POLITECNICO DI TORINO

Master degree course in Mechatronics Engineering

Master Degree Thesis

Model-Based Motion Planning for Quadruped Robots: from Simulation to Hardware Implementation

Supervisor
Prof. Giovanni Gerardo Muscolo

Co-supervisors:
Dott. Claudio Semini
Dott. Romeo Orsolino

Candidate
Angelo Bratta
s249839

Academic Year 2018/2019

Abstract

Legged robots have been widely investigated, in order to obtain outdoor motion, in which they could collaborate with humans. In order to do that, they have to adapt their movements to different environments in which they can work, keeping always the capability to complete the required task. Researchers have, therefore, dedicated big efforts in developing techniques which produce well suited trajectories that can be followed by the robot.

Starting from a simplified model, we propose two feasibility constraints to be included in a Single Rigid Body Dynamics-based trajectory optimizer in order to obtain robust motions in challenging terrain. The former finds an approximate relationship between joint torque limits and admissible contact forces without requiring the joints' configuration. The latter proposes a model of the leg to guarantee the avoidance of the collision with the environment. Such constraints have been included in a nonlinear non-convex optimization problem, implemented in a library named TOWR. We validate the feasibility of the trajectories both in simulation and on the Hydraulically actuated Quadruped robot, including experiments with non flat terrain.

The contributions of this thesis are the result of a 6 months project that was performed at the Dynamic Legged Systems (DLS) lab at IIT. A conference paper reporting the three mentioned contributions has been submitted for evaluation to the peer-reviewed International Conference on Robotics and Automation (ICRA), 2020. The conference paper is now under revision and the final outcome of this review process will be communicated in February 2020

Contents

List of Figures

List of Acronyms

IIT	Istituto Italiano di Tecnologia
DLS	Dynamic Legged Systems
HyQ	Hydraulically actuated Quadruped
LIP	Linear Inverted Pendulum
SLIP	Spring Loaded Inverted Pendulum
CD	Centroidal Dynamics
SRBD	Single Rigid Body Dynamics
LP	Linear Problem
QP	Quadratic Problem
NLP	Nonlinear Problem
DL	Deep Learning
RL	Reinforcement Learning
CoP	Center of Pressure
ZMP	Zero Moment Point
CWC	Contact Wrench Cone
CoM	Center of Mass
TOWR	Trajectory Optmization for Walking Robots
ROS	Robot Operating SystemROS
PD	Proportional-Derivative
WBC	Whole Body Controller

IK	Inverse Kinematics
HAA	Hip Abduction-Adduction
HFE	Hip Flexion-Extension
KFE	Knee Flexion-Extension

Chapter 1

Introduction

Starting from the '80s, legged robots became one of the most investigated fields of robotics.
They can be considered as an evolution of the wheeled robots, since they present higher mobility in case of irregular terrains. This characteristic can be exploited to explore and work on difficult, hostile and dangerous environments, such as nuclear power plants and disaster areas. Legged robot can, therefore, substitute the humans in discovering the environment and gathering informations, thanks to the use of sensors mounted onto the robot, like cameras and LIDAR. This aspect guarantees more safety for the men involved in the rescue operations, since they will be more conscious of the environment before going through it. Thanks to the robots, rescue operations could become more efficient and safe.
The motion of the robot must be reliable, so a high level of robustness is required. Due to the very different scenarios in which it can work, a legged robot has to be able to adapt itself to different kinds of terrains, keeping always the stability during the motion and the capability to complete the desired task. In addition, it has to be provided with reactive features and recovery strategies, in order to be able to reach the goal even if something is going different from what has been planned.

1.1 Aim

The aim of this thesis is to present an improvement of the Single Rigid Body Dynamics model for quadruped robot locomotion, in order to deal with non flat terrains. It is the result of a work performed at the Dynamic Legged System (DLS) at the Istituto Italiano di Tecnologia (IIT), in Genova.
This thesis deals with the two critical issues of such model and proposes:

- a novel approximate, robot-agnostic, projection of the joint torque limits into the task space. In this way we can express the relationship between the leg's configuration and the corresponding maximal contact forces at the end-effector. Keeping out the joints from the model, we are able to compute the boundary values of the contact force in the actual configuration in which the robot is. Imposing that contact forces are inside these limits assures implicitly that the resulting joint torques are feasible.
- a novel model of the leg's lower link to include into the trajectory optimization formulation the geometry of the leg, such as finite non-zero size of the robot's feet and shin's geometry.

Starting point of the work is the library TOWR, which is the implementation of a non linear Single Rigid Body Dynamics based trajectory optimizer.
As an experimental contribution, we have included the above mentioned improvement in TOWR library and then we have integrated it with the controller of the Hydraulically-actuated Quadruped robot (HyQ) of the DLS lab. We have successfully performed motion on a non-flat terrain (step of $10cm$). To the best of our knowledge this is the first time that a trajectory based on TOWR is deployed on a real robot on a non-flat terrain.

1.2 Methodologies

In order to guarantee the safety of the operators (and of the robot) proper simulations have been performed before doing experiments on the real hardware. First step has been done using the 3D visualizer RVIZ. After that, we have used an accurate software simulator like Gazebo, building the proper ROS structure to integrate the work with all the other elements of the robot. At the end of the simulation-based validation, we have worked with the robot, provided with sanity checks (possibility for the operator to pause the motion, possibility to automatic stop the motion when the robot has the four feet on the ground, etc...). At the end of the work, the robot has been able to step up onto a $10cm$ high pallet.
Thanks to this methodical approach, we have faced and fixed issues of increasingly complexity, working in the most efficient way and always in a safety environment

1.3 Outline

The thesis is organized in the following manner:
Chapter 2 introduces the block diagram of the motion structure of a robot, highlighting the trajectory planner (with its models and different approaches) and

stability criteria.
Chapter 3 analyses the details of the library TOWR.
Chapter 4 we give an overview of the DLS lab and of the HyQ robot.
Chapter 5 presents the concepts *Collision Avoidance Constraint* and *Force Polytopes Constraint*.
Chapter 6 explains the interface we have developed to integrate the trajectory planner and the controller of HyQ. In this chapter we present the obtained results both in simulation and on the robot.
Chapter 7 is dedicated to review the concepts investigated in the previous chapters and to discuss possible future works.

Chapter 2

State of the Art

Before going into the details of the contribution of this thesis, it is worth dedicating a chapter to the introduction the concept of trajectory planning, highlighting different approaches and giving a few examples. In addition, we present a few stability criteria which are strongly related to the trajectory planners.

2.1 Trajectory planners

Motion of a robot is the result of the composition of three main elements:

- *path generator*: the path is the sequence of the desired states in which the robot must go from the initial to the final position. Path generators, generally, compute paths which refer to the end effector, *i.e.* the tool located at the end of the limbs of the robot and which are usually assumed to be the only part that can make or break contacts with the environment. In this case, there are no informations about joint quantities and time (such as joint positions, torques or phase durations).
- *trajectory planner*: the trajectory represents the time law of the path. The trajectory planner considers also the kinematic and dynamic constraint of the robot (e.g. maximum speed, acceleration, torques). The trajectory can be expressed either in the task space, *i.e.* position and orientation of the end effector, or in the joint space, *i.e.* the space defined by joint angles displacements.
- *controller*: the controller is the element which guarantees that the robot follows the specified trajectory. It receives as input the error between actual and desired joint kinematic quantities (*i.e.* position and velocity) to compute the force/torque variables for the joint actuator. Most commonly used architectures are Proportional-Derivative and Proportional-Integral-Derivative. In

case of a trajectory in the task space, Inverse Kinematics is used to compute joint position and velocities.

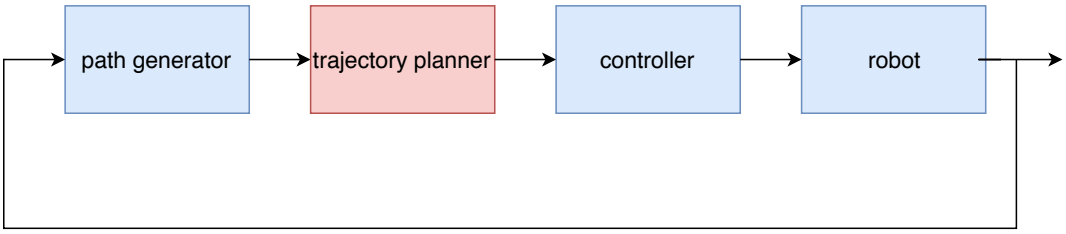

Figure 2.1 – Simplified block diagram corresponding to the motion scheme.

In this work we will focus on the trajectory planners.
The trajectory planning is a key element for the successful motion of the robot in the environment, especially in case of unstructured environments and irregular terrains. Because of the uncertainties and unmodeled dynamics which are present in the system, it is required to have a precise and robust trajectory with reduced computational time and effort to have frequent replanning. These two elements can be considered as conflicting, since precision and robustness can be achieved only if a complete, possibly nonlinear model, is used. On the contrary, the intrinsic complexity of these kinds of models increases the time and the effort required to obtain a solution. In addition, non linear models have local minima which can be wrongly considered the global one. The choice of the model is, thus, a crucial aspect in the design of trajectory planning algorithms, since it has to find the compromise between precision and computational velocity which is suited for the requirements. For this reason, the literature on this topic is typically split between the usage of simplified dynamic models and the usage of whole body models. Moreover, according to the time needed by the planner, it is possible to define the frequency of computation a new trajectory. In case of fast planner, *online* ones are used to start replanning while the robot is moving. This guarantees that the trajectory is computed starting from a more updated position of the robot.

2.1.1 Dynamic Models

Before analysing different techniques for the trajectory planning, it is worth introducing a brief but comprehensive explanation of different simplified dynamic models that can be used.

Linear Inverted Pendulum Model

Linear Inverted Pendulum Model (LIP) [9] [10] is the simplest model. It consists of a point mass, which corresponds to the Center of Mass (CoM) of the whole

robot. It is connected to a telescopic massless leg, which is fixed to the ground. Position of the foot depends on the angle of the leg with the vertical and the length of the leg. Such model has been exploited in trajectory planning, *e.g.* in [11]. Kajita *et al.* have used LIP to perform an outdoor walking with a bipedal robot.

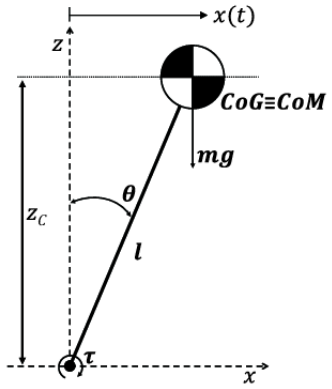

Figure 2.2 – Linear Inverted Pendulum

Spring Loaded Inverted Pendulum

Spring Loaded Inverted Pendulum (SLIP) [12], [13] considers a compliant leg. The most widespread approach to model the leg is the use of a spring of constant stiffness k. Due to presence of the spring and to the variable height of the CoM, the motion is the result of the transformation of gravitational and elastic potential energy into kinetic energy. From the point of view of energy variation it is, thus, possible to differentiate the walk into stance (foot in contact with the ground) and swing (foot in the air) phase. Stance phase is associated to variation in elastic potential energy and it can be divided in a sub-phase in which the variation is positive (leg is extended) and a sub-phase in which it is negative (leg is extending). A specular division can be done during the swing phase, exploiting the variation of gravitational energy. Assuming that there is no slippage when the foot is in contact, the dynamics of the robot is the sequence of the dynamics of the four sub-phases which have to be analysed independently. The efficiency of this model has been proved by [14], in which the authors have built a robot leg driven by the Series Elastic Actuator (SEA). This choice represents a suitable actuator system for interacting with the ground. In addition, Oh *et al.* [15] performed simulation with a SLIP based biarticular mechanism, embedded with a disturbance-observer-controller to have robustness in the motion.

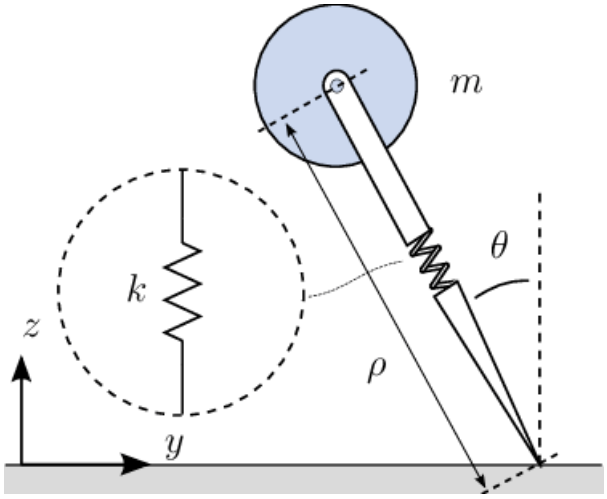

Figure 2.3 – Spring Loaded Inverted Pendelum [1]

Centroidal Dynamics

A third option is represented by the Centroidal Dynamics (CD) [16], [17] which exploits the full dynamics of a robot projected at its own CoM. The CD is a precise description of robot's dynamics in terms of its inputs (feet and CoM wrenches) and outputs (feet and CoM trajectories); for this reason, it should therefore not be considered as an approximate dynamic model. It is the first model in which joints are present, even if only to compute the inertia matrix of the robot. The matrix does not depend explicitly on the time, but it depends on the joints configuration that changes during the motion and it is, therefore, a time-varying quantity.

Single Rigid Body Dynamics

A simplification of the CD is the Single Rigid Body Dynamics (SRBD), where the robot is seen as a single rigid body with massless legs. In this case, robot's inertia is fixed and corresponds to its aggregate inertia in a predefined configuration. This implies that, unlike the CD, the robot's CoM matches the CoM of the base and it therefore does not move with respect to the base frame (*i.e.* the reference frame with the origin in the geometric center of the base of the robot and aligned with the robot itself). Due to its simplicity, SRBD is well suited to problems which require computational efficiency while dealing with complex terrains and possible non-coplanar contacts. In addition, it is a suitable approximation for robot with legs whose mass is negligible compared to the trunk's weight.

Whole Body Model

The Whole Body Model (WBM) is the more complex one, where also the joint variables are considered. In every time instant we are interested in configuration of the N joints ($\boldsymbol{q}$), position ($\boldsymbol{r}$) and orientation($\boldsymbol{\theta}$) of the CoM. Without going too much in depth, actuation torques depends on the inertia matrix, the Coriolis effect induced on a generic joint by other two joints, gravity force and contact force with the environment. All these quantities are affected by joint position, velocity and acceleration and their relationship is strongly nonlinear. Due to high number of variables involved in the model, WBM is used for trajectory optimization when a very precise and offline algorithm is required.

2.1.2 Online/Offline Optimization

The first approach for trajectory planning consists of solving an optimization problem, *i.e.* finding the optimal solution between all the feasible solutions of the specified problem. In other words, it means finding the value of the variable of interest such that the *objective function* (called also *cost function*) takes on the smallest value which satisfies the constraints.

$$\begin{aligned} \min_x \quad & f(x) \\ s.t \quad & g_i(x) \leq 0 \quad i = 1, ..., m \\ & h_j(x) = 0 \quad j = 1, ..., p \end{aligned} \tag{2.1}$$

where $f(x)$ is the *objective function* and $g_i(x)$ and $h_i(x)$ are respectively the *inequality* and *equality constraints*.
Optimization problems can be differentiated according to the type of objective function and constraints which are included in the formulation.
The first big differentiation which has to be done is betwen *convex* and *non-convex* problems:

- *convex*: a problem is defined *convex* if both the objective function and the constraints are convex *i.e.* if the segment which connects two points on the graph of the function lies above or on the graph, for every couple of arbitrary points). Immediate consequence of this is that the problem foresees either no solutions or global optimal solutions.
- non-convex: a problem is defined *non-convex* if either the objective function or at least one of the constraints are concave, *i.e.* they are not convex. The result is that a minimum can be either a local or a global optimum. Moreover, computation time increases with respect to convex problems and it is not possible to assert whether the problem has a solution or not.

According to the computation time and effort (which depend on elements such that complexity of the model, characteristics of the environment, etc...) trajectory planners can be considered as *online* or *offline* [18]. Online algorithms have a low computation time and they are able to compute the new trajectory during the motion of the robot. In this way the planner computes incrementally the trajectory from the robot state to the goal. Such approach allows the robot to react to changes in the environment or to tracking errors. Offline algorithms, instead, computes the entire trajectory from initial to goal state before the robot begins the motion.

Linear Problem

The easiest formulation for an optimization problem is represented by a *Linear Problem* (LP), in which both objective function and constraints are linear in the decision variable x. LPs are always convex and it is possible to find solutions also in case of problems with big number of variables involved. In case of problems in two variables, the solution can be found with the *graphical method*, which consists in drawing all the constraints and the objective function (they will be lines). Starting from these lines, it is possible to find the area of the valid solutions - called *Feasible Support Region* - and then the minimal value of objective function [19]. For an arbitrary number of variables, graphical method is not adequate, so the *Simplex method* has been introduced by George Dantzig in 1947 [20]. The method starts from the idea that the solution of a LP coincides with a corner of the Feasible Support Region, although there may be multiple optimal solutions. Simplex Method usually starts at a corner and then it moves to the neighboring corner that best improves the solution. When no more improvements can be made changing the corner, it means that the optimal solution has been found.

