ISTITUTO ITALIANO DI TECNOLOGIA, AND
UNIVERSITY OF GENOA

Skippy, the Balancing and Hopping Robot

by

Federico Allione

Thesis submitted for the degree of *Doctor of Philosophy* (XXXVI cycle)

February 2024

Prof. Roy Featherstone	Supervisor
Prof. Darwin Caldwell	Supervisor
Prof. Paolo Massobrio	Head of the PhD program

Thesis Jury:

Prof. Federico Colombo, *Polytechnic University of Turin*	External examiner
Prof. Marco Camurri, *Free University of Bozen-Bolzano*	External examiner

Department of Informatics, Bioengineering, Robotics and Systems Engineering
PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Declaration

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

Federico Allione
January 2024

Abstract

Legged robots have gained popularity recently, with quadrupeds being sufficiently reliable to leave the research labs and be deployed in real-world environments. Humanoids, on the other hand, still need to be developed more, and extensive research in the field of legged robots is still required. This thesis presents Skippy, a highly athletic, underactuated balancing and hopping monopedal robot. Skippy is a simple robot with only one leg and two actuators, and although its design is simple, complex dynamics govern it. Due to its intrinsic instability, it is difficult to control, making it a suitable testing machine for novel balancing and hopping control algorithms.

The beginning of this work focuses on the experimental validation of robotic hardware and software. It proposes a methodology to verify the correct behaviour of Skippy's sensorimotor system in all the possible scenarios it can encounter while operating (e.g. continuous hopping, crash landing after a wrong manoeuvre) through purpose-built testing apparatus. Furthermore, it presents the intermediate 2D balancing machines built before Skippy to validate the correct behaviour of the aforementioned sensorimotor system while balancing using Featherstone's balance controller. Then, the research described in this thesis extends Featherstone's balance controller to the case of a robot balancing on a rolling contact rather than on a single-contact point foot, and it validates it with an experiment on a purpose-built balancing machine. At last, this work shows Skippy in its 2D-constrained version, proving the robot's capability as a balancing and hopping machine.

Table of contents

List of figures

List of tables

Acronyms

AHRS Attitude and Heading Reference Systems.

BiSS-C Bidirectional Interface for Serial/Synchronous.

CAD Computer Aided Design.

CoM Centre of Mass.

CRC Cyclic Redundancy Check.

DoF Degrees of Freedom.

eQEP Enhanced Quadrature Encoder Pulse.

FFT Fast Fourier Transform.

FPGA Field Programmable Gate Array.

I2C Inter-Integrated Circuit.

IIT Istituto italiano di Tecnologia (Italian Institute of Technology).

IMU Inertial Measurement Unit.

LIDAR Light Detection And Ranging.

LQR Linear Quadratic Regularization.

MEMS Micro Electro-Mechanical System.

MPC Model Predictive Control.

MSU Michigan State University.

PD Proportional Derivative.

PID Proportional Integral Derivative.

PTPE Peak to Peak Error.

PWM Pulse-Width Modulation.

RMSE Root Mean Square Error.

RWP Reaction Wheel Pendulum.

SBR Self Balancing Robot.

SCADA Supervisory Control and Data Acquisition.

SLIP Spring Loaded Inverted Pendulum.

SPI Serial Peripheral Interface.

UniGe Università di Genova.

VI Virtual Instrument.

List of Publications

International Journals

1. F. Allione, R. Featherstone, P. Wensing, and D. G. Caldwell, "Balancing on a Rolling Contact", *IEEE Robotics and Automation Letters*, Early Access, p. 1-8, 2023, DOI: 10.1109/LRA.2023.3326696.

2. F. Allione, J. D. Gamba, A. E. Gkikakis, R. Featherstone, and D. G. Caldwell, "Effects of repetitive low-acceleration impacts on attitude estimation with micro-electromechanical inertial measurement units", *Frontiers in Robotics and AI*, vol. 10, 2023.

International Conferences with Peer Review

1. F. Allione, A. E. Gkikakis, and R. Featherstone, "Experimental demonstration of a general balancing controller on an untethered planar inverted double pendulum", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8292–8297, 2022.

2. F. Allione, B. R. P. Singh, A. E. Gkikakis, and R. Featherstone, "Mechanical shock testing of incremental and absolute position encoders", in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 52–57, 2021.

Chapter 1

Introduction

As legged robots become a part of our daily lives, we expect them to get better at moving around. They should be able to walk, run, and even jump like humans and animals do [10; 11; 12]. Meeting these expectations means these robots need improved designs and control systems before leaving the research labs.

