactions can be modeled as discrete and instantaneous plastic impact events (zero coefficient of restitution). Under such a model, the generalized velocities undergo a discrete change while the configuration remains unchanged. At impact, the generalized post-impact velocities $\dot{q}^-$ can be obtained using momentum conservation laws. An additional assumption on the post-impact velocity of the impacting foot to remain zero is utilized to obtain the following system of linear equations,

$$\begin{bmatrix} D(q) & -J_{st}^T(q) \\ J_{st}(q) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}^+ \\ F_{st} \end{bmatrix} = \begin{bmatrix} D(q)\dot{q}^- \\ \mathbf{0} \end{bmatrix}, \tag{2.25}$$

where F_{st} denotes the impact forces and $\dot{q}^-$ denotes the pre-impact velocities. We can obtain the expression for the post-impact velocities $\dot{q}^+$ as,

$$\dot{q}^+ = \Delta(q, \dot{q}^-), \left(q, \dot{q}^-\right) \in \mathcal{S}. \tag{2.26}$$

Hybrid Model

We can now put together the continuous-time dynamics in (2.6) and the discrete-impact dynamics (5.2) to obtain a hybrid dynamical model of locomotion,

$$\Sigma : \begin{cases} \dot{x} & = f(x) + g(x)u, \ (x, \tau) \notin \mathcal{S} \\ x^+ & = \Delta\left(x^-\right), \ (x, \tau) \in \mathcal{S} \end{cases} \tag{2.27}$$

Gait Generation

Having presented the hybrid dynamical model of legged locomotion, we now illustrate how this model can be used to generate periodic gaits for bipedal and quadrupedal locomotion. We will use these trajectories throughout Part I of this dissertation to develop model-based locomotion controllers. Specifically, we will obtain periodic solutions, also known as periodic gaits, for the hybrid dynamical system in (2.27). For the system of the form $\dot{x} = f(x, u)$, the problem of finding an optimal trajectory can be stated as

$$\begin{aligned} J(x(t), u(t)) = \min_{x(t), \tau(t)} \quad & \int_0^T L(x(t), \tau(t)) dt \\ & \tau(t)) dt \\ & c(x(t), \tau(t)) \leq 0, \quad 0 \leq t \leq T, \end{aligned} \tag{2.28}$$

where $L(\cdot)$ represent the running cost function, and $c(\cdot)$ represents the path constraints.

There are several approaches to numerically solve the trajectory optimization problem in (2.28). In this dissertation, we utilize the *direct collocation* method [19], which begins by transcribing the continuous-time trajectory optimization problem in (2.28) to a problem in discrete-time. This allows us to convert an infinite-dimensional problem, where the decision variables $x^*(t)$ and $\tau(t)^*$ are functions, to that of a finite dimensional constrained parameter optimization problem. Specifically, the time interval $t \in [0, T]$ is divided into a fixed number of uniformly distributed intervals, and the trajectory optimization problem turns to finding the optimal state and inputs at each of the discretization points.

In particular, the even-numbered nodes (e.g., t_0, t_2, ..., t_N) are called cardinal nodes, and the odd-numbered nodes between every two cardinal nodes are called interior nodes. At each discrete node of $t = t_i$, an approximation of state variables $x_i = x(t_i)$ and control inputs $\tau_i = \tau(t_i)$ is introduced as a set of optimization variables to be solved. The approximation of the slope of state variables $\dot{x}_i = \dot{x}(t_i)$ is also introduced as *defect variables* in the optimization.

The Hermite-Simpson methods then use piecewise continuous cubic interpolation polynomials to approximate the solution of the system over each interval between two neighboring cardinal nodes. This approximation can be fully determined by the approximated state variables and slopes at the cardinal nodes. Hence, if the approximated states x_i and slopes $\dot{x}_i$ at the interior nodes match the interpolation polynomial at time $t = t_i$, then the resulting piecewise polynomials are considered as an approximated solution of the system [58]. To find this approximated solution, e.g., the discrete representation of the states, the original

continuous time trajectory optimization problem can be converted to the following form given by

$$
\begin{aligned}
J(x_i, u_i) = \min_{\tau_i} \quad & \sum_{i=1}^{N-1} w_i L(x_i, \tau_i) \\
\text{st.} \quad & \dot{x}_i = f(x_i, \tau_i) \\
& c(x_i, \tau_i) \leq 0, \quad 0 \leq i \leq N \\
& \dot{x}_i - 3(x_{i+1} - x_{i-1})/2\Delta t_i + (\dot{x}_{i-1} + \dot{x}_{i+1})/4 = 0 \\
& x_i - (x_{i+1} + x_{i-1})/2 - \Delta t_i(\dot{x}_{i-1} - \dot{x}_{i+1})/8 = 0,
\end{aligned} \tag{2.29}
$$

where for all $i \in \{1, 3, \cdots, N-1\}$, where $\Delta t_i = t_{i+1} - t_{i-1}$ is the time interval between two cardinal nodes, and w_i is weighting factor of each node determined by the Gaussian quadrature [63]. Specifically, the last two constraints are called *collocation constraints*, which are determined by cubic interpolation polynomials. The above nonlinear programming problem can be solved by existing numerical NLP solvers such as `IPOPT`.

Common Constraints in Legged Locomotion

The path constraints $c(x, \tau)$ introduced in (2.28) encode several physical and user-defined constraints. Below, we highlight some typical constraints.

1. *Defect Constraints*: These are constraints on the estimated states from the optimizer and states obtained from the interpolation polynomial. A complete description of defect constraints can be found in [62].
2. *System dynamics*: The system dynamics, (2.1) are imposed as equality constraints.
3. *Periodicity of an orbit*: To obtain periodic gaits, the post impact states at the last node at the current domain are enforced to be equal to the first node of the next domain.
4. *Domain of admissibility*: The holonomic constraints in (2.5) and the constraints in (2.17), (2.21) or (2.23) depending on the contact type, must be satisfied to ensure holonomic and contact constraints are met.
5. *Gait Time*: The duration of the trajectory T is a decision variable that can be optimized for. (Alternately, the time step Δt_i can also be considered as a decision variable).
6. *Speed, Step Length and Foot clearance*: The desired average speed, step length, and vertical foot clearance with the ground can be specified.

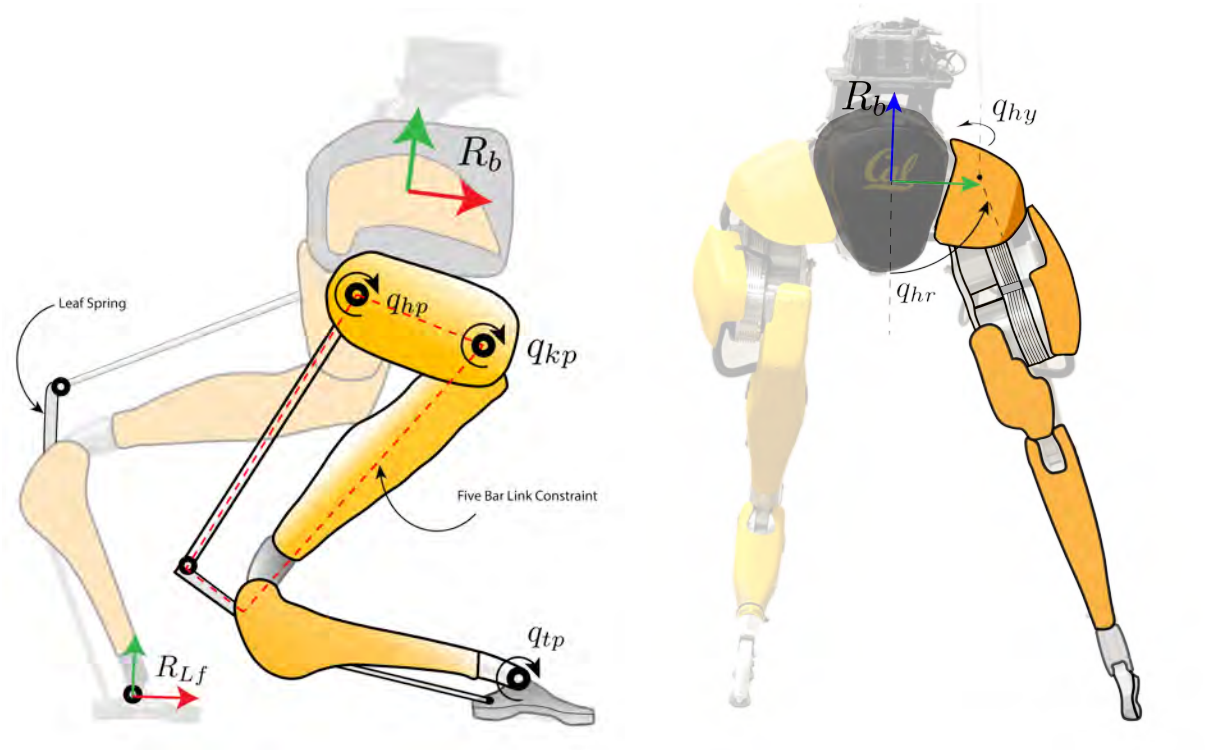


Figure 2.2: Figure illustrating the configuration of the Cassie bipedal robot.

2.2 Robot Description

Having presented the dynamics of a legged robot and a method to obtain gaits for locomotion, in this section, we briefly describe mathematical models of the various robots (Figure 2.1) considered in this dissertation.

Cassie Bipedal Robot

Cassie is a high-dimensional underactuated bipedal robot with an average height of 0.9m and weighing around 32kg. The configuration of the robot can be described by the variable $q = [p, R_b, q_L, q_R]$ with $p \in \mathbb{R}^3$ denoting the position of the base, $R_b \in SO(3)$ denoting the orientation, and $q_i \in \mathbb{R}^7, i \in \{L, R\}$ denotes the configuration variables of the left and right legs respectively. Specifically, each leg comprises of five actuated joints $q_{a,i} = [q_{hy,i}, q_{hr,i}, q_{hp,i}, q_{kp,i}, q_{tp,i}]$ denoting the hip yaw, hip roll, hip pitch, knee pitch and toe pitch, and two unactuated spring joints, $q_{s,i} = [q_{ks,i}, q_{as,i}]$, denoting the knee and ankle springs respectively. As described in Chapter 2, the dynamics of the robot can be expressed by the standard *manipulator equations* (2.1). Specifically for Cassie, these equations take the following form,

$$D\ddot{q} + C\dot{q} + G = B\tau + J_s^T \tau_s + J_{st}^T F_{st}, \tag{2.30}$$

where $D \in \mathbb{R}^{20\times 20}$ denotes the inertia matrix, $C \in \mathbb{R}^{20\times 20}$ denotes the Coriolis matrix, $G \in \mathbb{R}^{20}$ denotes the generalized gravity vector, $B \in \mathbb{R}^{20\times 10}$ is the actuation distribution matrix, $\tau_s \in \mathbb{R}^4$ are the spring torques, J_s is the Jacobian of the spring deflections, $\tau \in \mathbb{R}^{10}$ are the input joint torques, $F_{st} \in \mathbb{R}^{10}$ are the contact wrenches and $J_{st} \coloneqq \left[J_{st,1}^T, J_{st,2}^T\right]^T \in \mathbb{R}^{10\times 20}$ is the Jacobian of the contact poses. Specifically, we model the interaction between each foot and the ground as a line-contact, as described in Section 2.1.

In this chapter, we assume that the springs cannot deflect. This can be modeled as a holonomic constraint on the configuration variables of the robot.

Digit Humanoid Robot

Digit is a humanoid robot with 30 degrees of freedom (DoF) and 20 actuated joints as listed in (2.31)

$$\begin{aligned} q \;= [&q_x,\; q_y,\; q_z,\; q_{\text{yaw}},\; q_{\text{pitch}},\; q_{\text{roll}},\\ &q_{1L},\; q_{2L},\; \ldots,\; q_{12L},\; q_{1R},\; q_{2R},\; \ldots,\; q_{12R}]^T, \end{aligned} \tag{2.31}$$

where $(q_x, q_y, q_z, q_{\text{yaw}}, q_{\text{pitch}}, q_{\text{roll}})$ represent the floating base coordinate, $(q_1, q_2, ..., q_8)$ and $(q_9, q_{10}, q_{11}, q_{12})$ denote leg and arm joints, respectively. The corresponding DoFs are defined in (2.32)

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \begin{bmatrix} \text{hip roll} \\ \text{hip yaw} \\ \text{hip pitch} \\ \text{knee} \\ \text{shin} \\ \text{tarsus} \\ \text{toe pitch} \\ \text{toe roll} \end{bmatrix}, \begin{bmatrix} q_9 \\ q_{10} \\ q_{11} \\ q_{12} \end{bmatrix} = \begin{bmatrix} \text{shoulder roll} \\ \text{shoulder pitch} \\ \text{shoulder yaw} \\ \text{elbow} \end{bmatrix}. \tag{2.32}$$

The generalized coordinates of Digit are illustrated in Fig. 2.3, where Digit's arm is fully actuated with four DoFs: $q_9, q_{10}, q_{11}, q_{12}$, and Digit's leg has 8 DoFs with 6 of them being actuated: $q_1, q_2, q_3, q_4, q_7, q_8$. Note that q_5 and q_6 are passive and connected via leaf springs. In addition, q_7 and q_8 are actuated via rods by two motors attached at q_6.

A1 Quadruped Robot

The `Unitree A1` is a 10kg quadruped with 3 motors in each leg, with a total of 12 actuated joints and 6 underactuated base degrees of freedom (DoF). The configuration of the robot

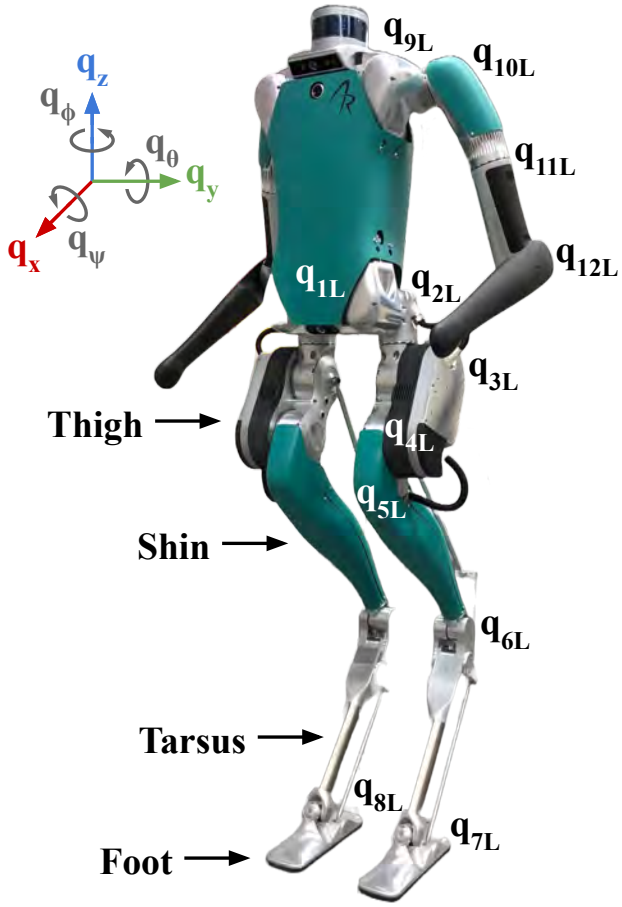


Figure 2.3: Digit humanoid robot model. Digit has 30 DoFs and 20 actuated joints, where each arm has 4 DoFs and each leg has 8 DoFs with 6 of them being actuated.

is represented by $q = \left[p^T, \Theta^T, q_{FR}^T, q_{FL}^T, q_{RR}^T, q_{RL}^T\right]^T \in \mathcal{Q} \subset \mathbb{R}^{18}$, where $p \in \mathbb{R}^3$ denotes the Cartesian position of the robot, $\Theta \in \mathbb{R}^3$ denotes the ZYX Euler angle representation of the orientation of the body, and $q_i \in \mathbb{R}^3,\ i \in \{FR, FL, RR, RL\}$ denotes the actuated joints of the front/rear right/left legs. The actuated joints include hip abduction, hip and knee pitch DoFs.

2.3 Chapter Summary

In this chapter, we presented a hybrid dynamical model for legged locomotion, including various contact types found in legged robots and physical constraints imposed on the contact wrench. Additionally, we presented a method to numerically obtain trajectories for the nonlinear hybrid locomotion model. Finally, in Section 2.2, we presented the models for

specific robots considered in this dissertation. We will use these models and trajectory generation methods throughout the dissertation to develop model-based feedback controllers. In the next chapter, we develop a planning and control method for a 2D biped for running with precise foot placement.

Part I

Model-Based Locomotion Control

Chapter 3

Bipedal Robotic Running on Stochastic Discrete Terrain

To navigate across discrete terrain with large displacements in the stepping locations, bipedal robots have to be able to perform agile and dynamic maneuvers such as jumping or running while also satisfying strict constraints on foot placement and ground contact forces. In this chapter, we analyze the problem of bipedal running over stochastically varying discrete terrain with large changes in step lengths. Specifically, our method is based on designing a library of running gaits that are two-step-periodic. We illustrate the capabilities of the proposed controller through numerical simulations of a five-link underactuated robot RABBIT, running over discrete terrain with step lengths that vary between 0.6m and 1.2m. This is about 1.5 times the robot's leg lengths and twice the step length that could have been achieved by walking.

3.1 Introduction

Legged robots have the promise of being able to serve in applications such as in space and urban exploration, as personal robots in homes and in search and rescue operations. It is their inherent morphology and mechanical structure that renders them potentially superior candidates over their wheeled counterparts. A key task in such applications is the ability to locomote over discrete footholds such as in unstructured environments like wooded paths or over a flight of stairs in indoor environments. This, however, introduces several challenges and constraints, including (a) Strict constraints on foot placement, (b) Friction constraints, and (c) Input constraints. Violation of any of these constraints will render the system unstable.

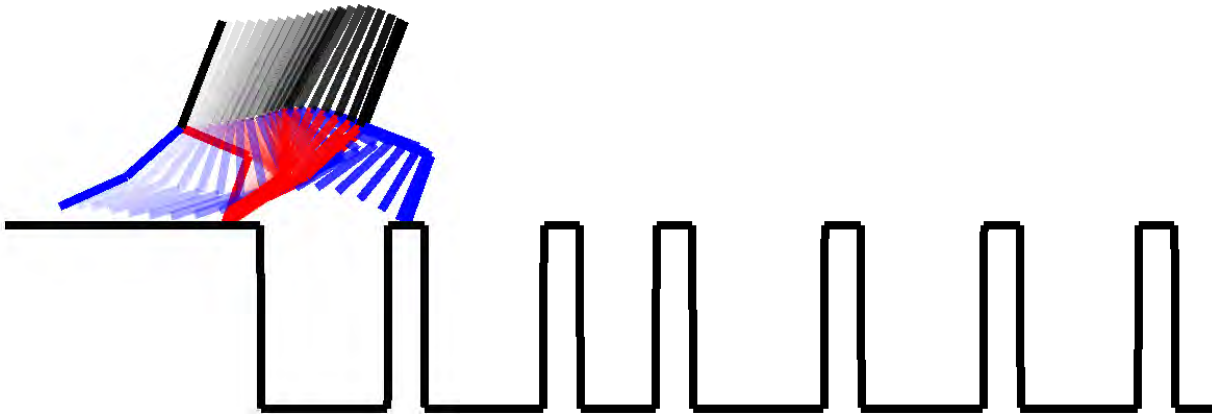


Figure 3.1: Illustration of the problem of the stepping stone. The goal of the feedback control design is for the bipedal robot to traverse over a set of discrete terrain with wide gaps. Such terrain can only be traversed by performing agile maneuvers like running or jumping.

Related Work

The problem of robotic legged bipedal walking over discrete terrain has been studied in the past, with a wide variety of techniques being used. Early methods for footstep planning, such as in [76], relied on simple models like the 3D inverted pendulum and cart-table models to generate walking patterns to walk over randomly generated stepping stones. In [113], the authors present a method based on the concept of capture points to control a bipedal robot to walk over discrete steps. The controller regulates the center-of-pressure on the stance foot so as to 'guide' the capture point to the desired stepping location. More recently, in [139], the authors present a centroidal momentum-based controller for a high-dimensional robot ATLAS, to walk over partial footholds, including line and point contact surfaces. In [36], a mixed integer quadratically constrained quadratic program is presented for footstep planning of a humanoid robot to walk on uneven terrain with obstacles. In [101], the authors present a method based on Control Barrier Functions (CBFs) to design feedback controllers for high-dimensional 3D robots to walk over stochastically generated discrete steps with changing step heights and step lengths. In [96], the authors propose a method that combines a one-step-periodic gait library approach with a CBF-based feedback controller that significantly improved the performance as in [100]. In [50], the authors propose a bio-inspired controller based on Central Pattern Generators (CPGs) to achieve step length and step height modulations over a wide range for bipedal walking.

With regards to bipedal robotic running, while numerous studies have been carried out, there have been limited studies on agile locomotion, like running, of legged systems over discrete terrain. One of the earliest works on running over rough terrain was by Hodgins and Raibert [65]. The authors presented three intuitive control designs for regulating the step length for a bipedal robot that could run over rough terrain, including stairs with changing step heights. In [107], the authors propose a reinforcement learning-based approach to control a physics-based legged character to navigate over terrain with wide gaps, steps, and obstacles. The authors are able to translate their method to a wide variety of high-dimensional characters, including a 21-link planar dog and a 7-link planar biped. In [38], the authors propose an intuitive dead-beat control strategy based on a point-mass model for running over 3D stepping stones. The authors perform numerical simulations on a point-mass model with massless legs for running over 3D stepping stones. In [57], the authors present a neuromuscular controller for bipedal running.

In our previous work [105, 103], we presented a method to design a feedback controller for an underactuated legged robot to walk over discrete terrain with stochastically varying step heights and step lengths. The method relied on pre-computing a library of a small number of walking gaits (through a nonlinear offline program) that were two-step-periodic and parametrized by the step lengths and step heights in the first and second steps. The controller then performed a bilinear interpolation between the different gaits based on the current and desired step lengths/heights during run-time. By switching among a set of two-step-periodic gaits, the controller was able to achieve aperiodic walking with precise footstep placement over randomly generated discrete terrain. The control method was successfully implemented on an underactuated bipedal robot, ATRIAS, over a wide range of complex terrain. In our most recent work [7], the two-step periodic gaits were also extended to quadrupedal robots.

Problem Statement and Approach

In this chapter, we study the problem of bipedal robotic running over randomly varying discrete terrain with large changes in step lengths (Figure 3.1). In particular, we develop a model-based feedback controller for an underactuated legged system that does not have knowledge of the entire terrain ahead of time but only the position of the next stepping location is known. By only using a one-step preview of the upcoming stepping location, the method renders itself capable of being combined with vision sensors (like cameras and LiDAR) to estimate the stepping locations.

Motivated by the success of the 'two-step-periodic' gait library approach for bipedal walking, we extend this approach to the case of bipedal running. A primary advantage of the proposed method is that, unlike most other methods that rely on simplifications of the system dynamics, it considers the full nonlinear hybrid dynamics of the system, both during offline gait generation and during the control phase. Given the current state-of-the-art in trajectory optimization for nonlinear hybrid systems and computational power, another advantage of

our method is that it is easy to implement on a physical system. A key reason for the success of the 'two-step-periodic' gait approach in [103] was it allowed for smooth transitions between walking gaits at different step-lengths/heights, thereby inherently preventing violations in friction cone and unilateral ground reaction force constraints and by using only a small number of gaits in the gait library. The method, therefore, seems promising to use in richer locomotion behaviors such as running and jumping.

The problem of robotic running over discrete footholds, however, places some additional challenges, which stems from the loss of control authority of the angular momentum during flight phases (i.e. when the robot is completely in the air). Additional challenges also arise due to the increased number of possible hybrid modes of the system. These are potential reasons for why the applicability of the 'two-step-periodic' gait library approach might not be straightforward. In particular, we show that by reasoning about the dynamics and by the proper choice of output variables to be controlled, the 'two-step-periodic' gait library approach can be extended to the case of running as well.

Contributions

The key contributions of this chapter are as follows:

1. We extend the 'two-step-periodic' gait library approach, initially presented in [105, 103] to develop a control strategy for bipedal robotic running over discrete footholds that follows strict constraints on the states and control inputs of the system;
2. By doing so, we are able to expand the capabilities of the robot to traverse over wider gaps in the discrete footholds that was not possible with only walking;
3. We show that, by a proper selection of output variables to be controlled, we can significantly reduce the error in the foot placement locations while running.

Organization

The rest of the chapter is organized as follows: In Section 3.2, we present the hybrid model for running, followed by the trajectory optimization and control design method in Section 3.3. Finally, in Section 3.4, we present results from numerical simulations of a five-link underactuated bipedal robot.

3.2 Dynamical Model for Running

In this section, we present a brief overview of the dynamical model of a planar five-link two-legged robot for running behaviors. These models will be used for generating optimal trajectories as well as for control synthesis. The specific robot under consideration is RABBIT. More details about the dynamical model can be found in [25].

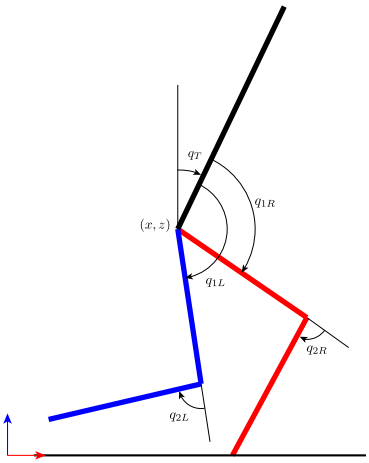


Figure 3.2: Generalized coordinates of the bipedal robot RABBIT.

Figure 3.2 illustrates a schematic diagram of the robot. The configuration variables, defined as $q := \begin{bmatrix} p^x_{hip} & p^z_{hip} & q_T & q_{1R} & q_{2R} & q_{1L} & q_{2L} \end{bmatrix}^T$ include the world frame position of the hip $\begin{bmatrix} p^x_{hip} & p^z_{hip} \end{bmatrix}^T$, world frame orientation of the torso q_T and the relative joint angles of the thigh q_1 and shin q_2 links. The subscripts L and R refer to the left and right links, respectively. For future reference, we define the actuated joints $q_a := \begin{bmatrix} q_{1R} & q_{2R} & q_{1L} & q_{2L} \end{bmatrix}$.

As noted in Chapter 2, the equations of motion have the form,

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B\tau + J^T_{st} F_{st}, \tag{3.1}$$

where $\tau \in \mathbb{R}^4$ are the control inputs that actuates each of the joints q_a, $F \in \mathbb{R}^2$ are the ground reaction forces at the stance foot and $J_{st} := \frac{\partial p_{st}}{\partial q} \in \mathbb{R}^{2\times 7}$ is the Jacobian of the stance foot position p_{st}. We note that during flight, since there are no external contact forces acting on the robot, $F_{st} \equiv 0$.

Like walking, the dynamics for running motions is also hybrid. Specifically, running comprises of alternating phases of single-support Σ_s and flight Σ_f phases as illustrated in Figure 3.3. The hybrid dynamics is written as

$$\begin{aligned} \Sigma_s : \begin{cases} \dot{x} & = f_s(x) + g_s(x)u, \ (x, \tau) \notin \mathcal{S}_{s\to f} \\ x^+ & = \Delta_{s\to f}(x^-), \ (x, \tau) \in \mathcal{S}_{s\to f} \end{cases} \\ \Sigma_f : \begin{cases} \dot{x} & = f_f(x) + g_f(x)\tau, \ x \notin \mathcal{S}_{f\to s} \\ x^+ & = \Delta_{f\to s}(x^-), \ x \in \mathcal{S}_{f\to s} \end{cases} \end{aligned} \tag{3.2}$$

Here $\mathcal{S}_{s\to f} := \{(x, u) \mid F^z(x, u) = 0\}$ is the switching surface corresponding to the transition between stance and flight domains and is defined as the set of states and control

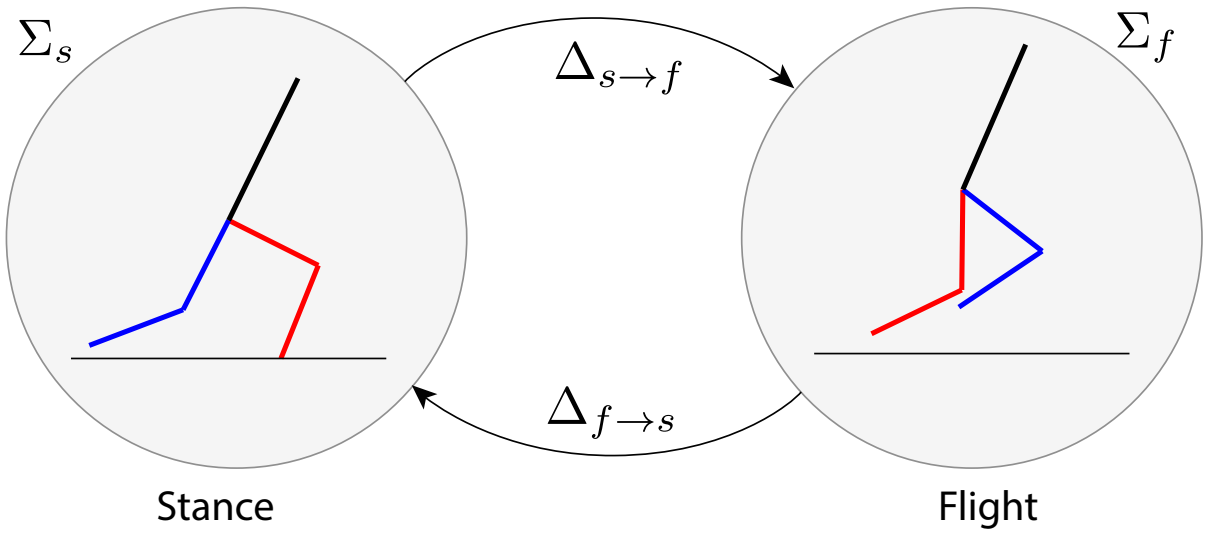


Figure 3.3: Illustration of the domains in running.

inputs such that the vertical ground reaction force $F^z(x,u)$ is zero (this corresponds to the case when the stance foot lifts off from the ground). Similarly, we define the switching surface $\mathcal{S}_{f\to s} := \{x \mid p^z_{sw}(x) = 0\}$ corresponding to the transition between flight to stance as the set of states such that the vertical component of the position of the swing foot is zero.

In addition, $\Delta_{s\to f} = \mathcal{I}$ is the identity operator and $\Delta_{f\to s}$ is obtained from rigid impact dynamics. The vector fields $f(x)$ and $g(x)$ are obtained for stance and flight phases using Lagrange's equations of motion (3.1).

In the next sections, we present our control approach based on the 'two-step-periodic' gait library.

3.3 Hybrid Zero Dynamics Based Control

In this section, we briefly present the Hybrid Zero Dynamics (HZD) framework [137, 138], which uses the dynamical model presented in Section 3.2 to generate periodic gaits and design feedback controllers for running. The HZD method begins by selecting a set of outputs y_a for the hybrid dynamical system as in (3.2). Driving these outputs to a set of desired quantities y_d defines how the various links of the robot move. The HZD controller then implements an Input-Output (IO) Linearizing controller to drive the outputs y_a to y_d.

Output Selection

In this section, we present our choice of outputs y_a (also known as virtual constraints). We note that several valid choices for the outputs exist. As in [105], a candidate for y_a is the set of actuated joint angles q_a. However, since we are interested in achieving precise step lengths, which is achieved through the flight phase, we choose the horizontal velocity of the center of mass as one of the outputs during the stance phase. This choice of output is motivated by the fact that the horizontal distance achieved during flight is dependent on the exit velocity of the stance phase (velocity of the center of mass at the instant before entering the flight phase). We also note that this is a *relative degree 1 output* [67] [11]. The complete set of outputs during the stance phase y_a^s and flight phase y_a^f is chosen as

$$y_a^s = \begin{bmatrix} v_{com}^x \\ q_1^{st} \\ q_1^{sw} \\ q_2^{sw} \end{bmatrix}, \tag{3.3}$$

$$y_a^f = \begin{bmatrix} q_T \\ q_2^{st} \\ q_1^{sw} \\ q_2^{sw} \end{bmatrix}. \tag{3.4}$$

The superscripts st and sw in y^s denote stance and swing legs respectively, and $st = L/R$ (and $sw = R/L$), depending on whether the Left or Right Leg is in stance respectively. In y^f, the superscripts st and sw denote the stance and swing legs in the preceding stance phase.

The desired outputs $y_d := y_d(\tau^p, \alpha^p)$ are parametrized by Bézier splines, where α^p are the coefficients of the Bézier spline and τ^p is a phase variable that monotonically increases from 0 to 1 and $p \in \{s, f\}$. Specifically, we will find the Bézier parameters α^p and phase variable parameters through a Nonlinear Program such that enforcing the outputs y_a to the desired outputs $y_d(\tau^p, \alpha^p)$ through a feedback controller will result in a *two-step-periodic* solution for the hybrid model in (3.2). This is schematically illustrated in Figure 3.4. We note that, there are several choices for the phase variable τ^p. In particular, we choose the normalized absolute stance leg-angle q_{LA}^{st} as the phase variable during stance and normalized time during flight,

$$\tau^s := \frac{q_{LA}^{st} - q_{LA,max}^{st}}{q_{LA,max}^{st} - q_{LA,max}^{st}}, \tag{3.5}$$

$$\tau^f := \frac{t - t_{min}}{t_{max} - t_{min}}, \tag{3.6}$$

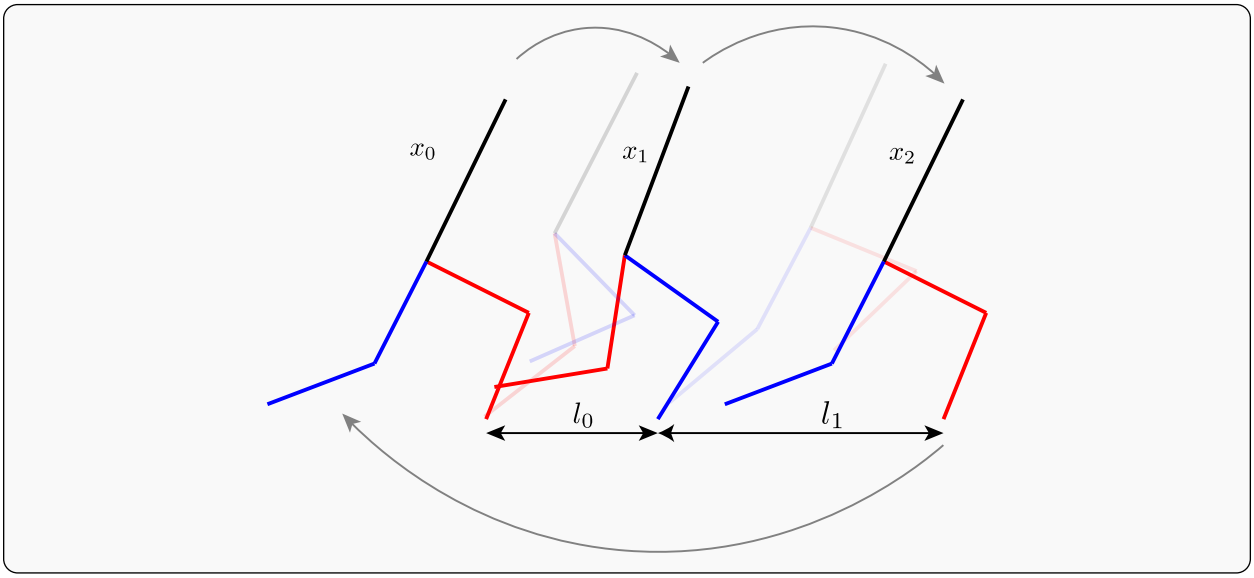


Figure 3.4: Illustration of two-step-periodic gait design. We optimize over two running steps with constraints on the step lengths l_0 and l_1 in the first and second running steps, respectively. The periodicity constraint enforces the states of the robot at the end of the second step x_2 to return to the states at the beginning of the first step x_0.

with q^{st}_{LA} defined as

$$q^{st}_{LA} := q_T + q^{st}_1 + \frac{q^{st}_2}{2} - \frac{\pi}{2}; \tag{3.7}$$

$q^{st}_{LA,max}, q^{st}_{LA,min}, t_{min}, t_{max}$ are constants to be determined. For future reference, we collect the constant parameters used in the gait phase variable definitions above into the following vectors,

$$\theta^s := \begin{bmatrix} q^{st}_{LA,max} \\ q^{st}_{LA,min} \end{bmatrix}, \tag{3.8}$$

$$\theta^f := \begin{bmatrix} t_{max} \\ t_{min} \end{bmatrix}. \tag{3.9}$$

Two-Step-Periodic Gait Design

Having presented the hybrid dynamical model for running and the choice of outputs y_a to be controlled, we now present a method to find the parameters α^s, α^f, θ^s and θ^f, such that the resulting gaits are two-step-periodic.

