MODULAR SAFETY-CRITICAL CONTROL OF LEGGED ROBOTS

by

Berk Tosun

B.S., Mechanical Engineering, Middle East Technical University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Evren Samur; his continuous support and understanding guided me through the challenges. I thank Assist. Prof. Sinan Öncü for his lasting support. I would also like to thank Prof. Dr. Duygun Erol Barkana for accepting to be a jury member of my thesis.

This project is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK #118E922).

I am grateful to the SoftExo research group for their generous support and hard work. My gratitude also goes to Ruben Grandia from ETH Zurich, Robotic Systems Lab for clarifying my questions.

I thank my dear friend and colleague Emre Tanfener who has helped me through countless long discussions.

Finally, I thank my family for their love and understanding: my father, Halil, and my brother Bilgehan for inspiring me; his lovely wife Özge and their children Alaz and Uraz for generously spreading joy; my brother Tolgahan for being there for me, mostly for a good laugh; my mother Saniye for her love, wit, and kindness; my sweetheart Hilal Say, for her patience, support and love.

ABSTRACT

MODULAR SAFETY-CRITICAL CONTROL OF LEGGED ROBOTS

With recent performance improvements, legged robots will soon enter our lives to stay. Various control algorithms are already employed in deploying existing robots, and many more algorithms are still in the making. Safety concerns during the operation of legged robots must be addressed to enable their widespread use and ease their development. Especially machine learning-based control methods would benefit from model-based constraints to improve their safety. This thesis presents a modular safety filter to improve safety, i.e., reduce the chance of a fall of a legged robot. The availability of a robot capable of locomotion is assumed, i.e., a nominal controller exists. During locomotion, terrain properties around the robot are estimated through machine learning which uses a minimal set of proprioceptive signals. A novel deep-learning model utilizing an efficient transformer architecture is used for terrain estimation. A quadratic program combines the terrain estimations with inverse dynamics and a novel control barrier function constraint to filter and certify nominal control signals. The result is an optimal controller that acts as a filter and the filtered control signal allows the safe locomotion of the robot. The resulting approach is general and could be transferred with low effort to any other legged system.

ÖZET

BACAKLI ROBOTLARIN MODÜLER GÜVENLİĞİ KRİTİK KONTROLÜ

Hızla yaygınlaşmakta olan ve performansı artan bacaklı robotlar, yakın gelecekte hayatlarımıza dahil olacak. Mevcut robotlarda halihazırda çeşitli kontrol algoritmaları kullanılmakta ve yeni kontrol metotları geliştirilmektedir. Bacaklı robotların yaygın kullanımlarını sağlamak ve geliştirme sürecini kolaylaştırmak için robotun güvenliği dikkatle ele alınmalıdır. Özellikle makine öğrenmesi kullanan kontrol yöntemlerinde güvenlik için model-temelli kısıtların da hesaba katılmasına ihtiyaç vardır. Bu tez, bacaklı bir robotun güvenliğini sağlamak, i.e., düşme olasılığını azaltmak, için modüler bir güvenlik filtresi sunmaktadır. Hareket edebilen bir robotun mevcut olduğu kabul edilmiştir, başka bir deyişle, bir nominal kontrolcü mevcuttur. Hareketli robotun çevresindeki arazi özellikleri, sadece propriyoseptif sinyaller kullanan makine öğrenimi yoluyla tahmin edilmiştir. Derin öğrenmede hızla yaygınlaşan verimli transformer mimarisi ile geliştirilen özgün bir derin öğrenme modeli, arazi özelliklerinin tahmini için kullanılmıştır. Karesel programlama ile arazi tahminleri, ters dinamikler ve özgün bir kontrol set fonksiyonu kısıtıyla birleştirilmiştir. Ortaya çıkan optimal kontrolcü bir filtre görevi görerek nominal kontrol sinyallerinin güvenli olduğunu sertifikalar, aksi halde kontrol sinyalini güvenliği ihlal etmeyecek şekilde değiştirir. Filtrelenmiş kontrol sinyali, robotun güvenli hareketiyle sonuçlanacaktır. Ortaya konan yaklaşım geneldir ve az bir çabayla diğer herhangi bir bacaklı robota aktarılabilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_g	Gravitational acceleration magnitude
$\boldsymbol{f}$	Part of control affine system - autonomous vector field
$\boldsymbol{g}$	Part of control affine system - control vector field
h	Control barrier function
h_e	Exponential control barrier function
$\bar{\boldsymbol{h}}$	Linear Inequality Vector
l	Length of pole in cart-pole system
m_c	Mass of cart in cart-pole system
m_p	Mass of pole in cart-pole system
n_c	Number of contact points
n_v	Number of total joints
n_{va}	Number of actuated joints
n_{vu}	Number of unactuated joints
p	Position of cart in cart-pole system
p_{bound}	Position bound for cart in cart-pole system
$\boldsymbol{q}$	Robot configuration
$\boldsymbol{q_a}$	Decomposed robot configuration - actuated joints
$\boldsymbol{q_u}$	Decomposed robot configuration - unactuated joints
t	Time
$\boldsymbol{u}$	Control signal
v_{bound}	Velocity bound for cart in cart-pole system
$\boldsymbol{v}$	Robot velocities
$\dot{\boldsymbol{v}}$	Robot accelerations
$\boldsymbol{x}$	State vector
$\boldsymbol{A}$	State (or system) matrix
$\boldsymbol{B}$	Input (or selection) matrix
C	Safe set
D	State space
$\boldsymbol{G}$	Gravity terms

$\bar{\boldsymbol{G}}$	Linear Inequality Matrix
$\boldsymbol{J_c}$	Geometric Jacobian relative to the base for feet in contact
$\boldsymbol{J_{flight}}$	Geometric Jacobian relative to the base for feet in flight state
$\boldsymbol{M}$	Mass matrix
$\boldsymbol{M_a}$	Decomposed mass matrix - actuated joints
$\boldsymbol{M_u}$	Decomposed mass matrix - unactuated joints
$\boldsymbol{S}$	Boolean array showing contact state of all feet
$SE(3)$	Special Euclidean group comprising 3D rigid body motions
$\boldsymbol{X}$	Concatenated optimization variables vector
α^{cbf}	Gain of control barrier function
α_1^{ecbf}	Gain one of exponential control barrier function
α_2^{ecbf}	Gain two of exponential control barrier function
θ	Angle of pole in cart-pole system
$\boldsymbol{\lambda}$	Concatenated constraint forces vector
λ_x	Constraint force component in x-axis
λ_y	Constraint force component in y-axis
λ_z	Constraint force component in z-axis
μ	Friction coefficient
$\tilde{\mu}$	Effective friction coefficient
φ	Step phase
$\boldsymbol{\cdot}^{\{i\}}$	Property related to i-th stance foot
$\boldsymbol{\cdot}^{\{j\}}$	Property related to j-th flight (swing) foot

LIST OF ACRONYMS/ABBREVIATIONS

3D	Three Dimensional
6D	Six Dimensional
dt	Simulator timestep
CBF	Control Barrier Function
CoM	Center of Mass
CWC	Contact Wrench Cone
DoF	Degree of Freedom
ECBF	Exponential Control Barrier Function
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
QP	Quadratic Program
URDF	Unified Robot Description Format
ZMP	Zero Moment Point

1. INTRODUCTION

1.1. Background

Although wheels offer a more efficient mode of transportation, it is common to replace them with legs in the case of robot locomotion. As in nature, legs enable robots to travel in challenging environments such as rough terrain, climb stairs and reach tight spots, which might be impossible for wheeled robots. The last two decades have seen significant improvements in legged robots. Leading firms have started commercializing their products; legged robots will become increasingly common in industries starting with environments such as warehouses and oil or gas plants. The next step for legged robots is integrating them further into human lives and it must be done with an emphasis on safety.

In the scope of this work, we will consider the safety of a legged robot as maintaining its ability to locomote; it can be also stated simply but roughly as not falling. It must be noted that there is no concrete definition of safe locomotion as there is no agreed way of quantifying their safety [1, 2].

Contact plays a key role in the dynamics of legged systems. To accomplish their tasks, the robots must generate proper contact forces through their interaction with the environment [3]; in the scope of legged robots and for their main task of locomotion, the environment is reduced to terrain alone. In other words, the terrain is a part of the legged robots' dynamics and it enters the dynamics via contact. Thus, achieving safe legged locomotion requires the control systems to consider the limitations of the terrain, i.e., legged robots must become contact-aware to operate safely.

Legged robots are switched systems. A switched systems' dynamics transition based on events. In the case of a legged robot, changes in the contact states, i.e., making and breaking contact with the ground correspond to the events. After an event, the robot's dynamics change and stay that way until the next event. Both the

environment and the robot's morphology affects the system dynamics and it results in a complex system. In other words, the dynamics of a legged robot are fully defined only by considering the terrain and contact states of its feet, in addition to the ordinary states of position and velocity of its joints. In this text, we frequently talk about the dynamics of legged robots, and it refers to this full description of dynamics which includes contact.

1.2. Motivation

Autonomous robotic devices require their safe operation as one of their fundamental properties. In current literature, the stability of legged robots is generally considered for the case of rigid flat ground such that the complex dynamics of the system are simplified. However, in real-world situations, this simple assumption is often violated; the point of having legs is to clear challenging terrain. Current robots try to handle different types of terrain as a disturbance, and they can tolerate only some of them. To overcome this challenge, some recent work focused on estimating the effect of the terrain [4,5]. More recent work [6,7] have shown only proprioceptive, i.e., blind, measurements combined with deep learning are effective in considering terrain effects while generating control actions. However, there has been no work on a dedicated proprioceptive terrain estimator; in this work, we construct a blind terrain estimator so that the robot gains access to the terrain properties with no additional sensors. With the proposed approach, only the sensors required to control a robot's actuators should be sufficient; thus, there is no need for the cost and complexity of additional sensors such as cameras and lidars. Though additional sensors have their place, a blind estimator can be integrated into most robots with ease.

Another recent development is the introduction of the Control Barrier Function (CBF) framework to guarantee the safe operation of dynamics systems such as legged robots [8]. This work uses CBF to generate constraints such that rough terrain can be safely navigated. The terrain can be estimated by the previously described blind terrain estimator and fed into the CBF constraint to generate an adaptive safety filter. The resulting system will function as a filter, allowing it to be integrated easily.

As the proposed system is modular, it can be integrated with a minimally intrusive fashion into complex control systems, to allow them to execute safe control actions.

In the last decade, machine learning has revolutionized many fields such as computer vision and natural language processing. Robotics significantly benefit from machine learning too, mainly through deep reinforcement-learning [9]. Some of the most impressive, high-performance controllers are obtained with such methods [7]. However, learning-based controllers can be based on simple models or no model at all, i.e., model-free. Combining such controllers with model-based safety constraints would be an effective way to address safety concerns. Thus, our proposed modular safety filter would be especially valuable in the testing and eventual deployment of learning-based systems.

1.3. Aims

The study's long-term goal is to improve the safe operation of robots, by improving their contact awareness. The short-term goal is to add an independent module to a legged robot to guarantee it remains in safe states, reducing the chance of a fall. The central hypothesis is to achieve better safety-critical control performance by estimating the full dynamics of the system.

The first objective is synthesizing a modular, minimally intrusive filter that regulates the control signal to guarantee safety. Using the CBF framework, a safety filter can be synthesized to filter control signals using optimal control in the form of a quadratic program. The resulting module can easily be added to an existing control system, making it a good choice for complex control systems.

The second objective is updating safety constraints by estimating full dynamics which is mainly influenced by the terrain. For control purposes, the dynamics of a legged robot could be significantly simplified by terrain assumptions. By implementing a proprioceptive terrain estimator, the robot will get access to the terrain properties which will include some of the unmodeled dynamics. By taking the terrain into account,

a safer control approach can be achieved. The terrain estimation can be used in the proposed safety filter; alternatively, any other control system can use it as an input.

1.4. Outline

Chapter 2 introduces the literature and related work concerning the problem at hand. The dynamics of floating base robot systems and their safety are discussed. Then, mathematical frameworks which will be used in further chapters are introduced.

Chapter 3 details the proposed solution for safer legged locomotion. The ideas introduced in Chapter 2 are put to work. Several CBF formulations are made in increasing complexity. A novel terrain estimator and a novel CBF constraint are detailed and discussed in this chapter.

Chapter 4 includes the implementation of the methods discussed in the previous chapter. First, the simple system of cart-pole is realized and then the developed methods are extended to a quadruped robot.

In Chapter 5, the performance of the proposed methods is evaluated in simulation environments. The results are reported and discussed.

Chapter 6 concludes the thesis with an outline.

2. PRELIMINARIES AND RELATED WORK

2.1. Dynamics of a Legged Robot

2.1.1. Equations of Motion

Contact is what makes legged systems both interesting and challenging. Legged systems are capable of making and breaking contact with their environment. Links of a robot can transition between 3 contact modes as shown in Figure 2.1.

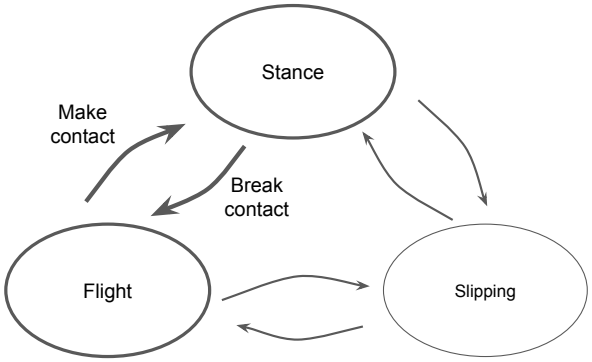

Figure 2.1. All possible contact modes and their transitions.

Constraint forces, also called reaction forces, are introduced into the equation of motion through the contacts in slipping or stance mode. A generic case of a robot capable of making and breaking contact is shown in Figure 2.2.

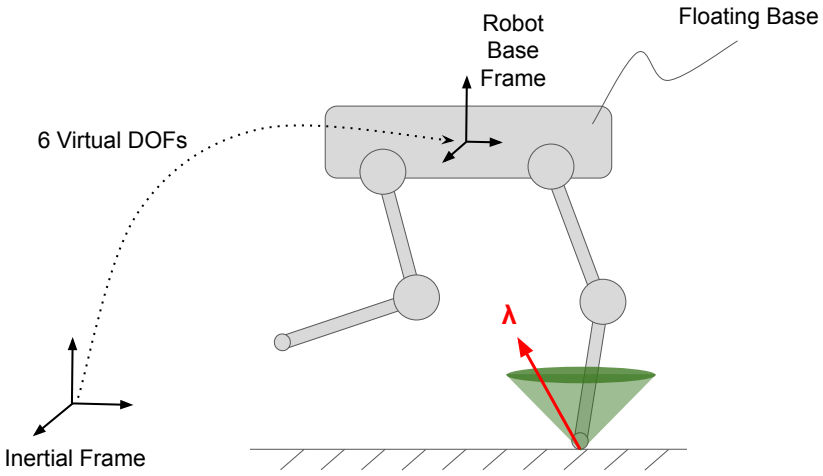

Figure 2.2. Floating base robot with one point foot in contact. The foot is in stance mode, as the reaction force $\boldsymbol{\lambda}$ lies inside the friction cone.

A legged robot is an underactuated control system [10], meaning it has less number of actuators than its degrees of freedom (DoF). Define n_{va} as the number of actuated joints and n_{vu} as the number of unactuated joints. The total DoF is defined as $n_v = n_{va} + n_{vu}$. Underactuation implies $n_{va} < n_v$; as a result, we have limited control authority over the system. The structure of the dynamics will be explored later in this section.

In the scope of robot dynamics [3, 11, 12], contact is very commonly modeled as rigid point contacts. Compliant contact models are generally avoided due to their numerically stiff, unstable condition. For systems such as legged robots, which can make contact in various combinations, a popular choice is to use excess coordinates, i.e., flight coordinates. In this case, a link is selected as the floating base link, and a base frame is assigned to it; then the floating base frame and an inertial frame are connected through 6 virtual, unactuated DoFs. The resulting equation of motion allows us to represent the dynamics concisely:

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{H}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{B}\boldsymbol{u} + \boldsymbol{J_c}(\boldsymbol{q})^T\boldsymbol{\lambda}, \tag{2.1}$$

where

- $\boldsymbol{q} \in \mathbb{R}^{n_{q_u}+n_{va}}$ is the robot configuration vector which includes both actuated and unactuated DoFs. Making the same statement formally: $\boldsymbol{q} = (\boldsymbol{q_u}, \boldsymbol{q_a})$. n_{q_u} is the number of parameters used to represent floating base configuration. In this work, quaternions are used to represent base orientation. Thus $n_{q_u} = 7$; 3 to represent translation, 4 for orientation.
- $\boldsymbol{v} \in \mathbb{R}^{6+n_{va}}$ is the robot velocity, similar to $\boldsymbol{q}$, $\boldsymbol{v} = (\boldsymbol{v_u}, \boldsymbol{v_a})$. Note $\dot{\boldsymbol{q}} \neq \boldsymbol{v}$ unless Euler angles are used to represent base orientation. $\boldsymbol{q}_u \in SE(3)$ includes rotations that can be represented in different ways; regardless of the configuration of q_u, the velocity of the floating base is represented by six parameters.
- $\dot{\boldsymbol{v}} \in \mathbb{R}^{6+n_{va}}$ is the robot accelerations.
- $\boldsymbol{u} \in \mathbb{R}^{n_{va}}$ is the control inputs, i.e., joint torques.
- $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_v \times n_v}$ is the mass matrix.
- $\boldsymbol{H}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^{n_v}$ is the nonlinear effects: centrifugal, Coriolis, and gravity terms.