Quadrupeds have recently become sufficiently reliable and robust to leave a laboratory's protective and controlled environment to enter the real world. Their hardware and software allow them to withstand the unexpected and unmodelled challenges the outer world can produce. Some of them, such as ANYmal [13] and Spot [14], have already been deployed in real working environments, mainly for industrial plant inspection.

Humanoids, on the other hand, are not completely ready to leave research labs, being still too slow and clumsy for daily activities. Although Atlas [15] and Digit [16] showed dynamic and athletic behaviours, humanoids cannot still be relied on for being introduced to the real world, although Digit is currently being tested in warehouses. The articulated mechanical structure, combined with highly complex tasks (such as walking), makes controlling this type of robot very demanding in terms of computational power and energy consumption.

This thesis focuses on realising and controlling a highly athletic, underactuated monopedal robot, with only one leg and two actuators. However, even this simple robot is governed by complex dynamics and is difficult to control due to its intrinsic instability. This work presents an in-depth experimental analysis of the challenges faced in the robot's development process, and the results can be used to inform the design of more complicated legged robots, such as bipeds and quadrupeds.

Figure 1.1 Skippy, the balancing and hopping robot

1.1 Motivation

This thesis is part of a larger project called Skippy. The project's objective is to design and build Skippy, a highly athletic monopedal robot, able to balance and hop in 3D using only two motors, see Figure 1.1. Skippy's design is technology inspired, meaning it is actively pursuing the best possible performance with the current technology in terms of actuation strategy, materials and design. The design of Skippy has been the objective of intensive study by one member of the Skippy team, Gkikakis [1]. In his work, the mechanical design and the desired robot's behaviour are co-optimized to achieve high performance, under the assumption that *'it is easier to increase the complexity of a high-performance robot than to increase the performance of a highly complex robot'* [17]. The objective is the realisation of the simplest possible robot physically able to perform the desired tasks, such as balancing and hopping, with a simple and light weight controller.

From the control point of view, Skippy's balance controller relies on Featherstone's controller [18], which has been further developed and expanded by two other members of the team, Singh [19] and Gamba [20]. This thesis pushes the Skippy project a step further

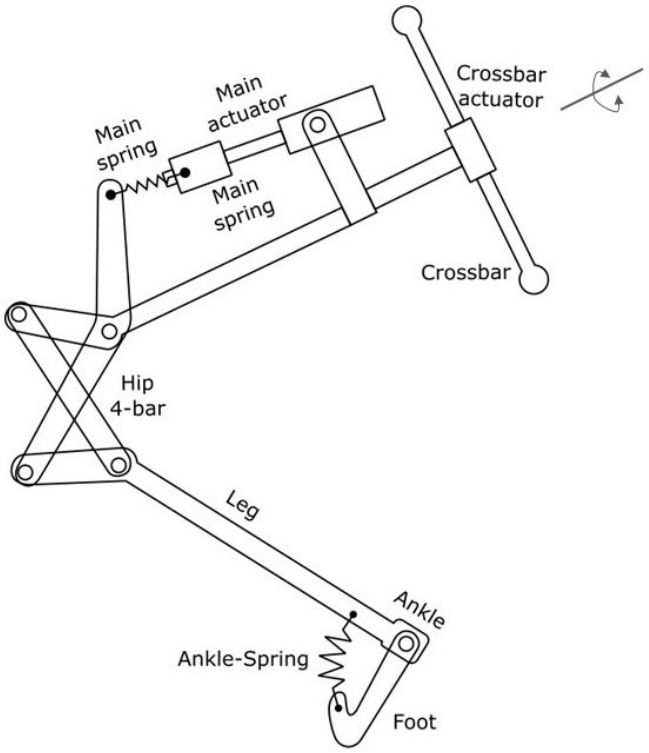


Figure 1.2 Conceptual diagram of Skippy [1].

by building Skippy, validating the previous simulation studies, and extending even more Featherstone's balance controller.

Skippy's hardware has been designed and built in stages. Its modular structure allows for a progressive development of both hardware and software. At each development step, extra complexity is added, making the robot more difficult to control but at the same time more performing. The structure of this thesis follows this trend by presenting the developed machines from the simplest to the most complex.

This thesis presents the experimental results obtained through the intermediate steps in developing Skippy, together with the controller that allowed for such results.

1.2 Objectives and Contributions

The main goal of the Skippy project is to design, build and control a highly athletic monopedal balancing and hopping machine called Skippy. Skippy is technology inspired, designed to be fully autonomous, untethered and battery-operated. All computation takes place on board, and the operator can start, stop and log all the experiments remotely. Skippy, in its final configuration, will be able to balance and hop in 3D using only two actuators, as shown in

Figure 1.2. The mechanism is planar except for the crossbar, which rotates about the axis shown. The main actuator has two functions: balancing and hopping in the sagittal plane, while the crossbar actuator keeps that plane vertical. Skippy is designed to be robust enough to withstand falls and crash landings due to mistakes and accidents, making it a reliable platform for testing learning algorithms.

