ISTITUTO ITALIANO
DI TECNOLOGIA
DYNAMIC LEGGED SYSTEMS

Università di Pisa

Corso di Laurea magistrale
in Ingegneria Robotica e dell'Automazione

On Slip Detection for Quadruped Robots

Supervisors:
Dr. Geoff Fink
Dr. Claudio Semini

University Advisor:
Prof. Lucia Pallottino

Author:
Ylenia Nisticò

Anno Accademico 2020/2021

This work is the result of an internship for Master's Thesis conducted at the Dynamic Legged Systems (DLS) at Istituto Italiano di Tecnologia (IIT), Genoa.

Tutors:
Dr. Geoff Fink
Dynamic Legged Systems (DLS) lab, Istituto Italiano di Tecnologia (IIT), Italy
Dr. Claudio Semini
Dynamic Legged Systems (DLS) lab, Istituto Italiano di Tecnologia (IIT), Italy

University Advisor:
Prof.ssa Lucia Pallottino
Research Center "E. Piaggio", Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Italy

"Nell'anno '99 di nostra vita
io, Francesco Guccini, eterno studente
perché la materia di studio sarebbe infinita
e soprattutto perché so di non sapere niente"
- Francesco Guccini
da *Addio, Stagioni*

Contents

List of Figures ii

List of Tables iv

Abstract vi

List of Acronyms vii

1 Introduction 1
- 1.1 State of the art . . . 2
  - 1.1.1 Proprioceptive Terrain-Aware Locomotion . . . 3
  - 1.1.2 Contact detection and localization . . . 5
  - 1.1.3 Slip detection and recovery . . . 6
- 1.2 Aim of this thesis . . . 8
- 1.3 Methodologies . . . 9
- 1.4 Outline . . . 9

2 Background theory 10
- 2.1 State Estimation . . . 10
  - 2.1.1 Linear systems theory . . . 10
  - 2.1.2 Probability theory . . . 13
  - 2.1.3 Propagation of the states and covariances . . . 16
- 2.2 The Kalman filter . . . 17
  - 2.2.1 The Discrete-time Kalman Filter . . . 18
  - 2.2.2 The Continuous-time Kalman filter . . . 20
- 2.3 The Extended Kalman Filter . . . 21
  - 2.3.1 The continuous-time extended Kalman filter . . . 21
  - 2.3.2 The discrete-time extended Kalman filter . . . 22
  - 2.3.3 The hybrid extended Kalman filter . . . 22

3 Modelling and Sensing 24
- 3.1 Quadruped Robot HyQ . . . 24
- 3.2 Robot Modelling . . . 25
  - 3.2.1 Onboard Sensors . . . 27
- 3.3 Controller . . . 28
  - 3.3.1 Reactive Controller Framework . . . 28

4 Methods **30**
4.1 Baseline Approach 30
4.1.1 One leg slip detection 30
4.1.2 Multiple leg slip detection 31
4.2 A novel approach for slip detection 33

5 Simulations Results **38**
5.1 Crawl 38
5.1.1 One Leg on an ice-patch 38
5.1.2 Comparison with the baseline approach 41
5.1.3 Multiple Legs on ice-patches 43
5.1.4 Comparison with the baseline approach 46
5.2 Trot 49
5.2.1 One Leg on ice-patch 49
5.2.2 Multiple Legs on ice-patches 53

6 Experimental Results **57**
6.1 Crawl 57
6.1.1 Walking on non-slippery ground 57
6.1.2 Walking on slippery ground 59

7 Conclusion **65**
7.1 Future Works 65

Appendices **66**

A Technical specifications of the robot **67**

B Forward Kinematics **68**

List of Figures

1.1 Popular legged robots 2
1.2 HyQ on several terrains with different compliances 5
1.3 An anti-skid foot prototype 8

2.1 Probability Density Function 14

3.1 HyQ robot 24
3.2 Location on Torso and Robot Base Coordinate Frame. 25
3.3 Slip dynamics 27
3.4 Overview of the locomotion framework. 29

4.1 Trends of ΔV and $\overline{\Delta V}$ with and without scaling in a simple trotting task with variable feed rate 34
4.2 Desired and actual foot velocity in a simulation task of crawling. 35

5.1 HyQ walking to an ice slab. 39
5.2 Shapes of $\overline{\Delta V}$ and ΔP: one leg on ice slab 40
5.3 Slip detection: one leg on ice slab 40
5.4 LF foot position along x-axis and y-axis 41
5.5 Shape of ΔV: baseline approach 42
5.6 Slip detection: baseline approach 42
5.7 LF leg position along x-axis and y-axis 43
5.8 Multiple legs on ice slabs 43
5.9 Shapes of $\overline{\Delta V}$ and ΔP: multiple legs on ice slabs 44
5.10 Slip detection: multiple legs on ice slabs 44
5.11 LF leg position along x-axis and y-axis 45
5.12 RF leg position along x-axis and y-axis 45
5.13 LH leg position along x-axis and y-axis 45
5.14 RH leg position along x-axis and y-axis 46
5.15 Shapes of ΔV: baseline approach 46
5.16 Slip detection: baseline approach 47
5.17 LF leg position along x-axis and y-axis. 47
5.18 RF leg position along x-axis and y-axis. 48
5.19 LH leg position along x-axis and y-axis. 48
5.20 RH leg position along x-axis and y-axis. 48
5.21 One leg on ice slabs 49
5.22 Shapes of $\overline{\Delta V}V$ and ΔP: one leg on ice slab 50

5.23 Slip detection: one leg on ice slab 50
5.24 LF leg position along x-axis and y-axis 51
5.25 Shapes of ΔV: baseline approach. 51
5.26 Slip detection: baseline approach 52
5.27 LF leg position along x-axis and y-axis. 52
5.28 Multiple legs on ice slabs 53
5.29 Slip detection: multiple legs on ice slabs 53
5.30 Shapes of $\overline{\Delta V}$ and ΔP: multiple legs on ice slabs 54
5.31 LF leg position along x-axis and y-axis. 54
5.32 RF leg position along x-axis and y-axis. 54
5.33 Shapes of ΔV: baseline approach 55
5.34 Slip detection: baseline approach 55
5.35 LF leg position along x-axis and y-axis 56
5.36 RF leg position along x-axis and y-axis 56

6.1 57
6.2 Slip detection: HyQ moving to a non slippery ground 58
6.3 Shapes of $\overline{\Delta V}$ and ΔP: HyQ moving to a non slippery ground 58
6.4 HyQ walking on a slippery pallet 59
6.5 Slip detection: multiple legs on slippery ground 59
6.6 Shapes of $\overline{\Delta V}$ and ΔP: multiple legs on slippery ground 60
6.7 LF leg position along x-axis and y-axis 60
6.8 RF leg position along x-axis and y-axis 60
6.9 LH leg position along x-axis and y-axis 61
6.10 RH leg position along x-axis and y-axis 61
6.11 Slip detection: baseline approach 62
6.12 Shapes of ΔV: baseline approach 62
6.13 LF leg position along x-axis and y-axis 62
6.14 RF leg position along x-axis and y-axis 63
6.15 LH leg position along x-axis and y-axis 63
6.16 RH leg position along x-axis and y-axis 63

B.1 Base frame and joint frames used to compute the forward kinematics . . . 68

List of Tables

5.1 Parameters of the simulation test: one leg on ice patch 39
5.2 Parameters of the simulation test: multiple legs on ice patch 43
5.3 Parameters of the simulation test: one leg on ice patch 49
5.4 Parameters of the simulation test: multiple legs on ice patch 53

6.1 Parameters of the experimental test: one leg on slippery patch 58

A.1 Technical specifications of the HyQ robot 67

Abstract

Legged robots are a class of robotic systems constructed to operate in dangerous and unstructured environments. They will outperform wheeled and tracked systems on uneven, rough and soft terrain. To perform dynamic whole-body locomotion and to traverse difficult terrain, legged robots require sophisticated control strategies to generate appropriate trajectories, and navigate without falling or getting stuck. Autonomous legged navigation is based on the close relationship between locomotion and perception, crucial to operate outside the laboratory. The robot has to perceive the environment and detect any change, in order to autonomously take decisions based on what it perceived. For instance, when the ground is slippery, common control techniques and state estimation algorithms may not be effective because the ground is commonly assumed to be non-slippery. This thesis addresses the problem of slip detection, a first fundamental step to implement a robust controller that allows the robot to traverse on slippery ground. We propose an approach independent of the gait type and independent of estimations of position and velocity of the robot in an inertial frame, that is usually prone to drift problems. We validated the approach on the hydraulically actuated quadruped robot HyQ , from the Istituto Italiano di Tecnologia (IIT), and we compare it against a state-of-art slip detection algorithm. We demonstrate the effectiveness of our approach and we show that the proposed method results in a better slip detection, which avoids drift-related problems, is not influenced by usual state estimation errors that can generate false positives, can be applied to detect the slippage of one or multiple legs indiscriminately and can adapted to different gait type.

List of Acronyms

DLS Dynamic Legged Systems

IIT Istituto Italiano di Tecnologia

MIT Massachusetts Institute of Technology

ROS Robot Operating System

HyQ Hydraulically actuated Quadruped

TAL Terrain-Aware Locomotion

PTAL Proprioceptive Terrain-Aware Locomotion

ETAL Exteroceptive Terrain-Aware Locomotion

WBC Whole-Body Control

pWBC Passive Whole-Body Control

STANCE Soft Terrain Adaptation aNd Compliance Estimation

F/T Force/Torque Sensors

GM Generalized Momenta

GRF Ground Reaction Force

IMU Inertial Measurement Unit

UKF Unscented Kalman Filter

HMM Hidden Markov Model

RV Random Variable

PDF Probability Distribution Function

pdf Probability density function

EKF Extended Kalman Filter

DoF Degree of Freedom

HAA Hip Adduction-Abduction

HFE Hip Flexion-Extension

KFE Knee Flexion-Extension

LIDAR LIght Detection And Ranging

SLAM Self Localization And Mapping

RCF Reactive Controller Framework

LF Left-Front

RF Rigth-Front

LH Left-Hind

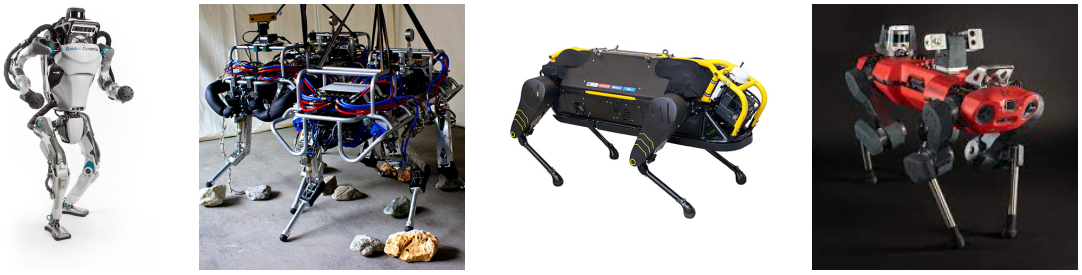
RH Right-Hind

Chapter 1

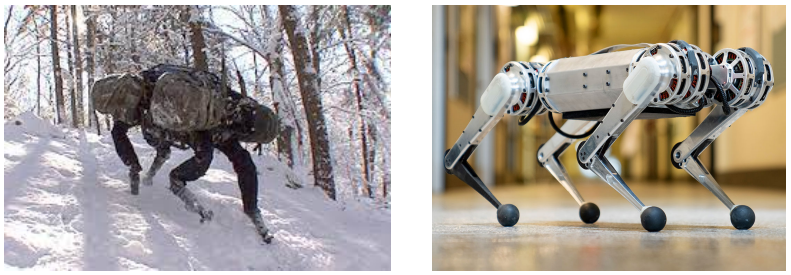
Introduction

We live in an era of rovers on Mars [1], drones surveying Earth (and Mars [2]), and self-driving cars. Similarly, legged robots have gained an increasing popularity over the past few decades. This is because they have the potential to operate in unstructured and dangerous environments, and to traverse difficult terrain where existing vehicles cannot go. For example, wheeled platforms perform poorly over soft or un-even terrain versus legged robots. As a result, half the Earth's landmass remained inaccessible to existing wheeled and tracked vehicles, whereas a much greater area can be reached by animals on foot [3]. In the near future, we will be able to build legged robots that can go to the places that animals can now reach. The reason why legs provide better mobility in rough terrain is that they can use isolated footholds that optimize support and traction. This means that a legged system can choose among the best footholds in the reachable terrain. On the other hand, wheeled platforms require a continuous path of support. A legged system also has greater agility in overcoming obstacles, and its performance can be independent of the roughness of the ground. This means that a legged robot is able to smoothly carry a load, even in case of pronounced variations in the terrain.

There is a further reason for exploring legged machines: to gain a better understanding of human and animal locomotion. Animals demonstrate great mobility and agility. Forms of locomotion on land include walking, running, hopping or jumping, dragging and crawling or slithering. Each step also requires energy to overcome inertia, balance is required for movements on land. We are still at a primitive stage in understanding the control principles that underlie walking and running. One way to learn more about plausible mechanisms for animal locomotion is to build legged machines, performing similar locomotion tasks. Then, studying their control systems and mechanical structures designed to solve similar problems, we can gain new insights into these problems and learn about possible solutions.



(a) Atlas [4] (b) HyQ [5] (c) HyQReal [6] (d) ANYmal [7]



(e) BigDog [8] (f) MIT Cheetah [9]

Figure 1.1: Some popular legged robots: they range from bipeds (a) to fully-hydraulically actuated quadrupeds (b),(c),(e), to fully electrically actuated quadruped robots (d),(f)

1.1 State of the art

The field of legged robotics has seen significant progress in recent years. The robots have acquired amazing capabilities and some of them have reached a point where they can actually leave lab environments and carry out tasks in real-world scenarios. For example, the performance of Atlas [4] (Fig. 1.1a), a bipedal robot of Boston Dynamics, continues to surprise us. In the most recent video, the robot demonstrated incredible athletic abilities by doing parkour [10]. On the other hand, among quadruped robots, the Hydraulically actuated Quadruped robots HyQ and HyQReal [5,6] were designed at the Istituto Italiano di Tecnologia (IIT) (Genoa) [11] to study highly dynamic motions (e.g. running, hopping, jumping), and to navigate over rough terrain (Fig. 1.1b,1.1c). HyQReal demonstrated a strong athletic ability, by pulling an aircraft in an experiment. The research group of Swiss Federal Institute of Technology (Zurich) developed the quadruped robot ANYmal [7], for autonomous operation in challenge environments [12]. Massachusetts Institute of Technology (MIT) introduced the electric actuated quadruped robot MIT Cheetah (Fig. 1.1f) [13] in 2013 and then MIT Cheetah 1,2,3 [14,15] to study the high speed quadrupedal locomotion. One of the most popular quadruped robot is Spot [16], constructed by Boston Dynamics Corporation. Spot is a versatile quadruped robot, suitable for many applications. It demonstrated great ability in navigating several kinds of terrain while sensing its environment. In 2019, Unitree Robotics released Aliengo [17], an electric quadruped robot which demonstrated excellent athletic performance, such as fast running, jumping, climbing in place after falling to the ground.

It is important that the improvements achieved on a hardware and control level go hand in hand with the perceptive capabilities of the robotic platforms. As part of this,

state estimation adopts a central role since estimated quantities are often prerequisites for other tasks such as balance control, trajectory planning, target tracking, or terrain mapping [18]. The strong dependency of other tasks on the estimated outputs imposes high reliability requirements from the state estimation. Missing, delayed, or bad estimates can quickly lead to failures of the robotic platform causing potential damage to the robot or its surroundings. This is the first reason that drives us to the use of simple sensor modalities: by employing sensors with low data processing complexity the framework is less prone to possible failures. For instance, inertial measurements require much less processing than image streams and are less affected by bad environmental conditions such as poor illumination or missing texture. Inertial sensors are nowadays often available on robotic platforms and, arguably, provide a very reliable source of information. While attitude estimates can be generated from inertial measurements only, position or velocity estimates are very inaccurate due to the underlying numerical integration of acceleration measurements. On the other hand, legged robots are often also equipped with kinematic sensors such as joint encoders.

This thesis focuses on state estimation using *proprioceptive sensors*, which only measure information internal to the robot (e.g. acceleration, motor speed, joint angles).

1.1.1 Proprioceptive Terrain-Aware Locomotion

Legged robots are moving out of research labs into the real-world with the promise of aiding humans in various applications. For this reasons, legged robots are expected to traverse terrains that are usually dynamic, unexplored, and uncertain. The core problem is that the terrain that the robot traverse introduces a large amount of uncertainty. The robots have to be able to perform several tasks: to sense the world around them and the terrain they are traversing, to understand the geometrical and physical properties of the terrain, to plan their motion based on the understanding of the terrain and its own limitations, and to quickly adapt this planned motion in case of unforeseen situations (e.g. falling, slipping, external pushes, etc.). This means that the robots have to be *terrain aware*. That is to say, the robot is able to perceive and understand the surrounding terrain, and is able to take decisions based on that. The robot has to have a good knowledge of its surroundings and use whatever sensors it has to perceive these surroundings and act upon them.

Terrain-Aware Locomotion (TAL) can be categorize into Proprioceptive Terrain-Aware Locomotion (PTAL) and Exteroceptive Terrain-Aware Locomotion (ETAL) [19]. ETAL relies on terrain information acquired using the robot's visual sensing. PTAL relies on the internal robot measurements, using information acquired by proprioceptive sensors. ETAL strategies have the advantage of being able to "look "at the environment in front of the robot. This allows both to select the best footholds based on the terrain information the capabilities of the legs (foothold selection [20, 21]), and to deduce some properties of the ground from images using deep learning [22, 23].

There are many scenarios where it is difficult to obtain visual feedback, for example, foggy areas or with dense vegetation. In these casses, PTAL strategies are more effective than ETAL ones. One common PTAL strategy is to localize and detect constacts. The robot interacts with the environment mainly through feet-ground contacts. Detecting

and localizing contacts is possibile because proprioceptive sensors measure the internal robot states. Some PTAL strategies rely on the joint position, velocity and/or torque measurements to detect and localize contacts [24–26]. Furthermore, PTAL strategies are used to deduce the physical and geometrical properties of the terrain and adapt accordingly. For instance, in [27] PTAL strategies have been adopted in locomotion planning and control over terrain with different impedance parameters. The robot was able to detect changes in the terrain impedance, and act upon it online. However, since PTAL uses only proprioceptive sensors, the action from these strategies are limited to corrective operations. The robot's internal states is not sufficient to predict future robot-terrain interactions: an action has to happen first before starting a reactive strategy. For example, the foot has to collide before triggering a step reflex, the shin has to collide to detect an obstacle [25], or the foot has to touch the terrain to infer its physical properties.

This thesis focuses on Proprioceptive Terrain-Aware Locomotion, applied at the level of state estimation. In particular, the thesis investigates locomotion on slippery terrain. The final goal is to develop a new algorithm for the leg-slip detection during the motion. In the following sections some of the most relevant work related to aforementioned topics are presented.

Locomotion over Soft Terrain

Walking on soft terrain is difficult because there are unmodeled contact dynamics that standard Whole Body Controllers (WBCs) are not accounting for. These uncertainties affect the locomotion, stability and performance of the system. Most of the work done on state estimation for legged robots is designed for rigid contacts, and does not take into account the physical parameters of the terrain. To be *terrain aware*, the robot should be able to adapt to terrains with different impedances. In [27, 28], the authors presented a Passive Whole-Body Control (pWBC) framework for quadruped robots, aware of the terrain geometry and friction properties and then they extended the PTAL capabilities of the previously presented pWBC, to adapt it to multiple terrains with different impedances (such as soft terrain, Fig. 1.2).

In [27], a soft terrain adaptation algorithm called Soft Terrain Adaptation aNd Compliance Estimation (STANCE) is proposed. It consists of a Compliant Contact Consistent Whole-Body Control that is aware of the terrain impedance, and an online Terrain Compliance Estimator that senses and estimates the terrain impedance. They showed that STANCE can adapt online to any type of terrain compliance (still or soft), with aggressive maneuvers, different forward velocities and external disturbances. They validated the algorithm in simulations and experiments, allowing the HyQ robot to adapt online to terrains with different compliances without pre-tuning. The robot successfully dealt with the transition between different terrains and showed the ability to differentiate between compliances under each foot.

In [29], to understand *"how and why the soft terrain affect state estimation for legged robots"*, the authors utilized a state estimator that fuses IMU measurements with leg odometry designed with rigid contact assumptions. They experimentally validated this state estimator on HyQ trotting over both soft and rigid terrain, and demonstrated that soft terrain negatively affects state estimation, and that the state estimates have a noticeable drift over soft terrain compared to rigid terrain.

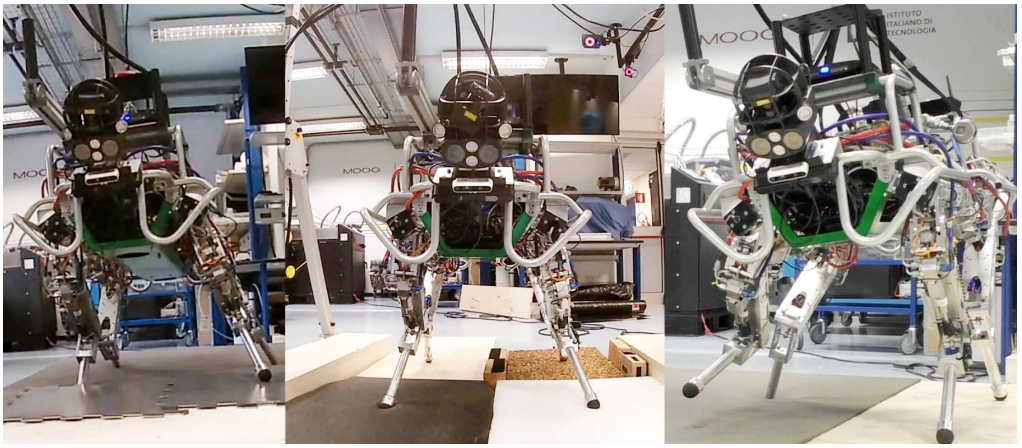


Figure 1.2: HyQ on several terrains with different compliances [29].

1.1.2 Contact detection and localization

Since legged robots operate in challenging and unstructured environments, unexpected collisions with obstacles (and self collisions) are lickely to occur. Many research works assume perfect knowledge of the environment and focus on generating motions at locations to avoid collisions. However, information about the topology of the environment might not be available in real-life applications. In these cases, the robot-environment interaction is not guaranteed to happen only at the end-effector (i.e. the foot). This situation is challenging because most advanced control strategies used to stabilize the trunk require the exact location of the contact points [28, 30, 31]. Consequently, to achieve robust locomotion, it is important to detect unexpected collisions and use them as feedback for stabilizing trunk controller. Relevant research effort has been devoted to finding methods that detect and estimate the location of arbitrary contacts occuring anywhere along the robot structure, not only at feet and hands. Some approaches use force/torque (F/T) sensors combined with a distributed skin to capture external forces that are applied to arbitrary locations [32–35]. However, most robots do not have this type of sensors as they significantly increase the complexity of the platform. Furthermore, most of these sensors are not capable nor designed to handle strong interaction forces; they are typically designed for tasks involving human-robot interaction.

Other methods for detecting and estimating external forces are based on model-based observers of generalized momenta (GM) [36]. The basic idea of these approaches is to compare computed torque with real torque to infer the action of external forces. Their advantage is that they require only proprioceptive measurements (torque and encoder measurements) without the need of additional force sensors. In [37] the first approach able to simultaneously detect collisions, find the contact link (not the contact location) and estimate the external forces is presented. Contact isolation in most GM approaches rely on thresholding the estimation coming from an observer (termed residual) that is a filtered version of the external wrench. However, most GM approaches are applied only to fixed-base robots. In [38] the authors developed an approach for floating-base robots (humanoids) in which they combined and compared the GM approaches with a F/T-sensor-based approach. In addition to contact isolation, some approaches try to estimate

the contact location at the price of increased complexity, e.g. in [24] researchers developed an optimization-based approach based on particle filters to detect and localize multiple contacts on a humanoid using only proprioceptive sensors.

In general, the majority of approaches based on proprioceptive measurements fail to determine the exact contact location on the most distal link. Also both in [38] and [24] rely on a very accurate model of the robot. Moreover, they are demonstrated only in simulation with a robot that is stationary when an external force is applied. They do not consider the implications of what happens when the robot contact is made during motion. When it comes to real applications, all the approaches based on filters or observers are strongly affected by modelling errors, sensor noise, offsets, friction and structural compliance. Moreover, due to the fact that they are filter-based, these approaches present an inherent delay (e.g. due to filter dynamics or the time it takes to reach the threshold to trigger detection) that often poses a limit on the gains of the observers.

In the context of quadruped robots, shin collisions are typically undesired events. Depending on the configurations of the robot [39], shin collisions can cause the robot to get stuck when moving forward or backward. During blind locomotions this situation becomes even more critical since the robot cannot acquire detailed information about the surface below it. In [25] the authors investigated the effect that systematic errors and delays in contact estimation have on quadruped locomotion performance. Initially, they described a study of the impact of shin collisions on dynamic locomotion, then they proposed a kinematic strategy to localize collisions on the most distal joint (i.e., the shin). They demonstrated with real experiments the increase in locomotion performance given by exploiting the knowledge of the online estimated shin contact in the stabilizing controller.

Another approach to localize contacts and collisions is proposed in a recent work [26]. Their method, implemented in 2D, is based on two assumptions: (i) the contact velocity that is in direction normal to the surface of the link must be zero, (ii) the contact velocity, just before the collision, must be positive. Their approach is simple because with only the usage of proprioceptive sensors, it is possible to detect the collision, providing a set of contact points, not only just one point, along the robot link. It can also be implemented easily, in order to obtain a single contact point, adding filtering or additional dynamic constraints such as acceleration.

1.1.3 Slip detection and recovery

One of the most open and challenging research problem in dynamic locomotion is slip detection and recovery. Most of the common methods perform trajectory planning and state estimation assuming no-slip conditions. This thesis mainly aims to investigate the slip detection making use of proprioceptive sensors.

The motion of a legged robot is based on two main phases: the *stance phase* in which the foot is in contact with the terrain, the *swing phase* in which the foot does not touch the ground and as a consequence no force is applied. Actually, most state-estimation algorithms rely on the assumption that the stance feet constraints are not violated (e.g. they are not moving or are supposed to move very little). Kinematic-based state estimation or odometry techniques [40, 41], which rely on the assumption that none of the stance feet is slipping, are prone to drift if the amount of slip is relevant (or if there is a compliance

between the base and the ground which is not modelled).

