POLITECNICO DI TORINO

MASTER's Degree in
MECHATRONICS ENGINEERING

Master's Degree Thesis

Dynamically Feasible Vision-Based Foothold Adaptations for Legged Locomotion

Supervisors

Prof. Giovanni Gerardo MUSCOLO

Dr. Octavio VILLARREAL

Prof. Giovanni BRACCO

Dr. Claudio SEMINI

Candidate

Luca CLEMENTE

ISTITUTO ITALIANO
DI TECNOLOGIA

04/2021

In collaboration with

DYNAMIC LEGGED SYSTEMS LAB

part of

ISTITUTO ITALIANO DI TECNOLOGIA

Summary

Dynamic locomotion on rough terrain is still a challenge for legged robots. To traverse complex scenarios reliably, information from proprioceptive and exteroceptive sensors may be required. Legged robots can be considered systems with both continuous and discrete dynamics, namely, a hybrid system. Their dynamics can be modelled continuously between two consecutive steps, but when a leg leaves or touches the ground, the dynamic description changes almost instantly, making them difficult to control. Moreover, the choice of the of foot locations has a high influence on the dynamic stability.

When dealing with rough terrain and complex scenarios, the problem of choosing the right foot location becomes even more difficult. Although there exist already legged robots capable of performing locomotion on rough terrains, most of them rely on reactive control or heuristic strategies. In some complex situations a high precision is required to choose the correct foot location and avoid falling. The absence of such vision-based strategies may be addressed to the heavy computational effort required to process visual data.

There exist actual strategies that rely on visual data to select foothold locations, but most of them consider only geometric aspects such as the robot kinematics (work space) and the terrain morphology. Dynamics is usually not considered in the evaluations due to the high-dimensionality of the problem caused by the large number of bodies and degree of freedom of legged robots. The absence of dynamics in such evaluations can lead to failures, for example choosing a foothold in which the robot cannot generate enough ground reaction forces to move as expected.

We propose a foothold adaptation criterion that evaluates the transition feasibility between contact switches. It includes dynamics to keep into account the future dynamical behaviour of the robot. We integrate this criterion with a vision-based foothold adaptation (VFA) strategy that considers the robot kinematics and the terrain morphology. The devised criterion is able to evaluate the terrain using information coming from the robot vision sensors, as well as its states (position, velocity and acceleration) to determine the existence of a dynamic trajectory that connects the current states of the robot with a set of desired states. Furthermore, the method generates a dynamically feasible trajectory for the robot to follow

as a natural outcome of the evaluation. We show that the criterion is suitable for both static and dynamic gaits, even when the dynamic equations of motion describe an underactuated system, such as trot, which implies that orientation and position cannot be fully controlled at the same time. We perform simulation studies on the quadruped robot HyQ to evaluate the dynamic transition feasibility between consecutive footholds while performing locomotion and we use the computed trajectory on a motion planner to demonstrate the consistency of the motion.

Acknowledgements

Working side by side with the people of the Dynamic Legged Systems Lab at Istituto Italiano di Tecnologia has been one of the most rich and beautiful experience of my life. Although there have been harsh and stressful times, the knowledge acquired working in this welcoming and hi-tech environment is priceless. The combination of the avant-guard technology and the niceness of the talented people creates the perfect environment to overcome any challenge, knowing that you can always count on the expertise of others.

The first mention goes to Octavio Villarreal who has been the pillar of my experience, guiding me through the many difficulties that entering the scientific world entails. His knowledge and skills have been a very valuable help to reduce the harshness of this journey.

Prof. Giovanni Gerardo Muscolo also had an important important role during my experience. Although current limitations led to some communication difficulties, he was always ready to provide me advices and opinions to point me in the right direction.

Another important mention goes to Dr. Claudio Semini, head of the Dynamic Legged Systems Lab. His leadership has been inspiring and a source of a sense of safety, supporting all the people of the DLS lab with his incredible experience.

But the most important mention goes to my family. *Grazie per aver sempre creduto in me, supportandomi giorno dopo giorno nelle mie scelte. Grazie per avermi permesso di crescere e di inseguire i miei sogni, anche se questo singnifica compiere dei sacrifici. Devo a voi ciò che sono oggi, è merito del vostro instancabile impegno se sono la persona che sono oggi.*

Table of Contents

List of Tables

List of Figures

Glossary

CNN Convolutional Neural Network

CoM Center of Mass

contact switch instant of time in which a leg change phase and a contact is made or broken with ground.

DoF Degree of Freedom

duty factor it is the fraction of the stride period in which a limb is in contact with the ground.

dynamic transition it is meant as moving through one or more transients where dynamics apply.

foothold foot-ground contact point.

gait pattern of movements of the limbs during locomotion.

GRF Ground Reaction Force

heightmap it is a discrete terrain map that stores the heights of the terrain portion considered.

leg cycle it is defined as the smallest repeating sequence of phases that the leg perform.

lift-off the instant in which a leg break contact with the ground.

locomotion the ability of the robot to move itself in space.

nominal foothold it is the prediction of the landing position of the foot considered.

RBDM Rigid Body Dynamic Model

RF Reference Frame

RPY Roll, Pitch, Yaw

step frequency it is the frequency at which a step is taken.

touchdown the instant in which a leg makes contact with the ground.

Chapter 1

Introduction

To build moving machines is something that humans have desired for a long time. The idea of creating something that can be resemble to be alive, something that can decide for itself has always fascinated human kind. As the years go by we advance towards that goal, building more and more refined machines that are able to perform increasingly complex tasks.

One of the requirements to have a machine that mimics life is motion, as life cannot exists without it. Motion intended as moving through space is something with what we are used to as we constantly perform locomotion tasks moving from one place to another. Then the desire to build moving machines is a direct consequence of our concept of life. Nowadays there is a large variety of autonomous moving machines, spanning from flying drones to crawling snake-like robots. All categories of robots have different purposes, each of those with their pros and cons. We focus on the category of legged robots.

In this thesis we try to enhance the locomotion capabilities of robots over rough terrain using visual feedback. In this chapter we describe how this work contributes to solve this challenge and we provide a general outline of the thesis.

1.1 Motivation

Legged robots have proven to be a reliable choice when dealing with uneven terrain. Compared to their wheeled counterparts, they are able to deal with different environments with less effort and more effectively, although their complexity [1] is higher than wheeled systems. A legged robot can be considered holonomic in a two-dimensional space while a classic wheeled robot cannot, making them more suitable to traverse highly uneven terrain. The major difference between wheeled and legged robots is how they interact with the environment, i.e., how they exert forces to move through space. While wheeled robots push themselves forward by applying a torque on the wheels, legged robots can use their legs to exert forces in a desired location within their workspace. This makes legged robots more suitable to interact with environments where the terrain present some extreme features such as holes or small sharp obstacles.

The ability of choosing the right way of interacting with the environment is important for every system, but it is crucial for legged robots. Choosing where to place the feet is one of the main problems in legged locomotion, as the success of the task (e.g. walking) depends on the ability of exerting forces on the terrain. When the task is executed in a known environment, all the variables, such as obstacles or terrain features, can be accounted for in advance. This allows to compute a trajectory that the robot will follow during the execution of the task [2, 3, 4]. However, this approach may be reduced to previously known environments. Using vision systems [5] might impact positively the motion success rate, expanding the capabilities of the robots.

The inclusion of exteroceptive sensors (such as LIDAR or ultrasonic sensors) gives robots the ability to "read" the environment, capturing useful information to increase the reliability of locomotion. The terrain is analysed continuously to evaluate its morphology, i.e., if there are any holes, or if there are obstacles that might interfere with the locomotion. The detected terrain features can be processed to evaluate where to place the robot's feet [6].

An evaluation process able to discard footholds according to geometric-related aspect helps to reduce the chance of failure when executing a motion. However, considering only geometrical aspects could not be enough to avoid falling in complex scenarios. As an example, we can think about a foothold that is reachable from the robot but does not allow to exert the desired ground reaction force. It can

happen when the required force to achieve a motion exceeds the maximum force that the robot can generate. This limit could be reached at any moment during the motion, even when the leg is already on the ground. This highlight how evaluating dynamics considering the future behaviour of the robot can impact positively the success rate of the motion.

In this thesis we address the problem of foot placement with the aid of vision systems to detect terrain features. The potential foot locations are evaluated with a dynamic model to check if they can yield a consistent trajectory. This evaluation can lead to a better foot location in terms of dynamical stability.

1.2 Contributions

The main contribution of this thesis can be stated as follows:

1. A foothold selection criterion that considers the dynamic transition feasibility of a motion when taking a step which extends the method presented in [6] to be able to deal with dynamic gaits such as trot.

Additionally, a secondary contribution of this work is:

1. A generalized way to compute a Bézier sub-curve of a generic n order in a generic $\mathbb{R}^m$ space

1.3 Outline

The rest of the thesis is organized as follows: in Chapter 2 an overview of fundamental concepts and related work is provided; in Chapter 3 a background of the laboratory where this research was carried it, namely the Dynamic Legged Systems (DLS), is given with a brief description of the platform used for simulation studies, the quadruped robot HyQ; in Chapter 4 the dynamic feasibility problem is formulated and we detail the challenges that it entails; in Chapters 5 and 6 we present our method for tackling the problem addressed in Chapter 4; in Chapter 7 we present the results that we have obtained testing our algorithm both in simulation and in foothold selection; Chapter 8 marks the end of this thesis with some final considerations and future work possibilities.

Chapter 2

Related work

Legged robots is a concept that is nowadays widely spread in robotics. However it is a relatively old branch of robotics, placing itself in history around the 1960's [7]. The advantage that these machines could have in complex scenarios compared to their wheeled counterparts, made them really appetible in applications such as space exploration or rescue tasks. Since the 60's, these machines have traversed a period of fast evolution to become the agile machines that we know today, dancing to the notes of *Do you love me* (The Contours) [8].

The first step taken was to make them walking without falling. In [9] the pioneering work presented by *McGhee and Frank* on creeping gaits defined the fundamentals to create a stability criterion for legged robots, known as the *support polygon*. It states that, to achieve statical stability, the projection of the Center of mass (CoM) of a walking robot must lie within the convex hull of its contact points. This concept was extended by *Vukobratović and Juricic* in [10] where new concepts as the Zero-Moment Point (ZMP) and the Center of Pressure (CoP) were introduced. The ZMP is the point on the ground where the moment produced by the inertial and gravitational forces is equal to zero, while the CoP is defined as the equivalent application point of all the ground reaction forces. The usefulness of these tools is incontrovertible, being still used to control legged robots nowadays. They are simple and powerful tools to determine balance conditions on the ground. The ZMP has even been extended to non-flat terrains [11] by considering friction constraints.

However, if these tools are able to provide static equilibrium to a walking robot, they do not allow to have flight phases during the locomotion because of the

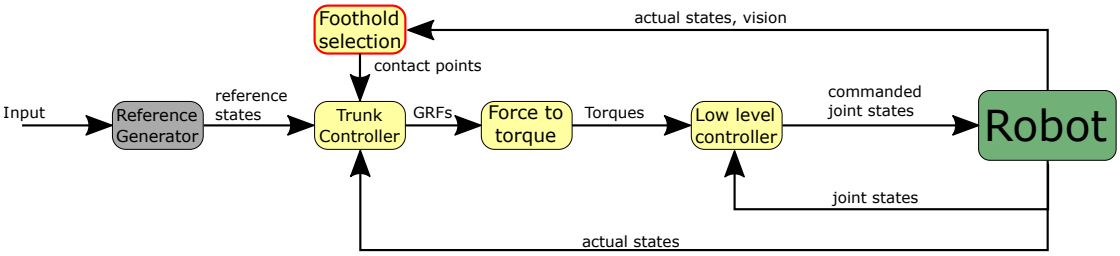


Figure 2.1: General logic scheme of a vision-aided legged robot control.

impossibility to apply forces without a proper contact with the ground. This limits the performances of the robot when a more dynamical motion is required. A wide variety of locomotion strategies have been developed to allows legged robots to be reliable independently of the static balance criteria. These strategies are the core for those robots capable of performing dynamic locomotion tasks. We refer to dynamic locomotion as those type of locomotions strategies where the body is not in static equilibrium while it is moving, but where the robot takes continuous steps to avoid falling. In the next section we provide an overview about the evolution of the dynamic locomotion in legged robots.

2.1 The legged locomotion problem

In this section we provide a summary of locomotion strategies used to achieve dynamic locomotion with legged robots. Achieving such type of locomotion is a very complex task because it involves several variables to take into account, such as balancing, attitude control, leg synchronization, generation of a contact sequence, foot trajectory generation and disturbance rejection. All of these components have to be computed within a short time [12]. To lighten the heavy computation burden, each component is dealt with separately using combination of approaches that consider different aspects of the process. The innovation brought by *Raibert* [13] in the 80's used a combination of classic control theory and heuristic approaches. The heuristic approach is a methodology that relies mostly on observation and experience rather than theory or mathematical derivations. Although they provide good solutions to complex problems, they are difficult to generalize due to their low affinity with theory.

Researchers have also developed their works drawing inspiration from nature. An example of nature-inspired result is the Central Pattern Generator (CPG).

CPGs are those neural circuits in animals capable of producing rhythmic output signals while receiving only simple input signals [14]. These neural structures have been replaced by artificial neural networks or system of coupled nonlinear oscillators to generate trajectories for legs in a synchronized fashion. this approach has been successfully used to control legged robots such as hexapods [15, 16], quadrupeds [17, 18] and bipeds [19, 20]. The evolution of legged robots led to having a more and more robust control strategy, heading toward locomotion in complex scenarios such as rough terrain or soft terrain [21]. Such control strategy can be roughly described as shown in figure 2.1. In this thesis we focus on the *Foothold selection* part, which is related to adjusting the landing position of a foot in swing according to some criteria.

2.2 Foothold selection in legged robots

Although there are many works about contact or foothold planning [22, 23, 24, 25], there are few examples of legged robots performing dynamic locomotion exploiting the knowledge about terrain morphology for contact selection, and even less that considers also dynamics. To actually move and achieve any motion, a system needs to exchange forces with the external world. The contact points are the locations in which the system can exchange forces with the environment through its limbs, resulting in an acceleration and, consequently, motion. Considering that the limbs have a finite length, the contact points have to change during the motion to continuously support the system.

Choosing correctly a sequence of contact point means searching for the sequence of foot locations that let the robot to exert forces to achieve the desired motion. Each type of mobile robot has its own different way to exchange forces with the environment. Comparing two main categories of ground mobile robots, i.e., wheeled and legged, we can explain how different the contact points sequence can be and why it is so important. In the case of wheeled robots, the contact points are the points in which the wheels are in contact with the ground. Wheels exert a force on the contact point opposite to friction in order to move in a certain direction and motion of the robot's base can be achieved without the need of a wheel losing contact from the ground. In the case of legged robots, the robot exerts a force against the contact points located at its feet, however, there is a limited amount of force that the robot can exert on the ground that depends on the robot's

configuration, friction and on the dynamics of the motion. To move, the robot needs to constantly make and break new contacts with the ground to overcome these limitations on the ground reaction forces. The set of contact points changes in number, having to lift a leg in order to move it to the next location and make a new contact with the ground, generating a discontinuous sequence of contact points in time. In both cases, the choice of contact points (planning) is fundamental to successfully complete the task.

Tonneau et al. [26] tackle the problem of contact positioning. The algorithm presented is mainly focused on generating a feasible reference trajectory and then finding the contacts to achieve a desired motion. They find equilibrium feasible robot configurations by generating and analysing the intersection of a *reachable space* with the environment. The two problems (trajectory and contact planning) are addressed in a decoupled fashion, computing the trajectory and then the contact positions. The main drawback, also mentioned in [26], is that the motion needs to evolve through equilibrium phases, then motions that require statically unstable phases, such as trotting, are excluded and cannot be generated.

There are algorithms that focus on selecting contacts on line to react to the environment properly. In [27, 28, 6, 29], the problem of foothold selection is addressed in different ways. *Villarreal et al.* [6] evaluate potential footholds for dynamic gaits by means of different criteria. A heuristic algorithm is used to evaluate the terrain morphology combined with robot's kinematics. The terrain is acquired using vision systems, allowing the algorithm to sense a previously unknown terrain. To speed up the evaluation process and make the algorithm suitable to run on line, a Convolutional Neural Network (CNN) is trained. To further improve such algorithm, *Villarreal et al.* [30] add an MPC controller to the previous foothold adaptation strategy. Other approaches [27, 28, 31] consider a similar methodology. They use an algorithm to evaluate terrain and kinematics to select footholds that can yield a feasible trajectory. More recently, the heavy computational load required by algorithms has been tackled with the adoption of GPU (Graphical processing Unit) computation [29]. They still consider the problem of foothold selection, but instead of using a neural network to reduce the computation times and make it suitable for on line execution, the computational complexity of the problem is addressed by performing the evaluation on a GPU. These works tackle foothold selection problem using multiple criteria that take into account kinematics, geometry and terrain morphology, but a dynamical criterion

applied to foothold selection is still missing.

To describe robot dynamics, different formulations have been proposed, resourcing to a simplified model [32]. Most of the proposed solutions to control the robot dynamics do not consider the possibility of changing your landing location to improve the reliability of the motion [33, 34]. Instead the rely on fast reactions to unexpected behaviours when stepping on undesired locations such as edges, obstacles or holes [35, 36, 30]. The main difference between foothold adaptation and foothold planning is that the former is able to change the landing position selected by the latter. The major drawback is that the computation times required are in the order of tens of seconds to generate a series of footholds over a specified time horizon for a desired trajectory. This makes the algorithm not suitable for online applications.

A remarkable approach which implements a dynamic model combined with an MPC is the one formulated by *Grandia et al.* [37]. They apply the concept of Control Barrier Functions (CBFs) to constrain the MPC, implementing this control strategy to ANYmal [38]. They achieve very good results in a stepping-stone environment, composed by several stones scattered on which the robot has to step without falling. The experiment is performed on a pre-mapped terrain, meaning that they lack a visual feedback to detect any environmental change that could happen during the execution of the task.

Lin et al. [25] apply a dynamic approach to the contact planning problem. This work addresses the formulation of a dynamical contact planner that is able to generate a contact sequence to achieve the desired motion. A centroidal dynamics model is used to describe the robot's dynamics, which comprises trajectory, linear and angular momentum. The algorithm is then used to train a neural network to reduce the computation times. In a more recent work, *Lin et al.* [39] improves the contact planner by adding increasing its robustness toward disturbance rejection. They employ neural networks to learn system capturability given initial CoM conditions. They also consider hands as a way to exert forces on the environment and recover. The method proposed is able to plan in advance a sequence of robust footholds which allow the robot to walk toward a predefined objective. However, the environments in which they perform simulations are pre-computed and static, which may cause problems when dealing with previously unknown environments.

An interesting approach about dynamics is proposed by *Orin et al.* [32]. It considers joint torque transformations using Jacobian matrices, relating joint space

with dynamics. This is useful to impose physical limits on joints torques, making the connection with the physical problem more realistic. A different technique more focused on decoupling the problem from the joint space, exploits the equations in Cartesian space [40] to directly relate Ground Reaction Forces (GRF) to the dynamics of the robot. We adopt this description of GRFs to relate them to the robot motion, finding a suitable set of forces that yield a feasible trajectory subject to physical constraints. The limitations related to the assumptions made in [40] are that they are considering this method for static locomotion (crawling) as the assumption of $\dot{\mathbf{L}} = 0$ is not compatible with more dynamical gaits. The major drawback is the computation time required for the algorithm, which negates the availability for on line applications.

A similar approach to the one proposed in this thesis is presented by *Tsounis et al.* [41]. Herein, they use a Dynamical Transition Feasibility measure to learn if a planned foothold yields a dynamically feasible trajectory basing on [40], as we do. They tackle the problem by modifying gaits to adapt the steps to the environment, training a Neural Network to perform the adaptation during locomotion. Our work differs from it by considering a time horizon composed by an arbitrary number of subsequent phases, while they consider only one phase ahead. Furthermore, we extend [40] and drop the assumption of imposing an angular momentum rate equal to 0. The condition $\dot{\mathbf{L}} = 0$ holds when slow and quasi-static motions are considered. When dealing with faster, dynamic motions this assumption might be too restrictive to reflect the real robot behaviour.

2.3 Discussion

In the previous sections we have seen how the problem of legged locomotion has been widely addressed by developing different control strategies. When introducing exteroceptive sensors, the robot is able to react to environmental changes by means of visual feedbacks (Fig. 2.1). With the inclusion of visual feedbacks, the robots are a step closer toward becoming eclectic machines, able to act properly in different situations. The ability of legged robots to overcome rough terrain is mainly due to their discontinuous contacts sequence, which makes them able to carefully pick the legs landing positions and avoid unsafe terrain areas. Although there are many works that address the problem of foothold selection, not all of them adopt visual feedbacks to correct the planned trajectories to adapt to a changing or previous

unknown environment. Whereas those employing visual feedbacks are able to detect terrain features (such as terrain height, edges, or holes) few of them perform foothold selection considering dynamics and contact transitions.

We propose a foothold evaluation criteria able to select the footholds that yield a dynamic feasible transition over a specified time horizon. This method makes use of visual feedback, being able to evaluate previously unseen terrain according to robot's dynamics. It considers also cases in which $\dot{\mathbf{L}} \neq 0$, making it suitable to perform foothold selection for more dynamical gaits. Moreover, it is able to generate a dynamically feasible trajectory and a set of forces considering the angular momentum rate $\dot{\mathbf{L}}$ along the evaluated time horizon. In the following chapter, we will describe briefly the platform on which we perform simulations, reporting its physical characteristics.

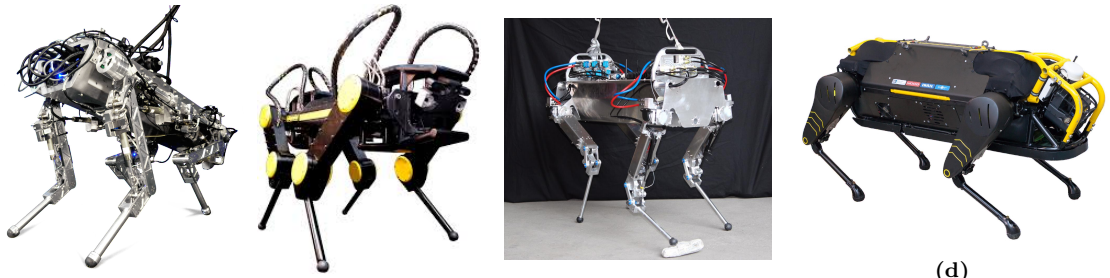
Chapter 3

System Overview

This thesis has been written as part of a collaboration with the Dynamic Legged Systems (DLS) lab at the Istituto Italiano di Tecnologia (IIT). This thesis focuses on improving dynamic locomotion of legged robots by including dynamic evaluations in the selection of contact locations. The method proposed in this thesis makes use of the Hydraulically actuated Quadruped robot (HyQ) to be tested and evaluated.

In this Chapter we describe briefly the history of the DLS lab and the platforms that have been developed until now, dedicating a section to the platform that will be used (HyQ).

3.1 Dynamic Legged Systems lab



(a) **(b)** **(c)** **(d)**

Figure 3.1: a) HyQ, **b)** HyQ2Max, **c)** MiniHyQ, **d)** HyQReal.

Istituto Italiano di Tecnologia[1] (IIT) is a Foundation financed by the State to conduct scientific research in the public interest, for the purpose of technological development. IIT's total staff comprises 1890 people from more than 60 countries, making IIT an international center of research that connects the world. The Headquarters located in Genova (GE) are the core of a scientific institute that focuses primarily on four fields: Technologies for Life Science (LifeTech), Computational Science, Nanomaterials and Robotics.

The Dynamic Legged Systems[2] (DLS) lab focuses its efforts in researching and developing high-performance, versatile legged robots. With the idea of being able to deal with scenarios dangerous for the human life, the DLS aims to develop quadrupeds that can help in those situations where it is needed to explore a hazardous environment, such as exploring a field after a natural disaster.

To perform such tasks, the DLS lab aims at including vision systems [42], external disturbance rejection [43] and reactive controllers [35] to deal with unknown scenarios.

The DLS lab has developed several robots since the first leg design [44]:

- *HYQ* [45] (Fig. 3.1a) - It is a hydraulic quadruped robot and it is the first legged robot developed in the DLS lab. We will use this platform to perform simulations and evaluations.
- *HyQCentaur* [46] - A hydraulic robotic arm has been added to the HyQ

[1] `https://www.iit.it/about-us`

[2] `https://dls.iit.it`

platform to extend its capabilities. The arm has six torque-controlled degrees of freedom.

- *HyQ2Max* [47] (Fig. 3.1b) - This platform was born as an evolution of HyQ, with the idea to build a stronger, more robust and versatile robot than HyQ. HyQ2Max has a larger joint maximum torque and higher joint limits, without a significant weight increase with respect to HyQ. This allows HyQ2Max to perform more dexterous motions, such as self-righting.
- *HyQReal* [48] (Fig. 3.1d) - It is the flagship platform of the DLS. It is a torque-controlled hydraulic quadruped, designed thanks to the experience gained with all its predecessors. It has proven its strength by pulling a small passenger air plane[3] (Piaggio P180 Avanti) at the Genova Airport.

3.2 The Hydraulically Actuated Quadruped Robot HyQ

In this dissertation, we perform simulations on the platform HyQ. The HyQ platform has been developed by Dr. Claudio Semini [45] in 2010, and it is still widely used in the DLS lab for research purposes. HyQ is a hydraulically actuated quadruped of about 100kg whose dimensions with fully-stretched legs are (1.0m × 0.5m × 0.98m)(Length × Width × Height). The legs weigh about 9kg and it is powered by three torque-control hydraulic joints, three per each leg: Hip Abduction-Adduction (HAA), Hip Flexion-Extension (HFE), Knee Flexion-Extension (KFE). Each joint is equipped with three sensors: a magnetic absolute encoder, a relative optical encoder and a force/torque feedback sensor. Moreover, an Inertial Measurement Unit (IMU) is present in the body. In Table 3.1 the main characteristics have been reported [45].

3.2.1 Exteroceptive sensors

Throughout the years of research, vision systems and exteroceptive sensors have been used on the HyQ [49] and HyQReal platforms. There are two main types of sensors:

[3] `https://youtu.be/pLsNs1ZS_TI`

Robot's mass	90kg
Leg's mass	9kg
Leg's length	0.339m to 0.789m
Foot radius	2cm
N° of joints	12
HAA range of motion	-90 ÷ +30 deg
HFE range of motion	-70 ÷ +50 deg
KFE range of motion	+20 ÷ +140 deg
Operating pressure	160 bar
Maximum torque HAA	120Nm
Maximum torque HFE	150Nm
Maximum torque KFE	150Nm

Table 3.1: HyQ platform characteristics

- *RGB-D* sensor (intel RealSense) - it is a depth sensor and it is used for visual odometry and mapping;
- *LiDAR* sensor (Velodyne) - it is mainly used to correct the pose using a scan matching based on the iterative closest point (ICP) algorithm [50].

We will employ the exteroceptive sensors to build a discrete heightmap which contains information about terrain height. The heightmaps acquired are of 30 × 30 and 66 × 66 cm. The resolution of the measurements is of 2cm, which yields grids of 15 × 15 and 33 × 33 points respectively.

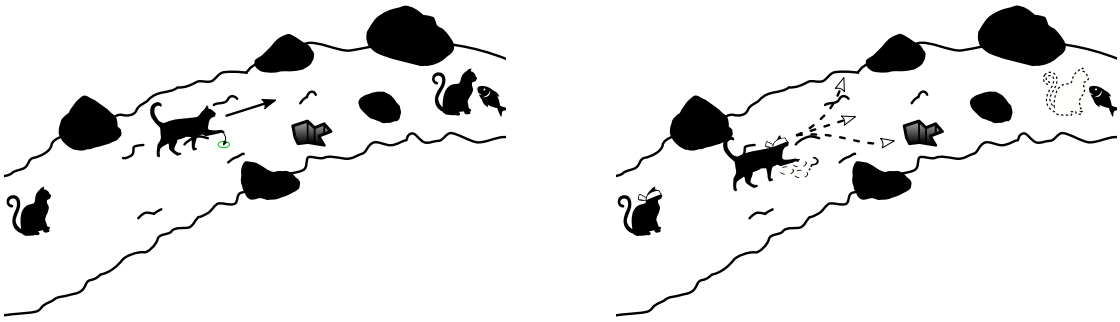
Chapter 4

Problem Formulation

Legged robot locomotion on rough terrain can become a hard task if not aided with environmental information. The knowledge of the environment as a mobile robot moves helps to evaluate how to act at any moment. To move over terrain, legged robots need to place correctly their feet. Aided with vision, a sequence of feet positions can be estimated such that the robot can avoid failures, such as falling. These sequence of positions in time are called a contact sequence. It needs kinematic and dynamic considerations to be correctly computed. It is needed to know where to exert forces and to adjust the robot's pose in space. The process of contact selection has to be constantly addressed during locomotion to adapt to a changing environment.

This chapter states the problem for the evaluation of potential contact locations. Our goal is to include dynamical stability when choosing a contact position during locomotion. We want to evaluate the potential landing locations of a leg in swing and discard those that do not yield a dynamically feasible motion in a given time horizon. Once all the unstable footholds have been discarded, an optimal foothold should be selected according to a criterion that captures the dynamic implications of the selection.

4.1 Vision-based Foothold Selection



(a) (b)

Figure 4.1: Example that shows the importance of vision during locomotion.

Consider a generic locomotion task (Fig. 4.1a), where a cat[1] has to walk across a pathway. The cat has to traverse on rough terrain to reach the end of the road. To successfully walk across the entire path, the cat has to choose where to place its feet to move forward. Based on its experience, it chooses the best place to locate its feet while walking to avoid falling and to reach its objectives such as spending less effort to move forward or avoid obstacles. The cat cannot plan in advance all the steps that it will take because if something changes in the environment it may fall or hit an obstacle. Then it has to choose where to place its feet while walking. After some time, the cat manages to reach the end of the path successfully reaching its objective. If the cat was blindfolded (Fig. 4.1b), reaching its destination would have been harder if not impossible because of all the obstacles in its path. Without visual feedback, the cat cannot choose where to place its feet according to the environment. It can only make decisions through proprioception and memory, leading to inaccurate walking and to an impossibility of reacting to environmental changes.

Having a vision-aided locomotion is useful to detect changes in the surroundings and to adapt continuously the locomotion to the terrain. Vision-based Foothold selection aims to embed vision systems in locomotion to select foot contact locations. With vision it is possible to acquire continuously terrain maps around the estimated foot landing positions. These maps are used to evaluate if there is the need of changing the foot landing positions (Fig. 4.2). Choosing among different contact

[1]Source: `http://www.onlinewebfonts.com/icon`, License: CC BY 3.0

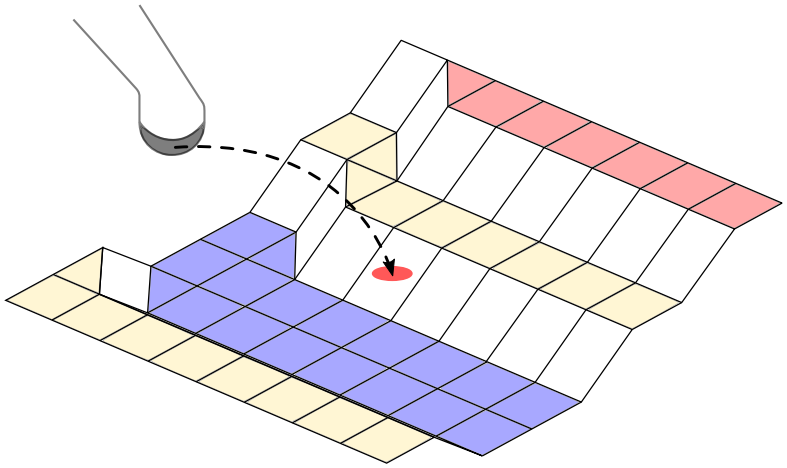


Figure 4.2: Example of heightmap where the estimated landing position (red circle) has to be discarded because near an edge.

points on rough terrain is a complex and computationally expensive process. The selection should consider both geometric [6] and dynamic aspects in order to discard unsafe foot locations. Geometry includes aspects such as the robot's kinematics, terrain morphology or leg collisions. While all of these aspects are fundamental to achieve a successful motion in space, they are not the only involved in locomotion. Other quantities such as the dynamic feasibility of the overall motion of the body or joint torque limits should also be considered when choosing where and how to take a step. Animals become aware of their limits through experiences during their life, allowing them to learn and act accordingly in different situations. Robots are built knowing what components they are made of, so their limits are known in advance. This makes us able to model their dynamics taking into account such limits.