My work mainly contributed to building Skippy and controlling its various 2D configurations. More in detail, my personal contribution is presented below, divided into preliminary work not reported in this thesis and the research activities described in this work.

The preliminary work consisted in

- CAD modelling and technical drawings of Skippy's mechanical components from a qualitative design,
- physical assembling of Skippy, and
- integrating the sensorimotor system.

The research work consisted in

- presenting and applying a methodology to test the sensorimotor system in all possible Skippy's working scenarios,
- implementing Featherstone's balance controller on three different 2D balancing machines (i.e. a reaction wheel pendulum, a rigid inverted double pendulum and a spring-loaded inverted double pendulum),
- extending Featherstone's balance controller to the case of robot balancing in 2D on rounded feet or wheels, and
- balancing and hopping with Skippy in 2D.

1.3 Thesis Overview

The structure of this work follows the steps taken in building Skippy, and it is organised as follows.

- Chapter 1: This chapter introduces this thesis with a general overview of the contents. It presents the motivations, objectives and contributions of this work.

- Chapter 2: This chapter covers the necessary background to understand the motivation and scope of this thesis. It starts with a brief outlook on legged robots and then presents state-of-the-art research in the field of balancing and hopping machines, with the main focus on underactuated robots. It then introduces the problem of testing the sensorimotor system of the robot. The final topic of this chapter is a literature review of the balance controllers used in robotics.

- Chapter 3: This chapter describes the experiments performed to test the behaviour of Skippy's sensorimotor system. It tests the encoders, motor and control unit in case of high impact shocks (e.g. foot hitting the ground after a high hop or robot crash landing after a wrong manoeuvre), and the inertial measurement unit in case of a low acceleration continuous motion (e.g. robot continuously hopping).

- Chapter 4: This chapter presents Featherstone's balance controller, which is the balance controller used in this thesis. It then proposes a simplified version of such a controller for the special case of a reaction wheel pendulum; eventually, it tests it on a purpose-built reaction wheel pendulum.

- Chapter 5: This chapter experimentally proves Featherstone's balance controller on an inverted double pendulum. The focus is on the experimental procedure and results, relying on the theory already shown in the previous chapter.

- Chapter 6: This chapter extends Featherstone's balance controller to the case of a robot that balances on rounded feet or wheels on flat horizontal surfaces. It starts presenting the controller and validating it in simulation and with a real robot. Then, the controller is further extended to the case of a robot balancing on a slope, and simulation experiments prove such an extension.

- Chapter 7: This chapter introduces Skippy in its 2D version. The robot proved to be able to balance and hop without losing balance. Furthermore, this chapter proves the robustness of Featherstone's controller to unmodelled robot parameters, being the spring-loaded leg of Skippy, of which the controller is completely unaware.

- Chapter 8: This chapter is the conclusion of this thesis. It summarises the contribution of this work and proposes directions for future works.

Chapter 2

Background

Legged mobile robots have become increasingly important in the field of robotics, serving as both research platforms for testing advanced motion control algorithms and as specialized tools for automated tasks in industry. Examples of these robots include HyQReal [21], Spot [14], ANYmal [11], and UniTree Go1 [22], all of which demonstrate the capabilities of quadrupedal robots, Figure 2.1. These robots are designed to be agile and versatile, allowing them to navigate difficult terrain, jump over obstacles, and perform manoeuvres that were previously impossible for slow or statically balanced robots. Another trend in legged locomotion consists in hybrid wheel-leg design, where the robots use wheels to move faster in flat terrain and legs to go through unstructured and rough terrains. Examples of this type of robot are shown in Figure 2.2, and are CENTAURO [23], ANYmal with wheels [24] and Mantis [25]. These robots vary significantly in shape and size but have the same motion strategy; they rely on efficient wheeled locomotion when moving on structured terrains, and they overcome obstacles using their legs. Another example of hybrid locomotion can be found in [26], where the authors designed a hopping Segway-like robot which is supposed to use wheeled locomotion in even and structured terrains and hop over obstacles by means of a spring-loaded approach. There are also examples of bio-inspired hybrid locomotion such as MSU Tailbot [27] (Figure 2.2d), which merges wheeled locomotion, jumping abilities to overcome obstacles and aerial manoeuvring ability of lizards. Another branch of legged locomotion uses humanoid robots such as Atlas [15], Digit [16], WALK-MAN [28] or TALOS [29] to explore bipedal locomotion, Figure 2.3. All these robots are designed and built to be highly complex, and then it is the work of the control system to achieve high performance. This is essentially the opposite design process to that followed in this thesis and in the Skippy project, where the highest performance is sought by means of a robot deliberately designed to be as simple as possible.