One of the earlier work on slip detection and recovery is from *Takemura et al.* [42], who presented both a *long term* and *short term* strategy for slip recovery. The former aims to change gait frequency and stride length when approaching slippery surfaces. However, changing locomotion parameters to address slippage can be successful only on terrain with limited roughness and moderate slipperiness. Conversely, if very challenging environment is considered, the occurrence of slippage might result in unrecoverable loss of stability because any other foot-step can be infeasible. At this extent, *Takemura* proposed to instantaneously add a force to have the ground reaction forces back in the friction cone. This approach has several shortcomings: (i) it is based on the idea that the normal is properly estimates; (ii) the required force is the result of the robot motion in interaction with the environment. More precisely, the GRFs can only be controlled to a limited extent in the null-space of the contact constraints. In addition to this the maximum applicable total normal force is constrained by the robot weight.

More recently, in one of the online videos, the BigDog robot (Fig. 1.1e) demonstrated to successfully recover from slipping on ice [8]. However, to date, no experimental results have been published and no details have been reported on the repeatability of the used approach.

In [18] a state estimation approach for legged robots based on kinematic velocity measurements at the ground contacts is introduced. The obtained information is fused with measurements from an on-board IMU by means of an unscented Kalman filter (UKF). The provided nonlinear observability analysis showed that, for general robot motions, all states were observable except for the global position and the yaw angle. This resulted in a filter which accurately estimates the inclination angles (roll and pitch) as well as the velocities of the robot. It also avoided unnecessary assumptions on the shape of the floor or on the employed gait pattern and was robust to a certain amount of foot slippage. This filter was implemented on the legged robot StarlETH [9] (Fig. **??**) and it enabled dynamic locomotion over uneven and labile terrain.

In [43] the authors presented a methodology for slip detection and estimation of the friction parameters, plus a recovery strategy which exploits the capabilities of a whole body controller, implemented for locomotion, which optimizes for the ground reaction forces (GRFs). The estimation makes use only of proprioceptive sensors. This method is fully described in chapter 4, since this approach is used as a baseline for the novel algorithm presented in this thesis project.

More recently, in [44] a probabilistic approach for contact and slip estimation, based on a Hidden Markov Model (HMM) is developed. A slip recovery approach relied on invasive impedance control and friction modulation. They performed field tests on a frozen ground, verifying that the presented pipeline could successfully stabilize ANYmal [7] (Fig. 1.1d), while losing traction.

However, the aforementioned works have attempted to improve kinematical-inertial estimation accuracy by detecting unstable contacts and reducing their influence on the overall estimation. Some approaches model the contact location as being fixed and affected only by Gaussian noise. This modeling is incorrect in several conditions, such as non-rigid terrain.

Alternatively, some works have focused on the design of the foot. For example, in

[45] an anti-skid foot which conditionally utilizes two types of foot pads is introduced: the primary foot pad with rubber and the complementary foot pad with the linearly-constrained spine mechanism and anchoring spines (Fig. 1.3). The foot switches the foot pad from the first to the second only when the primary foot pad slips. To detect the slippage on the primary foot pad, a passive slip detection mechanism is developed. In addition, to utilize the complementary foot pad only when the primary foot pad slips, a *lock* & *release* mechanism was designed.

Figure 1.3: The anti-skid foot prototype presented in [45]

These approaches are of limited applicability to legged robots, because they need a high cost force/torque sensor to be attached to the foot tip. However, due to the repetitive impacts with the ground, in the long run, this can result in a damage of the sensor. Furthermore, during locomotion, the touchdown event can create discontinuities in the force signal and jeopardize the detection. As a matter of facts, it is not easy to measure the instant when the force oscillation, due to the touch-down, has setted down, in order to have a detection without false positives. Conversely, a detection strategy based on kinematics is preferable in the context of legged robots where ground impacts are the order of the day.

Overall, conventional systems for legged locomotion over rough terrain enhance in complexity as more scenarios are considered, are extremely labor-intensive to develop and maintain, and remain vulnerable to situations beyond their controller design implementation.

1.2 Aim of this thesis

Perception and state estimation play an important role to achieve breakthroughs in dynamic whole-body locomotion for quadruped robots. This means that legged robots should be able to perceive their surrounding environment, detect any sudden change, and take decisions based on that. The detection aspect in quadruped robots includes (but is not limited to): detecting and localizing contact points between the robot and an obstacle, detecting the terrain's impedance properties, and detecting slippage. Based on that, this thesis focuses on developing a new approach for slip detection, the first fundamental step to then implement a robust controller that allows the robot the locomotion on slippery ground. Once the robot detect slippage, it can trigger a reflex action and adapt it's motion

trajectories to not slip. We validated the proposed approach on the 90 Kg Hydraulically actuated Quadruped (HyQ) robot (Istituto Italiano di Tecnologia IIT), and we compared it against a state-of-art slip detection algorithm. We showed that our approach can result in a better slip detection, which is independent of the gait type and of the robot position and velocity estimation in an inertial frame, that is related to drift-problems.

1.3 Methodologies

To validate the objective of this thesis it was necessary, first of all, to provide a C++ implementation of the robot kinematics and dynamics. The next step was the C++ implementation of the proposed algorithm and its verification in a simulation environment. In this context, we took advantage of the 3D visualizer RVIZ [46] , to then switch to a more accurate software simulator such as Gazebo [47]. To integrate the work with the other elements of the robot it was essential to build the proper ROS [48] structure.

To check if the results were consistent with the reality, PlotJuggler [49] was used to study the real-time behaviour of the robot. Then all the datas collected from the simulations were developed in Matlab [50] in order to improve the phase of filtering, providing, as a consequence, a desired output thanks to a manually tuned procedure.
Finally, once the simulation part was completed, it was possible to proceed with the hardware experiments on the robot HyQ.

1.4 Outline

This thesis is structured as follows: Chapter 1.1 presents a part of the state of the art related to the problem of state estimation and proprioceptive locomotion for legged robots; chapter 2 introduces the theory on state estimation and filtering; chapter 3 describes the target platform HyQ; chapter 4 describes a novel algorithm of slip detection; chapter 5 shows the results obtained in simulations and the comparison with those obtained using a previous existing algorithm of slip detection; chapter 6 shows the experimental results and finally chapter 7 is dedicated to conclusions and future works.

Chapter 2

Background theory

2.1 State Estimation

There are some common issues we must face in robotic applications, particularly *state estimation* and *control*. The *state* of a robot is a set of quantities, such as position, orientation, and velocity, that, if known, fully describe the robot's motion over time. Here we focus entirely on the problem of estimating the state of a robot, putting aside the notion of control.

In this chapter, we introduce the classic estimation results for linear systems corrupted by Gaussian measurement noise. We then examine some of the extensions to nonlinear systems with non-Gaussian noise. After all, we derive the state estimation algorithm (the Extended Kalman Filter EKF)

2.1.1 Linear systems theory

Many processes in our world can be described by state-space systems. If we derive a mathematical model for a process, then we can use the tools of mathematics to control the process and obtain information about it. Linear systems theory is based on the fact that if we know the state of a system at the present time, and we can know all of the present and future inputs, then we can deduce the values of all future outputs of the system. State-space models can be generally divided into linear models and nonlinear models. A continuous-time, deterministic linear system can be described by the equations:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \tag{2.1.1}$$

where x is the state vector, u is the control vector and y is the output vector. Matrices A, B, C are appropriately dimensioned matrices. The A matrix is the system matrix, B is the input matrix and C is the output matrix.

Although most real processes are nonlinear. Nonlinear systems are often approximated as linear systems, because the mathematical tools available for estimation and control are much more accessible and understood for linear systems. That way we can use the tools that have been developed for linear systems to derive estimation or control algorithms.

Real systems always have some nonlinearities. The general form of a continuous-time nonlinear system can be written as

$$\begin{cases} \dot{x} = f(x, u, w) \\ y = h(x, v) \end{cases} \tag{2.1.2}$$

where $f()$ and $h()$ are arbitrary vector-valued functions. w indicates process noise and v indicates measurement noise. If $f()$ and $h()$ are explicit functions of the time t then the system is time-varying. Otherwise the system is time-invariant. In order to apply tools from linear systems theory to nonlinear systems, we need to linearize the nonlinear system. To see how this is done, we can expand the nonlinear system equation $f(x, u, v)$ around the nominal operating point $(\bar{x}, \bar{u}, \bar{w})$. We then obtain a linear system approximation as follows:

$$\dot{x} = f(x, u, w) \simeq f(\bar{x}, \bar{u}, \bar{w}) + \frac{\partial f}{\partial x}\Big|_0 (x - \bar{x}) + \frac{\partial f}{\partial u}\Big|_0 (u - \bar{u}) + \frac{\partial f}{\partial w}\Big|_0 (w - \bar{w}) = \dot{\tilde{x}} + A\tilde{x} + B\tilde{u} + L\tilde{w} \tag{2.1.3}$$

where the 0 subscript means that the function is evaluated at the nominal point $(\bar{x}, \bar{u}, \bar{w})$ and A, B and L are defined by the above equations. The *tilde* quantities in 2.1.3 are defined as

$$\begin{aligned} \tilde{x} &= x - \bar{x} \\ \tilde{u} &= u - \bar{u} \end{aligned} \tag{2.1.4}$$

Subtracting $\dot{\tilde{x}}$ from both side of equation 2.1.3 gives

$$\dot{\tilde{x}} = A\tilde{w} + B\tilde{u} + L\tilde{w} \tag{2.1.5}$$

Since w is noise, we will set $\bar{w} = 0$ so that $\tilde{w} = w$ and we obtain

$$\dot{\tilde{x}} = A\tilde{w} + B\tilde{u} + Lw \tag{2.1.6}$$

We see that we have a linear equation for $\dot{\tilde{x}}$ in terms of $\tilde{x}$,$\tilde{u}$ and w. As long as the deviations from the nominal values remain small, the linearization will be accurate.
In a similar manner we can expand the nonlinear measurement equation given by 2.1.2 around a nominal operating point $x = \bar{x}$ and $\bar{v} = 0$. This results in the linearized measurement equation:

$$\tilde{y} = \frac{\partial h}{\partial x}\Big|_0 \tilde{x} + \frac{\partial h}{\partial v}\Big|_0 \tilde{v} = C\tilde{x} + Dv \tag{2.1.7}$$

where C and D are defined by the above equation and $\tilde{y} = y - \bar{y}$. 2.1.3 and 2.1.7 comprise a linear system that describes the deviations of the state and output from their nominal values.

Most systems in the real world are described with continuous-time dynamics . However, state estimation and control algorithms are almost always implemented in digital electronics. This often requires a transformation of continuous-time dynamics to discrete-time dynamics.
The solution of a continuous-time linear system 2.1.1 is given by

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{2.1.8}$$

Let $t = t_k$ (some discrete time point) and let the initial time $t_0 = t_{k-1}$ (the previous discrete time point). Assume that $A(\tau)$, $B(\tau)$ and $u(\tau)$ are approximately constant in the interval of integration. We then obtain

$$x(t_k) = e^{A(t_k - t_{k-1})}x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{A(t_k-\tau)}d\tau Bu(\tau_{k-1}) \tag{2.1.9}$$

Now define $\Delta t = t_k - t_{k-1}$, and substitute for τ in the above equation to obtain

$$x(t_k) = e^{A\Delta t}x(t_{k-1}) + \int_0^{\Delta t} e^{A(\Delta t-\alpha)}d\alpha Bu(\tau_{k-1}) = e^{A\Delta t}x(t_{k-1}) + e^{A\Delta t}\int_0^{\Delta t} e^{-A\alpha}d\alpha Bu(t_{k-1}) \tag{2.1.10}$$

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} \tag{2.1.11}$$

where x_k, F_k, G_k and u_k are defined by the above equation. This is a linear dicrete-time approximation to the continuous-time dynamics given in 2.1.1. The transformation from a continuous-time system to a discrete-time system is called *discretization*. At the end of the discretization we obtain a linear system defined as

$$\begin{cases} x_{k+1} = Fx_k + Gu_k \\ y_k = Hx_k \end{cases} \tag{2.1.12}$$

To compute the G matrix we approximate the integral of 2.1.10. The most accurate approximations, in order of increasing computational effort, are *rectangular integration, trapezoidal integration* and *fourth-order Runge-Kutta integration*:

- Rectangular integration (also called *Euler integration*). It uses rectangles to approximate the area under the curve. The interval of integration could be sub-divided into n smaller intervals of equal lengths and, assuming the single time interval $(t_{n+1} - t_n)$ small, n rectangles can be used to approximate the integral:

$$x(t_n) \simeq x(0) + \sum_{k=0}^{n}\int_{t_k}^{t_{k+1}} f[x(t_k), u(t_k), t_k]dt = x(0) + \sum_{k=0}^{n} f[x(t_k), u(t_k), t_k]T \tag{2.1.13}$$

- Trapezoidal integration. This method is based on the same idea of the previous one but instead of approximating each area as a rectangle, it approximates each area as a trapezoid.

- Range-Kutta integration. In order to further improve the integral approximation, this approach performs additional function calculations at each time step. $n_{th}-order$ Runge-Kutta integration is the approximation of an integral by performing n function calculations at each time step. This kind of integration is more computationally demanding than rectangular or trapezoidal integration, but it also provides for greater accuracy.

2.1.2 Probability theory

In an idealized world, the robot might incorporate sensors that can measure, without error, the state x. Real sensors are characterized by noise and, more important, by range limitations. In our attempt to filter a signal, we will be trying to extract meaningful information from a noisy signal. In order to accomplish this, we need to know something about what noise is, some of its characteristics and how it works. The probabilistic approach generalizes this idealized view by modeling robot sensors by conditional probability distributions. Firs of all it is necessary to introduce some definitions. We define a random variable (RV) as a functional mapping from a set of experimental outcomes (the domain) to a set of real numbers (the range). The outcome of a particular experiment is not an RV, it becomes certain. This distinction between an RV and its realization is important for understanding the concept of probability. An RV can be either continuous or discrete. The most fundamental property of an RV X is its probability distribution function (PDF) $F_X(x)$ defined as

$$F_X(x) = P(X \leq x) \tag{2.1.14}$$

In the above equation, $F_X(x)$ is the PDF of the RV X, and x is a nonrandom independent variable or constant. Some properties of the RV are

$$\begin{array}{rcll} F_X(x) & \in & [0,1] & \\ F_X(-\infty) & = & 0 & \\ F_X(\infty) & = & 1 & \\ F_X(a) & \leq & F_X(b) & if \quad a \leq b \\ P(a < X \leq b) & = & F_X(b) - F_X(a) & \end{array} \tag{2.1.15}$$

Th probability density function pdf $f_X(x)$ is defined as the derivative of the PDF.

$$f_X(x) = \frac{dF_X(x)}{dx} \tag{2.1.16}$$

Some properties of the pdf that can be obtained from this definition are

$$\begin{array}{rcl} F_X(x) & = & \int_{-\infty}^{x} f_X(z)dz \\ ff_X(x) & \geq & 0 \\ \int_{-\infty}^{infty} f_X(x)dx & = & 1 \\ P(a < x \leq b) & = & \int_a^b f_X(x)dx \end{array} \tag{2.1.17}$$

PDF and pdf can be *conditional*. The conditional distribution and density of the RV X given the fact that event A occurred are defined as

$$F_X(x|A) = P(X \leq x|A) = \frac{P(X \leq x, A)}{P(A)} \tag{2.1.18}$$

$$f_X(x|A) = \frac{dF_X(x|A)}{dx} \tag{2.1.19}$$

The average value of an RV X over a large number of experiments is the *exptected value*, also be called the *expectation*, the *mean* or the *average* of the RV. Suppose that we run

the experiment N times and observe a total of m different outcomes. We observe the outcome A_1 occurs n_1 times, A_2 occurs n_2 times, ... , and A_m occurs n_m times. Then the expected value of X is computed as

$$E(X) = \frac{1}{N}\sum_{i=1}^{m} A_i n_i \tag{2.1.20}$$

We can compute the expected value of X also as

$$E(X) = \int_{-\infty}^{\infty} x f_X(x) dx \tag{2.1.21}$$

A measure of how much variability there is in an RV is given by the *variance*. It is a measure of how much we expect the RV to vary from its *mean*. The formal definition is

$$\sigma_x^2 = E[(X - \bar{x})^2] = \int_{-infty}^{infty} (x - \bar{x})^2 f_X(x) dx \tag{2.1.22}$$

We use the notation $\tilde{X}(\bar{x}, \sigma^2)$ to indicate that X is an RV with a mean of $\bar{x}$ and a variance of σ^2.

An RV is called *Gaussian* or normal if its pdf is given by

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{[\frac{-(x-\bar{x})^2}{2\sigma^2}]} \tag{2.1.23}$$

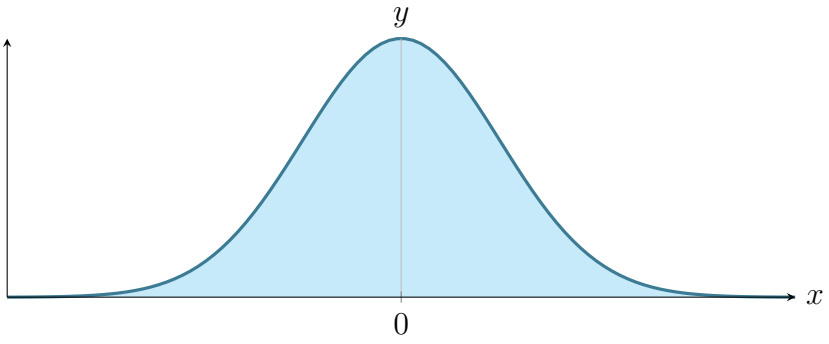


Figure 2.1: pdf of a Gaussian RV with mean 0 and variance 1

Figure 2.1 shows the pdf of a Gaussian RV with mean of zero and variance of one. If the mean changes, the pdf will shift to the left or right. If the variance increases, the pdf will spread out. If the variance decreases, the pdf will be squeezed in. The PDF of a Gaussian RV is given by

$$F_X(X) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{[\frac{-(z-\bar{x})^2}{2\sigma^2}]} dz \tag{2.1.24}$$

Two different events are *independent* if the occurrence of one event has no effect on the probability of the occurrence of the other event. This concept can be extended to say that random variables X and Y are *independent* if they satisfy the following relation:

$$P(X \leq x, Y \leq y) = P(X \leq x)P(Y \leq y) \quad \forall x, y \tag{2.1.25}$$

This implies

$$\begin{aligned} F_{XY}(x,y) &= F_X(x)F_Y(y) \\ f_{XY}(x,y) &= f_X(x)f_Y(y) \end{aligned} \tag{2.1.26}$$

We define the *covariance* of two scalar RVs X and Y as

$$C_{XY} = E[(X - \bar{X})(Y - \bar{Y})] = E(XY) - \bar{X}\bar{Y} \tag{2.1.27}$$

The *correlation coefficient* of two scalar RVs X and Y is

$$\rho = \frac{C_{XY}}{\sigma_x \sigma_y} \tag{2.1.28}$$

The correlation coefficient is a normalized measurement of the independence between two RVs X and Y. If X and Y are independent, then $\rho = 0$ (although the converse is not necessarily true). The *correlation* of two scalar RVs X and Y is

$$R_{XY} = E(XY) \tag{2.1.29}$$

Two RVs are said to be uncorrelated if $R_{XY} = E(X)E(Y)$. From the definition of independence, we see that if two RVs are independent then they are also uncorrelated. Independence implies uncorrelatedness, but uncorrelatedness does not necessarily imply independence. However, in the special case in which two RVs are both Gaussian and uncorrelated, then it follows that they are also independent.

A random variable can change with time. In this case we are talking about a *stochastic process* $X(t)$. Since a stochastic process is an RV that changes with time, it has a distribution and density function that are function of time. If $X(t)$ is a random vector of n elements the PDF and pdf of $X(t)$ are

$$F_X(x,t) = P[X_1(t) \leq x_1 and \cdots X_n(t) \leq x_n(t)] \tag{2.1.30}$$

$$f_X(x,t) = \frac{d^n F_X(x,t)}{dx_1 \cdots dx_n} \tag{2.1.31}$$

The mean and the covariance of $X(t)$ are also functions of time:

$$\bar{x} = \int_{-infty}^{infty} x f(x,t) dx \tag{2.1.32}$$

$$C_X(t) = E[X(t) - \bar{x}(t)][X(t) - \bar{x}(t)] = \int_{-infty}^{infty} [x - \bar{x}(t)][x - \bar{x}(t)]^T f(x,t) dx \tag{2.1.33}$$

The correlation between two RVs $X(t_1)$ and $X(t_2)$ is called the *autocorrelation* of the stochastic process $X(t)$:

$$R_X(t_1, t_2) = E[X(t_1)X^T(t_2)] \tag{2.1.34}$$

The autocovariance of a stochastic process is defined as

$$C_X(t_1, t_2) = E[X(t_1) - \bar{X}(t_1)][X(t_2) - \bar{X}(t_2)]^T \tag{2.1.35}$$

If the RV $X(t_1)$ is independent from the RV $X(t_2)$ for all $t_1 \neq t_2$ then $X(t)$ is called *white noise*. Otherwise $X(t)$ is called *colored noise*.

In optimal filtering research and experiments, we often have to simulate correlated white noise. That is, we need to create random vectors whose elements are correlated with each other according to some predefined covariance matrix. Suppose we want to generate an n-element random vector w that has zero mean and covariance Q:

$$Q = \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{1n} \\ \vdots & & \vdots \\ \sigma_{1n} & \dots & \sigma_n^2 \end{bmatrix} \tag{2.1.36}$$

Since Q is a covariance matrix, we know that all of its eigenvalues are real and nonnegative. We can therefore denote its eigenvalues as μ_k^2.

$$\lambda(Q) = \mu_k^2 \quad (k = 1, ..., n) \tag{2.1.37}$$

We can generate an n-element random vector w with a covariance matrix of Q following this steps:

- Find the eigenvalues of Q, and denote them as $\mu_1^2, ..., \mu_n^2$
- Find the eigenvectors of Q, and denote them as $d_1, ..., d_n$ such that

$$\begin{array}{rcl} D & = & [d_1 \cdots d_n] \\ D^{-1} & = & D^T \end{array} \tag{2.1.38}$$

- For $i = 1, ..., n$ compute the random variable $v_i = \mu_i r_i$, where each r_i is an independent random number with a variance of 1 (unity variance).
- Set $w = Dv$

2.1.3 Propagation of the states and covariances

Suppose we have the following linear discrete-time system:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \tag{2.1.39}$$

where u_k is a known input and w_k is a Gaussian zero-mean white noise with covariance Q_k. The mean of the state x_k and the covariance of x_k change with time. We can use 2.1.39 and 2.1.40 to obtain their expressions:

$$\bar{x}_k = E(x_k) = F_{k-1}\bar{x}_{k-1} + G_{k-1}u_{k-1} \tag{2.1.40}$$

$$\begin{array}{rcl} (x_k - \bar{x}_k)(\cdots)^T & = & (F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} - \bar{x}_k)(\cdots)^T \\ & = & [F_{k-1}(x_{k-1} - \bar{x}_{k-1}) + w_{k-1}][\cdots]^T \\ & = & F_{k-1}(x_{k-1} - \bar{x}_{k-1})(x_{k-1} - \bar{x}_{k-1})^T F_{k-1}^T + w_{k-1}w_{k-1}^T + \\ & & F_{k-1}(x_{k-1} - \bar{x}_{k-1})w_{k-1}^T + w_{k-1}(x_{k-1} - \bar{x}_{k-1})^T F_{K-1}^T \end{array} \tag{2.1.41}$$

We therefore obtain the covariance ok x_k as the expected value of the above expression. Since $(x_{k-1} - \bar{x}_{k-1})$ is uncorrelated with w_{k-1}, we obtain

$$P_k = E[(x_k - \bar{x}_k)(\cdots)^T] = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1} \tag{2.1.42}$$

It is interesting to consider the conditions under which 2.1.42 has a steady-state solution. That is, suppose that $F_k = F$ is a constant and $Q_k = Q$ is a constant.
Now let us look at the solution of the linear system of equation 2.1.39:

$$x_k = F_{x,0}x_0 + \sum_{i=0}^{k-1}(F_{k,i+1}w_i + F_{k,i+1}G_iu_i) \tag{2.1.43}$$

The matrix F_k is the state transition matrix of the system and is defined as

$$F_{k,i} = \begin{cases} F_{k-1}F_{k-2}\dots F_i & k > i \\ I & k = i, \\ 0 & k < i \end{cases} \tag{2.1.44}$$

Notice from equation 2.1.43 that x_k is a linear combination of x_0, w_i and u_i. If the input sequence u_i is known, then it is a constant and can be considered to be a sequence of Gaussian random variables with zero covariance. If x_0 and w_i are unknown but are Gaussian random variables, then x_k in Equation 2.1.43 is a linear combination of Gaussian random variables. Therefore x_k is itself a Gaussian random variable. But we computed the mean and covariance of x_k in Equations 2.1.40 and 2.1.42. Therefore

$$x_k \sim N(\bar{x}_k, P_k) \tag{2.1.45}$$

This completely characterizes x_k in a statistical sense since a Gaussian random variable is completely characterized by its mean and covariance.

The reviewed basic concepts of probability, random variables and stochastic processes. are fundamental to the Kalman filter that is derived in the Chapter 2.2

2.2 The Kalman filter

The Kalman filter operates by propagating the mean and the covariance of the state through time. The approach to deriving the Kalman filter will involve the following steps:

- We start with a mathematical description of a dynamic system whose states we want to estimate.
- We implement equations that describe how the mean of the state and the covariance of the state propagate with time.
- We take the dynamic system that describes the propagation of the state mean and covariance, and implement the equations.
  - The mean of the state is the Kalman filter estimate of the state

  - The covariance of the state is the covariance of the Kalman filter state estimate

- Every time that we get a measurement, we update the mean and covariance of the state.

2.2.1 The Discrete-time Kalman Filter

Suppose we have a linear dicrete-time system given as follow:

$$\begin{cases} x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \\ y_k = H_k x_k + v_k \end{cases} \tag{2.2.1}$$

The noise processes w_k and v_k are white, zero-mean, uncorrelated, and have known covariance matrices Q_k and R_k, respectively:

$$\begin{array}{rcl} w_k & \sim & (0, Q_k) \\ v_k & \sim & (0, R_k) \\ E[w_k w_j^T] & = & Q_k \delta_{k-j} \\ E[v_k v_j^T] & = & R_k \delta_{k-j} \\ E[v_k w_k^T] & = & 0 \end{array} \tag{2.2.2}$$

where δ_{k-j} is the Kronecker delta function; that is $\delta_{k-j} = 1$ if $k = j$, and $\delta_{k-j} = 0$ if $k \neq j$.