- $\boldsymbol{B} \in \mathbb{R}^{n_v \times n_{va}}$ is the selection matrix, $\boldsymbol{B} = [\mathbf{0}_{n_{va} \times n_{vu}}\ \boldsymbol{I}_{n_{va}}]^T$.
- $\boldsymbol{J_c}(\boldsymbol{q})^T \in \mathbb{R}^{n_v \times 3n_c}$ is the contact jacobian transposed. Define n_c as the number of contact points. For each contact point, there will be a 3D constraint force.
- $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$ is the vector of constraint forces, i.e., Lagrange multiplier.
- $\boldsymbol{J_c}(\boldsymbol{q})^T\lambda$ is the vector of contact forces, which must be constantly updated based on the contact status.

There is little work in the literature regarding the slipping mode of contact. It is interesting that nature also shies away from it. Following the current literature, we will model contact points in either stance or flight mode. A contact point in stance mode means the contact point is stationary until contact is broken. It implies a kinematic constraint is introduced due to fixed rigid contact points:

$$\boldsymbol{J_c}(\boldsymbol{q})\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_{\boldsymbol{c}}(\boldsymbol{q})\boldsymbol{v} = \mathbf{0}. \tag{2.2}$$

Equation (2.1) and Equation (2.2) define the constrained dynamics of a legged robot. Equation (2.1) can be decomposed into unactuated and actuated parts for further inspection:

$$M_u(q)\dot{v}_u + H_u(q, v) = J_{cu}(q)^T\lambda, \tag{2.3}$$

$$M_a(q)\dot{v}_a + H_a(q, v) = u + J_{ca}(q)^T\lambda. \tag{2.4}$$

Equation (2.3) shows that floating base, i.e., unactuated DoF, can be controlled only indirectly through contact forces [3]. In other words, underactuation limits the robot's center of mass (CoM) acceleration. Since contact forces cannot pull, the CoM cannot accelerate toward the ground faster than gravitational acceleration. The horizontal thrust generated by ground reaction forces is subject to friction and thus limited. Various past works and nature itself show that it is possible to achieve excellent control, even with all the discussed constraints [13].

Constraint forces, i.e., contact forces (in the scope of this work, we will assume a robot only makes contact with the ground; thus, they can be thought of as ground reaction forces as well) can only push; they cannot pull. A 3D contact force exists for each contact point, and three mutually orthogonal vectors can represent it. To

be concise, we will use a coordinate frame aligned with the inertial coordinate frame. Then the contact force at each contact point can be decomposed into its components $\lambda_x, \lambda_y \lambda_z$. Since we consider the case of horizontal locomotion, we can assume the normal component will align with the z-axis. This is commonly stated as the unilateral contact constraint:

$$\lambda_z > 0. \tag{2.5}$$

In line with the literature, we use Coulomb friction to model the friction. Lateral components of the contact forces must lie within the friction cone as illustrated in Figure 2.2. The following constraints must be satisfied; otherwise, the contact switches to the slipping mode:

$$\mu\lambda_z \geq ||\lambda_x||_2, \tag{2.6}$$

$$\mu\lambda_z \geq ||\lambda_x||_2, \tag{2.7}$$

where μ is the static friction coefficient, $|| \bullet ||_2$ is Euclidean norm.

2.1.2. Quantifying Safety: Viability

For traditional control systems, stability is analyzed by established measures such as eigenvalues or phase margins. Legged robots require special attention due to the complexity of quantifying the robot's safety. Safe locomotion of legged robots is seen as a problem of viability instead of Lyapunov stability [2, 14].

The viability kernel is defined as the set of all states where the system remains safe, i.e., it can keep operating. For legged robots, all states where the robot can keep its ability to locomote are included in the system's viability kernel. By definition, any state that is out of the viability kernel cannot return to the viability kernel, and thus it eventually ends up in a failed state. Figure 2.3 illustrates the same statement visually by providing some examples.

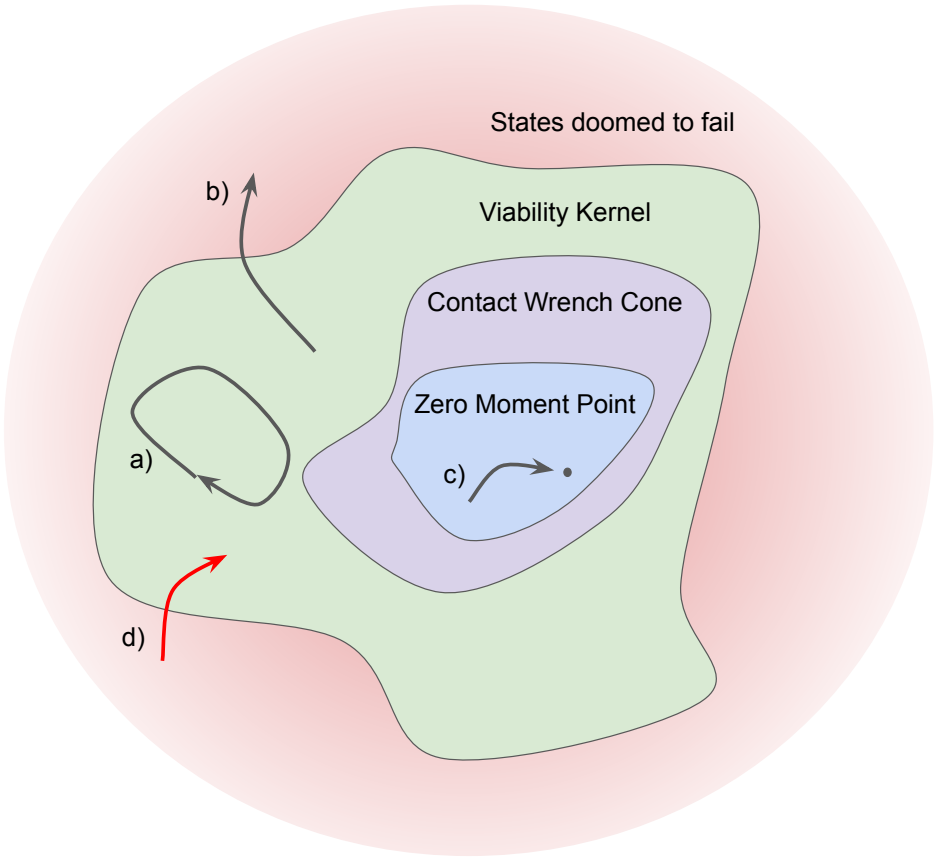

Figure 2.3. Evolution of some sample states and their relation with the viability kernel: a) Limit cycle, stable walking; b) Robot leaving the viability kernel, e.g., it tips over; c) Robot takes a single step while remaining statically stable for the whole duration; d) Impossible state transition.

Legged robot controllers must consider the viability kernel and show effort to stay in it. Legged robots are high DoF, switched dynamical systems; thus, it is intractable to compute a viability kernel for them. However, it is possible to define safety criteria as subsets of the viability kernel. There are several of them that are used widely in the literature:

- Zero Moment Point (ZMP) [15]: Oldest and simple criteria; it requires the robot's CoM to lie in the convex polygon created by the contact points at all times.
- Contact Wrench Cone (CWC) [16]: The set of all possible feasible forces is considered.

- N-step capturability [17]: The set of states where the robot can come to a stop in N-steps at all times.

The established safety criteria have their uses. However, their effect on the resulting closed-loop control system must be considered thoroughly. For example, ZMP requires the robot to always remain statically stable. Such control results in unnatural, slow, and highly energy-inefficient locomotion. CWC improves on ZMP and allows the controller to use a wider set of the viability kernel. Recent literature shows that learning-based robot controllers overperform controllers synthesized by the established safety criteria [7, 9]. Learning-based robot controllers do not necessarily consider any such safety criteria.

2.2. Control Barrier Functions

Control Barrier Function (CBF) [8] arises from a mathematical framework that lends itself to the synthesis of controllers which possess set forward invariance, meaning if the system states are in a certain set initially, it stays in that set for all future timesteps. As a model-based approach, CBF enables theoretical guarantees by checking and restricting derivatives. There have been various applications of CBF on different systems, such as automobiles, quadrotors, and legged robots.

The CBF framework requires us to define a CBF that denotes the system's safety. Define the system states as $\boldsymbol{x} \in D$, where D is the entire state space of the system. Then CBF, $h : D \to \mathbb{R}$, is defined so that returned scalar value must always be positive for safe states, i.e., safe if $h \geq 0$. The framework allows the construction of linear inequality constraints that can be used in optimal controllers. Since CBF relies on a model, it is sensitive to modeling errors common in complex systems such as legged robots. Different learning-based methods are shown to solve such problems [18].

2.2.1. Modular Safety-Critical Control

CBF discussed in section 2.2 are commonly utilized to construct modular safety filters. As shown in Figure 2.4, a safety filter can be inserted into an existing control system after the current, or nominal, controller. Following the literature, we will implement the safety filters as optimal controllers which minimize the interference to the nominal control signal. We make use of a highly efficient convex optimization method known as quadratic programming (QP). In a QP, we can include linear inequality constraints allowing us to incorporate CBF constraints.

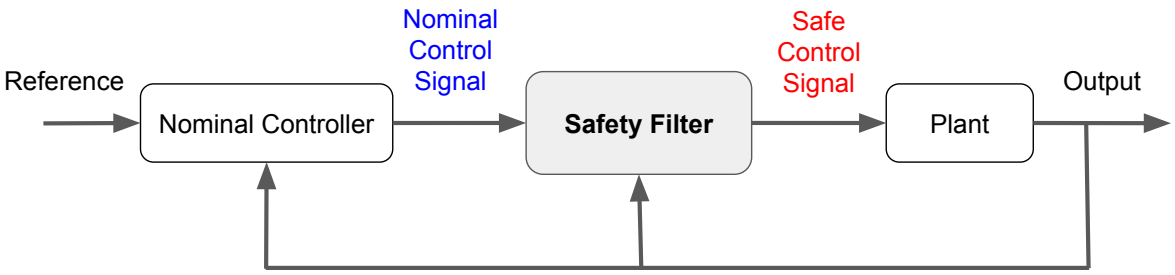

Figure 2.4. A safety filter acting as a filter to modify the nominal control signal.

The safety filter is minimally intrusive, i.e., it will only interfere with the nominal control signal near the boundary of the safe set to make sure the system remains in the safe states, which is illustrated in Figure 2.5.

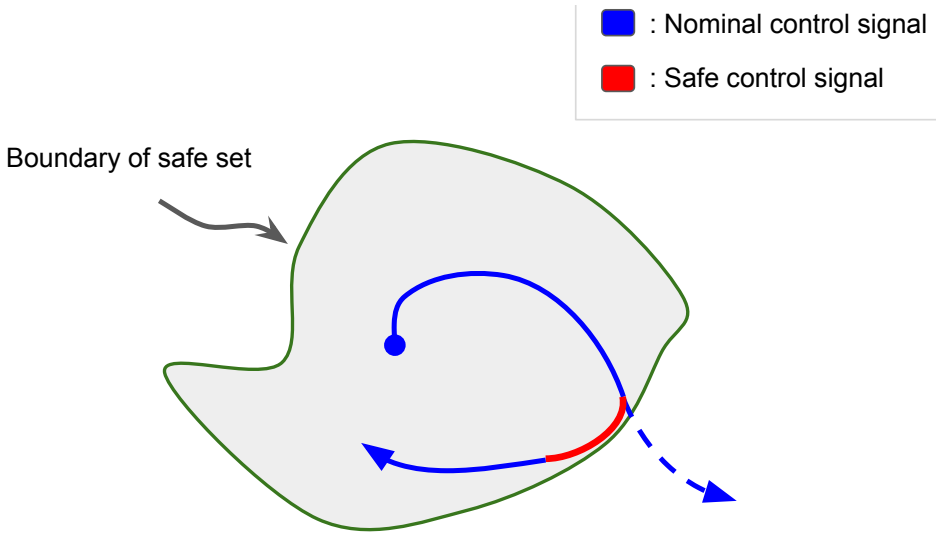

Figure 2.5. Evolution of a dynamic system's states showing set forward invariance with the help of a safety filter. The safety filter plays an active role near the boundary to keep the system in a safe set.

Considering the discussion from Section 2.1.2, CBF are used for restricting parts of the state space to avoid evolutions leaving the viability kernel. Instead of using classical safety criteria such as ZMP, we incorporate the physical limits of the system of interest and intuitive CBF constraints.

2.3. Terrain Estimation by Data-Based Spatial-Temporal Modeling

2.3.1. Graph models

Machine learning, especially deep learning, had a lot of success in various fields recently. There has been extensive work on Graph Neural Networks for graphs with arbitrary structures. Among them, Graph Convolutional Network Network is introduced by [19]. [20] has been the first to extend their work into the skeleton-based action recognition domain. Following their high success, various methods are introduced [21]. The measured quantities, such as 3D coordinates, of the joints are represented by constructing a graph. The quantities for each joint at each time step are represented as a node in the graph. Since the nodes are distributed over both spatial and temporal dimensions, connecting the nodes results in a spatial-temporal model.

2.3.2. Transformers

Transformer architecture has retained its popularity in deep learning since its introduction by [22]. They remain the *de facto* architecture for natural language processing with highly successful applications [23, 24]. By allowing powerful modeling of long-range relationships, transformers obtain an edge over other alternatives in some applications.

Recently, the work on the transformer expanded to different domains, mainly computer vision [25, 26]. We see that they can achieve impressive results with scale and tedious training.

The self-attention employed in the transformer gives it its expressive power, but it is costly to compute. It especially suffers from quadratic memory complexity. Recent studies on transformers enabled various methods to approximate memory-intensive computation of the original transformer. With a growing number of methods, they, called efficient transformers, enable the modeling of long sequences [27, 28].

The regular transformer with the self-attention mechanism is roughly equivalent to a graph network working on a fully-connected graph [29]. Extending the discussion to the efficient transformer architectures, we can think of sparse graphs instead of fully-connected ones.

2.3.3. Modeling Terrain Properties

It is plausible to use deep learning for modeling the terrain properties. Previous work has shown that it is possible to estimate terrain properties by only using proprioceptive measurements [30]. Namely, the main ones are:

- Joint torques
- Joint positions
- Joint velocities
- Base velocity
- Desired velocity

By storing the mentioned measurements for short amounts of time and processing them accordingly, it is possible to achieve very effective reinforcement learning models. Further analysis of the models shows that the model can accurately reconstruct the terrain. With terrain properties, two important ones affecting the robot dynamics are meant: friction coefficient and a rough estimate of the elevation map of the terrain, i.e., macroscopic surface roughness.

A similar problem is skeleton-based action recognition, where graph-based models have enjoyed success [20]. The measurements are structured as a graph and processed.

3. METHODS

3.1. Modular Safety-Critical Control

In the scope of this thesis, two independent modules are developed to enable modular safety-critical control. The first is the safety filter, detailed in Section 3.2, an optimal controller that minimally modifies the nominal control signal and ensures that safety constraints are satisfied. The second is the terrain estimator, discussed in Section 3.3, which estimates terrain properties such as friction. As a legged robot navigates, its environment changes; since its dynamics depend on the environment, updating constraints, would allow the robot to adapt to the terrain, allowing safe locomotion. In our scope, the constraints are targeted towards safety; thus, resulting control will be safer. Figure 3.1 illustrates the proposed system visually; a two-level control loop with different frequencies is shown. Such control schemes are getting common in robotics and are well motivated [31, 32].

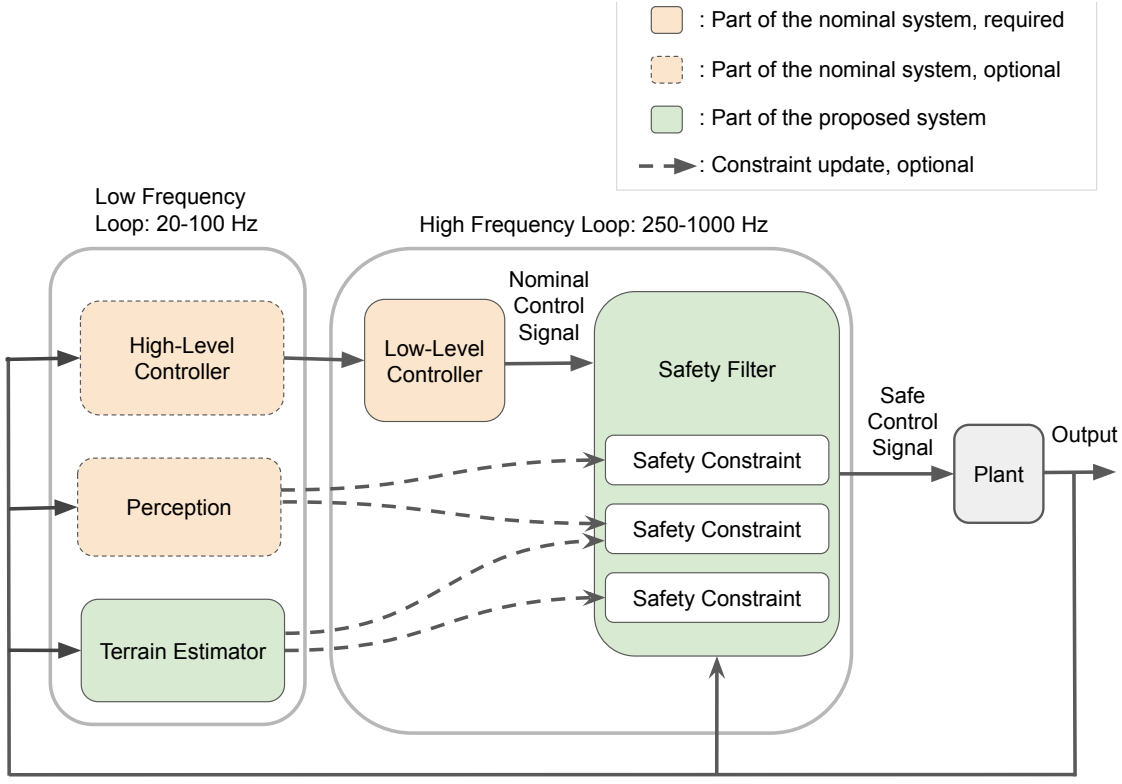

Figure 3.1. Block diagram of a two-level closed-loop control system, as commonly found in robotics.