(a) (b)

(c) (d)

Figure 2.1 Quadruped robots: (a) Istituto Italiano di Tecnologia (IIT) HyQReal, (b) Boston Dynamics Spot, (c) ANYbotics ANYmal, and (d) Unitree Go1.

(a) (b)

(c) (d)

Figure 2.2 Hybrid robots: (a) IIT Centauro, (b) Università di Genova (UniGe) Mantis, (c) ANYbotics ANYmal with Wheels, and (d) MSU Tailbot.

(a)

(b)

(c)

(d)

Figure 2.3 Humanoid robots: (a) IIT Walk-Man, (b) Agility Robotics Digit, (c) Boston Dynamics Atlas, and (d) Pal Robotics Talos.

This thesis mainly focuses on dynamic balancing on a purpose-built underactuated one-legged robot called Skippy which balances on a single point. Skippy is a modular robot and can be configured in various ways, which can be used to explore 2D balancing, 3D balancing and hopping.

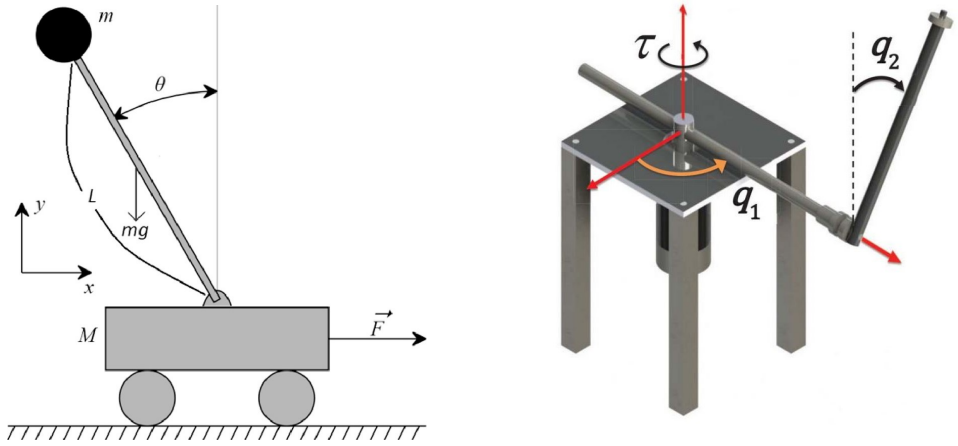
Multiple examples can be found in the literature of balancing machines that can travel in space by means of hops, such as Raibert's one-leg robot [30] or Carlési and Chemori's Kangaroo [31]. Other examples try to mimic animal jumping behaviours, such as Zuo, Lin and Wang's kangaroo [32] or the hopping machine of Zhang et al. [33] that mimics the locomotion of locusts. All these robots have in common the ability to travel in space (or plane) using hops. Some of them, such as Zhang et al. [33], use a pole leg (a second leg) to self-right and steer.

Skippy is intended to push a step further the mobility of hopping robots. It is designed to balance and reorient itself in 3D and then to hop to move in space. Furthermore, Skippy will be able to self-right after every crash land, regardless of the robot's orientation, which is an improvement of the self-righting technique proposed by Zhang in [34] where the robot cannot self-right when it falls on its backside.

2.1 Underactuated Robots

An underactuated robot has fewer actuators or control inputs than the Degrees of Freedom (DoFs) in its mechanical structure. In simpler terms, an underactuated robot has more motion freedoms than it has control mechanisms to directly command each of those freedoms, opposite to fully actuated robots, where each DoF is directly controlled.

Underactuated robots often rely on passive dynamics, such as gravity or mechanical coupling between joints, to achieve desired movements and behaviours. By leveraging these passive dynamics, underactuated robots can exhibit efficient and agile locomotion, adaptive responses to disturbances, and environmental compliance. The reduced number of actuators can lead to more lightweight and cost-effective designs. The design of underactuated robots often prioritizes efficiency in terms of energy consumption and control effort. Underactuated robots can operate with reduced power consumption and mechanical wear by minimizing the number of actuators required for a given task. Underactuated robots find applications in various fields, such as legged robotics, robotic hands and manipulators, aerial robotics, and underwater robotics. The rest of this section will analyze balancing and hopping underactuated robots.