The goal is to estimate the state x_k based on the knowledge of the system dynamics and the availability of the noisy measurements y_k. The amount of information that is available for our state estimation varies depending on the particular problem. If we have all of the measurements up to and including time k available for use in our estimate of x_k, then we can form an *a posteriori* estimate, which we denote as $\hat{x}_k^+$. One way to form the *a posteriori* state estimate is to compute the expected value of x_k conditioned on all of the measurements up to and including time k:

$$\hat{x}_k^+ = E[x_k | y_1, y_2, \cdots, y_k] = a\ posteriori - estimate \tag{2.2.3}$$

If we have all of the measurements before (but not including) time k available for use in our estimate of x_k, then we can form an *a priori* estimate, which we denote as $\hat{x}_k^-$. To form the *a priori* state estimate we compute the expected value of x_k conditioned on all of the measurements before (but not including) time k:

$$\hat{x}_k^- = E[x_k | y_1, y_2, \cdots, y_{k-1}] = a\ priori - estimate \tag{2.2.4}$$

It is important to note that $\hat{x}_k^+$ and $\hat{x}_k^-$ are both estimate of the same quantity; they are both estimate of x_k. However, $\hat{x}_k^-$ is the estimate of x_k *before* the measurement y_k is taken into account, and $\hat{x}_k^+$ is the estimate of x_k *after* the measurement y_k is taken into account.

In the notation that follows, $\hat{x}_0^+$ denotes the initial estimate of x_0 before any measurements are available. The first measurement is taken at time $k = 1$. Since we do not have any measurements available to estimate x_0, it is reasonable to form $\hat{x}_0^+$ as the expected value of the initial state x_0:

$$\hat{x}_0^+ = E(x_0) \tag{2.2.5}$$

The term P_k denotes the covariance of the estimation error. P_k^- denotes the covariance of the estimation error of $\hat{x}_k^-$, and P_k^+ denotes the covariance of the estimation error of $\hat{x}_k^+$:

$$\begin{array}{rcl} P_k^- & = & E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] \\ P_k^+ & = & E[(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T] \end{array} \tag{2.2.6}$$

$$\begin{array}{rcl} \hat{x}_{k-1}^-|\hat{x}_{k-1}^+ & \longrightarrow & \hat{x}_k^-|\hat{x}_k^+ \\ P_{k-1}^-|P_{k-1}^+ & \longrightarrow & P_k^-|P_k^+ \end{array} \tag{2.2.7}$$

After we process the measurement at time $(k-1)$, we have an estimate of x_{k-1} (denoted $\hat{x}_{k-1}^+$) and the covariance of that estimate (denoted P_{k-1}^+). When time k arrives, before we process the measurement at time k we compute an estimate of x_k (denoted $\hat{x}_k^-$) and the covariance of that estimate (denote P_k^-). Then we process the measurement at time k to refine our estimate of x_k. The resulting estimate of x_k is denoted $\hat{x}_k^+$ and its covariance is denoted P_k^+

We begin the estimation process with $\hat{x}_0^+$. Given $\hat{x}_0^+$, we want to set $\hat{x}_1^- = E(x_1)$. We know that $\hat{x}_0^+ = E(x_0)$ and we know from eq. 2.1.40 how the mean ok x propagates with time: $\bar{x}_1^- = F_{k-1}\bar{x}_{k-1}^+ + G_{k-1}u_{k-1}$. We therefore obtain:

$$\hat{x}_1^- = F_0\hat{x}_0^+ + G_0u_0 \tag{2.2.8}$$

The reasoning can be extended to obtain the following more general equation:

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} \tag{2.2.9}$$

This is called the time update equation for $\hat{x}$. From time $(k-1)^+$ to time k^- we do not have any additional measurement available to help us update our state estimate between time, so we should just update the state estimate based on our knowledge of the system dynamics.

Next we need to compute the time update equation for P, the covariance of the state estimation error. We begin with P_0^+, which is the covariance of our initial estimate of x_0. In general, P_0^+ represents the uncertainty in our initial estimate of x_0. If we know the initial state perfectly, then $P_0^+ = 0$. If we have absolutely no idea the value of x_0, then $P_0^+ = I$.

$$\begin{array}{rcl} P_0^+ & = & E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T] \\ & = & E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{array} \tag{2.2.10}$$

Given P_0^+ we can compute P_1^- recalling from Equation 2.1.42. We therefore obtain

$$P_1^- = F_0P_0^+F_0^T + Q_0 \tag{2.2.11}$$

The reasoning can be extended to obtain the following more general equation:

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{2.2.12}$$

This is called the time-update equation for P.

We have derived the time-update equations for $\hat{x}$ and P. Now we need to derive the measurement-update equations for $\hat{x}$ and P.

The only difference between $\hat{x}_k^-$ and $\hat{x}_k^+$ is that $\hat{x}_k^+$ takes the measurement y_k into account:

$$\begin{array}{rcc} K_k & = & P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1} \\ & = & P_kH_k^TR_k^{-1} \\ \hat{x}_k & = & \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1}) \\ P_k & = & (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T \\ & = & (P_{k-1}^{-1} + H_k^TR_k^{-1}H_k)^{-1} \\ & = & (I - K_kH_k)P_{k-1} \end{array} \tag{2.2.13}$$

where $\hat{x}_{k-1}$ and P_{k-1} are the estimate and its covariance *before* the measurement y_k is processed, and $\hat{x}_k$ and P_k are the estimate and its covariance *after* the measurement y_k is processed.
In Equation 2.2.13 we can replace $\hat{x}_{k-1}$ with $\hat{x}_k^-$, P_{k-1} with P_k^-, $\hat{x}_k$ with $\hat{x}_k^+$ and P_k with P_k^+. This results in

$$\begin{array}{rcc} K_k & = & P_k^-H_k^T(H_kP_k^-H_k^T + R_k)^{-1} \\ & = & P_k^+H_k^TR_k^{-1} \\ \hat{x}_k & = & \hat{x}_k^- + K_k(y_k - H_k\hat{x}_k^-) \\ P_k & = & (I - K_kH_k)P_k^-(I - K_kH_k)^T + K_kR_kK_k^T \\ & = & [(P_k^-)^{-1} + H_k^TR_k^{-1}H_k]^{-1} \\ & = & (I - K_kH_k)P_k^- \end{array} \tag{2.2.14}$$

These are measurement-update equations for $\hat{x}_k$ and P_k. The matrix K_k in the above equations is called the *Kalman filter gain* and the quantity $(y_k - H_k\hat{x}_k^-)$ is the innovation.

2.2.2 The Continuous-time Kalman filter

The majority of Kalman Filter applications are implemented in digital computers. However there are still opportunities to implement Kalman filters in continuous time (i.e. in analog circuits). For completeness the derivation of the continuous-time filter is reported in the following:

- The continuous-time system dynamics and measurement equations are given as:

$$\begin{array}{rcc} \dot{x} & = & Ax + Bu + w \\ y & = & Cx + v \\ w & \sim & (0, Q_c) \\ v & \sim & (0, R_c) \end{array} \tag{2.2.15}$$

  Note that $w(t)$ and $v(t)$ are continuous-time white noise processes.

- The continuous-time Kalman filter equations are given as

$$\begin{array}{rcc} \hat{x} & = & E[x(0)] \\ P(0) & = & E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T] \\ K & = & PC^TR_c^{-1} \\ \dot{\hat{x}} & = & A\hat{x} + Bu + K(y - C\hat{x}) \\ \dot{P} & = & -PC^TR_c^{-1}CP + AP + PA^T + Q_c \end{array} \tag{2.2.16}$$

2.3 The Extended Kalman Filter

All of the discussion to this point has considered linear filters for linear systems. Unfortunately, linear systems do not exist. All systems are ultimately nonlinear. In this case, we need to explore nonlinear estimator.

In this chapter we focuse on the Extended Kalman Filter (EKF). It is based on the idea of linearizing the nonlinear system around the Kalman filter estimate, and the Kalman filter estimate is based on the linearized system.

In the following sections we present the EKF for continuous-time systems with continuous-time measurements, then the EKF for discrete-time systems with discrete-time measurements, and finally the hybrid EKF, which is the EKF for continuous-time systems with discrete-time measurements. The hybrid EKF is the state-estimator used in this thesis.

2.3.1 The continuous-time extended Kalman filter

The system equations are given as

$$\begin{aligned} \dot{x} &= f(x,u,w,t) \\ y &= h(x,v,t) \\ w &\sim (0,Q) \\ v &\sim (0,R) \end{aligned} \tag{2.3.1}$$

Compute the following partial derivative matrices evaluated at the current state estimate

$$\begin{aligned} A &= \left.\frac{\partial f}{\partial x}\right|_{\hat{x}} \\ L &= \left.\frac{\partial f}{\partial w}\right|_{\hat{x}} \\ C &= \left.\frac{\partial h}{\partial x}\right|_{\hat{x}} \\ M &= \left.\frac{\partial h}{\partial v}\right|_{\hat{x}} \end{aligned} \tag{2.3.2}$$

Compute the following matrices

$$\begin{aligned} \tilde{Q} &= LQL^T \\ \tilde{R} &= MRM^T \end{aligned} \tag{2.3.3}$$

Execute the following Kalman filter equations:

$$\begin{aligned} \hat{x}_0 &= E[x(0)] \\ P(0) &= E[(x(0)-\hat{x}(0))(x(0)-\hat{x}(0))^T] \\ \dot{\hat{x}} &= f(\hat{x},u,w_0,t) + K[y - h(\hat{x},v_0,t)] \\ K &= PC^T\tilde{R}^{-1} \\ \dot{P} &= AP + PA^T + \tilde{Q} - PC^T\tilde{R}^{-1}CP \end{aligned} \tag{2.3.4}$$

2.3.2 The discrete-time extended Kalman filter

The system and measurement equations are given as follows:

$$\begin{array}{rcc} x_k & = & f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \\ y_k & = & h_k(x_k, w_k) \\ w_k & \sim & (0, Q_k) \\ v_k & \sim & (0, R_k) \end{array} \tag{2.3.5}$$

Initialize the filter as follows:

$$\begin{array}{rcc} \hat{x}_0^+ & = & E(x_0) \\ P_0^+ & = & E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{array} \tag{2.3.6}$$

For $k = 1, ., \cdots$, performs the following:

- Compute the following partial derivative matrices:

$$\begin{array}{rcl} F_{k-1} & = & \frac{\partial f_{k-1}}{\partial x}\Big|_{\hat{x}_{k-1}^+} \\ L_{k-1} & = & \frac{\partial f_{k-1}}{\partial w}\Big|_{\hat{x}_{k-1}^+} \end{array} \tag{2.3.7}$$

- Perform the time update of the state estimate and estimation-error covariance as follows:

$$\begin{array}{rcc} P_k^- & = & F_{k-1} P_{k-1}^+ F_{k-1}^T + L_{k-1} Q_{k-1} L_{k-1}^T \\ \hat{x}_k^- & = & f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0) \end{array} \tag{2.3.8}$$

- Compute the following partial derivative matrices:

$$\begin{array}{rcl} H_k & = & \frac{\partial h_k}{\partial x}\Big|_{\hat{x}_k^-} \\ M_k & = & \frac{\partial h_k}{\partial v}\Big|_{\hat{x}_k^-} \end{array} \tag{2.3.9}$$

- perform the measurement update of the state estimate and estimation-error covariance as follows:

$$\begin{array}{rcc} K_k & = & P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \\ \hat{x}_k^+ & = & \hat{x}_k^- + K_k[y_k - h_k(\hat{x}_k^-, 0)] \\ P_k^+ & = & (I - K_k H_k) P_k^- \end{array} \tag{2.3.10}$$

2.3.3 The hybrid extended Kalman filter

The system equations with continuous-time dynamics and discrete-time measurements are given as follows:

$$\begin{array}{rcc} \dot{x} & = & f(x, u, w, t) \\ y_k & = & h_k(x_k, v_k) \\ w(t) & \sim & (0, Q) \\ v_k & \sim & (0, R_k) \end{array} \tag{2.3.11}$$

Initialize the filter as follows:

$$\begin{array}{rcc} \hat{x}_0^+ & = & E[x_0] \\ P_0^+ & = & E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{array} \tag{2.3.12}$$

For $k = 1, 2, \cdots$, perform the following:

- Integrate the state estimate and its covariance from time $(k-1)^+$ to time k^- as follows:

$$\begin{array}{rcl} \dot{\hat{x}} & = & f(\hat{x}, u, 0, t) \\ \dot{P} & = & AP + PA^T + LQL^T \end{array} \tag{2.3.13}$$

  We begin this integration process with $\hat{x} = \hat{x}_{k-1}^+$ and $P = P_{k-1}^+$. At the end of this integration we have $\hat{x} = \hat{x}_k^-$ and $P = P_k^-$.

- At time k, incorporate the measurement y_k into the state estimante and estimation covariance as follows:

$$\begin{array}{rcl} K_k & = & P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \\ \hat{x}_k^+ & = & \hat{x}_k^- + K_k(y_k - h_k(\hat{x}_k, 0, t_k)) \\ P_k^+ & = & (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k M_k R_k M_k^T K_k^T \end{array} \tag{2.3.14}$$

  H_k and M_k are the partial derivatives of $h_k(x_k, v_k)$ with respect to x_k and v_k, and are both evaluated at $\hat{x}_k^-$.

Chapter 3

Modelling and Sensing

This section first describes the legged robot HyQ, that was used in this work, as well as its reference frames, dynamic model and its sensors. The second part of this section describes the locomotion control software, used for the thesis.

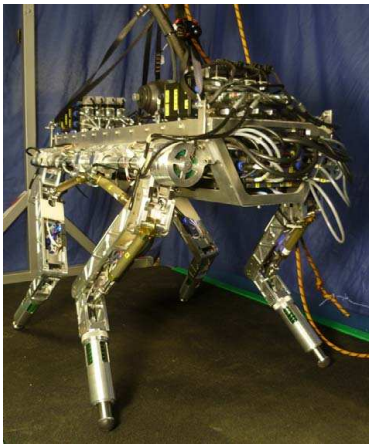


Figure 3.1: HyQ robot ([5])

3.1 Quadruped Robot HyQ

HyQ (Fig. 3.1) is a fully torque-controlled Hydraulically actuated Quadruped robot developed in the Department of Advanced Robotics at the Istituto Italiano di Tecnologia. HyQ stands 1 meter tall and weighs approximately 90 kg. It has 12 torque-controlled joints powered hydraulic actuators. This combination allows to perform powerful motions, hard to achieve with the traditional electrically actuated robots. HyQ is designed to navigate over rough terrain and perform highly dynamic tasks such as jumping and running with different gaits. For more information about the robot the reader can refer to [5]. HyQ consists of a torso and four identical legs, arranged in the *forward/backward* configuration, with the front and hind knees point to each other. The four legs are defined as follows: (i) LF is the left front leg (ii) RF is the right front leg (iii) LH is the left hind leg (iv) RH is the right hind leg.

There are three actuated joints for a total of 12 active DoF, and 4 passive DoF. The actuated joint are:

- HAA: Hip Abduction/Adduction, between the torso and hip assembly. This joint is responsible for lateral leg motion.
- HFE: Hip Flexion/Extension, between hip assembly and upper leg. It allows the motion that brings the knee closer/further from the HAA.
- KFE: Knee Flexion/Extension, between upper and lower leg. It allows motion in the same plane of HFE.

Curious reader can find a summary of technical specification of the robot in Appendix A.

3.2 Robot Modelling

This section describes first the reference frames of HyQ, followed by the dynamic model of the robot.

Reference Frames

We define the following reference frame: the body frame $\mathcal{B}$ located at the geometric center of the trunk (robot torso) and the world frame $\mathcal{W}$, an inertial frame whose origin coincides with a fixed point on Earth. As shown in the Fig. 3.2, the basis of the body frame are orientated forward, left and up. When the robot is in its starting position, the base frame coincides with the world frame, considering an offset along z-axis of exactly the height of the robot.

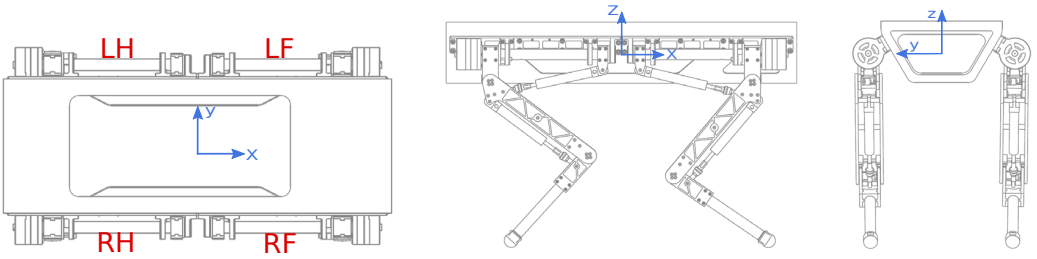


Figure 3.2: Location on Torso and Robot Base Coordinate Frame. From left to right: top view, side view, back view

Dynamic Model

The dynamics of the robot is obtained starting from the assumption that all of the external forces are exerted on the feet and the joint accelerations are zero [29]. When a foot is not in contact with the terrain it is in the *swing* phase and it is assumed that no external forces are exerted on it. When the contact occurs, the foot is in *stance* phase, and the ground exerts forces on it (Ground Reaction Forces GRFs). The equation is:

$$M(\bar{x})\ddot{\bar{x}} + h(\bar{x}, \dot{\bar{x}}) = \bar{\tau} + J^T F_{grf} \tag{3.2.1}$$

where $\bar{x} = [x^T \eta^T q^T]^T \in R^{18}$ is the generalized robot states, $\dot{\bar{x}} \in R^{18}$ is the corresponding generalized velocities, $\ddot{\bar{x}} \in R^{18}$ is the corresponding generalized accelerations, $x \in R^3$ is the position of the base, $\eta \in R^3$ is the attitude of the base, $q \in R^{12}$ is the vector of joint angles of the robot, $M \in R^{18\times 18}$ is the joint-space inertia matrix, h is the vector of Coriolis, centrifugal and gravity forces, $\bar{\tau} = ([0 \quad \tau^T]^T) \in R^{18}, \tau \in R^{12}$ is the vector of actuated joint torques. $F_{grf} \in \mathbb{R}^{12}$ is the vector of GRFs, the forces exerted by the terrain when there is the contact foot-ground. $J \in R^{18\times 12}$ is the floating base Jacobian.
For the purposes of this thesis, it is fundamental to know the contact state of the robot. The contact states for each foot are given by the contact status α and the GRF F_{grf}. The contact states are estimated by the state estimator: the GRFs from the torques and the joint states [29]. The contact status α is boolean variable, defined for each leg, whose value is 1 when the foot is in contact, 0 otherwise. The contact ($\alpha = 1$) is detected when the GRFs F_{grf} exceed a threshold F_{min}:

$$F_{grf} = \alpha J_{st}^{-T}(M(\bar{x})\ddot{\bar{x}} + h(\bar{x}, \dot{\bar{x}}) - \bar{\tau}) \tag{3.2.2}$$

Then

$$\alpha = \begin{cases} 1 & J_{st}^{-T}(M(\bar{x})\ddot{\bar{x}} + h(\bar{x}, \dot{\bar{x}}) - \bar{\tau}) > F_{min} \\ 0 & otherwise \end{cases} \tag{3.2.3}$$

The translational and rotational kinematics, and the translational dynamics of the robot as a single rigid body in $\mathcal{W}$ are

$$\dot{x}^w = v^w \quad \dot{v}^w = a^w + g^w \quad \dot{R}_b^w = R_b^w S(\omega^b) \tag{3.2.4}$$

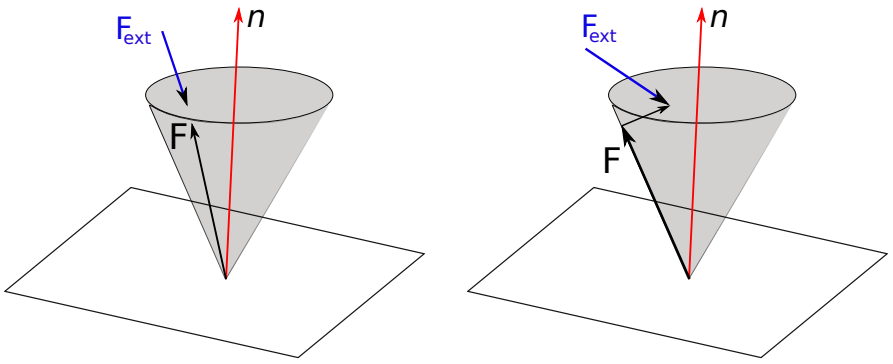
where $x^w \in R^3$, $v^w \in R^3$, $a^w \in R^3$ are the position, velocity and acceleration of the base in $\mathcal{W}$, respectively, $R_b^w \in SO(3)$ is the rotation matrix from $\mathcal{B}$ to $\mathcal{W}$, and ω^b is the angular velocity of the base in $\mathcal{B}$. The skew symmetric matrix function is $S()$.

Friction Cone

The purpose of this thesis is to investigate the problem of slip detection, proposing an effective and easily applicable solution to quadruped robots. To better understand what happens to the leg when it slips it is useful to understand the dynamics of the slippage.

We can consider a point foot on a frictional plane (3.3). In 3.3a the GRF F is able to balance the external force F_{ext} and the foot velocity in the world frame is $v = 0$. The terrain can balance only with a GRF F constrained to lie on the boundary of the cone satisfying $||\tilde{F}_t|| = \mu\|F\|$. When an external load, which requires a force out of the friction cone to be balanced, is applied the foot starts moving because there is a force accelerating it ($v \neq 0$). In this situation (fig. 3.3b) the foot is slipping.

Simplifying the problem we could say that, when the foot is in contact with the ground, during the contact its velocity expressed in an inertial frame ($\mathcal{W}$) should be zero. If this does not happen, then a slip is in progress. More details are described in Chap. 4



(a) (b)

Figure 3.3: Slip dynamics: the friction cone

3.2.1 Onboard Sensors

The quadruped robot is equipped with a six-axis IMU on the trunk (3 DoFs gyroscope and 3 DoFs accelerometer), and every joint contains an encoder and torque sensor. The accelerometer measures specific force $f_s^b \in R^3$.

$$f_s^b = a^b + g^b \tag{3.2.5}$$

where $a^b \in R^3$ is the acceleration of the body in $\mathcal{B}$ and $g^b \in R^3$ is the acceleration due to gravity in $\mathcal{B}$. The gyroscope directly measures angular velocity $\omega^b \in R^3$ in $\mathcal{B}$. The encoders are used to measure the joint position $q_i \in R$ and joint speed $\dot{q}_i^R$. The pose of each joint is assumed to be exactly known. The torque sensors in the joints directly measure torque $\tau_i \in \mathbb{R}$. The measured values af all the sensors contain a bias and noise: $\tilde{x} = x + b_x + n_x$ where $\tilde{x}$, b_x, and n_x are the measured value, bias and noise of x, respectively. For the model we assume that all of the biases are constant or slowly time-varying, and all of the noise variables have zero mean and a Gaussian distribution.

HyQ is equipped with *proprioeptive* and *exteroceptive* sensors. For the experiments done during this project of thesis, the robot is completely blind and the informations are obtained exclusively from *proprioceptive* sensors data.

In the following a brief description of the sensors the robot is equipped with [52].

The *proprioceptive* sensors mounted on HyQ are:

- Encoders measuring the joints positions q and velocity $\dot{q}$, used to evaluate position, velocity and acceleration of the knee and of the foot. In each joint there are two encoders:
  - Absolute encoders (AMS Programmable Magnetic Rotary Encoder - AS5045) used to measure the joint position when the robot is first turned on.
  - Relative/optical (Avago Ultra Miniature, High Resolution Incremental Encoder - AED-3300-TE1) used to measure how far the joint has moved at each step.
- In the trunk of the robot there are two IMUs: a military grade fibre optic KVH-1775 and a MEMS-based high-end consumer grade Lord MicroStrain 3DM-GX5-15

- Every joint contains a F/T sensor:
  - HFE and KFE joints are moved by pistons, so they are equipped with *loadcells* (Burster Subminiature Load Cell - 8417-6005) measuring forces.
  - HAA joint is equipped with a custom made torque sensor based on strain-gauges, detecting the torque τ.

The *exteroceptive* sensors mounted on HyQ are:

- RGB-D cameras (Realsense)
- LIDARs mainly used for SLAM purposes (Velodyne puck)

3.3 Controller

HyQ can perform different kind of locomotion. The locomotion framework used in this work is based on the *Reactive Controller Framework RCF* described in [53]. In the following a brief description of the RCF is presented.

3.3.1 Reactive Controller Framework

The RCF is mainly used for uneven and rough terrain thanks to its capability of performing robust dynamic locomotion. RCF is a modular framework that was first designed to receive informations only provided by the *proprioceptive* sensors, then its usage was extended considering *exteroceptive* sensors as well.

The framework comprises two main modules: a *motion control* block and a *motion generation* block. The *motion generation* block assigns the trajectories of the feet with respect to the *horizontal frame* $\mathcal{H}$, a frame whose xy-plane is always horizontal (this frame has the same yaw angle as the robot with respect to the world frame $\mathcal{W}$). Choosing this frame, the generation of the trajectories is independent from the trunk attitude, so it is possible to deal with non-flat terrain.

The motion control module provides corrective actions in order to obtain the desired trunk motion. RCF contains some other modules used for increasing the reliability of robot locomotion in difficult terrains (i.e. push recovery, foot collision detection, terrain adjustment etc.). The contribution of this thesis is to improve the performance of the *state estimation* paying attention to the problem of the *slip detection*. An overview of the RCF framework is presented in Fig. 3.4.

Trot and Crawl

HyQ can perform two locomotion behaviours [54]: a dynamically stable trot and a crawl-gaited style. The first allows fast locomotion over regular terrain varying inclination. The other one is more suitable for irregular and non-continuous terrains.

During the trotting the robot moves its legs in pairs: when LF and RH are in stance phase, the other two are in swing, and vice-versa. The trotting controller is able to trot

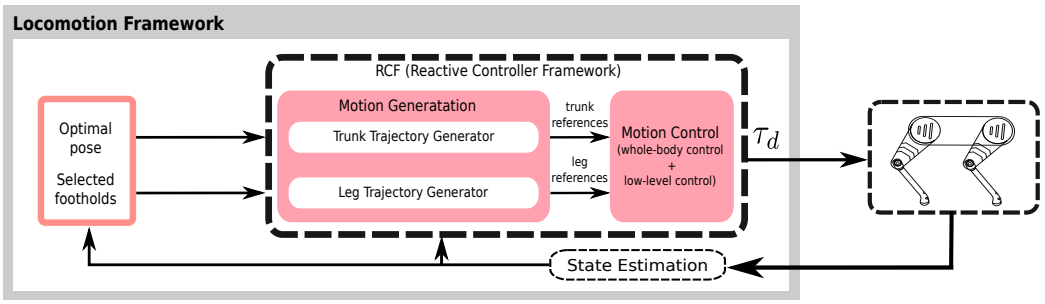


Figure 3.4: Overview of the locomotion framework.

with a speed up to 0.5 m/s. It allows the robot to overcome obstacles up to 5 cm without difficulty.

