Online Optimization-based Gait Adaptation of Quadruped Robot Locomotion

Angelo Bratta

Istituto Italiano di Tecnologia, Italy
Università degli Studi di Genova, Italy

Thesis submitted for the degree of:
Doctor of Philosophy (35^{th} *cycle*)

December 21, 2022

Angelo Bratta
Online Optimization-based Gait Adaptation of Quadruped Robot Locomotion
Candidate student for the *PhD Program in Bioengineering and Robotics*
Curriculum: *Advanced and Humanoid Robotics*
Genoa, Italy - December 2022

Tutor:
Dr. Michele Focchi
Dynamic Legged Systems Lab.,
Istituto Italiano di Tecnologia

Co-Tutor:
Dr. Claudio Semini
Dynamic Legged Systems Lab.,
Istituto Italiano di Tecnologia

Head of the PhD program:
Prof. Paolo Massobrio
Dipartamento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi,
Università degli Studi di Genova

Reviewers:
Prof. Paolo Pietro Arena
Dipartimento di Ingegneria Elettrica Elettronica e Informatica,
Università degli Studi di Catania

Dr. Nicolas Mansard
LAAS- CNRS,
Université de Toulouse

Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

December 21, 2022 Angelo Bratta

Abstract

Quadruped robots demonstrated extensive capabilities of traversing complex and unstructured environments. Optimization-based techniques gave a relevant impulse to the research on legged locomotion. Indeed, by designing the cost function and the constraints, we can guarantee the feasibility of a motion and impose high-level locomotion tasks, *e.g.,* tracking of a reference velocity. This allows one to have a generic planning approach without the need to tailor a specific motion for each terrain, as in the heuristic case. In this context, Model Predictive Control (MPC) can compensate for model inaccuracies and external disturbances, thanks to the high-frequency replanning.

The main objective of this dissertation is to develop a Nonlinear MPC (NMPC)-based locomotion framework for quadruped robots. The aim is to obtain an algorithm which can be extended to different robots and gaits; in addition, I sought to remove some assumptions generally done in the literature, *e.g.,* heuristic reference generator and user-defined gait sequence.

The starting point of my work is the definition of the Optimal Control Problem to generate feasible trajectories for the Center of Mass. It is descriptive enough to capture the linear and angular dynamics of the robot as a whole. A simplified model (Single Rigid Body Dynamics model) is used for the system dynamics, while a novel cost term maximizes leg mobility to improve robustness in the presence of nonflat terrain. In addition, to test the approach on the real robot, I dedicated particular effort to implementing both a heuristic reference generator and an interface for the controller, and integrating them into the controller framework developed previously by other team members.

As a second contribution of my work, I extended the locomotion framework to deal with a trot gait. In particular, I generalized the reference generator to be based on optimization. Exploiting the Linear Inverted Pendulum model, this new module can deal with the under-actuation of the trot when only two legs are in contact with the ground, endowing the NMPC with physically informed reference trajectories to be tracked. In addition, the reference

velocities are used to correct the heuristic footholds, obtaining contact locations coherent with the motion of the base, even though they are not directly optimized.

The model used by the NMPC receives as input the gait sequence, thus with the last part of my work I developed an online multi-contact planner and integrated it into the MPC framework. Using a machine learning approach, the planner computes the best feasible option, even in complex environments, in a few milliseconds, by ranking online a set of discrete options for footholds, i.e., which leg to move and where to step. To train the network, I designed a novel function, evaluated offline, which considers the value of the cost of the NMPC and robustness/stability metrics for each option.

These methods have been validated with simulations and experiments over the three years. I tested the NMPC on the Hydraulically actuated Quadruped robot (HyQ) of the IIT's Dynamic Legged Systems lab, performing omni-directional motions on flat terrain and stepping on a pallet (both static and relocated during the motion) with a crawl gait. The trajectory replanning is performed at high-frequency, and visual information of the terrain is included to traverse uneven terrain. A Unitree Aliengo quadruped robot is used to execute experiments with the trot gait. The optimization-based reference generator allows the robot to reach a fixed goal and recover from external pushes without modifying the structure of the NMPC. Finally, simulations with the Solo robot are performed to validate the neural network-based contact planning. The robot successfully traverses complex scenarios, e.g., stepping stones, with both walk and trot gaits, choosing the footholds online.

The achieved results improved the robustness and the performance of the quadruped locomotion. High-frequency replanning, dealing with a fixed goal, recovering after a push, and the automatic selection of footholds could help the robots to accomplish important tasks for the humans, for example, providing support in a disaster response scenario or inspecting an unknown environment.

In the future, the contact planning will be transferred to the real hardware. Possible developments foresee the optimization of the gait timings, *i.e.,* stance and swing duration, and a framework which allows the automatic transition between gaits.

Keywords : Legged Robots, Optimization, Trajectory Generation, Model Predictive Control, Neural Network.

Table of Contents

List of Figures

List of Tables

Glossary

Notation

x, α	scalars
$\mathbf{x}, \boldsymbol{\alpha}$	vectors
$\mathbb{R}, \mathbb{N}$	sets
$\mathbf{A}, \mathbf{B}$	matrices
$\mathbf{A}^\top$	matrix transpose
$\mathbf{A}^{-1}$	matrix inverse
$\mathbf{1}_n$	$n \times n$ identity matrix
$\mathbf{0}_n$	$n \times n$ zero matrix

Symbols

$\boldsymbol{\delta}$	sequence of gait status
$\boldsymbol{\omega}$	angular velocity
$\boldsymbol{\Phi}$	CoM orientation
D	duty factor
$\mathbf{h}$	hip location
$\mathbf{I}$	inertia tensor
$\mathbf{k}$	centroidal angular momentum
m_f	mobility factor
n_x	number of states
n_u	number of control inputs
n_p	number of model parameters
n_j	number of joints
n_a	number of actions
N	NMPC horizon

N_g	reference horizon
N_s	step horizon
o	offset
$\mathbf{p}_\mathrm{c}^\mathrm{g}$	predicted $X-Y$ CoM position by LIP optimization
$\bar{\mathbf{p}}_\mathrm{c}$	average $X-Y$ CoM position
$\mathbf{p}_\mathrm{f}$	Footholds sequence
$\mathbf{p}_\mathrm{n}$	position of neutral point
$\ddot{\mathbf{p}}_\mathrm{c}$	CoM acceleration
$\mathbf{q}_\mathrm{j}$	joint position
$\dot{\mathbf{q}}_\mathrm{j}$	joint velocity
$\mathbf{s}$	slack variables
T_f	response horizon
T_s	sampling time
T_c	cycle time
T_st	duration of the stance phase
T_sw	duration of the swing phase
$\mathbf{u}_\mathrm{r}$	input of the ContactNet
$\mathbf{u}^\mathrm{qp}$	GRFs computed by the QP Mapping
$\mathbf{u}^\mathrm{p}$	predicted GRFs by the NMPC
$\mathbf{v}_\mathrm{c}^\mathrm{g}$	predicted $X-Y$ CoM velocity by LIP optimization
$\mathbf{V}$	vectors in the dataset
$\hat{\mathbf{Y}}$	prediction of the ContactNet
$\mathbf{w}^\mathrm{g}$	ZMP position
$\mathbf{x}^\mathrm{p}$	predicted states by the NMPC
$\hat{\mathbf{x}}$	actual robot state
$\mathbf{x}^\mathrm{g}$	state of the optimized reference generator

Acronyms

CoM	Center of Mass
DDP	Differential Dynamic Programming
DoFs	Degrees of Freedom
GRFs	Ground Reaction Forces
HyQ	Hydraulically actuated Quadruped
HAA	Hip Adduction-Abduction
HFE	Hip Flexion-Extension
IP	Interior Point
KFE	Knee Flexion-Extension
LIP	Linear Inverted Pendulum
LF	Left Front
LH	Left Hind
LP	Linear Program
MBIP	Mixed-Binary-Integer Programming
MCTS	Monte Carlo Tree Search
MIP	Mixed-Integer Programming
MPC	Model Predictive Control
NLP	Nonlinear Program
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
PD	Proportional Derivative
QP	Quadratic Program
RF	Right Front
RH	Right Hind
RTI	Real Time Iteration
SLIP	Spring Loaded Inverted Pendulum
SQP	Sequential Quadratic Program
SRBD	Single Rigid Body Dynamics
TO	Trajectory Optimization
VFA	Visual Foothold Adaptation
WBC	Whole-Body Controller
ZMP	Zero-Moment Point

Chapter 1

Introduction

1.1 Motivation

Quadruped robots are versatile machines that use their sensors to navigate in complex scenarios by moving the trunk and applying forces on the ground. The developments in the hardware design and control algorithms allow the robots to obtain agile and stable motions even in complex environments. Thanks to these advancements, quadruped robots are increasingly being tested in real-world scenarios. In order words, they are moved out of our research laboratories.

The challenge of legged locomotion is represented by the generation of stable and robust trajectories to be followed. Optimization-based techniques gave a relevant impulse to the research on legged locomotion since we can guarantee the feasibility of a motion and impose high-level locomotion tasks, *e.g.,* tracking of a reference velocity. Researchers aim to design generic approaches, not tailored to a specific motion for each terrain, but that can react to external disturbances and changes in the environment. Among the different techniques, Model Predictive Control (MPC) is gaining momentum since they compute new trajectories online while the robot is moving.

Dealing with these concepts, the goal of this dissertation is to develop a Nonlinear MPC (NMPC)-based locomotion framework for quadruped robots for different prototypes and gaits. The developed methods focus at the same time on the computational efficiency to achieve high replanning frequency and on the accuracy of the approach to obtain reliable motions. In addition, this thesis sought to remove some assumptions generally done in the literature, *e.g.,* heuristic reference generator and user-defined gait sequence.

1.2 Contributions

The main novel contributions of this thesis are the following:

1. a 25 Hz NMPC framework which generates an omni-directional walk and rough terrain locomotion for the DLS lab's quadruped robot HyQ, taking into account the robot's mobility and the visual feedback from the environment.
   This contribution is validated both in simulations and experiments.

2. an optimization-based reference generator that provides the NMPC with physically-informed reference trajectories online. This module optimizes feasible paths for the CoM and coherent Ground Reaction Forces (GRFs) in the presence of the under-actuation of statically unstable gaits as the trot.
   Experiments and simulations are shown with the presented locomotion pipeline.

3. a machine learning-based contact planner which uses a neural network to rank a discretized set of possible actions and outputs the best feasible footholds, *i.e.,* which leg to move and where to step. Results are obtained in simulation.

Contribution 1 has been published in *IEEE Access* journal [Rathod et al., 2021].
Contribution 2 has been accepted for publication in the special issue *Legged Robots into the Real World* of *Robotics* journal [Bratta et al., 2022].
Contribution 3 is currently under review for publication [Bratta et al., 2023].

1.3 Outline

The rest of the dissertation is organized as follows:

- Chapter 2 summarizes the most relevant works on legged locomotion approaches, contact planners, and numerical optimization techniques.
- Chapter 3 introduces the locomotion framework and the NMPC.
- Chapter 4 presents the optimization-based reference generator.
- Chapter 5 describes the multi-contact online planner.
- Chapter 6 provides the final discussion and insight into future works.

Chapter 2

Related works

2.1 Legged Locomotion

In order to move, a robot needs to choose a trajectory for the motion of its base, and decide a feasible contact sequence for its feet to follow. In addition, its control poses complex problems related to the hybrid nature of the forces required to generate motion since the robot establishes and interrupts the contact between its feet and the ground. In this section, I introduce the relevant works in the literature for heuristics and optimization-based legged locomotion.

2.1.1 Heuristic Approaches

The term *heuristic* refers to a method of solving problems by finding practical ways of dealing with them. Heuristic approaches are not based on a mathematical formulation with formal proofs but rather on experience, intuition and practice. For example, [Raibert, 1986] devised an algorithm to control a 3D one-legged hopping machine. Raibert defined the *neutral point*, *i.e.,* the point about which the hopper has a symmetric motion of the body. The neutral point is computed as:

$$\mathbf{p}_{\mathrm{n}} = \mathbf{h} + \mathbf{v}_{\mathrm{c},(\mathrm{x,y})}\frac{T_{\mathrm{st}}}{2} + \mathrm{k}(\mathbf{v}_{\mathrm{cc},(\mathrm{x,y})} - \mathbf{v}^{\mathrm{d}}) \tag{2.1}$$

where:

- $\mathbf{p}_{\mathrm{n}} \in \mathbb{R}^2$ is the position of the neutral point on an horizontal plane
- $\mathbf{h} \in \mathbb{R}^2$ is the projection of the hip on the ground
- $\mathbf{v}_{\mathrm{c},(\mathrm{x,y})} \in \mathbb{R}^2$ is the X-Y component of CoM velocity

- $T_{\text{st}} \in \mathbb{R}$ is the duration of the stance phase
- $\text{k} \in \mathbb{R}$ is a gain
- $\mathbf{v}^{\text{d}} \in \mathbb{R}^2$ is the desired CoM velocity

Stepping on this point, the robot is able to keep walking at the forward velocity $\mathbf{v}^{\text{d}} \in \mathbb{R}^2$ without losing balance. Generalizing, an algorithm can use the neutral point to choose the footholds for multi-legged robots which move one leg at a time. To extend this idea to other gaits, [Raibert et al., 1986] designed a control system based on the concept of *virtual leg* [Sutherland and Ullner, 1984]: a pair of legs that lift and touch the ground in unison can be seen as a unique leg. In this way, a quadruped robot doing gaits such as trot, pace, or bound is equivalent to a biped robot. The control scheme is composed of three operations: 1) choosing the footholds of the feet that place the virtual feet in the desired location, 2) synchronizing the moment of the lift-off and touchdown of the members of the pair, and 3) computing the GRFs such that each leg delivers the same axial thrust to the ground.

Among the other examples of heuristic approaches, it is worth mentioning [Pratt et al., 1997] that presented the *virtual model control*, *i.e.,* a motion control which emulates virtual mechanical components to create the forces that are applied to the robot. This gives the possibility to tune the accuracy of the model, *e.g.,* using a simple spring or damper element in the task space [Khatib, 1987] or a more complex nonlinear element. The authors tested this approach successfully on Spring Turkey and Spring Flamingo [Pratt et al., 2001]. Simulations were also performed with a hexapod robot.

Important studies developed heuristic criteria to ensure the stability during the motion. The first one was proposed by McGhee and Frank and it refers to the *support polygon* [McGhee and Frank, 1968]: the robot is statically stable, *i.e.,* it does not fall if the projection of the CoM on the ground is inside the convex hull of the robot's feet. An example of support polygon is depicted in Figure 2.1. The relevance of the support polygon is indisputable. For example, the algorithm of the static crawl [Focchi et al., 2016, 2020] moves its base in the following support polygon (a triangle) before lifting a leg. The algorithm is thus a state-machine with four phases: move CoM, unload leg, swing leg, and load leg. The static crawl allows our HyQ to walk on flat terrain, but also inside a 50° inclined groove, as shown in Figure 2.1.

An extension of the support polygon led to the development of the Zero-Moment Point (ZMP)[Vukobratovic and Juricic, 1969], *i.e.,* the point on the ground where the total moment produced by the inertial and gravitational forces is equal to zero. Examples of works based on the ZMP are the simulation of bipedal walking on a spiral stairs achieved by a preview[1] tracking servo controller [Kajita et al., 2003] and the multicontact motion in a scenario with inclined platforms and wall with the humanoid HRP4 [Caron et al., 2017].

[1]The theory of the Preview Control was proposed by [Sheridan, 1966].

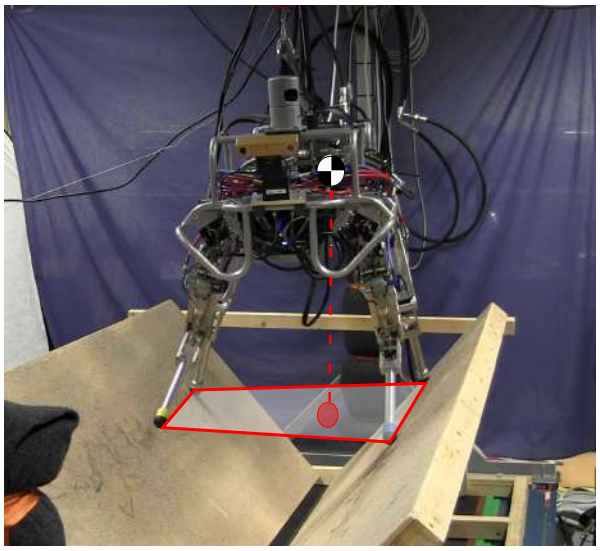

Figure 2.1: Picture of HyQ robot during a statically stable crawl locomotion inside a $50°$ inclined groove. The rectangle represents the *support polygon*, *i.e.,* the convex hull of the robot's feet. The black-and-white circle is the CoM. The robot is stable since the vertical projection of the CoM is inside the support polygon.

Heuristic approaches have proved their practical effectiveness by increasing locomotion autonomy, and they are still widely used in recent works but present some drawbacks:

- they are tailored for specific gaits;
- they cannot be easily generalized to any terrain and motion;
- they generally are reactive approaches, *e.g.,* [Focchi et al., 2013], so they do not foresee what will happen in the future but only react to what was happening in the past;

These reasons, combined with the improvements in mathematical theory and the computational power of computers, have motivated the research towards new optimization-based predictive locomotion planning. Nevertheless, heuristic terms are still exploited in the optimization approaches, *e.g.,* in the definition of the references for the cost function.

2.1.2 Optimization-Based Planning Approaches

Optimization-based techniques aim at finding the variables of interest as solution of an Optimal Control Problem (OCP)

$$\begin{aligned} \min_{\mathbf{x}(t),\mathbf{u}(t)} \quad & \int_{t_0}^{t_f} l\left(\mathbf{x}(t),\mathbf{u}(t)\right) + l_{\mathrm{f}}\left(\mathbf{x}(t_{\mathrm{f}})\right) \\ \text{s.t.} \quad & \text{Initial Condition}(\mathbf{x}(t_0)) \\ & \text{Dynamics Model}(\mathbf{x}(t),\mathbf{u}(t)) \\ & \text{Equality/Inequality Constraints}(\mathbf{x}(t),\mathbf{u}(t)) \end{aligned} \tag{2.2}$$

By defining the cost function $l\left(\mathbf{x}(t),\mathbf{u}(t)\right)$ and the constraints one can represent all range of movements and tasks. Methods based on the partial differential equation Hamilton-Jacobi-Bellman (HJB) equation have been developed to solve the continuous time problem of Eq. (2.2), but they are not suited for problems with many Degrees of Freedom (DoFs).

More often, the problem must be rewritten into a Nonlinear Program (NLP) which can be solved numerically, see Section 2.2.2 for details on how to transcribe an OCP and Section 2.2.3 for an overview on solution methods.

$$\min_{\mathbf{x}^{\mathrm{p}},\mathbf{u}^{\mathrm{p}}} \quad \sum_{k=0}^{N-1} \ell\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{a}_k\right) + \ell_{\mathrm{T}}\left(\mathbf{x}_N\right) \tag{2.3a}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_{\mathbf{0}}, \tag{2.3b}$$

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{a}_k\right), \qquad k \in \mathbb{I}_0^{N-1}, \tag{2.3c}$$

$$h\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{a}_k\right) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \tag{2.3d}$$

The variable $\mathbf{x} \in \mathbb{R}^{n_x}$ indicates the *states* of the system, $\mathbf{u} \in \mathbb{R}^{n_u}$ the *control input*, and $\mathbf{a} \in \mathbb{R}^{n_a}$ the *parameters*. The symbol $N \in \mathbb{R}$ indicates the number of nodes in which the horizon is discretized. Equation (2.3a) is composed of a running term $\ell\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{a}_k\right)$, a terminal cost $\ell_{\mathrm{T}}\left(\mathbf{x}_N\right)$ and it is used to represent the high-level locomotion task, *e.g.,* tracking desired velocities, minimizing the accelerations. Equation (2.3b) imposes the initial condition $\hat{\mathbf{x}}_{\mathbf{0}}$ for the state; the latter is generally measured from the robot or estimated. Equation (2.3d) is the so-called *path constraints*, *i.e.,* the constraints which are imposed at every node of the horizon N. An example of path constraint is imposing that the GRFs are inside the friction cone in order to prevent the foot from slipping. *Feasibility problems* are particular OCPs in which a cost function is not included, *i.e.,* is zero by construction. They neglect the additional features encoded in the cost functions and simply compute the values that satisfy all the given hard constraints.

The choice of the dynamics model of Eq. (2.3a) is crucial to obtain high-performance planners. The following section presents an overview of the most common models for optimization-based planning.

Dynamic models

Dynamic models are the equations used to describe how state changes over time. They are based on the laws of physics and on some modeling assumptions/intuitions; in the locomotion field, their goal is to describe analytically the coupled interactions between the robot and the environment, which generate the motion. Different models have been proposed to find the best trade-off between accuracy and complexity. Indeed, the more complex the chosen model system, the higher the effort (*i.e.,* computational time, hardware power) required to handle it. On the other hand, a detailed dynamics model can take into account all the characteristics of the robot platform, *e.g.,* joint limits, coupling terms in the inertia matrices, etc.

Being inspired by the motion of insects, [Full and Koditschek, 1999] proposed the idea that the movement can be split and described using a model with the least number of variables that shows a target behaviour. Similar models are called *templates* and can be grouped to define an *anchor* model. The impact of the work done by Full and Koditschek is evident in the model presented by [Kajita et al., 2001]. Indeed, under the assumption that there are no vertical and angular dynamics and infinite friction, the motion of a robot supported by a leg can be templated with a Linear Inverted Pendulum (LIP). The model connects the ZMP with the CoM position as follows:

$$\mathbf{p}_{\mathrm{c,(x,y)}} = \ddot{\mathbf{p}}_{\mathrm{c,(x,y)}} \frac{\mathbf{p}_{\mathrm{c,z}}}{g} + \mathbf{z} \tag{2.4}$$

where:

- $\mathbf{p}_{\mathrm{c,(x,y)}} \in \mathbb{R}^2$ is the CoM position in the plane perpendicular to the gravity vector
- $\ddot{\mathbf{p}}_{\mathrm{c,(x,y)}} \in \mathbb{R}^2$ is the CoM acceleration in the plane perpendicular to the gravity vector
- $\mathbf{p}_{\mathrm{c,z}} \in \mathbb{R}$ is the CoM position in a plane parallel to the gravity, *i.e.,* the robot height
- $g \in \mathbb{R} = 9.81\mathrm{m/s}^2$
- $\mathbf{z} \in \mathbb{R}^2$ is the location of the ZMP

In this case, the ZMP coincides with the contact point between the pendulum and the ground; if instead the contact surface is modeled with a foot, the ZMP is free to be any point inside that area (*cart-table* model [Kajita et al., 2003]). The Z component of the GRF balances gravity force relative to the mass m of the pendulum, while the horizontal forces $\mathbf{f}_{\mathrm{x,y}}$ is modeled through a repulsive spring with stiffness equal to $m\omega^2$, with $\omega = \sqrt{g/p_{\mathrm{c,z}}}$ the natural frequency of the linear pendulum.

$$\mathbf{f}_{\mathrm{(x,y)}} = m\omega^2(\mathbf{p}_{\mathrm{c,(x,y)}} - \mathbf{z}) \tag{2.5}$$

Over the years, the LIP has been used as a base for deploying new models. The Spring Loaded Inverted Pendulum (SLIP) [Schwind, 1998; Poulakakis and Grizzle, 2009] considers

the pendulum as a massless axially sprung leg attached to an extended body of mass m and moment of inertia I at a hip joint. Furthermore, the displacement between the CoM and the hip joint is considered. Thus the system is expressed as a function of the pitch angle and the CoM position. In addition, the conversion to polar coordinates allows one to compute the compressed spring length. The relevant contribution of this model is that it can describe a complete gait cycle [Holmes et al., 2006]; during the stance phase, the spring is compressed, and the body moves forward, *i.e.,* the angle formed by the line connecting foothold-CoM and the gravity increases. When the spring unloads at the nominal length, the force drops to zero, the leg loses contact with the ground, and the flight phase begins. When it is in the air, the body follows a ballistic trajectory due to the gravity, and finally, a new stance occurs when the uncompressed leg touches the ground.

The seminal work of [Orin et al., 2013] introduced the simplified model called *Centroidal Dynamics*. The robot is seen as a unique single rigid body whose force due to gravity is applied in its CoM. Orin *et al.* introduced the Centroidal Angular Momentum Matrix $\mathbf{A} \in \mathbb{R}^{3\times(6+n_j)}$, with n_j number of joints of the robot to compute the centroidal angular momentum $\mathbf{k}$ as a function of the robot posture (*i.e.,* CoM position $\mathbf{p}_\text{c}$, orientation $\boldsymbol{\Phi}$, and joint position $\mathbf{q}_\text{j}$), and the posture velocity (*i.e.,* linear velocity $\mathbf{v}_\text{c}$, angular rate $\boldsymbol{\omega}$, and joint velocity $\dot{\mathbf{q}}_\text{j}$).

$$\mathbf{k} = \mathbf{A}(\mathbf{p}_\text{c}, \boldsymbol{\Phi}, \mathbf{q}_\text{j}) \begin{bmatrix} \mathbf{v}_\text{c} \\ \boldsymbol{\omega} \\ \dot{\mathbf{q}}_\text{j} \end{bmatrix} \tag{2.6}$$

In other words, Eq. (2.6) maps the momentum of each link into the resultant one applied to the CoM. Even though position and orientation of the robot depend on the motion of each robot's link, we can represent the six DoFs that describe the position and orientation of the robot using the well-known linear and angular momentum equilibrium at its CoM.

$$\begin{aligned} m\ddot{\mathbf{p}}_\text{c} &= m\mathbf{g} + \sum_{i\in\mathbb{C}} \mathbf{f}_i \\ \dot{\mathbf{k}} &= \sum_{i\in\mathbb{C}} \mathbf{p}_{\text{fc},i} \times \mathbf{f}_i + \boldsymbol{\tau}_i \end{aligned} \tag{2.7}$$

where:

- $\mathbb{C}$ is the set of the indices of the feet in contact with the ground
- $\mathbf{f}_\text{i} \in \mathbb{R}^3$ is the GRF of the leg i
- $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector
- $\mathbf{p}_{\text{fc,i}} \in \mathbb{R}^3$ is the vector connecting the foot i and the CoM
- $\boldsymbol{\tau}_\text{i} \in \mathbb{R}^3$ is the contact torque at the foot i

A widely used model obtained from the Centroidal Dynamics is the Single Rigid Body Dynamics (SRBD). In this case, the centroidal inertia matrix $\mathbf{I} \in \mathbb{R}^3$ is assumed to be independent of the joint positions and generally equal to its value in a nominal configuration. The consequence of this assumption is that the model neglects the swing dynamics, *e.g.,* the re-orientation of the robot in the air during a jump or the motion of the arms for a humanoid robot; in essence, the inertia $\mathbf{I}$ is considered independent of the joint positions. This is a fair assumption either for quasi-static motions or for robots where the weight and the inertia of the trunk are higher than the ones of the limbs, as it generally happens for the quadrupeds; this limitation can be overcome by adding an external wrench, as explained in [Villarreal et al., 2020].

It is important to recall that the already mentioned models can be considered as *reduced* since their aim is to describe only the dominant dynamics. The opposite approach, instead, is to consider a legged robot as a floating-base kinematic tree of rigid bodies [Featherstone, 2007]. Differently from the Centroidal Dynamics model introduced before, in this case, equations can be derived that describe the dynamics of both CoM and joint quantities. In particular, we consider the robot as a fixed-base robot with n_j DoFs (one per each joint) and six additional virtual DoFs corresponding to CoM position and orientation with respect to the inertial (world) frame.
For the sake of clarity, I recall that I use the variable $\mathbf{q}_\mathrm{b} \in \mathbb{R}^3 \times SO(3)$ to describe the six DoFs related to the base (the *unactuated* part)[2] and the variable $\mathbf{q}_\mathrm{j} \in \mathbb{R}^{n_j}$ for the DoFs related to the joints (the *unactuated* part). The symbol $\mathbf{q}$ is stack of $\mathbf{q}_\mathrm{b}$ and $\mathbf{q}_\mathrm{j}$ and represents a point in the robot's configuration manifold. Similarly, $\dot{\mathbf{q}} \in \mathbb{R}^{6+n_\mathrm{j}}$ is the generalized joint space velocity vector, $\dot{\mathbf{q}} = [\dot{\mathbf{q}}_\mathrm{b}^\top, \dot{\mathbf{q}}_\mathrm{j}^\top]^\top$. With this notation the Eq. (2.6) is simply $\mathbf{k} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}}$.

The *Whole-Body Model* (also known as *Full Dynamics Model*) is obtained computing the Lagrangian L of the system [Siciliano, 2007].

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}) \tag{2.8}$$

where $T(\mathbf{q}, \dot{\mathbf{q}})$ is the kinetic energy and $U(\mathbf{q})$ is the potential energy.
The total kinetic energy is the sum of the kinetic energy of each link and corresponds to [Mansard, 2015]:

$$T = \sum_{\mathrm{i}=1}^{n_\mathrm{j}} T_\mathrm{i} = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M} \dot{\mathbf{q}} \tag{2.9}$$

with $\mathbf{M} \in \mathbb{R}^{(6+n_\mathrm{j})\times(6+n_\mathrm{j})}$ named Joint-space Inertia Matrix.
In the same way, potential energy U is the sum of the potential energy of the several links:

$$U = \sum_{\mathrm{i}=1}^{n_\mathrm{j}} U_\mathrm{i} = \sum_{\mathrm{i}=1}^{n_\mathrm{j}} m_\mathrm{i} g h_{\mathrm{q,i}} \tag{2.10}$$

[2]The dimension of the vector $\mathbf{q}_\mathrm{b}$ depends on the chosen parametrization of the orientation, *e.g.,* Euler Angles (6 variables) or quaternions (7 variables)

where m_i is the mass of the link and $h_{\mathrm{q,i}}$ the projection of the link height with respect to the gravity vector.
Solving the Lagrange Equation and considering the contact forces $\mathbf{f} \in \mathbb{R}^m$ we obtain the dynamics of the Whole-Body model (I refer the readers to [Mansard, 2015] for all the details):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{S}\boldsymbol{\tau} + \mathbf{J}_\mathrm{c}(\mathbf{q})^\top \mathbf{f} \tag{2.11}$$

The matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{6+n_\mathrm{j}}$ is the Coriolis matrix and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{6+n_\mathrm{j}}$ is the vector of gravity torques. They can be grouped in a unique vector $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$. The Jacobian matrix $\mathbf{J}_\mathrm{c} \in \mathbb{R}^{m\times(6+n_\mathrm{j})}$ maps the GRFs into torques applied at the feet into the generalized coordinates, while $\mathbf{S} \in \mathbb{R}^{(6+n_\mathrm{j})\times n_\mathrm{j}} = [\mathbf{0}_{6\times n_\mathrm{j}}, \mathbf{1}_{n_\mathrm{j}\times n_\mathrm{j}}]^\top$ is a selection matrix to model the under-actuation. Highlighting the under-actuated and the actuated (joint) dynamics Eq. (2.11) can be rewritten as:

$$\begin{bmatrix} \mathbf{M}_\mathrm{b}(\mathbf{q}) & \mathbf{M}_\mathrm{bj}(\mathbf{q}) \\ \mathbf{M}_\mathrm{jb}(\mathbf{q}) & \mathbf{M}_\mathrm{j}(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_\mathrm{b} \\ \ddot{\mathbf{q}}_\mathrm{j} \end{bmatrix} + \begin{bmatrix} \mathbf{h}_\mathrm{b}(\mathbf{q}, \dot{\mathbf{q}}) \\ \mathbf{h}_\mathrm{j}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_6 \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_\mathrm{cb}(\mathbf{q})^\top \\ \mathbf{J}_\mathrm{cj}(\mathbf{q})^\top \end{bmatrix} \mathbf{f} \tag{2.12}$$

Analyzing Eq. (2.12), it is evident that the floating base quantities (first row) are coupled with the part related to the kinematic tree (joints). For example, the joint accelerations $\ddot{\mathbf{q}}_\mathrm{j}$ generate a torque acting on the trunk of the robot. If this contribution is neglected, as it happens for the Centroidal Dynamics and the SRBD model, the system is affected by external disturbances. On the other hand, the nonlinearity of the terms involved in Eq. (2.12) and the higher number of DoFs ($6 + n_\mathrm{j}$) increase the computational effort required to use those equations. In addition, the dimension of the model, *i.e.,* the number of constraints, depends on the number of contact points (and so the elements m in the vector $\mathbf{f}$). The same model can be used to represent all kinds of robots, *i.e.,* monopeds, quadrupeds, hexapods, etc., but in all the cases, the value m changes during the motion.

Table 2.1 summarizes and compares the main dynamics models most frequently used in optimization-based planning.

Table 2.1: Types of dynamics models mostly used in optimization-based planning

Model	States	Non-flat ter.	Ang. Dynamics	Friction constr.	Torque constr.
LIP (walking)	2	×	×	×	×
LIP + Flywheel	3	×	✓	×	×
SLIP (Running)	3	✓	×	×	×
SRBD	6	✓	✓	✓	×
Centroidal Dyn.	6	✓	✓	✓	×
Whole-Body	$6+n_\mathrm{j}$	✓	✓	✓	✓

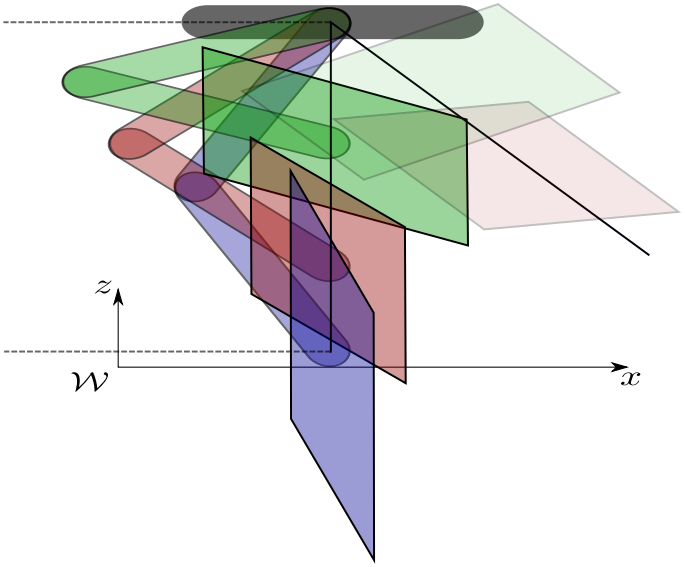

Figure 2.2: Representation of a planar leg with 2 DoFs in three different configurations. For each configuration the corresponding force polytope is shown (image taken from [Bratta et al., 2020])

State-of-the-art approaches

Several different OCPs and locomotion frameworks have been proposed. For example, [Winkler et al., 2018] introduced TOWR, a feasibility problem that computes optimal values of CoM quantities, orientation, GRFs, and foot position. Due to the large number of variables involved, the SRBD is used as a dynamics model. The path constraints impose that the feet are inside the workspace of the robot, that the GRFs are inside the friction cone, and the coherence between feet position and the terrain. Simulations with different legged robots and hardware experiments successfully validated the approach, but they also showed that a long computational time is required to obtain the solution.

TOWR must be considered as the starting point of this dissertation since, in my first work [Bratta et al., 2020], I modified its original formulation with two additional constraints. The first one finds an approximate relationship between contact forces and joint-torque limits, exploiting the theory of the polytopes [Chiacchio et al., 1997], see Figure 2.2; the second constraint, instead, overcomes the assumption of point feet done by the SRBD model by introducing a simplified kinematic model of the leg. Another extension of TOWR [Melon et al., 2020] added a learning-based scheme to initialize the nonlinear problem.

Alternatively, Sequential Linear Quadratic (SLQ) [Sideris and Bobrow, 2005] algorithm demonstrated their efficiency in Trajectory Optimization (TO) frameworks [Farshidian et al., 2017b]. Dealing with nonlinear dynamics, SLQ consists in rolling out the dynamics and then performing a linearization around the obtained trajectory. Approximating the cost with a quadratic function, the result is a Linear-Quadratic Optimal Control problem, which can be

solved in closed form with Riccati-like equations. In [Neunert et al., 2016], the authors were able to plan dynamic motions through contact changes on a whole-body 3D model without the need of pre-defining contact switches and contact points. They validated this approach in different cases, *e.g.,* trot, gallop, squat jumps.

Finally, TO benefits from the improvements in the control theory since they give the researchers new instruments to solve more complex problems. An atypical example is represented by the Differential Dynamic Programming (DDP) [Mayne, 1966]. Even though it was introduced back in the '60s, DDP is gaining popularity nowadays. It is based on Bellman's principle [Bellman, 1954][3] and recursively solves the problem Eq. (2.3) and computes the control sequence executing a forward pass and a backward pass. Given a nominal control sequence, the forward pass integrates the dynamics to obtain a nominal trajectory. The backward pass is then executed to generate a policy that updates the control sequence; In particular, the control inputs are expressed as locally-linear feedback policy of the state. The limitation of DDP is that the original formulation cannot handle other constraints than the dynamics. Examples of DDP-based approach are the hybrid system for bounding quadrupeds presented in [Li and Wensing, 2020], the hierarchical optimization of several tasks in the cost function [Geisert et al., 2017], and the integration of box inequality constraints on the controls [Tassa et al., 2014].

The common element among these TO algorithms is that they *couple* gait sequence, CoM trajectories, and GRFs, optimizing them at the same time. The advantage is that those methods guarantee both kinematic and dynamic feasibility. On the other hand, they suffer from the same limitations: large combinatorial space and high computational cost. The alternative is represented by the *decoupled approaches* [Escande et al., 2008; Bretl, 2006; Shkolnik et al., 2011; Dai et al., 2014]; they split the optimization into two sub-problems, one to choose the footholds (*contact planners*) and the following one for the centroidal quantities.

[Tonneau et al., 2018] presented an algorithm which computes a CoM path inside the contact reachable space (an approximation of the space where the static equilibrium feasibility is satisfied). A second problem extends the trajectory into a discrete sequence of contact configurations. The drawback of decoupled approaches is that they cannot guarantee that the selected footholds yield to a dynamically feasible trajectory for the robot's base. To handle this limitation, the Continuous Convex Resolution of Centroidal Dynamic Trajectories (C-CROC) [Fernbach et al., 2020] evaluates the footholds according to the transition feasibility criterion. In other words, considering two states separated by at most one contact creation and one contact break, C-CROC tests if there exists a dynamically and kinematically valid motion that connects the two states.

Finally, another example of decoupled approach is [Budhiraja et al., 2018]: a Centroidal

[3] *"An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions".*

Dynamics-based optimization is used to compute gait sequence and CoM trajectory, while the DDP algorithm finds the Whole-Body trajectories to follow the centroidal pattern.

Decoupled approaches and the research on fast optimization algorithms paved the way for online replanning, and MPC approaches.

2.1.3 Model Predictive Control

MPC is a control methodology applied to a wide range of different applications [Ruchika, 2013]. It was initially developed to control the transients of dynamic chemical systems with hundreds of inputs and outputs, subject to constraints [Qin and Badgwell, 2003] and is widely used in robot locomotion since it guarantees feasibility and allows the computation of updated trajectories while the robot is moving. To the best of my knowledge, the first paper to present the application of an MPC to a humanoid robot is [Wieber, 2006]. The author demonstrates that his approach is able to compensate for strong perturbations and improves the already mentioned ZMP Preview Control. Several researchers have extended their TO algorithm to perform online replanning, *e.g.,* [Li et al., 2022] as development of [Li and Wensing, 2020].

The MPC is based on the *receding horizon principle*: the problem of Eq. (2.3) is solved, and N control signals are computed, but only the first one is applied. After that, the initial conditions and the information coming from the environment are updated, and a new trajectory is computed. In this way, the MPC can react to changes in the environment and external disturbances and compensate for model inaccuracies.

The chosen model and constraints are strictly related to the replanning frequency: the faster the control is updated, the simpler the OCP of Eq. (2.3) can be. For example, in [Di Carlo et al., 2018], the model neglects the effect of nutation and procession, *i.e.,* the term $\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$, in the Centroidal Dynamics model of Eq. (2.7). In addition, only the yaw angle is considered for computing the orientation of the robot with respect to the inertial frame. Roll and pitch measurements are neglected since they are generally small values for flat terrain locomotion. Considering a short prediction horizon of 0.5 s, the problem is solved in 1 ms, and a frequency of 20-30 Hz is achieved. Thanks to the high frequency, despite using a simplified model, the approach is capable of obtaining robust locomotion at various speeds with the Cheetah 3 robot [Bledt et al., 2018].

In another work [Arena et al., 2021], a linear MPC is integrated with a Central Pattern Generator to control the steering of a quadruped robot. The authors presented an interesting analysis between the MPC framework and a PID approach; they showed that there are equivalent in some conditions, but in other cases, the PID loses for some time window the reference tracking signal.

Another example of simplified model-based MPC is [Villarreal et al., 2020], where SRBD is

used to compute dynamic trajectories for a simulation of HyQReal [Semini et al., 2019] to resist a wide range of disturbances and act preemptively to obstacles based on visual information.