(a) (b)

Figure 2.4 (a) Pendulum on a cart [2]; (b) Rotary inverted pendulum [3].

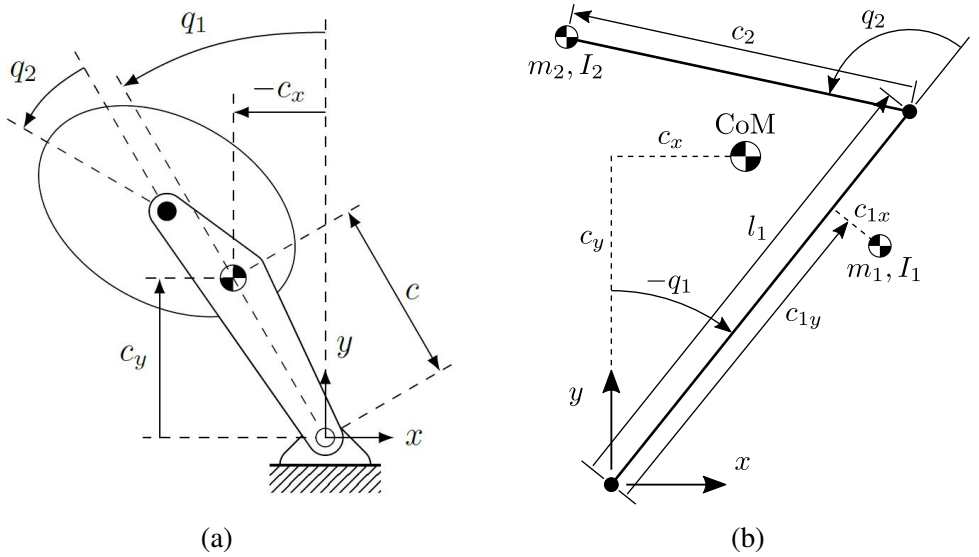
2.1.1 Balancing Robots

Underactuated balancing machines generally don't have direct control over the support point on which they balance. Literature is rich with examples of balancing machines often used to test and validate new control techniques, with the most common type being the inverted pendulum. It is because of its open-loop instability and highly nonlinear characteristics combined with a simple mechanical design and physical realization; furthermore, it can be used to describe bipedal locomotion [35]. The most common examples of the inverted pendulum are

- Pendulum on a Cart,
- Reaction Wheel Pendulum, and
- Double Inverted Pendulum.

Pendulum on a Cart

The pendulum on a cart is a well-known and deeply studied balancing device used extensively in education contexts. It consists of a single (or double) link inverted pendulum pivoted on a sliding cart (Figure 2.4a), or a rotating rod [36] (usually called the Furuta Pendulum [3]) (Figure 2.4b). In both cases, the system doesn't have direct control over the stick's angle but it has only horizontal or rotary control over the support. Multiple and various techniques have been tested over the years to balance such mechanisms, such as pole placement [37], neural



(a) (b)

Figure 2.5 (a) Reaction wheel pendulum [4]; (b) Inverted double pendulum [5].

networks [38], fuzzy-logic [2], sliding mode and Linear Quadratic Regulator (LQR) [39] for the linear model; and LQR control with refined Proportional Integrative Derivative (PID) [40] or Proportional Derivative (PD) and fuzzy logic [41] for the rotary model.

Reaction Wheel Pendulum

The Reaction Wheel Pendulum (RWP) is another example of a planar inverted pendulum, Figure 2.5a. The 3D extension of the RWP is called reaction mass pendulum [42], and the authors use it to map the momenta of a humanoid body. Applications of reaction wheel machines are of great interest because they are not only limited to terrestrial machines, like in [43], where the authors equip a quadruped Unitree A1 with two reaction wheels to increase its stability and disturbance rejection, but can also be found in aerospace devices [44], especially for precise attitude control of satellites [45]. The RWP consists essentially of a stick with an actuated symmetric link attached to the end whose rotation axis is parallel to the axis of rotation of the pendulum [46], which is generally fixed to the ground. In [47], the authors design, build and control a low budget RWP. They can then swing up and stabilize the pendulum using an energy-based strategy relying on an observer stabilizing controller and a two-stage bang-bang swing-up controller. Another example of RWP can be found in [4], where the authors apply for the first time a simplified version specific for RWP of the balance controller developed in [18] on a real robot called Tippy. Other examples of more sophisticated RWP can be found in Cubli [48], a cube-shaped robot able to balance both in

2D on a single edge and in 3D on a vertex. In [19], the author takes a step further in the generalization of the balance controller used in [4], trying to balance in simulation Tippy with an asymmetric reaction wheel; this modification essentially transforms the RWP into a inverted double pendulum, resembling Acrobot [49].