Quadratic Problem

A particular case of LP is the Quadratic Problem (QP), in which the objective function is quadratic in the optimizing variables and the constraints are linear [21]. Simplex Method cannot be applied, since the minimum does not coincide with a corner of the feasible region. QPs are generally solved either through gradient methods, such that gradient-based descent/ascent, or through Active Set Method [22] . In the former the next iteration step is proportional to the negative/positive gradient of the function. The latter is composed by two phases: the phase in which feasible points are computed, neglecting the objective function and the phase in which is minimized keeping the feasibility.

Nonlinear Problem

The most complex type of optimization problem is the Nonlinear Problem (NLP), which includes all the situations in which either objective function or at least one constraint are not linear. NLPs generally require high computational effort, as they are non convex and so may own multiple local minima. On the other end, they guarantee precision and reliability of the solution. Other drawback of the NLP is that an initial solution, called also *initial guess*, has to be provided to the solver. In general it can be stated that closer the initial guess is to the optimal solution easier will be for a solver to find a solution. More precise guesses require, instead, pre-processing work, like either solving a simplified similar problem [23] or design a second algorithm [24]. The choice of the initial guess is, thus, a compromise between these two elements.
Solver used to solve a NLP are generally based on gradient method and on Interior Point Method [25].

2.1.3 Reactive Behaviors

Optimization planners guarantee very good performances when the trajectory is perfectly followed, thanks to the action of the controller. On the contrary, unexpected obstacles, tracking errors or irregular terrains can be considered as events that cause changes in the trajectory. Starting from this idea *Reactive Behaviours planners* have been developed. While optimization-based trajectory planning requires good knowledge of the environment, reactive behaviours-based one foresees techniques to manage the unexpected events already mentioned before. Command input are computed based on values obtained by robot sensors.

2.1.4 Machine Learning

Widely used nowadays, the terms "*Machine Learning*" have been used for the first time by Arthur Lee Samuel in 1959, while a first rigorous definition has been provided only in 1997 by Tom Mitchell [26].
DEF1: *A computer program is said to* **learn** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*
As it can be understood by the definition, a Machine Learning Process is a process which does not receive as input a mathematical model, but which computes the output starting from a set of data. The higher the number of data (experience E in the definition), the better are the performances. It is thus evident that the *training* of a Machine Learning Algorithm, *i.e.* the design of the input data, is a very

important element of the algorithm itself. Two main different implementations are possible [27]:

- if couples of input-output data are provided, the algorithm is said *Supervised*. Starting from them, the algorithm has to learn the mathematical model to compute the output for the new data.
- if only a set of input data is provided, the algorithm is said *Unsupervised*. Since no output data are available, unsupervised algorithms can be used in completely unknown situations.

In the recent years, improvements on the Machine Learning Theory have made it possible to achieve a widespread usage of the above mentioned algorithms in many fields of the scientific research in robotics.
In the following, the two mostly used algorithms in trajectory planning for legged robots are introduced.

Deep Learning

The development of Deep Learning (DL) is due to the need to use Machine Learning into complex situations, in which more than one action is required. An algorithm of DL is composed by a certain number of layers, integrating in the so called *Neural Network* [28], see Fig.2.4. Layers are divided in three categories:

- first layer is called *input*.
- the intermediate layers are called *hidden*.
- last layer is called *output*.

Every hidden layer is responsible for a particular action. Lecun *et al.* [29] proposed an example which well explains how an algorithm of DL works: an image is an array of pixel values; the first layer recognizes the presence or absence of edges in the image. The second layer typically detects motifs, while the third layer assembles them into parts of familiar objects. The number of layers, *i.e.* the depth of the network, is a choice of the designer, while the layers are trained with only one general training algorithm.
For what concerns trajectory planning, some examples of DL-based trajectory planning are proposed, in order to highlight the importance of this approach.
Wei *et al.* [30] propose a DL algorithm which computes a path from the robot's current location to the goal, starting from a 2D map of the environment. [31] uses two DL algorithms: the former is an encoder network, *i.e.* a network which translates the point cloud map of the environment into a feature space, the latter

is a 9-layer DL which computes a collision-free trajectories into the feature space. An interesting usage of DL algorithms is to integrate them with optimization-based planners, in order to obtain more robust trajectories.
For example in [32], after the computation of the elevation map of the terrain, certain points are discarded, such as foothold out of the workspace of the robot, threshold on the terrain roughness, proximity to already discarded points, frontal or leg collision. A set of feasible points is obtained in this way. A Neural Network is used to adapt the landing position, which is the closest feasible point to the footholds obtained by the trajectory planner.

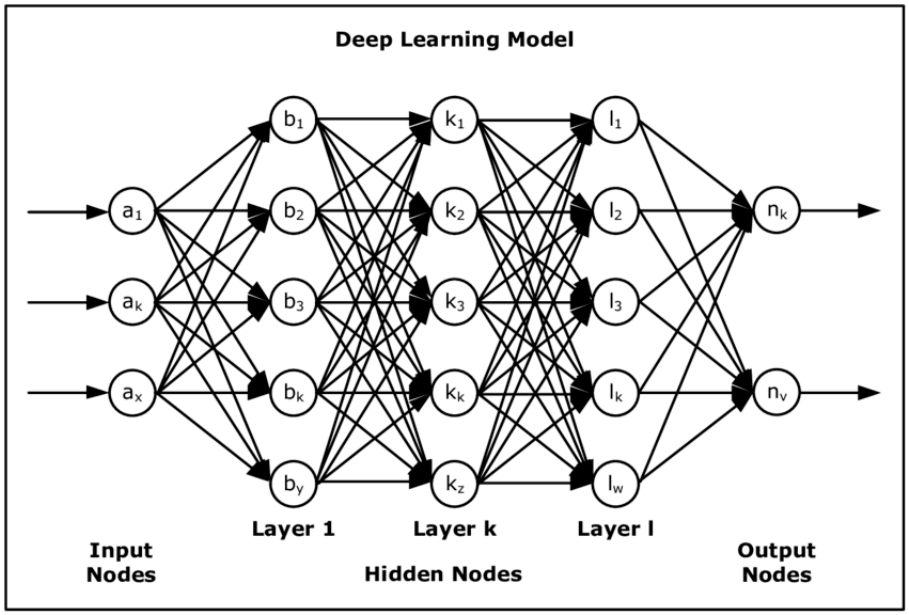

Figure 2.4 – 5-layer DL Neural Network [2]

Reinforcement Learning

Reinforcement Learning (RL) includes all the algorithms in which an agent takes decisions to maximize a cumulative reward and to improve the learning efficiency. The main characteristic of a RL algorithm is that an agent has to apply the best already tested actions (*exploitation*), but in order to do that, it has to find new actions (*exploration*) [33]. The exploration can be dangerous, because in the meanwhile the robot could takes catastrophic actions, *e.g.* it could fail. Singh [34] *et al.* have proposed an algorithm in which the set of actions that the RL agent can explore is restricted to the acceptable ones. Other examples of RL-based trajectory planner can be found in [35],

2.2 Stability Criteria

A *Stability Criterion* is the metric which is used to guarantee that the robot will not fall down during the motion. According to the type of terrain, more precise criteria have to be used.

2.2.1 Center of Pressure-Zero Moment Point

The easiest approach is to exploit the Center of Pressure (CoP), which is defined as the point on the ground with respect to which the horizontal momenta of the contact forces are null, [3]. It can be considered as the point of application of the resultant ground reaction forces. Starting from its definition, it is easy to understand that the CoP depends on the value of contact forces and foot position, thus it varies during the motion.

From a mathematical point of view, it is stated that the CoP is the point such

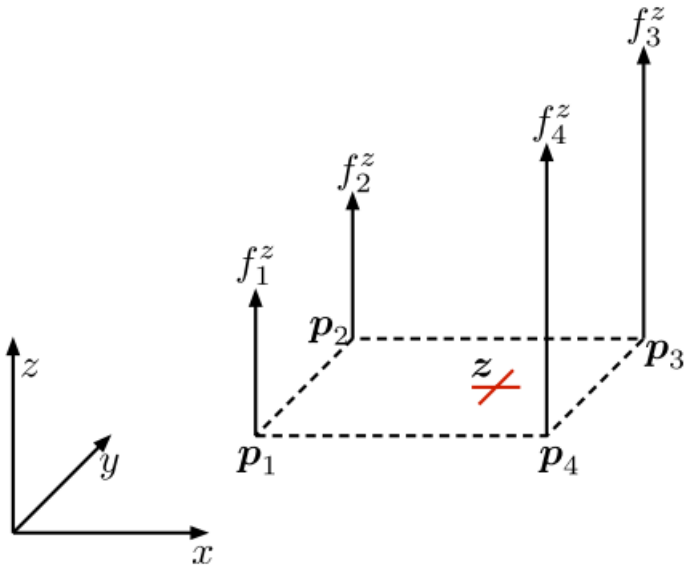

Figure 2.5 – Graphical representation of the CoP stability criterion [3]. Red cross corresponds to CoP. The dashed polygon is the convex hull of the four footholds p_i

that:

$$[\sum_i (\mathbf{p}_i - \mathbf{z}) \times \mathbf{f}_i]^{x,y} = \sum_i (\mathbf{p}_i^{x,y} - \mathbf{z}^{x,y}) \mathbf{f}_i^z = 0 \quad for = 1, \ldots, n_i \tag{2.2}$$

where:

- $\boldsymbol{z} \in R^3 = \text{CoP}$

- $\boldsymbol{p}_i \in R^3$ = foot position (authors assume that $\boldsymbol{p}_i^z$=0 on flat ground)
- $\boldsymbol{f}_i \in R^3$ = contact force
- n_i= number of feet of the robot.

Eq. 2.2 is valid only in case of flat terrains, in which all the contact are on the same plane. In presence of a non-flat terrain, it is possible to define only CoP for every contact surface independently, but a unique CoP cannot be determined.
Similarly to the CoP, it is possible to define the Zero Moment Point (ZMP):
DEF2: *" The ZMP is the point on the ground where the tipping moment, due to gravity and inertia forces, equals zero. The tipping moment being defined as the component of the moment that is tangential to the surface"*[36].
Sardain *et al.* [36] have mathematically demonstrated the coincidence between CoP and ZMP, so in the literature they are considered as the same thing.
Stability is guaranteed if the CoP/ZMP is inside the support polygon, *i.e.* the convex hull of the contact points $\mathbf{p}_i$ [3], [37], [38], [39].
DEF3: *"The convex hull of a set P of points is the enclosing convex polygon that contains P with smallest area"* [40].
Due to this limitation, CoP has to be generalized for uneven terrains.

2.2.2 Contact Wrench Cone

Looking for a stability criterion with a general validity, it is stated that stability can be assumed if the sum of gravity and inertial wrench (six-dimensional vector composed of forces and torques [41], [42]) of the CoM is inside the polyhedral convex cone of contact wrench robot-environment, called Contact Wrench Cone (CWC) [43].
The components of the resulting wrench (gravity + inertia) are:

$$\mathbf{f}_g = M(\mathbf{g} - \ddot{\mathbf{p}}_g) \tag{2.3}$$

$$\boldsymbol{\tau}_g = \mathbf{p}_g \times M(\mathbf{g} - \ddot{\mathbf{p}}_g) - \dot{\boldsymbol{L}} \tag{2.4}$$

$$\dot{\boldsymbol{L}} = \mathbf{I}\boldsymbol{\alpha} \tag{2.5}$$

with:

- M $\in \mathbb{R}$= total mass of the robot
- $\mathbf{g} \in R^3$=[0 0 -g]^T
- $\mathbf{p}_g \in R^3$= position of the center of gravity of the robot

- $\mathbf{L} \in R^3$ = angular momentum with respect to the center of gravity.
- $I \in R^{3\times3}$ = inertia matrix.
- $\boldsymbol{\alpha} \in R^3$=angular acceleration.

CWC can be considered as the admissible set of the total contact wrench, which is computed by summing up the individual contact wrench cone (ICWC) at each contact location. The most precise formulation of CWC foresees the Minkowsky sum of the ICWCs. Taking two cones A and B (identified by the set of their vertices), Minkowsky sum consists in summing every vertex of A with every vertex of B and then deleting the internal point of the resultant cone C (which are no more vertices of the cone) [44], see Fig. 2.6.

$$C = A + B = \{c \in R^n; \exists a \in A; \exists b \in B | \quad c = a + b\} \tag{2.6}$$

In case of a gradient-based nonlinear solver which requires the Jacobian of the constraints, Minkowsky sum cannot be used, since it is not differentiable due to the algorithm which reduces the number of vertices.

An alternative to the Minkowsky sum is the computation of the convex combina-

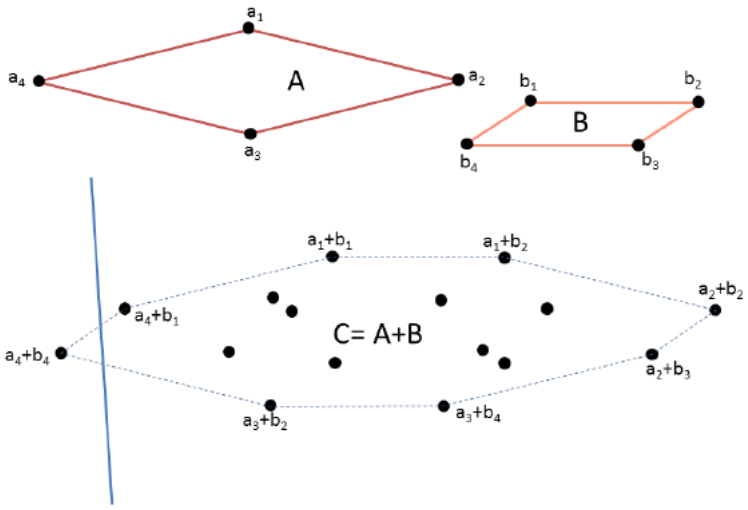

Figure 2.6 – Graphical representation of a Minkowski sum applied to two polygon A and B [4].Internal points of the resulting polygon C are not labelled, since they are not vertices and so discarded by the algorithm.

tion (*i.e.* linear combination with positive or null coefficient) of all the edges of all the ICWCs is considered, Eq. 2.7, 2.8.

$$\mathbf{f}_c = \sum_{k=1}^{K} (e_k^0 \mathbf{n}_k + \sum_{k=1}^{4} e_k^l \mathbf{t}_k^l). \tag{2.7}$$

$$\boldsymbol{\tau}_c = \sum_{k=1}^{K} \mathbf{p}_k \times (e_k^0 \mathbf{n}_k + \sum_{k=1}^{4} e_k^l \mathbf{t}_k^l). \tag{2.8}$$

with:

- $\mathbf{p}_k$ = k-th vertex of the support polygon (foot position).
- K = total number of vertices.
- e_k^l = non negative scalars.
- $\mathbf{t}_k^l \in \mathbb{R}^3$ = unit vectors tangent to $\mathbf{p}_k$
- $\mathbf{n}_k^l \in \mathbb{R}^3$=unit normal vector at $\mathbf{p}_k$ pointed to the robot.

The convex combination, however, is only suitable for the solution of feasibility problems, as it does not allow to estimate the stability margin (*i.e.* the distance between centroidal wrench and the CWC). Further details on the computation of the CWC are beyond the aim of this thesis.
With this formulation, the criterion is valid also for irregular terrains.
CWC has been widely exploited. Barthelemy *et al.* [45] have proposed a metric

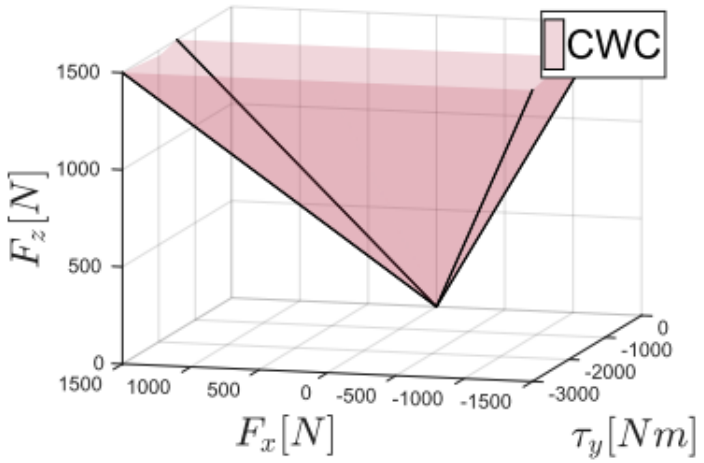

Figure 2.7 – Graphical representation of a Contact Wrench Cone, [4]. F_y, τ_x, τ_z are assumed to be zero, such that the cone can be graphically represented.

to quantify the robustness of the state of the robot, *i.e.* the maximum external force/torque (wrench) disturbance which the robot can resist without breaking static contact. The so called Contact Wrench Cone Margin has been used by [46] in a non convex formulation and by [47] in a convex one. CWC has been exploited in [4] to find a more precise description of the robot capability. Introducing thw Feasible Wrench Polytope (FWP), which is the intersection between the CWC

and the set of all the wrenches that a robot can generate considering its actuation limits (Actuation Wrench Polytope), Orsolino *et al.* stated that a robot is in a stable configuration if its aggregated wrench, as given by Eq. 2.3 and Eq. 2.4 , is inside the FWP.