Finally, the SRBD model is also used in [Ding et al., 2019]. The authors presented a formulation which linearizes rotation matrices, avoiding parameterizations like Euler angles and quaternions; in this way it prevents the issues of singularity and unwinding the phenomenon. Transcribing the problem into a Quadratic Program (QP), a replanning frequency of 160 Hz and trot and bound gaits were executed on a 5.5 kg electrical quadruped robot. Moreover, this work shows a back-flip maneuver in simulation. To deal with the same drawback, *i.e.,* singularity in the Euler angles, [Hong et al., 2020] performed a reparameterization of the robot orientation (on SO(3)) to orientation error in the tangent space of that manifold in a constrained NMPC;

A ZMP-based MPC, which decides the footstep placement of a human robot [Herdt et al., 2010] automatically. The importance of a similar approach is that footholds play a key role in the agility of the robot since their coherence with the CoM can improve the stability of the robot.

The so called *kinodynamics* models have been proposed to consider leg kinematics. For example, [Cebe et al., 2021] includes feet velocity in the controls, and [Grandia et al., 2019a] which considers the joints velocity as an input. In addition, the latter takes into account the limitations of the actuation bandwidth through frequency-dependent cost functions. The limitation of [Cebe et al., 2021], instead, is that a new trajectory is computed only at every step.

As already said for the TO algorithms, the computational efficiency of the DDP has been exploited by the MPCs. In [Koenemann et al., 2015], the authors described the first application of real-time whole-body MPC on a full humanoid robot, HRP-2. The DDP is implemented in MuJoCo[Todorov et al., 2012], and it is able to compute a 500ms trajectory in 50ms on a standard desktop computer. Another example of DDP-based method is [Grandia et al., 2019b] in which feedback MPC achieves stable locomotion under a low update rate (15 Hz). Indeed, a feedback policy obtained by the DDP compensates for the lack of fast update of the trajectory. In addition, this approach does not require a motion controller anymore.

The pioneering work of [Koenemann et al., 2015] showed that it is possible to use also the full model into MPC approaches. [Neunert et al., 2018] presented a whole-body MPC which also includes explicit contact dynamics to obtain trot and jump on both HyQ and ANYmal on flat terrain. Therefore, contact locations, sequences and timings are not prespecified but optimized by the solver. Exploiting Auto-Differentiation [Giftthaler et al., 2017] and code generation [Frigerio et al., 2012] the algorithm is executed at a replanning frequency of 190 Hz; to the best of the authors' knowledge and mine, the achieved frequency outperformed the state of the art of that time by at least one order of magnitude. On the other hand, the experimental results they presented are of very basic motions that do not really exploit the capabilities of the approach.

Most recent works

[Mastalli et al., 2022] proposed a hybrid predictive controller which combines feedback policies and tactile information. A feasibility-driven technique is used to reduce the computational cost. Such full-dynamics MPC can handle nonholonomy in the kinetic momenta, torque limits and friction-cone constraints.

A Whole-Body NMPC which integrates motion planner and perceptive information has been recently presented in [Grandia et al., 2022]. The terrain is encoded through a series of geometric primitives, *i.e.,* convex regions, and optimal footholds are chosen within those regions. Previous works of the same authors are the MPC with Control Lyapunov Functions [Minniti et al., 2022] and the *Terrain-Aware Motion Optimization for Legged Systems* (TAMOLS) [Jenelten et al., 2022].

Meduri *et al.* [Meduri et al., 2022] presented a biconvex dynamics solver and a kinematic formulation for a 20 Hz NMPC. BiConMP computes trajectories for both a quadruped (Solo12, in experiments) and a humanoid robot (Talos, in simulation) robot. In one of the hardware experiments, Solo12 gives a high-five to a person in front of it by first raising both its front legs at a height above its base and then reaching one of its leg out forward.

A bridge between TO and MPC is presented in [Bjelonic et al., 2022]: offline optimizations create motion libraries which are used as cost function of the MPC. The result is a complex, long-time horizon plan with reactive, short-time horizon solutions.

Finally, very recently, a complete review on optimization-based control for legged locomotion has been uploaded to arXiv [Wensing et al., 2022].

2.1.4 Contact Planner

A particular class of legged locomotion approaches is represented by the *Contact planners*. This section first presents an overview of the algorithms, which do not assume a predefined leg sequence but automatically discover it and choose the foothold configurations. In the second part, I mention some works that deal with foothold optimization, given which leg must swing.

Graph-based approaches demonstrated to be suitable for contact planning since they can cope with problems where the number of decision variables is unknown [Griffin et al., 2019]. However, these techniques present some limitations. For example, [Chestnutt et al., 2007] focused on A* search which adapts the actions to the local terrain during the navigation. Indeed, since the possible discrete actions are fixed throughout the planning process, the planner cannot find a solution if the required action is not present in the action sampling. With this method, the quadruped robot LittleDog was able to crawl over rough terrain. Even though the adaptive A* algorithm reduced the computation time with respect to standard

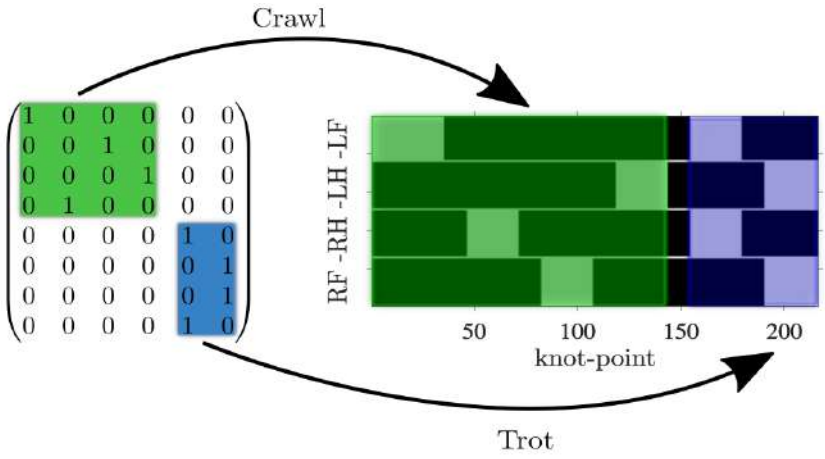

Figure 2.3: Example of transition matrix and gait phase. R, L, H and F represent right, left, hind and front legs (image taken from [Aceituno-Cabezas et al., 2018]).

A*, it is far from being executed in an MPC fashion since 92 s are required, on average, to compute a solution for rough environments.

Already introduced in Section 2.1.2, the TO presented in [Winkler et al., 2018] can be considered as a contact planner since the phase duration of each leg is an optimization variable which the solver can change. An additional constraint imposes that the sum of the phases coincides with the total duration of the motion. In [Posa et al., 2014], contact variables are avoided and substituted by the Linear Complementarity Constraint. It guarantees that either a force is applied or the end effector moves. In both cases, all variables are continuous.

A completely different approach is the one based on Mixed-Integer Programming (MIP). The MIP is an OCP in which one (or more than one) optimization variable is constrained to be an integer value.
The seminal work of Deits and Tedrake [Deits and Tedrake, 2014b] devised a MIP which handles obstacle avoidance, kinematic reachability, and rotation of footstep placements. The environment is decomposed into a set of convex regions where the foot can be safely placed using IRIS algorithm [Deits and Tedrake, 2014a]. A linear constraint ensures that if a footstep is chosen in a region, it lies on the fitting plane of the local terrain. A quadratic constraint imposes that each footstep is reachable from the previous step and step rotations are dealt with a piecewise linear approximation of sine and cosine. The proposed Mixed-Integer Quadratically Constrained Quadratic Program prevents from the need of nonconvex constraints, allowing the authors to solve the planning problem for a short horizon in less than 1 second.

[Aceituno-Cabezas et al., 2018] extended this idea to a Mixed-Integer Convex Optimization Problem (MICP) that simultaneously plans contacts and CoM motions. They introduced the binary *transition matrix* $\mathbf{T}$. The value of the element T_{ij} is 1 if the robot moves to i-th contact location at time j otherwise, it is 0, see Figure 2.3. Using this matrix, it is possible

to write constraints on the GRFs, *i.e.,* no force when the foot is in the air, and on the foot motion, *i.e.,* stationary leg when the leg is on the ground. Experiments with HyQ demonstrate that trajectories computed with this TO framework can deal with challenging nonflat terrains without requiring any pre-defined gait sequence.

The limitation of the MIP approaches is the high computation cost, which is mainly due to the branch-and-bound techniques [Clausen, 2003] to handle the combinatorial aspect of the problem, *i.e.,* the issue of selecting contact surfaces, as described in [Tonneau et al., 2020]. [Ponton et al., 2021] solved a MIP to find dynamically consistent contact sequences, combined with a kinodynamic optimization method to generate full-body movements. The authors of [Tonneau et al., 2020], instead, overcome the limitation of the computational cost using a convex relaxation of the problem thanks to an L1-norm minimization formulation. Given a set of surfaces, the MIP needs to find a point that belongs to exactly one surface. The authors showed how to reformulate this problem with a Linear Program (LP), substituting, thus, the branch-and-bound search. Simulations with HRP-2 and Talos demonstrated that this approach is 50 to 100 times faster than a normal MIP.

The Contact-Invariant Optimization [Mordatch et al., 2012], instead, incentivizes interaction with the environment to generate the desired behaviour. In fact, the formulation is endowed with scalar variables that indicate if a potential contact should be active in a given phase; they affect both the cost function and the dynamics (by enabling and disabling contact forces). The drawback of a larger combinatorial space is mitigated by the fact that it becomes better behaved and continuous.

Finally, another sub-class of problems uses the Monte Carlo Tree Search (MCTS). The MCTS decision process is represented by a tree where each node represents a *state*, *i.e.,* one of the possible choices for the contact configuration. A simulation policy is used to assign a reward to a node, while a tree policy decides which is the next node to expand. The goal of the MCTS is to choose the nodes' sequences that maximize the reward. Among the MCTS-based work, I highlight [Labbe et al., 2020] and [Amatucci et al., 2022]. The former demonstrated the efficiency of the MCTS in a pick-and-place task with a robotic arm; the latter applied it to an MPC framework for legged locomotion (in simulation). In particular, [Amatucci et al., 2022] obtained recently a high-frequency MPC which automatically discovers periodic gaits and adapts to external disturbance. A comparison with an implementation of the MCTS presented in [Aceituno-Cabezas et al., 2018] demonstrates, on average, that the MCTS of [Amatucci et al., 2022] is 2.72 times faster.

Foothold Selection

As explained in Section 2.1.2 and 2.1.3, decoupled approaches find footholds and then CoM quantities separately. In this section, I explain some foothold selection techniques.

The seminal work of Kolter *et al.* [Kolter et al., 2008] is one of the first applications of terrain awareness foothold selection. The authors rely on a trained Hierarchical Apprenticeship Learning algorithm. In order to select the best footholds in terms of robustness to deviations and slippage, the network is trained with collision probability maps and heightmaps collected a priori.

Discretizing a local heightmap into *templates* [Kalakrishnan et al., 2009], the work introduced in [Kalakrishnan et al., 2011] uses a learning regression method to rank the footholdsfootholds, allowing the LittleDog robot to traverse rough terrains. A similar work has been proposed by [Barasuol et al., 2015].

These works are based on expert user demonstration. In [Villarreal et al., 2019], the authors improved their previous work [Barasuol et al., 2013] using a heuristic algorithm that generates the ground truth of a self-supervised algorithm. In this way, a Convolutional Neural Network (CNN), named Visual Foothold Adaptation (VFA), is trained, and optimal corrections to nominal footholds are provided considering kinematics limits, terrain roughness, and collisions (both frontal and leg collision). Using CNNs demonstrated to be effective also for compliant feet that can mechanically adapt to the terrain profile [Bednarek et al., 2020].

As a general comment, machine learning approaches are more popular since they are more suitable for online evaluations. An example of pure optimization-based foothold selection algorithm is [Fankhauser et al., 2018]. In this work, a sequence of nominal footholds is computed heuristically based on the default step length and the initial condition of the robot. Considering a binary validity score taken from the evaluation of a 2.5D elevation map,[4] each foothold is discarded if it corresponds to an unfeasible foothold; the closest feasible foothold is selected as a candidate. In addition, it is checked for its reachability by running the pose optimizer with the current and the candidate foothold as input. A drawback is that a new foothold is computed only at every step, while algorithms as [Villarreal et al., 2019] are able to change the touchdown point also at any time during the swing phase.

2.2 Numerical Optimization Techniques

In the previous section, I presented an overview of optimization-based frameworks, *i.e.,* approaches in which the required variables (CoM trajectories, GRFs, footholds, etc...) are the solution of an OCP. In this section, I focus on how the OCP can be classified and which are the main techniques to write and solve them. Additional details can be found in [Diehl and Gros, 2017; Zanon, 2021].

[4] the readers are encouraged to refer to the previous work of the same authors [Fankhauser et al., 2014] for more details on the elevation map.

2.2.1 Classification

The goal of an OCP is to find the value of some quantities, named *optimization variables* which minimize a scalar *cost* function which depends on the optimization variables and references values, see Eq. (2.3). The values of the optimization variables should satisfy the relationship represented by the dynamic model and the *path constraints*, both equality and inequality.

Depending on the properties of the cost and the path constraints, the OCP can be divided into sub-families. The main classification is between *convex* and *nonconvex* OCP. In convex OCPs, both the cost and the constraints are convex.[5]
Recall: a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex in its domain $\mathbb{D}$ if:

$$f(\lambda x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall x, y \in \mathbb{D}, \quad \alpha \in [0;1]$$

Examples of convex functions are the affine transformation and the exponential, thus convex OCPs are the LPs, the QPs; depending on the constraints sometimes also NLPs are convex problems.

The main advantage of convex OCPs is that, if it exists, the optimal solution is unique [Boyd and Vandeberghe, 2004]. The most straightforward problem to be solved is the QP with linear constraints. A quadratic cost function $L(\mathbf{w})$ with $\mathbf{w}$ optimization variables corresponds to:

$$\frac{1}{2}\mathbf{w}^\top \mathbf{Q}\mathbf{w} + \mathbf{c}^\top \mathbf{w} + d$$

with the necessary condition that the matrix $\mathbf{Q}$ is positive definite, *i.e.,* symmetric matrix with all eigenvalues greater than zero. The solution of a QP with linear equality constraints can be found analytically. In the other cases, *i.e.,* QPs with inequality constraints or LP, the solution is not found in closed form, but numerically and it is still guaranteed to be a global minimum. As discussed in Section 2.1, convex approaches demonstrated good performance for locomotion applications and, in particular, low computation cost in MPC approaches.

A particular family of OCP is represented by the MIPs: some of the optimization variables are restricted to be integer quantities. In particular, in the Mixed-Binary-Integer Programming (MBIP), the value must be either 0 or 1. For example, in legged locomotion, a MBIP is obtained when a variable is used to encode whether a leg is in contact with the ground or not.

2.2.2 Transcription Methods

Equation (2.3) represents a finite-size problem which can be solved with numerical solvers. In reality, it is the result of a conversion (named *transcription*) of a continuous time OCP, Eq. (2.2), into a discrete one. The choice of the transcription method affects both accuracy

[5] in case of a maximization problem, the cost function must be concave.

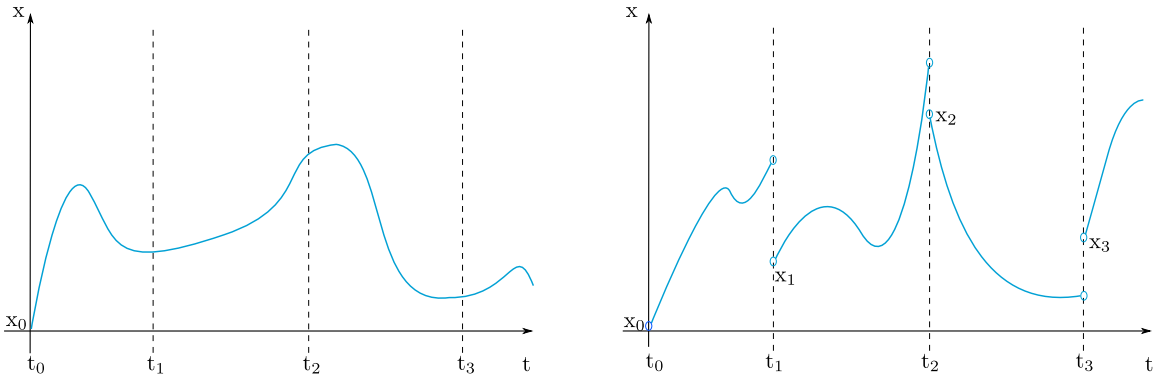

Figure 2.4: Direct (left) and multiple (right) shooting methods. The former simulates the state trajectory for the entire horizon. The latter divides the horizon into segments and integrates the states independently. Continuity constraints prevent from discontinuities at the beginning of each segment.

and numerical stability of the result. In this section, I introduce the two main transcription methods used in robotics [Nocedal and Wright, 2006]: *direct shooting* and *collocation*. Readers are encouraged to refer to [Betts, 2010] for further details on this topic.

In direct *single* shooting approaches, only the control input $\mathbf{u}$ are decision variables. Given the initial state condition, the values of the states are obtained by simulating the model along the entire horizon. This approach is suitable either for short horizons or simple dynamics. Indeed, the integrator functions used to simulate the dynamics tend to become highly nonlinear for long horizons. In particular, for unstable systems the states tend to diverge and the problem becomes badly scaled. A more accurate transcription method is the direct *multiple* shooting [Bock and Plitt, 1984], where the horizon is divided into smaller intervals, and the integration is performed independently for each segment; in other words, the transcription problem becomes a series of Initial Value Problem [Diehl et al., 2005]. A continuity constraint is imposed to nullify the *defect* (or *gaps*), *i.e.,* the difference between the last point of an integrated trajectory and the first value (*shooting node*) of the following interval. Even though the number of decision variables increases (control input and state at the shooting node), multiple shooting is widely used in robotics to prevent state diverging due to the complex robot dynamics. An example of direct and multiple shooting is represented in Figure 2.4.

In the collocation method, instead, both control and states are optimization variables. Similarly to the multiple shooting, the horizon is divided into segments and continuity is guaranteed by the constraint on the defect at the end points. The difference is that the state evolution is approximated by polynomials. The dynamics is imposed enforcing that it matches with the slope of the chosen polynomials in a discrete number of intermediate (*collocation*) points. The number of collocation points is a key design choice, since it affects the accuracy of the method [Hager et al., 2019].

2.2.3 Solution Methods

Legged locomotion benefits from the developments of the solvers needed to compute the optimal solution. Indeed, the choice of the solver is a key element of the design. In this section, I provide an overview of the most common solution methods and commercial solvers used in robotics at the time of writing this dissertation.

Let's consider the OCP for the variable $\mathbf{w}$, with $L(\mathbf{w})$ as cost function, $g(\mathbf{w})$ as equality constraints and $h(\mathbf{w})$ as inequality constraints.

$$\begin{aligned} \min_{\mathbf{w}} \quad & L(\mathbf{w}) \\ \text{s.t.} \quad & g(\mathbf{w}) = 0 \\ & h(\mathbf{w}) \leq 0 \end{aligned} \tag{2.13}$$

A general assumption is that $L(\mathbf{w})$, $g(\mathbf{w})$, and $h(\mathbf{w})$ are continuously differentiable.

The solution of Eq. (2.13) coincides with a systems of equations, named *KKT Condition* [Karush, 1939; Kuhn and Tucker, 1951].[6]

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla L(\mathbf{w}) + \boldsymbol{\lambda} \nabla g(\mathbf{w}) + \boldsymbol{\mu} \nabla h(\mathbf{w}) = 0 \tag{2.14a}$$

$$g(\mathbf{w}) = 0 \quad h(\mathbf{w}) \leq 0 \tag{2.14b}$$

$$\boldsymbol{\mu} \geq 0 \tag{2.14c}$$

$$\mu_{\mathrm{i}} h_{\mathrm{i}}(\mathbf{w}) = 0 \quad \forall \mathrm{i} \in \dim(h(\mathbf{w})) \tag{2.14d}$$

The function $\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is called Lagrangian, with $\boldsymbol{\lambda}, \boldsymbol{\mu}$ lagrangian multipliers; thus, Eq. (2.14a) corresponds to find its critical point, *i.e.,* the point where its gradient vanishes. Equation (2.14b) is referred to as *primal* feasibility, while Eq. (2.14c) is the *dual* feasibility. Finally, Eq. (2.14d) is the *complementarity slackness*. It impose that either the inequality or the multiplier $\boldsymbol{\mu}$ must be 0.

Newton's Algorithm

For OCP without inequality constraints, the simplest solution method is Newton's algorithm. It is an iterative approach based on the following equation

$$\begin{bmatrix} \nabla^2_{\mathrm{w,w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) & \nabla g(\mathbf{w}) \\ \nabla g(\mathbf{w})^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{w} \\ \boldsymbol{\lambda}_+ \end{bmatrix} = - \begin{bmatrix} \nabla L(\mathbf{w}) \\ g(\mathbf{w}) \end{bmatrix} \tag{2.15}$$

The optimization variables and the lagrangian multiplier can be updated by $\boldsymbol{\lambda}^+ = \boldsymbol{\lambda}_+ = \Delta\boldsymbol{\lambda} + \boldsymbol{\lambda}$, and $\mathbf{w}^+ = \mathbf{w} + \Delta\mathbf{w}$.

[6] KKT stands for the name of the researchers who developed them: Karush, Kuhn, and Tucker

In other words, Eq. (2.15) computes the directions along which the cost function reduces. The *linesearch* strategy computes the step size for $\mathbf{w}^+$ along that direction.

$$\begin{aligned} \mathbf{w}^+ &= \mathbf{w} + \alpha\Delta\mathbf{w} \\ \boldsymbol{\lambda}^+ &= (1-\alpha)\boldsymbol{\lambda} + \alpha\boldsymbol{\lambda}_+ \end{aligned} \tag{2.16}$$

with $\alpha \in [0, 1]$. Two approaches exist to perform a linesearch. The first one is the *exact linesearch* which solves a second problem to find explicitly the optimal value α^+ which minimizes $L(\mathbf{w} + \alpha\Delta\mathbf{w})$.
The alternative is the so called *backtracing* linesearch in which α is set equal to 1 and reduced until the Armijo's condition is satisfied [Armijo, 1966].

$$L(\mathbf{w} + \alpha\Delta\mathbf{w}) < L(\mathbf{w}) + \gamma\alpha\nabla L(\mathbf{w})\Delta\mathbf{w}$$

The variable γ is a design parameter.

Newton's algorithm consist of iteratively solving Eq. (2.15) and evaluating Eq. (2.16) till $\boldsymbol{\lambda}^+$ and $\mathbf{w}^+$ satisfy lagrangian and primal feasibility constraints of Eq. (2.14).

Solving Eq. (2.15) requires the hessian to be invertible. For this reason, a regularization of the matrix $\nabla^2_{\mathrm{w,w}}\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$ must be performed to modify the eigenvalues such that the matrix is positive definite and thus invertible.

In addition, the computation of the hessian (and of its inverse matrix) is time-consuming, so modified Newton's methods are obtaining by approximating that term, *e.g., Steepest Descent* (the hessian is the identity matrix), *Gauss-Newton* ($\nabla^2_{\mathrm{w,w}}\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla_{\mathrm{w}}\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})\nabla_{\mathrm{w}}\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})^\top$), and *BFGS*[7] (iterative process to update at each repetition the approximation).

Even though Newton's methods are the most popular solution techniques, they have two main drawbacks:

- they cannot handle the complementarity slackness, and thus the inequalities
- the convergence depends on the starting point of the iterative process. Indeed, in the case of an initial guess far from the optimal values, the solver could not be able to find a solution or converge to a local sub-optimal one.

Interior Point Methods

Interior Point (IP) methods were initially developed for LP [Karmarkar, 1984], but have been later extended to NLPs and nonconvex optimization [Wright, 2004]. Several different versions exist [Potra and Wright, 2000], I describe the two main ones: Barrier IP and the Primal-Dual IP.

[7] named after Broyden, Fletcher, Goldfarb and Shanno Fletcher [1987]

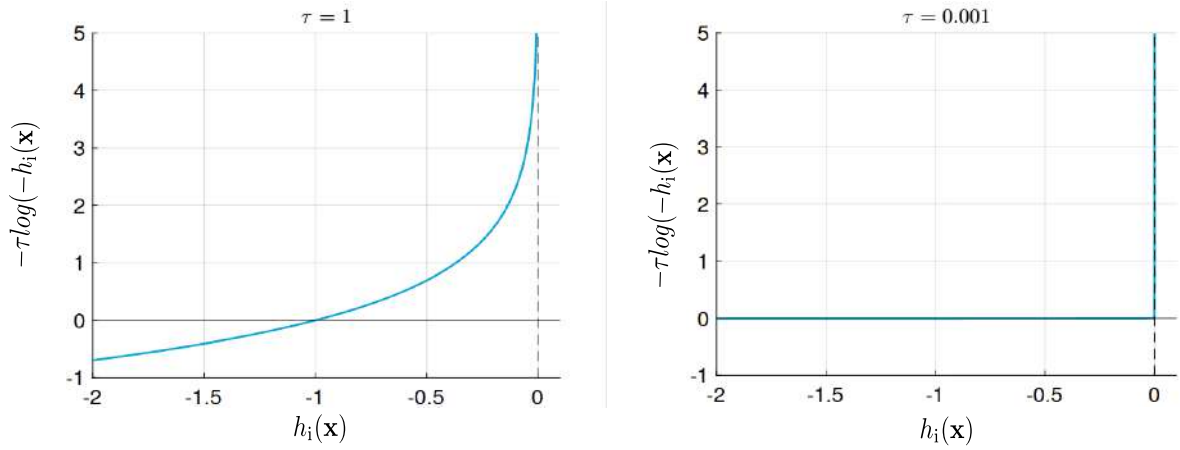

Figure 2.5: Representation of the logarithmic approximation of the inequality constraints for two different values of τ in a Barrier IP, (image taken from [Zanon, 2021].

Barrier IPs get rid of the complementarity slackness, modifying the cost function. The idea is that we add the *characteristics* function, *i.e.,* the function which is equal to 0 if the inequality constraint is satisfied and ∞ if it is violated.

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + c(\mathbf{w})$$
$$\text{with} \quad c(\mathbf{w}) = \begin{cases} 0 & \text{if} \quad h_\text{i}(\mathbf{w}) \leq 0 \\ \infty & \text{if} \quad h_\text{i}(\mathbf{w}) > 0 \end{cases}$$

Common choice for $c(\mathbf{w})$ is the logarithmic function

$$c(\mathbf{w}) = -\tau \sum_{\text{i}=0}^{\dim(h(\mathbf{w}))} log(-h_\text{i}(\mathbf{w}))$$

In other words, we heavily penalize a solution in which the inequality constraint is not satisfied. The accuracy of the approximation of the characteristic function with a logarithmic term increases for $\tau \to 0$, see Figure 2.5. The KKT conditions corresponding to IP Barrier Methods are:

$$\begin{aligned} &\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla L(\mathbf{w}) - \tau \sum_{\text{i}=0}^{\dim(h(\mathbf{w}))} h_\text{i}^{-1}(\mathbf{w})\nabla h_\text{i}(\mathbf{w}) + \lambda \nabla g(\mathbf{w}) = 0 \\ &g(\mathbf{w}) = 0 \\ &h(\mathbf{w}) \leq 0 \end{aligned} \tag{2.17}$$

Eq. (2.17) can be solved with Newton's method with linesearch, reducing iteratively τ towards zero. The drawback of this approach is represented by the term $h_\text{i}^{-1}(\mathbf{w})$ which becomes $h_\text{i}^{-2}(\mathbf{w})$ in the hessian of Eq. (2.15). For $h_\text{i}(\mathbf{w})$ close to zero the value of $h_\text{i}^{-2}(\mathbf{w})$ becomes high creating numerical issues when multiplied by a small value as τ. The *Primal-Dual* IP is obtained

introducing a new variable ν and rewriting Eq. (2.17) in the following way

$$\begin{aligned} &\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\nu},\boldsymbol{\mu}) = \nabla L(\mathbf{w}) + \boldsymbol{\nu}\nabla h(\mathbf{w}) = 0 \\ &g(\mathbf{w}) = 0 \quad h(\mathbf{w}) \leq 0 \\ &\boldsymbol{\nu} \geq 0 \\ &\nu_{\mathrm{i}} h_{\mathrm{i}}(\mathbf{w}) = -\tau \quad \forall \mathrm{i} \in \dim(h(\mathbf{w})) \end{aligned} \tag{2.18}$$

Primal-Dual IPs demonstrate high accuracy and computational speed and are the basis for commonly used solvers in robotics, *e.g.,* IPOPT [Wächter and Biegler, 2006] and HPIPM [Frison and Diehl, 2020].

Active-set methods

Active set methods [Wong, 2011], instead, split the inequality constraints into *active*, *i.e.,* $h_{\mathcal{A}}(\mathbf{w}) = 0$, and *inactive*, $h_{\bar{\mathcal{A}}}(\mathbf{w}) = 0$. The corresponding KKT conditions can be written as:

$$\begin{aligned} &\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda},\boldsymbol{\mu}_{\mathcal{A}}) = \nabla L(\mathbf{w}) + \boldsymbol{\lambda}\nabla g(\mathbf{w}) + \boldsymbol{\mu}_{\mathcal{A}}\nabla h_{\mathcal{A}}(\mathbf{w}) = 0 \\ &g(\mathbf{w}) = 0 \\ &h_{\mathcal{A}}(\mathbf{w}) = 0 \\ &\boldsymbol{\mu}_{\mathcal{A}} > 0 \quad h_{\bar{\mathcal{A}}}(\mathbf{w}) < 0 \end{aligned} \tag{2.19}$$

Active-set methods solvers guess which are the active constraints, solve the first three rows of Eq. (2.19), and then check if the conditions in the last row are satisfied. In particular, if $h_{\bar{\mathcal{A}}}(\mathbf{w}) = 0$ the corresponding index is added to the active set, while an index is removed from $\mathcal{A}$ if the value of its $\boldsymbol{\mu}_{\mathcal{A}}$ is negative. The process is iteratively repeated till the optimal solution is found. The active-set method is the base of the very famous Dantzig's *Simplex Algorithm* [Dantzig, 1963].

Sequential Quadratic Program

Sequential Quadratic Program (SQP) is a bridge between the original OCP of Eq. (2.13) and IP / active-set methods. Indeed, the approach consists in writing a quadratic approximation of the non linear problem whose solution corresponds to the Newton direction of the original non linear problem of Eq. (2.13):

$$\begin{aligned} \min_{\Delta\mathbf{w}} \quad & \frac{1}{2}\Delta\mathbf{w}^{\top}\nabla^2_{\mathrm{w,w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda},\boldsymbol{\mu})\Delta\mathbf{w} + \nabla f(\mathbf{w})^{\top}\Delta\mathbf{w} \\ \text{s.t.} \quad & g(\mathbf{w}) + \nabla g(\mathbf{w})^{\top}\Delta\mathbf{w} = 0 \\ & h(\mathbf{w}) + \nabla h(\mathbf{w})^{\top}\Delta\mathbf{w} \leq 0 \end{aligned} \tag{2.20}$$

Once solved, Eq. (2.20) returns the direction $\Delta\mathbf{w}$ and two new multipliers $\boldsymbol{\lambda}^{\mathrm{QP}}$ and $\boldsymbol{\mu}^{\mathrm{QP}}$.

The SQP algorithms perform iteratively the following operations:

1. write and solve Eq. (2.20)
2. update primal variable $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
3. update dual variable $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{\mathrm{QP}}$, $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}^{\mathrm{QP}}$
4. check if $(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is a solution of Eq. (2.13)

2.3 Summary and Discussion

In this chapter, I have presented the relevant works in the history of legged locomotion, starting from the pioneering heuristic algorithm [Raibert, 1986] till the MPCs developed nowadays. A subsection is dedicated to the contact planners, *i.e.,* approaches that simultaneously compute the gait sequence and the footstep locations, since they have room for exciting developments. Section 2.2 provides additional details on the OCP and on the state-of-the-art tools to solve them. Even though it is not evident at first sight, the transcription method and the characteristic of the solver are crucial design elements of an optimization framework. Indeed, they affect stability and computational cost.

The literature analysis shows that the research on legged locomotion is quickly reaching a mature state. As already mentioned, the challenge is represented by finding the trade-off between complexity and high replanning frequency. For example, [Winkler et al., 2018; Tonneau et al., 2020] need to be performed offline, and the MPCs such as the one in [Cebe et al., 2021] updates the trajectory only at every foot touchdown. In addition, the reference trajectories for the MPC still require heuristic terms and tedious tuning procedures. Finally, MIP and MCTS are able to tackle the problem of contact planning, *e.g.,* [Amatucci et al., 2022; Ponton et al., 2021], but so far they have only been shown with high computational costs or in simple environments (*i.e.,* flat terrain) [Neunert et al., 2018].

The goal of this dissertation is to design an NMPC which demonstrates stability and robustness in nonflat terrains, reacting to the changes of the environment.
In particular, this work enriches the literature on online replanning methods by presenting a locomotion framework that removes the assumptions of heuristic reference generation and user-defined gait sequence.

Chapter 3

Nonlinear Model Predictive Control for Legged Locomotion

3.1 Introduction

Legged locomotion is a challenging task due to the under-actuation of the base and the interaction of the robot with the ground. Indeed the GRFs have a hybrid nature and the feet must establish and break the contact with the ground. MPC approaches compute feasible trajectories for a horizon of motion and update them, based on the new state measurement to close the loop, while the robot is moving. In particular, NMPC formulations have the ability to handle both the nonlinear system dynamics and the constraints, explicitly. The replanning frequency allows the algorithm to compensate for model inaccuracies, external disturbances and changes in the environment.

This chapter presents a high-frequency NMPC planning approach applied to the HyQ robot. The significance of this work is that it demonstrates how a suitably formulated NMPC can tackle rough terrain locomotion, and provide the optimal base orientation, while being real-time feasible. Differently from previous works of our lab (*e.g.,* [Mastalli et al., 2020; Fahmi et al., 2019]), online optimizations are performed.

Leg mobility is the capability of the robot leg to arbitrarily change its foot position. Optimizing for leg mobility allows the NMPC to devise a robot base orientation and height that improves locomotion on rough terrain (in contrast with the motions on flat terrain shown in [Neunert et al., 2018]). To the best of my knowledge, it is the first time that an MPC approach considers leg mobility during the locomotion. This is particularly useful to achieve the *environment adaptation*.

The method presented in this chapter has been obtained by collaborating with the PhD student Niraj Rathod and Prof. Mario Zanon and Prof. Alberto Bemporad from IMT School for Advanced Studies in Lucca. It resulted in a joint publication [Rathod et al., 2021]. Niraj and his professors focused on defining the OCP in Python exploiting `acados` library features, *e.g.,* choice of the solver, integrator scheme, sampling time, etc. for the definition of the NMPC. In addition, Niraj performed the offline analysis to compute the mobility factor (Section 3.4.1).

Outline

This chapter is organized as follows: Section 3.2 gives an overview of the entire planning pipeline whereas Section 3.3 describes the NMPC setup. The leg mobility and the other novel features introduced in this work are explained in Section 3.4, while the generation of the references and the Whole-Body Controller (WBC) are detailed in Sections 3.5 and 3.6, respectively. The description of the solver and the Real Time Iteration (RTI) scheme for the NMPC is provided in Section 3.7. Further, Section 3.8 illustrates simulation and experimental results with the HyQ robot. Finally, Section 3.9 presents the conclusion and the discussion regarding the contents of this chapter.

3.2 Locomotion Framework

Figure 3.1 illustrates the planning pipeline of the entire locomotion framework. In this dissertation, I mainly focus on the *Reference Generator*, *NMPC*, *WBC*.

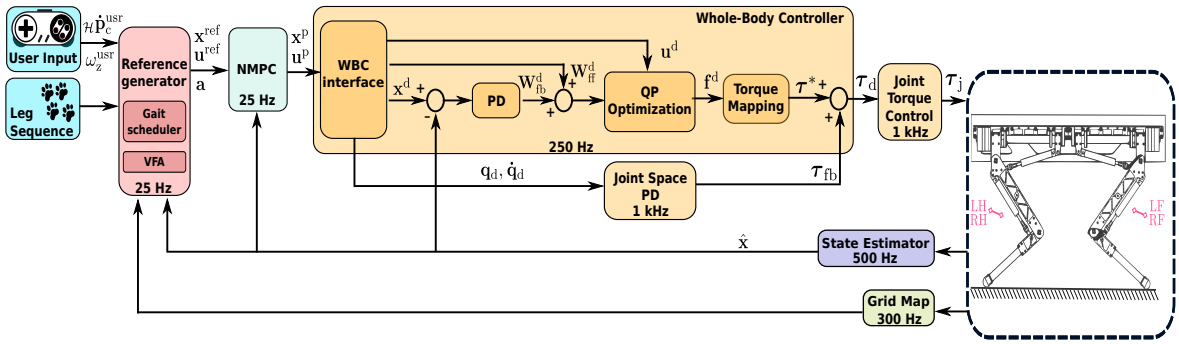

Figure 3.1: Block diagram of the planning pipeline with the NMPC in our locomotion framework. The reference generator provides the references ($\mathbf{x}^{\mathrm{ref}}$, $\mathbf{u}^{\mathrm{ref}}$) to NMPC after receiving the user inputs. Then, the NMPC passes optimal state $\mathbf{x}^{\mathrm{p}}$ and control $\mathbf{x}^{\mathrm{p}}$ trajectories to the Whole-Body Controller. The torque τ_{d} is given as reference to the low-level joint torque controller τ_{j}. The state estimator provides the state estimation $\hat{\mathbf{x}}$ to the required blocks. Finally, the heightmap is generated by Grid Map and given to the reference generator.

The *reference generator*, as discussed in Section 3.5, takes the user input (longitudinal, lateral and angular velocity), leg sequence (*e.g.,* Left Front (LF), Right Front (RF), Left Hind (LH),

Right Hind (RH)) gait timing, the initial state of the robot, and a map of the terrain to generate reference trajectories for the state $\mathbf{x}^{\text{ref}}$ (CoM position, orientation, linear and angular velocity) and control input $\mathbf{u}^{\text{ref}}$ (GRFs) required by the NMPC. While the quantities related to the gait are fixed, the user-defined velocities can be changed online while the robot moves. The reference generator also provides a vector of parameters $\mathbf{a}$ to the NMPC, that includes foot locations and sequences of contact status.
It is a heuristic module; in Chapter 4 I will present a more sophisticated version of this block, called *optimization-based reference generator*.

The NMPC runs at 25 Hz and delivers the optimal trajectories of the state $\mathbf{x}^{\text{p}}$ and control input $\mathbf{u}^{\text{p}}$, as detailed in Section 3.3. Reference generator and NMPC optimization are executed at the same frequency, such that updated references are provided at every execution of the optimization.

All the components of the Whole-Body controller (highlighted with the yellow box in Figure 3.1) are discussed in Section 3.6. The *WBC interface* interpolates the optimal state $\mathbf{x}^{\text{p}}$ at a rate of 250 Hz to generate a desired signal $\mathbf{x}^{\text{d}}$ for a Cartesian virtual impedance controller [Focchi et al., 2016]. The WBC interface also computes the feedforward wrench $\mathbf{W}_{\text{ff}}^{\text{d}}$ that is added to a feedback wrench $\mathbf{W}_{\text{fb}}^{\text{d}}$ that renders the Cartesian impedance. Moreover, the WBC interface provides the reference joint positions $\mathbf{q}_{\text{d}}$ and velocities $\dot{\mathbf{q}}_{\text{d}}$ to a Joint Space PD controller running at 1 kHz in parallel to the WBC. After acquiring the feedback and feedforward wrenches, a QP optimization computes the vector of desired GRFs $\mathbf{f}^{\text{d}}$ accounting for the friction cone constraints and penalizing the difference between $\mathbf{f}^{\text{d}}$ and $\mathbf{u}^{\text{p}}$ coming from the NMPC solution [Fahmi et al., 2019]. Then, $\mathbf{f}^{\text{d}}$ is mapped to the torque vector $\boldsymbol{\tau}^{*}$ that is added to the Joint Space PD torques $\boldsymbol{\tau}_{\text{fb}}$ resulting in the total desired torque $\boldsymbol{\tau}_{\text{d}}$. Ultimately, $\boldsymbol{\tau}_{\text{d}}$ is passed to a low-level joint torque controller as reference [Boaventura et al., 2015].

An online state estimator [Nobili et al., 2017] that runs at 500 Hz provides the estimation of the robot state $\hat{\mathbf{x}}$ to all the components inside our locomotion framework that require it. A dedicated on-board computer takes inputs from an RGB-D camera (RealSense) mounted in front of the robot and generates a 2.5D heightmap, *i.e.,* a two-dimensional discrete representation of the terrain where each pixel describes the height of a specific area. The *Grid Map* library [Fankhauser and Hutter, 2016] computes a new heightmap at a frequency of 30 Hz which is finally sent to the reference generator.

3.3 Nonlinear MPC Formulation

Let us define the decision variables as the predicted state and control input with $\mathbf{x}^{\text{p}} = \{\mathbf{x}_0, \dots, \mathbf{x}_N\}$ and $\mathbf{u}^{\text{p}} = \{\mathbf{u}_0, \dots, \mathbf{u}_{N-1}\}$, respectively, such that a NLP formulation can be

stated as:

$$\min_{\mathbf{x}^{\mathrm{p}},\mathbf{u}^{\mathrm{p}}} \quad \sum_{k=0}^{N-1} \ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) + \ell_{\mathrm{T}}\left(\mathbf{x}_N\right) \tag{3.1a}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_{\mathbf{0}}, \tag{3.1b}$$

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right), \qquad k \in \mathbb{I}_0^{N-1}, \tag{3.1c}$$

$$h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \tag{3.1d}$$

where, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the stage cost function; $\ell_{\mathrm{T}} : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal cost function. The initial condition of Eq. (3.1b) is expressed by setting $\mathbf{x}_0$ equal to the state estimate $\hat{\mathbf{x}}_{\mathbf{0}}$ received from the state estimator. The vector of model parameters $\mathbf{a}_k$ is not optimized but it is computed externally by the reference generator and provided to the optimization problem formulation. The nonlinear system dynamics are introduced by the equality constraints of Eq. (3.1c). Finally, the path constraints are included using Eq. (3.1d) which, for example, can be bounds on the decision variables. The NLP (3.1) is defined for a *prediction horizon* T that is divided into N discrete time *control intervals* of lengths $T_{\mathrm{s}} = \frac{T}{N}$. Hereafter, we will refer to T_{s} as the *sampling time*.