This thesis will present and test a planar balancing machine which can be described as a RWP and the balance controller used to control it. This is part of the *bend-swivel* balancing strategy that Skippy will use to balance in 3D, as anticipated in [50].

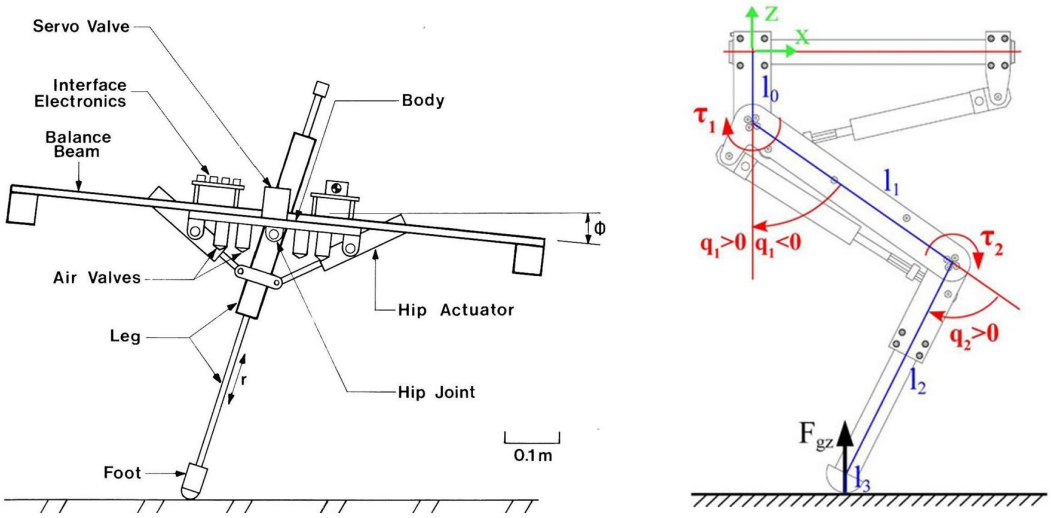
Inverted Double Pendulum

The inverted double pendulum consists of two links connected by a revolute joint (joint 2), as shown in Figure 2.5b, where the first link is connected to the ground through a revolute joint (joint 1). Such a model can be used to represent the dynamics of a biped standing on one point foot, where the configuration can be described as a inverted double pendulum with a controllable counterweight [51]. We can identify two types of underactuated inverted double pendulum; the first has joint 1 actuated and joint 2 passive, like Pendubot [52], the second type, instead, has the first joint passive and the second one actuated, similar to Acrobot [49]. These robots have been designed and used mainly to test newly developed non-linear control algorithms with the intent to stabilize the device in an unstable equilibrium point. In this thesis, greater attention will be devoted to the second type of machine because it is how Skippy will be modelled to balance in the sagittal plane.

2.1.2 Hopping Robots

In the case of hopping machines, we can divide the category into two main branches: bio-inspired and technology-inspired. To the first case belong the Kangaroo-like robots of [31] and [53], or the locust-inspired TAUB [54] or the frog-inspired Grillo [55] or flea-inspired [56]; a comprehensive review of biologically inspired jumping robots is presented in [57]. The second class, instead, includes robots like MSU Jumper [58], Acrobot [59], and Raibert's hopper [30], with the last two robots belonging to the category of the one-legged robot. Among these robots, some of them, such as Raibert's hopper [30] or Carlési and Chemori's Kangaroo [31] are capable of continuous hopping, others such as Zhang et al. jumping robot [33] or Scarfogliero's Grillo [55] can make only one hop at the time, being propelled by wind-up-and-release mechanisms.

One-legged robots are comprised of a single leg, typically with two or three links and at least one actuated joint. These robots serve as a basic model of mobile-legged robots and are



(a) (b)

Figure 2.6 (a) Linear actuation [6]; (b) Angular actuation [7].

frequently utilized as a testing ground for novel control algorithms or mechanical designs [7]. This thesis identifies two primary types of one-legged robots, distinguished by the type of actuated joint: linear (Figure 2.6a) and angular (Figure 2.6b).