3.2. Modular Safety Filters

As discussed in Section 3.1 and Section 2.2.1, we will include safety constraints by using modular safety filters. Since we implement the safety filter as an optimization problem, it is flexible and can contain multiple constraints. For the scope of this work, we will focus on CBF constraints.

The CBF framework allows different implementations. The following subsections implement a safety filter satisfying some CBF constraints in increasing complexity. First, a low-complexity control system, cart-pole, is utilized to implement and demonstrate different implementations. Then, we transfer the same developed implementation to a legged robot. Novel parts of our formulation are Section 3.2.1.4, Section 3.2.2 and Section 3.2.3.4.

3.2.1. Cart-Pole System: Naive Formulation

The classical control benchmark of the cart-pole problem is selected as the starting point. The 2-DoF system has one actuated, p, and one unactuated, θ, DoF. It is a simple system as we do not need to model contact dynamics, additionally, there are no switched dynamics since the systems never leave the ground. It is illustrated in Figure 3.2.

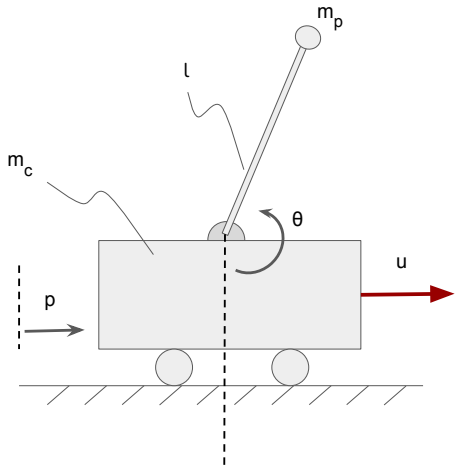

Figure 3.2. Illustration of the cart-pole dynamic system.

3.2.1.1. Dynamics. Since it is a relatively simple system, the equations of motion can be obtained relatively easily by manual methods, e.g., the method of Lagrange. Eventually, any process yields the equations of motion:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{B}u, \tag{3.1}$$

where

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} m_c + m_p & m_p l cos\theta \\ m_p l cos\theta & m_p l^2 \end{bmatrix}, \quad \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} 0 & -m_p l\dot{\theta} sin\theta \\ 0 & 0 \end{bmatrix}, \tag{3.2}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} 0 \\ m_p g l sin\theta \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{3.3}$$

Note, Equation (3.1) is written in a slightly different form than the general form shown in Equation (2.1). For the cart-pole system, instead of six, only one virtual DoF is introduced via p. Then, for $\boldsymbol{q} = (p, \theta)$, $\dot{\boldsymbol{q}} = \boldsymbol{v}$ holds allowing us to use $\dot{\boldsymbol{q}}$ for velocities equivalently.

Equation (3.1) is linearized about the pendulum-up configuration and written in state-space representation:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{p} \\ \ddot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m_c + m_p}{m_c l} a_g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m_p}{m_c} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ p \\ \dot{p} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{m_c} l \\ 0 \frac{1}{m_c} \\ 0 \end{bmatrix} u, \tag{3.4}$$

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u. \tag{3.5}$$

3.2.1.2. Nominal Control. Using the linearized dynamics in Equation (3.5), state control is achieved through an LQR controller. The obtained closed-loop system can simultaneously maintain the pendulum-up state and follow references for cart position. We will call the closed-loop system response as nominal control from now on.

Now that we have the nominal controller set up, we can construct some CBF to enforce safety constraints.

3.2.1.3. Enforce Limits on Cart Velocity: CBF. Regular CBF must have a relative degree of 1, e.g., the control inputs should be available in the first derivative of the CBF. For the cart-pole, this means we can only constrain the velocities: namely the linear velocity of the cart and the angular velocity of the pendulum. Following CBF enforces a symmetric bound on the cart velocity:

$$h(\boldsymbol{x}) = |x_4 - v_{bound}|, \tag{3.6}$$

$$= |\dot{p} - v_{bound}|. \tag{3.7}$$

Then as described in Section 2.2, the safe states are defined as $h(x) \geq 0$. We can enforce this limit by embedding the optimal control problem in a QP. This formulation is known as CBF-QP:

$$\arg\min_{u \in \mathbb{R}^1} \frac{1}{2}||u - u_{nominal}||_2^2 \tag{3.8}$$

$$s.t. \quad \dot{h}(\boldsymbol{x}, u) \geq -\alpha(h(\boldsymbol{x})). \tag{3.9}$$

where Equation (3.8) is the objective function that aims to minimally modify the nominal control signal; Equation (3.9) is the CBF constraint for the cart velocity, it combines the systems' dynamics with the CBF:

$$\dot{h}(\boldsymbol{x}, u) = L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x}), \tag{3.10}$$

where $L_{\boldsymbol{f}} h(\boldsymbol{x}) = \nabla h(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x})$ is the Lie derivative of $\boldsymbol{f}$ and h; $\boldsymbol{f}$ and $\boldsymbol{g}$ are used to define a nonlinear affine control system, in the general form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}. \tag{3.11}$$

Equation (3.11) can be obtained from the state equations, Equation (3.5).

3.2.1.4. Enforce Limits on Cart Position: ECBF. A more powerful CBF variant known as Exponential-CBF (ECBF) can be used in the case of control barrier functions for arbitrarily high-relative degrees [33]; i.e., ECBF generalizes CBF. For the cart-pole system we use it to enforce a symmetric bound on the cart position so that the cart cannot go further than the specified limit. The resulting CBF is of relative degree 2; since acceleration is related to the second derivative of the position.

To utilize the ECBF approach, we first start by defining a CBF for the cart position:

$$h(\boldsymbol{x}) = |x_3 - p_{bound}|, \tag{3.12}$$

$$= |p - p_{bound}|. \tag{3.13}$$

Since the time derivative of Equation (3.12) does not contain the control input, it is not possible to directly use it for the barrier constraint. We solve the issue using the ECBF approach, by defining an auxiliary function $h_e(\boldsymbol{x})$, called ECBF, as:

$$h_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{h}(\boldsymbol{x}) + \alpha_1(h(\boldsymbol{x})). \tag{3.14}$$

Similar to Equation (3.9), an inequality including the control input, u, will be obtained by differentiating $h_e(\boldsymbol{x}, \dot{\boldsymbol{x}})$:

$$\dot{h}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}, u) \geq -\alpha_2(h_e(\boldsymbol{x}, \dot{\boldsymbol{x}})). \tag{3.15}$$

Note, $\dot{h}(\boldsymbol{x})$ in Equation (3.14) is a conventional time derivative, whereas $\dot{h}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}, u)$ in Equation (3.15) requires a Lie derivative as given in Equation (3.10).

3.2.2. Cart-Pole System: Inverse Dynamics Formulation

So far, we have used the CBF-QP formulation given in Equation 3.8 to enforce the barrier constraints. Though it works perfectly fine for a low DoF system such as a cart-pole, the CBF-QP formulation is not the best option for higher DoF systems. It is especially problematic for legged systems which have switched dynamics based on their contact configuration. As explained in Section 2.1.1, the contact forces enter the dynamics through the term $J^T(q)\lambda$ where λ is the constraint forces. In addition, we have kinematic constraints due to rigid contacts. In CBF-QP formulation, we must embed the dynamics into the barrier inequality constraints, as shown in Equation (3.10). This requires two undesirable operations: inversion of the mass matrix and solving for λ [34]. Finding λ is challenging; it requires the assumption that kinematic contact constraints are satisfied. Inversion of the mass matrix tends to get numerically stiff [35]. Thus, it is preferable to have a different formulation without these issues.

We can avoid these issues by using the already mature inverse dynamic schemes. The CBF inequality constraints which are affine in the control input, joint torques can be directly integrated into the inverse dynamics formulation [34,36]. For the cart-pole, we get the following controller, which we will call as ID-CBF-QP:

$$\arg\min_{u} \frac{1}{2}||u - u_{nominal}||_2^2 \tag{3.16}$$

$$s.t. \quad \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{B}u \tag{3.17}$$

$$\dot{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, u) \geq -\alpha(h(\boldsymbol{q}, \dot{\boldsymbol{q}})) \tag{3.18}$$

where $\boldsymbol{q} = (x_1, x_3) = (\theta, p)$ is the robot configuration; Equation (3.17) enforces the physics and Equation (3.18) enforces the barrier constraint. Note, we do not need the Lie derivatives from Equation (3.10) anymore in Equation (3.18); the dynamics are already embedded into the problem by Equation (3.17). Though a regular CBF is shown in this formulation; it can be replaced with an ECBF without any additional effort.

3.2.3. Quadruped Robot: Inverse Dynamics Formulation

Moving on to legged robots, we focus on quadruped robots. A quadruped is preferred because:

- It allows relatively simpler dynamics due to their almost universal point feet design, compared to for example bipeds.
- Quadrupeds generally employ a design with light legs whose dynamics are fast and easier to control.
- There is extensive literature, some software, and access to different control schemes.

3.2.3.1. Dynamics. Quadrupeds are high DoF systems. Thus obtaining and computing their equations of motion is not trivial. We make use of Pinocchio [37], which is a carefully optimized library for modern rigid body algorithms for poly-articulated systems based on revisited Roy Featherstone's algorithms. Pinocchio can build models from URDF (Unified Robotics Description Format), which is the standard format for

representing robot models. Thus, we use Pinocchio with modular software to achieve a general solution, i.e., independent of the robot.

3.2.3.2. Nominal Control. We use some existing nominal controllers from the literature. The control approach does not matter, as long as it generates a control signal for the actuated joints, i.e., computes $\boldsymbol{u_{nominal}}$. Section 4.2.1 details the nominal controller.

In Section 3.1, a modern control scheme is introduced; though the shown system consists of two levels, one for planning and one for acting, note that there are no such restrictions. It is only expected that joint torques are generated by the nominal controller. We develop our objective function in the following sections based on $\boldsymbol{u_{nominal}}$. Alternatively, it is possible to formulate our optimization problem based on desired accelerations, $\dot{\boldsymbol{v}}_{\boldsymbol{desired}}$. Thus, different types of controllers can be supported. For example, if there exists some trajectory planning, it can be added to the formulation by only adjusting the objective function.

3.2.3.3. ID-CBF-QP. For the inverse dynamics controller, we have a similar formulation to that of the cart-pole system given in Section 3.2.2; however, we should add contact dynamics and its related constraints into the formulation:

$$\underset{\boldsymbol{X}=(\dot{\boldsymbol{v}},\boldsymbol{u},\boldsymbol{\lambda})}{\arg\min} \frac{1}{2}||\boldsymbol{u}-\boldsymbol{u_{nominal}}||_2^2 \tag{3.19}$$

$$s.t. \quad \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}}+\boldsymbol{H}(\boldsymbol{q},\boldsymbol{v})=\boldsymbol{B}\boldsymbol{u}+\boldsymbol{J_c^T}\boldsymbol{\lambda} \tag{3.20}$$

$$\boldsymbol{J_c}(\boldsymbol{q})\dot{\boldsymbol{v}}+\dot{\boldsymbol{J}}_{\boldsymbol{c}}(\boldsymbol{q})\boldsymbol{v}=0 \tag{3.21}$$

$$\lambda_z^{\{i\}}>0 \tag{3.22}$$

$$\tilde{\mu}\lambda_z^{\{i\}}\geq|\lambda_x^{\{i\}}| \tag{3.23}$$

$$\tilde{\mu}\lambda_z^{\{i\}}\geq|\lambda_y^{\{i\}}| \tag{3.24}$$

$$\dot{h}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{u})\geq-\alpha(h(\boldsymbol{q},\boldsymbol{v})) \tag{3.25}$$

$$-\boldsymbol{\tau_{max}}\leq\boldsymbol{u}\leq\boldsymbol{\tau_{max}}, \tag{3.26}$$

where we see the familiar terms from Section 2.1.1:

- Equation (3.19) is the objective function. Three different decision variables, concatenated into vector $\boldsymbol{X}$, will be optimized. The decision variables are:
  (i) $\dot{\boldsymbol{v}} \in \mathbb{R}^{n_v}$, accelerations for all degrees of freedom,
  (ii) $\boldsymbol{u} \in \mathbb{R}^{n_{va}}$, torques for actuated joints,
  (iii) $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$, concatenated ground reaction forces from each contact point; since we are working on robots with point feet, each stance foot contributes only one contact point.

  Note, $\dot{\boldsymbol{v}}$ can be included in the objective; for example, it can replace $\boldsymbol{u} - \boldsymbol{u_{nominal}}$ with $\dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}_{\boldsymbol{desired}}$. Or both $\boldsymbol{u}$ and $\boldsymbol{v}$ can be used with weights. This flexibility is one big advantage of optimization-based control.
- Constrained equations of motion are included as constraints:
  (i) Equation (3.20) is the generalized equation of motion.
  (ii) Equation (3.21) is the kinematic constraint to keep contact points stationary.
- Stance feet must maintain their contact mode; it can be done by including friction cone constraints:
  (i) Equation (3.22) is the unilateral contact constraint to avoid loss of contact for each stance foot, foot $\{i\}$.
  (ii) Equation (3.23) and Equation (3.24) are the linearized friction cone constraints to avoid slipping for each stance foot, foot y. We use a square pyramid to get an inner approximation; it requires the use of $\tilde{\mu} = \mu/\sqrt{2}$ as the effective friction coefficient.
- Equation (3.25) is the control barrier constraint.
- Equation (3.26) is the joint torque limits.

Inverse dynamics formulation for legged systems is quite powerful; it allows us to check for many failure cases since it takes contact forces into account, which might cause a fall in case the nominal controller does not consider them. If the nominal controller is already an ID-QP, one can simply add the CBF constraint to the formulation, achieving the same effect. Note that the formulation can be further extended.

3.2.3.4. Enforce Ground Clearance: ECBF. As an inequality constraint, we can incorporate CBFs into the optimization problem, as shown in Equation (3.25). There can be multiple CBF constraints as long as the QP remains feasible. In this instance, we define a novel CBF constraint to keep the feet from hitting the ground by including the terrain profile.

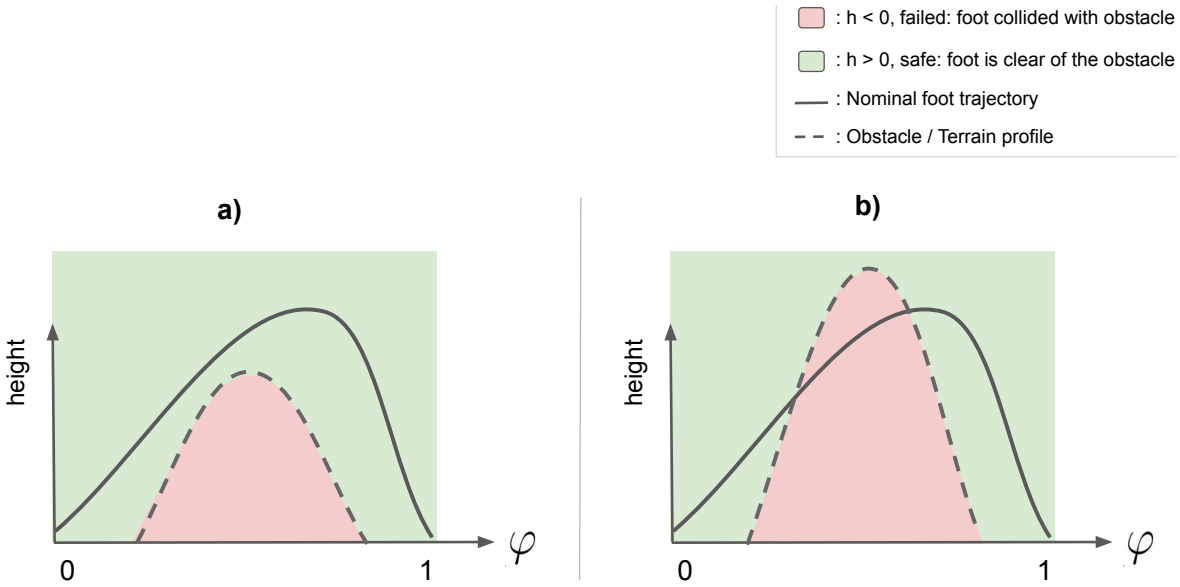

Figure 3.3. Two independent, sample cases of the ground clearance ECBF for an arbitrary obstacle for a single foot. a) Obstacle is already cleared by the nominal trajectory, no interference is required. b) Obstacle must be clarified by adjusting the nominal trajectory, ECBF can do it in an optimal sense.