3.3.1 Cost Function

The cost function is designed in order to impose high-level locomotion tasks.
The running term in Eq. (3.1a) is composed of three terms:

$$\ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) = \ell_{\mathrm{t}} + \ell_{\mathrm{m}} + \ell_{\mathrm{r}}, \tag{3.2a}$$

$$\ell_{\mathrm{t}} = \parallel \mathbf{x}_k - \mathbf{x}_k^{\mathrm{ref}} \parallel_{\mathbf{Q}}^2 + \parallel \mathbf{u}_k - \mathbf{u}_k^{\mathrm{ref}} \parallel_{\mathbf{R}}^2, \tag{3.2b}$$

$$\ell_{\mathrm{m}} = \parallel {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k} - {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k}^{\mathrm{ref}} \parallel_{\mathbf{M}}^2, \tag{3.2c}$$

$$\ell_{\mathrm{r}} = \parallel {}_{\mathcal{K}}\mathbf{u}_k \parallel_{\mathbf{P}}^2 \tag{3.2d}$$

- The tracking cost in Eq. (3.2b) is associated to state and control input and the reference trajectories $\mathbf{x}_k^{\mathrm{ref}}, \mathbf{u}_k^{\mathrm{ref}}$ are provided by the reference generator for each sampling instance k (refer to Section 3.5). For example, it guarantees that the robot follows the user-defined velocities.

- The mobility cost (3.2c) is a novel term which is introduced in this chapter. It accounts for improving the leg mobility by penalizing the difference between the hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}$ and the reference value ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ of maximum mobility. This cost allows the NMPC to optimize the robot base orientation (*e.g.,* align it automatically to the terrain shape) in order to increase the leg mobility which has as a desirable effect of having joint quantities far from kinematic limits during locomotion. The derivation of ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ is detailed separately in Section 3.4.1.

- In some locomotion scenarios [Focchi et al., 2016], to cope with uncertainties in the contact normal estimation and to increase robustness to external disturbances, it is desirable to have the GRFs $\mathbf{f}_i$ as close as possible to the center of the friction cone. This can be achieved by penalizing X-Y components of $\mathbf{u}$ in a frame $\mathcal{K}$ (see Figure 3.2) that is aligned to the normal of the contact and it is included in our cost function by a control input regularization term (3.2d), refer to Section 3.4.3 for the details.

The positive definite weight matrices $\mathbf{Q} \in \mathbb{S}_+^{n_x}, \mathbf{R} \in \mathbb{S}_+^{n_u}$, $\mathbf{M} \in \mathbb{S}_+^{n_e}$, $\mathbf{P} \in \mathbb{S}_+^{n_u}$ act as important tuning parameters in the NMPC formulation, since they determine the priority among the three different terms. Finally, we define the terminal cost $\ell_{\mathrm{T}} = \| \mathbf{x}_N - \mathbf{x}_N^{\mathrm{ref}} \|_{\mathbf{Q}_N}$ and use the weight matrix $\mathbf{Q}_N = \mathbf{Q}$ for this cost.

3.3.2 Robot Model

The NMPC exploits three reference frames: the world frame $\mathcal{W}$, the CoM frame $\mathcal{C}$, and the already mentioned contact frame $\mathcal{K}$; Figure 3.2 shows a schematic of the robot and location/orientation of the reference frames. While $\mathcal{W}$ is a fixed inertial frame, the CoM frame is aligned with the base of the robot and its origin is located always at the CoM. A variable with left subscript denotes its frame of reference. For example ${}_{\mathcal{C}}\boldsymbol{\omega}$ represents the angular velocity of the robot base expressed in the CoM frame $\mathcal{C}$. Note that, unless explicitly specified, all the relevant quantities are defined in the inertial frame $\mathcal{W}$.

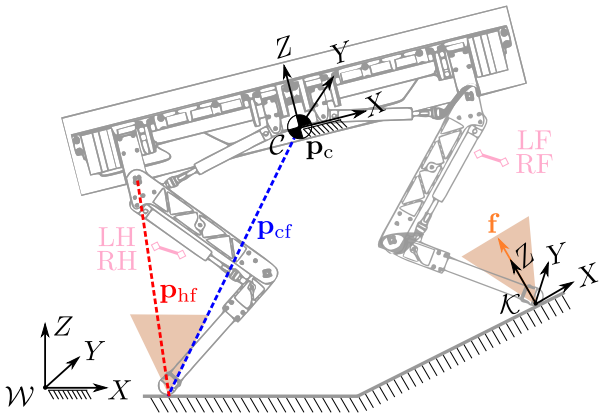

Figure 3.2: HyQ schematic showing the inertial frame ($\mathcal{W}$), the CoM frame ($\mathcal{C}$) attached to the CoM of the robot, and the contact frame ($\mathcal{K}$). The robot legs are shown in the *default* configuration.

As explained in Section 2.1.2, the choice of the model is a key element. In order to find the trade-off between accuracy and fast replanning, the dynamics is expressed by a simplified reduced-order SRBD model. Defined in a 6D space, it describes the translational and angular dynamics of the robot while neglecting the dynamics of its swinging legs. This is a valid

approximation for the HyQ robot because most of its mass is concentrated in the base, as mentioned in [Semini, 2010] (the mass of the base is 66.7 kg and the mass of each leg is 5.08 kg). The robot is approximated as a rigid body with the inertia computed considering the robot in a default leg configuration as shown in Figure 3.2.

To mitigate the effect of the approximations, the SRBD model is written in the CoM frame (specifically the angular dynamics); this choice yields a constant inertia tensor. Thus, the angular dynamic equations are much simpler *i.e.,* less nonlinear because the inertia tensor is not time-varying. In the SRBD model, GRFs are applied as inputs to control the position and orientation of the robot base. Recalling Eq. (2.7), the SRBD model is:

$$m\dot{\mathbf{v}}_{\mathrm{c}} = m\mathbf{g} + \sum_{i=1}^{4} \delta_i \mathbf{f}_i \tag{3.3a}$$

$${}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\, {}_{\mathcal{C}}\dot{\boldsymbol{\omega}} + {}_{\mathcal{C}}\boldsymbol{\omega} \times {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\, {}_{\mathcal{C}}\boldsymbol{\omega} = \sum_{i=1}^{4} \delta_i\, {}_{\mathcal{C}}\mathbf{p}_{\mathrm{cf},i} \times {}_{\mathcal{C}}\mathbf{f}_i \tag{3.3b}$$

where m is the robot mass, $\dot{\mathbf{v}}_{\mathrm{c}} \in \mathbb{R}^3$ is the CoM acceleration, $\mathbf{g}$ is the gravitational acceleration, $\mathbf{f}_i \in \mathbb{R}^3$ is the ground reaction force at foot i, ${}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}} \in \mathbb{R}^{3\times3}$ is the inertia tensor computed at the CoM frame origin, ${}_{\mathcal{C}}\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration of the robot's base, $\mathbf{p}_{\mathrm{cf},i} \in \mathbb{R}^3$ is the distance between the CoM position $\mathbf{p}_{\mathrm{c}} \in \mathbb{R}^3$ and the position $\mathbf{p}_{\mathrm{f},i} \in \mathbb{R}^3$ of foot i. The binary parameter $\delta_i = \{0, 1\}$ (*contact status*) defines whether the foot i is in contact with the ground and can therefore generate contact forces. The robot dynamics governed by Eq. (3.3) can be expressed as the continuous-time state-space model:

$$\begin{bmatrix} \dot{\mathbf{p}}_{\mathrm{c}} \\ \dot{\mathbf{v}}_{\mathrm{c}} \\ \dot{\boldsymbol{\Phi}} \\ {}_{\mathcal{C}}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{\mathrm{c}} \\ 1/m \sum_{i=1}^{4} \delta_i \mathbf{f}_i + \mathbf{g} \\ \mathbf{E}'^{-1}(\boldsymbol{\Phi})\, {}_{\mathcal{C}}\boldsymbol{\omega} \\ -{}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}^{-1}({}_{\mathcal{C}}\boldsymbol{\omega} \times {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\, {}_{\mathcal{C}}\boldsymbol{\omega}) + \sum_{i=1}^{4} \delta_i\, {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}^{-1}\, {}_{\mathcal{C}}\mathbf{p}_{\mathrm{cf},i} \times {}_{\mathcal{C}}\mathbf{f}_i \end{bmatrix} \tag{3.4}$$

The robot base orientation is represented by the sequence of Z-Y-X Euler angles[1] [Diebel, 2006] $\boldsymbol{\Phi} = (\phi, \theta, \psi)$ *i.e.,* roll (ϕ), pitch (θ) and yaw (ψ), respectively. The relation between the Euler Angles rates $\dot{\boldsymbol{\Phi}}$ and angular velocity ${}_{\mathcal{C}}\boldsymbol{\omega}$ is well-known and discussed in Appendix A for the sake of completeness.

The state $\mathbf{x}$ is the stack of CoM position, linear velocity, orientation and angular velocity $\mathbf{x} \in \mathbb{R}^{n_x} = (\mathbf{p}_{\mathrm{c}}, \mathbf{v}_{\mathrm{c}}, \boldsymbol{\Phi}, {}_{\mathcal{C}}\boldsymbol{\omega})$ with $n_x = 12$. The control input is represented by the GRFs $\mathbf{u} \in \mathbb{R}^{n_u} = (\mathbf{f}_1, \ldots, \mathbf{f}_4)$ with $n_u = 12$. Equation (3.4) can be concisely written as:

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a}(t)), \tag{3.5}$$

[1]Note that Euler angles can suffer from singularities that occur in certain configurations [Younes et al., 2012]. Because in this work we do not consider motions that involve such configurations, using Euler angles does not pose any issue. A singularity-free implementation is out of the scope of this work and is left for future research.

where $\mathbf{a} \in \mathbb{R}^{n_p} = (\mathbf{p}_\mathrm{f}, \boldsymbol{\delta})$ with $n_p = 16$ is a vector of parameters that includes the feet positions $\mathbf{p}_\mathrm{f} \in \mathbb{R}^{n_e}$, with $n_e = 12$ and the contact status $\boldsymbol{\delta} \in \mathbb{R}^4$.

The rigid-body dynamics in Eq. (3.5) is discretized using different numerical integration techniques [Quirynen, 2017; Butcher, 2003; Hairer et al., 1993, 1996] to obtain the discrete-time model:

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right), \tag{3.6}$$

which defines equality constraints (3.1c) imposed at every stage k to ensure that the state trajectory satisfies the system dynamics for the given control inputs.

One specific feature of legged robots is the need to ensure that the values of the GRFs are equal to zero for a swinging leg. This is typically done by introducing complementarity constraints [Posa et al., 2014], *e.g.,* [Cebe et al., 2021; Villarreal et al., 2020]. These constraints, however, pose several difficulties in solving the optimization problem, since the vast majority of the NLP algorithms cannot handle them and tailored solvers are required. Ultimately, this results in a significant increase in computation time. An alternative to complementarity constraints commonly adopted in the robotic community consists in removing the hybrid nature of the dynamics by providing the sequence of contact status $\boldsymbol{\delta}$ as input parameters in the state space model (3.5). In this manner, a contact mode δ_i is multiplied with the terms involving force $\mathbf{f}_i$ in Eq. (3.4) and the effect of that force in the linear and angular dynamics is nullified during the swing phase of the corresponding leg i. Hence, there is no more need to include complementarity constraints separately in Eq. (3.1),S which results in fewer constraints and, consequently, in a relatively smaller NMPC formulation.

3.3.3 Friction cone and unilateral constraints

Path constraints guarantee feasibility and robustness of the optimized trajectories. In particular, by imposing that the GRFs are inside the friction cone, it is ensured that the feet in stance do not slip. Friction cone constraints are encoded with their square pyramid approximation:

$$-\mu_i \mathbf{f}_{\mathrm{z},i} \leq \mathbf{f}_{\mathrm{x},i} \leq \mu_i \mathbf{f}_{\mathrm{z},i} \tag{3.7a}$$

$$-\mu_i \mathbf{f}_{\mathrm{z},i} \leq \mathbf{f}_{\mathrm{y},i} \leq \mu_i \mathbf{f}_{\mathrm{z},i} \tag{3.7b}$$

$$\underline{\mathbf{f}}_\mathrm{z} \leq \mathbf{f}_{\mathrm{z},i} \leq \overline{\mathbf{f}}_\mathrm{z} \tag{3.7c}$$

where, $\underline{\mathbf{f}}_\mathrm{z}$ and $\overline{\mathbf{f}}_\mathrm{z}$ are upper and lower bounds on GRFs Z component, respectively, and μ_i is the friction coefficient of the contact surface. Choosing $\underline{\mathbf{f}}_z$ greater than or equal to zero enforces unilateral constraints on the normal forces $\mathbf{f}_\mathrm{z}$. The friction cone and unilateral constraint are represented by $h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) \leq 0$ in the NMPC formulation, in Eq. (3.1d).

3.4 Locomotion-Enhancing Features

This section discusses the main distinctive features of the presented approach, focusing on the relevant contributions to improve the locomotion ability of our quadruped robot. These features are *mobility*, *force robustness*, and *ZMP margin*.

3.4.1 Mobility and Mobility Factor

Terrain adaptability is vital when it comes to the locomotion of the legged robots. Adjusting the posture of the robot depending on the environment is important for safe locomotion. A way to enable the NMPC to choose robot orientation adaptively to any terrain is to employ the concept of *mobility* [Focchi et al., 2017]. In order to rigorously discuss mobility in mathematical terms, it is defined in words as the attitude/capability of a manipulator (leg) to change end-effector position/orientation [Sciavicco and Siciliano, 2000] arbitrarily.

In order to penalize low-leg mobility in the cost function of Eq. (3.2c) the reference value of hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}^{\text{ref}}_{\text{hf}_k}$ must be provided. The goal of this section is to define a convenient metric to represent mobility and compute ${}_{\mathcal{C}}\mathbf{p}^{\text{ref}}_{\text{hf}_k}$ corresponding to the maximum value of such a metric. Among several ways to compute mobility, the velocity transformation ratio [Yoshikawa, 1984] allows one to evaluate mobility in a particular direction. However, the velocity transformation ratio cannot be used in this setting because it requires prior knowledge of the evolution of the relative foot position with respect to CoM. While foot position is known before the optimization, since it is computed by the reference generator, CoM position is an output of the CoM. Thus the velocity transformation cannot be evaluated in advance.

As an alternative approach, the volume of the *manipulability ellipsoid* is considered as a metric to evaluate mobility [Sciavicco and Siciliano, 2000].

$$\mathbf{v}(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^{\top})^{-1}\mathbf{v} = 1$$

A change in the volume of the manipulability ellipsoid with different leg configurations is visualized in Figure 3.3 (left). By inspecting Figure 3.3 (top right), it can be seen that the maximum volume V is in the vicinity of the most extended leg configuration because the mobility becomes very big in the X and Y directions, even if it is still very limited in the Z direction. However, because it is desirable to achieve good mobility in all the directions of the leg configuration, a better metric is the one that also accounts for the *isotropy* of the manipulability ellipsoid. A measure of the isotropy of an ellipsoid can be expressed as the inverse of its *eccentricity* E. Hence, a new manipulability index named *mobility factor* Eq. (3.8) can be defined in terms of both the eccentricity and the volume of the manipulability ellipsoid.

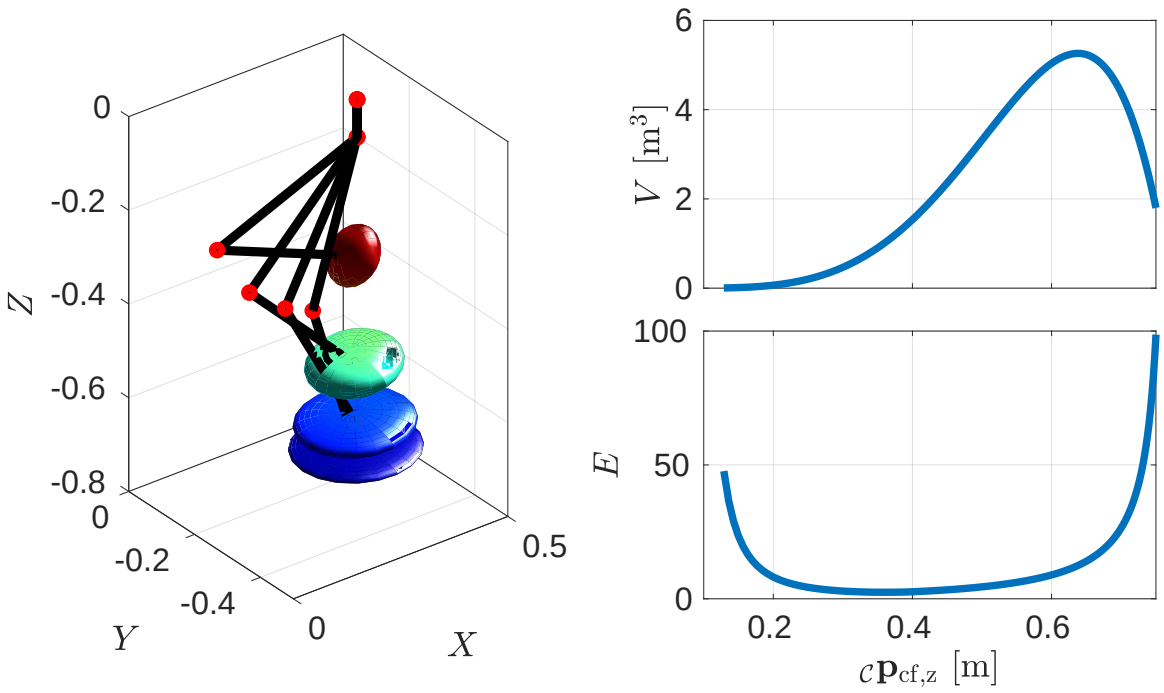

Figure 3.3: Manipulability ellipsoid changing with leg configuration (left) of the right front leg. Volume of the ellipsoid (top right) and Eccentricity of the ellipsoid (bottom right).

Again from Figure 3.3, it can be visualized that to keep a good mobility (left plot) in all directions, the volume (top right plot) should be maximized while the eccentricity (bottom right plot) should be as small as possible. Defining a foot Jacobian $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{3\times3}$ computed at a particular joint configuration $\mathbf{q}$, the volume V of a manipulability ellipsoid is evaluated as a product of the eigenvalues of $\left(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^\top\right)^{-1}$ while the eccentricity is the ratio between its maximum and minimum eigenvalue [Focchi et al., 2017]. First, the volume and eccentricity of manipulability ellipsoid are normalized by their ranges $\bar{V}$ and $\bar{E}$. Then the mobility factor is:

$$m_\mathrm{f} = \beta\frac{V}{\bar{V}} - \gamma\frac{E}{\bar{E}} \tag{3.8}$$

The minus sign in Eq. (3.8) represents conflicting contributions of the V and E in the definition of the mobility factor (*i.e.,* the goal is to achieve high volume and low eccentricity). Parameters β and γ are introduced to find the best trade-off between volume and eccentricity.

The mobility factor is a convex nonlinear function $m_\mathrm{f} : \mathbb{R}^3 \to \mathbb{R}$ that can be numerically evaluated inside the workspace of each leg. By selecting $\beta = 1$ and $\gamma = 4$, and after conducting a numerical analysis for all the feet positions in the workspace of a leg of the HyQ robot I found that an hip-to-foot vector of ${}_\mathcal{C}\mathbf{p}_\mathrm{hf} = (0, 0, -0.55)\,\mathrm{m}$ maximizes m_f. Figure 3.4 (left) shows a slice of the scalar function m_f in the X-Y plane for ${}_\mathcal{C}\mathbf{p}_{\mathrm{hf},z} = -0.55\,\mathrm{m}$ obtained for the RF leg. Instead, in Figure 3.4 (right) I plot m_f against the change of foot position in the Z direction

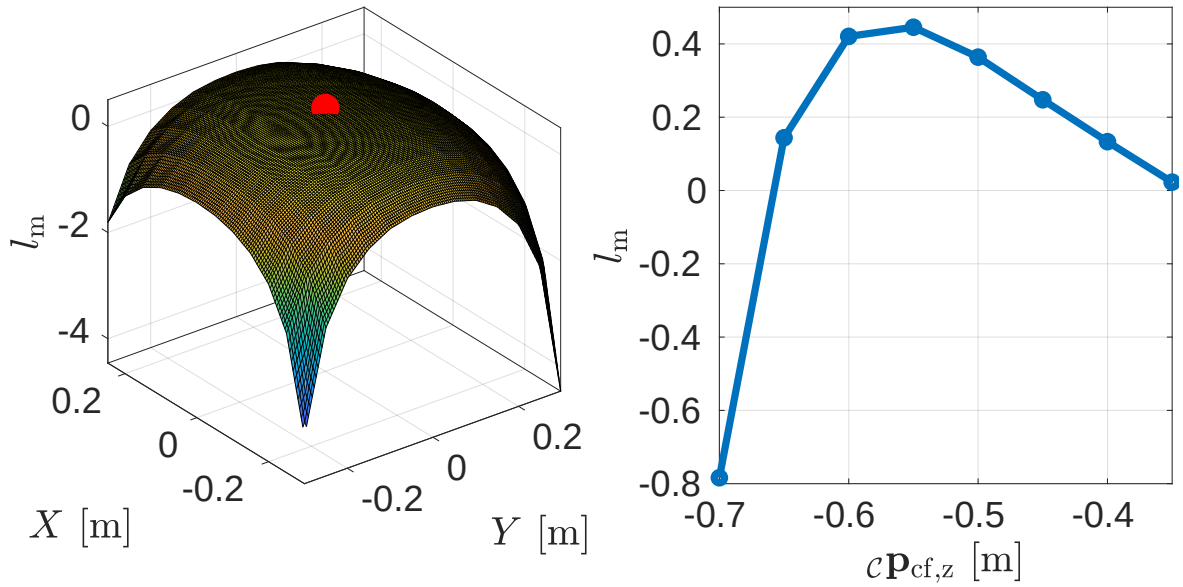

Figure 3.4: Slices of the mobility factor function for the RF leg: the left figure plots it against the X-Y components keeping Z constant. The red dot in the left plot represents the maxima. The right figure plots it against the Z component for a constant X-Y foot position.

considering the hip under the foot (X=0, Y=0) which clearly highlights $^{\mathcal{C}}\mathbf{p}_{\mathrm{hf,z}} = -0.55\,\mathrm{m}$ corresponding to the maximum value of the mobility factor m_{f} (*i.e.,* around 0.41). The output of this analysis is the reference for the hip-to-foot term $^{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ in the mobility cost Eq. (3.2c) for all the legs. The obtained numbers are specific for HyQ, but the same methodologies can be applied to any kind of robots.

In the mobility cost of Eq. (3.2c), the contact status δ_i multiplies the term corresponding to the i^{th} leg. Thus, the mobility cost solely accounts for stance legs because the robot can only use them to control its base orientation. The advantage of a mobility cot is that it enables NMPC to provide the optimal base orientation for a particular locomotion that retains mobility, there is no need to separately specify tracking cost for roll and pitch in the NMPC. This relieves a user from the burden of implementing a customized heuristic (*e.g.,* to align the robot base to the terrain), as was necessary in, *e.g.,* [Focchi et al., 2020; Gehring et al., 2015; Villarreal et al., 2020]. The relative tracking task for the CoM Z position is no longer required either, because maximizing the mobility in the Z direction automatically takes care of keeping an average distance of hips from the terrain to $^{\mathcal{C}}\mathbf{p}_{\mathrm{hf,z}}$, consequently keeping the robot base at a certain height.

Moreover, the yaw motion results by penalizing the mobility cost along the X-Y directions. This has the effect of driving the hips of the robot base over the feet, naturally aligning the base to the feet, similar to what was done in [Raiola et al., 2020]. However, a tracking cost on yaw was still necessary in the NMPC to track the heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}}$ commanded by the

user and to avoid oscillations.

Remark: The concept of mobility is model-independent hence, it can also be used with other models, such as full body dynamics in the MPC setting.

3.4.2 ZMP Margin

In legged locomotion, the robot is often operated close to unstable configurations which require a controller to continuously compensate for model inaccuracies and external disturbances while maintaining locomotion stability. However, in a configuration in which the ZMP [Wieber, 2006] is close to the boundary of the support polygon [Bretl and Lall, 2008] there is a higher chance that the robot can fall as a consequence of small perturbations.

The reference generator computes references for the GRFs by dividing the robot mass by the number of legs as explained in Section 3.5. Penalizing GRFs Z component heavily in the tracking cost of Eq. (3.2b) ensures that they stay close to the reference, consequently maintaining a higher loading on the diagonally opposite leg to the swinging one, and therefore maintaining some margin for the locomotion stability. Properly tuning the weights of Eq. (3.2) reduces the complexity of the formulation, since it relieves from the need to impose an additional constraint, *e.g.,* ZMP inside the support polygon.

To evaluate the locomotion stability, the *ZMP margin* is computed as the minimum of the distance of the ZMP from each support polygon edge, *i.e.,*

$$m_{\mathrm{c}} = \min(\mathbf{d}) \tag{3.9}$$

where $\mathbf{d}$ is a vector of the distances of ZMP projection (on a horizontal plane) from the support polygon edges.

3.4.3 Force Robustness

Similar to the considerations on mobility, in order to effectively compensate for disturbances acting on the system, robustness in the GRFs is required. The closer the GRF is to the friction cone boundary, the less lateral force the robot is able to withstand (due to perturbations) without suffering from slippage. This behaviour enhances the importance of an algorithm endowed with visual feedback from the environment.

An approach penalizing GRFs that are in the vicinity of the cone boundaries has been proposed in [Focchi et al., 2016; Fahmi et al., 2020] inside the WBC. These WBC based approaches instantaneously generate GRFs that are as close as possible to the normals of the cones while yielding the prescribed resultant wrench on the robot base. However, WBC does not account for the future state of the robot and hence, it leaves some room for the NMPC to compensate

for the contact normal estimation error and recover from external disturbances. Introducing these margins on GRFs from the cone boundaries is especially important in some scenarios, such as the V-shaped chimney reported in simulation in Section 3.8.3.

In this approach, a similar idea to [Focchi et al., 2016; Fahmi et al., 2020] is adopted providing the cost function with an additional term of Eq. (3.2d) in the NMPC, which penalizes the tangential components of GRFs in the contact frame $\mathcal{K}$ (see Figure 3.2). Having X-Y components smaller than Z component in $\mathcal{K}$ of GRFs determines that the resultant GRFs are as close as possible to the contact normals. The *Grid Map* module is responsible for extracting the normal of the terrains from the pointcloud of the images taken by the camera. The weight matrix $\mathbf{P}$ used in this cost is defined in Table 3.2 (Section 3.8).

3.5 Reference Generator

The NMPC requires a reference trajectory of the state and control input along with the model parameters *i.e.,* foot positions and contact status. This reference trajectory also serves as an initial guess for the first run of the NMPC. The references are generated for the length of control intervals N, and the reference generator is called before every iteration of the NMPC in order to obtain prompt adaptation to terrain changes and user set-point. The reference generator is based on heuristics and takes as inputs:

- the user commanded longitudinal and lateral CoM velocity ${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ in the horizontal frame $\mathcal{H}$,[2]
- user commanded heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}} \in \mathbb{R}$,
- current pose of the robot $(\mathbf{p}_{\mathrm{c}}, \boldsymbol{\Phi})$,
- current feet positions $\mathbf{p}_{\mathrm{f}} \in \mathbb{R}^{12}$,
- heightmap of the terrain

The output of the reference generator are:

- the references for the NMPC cost: states $\mathbf{x}^{\mathrm{ref}} \in \mathbb{R}^{n_x \times (N+1)}$, control $\mathbf{u}^{\mathrm{ref}} \in \mathbb{R}^{n_u \times N}$,
- parameters $\mathbf{a}$ of the model: sequence of the contact status $\boldsymbol{\delta}$ $(\in \mathbb{R}^{4\times N})$ and sequence of the foot locations $\mathbf{p}_{\mathrm{f}}$ $(\in \mathbb{R}^{12\times N})$,
- normals $\mathbf{n}$ $(\in \mathbb{R}^{12\times N})$ of the terrain at the foothold locations, which are provided as inputs to the NMPC for the cone constraints and to define the contact frame $\mathcal{C}$.

[2]The horizontal frame is placed like the CoM frame but with the Z-axis aligned with the gravity. In other words, it corresponds to $\mathcal{C}$ having zero pitch and roll.

First the X-Y components of the total velocity $\mathbf{v}_{\mathrm{c}}^{\mathrm{ref}} \in \mathbb{R}^3$ is computed; they depend on both $\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ and the X-Y components of the tangential velocity due to the heading velocity $\boldsymbol{\omega}^{\mathrm{usr}} = (0, 0, \omega_{\mathrm{z}}^{\mathrm{usr}})$.

$$\mathbf{v}_{\mathrm{c,(x,y)}}^{\mathrm{ref}} = \mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} + (\boldsymbol{\omega}^{\mathrm{usr}} \times \mathbf{p}_{\mathrm{c}}^{\mathrm{ref}})_{\mathrm{(x,y)}} \tag{3.10}$$

The X-Y CoM position $\mathbf{p}_{\mathrm{c,(x,y)}}^{\mathrm{ref}}$ is obtained by integrating the $\mathbf{v}_{\mathrm{c,(x,y)}}^{\mathrm{ref}}$ (rotated in the world frame) with the explicit Euler scheme. The references for CoM Z, roll and pitch are set to 0 because we do not track them in the NMPC cost of Eq. (3.2b). Instead, the reference for the yaw ψ is obtained by integrating the user-defined yaw rate $\dot{\psi}^{\mathrm{usr}}$ with $\dot{\boldsymbol{\Phi}}^{\mathrm{usr}} = \mathbf{E}^{-1}(\boldsymbol{\Phi}^{\mathrm{ref}})\boldsymbol{\omega}^{\mathrm{usr}}$ (see Appendix A for the transformation between angular velocity and Euler rates). The reference for angular velocity, instead, coincides with $\boldsymbol{\omega}^{\mathrm{usr}}$.

The references for GRFs $\mathbf{u}^{\mathrm{ref}}$ are calculated by simply dividing the total mass of the robot by the number of legs in stance. Dividing the forces equally onto the legs is correct only if the robot is static. Still, in the case of dynamic conditions, it is a better approximation than passing no references to the NMPC. This general limitation of the reference generators will be discussed and addressed in the following Chapter.

The sequence of contact status $\boldsymbol{\delta}$ and of footholds are computed by the *gait scheduler* and *robocentric stepping* strategy, respectively. It is important to mention that the reference generator does not compute the swing trajectories and they are obtained from the WBC interface discussed in Section 3.6.1.

3.5.1 Gait scheduler

The gait scheduler is logically decoupled from the reference trajectory generation and determines if a leg is either in swing or stance (δ_i) at each time instance for the entire gait cycle as shown in Figure 3.5.

The leg duty factor D_i and offsets o_i can be used to encode different gaits such as crawl, trot, and pace. The offset represents the moment in which the leg i begins a swing motion; the duty factor indicates the time in which the leg i is in stance. Figure 3.5 shows the values for a walk motion, *i.e.,* one legs move at a time. For example, the trot gait is obtained by choosing the same values of o_i for the two diagonal legs (LF-RH and RF-LH).

The gait scheduler implements a time parametrization $s \in [0, 1]$ (*stride phase*) which is normalized about the cycle time duration T_{c} such that the leg duty factor D_i and offsets o_i are independent of the cycle time. Each trigger l_i^{tr} (red arrow in Figure 3.5) corresponds to a new lift-off event. We can express the value of δ for leg i as:

$$\delta_i = \begin{cases} 1, & s < o_i \vee s > ((o_i + (1 - D_i)) \bmod 1) \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

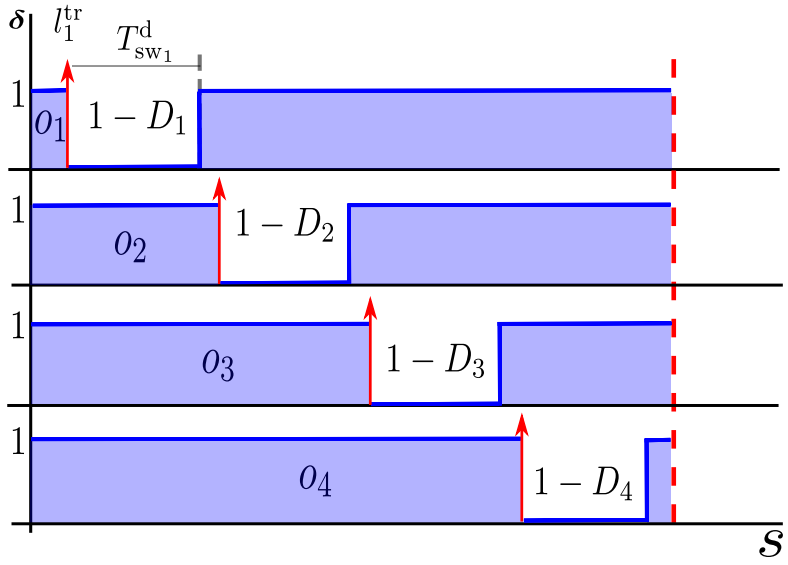

Figure 3.5: Gait schedule for a walk. Offsets $\mathbf{o} = [0.05, 0.3, 0.55, 0.8]$, duty-factors $\mathbf{D} = [0.85, 0.85, 0.85, 0.85]$. The red arrows represent the trigger l_i^{tr} for a swing leg i. The values of offsets and duty-factors can be used to encode all the gaits.

Every time the reference generator is called, it extracts N points from the gait schedule starting from an index called *gait counter*. It keeps the memory of the index of the gait schedule achieved by the previous call of the reference generator.

The synchronization between the first point of a contact sequence δ_{i_k} computed by the reference generator and the actual contact state of the robot is an important challenge, since it prevents the reference generator from computing a zero reference force while the leg is in stance and vice-versa. In case of premature or delayed touchdown events, the synchronization is lost and the gait counter is shifted backwards or forward to re-conciliate the planned touchdown with the actual touchdown as shown in Figure 3.6. This is a crucial feature when dealing with rough terrains. In particular, for the late touchdown, the gait counter is moved such that the touchdown happens at the second node of the horizon. This guarantees that at least the first value of the force (which is the only control input applied to the robot) is nonzero.

3.5.2 Robocentric stepping

The choice of the footholds is a key element in locomotion, since it deals with the kinematic limits of the robot. Inspired by [Raibert, 1986], I use a heuristic approach that continuously computes footholds consistent with the current position of the robot. To compute a foothold for a swinging leg i, we consider its hip position $\mathbf{h}_i$ instead of using the foot position at the moment of the lift-off. In this way a disturbance acting on the robot or a tracking error that

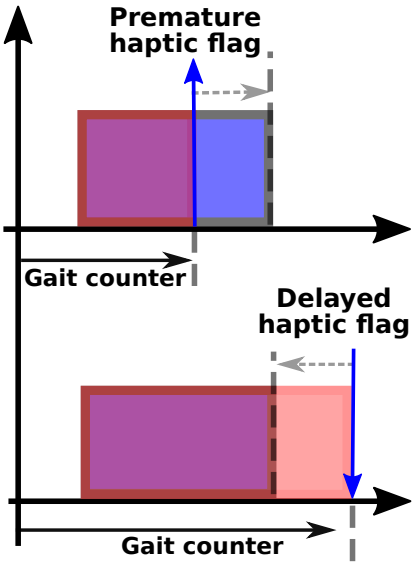

Figure 3.6: Fast-forwarding (top) and re-winding (bottom) of the gait counter to recover the synchronization between actual (haptic) and planned touchdown.

occurred during a swing can be recovered in the following swing. In this section, I drop the index i to simplify the notation. The lift-off trigger is a variable which is set to 1 at the first node of the swing phase:

$$l_k^{\mathrm{tr}} \triangleq \delta_k \wedge \overline{\delta}_{k+1}$$

The foot position $\mathbf{p}_{\mathrm{f}_{k+1}}$ is computed as:

$$\mathbf{p}_{\mathrm{f}_{k+1}} = \begin{cases} \mathbf{p}_{\mathrm{f}_k}^{\mathrm{td}} & l_k^{\mathrm{tr}} = 1 \\ \mathbf{p}_{\mathrm{f}_k} & l_k^{\mathrm{tr}} = 0 \end{cases} \tag{3.12}$$

Notice that at the lift-off condition $l^{\mathrm{tr}} = 1$ at instance k, $\mathbf{p}_{\mathrm{f}}$ is set equal to the touchdown point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and it is kept constant until the next lift-off event occurs. As already explained, $\mathbf{p}_{\mathrm{f}}$ has no effects during the swing on the dynamics of Eq. (3.3a) and Eq. (3.3b) since it is multiplied by $\delta = 0$. The X-Y component of the touchdown point is given by:

$$\mathbf{p}_{\mathrm{f}_k,(\mathrm{x,y})}^{\mathrm{td}} = \mathbf{h}_k + \alpha T_{\mathrm{sw}}^{\mathrm{d}}(\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} + (\boldsymbol{\omega}^{\mathrm{usr}} \times \mathbf{p}_{\mathrm{bh}})_{(\mathrm{x,y})}) \tag{3.13}$$

The second term in Eq. (3.13) represents the step length (red arrow in Figure 3.7) which is computed with respect to the hip instead of the previous foot location. Parameter α is an empirically chosen scaling factor. Parameter $T_{\mathrm{sw}}^{\mathrm{d}}$ is the default swing duration computed starting from user-defined offsets $\mathbf{o}$ and duty-factors $\mathbf{D}_{\mathrm{f}}$. The distance between the hip and the center of the base is denoted by $\mathbf{p}_{\mathrm{bh}} \in \mathbb{R}^3$. A 2.5D heightmap of the terrain is evaluated in correspondence of the touchdown point $\mathbf{p}_{\mathrm{f}_k,(\mathrm{x,y})}^{\mathrm{td}}$ to obtain $\mathbf{p}_{\mathrm{f}_k,\mathrm{z}}^{\mathrm{td}}$ that does not penetrate the terrain.

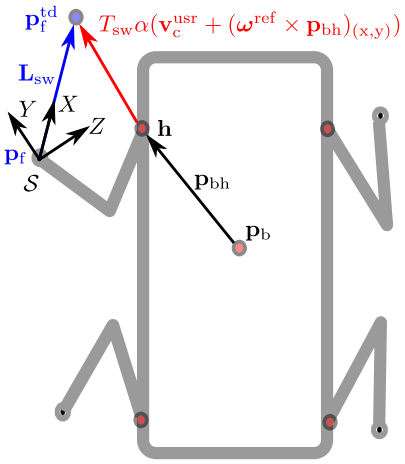

Figure 3.7: Representation of the robocentric stepping strategy and of the Swing Frame, located at the lift-off point. The red arrow shows the distance between the touchdown point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and the hip $\mathbf{h}$. The blue vector $\mathbf{L}_{\mathrm{sw}}$ connects lift-off and touchdown point.

If $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ is located near to an edge or leads to collisions (*e.g.,* of the foot or the shin) during the step cycle, this can be harmful to the robot's balance. To prevent this from happening, the robot acquires a local heightmap in the vicinity of the touchdown point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and adjusts the foot landing location using the VFA module presented in [Villarreal et al., 2019]. Even though the footholds are not directly optimized, the VFA improves the heuristics of the reference generator, providing the optimal value close to the nominal.

3.6 Whole-Body Controller

An overview of the WBC is provided in this section. The goal of this module is to track the planned trajectories $\mathbf{x}^{\mathrm{p}}$ and $\mathbf{u}^{\mathrm{p}}$ provided by the NMPC. The WBC first computes feedforward $\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}}$ and feedback $\mathbf{W}_{\mathrm{fb}}^{\mathrm{d}}$ wrenches from the planned trajectories and then the sum of these wrenches are mapped into GRFs through the QP optimization of Eq. (3.17). The WBC also maps the GRFs into the joint torques $\boldsymbol{\tau}^{*}$. This joint torque along with low-impedance feedback torque $\boldsymbol{\tau}_{\mathrm{fb}}$ results in the total torque $\boldsymbol{\tau}_{\mathrm{d}}$ required by the low-level joint torque control block. Refer to Figure 3.1 for the block representation of WBC inside our locomotion framework.

3.6.1 WBC interface

The NMPC runs at replanning frequency of 25 Hz whereas the WBC requires state and control inputs at 250 Hz (the *WBC frequency*). Hence, the WBC interface block re-samples the state and the control inputs at the WBC frequency. In particular, in order to obtain the desired

$\mathbf{u}^{\mathrm{d}}$ I use a zero-order hold filter of $\mathbf{u}^{\mathrm{p}}$. The planned states $\mathbf{x}^{\mathrm{p}}$ from the NMPC, instead, are re-sampled with a linear interpolation to obtain $\mathbf{x}^{\mathrm{d}}$.