Chapter 3

Trajectory Optimization for Walking Robots (TOWR)

This section is dedicated to introduce the library named TOWR (*Trajectory Optimization for Walking Robot*), developed by Alexander Winkler. This library is the result of the trajectory planner introduced in [48]. The formulated nonlinear optimization problem is a trajectory planner, based on the Single Rigid Body Dynamics (SRBD) , see section 2.1.2. Due to the non linearity of the problem, an Interior Point method [25] solver is used, implemented in the IPOPT library [49]. Default implementation of TOWR does not foresees a cost function $f(x)$ to minimize, but only a set of constraints to satisfy.
All the implementation details presented in this Chapter are due to the choice of the authors of the library and are not the only possibility.
The planner receives as input:

- initial position and orientation of the CoM;
- goal position and orientation of the CoM;
- total duration of the motion;
- gait, *i.e.* the duration of stance (foot in contact with the ground) and swing (foot in air) phases for each foot;

and is involved in finding values for:

- CoM linear position and velocity, orientation and angular velocity (Sec.3.1.1);
- feet linear position and velocity (Sec. 3.1.2);
- contact forces and first derivative (Sec. 3.1.3);

which satisfy the following constraints:

- Dynamic Constraints (Sec. 3.2.1);
- Kinematic Constraints (Sec. 3.2.2);
- Terrain Constraints (Sec. 3.2.3);
- Spline Acceleration Constraints (Sec. 3.2.4);
- Force Constraint (Sec. 3.2.5);
- Swing Constraint (Sec. 3.2.6);
- Gait Optimization (Sec. 3.2.7).

In order to improve the computational speed of the planner, a solution is considered feasible if the sum of the quantity of violation of each constraint is below a certain threshold. The lower the threshold, the better is the solution and, however, the higher is the computational effort.
Due to the nonlinearity of the problem, an Interior Point method [25] solver is used, implemented in the IPOPT library [49]. For what concerns initial guess, see Chapter 2, a linear interpolation between initial and the desired final state of the robot is performed. This can be done analytically and it, therefore, does not increase the computational load of the formulation.
From this moment on, all the quantities are expressed with respected to a fixed inertial World frame. Another reference frame that will be used is the CoM frame, which is located on the robot's CoM and has the same orientation of the World frame. The transformation between World frame and CoM frame is determined by a translation, correspondent to the CoM position in the World frame and a rotation due to the orientation of the trunk of the robot.

3.1 Minimal Parametrization of the Locomotion Problem

In this section a thorough description of the variables and parameters used by TOWR to describe a generic locomotion problem on rough terrain is provided.

3.1.1 CoM Parametrization

As already said, SRBD model considers the full motion projected in its CoM. Another important point for a robot is its *base*, *i.e.* the geometric center of the

trunk of the robot. Due to the presence of the legs and of some objects mounted on the trunk, CoM and Base are not equal. SRBD model considers only the CoM, assuming that the two points coincide. This assumption is well suited for robots with legs whose mass is negligible compared to the trunk's weight,. Instead, it determines a big inaccuracy of the model in the other cases.
The most relevant physical quantities used to describe the spatial motion of the CoM are the following:

- $\boldsymbol{r} \in R^3$: position of the CoM in the world frame.
- $\dot{\boldsymbol{r}} \in R^3$: derivative of the position of the CoM.
- $\boldsymbol{\theta} \in R^3$: orientation of the CoM, using Euler Angles ZYX.
- $\dot{\boldsymbol{\theta}} \in R^3$: derivative of the orientation of the CoM.

The optimizer discretizes these variables at constant time intervals dt. From this moment on, the discrete values are called *nodes*. Every dimension (x,y,z) of two consecutive nodes of $\boldsymbol{r}$ and $\boldsymbol{\theta}$ are interpolated with a third order polynomial - according to the Hermite Parametrization - in order to build a mathematical function, which is called *spline*. The generic third order polynomial $x(t)$ is equal to:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad with \quad 0 < t < \Delta T_i \tag{3.1}$$

where:

$$\begin{aligned} a_0 &= x_0 \\ a_1 &= \dot{x}_0 \\ a_2 &= -\Delta T_i^{-2}[3(x_0 - x_1) + \Delta T_i(2\dot{x}_0 + \dot{x}_1)] \\ a_3 &= \Delta T_i^{-3}[2(x_0 - x_1) + \Delta T_i(\dot{x}_0 + \dot{x}_1)] \end{aligned} \tag{3.2}$$

where:

- x_0 is the initial value in the i-th interval.
- x_1 is the final value in the i-th interval.
- $\dot{x}_{0,1}$ is the value of the first derivative.
- ΔT_i is the duration of the interval between the two nodes that have to be interpolated.

The Hermite Parametrization assures that the spline is continuous since, assuming that $x_i(t)$ and $x_{i+1}(t)$ are two consecutive polynomials, it can be proven that:

$$x_i(\Delta T_i) = x_{1,i}$$

and consequently:

$$x_{i+i}(0) = a_{0,i+1} = x_{0,i+1} = x_{1,i} = x_i(\Delta T_i)$$

The choice of using a third order polynomial is motivated by the fact that a higher order would require more boundary informations, *e.g.* the value of the acceleration in case of fifth order polynomial. The formulation would become more complex and, consequently, the computational time would increase. Accelerations have to be properly managed into a constraint of the formulation. The value of $\boldsymbol{r}$ and $\boldsymbol{\theta}$ at the generic time T can be found from the spline, finding at which interval T belongs and then using Eq. 3.1 at the correspondent polynomial. In the same way first and second derivative can be found deriving Eq. 3.1:

$$\dot{x}(t) = a_1 + 2a_2t + 3a_3t^2, \quad \text{with} \quad 0 < t < \Delta T_i \tag{3.3}$$

$$\ddot{x}(t) = 2a_2t + 6a_3t, \quad \text{with} \quad 0 < t < \Delta T_i \tag{3.4}$$

It is worth highlighting that while Eq. 3.3 and Eq. 3.4 applied to $\boldsymbol{r}$ coincide with linear velocity and acceleration, in case of $\boldsymbol{\theta}$ they do not represent angular velocity and acceleration.
Calling $\boldsymbol{\omega} \in R^3$ the angular velocity it results [50]:

$$\boldsymbol{\omega} = \boldsymbol{M} \begin{bmatrix} \dot{\gamma} \\ \dot{\psi} \\ \dot{\phi} \end{bmatrix} \tag{3.5}$$

with:

$$\boldsymbol{M} = \begin{bmatrix} 0 & -sin(\gamma) & cos(\psi)cos(\gamma) \\ 0 & cos(\gamma) & cos(\phi)sin(\gamma) \\ 1 & 0 & -sin(\psi) \end{bmatrix} \tag{3.6}$$

For what concerns angular acceleration $\boldsymbol{\alpha} \in R^3$, it can be computed deriving Eq.3.5:

$$\boldsymbol{\alpha} = \boldsymbol{M} \begin{bmatrix} \ddot{\gamma} \\ \ddot{\psi} \\ \ddot{\phi} \end{bmatrix} + \dot{\boldsymbol{M}} \begin{bmatrix} \dot{\gamma} \\ \dot{\psi} \\ \dot{\phi} \end{bmatrix} \tag{3.7}$$

with:

$$\dot{\boldsymbol{M}} = \begin{bmatrix} 0 & -cos(\gamma)\dot{\gamma} & -sin(\psi)cos(\gamma)\dot{\psi} - cos(\psi)sin(\gamma)\dot{\gamma} \\ 0 & -sin(\gamma)\dot{\gamma} & -sin(\psi)sin(\gamma)\dot{\psi} + cos(\psi)cos(\gamma)\dot{\gamma} \\ 0 & 0 & -cos(\psi)\dot{\psi} \end{bmatrix} \tag{3.8}$$

As initial guess for the optimizer, a straight line between starting and goal state is chosen for $\boldsymbol{r}$ and $\boldsymbol{\theta}$. Initial guesses for $\dot{\boldsymbol{r}}$ and $\dot{\boldsymbol{\theta}}$ are, instead, a constant value for every node ($\dot{\boldsymbol{r}}_i$) equal to the difference between the goal ($\boldsymbol{r}_{goal}$)) and the initial node($\boldsymbol{r}_{in}$), divided by the total time (T_{total}), i.e. the slope of the lines of r and θ.

$$\dot{\boldsymbol{r}}_i = (\boldsymbol{r}_{goal} - \boldsymbol{r}_{in})/T_{total} \quad \text{for } i = 1, \ldots, \text{number of nodes} \tag{3.9}$$

3.1.2 End-Effector Parametrization

End Effector variables are composed by two 3D vectors for each foot:

- $\boldsymbol{p}_i \in R^3$: position of the foot of the robot in the world frame;
- $\dot{\boldsymbol{p}}_i \in R^3$: velocity of the foot of the robot.

From now, unless specified, all the quantities refer to one single leg and we can thus drop the pedex
Considering the gait, $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$ must have a different behaviour in the two phases. Each phase is composed by an initial node, a final node and a certain number of equidistant nodes in the middle. The default number of nodes between to the initial and final nodes of each phase is one, however this number can be increased to improve the reliability of the feet trajectories.

Stance phase

The Stance phase is the interval of time in which the foot is in contact with the terrain; to avoid slippage, the velocity of the foot has to be always null. The initial node of the phase corresponds to the moment of touch down of the foot on the ground, while the final one corresponds to the moment of the lift off of the foot from the ground. The nodes of stance phases are called also *constant nodes*. In order to guarantee that the foothold does not change during the stance phase, $\dot{\boldsymbol{p}}_i$ of a constant node is not optimized, but set to zero. In this way, the point of touch down is optimized and it is kept constant during all the phase. For example, in Fig. 3.1 stance phase happens in the interval $0 - 0.2s$ and $0.42 - 1s$, during which phase both $\boldsymbol{p}_x, \boldsymbol{p}_z$ are constant.

Swing Phase

Stance phase is the interval of time in which the foot is in air. Initial and final values of swing phase are constant nodes, since they correspond respectively to a final and an initial node of a stance phase. Only the middle nodes are *not constant nodes* and they are therefore both optimized in position and velocity. In Fig. 3.1

stance phase is in the time interval $0.2 - 0.42$ s. While x coordinate increases continuously, the z takes on the shape of a bell, with the apex corresponding to the non constant node.
Both stance and swing nodes are joined to obtain a single spline, according to Eq. 3.1. The main difference between the spline for the CoM variables and the one containing the end-effector variables is that, for the former, the discretization time ΔT_i is equal between all the couples of nodes while, for the latter, the discretization time is different for all the nodes.
A goal for $\boldsymbol{p}$ is not provided as an input. Final position of the foot is only a consequence of the required final base position and orientation. To compute the final value, the robot is assumed to be in the nominal configuration while reached the target and so:

$$\boldsymbol{p}_{fin} = \boldsymbol{r}_{fin} + \boldsymbol{R}(\boldsymbol{\theta})\boldsymbol{p}^B_{nom} \tag{3.10}$$

where:

- $\boldsymbol{R}(\boldsymbol{\theta}) \in R^{3\times3}$ = rotation matrix between CoM frame and World frame.
- $\boldsymbol{p}^B_{nom} \in R^3$ = nominal coordinate of the foot, expressed in the CoM frame.

Once $\boldsymbol{p}_{fin}$ has been computed, the initial guess for the optimized nodes can be chosen as the linear interpolation between initial and final node Initial guess for velocity is a constant trajectory equal to difference between the two values divided by the total duration of the motion.

3.1.3 Parametrization of the contact forces

The force variables in TOWR represent the contact forces between ground and foot and are composed by two 3D vectors for each foot:

- $\boldsymbol{f}$: value of the contact force;
- $\dot{\boldsymbol{f}}$: first derivative of the force;

The structure of the Force variables corresponds to the one of End-Effector variables, presented in Sec. 3.1.2. In this case, the default value of the force nodes between the initial and the final nodes of each phase has been chosen to be equal to two. This number, however, can be changed in order to modify the amount of feasibility of the trajectories of the contact forces.

Stance Phase

Since, during the stance phase, the foot is in contact with the terrain, a force has to be applied in order to not break the contact. In this case, stance nodes

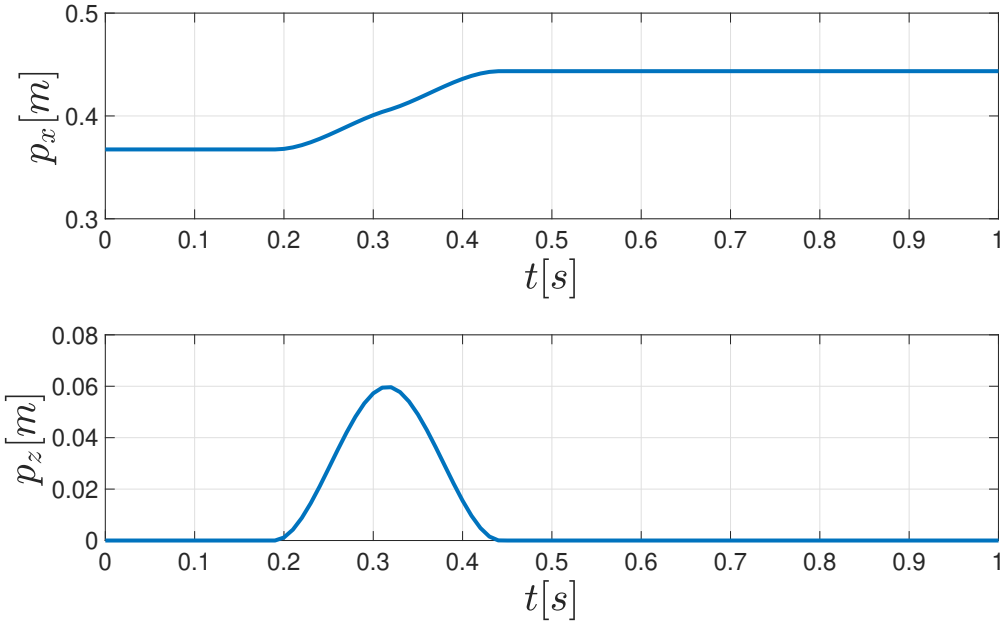

Figure 3.1 – x and z coordinate of left front foot position. These trajectories correspond to the first second of a $20cm$ walk in 2.4s with one cycle of crawl gait. In the interval $1 - 2.4s$ the values remain constant.

are the non constant nodes and, thus, the force and first derivative have to be optimized. Also in this case touchdown and lift-off are considered constant node and, therefore, they are swing ones.

Swing Phase

During the swing phase, the foot does not touch the ground, so it cannot apply any force. For this reason the force nodes during the swing phase are set to be constant nodes (i.e. they are not optimized) and the values of the force and its first derivative are set to zero.
Also in this case, Eq. 3.1 is used to build a third-order spline between the optimization nodes. Like for End-Effector variables, the discretization time is not constant.
As initial guess x and y coordinates of $\boldsymbol{f}_i$ and all the coordinates of $\dot{\boldsymbol{f}}$ are assumed to be constant and equal to zero, while z coordinate of $\boldsymbol{f}_i$, called ($\boldsymbol{f}_{ig}(z)$) is assumed to be constant and equal to the weight of the robot divided by the number of legs n_l:.

$$f_{ig}(z) = (mg)/n_l$$

with $g \in R = 9.80665\ m/s^2$.

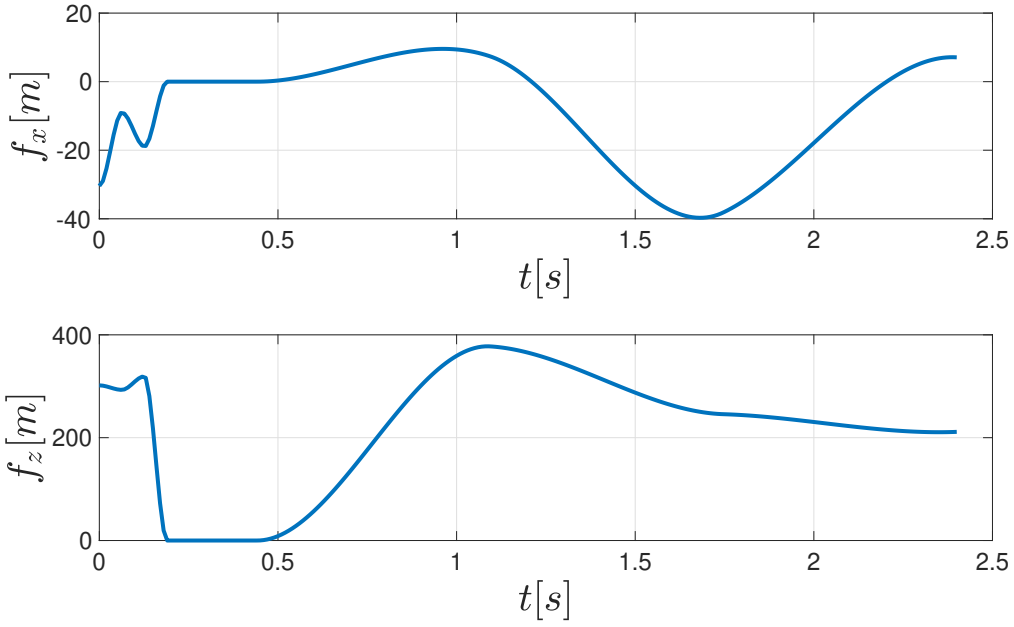

Figure 3.2 – x and z coordinate of the force of the left front foot foot. It corresponds to of a $20cm$ walk in 2.4s with one cycle of crawl gait

3.2 Constraints

As already said, the trajectory planner looks for a feasible solution for the variables introduced in section 3.1, *i.e.* a solution that satisfies all the defined constraints. In this section the default constraints introduced in TOWR are described. The formulation can be easily modified, in order to either implement and add new constraints or neglect some.
Every constraint can be written in the form:

$$b_{min} \leq g(x) < b_{max}$$

where b_{min} and b_{max} are the bounds and $g(x)$ is the constraint function, which can be nonlinear. b_{min}, b_{max} and $g(x)$ are expressed as column vector, whose rows depends on the number of equation of the correspondent constraint. For every equation, its Jacobian with respect to the optimal variables has to be expressed. The presence of the Jacobian imposes that $g(x)$ has to be differentiable.
All the constraints that involve $\boldsymbol{f}$ and $\boldsymbol{p}$ and their derivatives are evaluated for every foot separately.