We apply an ECBF to enforce ground clearance of the robot's end-effectors, i.e., its swing feet; so that its feet do not hit obstacles or adepts to uneven terrain while walking. Figure 3.3 describes the constraint by illustrating two sample cases. We start the formulation with a position-based CBF:

$$h(\boldsymbol{q}, \varphi) = {}^{W}\boldsymbol{p}_z^{\{j\}}(\boldsymbol{q}) - z^{\{j\}}(\varphi), \tag{3.27}$$

where $\varphi \in [0, 1]$ is the phase variable, denoting the progress of the swing phase; ${}^{W}\boldsymbol{p}_z^{\{j\}}(\boldsymbol{q}) : \mathbb{R}^{n_v} \to \mathbb{R}$ is the z-axis position (height) of the $\{j\}$th swing foot in the world frame; $z^{\{j\}}(\varphi) : \mathbb{R} \to \mathbb{R}$ maps the phase variable to desired ground clearance height for the $\{j\}$th swing foot in the world frame. $z(\varphi)$ can be thought as the obstacle heights over the gait period. Note, both $\boldsymbol{q}$ and φ are functions of time, t.

To use the ECBF approach, we need φ and its derivatives. In this work, we define the phase variable, φ as follows:

$$\varphi(t) = \frac{t \mod period_{gait}}{period_{gait}}. \tag{3.28}$$

Its time derivates are given by:

$$\dot{\varphi} = \frac{1}{period_{gait}}, \tag{3.29}$$

$$\ddot{\varphi} = 0. \tag{3.30}$$

Since the ground clearance CBF, h, is a position-based CBF, it cannot be directly used [38]. We have covered a similar case in Section 3.2.1.4; again, we encounter a control barrier function for a second relative degree safety constraint. We solve this problem as we did earlier; using the ECBF approach by defining an auxiliary function, h_e:

$$h_e(\boldsymbol{q}, \boldsymbol{v}, \varphi) = \dot{h}(\boldsymbol{q}, \boldsymbol{v}, \varphi) + \alpha_1 h(\boldsymbol{q}, \varphi). \tag{3.31}$$

Then, the ECBF constraint is:

$$\alpha_2 h_e(\boldsymbol{q}, \boldsymbol{v}, \varphi) + \dot{h}_e(\boldsymbol{q}, \boldsymbol{v}, \dot{\boldsymbol{v}}, \varphi) \geq 0. \tag{3.32}$$

Taking the derivatives and substituting, we get:

$$\alpha_2 h_e(\boldsymbol{q}, \boldsymbol{v}, \varphi) + {}^{W}\ddot{\boldsymbol{p}}_z^{\{j\}}(\boldsymbol{q}) - \ddot{z}^{\{j\}}(\varphi)\dot{\varphi} + \alpha_1({}^{W}\dot{\boldsymbol{p}}_z^{\{j\}}(\boldsymbol{q}) - \dot{z}(\varphi)^{\{j\}}\dot{\varphi}) \geq 0. \tag{3.33}$$

With the help of $\boldsymbol{J_{flight}}$, Jacobian of the flight feet, we recover one of the optimization variables; $\dot{\boldsymbol{v}}$:

$$\alpha_2 h_e(\boldsymbol{q}, \boldsymbol{v}, \varphi) + {}^{W}(\dot{\boldsymbol{J}}_{\boldsymbol{flight}}^{\{j\}}(\boldsymbol{q})\boldsymbol{v} + \boldsymbol{J}_{\boldsymbol{flight}}^{\{j\}}(\boldsymbol{q})\dot{\boldsymbol{v}})_z - \ddot{z}^{\{j\}}(\varphi)\dot{\varphi} + \alpha_1({}^{W}\dot{\boldsymbol{p}}_z^{\{j\}}(\boldsymbol{q}) - \dot{z}^{\{j\}}(\varphi)\dot{\varphi}) \geq 0. \tag{3.34}$$

Note, the ECBF constraint, Equation (3.32), does not directly include the control signal, u. However, the CBF framework requires u in the final constraint. Equation (3.32) can serve as a valid CBF because, $\dot{v}$ appears in affine relation to u in Equation (3.20) [36].

We use readily available, highly optimized QP solvers; as long as the inequality constraint is written in the standard form, we can use any off-the-shelf solver:

$$\bar{\boldsymbol{G}}\boldsymbol{X} \leq \bar{\boldsymbol{h}}, \tag{3.35}$$

where $\bar{\boldsymbol{G}}$ is the linear inequality matrix, $\bar{\boldsymbol{h}}$ is the linear inequality vector; note $\bar{\boldsymbol{G}}$, $\bar{\boldsymbol{h}}$ are not related to $\boldsymbol{G}$ and h, we use bar to avoid any confusion. Reorganizing Equation (3.34) to match the format of Equation (3.35), we get the inequality constraint in the standard form:

$$\begin{bmatrix} -\boldsymbol{J}^{\{j\}}_{\boldsymbol{flight}}(\boldsymbol{q}) & \boldsymbol{0}_{1\times n_{va}} & \boldsymbol{0}_{1\times 3n_c} \\ -\boldsymbol{J}^{\{j+1\}}_{\boldsymbol{flight}}(\boldsymbol{q}) & \boldsymbol{0}_{1\times n_{va}} & \boldsymbol{0}_{1\times 3n_c} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{v}} \\ \boldsymbol{u} \\ \boldsymbol{\lambda} \end{bmatrix} \leq \begin{bmatrix} \alpha_2 h_e(\boldsymbol{q},\boldsymbol{v},\varphi) + {}^{W}(\dot{\boldsymbol{J}}^{\{j\}}_{\boldsymbol{flight}}(\boldsymbol{q})\boldsymbol{v})_z - \ddot{z}^{\{j\}}(\varphi)\dot{\varphi} + \alpha_1(\dot{h}(\boldsymbol{q},\boldsymbol{v},\varphi)) \\ \alpha_2 h_e(\boldsymbol{q},\boldsymbol{v},\varphi) + {}^{W}(\dot{\boldsymbol{J}}^{\{j+1\}}_{\boldsymbol{flight}}(\boldsymbol{q})\boldsymbol{v})_z - \ddot{z}^{\{j+1\}}(\varphi)\dot{\varphi} + \alpha_1(\dot{h}(\boldsymbol{q},\boldsymbol{v},\varphi)) \\ \vdots \end{bmatrix}, \tag{3.36}$$

where we add one row for each foot in flight; since $\boldsymbol{J}^{\{j\}}_{\boldsymbol{flight}}(\boldsymbol{q}) \in \mathbb{R}^{1\times n_v}$ is a row vector.

3.2.4. Moving From Simulation to an Actual Robot

The formulation from Section 3.2.3.3 is sufficient for simulation purposes. However, integration into an actual robot requires more effort. A C++ controller implementation is the *de facto* standard in robotics to run the controller in an embedded system. Another important point is the real-time performance of our optimization problem; it is desired to have high-frequency control loops on the order of 250-1000 Hz. Therefore, it is desired to solve the optimization problem under 1 ms. The time complexity of the optimization problem is $O(n^3)$ where n is the number of decision variables. It is possible to exploit the structure of the dynamics to reduce the number of decision variables, thus, speeding up its execution.

Equation (3.20) and Equation (3.21) can be written with actuated and unactuated parts separated as shown in Equation (2.3). The control signal, $\boldsymbol{u}$, can be expressed as an affine function of other variables. Thus, it is removed from the formulation. This would require the objective function to be formulated with respect to $\dot{\boldsymbol{v}}$ as discussed

in Section 3.2.3.3. Though the problem size decreases moderately, the reductions in solution time are significant due to $O(n^3)$ time complexity [39].

A C++ implementation (with Python bindings) exists from the same group that created Pinocchio: TSID [39] is an optimization-based inverse-dynamics control library that uses Pinocchio for efficient computations. One can extend TSID such that CBF constraints are handled as well. This would achieve a faster, embedded-ready implementation of the formulation from Section 3.2.3.3. An example of such an implementation with a different, not open-sourced, package can be found in [36].

3.2.5. Moving From Point Foot to Planar Foot

Building on top of the previous section, the robot's feet geometry is the necessary change. For robots with a planar foot, point feet are replaced with planar rectangular surfaces. Any planar surface contact can be equally represented by several contact points placed at the vertices of the contact surface.

TSID has support for such geometries. It defines a special force-generator matrix to enable the mapping between multiple 3-D force vectors and 6-D motion vectors [39].

3.3. Data-Based Terrain Estimation

3.3.1. Network Architecture

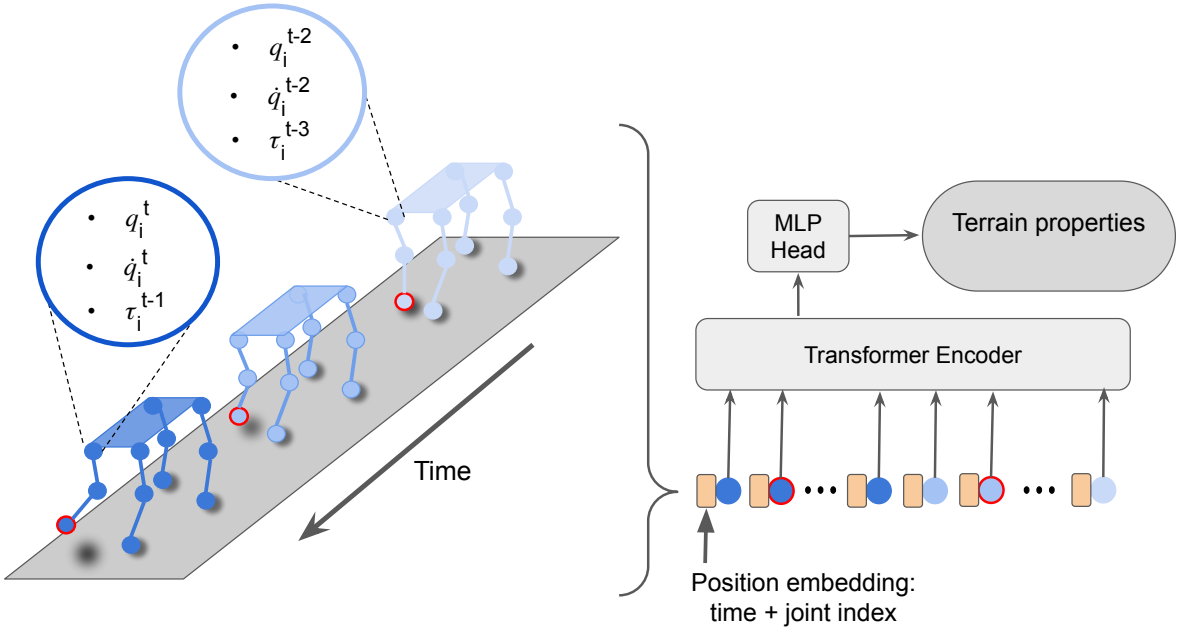

Figure 3.4. The robot moves one of its feet, shown with red, over time. Records for each joint at each time step are gathered and fed into the neural network. The network predicts terrain properties.

Section 2.3.3 discusses the existing literature on modeling terrain with machine learning. In this section, we propose a new approach in the same vein. Instead of constructing a sparse graph by hand, we use a much denser graph in which every node is connected. It is possible to achieve such a model using Transformer architecture [22]. The full attention mechanism powering the Transformers is loosely equivalent to a complete graph model. In this sense, each node in the graph becomes a token. The tokens are constructed by combining the mentioned proprioceptively available measurements. The full attention mechanism is not suited for long sequences; it has $O(n^2)$ time and memory complexity, where n is the number of elements in the sequence. To remedy this issue, a class of transformers known as efficient transformers are introduced [27]; they exploit the structure of the naive full-attention to achieve better algorithmic complexity. For our purposes, we use a popular one, Linformer [28]. Linformer reduces the $O(n^2)$ complexity to $O(n)$, allowing us to use it for long sequences with little loss of expression.

The proposed network is illustrated in Figure 3.4. Network architecture draws heavy inspiration from the vision transformer [25]. Proprioceptive inputs of the neural network are gathered for each joint at each timestep, and they are composed of the following:

- joint position, q
- joint velocity, $\dot{q}$
- joint torque, τ

Measurements of the joint quantities over a time window are concatenated with position embeddings and fed into the network. Since the torque values for the current time frame are not available, the torque values are shifted one step into the future to match the size of the tokens in all time frames. Using such a model, estimated terrain properties can be fed into the ID-CBF-QP described in Section 3.2.3.3:

- Friction cone constraints, Equation (3.23) and Equation (3.24), can be improved by updating μ.
- CBF ground clearance constraint, Equation (3.25), can be improved by updating $z(\varphi)$ in Equation (3.27).

3.3.2. Data Collection

To train the terrain estimation model described in Section 3.3.1, large amounts of data are required. To obtain such data, we can use Nvidia Omniverse Isaac [40]. It is a GPU-accelerated simulator that can achieve an order of faster simulations than the common CPU-based simulators.

Building on the work of [41], we can use Nvidia Isaac to simulate and gather data for legged robots. Figure 3.5 shows the simulation of 4096 parallel Unitree A1 quadruped robots, which is one of the popular choices for simulation and reinforcement-learning-based works. The simulation is able to reach 100,000 steps per second on an Nvidia RTX 3080 12GB GPU.

Collected data are stored as 3D arrays: the first dimension represents timesteps, the second is the environments, and the third is the measurements. We record the simulation observations to create a dataset. Due to the significant size of the records, care is required to keep batch sizes manageable; so they fit into memory. Table 3.1 displays values of the dimensions for each record. Measurements are given in Table 3.2.

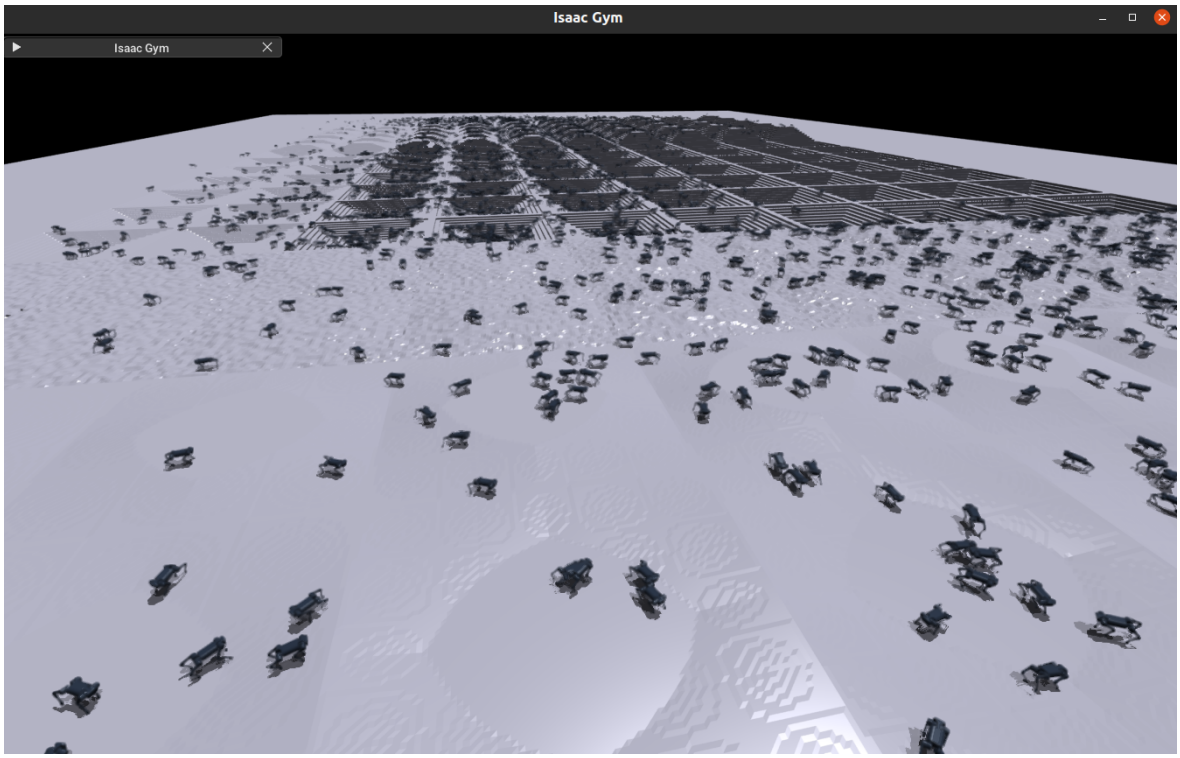

Figure 3.5. Nvidia Isaac simulates 4096 Unitree A1 quadrupeds over a terrain with varying roughness and friction. At each reset, robots are placed at different locations to experience different terrain conditions; the friction coefficient is replaced with a random value drawn from a uniform distribution.

Table 3.1. Raw measurements from Nvidia Isaac for each record.

Parameter	**Value**
Simulator timestep (dt)	0.005 s
Number of timesteps	120
Total time	0.6 s
Number of environments (robots)	4096
Number of measurements	252

Table 3.2. Definition of Nvidia Isaac measurements for a single robot for one timestep.