Motor Torque	$\lvert\tau\rvert \leq 10\,\mathrm{N\,m}$
Friction Cone	$\left\lvert\frac{F_{st}^x}{F_{st}^z}\right\rvert \leq 0.6$
Vertical Ground Reaction Force during stance	$F_{st}^v \geq 0\,\mathrm{N}$
Swing Foot Clearance during stance	$h_f\rvert \geq 0.05\,\mathrm{m}$

Table 3.1: Optimization constraints

The two-step-periodic gait design involves obtaining gait parameters such that the post-impact states of the system after two running steps return to the initial states at the start of the first step. The gaits are parametrized by the step lengths in the first and second running step l_0 and l_1, respectively and we define the set of parameters as

$$\mathcal{P}\left(l_0, l_1\right) := \{\alpha^s\left(l_0, l_1\right) \alpha^f\left(l_0, l_1\right), \theta^s\left(l_0, l_1\right), \theta^f\left(l_0, l_1\right)\}. \tag{3.10}$$

Subsequently, we find parameters $\mathcal{P}\left(l_0, l_1\right)$ for $(l_0, l_1) \in L \times L$ to build a library of gaits,

$$\mathcal{G} := \{\mathcal{P}\left(l_0, l_1\right) \mid (l_0, l_1) \in L \times L\}, \tag{3.11}$$

where L is a predefined set of step lengths. Specifically, we choose $L = \{0.6, 0.8, 1.0, 1.2\}$, with a total of 16 gaits in the gait library.

As described in Chapter 2, the problem of obtaining the gait library $\mathcal{G}$ is cast as a nonlinear program. In particular, the objective function considered here is the integral of squared torques over step length:

$$J = \int_0^T ||\tau(t)||_2^2 \, dt. \tag{3.12}$$

and constraints for the optimization are formulated as in Table 3.1.

In addition to the above constraints, we also need to guarantee the periodicity of the gait through the *periodicty constraints*:

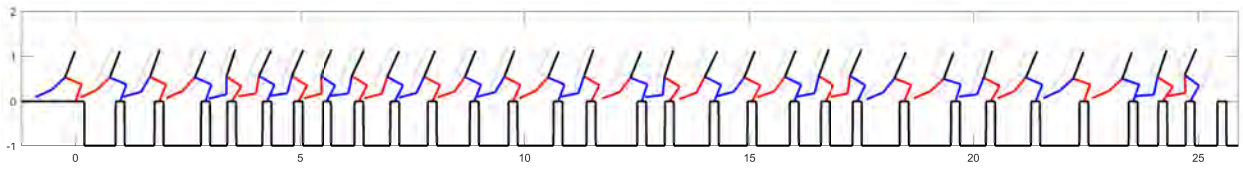


Figure 3.5: Snapshots of the robot running over the discrete footholds using the control method presented in this chapter.

1. The initial state at the start of the first stance phase is given by $x = x_0^+$with corresponding (initial) step length l_0.
2. *Transition constraints between stance and flight:* The state at the end of the first stance phase is equal to the state at the beginning of the first flight phase (corresponding to the step length l_0).
3. *Step Length constraint:* The step-length constraint is enforced as the difference between the position of the stance foot at the beginning of the flight phase and the position of the swing foot at the end of the flight phase being equal to the desired step-length l_0.
4. The state at the end of the first flight phase (before impact) is $x = x_1^-$ with (resulting) step length l_0.
5. Impact constraints at the end of the first flight phase are enforced as $x_1^+ = \Delta(x_1^-)$.
6. The initial state at the start of the second stance phase is given by $x = x_1^+$ with a corresponding (initial) step length of l_1.
7. *Transition between Stance and Flight phase:* The constraint is enforced as in 2 between the second stance and flight phase.
8. The state at the end of the second flight phase (before impact) is $x = x_2^-$ with (resulting) step length of l_2.
9. Impact constraints at the end of the second step are enforced as $x_2^+ = \Delta(x_2^-)$.
10. Periodic constraints are then enforced as $x_2^+ = x_0^+$, resulting in $l_2 = l_0$.

The generation of the two-step-periodic running gaits using direct collocation with the specifications mentioned above involves discretization of each phase in time by a specified number of nodes N,

$$0 = t_0 < t_1 < t_2 < \cdots < t_N = T, \tag{3.13}$$

where T represents the time to impact. In particular, we use $N = 10$ for each phase and use the method of Direct collocation to solve the following trajectory optimization problem,

$$\begin{aligned} J = \min_{u(t)} \quad & \int_0^T ||\tau(t)||_2^2 \, dt \\ \text{st.} \quad & x(t) = \int_0^t f(x(t)) + g(x(t))\tau(t)dt \\ & c(x(t), \tau(t)) \leq 0, \quad 0 \leq t \leq T. \end{aligned} \tag{3.14}$$

Here, $c(x(t), \tau(t))$ represents the physical constraints described in Table 3.1 as well as *periodicity constraints*. The desired gait parameters $\mathcal{P}(l_0, l_1)$ can be extracted from the optimal state trajectories through a simple Bézier curve fit. We use the open-source optimization and simulation toolbox `FROST` [61] to perform the above optimization. We refer the reader to [86] for more details on the specifics of the trajectory optimization scheme. We then generate the gait library $\mathcal{G}$ by obtaining the parameters $\mathcal{P}(l_0, l_1)$ for different values of $(l_0, l_1) \in L \times L$ through the NLP described above.

Control Design

We use an input-output linearizing controller as in [137, 138]. We first define the outputs to be regulated to zero as the difference between the actual and desired quantities y_a and y_d as:

$$y^p := y_a^p - y_d^p, \; p \in \{s, f\}. \tag{3.15}$$

We further differentiate between the relative degree one and relative degree two outputs during stance as

$$y_1^s := \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} (y_a^s - y_d^s), \tag{3.16}$$

$$y_2^s := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (y_a^s - y_d^s). \tag{3.17}$$

The IO Linearizing controller is then given by,

$$u = (A^p)^{-1} \left(-B^p(x, \alpha^p, \theta^p) + v^p\right), \; p \in \{s, f\}, \tag{3.18}$$

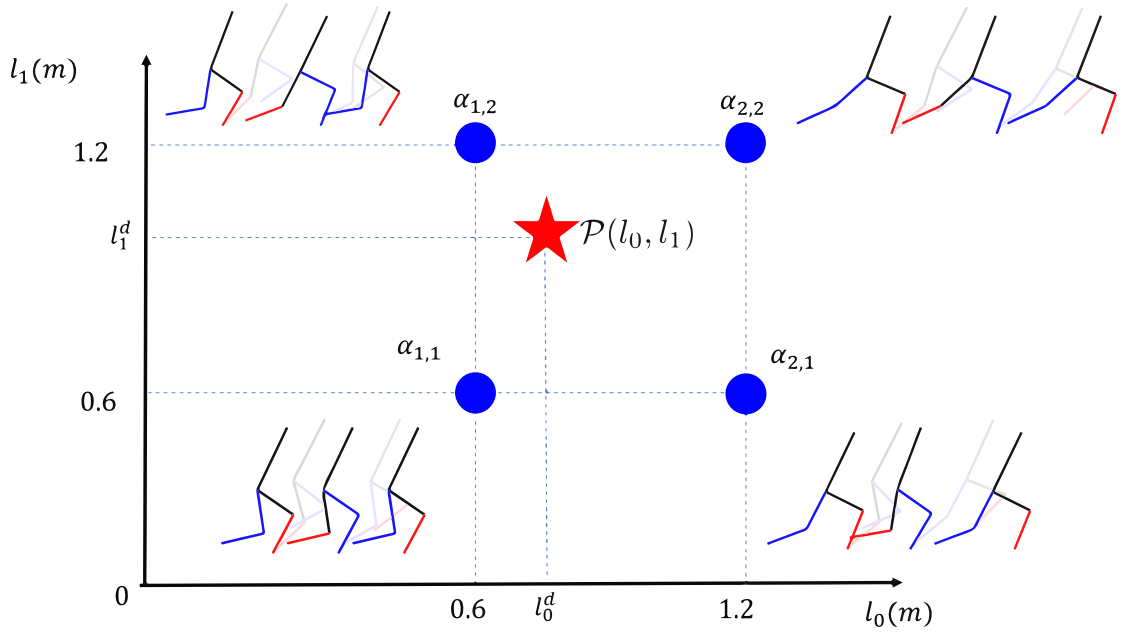


Figure 3.6: Illustration of gait interpolation. The desired gait parameters are obtained based on the desired step length of the previous step l_0^d and the desired step length of the current step l_1^d. Points marked by blue circles denote gait parameters in the gait library. Red star denotes the gait parameters based on the desired step lengths and obtained using bilinear interpolation of the existing gaits in the gait library.

where A^p is the decoupling matrix and is defined as

$$A^s := \begin{bmatrix} L_g y_1^s \\ L_g L_f y_2^s \end{bmatrix}, \tag{3.19}$$

$$A^f := L_g L_f y^f, \tag{3.20}$$

and B^p is defined as

$$B^s := \begin{bmatrix} L_f y_1^s \\ L_f^2 y_2^s \end{bmatrix}, \tag{3.21}$$

$$B^f := L_f^2 y^f. \tag{3.22}$$

Here, $L_f y$ and $L_g y$ denote the Lie-derivatives of the output y with respect to the vector fields f and g. Further, v is a feedback term, which could be, for example, a linear feedback

controller

$$v^s := \begin{bmatrix} -K_1 y_1^s \\ -K_2 y_2^s - K_3 \dot{y}_2^s \end{bmatrix}, \tag{3.23}$$

$$v^f := -K_4 y^f - K_5 \dot{y}^f, \tag{3.24}$$

where $K_i > 0, i = 1, 2, ...5$ are appropriate gain matrices.

The specific values of the parameters α^p and θ^p depend on the desired step lengths of the preceding step l_0^d and current step l_1^d. The superscript d represents the desired quantity. We restrict the desired step lengths l_1^d to be within the range of L. This is schematically illustrated in Figure 3.6 We use bilinear interpolation of the gait parameters $\mathcal{P}$ (3.10) as in [105] to compute the gait parameters $\mathcal{P}(l_0, l_1)$ corresponding to the step lengths l_0^d and l_1^d.

Remark 3.1. *In our method, we perform a linear interpolation of the Bézier parameters that parametrize the periodic gaits rather than the time trajectories of the states. The interpolated gait is, therefore also a smooth Bézier curve. The primary motivation behind doing so is since the desired step length is different in every step, planning for each of these transients would cause an explosion of planned trajectories. Instead, we plan for periodic gaits - specifically a two-step periodic gait to build a library of gaits $\mathcal{G}$ as in (3.11). During implementation, depending on the current step length l_0 and the desired step length l_1 of the next step, we select the four closest gaits (in terms of step length) from $\mathcal{G}$ and perform a bilinear interpolation (See Figure 3.6) resulting in $\mathcal{P}(l_0, l_1)$. Specifically, we only use the first step of the interpolated two-step periodic gait and then switch to another interpolated gait at the end of the first step. This makes the transients smoother (as opposed to large jumps in the desired outputs which generally causes a violation of unilateral constraints such as friction constraints and input constraints) as the exit state at the end of the first step is close to the entry state of the periodic gait used for the second step.*

Remark 3.2. *The proposed method performs a* linear *interpolation as opposed to a nonlinear interpolation. There are certainly several ways to represent this nonlinear model. For example, in [33], the authors propose Support Vector Machines (SVMs) and neural network model to interpolate between the different gaits.*

3.4 Numerical Validation

In this section, we present our numerical results from simulation of the five-link underactuated robot RABBIT. We performed multiple simulations with randomly varying desired step lengths. The desired step lengths were sampled from a uniform distribution between 0.6m and 1.2m, which is about twice the desired step length reported in [100] and about 1.5 times the robot's leg length. Figure 3.5 presents snapshots of the robot running over one realization of the discrete terrain. Figure 3.7 illustrates the footstep locations along with the

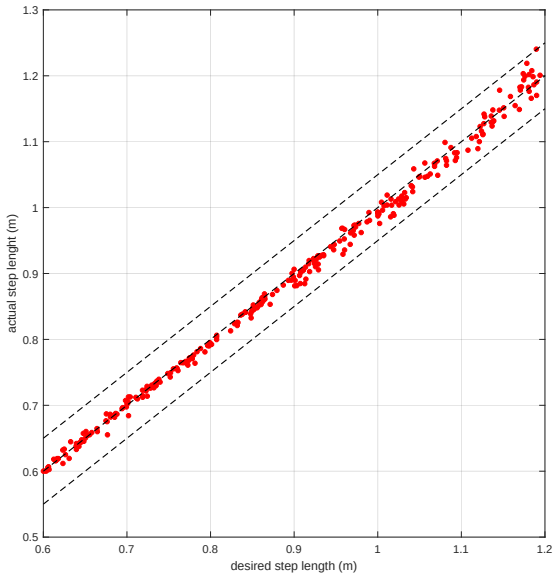


Figure 3.7: Resulting location of footsteps from 300 steps, over 10 experiments (30 steps per experiment). Outer dashed lines indicate a 5cm deviation from the desired step length. Red dots denote the actual value of the step length obtained from simulation.

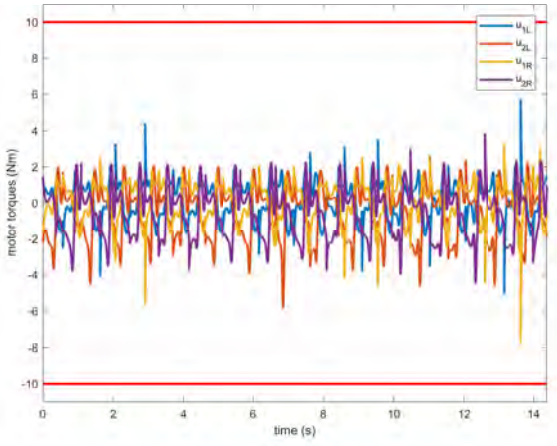


Figure 3.8: Motor Torques from one of the simulations. The top and bottom red lines indicate the maximum and minimum allowable control inputs.

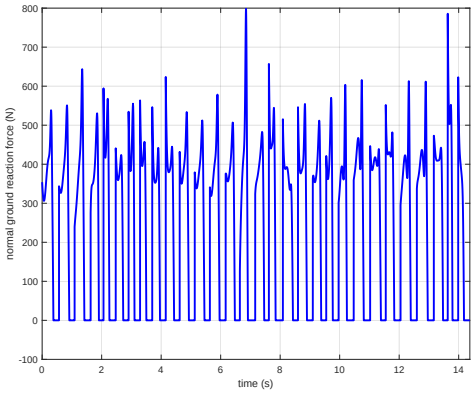


Figure 3.9: Vertical Ground Reaction Forces from one of the simulations.

desired stepping locations from ten such simulations with 30 steps per simulation. We note that the average step-length error increases as the desired step-length increases.

In all our simulations, the foot placement was accurate to ±5cm of the desired step lengths while all other constraints, such as input limits (Figure 3.8) and unilateral vertical ground reaction force (Figure 3.9) constraints were met. The average running velocity was 1.8m/s.

Remark 3.3. *As mentioned in Section 3.3, a candidate choice for the outputs during the stance phase are the actuated joint angles q_a (all outputs have relative degree two). However, the results we obtained using these outputs were very different from those obtained using the outputs defined in (3.3) with one relative degree one output. In particular, we observe a significantly poor performance in the placement of footsteps with a maximum error of 39*cm*. We attribute the success of the outputs in (3.3) to the fact that the horizontal displacement of the center of mass depends solely on its exit velocity during stance. Regulating the exit velocity directly during stance will potentially lead to an accurate step length at the end of the flight phase and hence smaller errors in step length. Figure 3.10 illustrates the performance of the controller using the actuated joints q_a as the outputs.*

3.5 Chapter Summary

In this chapter, we presented a control strategy for bipedal robotic running over stochastically varying discrete terrain and potentially increased the range of step lengths to twice that could have been achieved only by walking. The controller maintains the stability of the robot while respecting critical safety constraints such as constraints on foot placement and ground reaction forces. With a proper choice of the outputs to be controlled, the resulting step length from simulation is accurate to 5cm of the desired step length.

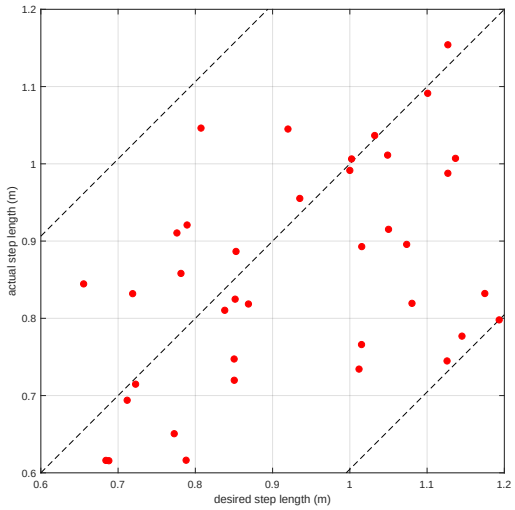


Figure 3.10: Resulting location of footsteps from 50 steps from a single simulation using q_a (vector relative degree 2) as the outputs during the stance phase. Outer dashed lines indicate a 39cm deviation from the desired step length. Red dots denote the actual value of the step length obtained from simulation.

While we are yet to formally prove any theoretical guarantees on switching between the different periodic gaits, we provide some intuitive explanation about the success of the method as in Remark 3.1. A potential direction to address this is the stability conditions for switching controllers between different exponentially stable periodic orbits as provided in [92]. Moreover, the method here is simple to implement and requires a small number of two-step periodic gaits to run over a terrain with a wide range of step lengths.

While this chapter focused on developing feedback controllers for a planar bipedal robot, in the next chapter, we develop a locomotion controller for a 3D bipedal robot `Cassie`. In particular, we develop an optimization-based controller that can handle physical constraints such as friction constraints, and we validate our approach through numerous hardware experiments.

Chapter 4

Geometric Variational Model Predictive Control

In Chapter 3, we introduced an HZD-based control method for a planar 2D bipedal robot. HZD is a popular approach that was initially developed for 2D bipedal robots and has been extended to several 3D bipedal robots in recent years. At its core, the HZD approach begins by finding a periodic orbit that incorporates the full nonlinear and hybrid dynamics of the robot and includes physical constraints, such as friction and input limits, as well as user-defined constraints, such as walking speed and step length, through a constrained nonlinear program. Once a periodic solution is found, an input-output linearizing controller is utilized to drive the outputs to zero. In addition to requiring the true nonlinear dynamics of the robot, the input-output linearizing controller does not enforce physical constraints such as unilateral ground reaction force and friction constraints or limits enforced by the robot's actuators. More recent work has leveraged advances in optimal controls to incorporate these constraints in the feedback control design. These methods solve a pointwise QP at every control iteration to track the desired outputs computed offline. However, a drawback of such approaches is the lack of predictive capability.

In this chapter, we take a departure from the HZD method and develop a model predictive controller based on a reduced order model which is geometrically consistent with the underlying configuration manifold structure of this model. We illustrate the efficacy of our approach through several numerical simulations and hardware experiments on the `Cassie` biped.

4.1 Related Work

MPC is a widely used method to control legged robots but requires linearization of CoM dynamics to simplify the underlying optimization for efficient real-time computation. A common approach to linearizing the CoM dynamics involves a small angle approximation

of the body roll and pitch, and a *Jacobian* linearization of the orientation dynamics [37, 146]. However, the small-angle approximation restricts the domains in which the model is valid, especially on uneven terrain where the robot might experience high angular velocities and pitch due to disturbances. Additionally, since the dynamics of the robot body evolve on the $SE(3)$ manifold, singularity issues arise in the Jacobian linearization process. Euler discretization of the continuous-time orientation dynamics also results in the loss of the underlying geometric structure of the $SO(3)$ manifold and, as a result, the discrete-time dynamics are not energy preserving [124].

This has led to research in geometric variation-based optimal control approaches [26, 141, 66] that linearize the quadruped dynamics using *rotation matrices* instead of Euler angles. The resulting linearization is coordinate-free and does not suffer from singularities. However, [26] does not consider discrete-time dynamics of the linearized system required for MPC, and [66] use forward Euler to discretize the orientation dynamics. Euler discretization of the orientation dynamics, however, results in the loss of important mechanical properties like energy and momentum conservation, and the discrete-time dynamics may not evolve in the $SO(3)$ manifold [124]. More recently in [32], a nonlinear MPC utilizing Lie group integrators was presented to achieve stable hopping on a monoped. In [131], an error-state MPC is proposed where the linearized dynamics are derived on the Lie Algebra.

4.2 Geometric Model Predictive Control

We now present our Geometric Variational MPC (GVMPC) framework, which outputs the contact forces of the stance legs, with the objective of stabilizing the robot's CoM trajectory and body orientation. GVMPC applies variation-based linearization [141] to a reduced-order model of the robot while ensuring that the discretized system is energy-conserving. Similar to prior works, we model the robot as a single rigid body actuated by linear forces and moments about its CoM.

Discretization

We begin by formulating a discrete-time model of the rigid-body dynamics as required by the MPC. Inspired by [124] we consider the system Lagrangian discretized using the Trapezoidal rule with a time step of $\Delta t \coloneqq t_{k+1} - t_k$,

$$\mathcal{L}_k \approx \int_{t_k}^{t_{k+1}} \mathcal{L} dt = \mathcal{L}\Delta t, \tag{4.1}$$

where $\mathcal{L}$ is the Lagrangian in continuous time. To obtain the discrete-time dynamics of the system, we equate the action sum to zero,

$$\Sigma_{k=0}^{N-1} \delta\mathcal{L}_k + \delta\mathcal{W}_k = 0. \tag{4.2}$$

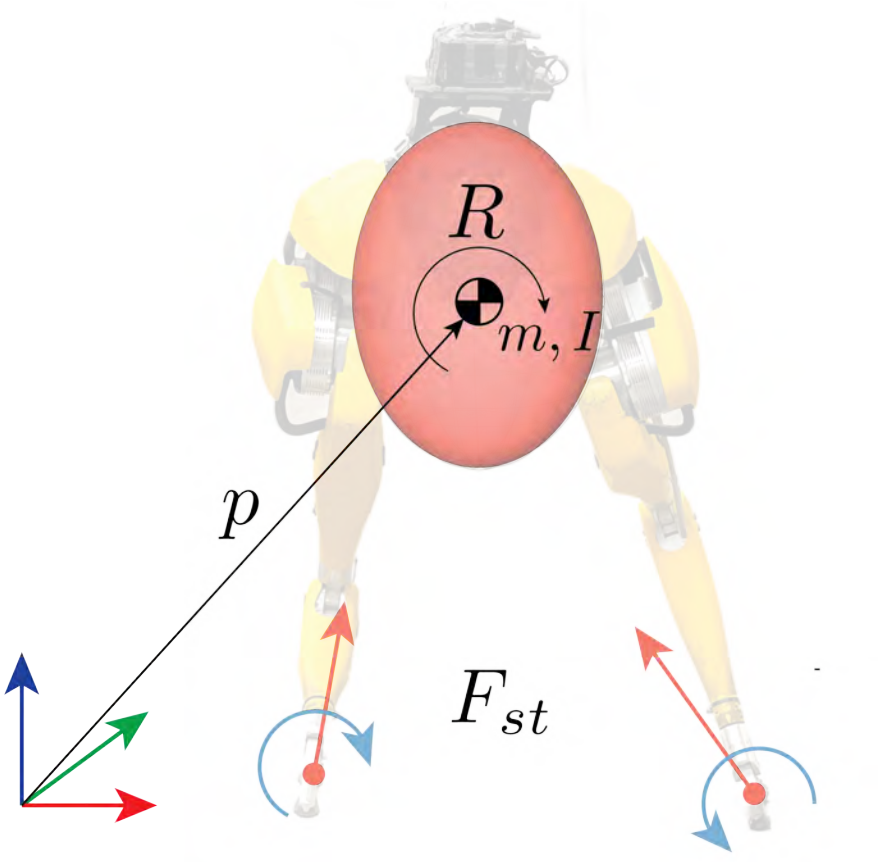


Figure 4.1: A rigid-body reduced order model for Cassie.

$\delta\mathcal{W}_k$ is the infinitesimal work done by the force f_k and moment τ_k obtained as,

$$\delta\mathcal{W}_k := \Delta t \left(f_k \cdot \delta p_k + \tau_k \cdot \delta\eta_k \right), \tag{4.3}$$

where δp_k is an infinitesimal displacement and $\delta\eta_k \in \mathbb{R}^3$ can be interpreted as an infinitesimal change in orientation.

The discrete-time equations of motion for the rigid-body dynamics are then,

$$p_{k+1} = p_k + \dot{p}_k \Delta t, \tag{4.4}$$

$$\dot{p}_{k+1} = \dot{p}_k + \Delta t \mathbf{g} + \frac{f_{k+1}}{m} \Delta t, \tag{4.5}$$

$$R_{k+1} = R_k \Delta R_k, \tag{4.6}$$

$$I\omega_{k+1} = \Delta R_k^T I \omega_k + \Delta t \tau_{k+1}, \tag{4.7}$$

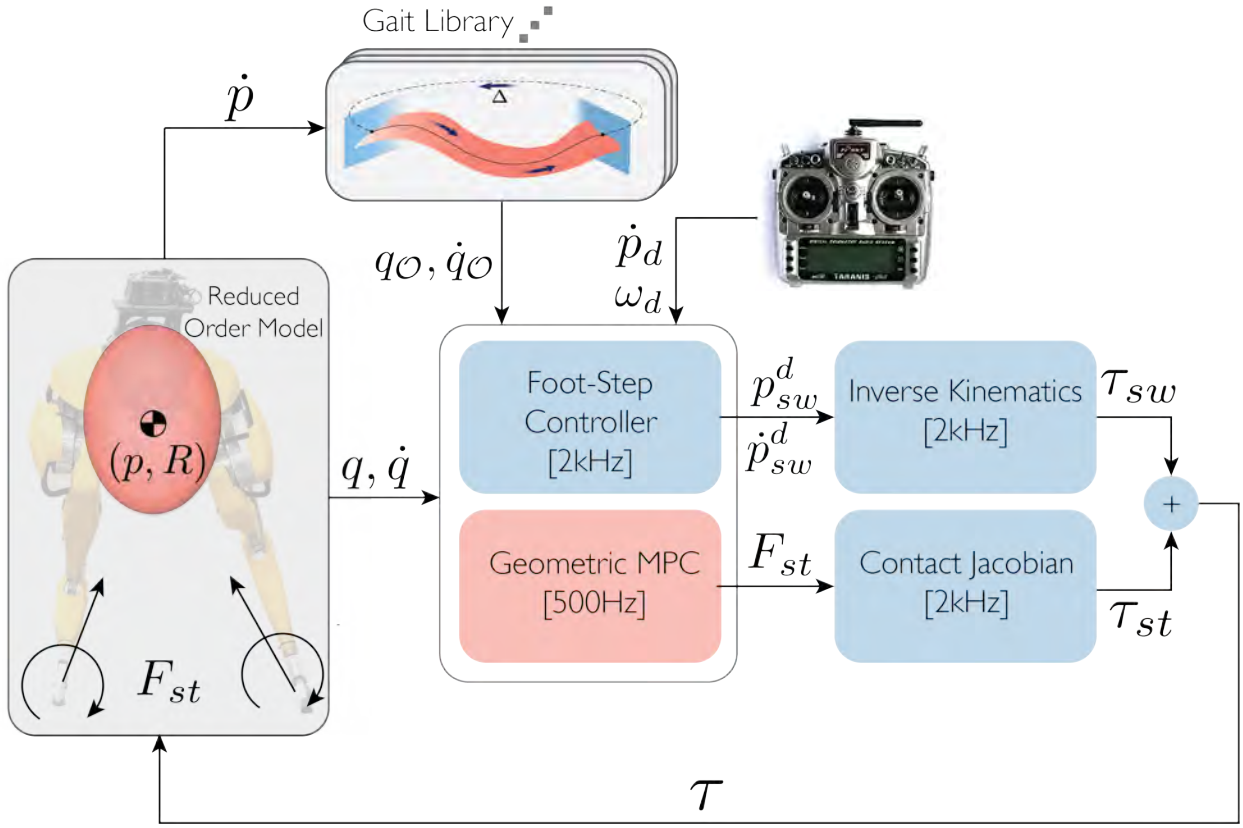


Figure 4.2: Proposed MPC framework applied to Cassie. Blocks in blue are run at $2kHz$ on the real-time computer. The geometric MPC is run at $500Hz$ on an `Intel NuC` computer, which communicates via `UDP` to the real-time computer. The desired velocity $\dot{p}_d$ and angular rate ω_d are sent by the user through the radio.

where $R_k \in SO(3)$ denotes the rotation matrix, $I \in \mathbb{R}^{3\times3}$ is the inertia tensor, $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector. $\Delta R_k \coloneqq \exp\left(\Delta t \hat{\omega}_k\right)$ denotes the change in orientation of the body from time t_k to time t_{k+1}, where the exponential map $\exp : \mathfrak{so}(3) \to SO(3)$ maps a skew-symmetric matrix to a rotation matrix. We define the state of the rigid body to be $\xi_k := \left[p_k^T, \dot{p}_k^T, R_k^T, \omega_k^T\right]^T$ and the input to be $F_k := \left[f_k^T, \tau_k^T\right]^T$.

Linearization

Having obtained the discrete-time model of the system, we next compute a *variation-based* linearization [141] of the nonlinear discrete-time dynamics around a reference trajectory. The resulting linearized model will be locally valid on the $SE(3)$ manifold and will be used to formulate our MPC problem as a quadratic program (QP) that can be solved in real-time. To compute the linearization, we take *infinitesimal variations* around a reference state.

Since the position and velocity dynamics in (4.4) and (4.5) are already linear, we turn to the linearization of the orientation dynamics (4.6) and (4.7). The variations on $SO(3)$ with respect to a reference trajectory $R_k^d \in SO(3)$ is given by,

$$\delta R_k = R_k^d \hat{\eta}_k, \tag{4.8}$$

where $\eta_k \in \mathbb{R}^3$ and $\hat{\eta}_k$ maps $\mathbb{R}^3 \to \mathfrak{so}(3)$ such that $\hat{a}b = a \times b$ for all $a, b \in \mathbb{R}^3$, where $\times$ is the vector cross product. The variation in the angular velocity is

$$\delta\omega_k = \frac{1}{\Delta t}\left(\Delta R_k \eta_{k+1} - \eta_k\right). \tag{4.9}$$

Using the variations in (4.8), and from the nonlinear discrete-time dynamics of the rotation matrix in (4.6), we get the linear discrete-time system about a reference as,

$$\begin{aligned}
R_{k+1} &= R_k \exp\left(\hat{\omega}_k \Delta t\right), && (4.10)\\
\delta R_{k+1} &= \delta R_k \exp\left(\hat{\omega}_k^d \Delta t\right) + R_k^d \delta\left(\exp\left(\hat{\omega}_k \Delta t\right)\right), && (4.11)\\
\Rightarrow R_{k+1}^d \hat{\eta}_{k+1} &= R_k^d \hat{\eta}_k \exp\left(\hat{\omega}_k^d \Delta t\right) + R_k^d \hat{\omega}_k \exp\left(\hat{\omega}_k^d \Delta t\right), && (4.12)\\
\Rightarrow \hat{\eta}_{k+1} &= \underbrace{R_{k+1}^{d^\top} R_k^d}_{\exp\left(\hat{\omega}_k^d \Delta t\right)^{-1}} \left(\hat{\eta}_k \exp\left(\hat{\omega}_k^d \Delta t\right) + \widehat{\delta\omega}_k \exp\left(\hat{\omega}_k^d \Delta t\right)\right), && (4.13)\\
\Rightarrow \hat{\eta}_{k+1} &= \exp\left(\hat{\omega}_k^d \Delta t\right)^{-1} \hat{\eta}_k \exp\left(\hat{\omega}_k^d \Delta t\right) + \exp\left(\hat{\omega}_k^d \Delta t\right)^{-1} \widehat{\delta\omega}_k \exp\left(\hat{\omega}_k^d \Delta t\right), && (4.14)\\
&= \widehat{\Delta R_k^{d^\top} \eta_k} + \Delta t \widehat{\Delta R}_k^{d^\top} \delta\omega_k \quad \left[\because R^\top \hat{\omega} R = \widehat{R^\top \omega}\right], && (4.15)\\
\Rightarrow \eta_{k+1} &= \Delta R_k^{d^\top} \eta_k + \Delta t \Delta R_k^{d\top} \delta\omega_k. && (4.16)
\end{aligned}$$

Similarly, the linearized discrete-time dynamics for the angular velocity is obtained from (4.7), (4.8) and (4.9) as,

$$\begin{aligned}
\delta\left(I\omega_{k+1}\right) &= \delta\left(\Delta R_k^\top I \omega_k + h\tau_{k+1}\right), && (4.17)\\
\Rightarrow I\delta\omega_{k+1} &= \delta\Delta R_k^\top I\omega_k^d + \Delta R_k^{d^\top} I\delta\omega_k + h\delta\tau_{k+1}, && (4.18)\\
\Rightarrow I\delta\omega_{k+1} &= \Delta t\left(\delta\hat{\omega}_k \Delta R_k^d\right)^\top I\omega_k^d + \Delta R_k^{d^\top} I\delta\omega_k + \Delta t\delta\tau_{k+1}, && (4.19)\\
\Rightarrow I\delta\omega_{k+1} &= \Delta t \Delta R_k^{d^\top} \delta\omega_k^\top I\omega_k^d + \Delta R_k^{d^\top} I\delta\omega_k + h\delta\tau_{k+1}, && (4.20)\\
&= -\Delta t \Delta R_k^{d^\top} \delta\omega_k I\omega_k^d + \Delta R_k^{d^\top} I\delta\omega_k + \Delta t\delta\tau_{k+1}, && (4.21)\\
&= \Delta t \Delta R_k^{d^\top} \left(\widehat{I\omega_k^d}\right)\delta\omega_k + \Delta R_k^{d^\top} I\delta\omega_k + \Delta t\delta\tau_{k+1}, && (4.22)\\
I\delta\omega_{k+1} &= \Delta t \Delta R_k^{d^\top} \left(\Delta t\widehat{I\omega_k^d} + I\right)\delta\omega_k + \Delta t\delta\tau_{k+1}. && (4.23)
\end{aligned}$$

Putting together the linear and angular components, the linearized discrete-time system is given by,

$$\delta\xi_{k+1} = A_k\delta\xi_k + B_k\delta F_k, \tag{4.24}$$

where $\delta\xi_k \coloneqq \left[\delta p_k^T, \delta\dot{p}_k^T, \eta_k^T, \delta\omega_k^T\right]^T$ is the error state of the linearized system.