3.2.1 Dynamic Constraint

As mentioned before, the trajectory planner is based on Single Rigid Body Dynamics, which is governed by the following dynamic equation is:

$$m\ddot{\mathbf{r}} = \sum_{i=1}^{n_i} \mathbf{f}_i(t) - m\mathbf{g} \tag{3.11}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \mathbf{I}\boldsymbol{\omega}(t) = \sum_{i=1}^{4} (\mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t))) \tag{3.12}$$

where $I \in R^{3\times3}$ is the constant inertia matrix.
The Dynamic Constraint is composed of 6 equations and since it is an equality constraint, $b_{min} = b_{max} = 0$.
This constraint is evaluated for every node of CoM nodes (*i.e.* at every fixed time interval defined by the variable *dt*).

3.2.2 Kinematic Constraint

In SRBD formulation, CoM variables and End Effector variables are completely decoupled, but in the real world the foot has to be inside the workspace of the robot, *i.e.* it has to lie within a maximum distance from the CoM, in order to have a feasible configuration for the robot. In other words, imposing that the foot is inside a good approximation of the workspace guarantees that the kinematic limits of the joints of the robot are not overcome. This constraint guarantees that the distance between foot and CoM, lies in a box around the nominal position:

$$\boldsymbol{p}^B_{nom} - \boldsymbol{l}_{min} < \boldsymbol{p}^B < \boldsymbol{p}^B_{nom} + \boldsymbol{l}_{max} \tag{3.13}$$

where:

$$\boldsymbol{p}^B = \boldsymbol{R}(\boldsymbol{\theta})^T(\boldsymbol{p} - \boldsymbol{r}) \tag{3.14}$$

where $\boldsymbol{l}$ is the dimension of the cube and $\boldsymbol{R}(\boldsymbol{\theta})^T$ is the rotation matrix from World frame to CoM frame. The Kinematic Constraint is composed by three equations per foot and it is evaluated every *dt*. If an instant does not correspond to a node in $\boldsymbol{p}$ and $\boldsymbol{f}$, it is computed evaluating the spline at that instant.

3.2.3 Terrain Constraint

This constraint is used to guarantee that:

- if a node of $\boldsymbol{p}$ belongs to stance phase, its z coordinate coincides with the height of the terrain;

- if a node of $\boldsymbol{p}$ belongs to swing phase, its z coordinate has to be positive in order to lift the foot;

It is evaluated for every node of $\boldsymbol{p}$ and is composed of one equation per node of every foot.

3.2.4 Spline Acceleration Constraint

As already said $\boldsymbol{r}, \dot{\boldsymbol{r}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}$ are optimized variables, while second derivatives are computed using the finite differences method. Discontinuity in the second derivative in presence of nodes can cause dangerous jumps and very fast movement of the robot. This constraint guarantees that the final value of the second derivative of a polynomial in the spline is equal to the first value of the following polynomial. In this way the linear and angular accelerations.
For the generic variable x:

$$\ddot{x}_i(\Delta T_i) = \ddot{x_{i+1}}(0) \tag{3.15}$$

This constraint is evaluated at every interval dt for CoM variables and it is composed of six equations equations (three for $\ddot{\boldsymbol{r}}$ and three for $\ddot{\boldsymbol{\theta}}$).

3.2.5 Force Constraint

In order to model the contact between terrain and foot, Coulomb friction has been used. Its equations depend on the friction coefficient μ and on the normal and tangential components to the surface. SRBD does not foresee a true stability

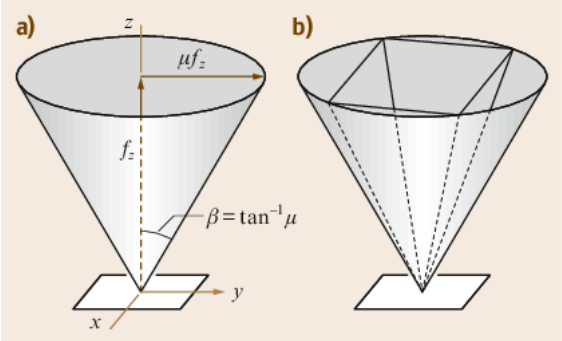

Figure 3.3 – Contact Friction Cone and its linearized pyramid[5]

criterion, but we can state that, if the ground reaction force lies in the cone, the contact between foot and ground is not broken and the robot will not fall during the movement.
The accuracy of this approximation depends on the number of lateral faces of the pyramid. In this case the contact friction cone is linearized by a pyramid. This

allows to approximate the Coulomb friction model with a set of linear constraints rather than a single quadratic constraint:

$$\underline{d} < C\boldsymbol{f} < \bar{d} \tag{3.16}$$

$$C = \begin{bmatrix} (-\mu\boldsymbol{s} + \boldsymbol{t}_1)^T \\ (-\mu\boldsymbol{s} + \boldsymbol{t}_2)^T \\ (\mu\boldsymbol{s} + \boldsymbol{t}_2)^T \\ (\mu\boldsymbol{s} + \boldsymbol{t}_1)^T \end{bmatrix} \quad \underline{d} = \begin{bmatrix} -\infty \\ -\infty \\ 0 \\ 0 \end{bmatrix} \quad \bar{d} = \begin{bmatrix} 0 \\ 0 \\ \infty \\ \infty \end{bmatrix} \tag{3.17}$$

where:

- $\mathbf{s} \in R^3$ is the normal direction to the terrain at the contact point.
- $\mu \in R$ is the friction coefficient.
- $\mathbf{t}_1$, $\mathbf{t}_2 \in R^3$ are the two tangential directions to the terrain at the contact point.

In addition, it must not happen that the ground pulls the feet of the robot, therefore, $\boldsymbol{f}\boldsymbol{s}^T$ must be must be larger than zero. Besides that, $\boldsymbol{f}\boldsymbol{s}^T$ must also be smaller than a predefined maximum value in order to avoid too large torques to the robot's actuators.
The constraint is evaluated in every node of $\boldsymbol{f}$. As a consequence, the Force Constraint, as defined in TOWR, does not prevent the force from either being outside the friction cone or negative between two nodes.

3.2.6 Swing Constraint

The constraint guarantees that x and y coordinate of a node of $\boldsymbol{p}$ in swing is the middle values between the previous and the following node. For $\dot{\boldsymbol{p}}$, instead, x and y coordinate are set equal to one third of the distance between the previous and the following node. This constraint is evaluated for all the swing nodes of $\boldsymbol{p}$ and it is, therefore, not active on the stance nodes.

3.2.7 Gait Optimization

In all the previous parts of this section we have considered the gait as fixed and decided a-priori by the user. In some situations, it would be better to leave to the solver the possibility to change the duration of the phases to obtain a more suitable gait for the required task. For this reason, the extra variable *Phase Duration* and the extra constraint *Total Duration Constraint* can be added.
Phase Duration contains the duration of each phase. A single Phase Duration

variable is associated to every foot of the robot. Maximum and minimum phase duration are provided. While finding the feasible values for Phase Duration variables, the solver has to respect the *Total Duration Constraint*, *i.e.* the sum of the duration of each phase has to be equal to the total duration of the motion. This is necessary to make sure that the total duration of the phases of each feet is the same and the motion ends in the same time instant for all the feet.

3.3 Results

The computational efficiency of the formulation and of the implementation in the library TOWR has been demonstrated in [48]. The library has been tested in simulation on various terrains, robots (monoped, biped, quadruped) and gaits, showing good performances in computational time. In addition, hardware experiments have been performed on flat terrains by Winker *et al.* [48] with the quadruped robot ANYmal [51].

Chapter 4

Experimental Setup

This thesis has been carried out at the Dynamic Legged Systems (DLS) at the Istituto Italiano di Tecnologia (IIT). The aim of this thesis is to improve the capability of motion of legged robots, through model-based optimization. The target platform of this project is the Hydraulically actuated Quadruped robot (HyQ).
In the first section of this Chapter we briefly describe the history of the DLS lab and, in the second section, we give an overview of the HyQ robot.

4.1 The Dynamic Legged Systems (DLS) lab

IIT is a research institute funded in 2003. It is owned by the *Ministero dell'Economia e delle Finanze*, but is administered by the *Fondazione IIT*, according to the Private Law of Italian Civil Code. IIT's total staff is comprised of more than 1700 people from 60 countries. Its aim is to conduct scientific research in the public interest, for the purpose of technological development [52]. It is composed of a *Central Research Laboratories* (a network of four hubs) in Genova, 11 research centres in Italy and two outstations located abroad (at MIT and at Harvard in the US). The headquarter in Genova includes four research domains: Robotics, Nanomaterials, Lifetech and Computational Sciences.
One of the laboratories which deal with the robotics field is the Dynamic Legged Systems (DLS) lab. This research group focuses on the design and control aspects of agile legged robots [53]. Being always focused on the idea that legged robots can be exploited to work in hostile environments, such as disaster areas, DLS' members investigate the field of the control and motion planning to obtain robust architectures against external disturbances. In addition, they are involved in giving to the robot the capability of perceiving the outdoor environment to go through it. Starting from the first prototypes of the legs (2008), DLS has presented

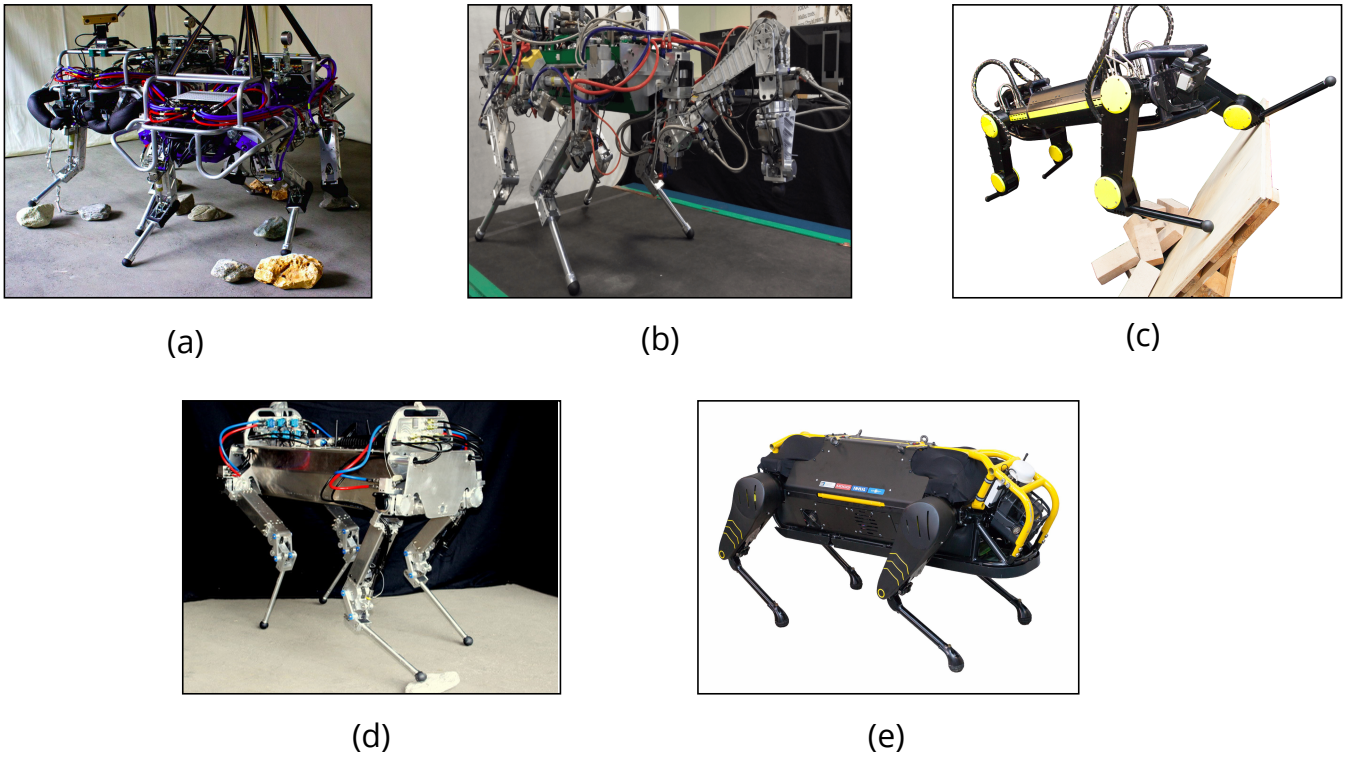

Figure 4.1 – HyQSilver (a), HyQCentaur (b), HyQ2Max (c), HyQMini (d), HyQReal (e)

several robots:

- *HyQ* (2010) [54], Fig. 4.1a: it is the robot we have used for this thesis. We will introduce it in the next section;
- *HyQCentaur* (2015) [55], Fig. 4.1b: HyQ has been provided with an arm in order to enable manipulation and to improve body stabilization. The arm is compact (0.743m fully extended), light-weight (12.5kg), and fast (maximum 4m/s no-load speed at end-effector). The arm has an hydraulic actuation and six torque-controlled degrees of freedom;
- *HyQ2Max* (2015)[56], Fig. 4.1c: HyQ2Max has been developed with the goal to build a stronger, more robust and versatile robot. With respect to HyQ, HyQ2MAX has a far larger joint range and higher joint output torque, without an increase in the weight of the robot. All the sensitive hardware parts (i.e. electronics, sensors, valves and actuators) have been embedded inside a protection shied, which makes the robot more robust to possible falls and to the rain. This allows HyQ2Max to perform dexterous motions such as self-righting (*i.e.* whenever the robot lies on its back consequently to a fall, it is able to turn back on his belly and stand up again);

- *HyQMini* (2015)[57], Fig. 4.1d: it is the smallest robot of the DLS lab. HyQMini has the same leg length as HyQ, but the length of its links is 15% less in flex configuration. It weighs only 35 kg and it is provided with miniature hydraulic actuators;

- *HyQReal* (2019), Fig. 4.1e: it is the newest robot of the DLS lab, presented in the International Conference on Robotics and Automation (ICRA) in May 2019. The robot is endowed with a 48V battery, two hydraulic onboard pumps, two computers and it is the result of a big mechanical redesigning of HyQ. Autonomy and robustness of this robot has been demonstrated pulling a small passenger airplane (Piaggio P180 Avanti) at the Genova Airport.

4.2 The Hydraulically actuated Quadruped (HyQ) robot

In order to verify the validity of the theoretical contribution of this thesis, we have performed experiments and simulations on Hydraulically actuated Quadruped (HyQ) [54], see Fig. 4.2. HyQ has been developed in 2010 by Dr. Claudio Semini and, from that moment on, has been widely exploited for research by his team. Now, two identical versions of this robot exist: HyQGreen, which is the one we have used for this thesis and HyQBlue, which is loaned to other universities or research centers. HyQ is a 90kg-quadruped robots, whose maximum dimensions, corresponding to the fully extended legs are: 1.0m x 0.5m x 0.98m (Length x Width x Height). Legs weigh around $9kg$ and the feet are $2cm$ radius spheres. The robot is provided with hydraulic actuation: the trunk is not actuated, while there are 12 torque-controlled joints, three per leg:

- Hip Abduction-Adduction (HAA);

- Hip Flexion-Extension (HFE);

- Knee Flexion-Extension (KFE);

Every joint is equipped with three sensors: a magnetic absolute encoder for the initialization of the joint positions, a relative optical encoder used for the joint's position control and a force/torque feedback sensor. In addition, an inertial measurement unit (IMU) is mounted on the torso of the robot (3-axis acceleration, 3-axis gyro and compass sensor).
Table 1 resumes the main parameters of HyQ, taken from [58]:

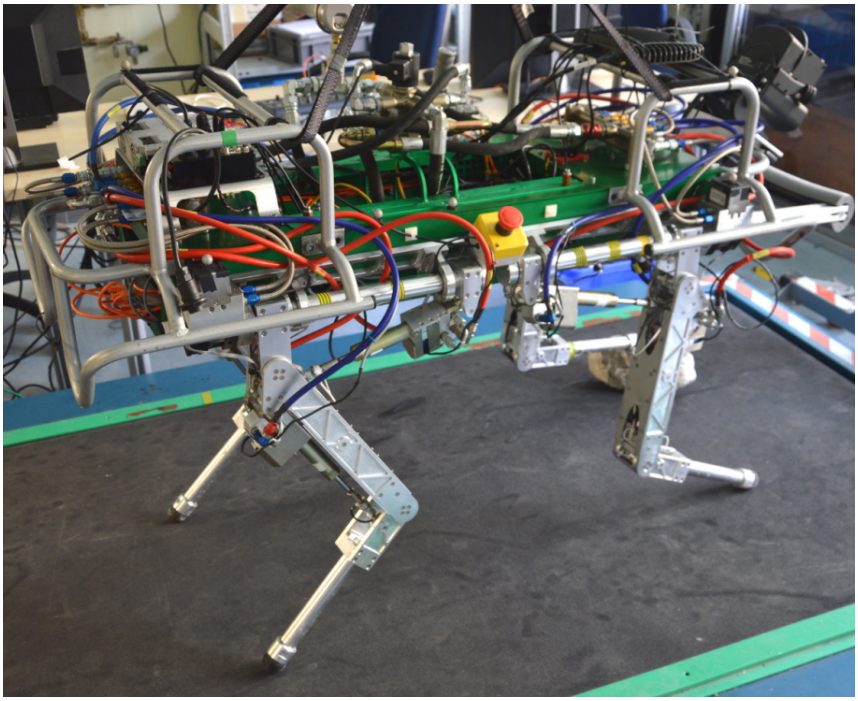

Figure 4.2 – Side view of HyQGreen [6]

Description	**Value**
Robot's mass	$90kg$
Leg's mass	$9kg$
Operating pressure	$160bar$
Dimensions (fully stretched legs)	$1.0m \times 0.5m \times 0.98m$
Leg length	from $0.339m$ to $0.789m$
Foot radius	$2cm$
Number of joints	12
HAA range of motion	-90° +30°
HFE range of motion	-70° +50°
KFE range of motion	-20° +140°
Maximum torque HAA	$120Nm$
Maximum torque HFE	$150Nm$
Maximum torque KFE	$150Nm$

The HyQ robot has so far proven significant navigation capabilities through rough terrains. The objective of this thesis is to endow the robot with improved motion planning skills that will allow the robot to cope with even larger obstacles and more complex terrains.