Name	**Size**
Base linear velocity	3
Base angular velocity	3
Projected gravity	3
Commanded base linear velocity	3
Joint positions	12
Joint velocities	12
Joint torques, applied in the last step	12
Height map grid around robot feet	187
Feet contact state	4
Friction coefficient	1
Joint torques, about to be applied	12
Total	252

4. IMPLEMENTATIONS

Theoretical methods discussed in Chapter 3 are realized in simulation environments. I have implemented a software package to work on mechanical systems via optimal control. The complete source code will be made available in [42]. Our implementation intends to be a general-purpose dynamical system control package for research. The main features are implemented while working on the simpler cart-pole system, discussed in Section 4.1. We have further specialized the package for working with legged robots; a quadruped robot is used during the implementation, details are given in Section 4.2.

Our software package uses PyBullet [43] for simulation and Pinocchio for rigid body computations. PyBullet is used only for simulation and observation; all calculations are carried out via Pinocchio. This would allow an easy transition to an actual robot, where highly efficient and embedded-ready Pinocchio is fully utilized. For optimization, we use the interface provided by qpsolvers [44]; through it, we have access to popular, powerful solvers such as CVXOPT [45], OSQP [46].

Different CBF formulations discussed in Chapter 3 result in different constraints. We have an implementation that allows us to define and add constraints easily to the optimization problem. After the controller is set up the performance can be evaluated by running simulations in different environments and with different initial conditions.

4.1. Safe Control of Cart-Pole Sytem

We have applied the safe control methods discussed in the previous chapter to the cart-pole system, described in Section 3.2.1. For the cart-pole system, we have two different implementations:

- A minimal implementation for cart-pole is developed [47]. Instead of simulation, ode45 implementation of Scipy [48] is used to integrate the equations of motion.

- Cart-pole system is also implemented in our main package [42] as well, where we use a rigid body dynamics simulator, PyBullet. Figure 4.1 shows the simulation environment in PyBullet.

The results from both implementations, direct integration of equations of motion and PyBullet simulation, agree as expected. While using the simulation environment a frictionless, flat plane is used thus the contact forces do not play any critical role. The simpler implementation with integrating equations of motion yields the same results.

4.1.1. Nominal System

The python-control [49] package is utilized to define the system dynamics given in Section 3.2.1.1. Then a LQR controller is constructed as the nominal controller.

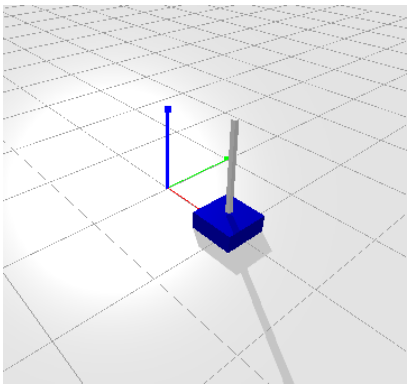

Figure 4.1. PyBullet simulation of cart-pole while moving in a frictionless plane.

Table 4.1 shows the system parameters; Table 4.2 shows the parameters used for the nominal controller.

Table 4.1. Cart-pole system parameters.

Parameter	**Value**
m_c	5 kg
m_p	2 kg
l	1.5 m

Table 4.2. Cart-pole LQR parameters.

Parameter	**Value**
$diag(Q)$	$[10.0, 1.0, 10.0, 100.0]$
R	1.0

4.2. Safe Control of Quadruped Robot

We have further developed our software package to include the safe control methods for legged robots, which are explained in Section 3.2.3. In the scope of this thesis, we limit the work to a quadruped robot with point feet. Again PyBullet provides the simulation environment, and Pinocchio provides rigid body dynamics algorithms for required model calculations such as mass matrix, jacobians, etc.

The selected quadruped robot is Unitree A1, shown in Figure 4.2. A1 is a popular choice for such purposes [50]. A1 has 12 actuated joints, 3 for each leg; it weighs 12 kilograms. The robot's specifications are available online, the details can be found in the openly available URDF.

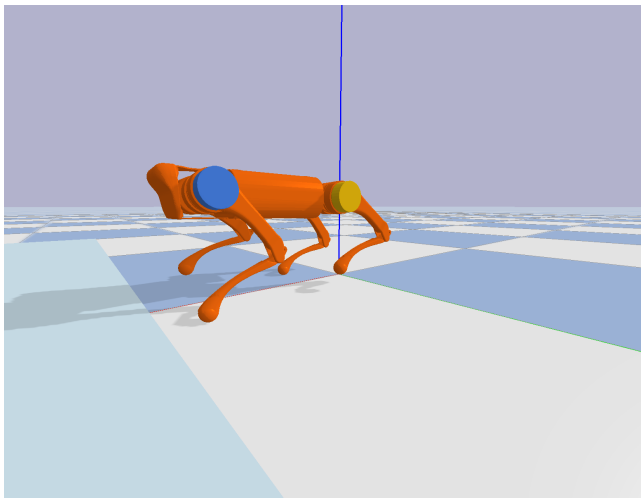

Figure 4.2. PyBullet simulation of Unitree A1 with the trotting gait.

4.2.1. Nominal System

A simple control scheme is applied for the nominal control of the robot. Figure 4.3 shows the control diagram used in this section. Though the used system is simpler than the two-loop version shown in Figure 3.1, the safety filter can be directly integrated into such, more complex systems.

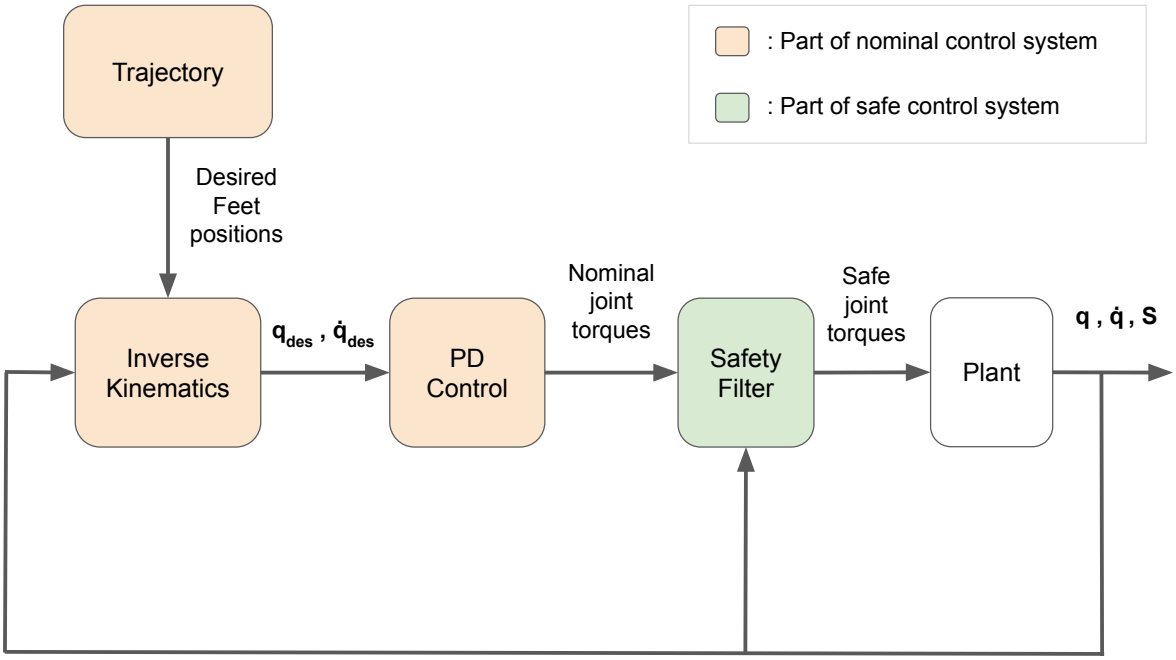

Figure 4.3. Control diagram of the quadruped robot. The plant is simulated in Pybullet.

As shown in Figure 4.3, an open loop trajectory is used. The trajectory for one foot over a step cycle is shown in Figure 4.4. Though the A1 robot can move in 3D, our implementation uses a planar motion for each foot. Right legs lead left legs by half a period, resulting in a trotting gait, which can be seen in Figure 4.2. It should be noted that the trotting gait is a form of dynamic walking; which cannot be achieved by simple methods such as ZMP. Based on the sampled trajectory, desired robot configuration and velocity are forwarded to a PD controller, where we get the nominal joint torques.

The nominal control signal, in this case, nominal joint torques are one of the two inputs to the safety filter. The safety filter computes the safe joint torques by minimally modifying the nominal torque values as detailed in Equation (3.19). The

robot state is provided to the safety filter as its second input by the feedback loop. S denotes a boolean array of size four which includes the contact status of each foot. By using S, we formulate proper constraints for each foot: if the foot is in stance mode, then it must obey the friction cone constraints; if it is in flight mode, it must follow the trajectory and also the ground clearance ECBF constraint given by Equation (3.36).

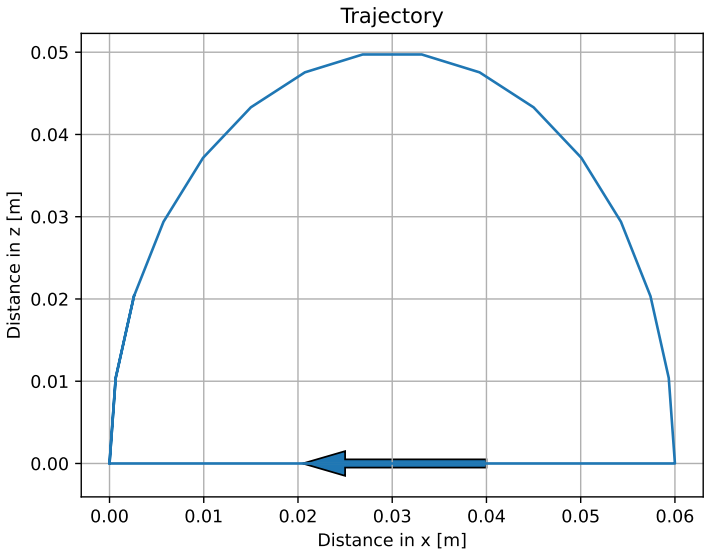

Figure 4.4. Openloop feet trajectory profile in the xz-plane. Front right and rear left move in unison, then front left and rear right also move in unison. There is a phase difference between the two diagonal pairs, resulting in a trotting gait.

4.3. Data-Based Terrain Estimation

The network architecture specified in Section 3.3.1 is implemented with PyTorch [51]. A dataset is created with the help of Nvidia Isaac by the means of GPU simulation as discussed in Section 3.3.2. We have split the dataset into three sets: train, validation, and test. Hyperparameters of the network are optimized with random search. The model with the best validation performance is selected and its generalization capabilities are verified with the test split.

5. RESULTS AND DISCUSSION

We developed a flexible implementation as detailed in Chapter 4. In this chapter, our implementation is used to evaluate the performance of the proposed methods in simulation environments for the two dynamical systems of interest: the cart-pole system and the quadruped robot.

5.1. Cart-Pole Simulation

In the methods, we have detailed two ways of formulating safety filters for the cart-pole system: the naive formulation given in Section 3.2.1 and the inverse dynamics formulation given in Section 3.2.2. We use the naive formulation, CBF-QP, in this section; though, both formulations yield the same results. The nominal controller used for the simulations below is described in Section 4.1.1.

5.1.1. CBF: Cart Velocity

CBF for cart velocity limits discussed in Section 3.2.1.3 is implemented, and the related parameters are shown in Table 5.1. Two simulations are performed, they only differ on whether the safety filter is enabled or disabled. Figure 5.1 shows the safety filter enabled case, we see that as the cart velocity reaches the velocity limit specified by the CBF constraint, v_{bound}, the control signal is modified temporarily. For other timesteps, the control signal remains unaltered. Figure 5.2 shows the safety filter disabled case for contrast. We observe that the CBF framework works as expected, one can note the safety filter interference causes a sudden change in the control signal.

Table 5.1. Cart-pole CBF parameters.

Parameter	**Value**
v_{bound}	0.45 m/s
α^{cbf}	10

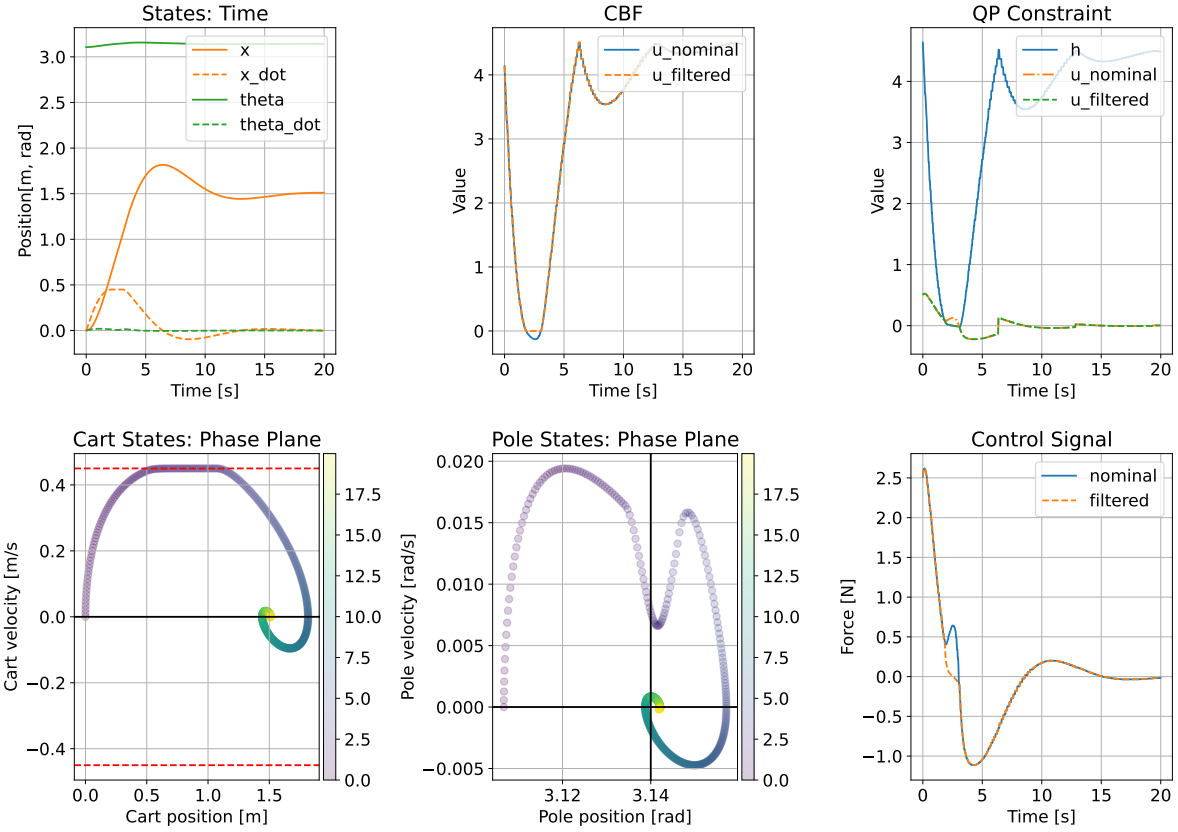

Figure 5.1. Cart-pole CBF-QP, safety filter enabled. CBF limits cart velocity.

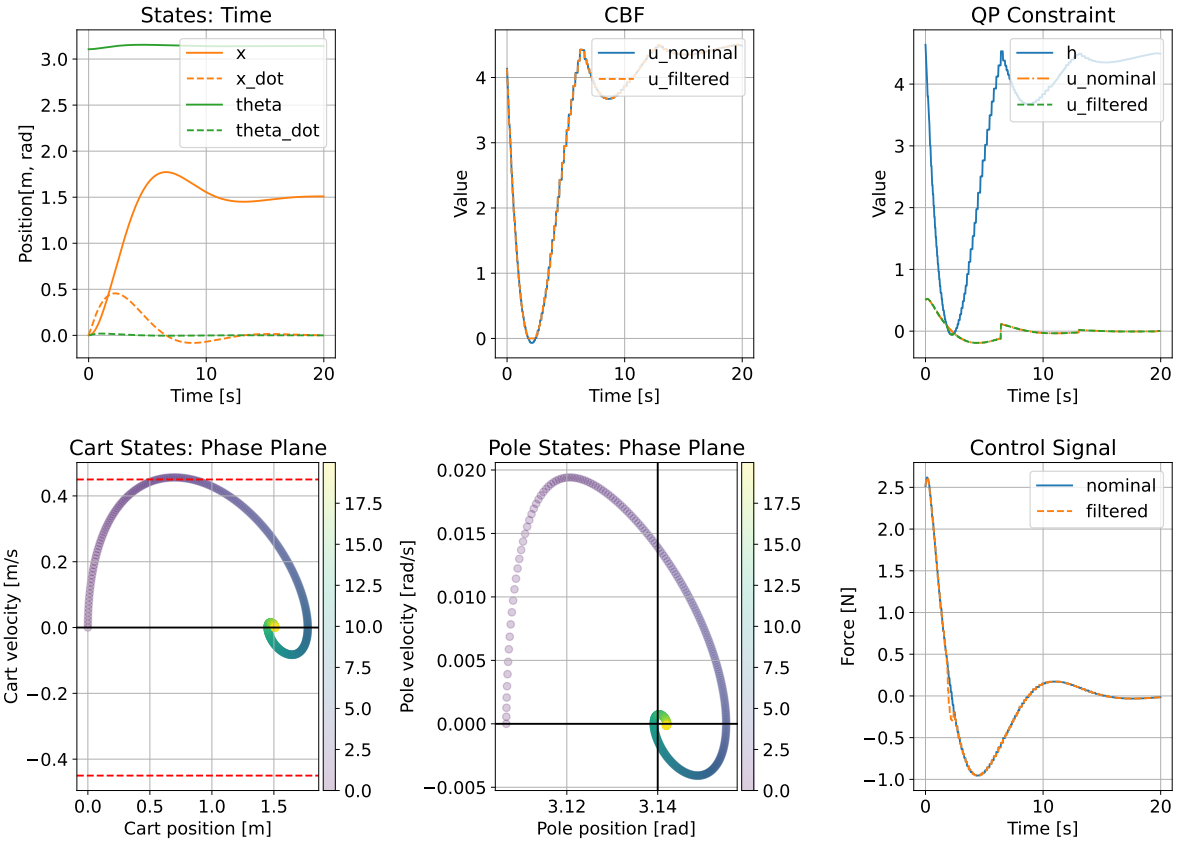

Figure 5.2. Cart-pole CBF-QP, safety filter disabled. CBF limits cart velocity.