With the consideration of a dynamic model during locomotion, the motion itself could benefit from it thanks to different aspects. A dynamic model takes into account forces and accelerations that play an important role. Having a model that can describe the relationship that exists between forces and motion lead to a better control of the robot's movement in space. Moreover, it can be used to describe and predict the evolution in time of a given system, knowing at least a set of initial conditions. In this case we want to use dynamics to foresee the future behaviour of the robot given the initial set of states in order to drive it to a final set of states. To achieve this, due to the hybrid (i.e., discrete and continuous time dynamics) nature of legged locomotion, a sequence of contact points has to be defined, i.e.,

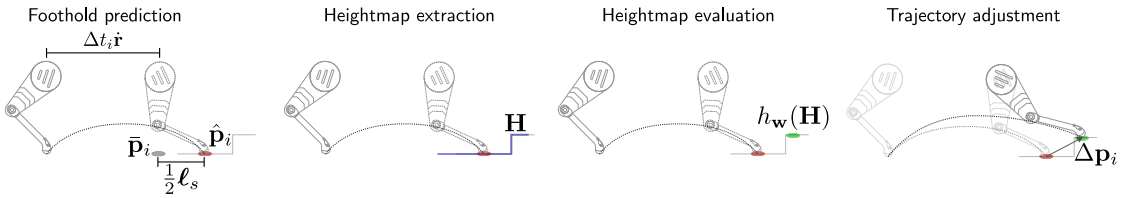


Figure 4.3: Phases of the VFA. From left (first) to right (last). (Source: [52]).

where it has to place its feet. Since contact points are the only way to exchange forces with the environment, the robot's behaviour depends on them.

Consider an example where a robot is moving forward. Let the foothold selection be performed only on kinematic quantities. Let's say that we want to have the robot come to a full stop after a certain number of steps, i.e., we want our system to be *capturable* [51]. Once the feet are on the ground, they cannot be moved assuming no slips. A bad choice of any landing position may lead to failures, such as falling. Geometrical aspects can be accounted to select footholds that are within the kinematic limits of the robot or to avoid undesired landing locations such as holes or sharp edges. However, it might be possible that the robot might not be able to generate the necessary ground reaction forces to achieve a certain motion for a specific foothold. This is information that the geometry of the locomotion phenomenon cannot provide. It would be useful to know the footholds that lead to undesired behaviors due to leg collisions or non-reachable foothold locations and discard them (see Fig. 4.2). Then considering an amount of possible contact points, it might happen that not all of them are compatible with the motion. Among the non-discarded feet locations, assuming that there are multiple compatible locations, only one has to be chosen to actually place the foot. This means choosing an *optimal foothold*, as the cat does while choosing where to place its paws. In [6] the optimal foothold is selected using a heuristic criterion that aims at disturbing the least the locomotion. Although this choice is reasonable and works well also with dynamic gaits, it does not embed any knowledge about the feasibility of the motion.

The evaluation described in [6] is divided in four main phases (see Fig. 4.3):

1. A *nominal foothold* is predicted based on the predefined trajectory of the leg and since the body moves as the legs are in swing, also the velocity of the trunk is used to compute the nominal foothold.

2. A *heightmap* is acquired in the vicinity of the nominal foothold. A heightmap is

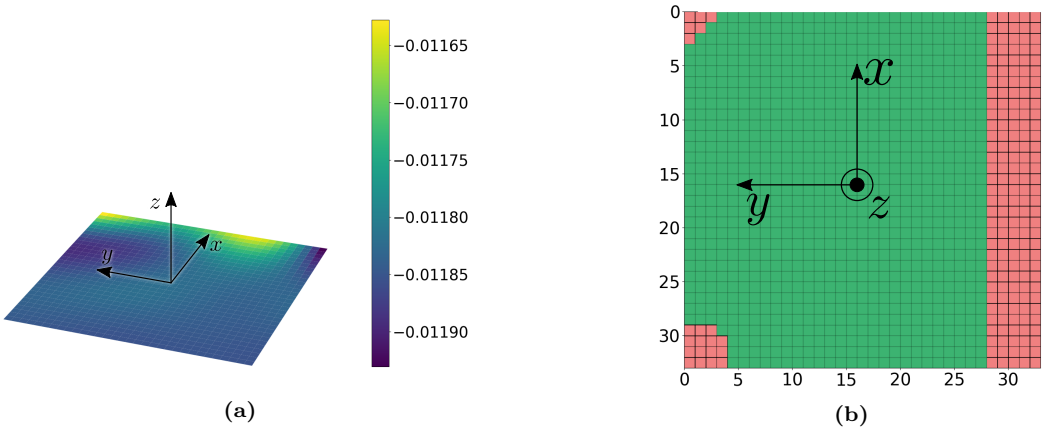


Figure 4.4: a) Acquired heightmap. **b)** VFA evaluation output.

a discrete representation of the terrain, where each pixel contains information on the height of that specific area.

3. The map is evaluated in search of feasible footholds (safe foot landing positions). Among the feasible footholds, an optimal foothold is chosen as the closest to the nominal feasible foothold.

4. Once the optimal foothold has been selected, the trajectory of the leg is continuously adjusted to adapt to the chosen location.

The foot trajectory is described by a half ellipse, where the major axis correspond to the step length. The nominal foothold is then computed using the following approximation:

$$\hat{\mathbf{p}}_i = \overline{\mathbf{p}}_i + \frac{1}{2}\mathbf{l}_s + \Delta t_i \dot{\mathbf{c}}_i \tag{4.1}$$

Where $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the nominal foothold position for leg i, $\overline{\mathbf{p}}_i \in \mathbb{R}^3$ is the center of the ellipse of leg i, Δt_i is the remaining swing time of leg i, $\mathbf{l}_s \in \mathbb{R}^3$ is the step length vector, and $\dot{\mathbf{c}} \in \mathbb{R}^3$ is the velocity of the base.

The evaluation process is a heuristic algorithm (VFA) that embeds evaluation criteria. These criteria check all the potential footholds in a given heightmap to test their feasibility according to geometrical aspects. The considered criteria are:

- *Kinematics* - evaluates if the foothold is outside the workspace of the leg
- *Terrain Roughness* - it accounts for the slope of the terrain, compute a threshold that if violated, marks the foothold as not feasible

- *Frontal Collision* - it checks for foot collisions. If a foot intersects the terrain at any point along its trajectory, the foothold is discarded
- *Leg Collision* - as the frontal collision criterion, it accounts for collisions but for both leg limbs for the whole leg cycle

The VFA algorithm processes a heightmap to evaluate it according to its criteria. Each foothold (map pixels) (Fig. 4.4a) is tested with all criteria to state if it is feasible or not. The evaluation produces a new map that contains the feasibility information about the heightmap, marking every feasible foothold (Fig. 4.4b). The criteria used to perform an evaluation consider only geometric aspects and an optimal foothold is selected among the ones that belong to the feasible set. The optimal one is selected as the feasible foothold closer to the nominal one. In this example, the nominal and optimal foothold are the exact same foothold, as the nominal foothold is also feasible.

This evaluation process allows to discard those footholds that are outside the workspace of the robot, that lead to collisions or are located on undesired locations such as holes or edges. All the footholds that are VFA-feasible (using VFA-feasible to describe a foothold that fulfil all VFA criteria) do not take into account any dynamic criterion. It may happen that a foothold is within the workspace but the forces required to achieve a motion using that foot position cannot be generated by the robot. Footholds with such conditions are not dynamically feasible.

In this chapter we explore an evaluation criterion that embeds dynamics in foothold selection to improve the geometric-related criteria presented in [6]. We consider a time horizon for which the footholds must yield a feasible motion, including also the various steps and discontinuities (contact switches) that arise. The footholds are discarded if they don't yield a dynamically feasible transition along the entire horizon considered.

4.2 Dynamic Transition Feasibility

The main problem addressed is to evaluate the terrain in search for dynamically feasible footholds and then to select the optimal foothold according to a quantitative measure of dynamic feasibility. This means that, after the analysis of the terrain and robot morphology, a set of dynamically "feasible" foot locations are selected. These contact points are chosen considering the robot's future behavior during a

specified time horizon. The feasibility of a foothold is asserted by checking if it can yield a dynamic transition for the considered time horizon. To move between two points in space, the robot goes through a sequence of phases. Each phase consists in a combination of swinging and standing legs. These phases can be also considered from each leg's point of view. Considering a single leg, it will go through a defined number of phases during a given time interval (Fig. 4.5).

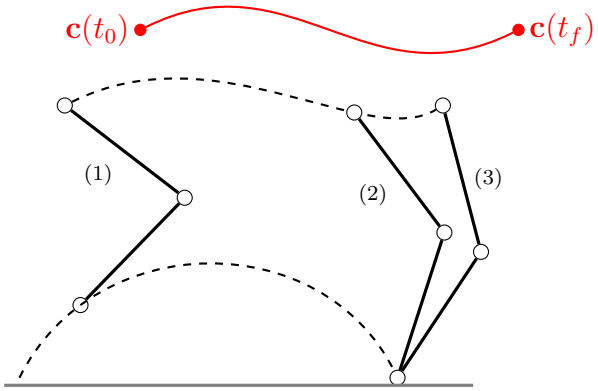


Figure 4.5: Considering one single leg, we can divide its movement in three different phases: (1) the leg is in swing and the foot is moving along its trajectory (lower dashed); (2) leg makes contact with the ground; (3) leg stays in contact with the ground and the hip moves forward as other legs are lifted and moved. The CoM trajectory (red) and the hip trajectory (upper dashed) are considered in the world reference frame.

The connecting point between subsequent phases is called *contact switch*, which is also the instant in which a leg touches (touchdown) or leaves (lift-off) the ground. Including all the contact switches that happen in the given time horizon is useful to evaluate the feasibility of the analyzed foothold ensuring the dynamic stability of the future robot's behavior. The idea is then to analyze the vicinity of the *nominal foothold* to create a set of feasible footholds that ensure dynamic stability throughout the entire predicted motion.

From now on, the analysis is focused on the nominal foothold (i.e., the predicted foothold based on the default trajectory of the leg and the velocity of the robot's body), unless differently stated. This approach takes inspiration from [40] and relies on the Rigid Body Dynamic Model RBDM[2], which is a simplified dynamic model that considers the robot as a single rigid body that has the mass in its CoM. But, in contrast with [40] the main purpose of the developed algorithm is not to

[2]The reason behind this choice is explained in 5.1

generate a reference trajectory (although it is possible), but to find if a trajectory exists given the initial and final sets of states. The time horizon considered for the evaluation covers an entire leg cycle. During this time interval, multiple contact switches occur, leading to an increase of the problem complexity and to higher computation times. The complexity is a direct consequence of the fact that the contact switches are discontinuities in motion since they change the way in which the robot exerts forces on the ground.

A generic CoM state $\mathbf{x}(t)$ is defined as an array of values associated to the CoM's configuration. This array comprises all the quantities involved in the RBDM and it can be written as:

$$\mathbf{x}(t) = [\mathbf{c}(t)\ \dot{\mathbf{c}}(t)\ \ \ddot{\mathbf{c}}(t)\ \ \boldsymbol{\Theta}(t)\ \ \dot{\boldsymbol{\Theta}}(t)\ \ \ddot{\boldsymbol{\Theta}}(t)]^\top \tag{4.2}$$

$$\mathbf{c}(t) = [c_x\ \ c_y\ \ c_z]^\top \tag{4.3}$$

$$\boldsymbol{\Theta}(t) = [\phi\ \ \gamma\ \ \psi]^\top \tag{4.4}$$

Where $\mathbf{c}$ is the CoM trajectory, $\boldsymbol{\Theta}$ is the CoM orientation trajectory and the other quantities are defined as derivatives. The main objective is to find a feasible transition between two sets of states, namely, the trajectory that connects the two states as a function of time $\mathbf{f}(t)$ such that:

$$\mathbf{f}(t) : \mathbf{x}(t_0) \rightarrow \mathbf{x}(t_f) \qquad t \in [t_0, t_f] \qquad t_f > t_0$$

Five out of the seven states components can be derived through differentiation of the other two states. Velocity ($\dot{\mathbf{c}}$) and acceleration ($\ddot{\mathbf{c}}$) are derivatives of the CoM's trajectory ($\mathbf{c}$). The angular rate of change with respect to time ($\dot{\boldsymbol{\Theta}}$) and the angular acceleration ($\ddot{\boldsymbol{\Theta}}$) are derivatives of orientation ($\boldsymbol{\Theta}$). These relationships must be considered when choosing a function $\mathbf{f}(t)$. To account also for the derivative relationships, it is sufficient to choose $\mathbf{f}(t)$ only for the two primitive states ($\mathbf{c}$, $\boldsymbol{\Theta}$) and then obtain the other states by analytical differentiation.

Along with the two states $\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$, also the footholds are required to fully define the motion. They are essential to describe the type of locomotion that the robot has to perform because the gait parameters (such as step frequency, duty factor, CoM forward speed) change the dynamical behavior of the robot itself. Since the robot needs to walk to move its CoM in space, all the contact switches happening in the time horizon between these two states have to be considered as well. To account for these contact switches and to quantify them, a *contact switch*

(a) CoM's trajectory. **(b)** CoM orientation.

Figure 4.6: Examples of trajectories in space. Only position and orientation trajectories are represented to avoid overlapping lines.

horizon can be defined.

Analogously to a time horizon, a contact switch horizon defines a time interval. A contact switch horizon defines the period of time that spans from t_0 to t_f, expressing it as the number of contact switches happening in that period of time. The usefulness of this new definition becomes clear when considering that the time needed to complete a gait cycle (Fig. 4.5) is not always constant because it depends on the initial conditions. The quantity that is constant during a gait cycle is the number of contact switches happening during the cycle itself. The contact switches horizon concept is then a time transformation that converts a not constant quantity to a constant quantity. The period of time considered in the evaluation can be computed as follows:

$$T = t_f - t_0 = \frac{1}{f_s} - t_e \tag{4.5}$$

Where f_s is the step frequency and t_e is the phase elapsed time. The latter refers to the time elapsed since the phase (swing or stance) that the robot is executing has started. From this equation we can see that the period of time T is not constant as the walking motion is performed. Even considering f_s as a constant, which is not exactly true, T depends on the elapsed time and then it changes at every time instant according to t_e. If we use instead a contact switches horizon, the considered horizon does not change at every instant but it remain constant along the whole phase considered. The considered contact switch horizon is the number of contact switches that the robot has to perform to complete the cycle of the leg considered (Fig. 4.5).

To drive the robot from the initial state ($\mathbf{x}(t_0)$) to the desired final state ($\mathbf{x}(t_f)$), we can define a trajectory for the CoM. This trajectory represents the evolution of the CoM's motion in space, so it is a spatial trajectory defined by its components

(x, y, z) in a Cartesian space (Fig. 4.6a), with respect to the world reference frame ($\mathcal{W}$).

In the RBDM not only linear quantities are considered. *Angular quantities* play a fundamental role in the description of the body's motion. They describe the rigid body's orientation, angular rate, and angular acceleration with respect to a fixed RF. As done for the CoM's trajectory, we would like to describe the evolution of the orientation as a trajectory along the contact switch horizon. This would allows to express the angular rate and acceleration through differentiation, although the obtained trajectory would not have a representation in Cartesian space but rather in an angles space (4.6b).

4.2.1 Robot Dynamics

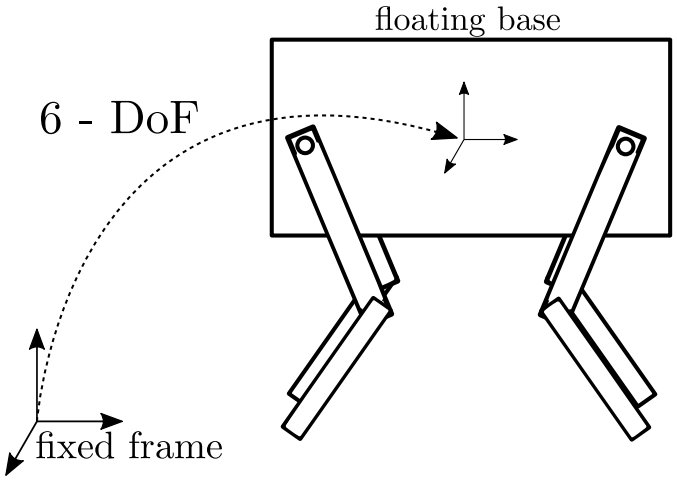


Figure 4.7: Fictitious 6 - DoF that connects fixed frame with the floating base.

We need to describe the evolution in time of our system's dynamics. To do that, we need to model the system using a mathematical description that takes into account the evolution in time of the system components. That description is referred as *dynamic model*. A generic model of a robot mechanism can be represented using a *connectivity graph*, which is a collection of nodes where each node represents a joint. The graph must always be connected to a fixed base or reference frame. If we want to represent a mobile robot (such as a legged robot) it is necessary to add a fictitious 6-DoF joint between a fixed base and a link in the robot [53]. The chosen link is called *floating base*. Then if the system considered has n_s degrees of freedom (DoFs), adding the extra joint we have a $n'_s = 6 + n_s$

DoF system. Note that a 6-DoF joint imposes no constraints and then it does not alter the physical properties of the system. It allows any translation (3 DoF) and any rotation (3 DoF).

The dynamic model for a generic n-DoF mobile system (floating base) can be described by a joint-space formulation [53]:

$$\mathbf{H}(\mathbf{q})\dot{\boldsymbol{\nu}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\boldsymbol{\nu} + \boldsymbol{\tau}_g(\mathbf{q}) = \boldsymbol{\tau} + \mathbf{J}^\top \mathbf{f}_e \tag{4.6}$$

Where $\mathbf{H}(\mathbf{q}) \in \mathbb{R}^{n'_s \times n'_s}$ is the so called the joint space inertia matrix, $\boldsymbol{\nu} = [\mathbf{v}^\top \quad \dot{\mathbf{q}}^\top]^\top \in \mathbb{R}^{n'_s}$ is the vector of the generalized velocities where $\mathbf{v} = [\dot{\mathbf{c}}^\top \ \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$ is the floating base velocity and $\dot{\mathbf{q}} \in \mathbb{R}^{n_s}$ represents the vector of joint velocities. $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n'_s \times n'_s}$ is a matrix such that $\mathbf{C}\boldsymbol{\nu}$ is the vector of Coriolis and centrifugal terms, $\boldsymbol{\tau}_g(\mathbf{q}) \in \mathbb{R}^{n'_s}$ is the vector of gravity terms, $\boldsymbol{\tau} \in \mathbb{R}^{n'_s}$ is the vector of joint torques and $\mathbf{J}^\top \mathbf{f}_e \in \mathbb{R}^{n'_s}$ is the term that accounts for external forces, where $\mathbf{J} \in \mathbb{R}^{n'_s \times n'_s}$ is the contact points Jacobian.

We can split equation (4.6) in two parts associated to the unactuated and the actuated parts of the system:

$$\begin{bmatrix} \mathbf{H}^u & \mathbf{H}^{ua} \\ \mathbf{H}^{au} & \mathbf{H}^a \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \ddot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{C}^u \mathbf{v} \\ \mathbf{C}^a \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\tau}_g^u \\ \boldsymbol{\tau}_g^a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_u^\top(\mathbf{p}) \\ \mathbf{J}_a^\top(\mathbf{p}) \end{bmatrix} \mathbf{f}_e \tag{4.7}$$

Where $\mathbf{p} = [\mathbf{p}_1^\top \quad \dots \quad \mathbf{p}_{nc}^\top] \in \mathbb{R}^{3 \times nc}$ is the stacked vector of contact positions and nc is the number of contacts. The superscripts u, a, and ua stand for unactuated, actuated and the cross terms between unactuated and actuated parts, respectively.

To find how to employ this dynamics model, we can rewrite the model in such a way that the body states, the forces, and the foothold locations are explicitly stated. From (4.2) we can associate linear and angular quantities to the model:

$$\dot{\mathbf{v}} = [\ddot{\mathbf{c}}^\top \quad \dot{\boldsymbol{\omega}}^\top] \tag{4.8}$$

$$\mathbf{v} = [\dot{\mathbf{c}}^\top \quad \boldsymbol{\omega}^\top] \tag{4.9}$$

$$\boldsymbol{\omega} = \underbrace{\begin{bmatrix} \cos\gamma\cos\psi & -\sin\psi & 0 \\ \cos\gamma\sin\psi & \cos\psi & 0 \\ -\sin\gamma & 0 & 1 \end{bmatrix}}_{T(\gamma,\psi)} \dot{\boldsymbol{\Theta}} \tag{4.10}$$

Where the relationship between $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\Theta}}$ is given by the matrix $T(\gamma,\ \psi) \in \mathbb{R}^{3\times3}$. Note that the body states depend on the joint variables because the motion itself is a direct consequence of the motion of the joints. Making the dependence explicit, we can say that $\mathbf{c}(t, \mathbf{q}(t))$ for the CoM's trajectory. It is worth noticing that the CoM's velocity and acceleration are dependent on $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$.

Moreover, the contact Jacobians are computed using the feet positions. We can highlight this dependency by stating it explicitly. Consider the rightmost term of equation (4.7), we can write the unactuated part as:

$$\mathbf{J}_u^\top(\mathbf{p})\mathbf{f}_e = \begin{bmatrix} \mathbf{S}(\mathbf{p}_1) & \dots & \mathbf{S}(\mathbf{p}_{nc}) \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{nc} \end{bmatrix} \tag{4.11}$$

Where $\mathbf{S}(\cdot)$ stands for the skew-symmetric operator used to represent a cross product:

$$\mathbf{a} \times \mathbf{b} = \mathbf{S}(\mathbf{a}) \times \mathbf{b} \tag{4.12}$$

Each contact point $\mathbf{p}_i \in \mathbb{R}^3$ is a three-dimensional vector defined in a Cartesian space. From this relationship, we can see how the correct foothold placement is really important to correctly drive the system, as it has a direct influence on the inputs of the system.

Since we are interested in controlling the base, we focus on solving the top part of (4.7). It is related with the unactuated part of the system:

$$\mathbf{H}^u\dot{\mathbf{v}} + \mathbf{H}^{ua}\ddot{\mathbf{q}} + \mathbf{C}^u\mathbf{v} + \boldsymbol{\tau}_g^u = \mathbf{J}_u^\top\mathbf{f}_e \tag{4.13}$$

Equation (4.13) can be decomposed into the linear and rotational dynamics of the system:

$$m \cdot (\ddot{\mathbf{c}} - \mathbf{g}) = \sum_{i}^{nc} \mathbf{f}_{e,\,i} \tag{4.14}$$

$$\mathbf{I}(\mathbf{q})\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}(\mathbf{q})\boldsymbol{\omega} + \mathbf{H}^{ua}\ddot{\mathbf{q}} = \sum_{i}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i} \tag{4.15}$$

A stable locomotion requires not only great knowledge about the operational space of the robot (kinematics). The knowledge of the actual workspace of the system could not be sufficient in some situations where highly dynamical motions are involved. Then to achieve dynamically feasible locomotion also dynamics must be considered. A way to consider dynamics is to evaluate the robot's behaviour when selecting footholds.

Foothold selection is a complex process where the terrain is analysed to choose where to place the legs in order to take a step. The choice must be carefully weighted in order to grant stability in the future and avoid unwanted behaviours, e.g., falling. The most intuitive way of conceiving a robot falling is trying to reach configurations outside the kinematic workspace of the robot. However, a robot may also fall due to the dynamic infeasibility of a required motion, which is a more abstract concept. Herein, we refer to dynamic feasibility as the ability to maintain forces and accelerations compatible with a trajectory for a given period of time. We would like to evaluate this dynamic feasibility when selecting footholds.

A foothold is feasible if it is able to yield a trajectory that fulfils the dynamic equation (4.6). We need to find the functions $\mathbf{c}(t), \boldsymbol{\Theta}(t)$ that are able to describe accurately the robot's trajectory and its dynamics for every potential foothold in a heightmap. If such function exists, then the foothold examined is deemed as feasible.

Solution of the transition feasibility problem

The objective is to find an evolution in time of the robot's linear trajectory and its orientation, given by the functions $\mathbf{c}(t)$ and $\boldsymbol{\Theta}(t)$. We know the initial state and we choose a final desired state where to drive the robot. Given those sets of two states, we can then formulate a way to embed the dynamic model considering as unknowns the quantities required for the motion, i.e., $\mathbf{c}(t)$, $\boldsymbol{\Theta}(t)$, $\boldsymbol{\tau}(t)$ and $\mathbf{f}_e(t)$, namely, CoM

linear trajectory, robot's orientation, joint torques and external forces. With the states relationships in mind (4.8), (4.9) and (4.10), a way to find all the required quantities is to set up a feasibility problem of the form:

$$\begin{aligned}
&\text{find} && \mathbf{c}(t), \boldsymbol{\Theta}(t), \boldsymbol{\tau}(t), \mathbf{f}_e && \text{(4.16a)}\\
&\text{such that} && \mathbf{H}^u\dot{\mathbf{v}} + \mathbf{H}^{ua}\ddot{\mathbf{q}} + \mathbf{C}^u\mathbf{v} + \boldsymbol{\tau}_g^u = \mathbf{J}_u^\top(\mathbf{p})\mathbf{f}_e && \text{(4.16b)}\\
& && \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \leq \mathbf{q}_{max}, \dot{\mathbf{q}}_{max}, \ddot{\mathbf{q}}_{max} && \text{(4.16c)}\\
& && \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} && \text{(4.16d)}
\end{aligned}$$

Where there are constraints on the joint kinematic (4.16c) and torque (4.16d) limits.

4.3 Discussion

In this chapter we have modelled the dynamics of a floating base robot, describing it using a joint-space formulation (4.6). It is composed by two part, namely, the unactuated and the actuated part, but we focus only on the unactuated part (4.13) since we are interested in controlling the base. We can further divide the unactuated part, separating the linear and rotational dynamics of the system (4.14) and (4.15).

To find all the quantities that are needed to completely describe the motion of the base, we set up an optimization problem (4.16a) that considers the unactuated dynamics to find the evolution in time of the robot's linear trajectory and its orientation.

Including dynamic feasibility as a foothold evaluation criterion presents itself as a highly complex problem. Both the nonlinear and hybrid dynamics, as well as the large dimensionality of the problem can lead to intractability of the problem. The feasibility problem should be solved for each potential foothold location in a heightmap, as the VFA algorithm does with geometric-related criteria (Fig. 4.4). Furthermore, in (4.13) the complexity of the unactuated part is highlighted. Such complexity is given by the nonlinearities introduced by the rotational component (4.15).

Our approach aims at building a simplification of this feasibility problem, expressing the quantities in a Cartesian space and not in a joint space and trying to reformulate the problem as convex optimization. Convexity is a useful property

because it guarantees to have a solution space with a unique minimum. In the next chapter, we try to avoid the nonlinearities using a different and simplified formulation of the problem. By exploiting the Neuton-Euler formulation for the Rigid Body Dynamic Model (RBDM) and describing the trajectory of the CoM as a polynomial curve, we avoid the nonlinearity of the problem

Chapter 5

Dynamically Feasible Vision-Based Foothold Adaptations for Legged Locomotion

Solving the problem with a full dynamics model is a complex task. We choose to tackle the problem considering a simplified dynamics model, namely the Rigid Body Dynamic Model (RBDM). With this model we neglect the leg inertia and we consider that the total mass of the robot is concentrated at the Center of Mass (CoM). This assumption allows us to consider the robot as a single rigid body that moves in space. Although this criterion does not consider kinematic limits of the robot, they are included using as input the feasible map obtained with [6].

To formulate the convex optimization problem, we extend the method proposed in [40], namely Convex Continuous Continuous and Convex Resolution of Centroidal dynamic trajectories for legged robots in multi-contact scenarios (C-CROC) to evaluate multiple foothold locations along a given contact switch horizon. Furthermore, we extend the method to be able to find feasible trajectories with underactuated dynamics such as trot. This allows us to use our evaluation also for dynamic gaits, without compromising the convexity of the problem.

5.1 Rigid Body Dynamics Model (RBDM)

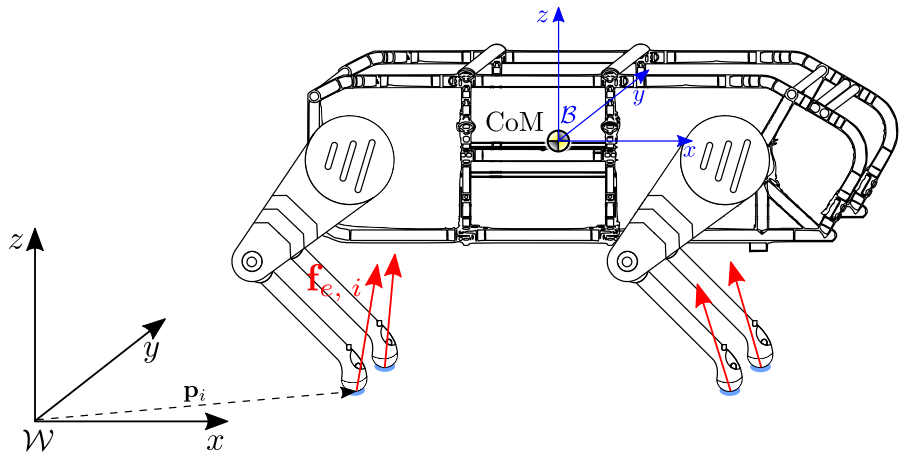


Figure 5.1: Schematic of the HyQ robot. The reference frame used to express all the quantities is $\mathcal{W}$ (world).