The crawl-gait is a slow locomotion. The velocity of the trunk never exceeds 0.05 m/s. In this case the gait is cyclic and utilizes a static stability criterion to produce the quadruped's walking pattern. The pattern is: LH to LF to RH to RF.

For the simulation and experimental tests (Chap. 5 and 6) we used both these methods of locomotion.

Chapter 4

Methods

The method of slip detection proposed in this thesis uses as baseline approach in *Focchi et al.* [43]. The authors introduce a methodology for the slip detection of quadruped robots and a recovery strategy from slippage during locomotion. In this chapter we briefly introduce this approach and then we move on the explaination of a novel method of slip detection. In Chapter 5 and 6 we compare the results obtained from baseline approach and the novel algorithm.

4.1 Baseline Approach

The proposed strategy to address the problem of slip detection in [43] is based on kinematics. They implemented two different strategies at the velocity level: (i) an approach to distinguish the slippage of only one leg based on $\dot{x}_f^b$, the stance feet velocities in the body frame ($\mathcal{B}$) ; (ii) a further approach for the case of two or more slipping legs, using $\dot{x}_f^w$, the stance feet velocities in the world frame ($\mathcal{W}$). The procedure for obtaining the robot legs kinematics is described in the appendix B.

4.1.1 One leg slip detection

The strategy proposed in [43] planned to compare the values of stance feet velocities in the body frame $\dot{x}_f^b$ and discriminate the outlier with appropriate statistical tools. At each control loop the median of the norms of the stance feet velocities is computed. A slipping leg should be the one whose velocity deviates the most from the median, beyond a certain threshold ϵ, tuned experimentally. During the locomotion the detection algorithm is continuously checking, within the set of active stance legs, if there is any slippage. Each leg has a flag β associated with it. Whenever a slip is detected, the value of β instantaneously switches from 0 to 1.

Pseudo-code implementation of the slip detection for one leg of a legged robot:

Algorithm 1 detectSlippageOneLeg($\alpha, \left\|\dot{x}^b_{f_i}\right\|$) [43]

for each stance leg i **do**
    $\|v\|_i \leftarrow \left\|\dot{x}^b_{f_i}\right\|$;
end
$M \leftarrow median([\|v\|])$;
for each stance leg i **do**
    $\beta_i \leftarrow |\|v\|_i - M| > \epsilon$;
end

where $\alpha \in \mathbb{R}$ is the contact status (Eq. 3.2.3), $\|v\|_i \in \mathbb{R}$ is the norm of the i-th stance foot velocity, $\dot{x}^b_{f_i} \in \mathbb{R}^3$ is the velocity of foot i expressed in the base frame $\mathcal{B}$ and $[]\|v\|] \in \mathbb{R}^4$ is the vector of the stance feet velocities norms.

4.1.2 Multiple leg slip detection

A more complicated situation is when two or more legs are slipping at the same time. In this case, it is hard to detect with the median approach which legs are slipping or in stance. In [43] they propose of checking which of the feet velocities $\dot{x}_f$ are kinematically consistent with the base velocity $\dot{x}^b_b$. This can help to discriminate the slipping legs. In this case the most intuitive way is to verify that the Cartesian velocities of the stance feet $\dot{x}^w_f$ are all *zero* in an inertial frame $\mathcal{W}$. $\dot{x}^w_f$ can be written as follows:

$$0 \simeq \dot{x}^w_f = \dot{x}^w_b + R^w_b(\dot{x}^b_f + \omega^b_b \times x^b_f) \tag{4.1.1}$$

where $R^w_b \in \mathbb{R}^{3\times3}$ is the rotation matrix representing the orientation of the robot base, $x^b_f \in \mathbb{R}^3$, $\dot{x}^b_f \in \mathbb{R}^3$ are, respectively, the position and velocity of the foot expressed in the base frame $\mathcal{B}$, ω^b_b is the angular velocity, measured by an on-board IMU sensor, while $\dot{x}^w_b \in \mathbb{R}^3$ is the base linear velocity. Accordingly to (4.1.1), computing $\dot{x}^w_f$ requires an estimation of the base linear velocity $\dot{x}^w_b$. For this purpose, a short-time integration of the base linear acceleration, measured by the IMU accelerometers, is executed. Afterward in [43] the authors proposed a strategy for the slip recovery after the detection phase. This strategy is based on the idea of correct the surface normal toward the estimated one resulting in GRFs which were back inside the real friction cone [55].

The slip detection method previously described has some down sides. It was tested using only *crawl* (chap. 3.3.1) as gait type. The *One leg slip detection* method, clearly cannot be applied to different locomotion, e.g. *trotting*, because in this case the legs have pairwise different velocities (two legs are in stance phase, the other two are in swing phase) and the accuracy might deteriorate. On the other hand, the *Multiple leg slip detection* method relies on velocities expressed in the *world frame*. The base velocity in the world frame is influenced by errors in the state estimation, which can result into false positives in the slip detection. Moreover, this approach relies on IMU and on the integration of the acceleration measured by it. It is known that integrating accelerometers is prone to

drift and, furthermore, it is fundamental to perform a *short-time* integration to avoid divergence issues.

This method was tested only in simulation. The authors did not perform experiments on a real robot, therefore it cannot be said with certainty that it is actually robust.

The abovementioned are the main problems we set out to solve by proposing a new slip detection algorithm. The main reason that led us to choose the algorithm just described as a baseline is that also the *new slip detection method* exploits the kinematics of the robot. We think that a kinematics approach is more suitable to address the problem of slip detection for legged robots than a force based approach. The force based approaches require the availability of force sensors located at the contact point (e.g. foot-tip). Repetitive impacts with the ground can cause the damage of the sensor. Furthermore, during locomotion, the touchdown event can create discontinuities in the force signals and generate false positives in the detection.

A detailed description of the novel slip detection algorithm is proposed in the next section.

4.2 A novel approach for slip detection

The idea behind the proposed method is to overcome the aforementioned problems, using the robot state $\bar{x} = [x^T \eta^T q^T]^T \in R^{18}$ to perform a reliable slip detection; $x \in R^3$ is the position of the base, $\eta \in R^3$ is the attitude of the base, $q \in R^{12}$ is the vector of joint angles of the robot. As already mentioned, the most intuitive way to detect slippage is to check if the stance foot velocity is not zero. But computing the velocity in the world frame requires the estimation of the base linear velocity (4.1.1). To estimated $\dot{x}_b^w$ we need to integrate the base linear acceleration $\ddot{x}_b^w$, measured by the IMU, running into drift problems. For this reason we move to the idea of detect the slippage using feet velocities expressed in the base frame $\mathcal{B}$. In fact, a leg subject to slippage would be the one whose velocity at the foot-level would deviate more from the desired one. An approach based on the feet velocities in the base frame is more robust because they directly depend on direct sensor measurements (e.g. encoders). A measure of the deviation from desired velocity can be given by:

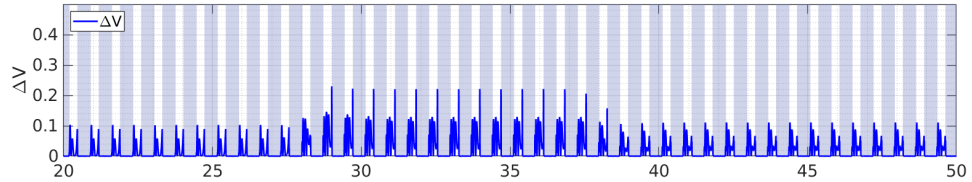
$$\Delta V = \left\| {}_d\dot{x}_f^b - \dot{x}_f^b \right\| = \sqrt{\sum_{i=x,y,z} \left({}_d\dot{x}_{f_i}^b - \dot{x}_{f_i}^b \right)^2} \tag{4.2.1}$$

where n = 3 are the components of the velocity vector $\dot{x}_f^b = [\dot{x}_{f_x}^b \dot{x}_{f_y}^b \dot{x}_{f_z}^b]^T \in \mathbb{R}^3$. ΔV is the norm of the difference between the desired and actual feet velocities in the base frame $\mathcal{B}$. We could think of detecting a slip, during the stance phase, when the ΔV value exceeds a certain limit (e.g. a threshold ϵ). Actually, just calculating the norm is not a reliable tool for detecting slippage. We are not considering that the difference between the desired and actual velocity increases along the prevalent direction of motion because of a greater tracking error. Furthermore, if in general we want to change the velocity of the robot during its motion, we would need to adjust the threshold value ϵ accordingly (Fig. 4.1a). So to ensure that the direction and velocity of motion do not affect the calculation of ΔV we introduce a "weight"to scale each component of the vector ${}_d\dot{x}_f^b - \dot{x}_f^b$:

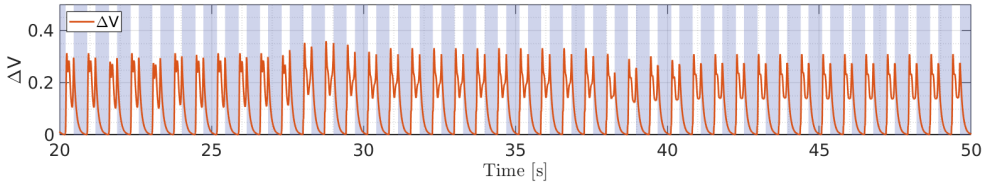
$$\overline{\Delta V} = \sqrt{\sum_{i=1}^{n} \left(\frac{{}_d\dot{x}_{f_i}^b - \dot{x}_{f_i}^b}{{}_d\dot{x}_{f_i}^b} \right)^2} \tag{4.2.2}$$

Through the division by each component of ${}_d\dot{x}_f^b$, we minimize the impact of a possible greater tracking error in one of the three directions and, above all, we ensure to have a $\overline{\Delta V}$ that remains more or less similar for the duration of the motion, even if we increase the desired body velocity (Fig. 4.1b). For a purely mathematical question, that is to avoid divisions by quantities close to 0, a margin m is added to the denominator. Its value can be experimentally tuned. For the same reason, we calculate $|{}_d\dot{x}_{f_i}^b|$ in order to ensure that the denominator is always $\neq 0$. In the end we have:

$$\overline{\Delta V} = \sqrt{\sum_{i=1}^{n} \left(\frac{{}_d\dot{x}_{f_i}^b - \dot{x}_{f_i}^b}{|{}_d\dot{x}_{f_i}^b| + m} \right)^2} \tag{4.2.3}$$



(a) ΔV during a simulation task of trotting (no slippage)



(b) $\overline{\Delta V}$ after the scaling during the same task of trotting (no slippage)

Figure 4.1: Trends of ΔV and $\overline{\Delta V}$ with and without scaling in a simple trotting task with variable feed rate

When a slippage occurs, the value of $\overline{\Delta V}$ increases. For this reason we can impose a limit to $\overline{\Delta V}$, beyond which the further increase is considered a slip:

$$\overline{\Delta V} = \sqrt{\sum_{i=1}^{n} \left(\frac{{}_d\dot{x}^b_{f_i} - \dot{x}^b_{f_i}}{|{}_d\dot{x}^b_{f_i}| + m} \right)^2} > \epsilon_v \tag{4.2.4}$$

ϵ_v is a threshold that we can call "conditioned". Its value, in fact, is set taking into account the phase of the foot motion for each leg i: a leg cannot slip during the *swing phase*, so ϵ_v can be set to ∞. During the *stance phase*, on the other hand, it is assumed that the value of $\overline{\Delta V}$ increases more whereas a slip occurs. So, in this case, ϵ_v is chosen as the level below which about a certain percentage p of the $\overline{\Delta V}$ value falls. P is manually tuned.

$$\epsilon_v = \begin{cases} \infty & swing\,phase \\ percentile(\overline{\Delta V}, p) & stance\,phase \end{cases} \tag{4.2.5}$$

In case of slippage, another important question is: *How far we slip?* To answer, once again we can use robot kinematics. A quantitative measure of the slipping length can be derived from the measurement of the foot *position*. Indeed, during slippage the foot position changes by deviating from the desired one. This deviation can be quantified by calculating:

$$\Delta P = \left| \left\| {}_d x^b_{f_i} \right\| - \left\| x^b_{f_i} \right\| \right| \tag{4.2.6}$$

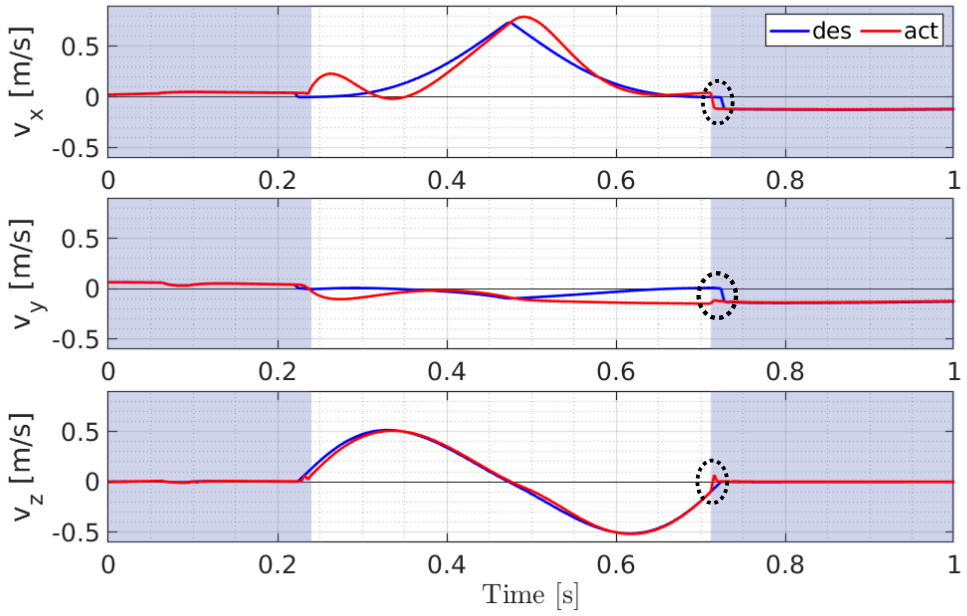


Figure 4.2: Desired (blue line) and actual (red line) foot velocity in a simulation task of crawling (no slippage). The light gray background indicates that the foot is in stance.

ΔP is also important for the slip detection for another main reason: at the beginning of a foot contact with the ground there is a little time interval, in which the difference between desired and actual foot velocities does not instantaneously decrease (4.2). This may be due to multiple causes: (i) an actual, but small and perhaps not important slippage, (ii) delay in control and (iii) the current implementation of stance detection (Chap. 3.2). The current measurement of the contact states of each leg at every time step is required. The contact state is given by *contact status* α, and GRFs. In order to detect foot slipping, we are primarily interested in contact status α, a variable that detects if a leg is in contact with the ground or not. The contact is detected when the GRF F_i exceed a certain value F_{min}. A GRF F_i for each leg i is computed using the actuated part of the dynamics (Chap. 3.2, Eq. 3.2.1), basing on two important assumptions: (i) there are no external forces on the swing-foot and the only force exerted on the stance-foot is the reaction exerted by the ground, (ii) we set accelerations to zero to calculate the inverse dynamics:

$$F_i = -\alpha_i (J_i^T(q_i))^{-1}(\tau_i - h_i(\bar{x}_i, \dot{\bar{x}}_i)) \tag{4.2.7}$$

where $J_i \in \mathbb{R}^{3x3}$ is the foot Jacobian for the $i-th$ leg, $\tau_i \in \mathbb{R}^3$ is the vector of joint torques of i, $h_i \in \mathbb{R}^3$ is the vector of centrifugal, Coriolis, gravity torques of i and $\alpha \in [0, 1]$ is the contact status selecting if the foot is on the ground ($\alpha = 1$) or not ($\alpha = 0$). A threshold of F_i is used to calculate α_i . Hence:

$$\alpha_1 = \begin{cases} 1 & \left\| (J_i^T)^{-1}(\tau_i - h_i) \right\| > F_{min} \\ 0 & otherwise \end{cases} \tag{4.2.8}$$

where $F_{min} > 0$. We are assuming that there exists a force threshold F_{min} that determines if the foot is in contact with the environment. This means that there is a small delay in detecting contact with the ground, and as can be clearly seen from the Fig. 4.2, at the beginning of the stance phase the error between the desired foot velocity and the actual one increases for a very short interval of time. Consequently this can facilitate a little slippage of the foot in this time interval, which however is considered negligible ($\sim$ 1 , 2 cm). To make sure it is not detected by our algorithm we add a condition on ΔP:

$$\Delta P = \left| \left\| {}_d x^b_{f_i} \right\| - \left\| x^b_{f_i} \right\| \right| > \epsilon_p \tag{4.2.9}$$

If the value of ΔP remains below a threshold ϵ_p, the slip is considered acceptable. ϵ_p has a constant value, tuned experimentally. Therefore, a slippage is detected when both $\overline{\Delta V}$ and ΔP exceed their respective thresholds.
We can introduce a flag $\beta_i \in [0, 1]$ for each leg i whose value is 1 if there is a slip detection, 0 otherwise.

Having said this, the pseudo-code implementation of the proposed algorithm is:

Algorithm 2 detectSlippage($\dot{x}^b_{f_i}$,${}_d\dot{x}^b_{f_i}$,$x^b_{f_i}$,${}_d x^b_{f_i}$)

$\overline{\Delta V} \leftarrow scaled\, diff(\dot{x}^b_{f_i}, {}_d\, \dot{x}^b_{f_i})$;
$\Delta P \leftarrow diff(x^b_{f_i}, {}_d\, x^b_{f_i})$;
for each stance leg i **do**
    $\beta_i \leftarrow (\overline{\Delta V} > \epsilon_v)\, \&\, (\Delta P > \epsilon_p)$;
end

where $\dot{x}^b_{f_i}$ and ${}_d\dot{x}^b_{f_i}$ are the *actual* and *desired* foot velocity for each leg i in $\mathcal{B}$, while $x^b_{f_i}$ and ${}_d x^b_{f_i}$ are the *actual* and *desired* foot position for each leg i in the *base frame* $\mathcal{B}$. ϵ_p is the threshold for ΔP. ϵ_v is the threshold for $\overline{\Delta V}$.

To summarize:

- The approach proposed to address the problem of slip detection is based on the kinematics of the legs: for each foot we obtain the desired and actual velocity and position expressed in the base frame.
- We compute $\overline{\Delta V}$ and ΔP, two quantities indicative of how much the slippage causes a deviation from "normal"conditions
- We threshold $\overline{\Delta V}$ and ΔP taking into account the contact status
- For each leg we obtain a flag that changes its value from 0 to 1 *if* and *when* a slip occurs.

This method has several advantages. It detects the slippage basing on feet velocities and positions expressed in $\mathcal{B}$. In this way we avoid al the drift-related problems that arise during state estimation. Errors in the state estimation can affect detection by generating false positives. The approach can be used to detect the slippage of one or multiple slipping

legs indiscriminately. It can be adapted to different types of locomotion, because each foot is treated separately from the others. We show this in Chapter 5.

In Chapters 5 and 6 there are the simulation and experimental results, and a comparison with the baseline approach.

Chapter 5

Simulations Results

In order to demonstrate the effectiveness of our implementation, we carried out a set of tests, simulating the dynamic model of HyQ walking on very slippery surfaces. Namely, ice slabs, with a dynamic friction coefficient μ equal to 0.08, corresponding to the friction between tire and ice. First of all a C++ implementation of the proposed algorithm was made. Then we created a simulation scenario in Gazebo where we required HyQ to step over ice slabs, while crawling and trotting. To check in a fast way and to tune the parameters of the proposed algorithm we used the tool PlotJuggler, that allows to load data from files and connect to live streaming of data. In this case it was used to open rosbags and subscribe to ROS topics. Then, all the datas were collected and post-processed on Matlab.

In this simulation tests, the motion generation and the control is done by the Reactive Controller Framework (RCF). The RCF has a trunk stabilization module that uses leg contact Jacobians, that, by default considers all contacts happening at the foot level. The trunk controller also uses the information about the surface inclination and the friction coefficient to generate GRFs that respect the associated friction cone.

We perform the tests presented in this chapter also using the baseline approach (Chap. 4.1), in order to make a comparison with the new proposed method and underline its strengths.

5.1 Crawl

5.1.1 One Leg on an ice-patch

A transition from walking on flat terrain to an ice slab ($\mu = 0.08$) is a good template to demonstrate the effectiveness of the algorithm. Fig. 5.1 shows the sequence of a movement: the robot starts walking from "normal "ground and then, moving forward, the robot walks on a ice-sheet. In this first simple demonstration the leg LF is on the slippery ice slab and therefore only the respective foot slides for a few moments.

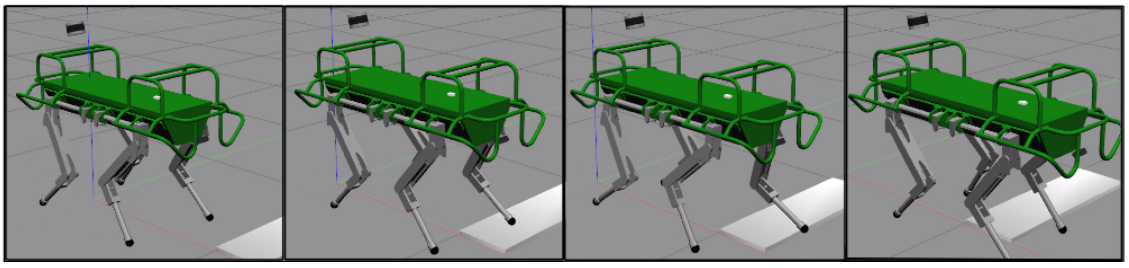


Figure 5.1: HyQ walking to an ice slab.

Aim of the slip detection is not only to detect slippage in the first moments in which it occurs, but also to be able to identify with some certainty which leg(s) is(are) slipping. For this simulation test we used the following parameters:

ϵ_v	$prctile(\overline{\Delta V}, 99\%)$
ϵ_p	0.03
m	0.3

Table 5.1: Parameters of the simulation test: one leg on ice patch

ϵ_v is the threshold for $\overline{\Delta V}$ ((4.2.4)), ϵ_p is the threshold for $\overline{\Delta V}$ ((4.2.9)), m is the margin used in (4.2.3). In Fig. 5.2 the shapes of ΔV and ΔP are shown. They were computed as explained in Chapter 4.2 and then filtered using a first-order low pass filter. The light blue stripes indicate that the contact status is one (foot in contact with the terrain). The light red stripes indicate the slippage. From the figure it is clear that the slippage occurs between the second and third second. We can see that the change in velocity in the sliding leg (LF) affects the $\overline{\Delta V}$ shapes of the other legs as well. This is because the velocities are expressed in the base frame $\mathcal{B}$ and means that, when a foot is slipping, the base changes its position for some instants. Then the velocity vectors of the other legs change (top figures in each subplots). Adding a further constraint on ΔP (Eq. 4.2.9) prevents this from affecting the correct detection for the other legs. Fig. 5.3 shows the flags (red) that identifies the sliding.

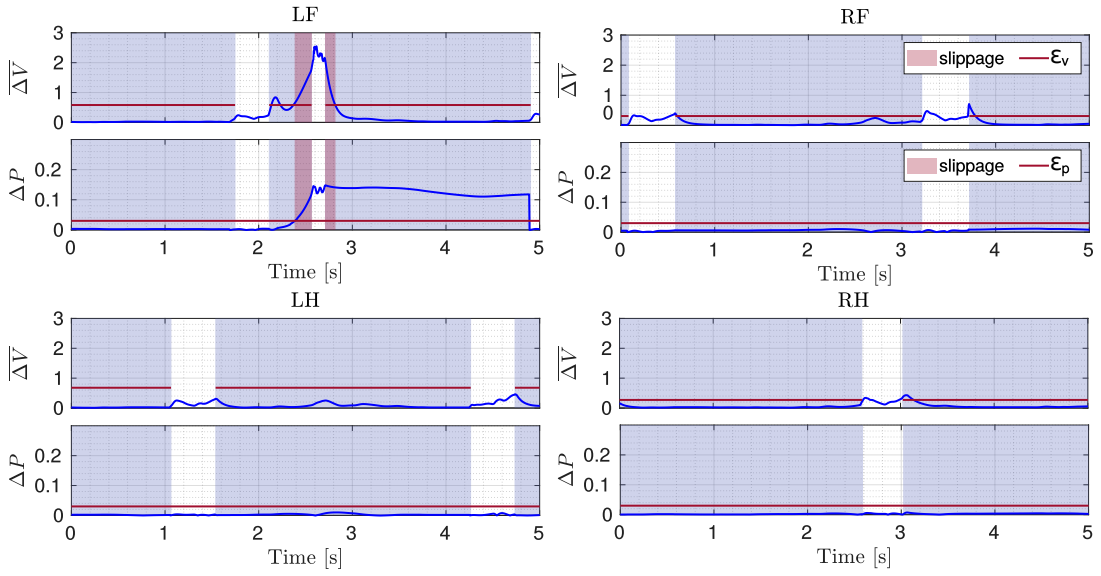


Figure 5.2: Shapes of ΔV and ΔP with the respective thresholds.

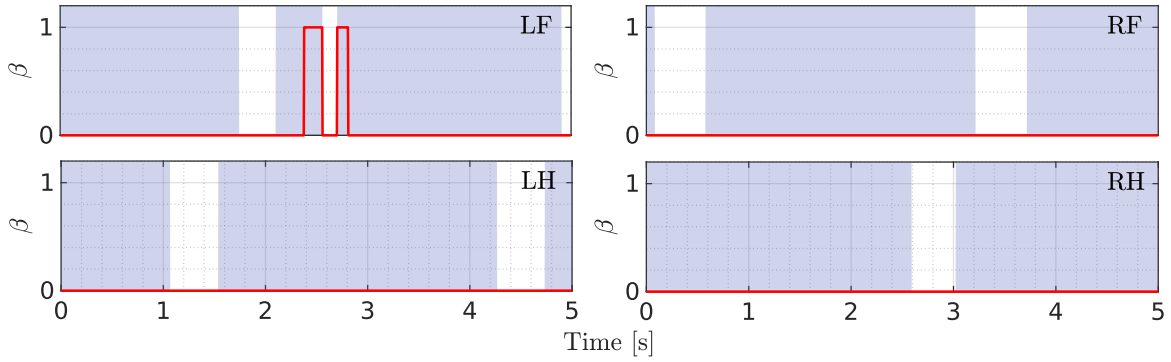


Figure 5.3: Flags indicating slippage when the robot move to ice slab

It is clear that the slip detection is correctly done: the flags switch from zero to one for LF, while the other three flags remain, as expected, at zero for the duration of the simulation.