The matrices A_k and B_k are given by,

$$A_k \coloneqq \begin{bmatrix} \mathbb{I}_3 & \Delta t\mathbb{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbb{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \Delta R_k^{d^T} & \Delta t\Delta R_k^{d^T} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & a_\omega \end{bmatrix}, \tag{4.25}$$

$$B_k \coloneqq \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ \frac{\Delta t\mathbb{I}}{m} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \Delta t I^{-1} \end{bmatrix}, \tag{4.26}$$

$$a_\omega \coloneqq I^{-1}\Delta R_k^{d^T}\left(\Delta t\widehat{I\omega_k^d} + I\right).$$

The linear discrete-time dynamics in (4.24) represents the evolution of the infinitesimal variations on the manifold around a reference trajectory. These variations represent the distance between two points on the manifold. Under the assumption that the actual rotation matrix R_k is close to the desired rotation matrix R_k^d, the variation $\delta\xi_k$ can be approximated as

$$\delta\xi_k \approx \begin{bmatrix} p_k - p_k^d \\ \dot{p}_k - \dot{p}_k^d \\ \frac{1}{2}\left(R_k^{d^T}R_k - R_k^T R_k^d\right)^\vee \\ \omega_k - R_k^T R_k^d \omega_k^d \end{bmatrix}, \tag{4.27}$$

where the vee map $\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the inverse of the hat operator, so that $\hat{x}^\vee = x$, $\forall x \in \mathbb{R}^3$. The last two terms in (4.27) denote the errors on the tangent bundle $TSO(3)$ manifold [80, 22]. With this approximation, the dynamics in (4.24) represents the evolution of the error on the manifold locally around the reference trajectory ξ^d.

Geometric MPC-QP

Given the desired CoM states ξ^d generated from the motion library at the current trotting step, we compute the initial error state $\delta\xi(0)$ as in (4.27) and solve the following QP,

$$F^*_{st,k} = \underset{F_{st,k},\delta\xi_k,\delta F_k}{\arg\min} \|\delta\xi_N\|_P + \Sigma_{k=0}^{N} \left(\|\delta\xi_k\|_Q + \|\delta F_k\|_R + \|F_{st,k} - F^0_{st}\|_{Q_{st}}\right) \tag{4.28}$$

$$\text{s.t.} \quad \delta\xi_{k+1} = A_k(\xi^d_k)\delta\xi_k + B_k(\xi^d_k)\delta F_k, \tag{4.29}$$

$$F_{st,k} \in \mathcal{K}^{line}_{\text{st}}(\mu, \gamma, l_f), \tag{4.30}$$

$$0 \leq F^z_{st_i} \leq c^i_k \bar{F}_{st}, \quad i \in \{0,1,2,3\} \tag{4.31}$$

$$G_c F_{st,k} = \delta F_k + \begin{bmatrix} m\mathbf{g} \\ \mathbf{0}_{3\times 1} \end{bmatrix}, \tag{4.32}$$

$$\delta\xi_0 = \delta\xi(0), \tag{4.33}$$

where (4.30) denotes the line-contact constraints defined in (2.21), (4.33) denotes the CoM wrench and contact forces, with G_c denoting the grasp-map [94]. The inequality in (4.31) represents the unilateral constraints on the vertical ground reaction forces at the feet; $c^i_k \in \{0,1\}$ denotes the binary contact state of foot $i \in \{L, R\}$ (L and R denoting *Left* and *Right* foot respectively). The above QP outputs the desired contact wrenches $F^*_{st,k}$ for the stance feet. For legs in swing, the contact forces are set to zero by the constraint in (4.31). The term $\|F_{st,k} - F^0_{st}\|_{Q_{st}}$ in the cost incentivizes the predicted contact wrench $F_{st,k}$ to remain close to the desired contact wrench applied at the previous timestep F^0_{st}. We implement the above QP using the `OSQP` solver [127], with a horizon length of 20 and timestep of 0.02s, which is equivalent to the desired time period of the walking gait. The stance-leg torques are obtained through the quasi-static relation

$$\tau_{st} = -J^T_{st} F^*_{st,0}. \tag{4.34}$$

On hardware, we also implement an inverse-kinematics-based control to track the desired rigid body states ξ^d, making the total stance leg torque

$$\tau_{st} = -J^T_{st} F^*_{st,0} + \tau_{ik,st}\left(\xi, \xi^d\right). \tag{4.35}$$

Additionally, for walking, we set the desired $x-y$ position of the center of mass to be equal to the actual $x-y$ position, i.e. $S_{xy} p^d_k \equiv S_{xy} p_k$, where $S_{xy} := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$.

Remark 4.1. *Since the horizon length of the MPC spans one walking step, the desired contact state c^i_k of either foot can change during this period. As our MPC does not optimize over the contact states, the grasp map G^k_c and desired contact state c^i_k (which are parameters of the MPC problem) must be carefully chosen when formulating the constraints at every timestep. For $k = 0$, we utilize the true contact state of each foot obtained by thresholding the estimated spring forces and grasp map obtained from forward kinematics. For $k > 0$, we set the desired contact state c^i_k depending on the time remaining in the current gait cycle. When $k\Delta t$ exceeds the time remaining in the gait, the desired contact state for each leg*

Parameter Description	Symbol	Numerical Value
State Stage Cost	Q	`diag`[500, 500, 2300 500, 500, 600, 80, 20, 10, 20, 5, 1]
Input Stage Cost	R	`diag`[0.005, 0.005, 0.001 0.005, 0.05, 1]
Contact Force Difference Stage Cost	Q_{st}	`diag`[10^{-2}, 10^{-2}, 10^{-2}, 10^{-4},10, 10^{-2}, 10^{-2}, 10^{-2}, 10^{-4},10]
Terminal Cost	P	`dlqr` $(A_k(\mathbf{0}), B_k(\mathbf{0}), Q, R)$
Maximum Vertical Contact Force	$\bar{F}_{st}^z$	1000N
Coefficient of static friction	μ	0.6
Coefficient of twisting friction	γ	0.1
Foot Length	l_f	$0.18m$
Horizon Length	N	20

Table 4.1: Numerical values of the MPC parameters implemented on the Cassie robot hardware.

is switched. Additionally, the grasp map for the swing leg is computed based on its target stepping location obtained from (4.36).

We tabulate the numerical values of the parameters in the Geometric MPC in Table 4.1.

Gait Library

Similar to [47], we utilize a library of periodic orbits for forward walking velocities ranging from $-0.5m/s$ to $1m/s$ obtained using the `FROST` toolbox [61]. The gaits have a step-time of $T_s = 0.4s$, and consist of alternating phases of *left stance* and *right stance* phases, with no double support phase. At every timestep of the low-level controller, a gait from the library is chosen depending on the forward velocity of the robot, from which nominal configuration variables and configuration velocities $q_{\mathcal{O}}, \dot{q}_{\mathcal{O}}$ on the periodic orbit are obtained as a function of a time-based phase variable $\tau_s := 1/T_s$. The desired reduced-order model states ξ^d are also obtained from the chosen gait.

Swing Leg Control

The periodic orbits obtained for Cassie are only marginally stable and require additional foot-stepping strategies on the swing leg to stabilize them. We implement a commonly used

Raibert heuristic to obtain a target stepping location $p^{tgt}_{sw,xy}$ at the end of a walking step to stabilize the robot,

$$p^{tgt}_{sw,xy} = p^{tgt}_{\mathcal{O},xy} + k^{sw}_p \left(\dot{p}_{xy} - \dot{p}^d_{xy}\right) + k_d \left(\dot{p}_{xy} - \bar{v}_{xy}\right), \tag{4.36}$$

where $p^{tgt}_{\mathcal{O},xy}$ denotes the nominal swing foot position obtained from the gait library, $\dot{p}_{xy}$ is the instantaneous forward, and lateral velocity of the center-of-mass of the robot, $\dot{p}^d_{xy}$ is the desired forward, and lateral velocity of the robot input by the user, and $\bar{v}_{xy}$ is the average velocity of the robot from the previous walking step. Next, we obtain the instantaneous desired swing foot position and velocity by interpolating between the foot position at the start of the gait to the target position using a bezier spine. The desired joint angles $q^d_{sw} \in \mathbb{R}^5$ and velocities $\dot{q}^d_{sw} \in \mathbb{R}^5$ for the swing legs are found using inverse kinematics and tracked using a PD controller with constant gravity compensation torque τ^g_{sw} for the swing knee pitch and hip roll joints,

$$\tau_{sw} = -k^{sw}_p \left(q_{sw} - q^d_{sw}\right) - k^{sw}_d \left(\dot{q}_{sw} - \dot{q}^d_{sw}\right) + \tau^g_{sw}, \tag{4.37}$$

where q_{sw} and $\dot{q}_{sw}$ are the instantaneous joint angles and joint velocities of the swing leg, respectively.

An overview of the proposed controller is illustrated in Figure 4.2. In the next section, we present our experimental results utilizing the proposed MPC method on the Cassie robot hardware.

4.3 Experiments

In this section, we present our main experimental results on Cassie bipedal robot. We implement our MPC on an external onboard computer with an `Intel i7-10710U` processor and communicate via UDP to the real-time computer.

Balancing and Crouching in Place

We begin by illustrating the dynamic balance capabilities of the proposed MPC framework. When the robot is balancing on both legs, it is fully actuated. Cassie is able to stably balance while being robust to perturbations in the form of pushes. Figure 4.3 illustrates a crouching maneuver performed by the robot where the desired height is input by the user.

Walking Experiments

Next, we perform several indoor walking experiments to test the robustness and efficacy of our approach. These experiments are summarized below.

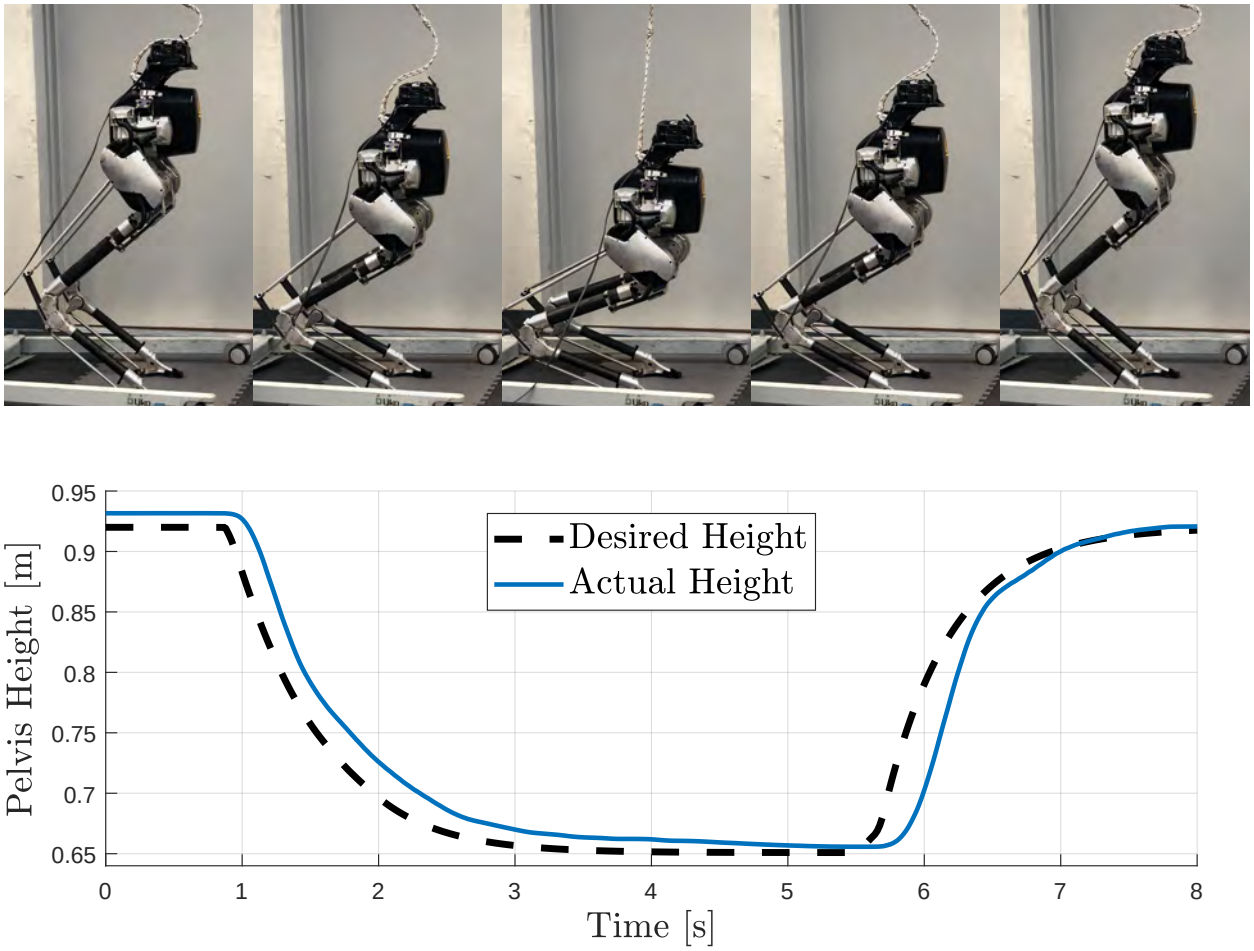
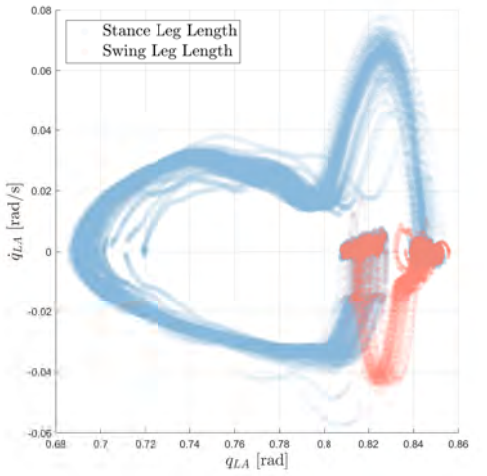
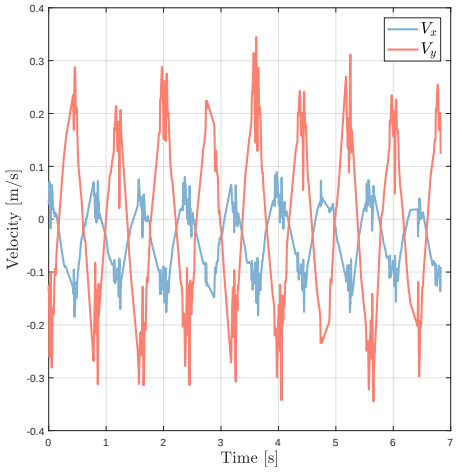


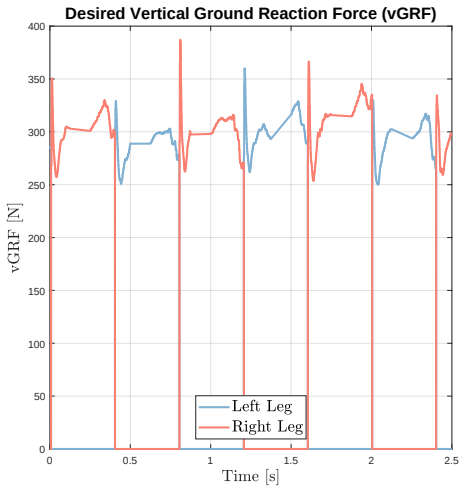
Figure 4.3: (Top) Snapshots of Cassie performing a crouching maneuver using the proposed approach. (Bottom) Plot illustrating tracking of the pelvis height. Note that the desired pelvis velocity is set to zero in this experiment since the desired height command is input by the user through the radio.



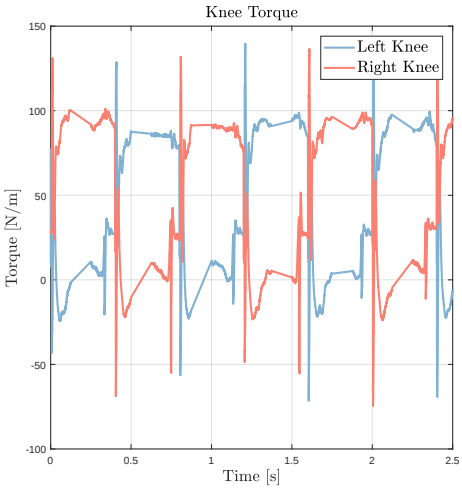
(a) Phase portrait of the leg length in stance and swing phases for stepping in place.



(b) Estimated forward and lateral velocity.



(c) Desired vertical ground reaction forces (vGRF) from MPC.



(d) Knee pitch torque for left and right legs.

Figure 4.4: Expriemental results for stepping in place using the proposed MPC approach.

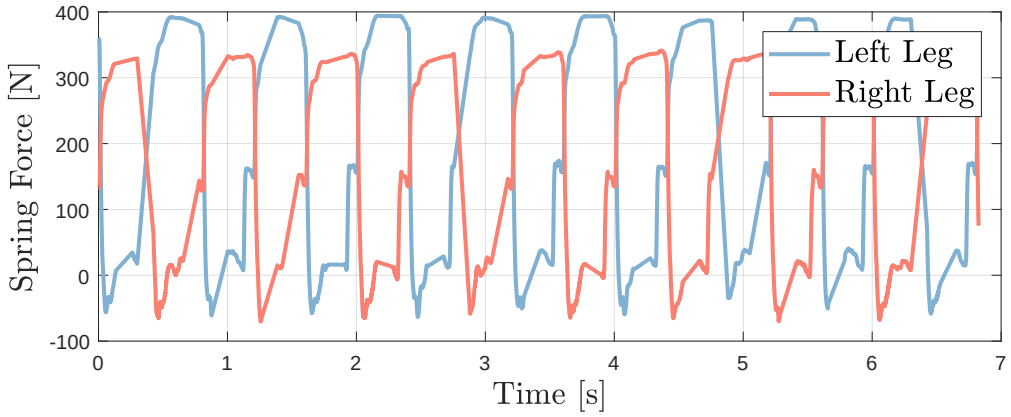


Figure 4.5: Estimated spring forces from hardware for stepping in-place.

Stepping In-Place and Push Recovery

We first illustrate our approach for stepping in place in a lab setting. The robot can stably step in place as illustrated by the phase portrait of the stance leg length in Figure 4.4a and the forward and lateral velocity in Figure 4.4b. Figure 4.4c illustrates the desired vertical ground reaction forces from the MPC. As mentioned earlier, the desired contact forces are converted to joint torques through a kinematic relationship using the jacobian transpose. The corresponding knee torques are illustrated in Figure 4.4d. In Figure 4.6, we show the gait tiles of the robot recovering from a push in the backward direction. The robot can quickly (in approximately 3s) recover from the push. Figure 4.5 illustrates the estimated spring forces which are used to obtain the contact state of each foot.

Forward, Backward, Lateral and Strafing Motions

We conduct numerous experiments to illustrate the versatility of our controller to walk forwards, backward, laterally, and diagonally. Figure 4.7 illustrates snapshots of the robot walking sideways and diagonally. Figure 4.8 illustrates the tracking performance of the controller.

Robustness to Ground Perturbations

Next, we test our controller against a variety of ground perturbations such as slopes and small step disturbances (Figure 4.9) and across randomly placed wooden planks and soft rubber mats (Figure 4.10). Additionally, we also demonstrate that the robot can sidestep onto small steps, as illustrated in Figure 4.11.

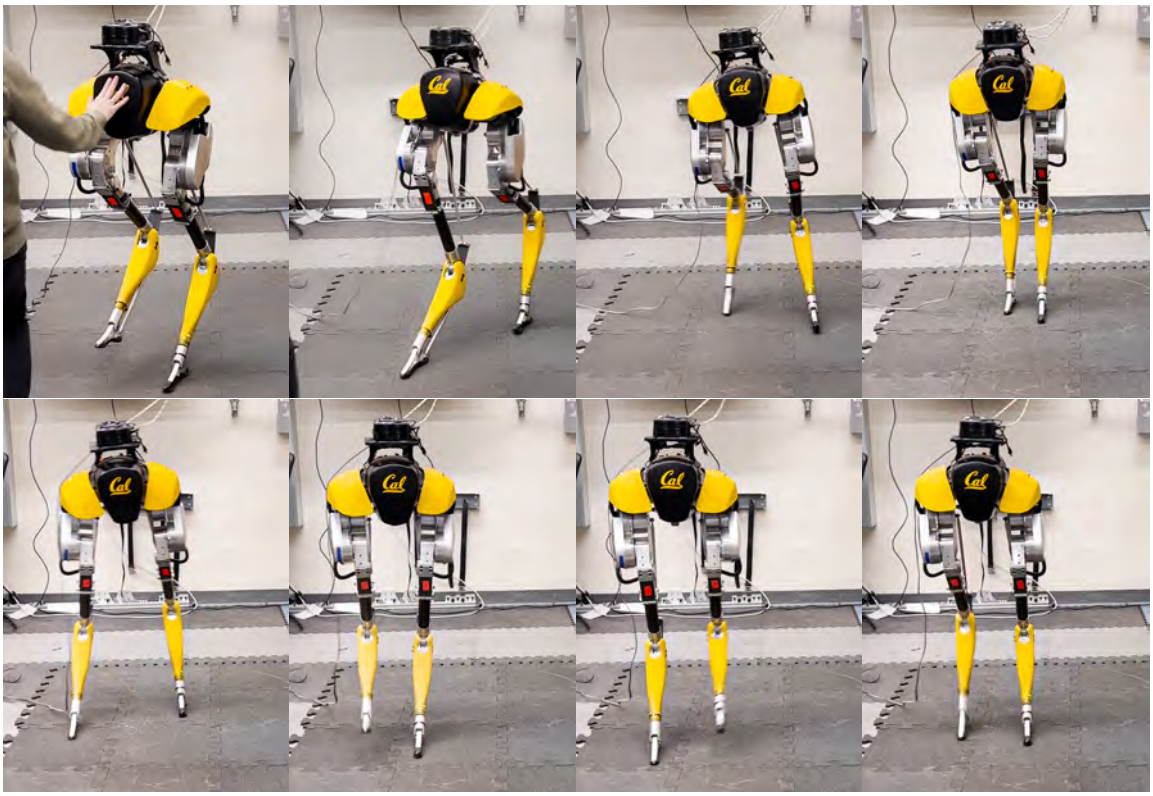


Figure 4.6: Snapshots of Cassie robot illustrating recovery from a push in the backwards direction. The first and the last tiles are about 3s apart.

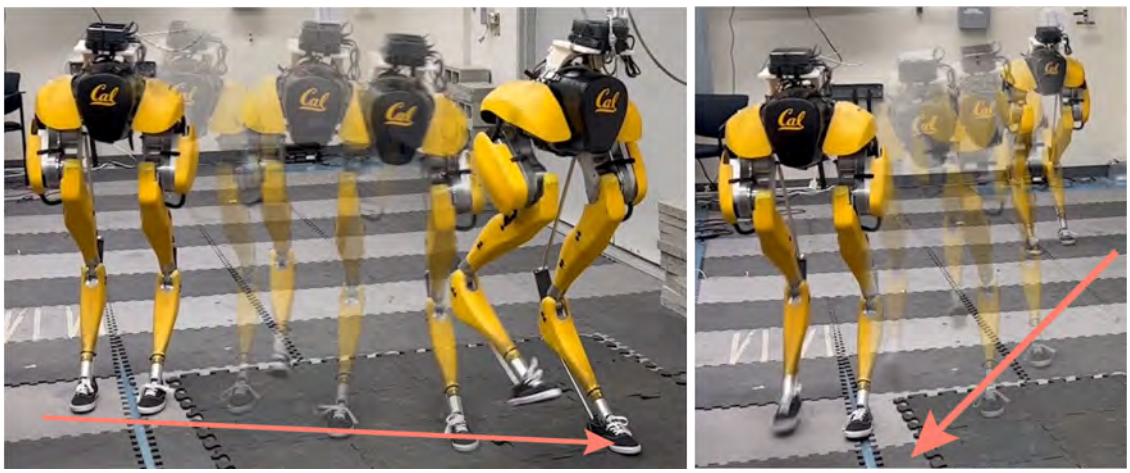


Figure 4.7: Snapshots of Cassie walking diagonally and sideways.

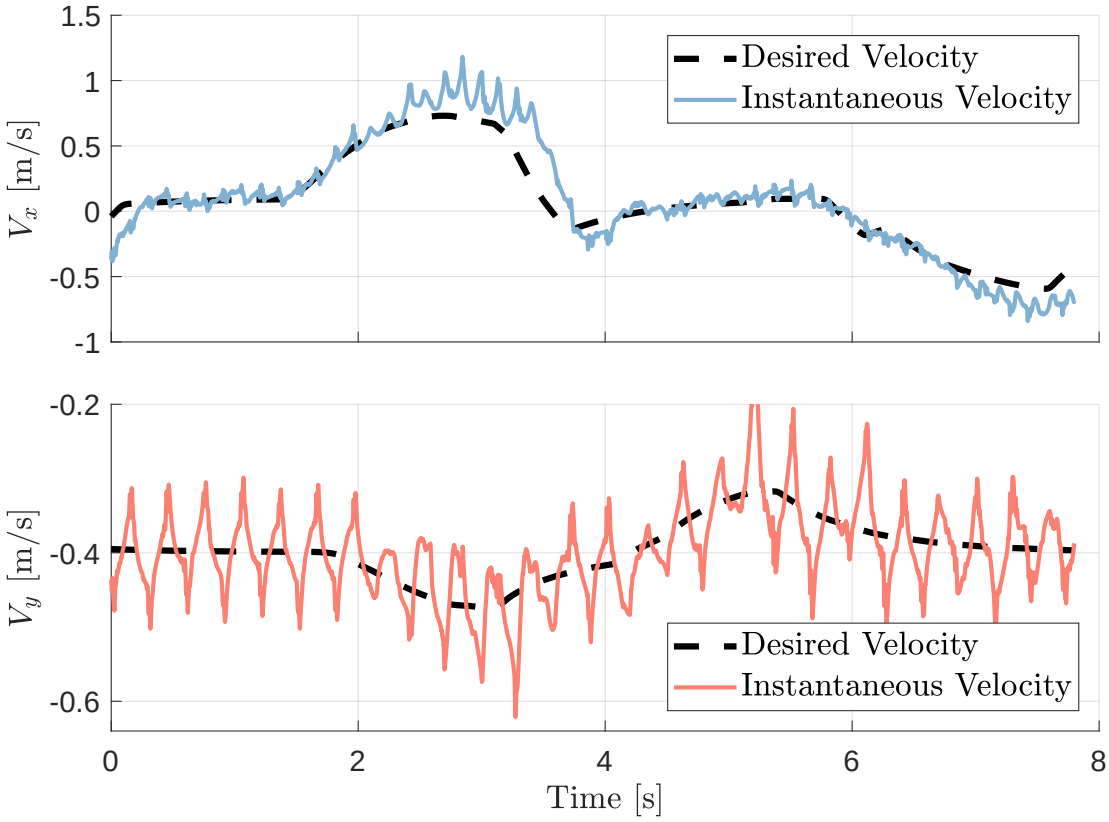


Figure 4.8: Estimated forward (Top) and lateral (Bottom) velocities while the robot is walking diagonally ($2s - 4s$) with a maximum forward velocity of $1m/s$ and a maximum lateral velocity of $0.5m/s$.

Outdoor Walking

We perform several outdoor experiments on a wide variety of rugged terrain, including large sloped concrete roads, sidewalks with irregularities, grass, loose ground such as mulch and step disturbances such as across curbs. Figure 4.12 highlights several outdoor experiments conducted across the UC Berkeley campus.

4.4 Chapter Summary

In this chapter, we proposed a linear MPC based on a rigid body model of the robot that evolves on the $SE(3)$ manifold. We utilized tools from differential geometry to derive linear discrete-time dynamics of the reduced-order model that is geometrically consistent with the underlying configuration manifold of this reduced-order model. We validated the proposed

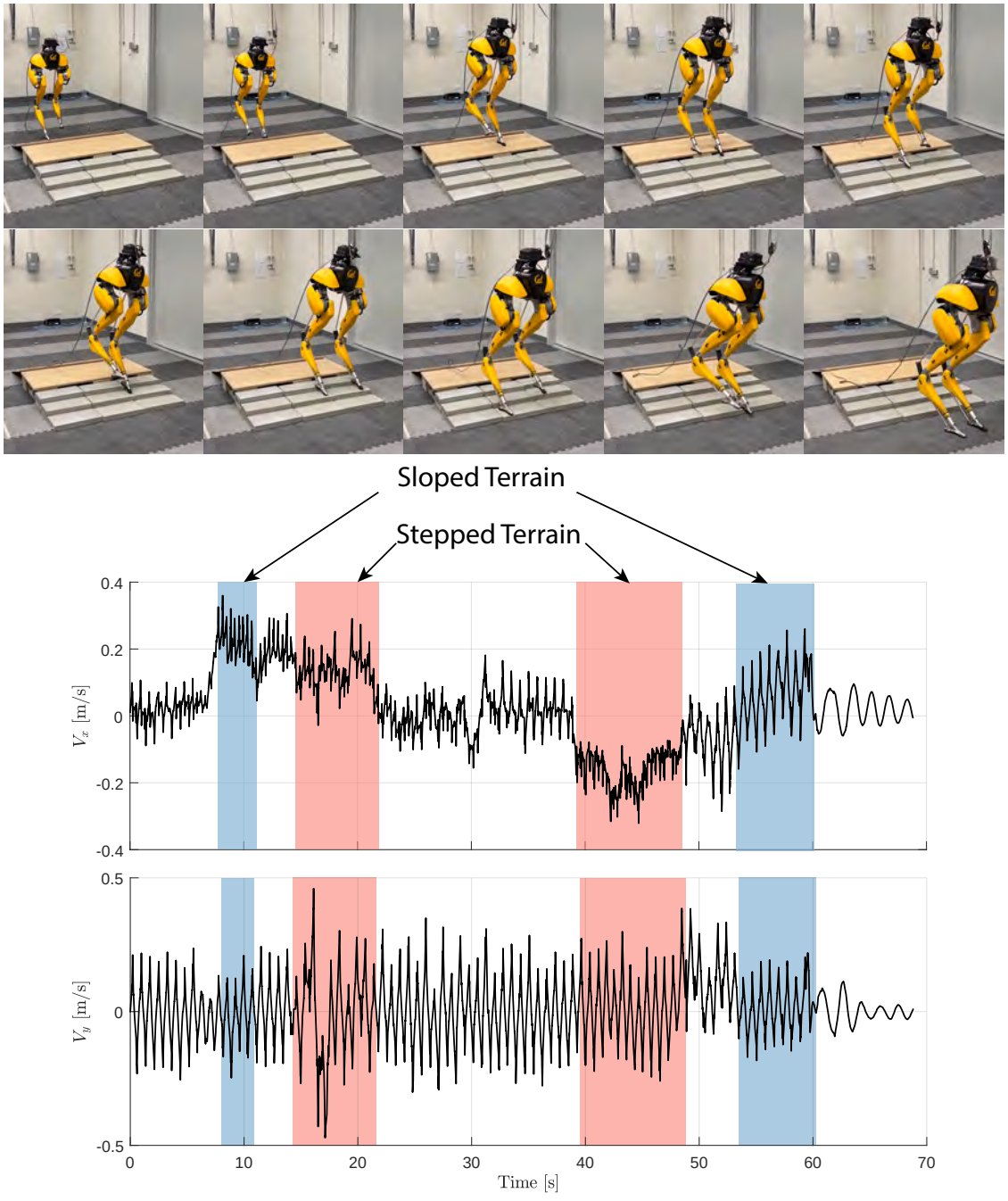


Figure 4.9: (Top) Snapshots of Cassie blindly negotiating a ramp of 20° and a flight of stairs of height 6cm using the proposed Model Predictive Controller. (Bottom) Estimated forward and lateral velocities experienced by the robot.

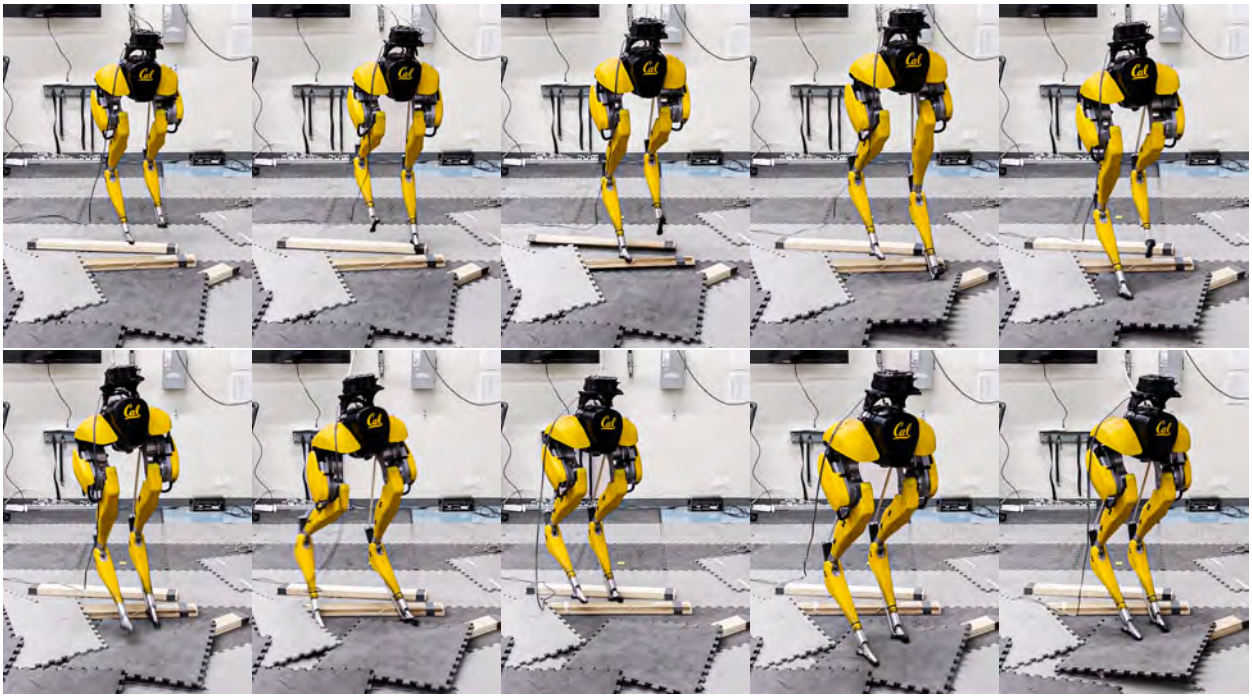


Figure 4.10: Snapshots of Cassie blindly walking over randomly placed wooden planks and soft rubber mats.