To tackle the complexity of the dynamic model described in (4.16a) there are different approaches. The approach that we consider is to use a simplified model representation of (4.6) that considers the unactuated dynamics of the robot (4.13), namely, the RBDM. It considers the robot as a rigid body moving through space subject to contact forces applied at each foot contact. The unactuated part can be decomposed in its linear and rotational part (4.14, 4.15). These two parts can be described with a Newton-Euler formulation [40], expressed in a fixed frame (world frame ($\mathcal{W}$) - Fig. 5.1):

$$m(\ddot{\mathbf{c}}(t) - \mathbf{g}) = \sum_{i=1}^{n} \mathbf{f}_{e,\,i}(t) \tag{5.1}$$

$$m\mathbf{c}(t) \times (\ddot{\mathbf{c}}(t) - \mathbf{g}) + \dot{\mathbf{L}}(\boldsymbol{\Theta},\ \dot{\boldsymbol{\Theta}}, \ddot{\boldsymbol{\Theta}}) = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{5.2}$$

Where:

- $m \in \mathbb{R}^1$ is the robot's mass
- $\mathbf{c}(t) \in \mathbb{R}^3$ is the CoM trajectory
- $\ddot{\mathbf{c}}(t) \in \mathbb{R}^3$ is the second derivative of $\mathbf{c}(t)$, i.e., the CoM's acceleration
- $\mathbf{g} \in \mathbb{R}^3$ is the gravity acceleration

- $\mathbf{f}_e \in \mathbb{R}^{12\times 1}$ is the stacked vector of ground reaction forces $[\mathbf{f}_{e,\,i},\ \ldots,\ \mathbf{f}_{e,\,nc}]^\top$, $\mathbf{f}_{e,\,i} \in \mathbb{R}^{3\times 1}$

- $\dot{\mathbf{L}} \in \mathbb{R}^3$ is the base angular momentum rate expressed in the world frame

- $\boldsymbol{\rho}_i \in \mathbb{R}^3$ is the ith contact point coordinate vector

- $nc \in \mathbb{R}^1$ is the total number of contact points

The reason behind the choice of this formulation is not only due to the complexity of the full dynamics model, but also to keep the feasibility problem convex. In [40], a different formulation of this model is presented. It is formulated using the Double Description method [54], which allows a non-trivial inequality formulation of the same problem:

$$\mathbf{H}\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g} + \dot{\mathbf{L}}) \end{bmatrix} \leq \mathbf{h} \tag{5.3}$$

Where $\mathbf{H}$ and $\mathbf{h}$ are a matrix and a vector, respectively, that depend on the contact locations, their normal and on the friction coefficients. Although it reduces the number of variables in the system since the forces are not included any more in the evaluation, computing $\mathbf{H}$ and $\mathbf{h}$ is a non-trivial operation and it is subject to occasional failures. In contrast to [40], we opt for an equality formulation:

$$\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g} + \dot{\mathbf{L}}) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \ldots & \mathbf{I}_3 \\ \mathbf{S}(\mathbf{p}_1) & \ldots & \mathbf{S}(\mathbf{p}_{nc}) \end{bmatrix} \mathbf{f}_e \tag{5.4}$$

We can then state a new feasibility problem that includes the simplified dynamic model as constraints. Assuming an ideal foot with infinite stiffness (no deformation hypothesis) and an infinitely hard terrain (no penetration hypothesis), the force exchanged by a single foot with the ground is composed by tangential components ($[x\ y]$ plane) and a normal component (along z direction). The robot can exert forces at the contact points by actuating the hip and knee joints to generate a torque at its joints and then a force at the end points of its limbs. To include the friction constraints, a linearized Coulomb friction cone model is considered (Fig. 5.2a). In this approximated model, the friction is assumed to be equal on both x and y directions.

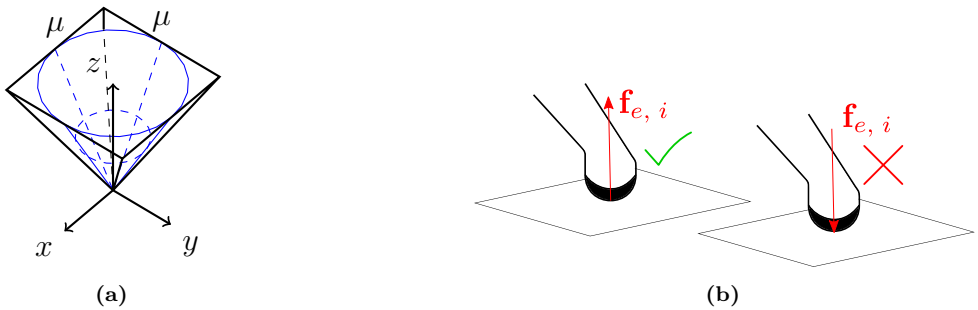


Figure 5.2: a) Friction cone approximation. Coulomb friction cone (blue) and its pyramidal approximation (black). **b)** Visualization of the unilaterality constraint. The forces depicted are GRFs, which means that are the forces that the ground exert on the robot.

$$\text{find} \qquad \mathbf{c}(t),\ \mathbf{f_e}(t) \tag{5.5a}$$

$$\text{such that} \qquad m(\ddot{\mathbf{c}}(t) - \mathbf{g}) = \sum_{i=1}^{n} \mathbf{f}_{e,\,i}(t) \tag{5.5b}$$

$$m\mathbf{c}(t) \times (\ddot{\mathbf{c}}(t) - \mathbf{g}) + \dot{\mathbf{L}}(\mathbf{\Theta},\ \dot{\mathbf{\Theta}}, \ddot{\mathbf{\Theta}}) = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{5.5c}$$

$$\mathbf{f}_e \leq \mathbf{f}_{e,\,max} \tag{5.5d}$$

$$\parallel f_x \parallel \leq \mu f_z \tag{5.5e}$$

$$\parallel f_y \parallel \leq \mu f_z \tag{5.5f}$$

$$f_z \geq 0 \tag{5.5g}$$

Where (5.5e, 5.5f) are the friction constraints, which are modelled as an approximation of the friction cone. The maximum force constraint (5.5d) limits the force according to the capabilities of the robot. The unilaterality constraint (5.5g) is a constraint that define the impossibility of "pulling" the ground, i.e., the force can only be exerted in one direction (towards the ground, "pushing") (Fig. 5.2b). To achieve a desired motion of the trunk, the robot can use its legs to push against the ground and generate ground reaction forces GRFs for each foot in contact.

Let's examine a bit further the components of the new feasibility problem, keeping in mind that we want to keep our problem convex. In our feasibility problem there are two nonlinearity sources, making the problem non-convex. One is caused by the term $\dot{\mathbf{L}}$, which has a nonlinear dependence on angular quantities and we will describe it in Section 5.5. The other non convexity is caused by the

cross product $\mathbf{c} \times \ddot{\mathbf{c}}$. It is a non convex term because it is a multiplication between two variables. Since they are not constrained in any way on their values, their product is not defined and then convexity cannot be ensured. To make up for this problem, the trajectory function $\mathbf{c}(t)$ must be carefully chosen. In the following, we formulate the convex optimization problem by constraining the trajectory of the CoM to belong to a space of Bézier curves similarly as how was it done in [40].

5.2 Convex formulation

Here we will analyse how the problem can be formulated in a convex form by choosing carefully the function to describe the trajectory of the CoM, $\mathbf{c}$. In order to choose a suitable function, we need to analyse the required characteristics that the function should address.

5.2.1 Trajectory properties analysis

The function $\mathbf{c}(t)$ must be chosen carefully because the physical consistency of the trajectory depends on it. The function $\mathbf{c}(t)$ has to connect two states in time, namely the initial state ($\mathbf{x}(t_0)$ or $\mathbf{x}_0$) and the final state ($\mathbf{x}(t_f)$ or $\mathbf{x}_f$). Then, the function has to fulfill some requirements in order to be considered suitable for the description of the problem. The requirements can be summarized as follows:

- *Continuity* - it is required to correctly address the physical behavior of the robot motion.
- *Derivability* - this requirement grants that the state's components that depend on the derivative relationship ($\dot{\mathbf{c}}$, $\ddot{\mathbf{c}}$ and $\dot{\boldsymbol{\Theta}}$, $\ddot{\boldsymbol{\Theta}}$) can be computed considering $\mathbf{c}$ and $\boldsymbol{\Theta}$ respectively as primitives. This requirement is an extension of the previous one.
- *Imposition of initial and final conditions* - this requirement is fundamental to relate the function to the physical quantities of the problem. It is related to how we are posing the problem, as we will see more in depth in the next section. The function must include the quantities defined at the initial and final instants to connect them in space.

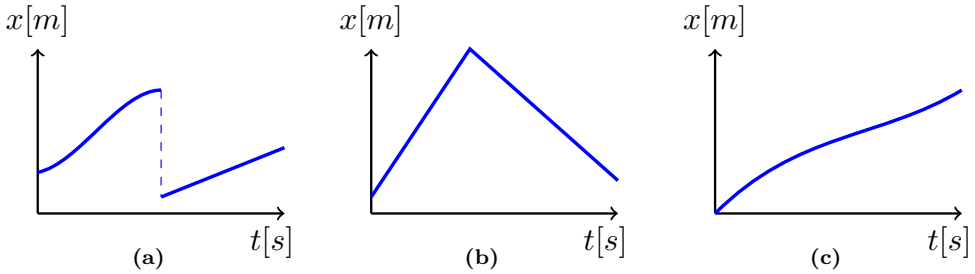


Figure 5.3: Examples of possible functions. The function depicted in (a) is not suitable because a rigid body can't move in space instantaneously. In (b) the function represented is still physically unfeasible because there are no jumps in space, a discontinuity would be present in its derivatives (velocity, acceleration, jerk), which is not physically possible. An example of physically meaningful function is shown in (c).

Fig. 5.3 shows an example that explain why the requirements can be reasonably assumed.

Linear	
Initial	**Final**
Position Velocity Acceleration	
3	**3**
6	

Angular	
Initial	**Final**
Orientation Orientation rate Angular acceleration	
3	**3**
6	

Table 5.1: In this tables the number of conditions to impose is resumed and reported at the bottom.

Trajectory free control point

A generic curve can describe a trajectory in space. It is considered *fully defined* when all of its coefficients are defined. The problem with having a fully defined curve is that it corresponds only to one specific trajectory in space. Considering that we want to describe the CoM's trajectory with a curve, a single solution constitutes a highly reduced solution space to find a trajectory to connect the set of initial and final states. There could be multiple trajectories that can connect two given states in space and time (Fig. 5.4a). In order to give the optimization a larger solution space to look for possible trajectories, we need to give to the function $\mathbf{c}(t)$ the ability to describe a set of trajectories instead of just one.

Having a free parameter yields a set of functions and then multiple solutions

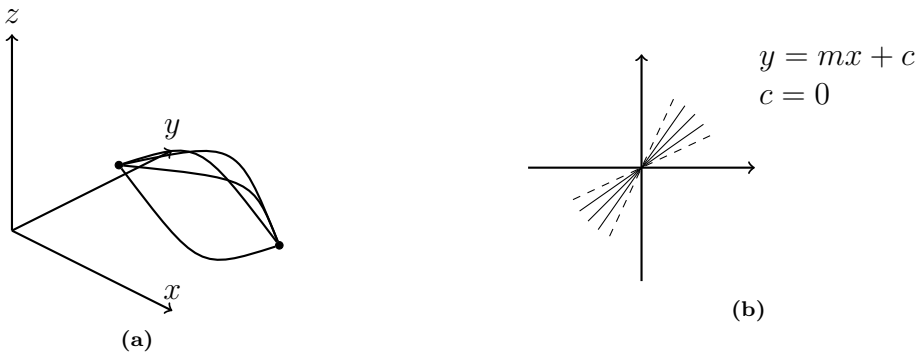


Figure 5.4: a) Example of 3D set of curves with the same initial and final conditions. **b)** Analogy with a set of straight lines with a free parameter.

consistent with physical parameters. This is analogue to the case of a straight line where only one of the two parameters is defined(5.4b).

In synthesis, we need a parametric curve that:

1. Can satisfy the physically meaningful requirements described by (5.2.1)
2. Can easily include a free parameter to generate a set of trajectories instead of just one
3. Can be included in the dynamics described by Equation (4.6)

As seen in 5.2.1, a possible choice is a polynomial function that depend on a parameter t, i.e., we need a parametric curve. This curve should have all the characteristics needed to bring physical consistency to the problem. A possible choice is to describe the trajectory using a *Bézier* curve. In the following we will describe how to formulate the dynamic feasibility problem in a convex fashion describing the trajectory of the CoM c using such curves and how we impose physical constraints on it. For an analytical description on Bézier curves, we refer the reader to Appendix A.

Bézier matrices representation

One can represent a Bézier curve using binomial coefficients as seen in Appendix A. However, this representation is equivalent to a description using matrices. We opt for the latter since it is easier to manipulate and it will become useful further on, when a decomposition of a Bézier curve into Bézier sub-curves is needed. A generic Bézier curve expression can be expressed in term of matrices. This notation is a

matrix notation that helps to compute a Bézier curve in a more systematic way. It will also be used when dealing with contact switches in the next chapters.

The expression of a generic Bézier curve (A.5) can be arranged in such a way that the matrix form becomes more evident. Considering a 5-th order Bézier curve to preserve readability, the starting expression is:

$$\begin{aligned}\mathbf{B}(u) =&\binom{5}{0}(1-u)^5\boldsymbol{\rho}_0 + \binom{5}{1}u(1-u)^4\boldsymbol{\rho}_1 + \binom{5}{2}u^2(1-u)^3\boldsymbol{\rho}_2 + \\ &+ \binom{5}{3}u^3(1-u)^2\boldsymbol{\rho}_3 + \binom{5}{1}u^4(1-u)\boldsymbol{\rho}_4 + \binom{5}{5}u^5\boldsymbol{\rho}_5\end{aligned} \tag{5.6}$$

The equation can be expanded by computing the binomial coefficients and rearranging it in the following way:

$$\begin{aligned}\mathbf{B}(u) =&(-u^5 + 5u^4 - 10u^3 + 10u^2 - 5u + 1)\boldsymbol{\rho}_0 + \\ &+(5u^5 - 20u^4 + 30u^3 - 20u^2 + u + 0)\boldsymbol{\rho}_1 + \\ &+(-10u^5 + 30u^4 - 30u^3 + 10u^2 + 0 + 0)\boldsymbol{\rho}_2 + \\ &+(10u^5 - 20u^4 + 10u^3 + 0 + 0 + 0)\boldsymbol{\rho}_3 + \\ &+(-5u^5 + 5u^4 + 0 + 0 + 0 + 0)\boldsymbol{\rho}_4 + \\ &+(u^5 + 0 + 0 + 0 + 0 + 0)\boldsymbol{\rho}_5\end{aligned} \tag{5.7}$$

Ignoring temporarily the control points $\boldsymbol{\rho}_i$, the expression can be written as the matrix multiplication:

$$\mathbf{B}(u) = [u^5\ u^4\ u^3\ u^2\ u\ 1]\begin{bmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.8}$$

Which can be rewritten as:

$$\mathbf{B}(u) = [1\ u\ u^2\ u^3\ u^4\ u^5] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix} \tag{5.9}$$

Adding back the control points we obtain the full form:

$$\mathbf{B}(u) = \underbrace{[1\ u\ u^2\ u^3\ u^4\ u^5]}_{\mathcal{T} \in \mathbb{R}^{1\times 6}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix}}_{\mathcal{M} \in \mathbb{R}^{6\times 6}} \underbrace{\begin{bmatrix} \boldsymbol{\rho}_0 \\ \boldsymbol{\rho}_1 \\ \boldsymbol{\rho}_2 \\ \boldsymbol{\rho}_3 \\ \boldsymbol{\rho}_4 \\ \boldsymbol{\rho}_5 \end{bmatrix}}_{\mathcal{P} \in \mathbb{R}^{6\times 3}} \tag{5.10}$$

Then a general Bézier curve of nth order in the mth dimensional space have the following form:

$$\mathbf{B}(u) = \mathcal{T}\mathcal{M}\mathcal{P} \tag{5.11}$$

Where $\mathcal{T} \in \mathbb{R}^{1\times(n+1)}$, $\mathcal{M} \in \mathbb{R}^{(n+1)\times(n+1)}$, and $\mathcal{P} \in \mathbb{R}^{(n+1)\times m}$.

The computation of the control points ($\boldsymbol{\rho}_i$) can be reduced to a Cauchy problem. If we consider the curve $\mathbf{B}(u)$ as the CoM position ($\mathbf{c}(t)$) we can obtain the velocity and the acceleration curves ($\dot{\mathbf{c}}(t)$ and $\ddot{\mathbf{c}}(t)$) by analytical differentiation ((A.11) and (A.13)). To compute the control points, we have to impose the initial and the final CoM states to the three curves (position, velocity and acceleration). The initial states are defined by the current robot states, while the final states are the desired states. Here we consider the binomial form, but the derivation using the matrix form is equivalent.

Initial	Final
$\boldsymbol{\rho}_0 = \mathbf{x}_{p,0}$	$\boldsymbol{\rho}_5 = \mathbf{x}_{p,f}$
$\boldsymbol{\rho}_1 = \frac{\mathbf{x}_{v,0} \cdot T}{n} + \boldsymbol{\rho}_0$	$\boldsymbol{\rho}_4 = \boldsymbol{\rho}_7 - \frac{\mathbf{x}_{v,f} \cdot T}{n}$
$\boldsymbol{\rho}_2 = \frac{\mathbf{x}_{a,0} \cdot T^2}{n(n-1)} + 2\boldsymbol{\rho}_1 - \boldsymbol{\rho}_0$	$\boldsymbol{\rho}_3 = \frac{\mathbf{x}_{a,f} \cdot T^2}{n(n-1)} + 2\boldsymbol{\rho}_6 - \boldsymbol{\rho}_7$

Table 5.2: Computation of the curve's control points.

Considering a 5^{th} order curve for the CoM position:

$$\mathbf{c}\left(\frac{t}{T}\right) = \binom{5}{0}\left(1-\frac{t}{T}\right)^5 \boldsymbol{\rho}_0 + \binom{5}{1}\frac{t}{T}\left(1-\frac{t}{T}\right)^4 \boldsymbol{\rho}_1 + \ldots + \binom{5}{5}\left(\frac{t}{T}\right)^5 \boldsymbol{\rho}_5 \quad (5.12)$$

$$\dot{\mathbf{c}}\left(\frac{t}{T}\right) = \binom{4}{0}\left(1-\frac{t}{T}\right)^4 \hat{\boldsymbol{\rho}}_0 + \binom{4}{1}\frac{t}{T}\left(1-\frac{t}{T}\right)^3 \hat{\boldsymbol{\rho}}_1 + \ldots + \binom{4}{4}\left(\frac{t}{T}\right)^4 \hat{\boldsymbol{\rho}}_4 \quad (5.13)$$

$$\ddot{\mathbf{c}}\left(\frac{t}{T}\right) = \binom{3}{0}\left(1-\frac{t}{T}\right)^3 \tilde{\boldsymbol{\rho}}_0 + \binom{3}{1}\frac{t}{T}\left(1-\frac{t}{T}\right)^2 \tilde{\boldsymbol{\rho}}_1 + \ldots + \binom{3}{3}\left(\frac{t}{T}\right)^3 \tilde{\boldsymbol{\rho}}_3 \quad (5.14)$$

Where the variable $\left(\frac{t}{T}\right)$ is considered to stress the importance of having a normalized time. T is the duration of the time horizon considered. The control points of the k^{th} order derivative of a Bézier curve depend on the control points of the $k^{th}-1$ order derivative through the relationship:

$$\boldsymbol{\rho}_i^k = (n-k) \cdot (\boldsymbol{\rho}_{i+1}^{k-1} - \boldsymbol{\rho}_i^{k-1}) \quad (5.15)$$

Where n is the order of the $k^{th}-1$ order derivative curve. By evaluating the curves in the points $t=0$ and $t=T$ and exploiting the control points relationship (5.15) we can impose the initial and the final values to the control points of the original curve (Table 5.2).

Free parameter of a Bézier curve as a control point

A Bézier curve is fully defined when all its control points have been computed. Once that we have fully defined the curve along the contact switch horizon[1], we have a complete description of the trajectory. However, this means that the solution of the

[1]See Section 4.2

feasibility problem described in (5.5) can only be solved if this exact trajectory can be achieved by the robot for each specific foothold. In other words, the problem would over constrained and the solution space for the problem would be heavily reduced. To increase the solution space, we can add a degree of freedom to the curve, a free parameter. In the case of a Bézier curve, an extra control point can be used as a free parameter. We understand that this trajectory does not include the totality of the solution space, however we deem this to be sufficiently big to avoid infeasibility in the problem, while still keeping the problem formulation convex.

To include the new control point in the curve, we need a higher order curve. Adding only one control point, the needed curve is a Bézier curve of the 8^{th} order. Changing order of the curve means that a new curve should be used, including the need of computing again all the control points. But since the computation of the control point is performed in the same way for any Bézier curve of any order, we can just "insert" the new control point in the middle of the curve, keeping the control points computed for our previously fully defined curve. We can insert the new control point in the following way:

$$\begin{bmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \boldsymbol{\rho}_3 & \boldsymbol{\rho}_\epsilon & \boldsymbol{\rho}_5 & \boldsymbol{\rho}_6 & \boldsymbol{\rho}_7 \end{bmatrix}$$
$$\uparrow \qquad \downarrow$$
$$\tilde{\boldsymbol{\rho}} \qquad \boldsymbol{\rho}_i \rightarrow \boldsymbol{\rho}_{i+1}$$

The new set of control points is:

$$\begin{bmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \boldsymbol{\rho}_3 & \boldsymbol{\rho}_\epsilon & \boldsymbol{\rho}_5 & \boldsymbol{\rho}_6 & \boldsymbol{\rho}_7 & \boldsymbol{\rho}_8 \end{bmatrix} \tag{5.16}$$

Where the unknown control point is $\boldsymbol{\rho}_\epsilon$. Being its coordinates unknown it can be considered as a Degree of Freedom (DoF) of the curve.

5.2.2 Convex problem formulation

The problem formulated might seem not convex at first glance because of (5.2). In fact, the cross product between two decision variables (5.2) is in general a sign of non-convexity. But exploiting the Bézier curve expressions, we can prove that the cross product expressed as a Bézier curve with one free control point, yields a linear relation with respect to said control point.

For the sake of clarity, we report here the cross product that we are going to analyse:

$$m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) = m\mathbf{c} \times \ddot{\mathbf{c}} - m\mathbf{c} \times \mathbf{g} \tag{5.17}$$

To check for convexity, we have to verify that the product $\mathbf{c} \times \ddot{\mathbf{c}}$ is a convex quantity, as it is a cross product between a decision variable and its second derivative.

Consider an 8^{th} order Bézier curve with a free control point (unknown variable). Here we use the binomial representation of a Bézier curve, since it is equivalent with the matrix notation. As shown in equation (A.6) from Appendix A, we can write the full equation describing the trajectory of the CoM as:

$$\mathbf{c}(u) = \binom{8}{0}(1-u)^8\boldsymbol{\rho}_0 + \ldots + \binom{8}{8}(u)^8\boldsymbol{\rho}_8 \tag{5.18}$$

We know that all the control points $\boldsymbol{\rho}_i$ are known terms with the exception of $\boldsymbol{\rho}_\epsilon$. Then we rearrange the equation in such a way to have all the known terms grouped together:

$$\mathbf{c}(u) = \underbrace{\binom{8}{0}(1-u)^8\boldsymbol{\rho}_0 + \ldots + \binom{8}{8}(u)^8\boldsymbol{\rho}_8}_{\boldsymbol{\Lambda}\in\mathbb{R}^3} + \underbrace{\binom{8}{4}u^4(1-u)^4}_{\zeta\in\mathbb{R}}\boldsymbol{\rho}_\epsilon \tag{5.19}$$

Where $\boldsymbol{\Lambda}(\boldsymbol{\rho}, u)$ is the known term and depends only on the known control points, while $\zeta(u)$ is the term that multiplies the unknown variable $\boldsymbol{\rho}_\epsilon$. The Bézier expression can now be written as:

$$\mathbf{c}(u) = \boldsymbol{\Lambda}(\boldsymbol{\rho}, u) + \zeta(u)\boldsymbol{\rho}_\epsilon \tag{5.20}$$

To prove that the cross product analysed is a convex quantity, we need to express the second derivative $\ddot{\mathbf{c}}$ in terms of the control points of the non differentiated curve.

A generic k^{th} order derivative of a Bézier curve can be obtained as:

$$\mathbf{B}^k(u) = \sum_{i=0}^{n-k} b_{n-k,i}(u)\boldsymbol{\rho}_i^k \tag{5.21}$$

$$\boldsymbol{\rho}_i^k = (n-k)\cdot(\boldsymbol{\rho}_{i+1}^{k-1} - \boldsymbol{\rho}_i^{k-1}) \tag{5.22}$$

Since the k^{th} derivative of a bezier curve is a Bézier curve of the $(n-k)^{th}$ order[2], where n is the order of the non differentiated curve, we have a $8-2=6^{th}$ order curve for the second derivative ($k=2$). Then we can expand the expression of the second derivative control points (A.15) to relate them to the non differentiated curve's control points:

$$\tilde{\boldsymbol{\rho}}_i = (n-1)(\hat{\boldsymbol{\rho}}_{i+1} - \hat{\boldsymbol{\rho}}_i) = n(n-1)(\boldsymbol{\rho}_{i+2} - 2\boldsymbol{\rho}_{i+1} + \boldsymbol{\rho}_i)) \tag{5.23}$$

We rearrange the equation such that the known and unknown terms are clearly separated, similar to (5.19):

$$\ddot{\mathbf{c}}(u) = \tilde{\boldsymbol{\Lambda}}(\boldsymbol{\rho}, u) + \tilde{\zeta}(u)\boldsymbol{\rho}_\epsilon \tag{5.24}$$

$\tilde{\boldsymbol{\Lambda}} \in \mathbb{R}^3$ and $\tilde{\zeta} \in \mathbb{R}$ have the same meaning as in (5.20) but for the second derivative.

Using the compact forms, the cross product between $\mathbf{c}(t)$ and $\ddot{\mathbf{c}}(t)$ can be written as:

$$\mathbf{c} \times \ddot{\mathbf{c}} = (\boldsymbol{\Lambda} + \zeta\boldsymbol{\rho}_\epsilon) \times (\tilde{\boldsymbol{\Lambda}} + \tilde{\zeta}\boldsymbol{\rho}_\epsilon) \tag{5.25}$$

$$\mathbf{c} \times \ddot{\mathbf{c}} = \boldsymbol{\Lambda} \times \tilde{\boldsymbol{\Lambda}} + \boldsymbol{\Lambda} \times \tilde{\zeta}\boldsymbol{\rho}_\epsilon + \zeta\boldsymbol{\rho}_\epsilon \times \tilde{\boldsymbol{\Lambda}} + \underbrace{\zeta\boldsymbol{\rho}_\epsilon \times \tilde{\zeta}\boldsymbol{\rho}_\epsilon}_{0} \tag{5.26}$$

$\boldsymbol{\rho}_\epsilon$ is an unknown vector and the source of non convexity can be cancelled thanks to the property of the cross product, where the cross product between two parallel vectors is always 0. Since the cross product is performed on the very same vector, they are parallel and it can be cancelled making the problem *convex*.

5.3 Problem feasibility

5.3.1 Constraint polytopes

A Bézier curve can be seen as the analytical representation of a physical quantity in 3D space. To preserve physical consistency of the analytical curve, it is subject to constraints that represent the dynamic model of the floating-base system. The constraints can be related to different physical elements, such as dynamical behaviour, or friction limits. To verify if a trajectory is physically consistent, it has

[2]See Equation (A.15)

to satisfy the constraints that describe the dynamics of the system (5.1) and (5.2).

The visualization of such constraints applied to the curve is not an easy task since they are mostly related to physical concepts rather than spatial quantities. The constraints considered can be represented by half-spaces *half-spaces* (closed regions of space). A half-space is defined as one of the two parts into which a hyperplane (a plane in a generic m-space) divides and affine space (m-space). By intersecting multiple half-spaces we can obtain a closed portion of space. This portion of space is referred as a *polytope*. The inside of the polytope is where all the constraints are valid at the same time, therefore it represents the feasible space.

Definition. *A polytope is a geometric object with flat sides. An n-polytope is associated to an n dimension. As examples, in 2-dimensional space they are called 2-polytopes and in 3-dimensional space 3-polytopes.*

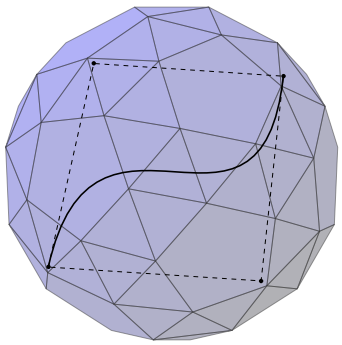


Figure 5.5: Example of constraints polytope applied to a curve.

Considering the n-dimensional generic space where the Bézier curve is defined, the constraints can be represented as an n-polytope.

The polytope can be considered as a way of visualizing constraints (Fig. 5.5), since the trajectory is bounded by the constraints to lie within an n-dimensional polytope. They make a distinction between the physical possible solutions (inside the polytope) and the physically not consistent solutions (outside the polytope).

5.3.2 Discrete and continuous time domain evaluations

All the considerations reported so far are formulated in the continuous time domain. But to actually solve the problem, we need to discretize the problem. The discretization can be achieved through time sampling, i.e., dividing a time interval in a finite amount of smaller intervals. The time between two subsequent

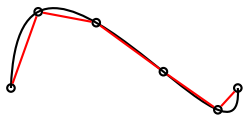


Figure 5.6: Example of continuous time (black) and interpolated discrete time (red) Bézier curve.

time intervals is called sampling time (Δt). The use of this description does not accurately describe the real world because the information between two subsequent discrete time instants is not defined. They can be obtained through interpolation, but in general they differ from the real behaviour (Fig. 5.6). With a smaller Δt better results can be achieved, but it still differs from the real case. It is important to remember that an optimization problem is built using variables, which can be either a value or an array of values. This means that a variable can represent only a single time instant. To consider multiple time time instants, we need to associate a variable to each instant in our time horizon. Increasing the amount of time instants considered means having an increasing number of variables to consider, leading to longer computation times.

In Fig. 5.7 a constraints polytope is applied to a curve computed in discrete and in continuous time. By computing the curve in discrete time, its feasibility is verified as it lies completely inside the 2-polytope. But when checking the feasibility of the real curve, i.e., the one computed in continuous time, it becomes clear that there is the chance of having an unfeasible solution (it lies also outside the boundaries). This means that we may find a non dynamically feasible solution when computing the curve in discrete time, leading to undesired behaviours in the real case as it is described by the continuous curve.

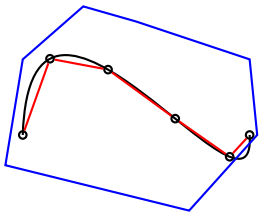


Figure 5.7: Example of constraint polytope (blue) applied to a continuous (black) and a discrete (red) description of the same curve. The feasibility check is performed at equally spaced points on the curve.

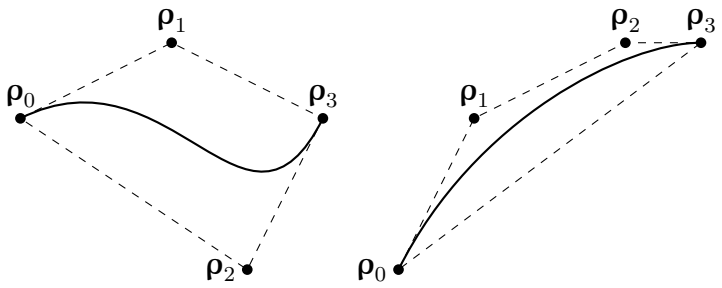


Figure 5.8: 2D examples of Bézier curve's convex hull property

To formulate the problem in such a way that it is also valid in continuous time, we can exploit a property of the Bézier curves to make sure that the whole curve is inside the polytope even with a discrete time formulation.

Property. *A generic Bézier curve of any order is contained entirely in the convex hull of its control points.*

$$\mathbf{B}(t) \subset conv(\boldsymbol{\rho}) \tag{5.27}$$

This property can be exploited to validate the formulation also for continuous time. The validation for continuous time is important since it allows to achieve physical consistency. Validating the results for the continuous time is not always possible given the discrete nature of the majority of the computational methods used. The method here described is taken from [40], where it is actually used to solve the discretization problem.

In Section 5.2.1 the need for a free control point ($\boldsymbol{\rho}_\epsilon$) has been described. Considering the CoM trajectory (Fig. 5.4a), leaving the free control point undefined leads to have an infinite amount of suitable curves that can connect two points in space. While all of these curves can analytically establish a connection in space, most of them are not physically achievable. The objective is then to place correctly $\boldsymbol{\rho}_\epsilon$ such that the Bézier curve that it describes remains physically consistent with respect to the model described by (5.1) and (5.2).