5.1.2. ECBF: Cart Position

ECBF for cart position limits discussed in Section 3.2.1.4 is implemented; Table 5.2 shows the related parameters for the ECBF. Similar to the previous CBF case, two simulations are performed, one with the safety filter and the other with the safety filter off. Figure 5.3 show the case with the safety filter on, and Figure 5.4 show the filter off case. Similar to the CBF constraint from the previous section, we see that the ECBF constraint works as expected and stops the pole from going past the position limit, p_{bound}.

Compared to the previous CBF case, the effect is harder to observe as p_{bound} is carefully selected to minimally change the nominal closed-loop response; lowering p_{bound} and further constraining the system causes a larger modification to the control signal by the safety filter. Sudden, large control signals push the system out of the pole-up configuration, since the nominal LQR controller is operated to work with linearized dynamics around the pole-up configuration, the system cannot recover and the pole drops down. This case demonstrates one of the main concerns of the safety filters: the safety filter does not consider the capabilities of the nominal controller. The problem of conflicting with the nominal controller can be reduced in different ways:

- Careful tuning of parameters both for the safety filter and the nominal controller.
- Safety filter takes the nonlinear dynamics into account and can operate the system regardless of its configuration. To match the two controllers, the nominal controller can be extended or replaced with, for example, trajectory optimization.

Table 5.2. Cart-pole ECBF parameters.

Parameter	**Value**
p_{bound}	1.77 m
α_1^{ecbf}	10
α_2^{ecbf}	5

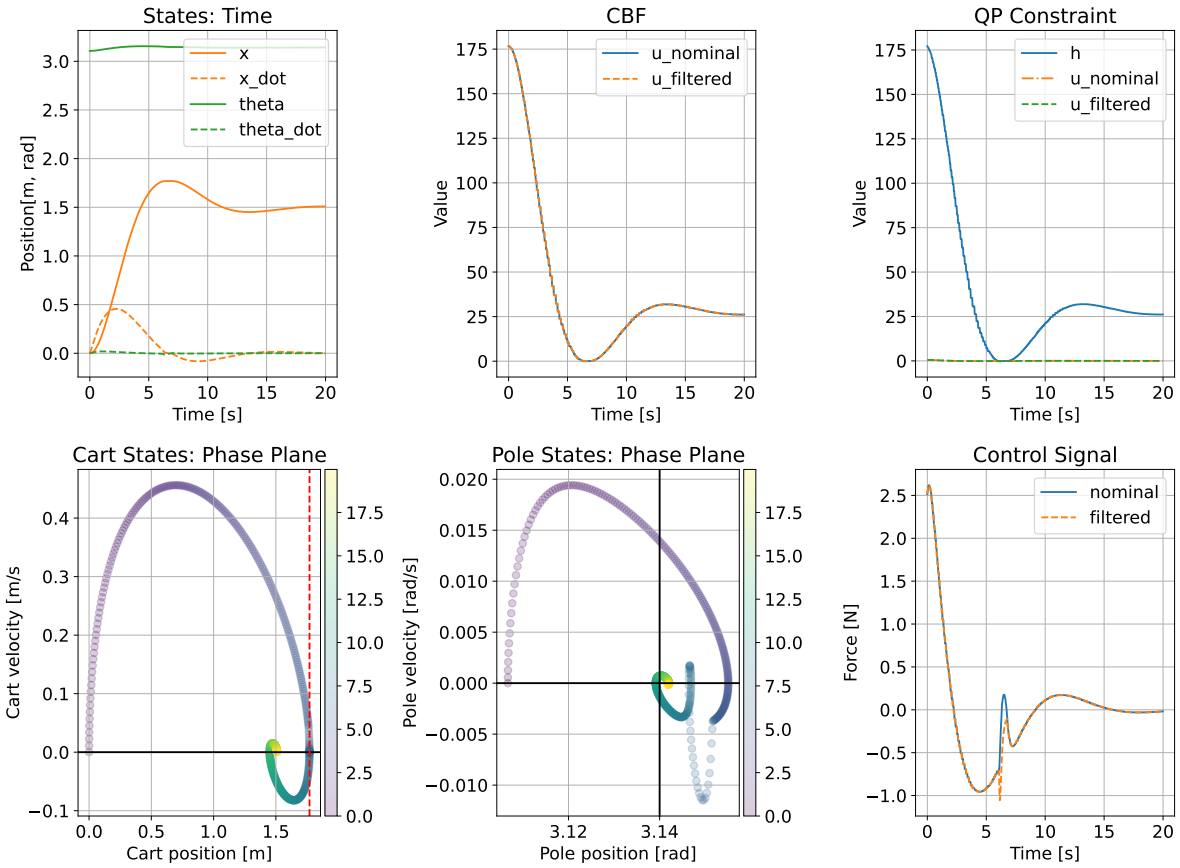

Figure 5.3. Cart-pole ECBF-QP, safety filter enabled. ECBF limits cart position.

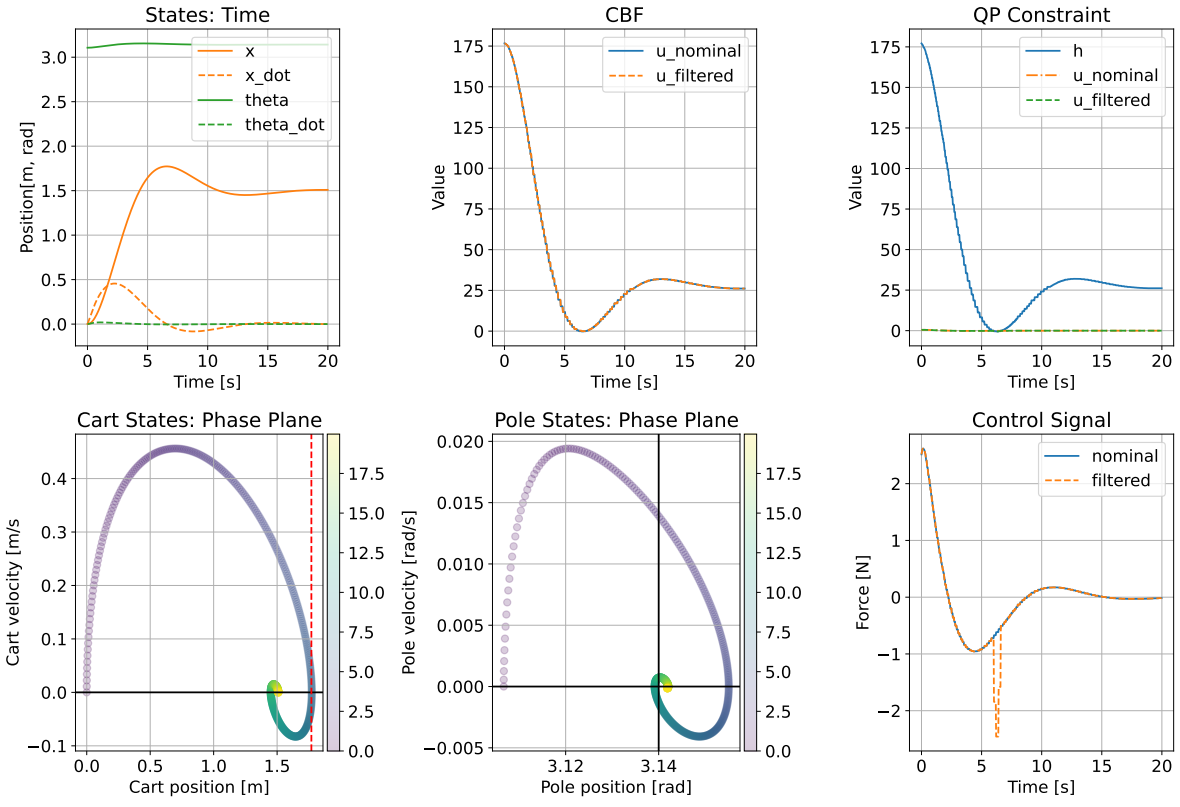

Figure 5.4. Cart-pole ECBF-QP, safety filter disabled. ECBF limits cart position.

5.2. Quadruped Robot Simulation

In this section, we inspect the effectiveness of our method for the control of legged systems detailed in Section 3.2.3. We use a quadruped robot in this work, the robot and nominal control approach are explained in Section 4.2.

5.2.1. Friction Cone

As detailed in Section 3.2.3.3, the optimal control problem for legged systems works with three optimization variables; one of them is $\boldsymbol{\lambda}$, the constraint forces or, in our specific case, ground reaction forces. By adding $\boldsymbol{\lambda}$ to the optimization problem, we can also apply constraints to it. Friction cone constraints are specified by Equation (3.22), Equation (3.23), and Equation (3.24). Assuming the friction coefficient is valid, the robot will never slip while walking as long as the friction cone constraint is satisfied. Thus, a major mode of failure will be limited. According to our implementation as a safety filter, the control will be adjusted only when the friction cone is about to be violated.

First, we would like to confirm the constraint forces from the optimal control solution match with the actual values. Figure 5.5 demonstrates the optimal controller can predict the constraint forces closely while walking in the simulation environment:

- For the vertical component, the mean absolute error is 3.0505 N.
- For the lateral components, the mean absolute error is 1.8463 N.

Then. to demonstrate the effectiveness of the friction constraint, a low friction value is set for the constraints: $\mu = 0.2$. The resulting control and ground reaction forces are shown for the front left foot in Figure 5.6, only a single arbitrarily selected foot is shown to keep the figure clean. In the figure, only the lateral forces are shown to keep it comprehensible; furthermore, the left-out vertical component has negligible change because it is hardly controllable. We observe spikes in the figure as the robot's foot impacts the ground, the simulator considers the impact only for a single timestep.

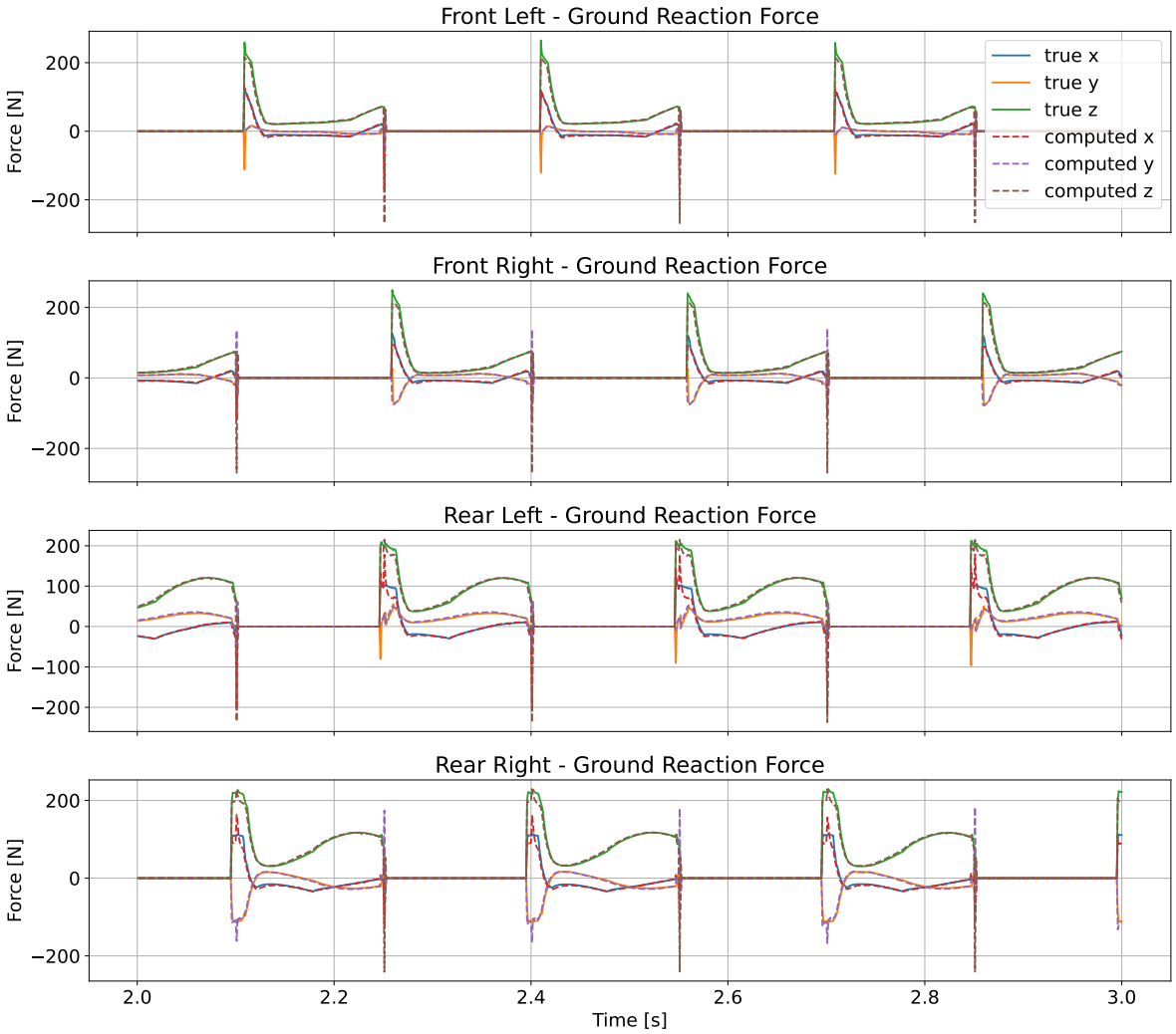

Figure 5.5. Comparison of optimized and measured ground reaction forces. The robot is walking with a trotting gait on flat ground.

We see that the ground reaction forces, $\boldsymbol{\lambda}$, are properly calculated and executed by the proposed control architecture. Results from the low friction environment given in Figure 5.6, show that the robot can walk with lateral ground reaction forces about half of the original. Since our nominal controller uses inverse kinematics, it does not take forces into account. By introducing the safety filter we compute inverse dynamics and apply friction cone constraints; thus, allowing us to eliminate foot sliding and enable safer locomotion.

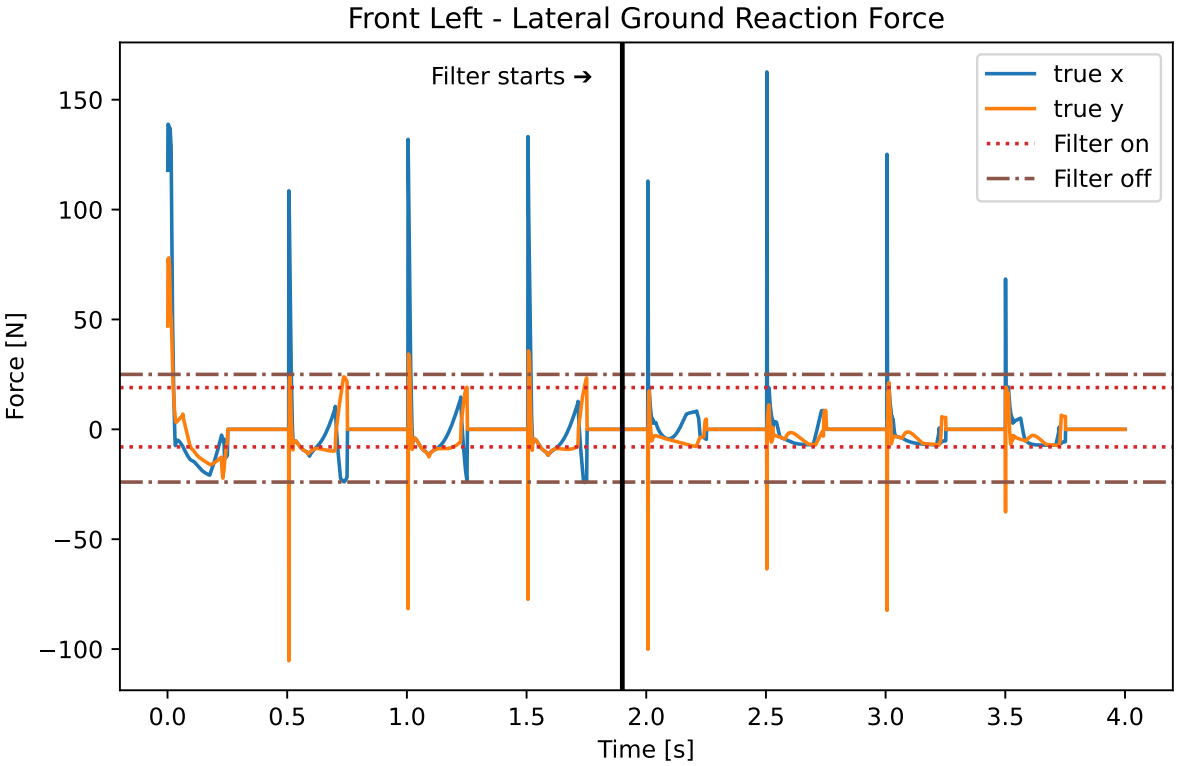

Figure 5.6. Decrease in lateral forces due to friction constraint after the filter is activated. The robot walks normally until the filter starts; then filter reduces lateral (tangential) forces.