The rigorous approach is to use the model of Eq. (3.3) to predict the evolution of the system in the T_{s} time interval, considering the $\mathbf{u}^{\mathrm{p}}$ coming from the NMPC, but for the motions considered in this work the result is very similar, so a linear interpolation is a fair approximation.

Finally, a feedforward wrench $\mathbf{W}_{\mathrm{ff}} \in \mathbb{R}^6$ is computed from the desired GRFs $\mathbf{u}^{\mathrm{d}}$ as:

$$\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}} = \begin{bmatrix} \sum_{i=1}^{4} \mathbf{u}_i^{\mathrm{d}} \\ \sum_{i=1}^{4} \mathbf{p}_{\mathrm{cf},i}^{\mathrm{d}} \times \mathbf{u}_i^{\mathrm{d}} \end{bmatrix} \tag{3.14}$$

3.6.2 Feedback wrench

[Focchi et al., 2016] presented the implementation of a Cartesian impedance which is reported here for the sake of completeness:

$$\mathbf{W}_{\mathrm{fb}}^{\mathrm{d}} = \mathbf{K} \begin{bmatrix} \mathbf{p}_{\mathrm{c}}^{\mathrm{d}} - \mathbf{p}_{\mathrm{c}} \\ \mathbf{e}({}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}^{\top}\, {}_{\mathrm{w}}\mathbf{R}_{\mathrm{d}}) \end{bmatrix} + \mathbf{D} \begin{bmatrix} \mathbf{v}_{\mathrm{c}}^{\mathrm{d}} - \mathbf{v}_{\mathrm{c}} \\ \boldsymbol{\omega}_{\mathrm{b}}^{\mathrm{d}} - \boldsymbol{\omega}_{\mathrm{b}} \end{bmatrix} \tag{3.15}$$

where ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}$ and ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{d}} \in \mathbb{R}^{3\times3}$ are the rotation matrices representing the actual and desired orientation of the base with respect to the inertial frame, respectively, $\mathbf{e}(\cdot) : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is a mapping from a rotation matrix to the associated rotation vector. Matrices $\mathbf{K}$ and $\mathbf{D}$ are diagonal matrices containing the proportional and derivative gains and they can be interpreted as impedances.

Remark: At each replanning instance of NMPC, the state reference is computed from the current state of the robot $\hat{\mathbf{x}}_{\mathbf{0}}$. Thus, at each replanning instance, the feedback term is nullified.

3.6.3 Projection of the GRFs

While the feedforward wrenches $\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}}$ provided by MPC satisfy the friction cone and unilateral constraints by construction, this guarantee is lost with the addition of the feedback term $\mathbf{W}_{\mathrm{fb}}^{\mathrm{d}}$ to the wrenches. Therefore, one needs to project the total wrenches $\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}} + \mathbf{W}_{\mathrm{fb}}^{\mathrm{d}}$ onto the set of wrenches that satisfy the constraints.

Let us define with c the number of stance feet, and let us write the matrix representation

$$\underbrace{\begin{bmatrix} \mathbf{I} & \dots & \mathbf{I} \\ [\mathbf{p}_{\mathrm{cf},1}\times] & \dots & [\mathbf{p}_{\mathrm{cf},c}\times] \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_c \end{bmatrix}}_{\mathbf{f}} = \underbrace{\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}} + \mathbf{W}_{\mathrm{fb}}^{\mathrm{d}}}_{\mathbf{b}} \tag{3.16}$$

Equation (3.16) is derived from a simplified SRBD model and allows us to map the desired wrenches into GRFs. The system (3.16) has 6 equations and $3c$ unknowns, so for a crawl motion, it has infinite solutions. Thanks to this redundancy, it is possible to solve the following QP to compute the desired GRFs $\mathbf{f}^{\mathrm{d}}$:

$$\mathbf{f}^{\mathrm{d}} = \underset{\mathbf{f}}{\operatorname{argmin}} \ \| \mathbf{A}\mathbf{f} - \mathbf{b} \|^2_{\mathbf{S}} + \| \mathbf{f} - \mathbf{u}^{\mathrm{d}} \|^2_{\mathbf{T}} \tag{3.17a}$$

$$\text{s.t.} \quad \underline{\mathbf{d}} \leq \mathbf{C}\mathbf{f} \leq \overline{\mathbf{d}} \tag{3.17b}$$

The term $\| \mathbf{f} - \mathbf{u}^{\mathrm{d}} \|^2_{\mathbf{T}}$ Eq. (3.17) allows the tracking of the desired forces $\mathbf{u}^{\mathrm{d}}$ received from the NMPC. Matrices $\mathbf{S} \in \mathbb{S}^6_+$ and $\mathbf{T} \in \mathbb{S}^{12}_+$ are positive-definite weight matrices. Inequality (3.17b) encodes the friction cone and unilateral constraints similar to Eq. (3.7). Using the approximation with square pyramids, Eq. (3.17b) is equivalent to:

$$\mathbf{C} \triangleq \begin{bmatrix} \mathbf{C}_0 \dots 0 \\ \vdots \ddots \vdots \\ 0 \dots \mathbf{C}_{\mathrm{c}} \end{bmatrix} \qquad \underline{\mathbf{d}} \triangleq \begin{bmatrix} \underline{\mathbf{d}}_0 \\ \vdots \\ \underline{\mathbf{d}}_{\mathrm{c}} \end{bmatrix} \qquad \overline{\mathbf{d}} \triangleq \begin{bmatrix} \overline{\mathbf{d}}_0 \\ \vdots \\ \overline{\mathbf{d}}_{\mathrm{c}} \end{bmatrix}$$

with:

$$\mathbf{C}_{\mathrm{i}} = \begin{bmatrix} (-\mu_{\mathrm{i}}\mathbf{n}_{\mathrm{i}} + \mathbf{t}_1)^\top \\ (-\mu_{\mathrm{i}}\mathbf{n}_{\mathrm{i}} + \mathbf{t}_2)^\top \\ (\mu_{\mathrm{i}}\mathbf{n}_{\mathrm{i}} + \mathbf{t}_1)^\top \\ (\mu_{\mathrm{i}}\mathbf{n}_{\mathrm{i}} + \mathbf{t}_1)^\top \\ \mathbf{n}_{\mathrm{i}}^\top \end{bmatrix} \qquad \underline{\mathbf{d}}_{\mathrm{i}} = \begin{bmatrix} -\infty \\ -\infty \\ 0 \\ 0 \\ \mathbf{f}_{\min} \end{bmatrix} \qquad \overline{\mathbf{d}} = \begin{bmatrix} 0 \\ 0 \\ \infty \\ \infty \\ \mathbf{f}_{\max} \end{bmatrix}$$

where $\mathbf{n}_{\mathrm{i}} \in \mathbb{R}^3$ is the direction normal to the terrain, $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3$ are the tangential components and $\mathbf{f}_{\min}, \mathbf{f}_{\max} \in \mathbb{R}$ are the bounds for the normal force.

It is important to note that gravity compensation is already incorporated in the NMPC formulation through the SRBD model and thus considered in $\mathbf{W}^{\mathrm{d}}_{\mathrm{ff}}$.

3.6.4 Mapping GRFs to Joint Torques

The GRFs $\mathbf{f}^{\mathrm{d}}$ must be mapped into joint torques $\boldsymbol{\tau}^*$. by exploiting the joint dynamics, see Eq. (2.11):

$$\boldsymbol{\tau}^* = -\mathbf{J}(\mathbf{q})^\top \mathbf{f}^{\mathrm{d}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.20}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{n_u \times n}$ is the contact Jacobian and $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ the vector of gravity/Coriolis terms in the leg joint dynamics. The number of joints is denoted by n.
In this case, the joint acceleration contribution $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$, is neglected because it is minimal with respect to the other terms.

3.6.5 Joint-Space PD

A 1 kHz Joint-Space Proportional Derivative (PD) controller is put in cascade with the WBC before sending torques to the low-level controller. It is fundamental to track the desired trajectories of the swinging legs and to increase the robustness in case a foot loses contact with the ground.

The WBC interface provides the joint trajectories $\mathbf{q}_\mathrm{d}$ and $\dot{\mathbf{q}}_\mathrm{d}$ required by the Joint-Space PD. To compute them, inverse kinematics is evaluated which in turn needs the swing trajectory $\mathbf{p}_\mathrm{f}^\mathrm{sw}$.

The swing trajectory generation takes advantage of the swing frame $\mathcal{S}$ [Raiola et al., 2020]. A drawing of this reference frame is provided in Figure 3.7. The X-axis is aligned with the vector that links lift-off and touchdown point ($\mathbf{L}_\mathrm{sw}$), Y-axis is perpendicular to the X-axis of the swing frame and to the Z-axis of the world frame. Finally the Z-axis is such that $\mathcal{S}$ is a counter-clockwise coordinate system. The origin of the swing frame $\mathcal{S}$ coincides with the lift-off point. In this way the swing trajectory lies on the X-Z plane and it is shaped as a semi-ellipse with $\mathbf{L}_\mathrm{sw}$ and H_sw as lengths of the axes:

$$ {}_{\mathcal{S}}\mathbf{p}_f^\mathrm{sw} = \begin{bmatrix} \frac{\mathbf{L}_\mathrm{sw}}{2}(1-\cos(\pi f_\mathrm{sw} t_\mathrm{sw})) \\ 0 \\ H_\mathrm{sw}\sin(\pi f_\mathrm{sw} t_\mathrm{sw}) \end{bmatrix} \tag{3.21} $$

where t_sw is the time elapsed from the beginning of a swing and $f_\mathrm{s} = 1/T_\mathrm{sw}^\mathrm{d}$ is the swing frequency. The mapping of ${}_{\mathcal{S}}\mathbf{p}_\mathrm{f}^\mathrm{sw}$ and its derivative in the inertial frame $\mathcal{W}$ allows one to obtain $\mathbf{p}_\mathrm{f}^\mathrm{sw}$ and $\dot{\mathbf{p}}_\mathrm{f}^\mathrm{sw}$, respectively. Finally, after evaluating the relative foot position with respect to the center of the trunk of the robot and similarly the relative velocity, the inverse kinematics can obtain $\mathbf{q}_\mathrm{d}$ and $\dot{\mathbf{q}}_\mathrm{d}$

Remark: The variable $\mathbf{p}_\mathrm{c}$ is the CoM of the robot. While in the X-Y directions the robot is symmetric and so $\mathbf{p}_{\mathrm{c},(\mathrm{x},\mathrm{y})}$ is very close to the center of the trunk, the Z coordinate is shifted downwards due to the effect of the legs. More in general, the mapping of CoM quantities into base ones, *i.e.,* related to the center of the trunk, depends on the values of $\mathbf{q}, \dot{\mathbf{q}}$.

3.7 Real-time iteration for NMPC

One of the main drawbacks of NMPC is its computational burden, thus efficient tailored algorithms are necessary in order to achieve fast sampling rates for complex systems with fast dynamics. In this work, I focus on direct multiple shooting methods derived from SQP that have been specifically developed for real-time NMPC [Diehl et al., 2005].

In multiple shooting methods both state $\mathbf{x}$ and control input $\mathbf{u}$ are decision variables unlike in single shooting where the decision vector only includes the control input. This does not

increase the computational complexity with respect to single shooting (where computations are moved from linear algebra to the evaluation of derivatives). Furthermore, multiple-shooting allows one to provide an initial guess also for the state trajectory, which is typically beneficial for unstable systems in an NMPC context [Diehl et al., 2005].

SQP is a popular algorithm which solves an NLP by iteratively solving local quadratic approximations (QPs) of the problem [Gros et al., 2020]. At each SQP iteration, the solution from the previous step is recycled to define an initial guess $(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}})$, which is then used to construct a QP approximation of the NLP (3.1), given by

$$\min_{\Delta\mathbf{x},\Delta\mathbf{u}} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix}^\top \mathbf{H}_k \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} + \mathbf{J}_k^\top \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} \tag{3.22a}$$

$$\text{s.t.} \quad \Delta\mathbf{x}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0^{\mathrm{L}}, \tag{3.22b}$$

$$\Delta\mathbf{x}_{k+1} = \mathbf{A}_k\Delta\mathbf{x}_k + \mathbf{B}_k\Delta\mathbf{u}_k + \mathbf{r}_k, \tag{3.22c}$$

$$\mathbf{C}_k\Delta\mathbf{x}_k + \mathbf{D}_k\Delta\mathbf{u}_k + \mathbf{h}_k \geq 0, \tag{3.22d}$$

where, $\Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^{\mathrm{L}}, \Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^{\mathrm{L}}$, $\hat{\mathbf{x}}_0$ is the current system state, and

$$\mathbf{A}_k = \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, \qquad \mathbf{B}_k = \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}},$$

$$\mathbf{C}_k = \left.\frac{\partial h(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, \qquad \mathbf{D}_k = \left.\frac{\partial h(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}},$$

$$\mathbf{r}_k = g\left(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}}, \mathbf{a}_k\right) - \mathbf{x}_{k+1}^{\mathrm{L}}, \qquad \mathbf{h}_k = h\left(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}}, \mathbf{a}_k\right)$$

$$\mathbf{J}_k = \mathbf{W}_k \begin{bmatrix} \mathbf{x}_k^{\mathrm{L}} - \mathbf{x}_k^{\mathrm{ref}} \\ \mathbf{u}_k^{\mathrm{L}} - \mathbf{u}_k^{\mathrm{ref}} \end{bmatrix} \tag{3.23}$$

Matrix $\mathbf{H}_k$ is the diagonal blocks of a suitable approximation of the Lagrangian Hessian. Since our problem relies on a least-squares cost, we adopt the popular Gauss-Newton Hessian approximation [Gros et al., 2020] that gives $\mathbf{H}_k = \mathbf{W}_k$.

While in SQP one solves several QPs until convergence is reached, the RTI scheme consists in solving a single QP per sampling time. This is motivated by the observation that in NMPC two subsequent problems have very similar solutions. Therefore, by reusing the solution of the previous NMPC problem, one obtains a very good initial guess for the next problem, which essentially only needs to correct for external perturbations and model mismatch. For all details on the RTI scheme, I refer to [Diehl et al., 2005; Gros et al., 2020] and references therein.

An interesting discussion arises from that the guess $(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}})$ is known *before* the next state measurement is available; this means that one can already evaluate the functions and their derivatives (3.23) before the initial state $\hat{\mathbf{x}}_0$ is available. Consequently, the QP can be constructed and prepared beforehand; note that this also includes the first factorization of the QP Hessian. Once $\hat{\mathbf{x}}_0$ is available, one only has to finish solving the QP. Therefore, while the overall sampling time must still be long enough to prepare the next QP, the latency between the time at which $\hat{\mathbf{x}}_0$ is available and the time at which the control input can be applied to the system is very small.

In the RTI scheme proposed above, the functions and their derivatives (3.23) are evaluated along a guess obtained from the previous solution, rather than along the reference trajectory. In this way a better approximation of the original problem is obtained.

Another important aspect to highlight is that there exist several approaches to compute Eq. (3.23). One choice consists of first linearizing the continuous-time system dynamics and then using the matrix exponential to obtain a discrete-time linear system. This approach presents some advantages, but can be computationally demanding. For the time-varying and infeasible references, however, it is preferred to *first discretize and then linearize* [Gros et al., 2020]. In this work, due to the presence of time-varying and infeasible references, the usage first discretize and then linearize approach is preferable. An advantage of this approach is that after numerically approximating the discrete-time dynamics, the linearization can be obtained at a desired accuracy.

A very popular way to obtain discrete-time dynamics is with the explicit Euler integrator, which is computationally inexpensive, but can be inaccurate and unstable. Therefore, it is usually more efficient to resort to higher-order integration schemes, such as,*e.g.,* the popular Runge-Kutta methods. Finally, there also exist implicit integration schemes, which require more computations per step, but they are typically much more stable and accurate than explicit schemes for some classes of systems. Unfortunately, the selection of the least computationally demanding integrator which delivers sufficient accuracy depends on the problem setting and cannot be known a priori. Typically it requires some trial-and error approach, which can be educated using some guidelines based on the theoretical properties of each integrator [Quirynen, 2017; Butcher, 2003; Hairer et al., 1993, 1996; Quirynen et al., 2014].

This work relies on the RTI implementation provided by `acados` [Verschueren et al., 2019], which consists of tailored efficient implementations of QP solvers, numerical integration schemes, and all other components of the RTI scheme.

3.8 Results

This section discusses the implementation details and results obtained from the simulations and experiments with the NMPC scheme proposed in Section 3.3.

The videos of both simulation and experiments can be found at the following link: https://youtu.be/r0-KIiw0eWM.

3.8.1 HyQ

Presented in 2011, HyQ is the first quadruped produced by IIT's DLS[3] during its twelve-years history. The robot weighs 87 kg and operates connected to a hydraulic pump and an external power supply. An IMU-based State Estimator [Nobili et al., 2017] is used instead of a motion capture system allowing one to perform experiments in outdoor spaces.

Each leg has three DoFs: Hip Adduction-Abduction (HAA), Hip Flexion-Extension (HFE), Knee Flexion-Extension (KFE). The HAAs are composed of hydraulic rotary motors equipped with torque sensors. The HFEs and the KFEs, instead, use hydraulic linear cylinders; the torques are estimated by measuring the linear forces at the cylinder. Joint positions are measured by the encoders, while joint velocities are obtained through differentiation of the position signals; the GRFs are estimated from the joint quantities. Please refer to [Semini, 2010] for additional details on the hardware design of HyQ.

3.8.2 Implementation details

To check the efficiency of the RTI-based NMPC algorithm with the proposed features mentioned in Section 3.4, several simulations and experiments are proposed in challenging scenarios. The simulation and experiments were performed on the HyQ robot of mass $m = 87\,\text{kg}$. The CoM is computed considering the mass of the individual link of the robot and the actual position of the links' CoM. The position of the links' CoM in their local frame is obtained from their CAD models. That information is also used to find the relationship between CoM and the center of the trunk for the swing generation (Section 3.6.5).

The feedback gains used in the WBC are $\mathbf{K} = \text{diag}(1500, 1500, 1500, 100, 100, 100)$ and $\mathbf{D} = \text{diag}(1000, 1000, 1000, 50, 50, 50)$. The weights for the QP (Eq. (3.17)) $\mathbf{S} = \text{diag}(5, 5, 10, 10, 10, 10)$ and $\mathbf{T} = \text{diag}(1000, \cdots, 1000)$.

The parameters and weights used by the NMPC are reported in Table 3.1 and 3.2, respectively. In all of the simulations and experiments, there are no any weights on the CoM position ($\mathbf{p}_c$), roll (ϕ) and pitch (θ) tasks because the NMPC is able to sort out these CoM quantities autonomously, *e.g.,* with the mobility term.

[3]Website: https://dls.iit.it/

Table 3.1: NMPC parameters

Parameter	Symbol	Value	Unit
Number of state	n_x	12	-
Number of control inputs	n_u	12	-
Number of model parameters	n_p	16	-
Prediction horizon	T	2	s
Sampling time	T_s	0.04	s
Number of control intervals	N	50	-
Friction coefficient	μ	0.7	-
GRFs lower bound	$\underline{\mathbf{f}}_\mathrm{z}$	0	N
GRFs upper bound	$\overline{\mathbf{f}}_\mathrm{z}$	500	N
Hip-to-foot distance reference	${}_\mathcal{C}\mathbf{p}^{\mathrm{ref}}_{\mathrm{hf},i}$	$(0.0, 0.0, -0.55)$	m

Table 3.2: Weights used in the NMPC

Cost	Weight	Value
State	$\mathbf{Q}_{\mathbf{p}_\mathrm{c}}$	diag(0, 0, 0)
	$\mathbf{Q}_{\mathbf{v}}$	diag(100, 100, 100)
	$\mathbf{Q}_{\mathbf{\Phi}}$	diag(0, 0, 100)
	$\mathbf{Q}_{\boldsymbol{\omega}}$	diag(100, 100, 1000)
Force	$\mathbf{R}_\mathrm{x}$	1×10^{-3}
	$\mathbf{R}_\mathrm{y}$	1×10^{-3}
	$\mathbf{R}_\mathrm{z}$	8×10^{-4}
Mobility	$\mathbf{M}_\mathrm{x}$	1×10^{-4}
	$\mathbf{M}_\mathrm{y}$	2×10^{-3}
	$\mathbf{M}_\mathrm{z}$	1000
Force robustness	$\mathbf{P}_\mathrm{x}$	100
	$\mathbf{P}_\mathrm{y}$	100
	$\mathbf{P}_\mathrm{z}$	1

Discretization

For the discretization of the dynamic constraints of Eq. (3.5), I mainly investigated two integration schemes, *i.e.,* a single step of the explicit Euler of order 1 and implicit midpoint

method due to their low computational complexity that favors the real-time implementation needs.

The explicit Euler of order 1 is the most basic explicit methods and it is based on the truncated Taylor expansion:

$$x_{n+1} = x_n + (t_{n+1} - t_n) f(x_n, t_n)$$

The implicit midpoint method is a second order Range-Kutta discretization method and its formulation is:

$$x_{n+1} = x_n + f\Big(\frac{x_n + x_{n+1}}{2}, t_n + \frac{t_{n+1} - t_n}{2}\Big)$$

Among the two, the midpoint demonstrated better performance because of its stability and accuracy properties. The achieved accuracy did not require more complex schemes, *e.g.,* Range-Kutta of order 4.

The sampling time $T_s = 40\,\text{ms}$ was chosen and it was sufficient to conduct NMPC computation *online* along with the other necessary computations for the replanning.

NMPC software

The RTI scheme described in Section 3.7 is implemented in the `acados` software package. Since `acados` comes with a Python interface allowing rapid prototyping, we first tuned the algorithm in simulation and then used the generated `C`-code to perform real experiments. The problem is solved with the QP solver High-Performance Interior Point Method (HPIPM) [Frison and Diehl, 2020], which exploits the sparsity structure of the MPC QP sub-problem (3.22), and supports inequality constraints. Details on Interior Point have been described in Section 2.2.3.

The computation time required by the NMPC was in the range of 5-7 ms with the prediction horizon of 2 s and control intervals N equal to 50 on the on-board computer (a Quad Core Intel Pentium PC104 @ 1GHz) of HyQ for all the experiments. This computation time corresponds to the feedback phase of the RTI scheme where the QP of Eq. (3.22) is solved after receiving the current state of the robot. The preparation phase of the RTI takes about 2-3 ms which is a fraction of the sampling time we chose. Refer to Section 3.7 for more details on these phases of the RTI scheme. Even though the computation time of NMPC is mostly consistent, some outliers happened. Hence a conservative approach to run the NMPC at 25 Hz guarantees that the computation time stays always less than 40 ms. Besides the computation time of the NMPC, the replanning frequency of 25 Hz also accounts for the time required by other blocks such as reference generator so that the total computation time does not exceed 40 ms.

Integration with the locomotion framework

The NMPC is integrated in a ROS node that publishes $\mathbf{x}^{\mathrm{p}}$ and $\mathbf{u}^{\mathrm{p}}$ at a frequency of 25 Hz. The on-board computer along with our locomotion framework (WBC Interface, WBC, etc., illustrated in Fig. 3.1) runs a real-time node that subscribes to the topic of the NMPC ROS node. ROS is not a real-time operating system, so it can introduce quite a significant and unpredictable communication delay if the NMPC is run on an external (*e.g.,* more powerful) computer. These delays are difficult to compensate for and they can cause a loss of synchronization between the NMPC ROS node and the WBC interface. Therefore, the NMPC node is launched natively on the on-board computer to avoid communication delays between two different computers.

Even though a more powerful dedicated off-board computer for the NMPC ROS node could be exploited, better performance in the overall implementation is obtained by avoiding the communication delays of ROS.

3.8.3 Simulations

Simulations in Gazebo [Koenig and Howard, 2004] shows the NMPC planner in action on challenging terrain.

Gazebo is a 3D simulator designed to reproduce the dynamic environments a robot may encounter. The simulated environment is populated by *model* with mass, velocity, friction, and numerous other attributes that allow them to behave realistically when pushed, pulled, knocked over, or carried. In particular, robots are dynamic structures composed of rigid bodies connected via joints. Forces, both angular and linear, can be applied to surfaces and joints to generate locomotion and interaction with an environment.

The main simulations are pallet crossing, walking over unstructured rough terrain and walking into a V-shaped Chimney.

Pallet crossing

In this simulation, HyQ traverses pallets of different heights, placed at varying distances from each other.

This simulation highlights the importance of including mobility in the NMPC formulation (3.2c). In particular, the simulation scenario includes a set of pallets, each one of 1 m length, with variable heights between 0.13 and 0.17 m and placed at unequal gap lengths ranging from 0.2 and 0.7 m. Multiple trials are performed commanding the robot to move forward at different velocities *i.e.,* 0.05 m/s and 0.1 m/s to show the repeatability of our approach.

To avoid stepping on undesired locations such as pallet edges and to prevent foot or shin collisions, the nominal footholds are adjusted by using the VFA (refer to Section 3.5.2).

Figure 3.8 shows the results of five different trials for each of the commanded velocities. The top plot shows the pitch angle θ of the robot as it traverses the scenario. Since the robot is only commanded to move along its X direction with a constant forward velocity and the foot locations are provided as known quantities to the NMPC, the adjustment in pitch is the result of minimizing the deviation from the hip-to-foot distance configuration corresponding to high mobility for all four legs. Without this feature, the robot would maintain a constant horizontal orientation (see Fig. 3.9) eventually reaching low mobility in some legs.

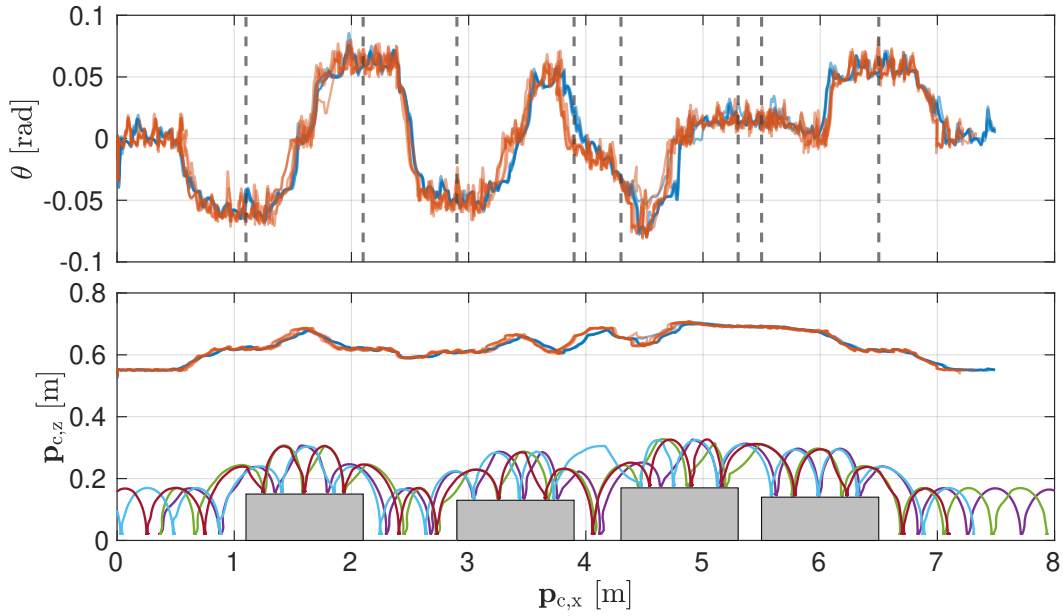

Figure 3.8: Simulation of pallet crossing scenario for five different trials commanding the robot to cross at 0.1 $\mathrm{m/s}$ (blue) and 0.05 $\mathrm{m/s}$ (red). The top graph shows the pitch angle and the dashed vertical lines indicate the edges of the pallets for that specific location in the plot. The bottom graph shows the CoM Z position for all the trials and the feet trajectories for one of the trials performed at 0.1 $\mathrm{m/s}$. The color of the swings are related to the different legs.

In addition, the robot is able to walk on randomly generated rough terrain (using terrain generation tool by [Abbyasov et al., 2020], see Figure 3.10) with the forward velocity of 0.3 m/s further stressing the advantages of mobility cost mentioned earlier. Figure 3.10 (bottom) shows the heightmap built by the robot during the motion. Different colors (*e.g.*, light blue and pink) represent points with different heights. The red disks indicate the nominal footholds in the prediction horizon computed by Eq. (3.13). The green disks refer to the foot location corrected by the VFA to avoid holes and be coherent with the terrain.

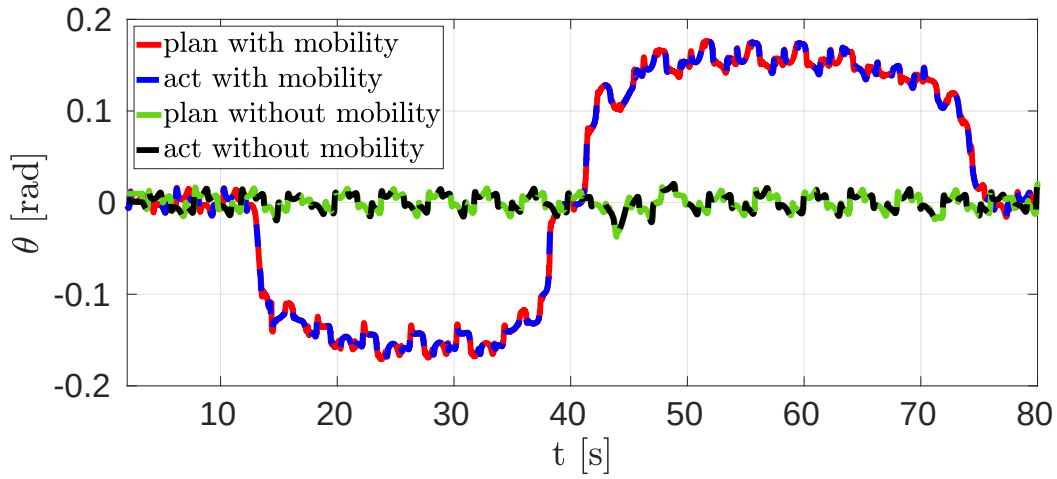

Figure 3.9: Comparison of the robot pitch θ in simulation with and without mobility cost in the NMPC. The red and green lines represent the planned pitch values delivered by the NMPC. The blue and black dashed lines are the actual pitch of the robot. Without mobility cost, the robot maintains a horizontal orientation, whereas the robot pitches to improve leg mobility when it is included.

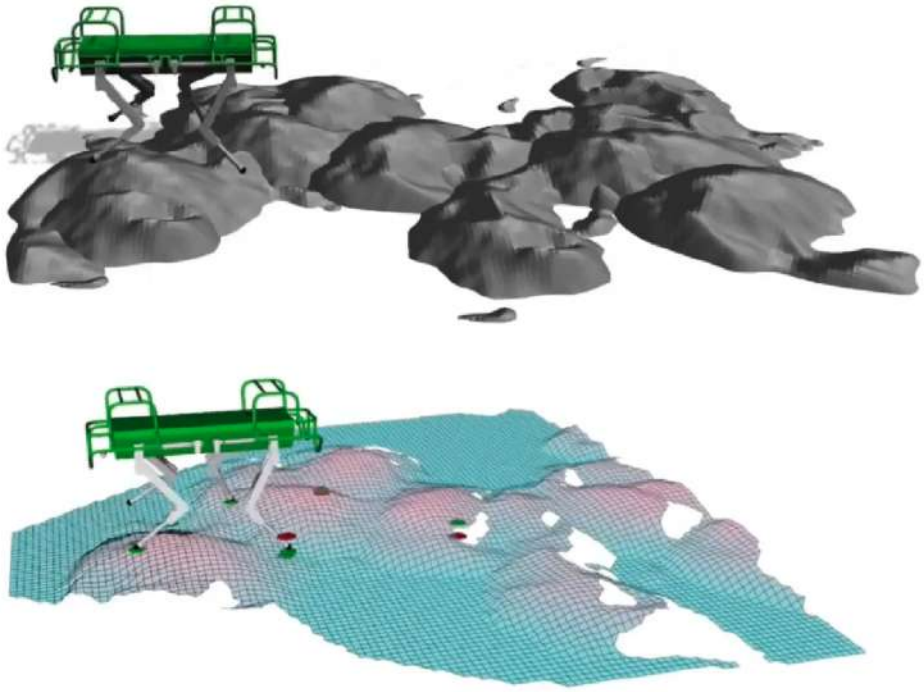

Figure 3.10: Simulation: HyQ traverses a randomly generated rough terrain thanks to the mobility feature. Red disks show nominal footholds, which are then corrected by the VFA (green disks).

Walk into a V-shaped chimney

In this simulation, HyQ walks at $0.03\,\mathrm{m/s}$ commanded velocity in the X direction into a V-shaped chimney with friction coefficient $\mu = 0.7$ and walls inclined at $35°$ to the ground.

This simulation exploits the cone constraints and force robustness cost defined inside the NMPC formulation that is vital for the success of this task. The robot receives an online update of the map of the environment through an on-board camera to get information about the normals at the location of the contact. These normals are used to formulate the force robustness cost of Eq. (3.2d) in the contact frame $\mathcal{K}$. With this cost, the NMPC provides optimal GRFs to stay close to the normals of the friction cones at the contacts. An image of this scenario is reported in Figure 3.11.

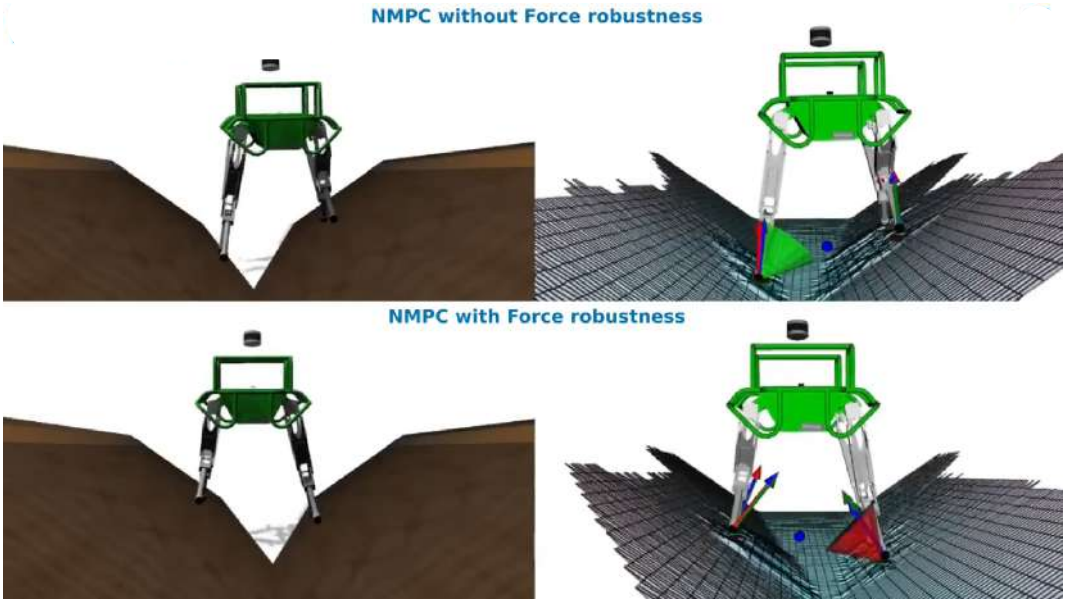

Figure 3.11: Comparison of the robot walking into a V-shaped chimney with and without force robustness (Eq. (3.2d)) cost in the NMPC. The red, green and blue lines arrows represent planned $\mathbf{f}^p$, the WBC $\mathbf{f}^d$ and the measured forces respectively. The image on the top refers to the case without the force robustness. The RF GRFs are aligned with the border of the friction, providing no robustness. The force robustness instead (bottom) drives the solution towards the center of the cone

Without the cone constraints, the robot slips while climbing the chimney and ultimately falls. When the force robustness cost is enabled, the forces are regularized to stay in the middle of the cones, thanks to the robustness feature described in Section 3.4.3. In this case, the robot walks successfully into the chimney.

In Figure 3.12, it can be seen that the longitudinal and lateral components of the GRFs at the LF foot stay within the bound μf_z (in red) imposed by the cone constraints. Moreover, Figure 3.13 plots the normal versus the tangential force of the GRF together with the cone bound μf_z (red line). The picture shows that the GRFs stays well within the bound without any violation. Therefore, including the force regularization term enables the NMPC to account for the estimation error in the orientation of contact normals and increase robustness to the

external disturbances.

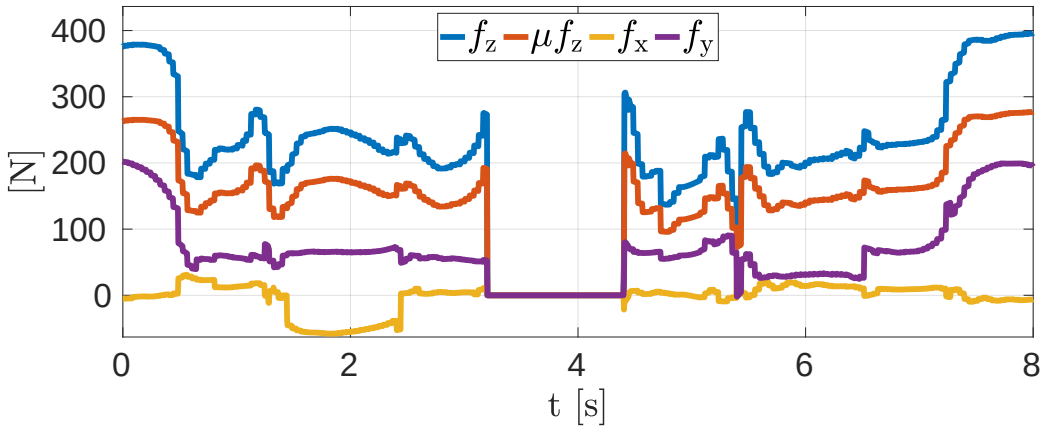

Figure 3.12: Walk into a V-shaped chimney simulation: GRFs of LF leg for a single gait cycle with cone constraints and regularization cost. Both the longitudinal f_x and lateral f_y lie conservatively within the bound μf_z imposed by cone constraints.

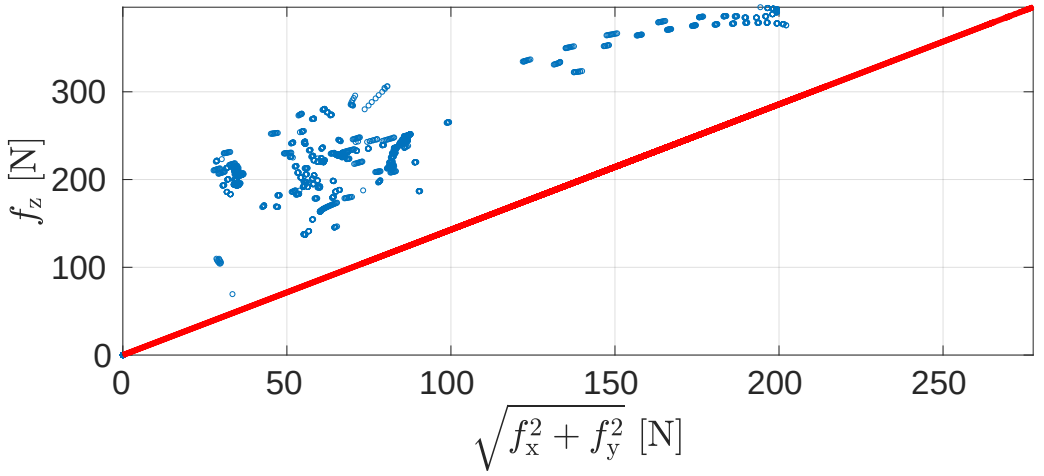

Figure 3.13: Walk into a V-shaped chimney simulation: Normal force f_z versus tangential force $\sqrt{f_x^2 + f_y^2}$ of the LF leg for a single gait cycle expressed in the contact frame. The red line is the cone bound μf_z.

Apart from the simulation mentioned above, the video[4] shows two additional simulations: a first regarding the ZMP margin (refer to Section 3.4.2) and a second that demonstrates the importance of a replanning at a higher rate.

For the ZMP margin simulation, the robot is pushed with 200 N of lateral force for 1 s both in case of sufficient (higher weight on GRFs Z) and no (lower weight on GRFs Z) ZMP margin. The ZMP margin plots for this simulation can be seen in Figure 3.14 where the ZMP margin

[4] https://youtu.be/r0-KIiw0eWM

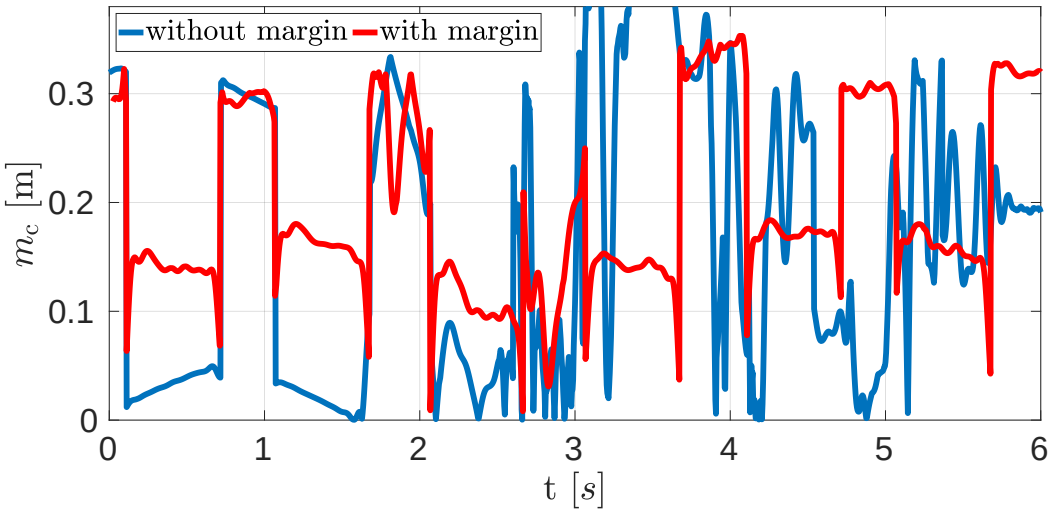

Figure 3.14: Plot of the ZMP margin m_c used to measure locomotion stability. The robot is pushed immediately after $2\,\mathrm{s}$ with lateral force of $200\,\mathrm{N}$ while walking on a flat terrain at $0.1\,\mathrm{m/s}$ CoM X velocity. The discontinuities are due to the switching between 3/4 stance legs in a crawl gait.

is improved in case of the red line compared to the blue one because the GRFs Z components are penalized relatively more (100 times) for the red line. Because of the improved margin the robot walks stably, whereas it falls while walking when there is no margin.