A way to guarantee that the feasibility evaluated in discrete time match an actual feasibility in continuous time is by exploiting the convex hull property (5.9). We can enforce the control points that describe the Bézier curve to lie inside the

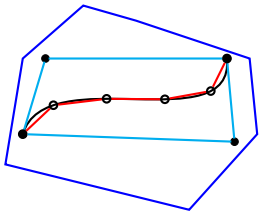


Figure 5.9: Example of constraint polytope (blue) applied to a continuous (black) and a discrete (red) description of the same curve. The check is performed on the control points. The convex hull of the control points (cyan) contains entirely the curve itself.

constraint polytope.

5.4 Trajectory discontinuities

Legged locomotion can be described as a *hybrid system*. A hybrid system is a system that exhibits both continuous and discrete dynamics. Locomotion can be decomposed in a sequence of alternating swing and stance phases (Fig. 4.5). The phases are divided by discrete events in time, i.e., when a leg makes or break contact with the ground (contact switch). Each phase's dynamics can be described using (5.1) and (5.2) considering the contact points relative to that specific phase (Fig. 5.10). At each contact switch the contact points change, increasing (swing → stance) or decreasing (stance → swing) and creating discontinuities.

In Figure 5.10 we can see how the contact points are not continuous in time,

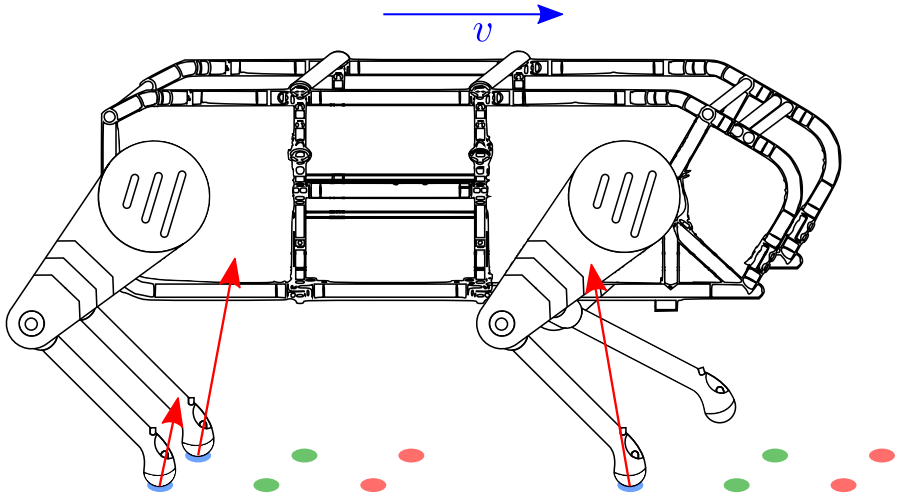


Figure 5.10: Future step positions that the robot will take. Here it is taking a step forward with its left-front leg (LH).

since they are defined only when the feet are in contact with the ground and not in swing. The sequence of contact points will be a collection of points on the ground defined only for specific time intervals.

Since the equations of motion include ground reaction forces, i.e., the forces at the contact points, we need to take into account the discontinuities. The constraints themselves, i.e., the equations of motion (5.1, 5.2), do not embed discontinuities since they define a continuous dynamics system. They can model only a single phase. To include discrete events in the description of the system, we need to formulate the constraints according to the phase considered, according to the phase's characteristics, namely the contact locations. As in (5.3.1), the constraints can be imagined as n-polytopes in the space of the trajectory. Being the trajectory in 3-D, we can visualize a 3-polytope for each of the phases that occur during the time span considered (Fig. 5.11).

In Fig. 5.11 the polytopes describe a visualization example of the constraints that we can apply to a curve to verify its feasibility. The fact that two polytopes overlap means that both sets of constraints are valid at the same time. This means that for all the possible conditions included in the two overlapping polytopes, there exists at least a solution that satisfies both set of constraints.

Considering Fig. 5.11, we can associate the overlapping regions to the contact switches because to break or make contacts with the ground correctly the CoM trajectory should be always feasible (inside a constraints polytope). When a contact switch occurs, the set of constraints changes making the polytope to change accordingly. To make sure that the trajectory yields a dynamically feasible motion, the contact switch must occur inside the overlapping area, such that at the beginning of the next phase the trajectory is already inside the relative polytope.

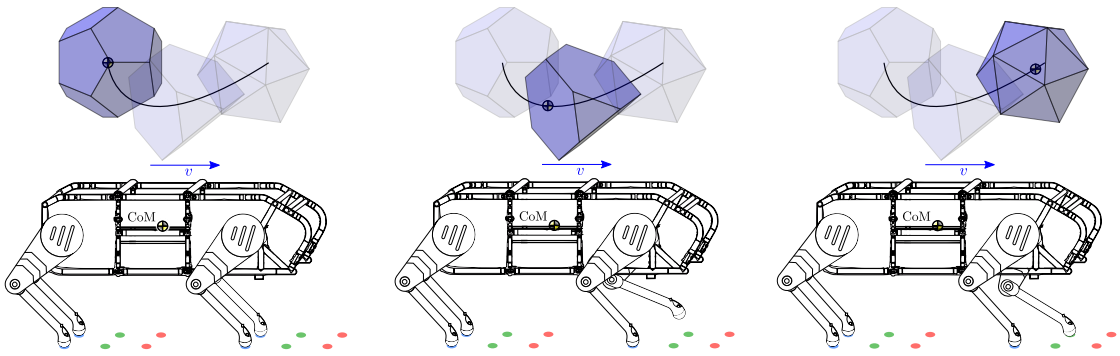


Figure 5.11: Example of constraints relative to three subsequent phases. Each 3-polytope represents a different set of constraints.

Considering a 2-dimensional case for the sake of clarity in notation and visualization, the overlapping polytopes can be applied to a Bézier curve to visualize the role of the constraints.

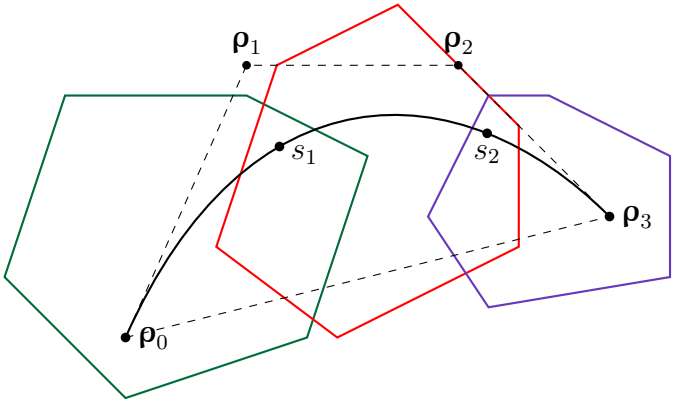


Figure 5.12: Example of overlapping constraint polytopes applied to a Bézier curve. s_1 and s_2 are the points in which contact switches occur. Inspired from Fig. 4 in [40]

In Figure 5.12, three overlapping polytopes are applied to a 3$^{\text{rd}}$ order Bézier curve. Observing the curve itself, we can state that the curve is feasible because it is always inside of at least one polytope. Although the feasibility of the curve is certain, applying the method used in Figure 5.7 the curve is marked as "infeasible" because the convex hull of its control points is not fully contained in the polytopes. This leads to a reduction of the solution space, forcing to discard solutions that are feasible.

To increase the solution space, we can consider three equivalent curves instead of considering the curve as a whole. The main curve can be cut at s_1 and s_2 resulting in three sub-curves, each representing a single phase. Each curve has its own new control points that can then be used to check if they belong to their relative constraint polytope.

The analytical derivation relies on the fact that the Bézier curves ($\mathbf{B}(u)$) are parametric curves ($u \in [0,\ 1]$), where u is the parameter of the curve, usually associated to time t. In general though, the time interval considered is defined between two generic time instants $t \in [t_i,\ t_{i+1}]$, $t_{i+1} > t_i$ which can be different from the required parametric interval $[0,\ 1]$. This means that in general $u \neq t$. In order to correctly associate time t with the parameter of the curve u, time is normalized. The time normalization process is used to rescale the time interval to $[0,\ 1]$.

Defining as *total duration* the time interval considered $T_d = t_{i+1} - t_i$, the

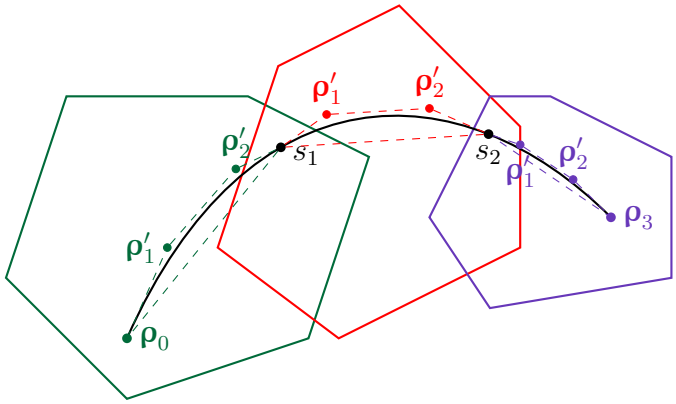


Figure 5.13: Decomposition of the Bézier curve into three equivalent sub-curves.

parameter of the curve can be associated to time as:

$$u = \frac{t - t_i}{T_d} \in [0,\ 1] \tag{5.28}$$

To reduce the notation complexity, here time is considered as already normalized, unless indicated otherwise.

Consider an arbitrary Bézier curve in a generic normalized time interval $t \in [0,\ 1]$ (Fig. 5.14). Consider two generic time instants $z,\ z_1 \in [0,\ 1]$, where $z_1 > z$:

$$t = \{t \in [z,\ z_1] |\ \ z_1 > z,\ z_1, z \in [0, 1]\} \tag{5.29}$$

We would like to decompose this curve into multiple sub-curves defined by the segments from $t = 0$ to $t = z$, from $t = z$ to $t = z_1$ and from $t = z_1$ to $t = 1$. Considering $\mathcal{T}$ from (5.11) for a 3$^{\text{rd}}$ order curve for the sake of readability, we can substitute the new time interval defined as $z + (z_1 - z)t$ to the parameter u, obtaining a generic $\mathcal{T}_i$:

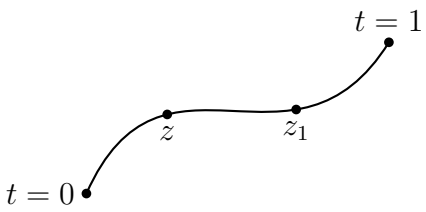


Figure 5.14: Generic 3$^{\text{rd}}$ order Bézier curve where a generic normalized time interval ($z \to z_1$) has been defined

$$\mathcal{T}_i = [1 \;\; (z + (z_1 - z)t) \;\; (z + (z_1 - z)t)^2 \;\; (z + (z_1 - z)t)^3] \tag{5.30}$$

(5.30) can be written as the product between two matrices:

$$\mathcal{T}_i = \mathcal{T} \underbrace{\begin{bmatrix} 1 & z & z^2 & z^3 \\ 0 & z_1 - z & z(z1 - z) & 3z^2(z_1 - z) \\ 0 & 0 & (z1 - z)^2 & 3z(z_1 - z)^2 \\ 0 & 0 & 0 & (z_1 - z)^3 \end{bmatrix}}_{\mathcal{Z} \in \mathbb{R}^{4\times 4}} \tag{5.31}$$

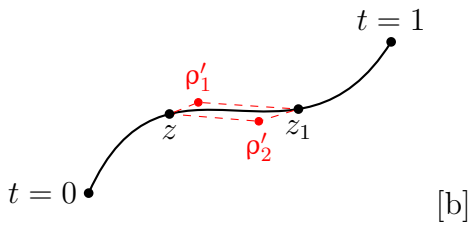
Recalling (5.11), we can write the expression for a generic equivalent sub-curve (piece of curve defined by z and z_1) as:

$$\mathbf{B}_i(t) = \mathcal{T}_i \mathcal{M} \mathcal{P} = \mathcal{T} \mathcal{Z} \mathcal{M} \mathcal{P} \tag{5.32}$$

The sub-curve generated depends on the control points of the original curve $\mathcal{P}$. This sub-curve is still a Bézier curve and can be described with its own set of control points. To find the set of control points that describe this piece of curve, we can rearrange the equation as:

$$\mathbf{B}_i(t) = \mathcal{T} \mathcal{M} \underbrace{\mathcal{M}^{-1} \mathcal{Z} \mathcal{M}}_{\mathcal{Q} \in \mathbb{R}^{4\times 4}} \mathcal{P} = \mathcal{T} \mathcal{M} \underbrace{\mathcal{Q} \mathcal{P}}_{\mathcal{P}_i \in \mathbb{R}^{4\times 3}} = \mathcal{T} \mathcal{M} \mathcal{P}_i \tag{5.33}$$

Since $\mathcal{M}$ is a square matrix, we can always multiply by $\mathcal{M}\mathcal{M}^{-1} = \mathbf{I}$.



[b]

Figure 5.15: Generic equivalent sub-curve's convex hull of its control points (red).

The resulting expression for the generic equivalent sub-curve (5.33) allows to compute the control points of the sub-curve itself. This new set of control points can be used to compute the convex hull to test the feasibility of the curve (5.13). Note that this is an alternative way of decomposing curves with respect to the one

used in [40], namely the De Casteljau algorithm.

This method can be easily generalized for a generic $n-th$ order curve. Consider a generic matrix $\{\mathcal{Z}_\infty | \mathcal{Z}_\infty \in \mathbb{R}^{n\times n}, n \to \infty\}$ built as shown in (5.31). A generic Bézier curve $B_k(u)$ of order k can be divided into multiple sub-curves using a matrix $\mathcal{Z}_k \in \mathbb{R}^{(k+1)\times(k+1)}$ which is a *sub-matrix* of $\mathcal{Z}_\infty$, obtained as:

$$\mathcal{Z}_\infty = \begin{bmatrix} & \overbrace{\quad}^{\mathcal{Z}_2} & & \overbrace{\quad}^{\mathcal{Z}_3} & \\ 1 & z & z^2 & z^3 & \cdots \\ 0 & z_1 - z & 2z(z_1 - z) & 3z^2(z_1 - z) & \cdots \\ 0 & 0 & (z_1 - z)^2 & 3z(z_1 - z)^2 & \cdots \\ 0 & 0 & 0 & (z_1 - z)^3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.34}$$

5.5 Angular momentum rate

Now we will focus on the third nonlinearity term in the feasibility problem (5.5a), which is due to the $\dot{\mathbf{L}}$ term, related to the angular behaviour of the robot (orientation, angular velocity and angular acceleration). We would like to express it in terms of angular quantities because they bear a more intuitive meaning in terms of locomotion. To show that the $\dot{\mathbf{L}}$ is a nonlinear function of angular quantities, we can derive it analytically. It can be obtained by differentiating the angular momentum expression $\mathbf{L} = \mathbf{I}\omega$:

$$\dot{\mathbf{L}} = \dot{\mathbf{I}}\boldsymbol{\omega} + \mathbf{I}\dot{\boldsymbol{\omega}} \tag{5.35}$$

$$\mathbf{I}_\mathcal{W} = {}^\mathcal{W}\mathbf{R}_\mathcal{B}\, \mathbf{I}\, {}^\mathcal{W}\mathbf{R}_\mathcal{B}^\top \tag{5.36}$$

Where ${}^\mathcal{W}\mathbf{R}_\mathcal{B}^\top(\boldsymbol{\Theta}) \in \mathbb{R}^{3\times3}$ is the rotation matrix between base $\mathcal{B}$ and world $\mathcal{W}$ reference frame, $\mathbf{I}_\mathcal{W}(\boldsymbol{\Theta})$ is the moment of inertia of the robot expressed in the world frame. Since the derivative of a generic rotating vector $\boldsymbol{A}$ is:

$$\frac{d\mathbf{A}}{dt} = \frac{d_r\mathbf{A}}{dt} + \boldsymbol{\omega} \times \mathbf{A} \tag{5.37}$$

Where $\dfrac{d_r\mathbf{A}}{dt}$ is the apparent time derivative of $\mathbf{A}$ in the rotating frame, while $\boldsymbol{\omega}$ is the angular speed of vector $\mathbf{A}$ with respect to fixed reference frame.

In this case, $\mathbf{I}$ is assumed fixed with respect to base frame $\mathcal{B}$, then $\frac{d_r \mathbf{I}}{dt} = 0$ and then:

$$\dot{\mathbf{I}} = \boldsymbol{\omega} \times \mathbf{I} \tag{5.38}$$

$\dot{\boldsymbol{\omega}}$ expression can be obtained by differentiation of (4.10):

$$\dot{\boldsymbol{\omega}} = \dot{\mathbf{T}}(\gamma, \psi, \dot{\gamma}, \dot{\psi})\dot{\boldsymbol{\Theta}} + \mathbf{T}(\gamma, \psi)\ddot{\boldsymbol{\Theta}} \tag{5.39}$$

Then, combining (5.35) with (5.38), (5.36) and (5.39):

$$\dot{\mathbf{L}} = \mathbf{T}(\gamma, \psi)\dot{\boldsymbol{\Theta}} \times \mathbf{I}_{\mathcal{W}}(\boldsymbol{\Theta})\mathbf{T}(\gamma, \psi)\dot{\boldsymbol{\Theta}} + \mathbf{I}_{\mathcal{W}}(\boldsymbol{\Theta}) \cdot (\dot{\mathbf{T}}(\gamma, \psi, \dot{\gamma}, \dot{\psi})\dot{\boldsymbol{\Theta}} + \mathbf{T}(\gamma, \psi)\ddot{\boldsymbol{\Theta}}) \tag{5.40}$$

Which is a highly nonlinear term. Although this terms could also be included in the formulation using a nonlinear solver, in this thesis we decide to avoid any nonlinear term as solving a nonlinear problem takes more time and may lead to getting stuck in local minima. We then decided to keep the formulation convex by avoiding any nonlinear term.

In the next chapter, we will deal with this nonlinearity, initially following the approach of [40], where the derivative of the angular momentum is set to zero (i.e., $\dot{\mathbf{L}} = 0$). However, as it will be shown later, this condition does not hold for highly dynamic motions such as trot, due to the underactuated condition of the system when less than three legs are in contact with the ground. Keeping this condition leads to infeasiblity when solving the problem described in this chapter (5.5a). We will remove this condition and examine cases where $\dot{\mathbf{L}} \neq 0$, showing a way to constrain and bound the angular momentum rate considering the actual limits of the robot.

Chapter 6

Convex Formulation of the Transition Feasibility Problem for Non-zero Angular Momentum Rate

In the previous chapter we have seen how describing the system with a RBDM and choosing the function to describe the CoM trajectory as a Bézier curve, we can eliminate most of the nonlinearities that make the problem not convex. However, if we also try to specify a trajectory for the angular quantities through the derivative of the angular momentum rate, the constraint related to $\dot{\mathbf{L}}$ becomes nonlinear. To avoid having such nonlinear term, a possible solution is to consider $\dot{\mathbf{L}} = 0$, as done in [40]. This choice has two potential drawbacks:

1. It might result in infeasible solutions when the initial conditions are not suitable to reach 0;
2. It is a very hard-to-fulfil constraint for more dynamic motions.

In the next sections we will show two different approaches that tackle the nonlinearity introduced by the angular momentum rate. We will see how assuming $\dot{\mathbf{L}} = 0$ is a reasonable choice when dealing with quasi-static motions (such as crawl) but it becomes a source of infeasibility when more dynamic motions are considered (such as trot). During a crawl the robot has the ability to track both linear and

angular quantity, in trot it is not possible due to the lower number of legs in contact in stance during a swing phase (*underactuated* system).

6.1 Angular momentum rate error minimization

We now consider the condition $\dot{\mathbf{L}} = 0$ along the entire time horizon computed. Instead of imposing it by not considering it in the optimization problem, we show how it is possible to include it keeping the problem convex, trying to reduce the error with respect to a desired behaviour using a tracking term in the cost function.

The analytical expression of $\dot{\mathbf{L}}$ (5.40) is nonlinear since it contains cross products between optimization variables and trigonometric functions of optimization variables (in matrix $T(\boldsymbol{\theta})$). To ensure convexity while including $\dot{\mathbf{L}}$ in the optimization, we define a reference value for the angular momentum rate $\dot{\mathbf{L}}_{ref}$ which can be computed using (5.40). Along with the definition of the reference, we include in the optimization problem $\dot{\mathbf{L}}$ as an optimization variable. Then the assumption $\dot{\mathbf{L}} = 0$ is replaced by the tracking term $\| \dot{\mathbf{L}} - \dot{\mathbf{L}}_{ref} \|_2^2$ and $\dot{\mathbf{L}}_{ref} = 0$ which penalizes the deviation from the reference angular momentum rate.

According to (5.40), to have $\dot{\mathbf{L}} = 0$ the angular velocities and accelerations should be constantly 0 along the entire trajectory, which might not be always possible. Tracking $\dot{\mathbf{L}}_{ref}$ built considering initial and desired conditions, we can consider all those cases with non-zero initial or desired angular velocity and acceleration.

The optimization problem that we want to solve has now a cost function to consider the deviation from the requested $\dot{\mathbf{L}}_{ref}$:

$$\min_{\boldsymbol{\rho}_4,\, \mathbf{f_e}(t),\, \dot{\mathbf{L}}} \quad \| \dot{\mathbf{L}} - \dot{\mathbf{L}}_{ref} \|_2^2 \tag{6.1a}$$

$$\text{such that} \quad m(\ddot{\mathbf{c}}(t) - \mathbf{g}) = \sum_{i=1}^{n} \mathbf{f}_{e,\,i}(t) \tag{6.1b}$$

$$m\mathbf{c}(t) \times (\ddot{\mathbf{c}}(t) - \mathbf{g}) + \dot{\mathbf{L}} = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{6.1c}$$

$$\mathbf{f}_e \leq \mathbf{f}_{e,\,max} \tag{6.1d}$$

$$\| f_x \| \leq \mu f_z \tag{6.1e}$$

$$\| f_y \| \leq \mu f_z \tag{6.1f}$$

$$f_z \geq 0 \tag{6.1g}$$

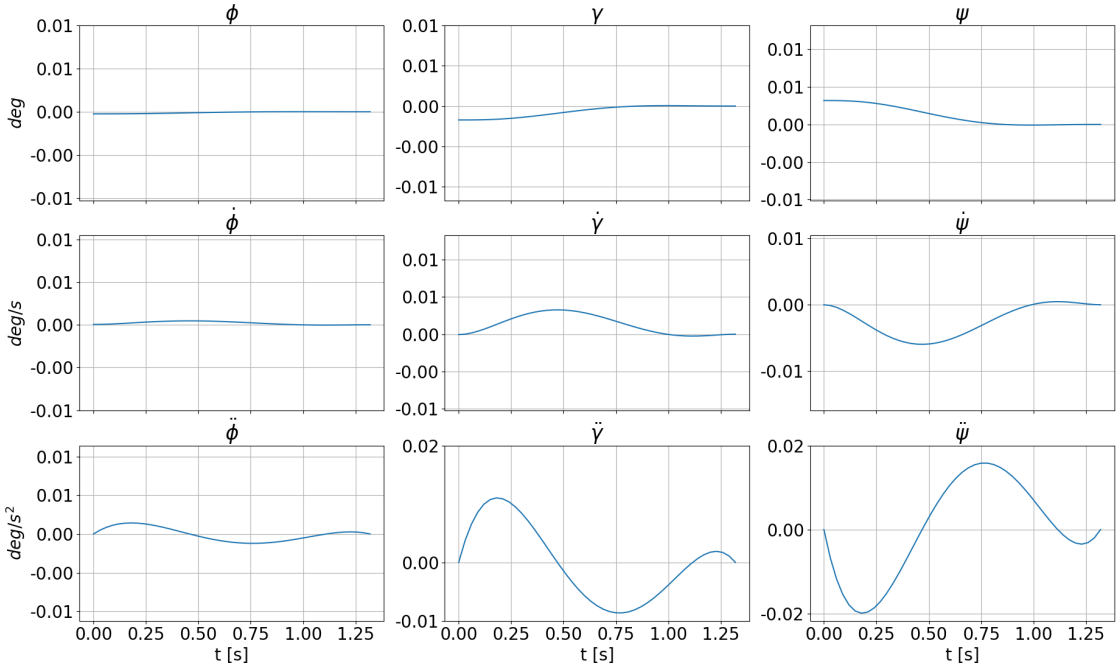


Figure 6.1: Example of ideal angular quantities computed as a Bézier curve.

It is worth noticing that $\dot{\mathbf{L}}$ does not depend any more on angular quantities as it is now defined as an optimization variable. The reference angular momentum rate reference $\dot{\mathbf{L}}_{ref}$ is computed using equation (5.40). But it requires the evolution in time of the angular quantities, namely $\boldsymbol{\Theta}$, $\dot{\boldsymbol{\Theta}}$ and $\ddot{\boldsymbol{\Theta}}$.

We design a desired orientation by solving a small optimization problem. We define the orientation curve as a Bézier curve of the 6^{th} order with a free control point. Then we impose as constraints a boundary on the roll, pitch, and yaw angles that the robot can safely reach:

$$\min_{\boldsymbol{\rho}_{4,\Theta}} \quad \frac{1}{R} \tag{6.2a}$$

$$\text{s.t.} \quad \| \boldsymbol{\Theta} \| \leq \boldsymbol{\Theta}_{max} \tag{6.2b}$$

The result of this optimization problem is the free control point $\boldsymbol{\rho}_{4,\Theta}$ which is then used to build a Bézier curve for the orientation (5.11). Although we are solving the problem minimizing the curvature, the orientation curve has to account for initial angular velocity and acceleration. A Bézier curve can be differentiated as described in Appendix A. We can then compute also the angular velocities (roll,

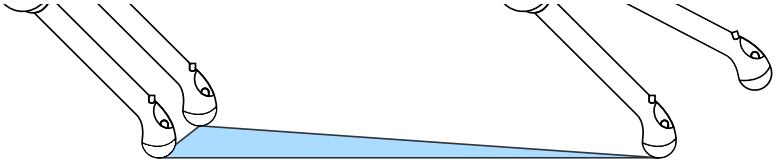


Figure 6.2: Example of support polygon during a swing phase.

pitch, yaw rates) and accelerations (Fig. 6.1) curves to have a complete angular reference with which compute $\dot{\mathbf{L}}_{ref}$.

The procedure described so far to minimize the error on $\dot{\mathbf{L}}$ imposing the reference $\dot{\mathbf{L}}_{ref} = 0$ holds for quasi-static gait types such as *crawl*. The crawl is a quadrupedal gait where only one leg is moved at a time, guaranteeing that at least three legs are in contact with the ground.

The ability to track a 0 angular momentum rate with this kind of motion can be explained by analysing the characteristics of crawl. The reason behind its stability resides in the concept called *support polygon*. The support polygon (SP) is the polygon that can be drawn considering the contact points (feet) as vertices (6.2). To consider the motion statically stable, the projection of the CoM on the SP's plane must always be contained inside the SP itself. If we consider the sequence of phases during crawl in terms of sequence of support polygons (Fig. 6.4), the

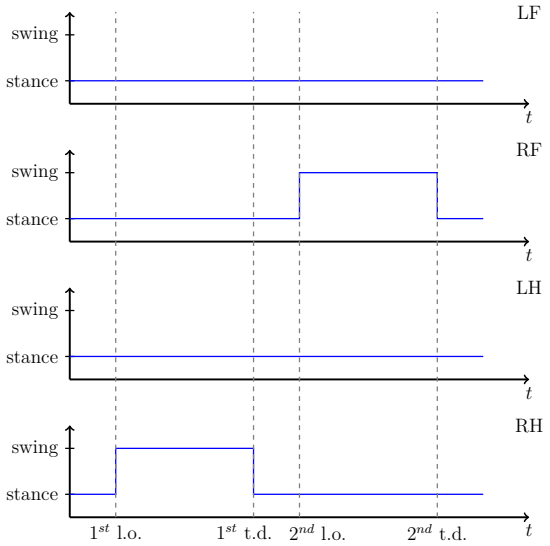


Figure 6.3: Plot that shows the time spent in air (swing) or on ground (stance) for each leg during crawl. Gray lines mark contact switches between phases. The sequence of legs moved is [RH - RF - LH - LF]. Here only half of the cycle is shown. Note that lift-offs (l.o.) and touchdowns (t.d.) happen at each start/end of each phase. The duration of the swing phase depends on the step frequency and duty factor (how much time the leg does spend in the air). It can be computed as $t_{swing} = (1 - duty\ factor)/step\ frequency$.

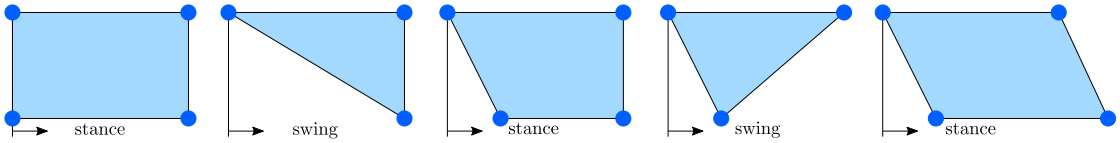


Figure 6.4: Example of a sequence of support polygons in a top-view during crawl. The arrow points toward the heading direction. The leg sequence is [RH - RF - LH - LF].

system can generate forces in any direction (unilaterality constraint still apply), provided that the CoM projection on the SP's plane is contained in the support polygon. In Fig. 6.3 we report the sequence of phases considering their time span.

6.1.1 Angular momentum rate boundaries

While the assumption of imposing $\dot{\mathbf{L}} = 0$ is fair for slow and quasi-static motions, like crawl, it does not hold any more when dealing with more dynamic motions, such as *trot* (Fig. 6.5). The trot is a type of gait where two legs (coupled in diagonal) are in air at the same time during the swing phase (Fig. 6.6 and 6.7), making the support polygon to collapse to a single line.

Having a support line instead of a support polygon during trot, the system is underactuated. The underactuated condition is a direct result of the fact that we can exert forces only along the support line, limiting the directions in which we can generate a moment. To execute a trajectory with an underactuated system, we have to sacrifice the tracking of $\dot{\mathbf{L}}$. We cannot guarantee the accurate tracking of the angular momentum rate during highly dynamic motion such as trot. But considering $\dot{\mathbf{L}}$ as an optimization variable, as we did previously, we can compute the required value to make the trajectory feasible. The problem with the current formulation is that $\dot{\mathbf{L}}$ is an unbounded quantity, being only related by the tracking term $\| \dot{\mathbf{L}} - \dot{\mathbf{L}}_{ref} \|_2^2$. The unconstrained variable could also assume values that verify the model equations (5.4) but that are not physically achievable by the robot. To avoid the divergence of $\dot{\mathbf{L}}$ and keep the angular momentum rate within the capabilities of the robot, we define an upper and a lower boundary on $\dot{\mathbf{L}}$.

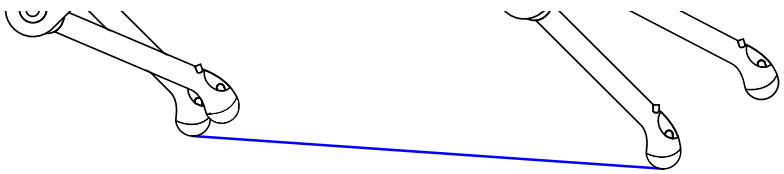


Figure 6.5: Example of support line during a swing phase in trot.

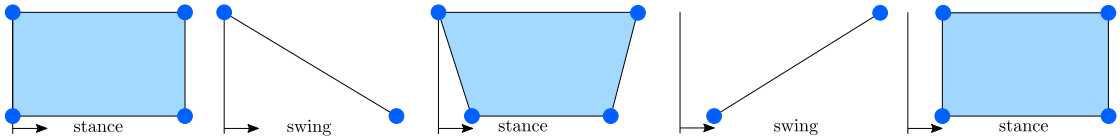


Figure 6.6: Example of a sequence of support line in a top-view during trot. The arrow points toward the heading direction. The leg sequence is [LF/RH - RF/LH].