5.2.2. Ground Clearance

We have discussed an ECBF to allow the robot to clear obstacles or navigate uneven terrain in Section 3.2.3.4. Arbitrary obstacle trajectories can be given as input to the safety filter; however, care must be taken to avoid infeasible solutions.

Figure 5.7 and Figure 5.8 show the simulation results in the presence of the ground clearance, i.e., ECBF, constraint for different obstacle profiles. In the figures, we observe that the ECBF framework works as expected and keeps the foot clear of the obstacle with minimal interference to the nominal control signal. We have used polynomials for demonstration, this is not necessary; our formulation only requires the obstacle profile to be twice differentiable, thus any function satisfying that property can be used.

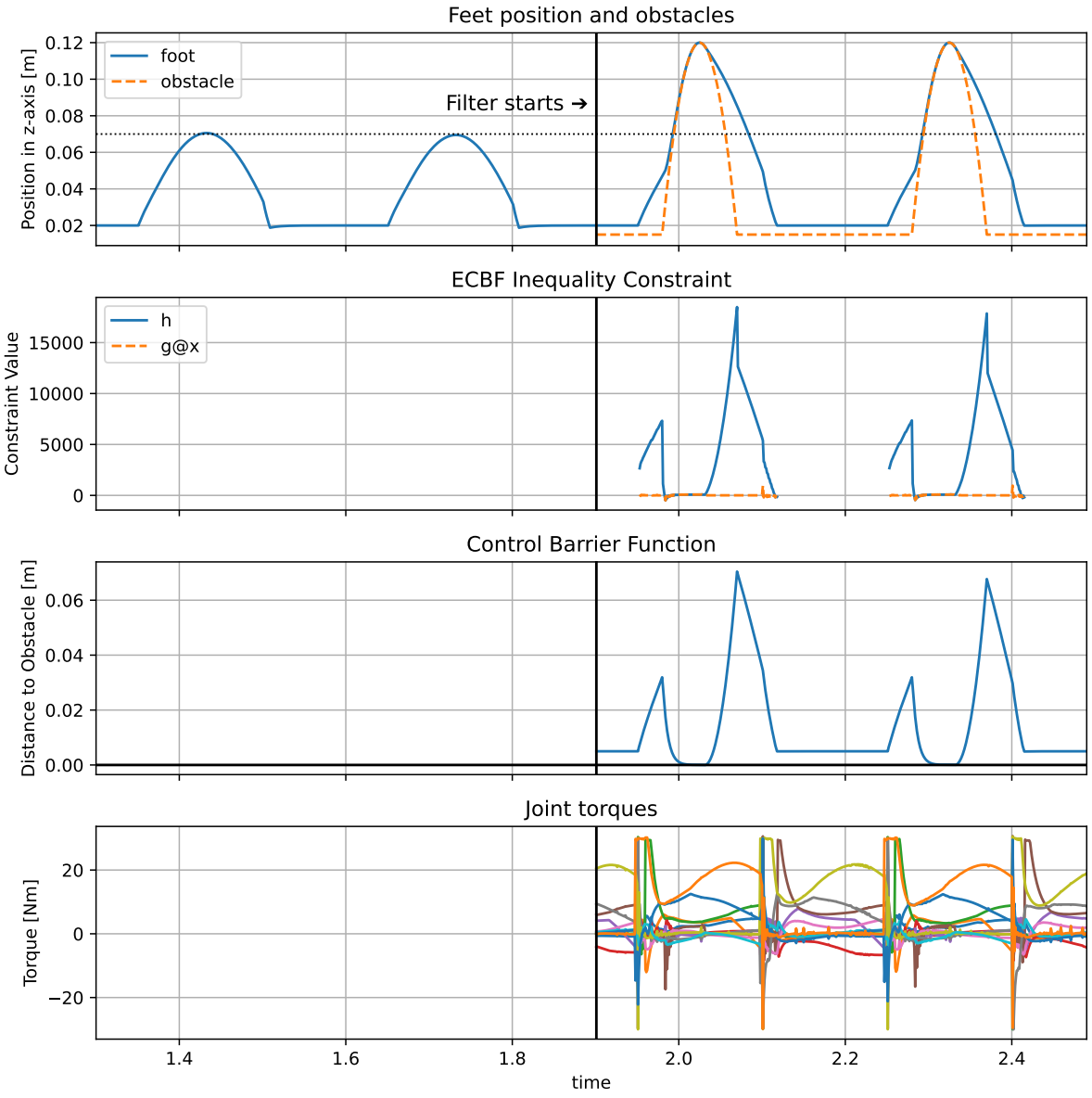

Figure 5.7. Simulation results with ground clearance constraint as a 2nd order polynomial. ECBF Constraint is active after the filter starts, and it is only defined during flight phases of the foot.

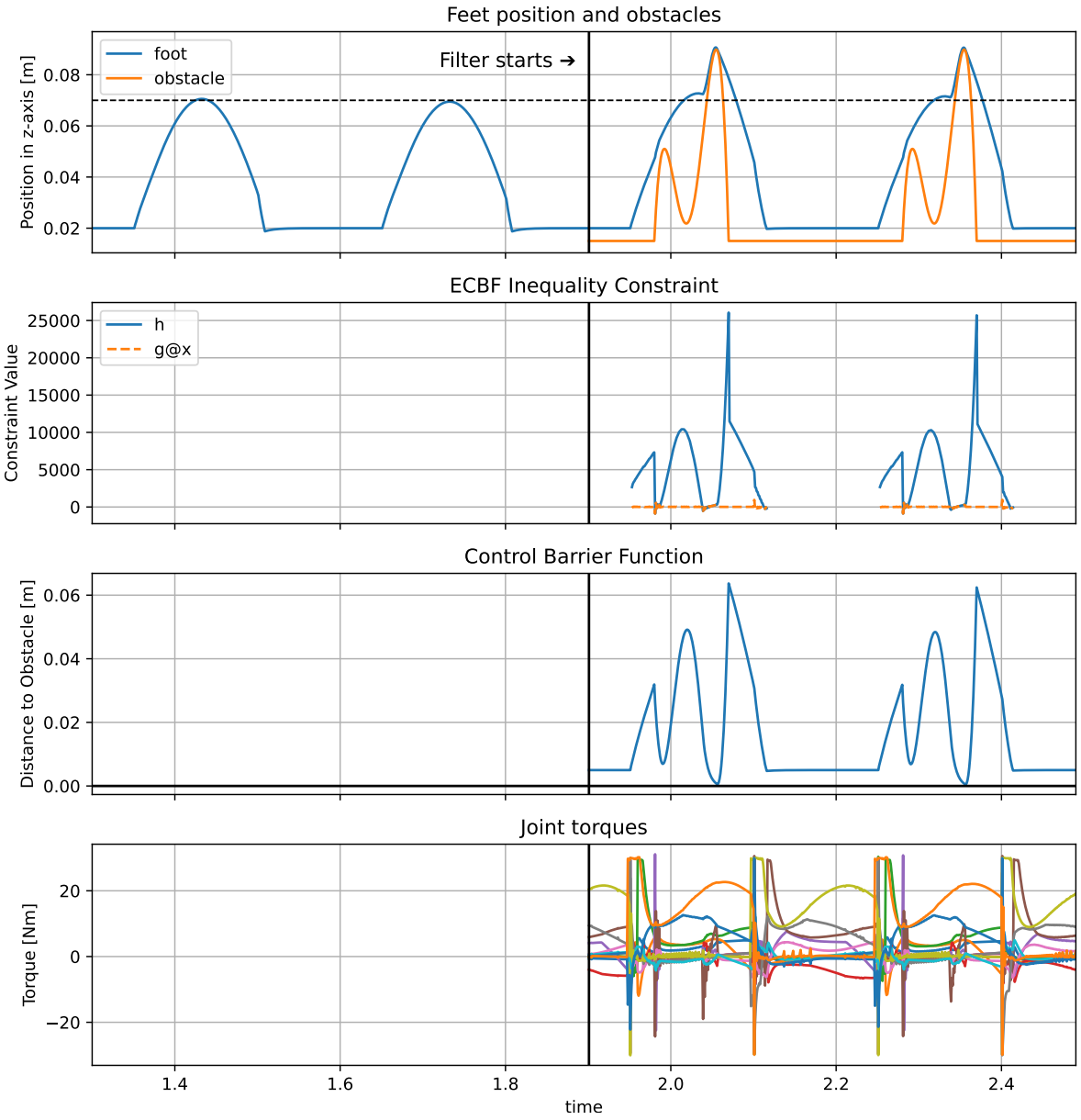

Figure 5.8. Simulation results with ground clearance constraint as a 4th order polynomial. ECBF Constraint is active after the filter starts, and it is only defined during flight phases of the foot.

5.3. Terrain Estimator

The model described in Section 3.3 is implemented; only proprioceptive measurements are used as features which are shown in Table 5.3. Raw measurements from the Isaac simulator are processed as shown in Table 5.4; dt is reduced by subsampling so that longer time windows are fed into the network. Our analysis shows network performance significantly increases with time windows (total time) of around 1 second.

A set of hyperparameters yielding good results are given in Table 5.5. Loss curves and evolution of model losses are shown in Figure 5.9 and Figure 5.10.

Using randomly sampled 3264 data points from the test split, the model can predict the friction coefficient with a mean absolute error of 0.0720; Figure 5.11 shows the regression performance of predictions and true values. Figure 5.12 shows that the error residuals are symmetric, implying a good fit.

Table 5.3. Selected features as input for the neural network model.

Name	**Dimension per timestep**
Joint configurations	12
Joint velocities	12
Joint torques	12

Table 5.4. Subsampled measurements from Nvidia Isaac for each record.

Parameter	**Value**
dt	0.01 s
Number of timesteps	240
Total time	2.4 s
Number of environments (robots)	4096
Number of measurements	252

Table 5.5. Network hyperparameters.

Parameter	**Value**
Learning rate	0.001
Batch size	64
Epochs	100
Time frames	40
Time between frames	0.03 s
Nodes in a time frame	12
Features per node	3
Positional embedding dimension	32
Depth attention layers	4
Number of attention heads	4
Hidden layer dimension	128
Dropout	0.2
Linformer-K dimension	64

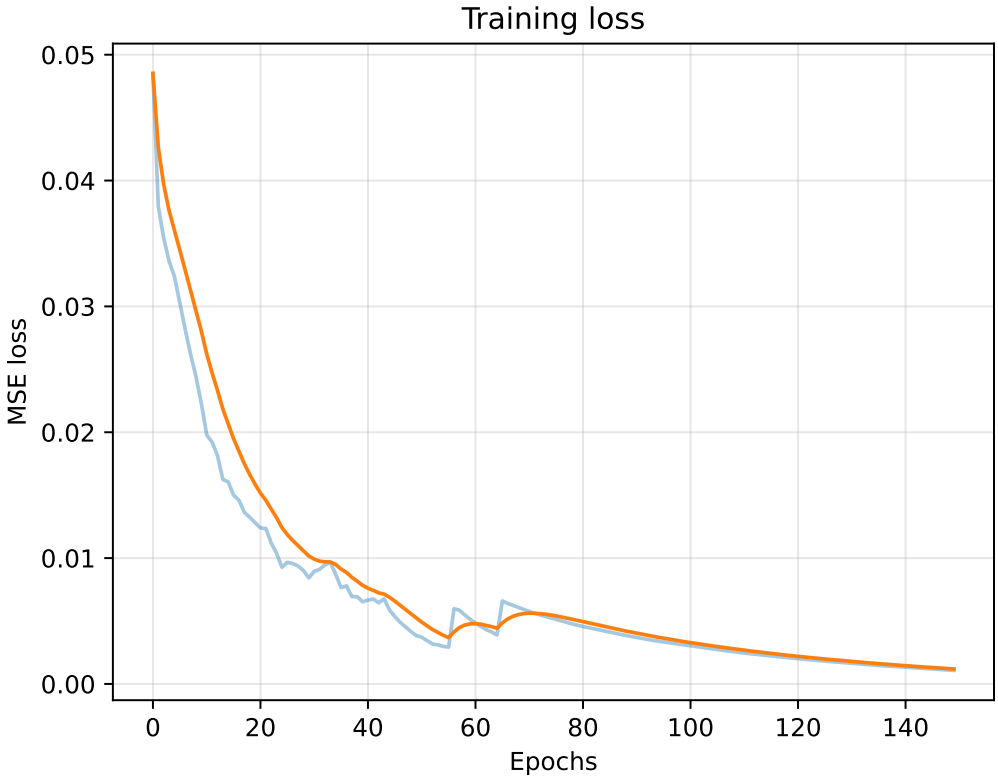

Figure 5.9. Training loss shown with both raw and smoothed curves.

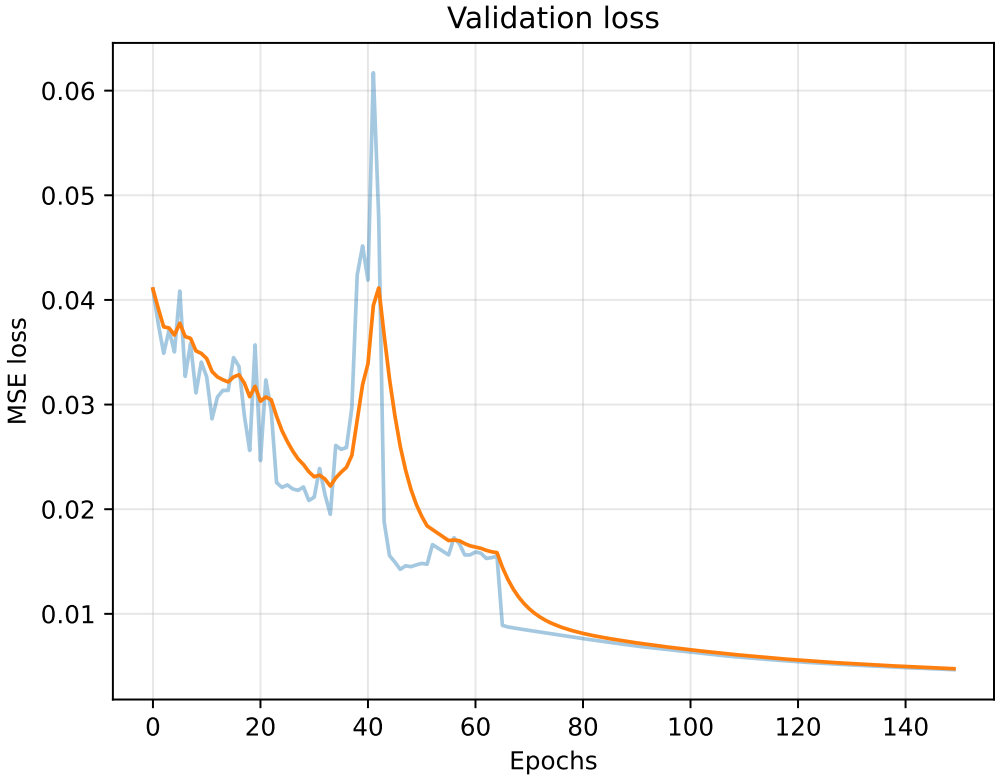

Figure 5.10. Validation loss shown with both raw and smoothed curves.

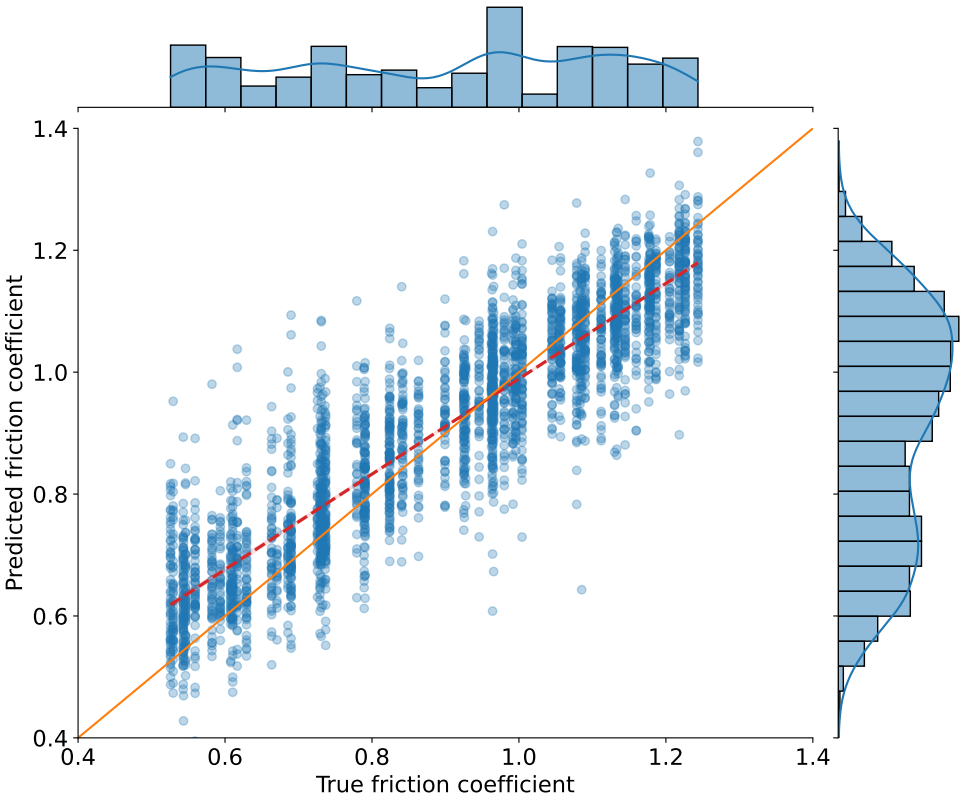

Figure 5.11. Regression performance of the model.