To better understand when the detection occurs, in Fig.5.4 the position along the x-axis and y-axis for LF. The red markers indicate that the slipping flag switch to the value one because the slippage is occurring.

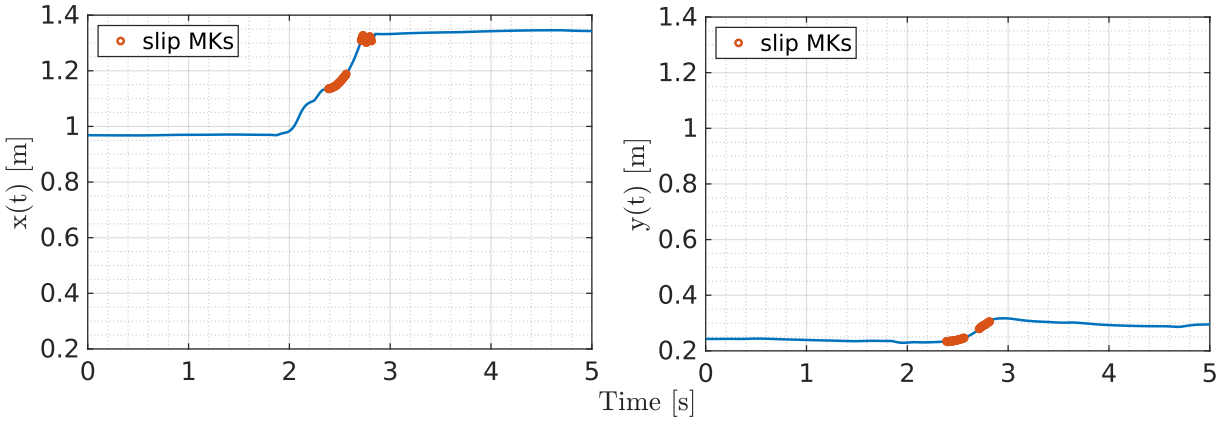


Figure 5.4: LF foot position along x-axis and y-axis during the simulation. Red markers MKs indicate the slippage

The trajectory in fig. 5.4 is obtained plotting $x_f^w \in \mathbb{R}^3$ the foot position in the world frame $\mathcal{W}$: $x_f^w = x_b^w + R_b^w x_f^b$, where $x_b^w \in \mathbb{R}^3$ is the position of $\mathcal{B}$ with respect to $\mathcal{W}$. $R_b^w \in \mathbb{R}^{3\times3}$ is the rotation matrix representing the orientation of $\mathcal{B}$. $x_f^b \in \mathbb{R}^3$ is the foot position in $\mathcal{B}$. Red markers (MKs) indicate that a foot slip has occurred.

5.1.2 Comparison with the baseline approach

As already mentioned we implemented the algorithm presented in [43] to make a comparison between the proposed approach and an already existing method. For this purpose we implemented the algorithm in C++ and used PlotJuggler for the tuning. Then we took the data recorded from the previous simulation test (Chap. 5.1.1) in order to have the same scenario in which test both the implementations. In this simulation the slippage interested only one leg (LF) of the robot. This is the reason why we implemented the algorithm described in Chap. 4.1 (*Detection of the slippage of one leg*). To briefly summarize the working principle of the algorithm: we compute the stance feet velocities in the base frame, then we compute the median of these velocities. A flag with value 0 is associated to each leg. If the velocity of one stance foot deviates from the median beyond a certain threshold, the flag value switch to 1 and the slippage is detected for this leg. In the following figures we call ΔV the expression $\left|\left\|\dot{x}_f^b\right\| - M\right|$, where $\dot{x}_f^b \in \mathbb{R}^3$ is the foot velocity in $\mathcal{B}$ and M is the median of the stance feet velocities. The threshold is $\epsilon_v = 0.4$. The slippage occurs between the second and third second. During this interval ΔV increases until it reaches a maximum peak (Fig. 5.5).

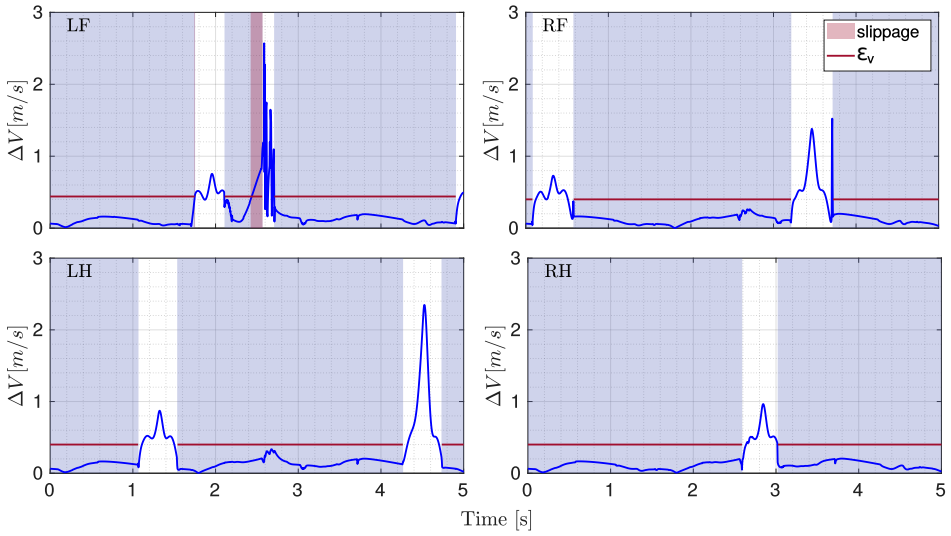


Figure 5.5: Shape of ΔV with the respective threshold. From top-left to bottom-right: LF - RF - LH - RH

In Fig. 5.6 we see that the LF-flag indicating slippage changes correctly its value from 0 to 1 in the time interval in which the slippage occurs. As expected, the flags associated with the other legs remain correctly at 0. However, we note that the flag goes instantaneously to the value 1 two times before the start of the real slipping. We also note that this happens at the beginning and at the end of the stance phase. This should be due to the fact that ΔV does not always instantly go below the threshold. The reason could be in the current implementation of contact detection. Since there are no further constraints other than the one on ΔV, it is very likely to incur false positives and therefore detect a slip even where there is actually no.

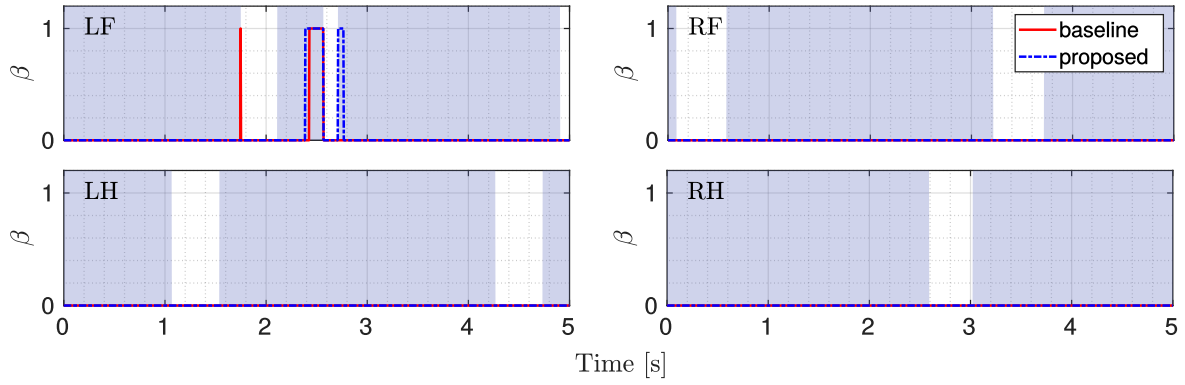


Figure 5.6: Flags indicating slip detection

In Fig. 5.7 we show how the LF-foot position changes during the simulation. From this figure we can clearly see that the sliding occurs mainly in the y-direction (subplot on the right). The yellow markers (MKs_{BL}) indicate that the baseline algorithm has detected a slippage. The red markers (MKs_{NEW}), represent the slip detection performed by the proposed algorithm.

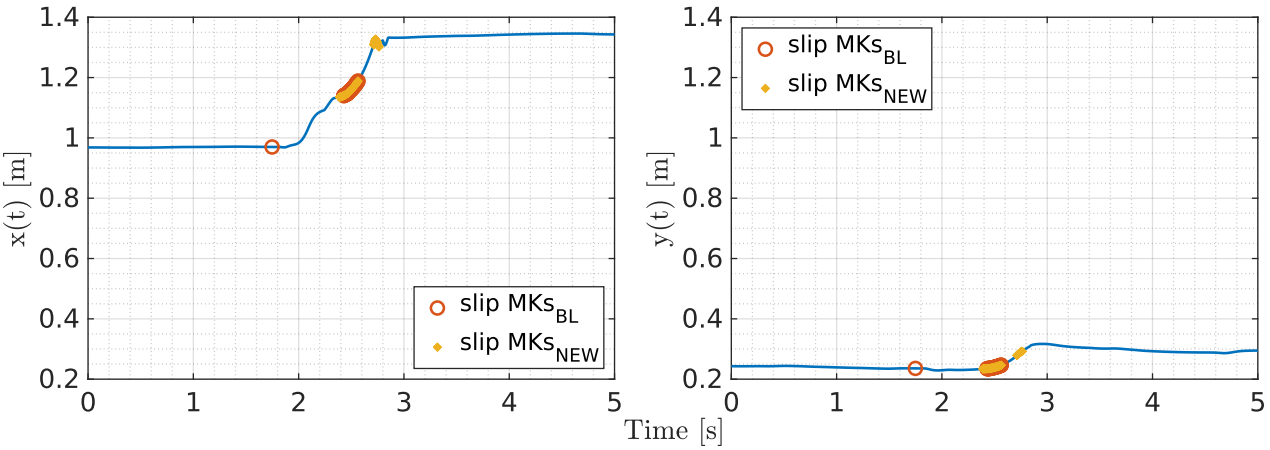


Figure 5.7: LF leg position along x-axis and y-axis during the simulation

5.1.3 Multiple Legs on ice-patches

A more challenging situation is when the robot walks to ice slabs and, during the motion, all the legs are slipping. As it is possible to see from Fig. 5.8 the slippage affects legs in the order LF-RF-LH-RH.

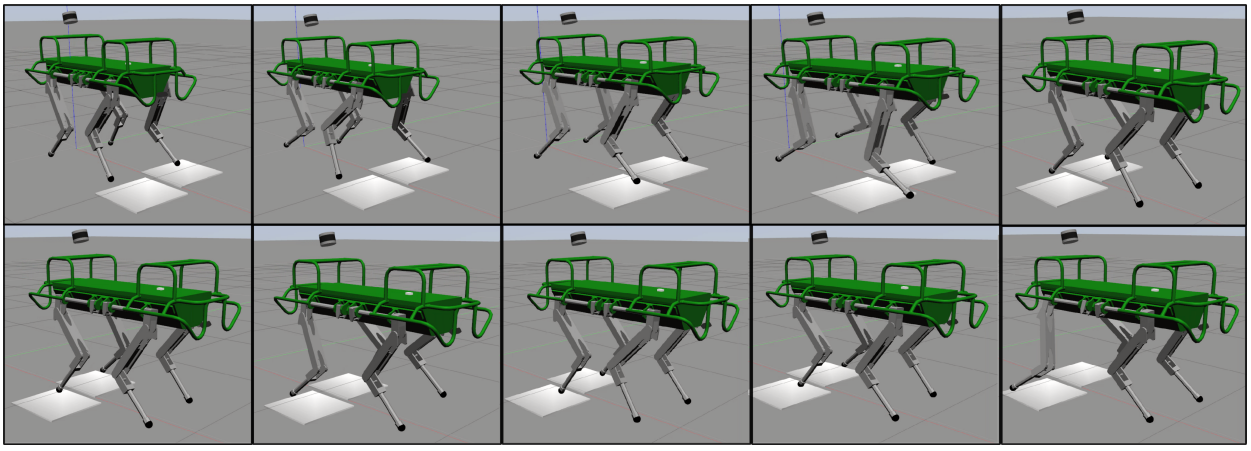


Figure 5.8: Multiple legs on ice slabs

The parameters used for this simulation are the same of the previous one:

ϵ_v	$prctile(\overline{\Delta V}, 99\%)$
ϵ_p	0.03
m	0.3

Table 5.2: Parameters of the simulation test: multiple legs on ice patch

Fig. 5.9 shows the shapes of $\overline{\Delta V}$, ΔP, the contact status and the slippage (red stripes). Fig. 5.10 shows the flags identifying the sliding.

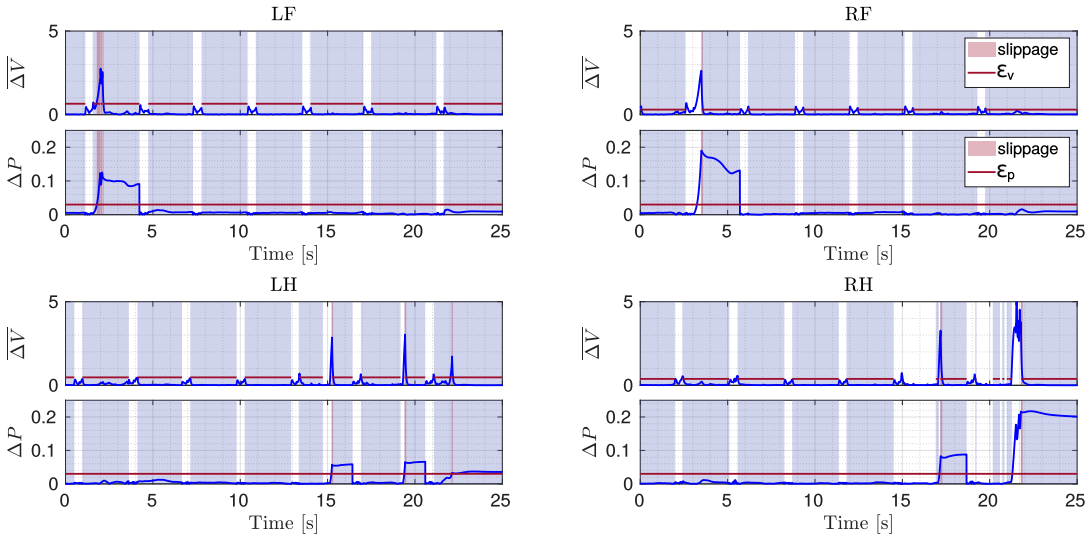


Figure 5.9: Shapes of $\overline{\Delta V}$ and ΔP with the respective thresholds

In Fig. 5.10 we show the flag for each leg. On the right side there is a zoom in the particular time interval the slippage occurs.

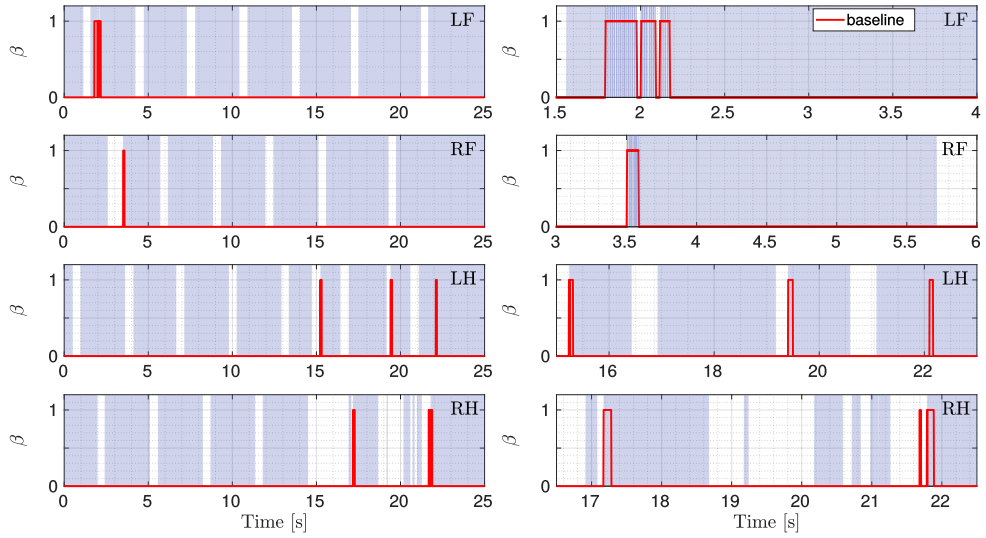


Figure 5.10: The slippage for each leg during the "multiple-legs "ice slab simulation. In the simulation all four legs walked over and slipped on the ice

In Figs. 5.11-5.14 we show the feet positions along x-axis and y-axis for LF, RF, LH, RH, with the markers indicating slippages.

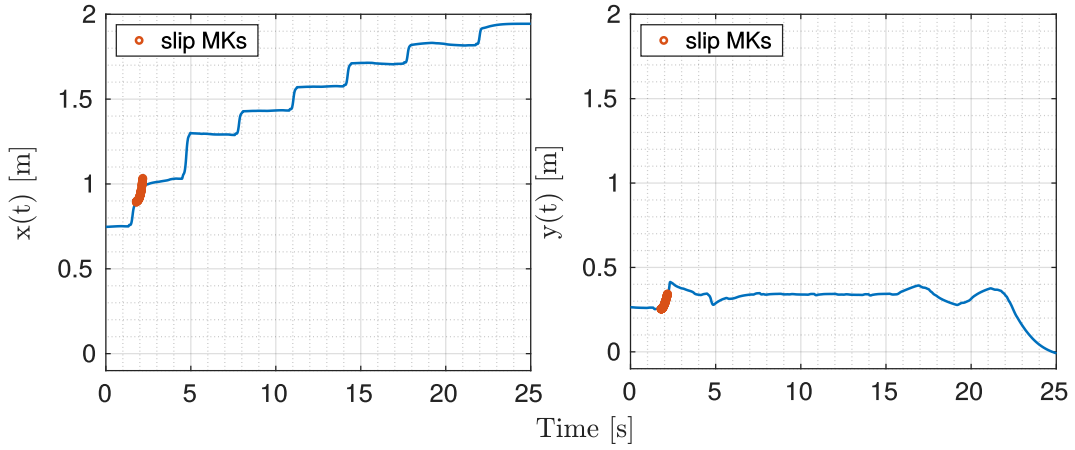


Figure 5.11: LF leg position along x-axis and y-axis during the simulation

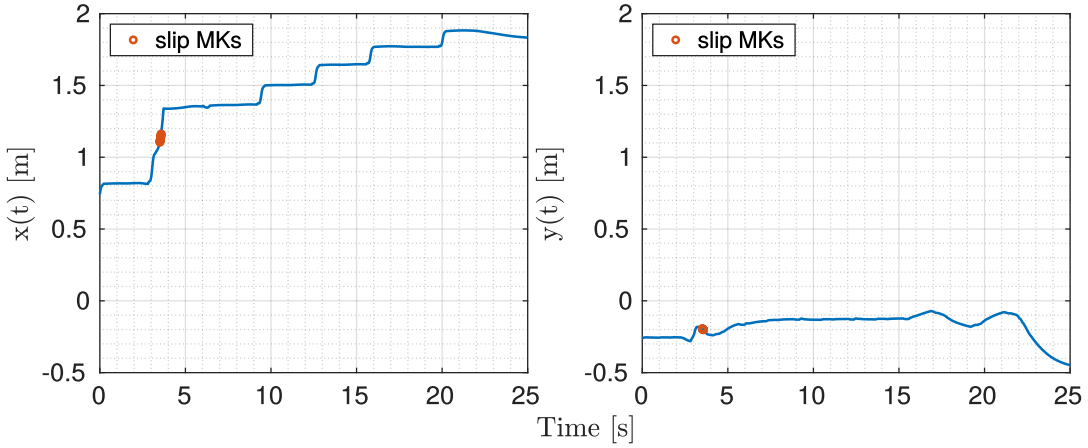


Figure 5.12: RF leg position along x-axis and y-axis during the simulation

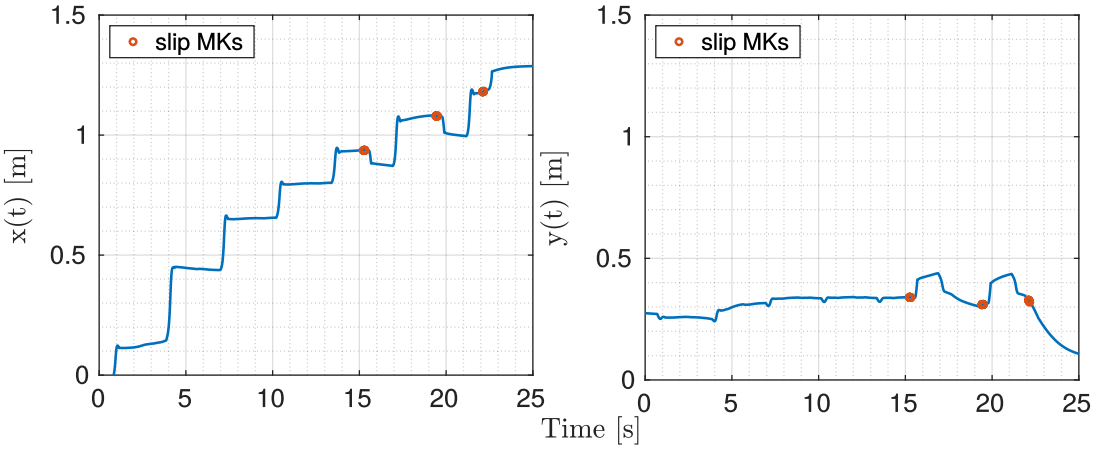


Figure 5.13: LH leg position along x-axis and y-axis during the simulation

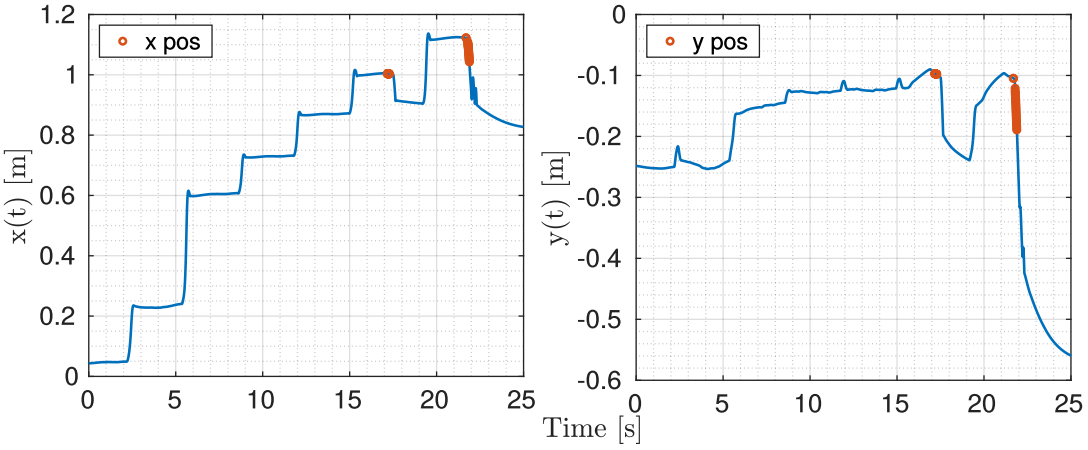


Figure 5.14: RH leg position along x-axis and y-axis during the simulation

5.1.4 Comparison with the baseline approach

In this test the sliding is never simultaneous, it occurs one leg at a time. So we implemented the strategy described in 4.1 for detect the slippage of one leg. We chose for the threshold the same value of the previous test, that is $\epsilon_v = 0.04$.

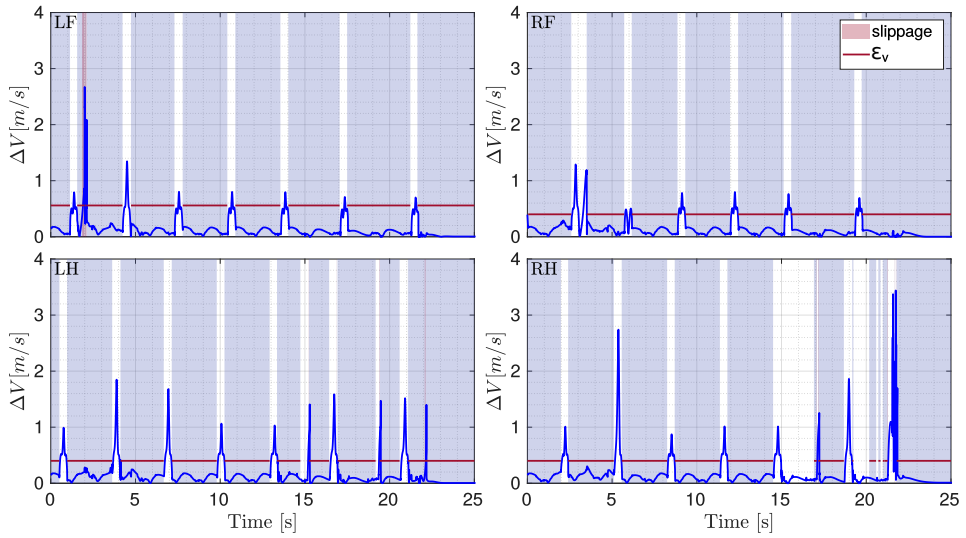


Figure 5.15: Shapes of ΔV with the threshold.

In Fig.5.16 we show the two flags obtained using the proposed approach (blue lines) and the baseline (red lines). Our approach is more reliable in detection: it is faster and the flag remains at one for the entire duration of the sliding (we manually watched each foot and then say when it is sliding or not). Furthermore, in the case of the RF leg we see the baseline approach is not detecting a slippage happening around the second 4 (as can be seen from the figure 5.18).