The angular momentum rate is strictly related to the external moments acting on the system (5.2), so we compute the Maximum Moment (MM) that the robot can generate to limit the maximum achievable angular momentum rate. The maximum moment that the robot can potentially generate is heavily dependent on the foothold considered. To compute the MM boundaries, we set a new optimization problem that is able to find the best configuration of forces such that we can generate the maximum angular momentum, namely the upper boundary:

$$\max_{\mathbf{f}_e} \quad \| \mathbf{M} \|_2^2 \tag{6.3a}$$

$$\text{such that} \quad \mathbf{M} = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{6.3b}$$

$$\mathbf{J}_c^\top \mathbf{f}_e \leq \boldsymbol{\tau}_{max}{}^{1} \tag{6.3c}$$

$$\mathbf{f}_e \leq \mathbf{f}_{e,\,max} \tag{6.3d}$$

$$\| f_x \| \leq \mu f_z \tag{6.3e}$$

$$\| f_y \| \leq \mu f_z \tag{6.3f}$$

$$f_z \geq 0 \tag{6.3g}$$

Where $\mathbf{J}_c$ is the contact Jacobian which depend on robot configuration and $\boldsymbol{\tau}_{max}$ is the maximum torque available for each joint. To find the lower boundary on the angular momentum we solve the same problem but minimizing $\| \mathbf{M} \|_2^2$. These two boundaries are then used in the main optimization problem as constraints to limit

[1]Here we do not consider the term related to the Coriolis forces to keep the problem convex. This has been shown to be a fair assumption [55, 36]

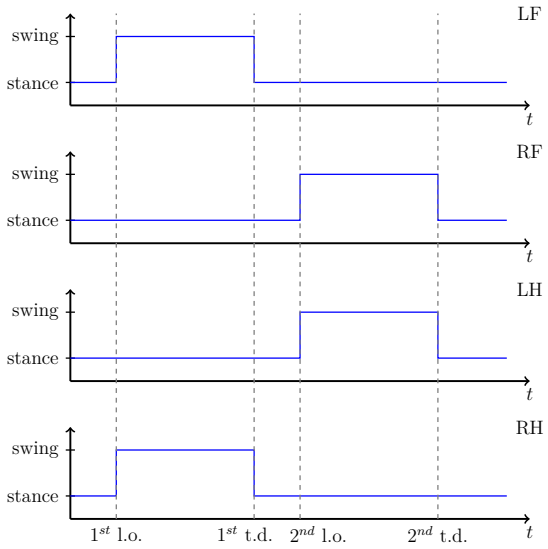


Figure 6.7: Plot that shows the time spent in air (swing) or on ground (stance) for each leg during trot. Here we are assuming that the lift-offs and the touchdowns are happening at the same time. The leg sequence is [LF/RH - RF/LH]. Note that lift-offs (l.o.) and touchdowns (t.d.) happen at each start/end of each phase. The duration of the swing phase depend on the step frequency and duty factor (how much time the leg does spend in the air). It can be computed as $t_{swing} = (1 - duty\ factor)/step\ frequency$.

the maximum value of $\dot{\mathbf{L}}$. We build it as:

$$\min_{\boldsymbol{\rho}_4,\, \mathbf{f_e}(t),\, \dot{\mathbf{L}}} \quad \| \dot{\mathbf{L}} - \dot{\mathbf{L}}_{ref} \|_2^2 \tag{6.4a}$$

$$\text{such that} \quad m(\ddot{\mathbf{c}}(t) - \mathbf{g}) = \sum_{i=1}^{n} \mathbf{f}_{e,\,i}(t) \tag{6.4b}$$

$$m\mathbf{c}(t) \times (\ddot{\mathbf{c}}(t) - \mathbf{g}) + \dot{\mathbf{L}} = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{6.4c}$$

$$\mathbf{MM}_{low} \leq \dot{\mathbf{L}} \leq \mathbf{MM}_{up} \tag{6.4d}$$

$$\mathbf{f}_e \leq \mathbf{f}_{e,\,max} \tag{6.4e}$$

$$\| f_x \| \leq \mu f_z \tag{6.4f}$$

$$\| f_y \| \leq \mu f_z \tag{6.4g}$$

$$f_z \geq 0 \tag{6.4h}$$

Although this formulation allows to include $\dot{\mathbf{L}}$ in the optimization process keeping it convex, the MM boundaries are an approximation of the real value achievable by the robot. We need to compute $\mathbf{J}_c$, which requires to know the joint configuration

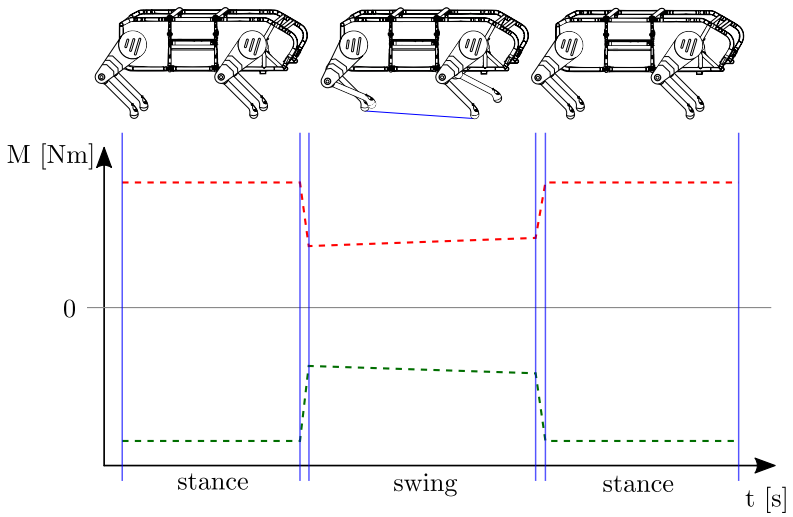


Figure 6.8: Qualitative representation of the MM boundaries associated to each phase.

and then to know the exact location of the CoM along the entire motion. Since the position is unknown, we compute the MM boundaries only at key points during the motion considered, which are at the beginning and at the end of the entire motion and right before and after a contact switch (Fig. 6.8). To know the position of the CoM at those evaluated points, we consider that the CoM follows a straight trajectory with constant velocity and 0 acceleration. This method for computing the MM boundaries makes the result subject to errors as the real trajectory is modelled as a Bézier curve. To take into account this error, we scale the maximum torque available at each joint by a confidence factor α. A confidence is a scaling factor that is based on how much confident we are about the CoM position prediction:

$$\overline{\boldsymbol{\tau}}_{max} = \alpha \boldsymbol{\tau}_{max} \tag{6.5}$$

6.2 Evaluation over an extended time horizon

If we consider a time horizon that extends indefinitely in time, problems may arise when searching for a feasible solution. As the number of phases considered in the time horizon increases, the chance of not finding a feasible solution increases as well. This happens because we are describing the trajectory as a curve with a single degree of freedom, which means having limited control over the curve's shape. A visualization of how extending the evaluation over a large time horizon can lead to

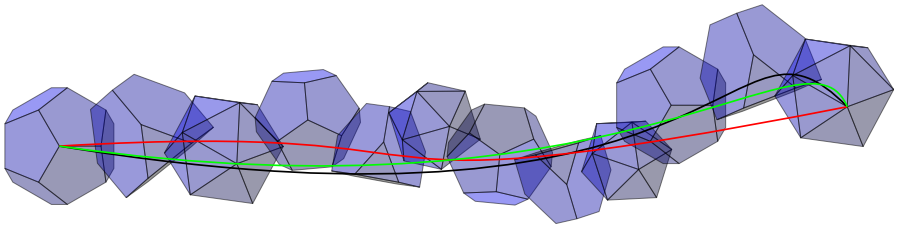


Figure 6.9: Different curves attempt to find a trajectory that is feasible, i.e., contained in all the polytopes at the same time.

infeasible solutions is depicted in Fig. 5.11. If we try to draw a Bézier curve with a single control point, we could manage to guarantee that the curve is inside part of the polytopes, but not all of them simultaneously (Fig. 6.9) as requested from the feasibility criterion defined in Section 5.4.

To generate a trajectory that is able to overcome the problem of having only one free control point including an indefinitely long time horizon, we define the optimization problem between two states (initial and final) as a sequence of multiple optimization sub-problems between a reduced number of phases, generally 2 or 3 phases. In this way it is possible to generate a trajectory composed by multiple Bézier curves such that we have only one free control point for each curve. A representation of the constraints when multiple optimizations are considered is shown in (Fig. 6.10).

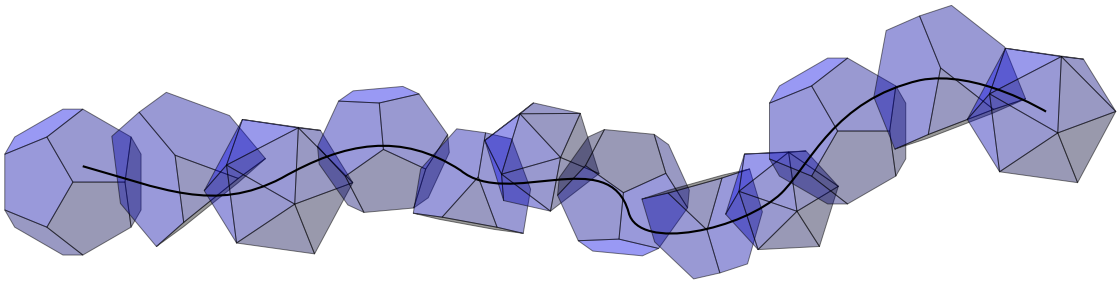


Figure 6.10: Now the curve is composed by multiple sub-curves and the time horizon can be extended indefinitely.

Chapter 7

Evaluations and Results

The main objective behind this thesis is to create an evaluation method that considers dynamics when dealing with foothold selection. This evaluation method consists in finding all those footholds that yield feasible transitions during dynamic locomotion. Among all these footholds, the evaluation has to select the optimal one according to a cost function, which will be described in the following sections. To reach the intended objective, in the last chapter we have described an evaluation method that has the purpose of finding a dynamically feasible trajectory over a finite time horizon, including transitions between phases (contact switches). This method can be implemented in foothold selection by considering each foothold as a different optimization problem with same initial and final states, but with different foothold positions.

In the following sections we will show the results obtained in simulations adopting the method proposed as a trajectory generator, to prove that the trajectories generated are physically consistent. We then show evaluations of heightmaps performed offline to prove that it can be used as a foothold selection criterion, being able to discard footholds not dynamically feasible. Moreover, we will present a simple cost function to evaluate the cost of each foothold based on the optimization problem described in Equation (6.1a).

7.1 Implementation details

In Figure 7.1 we show the pipeline that defines the control strategy used in simulation. The Dynamic Foothold Evaluation Planner (DFEP) is the algorithm

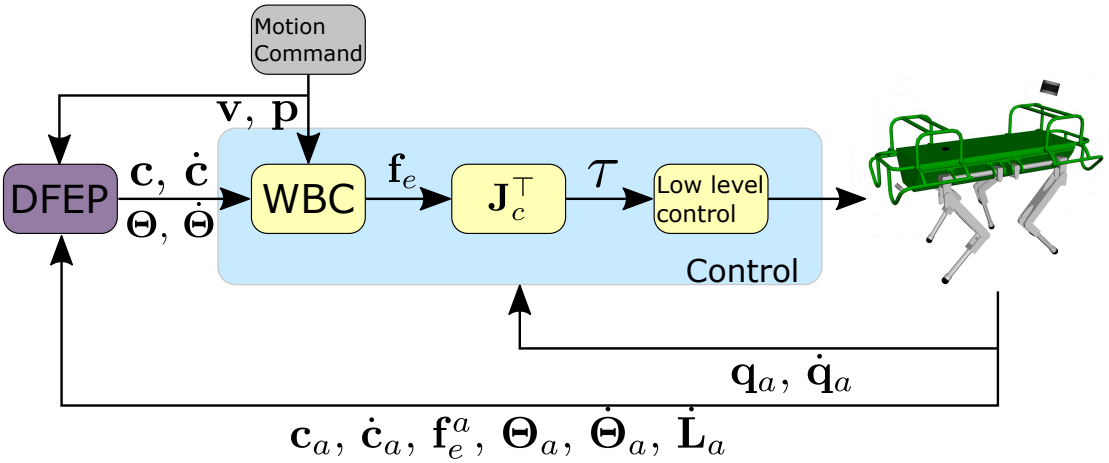


Figure 7.1: Diagram of the pipeline used to control the HyQ robot. Note that $\mathbf{v} \in \mathbb{R}^3$ is the commanded velocity and $\mathbf{p} = [\mathbf{p}_1 \ \dots \ \mathbf{p}_4]^\top$ is the foothold position. The The Dynamic Foothold Evaluation Planner (DFEP) generate reference trajectories, required GRFs and $\dot{\mathbf{L}}$ acquiring the actual states from the robot and the desired motion command ($\mathbf{v} \in \mathbb{R}^3$ is the desired velocity and $\mathbf{p} \in \mathbb{R}^{4\times3\times n_h}$ are the predicted footholds, where n_c is the prediction horizon). Only the references (CoM position and velocity and angular position and velocity) are then passed to a Whole Body Controller (WBC) which computes a set of GRFs to achieve the reference trajectory. The forces are then transformed in joint torques using the contact Jacobians ($\mathbf{J}_c$) and then commanded to the robot using a low-level controller.

presented in the previous chapters used as a motion planner. The The Dynamic Foothold Evaluation Planner (DFEP) generate reference trajectories, required GRFs and $\dot{\mathbf{L}}$ acquiring the actual states from the robot and the desired motion command ($\mathbf{v} \in \mathbb{R}^3$ is the desired velocity and $\mathbf{p} \in \mathbb{R}^{4\times3\times n_h}$ are the predicted footholds, where n_c is the prediction horizon). Only the references (CoM position and velocity and angular position and velocity) are then passed to a Whole Body Controller (WBC) [55] which computes a set of GRFs to achieve the reference trajectory. The forces are then transformed in joint torques using the contact Jacobians ($\mathbf{J}_c$) and then commanded to the robot using a low-level controller. Using a state estimator that gives CoM position, velocity, orientation and angular velocity, we provide the feedback to both DFEP and control logic such that we run a closed loop control architecture.

We run simulations using Gazebo [56] simulator to reproduce dynamics and sensors. The DFEP communicates with the WBC using ROS [57] *topics*. We run the DFEP in a ROS node, publishing the references in a topic. The references are resampled to match the update frequency of the WBC (25Hz - every 40ms). The WBC runs in another ROS node at a higher frequency (250Hz - 4ms) to ensure the correct error tracking thanks to the feedback loops. It receives the references

in samples by subscribing to the DFEP topic. To synchronize the two processes (reference update and control loop) the WBC resamples two subsequent references samples to match its update frequency.

7.1.1 Whole Body Controller (WBC)

One the WCB receives the reference trajectories (desired), it computes the desired linear and angular acceleration using a PD control law:

$$\ddot{\mathbf{c}}_d = K_{pcom}(\mathbf{c} - \mathbf{c}_a) + K_{dcom}(\dot{\mathbf{c}} - \dot{\mathbf{c}}_a) \tag{7.1}$$

$$\dot{\boldsymbol{\omega}}_d = K_{pbase} e(R_b^d - R_b^\top) + K_{dbase}(\boldsymbol{\omega} - \boldsymbol{\omega}_a) \tag{7.2}$$

Where $\ddot{\mathbf{c}}_d \in \mathbb{R}^3$ is the desired linear acceleration, all the $K_* \in \mathbb{R}^{3\times3}$ are the PD gains, $e(\cdot) : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is a mapping from a rotation matrix to the associated orientation vector. $R_b \in \mathbb{R}^{3\times3}$ and $R_b^d \in \mathbb{R}^{3\times3}$ are rotation matrices representing the actual and desired orientation of the base with respect to the world reference frame and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the desired angular velocity of the base, related to the euler angles rate $\dot{\boldsymbol{\Theta}}$ through a transformation matrix (4.10).

The Ground Reaction Forces (GRFs, $\mathbf{f}_e$) are computed using a Centroidal Dynamics model:

$$\underbrace{\begin{bmatrix} \mathbf{I} & \dots & \mathbf{I} \\ \mathbf{S}(\mathbf{p}_1) & \dots & \mathbf{S}(\mathbf{p}_{nc}) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_{e,\,1} \\ \vdots \\ \mathbf{f}_{e,\,nc} \end{bmatrix}}_{\mathbf{f}_e} = \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}}_d + \mathbf{g}) \\ \mathbf{I}_g \dot{\boldsymbol{\omega}}_d \end{bmatrix}}_{b} \tag{7.3}$$

At every control loop, the following quadratic program is solved to compute the desired GRFs:

$$\begin{aligned} \mathbf{f}_e^d = & \, argmin(\mathbf{A}\mathbf{f}_e - b)^\top S(\mathbf{A}\mathbf{f}_e - b) + \alpha \mathbf{f}_e^\top W \mathbf{f}_e \\ & \text{s.t. } \underline{d} < C\mathbf{f}_e < \overline{d} \end{aligned} \tag{7.4}$$

where $S \in \mathbb{R}^{6\times6}$ and $W \in \mathbb{R}^{3nc\times3nc}$ are positive-definite weight matrices, $\alpha \in \mathbb{R}$ weighs the secondary objective, $C \in \mathbb{R}^{p\times3nc}$ is the inequality constraint matrix that defines friction constraints and upper and lower bounds in the z direction,

$\underline{d}, \overline{d} \in \mathbb{R}^p$ the lower/upper bound respectively, with p being the number of inequality constraints and nc the number of contact points.

Once the GRFs have been computed, they are converted to joint torques further composed by two elements. The first element, called feed forward torques, is obtained by mapping the forces in the joint space $\boldsymbol{\tau}_{ff}$:

$$\boldsymbol{\tau}_{ff} = -S\mathbf{J}_c^\top \mathbf{f}_e \tag{7.5}$$

Where $\mathbf{J}_c \in \mathbb{R}^{nc\times(n+6)}$ is the stacked jacobian of the contact points, $S = [\mathbf{I}_{n\times n}\ \mathbf{0}_{n\times 6}]$ is a selection matrix that selects the actuated joints and n is the number of joints.

The second element is a PD joint-position controller with low gains, introduced for safety reasons and to move the swinging leg. During the swing motion, the gains of the leg in swing are increased to improve tracking capabilities. The desired joint torques $\boldsymbol{\tau}$ are then computed as:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{ff} + PD(\mathbf{q}, \dot{\mathbf{q}}, c_{st}) \tag{7.6}$$

Where c_{st} is a boolean vector that represent the stance condition of the legs. The computed joint torques are then sent to a low-level controller that tracks the desired joint torques. To close the control loop, from the robot the actual quantities ($_a$) are measured and used as feedback. The computation times required from the DFEP are not suitable to be used in a high frequency control loop (it requires about 0.25s to perform an evaluation), so the frequency at which we update the references is limited. We update the reference quantities at each lift-off, providing a reference up to the next lift-off event. In this way we have a reference for the current step at each step.

Adopting this configuration, we have two different ways of computing forces. The DFEP itself computes forces by solving the optimization problem, but they are not used to control the robot. We use them as a comparison between our method and the simulation. In the next sections, we show the output of the WBC and the DFEP show a sufficient degree of similarity.

7.2 Simulations

Imposing a final zero orientation

Here we show a simulation performed by imposing a final orientation $\mathbf{\Theta} = [0, 0, 0]^\top$. As already explained, we limit this evaluation to a single step evaluating the trajectory at lift-off. With an evaluation performed only at every lift-off, we can evaluate the trajectory only when the robot is moving, excluding all those cases when the robot starts from zero velocity. To account for these situations, we add the initial stance phase to the evaluation, computing always until the next lift-off. The simulation is performed assuming a flat ground environment.

In Fig. 7.2 we can see the trajectory, velocity and acceleration computed with the DFEP. We can see how the x and y components ($\mathbf{c}_x$, $\mathbf{c}_y$) are shaped to keep the CoM as much as possible inside the support polygon during swing. Considering that the leg to move is the RH, we can see how the CoM moves to provide a stable motion while reaching a safe position to grant stability for the next swing phase. On z axis ($\mathbf{c}_z$) we can see a flat trajectory.

The velocity plots show that starting from a zero velocity condition, at the end of the prediction horizon, we reach the desired velocity on all three axes, which

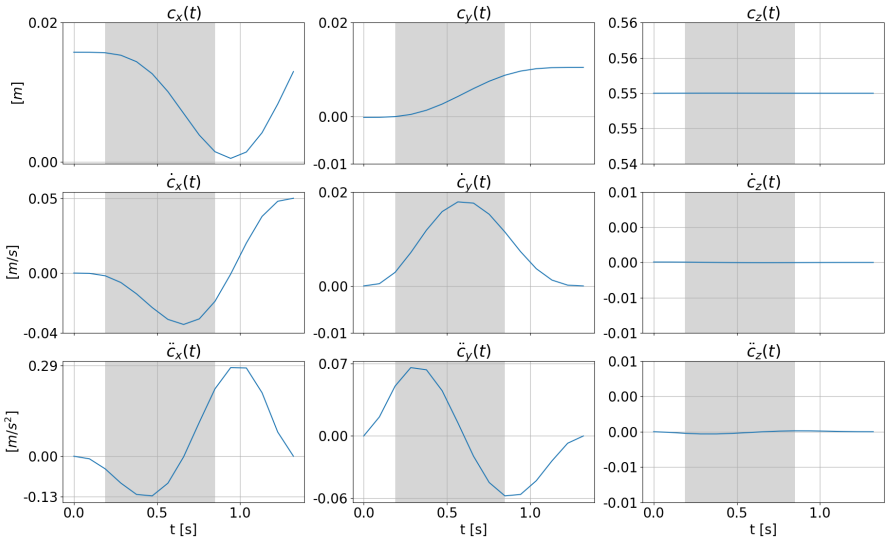


Figure 7.2: Plots of CoM position, velocity and acceleration with respect to time. These plots are divided in components (x, y, z - columns) and plotted quantity (position, velocity and acceleration - rows). The gray areas represent the swing phase of the robot. The initial stance phase is smaller than the final because we consider half of the stance time when starting from 0 velocity before the first leg lift.

has been commanded as $\mathbf{v} = [0.05,\ 0,\ 0]^\top m/s$. Note that no condition has been specified for the intermediate states.

The acceleration is related to the shape of velocity plot, being its derivative. We can see how the accelerations required to achieve such motion are pretty low, which reflects in low GRFs required to achieve such motion.

In the forces plot (Fig. 7.3) the GRFs required to achieve the motion are computed. They are not used as a reference for the WBC, but instead they are used to compare the quality of DFEP, as we will show later. The x and y components are the forces that drives the robot on the x and y axes, causing the acceleration behaviour (Fig. 7.2). The z component shows how during the swing phase (grey area) the swinging leg (RH) does not exert any force. From this plots we can see more clearly the hybrid nature of a legged robot. The forces are continuous in each phase but present a discontinuity when the contacts change, i.e., at each contact switch.

The legs support the weight of the robot. While in stance the forces are evenly distributed among all the legs, during swing, the robot is supported mainly by two out of the three legs in contact with the ground. This behaviour is caused by the position of the projection of the CoM with respect to the four contact points. The furthest the projection from a foot, the lower the force applied on it. In this case we can see how the lowest force is applied on the LF leg, which is the diagonally

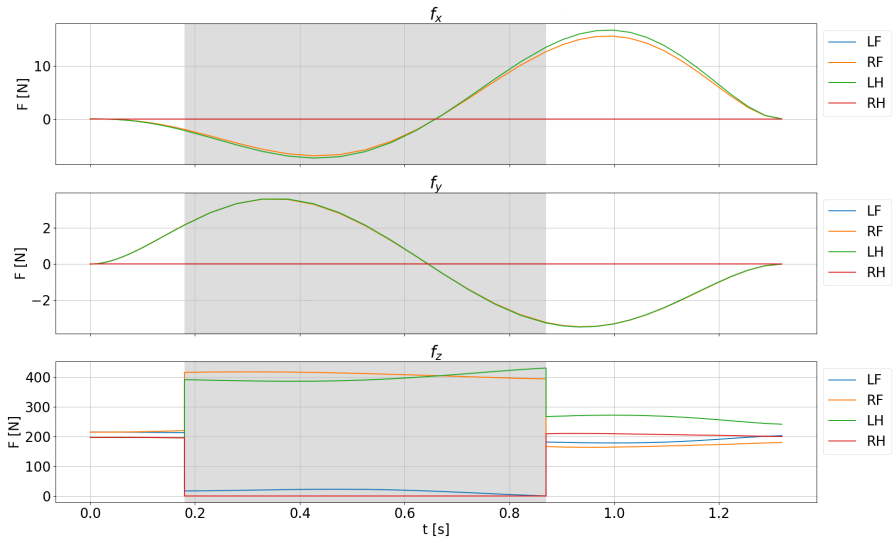


Figure 7.3: Plots of computed GRFs. The rows are the components of the forces (x, y, z), where each color indicates a different leg (LF, RF, LH, RH). The grey areas represent the swing phase of the robot.

opposite of the swinging leg (RH). The lowest force yields information about the CoM actual behaviour from another perspective. Since the force is low but not zero, we can assert that the the CoMs projection is relatively far from the LF leg but still inside the support polygon. As the time passes we can see that the force decreases, which means that the CoM is moving away from the LF leg, but still inside the support polygon as it doesn't reach zero until the end of the support phase. The same considerations can be made observing the highest forces. Since the RF and the LH legs are diagonally opposite, the z components of the forces that those leg exert follow a proportionally inverse relationship. We can see from their behaviour that the CoM moves toward the LH leg. The same considerations applies to all the ground reaction forces plots that we will see. The last stance phase shows a different distribution of forces with respect to the first stance phase, indicating that the CoM is not located at the center of the support polygon.

As we can see in Fig. 7.4, the angular momentum rate designed using (5.40) and the angular momentum computed as an optimization variable assume the same values along the entire horizon. An important observation about the angular momentum rate as an optimization variable is that it is considered a secondary objective. This is the consequence of having defined hard constraints for the trajectory ((6.1b) and (6.1c)) and only a tracking term on $\dot{\mathbf{L}}$. Although no hard constraints have been defined on $\dot{\mathbf{L}}$, we are able to track it completely.

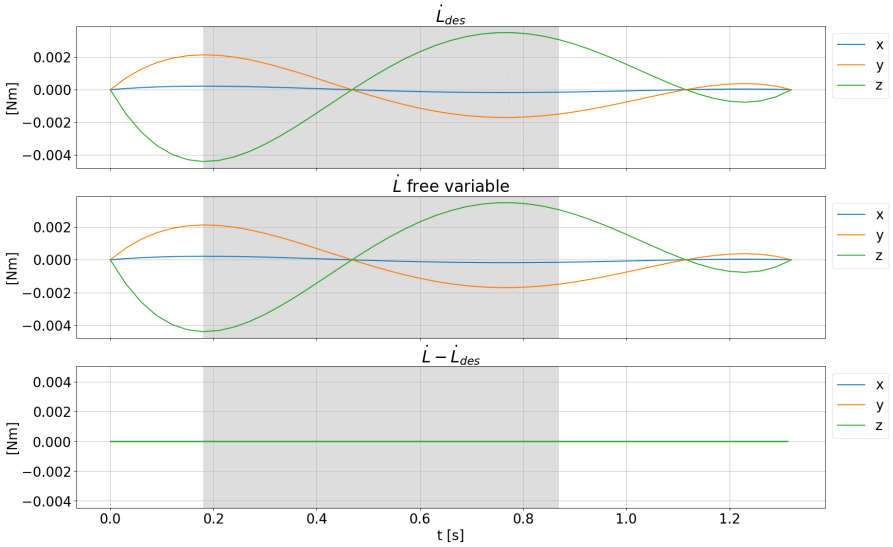


Figure 7.4: Plots of reference and actual $\dot{\mathbf{L}}$. The last plot shows the tracking error between the desired and the computed one. Each color indicates a different component (x, y, z). The grey areas represent the swing phase of the robot.

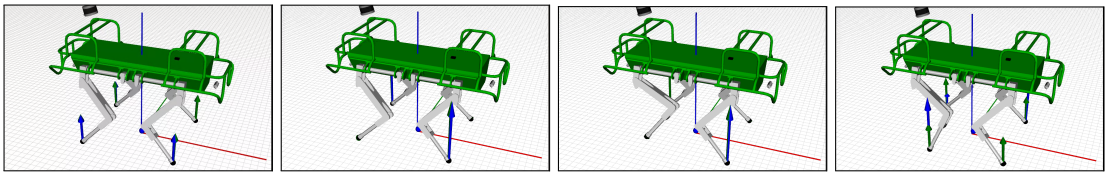


Figure 7.5: From left to right, sequence of pictures taken during simulation, where the swinging leg is the Right Hind (RH).

The optimization problem results are then sent to the Whole Body Controller (WBC) to drive the robot according to the references. In figure 7.5 we can see the robot taking a step during simulation.

In Fig. 7.6 we can see the references and the actual quantities compared. The CoM trajectory achieved using the control structure presented in Fig. 7.1, recorded in simulation, track with a good precision the trajectory provided to the WBC. The successful tracking of such trajectory is an indirect first proof of the actual feasibility of the computed trajectory. This behaviour can be further confirmed by examining the forces plots (Fig. 7.7).

The forces plot (Fig. 7.7) represent a more direct proof of the validity of DFEP. If we consider only the x and y components of the DFEP (Optimizer) and WBC (Simulation) forces, we see that they have different behaviours. The difference is caused by the different approach considered when computing the forces. The WBC uses a different optimization and cost function with respect to the DFEP, causing a similar order of magnitude in the values but a different behaviour. The principal and most interesting aspect of this plot can be observed analysing the z component

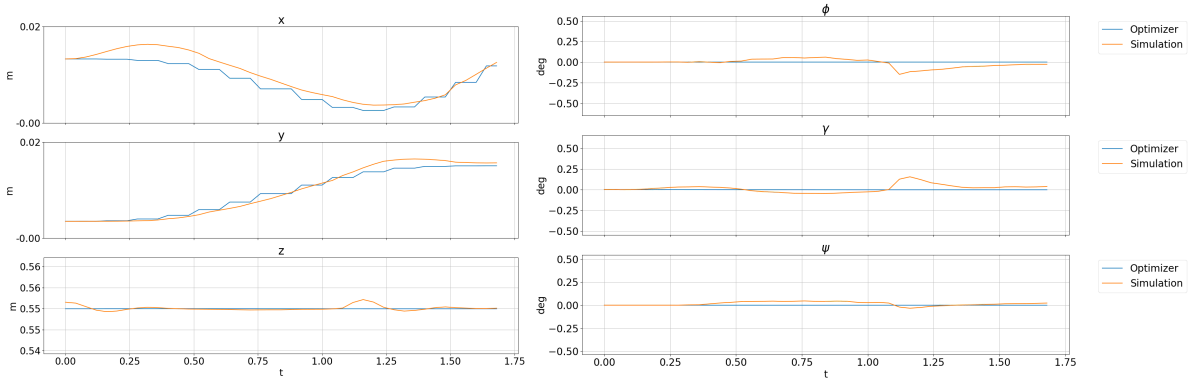


Figure 7.6: Plots of references (Optimizer) and actual (Simulation) tracked quantities (CoM position and orientation). The references are resampled to match the update frequency of the WBC (40ms), while the simulation quantities are smoother due to their lower sampling time (4ms). We can see how the DFEP quantities are accurately tracked.