Chapter 5

Geometry and Actuation Consistency

Chapter 2 has been dedicated to the analysis of the state-of-the-art approach in trajectory planning and stability criterion. In this chapter drawbacks of the SRBD, specially applied to quadruped robots, will be highlighted, presenting possible solutions to improve it. Two main novel approaches will be faced: joint-torques limit and collision with the environment [59].

5.1 Joint-Torque Limits

SRBD only foresees the presence of the CoM and the robot's feet, while joints variables are not considered. It is possible, thus, that a motion planner based on this model finds a trajectory which satisfies all the constraints of the SRBD, but violates feasibility constraints of the actual robot. Such constraints include the joint kinematic limits and the joint-torque limits. In addition, the model is non-convex and these constraints are nonlinear. A solution that violates these constraints has to be considered infeasible and can be dangerous for the robotic hardware unless carefully managed.
The joint-torque limits constraint problem is usually only addressed at the controller level [60, 8]. In order to explicitly consider this limit at the planning stage, Ding *et al.* [61] convexify the nonlinear joint-torque constraint such that it can be added to a Mixed Integer Quadratically Constrained Program (MIQCP). This formulation is suitable for convex optimization and it is thus computationally efficient. The decision to employ a unique outer bounding ellipsoid as an approximation of the force ellipsoids, however, discards the important (configuration dependent) information regarding the relationship between the leg's configuration and shape

of the force ellipsoids (*i.e.* the contact force is constant and it does not depend on the foot position). Another approach consists, instead, in using the force polytopes (see section 5.1.1) to map the joint torques limits into a set of admissible centroidal wrenches [4] or CoM positions [62]. The former approach however, is not suitable for online foothold optimization because of the computational effort related to the Minkowsky sum, while the latter is only suitable for static gaits.

Contribution

In this section we present a novel approximate, robot-agnostic, projection of the joint torque limits into task space. In other words, it represents a novel approach to describe the existing relationship between the leg's configuration and the corresponding maximal contact forces at the end-effector. If the force respects the maximum value of this constraint, we can state that also the joint-torques are inside their limits. Since leg configuration depends on the joints, which are not included in the model, footholds are exploited to express the leg geometry. The result is that the trajectory planner automatically changes either the value of the force to satisfy joint-torque constraint proposed in this thesis or the foot position to find a foothold in which that force respects the constraint. This aspect is usually not captured by standard simplified models, like SRBD and Centroidal Dynamics.

5.1.1 Force Polytopes

A *d-polytope* is the convex hull of a finite number of points in $\mathbb{R}^d$; the 2-polytope is a polygon, the 3-polytope is a polyhedron. It has been stated that a polytope can be expressed also as the intersection of half-planes (Double Description [63]).Assuming that torques limits correspond to an hypercube (cube $\in \mathbb{R}^n$), the *Force Polytopes* $\mathcal{A}$ [64] represents the mapping of torque limits into the task space, exploiting the Jacobian of the legs $\boldsymbol{J}((q))$:

$$-\boldsymbol{\tau}^{lim} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}^{lim} \tag{5.1}$$

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{f} \tag{5.2}$$

where $\boldsymbol{q} \in R^n$ is the vector of joint displacements and $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of the joint-torques, with n equal to the number of joints of the leg of the robot.
Substituting Eq. 5.2 in Eq. 5.1, it results [65]:

$$\mathcal{A} = \{\boldsymbol{f} \in \mathbb{R}^3 \quad | \quad -\boldsymbol{\tau}^{lim} \leq \boldsymbol{J}(q)^T \boldsymbol{f} \leq \boldsymbol{\tau}^{lim}\} \tag{5.3}$$

If torque limits are considered as hypersphere ($\boldsymbol{\tau}\boldsymbol{\tau}^T < 1$), a *Force Ellipsoid* $\mathcal{E}$ is obtained

$$\mathcal{E} = \{\boldsymbol{f} \in \mathbb{R}^3 \quad | \quad \boldsymbol{f}^T \boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{f} \leq 1\} \tag{5.4}$$

The main difference between ellipsoid and force polytopes is that while the former is an approximation, the latter computes a non constant polytopes that is affected by actual configuration of the joints.

5.2 Polytopes Morphing

A trajectory planner that does not respect the torque limits can be dangerous, especially if the robot needs to take on complex configurations in order to negotiate rough terrains such as steps and non-coplanar contacts. Force polytopes can be used to include joint-torque limits, but they require the knowledge of the joint-positions to compute the bounds in which the force at the end effector has to lie. The idea of the new approach that we proposed in [59] is to compute, offline, the polytopes corresponding to a certain number of configuration of the leg, through Eq. 5.3. Since the computation is not done in the formulation of the model, joint displacements $\boldsymbol{q}$ can be used. Once the default configuration polytopes are obtained, it is possible to compute the polytopes corresponding to every configuration as a linear interpolation of the default ones. In this way, actual value of $\boldsymbol{q}$ are not needed to be part of the model.

In Fig. 5.1 we can see a planar example where three force polytopes $\mathcal{A}_k$ (with $k = 1 \ldots 3$) are computed for the three different configurations of the same planar leg. In this example, we decided to pick three polytopes, because it allows us to sample the force polytopes at the configuration of maximum retraction, of maximum extension and at the leg's nominal configuration. In order to map foot position into leg extension we introduce two parameters:

1. l represents the distance between the foot and the hip joint, see Eq. 5.5
2. α represents the angle between the vertical and the line that connects the foot to the hip joint, see Eq. 5.8.

Looking at Fig. 5.1, we can notice that in all the three cases the end-effector is located on the same line with respect to the hip joint. Only the distance l_k changes in the three configurations.

$$l = \sqrt{(p_x^b - h_x^b)^2 + (p_z^b - h_z^b)^2} \tag{5.5}$$

From this moment on, we will use the halfspace description for the polytopes.

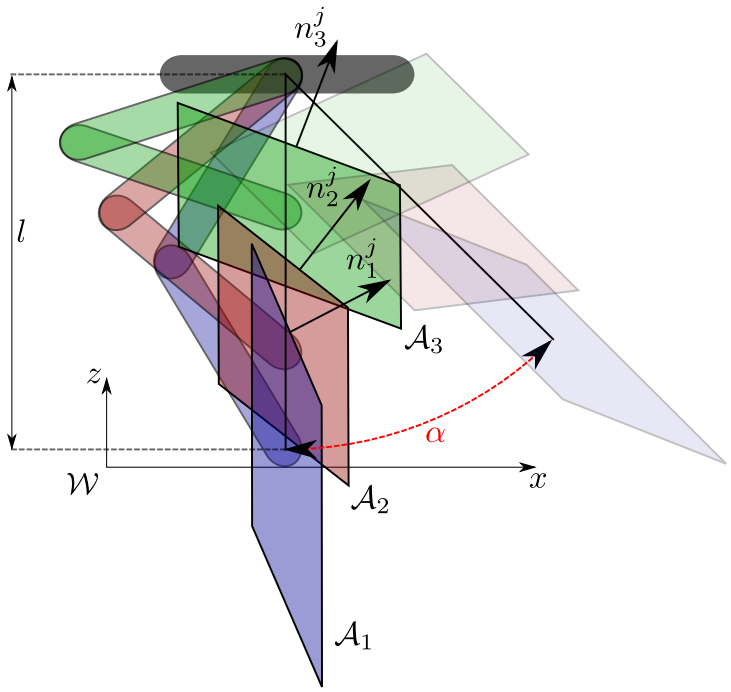

Figure 5.1 – Representation of a planar leg with 2 degree of freedom in three different configurations. For each configuration the corresponding force polytope is shown. The angle α represents the tilt angle of leg (*i.e.* angle between the vertical and the line connecting the foot and hip joint). l is instead the distance between foot and hip joint.

The considered leg has two degrees of freedom and, therefore, all the force/wrench polytopes are made of $2n = 4$ halfspaces. Each halfspace can be represented by its normal unit vector $\boldsymbol{n}_k^j \in R^m$ (with $j = 1 \dots 2n$), to which is associated an angle θ and the known term $d_k^j \in R$. For instance, the generic force polytope $\mathcal{A}_k$ can be expressed as:

$$\mathcal{A}_k = \left\{ \boldsymbol{f} \in R^m \quad | \quad \boldsymbol{A}_k(\boldsymbol{q})\boldsymbol{f} \leq \boldsymbol{d}_k(\boldsymbol{q}) \right\} \tag{5.6}$$

where m is the dimension of the contact force (e.g $m = 2$ in the planar case of Fig. 5.1 with a point foot) and:

$$\boldsymbol{A}_k(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{n}_k^1 \\ \vdots \\ \boldsymbol{n}_k^{2n} \end{bmatrix} \tag{5.7}$$

Eq. 5.6 is another possible formulations of Eq. 5.3.
The rows of the $\boldsymbol{A}_k(\boldsymbol{q})$ matrix correspond to the normalized rows of the leg's Jacobian $\boldsymbol{J}(\boldsymbol{q})$. As we are interested in the way the matrix $\boldsymbol{A}_k(\boldsymbol{q})$ changes with respect to a variation of the foot $\boldsymbol{p}$ in the cartesian space, we should then analyse the quantity $d\boldsymbol{J}(\boldsymbol{q})/d\boldsymbol{p}$. This is, however, robot-specific and goes against the

assumption of SRBD. For this reason, we use the linear interpolation between default pre-defined polytopes. Once the polytopes have been calculated, the approach is robot-agnostic. Using polytopes interpolation, we keep the relationship between foot position and maximal contact forces without the need of explicitly knowing the morphology of the considered robot.

For that purpose, we assume that the transformation between the corresponding halfplanes of two polytopes $\mathcal{A}_a$ and $\mathcal{A}_b$ consists of an homogeneous transformation instead of considering the actual dependency from the robot's configuration manifold $\boldsymbol{q}$. In particular, the homogeneous transformation is composed of a rotation of the leg's tilt angle α and a linear interpolation performed on the term d^j. More in depth, α corresponds to the angle between the vertical and the line that connects the foot to the hip joint (see Fig. 5.1):

$$\alpha = \arctan\left(\frac{p_x^b - h_x^b}{p_z^b - h_z^b}\right) \tag{5.8}$$

where (h_x^b, h_z^b) is the fixed position of the hip joint and (p_x^b, p_z^b) is the foot position with respect to the base frame. $\boldsymbol{R}(\alpha) \in R^{2\times 2}$ rotates the obtained force polytope by the angle α in such a way to align it to the leg. We can therefore consider l and α as the polar coordinates of the foot with respect to the hip joint.

The generic normal unit vector $\boldsymbol{n}^j(l)$ is found as the geodesic average [66] of the two neighboring values l_a and l_b. Exploiting the well known formula of a line passing between two points and considering l as the independent variable and θ as the dependent variable we obtain:

$$\theta = \frac{l - l_a}{l_b - l_a}(\theta_b - \theta_a) + \theta_a \tag{5.9}$$

and thus:

$$\boldsymbol{n}^j = \boldsymbol{R}(\alpha)\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix} \tag{5.10}$$

θ_a and θ_b are the angles corresponding to the two *predefined* normal vectors $\boldsymbol{n}_a = [\cos(\theta_a), \sin(\theta_a)]^T$ and $\boldsymbol{n}_b = [\cos(\theta_b), \sin(\theta_b)]^T$ closest to the value of l (see Fig. 5.2). The generic known term d can be found as a linear interpolation between the values d_a and d_b:

$$d = \frac{l - l_a}{l_b - l_a}(d_b - d_a) + d_a \tag{5.11}$$

Repeating this computation for all the halfplanes of the polytope, the Eq. 5.6 and Eq. 5.8 can be used to compute the polytopes corresponding to the foot position $\boldsymbol{p}$ and the actual leg configuration. Due to the presence of three default

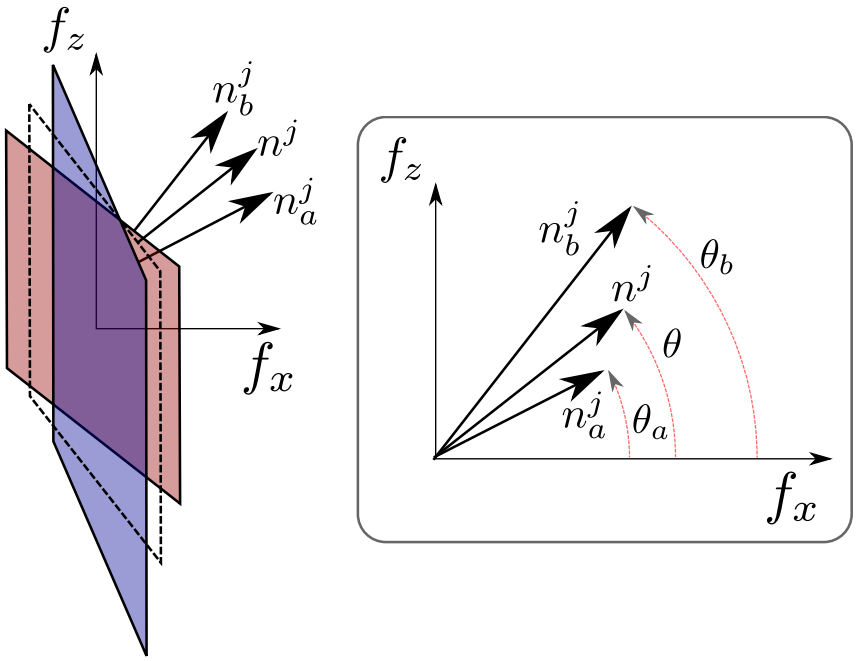

Figure 5.2 – Geodesic average of the unit vectors $\boldsymbol{n}_a$ and $\boldsymbol{n}_b$ normal to the halfplanes of the force polytopes $\mathcal{A}_a$ and $\mathcal{A}_b$. The generic normal vector $\boldsymbol{n}^j$ is obtained by linear interpolation of the angles θ_a and θ_b.

polytopes $\mathcal{A}_k$, with $k = 1, \ldots, 3$, the generic variables $(\cdot)_a$ and $(\cdot)_b$ take on the values corresponding to the two neighboring force polytopes $\mathcal{A}_a$ and $\mathcal{A}_b$:

$$\begin{cases} (\cdot)_a = (\cdot)_1^j, \quad (\cdot)_b = (\cdot)_2^j \quad \text{if} \quad l_1 \leq l \leq l_2 \\ (\cdot)_a = (\cdot)_2^j, \quad (\cdot)_b = (\cdot)_3^j \quad \text{if} \quad l_2 < l \leq l_3 \end{cases} \tag{5.12}$$

Eq. 5.10, 5.9 and 5.11 represent a polytope morphing among the three reference polytopes computed at the three predefined configurations.
Despite the nonlinearity given by the trigonometric terms in Eq. 5.10, this morphing represents a significant simplification because it does not require the knowledge of the leg's Jacobian matrix at the considered leg configuration. In addition this relationship is differentiable and it can, therefore, be used in optimization-based motion planning.

5.3 Collision with the environment

As already said, the SRBD considers the full dynamics of a robot, as its projection in the CoM. Feet are considered as points which interact with the environment, see Eq. 3.11 and Eq. 3.12. Imposing that the z coordinate of the foot in the World frame is always larger than the terrain height can guarantee can guarantee that there is no penetration in the terrain. In case of uneven terrains,

this constraint is often not enough to avoid collisions between the leg of the robot (*i.e.* shin collision) and the environment and proper improvements have to be adopted.

5.3.1 Foot radius

As introduced in Chapter 2, the SRBD considers point feet, while in the real robots feet always have a non-negligible size. The assumption of point feet can be unacceptable for humanoids robot, in which foot shape design is an important part of the construction of the robot. Ouezdou *et al.* [67] compared the results obtained using four different models for the feet: plate, flexible, active and hybrid flexible active. Without going into too much details, they demonstrated that foot shape affects the motion of a human robot. For what concerns quadruped robots, instead, they generally present spherical feet. To solve this uncertainty in the model, the radius of HyQ's spherical feet has been added in the formulation in such a way to discard solutions that may lead to undesired collisions during swing phase. Unsafe footholds, for example, happen when the spherical foot does not step entirely on the terrain (*e.g.* edge of a step). They could cause slippage and thus tracking errors with respect to the reference base trajectory and thus eventually the impossibility to complete the desired task. To overcome this, we force the planner to find a foothold $\boldsymbol{p}$ in which the terrain height is constant at a radius r before and behind the considered foothold along the robot's direction of motion, see Fig. 5.4). The value of r is $2cm$ in the case of the HyQ robot.