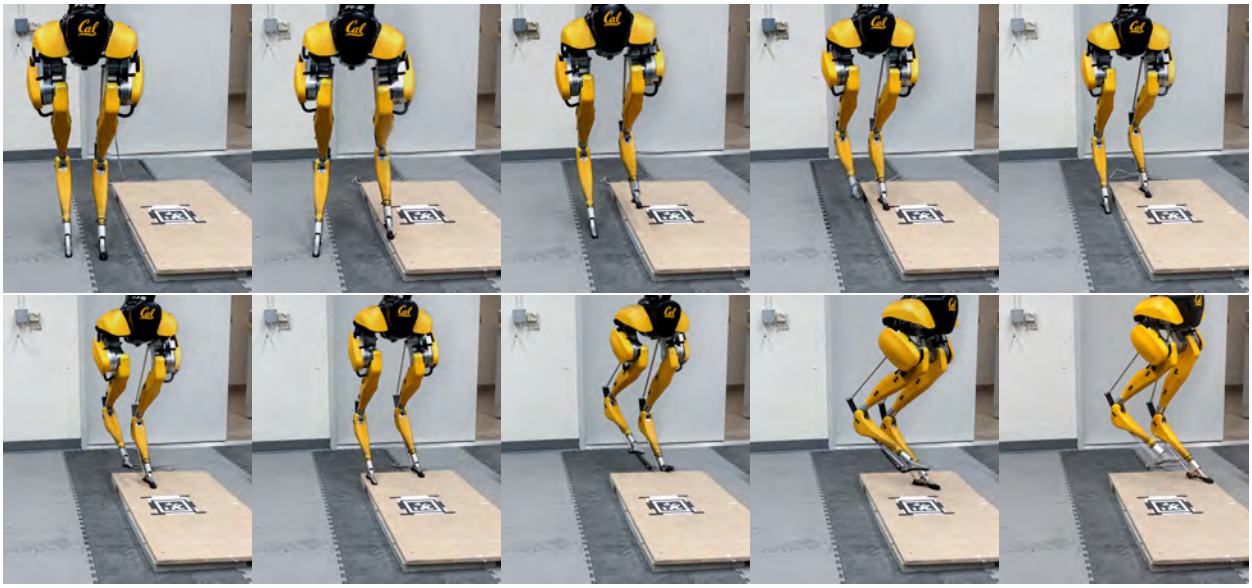


Figure 4.11: Snapshots of Cassie blindly sidestepping over a raised platform of height $8cm$.

Figure 4.12: Snapshots of outdoor walking experiments. Using the proposed approach, Cassie is able to negotiate a wide variety of terrain, including paved concrete roads and sidewalks, grass, and loose soft ground such as mulch. Cassie is also able to tackle large sloped terrains (uphill and downhill), as well as step across curbs. Note that in all experiments, Cassie is blind and cannot perceive the surrounding terrain.

method through several hardware experiments on the Cassie bipedal robot. In the next chapter, we extend the MPC framework developed in this chapter and the gait generation method for aperiodic locomotion developed in Chapter 3 to quadruped robots walking on discrete footholds using visual feedback from a depth camera.

Chapter 5

Vision-Aided Dynamic Quadrupedal Locomotion on Discrete Terrain using Motion Libraries

In this chapter, we present a framework rooted in control and planning that enables quadrupedal robots to traverse challenging terrains with discrete footholds using visual feedback. Nav-

Figure 5.1: The A1 quadruped robot walking over a random discrete terrain using our proposed approach.

igating discrete terrain is challenging for quadrupeds because the motion of the robot can be aperiodic, highly dynamic, and blind for the hind legs of the robot. Additionally, the robot needs to reason over both the feasible footholds as well as the base velocity in order to speed up or slow down at different parts of the discrete terrain. To address these challenges, we build an offline library of periodic gaits which span two trotting steps, and switch between different motion primitives to achieve aperiodic motions of different step lengths on a quadrupedal robot. The motion library is used to provide targets to a geometric model predictive controller which outputs the contact forces at the stance feet. To incorporate visual feedback, we use terrain mapping tools and a forward facing depth camera to build a local height map of the terrain around the robot, and extract feasible foothold locations around both the front and hind legs of the robot. Our experiments show a small scale quadruped robot navigating multiple unknown, challenging and discrete terrains in the real world.

Video of experiments can be found here: https://youtu.be/3HAUvSsQYjs

5.1 Introduction

Legged robots have the unique capability to traverse a wide variety of challenging and rough terrain, including terrains with gaps and discrete footholds. To navigate such terrain, a legged robot needs to precisely place its feet on feasible footholds while maintaining its overall stability. This requires planning over multiple footsteps and desired robot motion between the footsteps. For example, the robot might need to slow down and take a few steps on the same foothold before speeding up and stepping over a large gap. Moreover, for unknown discrete terrain, the robot needs to make these decisions in real-time while navigating; stopping might make the robot unstable and gaps harder to cross. This results in a high-dimensional and complex optimization problem with a limited computing time budget. Discrete and uneven terrains also present an additional challenge of controlling the robot, as such terrains can result in the robot pitching, rolling and experiencing high angular velocities.

Related Work

Legged locomotion on discrete terrains, such as across stepping stones, is an active area of research with methods ranging from reduced-order models to learning-based approaches. We summarize different research directions here:

Reduced Order Models: In [76], the authors propose a reduced order cart-pole model to generate gaits for a bipedal robot to walk on randomly placed stepping stones. [113] presents a method to regulate the center-of-pressure to guide the robot leg onto a discrete foothold. More recently, in [35], a reduced-order linear inverted pendulum model is presented to regulate the angular momentum about the stance foot at discrete impacts through the

vertical center of mass velocity. A QP-based controller is then used to track outputs for 2D bipedal robots to walk on discrete terrain.

Optimal Control: Optimization-based controllers such as Control Barrier Functions (CBFs) and Model Predictive Control (MPC) can enforce state and input constraints. In [101], a CBF-based approach regulates the foot positions of a bipedal robot around a nominal periodic gait, to step on discrete footholds. This method is extended in [106] to use a library of walking gaits. The work in [102, 104, 3] leverages the use of two-step periodic gaits, computed offline through trajectory optimization, to transition between gaits of different step lengths online. In [48], a multi-layered optimal control framework is presented that combines CBFs with MPC for precise foot placement over a planning horizon. Several other works [89, 69, 140, 46] have also explored using trajectory optimization for dynamic legged locomotion.

Reinforcement Learning: The work in [133] proposes to learn a high-level footstep planner, that takes in the local height map of the terrain as input, and outputs a sequence of desired footstep locations. A low-level joint controller is then learned to track these footsteps. In [144], the authors propose learning the desired accelerations for a centroidal model of a quadruped and use a heuristic approach to plan for footsteps on discrete terrain. The work in [142] proposes a curriculum with varying levels of difficulty to learn a policy for various bipedal robots to walk across stepping-stones. Several methods such as in [123, 147, 88] and [134] have also explored combining learning-based approaches, particularly for vision-based footstep planning along with model-based low-level joint control.

Approach and Primary Contributions

In this work, we study the problem of dynamic locomotion for quadrupedal robots across discrete terrain using visual feedback. Our primary contributions in this work are threefold. First, we extend our prior work for bipedal robots in [102, 104, 3], for solving footstep selection problem by building a library of *two-step periodic* gaits, to quadrupedal robot locomotion. By pre-computing an offline library of two-step periodic gaits, parametrized by the step lengths in the first and second steps, transition between different step lengths can be achieved online by switching between the different motion primitives. Moreover, with trajectory optimization tools such as [64], an offline library with several hundred gaits can be computed within tens of minutes. Unlike bipedal robots, however, additional kinematic constraints exist between the front and hind limbs of the quadruped. To overcome this, we propose to create a motion library of two-step-periodic *trotting* gaits comprising of four stance phases (equivalent to four bipedal steps). Next, to stabilize these gaits, we extend the proposed coordinate-free MPC presented in Chapter 4 to a quadrupedal robot.

Finally, using terrain mapping frameworks [40, 39] with a forward-facing depth camera, we incorporate visual feedback and experimentally validate our proposed approach on a `Unitree A1` quadrupedal robot to navigate across multiple unknown, challenging discrete terrains.

5.2 Hybrid Model of Trotting

In this section, we introduce the necessary background and notations for the robot dynamics model considered in our approach. Our formulation of the dynamics is derived from prior work on hybrid dynamics, as in [104]. We utlize the model of the A1 robot presented in Chapter 2. Below, we provide a hybrid model for a trotting gait.

Continuous Dynamics: As mentioned in Chapter 2, the dynamics of the continuous phases can be represented by the *Manipulator equations*:

$$\begin{aligned} D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) &= B\tau + J_{st}^T F_{st} \\ J_{st}\ddot{q} + \dot{J}_{st}\dot{q} &\equiv 0, \end{aligned} \tag{5.1}$$

where D is the inertia matrix, C the Coriolis terms, G gravitational terms, B a selection matrix. J_{st} denotes the contact Jacobian, F_{st} denotes the contact forces at the feet, and $\tau \in \mathbb{R}^{12}$ denotes the motor torques.

Impact Dynamics: The collision of the feet with the ground is modeled as an instantaneous rigid impact, and the post-impact velocities $\dot{q}^+$ is obtained by solving the linear system of equations

$$\begin{bmatrix} D(q) & -J_{st}^T(q) \\ J_{st}(q) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}^+ \\ F_{st} \end{bmatrix} = \begin{bmatrix} D(q)\dot{q}^- \\ \mathbf{0} \end{bmatrix}. \tag{5.2}$$

Hybrid Model: We model each trotting step with two alternating phases of double-support (DS), where the diagonal feet are in contact, and quadruple-support (QS), where all four feet are in contact, as illustrated in Fig. 5.2. Combining (5.1) and (5.2), we obtain a hybrid dynamical model for trotting as,

$$\begin{aligned} \Sigma_{ds}: \begin{cases} \dot{x} &= f_{ds}(x) + g_{ds}(x)\tau, \ x \notin \mathcal{S}_{ds\to qs} \\ x^+ &= \Delta_{ds\to qs}\left(x^-\right), \ x \in \mathcal{S}_{ds\to qs} \end{cases} \\ \Sigma_{qs}: \begin{cases} \dot{x} &= f_{qs}(x) + g_{qs}(x)\tau, \ (x,\tau) \notin \mathcal{S}_{qs\to ds} \\ x^+ &= \Delta_{qs\to ds}\left(x^-\right), \ (x,\tau) \in \mathcal{S}_{qs\to ds} \end{cases}, \end{aligned} \tag{5.3}$$

where $x := [q^T, \dot{q}^T]^T$ is the state of the robot, $f_{ds}(x)$, $g_{ds}(x)$ and $f_{qs}(x)$, $g_{qs}(x)$ denote the vector-fields in the DS and QS domains respectively and are obtained from (5.1). The switching surface $\mathcal{S}_{ds\to qs} := \{x \mid p_{sw}^z(x) = 0, \dot{p}_{sw}^z(x) < 0\}$ is defined to be the set of states where the vertical component of the swing foot position is zero and the vertical swing foot velocity is less than zero. $\mathcal{S}_{qs\to ds} := \{(x,\tau) \mid \lambda_c^z(x,\tau) = 0\}$ corresponds to the set of states and control inputs where the vertical ground reaction force at the stance feet $\lambda_c^z(x,\tau) \equiv 0$ (when the stance foot lifts off from the ground). The reset map $\Delta_{ds\to qs}$ is obtained from impact dynamics (5.2) and $\Delta_{qs\to ds} = \mathbb{I}$ is the identity operator.

5.3 Approach

We now present our proposed approach of trajectory optimization and model-based low-level robot control using geometric MPC. We begin by building a motion library consisting of gaits parametrized by step length and optimized to minimize the total energy over a step, subject to dynamics and periodicity constraints. The low-level controller takes optimized CoM trajectories and footstep locations from the gait library to generate desired joint torques applied to the robot. Lastly, we describe the localization and terrain mapping framework we use in real-world experiments.

Trajectory Optimization

In this section, we present a method to generate a motion library of trotting gaits that achieve foot placements of different step lengths. In particular, we obtain gaits that are 'two-step' periodic, such that the post-impact states of the robot after two trotting steps return to the initial states at the start of the first step. The gaits are parametrized by the step-lengths $l_0, l_1 \in \mathbb{R}^2$ as indicated in Fig. 5.2. The step lengths l_0 and l_1 each represent a pair of distances between the left and right pairs of feet. The goal of trajectory optimization is to find gait parameters $\gamma(l_0, l_1)$ for various step-length pairs to construct a library of gaits denoted by $\mathcal{G} := \{\gamma(l_0, l_1) \mid (l_0, l_1) \in \mathbb{L} \times \mathbb{L}\}$, where $\mathbb{L} := L \times L$ is a predefined set of step length pairs. Specifically, we choose $L = \{-0.2, -0.1, 0.0, 0.1, 0.2\}$m, with a total of 5^4 gaits in the library. $\gamma(l_0, l_1)$ comprises of the trajectory parameters for the base linear and angular velocities and body height and orientation, which serve as reference states for the MPC as detailed in Section 5.3.

We utilize the Direct Collocation method described in Chapter 2, which involves discretizing each phase in time by a specified number of nodes N [62], to minimize energy over the entire trajectory, subject to dynamics and additional constraints $c_i(x_i(t), \tau_i(t))$,

$$
\begin{aligned}
(x^*(\cdot), \tau^*(\cdot)) = \underset{x(t),\tau(t)}{\arg\min} \quad & \Sigma_i \int_0^T ||\tau(t)||_2^2 \, dt \\
\text{st. } x(t) = & \int_0^T f_i(x(t)) + g_i(x(t))\tau(t) dt, \\
& c_i(x(t), \tau(t)) \leq 0, \quad 0 \leq t \leq T, \ \forall i \in \mathcal{I}.
\end{aligned} \tag{5.4}
$$

Here, $\mathcal{I}$ denotes the set of all discrete phases, $c_i(x(t), \tau(t))$ encodes physical constraints such as state and input limits, friction constraints as well as periodicity and step length constraints. The desired gait parameters $\gamma(l_0, l_1)$ can then be extracted from the optimal state trajectories $x^*(\cdot)$. We use the open-source toolbox `C-FROST` [64] to model and solve the above optimization problem. We refer the reader to [104] for specific details on the trajectory optimization formulation.

Remark 5.1. *The generation of periodic gaits for quadrupeds with varying step lengths poses additional challenges and constraints compared to bipedal robots in [104]. These challenges*

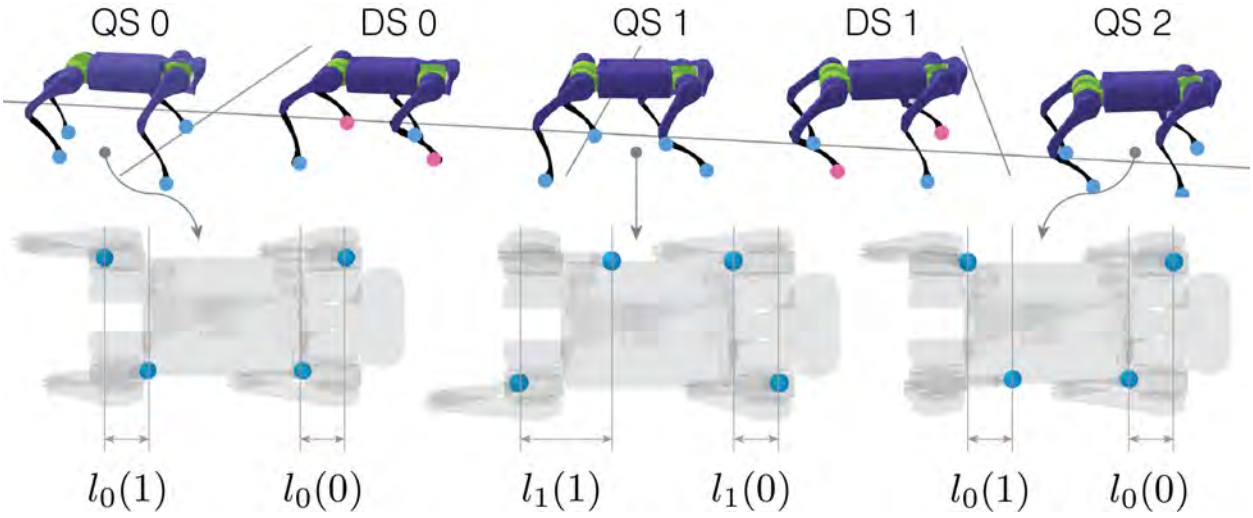


Figure 5.2: (Top) A trotting gait consists of two DS and QS domains as indicated by the figures marked from QS 0 to DS 1. For a 'one-step' periodic gait, the state at the beginning of the next step (QS 2) must coincide with the initial state of the previous step (QS 0). (Bottom) A 'one-step' periodic trotting gait is overly restrictive to capture all possible transitions between l_0 and l_1. When l_0 and l_1 are chosen independently, 'one-step' periodic solutions for a trotting gait do not exist (the configuration of the robot in QS 2 does not coincide with the configuration in QS 0). To obtain 'one-step' periodic trotting gaits, l_0 and l_1 are constrained by $l_0(0) + l_1(1) = l_0(1) + l_1(0)$. A 'two-step' periodic trotting gait used in this chapter consists of four DS and four QS phases and provides sufficient flexibility to choose l_0 and l_1 independently.

arise from kinematic constraints between the left and right limbs. In particular, to independently choose the step-length pairs l_0 and l_1, and to also induce periodicity constraints, we consider 'two-step' periodic trotting gaits which comprise of four QS and four DS phases (as opposed to 'one-step' periodic gaits). To visualize the requirement of a 'two-step' periodic gait, we first consider a 'one-step' periodic gait. When l_0 and l_1 are chosen independently, the net displacements of the left and right pairs of feet during a step is not necessarily equal. As a result, the four feet of the robot can move closer together or further apart during a step, resulting in a gait that is not *periodic. This is illustrated in Fig. 5.2, where the chosen step-lengths $l_0(0) = l_0(1) = l_1(0) = 0.1m$ and $l_1(1) = 0.2m$ result in the four feet moving closer together at the end of a step (the configuration in QS 2 does not coincide with QS 0). Additional constraints on l_0 and l_1 must be placed to obtain 'one-step' periodic gaits. In particular, the net displacements of the left and right pairs of feet during a step must be equal. This is captured by the constraint $l_0(0) + l_1(1) = l_0(1) + l_1(0)$. A 'two-step' periodic gait, on the other hand, consists of two additional DS and QS phases. By appropriately choosing the step lengths in these phases, the net displacements of the left and right pairs of feet in two*

(a) Aligned terrain

(b) Staggered Terrain

(c) Random Terrain

(d) A1 robot on discrete terrain, visualized from real-world data.

Figure 5.3: Different terrains tested in our experiments, and visualization of a local map built on the robot.

steps can be made equal while still being able to choose l_0 and l_1 independently.

Footstep Planning and Gait Selection

Once we have created the gait library, we can extract desired gait variables by querying motions that satisfy the environment foothold constraints and that start from the current state of the robot.

Footstep Planning: To choose the desired foothold location, we first query the gait library to obtain a nominal foothold location based on the current configuration and center-of-mass velocity of the robot as well as a nominal desired center-of-mass velocity. Similar to [144], we then chose the desired step length closest to the nominal foothold location and on the feasible terrain.

Gait Selection: Given the current state of the robot and the feasible footstep map, we extract a gait from the library based on the current step-length l_0 and the desired step-length l_1 through bi-linear interpolation of the gait library [104]. This returns the desired states for a reduced-order rigid-body model considered in the MPC controller. This update allows us to re-target the desired CoM velocities to be consistent with the desired step lengths.

Geometric Model Predictive Control for Stance Legs

We utilize the Geometric MPC developed in Chapter 4 to obtain the stance leg torques. The desired reduced-order model states are obtained from the gait library depending on the chosen foothold location.

Swing leg control

For the swing-leg control, we implement an output PD controller to follow the desired foot trajectory,

$$\tau_{sw} = J_{sw}^T \left(-K_p^{sw}(p^{sw} - p_d^{sw}) - K_d^{sw}(\dot{p}^{sw} - \dot{p}_d^{sw}) \right). \tag{5.5}$$

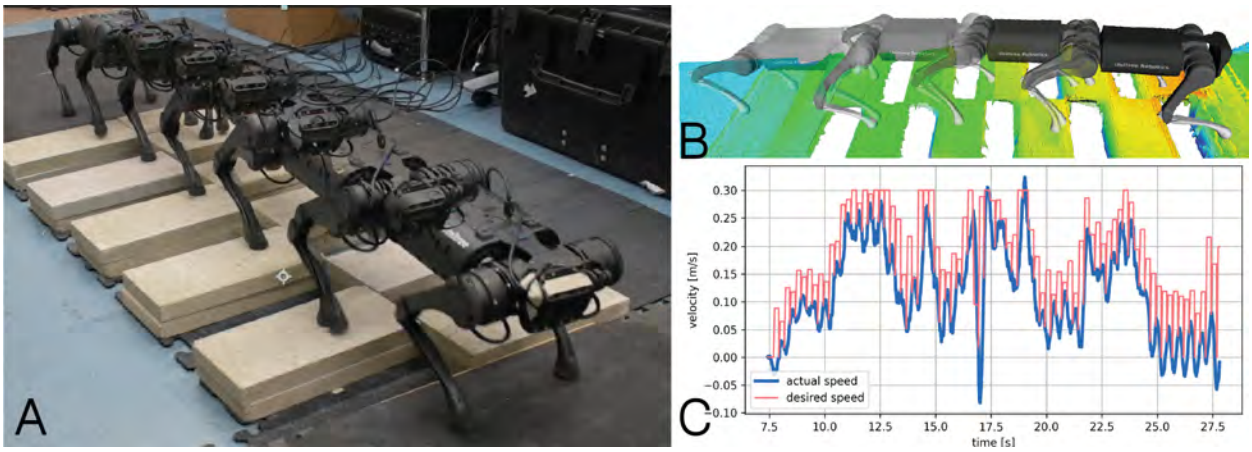


Figure 5.4: (A) Snapshots of the robot, (B) visualization of the terrain map illustrating the foot-placement of the robot on the stepping stones, and (C) forward velocity of the robot from real world data.

The desired foot trajectories are parametrized by Bézier polynomials such that the initial desired position is located at the true foot position at the start of a swing phase, and the final position based on the desired step-length, obtained through a footstep planner.

Localization and Mapping

We use a forward-facing depth camera to perceive the terrain, which makes it challenging to pick feasible footsteps for hind limbs. This requires building a local map of the robot by fusing a history of depth images that the robot sees, as well as the estimate of its own inertial pose, in order to build a local map of the terrain around the robot. We fuse two libraries to achieve this:

Localization: We implement contact-aided invariant EKF from [60] to localize the robot in the world. The binary contact information required by the EKF, is obtained through contact force sensors located at the feet.

Mapping: We utilize the probabilistic robot-centric mapping framework developed in [39, 40] to obtain a height-map of the terrain. Localization estimates from the EKF and depth images from the robot camera are fused by the mapper to build a local map around the robot. We distinguish between stepable and un-stepable terrain based on the height and normal direction and add a 5cm threshold at the edges between these regions to account for inaccuracies in the foot placement controller and state estimation. The local map based on previously observed depth images is used for picking footholds for the hind limbs, eliminating the issue of lack of perception towards the back of the robot. In the future, this localization and mapping framework can be replaced by learning-based approaches, which can automatically build a history of feasible footholds.

	Heuristic	Jacobian with Gait library	GVMPC (ours)
Aligned	0/3	1/3	3/3
Staggered	0/3	0/3	2/3

Table 5.1: Success rates of the three controllers on different terrains over 3 hardware runs on the A1 robot. Our approach (GVMPC) outperforms the baseline controllers on aligned and staggered terrains. The failure mode of GVMPC on the staggered terrain is due to the stance foot slipping at the edge of the terrain. All controllers use the same vision feedback.

5.4 Experiments

We demonstrate the robustness of our approach on the `Unitree A1` quadruped (Fig. 5.3d) on a diverse set of terrains with discrete footholds (Fig. 5.3). These terrains consist of concrete blocks of size $6'' \times 16''$. The gap lengths between blocks range between $7cm$ and $18cm$, and can be in different orientations. The robot is required to move forwards while avoiding the gaps. The gap lengths are the same for the left and right legs in the *aligned* terrain (Fig 5.3a), different in *staggered* terrain (Fig. 5.3b) or random in *random* terrain (Fig. 5.3c). Random terrains pose additional constraints on the lateral foot placement. The nominal commanded velocity is $0.25m/s$ and is updated by the gait library based on the desired foot position.

First, we compare our approach to the baseline in [144] (*Heuristic*), which uses the closest stepping location to a Raibert-like footstep and a Jacobian linearized rigid-body model without any motion libraries. We use the implementation in [30]. This baseline tests the robustness of our approach over other heuristic approaches from literature shown on discrete terrain walking. Next, we incorporate motion libraries to this baseline stance controller (*Jacobian with Gait Library*) and query CoM velocity and footstep location from the motion library. This experiment illustrates the need for geometric MPC on uneven terrain. Together, these experiments study the performance of our whole framework against heuristic approaches from literature, as well as the importance of geometric MPC on uneven terrain. Table 5.1 summarizes the success rates of the three controllers on different terrains over three hardware runs on the A1 robot.

We observe that the Heuristic approach is not able to successfully navigate any of the terrains. This is because the robot needs to speed up or slow down depending on the size of the gaps. Since the Heuristic baseline only changes the footstep position but maintains a constant CoM velocity, it is easily destabilized when walking over large gaps. The second baseline, which uses the gait library, is able to cross the aligned terrain in 1 trial but fails on the staggered terrain. The Jacobian linearized model does not regulate the CoM velocities and orientations well in our experiments, causing the robot to go unstable. The instability is caused more in the lateral direction, pointing towards foot placement feedback going unstable

due to lateral and roll angular velocities. The failure mode in the staggered experiment for the GVMPC is due to the stance foot slipping at the edge of the terrain.

Additionally, we conducted two runs of experiments on the random terrain, which is significantly more complicated and needs precise foot placement, and CoM position and orientation planning. Our approach is able to navigate this terrain in 2/2 experiments. These experiments demonstrate that our proposed Geometric MPC is able to robustly stabilize the robot from a more extensive set of states around the desired trajectory.

Robustness to missed steps

It is possible for either the perception system or state estimator to fail. Additionally, the foot can slip around the edges of a stepping stone. In such scenarios, it is important for the locomotion controller to recover from such failures. In Figure 5.5, we illustrate one such instant where the foot misses the stepping stone due to an error in the state-estimator, but is able to recover from such disturbances.

5.5 Chapter Summary

In this chapter, we present a planning and controls framework for vision-aided navigation for quadrupedal robots in challenging terrain. The method leverages offline computation of a library of gaits parametrized by step lengths and an on-board geometric MPC that takes into account the underlying geometric structure of the reduced-order rigid-body model, in both the discretization and linearization of the dynamics. Combining our proposed method with existing state-of-the-art tools for localization and mapping, we demonstrate the successful implementation of quadruped locomotion on discrete terrain. While the primary focus of this work is locomotion on discrete terrain with varying step lengths, our method can potentially be extend to terrains with varying step widths as well as for turning on discrete terrain.

A drawback of our method is it requires the elevation map to be segmented into *stepable* and *un-stepable* regions, which is currently achieved by thresholding the height and normal vector direction of the elevation map and providing a safety margin from the edge of a stone. For small-scale robots such as the A1, a gap between two stepping stones can be occluded in the resulting depth image due to the low nominal height of the robot. This can result in inaccurate segmentation of the height map if the threshold and safety margins are not chosen appropriately. Our approach also utilizes an invariant EKF to estimate the position of the robot on the elevation map. Any drift in the position estimate between steps can lead to inaccurate foot placement. The EKF on the A1 robot was particularly challenging to tune due to its compliant feet and the behavior of the contact sensor located at the foot on different surfaces.

This brings us to the end of Part I of the thesis, where we developed model-based feedback controllers to achieve agile and dynamic legged locomotion on several robotic platforms. One

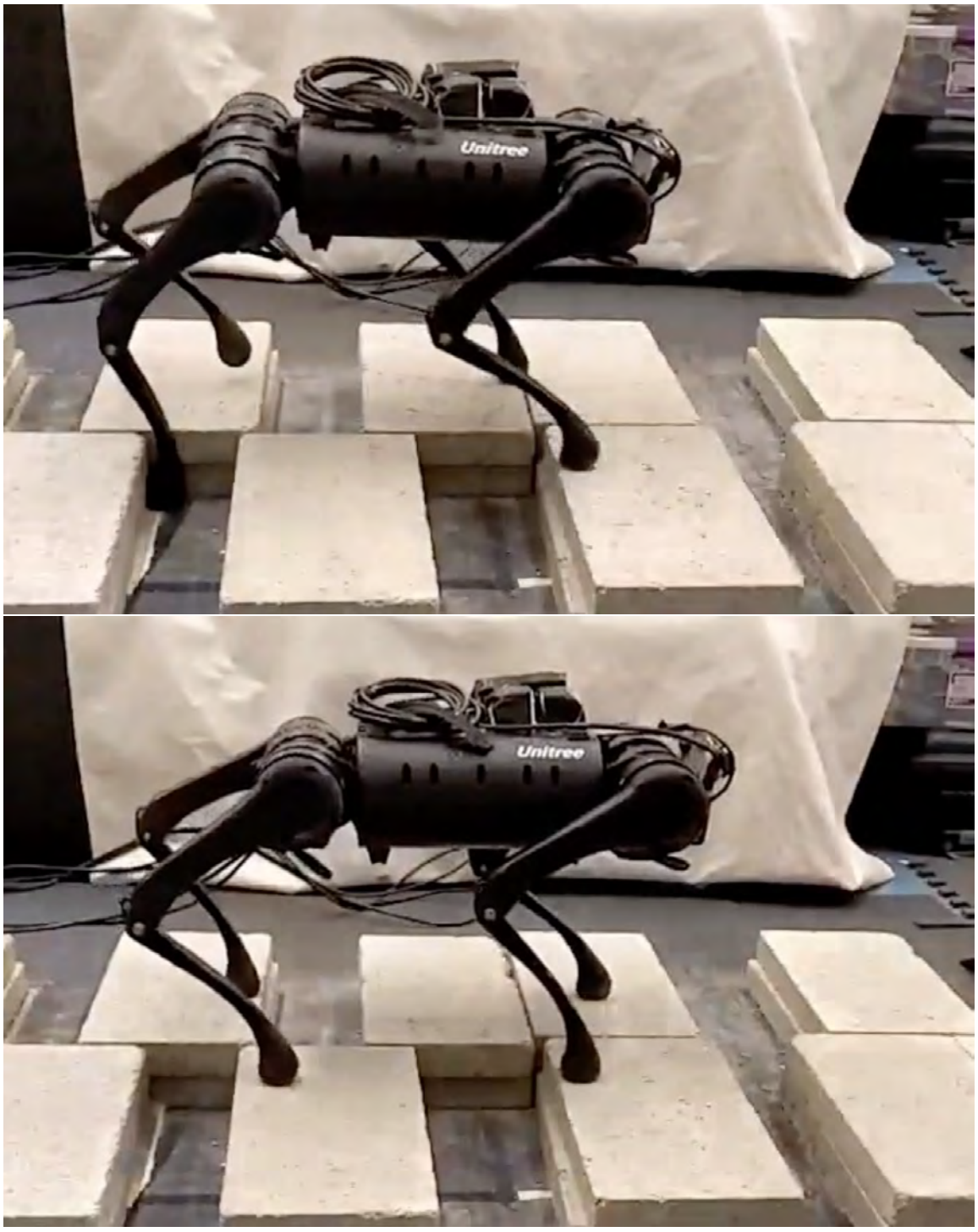


Figure 5.5: Robot recovering from a missed foot placement due to an error in the state estimate.

of the major limitations of model-based methods is the requirement for an accurate representative model of the system. Additional uncertainties can arise from environmental factors such as reduced friction or external force perturbations. In such scenarios, the performance of model-free methods can significantly degrade. Recent developments in model-free RL methods have shown impressive results on systems with highly uncertain dynamics. Such methods, however, are highly sample-inefficient. In the next part of the thesis, inspired by model-based control literature, we develop novel reward functions for RL problems that include CLF and CBF terms. By including these terms, our method can learn safe, stabilizing controllers for systems with high model uncertainty with only seconds or a few minutes of training data.

Part II

Combining Model-Based Control with Model-Free Policy Optimization

Chapter 6

Combining Model-Based Design and Model-Free Policy Optimization to Learn Safe, Stabilizing Controllers

This chapter introduces a framework for learning a safe, stabilizing controller for a system with unknown dynamics using model-free policy optimization algorithms. Using a nominal dynamics model, the user specifies a candidate Control Lyapunov Function (CLF) around the desired operating point, and specifies the desired safe-set using a Control Barrier Function (CBF). Using penalty methods from the optimization literature, we then develop a family of policy optimization problems which attempt to minimize control effort while satisfying the pointwise constraints used to specify the CLF and CBF. We demonstrate that when the penalty terms are scaled correctly, the optimization prioritizes the maintenance of safety over stability, and stability over optimality. We discuss how standard reinforcement learning algorithms can be applied to the problem, and validate the approach through simulation. We then illustrate how the approach can be applied to a class of hybrid models commonly used in the dynamic walking literature, and use it to learn safe, stable walking behavior over

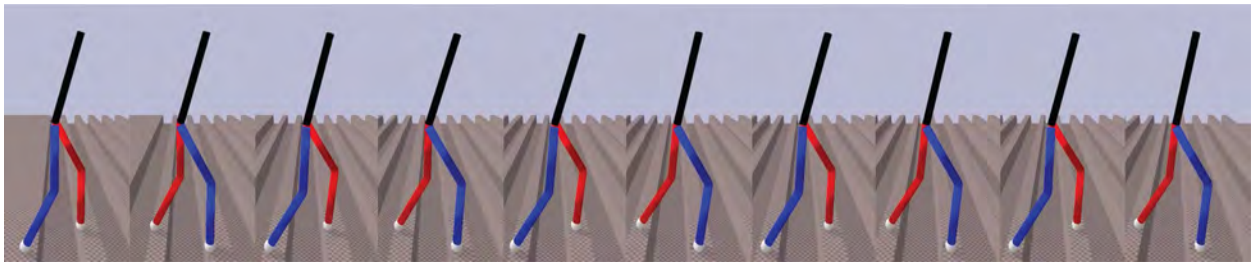


Figure 6.1: Snapshots of the RABBIT walking across a field of stepping stones using our proposed approach. The mass and inertia of the links are scaled by three times introducing significant model uncertainty.

a randomly spaced sequence of stepping stones.