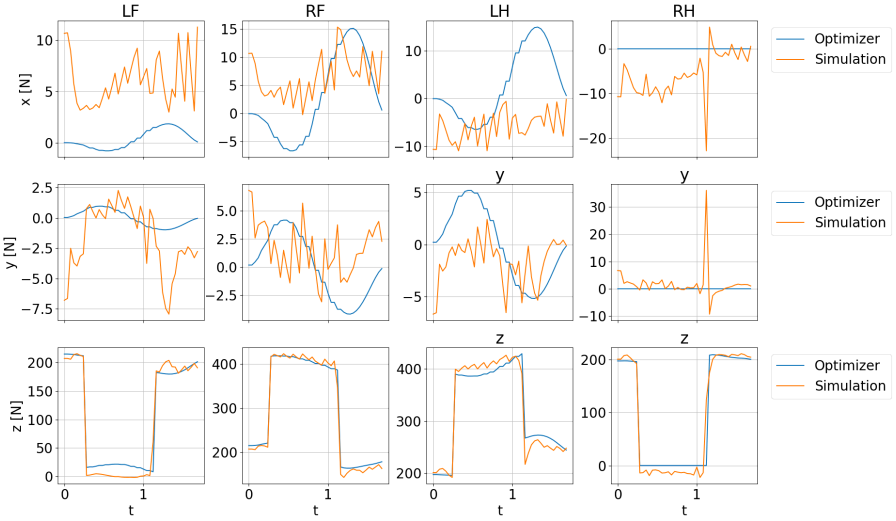


Figure 7.7: Plots of DFEP (Optimizer) and actual (Simulation) forces. On the rows we find the components (x, y, z) and on the columns the different legs (LF, RF, LH, RH). The forces obtained with two different methods (DFEP and simulation) are compared to show the similar behaviour that prove the correctness of the method proposed.

of the forces. The strong similarities between the DFEP and the WBC forces is a sufficient proof of the correctness of the method proposed in this thesis. The main reason is that the DFEP computes the forces required to achieve the trajectory generated. Since both forces and trajectory are optimization variable, they store

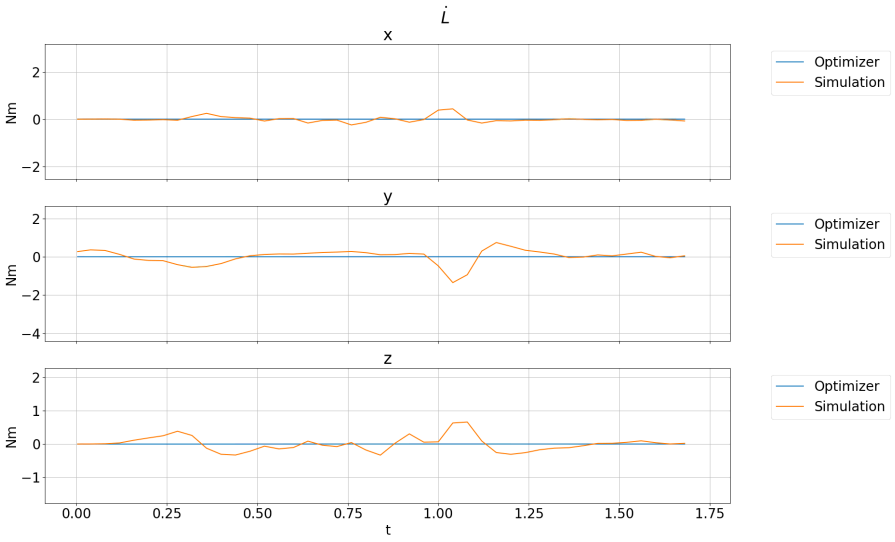


Figure 7.8: Plots of DFEP (Optimizer) and actual (Simulation) $\dot{\mathbf{L}}$. In each row a component (x, y, z) is shown. It is not used as a reference neither controlled by the WBC. The tracking of $\dot{\mathbf{L}}$ is a consequence of the trajectory and orientation provided as a reference.

no information about the actual forces applied on the system at the initial time instant. On top of that, the DFEP computes forces for the entire time horizon without any feedback to correct the behaviour or provide information along the motion. The prediction of forces behaviour reveals to be very accurate on the z axis, showing that the DFEP constitutes a valid method to generate feasible dynamics trajectories.

One of the most promising results can be seen in 7.8, where we compare the angular momentum rate computed with DFEP ($\dot{\mathbf{L}}$) and the actual one measured in simulation. We can see how even without providing any information about $\dot{\mathbf{L}}$ as reference (e.g., forces), or without having any feedback term able to reduce the error, the actual $\dot{\mathbf{L}}$ behaviour is very close to the one predicted by the DFEP. This tracking is a natural consequence of the trajectory and orientation used as reference. Although small bumps still occur in the measured data, mainly caused by noisy measurements or impacts with the ground, the general behaviour of the angular momentum rate is tracked.

Imposing a final orientation different than zero

In this trial, we impose a desired final orientation $\mathbf{\Theta} = \left[\frac{\pi}{24}, \frac{1.5\pi}{24}, 0\right]$. The simulation is set-up exactly as the previous case. In figure 7.9 we present some frames of the successful crawl.

In Fig. 7.10 the trajectory generated by the DFEP is shown. We can see that it is very similar to the previous case, in which the desired orientation was imposed to $\mathbf{\Theta} = [0,\, 0,\, 0]\, rad$. This is an expected result as the different orientation should have a minor impact on the linear quantities. Also here we can see that the trajectory is shaped in such a way to keep the CoM as much as possible inside the support polygon during each phase, while accelerating from a zero velocity to a desired velocity. The main difference between this and the previous case is highlighted by

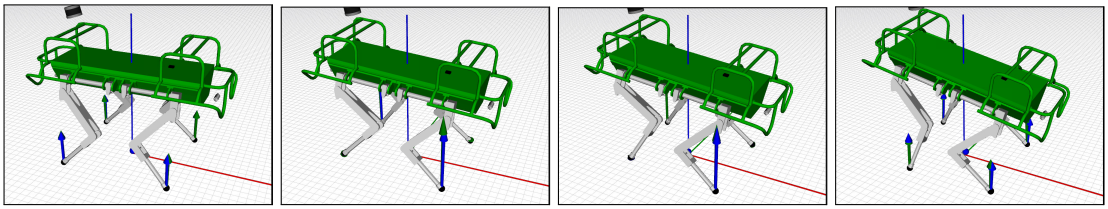


Figure 7.9: From left to right, sequence of pictures taken during simulation, where the swinging leg is the Right Hind (RH). Here a final orientation different than 0 is imposed.

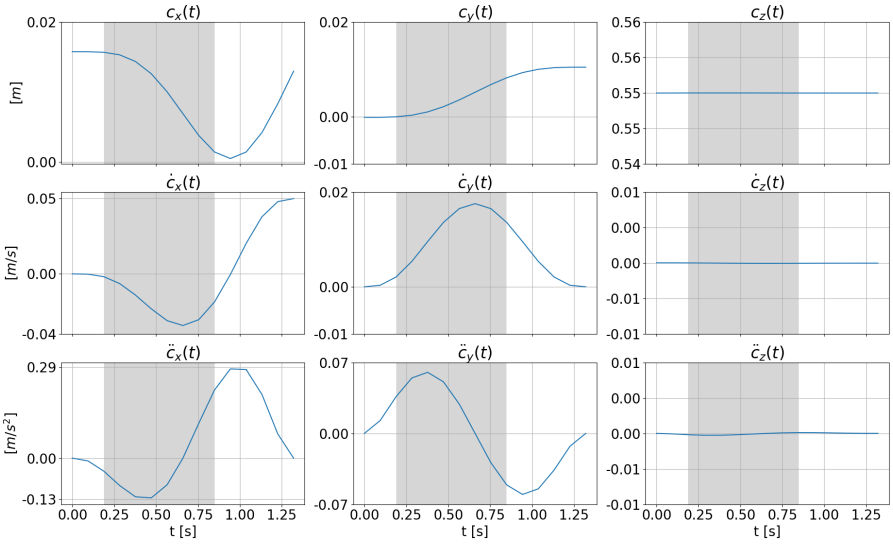


Figure 7.10: Plots of CoM position, velocity and acceleration with respect to time. These plots are divided in components (x, y, z - columns) and plotted quantity (position, velocity and acceleration - rows). The gray areas represent the swing phase of the robot. The initial stance phase is smaller than the final because we consider half of the stance time when starting from 0 velocity before the first leg lift.

the next plots.

In the forces plots (Fig. 7.11) we can see the first major difference with respect to the previous case and one of the most important aspects about the DFEP. Comparing the x and y components of this and the previous case forces, we can

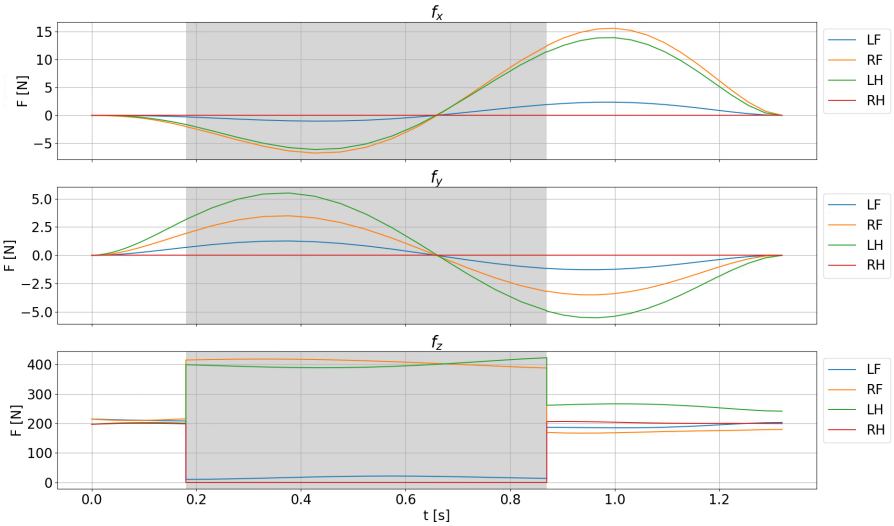


Figure 7.11: Plots of computed GRFs. The rows are the components of the forces (x, y, z), where each color indicates a different leg (LF, RF, LH, RH). The grey areas represent the swing phase of the robot.

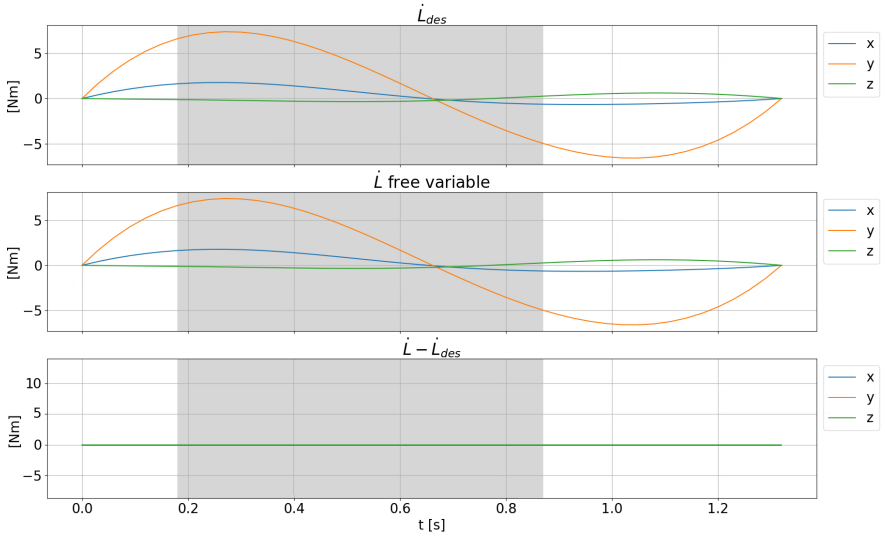


Figure 7.12: Plots of reference and actual $\dot{\mathbf{L}}$. The last plot shows the tracking error between the desired and the computed one. Each color indicates a different component (x, y, z). The grey areas represent the swing phase of the robot.

see how although the trajectory computed is the same, the forces component present some differences. If the trajectory is not the cause of the different forces behaviours,they can only be explained by the variation of the $\dot{\mathbf{L}}$ variable. Although angular quantities are not directly involved in the optimization problem that we are solving (6.1a), the angular momentum rate variable $\dot{\mathbf{L}}$ is tracking a term which depend on angular quantities $\dot{\mathbf{L}}_{ref}$. The higher and different forces computed in this case are then used to generate the moments needed to cause the variation on the angular momentum, i.e., to track the angular momentum rate and then, implicitly, angular quantities.

The other major difference with respect to the previous case is shown in the angular momentum rate plot (Fig. 7.12). This time the desired angular momentum is higher than the previous case by three orders of magnitude. In this test, we are imposing a final desired orientation different than $\boldsymbol{\Theta} = [0, 0, 0]$. To reach such final orientation, the robot needs to achieve a certain angular velocity and acceleration, i.e., it needs to change its angular momentum to change orientation. If the angular momentum changes in time, also the angular momentum rate will be different than 0. We compute this variation using (5.40), shown in the plot as $\dot{\mathbf{L}}_{des}$. By tracking its behaviour with the optimization variable $\dot{\mathbf{L}}$, we are able to completely reproduce its variation in time, compensating for the lack of angular quantities (orientation, angular velocity and acceleration) in the optimization problem.

We can see from Fig. 7.13 how the WBC is able to track the quantities provided as a reference. By comparing the trajectory and orientation computed by the DFEP and the ones measured in simulation, we can assert that the two are close enough to prove the feasibility of the generated trajectory. As described in the previous case, by comparing the forces recorded in simulation and generated by the DFEP, we can further confirm this assertion.

In Fig. 7.14 we can again compare the forces obtained with the two different method (DFEP and simulation). As in the previous case, the components on the x and y components are not very similar despite having values of the same order of magnitude. This is caused by the different approach used to compute them and also because of noise when dealing with such low forces. By comparing z component of the forces, we find again a similarity between the compared data that proves the validity of our model. A further confirm of the correctness of the method presented in this thesis, can be found comparing the optimization variable $\dot{\mathbf{L}}$ and the values measured in simulation.

In Fig. 7.15 we can compare the angular momentum rate computed with DFEP with the one measured from simulation. In the previous case we have seen that completely tracking a zero angular momentum rate is possible while crawling. In this case we set a final desired orientation different than $\mathbf{\Theta} = [0,\ 0,\ 0]$, to smoothly change the final pose of the robot. The angular momentum rate measured in simulation during the execution of the mixed motion (moving CoM while changing

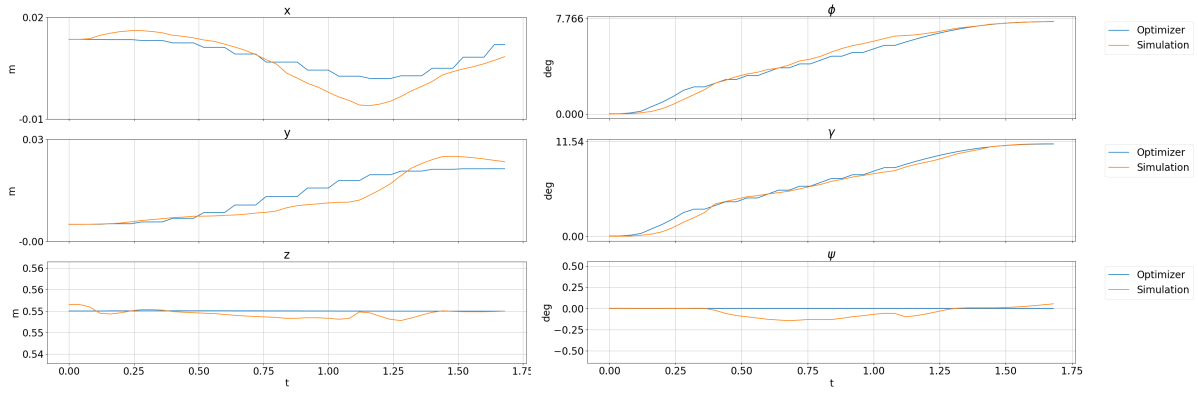


Figure 7.13: Plots of references (Optimizer) and actual (Simulation) tracked quantities (CoM position and orientation). Be aware that $\frac{\pi}{24}\ rad = 7.5\ deg$ and $\frac{1.5\pi}{24}\ rad = 11.25\ deg$. The references are resampled to match the update frequency of the WBC (40ms), while the simulation quantities are smoother due to their lower sampling time (4ms). We can see how the DFEP quantities are tracked reasonably well.

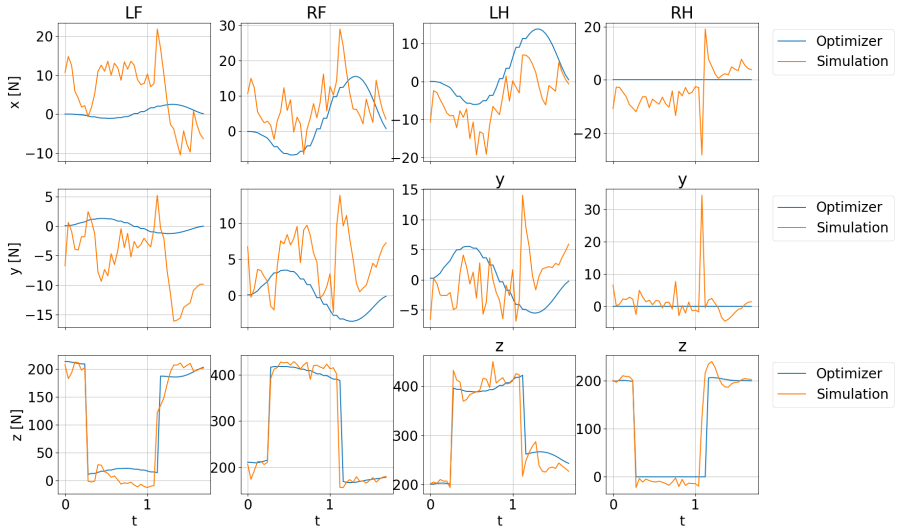


Figure 7.14: Plots of DFEP (Optimizer) and actual (Simulation) forces. On the rows we find the components (x, y, z) and on the columns the different legs (LF, RF, LH, RH). The forces obtained with two different methods (DFEP and simulation) are compared to show the similar behaviour that prove the correctness of the method proposed.

orientation) and the $\dot{\mathbf{L}}$ variable computed with DFEP have a very similar behaviour. All these factors, such as force and $\dot{\mathbf{L}}$ similarities, contribute to prove the validity of the proposed method, able to cope with the nonlinearities present in a RBDM dynamic model (5.4).

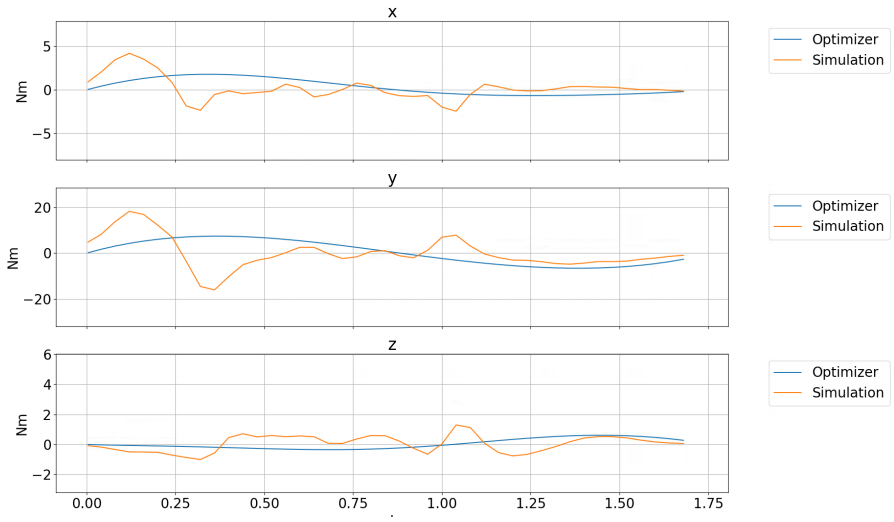


Figure 7.15: Plots of DFEP (Optimizer) and actual (Simulation) $\dot{\mathbf{L}}$. In each row a component (x, y, z) is shown. It is not used as a reference neither controlled by the WBC. The tracking of $\dot{\mathbf{L}}$ is a consequence of the trajectory and orientation provided as a reference.

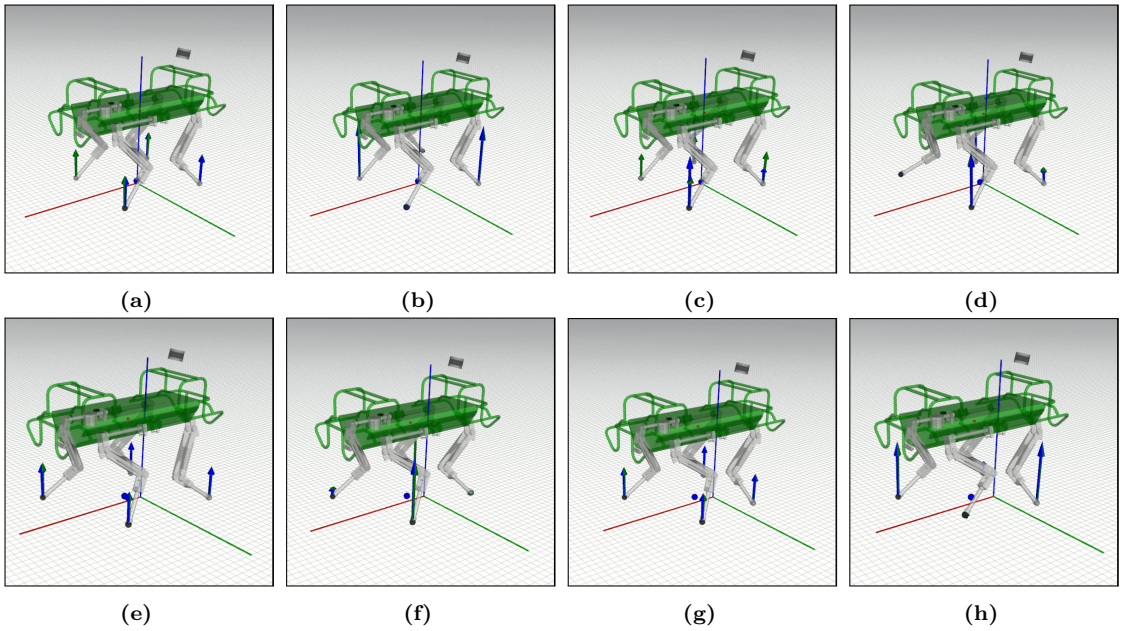


Figure 7.16: Snapshots of various phases achieved during the execution of the motion.

Long time horizon with zero orientation as target

In this test we chose to set as target an orientation of $\boldsymbol{\Theta} = [0, 0, 0]^\top$, and to keep the same configuration as the previous cases (flat ground). In this test we let the simulation run for multiple consecutive steps, to show that the trajectories computed step after step by the DFEP are physically achievable also for longer time horizons with respect to the previous cases shown. In figure 7.16 we show some frames acquired during the execution of the motion.

As done in the previous tests, in Fig. 7.17 we show the comparison between the DFEP and the actual simulation data. From these plots we can prove the stability over a long time horizon and multiple subsequent runs of the DFEP. Both linear and angular trajectories are tracked with a good precision over the entire simulation duration. By examining closely the x and y components of the linear trajectory we can see how the trajectory computed by the DFEP behaviour is not exactly the same at each phase. This asymmetric behaviour gives the evaluation an increased robustness with respect to generating periodic motions with no regard for initial conditions. The trajectory generated by the DFEP is adapted to the initial and final states each time that the evaluation is performed.

The comparison between computed and simulated forces (Fig. 7.18) shows once again the similarity between the two sets of forces, which are independent of

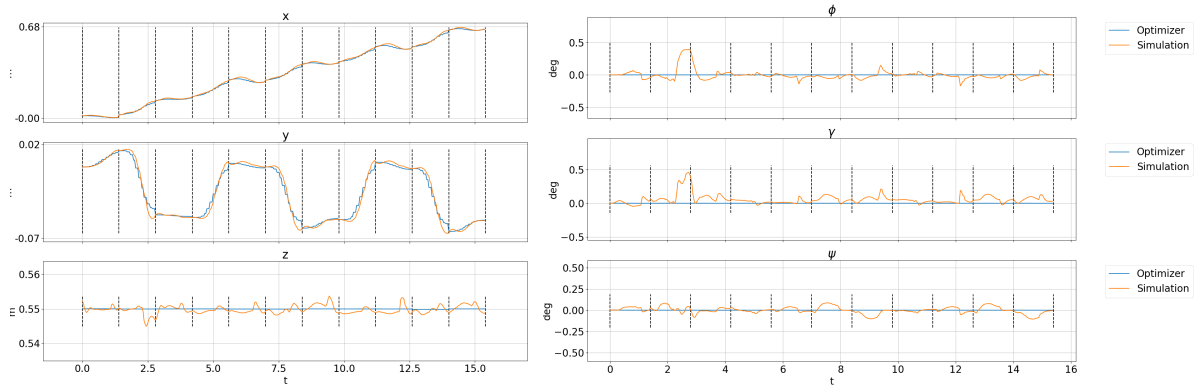


Figure 7.17: Plots of references (Optimizer) and actual (Simulation) tracked quantities (CoM position and orientation). The references are resampled to match the update frequency of the WBC (40ms), while the simulation quantities are smoother due to their lower sampling time (4ms). We can see how the DFEP quantities are tracked reasonably well. black dashed lines are delimiters to highlight different phases of the robot.

each other. The fact that they are independent but very similar also over a long time horizon, proves once again the validity of the quantities computed with the DFEP. We have run tests for longer time horizons with the same result as the one reported here. A shorter time horizon has been chosen to preserve plots readability. Although in this test we consider a constant orientation $\boldsymbol{\Theta} = [0,\ 0,\ 0]$ along the

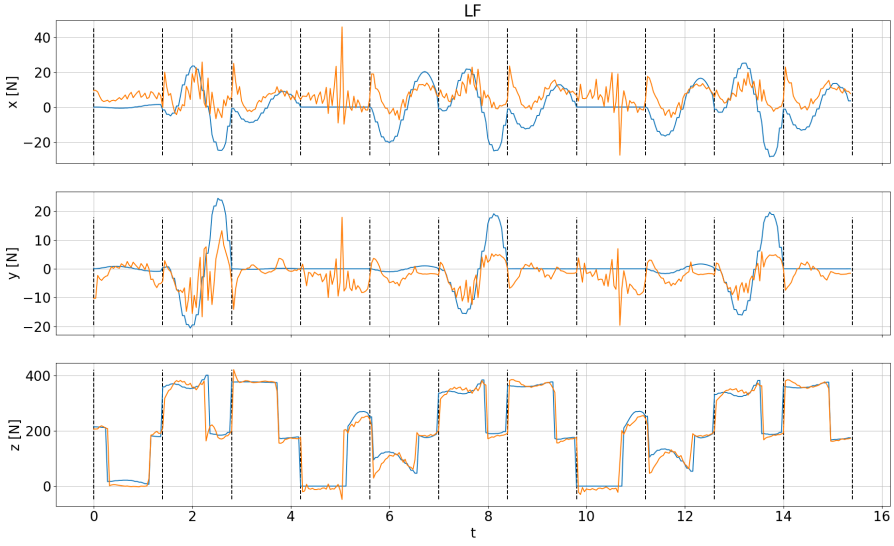


Figure 7.18: Plots of DFEP (Optimizer) and actual (Simulation) forces. On the rows we find the components (x, y, z). Here we show the force relative to only one leg (LF) for the sake of plot readability. The forces obtained with two different methods (DFEP and simulation) are compared to show the similar behaviour that prove the correctness of the method proposed. Black dashed lines are delimiters to highlight different phases of the robot.

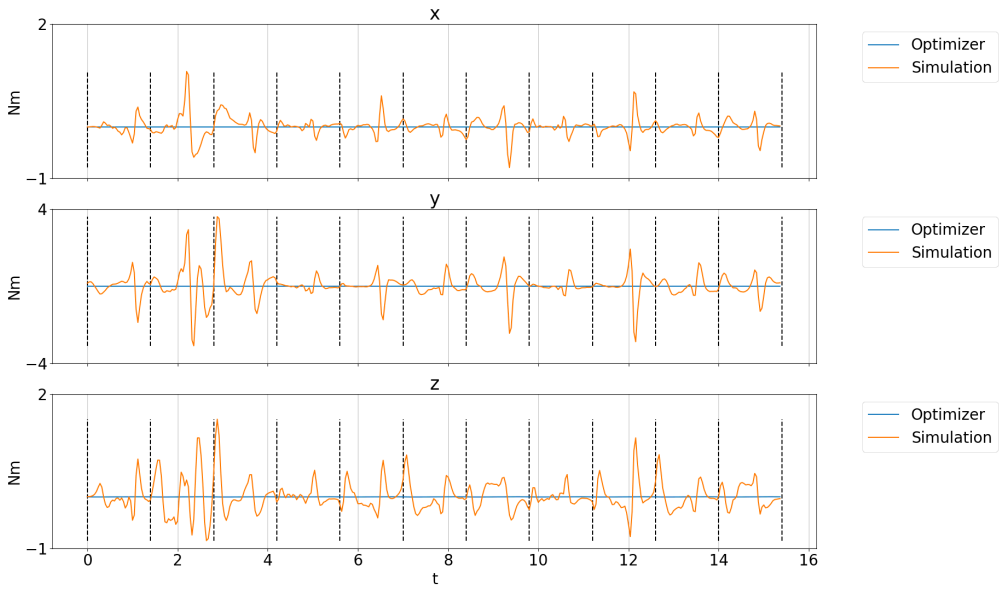


Figure 7.19: Plots of DFEP (Optimizer) and actual (Simulation) $\dot{\mathbf{L}}$. In each row a component (x, y, z) is shown. It is not used as a reference neither controlled by the WBC. The tracking of $\dot{\mathbf{L}}$ is a consequence of the trajectory and orientation provided as a reference. Black dashed lines are delimiters to highlight different phases of the robot.

entire time horizon, we have already proven that imposing a different desired final orientation yield the same results in terms of tracking. This is a sufficient proof to assert that the DFEP is stable over a long time horizon.

As shown previously for forces, also the angular momentum rate appears to be very similar to the one predicted by the DFEP (7.19). Here we see how even without a direct angular momentum rate feedback, $\dot{\mathbf{L}}$ is forced to 0. The angular momentum rate measured in simulation still presents some bumps due to unmodelled dynamics such as impacts, but the overall behaviour correspond very closely to the one computed by the DFEP.

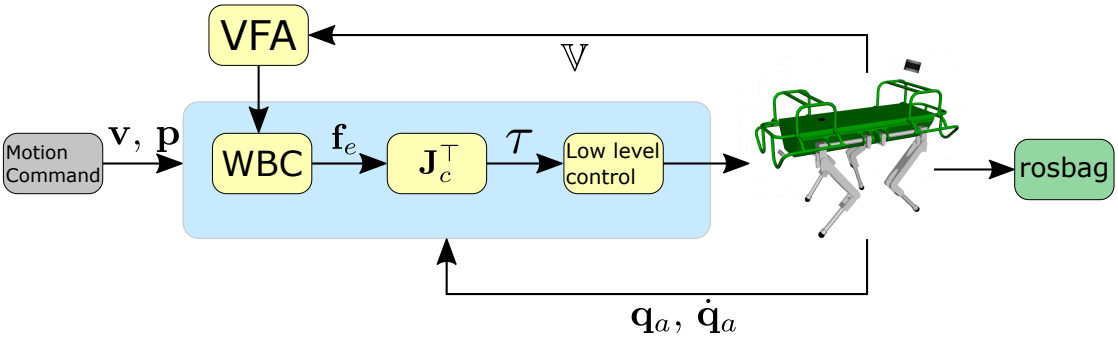


Figure 7.20: Diagram of the pipeline used to control the HyQ robot. Note that $\mathbf{v} \in \mathbb{R}^3$ is the commanded velocity and $\mathbf{p} = [\mathbf{p}_1 \ \dots \ \mathbf{p}_4]^\top$ is the foothold position. The motion command is passed to a Whole Body Controller (WBC) which computes a set of GRFs to achieve desired motion. The forces are then transformed in joint torques using the contact Jacobians ($\mathbf{J}_c$) and then commanded to the robot using a low-level controller.