Linear Actuation

The most famous and relevant hopping robot with linear actuation is Raibert's hopper [30]. More than 40 years ago, the author was able to build and control first a planar monopodedal hopper and then spatial one. The planar version of the robot consists of a telescopic leg and a torso connected by a hinge-type joint. The torso carries the actuators, while the leg is a pneumatic cylinder that acts as a spring. Position and orientation of the robot are estimated through the instrumented supporting boom. In the planar version, the robot has five motion DoF and only two actuators, one to stretch the leg and one to change the angle of the torso with respect to the leg. The robot has nine DoF and one extra actuator at the hip in its spatial version . The purpose of the extra actuator is to allow the leg to swing in any direction relative to the torso. The supporting boom is removed and all the sensors and interface electronics are contained in the robot's body. In both cases, the power supply and computational unit are not on board and are connected to the robot with an umbilical cord. Raibert broke the control of such a robot (modelled as a Spring-Loaded Inverted Pendulum (SLIP) [60; 61]) into three separate parts, consisting in controlling hop height, forward speed, and attitude of the torso,

allowing the robot to hop while moving in space continuously. The three-part controller has then been improved in recent years in [62], with the authors modifying the height controller.

Another example of a linearly actuated leg is presented in [63] where the authors modify Raibert's hopper to make it fully autonomous, and they replaced the pneumatic actuator of the leg with a voice coil actuator in parallel with two compression springs. In [64], the authors modify Raibert's hopper inserting a series elastic element between the actuator and the leg and they developed a high-order partial feedback linearization algorithm to control the robot's Centre of Mass (CoM).

Angular Actuation

Most quadruped robots have legs built with angular actuators. Legs' mechanisms can be serial chains, like in HyQ or ANYmal, or parallel chains, like in Delta-Quad [65] and Minitaur [66]. In [67], the authors use a scaled version of Minitaur's 5-bar mechanism leg to analyze the effects of leg configuration and leg length ratio on running stability and hop height. The experimental setup is similar to [68], with the leg connected to an instrumented 1.5 m aluminium rod with multiple functions such as constraining the motion of the leg, carrying the umbilical, and giving information about the robot position thanks to its encoders. The authors found the best ratio of the length of the links theoretically and validated it experimentally to achieve great speed, stability, and high hops.

Some monopedal robots try in a certain way to mimic the structure of animals' legs, being usually composed of a foot connected by a revolute joint (ankle) to a lower leg connected in turn via a generally actuated revolute joint (knee) to the upper leg, with some designs trying even to mimic the muscles behaviour [69]; others, like in [70], try to exploit the mechanical characteristics of the motors to design a monopod and a quadruped robot. One of the most famous examples of hopping monopedal robots is Salto-1P [71] which is an improved version of the Salto robot [72]. Salto-1P is an untethered lightweight robot with a total mass of 0.098 kg, less than 150 mm tall, and able to jump as much as 1.25 m vertically and 2 m horizontally, thanks to a series-elastic actuation for the leg, and it relies on thrusters to control its yaw heading. It has been intentionally designed to be a SLIP to have its control be as simple as possible based on Raibert's controller [30]. Unlike Raibert's hopper, Salto-1P is untethered. It is equipped with batteries to power the motors. However, it still depends on an external motion capture environment to provide position feedback and a ground station to estimate body velocity, desired leg lengths, and touchdown angles. The authors claim that using the attitude provided by the motion capture system prevents drifting due to the onboard gyro integration error.

An example of a spring-loaded hopping and running robot is the 2D bow leg robot [73] and its 3D version [74]. It is essentially a single leg made of a curved leaf spring, a foot, a freely pivoting hip and a string that holds the leg in compression. The leg servomotor controls the angle of the leg before impact; a second motor rewinds the bowstring to compress the spring determining the height of the following hop. The hopper is capable of hopping to a location and crossing various obstacles.

2.1.3 Skippy's Design Novelty

Skippy is intended to merge most of the features of the robots described in the Sections 2.1.1 and 2.1.2 into a single machine. It is designed to be fully autonomous, untethered, able to balance in 3D, continuously hop and make somersaults or other athletic manoeuvres. Skippy doesn't require any safety net or external protection system because it is designed to be shockproof and to be able to withstand crash landings due to accidents and mistakes. Moreover, all the computational effort is carried on onboard and with the operator able to start, stop, and log the experiment remotely.

Skippy will balance in 3D by merging the features of a double inverted pendulum and a reaction wheel pendulum in the so-called *bend-swivel* controller, first introduced in [50]. Skippy has only two actuators, one responsible for balancing the double inverted pendulum in the sagittal plane, the other actuating the reaction wheel to keep the sagittal plane vertical. The former is also accountable for making Skippy hop high; to increase the mechanical power and energy efficiency, curved regressive leaf springs are placed in series to the motor, and a passive spring-loaded ankle interfaces the leg to the ground.