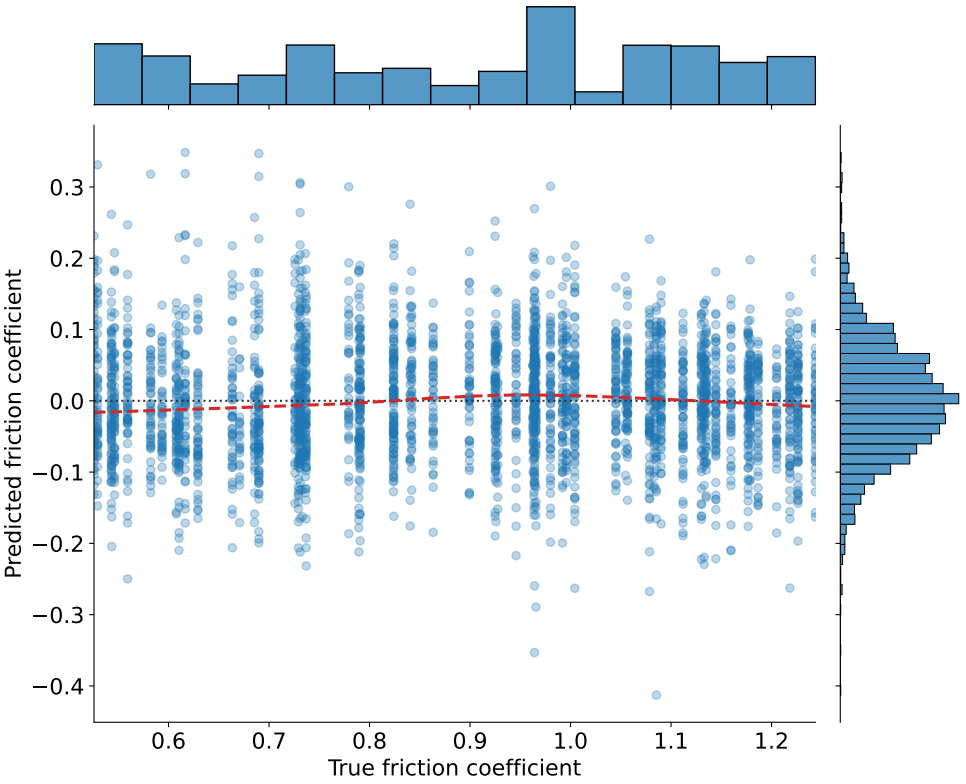

Figure 5.12. Error residuals with the dashed line showing locally weighted linear regression. Residuals are symmetrically distributed.

5.4. Discussion

We have shown the safety-critical performance of two underactuated dynamical systems in simulation environments. As the first system, the cart-pole demonstrated the potential and limitations of CBF; it tends to result in an aggressive control action and possible conflicts with the nominal controller. For the second system, we have used a quadruped robot to include contact-related effects. Nominal control of the robot with inverse-kinematics has shown its practical value but also its fragility. The inclusion of friction constraints had a strong positive effect on ensuring safety by stopping feet from slipping.

To include safety in a viability sense, we have turned to CBF. The CBF framework allows a flexible, intuitive synthesis of safety constraints; in the case of legged robots, working on the task space was straightforward. CBF framework allows room for creativity while defining safety, with the addition of the ECBF approach it becomes possible to use any combination of the robot's states to synthesize a CBF. There has already been some work on applying CBF to legged robots [18, 36, 38]; it is shown in physical systems that the modeling and measurement errors can be handled to a certain degree. However, larger deviations may cause failures. For such hard-to-model systems, few-shot learning-based methods are utilized and the system returns to functioning with high performance. Another concern with the CBF framework is implementing it as a safety filter; it can lead to suboptimal performance and conflicts with the nominal controller. For our implementations, we faced this issue while working on the cart-pole system for the position constraint as detailed in Section 5.1.2. Regarding a legged robot, it is common to employ Model Predictive Control (MPC) for the planning [32]; we have instead used a simpler scheme with open-loop trajectories. Having MPC for planning and a QP solver for execution opens the way to include constraints in one or both of them since both are optimal controllers. Recent work by [36], shows that having the constraint in the QP allows it to be enforced at a higher rate resulting in improved performance, as expected. Though, care must be given to aligning the two controllers; one can include complementing constraints in both to allow well-synchronized operation.

As we focus on including terrain effects, we formulated our CBF to take obstacles into account. One can combine it with the foot placement CBF found in the aforementioned works to achieve complete safe positioning of the feet in 3D space. It must be noted, there had been difficulties during the development, especially in tuning the barrier gains. Eventually, we achieved high-performance filtering of control signals. Framing safety in terms of model-based, intuitive functions allows a good way to quantify and guarantee legged robot performance, which will ease the deployment of such systems.

We have used blind sensing, i.e., proprioceptive estimation, in our machine-learning-based terrain estimator. Our work showed that the spatial-temporal relationships in legged robots should be evaluated in the order of seconds, not milliseconds. The transformer architecture has been applied to various domains already; we have shown it also performs well in understanding the contact dynamics of legged systems. Blind sensing is a promising, cost-effective approach to enable the wider use of legged robots. As detailed in Section 5.3, we see the friction coefficient can be predicted closely. By subtracting a small amount for possible deviations, one can get a conservative estimation of the friction coefficient which can then be used for all contact-related tasks.

6. CONCLUSION

This study presents a modern approach to safety-critical control of legged robots. We have constructed safety filters to allow minimally-intrusive, modular controllers that can be easily added to existing control architectures. For this purpose, the CBF framework is utilized to generate constraints. We have used the cart-pole system while working on the basics of safety filters which are implemented as optimal controllers. The inverse dynamics formulation of CBF which is suitable for legged robot applications has been developed first for the cart-pole system. Then, the developed approach is extended to a legged robot, a quadruped, to include contact-related effects. Proposed methods are implemented and evaluated in simulation.

In this work, we have developed a flexible optimal control software package that specializes in contact-aware robots. We have focused our efforts on legged systems which must be carefully controlled due to their underactuated nature. By defining constraints for the optimization problem, we have included several safety criteria; they cover some of the major sources of locomotion failures. In addition, we have developed a proprioceptive machine-learning model to estimate terrain properties. A blind terrain estimator can be easily added to most robots to increase their contact awareness which can then be used for high-performance control.

6.0.1. Contributions

In this work, following three contributions are made:

- An intuitive, high-level optimal-control-based safety filter framework is implemented in Python, which has direct support for Control Barrier Functions. The source code will be publically available [42, 47].
- A novel adaption of a state-of-the-art deep learning architecture, transformer, to model the friction of the terrain.

- A novel ECBF is formulated and implemented for ground clearance of the swing feet of legged robots.

6.0.2. Outlook

In this work, two seperate modules are developed for a quadruped robot: a safety filter and a terrain estimator. The effectiveness of each module has been shown independently; however, their combined performance is not explored in this work. Adding both the safety filter and terrain estimator to a powerful nominal control architecture such as ocs2 [52] would demonstrate their efficacy in greater confidence. Furthermore, embedded implementation on a real-time controlled robot is needed to verify all the findings from the simulation. Though they are addressed in Section 3.2.4 and Section 3.2.5, the work in this thesis has been limited to simulation alone. Further experimental work would be beneficial; intentional or unintentional simplifications from the simulation could be verified and if required fixed.

We followed the literature in our approach to the safety of legged robots. We have implemented an ECBF to enable ground clearance guarantees, combined with additional constraints it would serve well as a practical controller. However, there is no established method to evaluate the performance of legged robots; I believe a locomotion benchmark would accelerate progress in the field. Furthermore, improved definitions of locomotion safety, especially ones with a probabilistic sense would be valuable.

Our work could have benefited or had a different direction if the literature provided a more concrete understanding of deep learning. Most methods in the literature remain very practical; however, theoretical work is critical. There is a dire need for some technical framework that handles uncertainties and safety issues of machine learning models. Hopefully, the major interest in machine learning research will shed light on this area.

REFERENCES

1. Gong, Y. and J. Grizzle, "Zero Dynamics, Pendulum Models, and Angular Momentum in Feedback Control of Bipedal Locomotion", ArXiv:2105.08170, 2021.

2. Zaytsev, P., W. Wolfslag and A. Ruina, "The Boundaries of Walking Stability: Viability and Controllability of Simple Models", *IEEE Transactions on Robotics*, Vol. 34, No. 2, pp. 336–352, 2018.

3. Wieber, P.-B., *Holonomy and Nonholonomy in the Dynamics of Articulated Motion*, pp. 411–425, Springer Berlin Heidelberg, Berlin, 2006.

4. Carpentier, J. and N. Mansard, "Multicontact Locomotion of Legged Robots", *IEEE Transactions on Robotics*, Vol. 34, No. 6, pp. 1441–1460, 2018.

5. Focchi, M., R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell and C. Semini, "Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality", *Springer Tracts in Advanced Robotics*, pp. 165–209, Springer International Publishing, 2019.

6. Siekmann, J., K. Green, J. Warila, A. Fern and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning", ArXiv:2105.08328, 2021.

7. Miki, T., J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun and M. Hutter, "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild", *Science Robotics*, Vol. 7, No. 62, p. eabk2822, 2022.

8. Ames, A. D., S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath and P. Tabuada, "Control Barrier Functions: Theory and Applications", *18th European Control Conference (ECC)*, pp. 3420–3431, Naples, Italy, 2019.

9. Ibarz, J., J. Tan, C. Finn, M. Kalakrishnan, P. Pastor and S. Levine, "How to Train

Your Robot with Deep Reinforcement Learning: Lessons We Have Learned", *The International Journal of Robotics Research*, Vol. 40, No. 4-5, pp. 698–721, 2021.

10. Tedrake, R., "Underactuated Robotics", `https://underactuated.csail.mit.edu`, accessed on December 1, 2022.

11. Featherstone, R. and D. E. Orin, "Dynamics", *Handbook of Robotics*, pp. 35–65, Springer, Berlin, Heidelberg, 2008.

12. Budhiraja, R., J. Carpentier, C. Mastalli and N. Mansard, "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics", *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–9, 2018.

13. Holmes, P., R. J. Full, D. Koditschek and J. Guckenheimer, "The Dynamics of Legged Locomotion: Models, Analyses, and Challenges", *SIAM Review*, Vol. 48, No. 2, pp. 207–304, 2006.

14. Wieber, P.-B., "Viability and Predictive Control for Safe Locomotion", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1103–1108, 2008.

15. Vukobratović, M. and B. Borovac, "Zero-Moment Point — Thirty Five Years of Its Life", *International Journal of Humanoid Robotics*, Vol. 01, No. 01, pp. 157–173, 2004.

16. Hirukawa, H., S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara and M. Morisawa, "A Universal Stability Criterion of the Foot Contact of Legged Robots - Adios ZMP", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1976–1983, 2006.

17. Koolen, T., T. de Boer, J. Rebula, A. Goswami and J. Pratt, "Capturability-Based Analysis and Control of Legged Locomotion, Part 1: Theory and Application to Three Simple Gait Models", *The International Journal of Robotics Research*, Vol. 31, No. 9, pp. 1094–1113, 2012.

18. Rodriguez, I. D. J., N. Csomay-Shanklin, Y. Yue and A. D. Ames, "Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies", Vol. 168, pp. 1060–1072, 2022.

19. Kipf, T. N. and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", ArXiv:1609.02907, 2017.

20. Yan, S., Y. Xiong and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, No. 1, 2018.

21. Liu, Z., H. Zhang, Z. Chen, Z. Wang and W. Ouyang, "Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition", ArXiv:2003.14111, 2020.

22. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

23. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", ArXiv:1810.04805, 2019.

24. Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", ArXiv:1907.11692, 2019.

25. Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", *International Conference on Learning Representations (ICLR)*, 2021.

26. Chen, H., Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu and W. Gao, "Pre-Trained Image Processing Transformer", ArXiv:2012.00364, 2021.

27. Tay, Y., M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder and D. Metzler, "Long Range Arena: A Benchmark for Efficient Transformers", ArXiv:2011.04006, 2020.

28. Wang, S., B. Z. Li, M. Khabsa, H. Fang and H. Ma, "Linformer: Self-Attention with Linear Complexity", ArXiv:2006.04768, 2020.

29. Joshi, C., "Transformers are Graph Neural Networks", *The Gradient*, Vol. 12, 2020.

30. Lee, J., J. Hwangbo, L. Wellhausen, V. Koltun and M. Hutter, "Learning quadrupedal locomotion over challenging terrain", *Science Robotics*, Vol. 5, No. 47, p. eabc5986, 2020.

31. Roy, N., I. Posner, T. Barfoot, P. Beaudoin, Y. Bengio, J. Bohg, O. Brock, I. Depatie, D. Fox, D. Koditschek, T. Lozano-Perez, V. Mansinghka, C. Pal, B. Richards, D. Sadigh, S. Schaal, G. Sukhatme, D. Therien, M. Toussaint and M. Van de Panne, "From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence", ArXiv:2110.15245, 2021.

32. Bledt, G., M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245–2252, 2018.

33. Nguyen, Q. and K. Sreenath, "Exponential Control Barrier Functions for Enforcing High Relative-Degree Safety-Critical Constraints", *American Control Conference (ACC)*, pp. 322–328, 2016.

34. Reher, J., C. Kann and A. D. Ames, "An Inverse Dynamics Approach to Control Lyapunov Functions", *American Control Conference (ACC)*, pp. 2444–2451, 2020.

35. Featherstone, R., "An empirical study of the joint space inertia matrix", *In Inter-*

national Journal of Robotics Research, Vol. 23, pp. 859–871, 2004.

36. Grandia, R., A. J. Taylor, A. D. Ames and M. Hutter, "Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8352–8358, 2021.

37. Carpentier, J., G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse and N. Mansard, "The Pinocchio C++ library – A Fast and Flexible Implementation of Rigid Body Dynamics Algorithms and Their Analytical Derivatives", *IEEE International Symposium on System Integrations (SII)*, pp. 614–619, Paris, France, 2019.

38. Nguyen, Q., A. Hereid, J. W. Grizzle, A. D. Ames and K. Sreenath, "3D Dynamic Walking on Stepping Stones with Control Barrier Functions", *IEEE 55th Conference on Decision and Control (CDC)*, pp. 827–834, 2016.

39. Prete, A. D., N. Mansard, O. E. Ramos, O. Stasse and F. Nori, "Implementing Torque Control with High-Ratio Gear Boxes and without Joint-Torque Sensors", Vol. 13, No. 1, p. 1550044, 2016.

40. Makoviychuk, V., L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa and G. State, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning", ArXiv:2108.10470, 2021.

41. Rudin, N., D. Hoeller, P. Reist and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning", ArXiv:2109.11978, 2021.

42. Tosun, B., "taslel: Terrain-Aware Safe Legged Locomotion", 2022, `https://github.com/Berk-Tosun/taslel`, accessed on December 10, 2022.

43. Coumans, E. and Y. Bai, "PyBullet, A Python Module for Physics Simulation

for Games, Robotics and Machine Learning", http://pybullet.org, accessed on August 7, 2022.

44. Caron, S., "qpsolvers", https://github.com/stephane-caron/qpsolvers, accessed on October 14, 2022.

45. Andersen, M. S., J. Dahl and L. Vandenberghe, "CVXOPT: A Python Package for Convex Optimization", 2013, https://github.com/cvxopt/cvxopt, accessed on March 8, 2022.

46. Stellato, B., G. Banjac, P. Goulart, A. Bemporad and S. Boyd, "OSQP: An Operator Splitting Solver for Quadratic Programs", *Mathematical Programming Computation*, Vol. 12, No. 4, pp. 637–672, 2020.

47. Tosun, B., "cbf-cartpole", 2021, https://github.com/Berk-Tosun/cbf-cartpole, accessed on October 31, 2022.

48. Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, Vol. 17, pp. 261–272, 2020.

49. Murray, R., B. Grainera, S. Fuller and C. Rowley, "python-control", https://github.com/python-control/python-control, accessed on October 4, 2022.

50. Coumans, E., J. Peng and Y. Yang, "motion-imitation", 2018, https://github.com/erwincoumans/motion_imitation, accessed on August 26, 2021.

51. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", ArXiv:1912.01703, 2019.

52. Farshidian, F., M. Neunert, A. W. Winkler, G. Rey and J. Buchli, "An Efficient Optimal Planning and Control Framework for Quadrupedal Locomotion", *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.