7.3 Dynamic Foothold Evaluator (DFE)

In this section we implement our method to evaluate dynamic transition feasibility as an evaluation to select a foothold from a heightmap acquired using visual-feedback. In figure 7.20 we show the pipeline that defines the control strategy used. Unlike the previous case (DFEP) where the evaluator is used directly to control the robot as a motion planner, here we instead use it off-line to evaluate data recorded in simulation. In this step we added visual feedback $\mathbb{V}$ that acquires the terrain in a discrete fashion, generating grids of points that store the height of the terrain. The VFA evaluation [6] uses this feedback to provide a correction on the WBC output, adjusting the landing locations of the robot's feet.

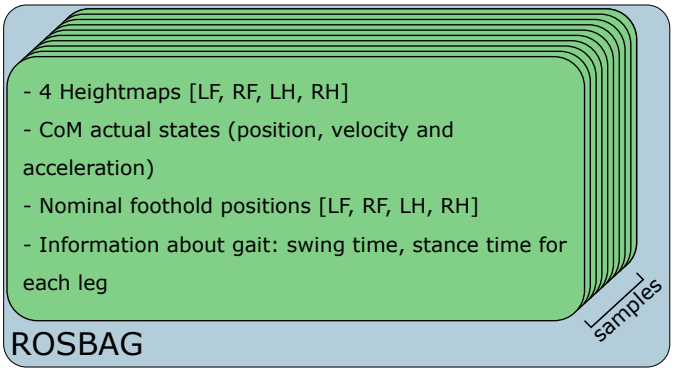


Figure 7.21: Rosbag file schematics. Each sample contains information on the heightmaps for each leg (LF, RF, LH, RH), on the actual CoM states, on angular quantities, on the nominal foothold positions and information about swing characteristics. The nominal foothold for a leg in swing is the predicted landing position of that leg. Swing and stance time are percentage values ($0 \rightarrow 1$) that store informations on the time elapsed in swing or in stance of each leg.

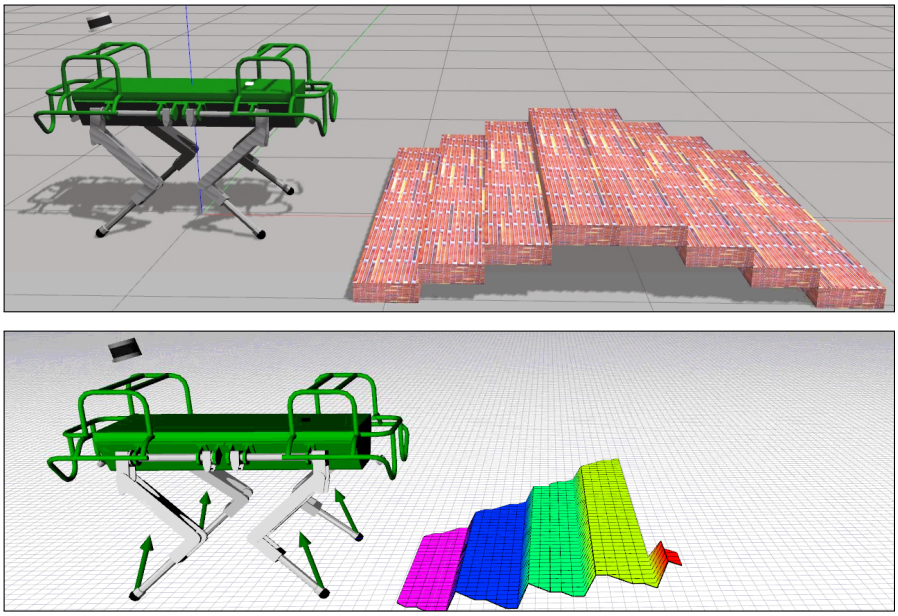


Figure 7.22: Complex scenario environment (top) and what the robot captures, used for visual feedback (bottom). The acquired heightmap is updated as the robot moves in the environment. The scenario used to perform the test is composed by a part of flat ground followed by a series of 8cm high steps.

We run simulations using Gazebo [56] simulator to reproduce dynamics and sensors. We recorded a set of data from simulation, stored in a log file (*rosbag*). Here we save the information about the robot states and the heightmaps at each time instant, organizing its structure as shown in Fig. 7.21.

To show the effectiveness of the evaluator, we consider a complex scenario where the robot has to climb stairs (Fig. 7.22).

We evaluate samples picked from the recorded rosbag file with the DFE, showing that it can be used as a foothold evaluator on a discrete heightmap. The DFE's core is the optimization problem described in the previous chapter, to which we added new cost terms to select an optimal foothold, which is the one with the lowest cost. The new cost terms introduced are:

1. $\| \mathbf{c} - \overline{\mathbf{c}} \|_2$ - which is a term that bias the evaluation toward more straight trajectories. This term allows us to have smoother trajectories and to reduce the oscillating motion of the trunk unless strictly needed for stability reasons;

2. $\| \mathbf{f}_e \|_2$ - used to minimize the ground reaction forces to achieve the motion. This term is useful to increase the efficiency of the motion, by exerting forces only when strictly needed;

3. $w \parallel \mathbf{h}_i - \mathbf{h}_n \parallel$ - where w is a weighting term and $\mathbf{h}_i$ is the foothold evaluated. It is a term used to discourage the footholds that are too far with respect to the nominal. This term is added to disturb the least possible the robot. The optimal foothold is "pulled" toward the nominal. This helps also to choose an optimal foothold when multiple footholds with similar costs are found, by preferring the nearest to nominal foothold.

The new cost terms defined are included in the optimization problem formulation, resulting in:

$$\min_{\boldsymbol{\rho}_4,\, \mathbf{f_e}(t),\, \dot{\mathbf{L}}} \quad Q_0 \parallel \dot{\mathbf{L}} - \dot{\mathbf{L}}_{ref} \parallel_2 + Q_1 \parallel \mathbf{c} - \overline{\mathbf{c}} \parallel_2 + Q_2 \parallel \mathbf{f}_e \parallel_2 + w \parallel \mathbf{h}_i - \mathbf{h}_n \parallel \tag{7.7a}$$

$$\text{such that} \quad m(\ddot{\mathbf{c}}(t) - \mathbf{g}) = \sum_{i=1}^{n} \mathbf{f}_{e,\,i}(t) \tag{7.7b}$$

$$m\mathbf{c}(t) \times (\ddot{\mathbf{c}}(t) - \mathbf{g}) + \dot{\mathbf{L}} = \sum_{i=1}^{nc} \mathbf{p}_i \times \mathbf{f}_{e,\,i}(t) \tag{7.7c}$$

$$\mathbf{f}_e \leq \mathbf{f}_{e,\,max} \tag{7.7d}$$

$$\parallel f_x \parallel \leq \mu f_z \tag{7.7e}$$

$$\parallel f_y \parallel \leq \mu f_z \tag{7.7f}$$

$$f_z \geq 0 \tag{7.7g}$$

Where $Q_0, Q_1, Q_2 \in \mathbb{R}^+$ are weighting coefficients, chosen to even out the different orders of magnitude in the cost function ($\mathbf{f}_e$ is in the order of thousands, while $\mathbf{c} - \overline{\mathbf{c}}$ is in the order of fractions of unit). This optimization problem will evaluate, for each sample, all the footholds that belong to the heightmap acquired.

7.3.1 Evaluations for Stair Climbing While Crawling

To acquire heightmaps in the case of a crawling motion, we acquired the surrounding area of the nominal foothold in form of heightmaps of 30cm × 30cm with a resolution of 2cm during the climbing of stairs. We evaluate 20 samples equally spaced in time (Fig. 7.23). In all samples evaluated, the evaluator was able to find feasible solutions to the optimization problem (7.7a). Each heightmap was preprocessed with the VFA algorithm to discard footholds (Fig. 7.24) to reduce the evaluation set for the DFE and to consider also geometric-related aspects. The DFE found

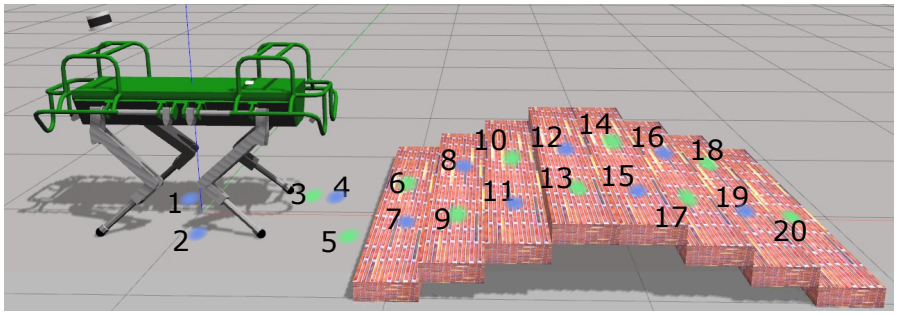


Figure 7.23: The samples evaluated are those associated with the nominal footholds marked in the picture. Two different colors have been used for the front (green) and rear (blue) legs.

that all the VFA-feasible footholds were dynamically feasible as well, which is a reasonable result given the size of the heightmap evaluated and the type of gait considered (slow and quasi-static). Here we report the result of only one sample to fully describe it. The sample shown in figure 7.24 is sample 9, where the Right Front (RF) leg has to land on a step.

The DFE generates a cost map according to the optimization function defined in (7.7a), storing the value of the cost function for each DFE-feasible foothold (Fig. 7.25). The foothold with the lower cost is selected as the *optimal foothold*. Each foothold evaluated can be further expanded to show the evaluated quantities for that specific foothold, namely, the CoM trajectory, velocity and acceleration, the ground reaction forces and both angular momentum rates (reference and optimization

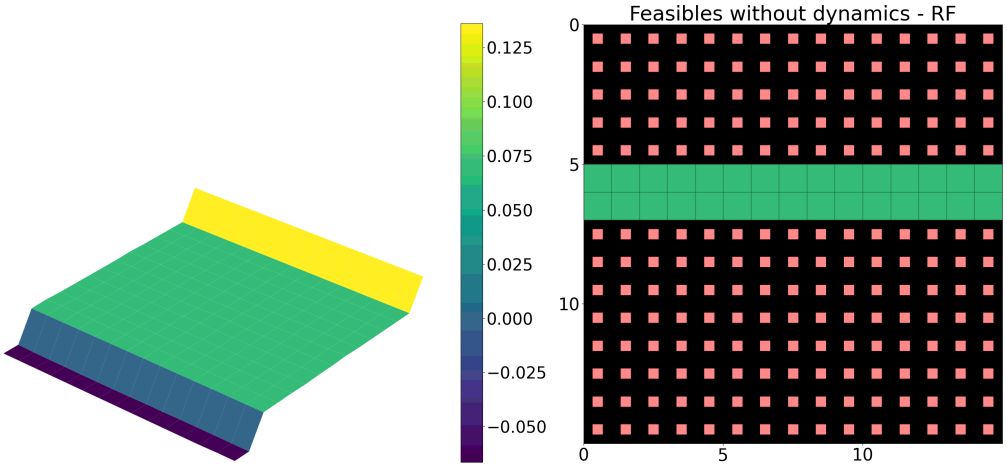


Figure 7.24: Heightmap acquired for the Right Front (RF) leg (on the left) is processed with the VFA algorithm (right) to discard footholds according to geometric-related aspects. Green squares mark the VFA-feasible footholds. The VFA discarded most of the footholds because they were close to both edges of the steps.

variable). Since we are evaluating the footholds with a cost function that bias the problem towards the minimization of forces (along with other terms), the cost map presents some differences in the cost of each foothold. This cost map is relative to the Right Front (RF) leg of the robot. Keeping this in mind, we can observe that the footholds on the right side have higher costs than footholds on the left side. This happens mainly for two reasons. The first reason is that the trajectory needed to achieve a stable motion has a higher curvature with respect to the one required by choosing footholds on the left side. The other reason is that the further from the body is the foothold considered, the harder it becomes to accurately track the $\dot{\mathbf{L}}$.

Here we show the plots relative to the foothold with the lowest cost. As already explained in 7.2, we perform the evaluation at each lift-off.

In Fig. 7.26 the trajectory computed with the DFE are reported. We can see how the trajectory changes to bring the CoM in stable positions along the motion. The trajectory shown here is relative to the foothold with the lowest cost, i.e., the optimal foothold. This means that the trajectory shown here is the one that let all terms considered in the cost function to assume the lowest value, and making this foothold the best choice for achieving the desired motion. Now the trajectory on the z axis is not any more flat like the previous case (DFEP), but it changes

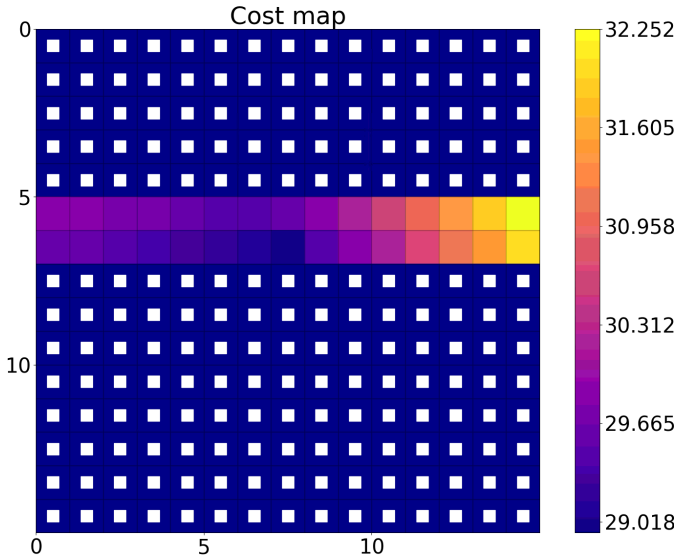


Figure 7.25: Output cost map. The white squares mark the discarded footholds. In this heightmap we can see that all the footholds VFA-feasible are also DFE-feasible since they have been found to have a cost. A cost value is associated to a foothold only if the foothold is DFE-feasible.

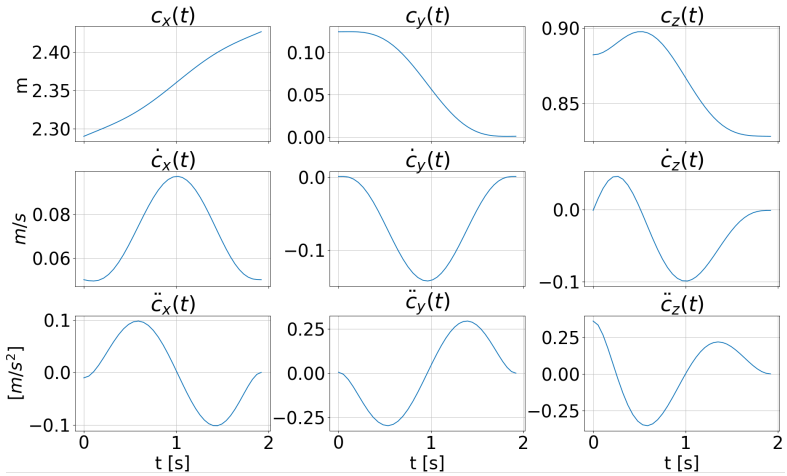


Figure 7.26: CoM position, velocity and acceleration generated by the the DFE during a crawling motion on stairs. Each column contain a different component (x, y, z), while each row a different linear quantity (position, velocity and acceleration).

according to the motion that we are planning to achieve on stairs.

The forces shown here (Fig. 7.27) are quite similar to the one computed while using the DFE as a planner (DFEP). The differences that we can see here are mainly due to the fact that now we are climbing stairs and we need extra force to move our CoM also along the z axis. It is the cause of having less evenly spread forces, specially on the moved leg (RF).

In the angular momentum rate plot (Fig. 7.28) we can see how the angular momentum can still be tracked completely. This means that during a crawling

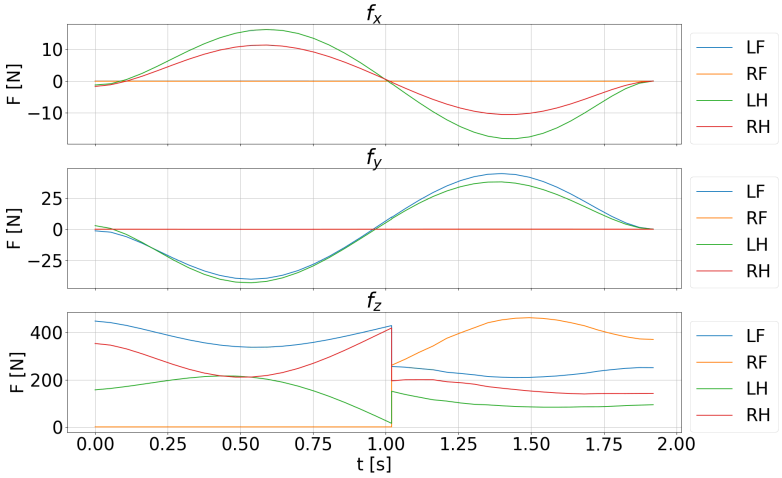


Figure 7.27: Forces needed to achieve the trajectory for a crawling motion. On each row we have different components (x, y, z) and each color represent a different leg (LF, RF, LH, RH).

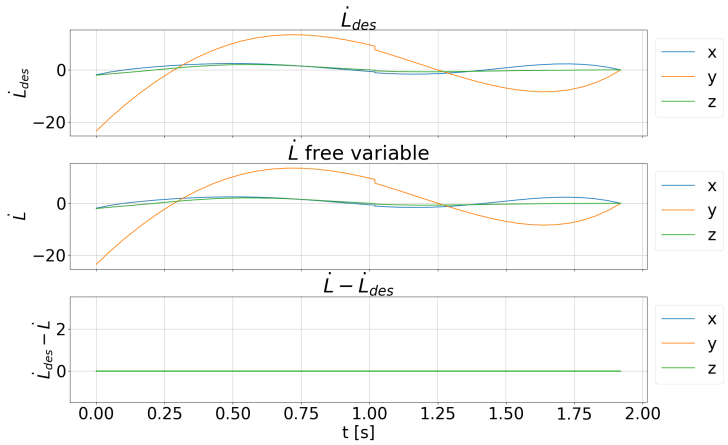


Figure 7.28: Angular momentum rate needed to achieve the motion. Here we compare the desired motion with the actual one, showing that the desired value could be tracked entirely. Each color represent a different component (x, y, z).

motion we have the ability to track both linear and angular quantities even when climbing stairs.

7.3.2 Evaluations for Stair Climbing While Trotting

We now proceed to evaluate the same scenario but changing the gait type to a more dynamical one, namely, the trot. While trotting, the system can be addressed as an underactuated system which means not being able to track both angular and linear quantities. In a first attempt we try to impose the condition $\dot{\mathbf{L}} = 0$, which results in finding only infeasible solutions. This means that executing a trotting motion imposing $\dot{\mathbf{L}} = 0$ is not physically possible.

By relaxing this constraint, we run the evaluation as in the previous case, i.e., creating an angular momentum rate reference and defining $\dot{\mathbf{L}}$ as an optimization variable.

Once again we acquire data from simulation evaluating 20 samples equally spaced in time, as did while crawling. The main difference with respect to crawl is that with two legs in air at the same time, we would need to evaluate combinations of footholds between two heightmaps (one for the rear leg and one for the front leg in swing), which would increase exponentially the computation time needed for each sample. To reduce it, we fix the rear foothold to be the nominal one (center of the heightmap). This time we increased the dimension of the acquired heightmap to a 33×33 discrete map with a resolution of 2cm to better highlight the results.

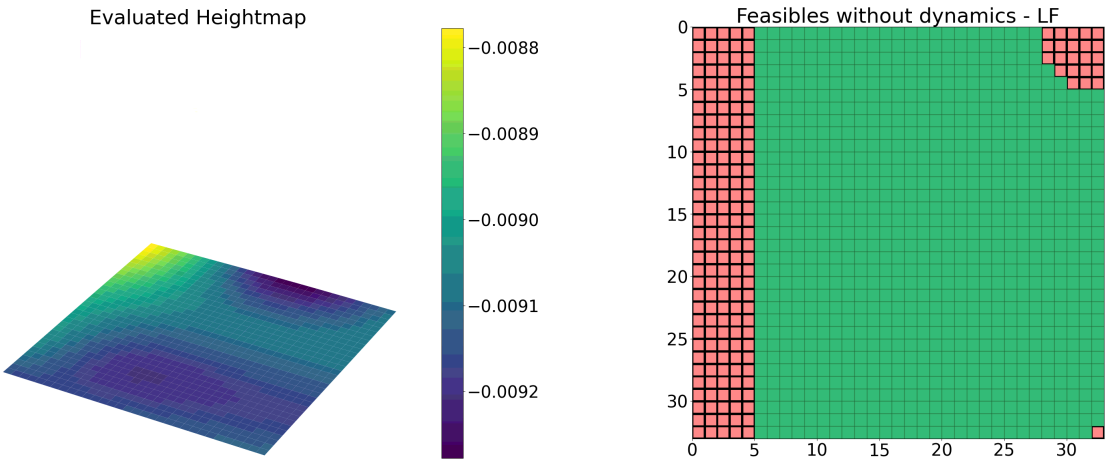


Figure 7.29: Heightmap acquired (left) is processed with the VFA algorithm (right) to discard footholds according to geometric-related aspects. Green squares mark the VFA-feasible footholds.

Without MM boundaries

In this first test we evaluate a heightmap without applying the boundaries described in Section 6.1.1. The evaluated heightmap is shown in figure 7.29, it is preprocessed with the VFA algorithm to discard all those footholds that are infeasible according to geometrical-related aspects. The sample evaluated in this test is the sample 3, related to the Left Front (LF) leg. We decided to change sample to emphasize the differences between the evaluation with and without moments boundaries. An important difference with respect to the previous case (crawl) is when we start the evaluation. Since trot is more dynamic, it is very difficult to track a trajectory for

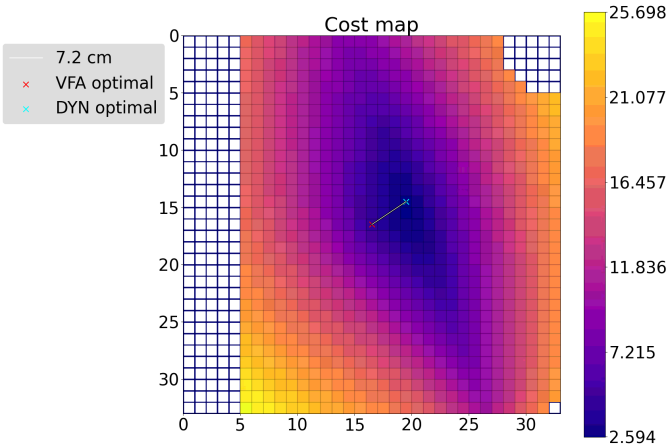


Figure 7.30: Output cost map. The white squares mark the footholds not evaluated or discarded. In this cost map the VFA and the DFE optimal footholds are different.

the short time that the rest of the swing represents. Instead, we take the current linear and angular velocities of the robot and assume that they will be the same for the rest of the swing to obtain the next landing location of the swinging feet. We then start the evaluation from the touchdown. This choice helps increasing the robustness of the evaluation since it evaluates the system when it is fully controllable, i.e., when we have all four legs on the ground.

The DFE outputs a cost map (Fig. 7.30) of the same dimension of the one that we provided. As we can see, the DFE has not discarded any foothold since the optimization term $\dot{\mathbf{L}}$ is not bounded and can take any value such that the constraints are verified. We will now plot the foothold with the highest cost to

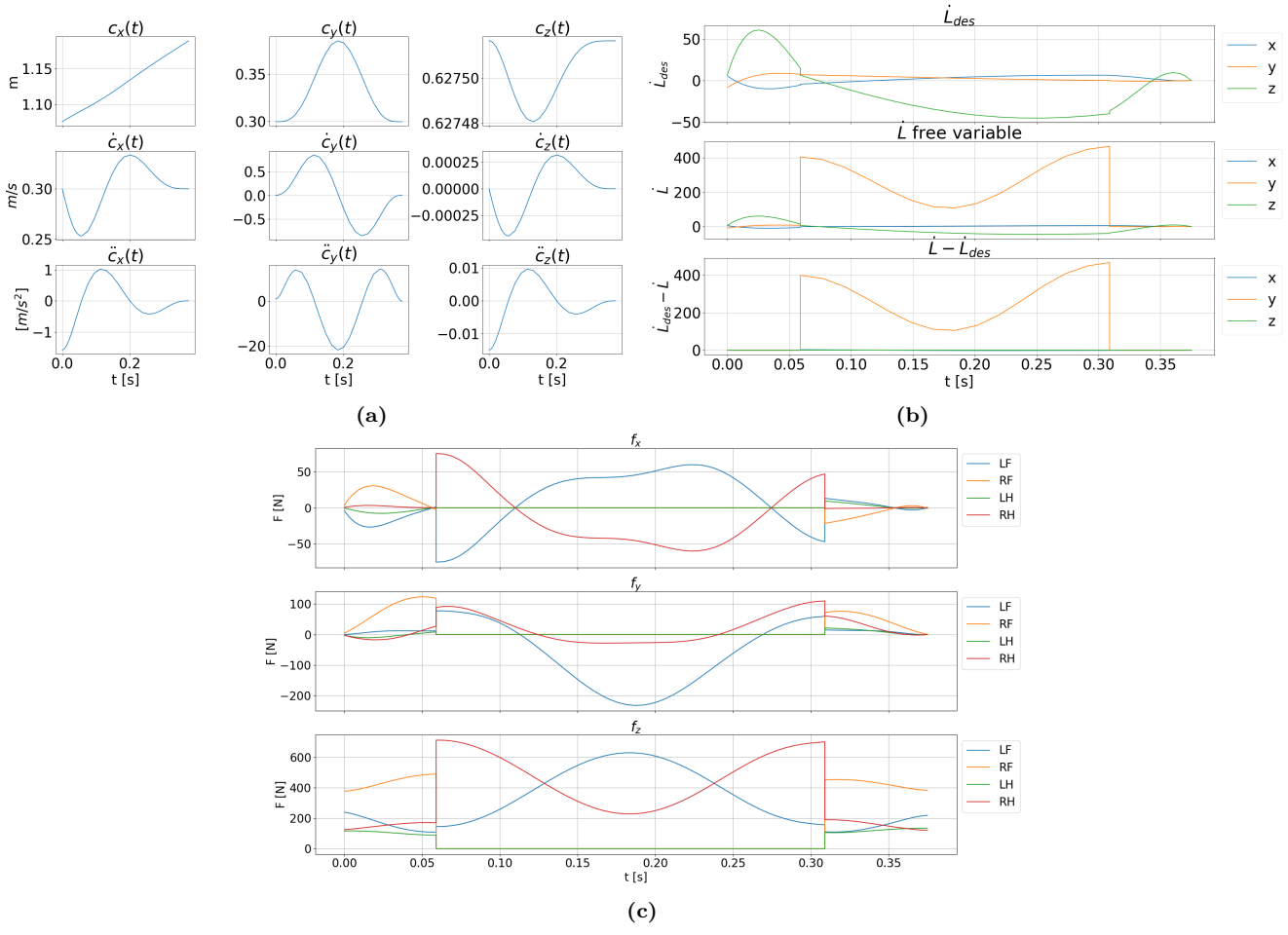


Figure 7.31: Plotting quantities relative to the highest cost foothold. In **a)** we can see the trajectory generated by the DFE, which is still biased toward having a straight trajectory. In **b)** the $\dot{\mathbf{L}}$ desired and optimized are shown. The difference between the two is not neglectable as it is an order of magnitude higher than the value of the desired one. In **c)** the forces required to achieve the motion are shown.

show the values assumed by the $\dot{\mathbf{L}}$ optimization variable. As we can see from 7.31b, $\dot{\mathbf{L}}$ assumes really high ($\simeq$ 400 Nm) values which might be physically infeasible. We will see that the foothold shown here is not feasible when applying the MM boundaries, as it exceeds the physical limits of the robot.

In the foothold plots (Fig. 7.31) the main characteristic that shows the importance of having physical boundaries on $\dot{\mathbf{L}}$, can be seen in the angular momentum rate plot, where the $\dot{\mathbf{L}}$ optimized variable reach values of an order of magnitude higher than the desired quantity. Although this makes the problem feasible, it might be not physically achievable by the robot if not properly bounded.

With MM boundaries

This test is carried out on the very same heightmap used in the previous test, but now introducing the MM boundaries. The cost map that the evaluator gives as output is depicted in figure 7.32. Here we can see how the set of feasible footholds is smaller than the previous case, having discarded those that are not able to yield an $\dot{\mathbf{L}}$ within the computed boundaries.

To show the effectiveness of the MM boundaries, we proceed to show the plots relative to the nearest feasible foothold with respect to the previous (now infeasible) shown. In figure 7.33b we can see how the angular momentum rate (here only $\dot{\mathbf{L}}$ is shown for the sake of clarity) is being limited by the MM boundaries. This is a limit case that shows the importance of having a physical limit on the optimization variable $\dot{\mathbf{L}}$ to have physically consistent results.

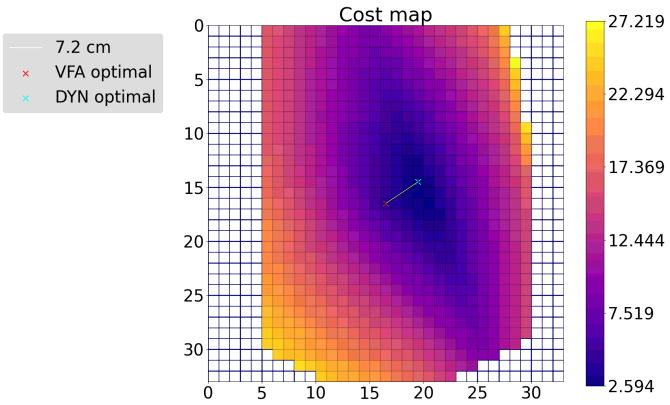


Figure 7.32: Output cost map. The white squares mark the footholds not evaluated or discarded. In this cost map the VFA and the DFE optimal footholds are the same.