In the second simulation, the robot is commanded with a constant CoM X velocity and heading velocity simultaneously. In the case of replanning at a lower rate of $0.8\,\mathrm{Hz}$, the robot becomes unstable and falls due to the increasing in model uncertainties and tracking errors. When the replanning is done at a lower frequency than $25\,\mathrm{Hz}$, the robot is in open-loop for the time interval between two consecutive replanning instances, hence, it is no more NMPC but an online open-loop trajectory optimization. On the other hand, at a higher replanning frequency of $25\,\mathrm{Hz}$ the robot walks successfully because the NMPC compensates for the model uncertainties and tracking errors.

3.8.4 Experiments

This section reported three different experiments which have been performed to demonstrate the real-time implementation of the NMPC running on the on-board computer of the robot. The video can be seen at the same link: https://youtu.be/r0-KIiw0eWM

Omni-directional walk

In this experiment, HyQ executes an omni-directional walk performed with the NMPC on a flat terrain. It validates that the NMPC computes feasible trajectories after receiving different

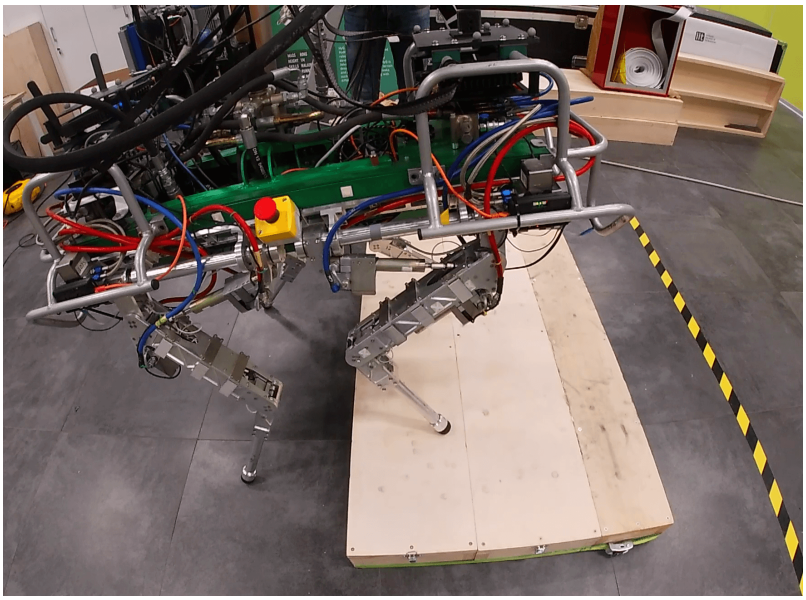

Figure 3.15: DLS' quadruped robot HyQ traversing a pallet with the mobility enhanced real-time NMPC.

velocity commands from the user while walking. Indeed, the robot is commanded with a longitudinal velocity ${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c,x}}^{\mathrm{usr}}$ by the user to walk forward/backward and then a lateral velocity ${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c,y}}^{\mathrm{usr}}$. Finally, a heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}}$ is commanded to turn in the left/right direction.

Figure 3.16 shows the CoM X-Y position and yaw angle of the robot base and it can be noticed that the actual values track very closely the planned trajectories provided by the NMPC.

Figure 3.17 depicts the deviation of the actual velocities from the reference values while following the planned trajectories form NMPC. It can be seen in Fig. 3.18 that the GRFs generated by the WBC are compliant with the planned values $\mathbf{u}^{\mathrm{p}}$ and again the actual values of GRFs track closely the planned values.

From these plots, it can be observed that the continuous replanning with NMPC plays an important role to achieve good tracking of the planned trajectory.

Traversing a static pallet

The purpose of this experiment is to demonstrate that the mobility cost (3.2c) incorporated in the NMPC formulation provides the necessary body pitch for the robot to traverse over a static pallet while maintaining good leg mobility. The pallet used in this experiment is 0.13 m in height and 0.8 m in length.

Figure 3.19 shows that the robot pitches up while climbing up the pallet and pitches down consequently while climbing down from the pallet. As shown in the previously mentioned video,

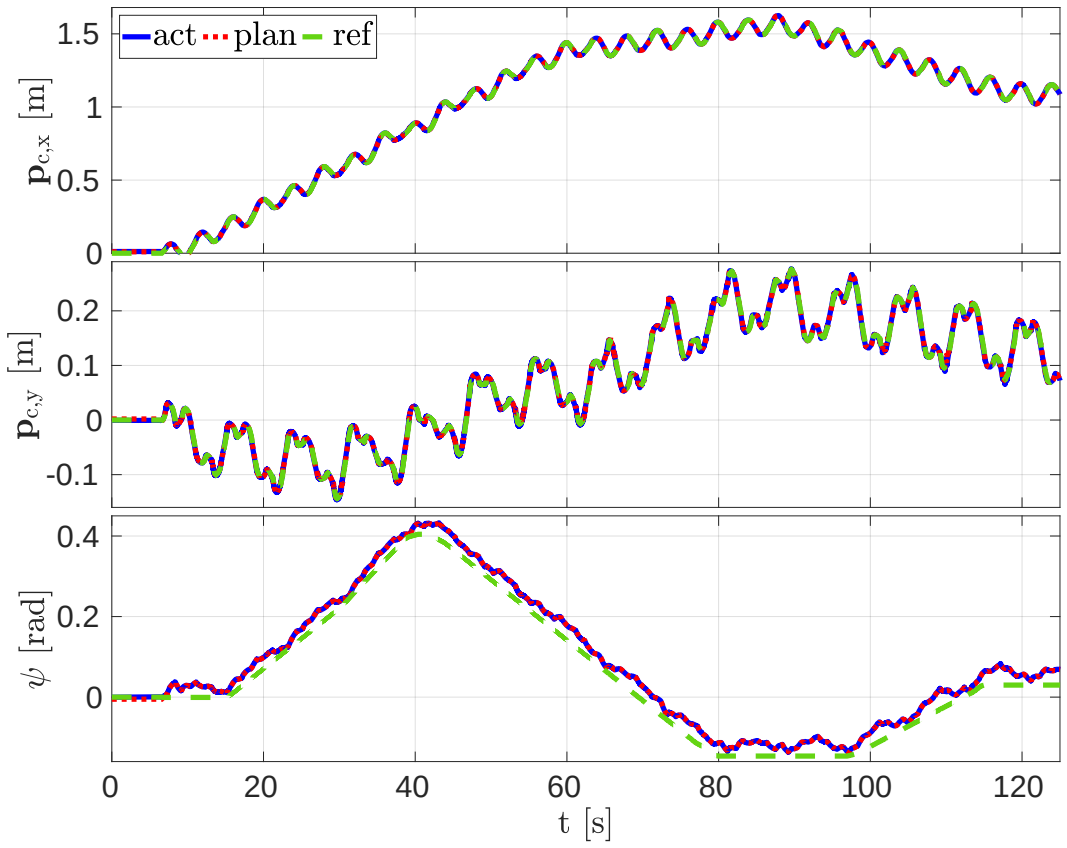

Figure 3.16: CoM X-Y position and yaw ψ in omni-directional walk experiment. The blue, dotted red and dashed green line represent the actual, planned and reference values, respectively.

in the simulation, the NMPC maintains the horizontal base orientation when the mobility cost is deactivated. This causes a reduced hip-to-foot distance while stepping up/down on the pallet ultimately resulting in low leg mobility. When mobility cost is activated, it directs the NMPC solution to achieve the necessary pitch; in this way, the hip-to-foot distance is maintained close to the reference value, and hence the leg mobility is improved. Moreover, the VFA provides the corrected foot position (*i.e.,* to avoid shin or feet collisions with the edges of the pallet) to the NMPC and this further enhances the overall locomotion.

Traversing a repositioned pallet

In this experiment, the NMPC is tested to plan the robot motion in real-time by adapting to the changes in the environment.

As it can be seen from the uploaded video, when the pallet (0.13 m in height and 0.8 m in length) is pushed in front of HyQ while walking, the heightmap detects the pallet and the VFA provides updated foot locations to the NMPC. The NMPC after receiving these updated footholds delivers a solution by pitching up the robot base in order to adapt to the change in

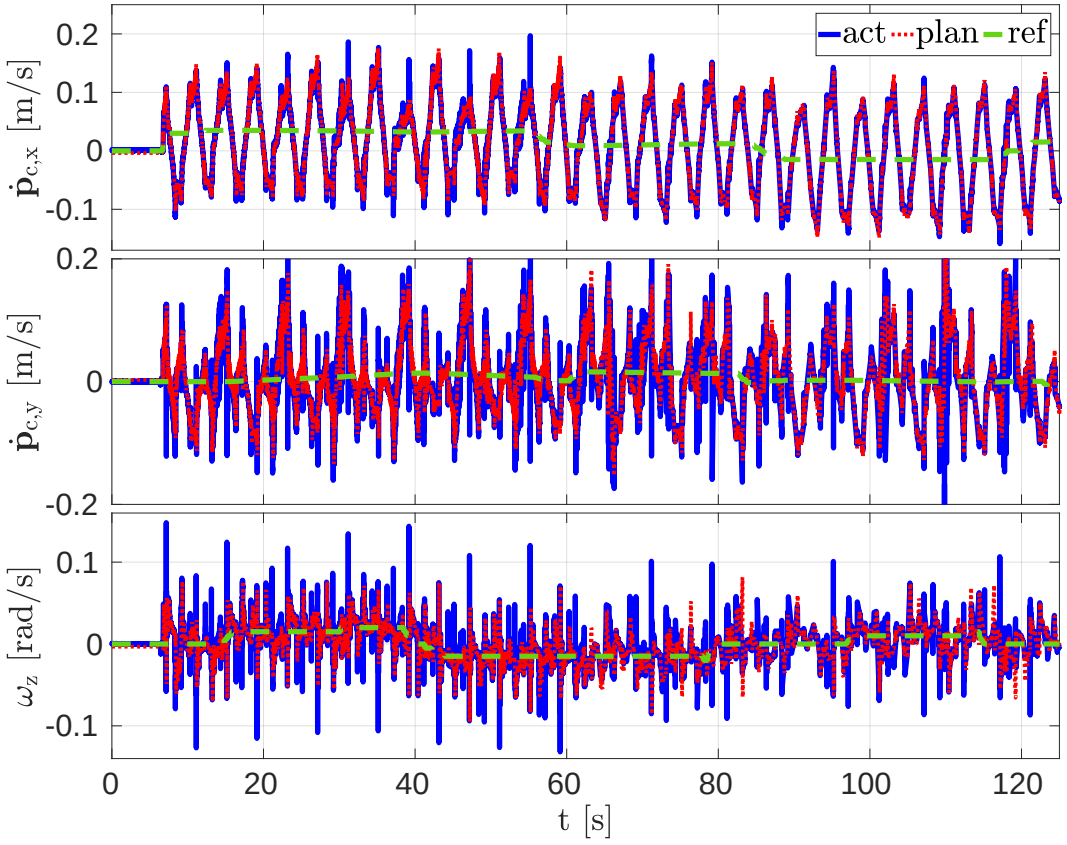

Figure 3.17: The longitudinal $\dot{\mathbf{p}}_{c,x}$, lateral $\dot{\mathbf{p}}_{c,y}$ and angular $\boldsymbol{\omega}_z$ velocity of the robot in omni-directional walk experiment. The blue, dotted red and the green line represent the actual, planned and reference values, respectively.

the environment while maintaining the mobility. Even though the mobility cost is defined for the stance legs, it is interesting to notice that the NMPC decides to adjust the base pitch while swinging the RF leg onto the pallet (Figure 3.20) by forecasting the change in the hip-to-foot distance at the touchdown. This experiment highlights the advantage of the predictive control over the traditional control approaches for its ability to incorporate the knowledge of the future states. It also validates the effectiveness of our mobility cost in the NMPC coupled with the VFA to adapt to the changes in the locomotion environment.

3.9 Discussion and Conclusion

3.9.1 Discussion

The MPC is a popular research topic which demonstrated significant improvements in recent years. The work presented in this chapter aims at improving state-of-the-art approaches by

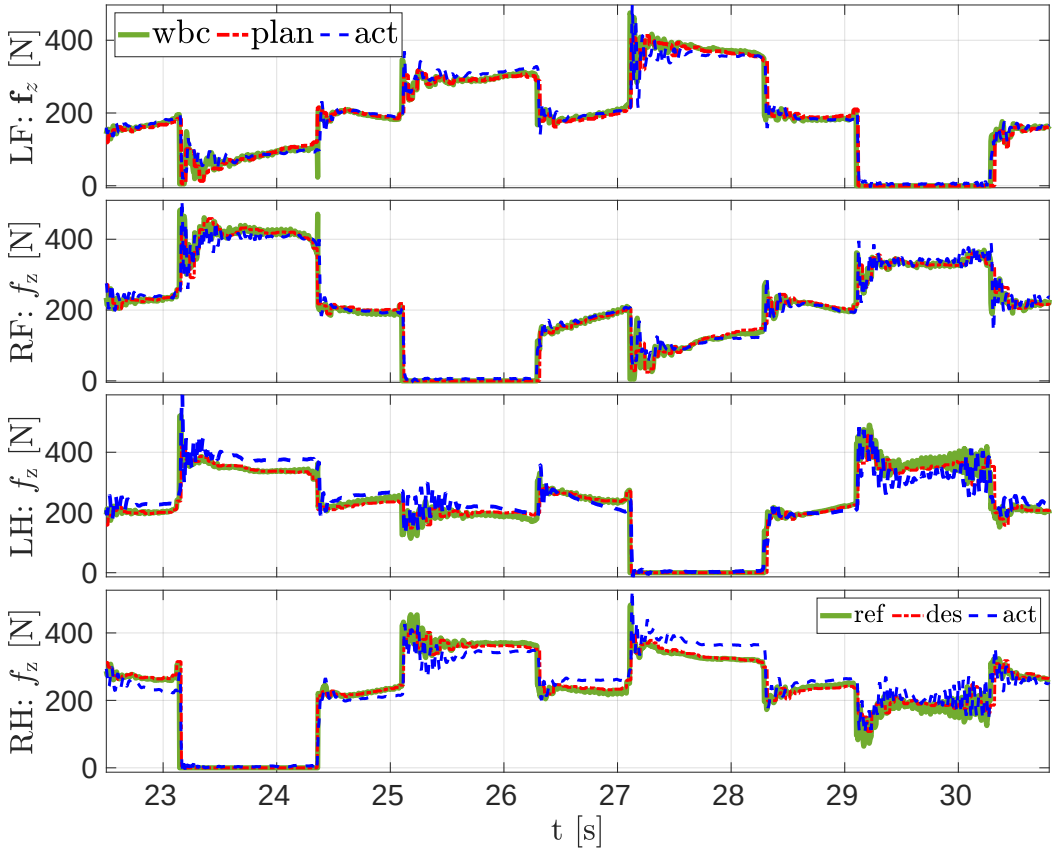

Figure 3.18: GRFs from one gait cycle in the omni-directional walk experiment (only one cycle for better visibility of the data). The green, dotted red and dashed blue line represent the output from WBC *i.e.*, $f^{\mathrm{d}}_{\mathrm{z},i}$, planned and actual values, respectively.

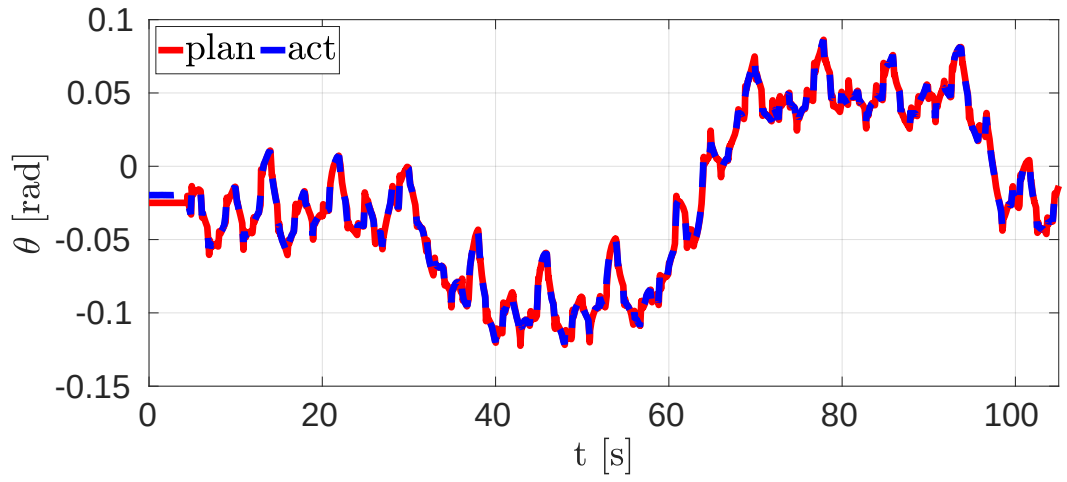

Figure 3.19: Planned (red) and Actual (dashed blue) pitch of the robot base while traversing a static pallet in the experiment at a commanded CoM X velocity of $0.03\,\mathrm{m/s}$.

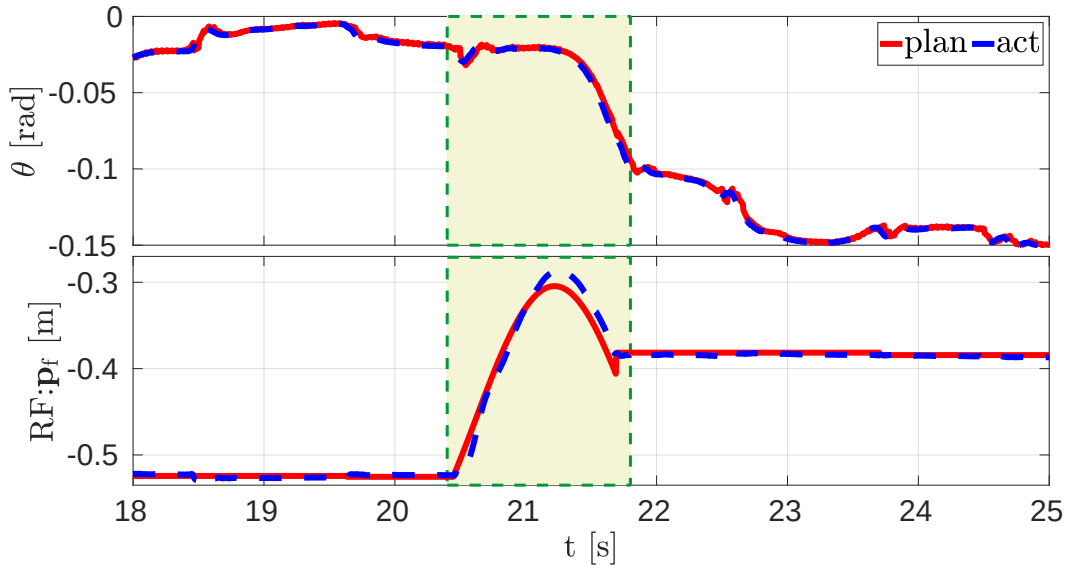

Figure 3.20: Robot base pitch achieved during the swing of RF leg while traversing a repositioned pallet in the experiment a commanded CoM X velocity of $0.05\,\mathrm{m/s}$. The red and dashed blue line are planned and actual values.

introducing a framework that runs at a (sufficient) high frequency and adapts the motion to the environment.

The RTI scheme for the NMPC enables one to close the loop at 25 Hz on the NMPC with a prediction horizon of 2 s. The replanning frequency is set to a conservative value to ensure that the synchronization is always satisfied, even in the presence of a few outliers. This choice has been validated with trials at lower frequencies and increases the robustness and reliability of the algorithm.

Other MPC works achieved higher replanning frequency, *e.g.,* [Neunert et al., 2018; Farshidian et al., 2017a], but in this work, I did not encounter the need for a faster replanning. Indeed, closing the loop on NMPC at 25 Hz allows us to compensate for the state drifts due to model inaccuracies and react to the change in the environment. Anyhow, the solver takes less than 10 ms on average to compute a new solution, so it is always possible to increase the frequency.

In addition, the strength of this formulation is that it is able to generate stable motions in non-flat terrain acquiring data from the heightmap (differently from [Neunert et al., 2018]) without the need to define heuristic references for orientation or kinematics limits. To the best of my knowledge, it is the first time that mobility has been addressed in an MPC fashion.

Limitations

The NMPC proved to be effective to compute feasible CoM and GRFs trajectories; anyway, it is worth highlighting limitations and possible improvements of the formulation. First of all,

the reference generator endows the NMPC with unfeasible, heuristic trajectories; for example, the X component of the reference GRFs is null even though the tracking of a linear X velocity is required. Consequently, tuning the weights becomes a crucial and challenging task that can degrade the performance. An optimization-based reference generator would overcome this limitation. Furthermore, feet locations and timings are parameters of the NMPC and thus are not optimized. In particular, the VFA improves the nominal position to deal with kinematic limits and morphology of the terrain but has no information on the optimal CoM trajectories.

3.9.2 Conclusion

In this chapter, I have presented a real-time NMPC which leverages optimization of leg mobility to achieve terrain adaptation. Including the contact sequence parameters (footholds and gait status) inside the SRBD model prevents one from adding the complementarity constraints explicitly.

In our NMPC, the mobility cost penalizes the hip-to-foot distance from a reference value corresponding to a high mobility factor, and hence it directs the NMPC to compute essential robot orientation to maintain high mobility while respecting the kinematic limits. This is evident from the pallet experiments where we also included VFA to correct undesired foot positions defined by the heuristics and avoid possible foot and shin collision. Accounting for the ZMP margin in our NMPC improved the locomotion stability of the robot in all of our experiments and simulation by keeping a sufficiently large ZMP margin from support polygon boundaries. Incorporating a force robustness term in the NMPC ensures that the GRFs stay close to the contact normals and hence, it enables the robot to cope with the estimation error of the orientation of the contact normals.

Omni-directional walk, traversing a static/relocated pallet and a set of simulations validate the locomotion framework on HyQ robot.

The work presented in this chapter has been published in [Rathod et al., 2021].

Chapter 4

Optimization-Based Reference Generator

4.1 Introduction

In the previous chapter, I introduced an NMPC-based locomotion framework for quadruped robots. In this framework, the reference generator module had the purpose of providing trajectories to be tracked in the cost function together with footholds and contact sequence. The NMPC computes the optimal trajectories which are then tracked by the WBC. This pipeline's schematic is shown in Figure 3.1.

The idea behind this work is that generally NMPC algorithms utilize reference trajectories computed from heuristics and a proper tuning of the weights is essential to obtain good performance. For instance, when a legged robot has to deal with some process noise (*e.g.,* react to disturbances from the environment) or track a specific goal with statically unstable gaits, the effectiveness of the algorithm can degrade.

Another shortcoming of a standard MPC implementation is related to the fact that the robot becomes "transparent" to external pushes, *i.e.,* at each sample the new trajectory starts from the *actual* position. Therefore an additional effort is required to track a specific *reference* Cartesian position or to return, after a push, to the original one. A common practice is to have user-defined values as a reference for the MPC [Di Carlo et al., 2018]. In these approaches the user would have to manually change the reference velocity in order to compensate for the deviation caused by the push. Another possible solution would be to add in parallel to the MPC a simple Cartesian PD control action to attract the CoM to a desired position [Grimminger et al., 2020]. However, this control strategy has some notable limitations, in particular:

1. the added wrench does not consider the wrench produced by the MPC layer (hence, there will be a fight between the two controllers with the risk of violating feasibility).
2. the PD is not aware of the hybrid dynamics of the legged robot, *i.e.,* the intermittent contacts and the (possible) under-actuation.
3. the PD approaches cannot change foot locations, which is crucial to deal with lateral pushes [Barasuol et al., 2013].

Another possibility to track a fixed goal is adding it to the cost function. This would solve the issue number 1 of the PD controller, because the constant position is embedded in the cost function of the MPC, hence the "conflict" is dealt with at the cost level resulting in feasible contact forces. However, this solution suffers from the fact that footholds cannot be changed, and thus the robot will be able to apply a limited resistive force before the legs lose their control authority (*e.g.,* when the CoM/ZMP goes out of the support polygon), and therefore is of restricted applicability.

Since, undoubtedly, accurately tuning the weights for the different cost terms is a tedious task [Bouyarmane and Kheddar, 2018] and having more physically meaningful references has been shown to improve performance [Bledt and Kim, 2019], the goal of this chapter is to improve the algorithm of Chapter 3 by including a novel optimization-based reference generator. Differently from [Bjelonic et al., 2022], the optimization is executed *online* at the same frequency as the NMPC.

The LIP model [Kajita et al., 2001] is used to compute reference trajectories for the CoM while taking into account the possible under-actuation of a gait (*e.g.,* in a trot). A QP Mapping is used then to compute the required forces to track those velocities. The foot locations are also corrected using the optimized reference to drive the robot towards the goal.
The effectiveness of the approach is validated both in simulations and experiments in three different scenarios:

(a) straight motion

(b) fixed lateral goal

(c) recovery after a push

I used the Aliengo robot of Unitree[1] shown in Figure 4.1. The proposed approach allows one to address all the limitations of the solution mentioned before.

[1] https://www.unitree.com/products/aliengo/

Figure 4.1: Picture of Unitree's Aliengo quadruped robot.

Outline

The rest of the chapter is organized as follows: Section 4.2 gives an overview of our planning framework, highlighting the main features of the reference generator. Section 4.3 describes the optimization problem with the LIP model and how it is used to compute CoM position velocity and GRFs references. Simulations and experiments with our Aliengo robot are presented in Section 4.4. Finally, discussion and conclusions are drawn in Section 4.5.

4.2 Locomotion Framework Description

Figure 4.2 gives an overview of the entire locomotion framework where we highlight the new reference generator. It is an extension of Figure 3.1, and it shows the structure of the new module. For the sake of clarity of the image, all the elements of WBC and PD Joint Controller have been represented with a unique block since they have not been modified.

The user decides the leg sequence, *i.e.,* type of gait (trot and walk) and order to move the legs, and the values of both linear $\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ and heading $\omega_{\mathrm{z}}^{\mathrm{usr}} \in \mathbb{R}$ CoM reference velocities, *i.e.,* the velocities that the robot should follow. Given the user-defined leg sequence, the *gait scheduler* (Section 3.5.1) returns the contact status (either swing or stance) $\delta_{i,k} \in \mathbb{R}$ of each leg i and for each time instant k in the *reference* horizon (divided into N_{g} intervals). [2]

The *robocentric stepping* module (Section 3.5.2) has the important task of computing footholds for each leg over the entire horizon. The sequences $\mathbf{p}_{\mathrm{f}}$ and $\boldsymbol{\delta}$ are the input parameters for the NMPC optimization.

If the error between the goal and average CoM position is lower than a threshold (Section 4.2.1), the reference generator does not perform optimizations and heuristic approaches (Eq. (3.10))

[2]Note that N_{g} can be different from the number of control intervals N used in the NMPC, with $N_{\mathrm{g}} \geq N$.

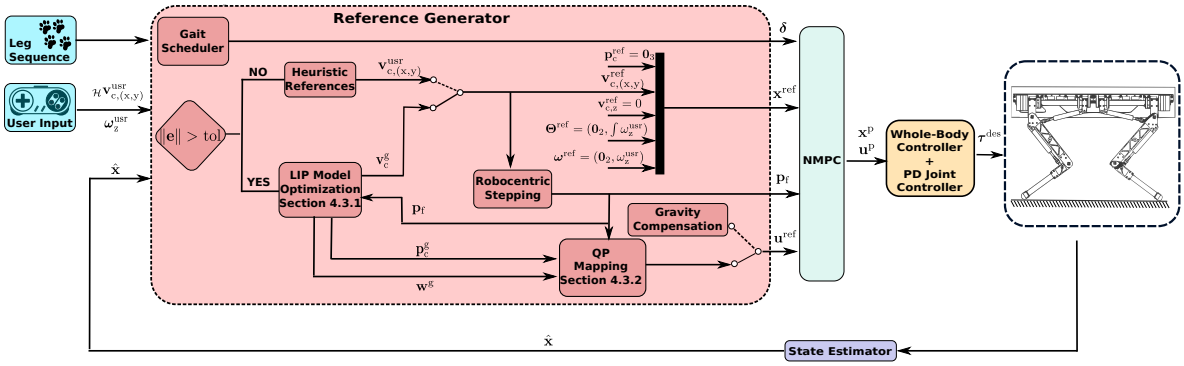

Figure 4.2: Block diagram of the proposed locomotion framework. Given the leg sequence, the user-defined velocities, and the actual state of the robot $\hat{\mathbf{x}}_\mathrm{c}$, the reference generator computes the gait status $\boldsymbol{\delta}$, the footholds $\mathbf{p}_\mathrm{f}$, the reference states $\mathbf{x}_\mathrm{c}^\mathrm{ref}$ and GRFs $\mathbf{u}^\mathrm{ref}$ for the NMPC at each control loop. If the norm of the error $\mathbf{e}$ between the goal and the average CoM position during the last gait cycle $\bar{\mathbf{p}}_\mathrm{c}$ is smaller than the threshold, simple heuristic techniques are used. In the other case, the reference generator is in the *optimize* status. CoM position and velocity to accomplish the task are computed with a LIP model; a QP mapping is used to obtain the GRFs corresponding to the ZMP computed by the LIP optimization. In addition, the CoM velocity is used to improve the footholds. Finally, the NMPC computes the *optimal* trajectories for CoM quantities $\mathbf{x}_\mathrm{c}^\mathrm{p}$ and GRFs $\mathbf{u}^\mathrm{p}$. A Whole-Body Controller and a PD-Joint controller compute the optimal torques $\boldsymbol{\tau}_\mathrm{d}$ that are sent to the robot.

are used to compute references for the NMPC. This condition is named *heuristic* reference generator. A description is provided in the previous chapter.

Instead, when the reference generator is in the *optimize* state, the sequences $\mathbf{p}_\mathrm{f}$ and $\boldsymbol{\delta}$ are the input parameters to a first-stage optimization that employs the LIP model (Section 4.3.1). Here we compute the optimal X-Y CoM trajectory $\mathbf{p}_\mathrm{c}^\mathrm{g}, \mathbf{v}_\mathrm{c}^\mathrm{g}$ to reach the goal $\mathbf{p}_\mathrm{c}^\mathrm{goal}$ where dynamic stability is satisfied (*i.e.,* ZMP always inside the support polygon). However, since the footholds are computed the first time with the simple heuristic approach, we need to recompute them using the optimized velocity.
To improve the accuracy, the LIP optimization can be repeated with the new set of footholds. These two operations, *i.e.,* LIP optimization and update of the footholds, can iterate until a stop condition is reached, *e.g.,* the maximum number of iterations or difference between two consecutive solutions below the defined threshold.

The CoM velocity trajectory $\mathbf{v}_\mathrm{c}^\mathrm{g}$ computed by the LIP model optimization is then used as velocity reference for the NMPC.[3] A QP-based mapping (Section 4.3.2) computes the references for the GRFs $\mathbf{u}^\mathrm{ref}$.

Finally, the output of the NMPC are the CoM trajectories (position, orientation, linear and

[3] as described in Section 3.5, reference CoM position, roll and pitch are not tracked, the reference for the yaw is obtained by integrating the user-defined yaw rate $\omega_\mathrm{z}^\mathrm{usr}$.

angular velocities) $\mathbf{x}_c^p$ and the GRFs $\mathbf{u}^p$ that will be sent to the Controller block. The Controller is composed of the 250 Hz WBC Interface (3.6.1), the 250 Hz WBC [Fahmi et al., 2019] and the 1 kHz PD-Joint controller. They generate the torque references $\boldsymbol{\tau}_d$ for the low-level joint controller of the robot. The State Estimator module [Nobili et al., 2017] provides the actual values $\hat{\mathbf{x}}_c$ of the robot at a frequency of 500 Hz.

4.2.1 Goal Setting and Status of the Reference Generator

As already explained in Section 4.1, the robot cannot follow a user-defined velocity in the presence of non-idealities or external disturbances,[4] due to the MPC transparency. For this reason, we define the goal $\mathbf{p}_c^{goal} \in \mathbb{R}$ as the X-Y position the robot would have reached if it had followed the user-commanded velocities $\mathbf{v}_c^{usr}$. The goal is updated at each iteration of the NMPC. It is initialized with $\mathbf{p}_c^{goal} = \hat{\mathbf{p}}_{c,(x,y)} + N_g \mathbf{v}_c^{usr} T_s$ at the beginning of an experiment and each iteration increments by $\mathbf{v}_c^{usr} T_s$. Variable T_s is the sampling time of the reference generator and it is equal to $1/f_s$ where f_s is the planning loop frequency.[5] As an alternative, the user can decide a fixed goal, either for one coordinate or both. We assume that there are no obstacles and the goal can be reached.

In order to change the status of the reference generator from *heuristic* to *optimize* in a meaningful way (*e.g.,* no activation due to the normal sway of the robot) the algorithm considers the *average* $\bar{\mathbf{p}}_c$ of the X-Y CoM position during the last gait cycle. To the average position, we add the offset due to the desired motion in the horizon N_g and compare it with the goal. The error is computed as: $\mathbf{e} = \mathbf{p}_c^{goal} - (\bar{\mathbf{p}}_c + N_g \mathbf{v}_c^{usr} T_s)$ and when its norm $\|\mathbf{e}\|$ goes beyond a threshold the reference generator status is set to *optimize*.

Algorithm 1 illustrates the pseudo-code of the different computation phases of the reference generator.

4.2.2 Formal Guarantees on Response Time

In an optimization problem, if we set the tracking of the goal as either a running cost or a terminal cost, the response will depend on the tuning of the weights of the cost itself. In addition, with this approach, we cannot impose a predefined time T_f in which the robot reaches the goal (*response horizon*).[6] For these two reasons a hard constraint to impose that

[4] the case of a fixed goal corresponds to a scenario in which $\mathbf{v}_{c,y}^{usr} = 0$.

[5] For the sake of simplicity, the sampling time T_s of the NMPC and the reference generator are set equal to the same value.

[6] The size of the horizon N_g will be linked to the response horizon T_f, therefore there will be a maximum value of T_f that can be achieved by the reference generator, related to the real-time constraint posed by the replanning frequency f_s. This will depend on the computational power of the machine where the optimization is run.

Algorithm 1 Reference Generator

Reference Generator status $\leftarrow$ HEURISTIC
$\mathbf{e} \leftarrow \mathbf{p}_{\mathrm{c}}^{\mathrm{goal}} - \bar{\mathbf{p}}_{\mathrm{c}} - \left(\int_0^{N_{\mathrm{g}}T_{\mathrm{s}}} \mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} dt\right)$
if $\|\mathbf{e}\| > tol$ **then**
    Reference Generator status $\leftarrow$ OPTIMIZE
$\mathbf{p}_{\mathrm{f}}, \boldsymbol{\delta} \leftarrow$ *Heuristic References*
if OPTIMIZE Reference Generator **then**
    while stop condition is not reached **do**
        $\mathbf{v}^{\mathrm{g}} \leftarrow$ *LIP Model Optimization*$(\mathbf{p}_{\mathrm{c}}^{\mathrm{goal}}, \hat{\mathbf{x}}_{\mathrm{c}})$
        $\mathbf{p}_{\mathrm{f}} \leftarrow$ *Robocentric Stepping*$(\mathbf{v}_{\mathrm{c}}^{\mathrm{g}})$
    $\mathbf{v}^{\mathrm{ref}} \leftarrow \mathbf{v}^{\mathrm{g}}$
    $\mathbf{u}^{\mathrm{ref}} \leftarrow$ *QP Mapping*$(\mathbf{p}_{\mathrm{c}}^{\mathrm{g}}, \mathbf{w}^{\mathrm{g}}, \mathbf{p}_{\mathrm{f}})$
else
    $\mathbf{v}^{\mathrm{ref}} \leftarrow \mathbf{v}^{\mathrm{usr}}$
    $\mathbf{u}^{\mathrm{ref}} \leftarrow$ *Gravity Compensation*
$\mathbf{p}_{\mathrm{c}}^{\mathrm{goal}} \leftarrow \mathbf{p}_{\mathrm{c}}^{\mathrm{goal}} + \mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} T_s$
solve the NMPC

the CoM position matches the goal can be added to guarantee a response interval equal to T_{f}, starting from the first time instant when the reference generator is set to the *optimize* status. In addition, we also need to impose the same hard constraint on all the samples after $M = T_{\mathrm{f}}/T_{\mathrm{s}}$; in this way once reached, the goal is continuously tracked by the reference generator. Indeed, at every iteration of the NMPC, the interval without constraint should shrink because otherwise the target would never be reached, see Fig. 4.3. For this reason we enforce hard constraints on the goal, with the number of samples M gradually reducing to zero. In this way, the goal is reached in a time that does not depend on the tuning of the weights. In practice, using hard constraints might lead the optimization to be trapped in an infeasible solution [Hult et al., 2019]. Therefore, an implementation with slacks variables in the constraints, that allows one to relax them, is preferable. Penalizing their value in the cost function of problem (4.2) guarantees that the robot reaches the target in a predefined time interval; if this is not possible, the solver will find the "best" feasible solution.

4.3 Optimized Reference Generator

As already mentioned, the task of the reference generator is to compute along a horizon N_{g} the linear (*i.e.,* longitudinal and lateral) CoM velocity reference trajectories $\mathbf{v}_{\mathrm{c}}^{\mathrm{g}}$ to reach a desired goal/recover from an external disturbance in a predefined time T_{f} and the trajectory of GRFs $\mathbf{u}^{\mathrm{ref}}$ to follow it.

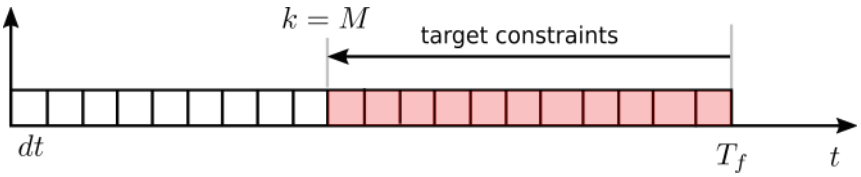

Figure 4.3: Pictorial representation of the goal constraints enforced to achieve a certain response time T_f. The highlighted blocks corresponds to the nodes of the LIP optimization in which the constraint (4.2f) is active. At every iteration the variable M is decreased such that the time to reach the target matches the predefined time T_f.

4.3.1 LIP Model Optimization

One of the features of the proposed reference generator is to take into account the intrinsic under-actuated nature of a robot when only two feet are on the ground [Chignoli and Wensing, 2020]. Therefore, we employ the simplest model that is able to capture this under-actuation: the LIP model [Kajita et al., 2001]. Indeed, this allows computing the optimal trajectory for the CoM, while imposing a desired behavior for the ZMP. The ZMP is defined as "a point on the ground at which the tangential component of the moment generated by the ground reaction force/moment becomes zero" [Harada et al., 2003], and its position determines the direction and magnitude of the CoM acceleration. Guaranteeing that the GRFs are such that the resulting ZMP is inside the support polygon ensures that they also satisfy the unilateral constraints [Vukobratovic and Borovac, 2004] (the legs can only push and not pull the ground); therefore they can be effectively realized by a real robot. Since the support polygon boils down to a line connecting the stance feet during a two-leg-stance phase, the ZMP will be able to move only on that segment. Even though the ZMP-based models have been efficiently used in MPC approaches, *e.g.,* [Bellicoso et al., 2018], the LIP model presents some assumptions (vertical and angular dynamics are neglected). It is worth remarking that, aiming to find a compromise between accuracy and computational efficiency, NMPC uses a more complex model (the SRBD [Orin et al., 2013]) that will consider these dynamics in the lower optimization stage. ion Let us define the state of the reference generator $\mathbf{x}^\mathrm{g} = \{\mathbf{x}^\mathrm{g}_0, \dots \mathbf{x}^\mathrm{g}_\mathrm{Ng}\} \in \mathbb{R}^{4\times(N_\mathrm{g}+1)}$, with $\mathbf{x}^\mathrm{g}_\mathrm{k} = \left[\mathbf{p}^\mathrm{g}_\mathrm{c,k}, \mathbf{v}^\mathrm{g}_\mathrm{c,k}\right]^\top \in \mathbb{R}^4$ the stack of X-Y CoM position and velocity at time k. The control inputs are $\mathbf{w}^\mathrm{g} \in \mathbb{R}^{2\times N_\mathrm{g}} = \{\mathbf{w}^\mathrm{g}_0, \dots \mathbf{w}^\mathrm{g}_{N_\mathrm{g}-1}\}$, with $\mathbf{w}^\mathrm{g}_k \in \mathbb{R}^2$ X-Y position of the ZMP at time k. Footholds $\mathbf{p}_\mathrm{f,k}$ and gait status $\boldsymbol{\delta}_\mathrm{k}$ for all the feet at each sample k are the parameters used to compute the support polygon at each node of the horizon N_g. Additional parameters are initial Z CoM position $\hat{\mathbf{p}}_\mathrm{c,z}$ and the gravity $g \in \mathbb{R} = 9.81\mathrm{m/s}^2$.