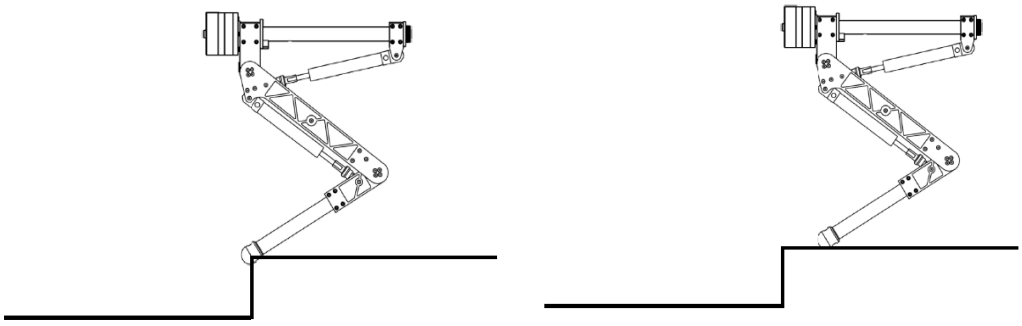

Figure 5.3 – Image on the left represents an unsafe foothold since the foot is very close to the edge and the foot does not step entirely on the terrain. Below image shows a safe foothold.

5.3.2 Shin Collision

Shin Collision refers to the impact between the leg's lower link and the terrain. The leg's volume plays an important role in the execution of successful motion plans, as it may lead to self-collisions or to collisions with the environment if not properly managed. Shin collisions, for example, may occur during a leg's stance phase as a consequence of a wrong choice of the foothold or it may occur during the leg's swing phase. Shin Collision cannot be avoided in SRBD, since the model of the leg is not included in the formulation.
A solution could be the inclusion of the complete geometry of the robot into the model, but this would require a complexity that goes against the aim of the SRBD. A state-of-art approach to the problem of avoiding shin collision consists in looking for a collision-free swing phase, considering the height map of the terrain and the robot configuration [68]. This approach exploits a Machine Learning algorithm, but it cannot be exploited in optimization-based trajectory planning. An alternative is presented in [69], in which the controller is able to detect the point of application of the ground reaction forces from the foot to the shin in case of collision. This method can guarantee safe navigation on challenging terrains, however, it is a pure reactive module which does not increase the robustness of the planner.

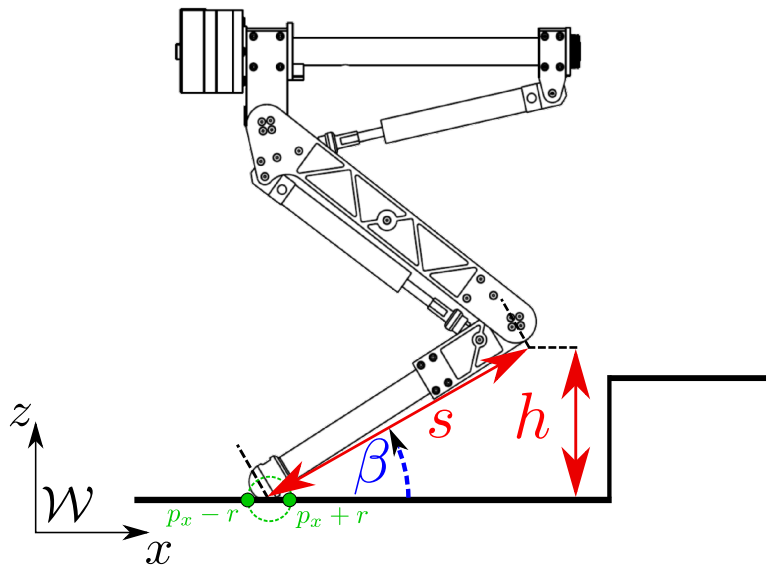

Figure 5.4 – Representation of one single leg of the HyQ robot. h is the knee height, s is the length of the lower link and β is the angle selected to keep the shin from hitting possible obstacles. r is the radius of the foot used to avoid edges and unsafe foothold.

Contribution

In order to avoid such collisions, we provide a simplified kinematic model of the leg to the motion planner [59]. We assume the lower link to be a straight line of length s (distance between the foot's contact point and the knee) with a fixed angle β with respect to the ground. The knee's projection on the ground is equal to $s\cos(\beta)$ and the height of the knee corresponds to $h = s\sin(\beta)$ and it can then be mapped along the direction of motion using the knowledge of the yaw angle γ. Knee collision can thus be avoided if the height of the knee is higher than the terrain on that point. Fig. 5.4 represents the model chosen for the $i-th$ leg in the x-z plane:

$$p_z + s \cdot \sin(\beta) > h_{terrain}(p_{knee}) \tag{5.13}$$

where:

$$\begin{aligned} p_{knee\_x} &= p_x + s\cos(\beta)\cos(\gamma) \\ p_{knee\_y} &= p_y + s\cos(\beta)\sin(\gamma) \end{aligned} \tag{5.14}$$

where $h_{terrain}(p_{knee}) \in \mathbb{R}$ is the height of the terrain in the 2D projection of the knee, computed through Eq. 5.14.

Besides checking for possible knee collisions, we also avoid shin collisions by imposing the constraint on a number of points between the foot and the knee. The lower limb length s is a constant robot parameter while a conservative value of β angle should be selected by looking at the maximum inclination that the leg can take on during a walk.

Chapter 6

Simulation and Hardware Results

In the previous Chapter we have presented a new formulation of SRBD model, which guarantees a robust trajectory for an hardware robot. The new formulation consists of adding two constraints named *Collision Avoidance* and *Force Polytopes*. They are nonlinear constraints, due to the presence of trigonometric terms in Eq. 5.14 (for Collision Avoidance Constraint) and in Eq. 5.10 (for Force Polytopes Constraints), but they are differentiable, so they can be integrated inside the TOWR library.
In order to perform simulations and then experiments on the real robot with a trajectory computed by TOWR, an interface between the trajectory planner and the controller has to be developed. The controller is responsible for computing the reference for the low-level variable (joint position and torques) of the robot and for tracking them. For our experiments, we have used the Hydraulically actuated Quadruped (HyQ) Robot [54] of the Istituto Italiano di Tecnologia of Genova, a 90 kg quadruped robot equipped with 12 torque-controlled joints (3 per leg), powered only by hydraulic actuators.
A thorough description of the controller implemented on HyQ [8] is beyond the aim of this thesis, but some important details will be reported. It is worth noticing that the robot is endowed with a series of sensors, such that all the main relevant quantities for locomotion (such as base velocity or ground reaction forces) can be can be directly or indirectly measured.
Since the communication between controller and trajectory planner is managed by ROS environment, a brief introduction to its main elements is presented in this Chapter. In the last section, we will present the obtained results, both in simulation and on the real hardware robot, of the proposed SRBD-based trajectory

planner.

6.1 Robot Operating System (ROS)

ROS, which stands for *Robot Operating System*, is defined as meta-operating system and provides a software platform to develop robotic applications [7]. The advantage of using ROS is that every code is reusable on different kind of applications and robots. In addition, it provides debugging and visualization tools, like *rqt*, and RVIZ, which does not require robot-specific tools [70]. The smallest unit of a ROS environment is called *node*. It can be assimilated to an executable program, implemented for one single purpose. It guarantees the reusability of ROS. Name, message type and URI addresses are associated to every node. Nodes are joined together to form ROS packages.
As already described, ROS is a communication based program. The main elements involved in the communication are [7]:

- *messages*: they are used by the nodes to exchange informations. A type is associated to every message: primitive types (integer, float, boolean) or user defined types.
- *topics*: they are defined as named buses over which nodes exchange messages [71].
- *publisher*: is a ROS node that writes a particular type of message on a topic.
- *subscriber*: is a ROS node that listens to a topic and when it reads a message, performs an action, generally called *Callback*.
- *service*: it is defined as a synchronous bidirectional communication between the service client that requests a service and the service server that is responsible for responding to requests [72].

6.2 Planner and Controller Integration

Due to its structure, ROS allows both the motion planner (TOWR) and the controller to publish and receive informations. Two already existing topics have been mainly exploited in the experiments that we carried out in this thesis:

1. the first topic is published by the controller and it is responsible of publishing the actual states to the planner (TOWR).
2. the second topic is published by the motion planner (TOWR) as is responsible of publishing the reference trajectory to the controller so that it can track it.

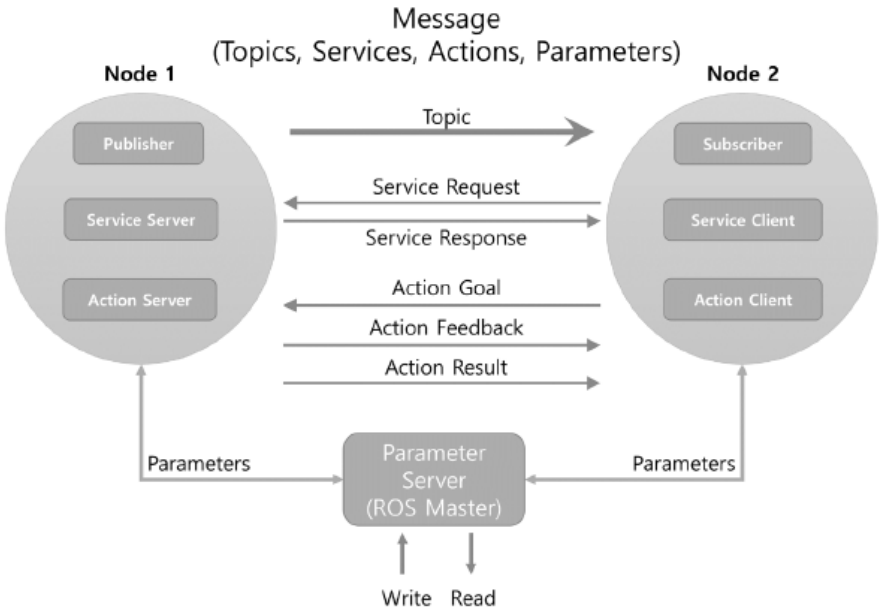

Figure 6.1 – Scheme of the communication between two nodes [7]

6.2.1 Controller to Planner Communication

TOWR has its own default initial condition, but we are interested in computing a trajectory starting from the actual configuration in which the robot is.
HyQ, through what is called *StateEstimator*, computes its actual state pose and twist (linear and angular velocities) of the base and feet position with a frequency of 250 Hz. In TOWR a subscriber to this topic is added. The callback of this subscriber sets TOWR initial condition equal to the ones received from the message. It makes sure that the trajectory optimization problem is always initialized with the most recent state of the robot.

CoM Position

First of all, the controller publishes the position of the geometric center of the trunk of the robot, which is different from the CoM of the robot, see Sec. 3.1.1. Knowing the actual configuration of the joints, the vector of the offset between CoM and base is computed by the controller. Since TOWR finds a trajectory for the robot's CoM, the base variable read from the message has to be corrected including this offset to obtain the initial condition for the robot's CoM of the robot in the world frame.

$$\boldsymbol{r}_{in}^{W} = \boldsymbol{b}_{c}^{W} + \boldsymbol{R}(\boldsymbol{\theta})\boldsymbol{o}^{B}; \tag{6.1}$$

where:

- $\boldsymbol{r_{in}^W} \in R^3$= initial position of the CoM, used by the trajectory planner, expressed in the World frame.
- $\boldsymbol{b_c^W} \in R^3$= actual base position of the robot expressed in the World frame read from controller message.
- $\boldsymbol{R}(\boldsymbol{\theta}) \in R^{3\times3}$ = rotation matrix between Base Frame (*i.e.* a reference frame aligned with CoM whose origin is the geometrical center of the base link) and World frame.
- $\boldsymbol{o^B} \in R^3$ = vector which represents the offset between CoM and Base expressed in the Base frame.

with:

$$\boldsymbol{\theta} = \begin{bmatrix} \phi \\ \psi \\ \gamma \end{bmatrix}$$

being the orientation of the base link and:

$$\boldsymbol{R}(\boldsymbol{\theta}) = \begin{pmatrix} c(\psi)c(\gamma) & c(\gamma)s(\phi)s(\psi) - c(\phi)s(\gamma) & c(\gamma)s(\psi)c(\phi) + s(\phi)s(\gamma) \\ s(\gamma)c(\psi) & s(\gamma)s(\phi)s(y) + c(\phi)c(\gamma) & c(\phi)s(\psi)s(\gamma) - s(\phi)c(\gamma) \\ -s(\psi) & c(\psi)s(\phi) & c(\phi)c(\psi) \end{pmatrix} \quad (6.2)$$

For the sake of simplicity,the letter c is used to indicate the *cos* and letter s replaces the word *sin*.

Foot Position

The controller publishes the feet position in the Base Frame, while TOWR requires their position in the World frame:

$$\boldsymbol{p_{in}^W} = \boldsymbol{R}(\boldsymbol{\theta})\boldsymbol{p_c^B} + \boldsymbol{b_c^W} \quad (6.3)$$

where:

- $\boldsymbol{p_{in}^W} \in R^3$ = initial position of the i-th foot,used by the trajectory planner, expressed in the World frame.
- $\boldsymbol{p_c^B} \in R^3$ = actual position of the i-th foot, expressed in the Base Frame, read from controller message

The robot has spherical feet of $2cm$ radius, so the z coordinate of the foot position in the World Frame is always$2cm$ above the terrain height (during stance phases). As already explained, TOWR, instead, has a model in which the feet are just points and therefore that value could be interpreted as foot non in contact with the terrain. To avoid this, and since the robot starts its movement with the four feet on the terrain, the z coordinate of the feet is set to zero in the initialization of the optimization problem. As a consequence, also the robot's z coordinate in the World frame has to be reduced by the foot radius in order to keep the same relative distance between the base of the robot and its feet.

6.2.2 Planner to Controller Communication

TOWR is used to obtain the trajectory that the robot has to follow to complete a certain task. Outputs of TOWR are splines for the following quantities:

- CoM position and orientation, with their derivative $(\boldsymbol{r}, \dot{\boldsymbol{r}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$;
- foot position and velocity $\boldsymbol{p}, \dot{\boldsymbol{p}}$ for all the four feet;
- contact force and its derivative $\boldsymbol{f}, \dot{\boldsymbol{f}}$ for all the four feet;

TOWR has been implemented to use XPP messages [73], in order to obtain a graphical visualization on the software RVIZ. HyQ's controller uses, instead, a type of messages, which is called *WholeBodyState*. The controller runs at a frequency of 250 Hz, so a WholeBodyState has to be computed every 0.004 s. The conversion between XPP and WholeBodyState is obtained, querying the splines of $\boldsymbol{r}, \boldsymbol{p}, \boldsymbol{f}$ and their derivative, through Eq. 3.1, 3.3, 3.4.
In order to compute the commands required to track the desired trajectory, HyQ's controller has to be provided with (see Sec. 6.2.3):

1. linear position, linear velocity and linear acceleration of the base of the robot in the World Frame;
2. angular position, velocity and acceleration of the base of the robot in the World Frame;
3. position, velocity and acceleration of robot joints;

As already mentioned, joint positions are not provided by TOWR, so Inverse Kinematics (IK) has to be used. The module which computes IK was already available on the DLS software framework and it requires the position, velocity and acceleration of the feet in the Base Frame.

Base Linear Position

Base position can be computed using the inverse equation of Eq. 6.1:

$$\boldsymbol{b}_{des}^{W} = \boldsymbol{r}_{des}^{W} - \boldsymbol{R}(\boldsymbol{\theta}_{des})\boldsymbol{o}^{B} \tag{6.4}$$

where $\boldsymbol{\theta_{des}}$ and $\boldsymbol{p}_{des}^{W}$ are received from the the message published by TOWR.

Base Linear Velocity

In order to compute Base Linear velocity $\dot{\boldsymbol{b}}_{des}^{W}$, starting from $\dot{\boldsymbol{r}}_{des}^{W}$ provided by TOWR, the equation of the kinematics of the rigid body can be exploited:

$$\dot{\boldsymbol{b}}_{des}^{W} = \dot{\boldsymbol{r}}_{des}^{W} + \boldsymbol{\omega}_{des} \times \boldsymbol{o}^{b} \tag{6.5}$$

with $\boldsymbol{\omega}_{des} \in R^3$ which is the desired angular velocity, computed starting form the derivative of Euler Angles received by TOWR and then applying Eq.3.5 and 3.6.

Base Linear Acceleration

As for the base linear velocity, the acceleration $\ddot{\boldsymbol{b}}_{des}^{W}$ is computed through the equations of the kinematics of the rigid body. Since the Coriolis terms $2\omega \times \dot{\boldsymbol{r}}_{des}^{W}$. are small, we have decided to neglect this term:

$$\ddot{\boldsymbol{b}}_{des}^{W} = \ddot{\boldsymbol{r}}_{des}^{W} + \boldsymbol{\omega}_{des} \times (\boldsymbol{\omega}_{des} \times \boldsymbol{o}^{b}) \tag{6.6}$$

Base Angular Quantities

Since CoM frame e Base frame are aligned, base orientation, velocity and acceleration coincides with the CoM ones:

- base orientation $\boldsymbol{\theta}_{des}$ is obtained by TOWR message;
- angular velocity $\boldsymbol{\omega}_{des}$ is computed through Eq. 3.5;
- angular acceleration $\boldsymbol{\alpha}_{des}$ is computed through Eq. 3.7.