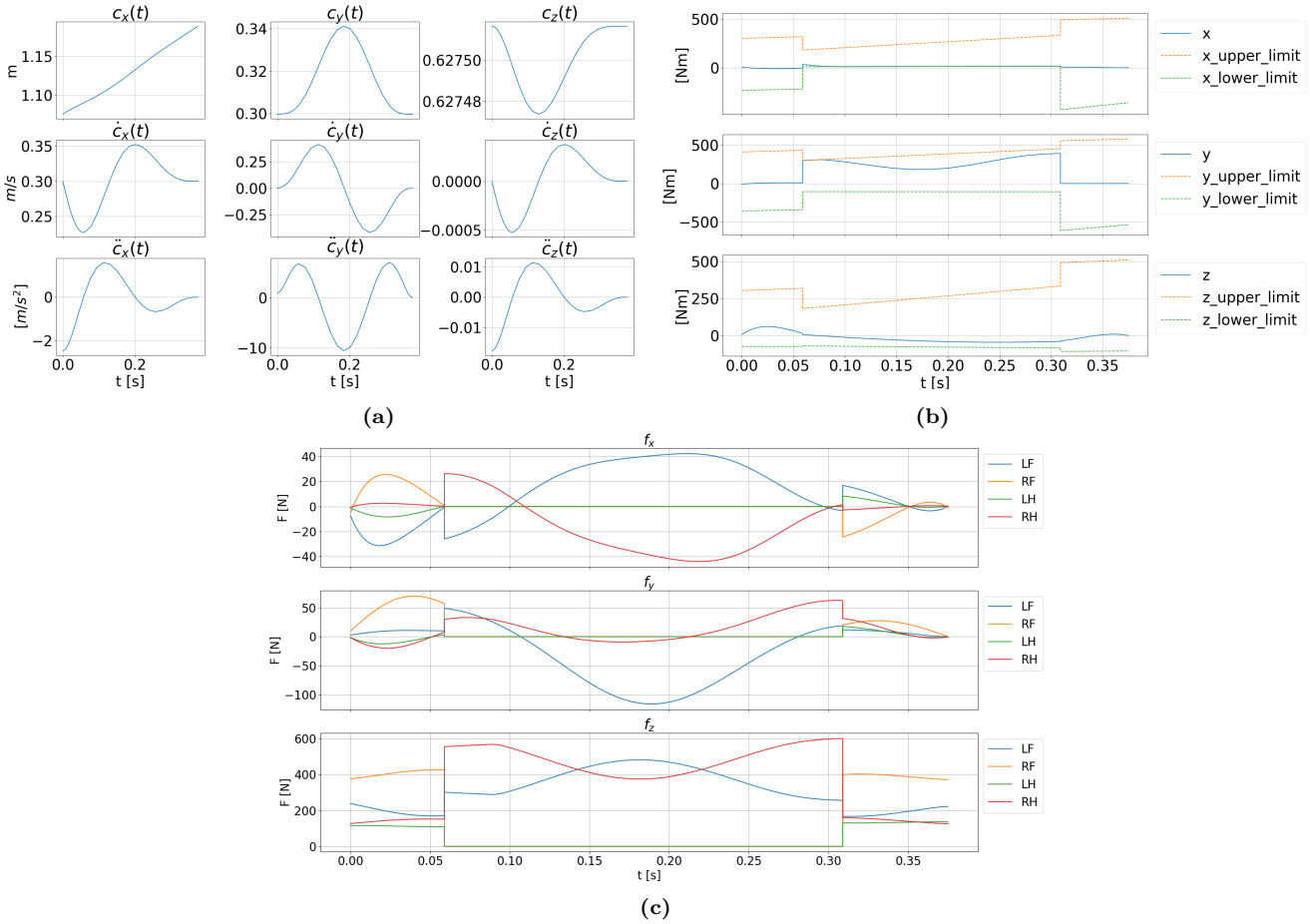


Figure 7.33: Plotting quantities relative to the highest cost foothold. In **a)** we can see the trajectory generated by the DFE, which is still biased toward having a straight trajectory. In **b)** only the $\dot{\mathbf{L}}$ optimization variable is shown, to better highlight how the boundaries are acting. The angular momentum rate is saturated by the boundaries, if still compatible with the motion. In **c)** the forces required to achieve the motion are shown..

The MM boundaries are physical limits that embed the real physical capabilities of the robot. Since they represent the maximum achievable moment by the robot, they are able to limit the angular momentum rate that the robot can achieve. If we consider the two output cost maps, we can see that the size of the feasible area in the vicinity of the nominal foothold is reduced. The further the foothold, the higher the joint torque required to apply a requested force in that specific point. This limit helps us in finding all those footholds that require a joint torque that exceed the maximum joint torque that the robot could achieve.

The cost maps shown so far, present a common characteristic on the distribution of the cost values. We can see how the area with footholds with lower costs can be found along a diagonal line that spans the entire map. Our hypothesis is that the

low cost area might be related to the support line, i.e., the line that connects the legs supporting the robot during swing. This might be related to the fact that as while crawling we have full control over both linear and angular quantities when the projection of the CoM lies within the support polygon, while trotting the tracking of both linear and angular quantities improves by exploiting the support line. As a future work we will investigate this relationship, to learn how to improve the tracking of a legged robot during high dynamic motions.

7.4 Discussions

In this last chapter we have seen how we were able to create a method to generate dynamically feasible CoM trajectories in a convex fashion. We have run tests in a Gazebo [56] simulation by employing the evaluator as a planner (DFEP, Section 7.1), showing that the trajectories are not only dynamically feasible, but also that the forces and $\dot{\mathbf{L}}$ computed with the DFEP predict with good accuracy the forces measured in simulation. In the three scenarios proposed, the DFEP has proven to be a reliable trajectory planner which relies on the actual states to compute a suitable trajectory to achieve the motion. It has proven to be also stable during long time horizons, leading to a stable crawl adapting the trajectory generated to the situation.

The optimization problem that the DFEP has to solve presents a high number of optimization variables, mainly due to the presence of $\mathbf{f}_e$ and $\dot{\mathbf{L}}$. The number of variables depend mainly on the number of discretization points considered. An approximation of the amount of optimization variables for a single optimization problem can be computed as:

$$n_{var} = n_{csw} * n_{samples}(\underbrace{(3 * n_c)}_{\mathbf{f}_{e,i}} + \underbrace{3}_{\dot{\mathbf{L}}_i}) \tag{7.8}$$

Where n_{csw} is the number of contact switches happening in the horizon considered, $n_{samples}$ is the number of discretization points used and n_c is the number of limbs (in our case $n_c = 4$ as HyQ is a quadruped).

Solving such optimization problem is time consuming (it takes about 0.35s) and computationally expensive, so it is still not suitable for real time applications. If we apply the DFEP to the problem of foothold selection (DFE, Section 7.3), we need

to solve the problem once per every foothold in the heightmap considered. This means that we need an amount of time that depend on the size of the heightmap and which takes too much time to be employed in real time as a foothold selection algorithm as it is.

Despite the slowness of the evaluation, the proposed method is able to generate a trajectory for more dynamic gaits (such as trot), where the condition $\dot{\mathbf{L}} = 0$ does not hold any more. By considering cases where $\dot{\mathbf{L}} \neq 0$ we can perform foothold selection for dynamic gaits.

It cannot be tested in simulation because of the long time required to compute a single heightmap. In the case of trot, where two legs move at the same time, we would have to consider two heightmaps at the same time, further increasing the evaluation times. To overcome the problem of real time, we intend to use the output of the DFE to train a Neural Network to perform foothold selection in shorter times, suitable for real time.

Chapter 8

Conclusions and future work

In this thesis we presented a method for foothold adaptation that considers the evolution of the robot dynamics to choose a suitable landing location for the feet. We formulated an evaluation method for potential footholds based on the concept of transition feasibility, which determines the existence of a dynamically feasible trajectory between a set of initial and final conditions. We used this concept to formulate an optimization problem. Said problem is highly nonlinear and can be highly computationally expensive to solve and prone to getting stuck in local minima. To cope with this, we reformulated the optimization as a convex problem, based on [40]. The method relies on two main tools: the use of a simplified model to describe the robot dynamics, namely, the Single Rigid Body Dynamics Model, and the description of the trajectory of the CoM as a Bézier curve. Since the method presented in [40] assumes zero rate of change of the angular momentum rate ($\dot{\mathbf{L}} = 0$), the method is not able to find solutions for highly dynamic motions, mainly due to the underactuation condition of the system. We extended this method to consider time varying angular momentum rates based on a desired trajectory for the angular quantities that describe the motion of the base (base orientation, angular rate and angular acceleration) and included a tracking cost in the optimization problem to minimize the deviations from this desire trajectory. This resulted in the possibility to track values of $\dot{\mathbf{L}}$ without compromising the feasibility of the problem during dynamic motions with underactuated dynamics such as trot. Furthermore, to prevent the optimization to provide solutions with non physically consistent values of the angular momentum rate $\dot{\mathbf{L}}$, we introduced the concept of Maximum Moments (MM), defined as the maximum moment that

the robot can counteract in a given configuration. We used the MM boundaries to limit the value of the angular momentum rate $\dot{\mathbf{L}}$ coming out of the optimization. Although we present this method applied to a quadruped, it is valid for any type of legged robot. We validated our proposed methods in two scenarios: using our evaluation as a motion planner (DFEP, Section 7.1) and to evaluate heightmap data collected from simulation (DFE, Section 7.3).

In the case of using it as planner we used the DFEP to generate reference trajectories for a Whole Body Controller [55] (WBC) which used such references to drive the robot in simulation. The references passed to the WBC are only the desired trajectory, velocity, orientation, and angular velocity. We compared the forces and $\dot{\mathbf{L}}$ computed using the DFEP and the forces and the $\dot{\mathbf{L}}$ acquired from simulation, seeing that the DFEP predicted with good accuracy the behaviour measured in simulation. This similarity and the fact that the robot was actually able to follow the desired trajectory, are proofs of the correctness of the DFEP.

In the case of the heightmap evaluator we showed how by solving multiple optimization problems it is possible to evaluate if there exist a dynamic transition between the initial and final state considered with a given foothold. It is possible to fully track the angular momentum rate $\dot{\mathbf{L}}$ in the case of slow and quasi-static motions, such as crawl. We built a cost function that was able to quantify the quality of the generated trajectory in terms of $\dot{\mathbf{L}}$ tracking, minimization of forces, distance of the foothold considered with respect to the nominal and deviation from a straight trajectory. We are able to choose an optimal foothold by the selecting the foothold with the lowest cost.

To include also more dynamic motions in the heightmap evaluation, i.e., motion that do not allow the complete tracking of both linear and angular quantities, such as trot, we designed boundaries to constrain $\dot{\mathbf{L}}$. Such boundaries are intended to define a physical limit on the maximum moment that the robot can counteract, basing them on the maximum joint torques available.

In this thesis we tried to address this problem finding a convex formulation that allowed to include different aspects of the locomotion. Although it has already yielded some promising results, it is far from its completeness, as we describe in the next section.

8.1 Future work

As stated so far this method takes into account dynamics to evaluate the transition feasibility for each foothold in a heightmap. Each foothold is evaluated by solving a convex optimization problem that embeds the equations of motion in form of constraints. Solving a convex problem requires time, more are the variables considered in the problem, more is the time required to solve the problem. At the moment of writing, solving each problem takes about 0.35s, which is a relatively small amount of time but it is not suitable for fast real-time applications. Considering that in a heightmap there are (worst case scenario, considering a 33×33 heightmap) an average of 980 footholds, to evaluate a full heightmap it would require around 5.7 minutes, which is definitely not suitable for high dynamical motions. If we consider also that when trotting there are two legs in the air at the same time, the time required for the evaluation increases as we need to evaluate both heightmaps at the same time, performing combinations between footholds (rear and front) to find the best configuration. As future work we want to compensate for the long computation time by using a Convolutional Neural Network (CNN) as it was done in [6]. However, training a NN while accounting for the robot dynamics requires a more sophisticated architecture with respect to the one shown in [6]. For example, the CNN shown in [6] only considers as input a heightmap, since it only deals with geometrical properties (kinematics and terrain morphology), however our inputs to the NN should also include the robot states and very likely not a single heightmap, as it has been shown in previous examples of NN [41]. Acquiring data from simulation, we can generate a data set that will be used to train the neural network drastically reducing the computation times and making it suitable for real time.

Once the algorithm is suitable for real time, we plan to test it in experiments using the HyQ platform [48] for both versions, i.e., as a planner (DFEP) and as a foothold adaptation criterion (DFE). The step right after making it suitable for real time is to increase the evaluation frequency, making it to evaluate continuously at each control loop.

Another future implementation planned is to use the algorithm in a hybrid way, using it for both selecting footholds and as a motion planner. Once the foothold has been selected using this method, the corresponding computed trajectories will be used as a reference, to better track the predicted dynamical behaviour. In this way

we can achieve more efficient trajectories, according to the cost function considered.

Finally, a probable direction in which to orient the future work is toward a nonlinear problem formulation. Solving directly a nonlinear problem might improve the computation of $\dot{\mathbf{L}}$ by including directly the nonlinear formulation (5.40) in function of angular quantities. This would allow us to compute directly the angular quantities required to achieve the angular momentum rate needed for the motion considered. Although solving a nonlinear problem is a more complex task and might take more time to solve, implementing it using a NN will reduce drastically the time required to find a solution.

Appendix A

Bézier curve

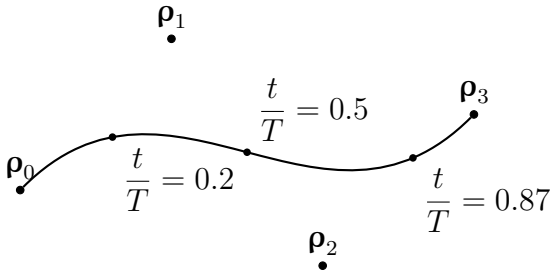


Figure A.1: 2D example of a generic Bézier curve with normalized time as parametric variable.

A Bézier curve is a parametric curve composed by a *Bernstein polynomial* (Figure A.1). A generic Bézier curve in a generic $\mathbb{R}^m$ space is a curve defined by a certain number of *control points* ($\boldsymbol{\rho} \in \mathbb{R}^m$), that can be seen as the analogue of coefficients for simple polynomial functions. A generic curve in a generic parametric variable u is defined as:

$$\mathbf{B}(u) = \sum_{i=0}^{n} b_{n,i}(u) \cdot \boldsymbol{\rho}_i \tag{A.1}$$

Where $\boldsymbol{\rho}_i$ is the ith control point, n is the *order* of the curve which is the analogue of the degree of a polynomial, and $b_{n,i}(u)$ is the *Bernstein basis polynomial*. It is defined as:

$$b_{n,i}(u) = \binom{n}{i} u^i (1-u)^{n-i} \tag{A.2}$$

The relation between number of control points (n_{cs}) and order of the curve is

the same that connects the number of coefficients and the degree of a polynomial function:

$$n_{cs} = n + 1 \tag{A.3}$$

We can write a generic Bézier equation by expanding (A.1):

$$\mathbf{B}(u) = b_{n,0}\boldsymbol{\rho}_0 + b_{n,1}\boldsymbol{\rho}_1 + \ldots + b_{n,n}\boldsymbol{\rho}_n \tag{A.4}$$

Combining (A.2) with (A.4) we can write a full expression for a generic Bézier curve of order n with $n+1$ control points:

$$\mathbf{B}(u) = \binom{n}{0}(1-u)^n\boldsymbol{\rho}_0 + \binom{n}{1}u(1-u)^{n-1}\boldsymbol{\rho}_1 \ldots \binom{n}{n}u^n\boldsymbol{\rho}_n \tag{A.5}$$

Analogously to the coefficients of a polynomial function, we can associate a condition to each control point of the curve. In order to correctly use a Bézier function with real quantities, the parametric variable ($u \in [0,1]$) has to be connected to real time. To do this, we can apply the substitution $u = t/T$, where $t \in [t_0, t_f]$ is a generic time instant and $T = t_f - t_0$ is the total period of time considered. The ratio t/T is a time normalization transformation. The Bézier curve expressed is now expressed in a normalized time domain (Figure A.1).

The same procedure described for the polynomial function can be applied to the Bézier curves as well to obtain the control points coordinates. Considering the initial time instant $t_0 = 0,\ u(t = t_0) = 0$ and the final time instant $t_f = T,\ u(t = t_f) = 1$ without loss of generality, we can compute the control points as follows:

$$\mathbf{B}(u) = \binom{7}{0}(1-u)^7\boldsymbol{\rho}_0 + \binom{7}{1}u(1-u)^6\boldsymbol{\rho}_1 + \ldots + \binom{7}{7}(u)^7\boldsymbol{\rho}_7 \tag{A.6}$$

$$\mathbf{B}(0) = \underbrace{\binom{7}{0}}_{1}\boldsymbol{\rho}_0 = \mathbf{x}_0 \longrightarrow \boldsymbol{\rho}_0 = \mathbf{x}_0 \tag{A.7}$$

$$\mathbf{B}(1) = \underbrace{\binom{7}{7}}_{1}\boldsymbol{\rho}_7 = \mathbf{x}_f \longrightarrow \boldsymbol{\rho}_7 = \mathbf{x}_f \tag{A.8}$$

Where $\mathbf{x}_0$ and $\mathbf{x}_f$ are two generic set of states (initial and final). Here we consider them as position and their derivatives are considered velocity and acceleration

respectively.

Derivatives

To compute the 6 control points left, we need to derive (A.6) to obtain the expressions for the velocity, acceleration and jerk. A Bézier curve's derivative is still a Bézier curve of a lower order. All the Bézier curve's derivatives depend on the control points of the original curve. The following analytical procedure can be exploited to compute the derivative of a Bézier curve.

The generic expressions for a k^{th} Bézier curve derivative and its control points are:

$$\mathbf{B}^k(u) = \sum_{i=0}^{n-k} b_{n-k,i}(u)\boldsymbol{\rho}_i^k \tag{A.9}$$

$$\boldsymbol{\rho}_i^k = (n-k)\cdot(\boldsymbol{\rho}_{i+1}^{k-1} - \boldsymbol{\rho}_i^{k-1}) \tag{A.10}$$

Consider $\mathbf{B}(u)$ as a Bézier curve of a generic order n. The derivative of $\mathbf{B}(u)$ with respect to u is defined as $\dot{\mathbf{B}}(u)$ and can then be computed as:

$$\dot{\mathbf{B}}(u) = \sum_{i=0}^{n-1} b_{n-1,i}(u)\hat{\boldsymbol{\rho}}_i \tag{A.11}$$

Where the ($\wedge$) symbol has been used to represent the control points relative to the first-order derivative of the Bézier curve, expressed as:

$$\hat{\boldsymbol{\rho}}_i = n\cdot(\boldsymbol{\rho}_{i+1} - \boldsymbol{\rho}_i) \tag{A.12}$$

The same process can be applied to any kth order derivative. For the 2^{nd} and the 3^{rd} derivative we have:

$$\ddot{\mathbf{B}}(u) = \sum_{i=0}^{n-2} b_{n-2,i}(u)\tilde{\boldsymbol{\rho}}_i \tag{A.13}$$

$$\dddot{\mathbf{B}}(u) = \sum_{i=0}^{n-3} b_{n-3,i}(u)\hat{\tilde{\boldsymbol{\rho}}}_i \tag{A.14}$$

Where the $\sim$ symbol represents the control points for the second derivative. The

Initial	Final
$\boldsymbol{\rho}_0 = \mathbf{x}_{p,0}$	$\boldsymbol{\rho}_7 = \mathbf{x}_{p,f}$
$\boldsymbol{\rho}_1 = \frac{\mathbf{x}_{v,0} \cdot T}{n} + \boldsymbol{\rho}_0$	$\boldsymbol{\rho}_6 = \boldsymbol{\rho}_7 - \frac{\mathbf{x}_{v,f} \cdot T}{n}$
$\boldsymbol{\rho}_2 = \frac{\mathbf{x}_{a,0} \cdot T^2}{n(n-1)} + 2\boldsymbol{\rho}_1 - \boldsymbol{\rho}_0$	$\boldsymbol{\rho}_5 = \frac{\mathbf{x}_{a,f} \cdot T^2}{n(n-1)} + 2\boldsymbol{\rho}_6 - \boldsymbol{\rho}_7$
$\boldsymbol{\rho}_3 = \frac{\mathbf{x}_{j,0} \cdot T^3}{n(n-1)(n-2)} + 3\boldsymbol{\rho}_2 - 3\boldsymbol{\rho}_2 + \boldsymbol{\rho}_0$	$\boldsymbol{\rho}_4 = \boldsymbol{\rho}_7 - 3\boldsymbol{\rho}_6 + 3\boldsymbol{\rho}_5 - \frac{\mathbf{x}_{j,f} \cdot T^3}{n(n-1)(n-2)}$

Table A.1: Computation example of the curve's control points.

control points can be computed as:

$$\tilde{\boldsymbol{\rho}}_i = (n-1) \cdot (\hat{\boldsymbol{\rho}}_{i+1} - \hat{\boldsymbol{\rho}}_i) \tag{A.15}$$

$$\hat{\tilde{\boldsymbol{\rho}}}_i = (n-2) \cdot (\tilde{\boldsymbol{\rho}}_{i+1} - \tilde{\boldsymbol{\rho}}_i) \tag{A.16}$$

Now that all the required derivatives expressions have been obtained, we can compute the control points by imposing initial and final conditions as done in (A.7) and (A.8). The control points obtained are reported in Table A.1.

Bibliography

[1] A. Mahapatra, S. S. Roy, and D. K. Pratihar. «Multi-Legged Robots—A Review». In: *Multi-body Dynamic Modeling of Multi-legged Robots*. Singapore: Springer Singapore, 2020, pp. 11–32. ISBN: 978-981-15-2953-5. DOI: 10.1007/978-981-15-2953-5_2.

[2] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng. «A Control Architecture for Quadruped Locomotion over Rough Terrain». In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 811–818.

[3] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. «Learning Locomotion over Rrough Terrain using Terrain Templates». In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 167–172.

[4] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal. «Compliant Quadruped Locomotion Over Rough Terrain». In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 814–820.

[5] M. Camurri, S. Bazeille, D.G. Caldwell, and C. Semini. «Real-time Depth and Inertial Fusion for Local SLAM on Dynamic Legged Robots». In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2015, pp. 259–264. DOI: 10.1109/MFI.2015.7295818.

[6] O. A. Villarreal Magaña et al. «Fast and Continuous Foothold Adaptation for Dynamic Locomotion Through CNNs». In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2140–2147.

[7] S.M. Song and K. J. Waldron. «Machines That Walk: The Adaptive Suspension Vehicle». In: Cambridge, MA, USA: MIT Press, 1988.

[8] Boston Dynamics. *Do you love me?* URL: https://www.youtube.com/watch?v=fn3KWM1kuAw&ab_channel=BostonDynamics.

[9] R.B. McGhee and A.A. Frank. «On the stability properties of quadruped creeping gaits». In: *Mathematical Biosciences* 3 (1968), pp. 331–351. ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(68)90090-4. URL: https://www.sciencedirect.com/science/article/pii/0025556468900904.

[10] M. Vukobratovic and D. Juricic. «Contribution to the Synthesis of Biped Gait». In: *IEEE Transactions on Biomedical Engineering* BME-16.1 (1969), pp. 1–6. DOI: 10.1109/TBME.1969.4502596.

[11] S. Caron, Q. Pham, and Y. Nakamura. «ZMP Support Areas for Multicontact Mobility Under Frictional Constraints». In: *IEEE Transactions on Robotics* 33.1 (2017), pp. 67–80. DOI: 10.1109/TRO.2016.2623338.

[12] Ludovic D. Maes, Marc Herbin, Rémi Hackert, Vincent L. Bels, and Anick Abourachid. «Steady locomotion in dogs: temporal and associated spatial coordination patterns and the effect of speed». In: *Journal of Experimental Biology* 211.1 (2008), pp. 138–149. ISSN: 0022-0949. DOI: 10.1242/jeb.008243. eprint: https://jeb.biologists.org/content/211/1/138.full.pdf. URL: https://jeb.biologists.org/content/211/1/138.

[13] M H Raibert. «Legged robots that balance». In: (Jan. 1985). URL: https://www.osti.gov/biblio/5606728.

[14] Auke Jan Ijspeert. «Central pattern generators for locomotion control in animals and robots: A review». In: *Neural Networks* 21.4 (2008). Robotics and Neuroscience, pp. 642–653. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2008.03.014. URL: https://www.sciencedirect.com/science/article/pii/S0893608008000804.

[15] Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. «CPG model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators». In: *Robotics and Autonomous Systems* 44.3 (2003). Best papers presented at IAS-7, pp. 171–179. ISSN: 0921-8890. DOI: https://doi.org/10.1016/S0921-8890(03)00067-8. URL: https://www.sciencedirect.com/science/article/pii/S0921889003000678.

[16] P. Arena, L. Fortuna, M. Frasca, and G. Sicurella. «An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.4 (2004), pp. 1823–1837. DOI: 10.1109/TSMCB.2004.828593.

[17] L. Righetti and A.J. Ijspeert. «Pattern generators with sensory feedback for the control of quadruped locomotion». In: June 2008, pp. 819 –824. ISBN: 978-1-4244-1646-2. DOI: 10.1109/ROBOT.2008.4543306.

[18] A. Badri-Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A.J. Ijspeert. «Towards Dynamic Trot Gait Locomotion: Design, Control, and Experiments with Cheetah-cub, a Compliant Quadruped Robot». In: *The International Journal of Robotics Research* 32 (July 2013). DOI: 10.1177/0278364913489205.

[19] L. Righetti and Auke Jan Ijspeert. «Programmable central pattern generators: an application to biped locomotion control». In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 2006, pp. 1585–1590. DOI: 10.1109/ROBOT.2006.1641933.

[20] R. HÉliot and B. Espiau. «Multisensor Input for CPG-Based Sensory—Motor Coordination». In: *IEEE Transactions on Robotics* 24.1 (2008), pp. 191–195. DOI: 10.1109/TRO.2008.915433.

[21] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini. «STANCE: Locomotion Adaptation over Soft Terrain». In: *IEEE Trans. Robot. (T-RO)* 36.2 (2020), pp. 443–457. DOI: 10.1109/TRO.2019.2954670.

[22] C. Mastalli et al. «Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion». In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1096–1103. DOI: 10.1109/ICRA.2017.7989131.

[23] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. «Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization». In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1560–1567. DOI: 10.1109/LRA.2018.2798285.

[24] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard. «An Efficient Acyclic Contact Planner for Multiped Robots». In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 586–601. DOI: 10.1109/TRO.2018.2819658.

[25] Y. Lin, B. Ponton, L. Righetti, and D. Berenson. «Efficient Humanoid Contact Planning using Learned Centroidal Dynamics Prediction». In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5280–5286. DOI: 10.1109/ICRA.2019.8794032.

[26] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard. «An Efficient Acyclic Contact Planner for Multiped Robots». In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 586–601. DOI: 10.1109/TRO.2018.2819658.

[27] D. Belter, J. Bednarek, H. Lin, G. Xin, and M. Mistry. «Single-shot Foothold Selection and Constraint Evaluation for Quadruped Locomotion». In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7441–7447.

[28] L. Chen et al. «CNNs based Foothold Selection for Energy-Efficient Quadruped Locomotion over Rough Terrains». In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 1115–1120.

[29] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter. «Perceptive Locomotion in Rough Terrain – Online Foothold Optimization». In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5370–5376.

[30] O. A. Villarreal Magaña, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini. *MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion*. 2019. arXiv: 1909.13842 [cs.RO].

[31] J. Bednarek et al. «CNN-based Foothold Selection for Mechanically Adaptive Soft Foot». In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10225–10232. DOI: 10.1109/IROS45743.2020.9340910.

[32] D. Orin, A. Goswami, and S. Lee. «Centroidal Dynamics of a Humanoid Robot». In: *Autonomous Robots* 35 (Oct. 2013). DOI: 10.1007/s10514-013-9341-4.

[33] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli. «Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds». In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1502–1509.

[34] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. «Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization». In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1560–1567.

[35] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E.R. De Pieri, and D.G. Caldwell. «A Reactive Controller Framework for Quadrupedal Locomotion on Challenging Terrain». In: May 2013, pp. 2554–2561. DOI: 10.1109/ICRA.2013.6630926.

[36] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. «Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control». In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594448.

[37] R. Grandia, A. Taylor, A. Ames, and M. Hutter. *Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control*. Nov. 2020. DOI: 10.13140/RG.2.2.17776.89605.

[38] M. Hutter et al. «ANYmal - a highly mobile and dynamic quadrupedal robot». In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 38–44. DOI: 10.1109/IROS.2016.7758092.

[39] Y. Lin, L. Righetti, and D. Berenson. «Robust Humanoid Contact Planning With Learned Zero- and One-Step Capturability Prediction». In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2451–2458. DOI: 10.1109/LRA.2020.2972825.

[40] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx. «C-CROC: Continuous and Convex Resolution of Centroidal Dynamic Trajectories for Legged Robots in Multicontact Scenarios». In: *IEEE Transactions on Robotics* PP (Jan. 2020), pp. 1–16. DOI: 10.1109/TRO.2020.2964787.

[41] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter. «DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning». In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3699–3706. DOI: 10.1109/LRA.2020.2979660.

[42] S. Bazeille, M. Camurri, J. Ortiz, I. Havoutis, D.G. Caldwell, and C. Semini. «Terrain mapping with a pan and tilt stereo camera for locomotion on a quadruped robot». In: *ICRA14 Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots (WMEPC14)*. 2014. DOI: 10.1007/s11370-014-0147-9.

[43] M. Focchi et al. «Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality». In: *Advances in Robotics Research: From Lab to Market: ECHORD++: Robotic Science Supporting Innovation*. Springer International Publishing, 2020, pp. 165–209. ISBN: 978-3-030-22327-4. DOI: 10.1007/978-3-030-22327-4_9. URL: https://doi.org/10.1007/978-3-030-22327-4_9.

[44] C. Semini, N.G. Tsagarakis, B. Vanderborght, Y. Yang, and D.G. Caldwell. «HyQ – Hydraulically Actuated Quadruped Robot: Hopping Leg Prototype». In: *Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. 2008, pp. 593–599. DOI: 10.1109/BIOROB.2008.4762913.

[45] C. Semini, N.G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D.G. Caldwell. «Design of HyQ - a Hydraulically and Electrically Actuated Quadruped Robot». In: *IMechE Part I: Journal of Systems and Control Engineering* 225.6 (2011), pp. 831–849. DOI: 10.1177/0959651811402275.

[46] B.U. Rehman, M. Focchi, M. Frigerio, J. Goldsmith, D.G. Caldwell, and C. Semini. «Design of a Hydraulically Actuated Arm for a Quadruped Robot». In: *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR)*. 2015. URL: http://www.worldscientific.com/worldscibooks/10.1142/9820.

[47] C. Semini et al. «Design of the Hydraulically-Actuated, Torque-Controlled Quadruped Robot HyQ2Max». In: *IEEE/ASME Transactions on Mechatronics* PP.99 (2016), pp. 1–1. ISSN: 1083-4435. DOI: 10.1109/TMECH.2016.2616284.

[48] C. Semini et al. «Brief introduction to the quadruped robot HyQReal». In: *Italian Conference on Robotics and Intelligent Machines (I-RIM)*. Rome, 2019, pp. 1–2.

[49] M. Camurri. «Multisensory State Estimation and Mapping on Dynamic Legged Robots». PhD thesis. Istituto Italiano di Tecnologia (IIT) and University of Genova, 2017.

[50] S. Nobili et al. «Heterogeneous Sensor Fusion for Accurate State Estimation of Dynamic Legged Robots». In: *Proceedings of Robotics: Science and Systems*. Boston, USA, 2017.

[51] T. Koolen, 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». In: *The International Journal of Robotics Research* 31 (July 2012), pp. 1094–1113. DOI: 10.1177/0278364912452673.

[52] Octavio Villarreal. «Bridging Vision and Dynamic Legged Locomotion». PhD thesis. Istituto Italiano di Tecnologia (IIT) and University of Genova, 2020.

[53] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 354023957X.

[54] K. Fukuda and A. Prodon. «Double Description method Revisited». In: *Combinatorics and Computer Science*. Ed. by Michel Deza, Reinhardt Euler, and Ioannis Manoussakis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 91–111.

[55] M. Focchi, del A. Prete, I. Havoutis, R. Featherstone, D.G. Caldwell, and C. Semini. «High-slope terrain locomotion for torque-controlled quadruped robots». In: *Autonomous Robots* 41.1 (2017), pp. 259–272. ISSN: 1083-4435. DOI: 10.1007/s10514-016-9573-1. URL: http://dx.doi.org/10.1007/s10514-016-9573-1.

[56] N. Koenig and A. Howard. «Design and use paradigms for Gazebo, an open-source multi-robot simulator». In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[57] A. Koubaa. *Robot Operating System (ROS): The Complete Reference (Volume 1)*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319260529.