6.1 Introduction

Following recent empirical successes from the reinforcement learning (RL) literature ([81]), there has been a renewed interest in data-driven methods for controller design in the case of model uncertainty ([17, 8]). However, despite the flexibility of model-free approaches, these methods are known to suffer from poor sample complexity since they do not take advantage of known structural properties of the control system. Moreover, the literature currently lacks constructive methods for designing learning problems which give the system designer fine-grained control over potentially competing global objectives, such as the rate of convergence to a desired operating point or the avoidance of an unsafe region of the state-space.

Fortunately, modern model-based control theory has developed many tools such as Control Lyapunov Functions (CLFs; [126]) and Control Barrier Functions (CBFs; [9]) which allow the system designer to constrain the pointwise closed-loop behavior of a given control system to ensure desired global properties (stability and safety, respectively) are achieved. When an accurate dynamics model is available, online optimization can be used to satisfy these pointwise constraints while minimizing a cost, such as control effort [9]. In effect, these approaches reduce the satisfaction of challenging global objectives to simple local decisions from the perspective of controller synthesis.

This chapter takes preliminary steps towards extending this design philosophy to the model-free setting by introducing a framework for systematically designing policy optimization problems over a parameterized learned controller which enforces a hierarchy of user-specified constraints on the closed-loop dynamics. To make the framework explicit, we focus on learning safe, stabilizing controllers using CLFs and CBFs and choose to prioritize safety over stability. We focus on the regime where the system designer has access to a dynamics model which may be highly inaccurate but is assumed to at least capture basic structural information about the real world plant. The model is used to construct a candidate CLF and CBF for the plant and a family of policy optimization problems are formulated which use penalty terms to discourage violations of the pointwise constraints imposed by these functions. This allows the system designer to carefully constrain the desired closed-loop behavior for the learned controller while also allowing for additional performance terms, such as minimizing control effort.

Our theoretical results demonstrate how to scale the penalty terms to control violations of the constraints and appropriately prioritize safety over stability and stability over performance. We first introduce the approach for classical control systems but then demonstrate how to extend the approach to the hybrid case via an application to a class of hybrid models which are frequently used in the dynamic walking literature ([53]). We discuss how to synthesize numerical approximations to the family of learning problems which can be solved using standard machine learning techniques, including state of the art reinforcement learning

algorithms. Simulation experiments are provided for both the continuous and hybrid cases, which demonstrate that our method is able to effectively learn safe, stabilizing controllers in the face of large amounts of dynamics uncertainty. We can reliably solve the policy optimization problems formulated over these systems using only a few minutes or even seconds of simulated data, representing a sharp increase in the sample efficiency usually found in the reinforcement learning literature [68, 81]. We conjecture that this is due to the large amount of structure embedded in the learning problem through the incorporation of CLF and CBF constraints, which reduce the search for an optimal safe, stabilizing controller to a set of local criteria at each point in the state space.

Related Work: The unification of Control Barrier Functions and Control Lyapunov Functions to synthesize safe, stabilizing controllers was first proposed in [9] using online quadratic programming. In the case of model uncertainty, robust formulations have been proposed ([98]). Learning based methods using supervised learning [128] or reinforcement learning [27] to learn the uncertain dynamics terms in the quadratic program have also been considered. These can be thought of as indirect learning methods, since they still require solving an optimization problem involving the learned components to calculate the desired controller. The primary downside of each of these approaches is that if the optimization is infeasible at a particular point then the control strategy will generally be undefined, which can be particularly difficult to rule out when learning unknown dynamics.

Building on our previous work [136], we introduce a framework for directly learning a safe, stabilizing controller for the system using model-free policy optimization algorithms. By directly learning the desired controller, our approach removes the need for solving a real-time optimization problem involving a potentially complex learned component, which may take a non-trivial amount of time to process during real-time applications. At points where it is infeasible to satisfy the desired constraints, our method provides a "best effort" control strategy which satisfies the constraints to the greatest degree possible, bypassing issues of feasibility.

6.2 Control Lyapunov Functions and Control Barrier Functions

Throughout most of the chapter we will consider control-affine systems of the form

$$\dot{x} = f(x) + g(x)u, \tag{6.1}$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ the input. We assume that $f\colon \mathbb{R}^n \to \mathbb{R}^n$ and $g\colon \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuously differentiable.

Control Lyapunov Functions: Control Lyapunov Functions (CLFs; [126]) are commonly used to construct a controller which stabilizes a system to either a desired operating point or a desired subset of the state-space [12]. Specifically, we say that the continuously

differentiable function $V\colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a *Control Lyapunov Function* if

$$\inf_{u\in\mathbb{R}^m} \nabla V(x)[f(x)+g(x)u] \leq -\sigma(x) \quad \forall x \in \mathbb{R}^n \setminus \{0\}, \tag{6.2}$$

where $\sigma\colon \mathbb{R}^n \to \mathbb{R}$ specifies a desired pointwise rate of decay. Here, V and σ are both assumed to be positive definite, and V is additionally assumed to be radially unbounded. Under these conditions, V can be viewed as a generalized energy function for (6.1), and condition (6.2) ensures that there exists a control which drives the system state asymptotically to the origin.

Control Barrier Functions: Inspired by barrier functions from the optimization literature, the level sets of Control Barrier Functions (CBFs) encode user-specified safety constraints. Many classes of CBFs have been proposed in recent years [9, 145, 97], but for concreteness throughout the chapter, we will use the class proposed in [145]. Specifically, we say that the function $h\colon \mathbb{R}^n \to \mathbb{R}$ is a *Control Barrier Function* if

$$\sup_{u\in\mathbb{R}^m} \nabla h(x)[f(x)+g(x)u] \geq -\alpha(h(x)) \quad \forall x \in \mathcal{C}, \tag{6.3}$$

where $\mathcal{C} = \{x \in \mathbb{R}^n\colon h(x) \geq 0\}$ is a safe set specified by the 0-super-level set of h, and $\alpha\colon (-b, a) \to \mathbb{R}$, with $a, b > 0$, is locally Lipschitz, strictly increasing, and $\alpha(0) = 0$.

It is natural to then search for a Lipschitz continuous control law which satisfies the pointwise constraints in (6.2) and (6.3) simultaneously. One candidate control law is given by solving a pointwise quadratic program (QP):

$$\begin{aligned} u^*(x) = \arg\min_{u\in\mathbb{R}^m} \ & \|u\|_2^2 \\ \text{s.t. } & \nabla V(x)[f(x)+g(x)u] \leq -\sigma(x) \\ & \nabla h(x)[f(x)+g(x)u] \geq -\alpha(h(x)) \end{aligned} \tag{6.4}$$

which aims to minimize control effort while satisfying the two pointwise constraints. Unfortunately, even if V and h are an actual CLF and CBF for the system, it may be impossible to satisfy both constraints simultaneously leading to infeasibility issues. A common heuristic is to add slack terms to one or both of the constraints to ensure feasibility of the problem at the cost of some violation of the constraints [9].

6.3 Learning Safe, Stabilizing Controllers for Uncertain Systems

While control laws similar to (6.4) have been successfully applied in a number of applications they have several practical limitations. Most importantly, these approaches require that an exact dynamics model is available to ensure that the pointwise constraints in (6.4) can be satisfied on the real-world system. Secondly, the infeasibility issues mentioned above mean that the controller may be undefined at certain points in the state-space, which can be

highly problematic during real-time operation. This motivates the method detailed below, which uses a candidate CLF and CBF to learn an optimal safe, stabilizing controller for an uncertain system using data collected from the plant. The method prioritizes satisfaction of the CBF constraint over the CLF constraint and removes the need for real-time optimization.

Specifically we will seek to safely stabilize the *plant*

$$\dot{x} = f_p(x) + g_p(x)u, \tag{6.5}$$

whose dynamics are unknown. We will also assume that a nominal *dynamics model* for the plant is available:

$$\dot{x} = f_m(x) + g_m(x)u. \tag{6.6}$$

We assume that the dynamics model has been used to synthesize a candidate CLF V (and rate σ) and CBF h (and rate α) for the unknown plant. Even though the dynamics of the plant are unknown, it is often reasonable to assume that the model captures enough basic structural information about the plant to guarantee that these functions are also a valid CLF and CBF for the real-world system. For example, in our simulated applications we design the candidate CLF using feedback linearization, which is guaranteed to be a CLF for the true system as long as the relative degree of the plant matches that of the model, a relatively weak assumption.

The learned controller $\hat{u}\colon \mathbb{R}^n \times \Theta \to \mathbb{R}^m$ is of the form

$$\hat{u}(x, \theta) = u_m(x) + \tilde{u}(x, \theta). \tag{6.7}$$

Here, $u_m\colon \mathbb{R}^n \to \mathbb{R}^m$ is a nominal controller supplied by the system designer which is derived from the nominal dynamics model, and $\tilde{u}\colon \mathbb{R}^n \times \Theta \to \mathbb{R}^m$ is a learned augmentation. The learned parameters $(\theta_1, \ldots, \theta_p) \in \Theta \subset \mathbb{R}^p$ are to be trained so as to select the optimal safe, stabilizing controller for the system.

Assumption 6.1. *The learned controller $\hat{u}\colon \mathbb{R}^n \times \theta \to \mathbb{R}^m$ is continuously differentiable in both of its arguments.*

Assumption 6.2. *The set of learned parameters Θ is a compact convex set.*

Our primary goal is to find a controller which satisfies the following infinite dimensional constraints, when possible:

$$\underbrace{-\nabla h(x)[f_p(x) + g_p(x)\hat{u}(x, \theta)] - \alpha(h(x))}_{\Delta_1(x,\theta)} \leq 0 \quad \forall x \in \mathcal{C}, \tag{6.8}$$

$$\underbrace{\nabla V(x)[f_p(x) + g_p(x)\hat{u}(x, \theta)] + \sigma(x)}_{\Delta_2(x,\theta)} \leq 0 \quad \forall x \in \mathcal{C}. \tag{6.9}$$

Here, the set $\mathcal{C} \subset \mathbb{R}^n$ is the safe-set defined by the 0-super-level set of h. In words, we want to train a controller $\hat{u}(\cdot, \theta)\colon \mathbb{R}^n \to \mathbb{R}$ which satisfies the safety and stabilization constraints

that the chosen CBF and CLF impose on the real-world system. We make the following assumption:

Assumption 6.3. *The safe set $\mathcal{C}$ is compact.*

Since it may not be possible to learn a controller which satisfies both sets of constraints simultaneously, our learning framework must be flexible enough to prioritize the safety objective over the stabilization objective when necessary. While we do not know the terms in $\Delta_1(x,\theta)$ and $\Delta_2(x,\theta)$ since the dynamics of the plant are unknown, these terms can be calculated for different values of $x \in \mathcal{C}$ and $\theta \in \Theta$ if measurements of $\dot{V}$ and $\dot{h}$ are available when collecting data from the plant.

In order to enforce these constraints while minimizing control effort, we will solve optimizations of the form

$$\mathbf{P}^{(\lambda_1,\lambda_2)}: \quad \min_{\theta\in\Theta} \mathbb{E}_{x\sim X} L^{(\lambda_1,\lambda_2)}(x,\theta),$$

where

$$L^{(\lambda_1,\lambda_2)}(x,\theta) = \|\hat{u}(x,\theta)\|_2^2 + \lambda_1 H(\Delta_1(x,\theta)) + \lambda_2 H(\Delta_2(x,\theta)),$$

the hinge map $H\colon \mathbb{R} \to \mathbb{R}$ is defined by $H(y) = \max\{0, y\}$ for each $y \in \mathbb{R}$, and the probability distribution $X\colon \mathcal{C} \to [0,1]$ is supported on $\mathcal{C}$. Here, X is understood to be the distribution of states visited when collecting samples from the real world plant during the learning process, and $\lambda_1, \lambda_2 \geq 0$ are penalty parameters to be chosen later.

Remark 6.1. *The requirement that X is supported on all of $\mathcal{C}$ is analogous to the persistency of excitation conditions found in the adaptive control literature [119], and ensures that the data is "rich enough" so that the correct controller is learned. Note that under this assumption the penalty terms $\mathbb{E}_{x\sim X}\lambda_1 H(\Delta_1(x,\theta))$ and $\mathbb{E}_{x\sim X}\lambda_2 H(\Delta_2(x,\theta))$ are positive if and only if the safety and stability constraints are violated, respectively, at some point $x \in \mathcal{C}$. Thus this richness requirement guarantees that violations of the pointwise constraints are appropriately penalized by the optimization. The theoretical guarantees we provide below are algorithm agnostic, and seek to characterize the global optimizers of the problem. Future work will seek to bound the performance of specific machine learning algorithms used to solve $\boldsymbol{P}^{(\lambda_1,\lambda_2)}$, which generally come in the form of probabilistic guarantees.*

Theoretical Analysis: We now demonstrate that violations of the safety and stability constraints can be decreased to a pre-specified tolerance by scaling the penalty terms appropriately. For simplicity, we assume there exists at least one set of parameters which satisfies the safety constraint:

Assumption 6.4. *There exists $\theta^* \in \Theta$ such that for each $x \in \mathcal{C}$ we have $\Delta_1(x,\theta^*) \leq 0$.*

Next, we build up some additional notation to simplify the statement of our theoretical results. First, define the maps $M_u, M_1, M_2 \colon \Theta \to \mathbb{R}_{\geq 0}$ by

$$M_u(\theta) = \mathbb{E}_{x\sim X}\|\hat{u}(x,\theta)\|_2^2,$$

$$M_1(\theta) = \mathbb{E}_{x\sim X} H(\Delta_1(x,\theta)),$$
$$M_2(\theta) = \mathbb{E}_{x\sim X} H(\Delta_2(x,\theta)).$$

For each chosen parameter $\theta \in \Theta$, $M_u(\theta)$ captures total energy exerted by the corresponding controller across the safe set, $M_1(\theta)$ is the extent to which the CBF constraint is violated, and $M_2(\theta)$ is the extent to which the CLF constraint is violated. Next, for each $\varepsilon_1 \geq 0$ define

$$\Theta_{\varepsilon_1} = \{\theta \in \Theta \colon M_1(\theta) \leq \varepsilon_1\},$$

which is the set of parameters for which the total violation of the CBF constraint is less than ε_1. We also define

$$\tilde{M}_2 = \min_{\theta\in\Theta_0} M_2(\theta), \tag{6.10}$$

which is the smallest extent to which the CLF constraint can be violated, subject to exact satisfaction of the CBF constraint, and is the ideal amount of violation of the CLF constraint that can be returned by our optimization problem. We then define for each $\varepsilon_1, \varepsilon_2 \geq 0$

$$\Theta_{\varepsilon_1,\varepsilon_2} = \{\theta \in \Theta_{\varepsilon_1} \colon M_2(\theta) \leq \tilde{M}_2 + \varepsilon_2\},$$

which is the set of parameters corresponding to learned controllers which violate the CBF and CLF constraints no more than $\varepsilon_1 \geq 0$ and $\varepsilon_2 \geq 0$ more than their ideal values.

We now present our first result, whose proof can be found in the Appendix:

Theorem 6.1. *There exist constants, $C_1, C_2, C_3 \geq 0$ such that if $\lambda_1 \geq \frac{C_1\lambda_2 + C_2}{\varepsilon_1}$ and $\lambda_2 \geq \frac{C_3}{\varepsilon_2}$ then each global optimizer θ^* of $\boldsymbol{P}^{(\lambda_1,\lambda_2)}$ satisfies $\theta^* \in \Theta_{\varepsilon_1,\varepsilon_2}$.*

The result indicates that if we choose $\lambda_2 \gg 0$ and $\lambda_1 \gg \lambda_2$ our optimization correctly enforces safety over stability, satisfying the two constraints to the desired tolerances. Within the set of desired controllers specified by $\Theta_{\varepsilon_1,\varepsilon_2}$, the optimization is then left to reduce the amount of control effort required to achieve these objectives. However, driving both tolerances to zero requires taking $\lambda_1, \lambda_2 \to \infty$.

One practical approach for ensuring exact satisfaction of the safety constraint for a finite value of the multipliers is to add a small amount of extra conservativeness to the pointwise CBF constraint. Specifically, letting $\Delta_1^\delta(\theta, x) = \Delta_1(x,\theta) + \delta$ for some small parameter $\delta > 0$, one can replace $\Delta_1(x,\theta)$ with $\Delta_1^\delta(x,\theta)$ in the loss $L^{(\lambda_1,\lambda_2)}(x,\theta)$. Due to the continuity of the problem data, driving $\mathbb{E}_{x\sim X} H(\Delta_1^\delta(\theta,x))$ to be sufficiently small (which can be done with finite values of λ_1) will ensure exact satisfaction of the original CBF constraint. A forthcoming article will address this point in greater detail.

However, the attractive properties mentioned above only apply to the global minimizers of $\mathbf{P}^{(\lambda_1,\lambda_2)}$, which in general will be non-convex, meaning that in practice only local minimizers to the problem can be found using common incremental machine learning algorithms. Thus, we seek conditions on the structure of the learned controller which ensure that the optimization problem is convex. Specifically, we analyze the case where the learned portion of the controller is of the form

$$\tilde{u}(x,\theta) = \sum_{k=1}^{p} \theta_k u_k(x), \tag{6.11}$$

where $\{u_k\}_{k=1}^p$ is a set of features.

Theorem 6.2. *Suppose that the learned augmentation in (6.7) is of the form (6.11), and that the set $\{u_k\}_{k=1}^p$ is linearly independent. Then $\boldsymbol{P}^{(\lambda_1,\lambda_2)}$ is strongly convex.*

Many well-known bases such as radial basis functions [117] or polynomials can be used to recover any continuous function up to a desired degree of accuracy by including enough terms in the expansion. It is an important matter for future work to include these methods in our framework, as it would enable users to design networks for the learned controller which are guaranteed to be able to satisfy the CLF and CBF constraints to a desired degree of accuracy. However, function approximation schemes of the form (6.11) may require a prohibitive number of bases elements to ensure that the desired function is accurately reconstructed in high dimensions. Thus, in practice, more compact function approximators such as feed-forward neural networks must be used in high dimensions. Unfortunately, such networks generally lead to non-convexities in $\mathbf{P}^{(\lambda_1,\lambda_2)}$.

Numerical Implementation via RL: In practice, our method uses finite difference approximations to $\dot{h}$ and $\dot{V}$ to compute the terms in Δ_1 and Δ_2, and then solves the resulting approximations to $\mathbf{P}^{(\lambda_1,\lambda_2)}$ using standard model-free reinforcement learning algorithms.

Specifically, we will assume that during the learning process the learned controller is sampled every $\Delta t > 0$ seconds, and will let $t_k = k\Delta t$ for $k \in \mathbb{N}$ denote the sampling instances. When the control $\hat{u}(x(t_k),\theta)$ is applied over the interval $[t_k, t_{k+1}]$ we have

$$\Delta_1(x(t_k)),\theta) = \underbrace{-\frac{h(x(t_{k+1})) - h(x(t_k))}{\Delta t} - \alpha(h(x(t_k)))}_{=:\tilde{\Delta}_1(x,\theta)} + O(\Delta t^2),$$

$$\Delta_2(x(t_k)),\theta) = \underbrace{\frac{V(x(t_{k+1})) - V(x(t_k))}{\Delta t} + \sigma(x(t_k))}_{=:\tilde{\Delta}_2(x,\theta)} + O(\Delta t^2).$$

Thus, for small $\Delta t > 0$ we approximate $L^{(\lambda_1,\lambda_2)}$ with

$$\tilde{L}^{(\lambda_1,\lambda_2)}(x,\theta) = \|\hat{u}(x,\theta)\|_2^2 + \lambda_1 H(\tilde{\Delta}_1(x,\theta)) + \lambda_2 H(\tilde{\Delta}_2(x,\theta))$$

and define the following reinforcement learning problem:

$$\tilde{\mathbf{P}}^{(\lambda_1,\lambda_2)}: \min_{\theta\in\Theta} E_{x_0\sim X}\left[\sum_{k=0}^{N} \tilde{L}^{(\lambda_1,\lambda_2)}(x_k,\theta)\right] \quad \text{s.t. } x_{k+1} = x_k + \int_{t_k}^{t_{k+1}} \left[f(x(t)) + g(x(t))\hat{u}(x_k,\theta)\right] dt \tag{6.12}$$

Here, $N \in \{1, 2, \dots\}$ is the length of the rollout for each experiment on the plant. Note that this is a standard form for reinforcement learning problems, which can be solved using any off-the-shelf algorithm. Future work will seek to provide correctness guarantees when specific learning algorithms are used to solve these approximations.

6.4 Simulations

Double Pendulum With Safety Constraint: We first apply the learning framework to the double pendulum in Figure 6.2 with two degrees of freedom $q = (\theta_1, \theta_2) \in \mathbb{R}^2$ and inputs $u = (\tau_1, \tau_2) \in \mathbb{R}^2$, where τ_i is a torque applied at the joints. The Lagrangian dynamics obey

$$M(q)\ddot{q} + \Gamma(q, \dot{q}) = Bu,$$

where $M(q)$ is the mass matrix and $\Gamma(q, \dot{q})$ collects the gravity and Coriolis terms. The overall state of the system is $x = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathbb{R}^4$.

The control objective is to stabilize the system to the origin, while ensuring that the y-position of the end-effector does not dip below the constraint depicted in Figure 6.2. In Figure 6.2 the origin corresponds to both arms pointing directly to the right. To guide the system towards the origin, the method from [12] is used to design a CLF of the form $V(x) = x^T P x$. We then design a CBF which ensures satisfaction of the safety constraint using the method of *exponential control barrier functions* (ECBFs) described in [97].

To set up the learning problem, we vary the dynamics parameters of the model (mass and length of arms) by 50 percent between the 'true' system dynamics and the nominal model used by the system designer. The learned controller is composed of a linear combination of 300 Gaussian radial basis functions distributed randomly throughout the state-space. We solve the reinforcement learning problem (6.12) with a rollout length of $N = 1$, penalty parameters $\lambda_1 = 1000$ and $\lambda_2 = 100$ and step-length of $\Delta t = 0.05$s. The Soft Actor Critic (SAC) algorithm from [56] is used to solve the problem. Figure 6.2 displays the performance of the learned controller after only 800 samples are collected, which corresponds to 40 seconds of data. The controller was tested from 20 initial conditions, maintaining safety and stability in each scenario.

Safe Bipedal Locomotion on Stepping Stones: We will now apply the presented method to the Hybrid Zero Dynamics (HZD) framework in order to learn an efficient, stable and safe walking controller for a bipedal robot walking on a discrete terrain of randomly spaced stepping stones. The robot is modelled as a hybrid system with impulse effects, as

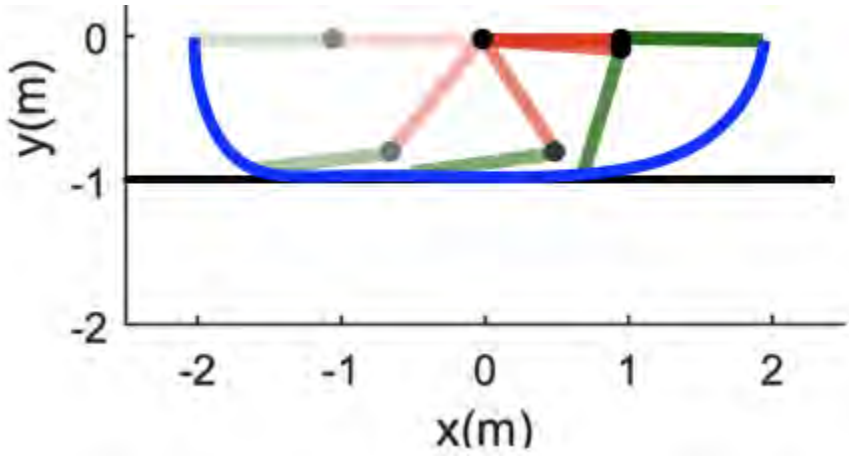


Figure 6.2: A trace of a trajectory for the double pendulum under the influence of the learned controller. The horizontal black line represents the safety constraint, while the blue curve traces the end-effector.

done in [12]:

$$\Sigma : \begin{cases} \dot{\eta} = f(\eta, z) + g(\eta, z)u, & \\ \dot{z} = p(\eta, z) & \text{when } (\eta, z) \notin \mathcal{S}, \\ \eta^+ = \Delta_X\left(\eta^-, z^-\right), & \\ z^+ = \Delta_Z\left(\eta^-, z^-\right) & \text{when } (\eta, z) \in \mathcal{S}, \end{cases} \tag{6.13}$$

where $\eta \in \mathcal{X} \subset \mathbb{R}^{n_a}$ are the actuated states, $z \in \mathcal{Z} \subset \mathbb{R}^{n_u}$ the unactuated states, and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ the control inputs. This model assumes alternating single support phases, where the swing foot is off the ground and the stance foot remains at a fixed point. Impact between the swing foot and the ground is modelled as a rigid impact and occurs when $(\eta, z) \in \mathcal{S}$, where $\mathcal{S}$ is a smooth switching manifold. In (6.13), $\eta^+ \in \mathcal{X}$ and $z^+ \in \mathcal{Z}$ are the post-impact states, while $\eta^- \in \mathcal{X}$ and $z^- \in \mathcal{Z}$ are the pre-impact states.

The method of Hybrid Zero Dynamics (HZD) aims to drive the actuated states to zero thereby constraining the system to evolve on a lower dimensional zero dynamics manifold $\Psi = \{(\eta, z) \in \mathcal{X} \times \mathcal{Z} : \eta = 0\}$, which contains a stable walking gait for the model. As in [12], the system can be stabilized to this surface using feedback linearization to construct a CLF for the actuated coordinates. Following the method in [99], we also design a CBF which takes in the relative distance between the current and subsequent stepping stones and forces the robot to step down on the next stepping stone during each impact event. Both of these functions are only used to constrain the evolution of the continuous dynamics, but are constructed so as to maintain safe, stable walking for the full hybrid dynamics. Because of this, we can directly apply our framework to overcome model uncertainty in the continuous dynamics.

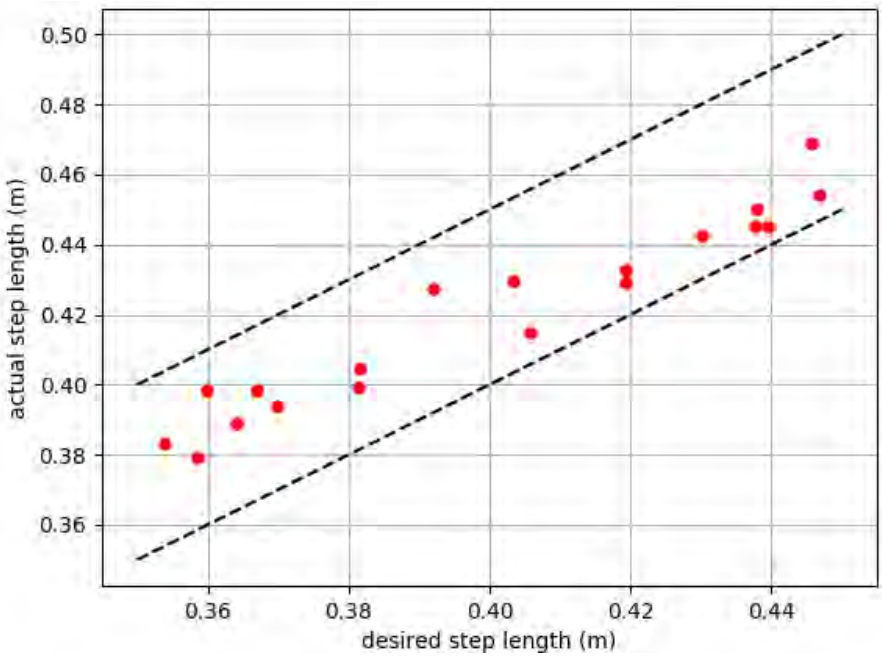


Figure 6.3: Plot of the desired step length vs actual step length achieved by the learned controller for the walking simulation. The black dashed lines indicate the necessary step length constraint required to successfully walk over stepping stones.

To set up the learning problem, model uncertainty is introduced by scaling the mass and inertia of each of the robot's links to be three times those of the nominal model. The learned policy takes the form of a neural network with two hidden layers of size 400×300, and *tanh* activation functions. The training data consists of rollouts of 2 consecutive walking steps with randomly perturbed initial conditions and desired step lengths l_d sampled uniformly from $\mathcal{L} := [0.35, 0.45]m$. We again use SAC to train the policy, with a time step of $\Delta t = 1/1000s$ for numerical simulations. The training process converges in about 200,000 time steps, corresponding to about 3 minutes and 20 seconds of data.

The trained policy is tested on 100 simulations of 10 walking steps each, with desired step lengths uniformly sampled from $\mathcal{L}$. The robot only has knowledge of the position of the next stepping stone. A simulation is considered as a failure if the robot fails to land on any of the desired stepping stones, or if it losses stability and falls. Out of the 100 simulations, 93 were successful using the learned controller, while only 26 simulations were successful with the nominal controller without the learning component. This ability of the learned controller to adapt to different required step lengths is clearly reflected in Figure 6.3. Snapshots of the simulation results is presented in Figure 6.1.

6.5 Chapter Summary

In the last section we have shown that our method can learn safe, stabilizing controllers for systems with high model uncertainty with only seconds or a few minutes of training data. However, there are some limitations of our approach. First, as expected by the use of reinforcement learning algorithms, fine tuning of the learning hyperparameters can be time consuming, and in order to get the results shown in this chapter an extensive search had to be conducted. It will be an important matter for future work to provide algorithm-dependent guarantees which characterize the solutions obtained by specific methods when solving (approximations to) $\mathbf{P}^{(\lambda_1,\lambda_2)}$. Future work will also seek to characterize the effects of input saturation on our theoretical guarantees.

Chapter 7

Lyapunov Design for Robust and Efficient Robotic Reinforcement Learning

Recent advances in the reinforcement learning (RL) literature have enabled roboticists to automatically train complex policies in simulated environments. However, due to the poor sample complexity of these methods, solving RL problems using real-world data remains a challenging problem. This chapter introduces a novel cost-shaping method which aims to reduce the number of samples needed to learn a stabilizing controller. The method adds a term involving a Control Lyapunov Function (CLF) – an 'energy-like' function from the model-based control literature – to typical cost formulations. Theoretical results demonstrate the new costs lead to stabilizing controllers when smaller discount factors are used, which is well-known to reduce sample complexity. Moreover, the addition of the CLF term 'robustifies' the search for a stabilizing controller by ensuring that even highly sub-optimal polices will stabilize the system. We demonstrate our approach with two hardware examples where we learn stabilizing controllers for a cartpole and an A1 quadruped with only seconds and a few minutes of fine-tuning data, respectively. Furthermore, simulation benchmark studies show that obtaining stabilizing policies by optimizing our proposed costs requires orders of magnitude less data compared to standard cost designs.

7.1 Introduction

A key challenge in robotics is reasoning about the long-horizon behavior induced by a control policy. This is because important system properties such as stability are inherently long-horizon phenomena. In reinforcement learning (RL), the *discount factor* implicitly controls how far into the future policy optimization algorithms plan when optimizing the objective specified by the user. Standard approaches to designing objective functions for robotic RL, such as penalizing the distance to a reference trajectory, inherently require a large discount

Figure 7.1: We learn precise stabilizing policies on hardware for the `Quanser` cartpole [114] (top) and the `Unitree A1` quadruped [116] (bottom) using only seconds and a few minutes of real-world data, respectively. A video of our experiments can be found here https://youtu.be/17kBfitE5n8

factor to learn control policies which stabilize the system [111, 43]. Unfortunately, problems with large discount factors can be extremely difficult to solve, often requiring vast data sets and careful tuning of hyper-parameters [41]. As a number of recent success stories have demonstrated [79, 78, 108, 109, 84, 16], ever-increasing computational resources can be used to solve these problems in simulation and deploy the resulting controllers directly on the real-world system. However, because it is impractical to model every detail of complex hardware platforms, achieving the best performance will require learning from real-world data.

This chapter introduces a cost-shaping framework which enables users to reliably learn stabilizing control policies with small amounts of real-world data by solving problems with small discount factors. Our approach uses *Control Lyapunov Functions* (CLFs), a standard design tool from the control theory literature [14, 126, 10, 12]. CLFs are 'energy-like' functions for the system which reduce the search for a stabilizing controller to a myopic one-step criterion. In particular, any controller which decreases the energy of the CLF at each instance of time will stabilize the system. Thus, CLFs reduce the long-horizon objective of stabilizing the system to a simple one-step condition. When a CLF is available and the dynamics are known, constructive techniques from the control literature can be used to synthesize a stabilizing controller. However, when there is uncertainty in the dynamics, it is difficult to guarantee that a controller will always decrease the value of the CLF, or that we have even designed a true CLF for the system.

Our approach is to 1) design an approximate CLF for the real-world system using an approximate dynamics model and 2) modify the 'standard' choice of cost functions mentioned

above by adding a term which incentivizes controllers which decrease the approximate CLF over time. This technique effectively uses the approximate CLF as supervision for reinforcement learning, enabling the user to embed known system structures into the learning process while retaining the flexibility of RL to overcome unknown dynamics. Indeed, as our analysis demonstrates, when our approach is used reinforcement learning algorithms implicitly learn to 'correct' the approximate CLF provided by the user. When the candidate CLF is close to being a true CLF for the system (in a sense we make precise below), a stabilizing controller can be efficiently learned by solving a problem with a small discount factor. Moreover, the addition of the approximate CLF 'robustifies' the search for a stabilizing controller by ensuring that even highly suboptimal policies will stabilize the system. Finally, in situations where it is too difficult to design a nominal CLF by hand, we demonstrate how one can be learned using a simulation model and the standard style of RL objective discussed above. Specifically, we use the value function learned by the RL algorithm as an approximate CLF for the real-world system. Altogether, beyond accelerating and robustifying RL, our approach also expands the applicability of CLF-based design techniques.

We apply this technique to develop data-efficient fine-tuning strategies, wherein a nominal controller developed using a simulation model is refined with small amounts of real-world data. For the A1 experiment, the nominal controller is a model-based control architecture [34], and we hand-design a CLF using a highly simplified linearized reduced-order model for the system. Even though this model is very crude, we are nonetheless able to learn a precise tracking controller for this 18 DOF system with only 5 minutes of real-world data. For the cartpole swing-up task we used the value function from a simulation-based RL problem as the candidate CLF for the real-world system, using the learning process described above. Our fine-tuning approach then learned a robust swing-up controller after observing only one 10 second trajectory from the real-world system.