To obtain $\mathbf{x}^{\mathrm{g}}$ and $\mathbf{w}^{\mathrm{g}}$ the following optimization problem is introduced:

$$\min_{\mathbf{x}^{\mathrm{g}},\mathbf{w}^{\mathrm{g}}} \quad \sum_{k=0}^{N_{\mathrm{g}}} \parallel \mathbf{p}_{\mathrm{c,k}}^{\mathrm{g}} - \mathbf{p}_{\mathrm{c}}^{\mathrm{goal}} \parallel_{\mathbf{Q}_{\mathrm{p}}}^{2} + \parallel \mathbf{v}_{\mathrm{c,k}}^{\mathrm{g}} \parallel_{\mathbf{Q}_{\mathrm{v}}}^{2} + \tag{4.1a}$$

$$\sum_{k=0}^{N_{\mathrm{g}}-1} \parallel \mathbf{w}_{\mathrm{k}}^{\mathrm{g}} - \mathbf{w}_{\mathrm{k}}^{\mathrm{ref}} \parallel_{\mathbf{Q}_{\mathrm{w}}}^{2} \tag{4.1b}$$

$$\text{s.t.} \quad \mathbf{x}_{0}^{\mathrm{g}} = \hat{\mathbf{x}}_{(\mathrm{x,y})} \tag{4.1c}$$

$$\mathbf{x}_{\mathrm{k+1}}^{\mathrm{g}} = \mathbf{x}_{\mathrm{k}}^{\mathrm{g}} + \begin{bmatrix} \mathbf{v}_{\mathrm{c,k}}^{\mathrm{g}} T_{\mathrm{s}} + \frac{T_{\mathrm{s}}^{2}\mathrm{g}}{2\hat{\mathbf{p}}_{\mathrm{c,z}}} \left(\mathbf{p}_{\mathrm{c,k}}^{\mathrm{g}} - \mathbf{w}_{\mathrm{k}}^{\mathrm{g}} \right) \\ \\ \frac{\mathrm{g}}{\hat{\mathbf{p}}_{\mathrm{c,z}}} \left(\mathbf{p}_{\mathrm{c,k}}^{\mathrm{g}} - \mathbf{w}_{\mathrm{k}}^{\mathrm{g}} \right) T_{\mathrm{s}} \end{bmatrix} \tag{4.1d}$$

$$\mathbf{w}_{\mathrm{k}}^{\mathrm{g}} \in \mathcal{S}(\mathbf{p}_{\mathrm{f,k}}, \boldsymbol{\delta}_{\mathrm{k}}) \qquad k \in \mathbb{I}_{0}^{N_{\mathrm{g}}-1} \tag{4.1e}$$

The terms (4.1a) and (4.1b) of the cost function aims to minimize the distance between CoM position and the goal, the norm of the velocities and the distance of the ZMP from the center of the support polygon $\mathbf{w}_{\mathrm{k}}^{\mathrm{ref}}$ (for robustness purposes). In addition, the minimization of the velocity term (4.1a) prevents having large velocities that could determine footholds outside of the workspace of the leg due to the robocentric stepping. Matrices $\mathbf{Q}_{\mathrm{p}} \in \mathbb{S}_{+}^{2}$, $\mathbf{Q}_{\mathrm{v}} \in \mathbb{S}_{+}^{2}$ and $\mathbf{Q}_{\mathrm{w}} \in \mathbb{S}_{+}^{2}$ are positive definite weighting matrices. The initial condition (4.1c) is expressed by setting $\mathbf{x}_{0}^{\mathrm{g}}$ equal to the corresponding values $\hat{\mathbf{x}}_{\mathrm{c}}$ received from the State Estimator. Equation (4.1d) corresponds to the discrete CoM dynamics for the LIP model. Equation (4.1e) imposes that the ZMP always lies inside the support polygon $\mathcal{S}(\mathbf{p}_{\mathrm{f,k}}, \boldsymbol{\delta}_{\mathrm{k}})$. The symbol $\mathbb{I}_{0}^{N_{\mathrm{g}}-1}$ indicates the set of integer numbers in the closed interval $[0\ ,\ N_{\mathrm{g}} - 1]$.

The optimization problem (4.1) does not have any guarantee on the response time. As already discussed in Section 4.2.2 a formulation with slack variables can be used to impose a predefined

instant in which the robot has to reach the target:

$$\min_{\mathbf{x}^{\mathrm{g}},\mathbf{w}^{\mathrm{g}},\mathbf{s}} \quad \sum_{k=0}^{N_{\mathrm{g}}} \| \mathbf{v}^{\mathrm{g}}_{\mathrm{c,k}} \|^2_{\mathbf{Q}_{\mathrm{v}}} + \sum_{k=0}^{N_{\mathrm{g}}-1} \| \mathbf{w}^{\mathrm{g}}_{\mathrm{k}} - \mathbf{w}^{\mathrm{ref}}_{\mathrm{k}} \|^2_{\mathbf{Q}_{\mathrm{w}}} + \tag{4.2a}$$

$$\sum_{k=0}^{N_{\mathrm{g}}} \left(\| \mathbf{s}_{\mathrm{k}} \|^2_{\mathbf{Q}_{\mathrm{s,q}}} + \mathbf{Q}_{\mathrm{s,l}} \mathbf{s}_{\mathrm{k}} \right) \tag{4.2b}$$

$$\text{s.t.} \quad \mathbf{x}^{\mathrm{g}}_{\mathrm{c},0} = \hat{\mathbf{x}}_{(\mathrm{x,y})} \tag{4.2c}$$

$$\mathbf{x}^{\mathrm{g}}_{\mathrm{c,k+1}} = \mathbf{x}^{\mathrm{g}}_{\mathrm{c,k}} + \begin{bmatrix} \mathbf{v}^{\mathrm{g}}_{\mathrm{c,k}} T_{\mathrm{s}} + \frac{T_s^2 \mathrm{g}}{2\hat{\mathbf{p}}_{\mathrm{c,z}}} \left(\mathbf{p}^{\mathrm{g}}_{\mathrm{c,k}} - \mathbf{w}^{\mathrm{g}}_{\mathrm{k}} \right) \\ \\ \frac{\mathrm{g}}{\hat{\mathbf{p}}_{\mathrm{c,z}}} \left(\mathbf{p}^{\mathrm{g}}_{\mathrm{c,k}} - \mathbf{w}^{\mathrm{g}}_{\mathrm{k}} \right) T_{\mathrm{s}} \end{bmatrix} \tag{4.2d}$$

$$\mathbf{w}^{\mathrm{g}}_{\mathrm{k}} \in \mathcal{S}(\mathbf{p}_{\mathrm{f,k}}, \boldsymbol{\delta}_{\mathrm{k}}) \qquad \mathrm{k} \in \mathbb{I}_0^{N_{\mathrm{g}}-1} \tag{4.2e}$$

$$\begin{bmatrix} \mathbf{s}_{\mathrm{k,x}} \\ \mathbf{s}_{\mathrm{k,y}} \end{bmatrix} \geq \begin{bmatrix} |\mathbf{p}^{\mathrm{g}}_{\mathrm{c,x,k}} - \mathbf{p}^{\mathrm{goal}}_{\mathrm{c,x}}| \\ |\mathbf{p}^{\mathrm{g}}_{\mathrm{c,y,k}} - \mathbf{p}^{\mathrm{goal}}_{\mathrm{c,y}}| \end{bmatrix} \qquad \mathrm{k} \in \mathbb{I}_M^{N_{\mathrm{g}}} \tag{4.2f}$$

$$\mathbf{s}_{\mathrm{k,x}}, \mathbf{s}_{\mathrm{k,y}} \geq 0 \qquad \mathrm{k} \in \mathbb{I}_0^{N_{\mathrm{g}}} \tag{4.2g}$$

The slack variable $\mathbf{s}_{\mathrm{k}} \in \mathbb{R}^{2\times 1}$ is added in the cost term (4.2b) and thanks to Eq. (4.2f), and Eq. (4.2g) it allows one to impose that the robot CoM coincides with the goal after M samples. Differently from the optimization problem (4.1), the cost function does not explicitly include a position term, since it is enforced with slacks into Eq. (4.2b). Matrices $\mathbf{Q}_{\mathrm{s,q}} \in \mathbb{S}^2_+$ and $\mathbf{Q}_{\mathrm{s,l}} \in \mathbb{R}^{1\times 2}$ are the additional weights for the quadratic and linear terms for the slack variables. Equation (4.2c), (4.2d), (4.2e) are the same as Eq. (4.1c), (4.1d), (4.1e) respectively.

4.3.2 QP Mapping

The reference generator computes CoM trajectories which the NMPC must follow, but the latter also requires reference GRFs $\mathbf{u}^{\mathrm{ref}} \in \mathbb{R}^{12\times N}$. Since the output of the optimization problem (4.1) or (4.2) is a trajectory for the ZMP $\mathbf{w}^{\mathrm{g}}$, we need to map this into a set of consistent GRFs, thus moving from a bi-dimensional to the higher dimensional space of contact forces. For this reason, let us define the set of indices of the legs in contact with the ground by $\mathbb{C}$. A parametric QP is solved to find the vector of GRFs $\mathbf{u}^{\mathrm{qp}} \in \mathbb{R}^{3|\mathbb{C}|}$ which corresponds to the ZMP location $\mathbf{w}^{\mathrm{g}}$. If a foot is in the swing phase, its GRFs are set to 0 by default:

$$\mathbf{u}^{\mathrm{ref}}_{\mathrm{i}} = \begin{cases} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i}} & \mathrm{i} \in \mathbb{C} \\ \mathbf{0}_{3\times 1} & \mathrm{i} \notin \mathbb{C} \end{cases} \tag{4.3}$$

where i is the leg index.

Note that we need to solve a QP for each sample k in the horizon N, therefore to avoid overloading the notation, we do not specify the subscript k in the following quantities.

The parameters of the model are the footholds $\mathbf{p}_{\mathrm{f}}$, the X-Y components of the CoM position $\mathbf{p}_{\mathrm{c}}^{\mathrm{g}}$, the ZMP trajectory $\mathbf{w}^{\mathrm{g}}$, the initial Z CoM position $\hat{\mathbf{p}}_{\mathrm{c,z}}$ and the mass of the robot $m \in \mathbb{R}$. Thus, for every k we solve:

$$\min_{\mathbf{u}^{\mathrm{qp}}} \quad \| \mathbf{u}^{\mathrm{qp}}_{(\mathrm{x,y})} \|^2_{\mathbf{Q}_{\mathrm{u}}} + \left\| \sum_{i \in \mathbb{C}} \left[\mathbf{p}_{\mathrm{f,i}} - \left[\mathbf{p}_{\mathrm{c}}^{\mathrm{g}}, \hat{\mathbf{p}}_{\mathrm{c_z}} \right]^{\top} \right]_{\mathrm{x}} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i}} \right\|^2_{\mathbf{Q}_{\mathrm{k}}} \tag{4.4a}$$

$$\text{s.t.} \quad \mathbf{w}^{\mathrm{g}}_{\mathrm{x,y}} = \frac{\sum_{i \in \mathbb{C}} \mathbf{p}_{\mathrm{f_i,(x,y)}} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i,z}}}{\sum_{i \in \mathbb{C}} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i,z}}} \tag{4.4b}$$

$$\sum_{i \in \mathbb{C}} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i,z}} = m\mathrm{g} \tag{4.4c}$$

$$\frac{g}{\hat{\mathbf{p}}_{\mathrm{c,z}}} (\mathbf{p}_{\mathrm{c}}^{\mathrm{g}} - \mathbf{w}^{\mathrm{g}}) = \frac{\sum_{i \in \mathbb{C}} \mathbf{u}^{\mathrm{qp}}_{\mathrm{i,(x,y)}}}{m} \tag{4.4d}$$

The first term of Equation (4.4a) is the regularization term on the X-Y components of the GRFs, with $\mathbf{Q}_{\mathrm{u}} \in \mathbb{S}^2_+$ as weighting matrix, while the second one minimizes the angular momentum rate, and it is weighted by $\mathbf{Q}_{\mathrm{k}} \in \mathbb{S}^3_+$. Variable $\left[\cdot \right]_{\mathrm{x}}$ represents the skew-symmetric matrix associated with the cross product. Equation (4.4b) is the definition of ZMP, Equation (4.4c) represents the gravity compensation. Equation (4.4d) guarantees that the X-Y CoM acceleration computed with the LIP model in the reference generator (left term) coincides with the one of the SRBD used in NMPC (right term). It is worth highlighting that when the robot is on two legs, the ZMP lies on a line, and so it can only move in a 1D manifold. During that phase, imposing the Eq. (4.4b) for the X component is enough to guarantee that also the Y coordinate of the ZMP respects the same constraint.

Along the horizon N_{g}, each problem is independent of the others, so they can be solved in parallel. In this way the computation effort remains low (a couple of ms to solve a QP problem with linear constraints) and therefore the new reference generator can be integrated into our high-frequency NMPC scheme.

4.4 Simulation and Experimental Results

The optimization-based reference generator endows the NMPC planner with the capability to recover from disturbances and avoid drifting in the face of non-idealities. To show its effectiveness, three template scenarios are tailored:

(a) straight motion

(b) fixed lateral goal

(c) recovery after a push

Table 4.1: Weights used in the Governor

Cost	Weight	Value
Velocities LIP	$\mathbf{Q}_{\mathrm{v}}$	$\mathrm{diag}(200, 300)$
ZMP	$\mathbf{Q}_{\mathrm{w}}$	$\mathrm{diag}(100, 350)$
Slack	$\mathbf{Q}_{\mathrm{s,q}}$ $\mathbf{Q}_{\mathrm{s,l}}$	$\mathrm{diag}(0, 1000)$ $[0, 1000]$
Forces QP mapping	$\mathbf{Q}_{\mathrm{u}}$	$\mathrm{diag}(100, 100)$
Angular Momentum Rate	$\mathbf{Q}_{\mathrm{k}}$	$\mathrm{diag}(1, 1, 1)$

I performed the simulations and experiments on the 22 kg quadruped robot AlienGo of Unitree. The trot is considered as a template gait for our experiments because of its inherently unstable nature. Indeed, any asymmetry in the real robot can make it drift when setting a pure forward velocity. The quadruped is thus not able to follow a straight line.

The variable T_{s} is set to 40 ms and the horizon length of both reference generator (N_{g}) and NMPC (N) are 50 nodes corresponding to an interval of 2 s; the planning loop frequency is set to 25Hz as already discussed in Chapter 3. The trot gait parameters are cycle time T_{c}= 1 s, duty factor $D = 0.65$. [7] The values of the weighting matrices are reported in Table 4.1. Without any lack of generalization, slack variables are considered only for the Y component.

Similarly to the NMPC, the HPIPM [Frison and Diehl, 2020] solver integrated into `acados` [Verschueren et al., 2019] library is used to find the solutions of the problem (4.2). The problem (4.4), instead, is solved using `eiquadprog` [Guennebaud et al., 2011], since it offers a more straightforward interface for the QPs.

Both reference generator and NMPC run on an Intel Core i7-10750H CPU @ 2.60 GHz.

For the sake of clarity, I recall that the adjective *reference* refers to the output of the reference generator, *planned* to the output of the NMPC, and *actual* to the real values measured by the State Estimator.

The video showing the achieved results, both in simulation and real experiment, can be seen at the following link:
https://youtu.be/Jp0D8_AKiIY.

[7] The duty factor is defined as the ratio between the duration of the stance phase and cycle time.

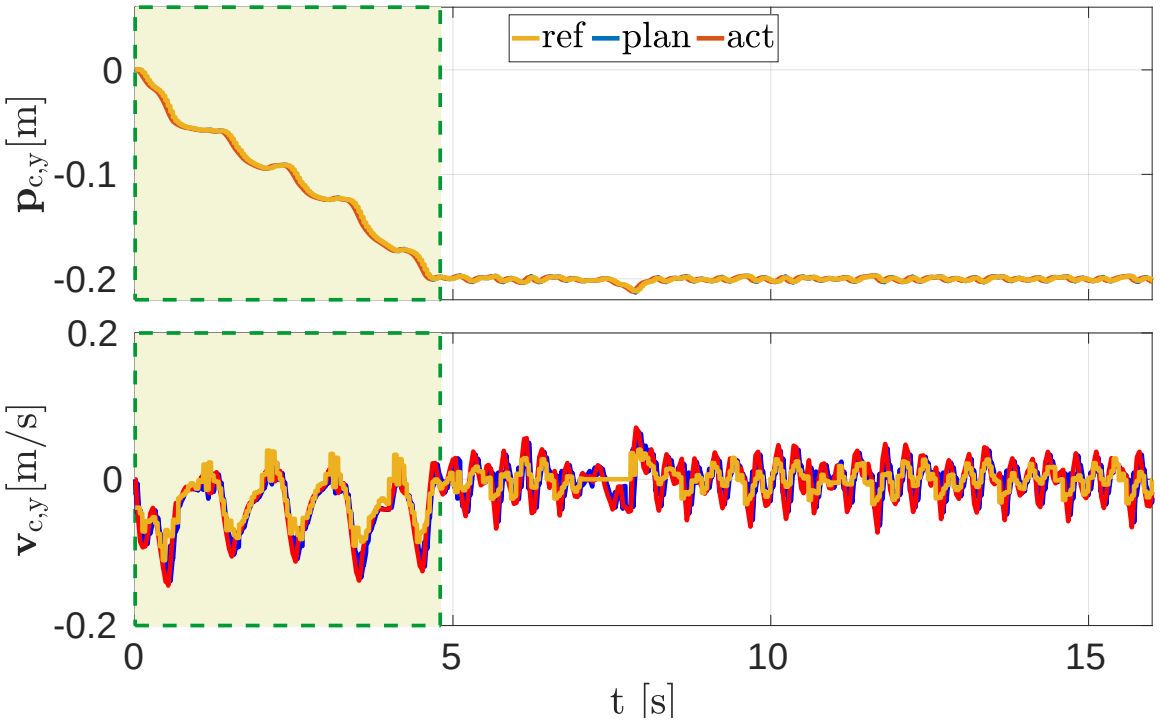

Figure 4.4: Simulation, scenario (b): CoM Y position and velocity in a scenario in which the robot has to reach a target of -0.2 m, with a response horizon of 4.8 s (120 iterations of the NMPC, shadowed blocks). Yellow lines are the output of the reference generator, which is used as reference for the NMPC. Blue lines corresponds to the planned trajectories, computed by the NMPC and tracked by the WBC. The red lines report the actual values.

4.4.1 Simulations

The four-leg-stance phase in a walking trot is the only moment of the gait where the robot has full control authority and is able to track the reference velocities. Neglecting this fact would lead to failure or unpredictably longer response times. In our algorithm, the reference generator already takes into account the under-actuation, enabling us to deal with this issue successfully. The simulations of the scenario (a) are only reported in the uploaded video.

Scenario (b)

Figure 4.4 refers to the simulation of the scenario (b) in which the robot has to go to a lateral target (-0.2 m on the Y coordinate) in a predefined time ($T_\mathrm{f} = 4.8$ s) and then keep it while walking. Since X-Y directions are decoupled in the LIP model, the X component of the goal $\mathbf{p}_{\mathrm{c,x}}^{\mathrm{goal}}$ is updated at each iteration of the NMPC according to Algorithm 1 and continuously tracked. The top plot demonstrates that the reference generator is able to compute reference CoM trajectories (yellow line) that allow the actual CoM values (red line) to accomplish the

task. Since the reference quantities satisfy the under-actuation of the gait, the velocities are nicely tracked by the NMPC (respectively yellow and blue line,in the bottom part of Figure 4.4) and consequently by the actual ones.

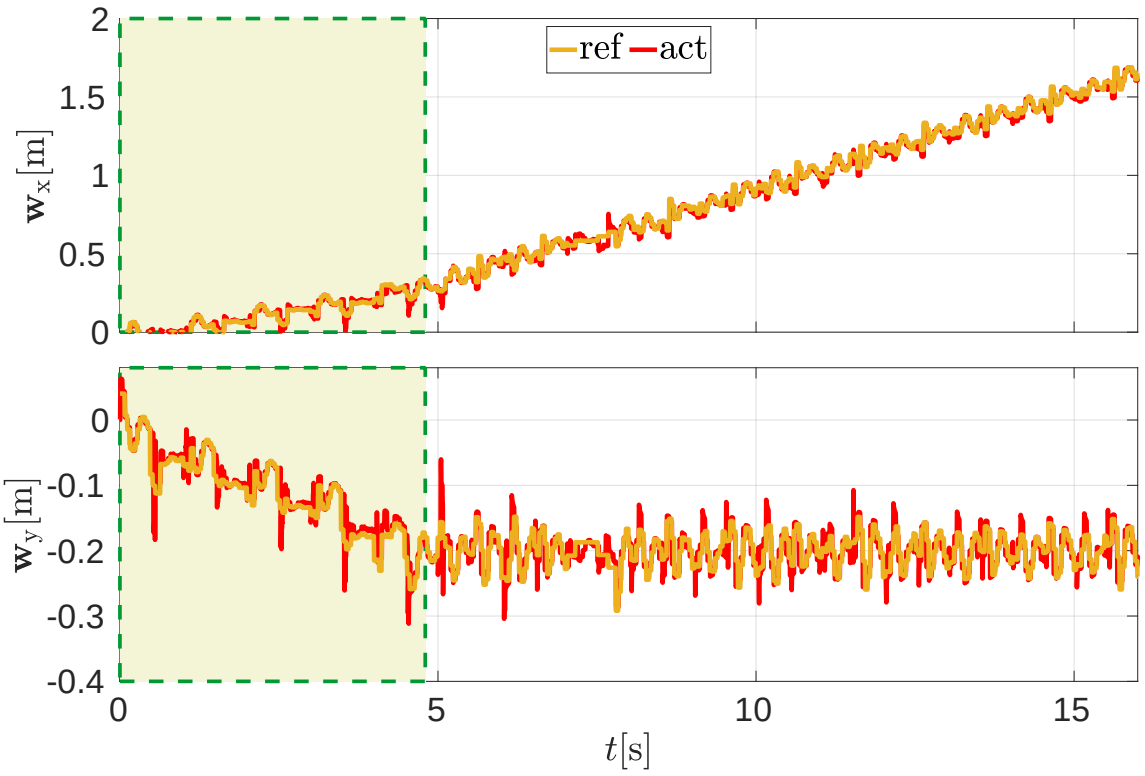

Figure 4.5: Simulation, scenario (b): X-Y ZMP location in a scenario in which the robot has to reach a target of -0.2 m along Y direction with a response time $T_\mathrm{f} = 4.8$ s (120 iterations of the NMPC). The yellow line corresponds to the output of the reference generator, while the red has been computed using Equation (4.4b).

Figure 4.5, instead, shows the reference $\mathbf{w}^\mathrm{g}$ and actual X-Y components of the ZMP locations (computed by Eq. (4.4b)) for the same simulation.

As it can be seen, the robot is able to track the reference values for the entire cycle, for both X-Y coordinates.

In this simulation the reference generator is always kept in the *optimize* status to show how the algorithm is able to track (once the target has been reached) a zero user-defined lateral velocity $\mathbf{v}^\mathrm{usr}_\mathrm{c,y}$.

Figure 4.6 reports scenario (b) with the same goal (-0.2 m) but different response horizon ($T_\mathrm{f} = 3,2$ s). Due to the smaller interval, the response is more aggressive and presents an overshoot which is then recovered within the 3 s interval. The task is achieved by simply modifying the response horizon T_f; no tuning of the weights of the cost function is required. Shaded areas highlight the response horizon. Figure 4.7 shows the tracking error between the

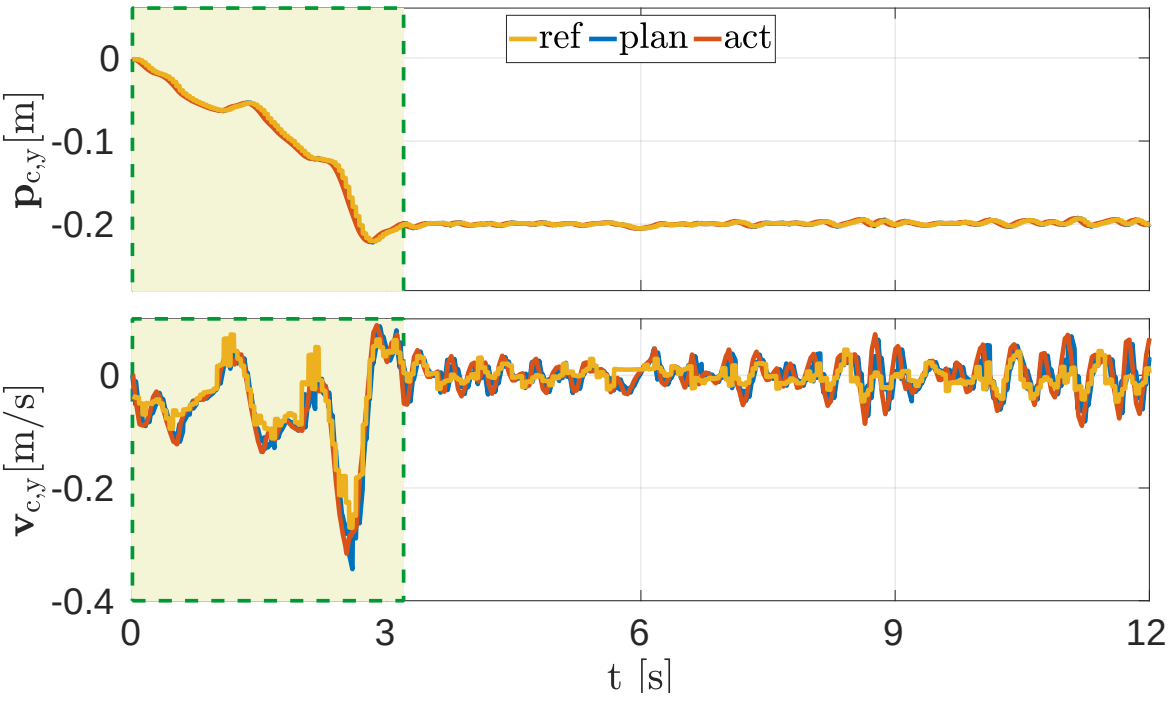

Figure 4.6: Simulation, scenario (b): CoM Y position and velocity when the robot has to reach a target of -0.2 m with a response interval of $T_f = 3,2$ s (80 iterations of the NMPC)). As in Fig. 4.4, yellow, blue, and red lines correspond respectively to reference, desired, and actual quantities.

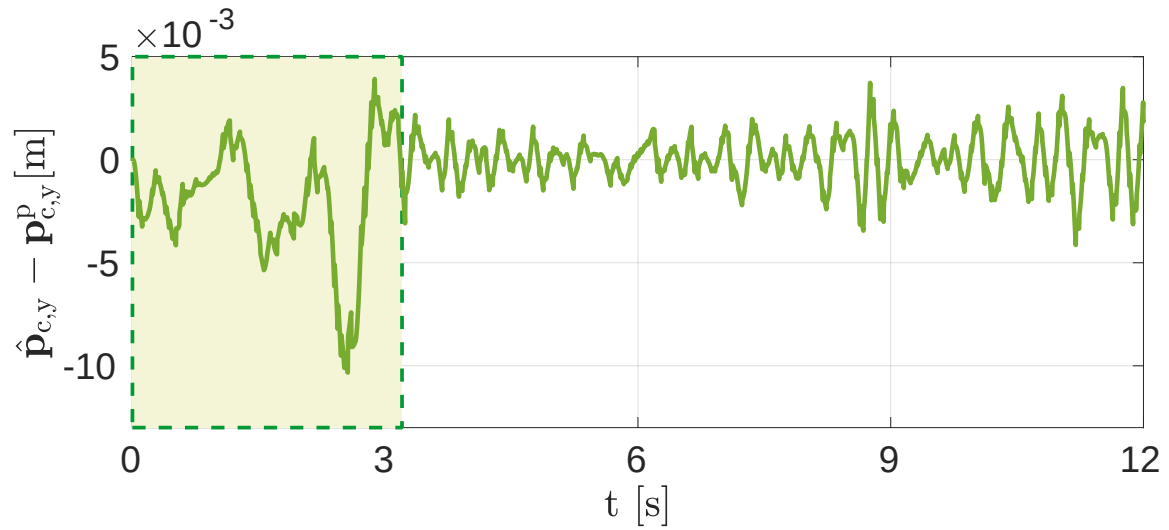

Figure 4.7: Simulation, scenario (b): goal of -0.2 m with $T_f = 3.2$ s. Tracking error between actual and desired Y CoM position ($\hat{\mathbf{p}}_{c,y} - \mathbf{p}^p_{c,y}$).

actual and desired Y CoM position. It confirms that the error is negligible and the desired values are tracked by the robot.

Scenario (c)

Next, to validate the approach a comparison with a simple cartesian PD approach in the scenario (c) is performed.

This is typically implemented computing a feed-forward force $f_{\mathrm{ff,y}} \in \mathbb{R}$ which depends on the error between the actual state and the goal, *i.e.,*

$$f_{\mathrm{ff,y}} = K_{\mathrm{p}} \left(\mathbf{p}_{\mathrm{c,y}}^{\mathrm{goal}} - \hat{\mathbf{p}}_{\mathrm{c,y}} \right) + K_{\mathrm{d}} \left(\mathbf{v}_{\mathrm{c,y}}^{\mathrm{usr}} - \hat{\mathbf{v}}_{\mathrm{c,y}} \right) \tag{4.5}$$

For the first 5 seconds of the motion the robot is pushed with a force of 15 N in the Y direction. The user velocity $\mathbf{v}_{\mathrm{c,y}}^{\mathrm{usr}}$ is zero, so the task is to come back to the initial Y position. In Figure 4.8 I report the actual Y CoM position $\hat{\mathbf{p}}_{\mathrm{c,y}}$ in three different cases.

The green line represents the same scenario considering the system as a second order with a critically damped response.

From the control theory, we know that the settling time is equal to $T_{\mathrm{f}} = \frac{4}{\zeta \omega_{\mathrm{n}}}$, with ζ equal to damping ratio and ω_{n} natural frequency. Rewriting the second-order equation of the mass/spring/damper as a function of ζ and ω_{n} we have $K_{\mathrm{p}} = m\omega_{\mathrm{n}}^2$ and $K_{\mathrm{d}} = 2\zeta\sqrt{mK_{\mathrm{p}}}$.
Merging the two expressions it results $T_{\mathrm{f}} = 4\sqrt{\frac{m}{K_{\mathrm{p}}}}$ for $\zeta = 1$.
Considering $T_{\mathrm{f}} = 4.8$ s and a total robot mass m $= 22$ kg, K_{p} is equal to 15 N/m and consequently $K_{\mathrm{d}} = 36\,\mathrm{Ns/m}$.

It is evident that the CoM is far from reaching the goal in the predefined time. Instead, The purple line represents the case in which the values of the proportional and derivative gain matrix are manually tuned so that the robot reaches the goal faster. The result is $K_{\mathrm{p}} = 170$ N/m and $K_{\mathrm{d}} = 122$ Ns/m. As it can be noticed in the plot, once the disturbance is removed, the robot moves towards the goal, but it is not able to follow it once it has been reached. In fact, the robot keeps drifting, moving away from the initial position.

It is possible to conclude thus that the approach with a feed-forward force results in a slow response, with a steady state error with respect to the goal. The common practice of setting the PD parameters considering the system behaving as a second order system (*i.e.,* spring/mass/damper) is not valid due to the under-actuation, showing the limitation of the approach. In addition, even hand-tuning the parameters does not allow the user to obtain the desired behaviour.

The result of the proposed approach, instead, is reported with light blue line. Once the disturbance has been removed, the reference generator computes a velocity trajectory that brings the robot towards the goal in the desired time T_{f} (shaded area) and without steady state error.

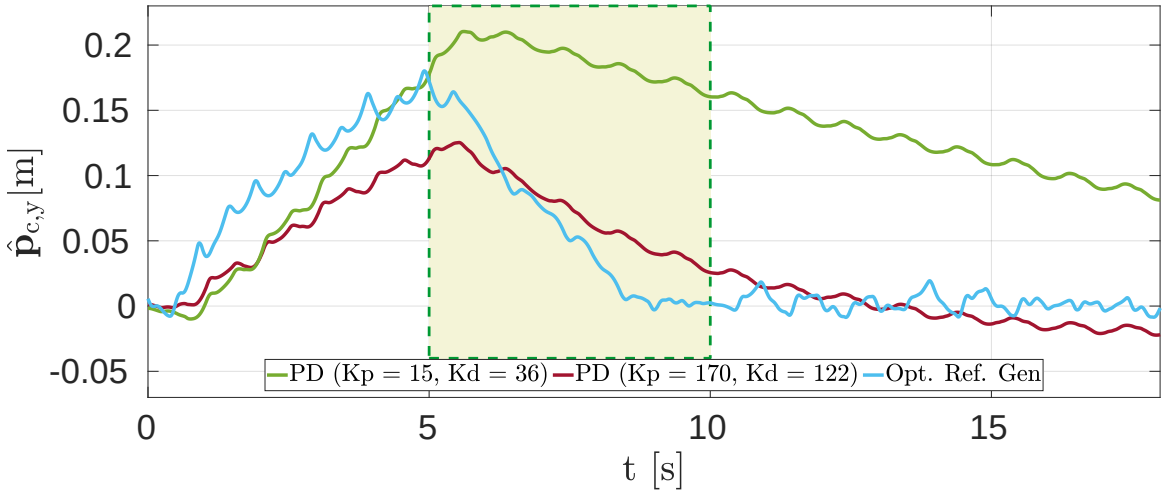

Figure 4.8: Simulation, scenario (c): comparison of the actual Y CoM positions between the PD force approach (green and purple lines) and our reference generator (light blue line). The goal is to come back to the initial position after a 15 N lateral push of 5 s. For the green line the values are chosen to impose a response time of 4.8 s considering a second order system response. For the purple line, instead, $\mathrm{K_p}$ and $\mathrm{K_d}$ are computed such that the system should have a critically damped response. In both cases the robot is not able to converge to the goal. With our reference generator (light blue line) the robot recovers its initial position after the push, without any steady-state error.

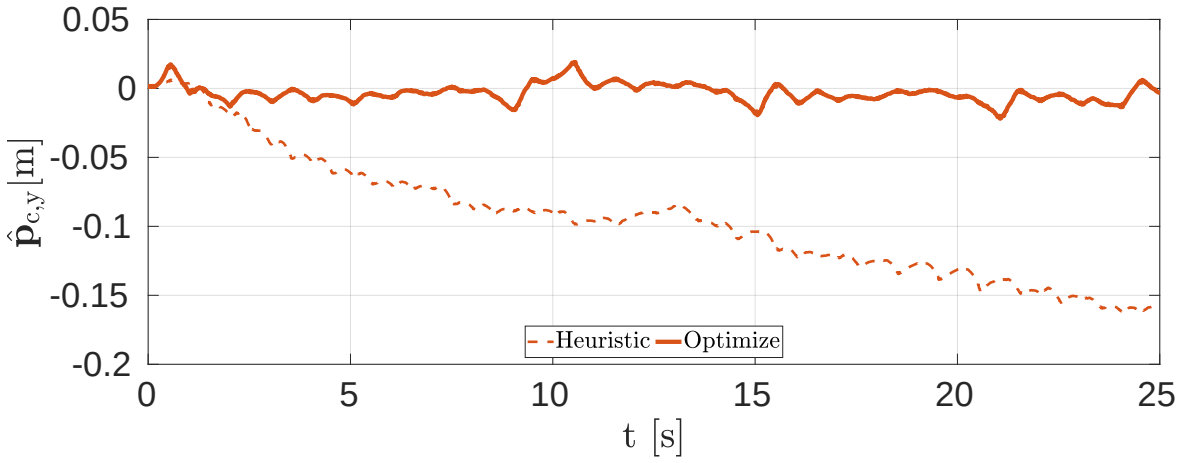

Figure 4.9: Experiment, scenario (a): Aliengo moving forward with zero lateral velocity set by the user. The dashed line represents the actual Y CoM position when the reference generator is forced to be always in *heuristic* status. The robot diverges from the goal of $\hat{\mathbf{p}}_{c,y} = 0$ and there is no part in the controller that brings it back. The continuous line shows the actual Y CoM position when the reference generator automatically changes its state according to the error $\mathbf{e}$. Thanks to the corrections of the *optimize* reference generator, Aliengo is able to stay close to the goal.

4.4.2 Experiments

In this section we present some experiments for the scenarios (a), (b), (c) carried out with the real robot platform. In these experiments, we employed the problem (4.1) without ensuring guarantees on the response time. As first experiment we performed scenario (a) on the robot, as illustrated in Fig. 4.9.

Scenario (a)

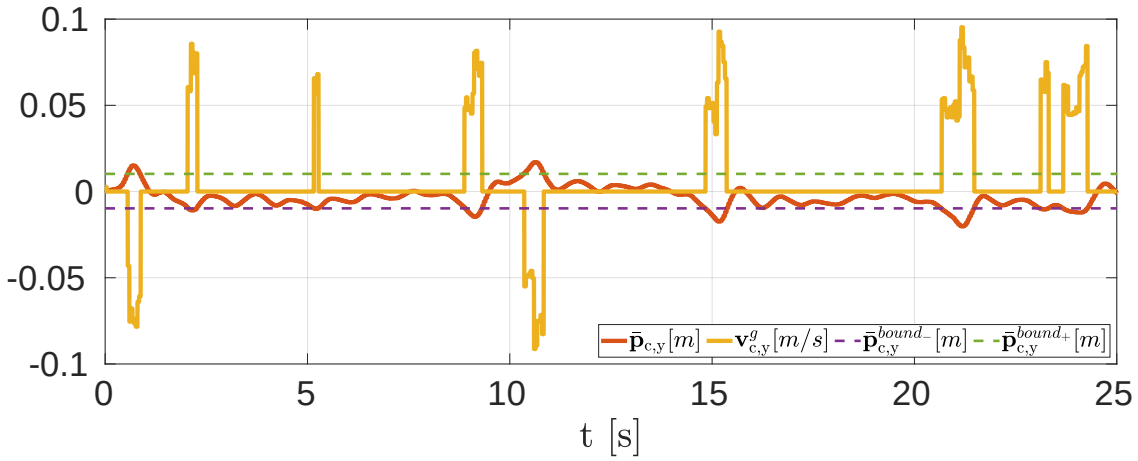

Figure 4.10: Experiment, scenario (a): Aliengo moving forward with zero user lateral velocity. The peaks in the reference velocity (yellow line) represent the moment in which the average Y position (red line) has passed the threshold (dashed lines) around the initial position and the reference generator is set to *optimize* status.

The figure demonstrates that the NMPC with only a heuristic reference generator (dashed lines) does not succeed in moving on a straight line for a statically unstable gait as trot.

Indeed, the robot suffers lateral and backward (less visible) drifts because of two reasons:

1. the trot being an unstable gait the CoM always diverges between two four-leg-stance events in opposite directions. Any little asymmetry in the robot results in a cumulative drift in one direction.

2. since Aliengo has C-shaped legs, they create nonzero moments about the pitch axis during a swing.

Enabling the optimal reference generator, instead, the robot is able to reach the goal and prevent the trajectory from drifting (see the continuous line in Figure 4.9).

Figure 4.10 shows the change in the reference lateral velocity (yellow line) done by the reference generator. When the average Y position ($\bar{\mathbf{p}}_{\mathrm{y}}$, red line) exceeds the bounds ($\bar{\mathbf{p}}_{\mathrm{y}}^{\mathrm{bound}}$, dashed

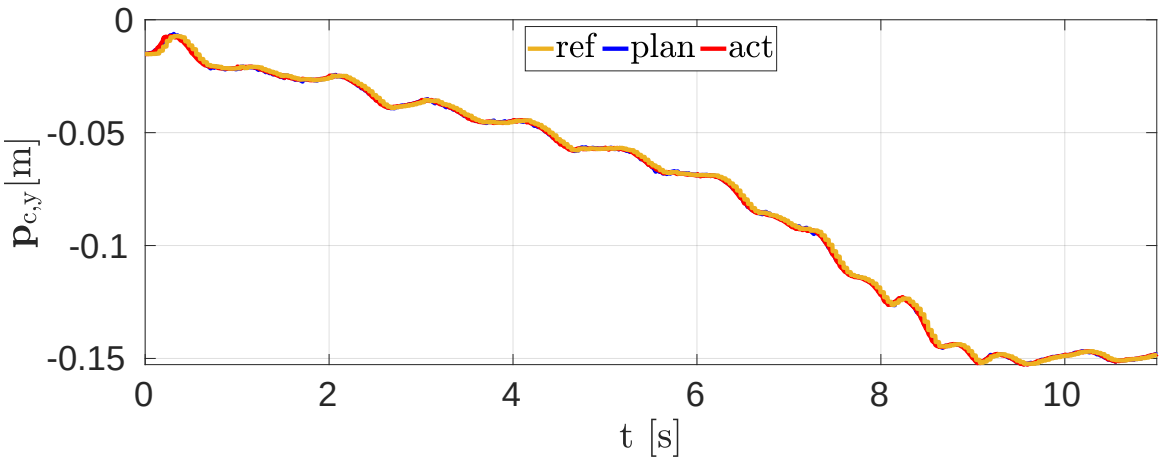

Figure 4.11: Experiment, scenario (b): Aliengo robot reaches the target position of -0.15 m and then keeps moving following the user-defined velocity.