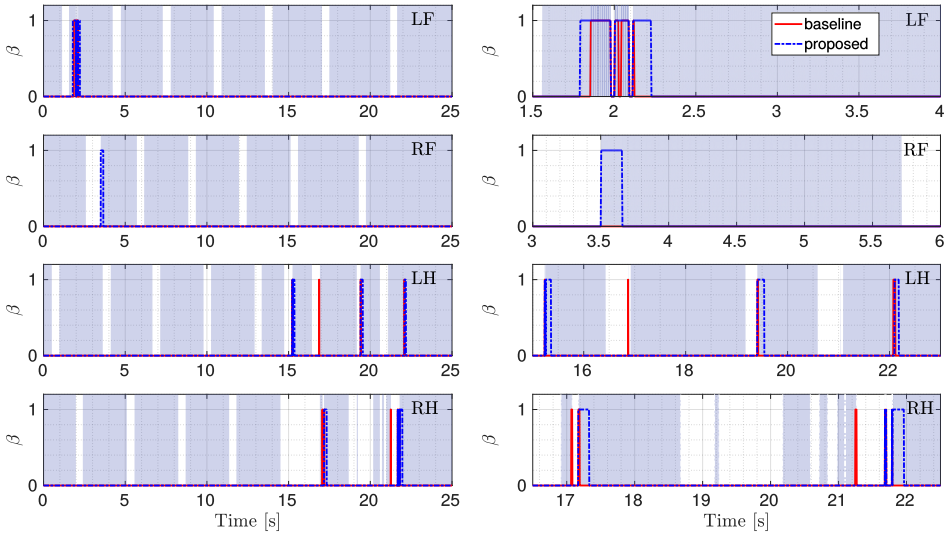


Figure 5.16: Comparison between the flags

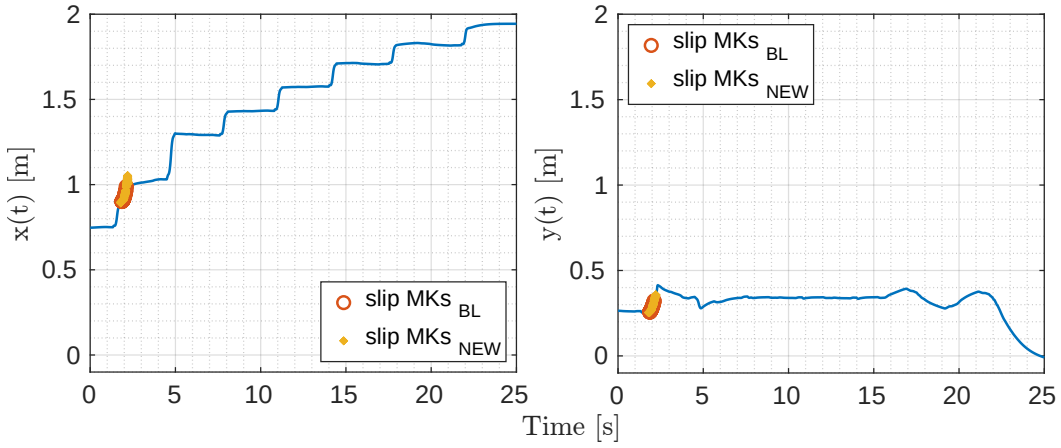


Figure 5.17: LF leg position along x-axis and y-axis during the simulation. Yellow markers correspond to the baseline flags = 1

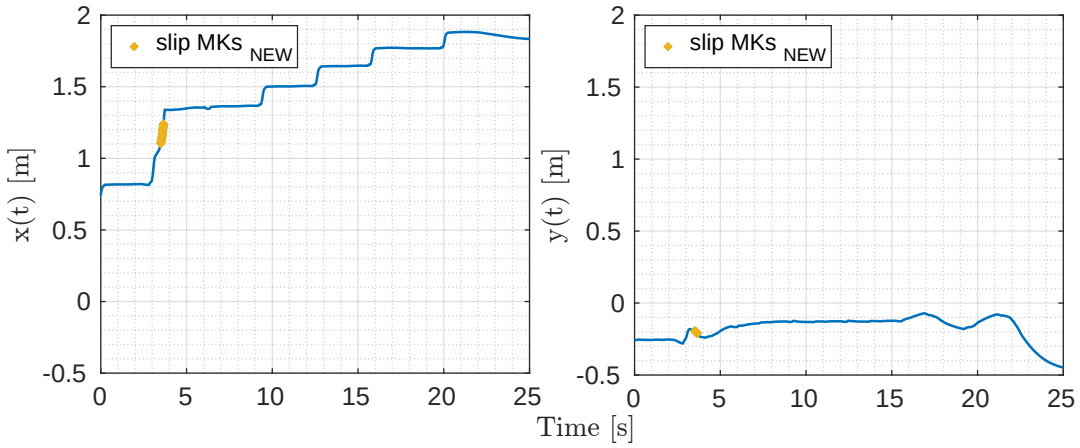


Figure 5.18: RF leg position along x-axis and y-axis during the simulation. Yellow markers correspond to the baseline flags = 1

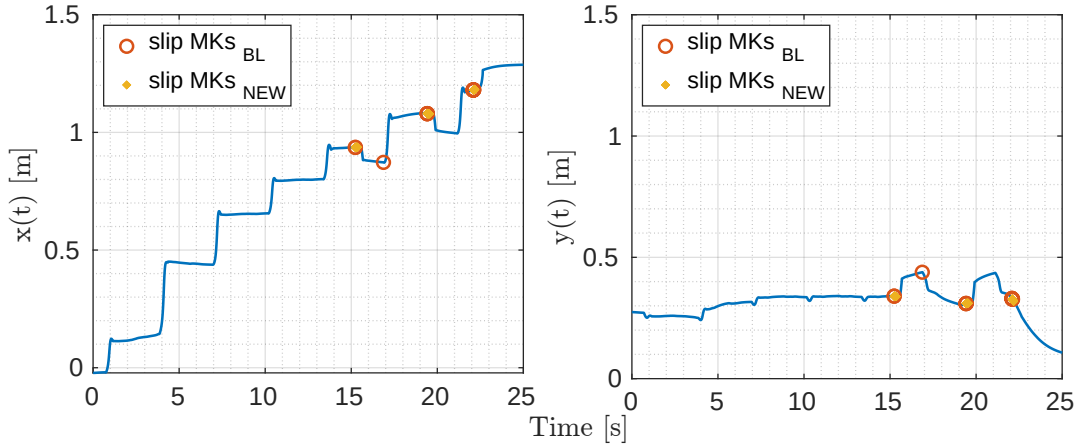


Figure 5.19: LH leg position along x-axis and y-axis during the simulation. Yellow markers correspond to the baseline flags = 1

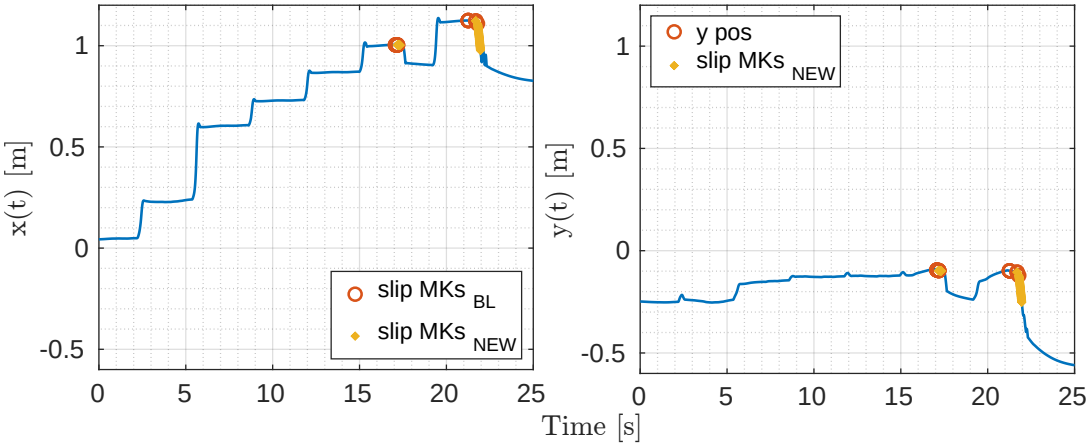


Figure 5.20: RH leg position along x-axis and y-axis during the simulation. Yellow markers correspond to the baseline flags = 1

5.2 Trot

To demonstrate that the proposed method can be used with different types of locomotion, we tested it in simulation with the robot trotting on slippery terrain. The quadrupedal trotting is a gait that allows the robot to reach higher speeds. It is more difficult to manage because the robot uses only two legs at the time (when LF and RH are in stance, RF and LH are in swing and vice versa) to achieve the desired velocity.
In the next sections we show the results obtained with the proposed slip detection algorithm and we compare them with those obtained using the baseline approach (Chap. 4.1).

5.2.1 One Leg on ice-patch

The robot starts walking from a non-slippery ground and then moves to ice-sheet. In this simple demonstration, the only leg going to the slab is LF. This is to test if, the slippage of one leg can affect the slip detection of the others.

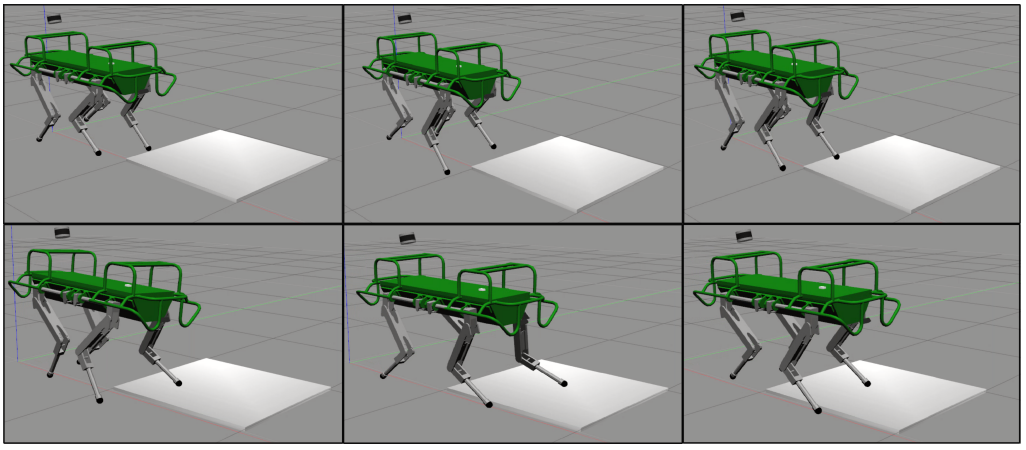


Figure 5.21: One leg on ice slabs

We used the following parameters:

ϵ_v	$prctile(\overline{\Delta V}, 95\%)$
ϵ_p	0.03
m	0.3

Table 5.3: Parameters of the simulation test: one leg on ice patch

We said that, if LF is in stance, RH is in stance as well, while the others are in swing (and certainly they cannot slip). From Fig. 5.22 we see that the slippage clearly occurs around the seconds 5 and 6. And from Fig. 5.22 and 5.23 we clearly see that the slip detection correctly performed on stance leg (LF) does not affect the slip detection of the

other stance leg RH (the value of its flag continues to be 0 thanks to the fact that ΔP remains below the detection threshold).

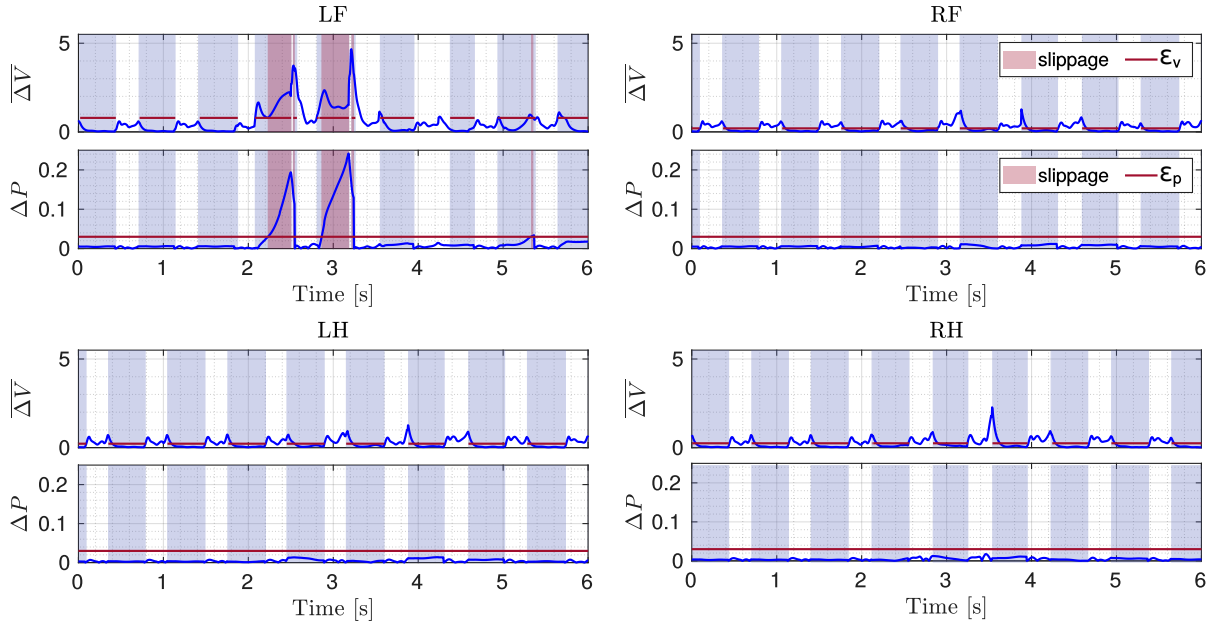


Figure 5.22: Shapes of $\overline{\Delta V}V$ and ΔP with the respective thresholds

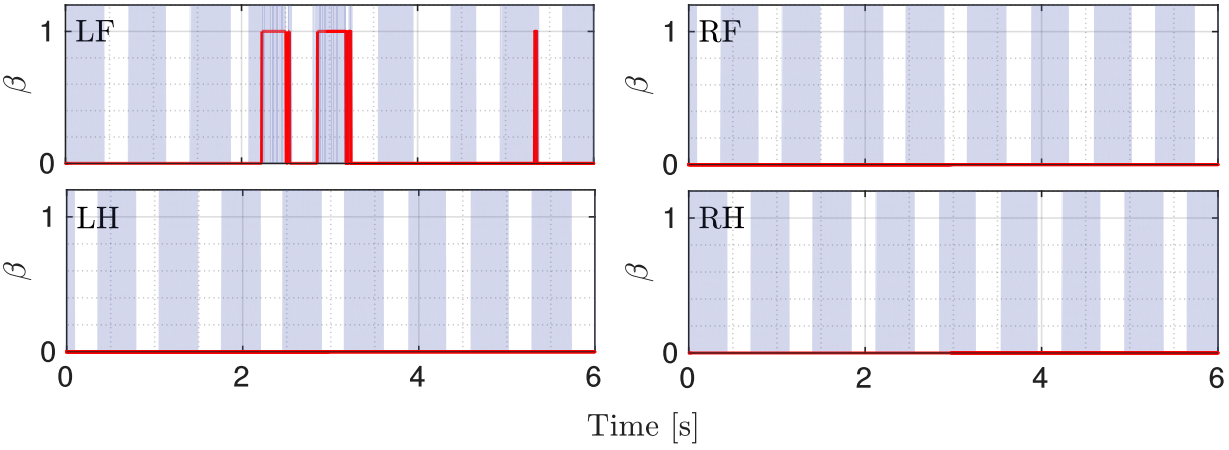


Figure 5.23: Flag indicating slippage

In Fig. 5.24 the evolution of the LF foot trajectory during the test.

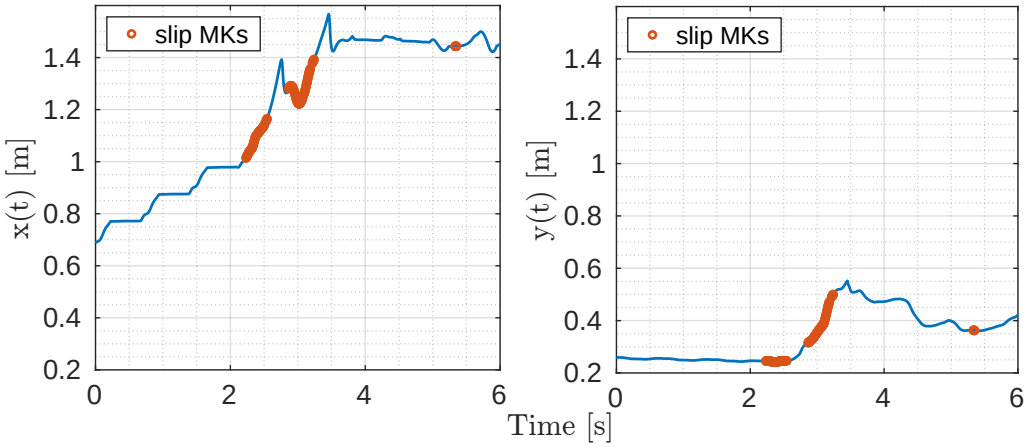


Figure 5.24: LF leg position along x-axis and y-axis during the simulation

Comparison with the baseline approach

In this case we implemented the method described in sec. 4.1.2. Although there is only one sliding leg in this test, the median-based approach could not be applied, because during the trot the legs have pairwise different velocities and there are only two stance legs on which to calculate the median of velocities.

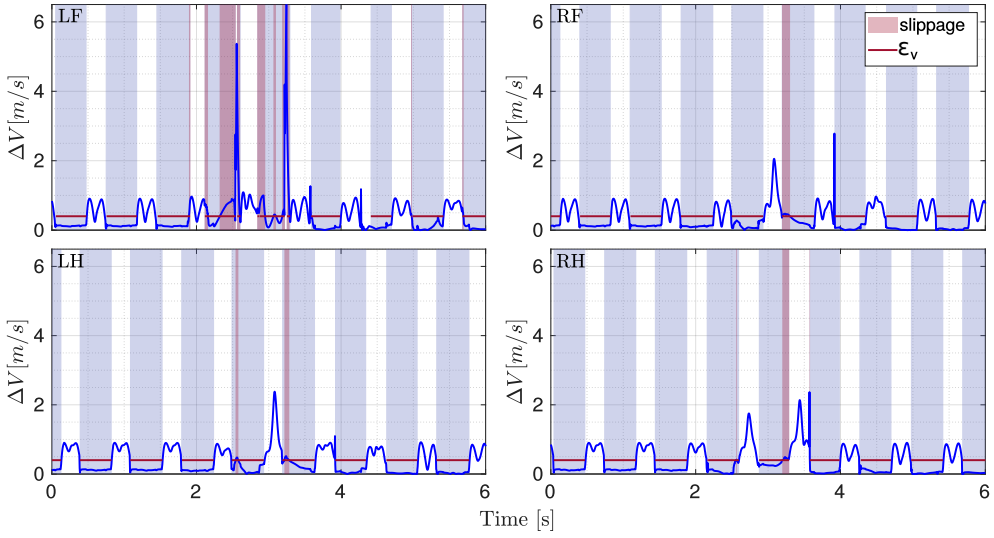


Figure 5.25: Shapes of ΔV with the threshold. From top-left to bottom-right: LF - RF - LH - RH

We note that there is a correct detection with both the algorithms in the interval 2-3 s, when the slippage actually occurs. But we also see that, with the baseline approach, in some instants the flag goes to 1 and then instantly returns to 0. We can consider them as false positives again probably due to the implementation of contact detection.

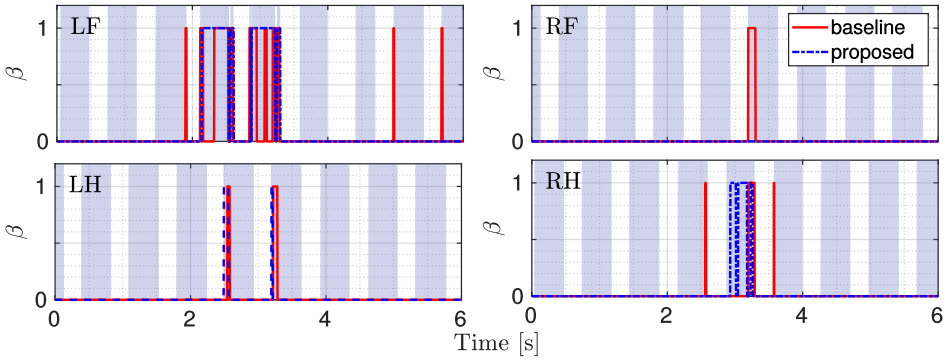


Figure 5.26: Comparison between the flags

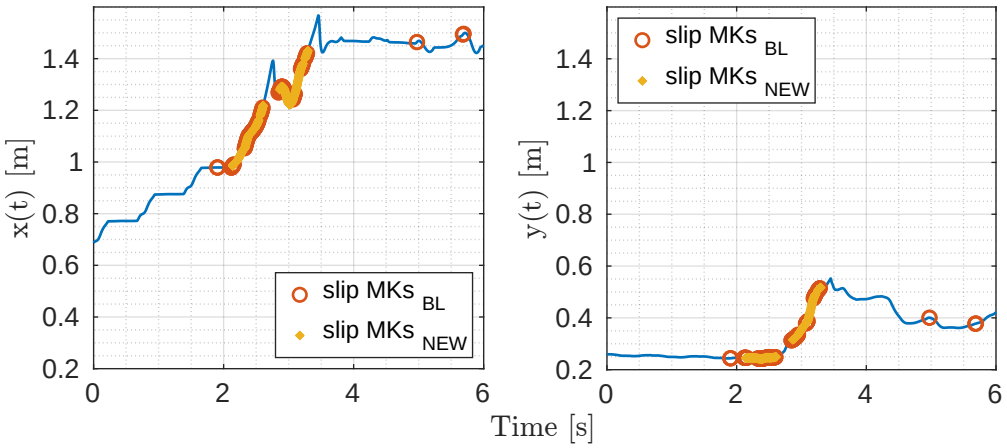


Figure 5.27: LF leg position along x-axis and y-axis during the simulation. Yellow markers correspond to the baseline flags = 1

5.2.2 Multiple Legs on ice-patches

As a last simulation result we show the case of slip detection in the case of several legs are on a slippery ground. In particular the robot moves to ice slabs and two legs are on ice-patches (LF and RF).

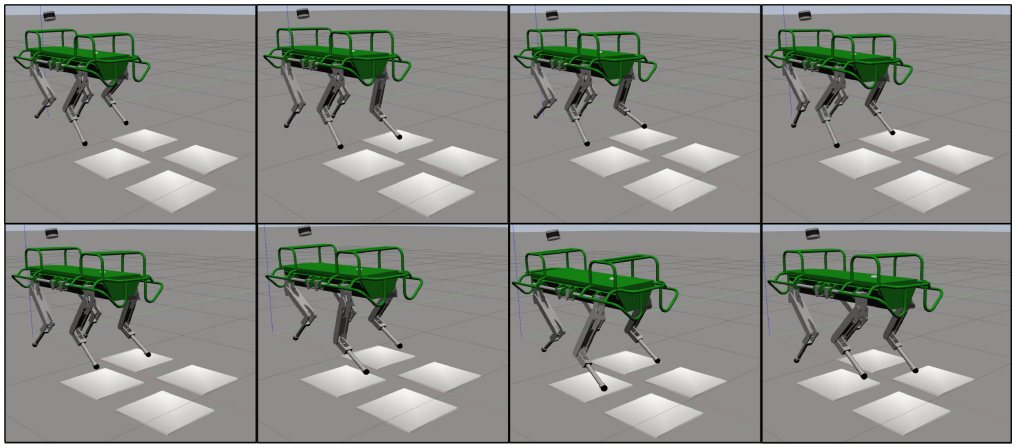


Figure 5.28: Multiple legs on ice slabs

We used the following parameters:

ϵ_v	$prctile(\overline{\Delta V}, 95\%)$
ϵ_p	0.03
m	0.3

Table 5.4: Parameters of the simulation test: multiple legs on ice patch

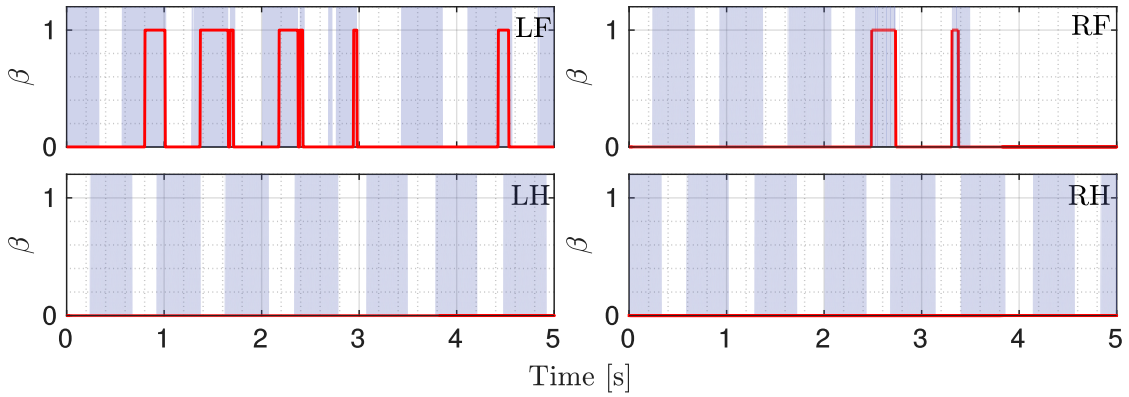


Figure 5.29: Flags indicating slippage

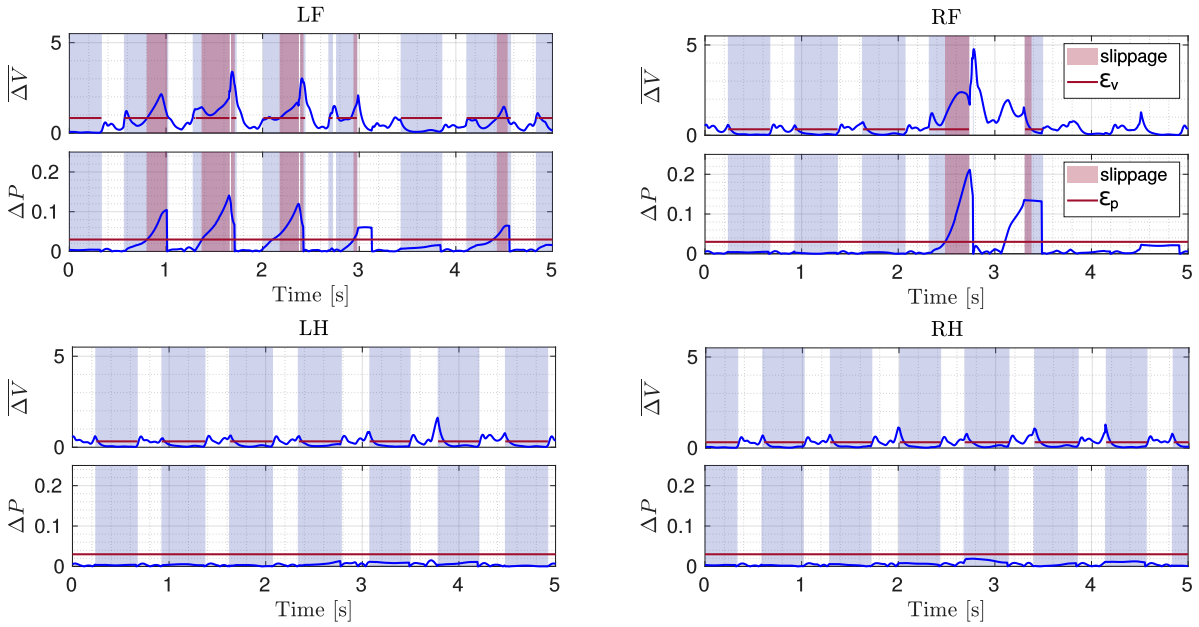


Figure 5.30: Shapes of $\overline{\Delta V}$ and ΔP with the respective thresholds

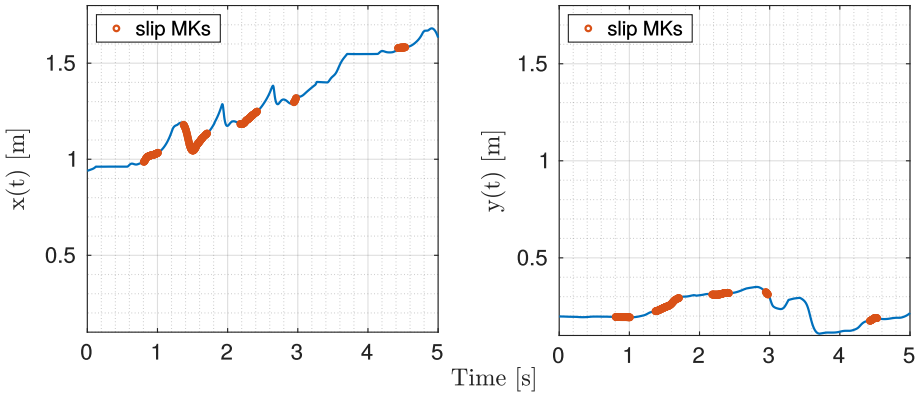


Figure 5.31: LF leg position along x-axis and y-axis during the simulation

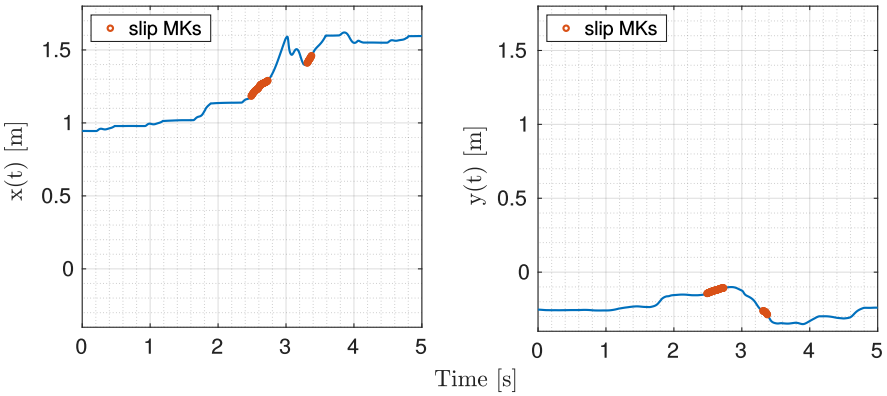


Figure 5.32: RF leg position along x-axis and y-axis during the simulation

Comparison with the baseline approach

Also in this case we implemented the baseline-algorithm presented in sec.4.1.2.