Foot Position in Base Frame

In order to compute foot position, the inverse equation of Eq. 6.3 is applied:

$$\boldsymbol{p}_{des}^{B} = \boldsymbol{R}(\boldsymbol{\theta_{des}})^{T}(\boldsymbol{p}_{des}^{W} - \boldsymbol{b}_{des}^{W}) \tag{6.7}$$

where $\boldsymbol{b}_{des}^{W}$ is computed through Eq. 6.4 and the other terms are received by TOWR.

Foot Velocity in Base Frame

TOWR provides velocity of the foot only in the world frame, so the following equations has to be applied:

$$\dot{\boldsymbol{p}}^B_{des} = \boldsymbol{R}(\boldsymbol{\theta_{des}})^T(\dot{\boldsymbol{p}}^W_{des} - \dot{\boldsymbol{b}}^W_{des}) \tag{6.8}$$

where $\boldsymbol{b}^W_{des}$ is computed through Eq. 6.5 and the other terms are received by TOWR.

Foot Acceleration in Base Frame

As done for foot velocity, foot acceleration can be computed through the equations of the kinematics of the rigid body:

$$\ddot{\boldsymbol{p}}^B_{des} = \boldsymbol{R}(\boldsymbol{\theta_{des}})^T(\ddot{\boldsymbol{p}}^W_{des} - \ddot{\boldsymbol{b}}^W_{des}) \tag{6.9}$$

where $\ddot{\boldsymbol{b}}^W_{des}$ is computed through Eq. 6.6 and the other terms are received by TOWR.

6.2.3 Robot's Controller

Robot's controller is composed by two elements:

- a Proportional-Derivative controller for every joint.
- a Whole Body Controller which computes the required torque to balance the Wrench required by the trajectory, see Fig. 6.2.

Joint Level Controller

A Joint Level Proportional-Derivative(PD) controller is used in all the 12 joints of HyQ robot:

$$\boldsymbol{\tau} = K_p(\boldsymbol{q} - \boldsymbol{q}_{des}) + K_d(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{des}) \tag{6.10}$$

where:

- $\boldsymbol{\tau} \in R^m$ = the torque command on the m joints ($m = 12$ for HyQ)
- $K_p, K_d \in R$ = proportional and derivative coefficient of the PD controller.
- $\boldsymbol{q} \in R^m$ = actual value of the joint position.
- $\boldsymbol{q}_{des} \in R^m$ = desired value for the joint position.

K_p and K_d of Eq. 6.10 have to be tuned in order to achieved required performances.

Whole Body Controller

The Whole Body Controller (WBC) solves a QP to obtain the optimal generalized accelerations and contact forces and mapping them to the desired joint torques. It is performed considering the full dynamics of the robot. More details can be found in [8]. For what concerns the aim of this thesis, we have to highlight that the required input for WBC are $\boldsymbol{b}^W, \dot{\boldsymbol{b}}^W_{des}, \ddot{\boldsymbol{b}}^W_{des}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}_{des}, \ddot{\boldsymbol{\theta}}_{des}$.
The orientation $\boldsymbol{\theta}$ is obtained from TOWR, while the other quantities can be computed respectively through Eq.6.4, 6.5, 6.6, 3.5 and 3.7.

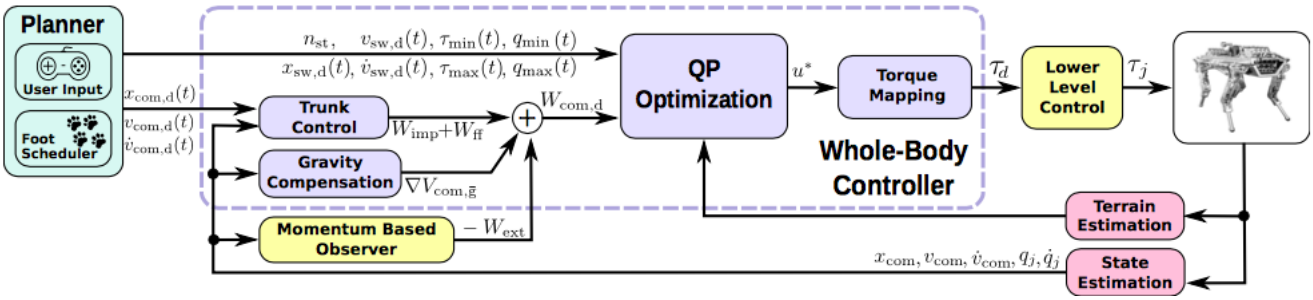

Figure 6.2 – Scheme of the Whole Body Controller of HyQ [8]

6.2.4 Deployability of TOWR trajectory

As we have seen, TOWR finds an optimal solution for a trajectory of the robot, starting from the actual state of the robot. This structure avoids discontinuities between desired and actual state. However, every time the robot has to move, a new trajectory has to be computed. Let's assume that TOWR computes a trajectory of $1m$ on a flat terrain. Once the robot has followed that the trajectory, if we want to perform another meter, we can assume that the previous trajectory can be reused for a new execution. This works well only provided that the final configuration of the robot at the end of the first trajectory is not too different from the initial configuration that was used to optimize the trajectory. Reusing the already computed trajectory, we would avoid to take time on computation, obtaining, therefore, a more efficient motion. In order to do this, two issues have to be faced: how to save the trajectory from TOWR and how to make that trajectory reusable. These two issues are better explained in the next two subsections.

Saving the Trajectory

Until now we assumed that TOWR publishes on a topic and that the subscriber subscribes to it. In addition HyQ framework foresees the possibility to use another module which receives the entire trajectory, *i.e.* a vector of WholeBodyState, and every $0.004s$ publishes the next state to the controller. Thanks to this module, we can use ROS bag [74] files.
Bag files subscribe to a topic and save all the messages published on that topic. Once the trajectory has been saved, the bag file can can be re-published at a later stage. When re-published (or re-played), the trajectory will be read by the controller which will then move the robot accordingly.

Reusability of the Optimal Trajectories

To make the trajectories obtained by TOWR resusable, we assign to let the controller the role of translating the trajectory computed by TOWR with respect to the actual state of the robot (*i.e.* we make sure that the optimal trajectory starts from the actual state rather than the desired CoM position) in the actual state of the robot TOWR with respect to the actual state of the robot (i.e. we make sure that the optimal trajectory starts from the actual state rather than the desired CoM position), without modifying the other properties of the trajectory itself.
Since the first state of the trajectory coincides with the actual state of the robot, see Sec. 6.2.1, subtracting it from all the states of the trajectory we obtain a trajectory that starts from zero. All the elements can be considered as relative displacement from the initial state.
From the controller side, Eq. 6.4 must be updated since the term $\boldsymbol{r}^W_{des}$ is now shifted of the initial state of the trajectory. Updated equation is:

$$\boldsymbol{b}^W_{des} = \boldsymbol{r}^W_{des} + \boldsymbol{r}^W_{act} - \boldsymbol{R}(\boldsymbol{\theta}_{des})\boldsymbol{o}^{\boldsymbol{b}} \tag{6.11}$$

where $\boldsymbol{r}^W_{act}$ is the actual CoM position at the moment when the trajectory has to be performed and it is,therefore, a constant for the entire trajectory.
We have used similar approach also for position of the feet. Starting from Eq. 6.12:

$$\boldsymbol{p}^B_{des} = \boldsymbol{R}(\boldsymbol{\theta_{des}})^T(\boldsymbol{p}^W_{des} + \boldsymbol{p}^W_{act} - \boldsymbol{b}^W_{des}) \tag{6.12}$$

where $\boldsymbol{p}^W_{act}$ is the actual CoM position at the moment the trajectory has to be performed. $\boldsymbol{b}^W_{des}$ is computed with Eq.6.12 and it is, therefore, a constant for the entire trajectory.
The trajectory of velocity and acceleration does not require any manipulation,

since they already start from zero.

6.3 Experimental Results

In this Section we present the validation results that have lead to the successful execution of the optimal trajectories generated by our motion planner on the HyQ robot both in simulation and on the real hardware [59]. For all the simulations and experiments we used an Intel® Core™ i5-4460 CPU @ 3.20GHz × 4 and all the nonlinear optimization problems were solved using an Interior Point method [25] solver, implemented in the IPOPT library [49].

6.3.1 Joint Torque Limits Approximation

The efficiency of the joint-torque limits approximation is demonstrated during a $1m$ walk on a flat terrain for $2.4s$ (three crawl gait cycles). Fig. 6.4 shows the Hip Abduction-Adduction (HAA), Hip Flexion-Extension (HFE) and Knee Flexion-Extension (KFE) joint-torques and the corresponding saturation limits of the HyQ robot.
The plots show the optimal trajectory obtained using the motion planner with (right) and without (left) the force polytope constraints. Our motion planner does not explicitly optimize over joint-torques $\boldsymbol{\tau}$, however it is possible to compute the $\boldsymbol{\tau}$ exploiting the dynamic equation of motion of each single leg:

$$\boldsymbol{\tau} = \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{f} \tag{6.13}$$

where:

- $\boldsymbol{M}$ = leg's inertial matrix.
- $\boldsymbol{c}$ = Coriolis term.
- $\boldsymbol{g}$ = gravity term.

$\boldsymbol{f}$ is the contact force as optimized by the motion planner and $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ can be obtained by inverse kinematics of the foot trajectory in the base frame $\boldsymbol{p}^B$ (assuming a fixed offset between base and robot's CoM).
We can see in the left plots that the desired torques violate the saturation limits of the actuators of the HyQ robot. This is justifiable considering that the baseline motion planner has no information about such saturation values and the only constraints acting on the contact forces are the linearized friction cones. The

Figure 6.3 – HyQ robot stepping up a pallet of $10cm$ in both simulation (Gazebo) and hardware experiment.

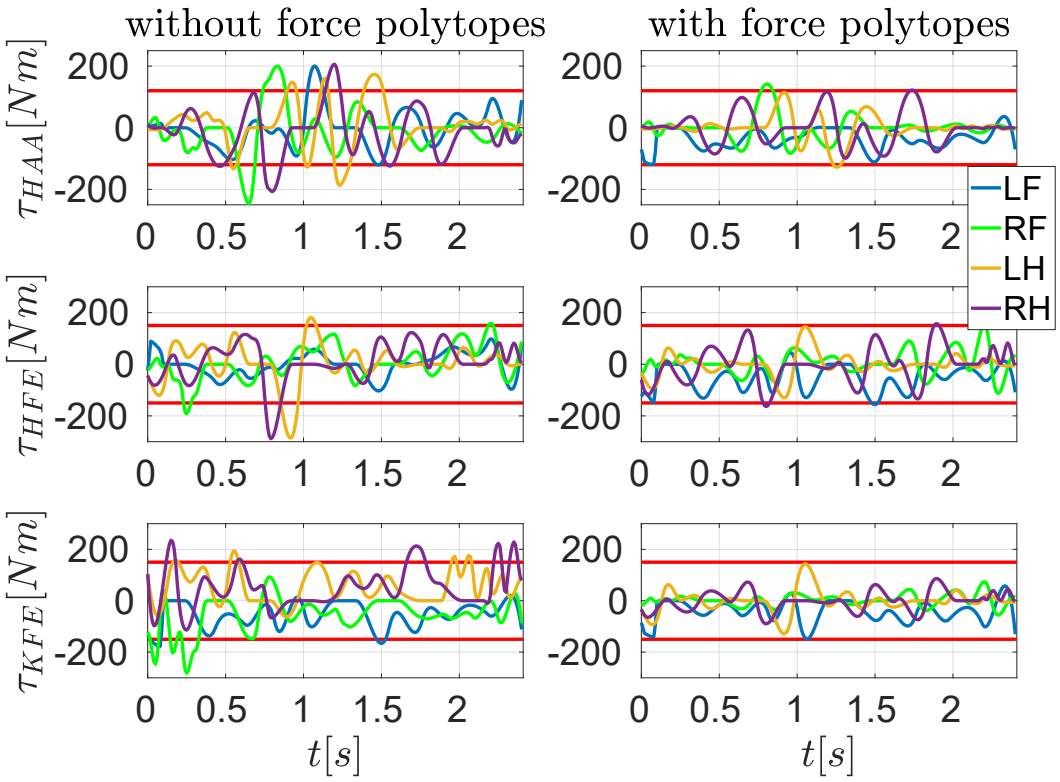

Figure 6.4 – This plot shows the joint torques of the HyQ robot over a $2.4m$ walk on a flat terrain. We can see, in the case where no force polytopes are considered (left plots), that the torques τ_{HAA}, τ_{HFE} and τ_{KFE} violate their limits multiple times during the walk. On the right plots, instead, we can see that the force polytope constraint is able to bias the planner towards a solution that respects all the limits.

plots on the right, instead, do not violate the torque limits of the robot thanks to the force polytope constraint included in the motion planner formulation. This is possible thanks to more extended configuration that the HyQ takes on during the walk and the standing phases. As a matter of fact, a force polytope with a larger

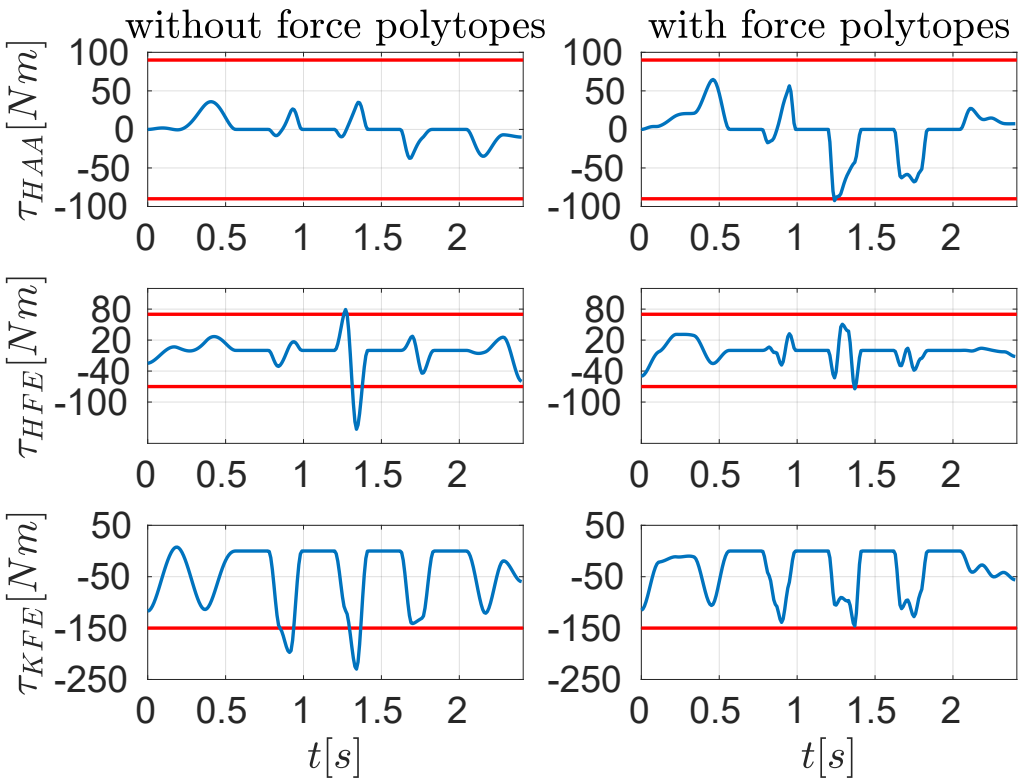

Figure 6.5 – This plot shows the joint torques of the monoped robot over a $2.4s$ motion on a flat terrain. We can see, in the case where no force polytopes are considered (left plots), that the HFE torque τ_{HFE} limit is violated twice during the hop (at $t = 0.9s$ and at $t = 1.3s$). On the right plots, instead, we can see that the force polytope constraint is able to bias the planner towards a solution that respects all the limits.

maximum normal force corresponds to this robot configuration (see Fig. 5.1) [62].

Fig. 6.5 shows similar results for the monoped robot (corresponding to a single leg of HyQ, see Fig. 5.4).

6.3.2 Collision with the Environment Avoidance

Shin collision avoidance and correct choice of the foothold become of paramount importance whenever a robot needs to negotiate rough terrains.
Exploiting the constraint that we described in Chapter 5, HyQ was able to walk for $1m$, performing three cycles of crawl in $11s$ to step onto $15cm$ high pallet in simulation and onto $10cm$ high pallet on the hardware robot. Fig. 6.6 shows the base position x (continuous line) and tracking error (dashed line) with respect to the desired trajectory computed by the planner in three following different versions:

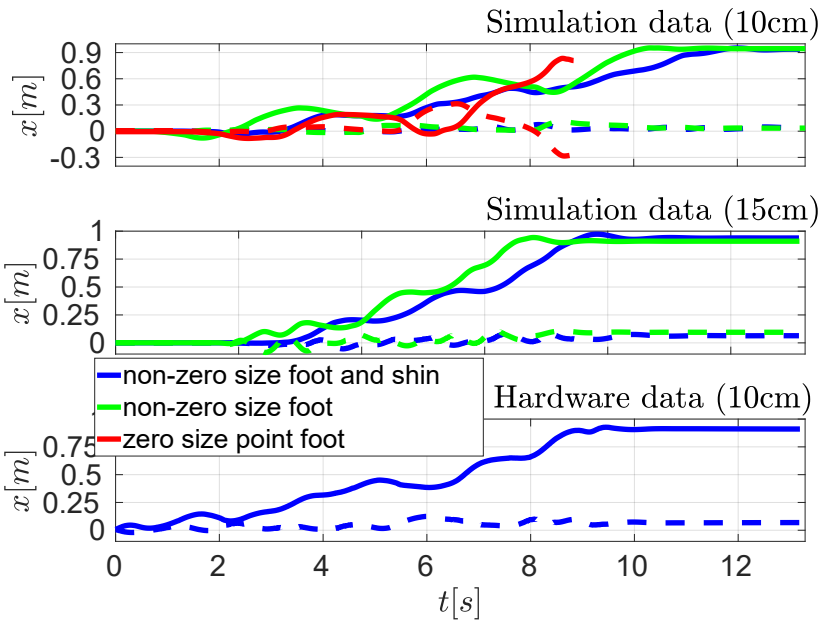

Figure 6.6 – Shin collision planning: tracking performances of simulations with three different terrain constraints for a walk of $1m$ with three crawl cycles. The thick lines represent the base position x while the dashed lines represent the tracking error with respect to the desired trajectory along the same x coordinate.