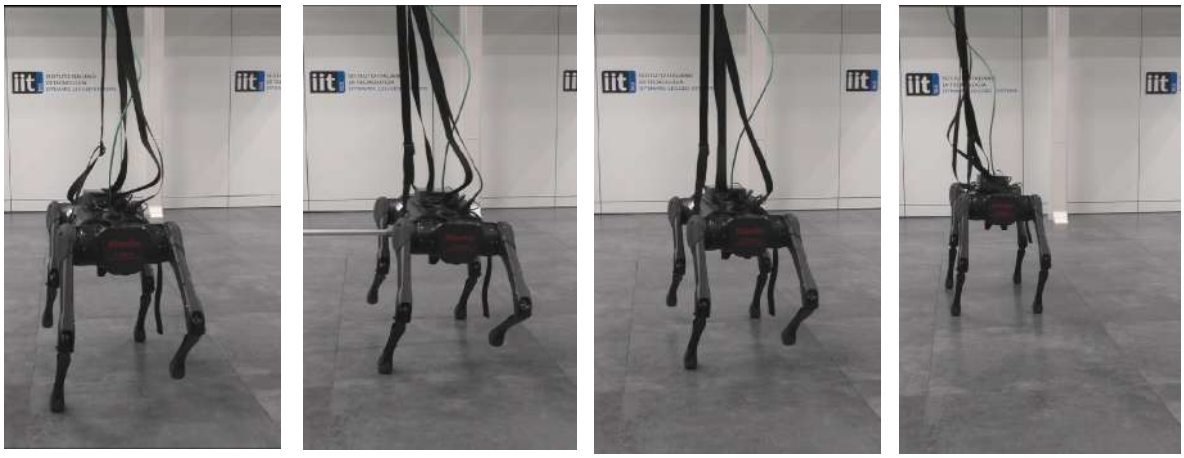

Figure 4.12: Experiments, scenario (c): sequence of screenshots. The robot moves forward (picture 1), and is suddenly pushed with a stick (picture 2). Once the push is removed (picture 3), the optimized reference generator automatically drives the robot back to its initial position. Finally, the robot follows the user-defined velocity (picture 4).

lines), the status is changed to *optimize* and the reference generator brings back the CoM close to the goal. A threshold of 1 cm around the constant goal has been chosen for the activation. Once the goal has been reached, the reference generator automatically resets to *heuristic*, and the reference velocity becomes equal to the user one (zero). The continuous changing of the status of the reference generator demonstrates the need to have an external module which corrects the reference trajectories during a trot.

Scenario (b)

Figure 4.11 shows the Y position of the robot in the scenario (b) with a fixed goal of -0.15 m. As in simulation, the robot is able to track the reference value, due to the fact that the velocity takes into account the under-actuation of the trot gait. In this case, we decided to keep the reference generator always set to *optimize* to demonstrate that it is able to work properly also when the target has been reached, compensating drifts as in scenario (a).

Scenario (c)

In the last experiment, we show how we can use our reference generator to react to external disturbances, see Figure 4.12. As already mentioned in the Section 4.1, the task is not to reject the disturbance, but to cope with it and later recover from its effect. An analysis of techniques to reject disturbances goes beyond the scope of this work.

Figure 4.13 shows the Y position of the CoM in a real hardware experiment in scenario (c) when the robot receives two manually applied pushes. The threshold on the error is set to 1 cm (dashed purple line). During the push, the robot tries to resist to the disturbance and, thanks to the high-frequency replanning of the NMPC, it is able to keep the stability and avoid falling. Once the pushing force is removed, the reference generator drives the robot back towards the initial position. As in the previous cases, the reference generator is set automatically to *optimize* when the robot is diverging from the goal.

The readers are encouraged to check the experiments corresponding to the mentioned results in the following video:
https://youtu.be/Jp0D8_AKiIY.

4.5 Discussion and Conclusions

4.5.1 Discussion

The reference generator is an important module for an NMPC approach; indeed it provides the references for the cost term and (some) required parameters. As already mentioned, this module can affect the performance of the entire framework. For example, doing some regularization with some heuristic policies allowed the controller to exploit the known dynamics of the system [Bledt et al., 2017]. Similarly [Bjelonic et al., 2022] endowed their MPC with an offline generated library of optimal reference trajectories demonstrating the importance of warm starting the optimization with feasible trajectories.

Motivated by this aspect, the goal of the work proposed in this chapter is to endow the already developed NMPC framework with *online* physically informed reference trajectories to

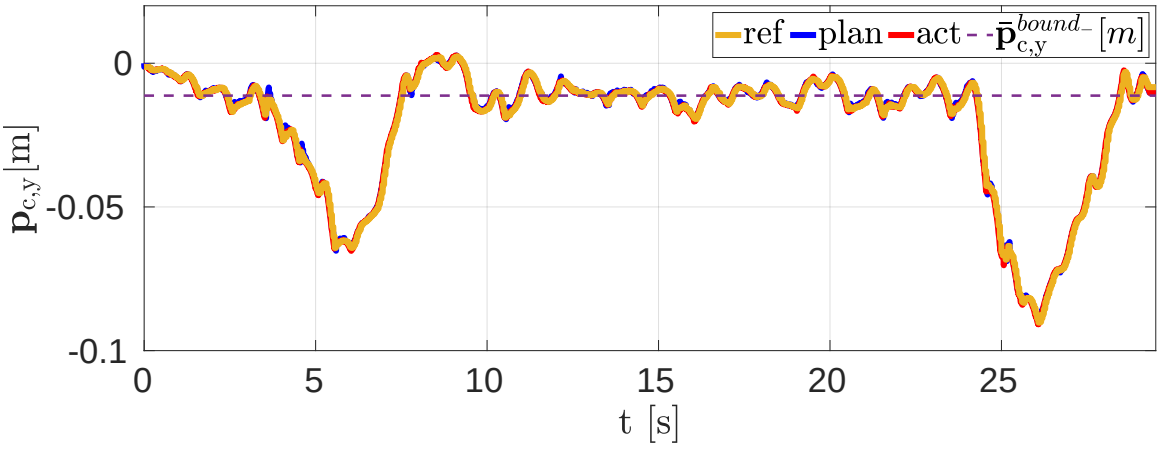

Figure 4.13: Experiment, scenario (c): CoM Y position of the robot. During the motion, the robot has been pushed twice and it automatically comes back to the initial position when the push is removed.

be tracked. Thanks to the reformulation of the problem of generating the trajectory as an OCP, the proposed reference generator deals with the under-actuation of the trot and imposes additional features, *i.e.,* reaching a fixed goal or recovering after a push. In addition, the heuristics of the robocentric stepping is improved with the output of the optimization.

Some alternatives to the presented approach are described in the Introduction. They are the PD wrench term, adding a position term in the cost function, and the user changing the reference velocities. Figure 4.8 shows a comparison in simulation with the NMPC + a PD feed-forward term . It would be interesting to extend the comparison to other state-of-the-art locomotion frameworks; the difficulty is due to the fact that the code is generally unavailable and some implementation details are not provided nor dataset of representative experiments are provided.

Finally, it is worth mentioning that this approach could also be used directly as a planner, since it computes feasible CoM and GRFs trajectories, according to the LIP model. The advantage is that the formulation is very lightweight, and thus high replanning frequencies can be achieved. Anyhow, the assumption of this model, *i.e.,* no vertical and angular dynamics, restricts the possible applications.

Limitations

Although the formulation has been tested in three different scenarios, the heading, *i.e.,* yaw, orientation cannot be considered. An extension of the approach needs to be investigated. In addition, the formulation which imposes a response horizon Eq. (4.2) has been tested only in simulation; real experiments with that formulation will be included in the future works. More

tests need to be performed to see if the LIP based approach (that is based on a flat terrain assumption) scales up effectively also to rough terrains.

4.5.2 Conclusion

This chapter presents a novel optimization-based reference generator for quadruped locomotion, which deals with the under-actuation of statically unstable gaits and with external disturbances. It exploits the LIP model to compute feasible reference trajectories that allow the robot to accomplish a tracking task. A QP mapping is used to determine the GRFs which corresponds to the ZMP location computed by the LIP model. Velocities and GRFs are used as *informative* tracking references by the 25 Hz lower-stage NMPC planner. This results in the absence of conflicting tasks in the cost function, which simplifies the tuning of the cost weights of the NMPC. The approach is validated by performing simulations and experiments with the 22 kg quadruped robot Aliengo in three different scenarios: (a) straight motion, (b) fixed lateral goal and (c) recovery after a push. In addition, I presented and validated in simulations a formulation with slack variables that can guarantee to reach the goal in a specified time, without the need to further tune any parameters. For the last scenario, we demonstrated that the simple solution of adding a Cartesian PD in parallel to the NMPC is not enough to return in a predefined time to the required position.
The work presented in this chapter has been accepted for publication in [Bratta et al., 2022].

Chapter 5

Contact Planner

Figure 5.1: Picture of the torque-controlled quadruped robot Solo12 [Grimminger et al., 2020].

5.1 Motivation

MPC approaches can automatically generate optimal trajectories for a legged robot and traverse complex environments, *e.g.,* [Di Carlo et al., 2018; Kim et al., 2019; Mastalli et al., 2022]. The previous chapters presented an NMPC scheme and an optimization-based reference generator, able to compute optimal values of CoM and GRFs. Simulations and experiments with HyQ [Semini et al., 2011] and Aliengo validate the proposed approach. The optimization-based reference generator improves the heuristic footholds to be coherent with the physically informed reference trajectories, but the gait sequence, *i.e.,* which leg needs to be moved, is still decided a priori.

Online motion planning for legged robots remains a challenging problem; indeed, including the gait status inside the formulation of an OCP causes an increase in the computational cost.

Few algorithms have been proposed in the literature, *e.g.,* [Deits and Tedrake, 2014b; Winkler et al., 2018; Amatucci et al., 2022], see Section 2.1.4. They demonstrated to be effective either in offline techniques or simple scenarios. An interesting idea is the *DeepQ Stepper* [Meduri et al., 2021]; it exploits the reinforcement learning to obtain a reactive contact planner for bipedal locomotion, which was limited to single contacts and cyclic gaits.

This chapter addresses these limitations and proposes an online, MPC friendly multi-contact planner - *ContactNet*, that can automatically generate arbitrary gaits, select footholds in complex terrains, *e.g.,* stepping stones, and recover from external perturbations. The solving time remains unaffected by terrain complexity. This is in stark contrast with other contact planning algorithms, such as the MIP [Deits and Tedrake, 2014b] or the MCTS [Amatucci et al., 2022].

ContactNet is based on a multi-output regression network [Schmid et al., 2022] that ranks a discrete set of allowed foothold locations. This information is then used to generate a contact plan. The ContactNet is trained offline using data generated with a novel cost function (see Section 5.3.2) that chooses footholds which maximize robustness, and stability and minimize trajectory generation cost. After training, the ContactNet provides the foothold plan to the NMPC to generate online a desired behavior.

To evaluate this approach, ContactNet generates *acyclic walk* and *acyclic trot* behaviors on the Solo12 robot [Grimminger et al., 2020] in simulation (Figure 5.1). ContactNet can automatically navigate terrains with holes, even though data collection for training is performed on flat terrain without holes, without any visual feedback during the training (unlike [Villarreal et al., 2019]).

Outline

This chapter is organized as follows: Section 5.2 describes the control architecture; Section 5.3 gives an overview of the Contact Planner, highlighting the main features of the ContactNet and how it is integrated into the locomotion pipeline. Section 5.4 presents the results of simulation with the Solo12 robot with the ContactNet for both walk and trot in different scenarios: acyclic gaits (Section 5.4.2), stepping stones (Section 5.4.3) random generated terrain (Section 5.4.4), and push recovery (Section 5.4.5). Finally, discussion and conclusions are drawn in Section 5.5.

5.2 Overall control architecture

Figure 5.2 shows the block scheme of the locomotion framework, and how the ContactNet has been integrated with the other elements already introduced.

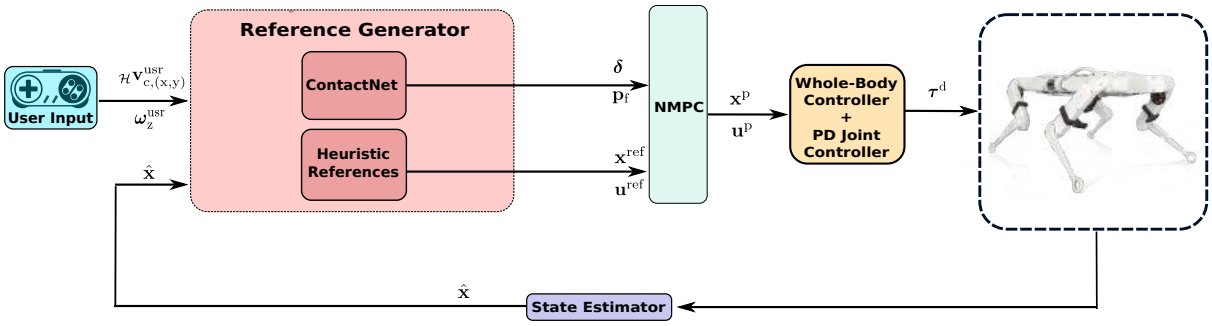

Figure 5.2: Block scheme of the entire locomotion framework. Given the user-defined velocities and the actual quantities, the ContactNet provides the sequence of footholds, *i.e.,* swing legs and touchdown points, at a frequency of 3.125 Hz (after each touchdown). Integration of the user-defined quantities compute the reference trajectories, see Section 3.5. Given the sequence as parameter, the NMPC computes CoM trajectory and GRFs tracked by a 1 kHz Centroidal WBC and an impedance controller.

The user decides the linear velocities ${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ that the robot should follow. Given the X and Y components of the footholds in the CoM frame $\mathcal{C}$ ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{f}} \in \mathbb{R}^8$, Z component of the CoM $\hat{\mathbf{p}}_{\mathrm{c,z}} \in \mathbb{R}$, actual CoM velocity $\hat{\mathbf{v}}_{\mathrm{c}} \in \mathbb{R}^3$ and reference velocities $\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}}$, the ContactNet returns the ranking of the discretized actions, according to the cost function described in Section 5.3.2; in particular, the best *feasible* footholds are chosen, *i.e.,* coherent with the terrain, is chosen. Reference velocities are integrated to compute the CoM position at the end of the *step horizon* N_{s}.[1]

Those positions and reference velocities are used together with the new foothold to re-evaluate the neural network after N_{s} and 2 N_{s}. Consequently, a sequence of 3 footholds is used to generate a contact plan $\mathbf{p}_{\mathrm{f}}$ and $\boldsymbol{\delta}$ for a horizon of $N = 3\ N_{\mathrm{s}}$. The swing times are preset depending on the chosen gait (discussed in detail below). This contact plan along with the reference CoM trajectories are provided to the trajectory optimizer to generate an optimal movement using our NMPC optimization.

The CoM trajectories are then tracked by a 1 kHz WBC [Grimminger et al., 2020][2], combined with a PD controller in Cartesian space for the swing trajectories. The swing trajectory is defined in the swing frame [Raiola et al., 2020]; a semi-ellipse represents the X component and a fifth-order polynomial the Z component.

At the end of each step horizon (frequency 3.125 Hz), another iteration of ContactNet and trajectory planner is executed in an MPC fashion. The motivation and consequence of this

[1] At the start of this work, I decided to use the heuristic reference generator. The optimization-based one of Chapter 4 will be integrated in the future.

[2] This work has been performed during my internship at the Machines in Motion lab of the New York University, so I used a WBC presented in one of their previous works. There are no substantial differences with the WBC used in Chapter 3 and 4 developed by [Fahmi et al., 2019]

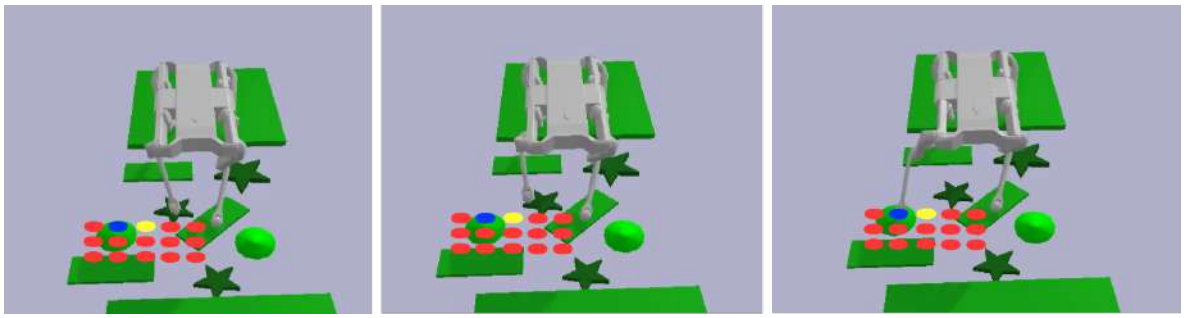

Figure 5.3: Example of the evaluation of the ContactNet on a terrain composed of stepping stones for one leg. Red disks represent some of the actions evaluated by the ContactNet. The network computes the ranking order, according to which the yellow disk is the one which minimizes the cost function (5.1). Knowing the terrain map of the terrain yellow is considered *unsafe* and discarded. The blue disk corresponds to the first action deemed *safe*.

choice for the re-planning frequency are discussed in Section 5.5.

5.3 ContactNet

This section describes the cost function and data generation approach used to rank footholds offline. After that, the details about the ContactNet are provided.

5.3.1 Footholds

The allowed stepping region of each leg is discretized into a fixed set of possible footholds. These footholds are defined at a fixed distances from the current hip location of the corresponding foot, similarly to the robocentric stepping described in Section 3.5.2. Subsequently, as the robot moves, the allowed foothold locations also change. An illustration of the foothold discretization is shown in Figure 5.3. Discretizing allowed footholds is reasonable because high precision is not generally necessary during contact planning and exact foothold location can eventually be further optimized during trajectory optimization. The experiments show that, despite losing the freedom of stepping anywhere in the leg workspace, very reliable behaviors can be generated by appropriately discretizing the space.

5.3.2 Cost Function

Now that we have a set of allowed footholds, the goal is to identify the best one given the terrain and state of the robot. For this, a novel cost function is used to rank all the foothold locations based on robot stability, robustness and trajectory optimization. The input is $\mathbf{u}_\mathrm{r} = [{}_\mathcal{C}\mathbf{p}_\mathrm{f}, \hat{\mathbf{p}}_{\mathrm{c},z}, \hat{\mathbf{v}}_\mathrm{c}, \mathbf{v}_\mathrm{c}^\mathrm{usr}]$, where ${}_\mathcal{C}\mathbf{p}_\mathrm{f} \in \mathbb{R}^8$ represents X and Y components of the foot location

in the CoM frame $\mathcal{C}$, $\hat{\mathbf{p}}_{\mathrm{c,z}} \in \mathbb{R}$ is the Z component of the CoM, $\hat{\mathbf{v}}_{\mathrm{c}} \in \mathbb{R}^3$ is the actual CoM velocity. Finally, the variable $\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ is the user-defined reference velocity. This vector fully describes the initial conditions and references of the robot. In addition, using foot locations in the CoM frame behaves as a normalization of the data which improves the robustness and accuracy of the network.

To evaluate a foothold, the NMPC is solved offline to generate a trajectory that moves the robot from the current configuration to the chosen one. For example, in Figure 5.3, if we consider the disk colored in blue, we generate a trajectory with a contact plan that moves the front right leg of the robot to that foothold while minimizing the trajectory optimization costs (velocity tracking, base reference etc., as discussed in Chapter 3). Using the generated trajectory, the chosen foothold is evaluated using the following cost function:

$$V = \sum_{k=0}^{N_s} V_{\mathrm{k}} + V_{\mathrm{N_s}} \tag{5.1}$$

where N_s is the *step horizon*, V_{k} is a running cost (evaluated at each point of the trajectory), and $V_{\mathrm{N_s}}$ is a terminal cost (evaluated only at the final point).

The running cost V_{k} consists of three terms:

$$V_{\mathrm{k}} = \gamma_{opt} V_{\mathrm{k,opt}} + \gamma_{stab} V_{\mathrm{k,stab}} + \gamma_{hip} V_{\mathrm{k,hip}}. \tag{5.2}$$

The first term corresponds to the cost of the optimization problem obtained from the NMPC, *i.e.,* tracking of references for states (CoM quantities) and control inputs (GRFs). It guarantees that a feasible trajectory that respects the dynamics and friction cone constraints exists, evaluating thus the transition feasibility presented by [Fernbach et al., 2020].
I recall that the approach with the ContactNet uses the SRBD model [Orin et al., 2013], but any other model could also be exploited. Note that $V_{\mathrm{k,opt}}$ is set to a large value if the solver does not converge to a solution (*i.e.,* a feasible motion can not be generated for the chosen foothold). The variable $V_{\mathrm{k,stab}}$, evaluates the stability of the motion. It computes the distance of the CoM to the support polygon. For instance, in a walk, this encourages foothold locations in which the robot is statically stable (CoM inside the support polygon, $V_{\mathrm{k,dist}} = 0$); for a trot, this maximizes the controllability of the robot. The last cost term $V_{\mathrm{k,hip}}$ enforces kinematic limits - it assigns a non-zero value when a leg in stance violates these limits. Even though the simplified model does not include joint values, it is possible to detect a violation of the kinematic limits if the distance between the foot and the hip exceeds a certain threshold. However, the exact position of the hip is known only at the initial condition of the trajectory. Since the horizon is short, we assume that the positional offset between the hip and the CoM remains constant for the entire trajectory. Further, a conservative threshold value is chosen to encourage the motion of one leg when it is close to the kinematic limits, similarly to [Bjelonic et al., 2021].

The term $V_{\mathrm{N_s}}$ in the cost function V takes into account the future actions of the robot (value function). It is defined as follows:

$$V_{\mathrm{N_s}} = \gamma_{cent} V_{\mathrm{cent}} - \gamma_{area} V_{\mathrm{area}}. \tag{5.3}$$

With V_{cent}, we introduce a penalization on the distance between the CoM and the center of the support polygon. Minimizing this quantity increases the number of subsequent stable actions. Finally, the quantity V_{area}, which is the area of the support polygon, is maximized. This helps improve the robustness of the contact configuration at the end of the trajectory by favoring large support areas. The numbers $\gamma_i \in \mathbb{R}$ scale the different cost terms.

5.3.3 ContactNet

Using the cost function V discussed previously, it is possible to automatically generate acyclic multi-contact plans for locomotion by simply taking the contact transition with the lowest cost. However, it is not feasible to evaluate online all the possible footholds by computing optimized trajectories. Indeed, one would need to solve $N_{\mathrm{a}} = 100+$ optimization problems. Consequently, to overcome this limitation, the presented approach trains offline a neural network that learns to rank the possible footholds using the cost function (5.1). This section discusses the data generation scheme and training procedure used to train the neural network ContactNet.

Data Generation

To train the ContactNet, a dataset is generated and contains many possible stepping situations that can appear on the robot. The robot is initialized in a randomly generated state (different joint position and velocity) and with a random reference CoM velocities in the range (-0.1,0.1 m/s) for both X and Y directions. An *episode* is executed and the robot is allowed to walk for a fixed duration of time. At each touchdown, the cost function (5.1) is evaluated, and the best foothold, *i.e.,* the one with the smallest V, is selected. Subsequently, a trajectory is generated with this contact plan for N_{s} and is tracked on the robot in simulation. Such episodes are run several times to generate a large dataset containing the optimal values of $\mathbf{u}_{\mathrm{r}}$ and $\mathbf{V}$. A new episode is restarted after a chosen number of steps (30 in this case) or when the robot falls down. Since it is not possible to know which initial step led to the final fall, in case of falling, the last 3 entries, *i.e.,* the last 3 steps are removed from the dataset. As a general comment, it is way more convenient to erase "good" entries than add "wrong" ones.

Obviously, generating the dataset is not real-time. Nevertheless, it takes little time - about 6-7 hours with a standard performance computer (3.7 GHz Intel Xeon processor) - for a machine learning approach.

ContactNet Training

The goal now is to learn a function $f_\theta : \mathbb{R}^{14} \to \mathbb{R}^{N_\mathrm{a}}$, which maps the current state of the robot $\mathbf{u}_\mathrm{r}$ to the list of the ranked footholds. The main advantage of a learning approach is that it guarantees low computational effort at runtime, allowing one to integrate it into the MPC framework.

In order to learn the ranking function for all the possible actions, a Multi-Output Regression Network [Schmid et al., 2022] is used. To do so, the vectors $\mathbf{V}$ in the dataset are sorted in decreasing order to create a vector $\mathbf{Y} \in \mathbb{R}^{N_\mathrm{a}}$; it assigns to each value of $\mathbf{V}$ its position in the sorted vector;[3] the smaller the cost for the action a, the higher its value in $\mathbf{Y}$. For robustness purposes, each entry is normalized by the total number of footholds (classes).

For the sake of clarity, we provide a small example:

- $\mathbf{V} = [0.8, 0.3, 1]$
- $\mathrm{sort}(\mathbf{V}) = [1, 0.8, 0.3]$
- sorted indexes for the entries in $\mathbf{V}$ are $[1, 2, 0]$
- $\mathbf{Y} = [1/3, 2/3, 0/3]$.

As a training loss, I use the mean squared error between the prediction $\hat{\mathbf{Y}} = f_\theta(\mathbf{u}_\mathrm{r})$ and $\mathbf{Y}$.

$$\min_\theta \quad \frac{1}{N_\mathrm{a}} \sum_{k=0}^{N_\mathrm{a}-1} (\mathbf{Y}_\mathrm{i} - f_\theta(\mathbf{u}_\mathrm{r})_\mathrm{i}) \tag{5.4}$$

ContactNet Evaluation

After training the ContactNet, one can quickly obtain the optimal foothold by sorting in decreasing order $\hat{\mathbf{Y}}$. In certain situations where terrain restrictions exist, one not only has to sort the $\hat{\mathbf{Y}}$ but also select a *safe* one, *i.e.,* the foothold which does not correspond to a hole on the ground, discarding the others. The knowledge of the terrain map is used to identify the subset of feasible footholds from the allowed ones. It is important to stress that visual information is exploited only at runtime to filter $\hat{\mathbf{Y}}$; indeed, the dataset is created only on flat ground.

An example of such a situation is shown in Figure 5.3, where the robot is expected to walk across stepping stones (green objects). The red circles correspond to all the allowed footholds of the LF leg that are ahead of the hip position[4]. The yellow disk represents the optimal foothold

[3] *Remark*: the count of the indices starts from zero.

[4] note that the allowed locations behind the actual foothold are not shown for image clarity but are also evaluated in this situation

location in $\hat{\mathbf{Y}}$ (that is the one with the minimum cost according to the ContactNet) but it must be discarded since there is no terrain below it. Consequently, the foothold corresponding to the blue disk must be selected, which is the best location that also respects the restrictions due to the terrain. The NMPC is then used to move the robot to the chosen foothold.

Remark: Ranking all of the discretized actions is crucial to navigate complicated terrain situations online. Note that there is no increase in contact planning time with terrain complexity because the ContactNet provides the ranks of the footholds all the time. This is in contrast with other optimization [Fernbach et al., 2020; Deits and Tedrake, 2014b] and sampling [Amatucci et al., 2022] based planners.

5.4 Results

This section presents the results obtained by performing simulations with Solo12, a 2.2 kg open-source torque-controlled modular quadruped robot. The entire framework runs on a Dell precision 5820 tower machine with a 3.7 GHz Intel Xeon processor. The simulations are performed using the PyBullet library [Coumans, 2013].
The video of the achieved results can be seen here:
https://www.dropbox.com/s/6y5k34wuf2eyyu3/ICRA23_2586_VI_i.mp4?dl=0

5.4.1 Neural Network Architecture

For all the experiments, the ContactNet is composed of 4 fully connected layers with 128 neurons each. All layers except the last one are activated with a ReLU function [Nair and Hinton, 2010]. As hyper-parameters [Smith, 2018] for training, the number of epochs is equal to 1000 with a batch size of 100 and a learning rate of 0.001.

The term *batch* refers to a sub-group of the dataset which is used to update the parameters of the function $f_\theta(\mathbf{u}_\mathrm{r})$ (*iteration*). This approach, called *mini-batch mode*, divides an epoch in several iterations[5] and demonstrated better performance than using the entries of the dataset at the same time. In each epoch different batches are created, such that the final result is not biased. The batch size is an important element of the training process [Lin, 2022]; in the case of the ContactNet the value of 100 has been chosen by trial and error.

The *learning rate* [Bishop, 1995] is probably the most important hyper-parameter[6] and determines how quickly the model is adapted to the problem, since it scales the update of the parameters of each iteration. The choice of the learning rate is the result of trade-off between velocity and accuracy: a small learning rate increases the training time, while with large learning rates the network could converge to a suboptimal solution.

[5] number of iterations = number of elements in the dataset / batch size

[6] "*If you have time to tune only one hyper-parameter, tune the learning rate*" [Goodfellow et al., 2016]

The input state $\mathbf{u}_\mathrm{r}$ is normalized to be in the range (-1,1) to improve the accuracy of the network. To evaluate the network's performance 70 % of the entries of the dataset are used as a training set and the remaining part as a test set. The top-5 metric determines the statistics of the network; in particular, let us consider a correct prediction if the first element of $\hat{\mathbf{Y}}$, *i.e.,* what the neural network outputs as the best action, is one of the first five elements according to the corresponding $\mathbf{V}$ stored in the dataset. This metric has a particular importance for the ContactNet since the best action will not always be feasible due to the requirements of the terrain.

5.4.2 Acyclic gaits

This subsection discusses the various parameters defined to generate the two gaits - walk and trot - discussed in the chapter.

Walk

In this motion, the robot is only allowed to move one leg at a time. Considering the same sampling time $T_\mathrm{s} = 40$ ms, the step horizon N_s is equal to 320 ms (8 Nodes) and it is composed of 120 ms of full stance phase (3 nodes), 160 ms (4 nodes) of swing phase, and the last node of full stance phase. The prediction horizon N used by the NMPC is composed of three step horizons, 960 ms, as discussed in Section 5.2.

The allowed stepping region for each leg is a 20 $\times$ 20 cm grid based on the kinematic limits of Solo12. This space is discretized into 25 footholds which are 5 cm apart, see Figure 5.3. Subsequently, the robot is allowed to choose from a total of $N_\mathrm{a} = 100$ possible footholds - 4×5^2 since we do not prescribe which leg needs to swing, only that one leg swings at a time.

Using the procedure discussed in Section 5.3.2, 1500 episodes generate the entries for the dataset. The resulting data had 43410 instances of $\mathbf{u}_\mathrm{r}/\mathbf{V}$. Using the generated dataset, the ContactNet is trained to predict the optimal footholds. The approach obtained an accuracy of 93.48/90.81 % in the training/test set according to the top-5 metric.

Trot

In the trot gait, two diagonal feet leaves the ground at the same time, *i.e.,* LF/RH or RF/LH. Subsequently, the possible action allows the robot to move either of these diagonal pairs at a particular instance. The total stepping region for each leg is a square size 10 $\times$ 10 cm. The foothold discretization resolution is still 5cm, 9 choices per leg. Consequently, at the start of a stepping horizon, there are a total of 162 foothold choices since two legs leave the ground at each step . All the other parameters are the same as the walk. The smaller grid for the trot is

justified by the quadratic relationship between actions and options per leg. In addition, the trot intrinsically moves each leg at a higher frequency, so a smaller step length is generally required.

Once again, 1500 episodes generate the dataset for this gait and train the ContactNet, obtaining 45000 instances. The neural network achieves an accuracy of 99.48/97.7 % in the training/test set.

As already explained, the ContactNet generates locomotion for user-defined velocities. The uploaded video shows the stability and accuracy of the ContactNet simulating a long horizon motion with change in the reference velocities in a terrain with holes.

5.4.3 Stepping stones scenario

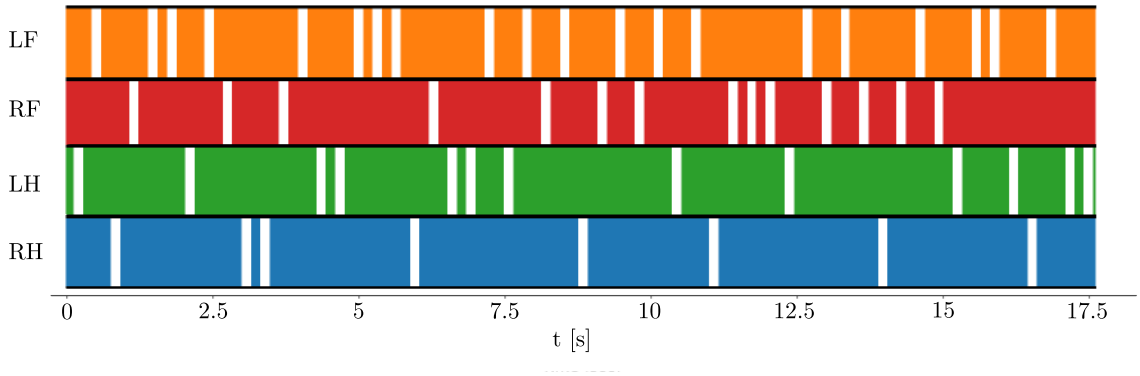

Figure 5.4: Simulation: gait schedule of a walk motion on a stepping stones scenario. White parts indicates moment in which that leg is in swing. The ContactNet finds a completely acyclic gait.

To demonstrate the effectiveness of the contact planner in an MPC framework a complex terrain is designed, composed of 8 sparse stepping stones of different shapes: 3 stars, 2 circles, and 3 rectangles, Figure 5.3. Two squares are positioned as starting and final points. The task is to traverse the terrain with a user-defined forward velocity of 0.05 m/s using the ContactNet trained for the walk gait. The proposed approach successfully navigates the terrain.

Figure 5.4 shows the resulting gait sequence for the entire motion. Each color indicates a different leg, and the full block denotes a stance phase. The white spaces indicate that the particular leg is in the swing phase at that time. Figure 5.4 shows that the motion is completely acyclic. For example, when the front legs are on the last square the CoM is closer to the hind legs automatically making the swing of the front legs preferable to prevent the CoM from going outside the support triangle. Hind legs are moved only when stability is guaranteed and a stepping stone is inside the set of actions.

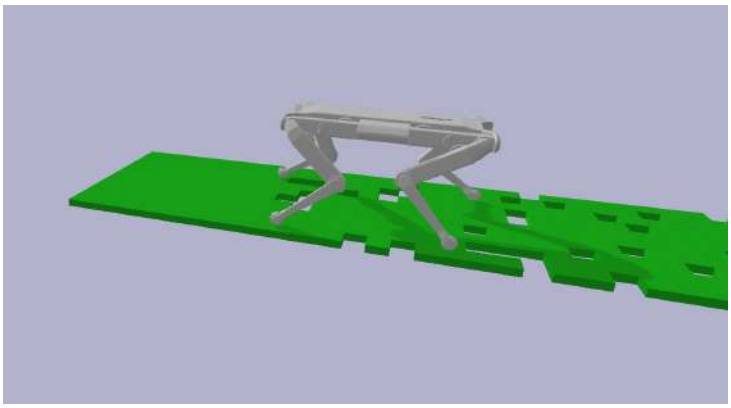

Figure 5.5: Simulation: Solo12 robot traversing generated terrain with randomly removed squares of 5x5 cm dimensions.

The result of the simulation is shown in the uploaded video. The ContactNet in average has chosen the 6th element of $\hat{\mathbf{Y}}$, with a maximum of 30 discarded elements for the swing of the LH leg at time around 10.0 s. [7] The average computation time for a complete iteration of the ContactNet, *i.e.,* computation of 3 subsequent actions, is 1.6 ms, which is much faster than the trajectory optimizer.

5.4.4 Random generated terrain

In this section, the ContactNet is used to navigate unstructured environments with various terrain constraints. The terrain is a rectangle 1.5x0.5 m obtained by placing 300 squares 5x5 cm. Starting from a number of $n = 50$ till $n = 110$ (17 % - 37 %), n blocks are removed.

I evaluate the success rate for both the walk and trot setups (see Figure 5.5) with a reference velocity of 0.05 m/s in the X direction. For each n, 100 different trials are executed to perform a statistical analysis. A trial is considered successful if all the four legs of the robot overcome the last block. In addition, to validate the robustness, the navigation task of both gaits is repeated with Gaussian noise applied to joint velocity measurement. This is performed to validate the robustness of the approach in conditions closer to the real robot.
The results are shown in Figure 5.6 and Figure 5.7.

The ContactNet has a high rate of success for both gaits, while, as expected, the walk guarantees better performance due to its intrinsic stability. The addition of noise does not cause a significant reduction in performance. This suggests that the framework would reliably work on a real robot. Note that the ContactNet remains robust to noise even though it was not trained for it, as is commonly done using domain randomizing techniques [Tobin et al.,

[7] at each iteration of the MPC the ContactNet computes a sequence of three actions. For the plot we consider only the first because it is the one that is executed.

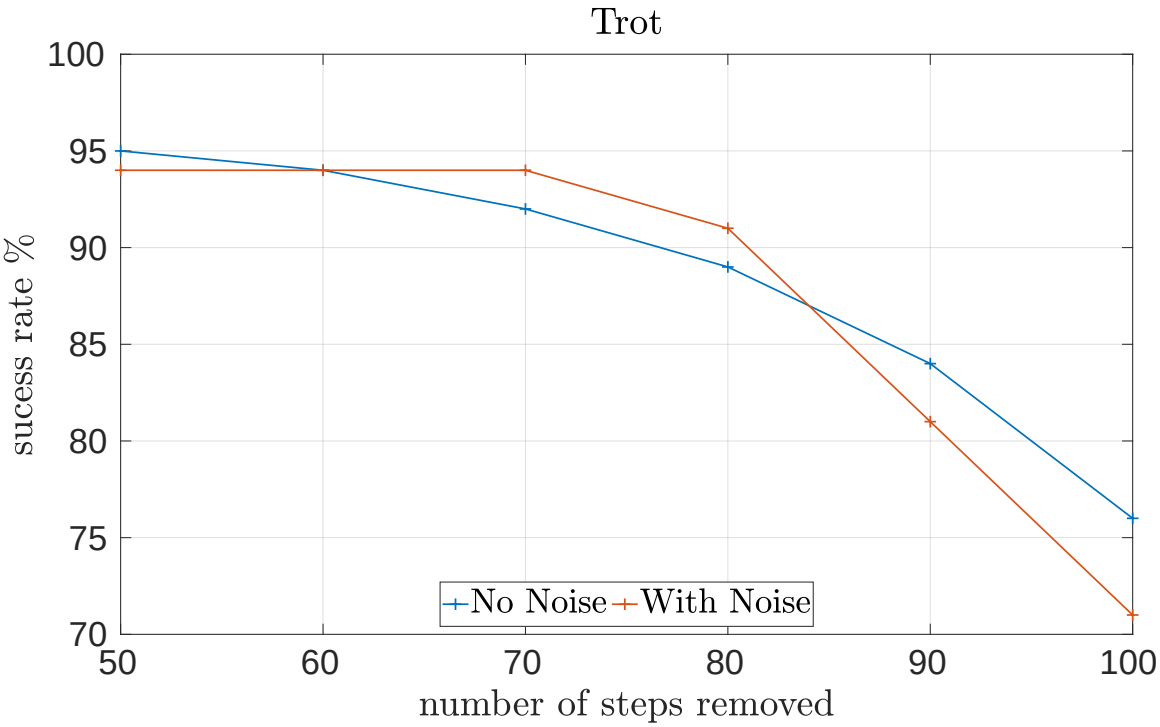

Figure 5.6: Successful rate of trot motion over the number of blocks removed with $\mathbf{v}^{\text{usr}} = [0.05, 0]\text{m/s}$. For each number of blocks we execute 100 trajectories with and without noise (Gaussian noise with zero mean and 0.01 variance).

2017]. Furthermore, no assumptions are made about removing blocks to guarantee that a real feasible path exists.

Another important element that deserves to be analyzed is the computational cost of the ContactNet. For the analysis we consider the time duration to perform 3 evaluations of the network (needed to generate the parameters to the NMPC block), along with the time of safe foothold selection based on the terrain from the vector $\hat{\mathbf{Y}}$.

Table 5.1 reports the mean value of total computational time for each value of n. It is worth highlighting that the time is low, around 1 ms, and does not change with the complexity of the terrain. This contrasts with other approaches, such as MIP and MCTS, which suffer from the dimension of the solution space. Indeed, the large number of constraints which must be included in the MIP will cause a significant increase in computational effort and reduce the possibility of converging to a feasible solution. In addition, the non-convexity of the star stones is hard to be considered in an analytic constraint. Similarly, given the high number of possibilities, the efficiency of the expansion phase in MCTS will quickly degrade too. The ContactNet, however, only has to query the terrain map until the first safe action is found. Equation (5.1) ensures that the chosen action is the *best* admissible one.