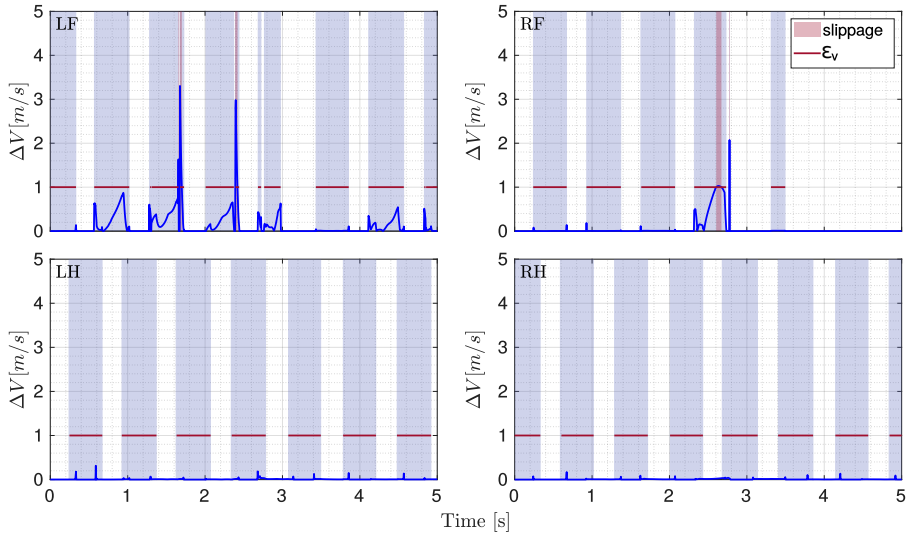


Figure 5.33: Shapes of ΔV with the threshold

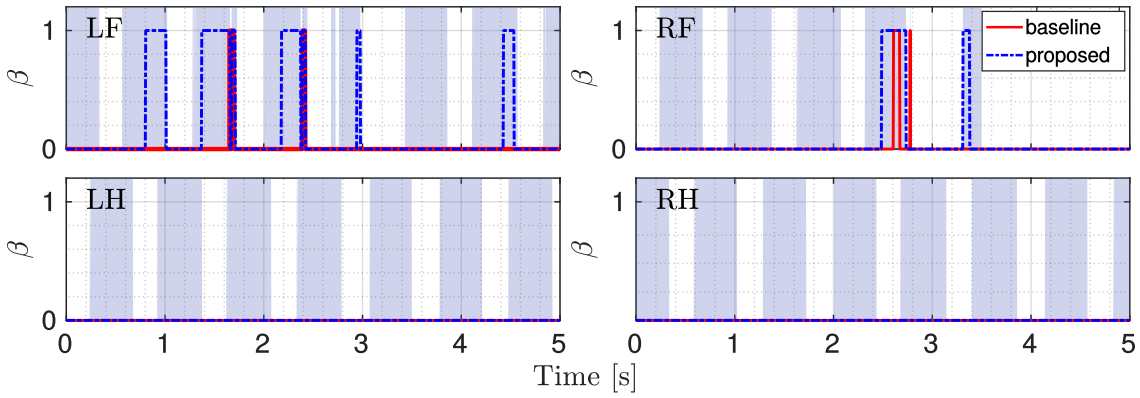


Figure 5.34: Flags indicating slippage

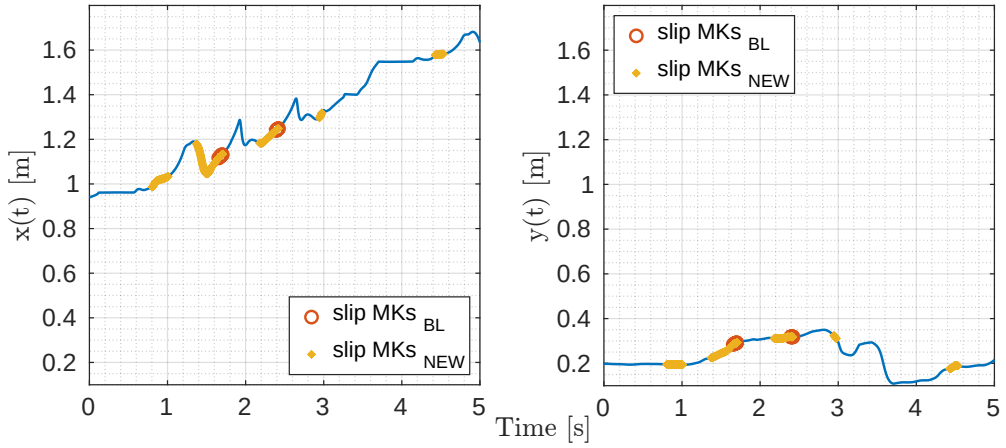


Figure 5.35: LF leg position along x-axis and y-axis during the simulation

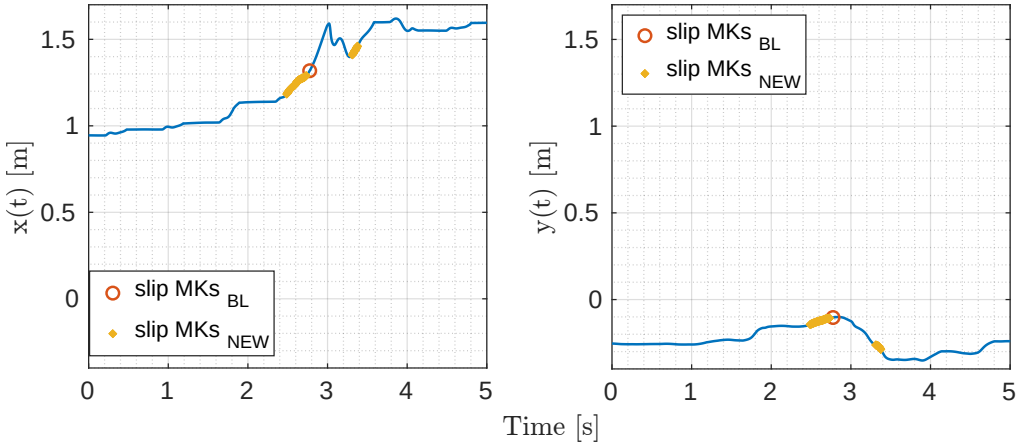


Figure 5.36: RF leg position along x-axis and y-axis during the simulation

From the simulations it is clear that both algorithms are able to detect single or multiple slipping, but the algorithm proposed in this project proves to be more robust, faster and, above all, detects slipping throughout its duration and not only in the first instants, which instead happens in the case of the baseline approach.

To further validate the method we did experiments on the HyQ robot (Chapter 6)

Chapter 6

Experimental Results

We validated the simulation presented in Chapter 5 on the real platform HyQ. In this chapter we show two experiments. In the first experiment the robot walks on a non-slippery terrain. This was done in order to have a ground truth and allows to adequately tune the parameters of our implementation. To prove the effectiveness of the slip detection, for the second experiment we used a slippery pallet for the robot to walk on. To make the pallet slippery, we sprinkled its surface with soap.

6.1 Crawl

For the experiments we used only crawl as gait, because it is more statically stable.

6.1.1 Walking on non-slippery ground

This is the first experiment done on the robot. Here the robot walks on a non-slippery terrain. This was necessary to tune the parameters used in the next experiment, in which the robot moves on a slippery pallet.

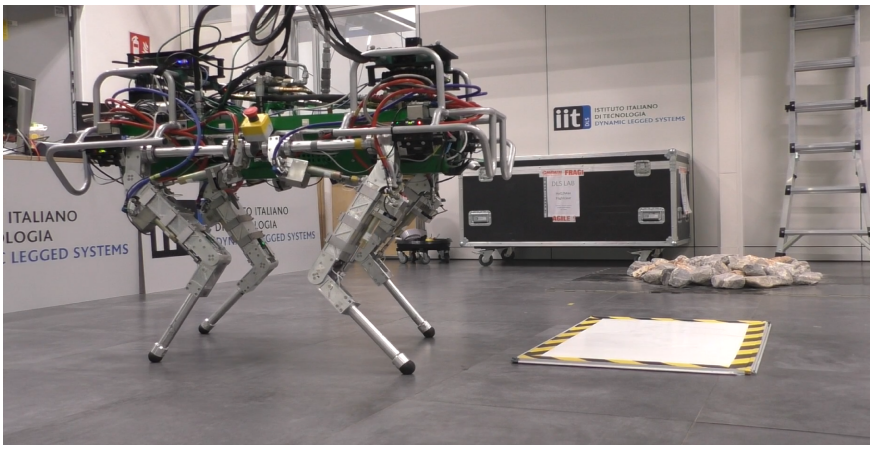


Figure 6.1

At the end we chose the following parameter:

ϵ_v	$prctile(\overline{\Delta V}, 95\%)$
ϵ_p	0.04
m	0.3

Table 6.1: Parameters of the experimental test: one leg on slippery patch

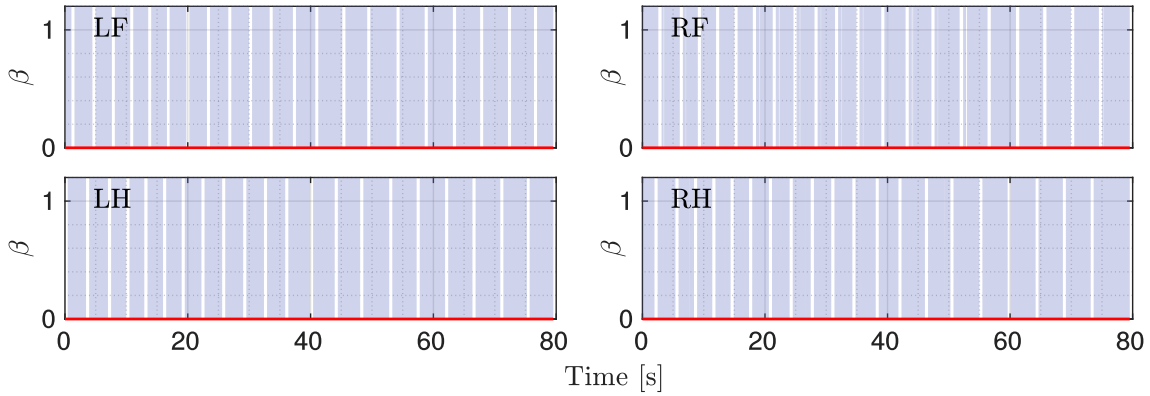


Figure 6.2: Flags indicating slippage

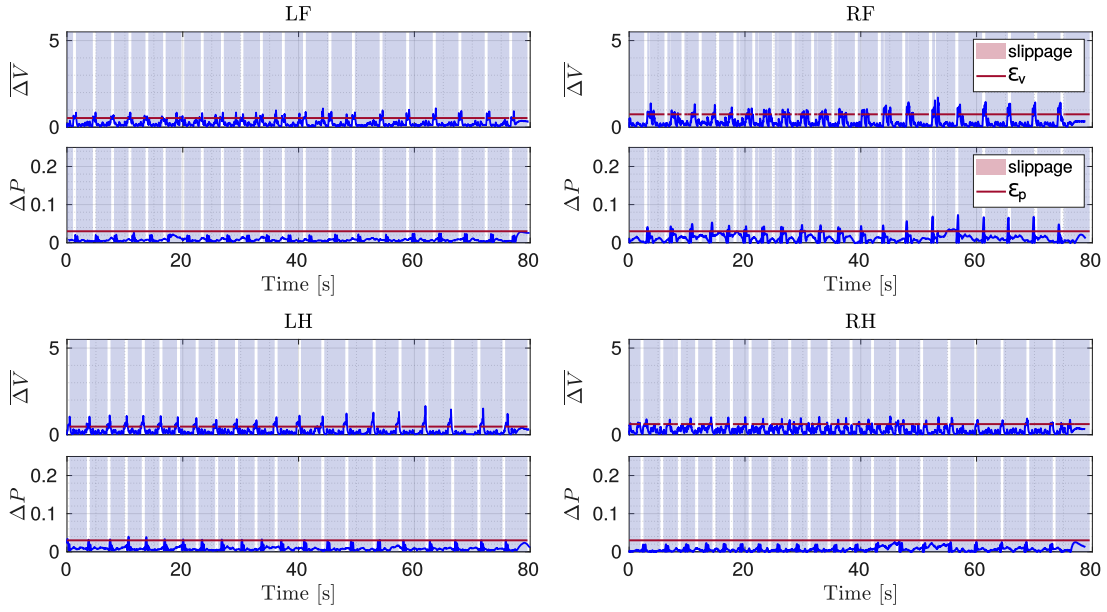


Figure 6.3: Shapes of $\overline{\Delta V}$ and ΔP with the respective thresholds

6.1.2 Walking on slippery ground

In this experiment the robot walked on a slippery pallet. Although only the legs LF and RF were placed in sliding conditions, during the experiment also for the leg LH, in contact with "normal"ground, a slipping was detected (Fig. 6.5).

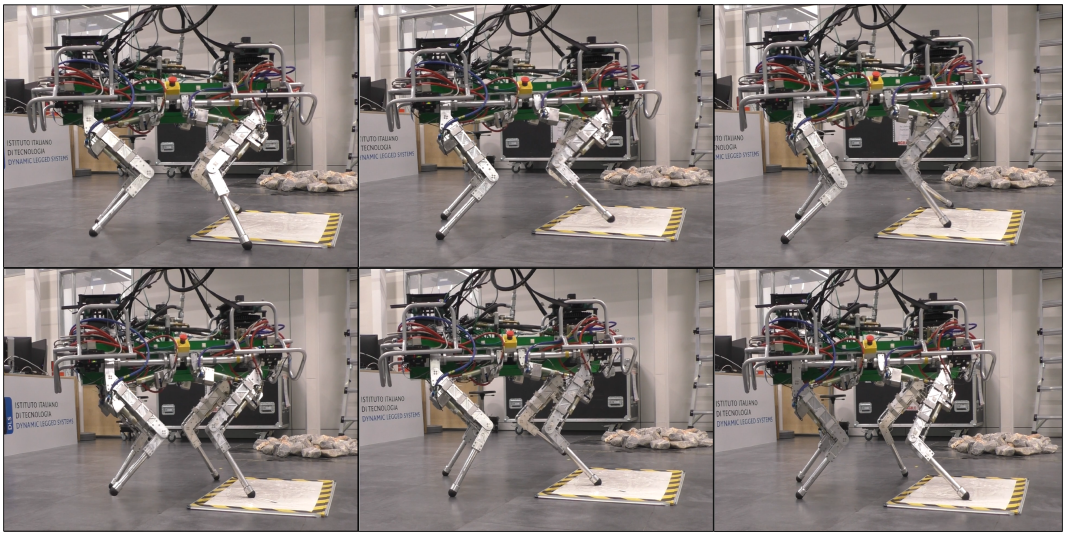


Figure 6.4: HyQ walking on a slippery pallet

The legs on slippery pallet are LF and RF. As it is possible to see from the plots all the detection are correctly performed.

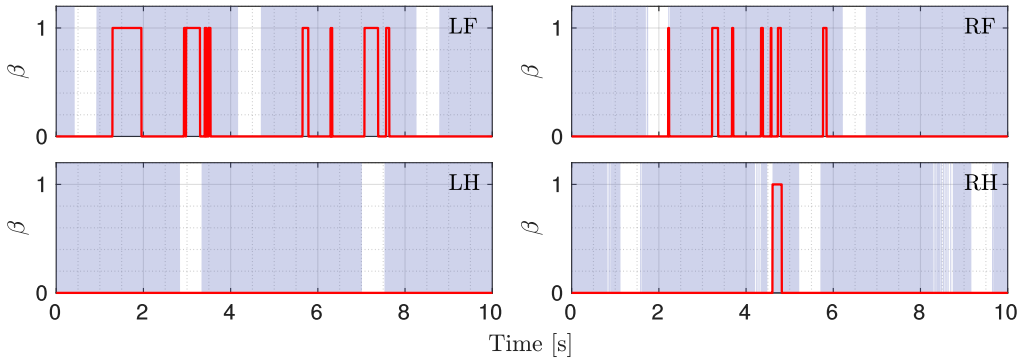


Figure 6.5: Flags indicating slippage

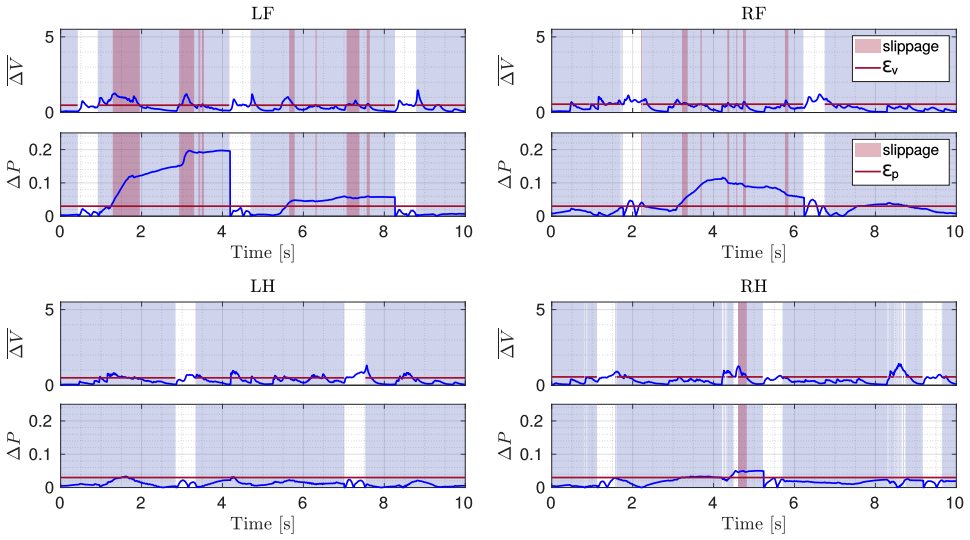


Figure 6.6: Shapes of ΔV and ΔP with the respective thresholds

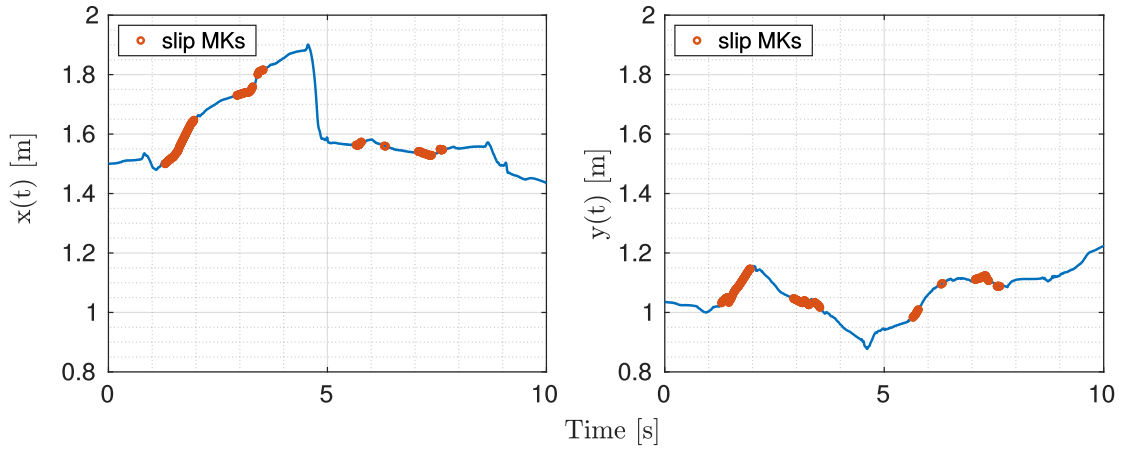


Figure 6.7: LF leg position along x-axis and y-axis during the experiment

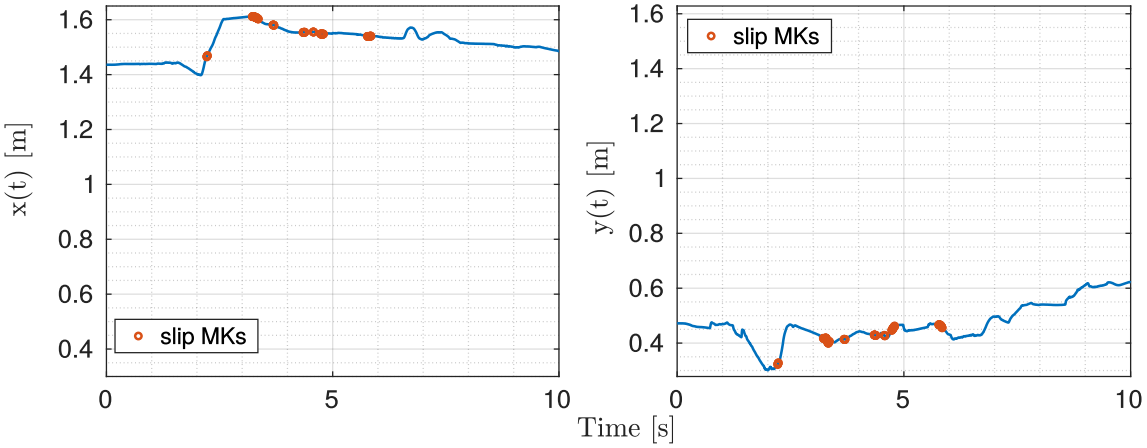


Figure 6.8: RF leg position along x-axis and y-axis during the experiment

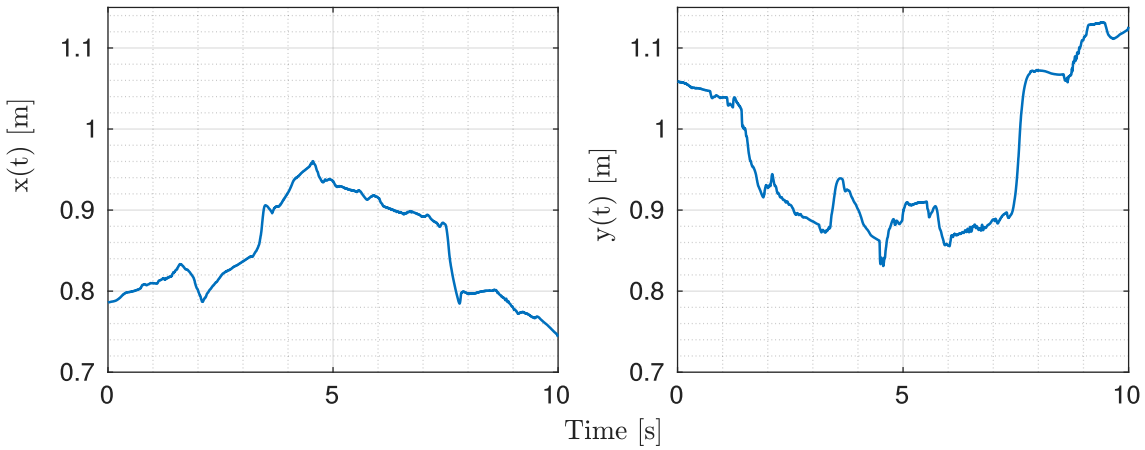


Figure 6.9: LH leg position along x-axis and y-axis during the experiment

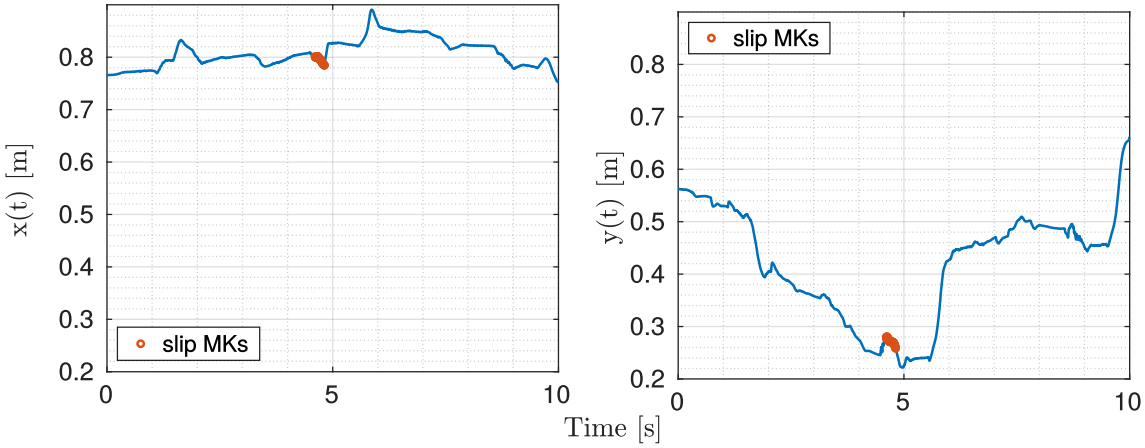


Figure 6.10: RH leg position along x-axis and y-axis during the experiment

Comparison with the baseline approach

During this experiment, legs LF and RF slipped at the same time, So, to make a comparison we implemented the method of "*Multiple leg slip detection*"We chose for the threshold the value $\epsilon_v = 1$.