1) Zero Size Point Foot and No Shin Collision Avoidance (red lines)

This corresponds to the formulation given in [48] and implemented in TOWR. Both shin collision and the foot size are neglected. The algorithm took $\sim 50s$ to find an optimal solution. In the upper plot of Fig. 6.6 we can see that, in this case, the tracking error (dashed line) grows until the experiment is stopped because the robot falls down. In this case, the robot collides with the corner of the edge, due to a non-robust choice of the foothold. We have demonstrated that even a terrain with a relatively low obstacle ($10cm$) cannot be overcome without explicitly considering the feet and leg geometry.

2) Non-Zero Size Point Foot and No Shin Collision Avoidance (green lines)

In this case we enforce in the planner a foot radius r of $2cm$, while we do not include any shin collision avoidance constraint. The computation time increased by 30 % compared to the first scenario ($\sim 70s$) in the case of $10cm$ high pallet. For the $15cm$ high pallet the solver took $105s$ to find an optimal solution. This constraint guarantees the successful navigation in the non flat terrain, but comparing the upper and the middle plot of Fig. 6.6 it can be seen that increasing of the height of the step will also increase the tracking error.

3) Non-Zero Size Point Foot and Shin Collision Avoidance (blue lines)

This version corresponds to the constraint described in Chapter 5. In case of a $10cm$ high pallet, the algorithm took twice as long as the first simulation ($\sim 100s$), while $130s$ were required in case of a $15cm$ high pallet. This increase in computational time is motivated by the increase in the complexity of the model. As robot paramete of Eq. 5.14, the lower link length of HyQ s is $0.3m$. Due to the symmetric design of HyQ we have selected inclination angle β=37° for the hind legs and β=127° for the front legs. The shin collision is thus possible on HyQ only with hind legs when walking up a step and front legs when walking down a step. This aspect is automatically captured by the sign of $sin(\beta)$ and $cos(\beta)$ in the definition of the constraint of Eq. 5.14. For this experiment we selected two points, besides the knee, to be checked against possible collisions. Unlike the previous version of the planner, in this case the tracking error did not increase in the case of higher step thanks to the larger robustness given by he shin collision avoidance constraint. To the best of author's knowledge, the 10cm pallet experiment represents the first time that a hardware implementation of a trajectory planner based on TOWR on a non-flat terrain has been performed.

Chapter 7

Conclusion

In this thesis we have presented two theoretical contributions consisting of two feasibility constraints aimed at increasing the robustness of trajectories that are optimized using the Single Rigid Body Dynamics model. The first constraint focuses on including the joint torque limits constraint and approximates the way these limits are mapped into admissible contact forces depending on the leg's configuration. We have developed the approach of interpolation of default robot's configuration force polytopes. The proposed approximation is robot-independent and is thus suitable for motion planning applications based on simplified robot-agnostic models, such as SRBD or Centroidal Dynamics.
The second constraint, instead, is able to describe and approximate the volume of the robot's legs in such a way to avoid undesired collisions between the lower limbs and the environment.
The experimental contribution of this thesis consists of the integration of TOWR with the controller of the HyQ robot. The obtained results demonstrate that SRBD, alone, is not sufficient for non flat terrains. The consideration of the foot shape is needed to complete the desired locomotion task on an uneven terrain and, additionally, the modelling of the lower link resolves the problem of shin collision and improves the tracking.

7.1 Future works

This thesis could be considered as the starting point for future works. In particular, new possible research scenarios are:

- development of strategies for online replanning (the solver optimizes while the robot walks). In this way the robot could perform a long time horizon walking without stopping to compute the next steps;

- usage of pre-computed feasible solutions for the warm start of every new nonlinear trajectory optimization. The computation time will be drastically reduced, since the initial guess is closer to the optimal solution and a small number of iterations is required;
- inclusion of an explicit stability criterion, such as CWC, in a SRBD-based trajectory planner;
- precise identification of the robot's inertial parameters (robot's Center of Mass and inertia tensors) to improve tracking capability;
- introduction of a numerical metric to understand whether a trajectory is reusable or not.

Appendix A

Computation of Jacobian

A.1 Force polytope's Jacobian

One vertex $\boldsymbol{f}^{lim}$ of the force polytope $\mathcal{A}_i$ the constraint's Jacobian would require the knowledge of the following quantity:

$$\begin{aligned} &\frac{d\boldsymbol{f}^{lim}(IK(\boldsymbol{p}))}{d\boldsymbol{p}} = \frac{d\boldsymbol{f}^{lim}(\boldsymbol{q})}{d\boldsymbol{p}} = \frac{d(\boldsymbol{J}^{-T}(\boldsymbol{q})\boldsymbol{\tau}^{lim})}{d\boldsymbol{p}} \\ &= \frac{d\boldsymbol{J}^{-T}(\boldsymbol{q})}{d\boldsymbol{p}}\boldsymbol{\tau}^{lim} + \frac{d\boldsymbol{\tau}^{lim}}{d\boldsymbol{p}}\boldsymbol{J}^{-T}(\boldsymbol{q}) \end{aligned} \tag{A.1}$$

The above relationship is highly nonlinear because of the trigonometric terms in the leg's Jacobian matrix and it requires the knowledge of the robot's kinematics which is against the SRBD assumption.

A.2 Force polytope's Simplified Jacobian

The simplified force polytope constraint's Jacobian $d\boldsymbol{g}/d\boldsymbol{x}$ with respect to the optimization variables $\boldsymbol{x}$, required by nonlinear optimization solvers based on the interior point method, can be deducted as follows:

$$\frac{d\boldsymbol{g}}{d\boldsymbol{x}} = \frac{d\boldsymbol{A}(\boldsymbol{p})}{d\boldsymbol{x}}\boldsymbol{f} + \frac{d\boldsymbol{f}}{d\boldsymbol{x}}\boldsymbol{A}(\boldsymbol{p}) - \frac{d\boldsymbol{d}(\boldsymbol{p})}{d\boldsymbol{x}} \tag{A.2}$$

where:

$$\frac{d\boldsymbol{A}(\boldsymbol{p})}{d\boldsymbol{x}} = \begin{bmatrix} -\sin(\theta_x) \\ \cos(\theta_x) \end{bmatrix} \cdot \frac{\theta_2 - \theta_1}{p_{2,x} - p_{1,x}} \cdot \frac{d\boldsymbol{p}}{d\boldsymbol{x}} \tag{A.3}$$

and:

$$\frac{d\boldsymbol{d}(\boldsymbol{p})}{d\boldsymbol{x}} = \frac{d_2 - d_1}{p_{2,x} - p_{1,x}} \tag{A.4}$$

We can see that the above relationship does not depend on the robot's kinematics and it can be therefore applied to arbitrary robots. The three polytopes employed for the morphing are, instead, robot specific and they can be computed offline.

Bibliography

[1] O. Arslan and U. Saranli, "Reactive planning and control of planar spring–mass running on rough terrain," *IEEE Transactions on Robotics*, vol. 28, pp. 567–579, 06 2012.

[2] W. Serrano, "Smart internet search with random neural networks," *European Review*, vol. 25, pp. 1–13, 02 2017.

[3] P. B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged systems," in *Springer Handbook of Robotics, 2nd Ed* (B. Siciliano and O. Khatib, eds.), Springer, 2016.

[4] R. Orsolino, M. Focchi, C. Mastalli, H.Dai, D. Caldwell, and C. Semini, "Application of wrench based feasibility analysis to the online trajectory optimization of legged robots," *IEEE Robotics and Automation Letters (RA-L)*, pp. 3363–3370, 2018.

[5] I. Kao, K. Lynch, and J. Burdick, *Contact Modeling and Manipulation*, pp. 647–669. 01 2008.

[6] M. Camurri, *Multisensory State Estimation and Mapping on Dynamic Legged Robots*. PhD thesis, University of Genoa, Italy and Italian Institute of Technology (IIT), 2017.

[7] `http://wiki.ros.org`.

[8] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passivity based whole-body control for quadrupedal locomotion on challenging terrain," *CoRR*, vol. abs/1811.00884, 2018.

[9] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 239–246 vol.1, 2001.

[10] R. Newburgh, *The Pendulum: A Paradigm for the Linear Oscillator*, vol. 13, pp. 37–47. 01 2005.

[11] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Kaneko, F. Kanehiro, and K. Y. Aist, "Biped walking control on uneven ground by linear inverted pendulum tracking," *The 2010 IEEE/RSJ International Conference on Intelligent*

Robots and Systems, vol. 2, pp. 3–6, 2010.

[12] H. Geyer and U. Saranli, *Gait Based on the Spring-Loaded Inverted Pendulum*, pp. 1–25. 01 2017.

[13] G. Antoniak, T. Biswas, N. Cortes, S. Sikdar, C. Chun, and V. Bhandawat, "Spring-loaded inverted pendulum goes through two contraction-extension cycles during the single-support phase of walking," *Biology Open*, vol. 8, no. 6, 2019.

[14] C. Lee and S. Oh, "Development, analysis, and control of series elastic actuator-driven robot leg," *Frontiers in Neurorobotics*, vol. 13, p. 17, 2019.

[15] S. Oh and K. Kong, "Realization of spring loaded inverted pendulum dynamics with a two-link manipulator based on the bio-inspired coordinate system," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 310–315, May 2014.

[16] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal Dynamics of a Humanoid Robot," *Auton. Robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[17] P. M. Wensing and D. E. Orin, "Improved Computation of the Humanoid Centroidal Dynamics and Application for Whole-Body Control," *International Journal of Humanoid Robotics*, vol. 13, no. 1, 2016.

[18] Z. Shiller, *Off-Line and On-Line Trajectory Planning*, vol. 29, pp. 29–62. 02 2015.

[19] J. Reeb and S. Leavengood, "Using the graphical method to solve linear programs," 01 1998.

[20] G. B. Dantzig, "Origins of the simplex method," 1990.

[21] P. Tondel, T. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit mpc solutions," vol. 39, pp. 1199 – 1204 vol.2, 02 2001.

[22] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[23] P. Tsiotras, E. Bakolas, and Y. Zhao, "Initial guess generation for aircraft landing trajectory optimization," 08 2011.

[24] P. Tu, "Fast initial guess estimation for digital image correlation," *CoRR*, vol. abs/1710.04359, 2017.

[25] H. Yamashita, "A global convergent primal-dual interior point method for constrained optimization," *Optimization Methods and Software*, vol. 10, pp. 443–469, 1998.

[26] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[27] S. B. Kotsiantis, "Supervised machine learning: A review of classification

techniques," in *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, (Amsterdam, The Netherlands, The Netherlands), pp. 3–24, IOS Press, 2007.

[28] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, 04 2014.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[30] G. Wei, D. Hus, W. Lee, S. Shen, and K. Subramanian, "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation," 10 2017.

[31] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion Planning Networks," pp. 2118–2124, 2019.

[32] O. A. Villarreal Magaña, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion Through CNNs," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.

[33] J. Johnson, J. Li, and Z. Chen, "Reinforcement learning: An introduction: R.s. sutton, a.g. barto," *Neurocomputing - IJON*, vol. 35, pp. 205–206, 11 2000.

[34] S. P. Singh, A. G. Barto, R. Grupen, and C. Connolly, "Robust reinforcement learning in motion planning," *Advances in neural information processing systems*, p. 655, 1994.

[35] D. Aranibar and P. Alsina, "Reinforcement Learning-Based Path Planning for Autonomous Robots," *EnRI-XXIV Congresso da Sociedade Brasileira de Computaçao*, p. 10, 2004.

[36] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure—zero moment point," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 34, pp. 630 – 637, 10 2004.

[37] F. Romano, D. Pucci, S. Traversaro, and F. Nori, "The Sensitivity of the Static Center of Pressure as a Criterion to assess Balancing Controllers Performance," *CoRR*, vol. abs/1610.0, 2016.

[38] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, pp. 1321–1326 vol.2, May 1998.

[39] A. Goswami, "Postural stability of biped robots and the foot-rotation indicator

(fri) point," *International Journal of Robotic Research - IJRR*, vol. 18, pp. 523–533, 06 1999.

[40] S. Asaeedi, F. Didehvar, and A. Mohades, "Alpha convex hull, a generalization of convex hull," 09 2013.

[41] R. Featherstone, "A Beginner's Guide to 6-D Vectors (Part 1)," *{IEEE} Robot. Automat. Mag.*, vol. 17, no. 3, pp. 83–94, 2010.

[42] R. Featherstone, "A Beginner's Guide to 6-D Vectors (Part 2) [Tutorial]," *{IEEE} Robot. Automat. Mag.*, vol. 17, no. 4, pp. 88–99, 2010.

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

[44] V. Delos and D. Teissandier, "Minkowski sum of HV-polytopes in Rn," in *eprint arXiv:1412.2562*, 2014.

[45] S. Barthélemy and P. Bidaud, *Stability Measure of Postural Dynamic Equilibrium Based on Residual Radius*, pp. 399–407. 01 2008.

[46] S. Caron, Q.-C. Pham, and Y. Nakamura, "Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics," in *RSS*, no. 8, 2015.

[47] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," *IEEE-RAS International Conference on Humanoid Robots*, pp. 579–586, 2016.

[48] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[49] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, mar 2006.

[50] `https://docs.leggedrobotics.com/kindr/cheatsheet_latest.pdf`.

[51] M. Hutter, C. Gehring, D. Jud, A. Lauber, D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. A. Höpflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44, 2016.

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

[53] `https://www.iit.it/research/lines/dynamic-legged-systems`.

[54] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ -A hydraulically and electrically actuated

quadruped robot," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[55] B. Ur rehman, M. Focchi, M. Frigerio, J. Goldsmith, D. Caldwell, and C. Semini, "Design of a hydraulically actuated arm for a quadruped robot," 09 2015.

[56] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the hydraulically actuated, torque-controlled quadruped robot hyq2max," *IEEE/ASME Transactions on Mechatronics*, vol. 22, pp. 635–646, April 2017.

[57] H. Khan, S. Kitano, M. Frigerio, M. Camurri, V. Barasuol, R. Featherstone, D. Caldwell, and C. Semini, "Development of the lightweight hydraulic quadruped robot -minihyq," 05 2015.

[58] C. Semini, *HyQ - Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, University of Genoa, Italy and Italian Institute of Technology (IIT), 2010.

[59] 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," *Under Review for IEEE International Conference on Robotics and Automation ICRA*, 2020.

[60] V. Samy, S. Caron, K. Bouyarmane, and A. Kheddar, "Post-Impact Adaptive Compliance for Humanoid Falls Using Predictive Control of a Reduced Model.".

[61] Y. Ding, C. Li, and H. W. Park, "Single Leg Dynamic Motion Planning with Mixed-Integer Convex Optimization," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7391–7396, 2018.

[62] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible region: an actuation-aware extension of the support region," *CoRR*, vol. abs/1903.07999, 2019.

[63] K. Fukuda and A. Prodon, "Double Description Method Revisited," in *Selected Papers from the 8th Franco-Japanese and 4th Franco-Chinese Conference on Combinatorics and Computer Science*, (Berlin, Heidelberg), pp. 91–111, Springer-Verlag, 1996.

[64] P. Chiacchio, Y. Bouffard-Vercelli, and F. Pierrot, "Force Polytope and Force Ellipsoid for Redundant Manipulators," *Journal of Robotic Systems*, vol. 14, no. 8, pp. 613–620, 1997.

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

[66] C. Gramkow, "On averaging rotations," *International Journal of Computer*

Vision, vol. 42, pp. 7–16, 1999.

[67] F. Ouezdou, S. Alfayad, and B. Almasri, "Comparison of several kinds of feet for humanoid robot," pp. 123 – 128, 02 2005.

[68] F. Doshi, E. Brunskill, A. Shkolnik, T. Kollar, K. Rohanimanesh, R. Tedrake, and N. Roy, "Collision detection in legged locomotion using supervised learning," *IEEE International Conference on Intelligent Robots and Systems*, pp. 317–322, 2007.

[69] M. F. V. Barasuol G. Fink and C. Semini., "On the Detection and Localization of Shin Collisions and Reactive Actions in Quadruped Robots," *22nd International Conference on Climbing and Walking Robots*, 2019.

[70] Y. Pyo, H. Cho, L. Jung, and D. Lim, *ROS Robot Programming*. ROBOTIS, 12 2017.

[71] `http://wiki.ros.org/Topics`.

[72] `http://wiki.ros.org/Services`.

[73] `http://wiki.ros.org/xpp`.

[74] `http://wiki.ros.org/rosbag`.