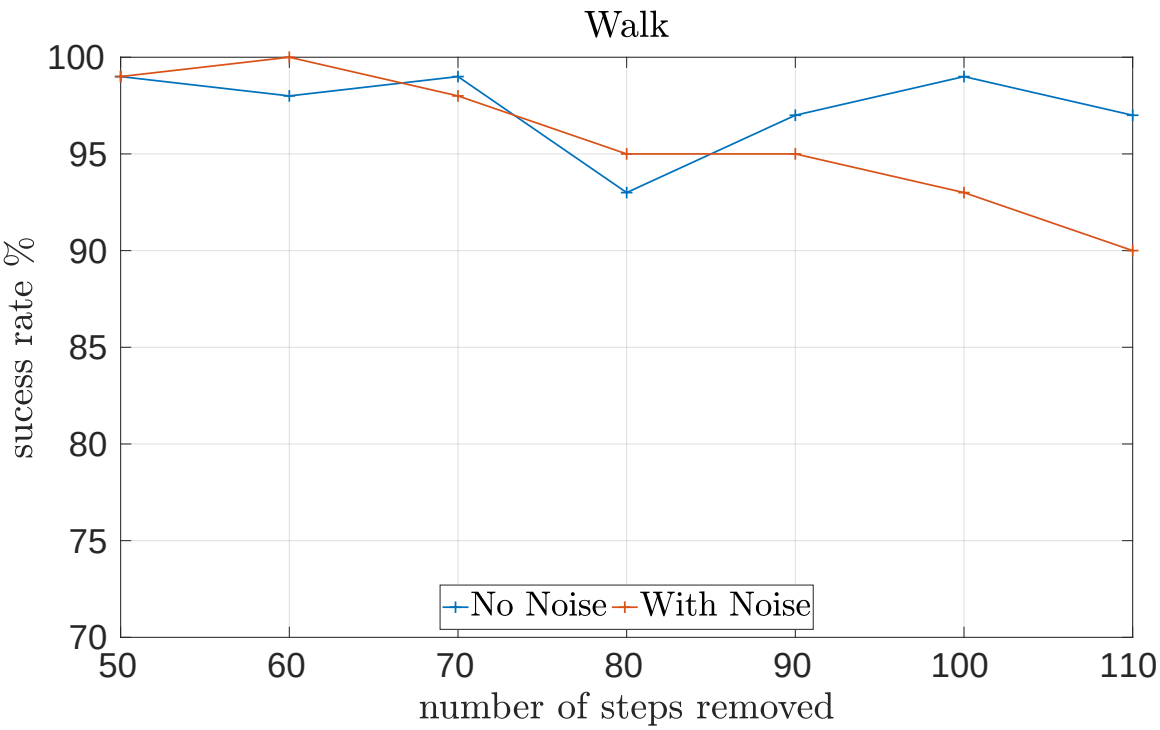

Figure 5.7: Success rate of walk motion over the number of blocks removed with $\mathbf{v}^{\text{usr}} = [0.05, 0]$ m/s. For each number of blocks we execute 100 trajectories with and without noise (Gaussian noise with zero mean and 0.01 variance).

Table 5.1: Computational time of the ContactNet

Number of steps removed	Walk [ms]	Trot [ms]
50	1.272	1.507
60	0.938	0.950
70	0.944	0.803
80	1.152	0.785
90	1.0584	0.874
100	0.9167	0.7899

5.4.5 Push Recovery

As a final result, we tested the robustness of the ContactNet pushing the robot for 1 s with external forces in the range of $\pm$5N in both $X - Y$ directions, while tracking a forward velocity. In addition some blocks 10x10 cm are randomly removed. While being pushed, the contact

planner adjusts footholds to counteract the external disturbance and avoid holes. Once the push is removed, the robot automatically recovers a stable configuration, for example by moving a leg which resulted to being close to the kinematic limits.

5.5 Discussion and Conclusion

5.5.1 Discussion

Obtaining footholds, *i.e.,* foot placement and leg sequence is one of the main challenges for a trajectory planner framework. Optimization techniques [Grandia et al., 2022] and learning approaches [Kalakrishnan et al., 2011; Villarreal et al., 2019] can provide foot locations with user-defined gait patterns, *e.g.,* the sequence LH, LF, RH, RF which has been shown to maximize static stability [McGhee and Frank, 1968]. While several algorithms perform foothold selection online, the additional difficulties of a MIP increase the computational time.

This chapter deals with those issues and presents an online multi-contact planning framework that can be easily integrated with existing trajectory optimization approaches through a multi-output regression network. The *ContactNet* ranks the discretized possible footholds, taking into account the optimization of the NMPC.

The main novelty is that it maintains a constant solve time while navigating complex terrains; indeed, by exploiting the knowledge of the heightmap, the ContactNet discards some options to provide the *best feasible* foot placement. As another advantage, transcribing the shape of the stepping stone in a mathematical formulation is not required; it avoids, thus, nonlinear constraints in the presence of the stars of Figure 5.5.

The ContactNet exploits all the four legs during the choice of a foothold. For the sake of clarity, I provide an example: let us imagine that the best option is to move the LF to a certain point. If this is not feasible, an algorithm of foothold correction such as [Villarreal et al., 2019] will avoid it by correcting the touchdown of the LF feet.
On the other hand, the ContactNet would take into account also moving the other legs, choosing for example, a touchdown point for the RF.

This chapter focuses on flat terrain with stepping stones; in particular, the ContactNet can avoid the holes, even though the training has been performed only with data on flat terrain. An extension of the approach would consider uneven terrain by discretizing the 3D stepping region and retraining the network. There is no need to make changes to the optimization problem since the results of Chapter 3 demonstrate that the NMPC is also efficient in those scenarios. Anyhow, uneven terrain requires additional components, such as a collision-free swing trajectory which goes beyond the scope of presenting the contact planning problem.

Currently, the ContactNet does not update the foothold during the swing phase. Throughout the experiments, the replanning frequency was demonstrated to be sufficiently robust to uncertainties in the environment. However, if the need arises for faster updates, the NMPC can compute the optimal trajectories with arbitrary initial contact configurations. After which, the cost function (5.1) can also be used to rank footholds during the swing phase.
Subsequently, the foothold could be adapted online by continuously predicting with the neural network during the swing.

Limitations

ContactNet has only been tested in simulation due to the lack of time before the submission of this dissertation. However, the analysis with external disturbances and sensor noise gave evidence that the results should transfer well to the real robot.

Moreover, two neural networks are needed for the walk and the trot. In future works, I aim to merge them into a single network, obtaining automatic transition between gaits.

5.5.2 Conclusion

In conclusion, this chapter proposed a multi-contact planner, *ContactNet*, capable of generating acyclic contact sequences in a few milliseconds, even in the presence of complex terrains. The ContactNet provides a sequence of footholds to the NMPC optimizer. To the best of my knowledge, the ContactNet outperforms all the state-of-the-art MIP and MCTS approaches in terms of computational performance. Simulations with the Solo12 robot are performed with walk (only one leg moves at a time) and trot (diagonal leg pairs move together) gaits in the presence of terrain composed of stepping stones. Robustness is demonstrated by injecting measurement noise and applying external disturbances.
The work presented in this chapter is currently under review for *IEEE ICRA* 2023 [Bratta et al., 2023].

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In the '70s, researchers worldwide started to dedicate their effort to building legged robots. It became immediately evident that having legs allows the robots to walk into uneven and complex terrains. Indeed, with respect to their wheeled counterparts, legged robots can break the contact with the ground and thus overcome obstacles and steps. To do so, they must follow a trajectory that guarantees reliability, stability, and robustness.

In this dissertation, I developed and presented an NMPC framework to obtain optimal values of CoM quantities and GRFs while adapting the footholds to the environment. In addition, I sought to remove heuristic components inside the computation of the references. The approach is generic and thus has been tested on three different quadruped robots (HyQ, Aliengo, Solo12).

Chapter 3 introduces the framework and the OCP formulation. A simplified model, the SRBD model is used to predict the trajectories of the robot in a horizon of 2 s. A novel mobility cost term is used to satisfy the kinematic limits and adjust the trunk orientation without requiring heuristic references. Heuristic footholds are based on user-defined quantities and are provided as parameters to the OCP. The RTI approach guarantees low computational time (5-7 ms), and thus a high-frequency replanning (25 Hz) is achieved.

Simulations and experiments with the HyQ robot validate the approach. In particular, HyQ performs an omni-directional motion, reacting to the changes in the user-defined velocities. By integrating visual information and online correction of the touchdown points (VFA [Villarreal et al., 2019]) into the pipeline, the robot is able to step on a pallet (both static and repositioned); the mobility allows for pitching up and down while walking.

Chapter 4 deals with the heuristic references and the intrinsic under-actuation of the statically unstable gait as the trot. An optimization-based reference generator provides a physically informed trajectory to be tracked by the NMPC. The LIP model and a QP mapping compute feasible GRFs, avoiding conflict tasks in the cost function. Moreover, the optimal velocities are used to improve the heuristic of the foothold selection.

This comprehensive locomotion framework is tested with experiments on the Aliengo robot in three different scenarios: (a) straight motion, (b) fixed lateral goal, and (c) recovery after a push. In addition, a simulation validates the novel formulation that imposes a user-defined time to reach the target through slack variables.

Chapter 5 removes the assumption of a heuristic touchdown point by presenting a multi-contact online planning. *ContactNet* is a multi-output regression network which ranks a discretized set of possible footholds. A novel offline cost function considers the NMPC, the robot stability, and robustness, and it is used to train the network. This approach can handle flat terrain with holes, *i.e.,* composed of stepping stones, without suffering from high computational cost.

Several analyses are performed in simulation with the quadruped robot Solo with both walk and trot motions. Adding sensor noise and applying external disturbances demonstrates that the approach has a high probability to also successfully work on the real platform.

I believe that the contributions of this dissertation improve the state-of-the-art approaches in a popular field, such as the MPC applied to legged locomotion.

6.2 Future Works

This section provides insight into future works and applications of this dissertation:

- the NMPC introduced in Section 3.3 demonstrated good results and has been integrated with the optimization-based reference generator and the contact planner. Further improvements can be achieved by increasing the replanning frequency and augmenting the complexity of the model.
  For example, an idea could be to consider joint quantities to satisfy actuation limits or to design collision-free swing trajectories.

- the optimization-based reference generator provides feasible X-Y CoM references to accomplish a task. Additional scenarios can be considered by extending the formulation to angular velocities.
  Moreover, a better validation of the approach foresees the tests with response horizon formulation, the investigation of how this approach scales up to rough terrains, and further comparative analyses with the state of the art.

- the *ContactNet* has been tested in several scenarios, but unfortunately, only in simulation; experiments with Solo or HyQ need to be performed. In addition, the architecture of the network could be improved to consider gait transitions and terrains with steps and gasps.
- finally, I aim to use the presented approaches to have a quadruped robot being part of our social life, *e.g.,* helping humans in a disaster response scenario or inspecting an unknown environment.

Appendix A

Angular Velocity

I employ the Z-Y-X convention Diebel [2006] for the Euler angles sequence $\mathbf{\Phi} = (\phi, \theta, \psi)^\top$ to represent the orientation of the robot base where, ϕ, θ and ψ are the roll, pitch and yaw, respectively. The angular velocity in inertial and CoM frame is related to the Euler angle rates with the following relations:

$$\boldsymbol{\omega} = \mathbf{E}(\mathbf{\Phi})\,\dot{\mathbf{\Phi}} \tag{A.1}$$

$$_{\mathcal{C}}\boldsymbol{\omega} = \mathbf{E}'(\mathbf{\Phi})\,\dot{\mathbf{\Phi}} \tag{A.2}$$

$\mathbf{E}(\mathbf{\Phi})$ and $\mathbf{E}'(\mathbf{\Phi})$ are the *Euler angle rates matrix* and *conjugate Euler angle rates matrix* respectively given by,

$$\mathbf{E}(\mathbf{\Phi}) = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{E}'(\mathbf{\Phi}) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{A.4}$$

Remark: $\mathbf{E}$ depends on pitch and yaw, whereas $\mathbf{E}'$ on roll and pitch. Thus, the Euler angle rates $\dot{\mathbf{\Phi}}$ is

$$\dot{\mathbf{\Phi}} = \mathbf{E}^{-1}(\mathbf{\Phi})\,\boldsymbol{\omega} \tag{A.5}$$

$$\dot{\mathbf{\Phi}} = \mathbf{E}'^{-1}(\mathbf{\Phi})\,_{\mathcal{C}}\boldsymbol{\omega} \tag{A.6}$$

References

Abbyasov, B., Lavrenov, R., Zakiev, A., Yakovlev, K., Svinin, M., and Magid, E. (2020). Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7232. https://doi.org/10.1109/ICRA40945.2020.9196621.

Aceituno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., Cappelletto, J., Grieco, J. C., Fernando-Lopez, G., and Semini, C. (2018). Simultaneous contact, gait and motion planning for robust multi-legged locomotion via mixed-integer convex optimization. In *IEEE Robotics and Automation Letters (RA-L)*. https://doi.org/10.1109/LRA.2017.2779821.

Amatucci, L., Kim, J.-H., Hwangbo, J., and Park, H.-W. (2022). Monte carlo tree search gait planner for non-gaited legged system control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4701–4707. https://doi.org/10.1109/ICRA46639.2022.9812421.

Arena, P., Sueri, P., Taffara, S., and Patanè, L. (2021). Mpc-based control strategy of a neuro-inspired quadruped robot. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. https://doi.org/10.1109/IJCNN52387.2021.9533394.

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1 – 3. https://doi.org/10.2140/pjm.1966.16.1.

Barasuol, V., Buchli, J., Semini, C., Frigerio, M., De Pieri, E. R., and Caldwell, D. G. (2013). A reactive controller framework for quadrupedal locomotion on challenging terrain. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2554–2561. https://doi.org/10.1109/ICRA.2013.6630926.

Barasuol, V., Camurri, M., Bazeille, S., Caldwell, D. G., and Semini, C. (2015). Reactive trotting with foot placement corrections through visual pattern classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5734–5741. https://doi.org/10.1109/IROS.2015.7354191.

Bednarek, J., Maalouf, N., Pollayil, M. J., Garabini, M., Catalano, M. G., Grioli, G., and Belter, D. (2020). Cnn-based foothold selection for mechanically adaptive soft foot. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10225–10232. https://doi.org/10.1109/IROS45743.2020.9340910.

Bellicoso, C. D., Jenelten, F., Gehring, C., and Hutter, M. (2018). Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):2261–2268. https://doi.org/10.1109/LRA.2018.2794620.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming*. SIAM Review. https://doi.org/10.1137/1.9780898718577.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc. https://doi.org/10.5555/525960.

Bjelonic, M., Grandia, R., Geilinger, M., Harley, O., Medeiros, V. S., Pajovic, V., Jelavic, E., Coros, S., and Hutter, M. (2022). Offline motion libraries and online mpc for advanced mobility skills. *The International Journal of Robotics Research*, 41(9-10):903–924. https://doi.org/10.1177/02783649221102473.

Bjelonic, M., Grandia, R., Harley, O., Galliard, C., Zimmermann, S., and Hutter, M. (2021). Whole-body mpc and online gait sequence generation for wheeled-legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8388–8395. https://doi.org/10.1109/IROS51168.2021.9636371.

Bledt, G. and Kim, S. (2019). Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323. https://doi.org/10.1109/IROS40897.2019.8968031.

Bledt, G., Powell, M. J., Katz, B., Carl0, J. D., Wensing, P. M., and Kim, S. (2018). MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. https://doi.org/10.1109/IROS.2018.8593885.

Bledt, G., Wensing, P. M., and Kim, S. (2017). Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT Cheetah. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4102–4109. https://doi.org/10.1109/IROS.2017.8206268.

Boaventura, T., Buchli, J., Semini, C., and Caldwell, D. (2015). Model-based hydraulic impedance control for dynamic robots. *Robotics, IEEE Transactions on*, 31(6):1324–1336. https://doi.org/10.1109/TRO.2015.2482061.

Bock, H. G. and Plitt, K.-J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608. https://doi.org/10.1016/s1474-6670(17)61205-9.

Bouyarmane, K. and Kheddar, A. (2018). On weight-prioritized multitask control of humanoid robots. *IEEE Transactions on Automatic Control*, 63(6):1632–1647. https://doi.org/10.1109/TAC.2017.2752085.

Boyd, S. and Vandeberghe, L. (2004). *Convex Optimization*. Cambridge University Press. https://doi.org/10.1017/CBO9780511804441.

Bratta, A., Focchi, M., Rathod, N., and Semini, C. (2022). Optimization-based reference generator for nonlinear model predictive control of legged robots. *Robotics*. accepted.

Bratta, A., Meduri, A., Focchi, M., RIghetti, L., and Semini, C. (2023). Contactnet: Online multi-contact planning for acyclic legged robot locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*. under review.

Bratta, A., Orsolino, R., Focchi, M., Barasuol, V., Muscolo, G. G., and Semini, C. (2020). On the hardware feasibility of nonlinear trajectory optimization for legged locomotion based on a simplified dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1427–1423. https://doi.org/10.1109/ICRA40945.2020.9196903.

Bretl, T. (2006). Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The International Journal of Robotics Research*, 25(4):317–342. https://doi.org/10.1177/0278364906063979.

Bretl, T. and Lall, S. (2008). Testing static equilibrium for legged robots. *IEEE Transactions on Robotics (T-RO)*, 24(4):794–807. https://doi.org/10.1109/TRO.2008.2001360.

Budhiraja, R., Carpentier, J., Mastalli, C., and Mansard, N. (2018). Differential dynamic programming for multi-phase rigid contact dynamics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1–9. https://doi.org/10.1109/HUMANOIDS.2018.8624925.

Butcher, J. C. (2003). *Numerical Methods for Ordinary Differential Equations*. J. Wiley.

Caron, S., Pham, Q.-C., and Nakamura, Y. (2017). Zmp support areas for multicontact mobility under frictional constraints. *IEEE Transactions on Robotics*, 33(1):67–80. https://doi.org/10.1109/TRO.2016.2623338.

Cebe, O., Tiseo, C., Xin, G., Lin, H.-C., Smith, J., and Mistry, M. N. (2021). Online dynamic trajectory optimization and control for a quadruped robot. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 12773–12779. https://doi.org/10.1109/ICRA48506.2021.9561592.

Chestnutt, J., Nishiwaki, K., Kuffner, J., and Kagami, S. (2007). An adaptive action model for legged navigation planning. In *IEEE-RAS International Conference on Humanoid Robots*, pages 196–202.

Chiacchio, P., Bouffard-Vercelli, Y., and Pierrot, F. (1997). Force polytope and force ellipsoid for redundant manipulators. *Journal of Robotic Systems*, 14:613–620. https://doi.org/10.1002/(SICI)1097-4563(199708)14.

Chignoli, M. and Wensing, P. M. (2020). Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797. https://doi.org/10.1109/ACCESS.2020.2980446.

Clausen, J. (2003). Branch and bound algorithms-principles and examples. In *Department of Computer Science, University of Copenhagen*.

Coumans, E. e. a. (2013). Bullet physics library. *Open source: bulletphysics. org*, 15(49):5.

Dai, H., Valenzuela, A., and Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 295–302. https://doi.org/10.1109/HUMANOIDS.2014.7041375.

Dantzig, B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Deits, R. and Tedrake, R. (2014a). Computing large convex regions of obstacle-free space through semidefinite programming.

Deits, R. and Tedrake, R. (2014b). Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 279–286. https://doi.org/10.1109/HUMANOIDS.2014.7041373.

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., and Kim, S. (2018). Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. https://doi.org/10.1109/IROS.2018.8594448.

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58:1–35.

Diehl, M., Bock, H. G., Diedam, H., and Wieber, P. B. (2005). Fast direct multiple shooting algorithms for optimal robot control. *Fast Motions in Biomechanics and Robotics,*, pages 65–93. https://doi.org//10.1007/978-3-540-36119-0-4.

Diehl, M. and Gros, S. (2017). *Lecture Notes on Numerical Optimal Control*. IMTEK - University of Freiburg.

Ding, Y., Pandala, A., and Park, H. W. (2019). Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. https://doi.org/10.1109/ICRA.2019.8793669.

Escande, A., Kheddar, A., Miossec, S., and Garsault, S. (2008). Planning support contact-points for acyclic motions and experiments on hrp-2. In *International Symposium on Experimental Robotics (ISER)*, pages 293–302. https://doi.org/10.1007/978-3-642-00196-3-35.

Fahmi, S., Focchi, M., Radulescu, A., Fink, G., Barasuol, V., and Semini, C. (2020). STANCE: Locomotion adaptation over soft terrain. *IEEE Transactions on Robotics (T-RO)*, 36(2):443–457. https://doi.org/10.1109/TRO.2019.2954670.

Fahmi, S., Mastalli, C., Focchi, M., and Semini, C. (2019). Passive whole-body control for quadruped robots: Experimental validation over challenging terrain. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):2553–2560. https://doi.org/10.1109/LRA.2019.2908502.

Fankhauser, P., Bjelonic, M., Dario Bellicoso, C., Miki, T., and Hutter, M. (2018). Robust rough-terrain locomotion with a quadrupedal robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768. https://doi.org/10.1109/ICRA.2018.8460731.

Fankhauser, P., Bloesch, M., Gehring, C., and Hutter, M. (2014). Robot-centric elevation mapping with uncertainty estimates. In *International Conference on Climbing and Walking Robots (CLAWAR)*, pages 5761–5768. https://doi.org/10.1142/9789814623353-0051.

Fankhauser, P. and Hutter, M. (2016). A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In *Robot Operating System, Vol. 1*, chapter 5, pages 99–120. Springer. https://doi.org/10.1007/978-3-319-26054-9_5.

Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., and Buchli, J. (2017a). Real-time motion planning of legged robots: A model predictive control approach. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. https://doi.org/10.1109/HUMANOIDS.2017.8246930.

Farshidian, F., Neunert, M., Winkler, A. W., Rey, G., and Buchli, J. (2017b). An efficient optimal planning and control framework for quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 93–100. https://doi.org/10.1109/ICRA.2017.7989016.

Featherstone, R. (2007). *Rigid Body Dynamics Algorithms*. Springer-Verlag,Berlin, Heidelberg. https://doi.org/10.1007/978-1-4899-7560-7.

Fernbach, P., Tonneau, S., Stasse, O., Carpentier, J., and Taïx, M. (2020). C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios. *IEEE Transactions on Robotics (T-RO)*, 36(3):676–691. https://doi.org/10.1109/TRO.2020.2964787.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, USA, second edition.

Focchi, M., Barasuol, V., Havoutis, I., Buchli, J., Semini, C., and Caldwell, D. G. (2013). Local reflex generation for obstacle negotiation in quadrupedal locomotion. In *International Conference on Climbing and Walking Robots (CLAWAR)*. https://doi.org/10.1142/9789814525534-0056.

Focchi, M., del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D. G., and Semini, C. (2016). High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, pages 1–14. https://doi.org/10.1007/s10514-016-9573-1.

Focchi, M., Featherstone, R., Orsolino, R., Caldwell, D. G., and Semini, C. (2017). Viscosity-based height reflex for workspace augmentation for quadrupedal locomotion on rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5353–5360. https://doi.org/10.1109/IROS.2017.8206430.

Focchi, M., Orsolino, R., Camurri, M., Barasuol, V., Mastalli, C., Caldwell, D. G., and Semini, C. (2020). Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality. *Springer Tracts in Advanced Robotics (STAR)*, pages 165–209. https://doi.org/10.1007/978-3-030-22327-4-9.

Frigerio, M., Buchli, J., and Caldwell, D. G. (2012). Code generation of algebraic quantities for robot controllers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2346–2351. https://doi.org/10.1109/IROS.2012.6385694.

Frison, G. and Diehl, M. (2020). Hpipm: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569. https://doi.org/10.1016/j.ifacol.2020.12.073.

Full, R. and Koditschek, D. (1999). Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332. https://doi.org/10.1242/jeb.202.23.3325.

Gehring, C., Bellicoso, C. D., Coros, S., Bloesch, M., Fankhauser, P., Hutter, M., and Siegwart, R. (2015). Dynamic trotting on slopes for quadrupedal robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5129–5135. https://doi.org/10.1109/IROS.2015.7354099.

Geisert, M., Del Prete, A., Mansard, N., Romano, F., and Nori, F. (2017). Regularized hierarchical differential dynamic programming. *IEEE Transactions on Robotics (T-RO)*, 33(4):819–833. https://doi.org/10.1109/TRO.2017.2671355.

Giftthaler, M., Neunert, M., Stäuble, M., Frigerio, M., Semini, C., and Buchli, J. (2017). Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22):1225–1237. https://doi.org/10.1080/01691864.2017.1395361.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Grandia, R., Farshidian, F., Dosovitskiy, A., Ranftl, R., and Hutter, M. (2019a). Frequency-aware model predictive control. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1517–1524. https://doi.org/10.1109/LRA.2019.2895882.

Grandia, R., Farshidian, F., Ranftl, R., and Hutter, M. (2019b). Feedback mpc for torque-controlled legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737. https://doi.org/10.1109/IROS40897.2019.8968251.

Grandia, R., Jenelten, F., Yang, S., Farshidian, F., and Hutter, M. (2022). Perceptive locomotion through nonlinear model predictive control. *arXiv*.

Griffin, R. J., Wiedebach, G., McCrory, S., Bertrand, S., Lee, I., and Pratt, J. (2019). Footstep planning for autonomous walking over rough terrain. In *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 9–16. https://doi.org/10.1109/HUMANOIDS43949.2019.9035046.

Grimminger, F., Meduri, A., Khadiv, M., Viereck, J., Wuthrich, M., Naveau, M., Berenz, V., Heim, S., Widmaier, F., Flayols, T., Fiene, J., Badri-Sprowitz, A., and Righetti, L. (2020). An open torque-controlled modular robot architecture for legged locomotion research. *Robotics and Automation Letters (RA-L)*, 5(2):3650–3657. https://doi.org/10.1109/lra.2020.2976639.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2020). From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1):62–80. https://doi.org/10.1080/00207179.2016.1222553.

Guennebaud, G., Furfaro, A., and Gaspero, L. D. (2011). eiquadprog.hh.

Hager, W., Hou, H., Mohapatra, S., and Rao, A. (2019). Convergence rate for an hp collocation method applied to unconstrained optimal control. *Computational Optimization and Applications*, 74. https://doi.org/10.1007/s10589-019-00108-7.

Hairer, E., Nørsett, S., and Wanner, G. (1993). *Solving Ordinary Differential Equations I.* Springer, Berlin, 2nd edition.

Hairer, E., Nørsett, S., and Wanner, G. (1996). *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems.* Springer, Berlin, 2nd edition.

Harada, K., Kajita, S., Kaneko, K., and Hirukawa, H. (2003). Zmp analysis for arm/leg coordination. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 75–81. https://doi.org/10.1109/IROS.2003.1250608.

Herdt, A., Diedam, H., Wieber, P.-B., Dimitrov, D., Mombaur, K. D., and Diehl, M. (2010). Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24:719 – 737. https://doi.org/10.1163/016918610X493552.

Holmes, P., Full, R. J., Koditschek, D. E., and Guckenheimer, J. M. (2006). The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Rev.*, 48:207–304. https://doi.org/10.1137/S0036144504445133.

Hong, S., Kim, J.-H., and Park, H.-W. (2020). Real-time constrained nonlinear model predictive control on so(3) for dynamic legged locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3982–3989. https://doi.org/10.1109/IROS45743.2020.9341447.

Hult, R., Zanon, M., Gros, S., and Falcone, P. (2019). Optimal coordination of automated vehicles at intersections: Theory and experiments. *IEEE Transactions on Control Systems Technology*, 27(6):2510–2525. https://doi.org/10.1109/TCST.2018.2871397.

Jenelten, F., Grandia, R., Farshidian, F., and Hutter, M. (2022). Tamols: Terrain-aware motion optimization for legged systems. *arXiv*.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1620–1626. https://doi.org/10.1109/ROBOT.2003.1241826.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 239–246. https://doi.org/10.1109/IROS.2001.973365.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2011). Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258. https://doi.org/10.1177/0278364910388677.

Kalakrishnan, M., Buchli, J., Pastor, P., and Schaal, S. (2009). Learning locomotion over rough terrain using terrain templates. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 167–172. https://doi.org/10.1109/IROS.2009.5354701.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395.

Karush, W. (1939). *Minima of Functions of Several Variables with Inequalities as Side Constraints*. (M.Sc. thesis). Dept. of Mathematics, Univ. of Chicago.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53. https://doi.org/10.1109/JRA.1987.1087068.

Kim, D., Di Carlo, J., Katz, B., Bledt, G., and Kim, S. (2019). Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv*.

Koenemann, J., Del Prete, A., Tassa, Y., Todorov, E., Stasse, O., Bennewitz, M., and Mansard, N. (2015). Whole-body model-predictive control applied to the hrp-2 humanoid. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3346–3351. https://doi.org/10.1109/IROS.2015.7353843.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154 vol.3. https://doi.org/10.1109/IROS.2004.1389727.

Kolter, J. Z., Rodgers, M. P., and Ng, A. Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 811–818. https://doi.org/10.1109/ROBOT.2008.4543305.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. *Proceedings of 2nd Berkeley Symposium*, pages 481–492.

Labbe, Y., Zagoruyko, S., Kalevatykh, I., Laptev, I., Carpentier, J., Aubry, M., and Sivic, J. (2020). Monte-carlo tree search for efficient visually guided rearrangement planning. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):3715–3722. https://doi.org/10.1109/LRA.2020.2980984.

Li, H. and Wensing, P. (2020). Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters (RA-L)*, 5:5448–5455. https://doi.org/10.1109/LRA.2020.3007475.

Li, H., Wensing, P., Zhang, T., and Yu, W. (2022). Zero-shot retargeting of learned quadruped locomotion policy using a hybrid kinodynamic model and predictive control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Lin, R. (2022). Analysis on the selection of the appropriate batch size in cnn neural network. In *International Conference on Machine Learning and Knowledge Engineering (MLKE)*, pages 106–109. https://doi.org/10.1109/MLKE55170.2022.00026.

Mansard, N. (2015). *Numerical Methods for Robotics*. LAAS-CNRS Geppetto Team.

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2020). Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics (T-RO)*, 36(6):1635–1648. https://doi.org/10.1109/TRO.2020.3003464.

Mastalli, C., Merkt, W., Xin, G., Shim, J., Mistry, M., Havoutis, I., and Vijayakumar, S. (2022). Agile maneuvers in legged robots: a predictive control approach. *arXiv*.

Mayne, D. (1966). A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95. https://doi.org/10.1080/00207176608921369.

McGhee, R. and Frank, A. (1968). On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351. https://doi.org/10.1016/0025-5564(68)90090-4.

Meduri, A., Khadiv, M., and Righetti, L. (2021). Deepq stepper: A framework for reactive dynamic walking on uneven terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2099–2105. IEEE. https://doi.org/10.1109/ICRA48506.2021.9562093.

Meduri, A., Shah, P., Viereck, J., Khadiv, M., Havoutis, I., and Righetti, L. (2022). Biconmp: A nonlinear model predictive control framework for whole body motion planning. *IEEE Transactions on Robotics (T-RO)*. To be published.

Melon, O., Geisert, M., Surovik, D., Havoutis, I., and Fallon, M. (2020). Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1410–1416. https://doi.org/10.1109/ICRA40945.2020.9196562.

Minniti, M. V., Grandia, R., Farshidian, F., and Hutter, M. (2022). Adaptive clf-mpc with application to quadrupedal robots. *IEEE Robotics and Automation Letters (RA-L)*, 7(1):565–572.

Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8. https://doi.org/10.1145/2185520.2185539.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *International Conference on International Conference on Machine Learning*, page 807–814. https://doi.org/10.5555/3104322.3104425.

Neunert, M., Farshidian, F., Winkler, A., and Buchli, J. (2016). Trajectory optimization through contacts and automatic gait discovery for quadrupeds. *IEEE Robotics and Automation Letters (RA-L)*, 2. https://doi.org/10.1109/LRA.2017.2665685.

Neunert, M., Stauble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M., and Buchli, J. (2018). Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):1458–1465. https://doi.org/10.1109/LRA.2018.2800124.

Nobili, S., Camurri, M., Barasuol, V., Focchi, M., Caldwell, D. G., Semini, C., and Fallon, M. (2017). Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. In *Proceedings of Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2017.XIII.007.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer. https://doi.org/10.1007/b98874.

Orin, D. E., Goswami, A., and Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35. https://doi,org/10.1007/s10514-013-9341-4.

Ponton, B., Khadiv, M., Meduri, A., and Righetti, L. (2021). Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics. *IEEE Transactions on Robotics (T-RO)*, 37(5):1661–1679. https://doi.org/10.1109/TRO.2020.3048125.

Posa, M., Cantu, C., and Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81. https://doi.org/10.1177/0278364913506757.

Potra, F. A. and Wright, S. J. (2000). Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302. https://doi.org/10.1016/S0377-0427(00)00433-7.

Poulakakis, I. and Grizzle, J. W. (2009). The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793. https://doi.org/10.1109/TAC.2009.2024565.

Pratt, J., Chew, C.-M., Torres, A., Dilworth, P., and Pratt, G. (2001). Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143. https://doi.org/10.1177/02783640122067309.

Pratt, J., Dilworth, P., and Pratt, G. (1997). Virtual model control of a bipedal walking robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 193–198. https://doi.org/10.1109/ROBOT.1997.620037.

Qin, J. S. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764. https://doi.org/10.1016/S0967-0661(02)00186-7.

Quirynen, R. (2017). *Numerical Simulation Methods for Embedded Optimization*. PhD thesis, Arenberg Doctoral School, Dept. Eng. Sci., KU Leuven, Leuven, Belgium, Dept. Math. Phys., Univ. Freiburg, Freiburg im Breisgau, Germany.

Quirynen, R., Vukov, M., Zanon, M., and Diehl, M. (2014). Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. *Optimal Control Applications and Methods*, 36:685–704. https://doi.org/10.1002/oca.2152.

Raibert, M., Chepponis, M., and Brown, H. (1986). Running on four legs as though they were one. *IEEE Journal on Robotics and Automation*, 2(2):70–82. https://doi.org/10.1109/JRA.1986.1087044.

Raibert, M. H. (1986). *Legged robots that balance*. MIT press.

Raiola, G., Mingo Hoffman, E., Focchi, M., Tsagarakis, N., and Semini, C. (2020). A simple yet effective whole-body locomotion framework for quadruped robots. *Frontiers in Robotics and AI*, 7:159. https://doi.org/10.3389/frobt.2020.528473.

Rathod, N., Bratta, A., Focchi, M., Zanon, M., Villarreal, O., Semini, C., and Bemporad, A. (2021). Model predictive control with environment adaptation for legged locomotion. *IEEE Access*, 9:145710–145727. https://doi.org/10.1109/ACCESS.2021.3118957.

Ruchika, N. R. (2013). Model predictive control: History and development. *International Journal of Engineering Trends and Technology (IJETT)*, 4(6):2600–2602. https://doi.org/10.1109/TRO.2020.2964787.

Schmid, L., Gerharz, A., Groll, A., and Pauly, M. (2022). Machine learning for multi-output regression: When should a holistic multivariate approach be preferred over separate univariate ones? In *arXiv*.

Schwind, W. J. (1998). *Spring Loaded Inverted Pendulum Running: A Plant Model*. PhD thesis, University of Michigan, USA.

Sciavicco, L. and Siciliano, B. (2000). *Modelling and Control of Robot Manipulators*. Springer-Verlag, Berlin, Heidelberg. https://doi.org/10.1007/978-1-4471-0449-0.

Semini, C. (2010). *HyQ – Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, Istituto Italiano di Tecnologia (IIT) and University of Genova.

Semini, C., Barasuol, V., Focchi, M., Boelens, C., Emara, M., Casella, S., Villarreal, O., Orsolino, R., Fink, G., Fahmi, S., Medrano-Cerda, G., Sangiah, D., Lesniewski, J., Fulton,

K., Donadon, M., Baker, M., and Caldwell, D. G. (2019). Brief introduction to the quadruped robot hyqreal. *Italian Conference on Robotics and Intelligent Machines (I-RIM)*.

Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F., and Caldwell, D. G. (2011). Design of HyQ -A hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 225(6):831–849. https://doi.org/10.1177/0959651811402275.

Sheridan, T. (1966). Three models of preview control. *IEEE Transactions on Human Factors in Electronics*, HFE-7(2):91–102. https://doi.org/10.1109/THFE.1966.232329.

Shkolnik, A., Levashov, M., Manchester, I. R., and Tedrake, R. (2011). Bounding on rough terrain with the littledog robot. *The International Journal of Robotics Research*, 30(2):192–215. https://doi.org/10.1177/0278364910388315.

Siciliano, B. (2007). *Springer Handbook of Robotics*. Springer-Verlag,Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30301-5.

Sideris, A. and Bobrow, J. E. (2005). An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems. *Proceedings of the 2005, American Control Conference, 2005.*, 4:2275–2280. https://doi.org/10.1109/TAC.2005.860248.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *ArXiv*.

Sutherland, I. E. and Ullner, M. K. (1984). Footprints in the asphalt. *The International Journal of Robotics Research*, 3(2):29–36. https://doi.org/10.1177/027836498400300203.

Tassa, Y., Mansard, N., and Todorov, E. (2014). Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. https://doi.org/10.1109/ICRA.2014.6907001.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. https://doi.org/10.1109/IROS.2017.8202133.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033.

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics (T-RO)*, 34(3):586–601. https://doi.org/10.1109/TRO.2018.2819658.

Tonneau, S., Song, D., Fernbach, P., Mansard, N., Taïx, M., and Del Prete, A. (2020). Sl1m: Sparse l1-norm minimization for contact planning on uneven terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6604–6610. https://doi.org/10.1109/ICRA40945.2020.9197371.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., Frey, J., Albin, T., Quirynen, R., and Diehl, M. (2019). Acados - a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*. https:/doi.org/10.1007/s12532-021-00208-8.

Villarreal, O., Barasuol, V., Camurri, M., Franceschi, L., Focchi, M., Pontil, M., Caldwell, D. G., and Semini, C. (2019). Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters*, 4(2):2140–2147. https://doi.org/10.1109/LRA.2019.2899434.

Villarreal, O., Barasuol, V., Wensing, P. M., Caldwell, D. G., and Semini, C. (2020). MPC-based controller with terrain insight for dynamic legged locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2436–2442. https://doi.org/10.1109/ICRA40945.2020.9197312.

Vukobratovic, M. and Borovac, B. (2004). Zero-moment-point - thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173. https://doi.org/10.1142/S0219843604000083.

Vukobratovic, M. and Juricic, D. (1969). Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering*, BME-16(1):1–6. https://doi.org/10.1109/TBME.1969.4502596.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57. https://doi.org/10.1007/s10107-004-0559-y.

Wensing, P., Posa, M., Hu, Y., Escande, A., Mansard, N., and Prete, A. D. (2022). Optimization-based control for dynamic legged robots. *arXiv*.

Wieber, P.-B. (2006). Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 137–142. https://doi.org/10.1109/ICHR.2006.321375.

Winkler, A. W., Bellicoso, C. D., Hutter, M., and Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters (RA-L)*, 3:1560–1567. https://doi.org/10.1109/LRA.2018.2798285.

Wong, E. (2011). *Active-Set Methods for Quadratic Programming*. PhD thesis, University of California, San Diego.

Wright, M. (2004). The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56. https://doi.org/10.1090/S0273-0979-04-01040-7.

Yoshikawa, T. (1984). Analysis and control of robot manipulators with redundancy. In *The First International Syposium*, pages 735–747. https://doi.org/10.1.1.18.7268.

Younes, A. B., Turner, J. D., Mortari, D., and Junkins, J. L. (2012). A survey of attitude error representations. *AIAA/AAS Astrodynamics Specialist Conference*. https://doi.org/10.2514/6.2012-4422.

Zanon, M. (2021). *Lecture Notes on Numerical Methods for Optimal Control*. IMT Lucca.

List of Publications

1. **A. Bratta**, R. Orsolino, M. Focchi, V. Barasuol, G. G. Muscolo and C. Semini, "On the Hardware Feasibility of Nonlinear Trajectory Optimization for Legged Locomotion based on a Simplified Dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1417-1423, doi: 10.1109/ICRA40945.2020.9196903.

2. N. Rathod, **A. Bratta**, M. Focchi, M. Zanon, O. Villarreal, C. Semini and A. Bemporad, "Model Predictive Control With Environment Adaptation for Legged Locomotion," in *IEEE Access*, 2021, vol. 9, pp. 145710-145727, doi: 10.1109/ACCESS.2021.3118957, *2021 IEEE Access Best Video Award (Part 2)*.

3. **A. Bratta**, N. Rathod, M. Zanon, O. Villarreal, A. Bemporad, C. Semini and M. Focchi, "Towards a Nonlinear Model Predictive Control for Quadrupedal Locomotion on Rough Terrain" in *Italian Conference in Robotics and Intelligent Machines*, 2021, *Finalist Best Student Paper Award*

4. L. Clemente, O. Villarreal, **A. Bratta**, M. Focchi, V. Barasuol, G. G. Muscolo, and C. Semini, "Foothold Evaluation Criterion for Dynamic Transition Feasibility for Quadruped Robots", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4679-4685, doi: 10.1109/ICRA46639.2022.9812434.

5. **A. Bratta**, M.Focchi, N. Rathod, and C. Semini "Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots" in *Robotics*, 2022.

6. **A.Bratta**, A.Meduri, M. Focchi, L. Righetti, and C. Semini, "ContactNet: Online Multi-Contact Planning for Acyclic Legged Robot Locomotion", *IEEE International Conference on Robotics and Automation (ICRA)*, 2023 (under review).