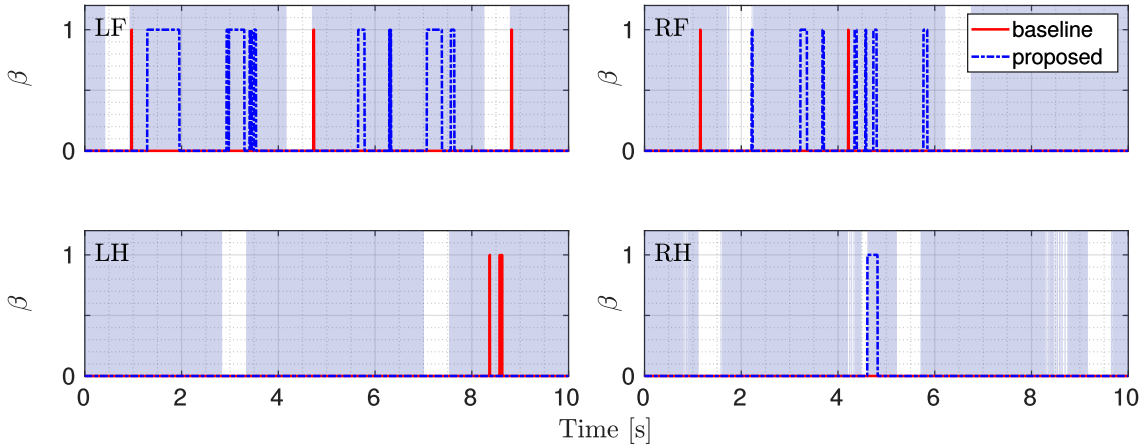


Figure 6.11: Flags indicating slippage

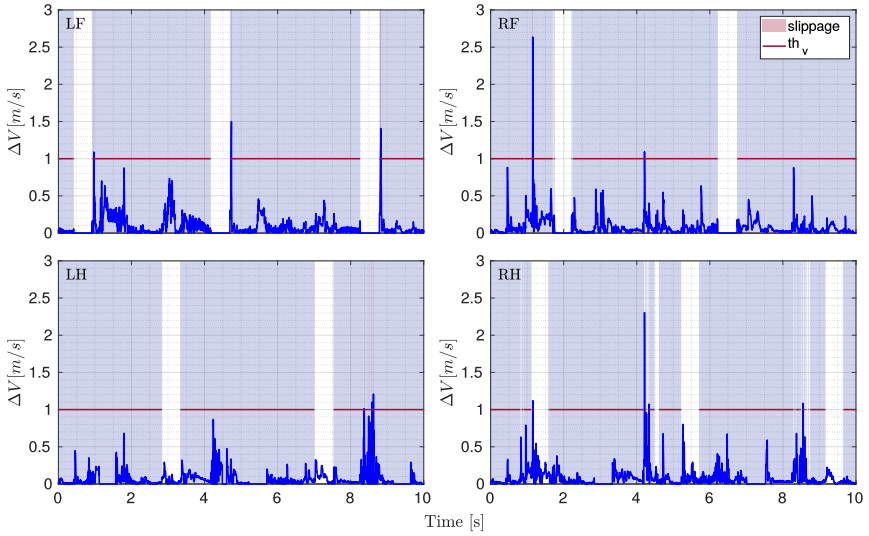


Figure 6.12: Shapes of ΔV with the respective thresholds

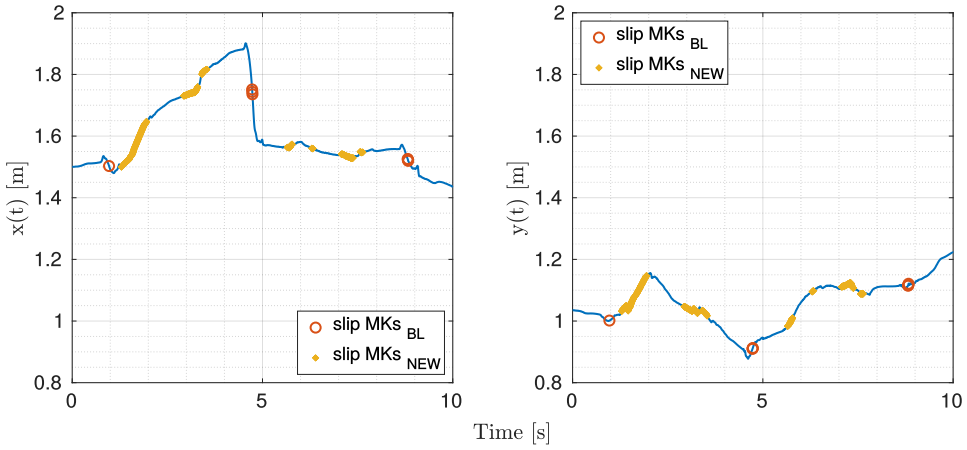


Figure 6.13: LF leg position along x-axis and y-axis during the simulation

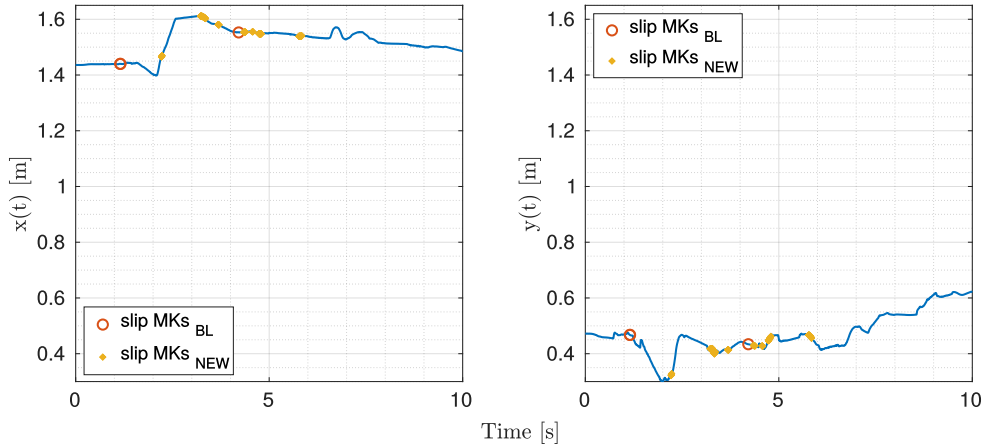


Figure 6.14: RF leg position along x-axis and y-axis during the simulation

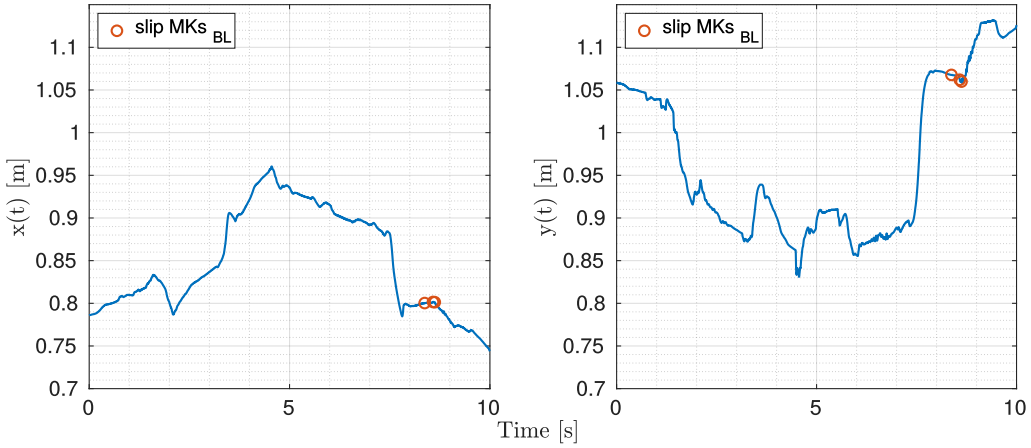


Figure 6.15: LH leg position along x-axis and y-axis during the simulation

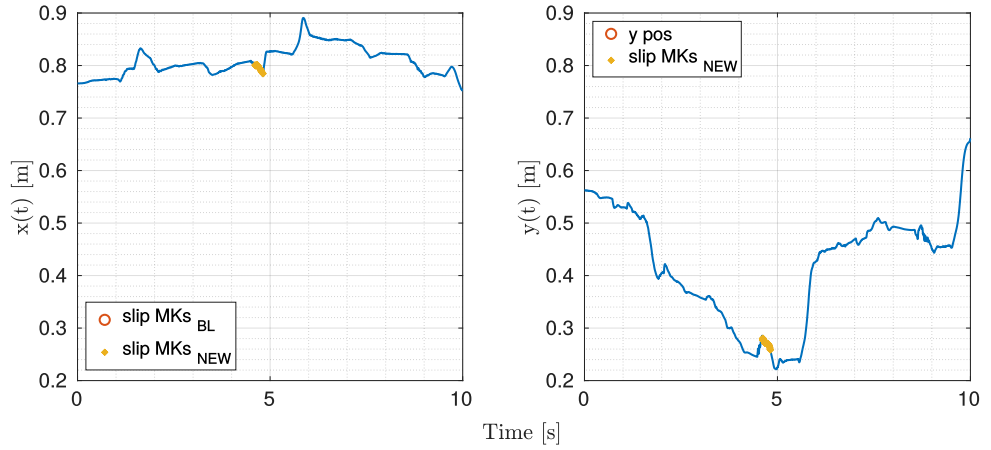


Figure 6.16: RH leg position along x-axis and y-axis during the simulation

Also in this case the results obtained with the proposed method are better, as they detect

the actual slips, there are no false positives and the flags are also able to show the duration of the slipping.

Chapter 7

Conclusion

In this thesis we presented a novel slip detection approach for legged robots based on kinematics, which makes use of velocity and position measurements at the ground contacts. In the field of legged robots, a kinematic-based approach is more suitable than a force-base approach, which involves the use of 6-axis force/torque sensors at the foot tips. The provided method shows that it is possible to detect a slippage quickly and effectively relying on the feet positions and velocities expressed in the base frame. This allows to avoid problems related to the drift, that usually happen when using the world frame. We proposed a method suitable for different types of locomotion and which is easily applicable to situations where the robot is required to change its velocity. Then we proved the effectiveness of the algorithm through the results obtained in simulation tests and in experiments. We also compared these results with those obtained using an already existing algorithm, showing that our implementation is more robust.

7.1 Future Works

An analysis of the maximum amount of slippage which is tolerable in the context of locomotion in order to preserve stability is part of future works. Furthermore we plan to implement an estimation of the friction properties of the terrain during the locomotion. This can be useful to set different level of "cautiousness", selecting more or less conservative gaits according to the situation. Then we can move on including the slip detection into the state estimation, on the estimation of the surface normal and an implementation of a recovery strategy based on the correction of the surface normal, that brings the GRFs back into the friction cone, when a slippage occurs. A recovery strategy is essential for locomotion on very slippery terrain as ice and in situation where the inclination of the terrain is wrongly estimated. In the future we can fuse the proposed approach with information coming from vision, that could provide a default value for the friction coefficient together with an estimate of its roughness. Moreover, the described algorithm should be tested in different type of scenarios, considering for instance soft terrain. Another important step will be the implementation of the proposed method at the level of others contact points: it should be useful to analyze the slippage of the shin when it enters in contact with an obstacle.

Appendices

Appendix A

Technical specifications of the robot

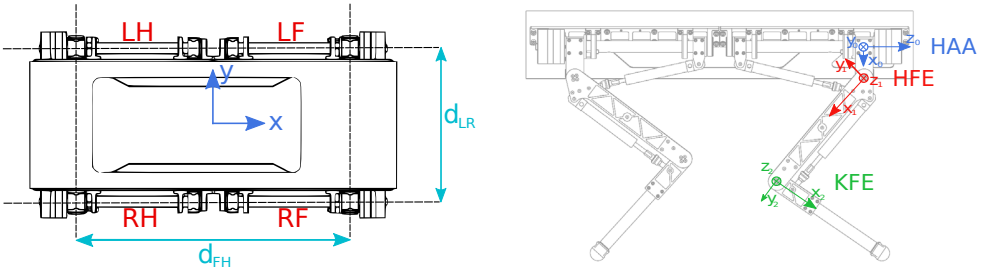
Table A.1 provides a summary of technical specifications of the robot and its actuators

Desctiption	**Value**
Dimensions	1.0 m × 0.5 m × 0.98 m (Length x Width x Height)
Leg length (HAA-ground)	from 0.339 m to 0.789 m
Distance left to right HAA axis	0.414 m
Distance left to right HFE axis	0.747 m
Weight	90 kg
Active DoF	12
Joint range of motion	120°
Hydraulic actuator type	double-acting cylinders
Maximum torque	145 Nm
Onboard sensors	joint position (relative and absolute) joint torque cylinder pressure foot spring compression IMU
Onboard computer	IntensePC with real-time Linux
Control frequency	1 kHz

Table A.1: Technical specifications of the HyQ robot

Appendix B

Forward Kinematics



(a) Top View (b) Side View

Figure B.1: Base frame and joint frames used to compute the forward kinematics

This is the procedure to obtain the *Transformation Matrix* of the leg RF. The same can be done for the other legs (LF, LH, RH) changing the distances between base and joint frames.

d_{12} is the distance between HAA and HFE; d_{23} is the distance between HFE and KFE; d_{34} is the distance between KFE and the foot.

$$T^{\mathcal{B}}_{HAA} = \begin{bmatrix} 0 & 0 & 1 & dfh/2 \\ 0 & 1 & 0 & -dlr/2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.0.1}$$

$$T^{HAA}_{HFE} = \begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 1 & d_{12}cos(\theta_1) \\ sin(\theta_1) & cos(\theta_1) & 0 & d_{12}sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.0.2}$$

$$T^{HFE}_{KFE} = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 1 & d_{23}cos(\theta_2) \\ sin(\theta_2) & cos(\theta_1) & 0 & d_{23}sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.0.3}$$

$$T_{foot}^{KFE} = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 1 & d_{34}cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & d_{34}sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.0.4}$$

Bibliography

[1] Alexandra Witze et al., *Nasa has launched the most ambitious mars rover ever built: Here's what happens next*, Nature **584** (2020), no. 7819, 15–16.

[2] Ned Potter, *A mars helicopter preps for launch: The first drone to fly on another planet will hitch a ride on nasa's perseverance rover - [news]*, IEEE Spectrum **57** (2020), no. 7, 06–07.

[3] Marc H Raibert, *Legged robots*, Communications of the ACM **29** (1986), no. 6, 499–514.

[4] Boston Dynamics, *Atlas, https://www.bostondynamics.com/atlas, accessed in september 2021.*

[5] Claudio Semini, Nikolaos Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin 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 **225** (2011), no. 6, 831–849.

[6] Claudio Semini, Victor Barasuol, Michele Focchi, Chundri Boelens, Mohamed Emara, Salvatore Casella, Octavio Villarreal, Romeo Orsolino, Geoff Fink, Shamel Fahmi, Gustavo Medrano-Cerda, Dhinesh Sangiah, Jack Lesniewski, Kyle Fulton, Michel Donadon, Mike Baker, and Darwin G Caldwell, *Brief introduction to the quadruped robot HyQReal*, Italian conference on robotics and intelligent machines (I-RIM), October 2019, pp. 1–2.

[7] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger, *Anymal - a highly mobile and dynamic quadrupedal robot*, 2016 ieee/rsj international conference on intelligent robots and systems (iros), 2016, pp. 38–44.

[8] Boston Dynamics, *Bigdog overview (updated march 2010), https://www.youtube.com/watch?v=cnzprsrwumq, accessed in september 2021.*

[9] Marco Hutter, Christian Gehring, Michael Bloesch, Mark A Hoepflinger, C David Remy, and Roland Siegwart, *Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion*, Adaptive mobile robotics, 2012, pp. 483–490.

[10] Boston Dynamics, *Atlas — partners in parkour, https://www.youtube.com/watch?v=tf4dml7fiwk, accessed in september 2021.*

[11] *https://www.iit.it/web/dynamic-legged-systems, accessed in september 2021.*

[12] Peter Fankhauser and Marco Hutter, *Anymal: a unique quadruped robot conquering harsh environments*, Research Features **126** (2018), 54–57.

[13] Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey Lang, and Sangbae Kim, *Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot*, 2013 ieee international conference on robotics and automation, 2013, pp. 3307–3312.

[14] Hae-Won Park, Patrick M Wensing, and Sangbae Kim, *High-speed bounding with the mit cheetah 2: Control design and experiments*, The International Journal of Robotics Research **36** (2017), no. 2, 167–192.

[15] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim, *Mit cheetah 3: Design and control of a robust, dynamic quadruped robot*, 2018 ieee/rsj international conference on intelligent robots and systems (iros), 2018, pp. 2245–2252.

[16] Erico Guizzo, *By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility*, IEEE Spectrum **56** (2019), no. 12, 34–39.

[17] Unitree Robotics, *Aliengo, https://www.unitree.com/products/aliengo/, accessed in september 2021.*

[18] Michael Bloesch, Christian Gehring, Peter Fankhauser, Marco Hutter, Mark A Hoepflinger, and Roland Siegwart, *State estimation for legged robots on unstable and slippery terrain*, 2013 ieee/rsj international conference on intelligent robots and systems, 2013, pp. 6058–6064.

[19] S. Fahmi, *On terrain-aware locomotion for legged robots*, Ph.D. Thesis, 2021.

[20] Dominik Belter and Piotr Skrzypczyński, *Rough terrain mapping and classification for foothold selection in a walking robot*, Journal of Field Robotics **28** (2011), no. 4, 497–528.

[21] Victor Barasuol, Marco Camurri, Stephane Bazeille, Darwin G. Caldwell, and Claudio Semini, *Reactive trotting with foot placement corrections through visual pattern classification*, 2015 ieee/rsj international conference on intelligent robots and systems (iros), 2015, pp. 5734–5741.

[22] Paul Filitchkin and Katie Byl, *Feature-based terrain classification for littledog*, 2012 ieee/rsj international conference on intelligent robots and systems, 2012, pp. 1387–1392.

[23] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter, *Where should i walk? predicting terrain properties from images via self-supervised learning*, IEEE Robotics and Automation Letters **4** (2019), no. 2, 1509–1516.

[24] Lucas Manuelli and Russ Tedrake, *Localizing external contact using proprioceptive sensors: The contact particle filter*, 2016 ieee/rsj international conference on intelligent robots and systems (iros), 2016, pp. 5062–5069.

[25] Victor Barasuol, Geoff Fink, Michele Focchi, D Caldwell, and Claudio Semini, *On the detection and localization of shin collisions and reactive actions in quadruped robots*, International conference on climbing and walking robots, 2019.

[26] Sean Wang, Ankit Bhatia, Matthew T Mason, and Aaron M Johnson, *Contact localization using velocity constraints*, 2020 ieee/rsj international conference on intelligent robots and systems (iros), 2020, pp. 7351–7358.

[27] Shamel Fahmi, Michele Focchi, Andreea Radulescu, Geoff Fink, Victor Barasuol, and Claudio Semini, *Stance: Locomotion adaptation over soft terrain*, IEEE Transactions on Robotics **36** (2020), no. 2, 443–457.

[28] Shamel Fahmi, Carlos Mastalli, Michele Focchi, and Claudio Semini, *Passive whole-body control for quadruped robots: Experimental validation over challenging terrain*, IEEE Robotics and Automation Letters **4** (2019), no. 3, 2553–2560.

[29] Shamel Fahmi, Geoff Fink, and Claudio Semini, *On state estimation for legged locomotion over soft terrain*, IEEE Sensors Letters **5** (2021), no. 1, 1–4.

[30] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt, *Design of a momentum-based control framework and application to the humanoid robot atlas*, International Journal of Humanoid Robotics **13** (March 2016), 1650007–1.

[31] C Dario Bellicoso, Fabian Jenelten, Christian Gehring, and Marco Hutter, *Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots*, IEEE Robotics and Automation Letters **3** (2018), no. 3, 2261–2268.

[32] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale, *Control of contact forces: The role of tactile feedback for contact localization*, 2012 ieee/rsj international conference on intelligent robots and systems, 2012, pp. 4048–4053.

[33] Serena Ivaldi, Matteo Fumagalli, Marco Randazzo, Francesco Nori, Giorgio Metta, and Giulio Sandini, *Computing robot internal/external wrenches by means of inertial, tactile and f/t sensors: theory and implementation on the icub*, 2011 11th ieee-ras international conference on humanoid robots, 2011, pp. 521–528.

[34] Yoshiyuki Ohmura, Yasuo Kuniyoshi, and Akihiko Nagakubo, *Conformable and scalable tactile sensor skin for curved surfaces*, Proceedings 2006 ieee international conference on robotics and automation, 2006. icra 2006., 2006, pp. 1348–1353.

[35] John Ulmen and Mark Cutkosky, *A robust, low-cost and low-noise artificial skin for human-friendly robots*, 2010 ieee international conference on robotics and automation, 2010, pp. 4836–4841.

[36] Alessandro De Luca and Raffaella Mattone, *Sensorless robot collision detection and hybrid force/-motion control*, Proceedings of the 2005 ieee international conference on robotics and automation, 2005, pp. 999–1004.

[37] Alessandro De Luca, Alin Albu-Schaffer, Sami Haddadin, and Gerd Hirzinger, *Collision detection and safe reaction with the dlr-iii lightweight manipulator arm*, 2006 ieee/rsj international conference on intelligent robots and systems, 2006, pp. 1623–1630.

[38] Jonathan Vorndamme, Moritz Schappler, and Sami Haddadin, *Collision detection, isolation and identification for humanoids*, 2017 ieee international conference on robotics and automation (icra), 2017, pp. 4754–4761.

[39] Michele Focchi, Romeo Orsolino, Marco Camurri, Victor Barasuol, Carlos Mastalli, Darwin G Caldwell, and Claudio Semini, *Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality*, Advances in robotics research: From lab to market, 2020, pp. 165–209.

[40] Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart, *State estimation for legged robotsconsistent fusion of leg kinematics and imu*, Robotics **17** (2013), 17–24.

[41] Annett Chilian, Heiko Hirschmüller, and Martin Görner, *Multisensor data fusion for robust pose estimation of a six-legged walking robot*, 2011 ieee/rsj international conference on intelligent robots and systems, 2011, pp. 2497–2504.

[42] Hiroshi Takemura, Masato Deguchi, Jun Ueda, Yoshio Matsumoto, and Tsukasa Ogasawara, *Slip-adaptive walk of quadruped robot*, Robotics and Autonomous Systems **53** (2005), no. 2, 124–141.

[43] Michele Focchi, Victor Barasuol, Marco Frigerio, Darwin G Caldwell, and Claudio Semini, *Slip detection and recovery for quadruped robots*, Robotics research, 2018, pp. 185–199.

[44] Fabian Jenelten, Jemin Hwangbo, Fabian Tresoldi, C Dario Bellicoso, and Marco Hutter, *Dynamic locomotion on slippery ground*, IEEE Robotics and Automation Letters **4** (2019), no. 4, 4170–4176.

[45] Jaejun Park, Do Hun Kong, and Hae-Won Park, *Design of anti-skid foot with passive slip detection mechanism for conditional utilization of heterogeneous foot pads*, IEEE Robotics and Automation Letters **4** (2019), no. 2, 1170–1177.

[46] *http://wiki.ros.org/rviz, accessed in september 2021.*

[47] *http://gazebosim.org/, accessed in september 2021.*

[48] *https://www.ros.org/, accessed in september 2021.*

[49] *http://wiki.ros.org/plotjuggler, accessed in september 2021.*

[50] *https://it.mathworks.com/products/matlab.html, accessed in september 2021.*

[51] Dan Simon, *Optimal state estimation: Kalman, h infinity, and nonlinear approaches*, John Wiley & Sons, 2006.

[52] Geoff Fink and Claudio Semini, *The DLS quadruped proprioceptive sensor dataset*, International conference on climbing and walking robots (CLAWAR), August 2020, accepted, pp. 1–8.

[53] Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson Roberto De Pieri, and Darwin G. Caldwell, *A reactive controller framework for quadrupedal locomotion on challenging terrain*, May 2013, pp. 2554–2561.

[54] Ioannis Havoutis, Jesus Ortiz, Stephane Bazeille, Victor Barasuol, Claudio Semini, and Darwin G Caldwell, *Onboard perception-based trotting and crawling with the hydraulic quadruped robot (hyq)*, 2013 ieee/rsj international conference on intelligent robots and systems, 2013, pp. 6052–6057.

[55] D. Stewart and J.C. Trinkle, *An implicit time-stepping scheme for rigid body dynamics with coulomb friction*, Proceedings 2000 icra. millennium conference. ieee international conference on robotics and automation. symposia proceedings (cat. no.00ch37065), 2000, pp. 162–169 vol.1.