Related Work

We outline how our approach departs from related work; Appendix B contains further discussion. **Discount Factors, Sample Complexity and Reward Shaping:** It is well-understood that the discount factor has a significant effect on the size of the data set that RL algorithms need to achieve a desired level of performance. Specifically, it has been shown in numerous contexts [18, 120, 93, 112] that smaller discount factors lead to problems which can be solved more efficiently. This has led to a number of works which explicitly treat the discount factor as a parameter which can be used to control the complexity of the problem alongside reward shaping techniques [73, 110, 41, 132, 24, 95]. Compared to these works, our primary contribution is to demonstrate how CLFs can be combined with model-free algorithms to rapidly learn stabilizing controllers for robotic systems.

Fine-tuning with Real World Data: Recently, there has been much interest in using RL to fine-tune policies which have been pre-trained in simulation [125, 75, 74, 87]. These methods typically optimize the same cost function with a large discount factor in both

simulation and on the real robot. In contrast, using our cost reshaping techniques, we solve a different problem with a smaller discount factor on hardware which can be solved more efficiently. In Appendix E, we show that our method outperforms typical fine-tuning approaches under moderate perturbations to the dynamics model.

Learning with Control Lyapunov Functions: A number of recent works have also tried to overcome the reality gap using data-driven methods to improve CLF-based controllers [130, 129, 136, 135, 23, 27]. While these methods work well when a true CLF for the real-world system is available, our method is more general as we can still efficiently learn stabilizing controllers when only an approximate CLF is available by modulating the discount factor used to optimize our cost.

7.2 Background and Problem Setting

Throughout this chapter we will consider deterministic discrete-time systems of the form:

$$x_{k+1} = F(x_k, u_k), \tag{7.1}$$

where $x_k \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time k, $u_k \in \mathcal{U} \subset \mathcal{X}$ is the input applied to the system at that time, and $F \colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is the transition function for the system. This general nonlinear model is broad enough to cover many important continuous control tasks for robotics. We will let Π denote the space of all control polices $\pi \colon \mathcal{X} \to \mathcal{U}$ for the system. To ease exposition, for our theoretical analysis we will focus on the case where the goal is to stabilize the system to a single point, namely the origin. Through our examples we will demonstrate how our cost-shaping technique can be leveraged to achieve more complicated tasks, and in Section 7.5 we outline a path for extending our theoretical results to these settings in future work.

Control Lyapunov Functions

Control Lyapunov Functions [14, 126, 10, 12] are 'energy-like' functions for the dynamics (7.1):

Definition 7.1. *We say that a positive definite function $W \colon \mathbb{R}^n \to \mathbb{R}$ is a* Control Lyapunov Function *(CLF) for (7.1) if the following condition holds for each $x \in \mathcal{X} \backslash \{0\}$:*

$$\min_{u \in \mathcal{U}} W(F(x, u)) - W(x) < 0. \tag{7.2}$$

The condition (7.2) ensures that for each $x \in \mathcal{X}$ there exists a choice of input which decreases the 'energy' $W(x)$. Any policy which satisfies the one-step condition $W(F(x, \pi(x))) - W(x) < 0$ can be guaranteed to asymptotically stabilize the system [77, 4] (see Appendix C for background on stability theory). Given a CLF for the system, model-based methods constructively synthesize a controller which satisfies this property using either closed-form

equations [126] or by solving an online (convex) optimization problem [42, 12] to satisfy (7.2). However, when the dynamics are unknown it is difficult to ensure that we have synthesized a 'true' CLF for the system.

Remark 7.1. *(Designing Control Lyapunov Functions) While there is no general procedure for designing CLFs by hand for general nonlinear systems, there do exist constructive procedures for designing CLFs for many important classes of robotic systems, such as manipulator arms [10] and robotic walkers [12] using structural properties of the system. Moreover, in our examples we will investigate how a CLF can be learned from a simulation model and how very coarse CLF candidates can be used to accelerate learning a stabilizing controller.*

Stability of Dynamic Programming and Reinforcement Learning

Here we investigate how a common class of cost functions found in the literature can be used to learn stabilizing controllers. In particular, we consider a running cost $\ell\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ of the form $\ell(x,u) = Q(x) + R(u)$, where $Q\colon \mathcal{X} \to \mathbb{R}$ is the state cost and $R\colon \mathcal{U} \to \mathbb{R}$ is the input cost. Both Q and R are assumed to be positive definite (in practice, both are usually quadratic). Given a policy $\pi \in \Pi$, discount factor $\gamma \in^{0,1}$, and initial condition $x_0 \in \mathcal{X}$, the associated long-run cost is:

$$V_\gamma^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k \ell(x_k, \pi(x_k)) \quad \text{s.t. } x_{k+1} = F(x_k, \pi(x_k)), \tag{7.3}$$

where $V_\gamma^\pi\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is the *value function* associated to π. Small discount factors incentivize policies which greedily optimize a small number of time-steps into the future, while larger discount factors promote policies which reduce the cost in the long-run. We say that a policy $\pi_\gamma^* \in \Pi$ is *optimal* if it achieves the smallest cost from each $x \in \mathcal{X}$:

$$V_\gamma^{\pi_\gamma^*}(x) = V_\gamma^*(x) := \inf_{\pi \in \Pi} V_\gamma^\pi(x), \qquad \forall x \in \mathcal{X},$$

where $V_\gamma^*\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is the *optimal value function*. Together V_γ^* and π_γ^* capture the 'ideal' behavior induced by the cost function (7.3). It is well-known [18] that the optimal value function will satisfy the Bellman equation:

$$V_\gamma^*(x) = \inf_{u \in \mathcal{U}} \left[\gamma V_\gamma^*(F(x,u)) + \ell(x,u)\right], \qquad \forall x \in \mathcal{X}, \tag{7.4}$$

and an optimal policy π_γ^* will satisfy $\pi_\gamma^*(x) \in \arg\min_{u \in \mathcal{U}} \left[\gamma V_\gamma^*(F(x,u)) + \ell(x,u)\right]$, $\forall x \in \mathcal{X}$. Unfortunately, it is impractical to directly search over Π to find a policy which meets these conditions. This necessitates the use of function approximation schemes (e.g. feed-forward neural networks) to instead represent a subset of policies $\hat{\Pi} \subset \Pi$ to search over. Indeed,

modern RL approaches for robotics randomly sample the space of trajectories to optimize problems of the form:

$$\inf_{\pi \in \hat{\Pi}} \mathbb{E}_{x_0 \sim X_0}\big[V_\gamma^\pi(x_0)\big], \tag{7.5}$$

where X_0 is a distribution over initial conditions. While this approach enables these methods to optimize high-dimensional policies, they are data-hungry, can display high-variance and thus frequently return highly sub-optimal policies when data is limited. To better understand the effect that this has on the stability of learned policies, for each $\pi \in \hat{\Pi}$ and $\gamma \in^{0,1}$ define the *optimality gap*:

$$\varepsilon_\gamma^\pi(x) = V_\gamma^\pi(x) - V_\gamma^*(x).$$

The temporal difference equation [18] dictates that for each $x \in \mathcal{X}$ the policy satisfies:

$$V_\gamma^\pi(x) = \gamma V_\gamma^\pi(F(x,\pi(x))) + \ell(x,\pi(x)). \tag{7.6}$$

From these equations we can obtain:

$$\begin{aligned} V_\gamma^\pi(F(x,\pi(x))) - V_\gamma^\pi(x) &= \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)V_\gamma^\pi(x)\big) && (7.7)\\ &= \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)[V_\gamma^*(x) + \varepsilon_\gamma^\pi(x)]\big) && (7.8)\\ &\leq \frac{1}{\gamma}\big(-Q(x) + (1-\gamma)[V_\gamma^*(x) + \varepsilon_\gamma^\pi(x)]\big), && (7.9) \end{aligned}$$

where we have first rearranged (7.6), then used $V_\gamma^\pi(x) = V_\gamma^*(x) + \varepsilon_\gamma^\pi(x)$, and finally we have used $\ell(x,\pi(x)) \geq Q(x)$. Inequalities of this sort are the building block for proving the stability of suboptimal polices in the dynamic programming literature [43, 111].

Remark 7.2. *(Value Functions as CLFs) By inspecting the cost (7.3) we see that V_γ^π is positive definite (since Q is positive definite). Thus, if the right-hand side of (7.9) is negative for each $x \in \mathcal{X} \setminus \{0\}$, this inequality shows that V_γ^π is a CLF for (7.1), and that π is an asymptotically stabilizing control policy. In other words, V_γ^π is a CLF which is implicitly learned during the training process. Indeed, many RL algorithms directly learn an estimate of the value function, a fact which we later exploit to learn a CLF for the cartpole swing up-task in Section 7.4 using the nominal simulation environment.*

Note that the right hand side of (7.9) will only be negative if $V_\gamma^*(x) + \varepsilon_\gamma^\pi(x) < \frac{1}{1-\gamma}Q(x)$. Since from (7.3) we know that $V_\gamma^*(x) > Q(x)$ for each $x \in \mathcal{X}$, even the optimal policy (which has no optimality gap) will only be stabilizing if γ is large enough. On the other hand, for a fixed $\gamma \in (0,1]$, this inequality also quantifies how sub-optimal a policy can be while maintaining stability. To make these observations more quantitative we make the following assumption:

Assumption 7.1. *For each $\gamma \in^{0,1}$ there exists $C_\gamma \geq 1$ such that $V_\gamma^*(x) \leq C_\gamma Q(x)$ for each $x \in \mathcal{X}$.*

Growth conditions of this form are standard in the literature on the stability of approximate dynamic programming [85, 111, 43, 44]. Note that, because the running cost ℓ is non-negative, we have $C_{\gamma'} \leq C_{\gamma''}$ if $\gamma' \leq \gamma''$. In particular, the constant C_1 upper-bounds the ratio between the one-step cost and the optimal undiscounted value function. When C_1 is smaller, the optimal undiscounted policy is more 'contractive' and approximate dynamic programming methods converge more rapidly to an optimal solution [85]. Thus, intuitively the constants $C_\gamma \geq 1$ will be smaller when the system is easier to stabilize. The following result is essentially a specialization of the main result from [44]:

Proposition 7.1. *Let Assumption 7.1 hold and let $\gamma \in^{0,1}$ and $\pi \in \hat{\Pi}$ be fixed. Further assume that there exists $\delta > 0$ such that for each $x \in \mathcal{X}$ we have i) $\varepsilon_\gamma^\pi(x) \leq \delta Q(x)$ and ii) $C_\gamma + \delta < \frac{1}{1-\gamma}$. Then, π asymptotically stabilizes (7.1).*

Proof. Combining conditions i) and ii) with equation (7.9) yields:

$$V_\gamma^\pi(F(x,\pi(x))) - V_\gamma^\pi(x) \leq \frac{2}{\gamma}\big(-1 + (1-\gamma)[C_\gamma + \delta]\big)Q(x). \qquad \square$$

Thus the RHS of the preceding equation will be negative-definite if $C_\gamma + \delta < \frac{1}{1-\gamma}$, which demonstrates the desired result.

Remark 7.3. *(Stability Properties of the Cost Function) In the following section we will derive an analogous result to Proposition 7.1 for the novel reshaped cost function we propose below. When comparing these results we will primarily focus on the effect of the constants $C_\gamma \geq 1$ (and the equivalent constants for the new setting). The C_γ constants can be used to bound how large of a discount factor is need to stabilize the system. In particular, Proposition 7.1 implies that the optimal policy will stabilize the system for each γ which satisfies $\gamma > 1 - \frac{1}{C_\gamma}$. The C_γ constants also characterizes how 'robust' the cost function is to suboptimal policies. In particular, for a fixed discount factor, the policy will stabilize the system if $\delta < \frac{1}{1-\gamma} - C_\gamma$. Thus smaller values of the C_γ constants permit more suboptimal policies.*

7.3 Lyapunov Design for Infinite Horizon Reinforcement Learning

Our method uses a positive definite candidate Control Lyapunov Function $W : \mathbb{R}^n \to \mathbb{R}$ for the nonlinear dynamics (7.1), and reshapes (7.3) to our proposed new long horizon cost $\tilde{V}_\gamma^\pi : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$:

$$\begin{aligned} \tilde{V}_\gamma^\pi(x_0) = &\sum_{k=0}^{\infty} \gamma^k \Big([W\big(F(x_k,\pi(x_k))\big) - W(x_k)] + \ell(x_k,\pi(x_k)) \Big) \\ &\text{s.t. } x_{k+1} = F(x_k,\pi(x_k)). \end{aligned} \tag{7.10}$$

As we shall see below, our method works best when W is in fact a CLF for the system, but still provides benefits when it is only an 'approximate' CLF for the system (in a sense we will make precise later). For each $\gamma \in^{0,1}$ the new optimal value function is given by:

$$\tilde{V}_\gamma^*(x) = \inf_{\pi \in \Pi} \tilde{V}_\gamma^\pi(x). \tag{7.11}$$

The new cost (7.10) includes the amount that W changes at each time step, and thus encourages choices of inputs which decrease W over time. In this case, the Bellman equation [18] dictates:

$$\tilde{V}_\gamma^*(x) = \inf_{u \in \mathcal{U}} \left[\gamma \tilde{V}_\gamma^*(F(x,u)) + \Delta W(x,u) + \ell(x,u)\right], \quad \forall x \in \mathcal{X}, \tag{7.12}$$

where $\Delta W(x,u) := W(F(x,u)) - W(x)$. To gain some intuition for the approach let us consider the two extremes where $\gamma = 0$ and $\gamma = 1$. In the case where $\gamma = 1$, by inspection we see that $\tilde{V}_1^* = V_1^* - W$ solves the Bellman equation. Plugging in this solution demonstrates that any optimal policy $\tilde{\pi}_1^*$ must satisfy $\tilde{\pi}_1^*(x) \in \arg\min_{u \in \mathcal{U}} [V_1^*(F(x,u)) + \ell(x,u)]$. This is precisely the optimality condition for the original cost (7.3) when $\gamma = 1$, and thus the set of optimal policies for the two problems coincide. Thus, in this case, by embedding the CLF in the cost we are effectively using W as a warm-start initial guess for the optimal value function. In the other extreme where $\gamma = 0$, from (7.12) we see that an optimal policy must satisfy $\tilde{\pi}_0^*(x) \in \arg\min_{u \in \mathcal{U}} \left[\Delta W(x,u) + \ell(x,u)\right]$. Thus, when $\gamma = 0$ the optimal policy attempts to greedily decrease the value of the candidate CLF and the one-step cost on the input. As we shall see below, when intermediate discount factors are used, optimal policies may instead decrease the value of W over the course of several steps.

Using the new cost function (7.10), each policy must satisfy the new difference equation:

$$\tilde{V}_\gamma^\pi(x) = \gamma \tilde{V}_\gamma^\pi\big(F(x,\pi(x))\big) + W\big(F(x,\pi(x))\big) - W(x) + \ell(x,\pi(x)). \tag{7.13}$$

In our stability analysis, we will use the following composite function as a candidate CLF for (7.1):

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) = W(x) + \gamma \tilde{V}_\gamma^\pi(x). \tag{7.14}$$

We provide an interpretation of this curious candidate CLF in Remark 7.4 below, but first perform an initial analysis similar to the one presented in the previous section. Defining for each $\pi \in \hat{\Pi}$, $\gamma \in^{0,1}$ and $x \in \mathcal{X}$ the new optimality gap:

$$\tilde{\varepsilon}_\gamma^\pi(x) = \tilde{V}_\gamma^*(x) - \tilde{V}_\gamma^\pi(x), \tag{7.15}$$

and following steps analogous to those taken in (7.7)-(7.9), we can obtain the following:

$$\begin{aligned}
\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi\big(F(x,\pi(x))\big) - \tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) &= -\ell(x,\pi(x)) + (1-\gamma)\tilde{V}_\gamma^\pi(x) && (7.16)\\
&= -\ell(x,\pi(x)) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\varepsilon}_\gamma^\pi(x)\big] && (7.17)\\
&\leq -Q(x) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\varepsilon}_\gamma^\pi(x)\big]. && (7.18)
\end{aligned}$$

Similar to the analysis in the previous section, we will aim to understand when the right-hand side of (7.18) is negative, as this will characterize when π stabilizes the system. One key difference between the inequalities (7.9) and (7.18) is that, while the original value function V_γ^* is necessarily positive definite, $\tilde{V}_\gamma^*$ can actually take on negative values since the addition of the CLF term allows the new running cost in (7.10) to be negative. As we shall see, this forms the basis for the stability and robustness properties our cost formulation enjoys when W is designed properly.

Remark 7.4. *(Learning Corrections to W) When the right hand side of (7.18) is negative for each $x \in \mathcal{X} \setminus \{0\}$, inequality (7.18) demonstrates that $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi$ is in fact a CLF for (7.1) and that π stabilizes the system (see Theorem 7.1). We can think of W as an 'initial guess' for a CLF for the system, while $\gamma\tilde{V}_\gamma^\pi$ is a 'correction' to W that is implicitly made by a learned policy π. Roughly speaking, the larger the discount factor, the larger this correction. Thus, the user can trade-off how much the learned policy is able to correct the candidate CLF W against the additional complexity of solving a problem with a higher discount factor, depending on how 'good' they believe the CLF candidate to be.*

We first state a general stability result for suboptimal policies associated to the new cost, and then discuss how the choice of W affects the stability of suboptimal control policies:

Assumption 7.2. *For each $\gamma \in^{0,1}$ there exists $\tilde{C}_\gamma \in \mathbb{R}$ such that $\tilde{V}_\gamma^*(x) \leq \tilde{C}_\gamma Q(x)$ for each $x \in \mathcal{X}$.*

Because the reshaped one-step cost $W(F(x,u)) - W(x) + \ell(x,u)$ can take on negative values, so can the $\tilde{C}_\gamma$ constants. Moreover, in this case it is possibe to have $\tilde{C}_{\gamma'} \geq \tilde{C}_{\gamma''}$ when $\gamma' \leq \gamma''$. This is because when larger discount factors are used, the optimal policy can benefit from decreasing W further into the future. The following stability result is analogous to Proposition 7.1:

Theorem 7.1. *Let Assumption 7.2 hold and let $\gamma \in^{0,1}$ and $\pi \in \hat{\Pi}$ be fixed. Further assume that there exists $\tilde{\delta} > 0$ such that for each $x \in \mathcal{X}$ we have i) $\tilde{\varepsilon}_\gamma^\pi(x) \leq \delta Q(x)$ and ii) $\tilde{C}_\gamma + \tilde{\delta} < \frac{1}{1-\gamma}$. Then, π asymptotically stabilizes (7.1).*

The proof is conceptually similar to the proof of Proposition 1; we delegate the proof to Appendix D for brevity. Indeed, note that the conditions for stability under the new cost are essentially identical to those for the previous cost in Proposition 7.1.

As alluded to in Remark 7.3, we will primarily focus on comparing how large the constants $C_\gamma \geq 1$ and $\tilde{C}_\gamma \in \mathbb{R}$ are for the two problems, as they control the discount factor required to learn a stabilizing policy and also the 'robustness' of the cost to suboptimal controllers. We provide two characterizations which ensure that $\tilde{C}_\gamma < C_\gamma$. The first condition is taken from the model-predictive control literature [71, 52], where CLFs are used as terminal costs for finite-horizon prediction problems. Proof of the following result can be found in Appendix D:

Lemma 7.1. *Suppose that for each $x \in \mathcal{X}$ the following condition holds:*

$$\inf_{u \in U} W(F(x,u)) - W(x) + \ell(x,u) \leq 0. \tag{7.19}$$

Then Assumption 7.2 is satisfied with constant $\tilde{C}_\gamma \leq 0$.

The hypothesis of Lemma 7.1 implies that *i*) W is a true CLF for the system and *ii*) W dominates the running cost ℓ, in the sense that W can be decreased more rapidly than ℓ accumulates. Effectively, this condition implies that it is advantageous for polices to myopically decrease W at each time step. Consequently, when this condition holds optimal polcies associated to the reshaped costs (7.10) will stabilize the system for any choice of discount factor.

The following definition generalizes this condition to cases where W may not be a true CLF for the system but can be decreased over several time-steps:

Definition 7.2. *We say that the candidate CLF W $\bar{\gamma}$-dominates the running cost ℓ if for each discount factor $\bar{\gamma} \leq \gamma \leq 1$ and $x \in \mathcal{X}$ we have $\tilde{V}_\gamma^*(x) \leq V_\gamma^*(x)$.*

The condition in (7.2) effectively provides a way of characterizing how 'close' W is to being a true CLF for the real-world system. In particular, the larger $\bar{\gamma}$ the further into the future RL algorithms must look to see the benefits of decreasing W. Our previous discussion, which showed that $\tilde{V}_1^* = V_1^* - W$, demonstrates that every candidate CLF 1-dominates the cost. Moreover, clearly W can only 0-dominate the original cost if it is a CLF for the system. While this condition is more difficult to verify for intermediate values of $\bar{\gamma}$, it provides qualitative insight into how even approximate CLFs for the system can still make it easier to obtain stabilizing controllers.

Remark 7.5. *(Robustness of reshaped cost) When the condition of Lemma 7.1 is satisfied we will have $\tilde{C}_\gamma \leq 0 < C_\gamma$, implying the new cost enjoys the desirable robustness properties discussed above. When W satisfies the 'approximate CLF' condition in Definition (7.2), it will only enjoy these benefits when the discount factor is large enough. We leave it as a matter for future work to provide quantitative estimates for the $\tilde{C}_\gamma$ constants in these regimes, and to provide sufficient conditions which ensure W $\bar{\gamma}$-dominates the running cost.*

7.4 Examples and Practical Implementations

We summarize the main results for each of our examples, but leave most details and plots to Appendix E. In every experiment we report, the soft actor-critic algorithm (SAC) [56] is used as the learning algorithm to optimize the various reward structures we investigate.

Velocity Tracking for A1 Quadruped: We apply our approach to train a neural network controller which augments and improves a nominal model-based controller [34] for a quadruped

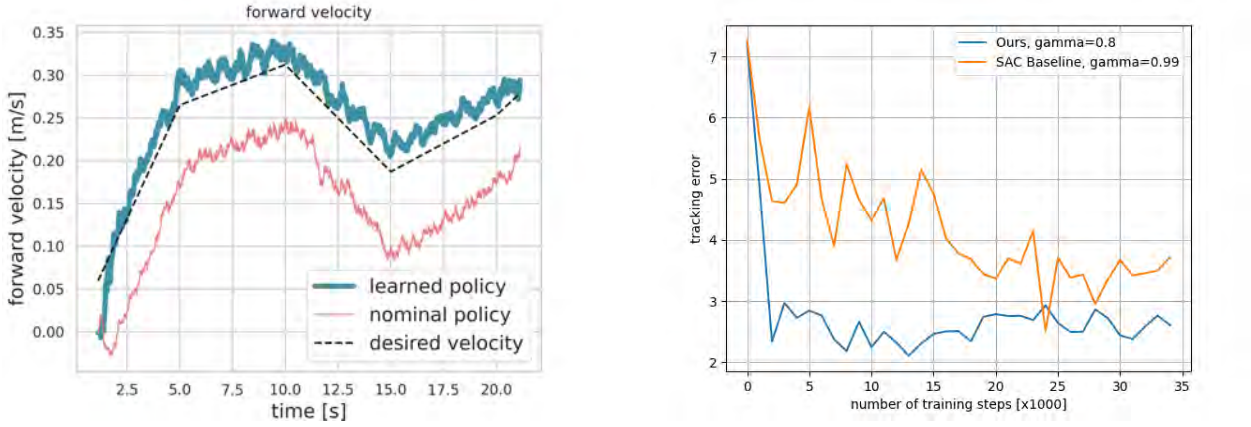


Figure 7.2: (Left) Plot illustrating improved velocity tracking of the learned policy (in dark green) compared to the nominal locomotion controller (in pink) to track a desired velocity profile (in dashed black line) using our proposed method on the `Unitree A1` robot hardware. (Right) Plot from the simulated benchmark study illustrating cumulative velocity tracking error (lower is better) over 10s rollouts at different stages of the training. In orange, we show the results of fine-tuning using SAC with a standard RL cost. In blue, we fine-tune using SAC with our reward reshaping method, with a candidate CLF designed on a nominal linearized model of the robot. In both cases, we plot the results using the discount factor that achieved the best performance.

robot using real-world data. As illustrated by the pink curve in Figure 7.2 (left), the nominal controller fails to accurately track desired velocities specified by the user. We design a CLF around the desired gait using a linearized reduced-order model for the system. We then collect rollouts of $10s$ on the robot hardware with randomly chosen desired velocity profiles, and solve an RL problem using our cost and a discount factor $\gamma = 0$. Our approach is able to learn a policy which significantly improves the tracking performance of the nominal controller within 5 minutes (30 episodes) of hardware data, as shown in Figure 7.2 (left). A video of these results can be found in https://youtu.be/l7kBfitE5n8, and more details are provided in Appendix E. Furthermore, in Figure 7.2 (right) we benchmark our approach in simulation against an RL agent trained with a 'standard' cost which penalizes the squared error with respect to the desired velocity. As this figure demonstrates, our method is able to rapidly decrease the average tracking error in only around 2 thousand steps from the environment. In contrast, the benchmark approach is only able to reach this level of performance for the first time after around 24 thousand steps.

A1 Quadruped Walking with an Unknown Load: We attach an un-modeled load to the A1 quadruped, that is equivalent to one-third the mass of the robot. Fine-tuning on hardware the same base controller from the previous set-up where the CLF is designed to stabilize to the target gait, our approach is able to significantly decrease the tracking error to about one-third its nominal value with only one minute of data collected on the robot

hardware as illustrated in Figure E.1 in Appendix E. Additionally, in Appendix E, we run a simulated benchmark comparison and verify that our method clearly out-performs the 'standard' cost baseline for this task.

Fine-tuning a Learned Policy for Cartpole Swing-Up: We fine-tune a swing-up controller for the `Quanser` cartpole system [114] using real-world data and an initial policy which was pre-trained in simulation but that does not translate well to the real system. Due to the underactuated nature of the system, synthesizing a CLF by hand is challenging. Thus, as alluded to previously, we use a 'typical' cost function of the form (7.3) and a discount factor of $\gamma = 0.999$ to learn a stabilizing neural network policy π_ϕ for a simulation model of the system. Given the discussion in Remark 7.2, we use the value function V_θ associated with the simulation-based policy as the candidate CLF ($W = V_\theta$) for our reward reshaping formulation (7.10). When improving the simulation-based policy π_ϕ with real-world data, we keep the parameters of this network fixed and learn an additional smaller policy π_ψ (so that the overall control action is produced by $\pi_\phi + \pi_\psi$) using our proposed CLF-based cost formulation. We solve the reshaped problem with a discount factor $\gamma = 0$ and collect rollouts of $10s$ on hardware. Our CLF-based fine-tuning approach is able to successfully complete the swing-up task after collecting data from just one rollout. After collecting data from an additional rollout, the controller is reliable and robust enough to recover from several pushes. A video of these experiments can be found in https://youtu.be/l7kBfitE5n8, and more details and plots of the results are provided in Appendix E. Furthermore, in Appendix E we provide a simulation study comparing a standard fine-tuning approach to our method, showing that our approach is able to more rapidly learn a reliable swing-up policy than the baseline and also achieves a higher reward.

Fine-tuning a Bipedal Walking Controller in Simulation: We also apply our design methodology to fine-tune a model-based walking controller [12] for a bipedal robot with large amounts of dynamics uncertainty. Model uncertainty is introduced by doubling the mass of each link of the robot. The nominal controller fails to stabilize the gait and falls within a few steps. To apply our method, we design a CLF around the target gait as in [12] to be used in our reward formulation. As a benchmark comparison, we also train policies with a reward which penalizes the distance to the target motion (no CLF term), as is most commonly done in RL approaches for bipedal locomotion which use target gaits in the reward [84]. Our approach is able to significantly reduce the average tracking error per episode after only 40000 steps of the environment (corresponding to 40 seconds of data), while the baseline does not reach a similar level of performance even after 1.2 million steps, as illustrated in Figure E.5 of Appendix E.

Inverted Pendulum with Input Constraints: Our final example demonstrates the utility of our method even when W is a crude guess for a CLF for the system, through the use of moderate discount factors. We illustrate this for a simple inverted pendulum simulator by varying the magnitude of the input constraints for the system. We use the procedure from [12] to design a candidate CLF for the system. Like many CLF design techniques,

this approach assumes there are no input constraints and encourages the pendulum to swing directly up. As the input constraints are tightened, W becomes a poorer candidate CLF, as there is not enough actuation authority to decrease W at each time step. Even in this case, in line with the discussion of Remark 7.5, if a proper discount factor is used, the addition of the candidate CLF in the reward enables our method to rapidly learn a stabilizing controller for each setting of the input bound. These results are presented in Appendix E.

7.5 Chapter Summary

As we have mentioned previously, our approach has several limitations. The cost-shaping technique we introduce in Section 7.3 only provides benefits when W is in-fact a reasonable guess for a CLF for the true system. This requires that the user has a dynamics model which captures the primary features of the environments which affect the structure of CLFs for the system. While the cart-pole simulations we provide in the Appendix E provide some intuition for when this will be the case, further research is needed to better understand in what scenarios we can see significant benefits from our method. Nonetheless, our two hardware experiments provide encouraging initial results which indicate that our method can rapidly learn stabilizing controllers using CLFs which are constructed using a nominal dynamics model. More broadly, there are many exciting avenues for further incorporating Lyapunov design techniques with RL, especially offline learning [82].

Chapter 8

Conclusion and Future Work

This chapter provides a summary of the work presented in this dissertation, concluding remarks, and directions for future work. This dissertation developed planning algorithms and feedback controllers inspired by model-based techniques to achieve stable, safe, and robust dynamic locomotion for legged robots such as quadrupeds and bipeds.

Navigating challenging terrain with discrete footholds requires precise foot placement and the satisfaction of physical constraints on the contact wrench exerted by the ground. Moreover, agile maneuvers such as running and jumping are required when the distance between these footholds is large. This thesis presented a controls and planning technique for a planar bipedal robot to achieve the task of running over a field of randomly placed stepping stones. The proposed method was validated through several numerical simulations. Specifically, a library of *two-step-periodic* trajectory libraries was developed for running at various step lengths. An HZD-based controller was then implemented to achieve stable running. A key observation for running was that the choice of output variables to control was crucial to obtain precise foot placement. The two-step periodic gaits presented in Chapter 3 also prove to be effective on other legged robots such as quadrupeds, as was demonstrated in Chapter 5.

One of the drawbacks of the HZD method presented in Chapter 3 is its inability to encode physical constraints such as friction in the feedback controller. Additionally, the method lacks predictive capabilities. This dissertation developed a coordinate-free MPC method based on the rigid body reduced-order model, which evolves on the $SE(3)$ manifold. The method was validated on several legged robotic hardware platforms, including the `Unitree A1` quadruped as well as `Cassie` biped. Several indoor and outdoor experiments were performed on a wide range of rough terrain to test the robustness of the proposed approach.

To navigate challenging terrain, robots need to be able to perceive the surrounding environment and plan trajectories in real-time to achieve safe navigation. Additionally, when perception systems fail, the underlying feedback controller must be robust to uncertainties in the environment. To address these challenges, the method of *two-step-periodic* gait library is

extended to a quadruped robot. The trajectories are stabilized by the proposed coordinate-free MPC developed in Chapter 4 to navigate a field of stepping stones using visual feedback through a depth camera. Several experiments and ablation studies are performed to demonstrate the efficacy of the proposed approach.

While model-based techniques provide fine-grained control around a desired behavior of the system, their performance degrades in the presence of model uncertainty. Learning-based techniques like RL, on the other hand, are able to synthesize complex behaviors for highly dynamic systems but suffer from poor sample complexity. Moreover, for tasks that require precise control, such as stepping stones, it is difficult to characterize the reliability of such methods. This dissertation develops novel cost-shaping methods inspired by CLFs and CBFs from model-based literature for RL to learn safe, robust, and efficient policies for legged robots. Through numerous simulations and experiments on hardware, the proposed method is able to quickly learn stabilizing and safe policies even under significant model uncertainty. The proposed method allows one to quickly finetune model-based as well as RL-based policies using only a few seconds to a few minutes of real-world data.

Directions for Future Work

Having presented a summary of this thesis, we now present some limitations of our work and future directions.

Two-Step-Periodic Gait Library and Nonlinear Model Predictive Control

One of the primary motivations of the *two-step-periodic* gait library approach was that it reduced the computational burden of finding trajectories online for legged robots, which can be computationally intractable. However, a major drawback of this approach is that the controller is restricted to the motions and gaits within the library.

Recent results have shown that it is possible to implement Nonlinear Model Predictive Control (NMPC) on 2D bipedal robots utilizing the whole-body dynamical model [45] as well as on quadrupedal robots utilizing a kino-dynamic model [49]. These methods leverage recent advances in optimization-based control. These methods have the ability to plan for the full joint configuration in real-time and partially or completely eliminate the requirement of trajectories generated offline.

Extensions to Rigid Body Model and Higher Dimensional Robots

The rigid body model presented in Chapter 4, provides a concise and effective reduced order model for quadruped and bipedal robots such as Cassie. The method can be easily extended to higher dimensional robots like Digit. Figure 8.1 illustrates simulation results on the Digit humanoid robot for balancing utilizing the proposed Geomertric MPC from Chapter 4. We use a planar surface contact model (2.24) instead of a line contact model for Cassie.

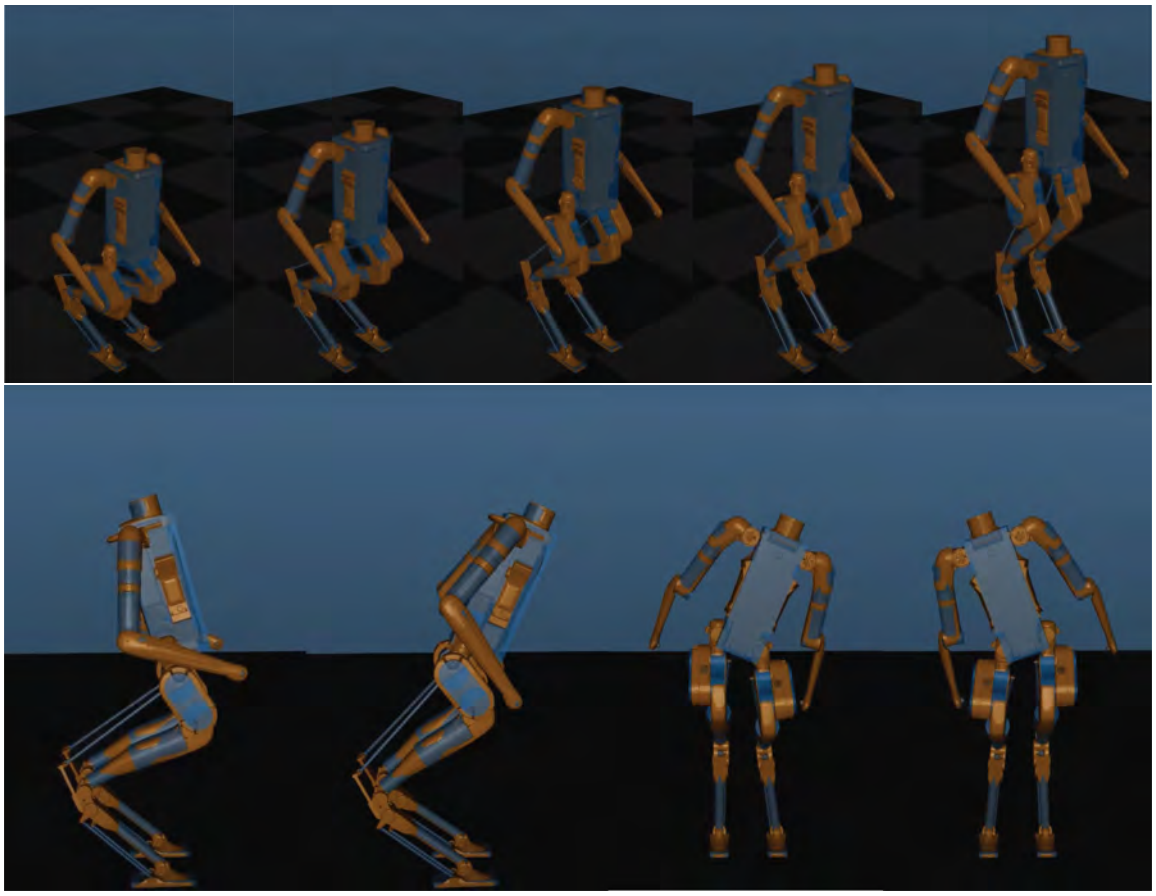


Figure 8.1: Simulation results in `MuJoCo` for (Top) Digit squatting in-place and (Bottom) re-orienting its body using the proposed geometric MPC in Chapter 4.

A primary motivation for using such a centroidal model without explicitly modeling the limbs is that they weigh significantly less compared to the mass of the main body and can be considered inertialess. However, this assumption breaks when the limbs are moving too quickly or if the mass and inertia of the legs is significant. Recent works have suggested using a kino-dynamic model based on the centroidal momentum dynamics. In this line of work, a kinematic model relating the joint velocities to the centroidal momentum is utilized, along with a dynamical model describing the evolution of the centroidal linear and angular momentum.

Extending from our approach in Chapter 4, we can capture the effect of the limbs on the change in inertia and centroidal momentum by explicitly modeling the limbs as additional rigid bodies. We can also capture change in inertia from the torso (due to arm swing, for example), by considering a variable inertia model.

Planning through Contacts and Mode Sequences

Another limitation of the approaches presented in this thesis is the inability to plan through contacts and mode sequences. From a model-based perspective, this is a challenging control problem due to the nonlinear and hybrid nature of the underlying system. This is a growing area of research, and several recent works [15, 29] have introduced frameworks to tackle this problem. Another possible approach is to use tools from reachability to such as in [28] where we extend the Hamilton-Jacobi reachability framework to account for changing contact conditions. Through this framework, we are able to compute a bigger Region of Attraction as well as a stabilizing controller that accounts for mode switches.

Combining Wheels with Legged Locomotion

While legged robots can enable locomotion on rugged terrain, on flat terrain, wheeled robots can outperform them in both speed and energy efficiency. To bridge this performance gap, several researchers [5, 13, 20, 83] have looked at combining wheeled platforms with legged robots to achieve locomotion on rough terrain as well as energy efficiency and speed on flat ground.

Bibliography

[1] Ananye Agarwal et al. "Legged locomotion in challenging terrains using egocentric vision". In: *arXiv preprint arXiv:2211.07638* (2022).

[2] *Agility Robotics Digit*. https://agilityrobotics.com/robots. Accessed: 2022-11-30.

[3] Ayush Agrawal and Koushil Sreenath. "Bipedal Robotic Running on Stochastic Discrete Terrain". In: *European Control Conference (ECC)*. Naples, Italy, June 2019, pp. 3564–3570.

[4] Ayush Agrawal and Koushil Sreenath. "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation." In: *Robotics: Science and Systems*. Vol. 13. Cambridge, MA, USA. 2017.

[5] Ayush Agrawal et al. "Experimental gait analysis of waveboard locomotion". In: *Dynamic Systems and Control Conference*. Vol. 50701. American Society of Mechanical Engineers. 2016, V002T22A011.

[6] Ayush Agrawal et al. "First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons". In: *IEEE Access* 5 (2017), pp. 9919–9934.

[7] Ayush Agrawal et al. "Vision-aided dynamic quadrupedal locomotion on discrete terrain using motion libraries". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 4708–4714.

[8] Anayo K Akametalu et al. "Reachability-based safe learning with Gaussian processes". In: *IEEE Conference on Decision and Control*. 2014, pp. 1424–1431.

[9] A. D. Ames et al. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876.

[10] Aaron Ames and Matthew Powell. "Towards the Unification of Locomotion and Manipulation through Control Lyapunov Functions and Quadratic Programs". In: *Lecture Notes in Control and Information Sciences* 449 (Jan. 2013), pp. 219–240. DOI: 10.1007/978-3-319-01159-2_12.

[11] Aaron D Ames. "Human-Inspired Control of Bipedal Walking Robots." In: *IEEE Trans. Automat. Contr.* 59.5 (2014), pp. 1115–1130.

[12] Aaron D Ames et al. "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[13] Jonathan Anglingdarma et al. "Motion Planning and Feedback Control for Bipedal Robots Riding a Snakeboard". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2818–2824.

[14] Zvi Artstein. "Stabilization with relaxed controls". In: *Nonlinear Analysis: Theory, Methods & Applications* 7.11 (1983), pp. 1163–1173.

[15] Alp Aydinoglu and Michael Posa. "Real-time multi-contact model predictive control via admm". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 3414–3421.

[16] Suneel Belkhale et al. "Model-based meta-reinforcement learning for flight with suspended payloads". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1471–1478.

[17] Felix Berkenkamp et al. "Safe model-based reinforcement learning with stability guarantees". In: *NeurIPS*. 2017, pp. 908–918.

[18] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[19] John T Betts. "Survey of numerical methods for trajectory optimization". In: *Journal of guidance, control, and dynamics* 21.2 (1998), pp. 193–207.

[20] Marko Bjelonic et al. "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2116–2123.

[21] *Boston Dynamics Atlas*. https://www.bostondynamics.com/atlas. Accessed: 2022-11-30.

[22] Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Vol. 49. Springer, 2019.

[23] Fernando Castañeda et al. "Gaussian Process-based Min-norm Stabilizing Controller for Control-Affine Systems with Uncertain Input Effects and Dynamics". In: *2021 American Control Conference (ACC)*. 2021, pp. 3683–3690.

[24] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. "Heuristic-guided reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13550–13563.

[25] Christine Chevallereau et al. "Rabbit: A testbed for advanced control theory". In: *IEEE Control Systems Magazine* 23.5 (2003), pp. 57–79.

[26] Matthew Chignoli and Patrick M Wensing. "Variational-based optimal control of underactuated balancing for dynamic quadrupeds". In: *IEEE Access* 8 (2020), pp. 49785–49797.

[27] Jason Choi et al. "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions". In: *Robotics: Science and Systems*. Corvalis, OR, July 2020.

[28] Jason J Choi et al. "Computation of Regions of Attraction for Hybrid Limit Cycles Using Reachability: An Application to Walking Robots". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4504–4511.

[29] Simon Le Cleac'h et al. "Fast contact-implicit model-predictive control". In: *arXiv preprint arXiv:2107.05616* (2021).

[30] Erwin Coumans. *Motion Imitation*. https://github.com/google-research/motion_imitation. 2021.

[31] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2019.

[32] Noel Csomay-Shanklin, Victor D Dorobantu, and Aaron D Ames. "Nonlinear Model Predictive Control of a 3D Hopping Robot: Leveraging Lie Group Integrators for Dynamically Stable Behaviors". In: *arXiv preprint arXiv:2209.11808* (2022).

[33] Xingye Da, Ross Hartley, and Jessy W Grizzle. "Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field". In: *2017 IEEE International Conference on Robotics and Automation*. 2017, pp. 3476–3483.

[34] Xingye Da et al. "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion". In: *Conference on Robot Learning*. PMLR. 2021, pp. 883–894.

[35] Min Dai, Xiaobin Xiong, and Aaron Ames. "Bipedal Walking on Constrained Footholds: Momentum Regulation via Vertical COM Control". In: *arXiv preprint arXiv:2104.10367* (2021).

[36] Robin Deits and Russ Tedrake. "Footstep planning on uneven terrain with mixed-integer convex optimization". In: *IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 279–286.

[37] Jared Di Carlo et al. "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control". In: *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2018, pp. 1–9.

[38] Johannes Englsberger, Pawel Kozłowski, and Christian Ott. "Biologically inspired deadbeat control for running on 3D stepping stones". In: *IEEE-RAS 15th International Conference on Humanoid Robots*. 2015, pp. 1067–1074.

[39] Péter Fankhauser, Michael Bloesch, and Marco Hutter. "Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization". In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), pp. 3019–3026. DOI: 10.1109/LRA.2018.2849506.

[40] Péter Fankhauser et al. "Robot-Centric Elevation Mapping with Uncertainty Estimates". In: *International Conference on Climbing and Walking Robots (CLAWAR)*. 2014.

[41] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. "How to discount deep reinforcement learning: Towards new dynamic strategies". In: *arXiv preprint arXiv:1512.02011* (2015).

[42] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science and Business Media, 2008.

[43] Vladimir Gaitsgory, Lars Grüne, and Neil Thatcher. "Stabilization with discounted optimal control". In: *Systems & Control Letters* 82 (2015), pp. 91–98.

[44] Vladimir Gaitsgory et al. "Stabilization with discounted optimal control: the discrete time case". In: (2016).

[45] Manuel Y Galliker et al. "Bipedal Locomotion with Nonlinear Model Predictive Control: Online Gait Generation using Whole-Body Dynamics". In: *arXiv preprint arXiv:2203.07429* (2022).

[46] Scott Gilroy et al. "Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2021, pp. 2132–2139.

[47] Yukai Gong et al. "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 4559–4566.

[48] Ruben Grandia et al. "Multi-layered safety for legged robots via control barrier functions and model predictive control". In: *arXiv preprint arXiv:2011.00032* (2020).

[49] Ruben Grandia et al. "Perceptive locomotion through nonlinear model predictive control". In: *arXiv preprint arXiv:2208.08373* (2022).

[50] Philippe Greiner et al. "Continuous Modulation of Step Height and Length in Bipedal Walking, Combining Reflexes and a Central Pattern Generator". In: *IEEE International Conference on Biomedical Robotics and Biomechatronics*. 2018, pp. 342–349.

[51] Gene Grimm et al. "Examples when nonlinear model predictive control is nonrobust". In: *Automatica* 40.10 (2004), pp. 1729–1738.

[52] Gene Grimm et al. "Model predictive control: for want of a local control Lyapunov function, all is not lost". In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 546–558.

[53] Jessy W Grizzle et al. "Models, feedback control, and open problems of 3D bipedal robotic walking". In: *Automatica* 50.8 (2014), pp. 1955–1988.

[54] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3389–3396.

[55] Tuomas Haarnoja et al. "Learning to walk via deep reinforcement learning". In: *arXiv preprint arXiv:1812.11103* (2018).

[56] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). eprint: 1801.01290.

[57] Matthew Harding et al. "Augmented Neuromuscular Gait Controller Enables Real-time Tracking of Bipedal Running Speed". In: *IEEE International Conference on Biomedical Robotics and Biomechatronics*. 2018, pp. 364–371.

[58] Charles R Hargraves and Stephen W Paris. "Direct trajectory optimization using nonlinear programming and collocation". In: *Journal of guidance, control, and dynamics* 10.4 (1987), pp. 338–342.

[59] Omar Harib et al. "Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking". In: *IEEE Control Systems Magazine* 38.6 (2018), pp. 61–87.

[60] Ross Hartley et al. "Contact-aided invariant extended Kalman filtering for robot state estimation". In: *The International Journal of Robotics Research* 39.4 (2020), pp. 402–430.

[61] Ayonga Hereid and Aaron D Ames. "FROST: Fast robot optimization and simulation toolkit". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2017, pp. 719–726.

[62] Ayonga Hereid et al. "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1447–1454.

[63] Ayonga Hereid et al. "Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization". In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 370–387.

[64] Ayonga Hereid et al. "Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4722–4729.

[65] Jessica K Hodgins and MN Raibert. "Adjusting step length for rough terrain locomotion". In: *IEEE Transactions on Robotics and Automation* 7.3 (1991), pp. 289–298.

[66] Seungwoo Hong, Joon-Ha Kim, and Hae-Won Park. "Real-Time Constrained Nonlinear Model Predictive Control on SO (3) for Dynamic Legged Locomotion". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3982–3989.

[67] Zhao Huihua, Shishir Nadubettu Yadukumar, and Aaron D Ames. "Bipedal robotic running with partial hybrid zero dynamics and human-inspired optimization". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1821–1827.

[68] J. Hwangbo et al. "Control of a Quadrotor With Reinforcement Learning". In: *Robotics and Auto. Letters* 2.4 (Oct. 2017), pp. 2096–2103.

[69] Koji Ishihara, Takeshi D Itoh, and Jun Morimoto. "Full-body optimal control toward versatile and agile behaviors in a humanoid robot". In: *IEEE Robotics and Automation Letters* 5.1 (2019), pp. 119–126.

[70] Ali Jadbabaie and John Hauser. "On the stability of receding horizon control with a general terminal cost". In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 674–678.

[71] Ali Jadbabaie, Jie Yu, and John Hauser. "Receding horizon control of the Caltech ducted fan: A control Lyapunov function approach". In: *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*. Vol. 1. IEEE. 1999, pp. 51–56.

[72] Ali Jadbabaie, Jie Yu, and John Hauser. "Unconstrained receding-horizon control of nonlinear systems". In: *IEEE Transactions on Automatic Control* 46.5 (2001), pp. 776–783.

[73] Nan Jiang et al. "The dependence of effective planning horizon on model accuracy". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. Citeseer. 2015, pp. 1181–1189.

[74] Ryan Julian et al. "Efficient adaptation for end-to-end vision-based robotic manipulation". In: (2020).

[75] Ryan Julian et al. "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning". In: *arXiv preprint arXiv:2004.10190* (2020).

[76] Shuuji Kajita et al. "Biped walking pattern generation by using preview control of zero-moment point". In: *IEEE International Conference on Robotics and Automation*. Vol. 3. 2003, pp. 1620–1626.

[77] Christopher M Kellett and Andrew R Teel. "Results on discrete-time control-Lyapunov functions". In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*. Vol. 6. IEEE. 2003, pp. 5961–5966.

[78] Ashish Kumar et al. "Rma: Rapid motor adaptation for legged robots". In: *arXiv preprint arXiv:2107.04034* (2021).

[79] Joonho Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science robotics* 5.47 (2020), eabc5986.

[80] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. "Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3)". In: *arXiv preprint arXiv:1003.2005* (2010).

[81] Sergey Levine et al. "End-to-End Training of Deep Visuomotor Policies". In: *Journal of Machine Learning Research* 17.1 (Jan. 2016), pp. 1334–1373. ISSN: 1532-4435.

[82] Sergey Levine et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint arXiv:2005.01643* (2020).

[83] Junheng Li, Junchao Ma, and Quan Nguyen. "Balancing Control and Pose Optimization for Wheel-legged Robots Navigating Uneven Terrains". In: *arXiv preprint arXiv:2109.09934* (2021).

[84] Zhongyu Li et al. "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots". In: *arXiv preprint arXiv:2103.14295* (2021).

[85] Bo Lincoln and Anders Rantzer. "Relaxing dynamic programming". In: *IEEE Transactions on Automatic Control* 51.8 (2006), pp. 1249–1260.

[86] Wen-Loong Ma et al. "Efficient hzd gait generation for three-dimensional underactuated humanoid running". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016, pp. 5819–5825.

[87] Zhao Mandi, Pieter Abbeel, and Stephen James. "On the Effectiveness of Fine-tuning Versus Meta-reinforcement Learning". In: *arXiv preprint arXiv:2206.03271* (2022).

[88] Gabriel B Margolis et al. "Learning to Jump from Pixels". In: *arXiv preprint arXiv:2110.15344* (2021).

[89] Carlos Mastalli et al. "Crocoddyl: An efficient and versatile framework for multi-contact optimal control". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2536–2542.

[90] Tad McGeer et al. "Passive dynamic walking". In: *Int. J. Robotics Res.* 9.2 (1990), pp. 62–82.

[91] Takahiro Miki et al. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 7.62 (2022), eabk2822.

[92] Mohamad Shafiee Motahar, Sushant Veer, and Ioannis Poulakakis. "Composing limit cycles for motion planning of 3D bipedal walkers". In: *IEEE Conference on Decision and Control*. 2016, pp. 6368–6374.

[93] Rémi Munos and Csaba Szepesvári. "Finite-Time Bounds for Fitted Value Iteration." In: *Journal of Machine Learning Research* 9.5 (2008).

[94] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[95] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *International conference on machine learning*. Vol. 99. 1999, pp. 278–287.

[96] Q Nguyen et al. "Dynamic walking on stepping stones with gait library and control barrier". In: *Workshop on Algorithimic Foundations of Robotics*. 2016.

[97] Quan Nguyen and Koushil Sreenath. "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints". In: *American Control Conference*. 2016, pp. 322–328.

[98] Quan Nguyen and Koushil Sreenath. "Robust Safety-Critical Control for Dynamic Robotics". In: *IEEE Transactions on Automatic Control* (2021).

[99] Quan Nguyen and Koushil Sreenath. "Safety-Critical Control for Dynamical Bipedal Walking with Precise Footstep Placement". In: *IFAC Analysis and Design of Hybrid Systems*. Atlanta, GA, Oct. 2015.

[100] Quan Nguyen and Koushil Sreenath. "Safety-critical control for dynamical bipedal walking with precise footstep placement". In: *The IFAC Conference on Analysis and Design of Hybrid Systems*. Vol. 48. 27. 2015, pp. 147–154.

[101] Quan Nguyen et al. "3D dynamic walking on stepping stones with control barrier functions". In: *IEEE Conference on Decision and Control*. 2016, pp. 827–834.

[102] Quan Nguyen et al. "Dynamic Bipedal Locomotion over Stochastic Discrete Terrain". In: *International Journal of Robotics Research (IJRR)* (Aug. 2018), pp. 1–17.

[103] Quan Nguyen et al. "Dynamic bipedal locomotion over stochastic discrete terrain". In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1537–1553.

[104] Quan Nguyen et al. "Dynamic Walking on Randomly-Varying Discrete Terrain with One-step Preview". In: *Robotics: Science and Systems (RSS)*. Boston, MA, July 2017.

[105] Quan Nguyen et al. "Dynamic Walking on Randomly-Varying Discrete Terrain with One-step Preview." In: *Robotics: Science and Systems*. Vol. 2. 3. 2017, pp. 384–399.

[106] Quan Nguyen et al. "Dynamic walking on stepping stones with gait library and control barrier functions". In: *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 384–399.

[107] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. "Dynamic terrain traversal skills using reinforcement learning". In: *ACM Transactions on Graphics* 34.4 (2015), p. 80.

[108] Xue Bin Peng et al. "Learning agile robotic locomotion skills by imitating animals". In: *arXiv preprint arXiv:2004.00784* (2020).

[109] Xue Bin Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[110] Marek Petrik and Bruno Scherrer. "Biasing approximate dynamic programming with a lower discount factor". In: *Advances in neural information processing systems* 21 (2008), pp. 1265–1272.

[111] Romain Postoyan et al. "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost". In: *IEEE Transactions on Automatic Control* 62.6 (2016), pp. 2736–2749.

[112] Romain Postoyan et al. "Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 487–492.

[113] Jerry E Pratt and Russ Tedrake. "Velocity-based stability margins for fast bipedal walking". In: *Fast Motions in Biomechanics and Robotics*. Springer, 2006, pp. 299–324.

[114] Quanser. *Linear servo base unit with inverted pendulum*. Apr. 2021. URL: https://www.quanser.com/products/linear-servo-base-unit-inverted-pendulum/.

[115] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[116] Unitree Robotics. *A1*. URL: https://www.unitree.com/products/a1/.

[117] Robert M Sanner and J-JE Slotine. "Gaussian networks for direct adaptive control". In: *IEEE Transactions on Neural Networks* 3.6 (1992), pp. 837–863.

[118] Shankar Sastry. *Nonlinear systems: analysis, stability, and control*. Vol. 10. Springer Science & Business Media, 1999.

[119] Shankar Sastry and Marc Bodson. *Adaptive control: stability, convergence and robustness*. Courier Corporation, 1989.

[120] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[121] Jonah Siekmann et al. "Blind bipedal stair traversal via sim-to-real reinforcement learning". In: *arXiv preprint arXiv:2105.08328* (2021).

[122] Jonah Siekmann et al. "Sim-to-real learning of all common bipedal gaits via periodic reward composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7309–7315.

[123] Avinash Siravuru et al. "Deep visual perception for dynamic walking on discrete terrain". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 418–424.

[124] Avinash Siravuru et al. "The Reaction Mass Biped: Geometric Mechanics and Control." In: *J. Intell. Robotic Syst.* 89.1-2 (2018), pp. 155–173.

[125] Laura Smith et al. "Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World". In: *arXiv preprint arXiv:2110.05457* (2021).

[126] Eduardo D. Sontag. "A 'universal' construction of Artstein's theorem on nonlinear stabilization". In: *Systems and Control Letters* 13.2 (1989), pp. 117–123.

[127] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2. URL: https://doi.org/10.1007/s12532-020-00179-2.

[128] Andrew Taylor et al. "Learning for safety-critical control with control barrier functions". In: *L4DC*. 2020, pp. 708–717.

[129] Andrew J Taylor et al. "A control lyapunov perspective on episodic learning via projection to state stability". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 1448–1455.

[130] Andrew J. Taylor et al. *Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems*. 2019. arXiv: 1903.01577 [cs.RO].

[131] Sangli Teng et al. "An Error-State Model Predictive Control on Connected Matrix Lie Groups for Legged Robot Control". In: *arXiv preprint arXiv:2203.08728* (2022).

[132] Chen Tessler and Shie Mannor. "Reward Tweaking: Maximizing the Total Reward While Planning for Short Horizons". In: *arXiv preprint arXiv:2002.03327* (2020).

[133] Vassilios Tsounis et al. "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3699–3706.

[134] Octavio Villarreal et al. "MPC-based controller with terrain insight for dynamic legged locomotion". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2436–2442.

[135] Tyler Westenbroek et al. "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers". In: *IFAC-PapersOnLine* 54.5 (2021), pp. 19–24.

[136] Tyler Westenbroek et al. "Learning min-norm stabilizing control laws for systems with unknown dynamics". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 737–744.

[137] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. "Hybrid zero dynamics of planar biped walkers". In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 42–56.

[138] Eric R Westervelt et al. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007.

[139] Georg Wiedebach et al. "Walking on partial footholds including line contacts with the humanoid robot atlas". In: *IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 1312–1319.

[140] Alexander W Winkler et al. "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1560–1567.

[141] Guofan Wu and Koushil Sreenath. "Variation-based Linearization of Nonlinear Systems Evolving on $SO(3)$ and S^2". In: *IEEE Access* 3 (Sept. 2015), pp. 1592–1604.

[142] Zhaoming Xie et al. "ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills". In: *Computer Graphics Forum*. Vol. 39. 8. Wiley Online Library. 2020, pp. 213–224.

[143] Zhaoming Xie et al. "Feedback control for cassie with deep reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1241–1246.

[144] Zhaoming Xie et al. "GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model". In: *arXiv preprint arXiv:2104.09771* (2021).

[145] Xiangru Xu et al. "Robustness of control barrier functions for safety critical control". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 54–61.

[146] Chenyu Yang et al. "Dynamic legged manipulation of a ball through multi-contact optimization". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 7513–7520.

[147] Wenhao Yu et al. "Visual-Locomotion: Learning to Walk on Complex Terrains with Vision". In: *5th Annual Conference on Robot Learning*. 2021.

Appendix A

Proof of Theorem 6.1

The overall loss can be written as

$$\mathbb{E}_{x\sim X} L^{(\lambda_1,\lambda_2)}(x,\theta) = M_u(\theta) + \lambda_1 M_1(\theta) + \lambda_2 M_2(\theta). \tag{A.1}$$

For convenience we write

$$\overline{M}_u = \max_{\theta\in\Theta} M_u(\theta) \qquad \underline{M}_u = \min_{\theta\in\Theta} M_u(\theta) \tag{A.2}$$

$$\overline{M}_1 = \max_{\theta\in\Theta} M_1(\theta) \qquad \underline{M}_1 = \min_{\theta\in\Theta} M_1(\theta) \tag{A.3}$$

$$\overline{M}_2 = \max_{\theta\in\Theta} M_2(\theta) \qquad \underline{M}_2 = \min_{\theta\in\Theta} M_2(\theta) \tag{A.4}$$

We first demonstrate that there exists $C_1, C_2 \geq 0$ such that if $\lambda_1 \geq \frac{1}{\varepsilon_1} C_1\lambda_2 + \frac{1}{\varepsilon_1} C_2$ then for each global optimizer $\theta^* \in \Theta$ of $\mathbf{P}^{(\lambda_1,\lambda_2)}$ we must have $\theta^* \in \Theta_{\varepsilon_1}$. To show this consider two points $\theta_1 \in \Theta_0$ and $\theta_2 \notin \Theta_{\varepsilon_1}$. Let $L_k = \mathbb{E}_{x\sim X} L^{(\lambda_1,\lambda_2)}(x,\theta_k)$ for $k \in \{1,2\}$. We have that:

$$L_1 \leq \overline{M}_u + \lambda_2 \overline{M}_2 \quad \text{ and } \quad \underline{M}_u + \lambda_1\varepsilon_1 + \lambda_2 \underline{\mathrm{M}}_2 \leq L_2.$$

Here, the first inequality follows from the fact that $M_1(\theta_1) = 0$ and the second inequality follows from the fact that $M_1(\theta_2) \geq \varepsilon_1$. Combining the inequalities yields:

$$L_1 \qquad \leq \qquad \left(\overline{M}_u - \underline{M}_u\right) \quad - \quad \lambda_1\varepsilon_1 \quad + \quad \lambda_2\left(\overline{M}_2 - \underline{M}_2\right) \quad + \quad L_2$$

Thus, we see that if we set $\lambda_1 > \frac{1}{\varepsilon_1} C_1\lambda_2 + \frac{1}{\varepsilon_1} C_2$ with $C_1 = \overline{M}_2 - \underline{M}_2$ and $C_2 = \overline{M}_u - \underline{M}_u$, then we must have that $L_1 < L_2$. Thus, any $\theta_2 \notin \Theta_{\varepsilon_1}$ cannot be a global minimizer if we choose the constants $C_1, C_2 \geq 0$ as above.

Next, we demonstrate that when we fix $\lambda_1 \geq \frac{1}{\varepsilon_1} C_1\lambda_2 + \frac{1}{\varepsilon_1} C_2$ as we vary λ_2, then there exists $C_3 \geq 0$ such that if we choose $\lambda_2 \geq \frac{1}{\varepsilon_2} C_3$ then any global optimizer θ^* of $\mathbf{P}^{(\lambda_1,\lambda_2)}$ must lie in $\Theta_{\varepsilon_1,\varepsilon_2}$. As established above, we already know that all optimizers of $\mathbf{P}^{(\lambda_1,\lambda_2)}$

must lie in Θ_{ε_1} in this case. Thus, we will now consider the two points $\theta_3 \in \Theta_{0,0}$ and $\theta_4 \in \left\{\theta \in \Theta_{\varepsilon_1} \colon M_2(\theta) > \tilde{M}_2 + \varepsilon_2\right\}$, with $\tilde{M}_2$ defined as in (6.10), so that θ_4 satisfies the desired tolerance for the CBF constraint but not the desired tolerance for the CLF constraint. Again let $L_k = \mathbb{E}_{x\sim X} L^{(\lambda_1,\lambda_2)}(x, \theta_k)$ for $k \in \{3, 4\}$. We then have that

$$L_3 \leq \overline{M}_u + \lambda_2 \tilde{M}_2 \quad \text{and} \quad \underline{M}_u + \lambda_2(\tilde{M}_2 + \varepsilon_2) \leq L_4$$

where we have used the fact that $M_1(\theta_3) = 0$, $M_2(\theta_3) = \tilde{M}_2$ and $M_2(\theta_4) \geq \tilde{M}_2 + \varepsilon_2$. Again combining the inequalities and rearranging terms we see that

$$L_3 \leq \left(\overline{M}_u - \underline{M}_u\right) - \lambda_2\varepsilon_2 + L_4.$$

Thus, we see that if we select $C_3 = \overline{M}_u - \underline{M}_u$ and put $\lambda_2 \geq \frac{1}{\varepsilon_2} C_3$ then it must be the case that $L_3 < L_4$ so that θ_4 is not a minimizer. Thus, we see that if we choose $\lambda_1 \geq \frac{1}{\varepsilon_1} C_1 \lambda_2 + \frac{1}{\varepsilon_1} C_2$ and $\lambda_2 \geq \frac{1}{\varepsilon_2} C_3$ with all of the constants chosen as above then all minimizers of $\mathbf{P}^{(\lambda_1,\lambda_2)}$ must lie in $\Theta_{\varepsilon_1,\varepsilon_2}$, as desired.

Appendix B

Additional Literature Review

Model Predictive Control: We briefly review stability results from the model predictive control (MPC) literature, focusing our discussion on the benefits of using a CLF as the terminal cost. In their simplest form, MPC control schemes minimize a cost functional of the form

$$\inf_{\hat{\mathbf{u}} \in \mathcal{U}^N} J^N_{MPC}(x_k, \hat{\mathbf{u}}) = \sum_{k=0}^{N-1} \big(Q(\hat{x}_k) + R(\hat{u}_k)\big) + \hat{W}(\hat{x}_N)$$
$$\text{s.t. } \hat{x}_{k+1} = F(\hat{x}_k, \hat{u}_k), \quad \hat{x}_0 = x_k,$$

where x_k is the the current state of the real-world system, $N \in \mathbb{N}$ is the prediction horizon, $\{\hat{x}_k\}_{k=0}^N$ and $\hat{\boldsymbol{u}} = \{\hat{u}_k\}_{k=0}^{N-1} \in \mathcal{U}^N$ are a predictive state trajectory and control sequence, Q and R are as above, and $\hat{W} \colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is the terminal cost which is assumed to be a proper function. The MPC controller then applies the first step of the resulting open loop control and the process repeats, implicitly defining a control law $u_{MPC}(x)$. The MPC cost $J^N_{MPC}(x_k, \cdot)$ can be thought of as a finite-horizon approximation of the original cost (7.3) (except that it is defined over an open-loop sequence of control inputs instead of being a cost over policies).

Stability results from the MPC literature focus primarily on the effects of the prediction horizon N and the choice of terminal cost $\hat{W}$. Under mild conditions, for any choice of terminal cost (including $\hat{W}(\cdot) \equiv 0$), the user can guarantee that the MPC scheme stabilizes the system on any desired operating region by making the prediction horizon N sufficiently large [70, 52]. Thus, there is a clear connection between the explicit prediction horizon N in MPC schemes and the discount factor γ, as both need to be sufficiently large if a stabilizing controller is to be obtained (since trajectory optimization problems with longer time horizons are generally more difficult to solve). Indeed, in [111] it was pointed out that the *implicit prediction horizon* $\frac{1}{1-\gamma}$, a factor which shows up in the stability conditions in Proposition 7.1, plays essentially the same role in stability analysis as N for an MPC scheme with no terminal cost when the running cost is $\ell = Q + R$. Thus, much like the 'typical' policy

optimization problems discussed in Section 7.2, MPC schemes with no terminal cost (or one which is chosen poorly) may require an excessively long prediction horizon to stabilize the system.

Fortunately, the MPC literature has a well-established technique for reducing the prediction horizon needed to stabilize the system: use an (approximate) CLF for the terminal cost $\hat{W}$ [72, 70, 52]. Indeed, roughly speaking, these results guarantee that for *any prediction horizon* $N \in \mathbb{N}$ the MPC scheme will be stabilizing if $\hat{W}$ is a valid CLF for the system. Extensive empirical evidence [71] and formal analysis [72] has demonstrated that well-designed CLF terminal costs reduce the prediction horizon needed to stabilize the system on a desired set and increase the robustness of the overall MPC control scheme [51]. Thus, in many ways our cost-reshaping approach can be seen as a way to obtain these benefits in the context of infinite horizon model-free reinforcement learning.

Appendix C

Asymptotic Stability and Lyapunov Theory

C.1 Asymptotic Stability and Lyapunov Theory

Next, we briefly introduce the elements from stability theory and Lyapunov theory which we use extensively throughout this chapter.

C.2 Notation and Terminology

We say that a function $W\colon \mathbb{R}^n \to \mathbb{R}$ is *positive definite* if $W(0) = 0$ and $W(x) > 0$ if $x \neq 0$. Let $\alpha\colon [0,\infty) \to [0,\infty)$ be a continuous function. We say that α is in class $\mathcal{K}$ (denoted $\alpha \in \mathcal{K}$) if $\alpha(0) = 0$ and α is strictly increasing. If in addition we have $\alpha(r) \to \infty$ as $r \to \infty$ when we say that α is in class $\mathcal{K}_\infty$ (denoted $\alpha \in \mathcal{K}_\infty$). Let $\beta\colon [0,\infty)\times[0,\infty)$ be a continuous function. We say that β is in class $\mathcal{KL}$ if for each fixed $t \in [0,\infty)$ the function $\beta(\cdot,t)$ is in class $\mathcal{K}$ and for each fixed $r \in [0,\infty)$ we have $\beta(r,t) \to 0$ as $t \to \infty$.

C.3 Basic Stability Results

Definition C.1. *We say that the closed loop system $x_{k+1} = F(x_k, \pi(x_k))$ is* asymptotically stable on the set $D \subset \mathbb{R}^n$ *if there exists $\beta \in \mathcal{KL}$ such that for each initial condition $x_0 \in D$ and $k \in \mathbb{N}$ the closed-loop trajectory satisfies:*

$$\|x_k\|_2 \leq \beta(\|x_0\|_2, k). \tag{C.1}$$

Analogously, if the preceding condition holds then we say that π asymptotically stabilizes (7.1).

In words, the definition says that π asymptotically stabilizes (7.1) if all trajectories of the closed-loop system $x_{k+1} = F(x_k, \pi(x_k))$ converge to the origin. Asymptotic stability is a difficult property to verify directly as it requires reasoning about the infinite-horizon behavior of trajectories. Lyapunov functions are a powerful analysis tool which can verify asymptotic stability with a 'one-step' criterion:

Definition C.2. *We say that the positive definite function $W\colon \mathbb{R}^n \to \mathbb{R}$ is a Lyapunov function for the closed-loop system $x_{k+1} = F(x_k, \pi(x_k))$ if for each $x \in \mathbb{R}^n$ we have:*

$$W(F(x, \pi(x))) - W(x) < 0. \tag{C.2}$$

Intuitively, the Lyapunov function W can be thought of as an energy-like function for the closed loop system $x_{k+1} = F(x_k, \pi(x_k))$. In this light, the condition (C.2) ensures that the 'energy' of the closed-loop system is decreasing at each point in the state-space. This condition guarantees that the closed-loop system is asymptotically stable [118], and is a simple algebraic condition. Note that while control Lyapunov functions are defined formally for the open-loop dynamics (7.1), a Lyapunov function is defined for a particular set of closed-loop dynamics. That is, a control Lyapunov function W for $x_{k+1} = F(x_k, u_k)$ becomes a Lyapunov function for the closed-loop dynamics $x_{k+1} = F(x_k, \pi(x_k))$ after we apply a control law π which satisfies $W(F(x, \pi(x))) - W(x) < 0$ for each $x \in \mathcal{X}$.

Appendix D

Missing Proofs and Intermediate Results

Lemma D.1. *The composite function* $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi = W + \gamma \tilde{V}_\gamma^\pi \colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ *is positive definite.*

Proof. Note that we can re-write the reshaped cost (7.10) as

$$\tilde{V}_\gamma^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k \Big([W(x_{k+1}) - W(x_k) + \ell(x_k, \pi(x_k))] \Big), \tag{D.1}$$

where $\{x_k\}_{k=0}^\infty$ is the state trajectory generated by the policy π from the initial condition $x_0 \in \mathcal{X}$. By rearranging terms we can rewrite this expression as:

$$\tilde{V}_\gamma^\pi(x_0) = -W(x_0) + (1-\gamma) \sum_{k=0}^{\infty} \gamma^k W(x_{k+1}) + \sum_{k=0}^{\infty} \gamma^k \ell(x_k, \pi(x_k)) > -W(x_0) + Q(x_0) \tag{D.2}$$

where we have used the fact that W and ℓ are both non-negative, and that $\ell(x_0, \pi(x_0)) > Q(x_0)$. Thus, using this expression we see that

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x_0) = W(x_0) + \gamma \tilde{V}_\gamma^\pi(x_0) > (1-\gamma) W(x_0) + \gamma Q(x_0), \tag{D.3}$$

Since Q and W are assumed to be positive definite functions this demonstrates that $\boldsymbol{\mathcal{V}}_\gamma^\pi$ is in fact positive definite, since a convex combination of positive definite functions is positive definite. The proof is concluded by noting that the choice of γ and π is arbitrary, and thus the conclusion that $\boldsymbol{\mathcal{V}}_\gamma^\pi$ is positive definite holds for all policies and discount factors. □

D.1 Proof of Theorem 7.1

Proof. Lemma D.1 demonstrates that $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi = W + \gamma \tilde{V}_\gamma^\pi \colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is a positive definite function. Using the hypotheses of the results with the inequality (7.18) we obtain

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi\big(F(x, \pi(x))\big) - \tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) \leq (-1 + (1-\gamma)[\tilde{C} + \tilde{\delta}]) Q(x). \tag{D.4}$$

Note that if $\tilde{C} + \tilde{\delta} < \frac{1}{1-\gamma}$ then the right hand side of (7.2) will be negative definite, which establishes that π asymptotically stabilizes the system. □

D.2 Proof of Lemma 7.1

Proof. Consider a policy $\bar{\pi} \in \Pi$ defined for each $x \in \mathcal{X}$ by:

$$\bar{\pi}(x) \in \arg\inf_{u \in \mathcal{U}} W(F(x,u)) - W(x) + \ell(x,u) \leq 0, \tag{D.5}$$

where the preceding inequality follows directly from the assumptions made in the Lemma. Next, for a given initial condition $x_0 \in \mathcal{X}$ let $\{x_k\}_{k=0}^{\infty}$ be the state trajectory generated by $\bar{\pi}$. The corresponding reshaped cost is given by

$$\tilde{V}_{\gamma}^{\bar{\pi}}(x_0) = \sum_{k=0}^{\infty} \gamma^k \Big([W\big(F(x_k, \bar{\pi}(x_k))\big) - W(x_k)] + \ell(x_k, \bar{\pi}(x_k)) \Big) \tag{D.6}$$

$$\leq \sum_{k=0}^{\infty} \gamma^k (0) \tag{D.7}$$

$$\leq 0, \tag{D.8}$$

which demonstrates the desired result, since the initial condition and discount factor were chosen arbitrarily. □

Appendix E

Additional Experiment Details

We now provide more details of the experimental results reported in Section 7.4 and also additional evaluations. While we have chosen to minimize costs in the main portion of the chapter, as this is more consistent with the notation used in the literature on Lyapunov theory and the stability of dynamic programming, most RL algorithms take in rewards that are to be maximized. Thus, for the sake of consistency with practical implementations, in this section we report the reward functions used in our code, which are simply the costs from before with the sign flipped.

For training from hardware data, we used asynchronous off-policy updates, similar to the framework presented in [54]. In particular, we have two separate threads, with one running episodes on the hardware system with the latest available policy and adding the transition data to the replay buffer, and the other one sampling from this buffer and performing the actor and critic updates. We only synchronize the policy network weights at the beginning of each episode.

E.1 A1 Quadruped Results

To illustrate the efficacy of our approach, we run two sets of experiments with the A1 robot: 1) accurately tracking a target velocity when the gains k_p and k_d are not well tuned (Section 7.4); and 2) accurately tracking the height of the robot with an unknown load attached to it. Here we provide additional details of experiments related to these experiments. For both settings, we use the locomotion controller presented in [34, Section 3.2] as our nominal baseline controller. This controller uses a linearized rigid-body model to formulate a quadratic-program (QP)-based controller to track a desired body pose of the robot. Specifically, the following QP is solved to obtain the ground reaction forces f for the feet in contact with the ground:

$$\begin{aligned} \min_{f} \ & \|\mathbf{M}f - \tilde{g} - \ddot{q}_d\|_Q + \|f\|_R \\ \text{s.t. } & f_z \geq 0, \\ & -\mu f_z \leq f_x \leq \mu f_z, \\ & -\mu f_z \leq f_y \leq \mu f_z, \end{aligned} \tag{E.1}$$

where $\mathbf{M}$ is the inverse inertia matrix of the rigid body, $\tilde{g} := [0, 0, g, 0, 0, 0]$ denotes the acceleration due to gravity and $\ddot{q}_d \in \mathbb{R}^6$ are the desired pose accelerations of the robot's body. In particular, the desired accelerations are obtained using a PD controller,

$$\ddot{q}_d = -k_p(q - q_d) - k_d(\dot{q} - \dot{q}_d), \tag{E.2}$$

with $q \in \mathbb{R}^6$ denoting the robot's body pose.

Next, we provide further details for each set of experiments on the A1 robot.

Velocity Tracking for A1 Quadruped

When the feedback gains $k_p, k_d \in \mathbb{R}^6$ are not well tuned, large tracking errors in the forward speed of the robot can persist as illustrated in Fig. 7.2 (left). To compensate for the increased tracking error, we learn a policy π_θ (MLP with two hidden layers of size 32×32) that outputs an additional acceleration term in (E.2), making the final desired acceleration $\ddot{q}_d = -k_p(q - q_d) - k_d(\dot{q} - \dot{q}_d) + \pi_\theta$. π_θ can therefore be viewed as a learned fine-tuning policy with respect to a model-based controller. The observations for the RL agent include the forward and lateral velocity, the roll and pitch orientation and the desired forward velocity of the robot. The actions include offsets to the desired forward and lateral accelerations.

The policy π_θ is learned directly on the robot hardware using a CLF W designed for the nominal rigid body dynamics of the robot following the procedure described in [12]. For training, we use SAC [56] with the reward $r_k = \frac{(W(F(x_k,u_k)) - W(x_k))}{\Delta t_k} + \lambda\|u_k\|^2$. The CLF term in the reward allows us to use a discount factor $\gamma = 0$, which considerably reduces the complexity of the learning problem. Indeed, within only 5 minutes of data collected from the robot hardware, our method is able to significantly reduce the tracking error in the forward velocity compared to the nominal locomotion controller, as shown in Figure 7.2 (left).

Height Tracking with an Unknown Load

In this experiment, we use the same base controller and an equivalent offset policy π_θ as in the previous set-up and attempt to track a target gait. The CLF is designed to stabilize to the target gait as in the previous experiment. Figure E.1 plots the tracking error of the learned controller versus the nominal controller after only 1 minute of training data. As the figure demonstrates, our approach is able to significantly decrease the error to about one-third its nominal value with only a small amount of data.

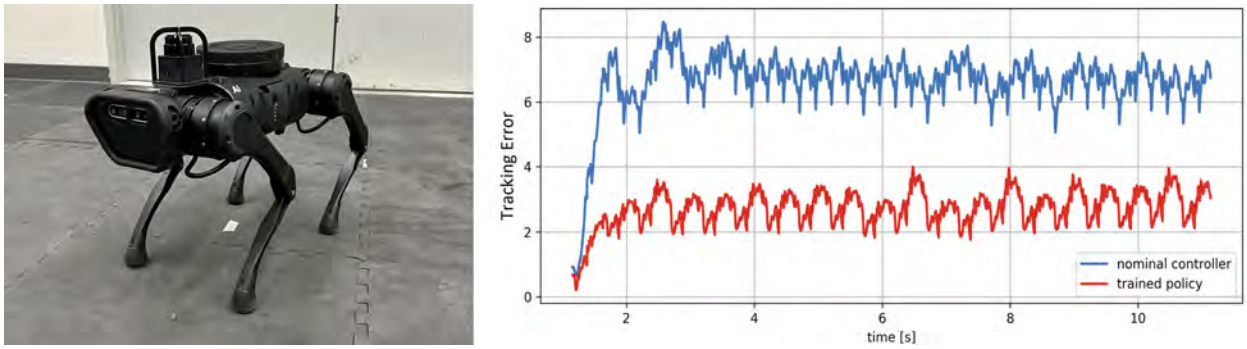


Figure E.1: Comparison between nominal controller and learned policy after training on 60s of real-world data on the A1 robot with an added 10lb weight. The learned policy is able to significantly reduce the tracking error caused by the added weight.

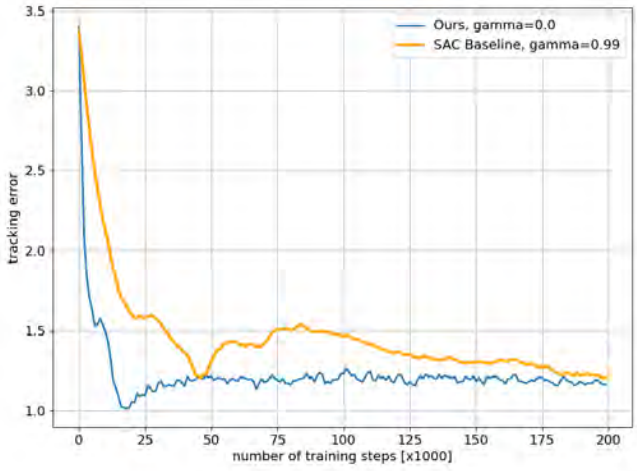


Figure E.2: Cumulative gait tracking error (lower is better) over 10s rollouts at different stages of the simulated fine-tuning benchmark comparison of the A1 quadruped with an unknown load. In orange, we show the results of fine-tuning using SAC with a standard RL cost which penalizes the distance to the desired gait with a discount factor of $\gamma = 0.99$. In blue, we plot the performance of our cost reshaping method with SAC and a discount factor of $\gamma = 0$. For both cost formulations, we plot the discount factor that led to the best performance.

To verify that our method out-performs the baseline for this task, we run a simulated benchmark comparison similar to the A1 simulation study for velocity tracking that was presented in Section 7.4 of the chapter. For this case, we reproduce the unknown load hardware experiment in simulation by adding a 10lb weight to the robot. When testing our method, we again use SAC with the same reward formulation from the hardware experiments above. For the baseline reward, we penalize the distance to the target that we want to track. Figure E.2 depicts the best results that we have been able to obtain for each cost formulation across different discount factors and training hyper-parameters. As Fig. E.2 depicts, our approach quickly converges to a stable walking controller which closely tracks the references after only around 22 thousand steps of the environment. The baseline does not match this performance until it has had access to around 48 thousand steps, and takes much longer to consistently approach the performance of our method.

E.2 Cartpole Results

We first provide plots and give additional details for the cartpole experiments presented in Section 7.4. Then, we present a comparison of the performance of our approach with respect to a typical fine-tuning method on a simulator of the cartpole system.

Additional Details of the Cartpole Hardware Fine-tuning Experiments

For the cartpole experiments presented in Section 7.4, we used a Quanser Linear Servo Base Unit with Inverted Pendulum [114], with a pendulum length of 60cm. The system has 4 states, $x = [p,\ \alpha,\ \dot{p},\ \dot{\alpha}] \in \mathbb{R}^4$, corresponding to the cart position p, the pendulum angle α, and their respective velocities. The control input is the voltage applied to the motor that actuates the cart $u \in \mathbb{R}$.

We first train a SAC agent in simulation using a 'conventional' RL reward that penalizes the distance to the equilibrium, control effort, and includes a penalty if the cart goes off-bounds $r(x_k, u_k) = -0.1\ (5\alpha_k^2 + p_k^2 + 0.05u_k^2) - 5 \cdot 10^3 \cdot \mathbb{1}(|p_k| \geq 0.3)$. The observations of the RL agent are state measurements, the actions are direct voltage commands with limits set to $|u| < 10$V as specified by the manufacturer, and the simulation is run at 100Hz. In order to obtain a stabilizing swing-up policy with this traditional reward, a high discount factor is needed, so we use $\gamma = 0.999$. After around 15 thousand seconds of simulation data with a learning rate of $5 \cdot 10^{-4}$, the RL agent learns to consistently swing-up and balance the pendulum at the upright position in simulation. However, when deployed on the cartpole hardware system, the policy from simulation fails to obtain successful swing-up behaviors due to the sim-2-real gap, as shown in the attached video.

To tackle these issues, we exploit the fact that SAC uses a feedforward neural network to approximate the discounted value function of the problem, and we use this approximate

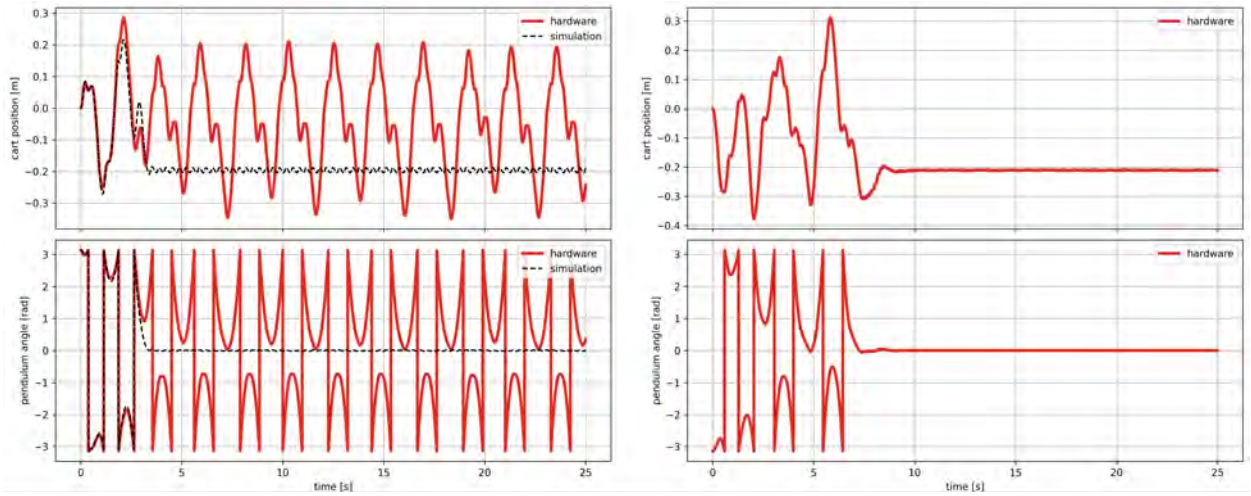


Figure E.3: Experimental plots of the cart position and pendulum angle of the cartpole system. (left) The policy trained only in simulation fails to bring the real cartpole system to the upright position; (right) by fine-tuning the learned policy with $20s$ of real-world data using our CLF-based reward function, we obtain a successful policy.

value function (after 18,600 seconds of data) as a CLF candidate to fine-tune the learned policy directly on hardware.

Thus, we learn on hardware a fine-tuning policy u_ψ (MLP with 2 hidden layers of 64×64) whose actions are added to the ones of the policy trained on simulation u_ϕ (MLP with 2 hidden layers of 400×300). The episodes are 10 seconds long, and the policy is run at 500Hz, with each episode consisting of 5000 data points. The action space limits for this new policy are set to $|u_\psi| < 4\text{V}$ but we still have a saturation of the total voltage $|u_\phi + u_\psi| < 10\text{V}$. The reward for this new policy is $\hat{r}(x_k, u_k) = \Delta V_\theta(x_k, u_k) - 0.1 \cdot (5\alpha_k^2 + p_k^2 + 0.05u_k^2)$, where V_θ is the value function network of the SAC agent that was trained in simulation. This allows us to set the discount factor $\gamma = 0$ for the offset policy learned on hardware and therefore greatly reduce the complexity of the learning problem. After only one episode of 10 seconds of real-world data we obtain a policy that manages to swing-up the pendulum to the upright position, and stabilizes it at the top. However, the behavior near the top is not smooth, and it fails for some different initial conditions. After training with another episode of 10 seconds of data, we obtain a policy that consistently manages to swing-up and balance the pendulum at the top, while the cart stays in-bounds. The plots in Fig. E.3 (right) show the cart position and the pendulum angle when deploying the fine-tuned policy in the real Quanser cartpole system. The plots in Fig. E.3 (left) show the results when using the policy that has been only trained in simulation, and how its performance is very different when deployed in simulation vs in hardware. A video with the results of the cartpole experiments can be found in https://youtu.be/l7kBfitE5n8, and a sequence of

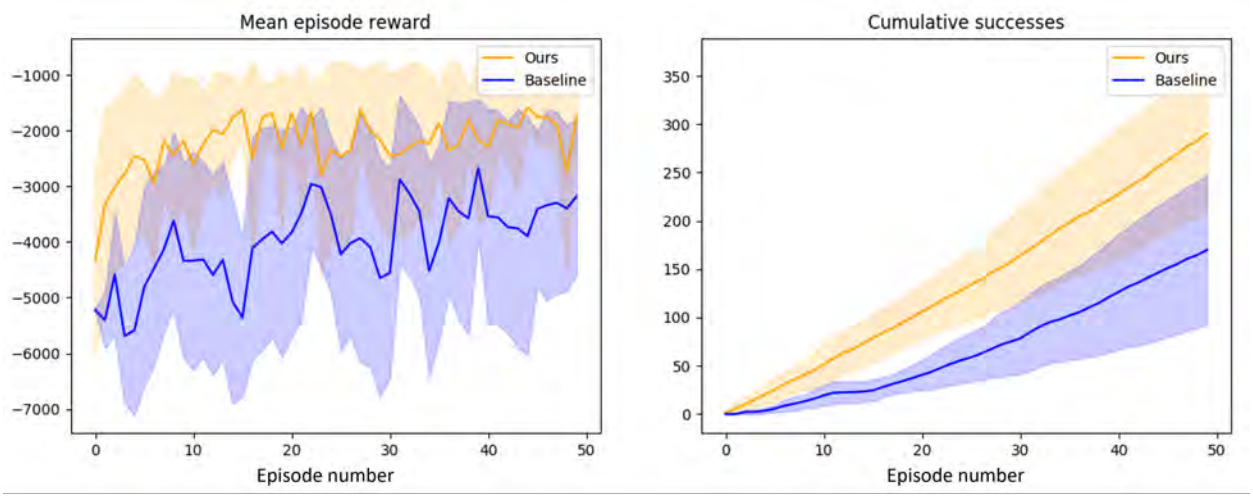


Figure E.4: Comparison of the simulation results of fine-tuning a cartpole swing-up policy after adding model mismatch. A policy trained on a nominal dynamics model of the cartpole fails when deployed on the new dynamics. In blue, we show the results of continuing to train the agent with the original costs and discount factor. In orange, we fine-tune using our reshaping method with the pre-trained value function and a discount factor of $\gamma = 0$. For each episode of training on the new dynamics model, we compare the performance of both methods when running the cartpole from 10 initial conditions: (on the left) the average original reward without the CLF term, and (on the right) the cumulative number of successful swing-ups. The plots show the mean and standard deviation of the results over 10 different training random seeds.

snapshots of a successful experiment that uses the fine-tuned policy can be found in Figure 7.1.

Cartpole Simulation Baseline Comparison with a Typical Fine-tuning Method

As explained at the beginning of the chapter, previous work has shown that using hardware data to fine-tune a policy that has been pre-trained in simulation is a powerful approach to tackle the sim-2-real gap problem (e.g. [125, 75, 74, 87]). These methods typically take the RL agent trained in simulation and continue its learning process using hardware data, the original cost function and discount factor (see e.g. [125]). In contrast, our proposed approach stops the simulation training of u_ϕ and learns a smaller offset policy u_ψ from hardware data using a separate learning process that has a different reward function $\hat{r}$ (with the CLF candidate being the learned value function in simulation) and a smaller discount factor (in this case $\gamma = 0$).

In Figure E.4, we compare in simulation the results of using this standard fine-tuning approach with those obtained with our method. For both approaches, we first pre-train a

policy π_ϕ and value function V_θ on a nominal set of dynamics using SAC and the reward $r(x_k, u_k) = -0.1\ (5\alpha_k^2 + p_k^2 + 0.05u_k^2) - 5 \cdot 10^3 \cdot \mathbb{1}(|p_k| \geq 0.3)$, and then perturb the parameters of the simulator to introduce model mismatch for the fine-tuning phase. Specifically, we increase the weight and friction of the cart by 200%; and the mass, inertia and length of the pendulum by a 25%. After doing this, we randomly sample 10 initial conditions around the downright position ($-0.05m \leq p_0 \leq 0.05m$, $-\pi + 0.05rad \leq \alpha_0 \leq \pi - 0.05rad$, $-0.05m/s \leq \dot{p}_0 \leq 0.05m/s$, $-0.05rad/s \leq \dot{\alpha}_0 \leq 0.05rad/s$). We label a trial as success if within 10 seconds of simulation, the pendulum is stabilized in the set $-0.12rad < \alpha < 0.12rad$, $-0.3rad/s < \dot{\alpha} < 0.3rad/s$ and the cart never gets out of bounds ($|p| < 0.3$). The policy u_ϕ trained with data from the nominal dynamics model does not succeed for any of the 10 initial conditions due to the model mismatch. The baseline in Figure E.4 is obtained by emptying the replay buffer and using data from the new environment to continue the training process of u_ϕ with the same reward $r(x_k, u_k)$. On the other hand, as with the hardware experiments, our method takes the learned value function V_θ from the nominal dynamics model and learns an offset policy u_ψ using the modified reward $\hat{r}(x_k, u_k) = \Delta V_\theta(x_k, u_k) - 0.1 \cdot (5\alpha_k^2 + p_k^2 + 0.05u_k^2)$. In Figure E.4, we plot for 10 training random seeds the average original reward $r(x_k, u_k)$ and the cumulative number of successes of the validation episodes ran from the initial conditions mentioned above. The x axis is the number of rollouts of fine-tuning data (each rollout consists of 10 seconds of data). As this figure clearly demonstrates, our approach is able to more rapidly learn a reliable swing-up controller than the baseline. Moreover, as the plot on the left displays, even though we are no longer optimizing for the original reward, by rapidly converging to a stabilizing controller our method still performs better on the original reward than the benchmark.

The above results show that our approach effectively serves to fine-tune policies when the dynamics of the system change. In fact, we have artificially added a severe model mismatch and shown that we can adapt to the new dynamics with a discount factor of 0. This is because the original value function is still a 'good' CLF candidate for the new system. However, if the change in the dynamics is drastic, or if the overall shape of the motion required to complete the task has to be greatly modified, then the value function from the original dynamics may not be a good CLF candidate, and our method might fail. We have observed that for the cartpole example our method is very robust to variations in the parameters of the cart dynamics (in fact, in the above example we are multiplying both friction and mass of the cart by a factor of 3), but that if we drastically reduce the length and mass of the pendulum by a 50%, our method fails. We hypothesize that this might be related to the underactuated nature of the pendulum dynamics. An interesting direction for future work would therefore be to study under which conditions the original value function retains the CLF properties for a new set of dynamics.

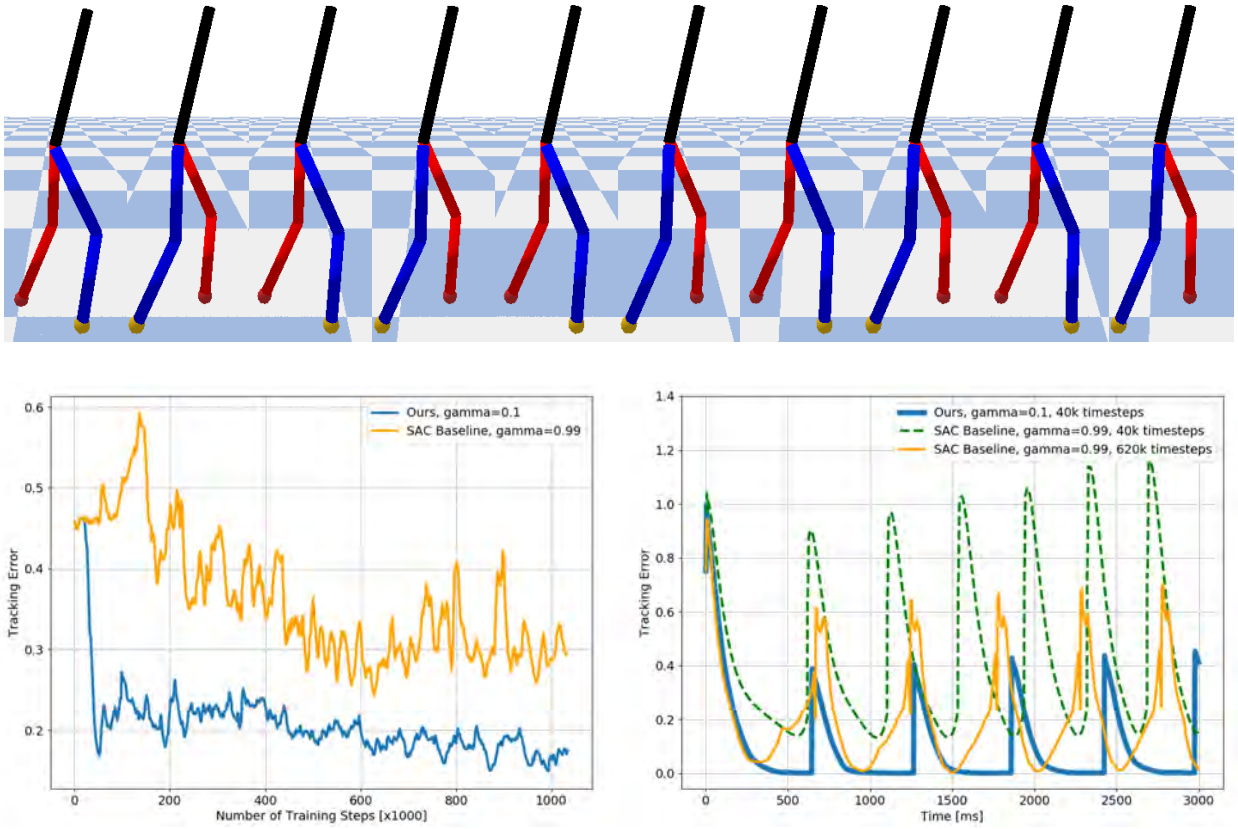


Figure E.5: (Top) Snapshots of RABBIT [25], a five-link bipedal robot, successfully walking with our learned controller in the PyBullet simulator [31]. (Bottom-Left) Average tracking error (lower is better) per episode at different stages of the training process when fine-tuning a model-based walking controller under model mismatch. In blue, using our CLF-based reward formulation and SAC, the robot learns a stable walking gait with only 40k steps (40 seconds) of training data. In orange, with a baseline that uses a typical reward penalizing the tracking error to the target gait, the training takes longer to converge and does not achieve the same performance. The results show the best performance for both method across different discount factors and training hyper-parameters. (Bottom-Right) Comparison of the tracking error of roll-outs of different learned walking policies. In blue, a policy learned with 40k steps of the environment using our CLF-based reward. In dashed green, a policy learned using the baseline reward with 40k steps of the environment. In orange, a policy learned using the baseline reward with 620k steps of the environment (best baseline policy). The jumps in tracking error occur at the swing-leg impact times. The policy learned with our reward formulation clearly outperforms the baseline, even when the baseline has 15 times as much data.

Bipedal Walking Results

In this section, we provide further details on applying our design methodology to fine-tune a model-based walking controller for a bipedal robot. As mentioned in Section 7.4, we first design a CLF around the target gait using the nominal model as in [12] to be used in our reward formulation. As a benchmark comparison, we also train policies with a typical reward which penalizes the distance to the target motion. For both approaches we use the SAC algorithm to optimize the policy. We plot the best performance we have been able to obtain from each method by sweeping across different discount factors and algorithm hyper-parameters in Figure E.5. In particular, the top of this Figure depicts snapshots of the stable walking controller our method obtains after only 40k steps of the environment, which corresponds to only 40 seconds of data given the 1kHz frequency of the controller. The bottom left depicts the average tracking error during the training process for both methods. Finally, the bottom-right plots the tracking error over a few representative rollouts. Note that the tracking error for both methods 'jumps' each time one of the feet impacts the ground. These jumps occur when the swing-foot impacts with the ground and are an unavoidable feature of the environment. Thus, in this context a stable walking controller needs to rapidly converge to the target motion over the course of the next step to maintain stability of the walking motion. As the learning curve demonstrates, our approach is able to significantly reduce the average tracking error per episode after only 40k steps of the environment, while the baseline does not reach a similar level of performance even after 1.2 million steps. As the rollouts in the bottom-right demonstrate, our method learns a desirable tracking controller which smoothly decreases the tracking error between each impact event after only 40 thousand steps. In contrast, after 40 thousand steps the baseline controller diverges from the target motion, corresponding to a fall after only a few steps. After 620 thousand steps, the baseline controller is able to maintain the stability of the walking motion, yet the tracking performance is notably worse than our method at 40 thousand steps, despite having access to around 15 times as many samples.

Inverted Pendulum Results

The states of the system are $x = (\theta, \dot{\theta}) \in \mathbb{R}^2$, where θ is the angle of the arm from the vertical position, and the input $u \in \mathbb{R}$ is the torque applied to the joint. In each of the reinforcement learning experiments reported in Section 7.4 for this system we sample initial conditions over the range $-\pi \leq \theta \leq \pi$ and $-0.1 < \dot{\theta} < 0.1$.

We first train a stabilizing controller using a 'typical' cost function of the form $r_k = -\|x_k\|_2^2 - 0.1\|u_k\|_2^2$, and then train a controller using the reshaped cost

$$r_k = -\left[W(F(x_k, u_k)) - W(x_k)\right] - \|x_k\|_2^2 - 0.1\|u_k\|_2^2.$$

We use the soft actor critic (SAC) algorithm [55] and each training epoch consisted of 5 episodes with 100 simulation steps each, where each time step for the simulator is 0.1 seconds. For both forms of cost function, we sweep across different values of discount factors

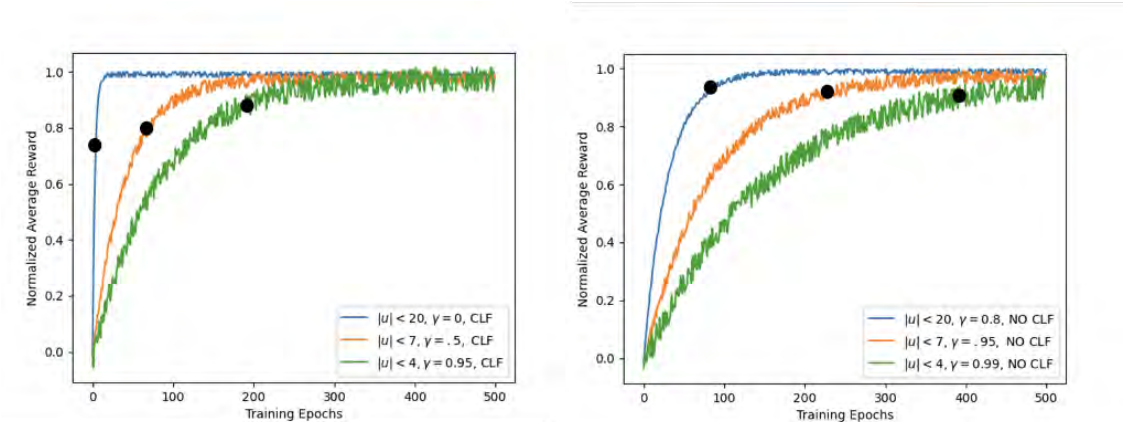


Figure E.6: Learning curves for an inverted pendulum system under different input constraints. The curves plotted correspond to the smallest discount factors that led to stabilizing policies. On the left, the obtained learning curves use a CLF in the reward. On the right, the reward does not include the CLF term. The black dots denote the first stabilizing policy for each training. For each setting we plot the learning curve for the discount factor that achieved the best performance.

(from $\gamma = 0$ to $\gamma = 0.95$ in increments of 0.05 and also tried $\gamma = 0.99$) to 1) determine which values of discount factors lead to stabilizing policies and 2) which discount factor allows the agent to learn a stabilizing controller most rapidly. To determine whether a given controller stabilizes the system we randomly sample 20 initial conditions and see if each trajectory reaches the set $\{x \in \mathbb{R}^n : \|x\|_2 < 0.05\}$ within 20 seconds of simulation. For each scenario, the smallest discount factor that lead to a stabilizing controller was also the discount factor that cause the agent to learn a stabilizing controller with the least amount of data.

Training curves for each of the critical values of the discount factor are depicted in Figure E.6 for each of the cost formulations and input constraints. Each curve indicates the average reward per epoch across 10 different training runs and reports the best results for each scenario after an extensive hyper-parameter sweep. We normalize each training curve so that a reward of 0 indicates the average reward during the first epoch, while a reward of 1 is the largest average reward obtained across all epochs. On each of the training curves the black dot denotes the first training epoch at which a stabilizing controller was obtained.

As illustrated by the plots in Figure E.6 (a), the addition of the CLF enables our method to more rapidly learn a stabilizing controller in each setting and consistently decreases the amount of data that is needed to learn a stabilizing controller, even when W is not a global CLF for the system. However, the effects are more pronounced when the input constraints are less restrictive and W is a better candidate CLF. For example, when $|u| < 20$ our approach is able to learn a stabilizing controller in 5 iterations, whereas it takes 92 iterations with the original cost (our approach takes $\sim 5.4\%$ as many samples). Meanwhile when

$|u| < 4$ our approach takes 198 iterations while the original cost takes 389 iterations (our approach takes $\sim 51\%$ as many samples).Moreover, we observe that larger discount factors are required when $|u| \leq 7$ and $|u| \leq 4$, as W becomes a poorer candidate CLF for these cases.