2.2 Onboard Sensors

Autonomous mobile robots rely only on onboard sensors to estimate their position and orientation in space. In particular, Skippy is equipped with absolute encoders to know the internal angles of the mechanism, quadrature encoders to measure motors' velocity, and an Inertial Measurement Unit (IMU) to estimate its absolute orientation.

Since Skippy is a hopping robot, it will be subjected to considerable impulses whenever it lands on the ground or makes a crash landing. As Skippy is a monopedal robot, there needs to be a considerable amount of built-in shock protection for it not to have fatal falls from heights as high as 3 metres. However, to determine the accurate amount of cushioning without bulking up the weight of the robot unnecessarily, Skippy needs to have just the right

amount of shock-protective foams and sensors that work while the robot is subjected to the worst-case shock profiles. To the author's knowledge, no previous work in the robotics literature focuses on the mechanical shock testing of single components. Prior publications only considered the effects of such impacts on the assembled robot [75] or in the design process of the robot itself [76], with the intent of avoiding or mitigating them.

Skippy is designed to hop continuously; this motion causes the system to withstand what is referred in this thesis as *low impact accelerations* for a prolonged period of time. This motion doesn't affect the optical sensors, such as the encoders. Still, it may cause unexpected behaviour in inertial sensors, such as a drift in the IMU measurements.

2.2.1 Inertial Measurement Unit

The accuracy of Micro-Electromechanical Systems (MEMS) sensors for state estimation during dynamic behaviours has been previously studied but under different working conditions to those considered in this thesis.

Position estimation using IMUs has been addressed by [77], where the authors propose a cascaded Kalman filter with a fusion of GPS and IMU data for trajectory tracking. In humanoid applications, high-quality measurement of the floating base orientation can be achieved with an IMU, but achieving high-precision positioning with low drift remains a significant challenge. [78] describes different optimisation strategies and a state estimation algorithm to execute walking over non-flat terrains with the Atlas humanoid [15]. The authors used an IMU mounted on the pelvis to obtain its pose and twist. To reduce the drift of the robot measurements, they used an inertial and kinematic estimator, which was proven unsuitable for accurate walking over distances of tens of meters. They finally added a LIDAR to achieve the task. Instead, [79] analyses the effects of IMU drift on a walking person by using an IMU to estimate the vertical motion of the foot. However, the experiment is relatively short since the subject takes only a few steps. [80] uses a MEMS IMU to estimate the position of a walking person in an indoor environment where no GPS signal can be used. It calculates the body's location while moving, but the precision required is not the same as the one required by most robot-control algorithms.

On the topic of orientation estimation, in [81], the authors attempt to reduce the effect of drift in orientation tracking for a virtual reality head-mounted display. The sensor is mounted directly on the headset, and it experiences continuous motion but with very low accelerations. [82] presents a three-dimensional model of a quadruped robot with six degrees of freedom at the torso and five at each leg executing a 3D trotting gait. During these

experiments, the IMU (mounted on the robot's torso) experiences a severe drift on the yaw signal. The authors believe this drift was caused by the magnetic field excited by the motor in the treadmill. [83] proposes a cascaded two-step Kalman filter to compensate for external ferromagnetic disturbance or large impulsive acceleration (such as a single jump) or medium-long acceleration (roller-blade outdoors). [84] avoids the use of magnetometers to estimate the foot progression angle by means of IMU data. However, the experiments are limited to only a few steps and only to people walking because the authors sustain that estimation algorithms can be prone to magnetic distortion and inaccuracies after walking starts and turns. The effects of continuous low amplitude accelerations (e.g. a person walking) on orientation estimation have been addressed by [85]. In this work, the authors collect accelerometer and gyroscope raw data from people walking on a treadmill. Then, they process those data offline using a custom filter to compensate for drift and estimate the correct orientation. Such a strategy is often not feasible for embedded systems, which typically lack memory and computational power and must rely on the sensors' orientation data in real time.

IMUs have become popular also in the field of wearable devices. In [86], the authors combine an air-pressure and IMU sensor to recognise human activities. At the same time, in [87], they use a chest-worn band equipped with a breathing sensor and tri-axis accelerometer to measure respiratory parameters and identify human activities. A comprehensive literature review can be found in [88], where the main focus is IMU-based wearable devices in sports medicine.

2.2.2 Sensor's Testing Novelty

One contribution of this thesis consists of systematically analysing the behaviour of Skippy's sensors in extreme and potentially troublesome situations employing a purpose-built testing apparatus. More specifically, the encoders have been tested with high acceleration impacts, mimicking a crash landing of Skippy after a hop or an athletic manoeuvre. Also, the behaviour of the IMU will be evaluated while subject to continuous low acceleration impacts, similar to those experienced by the robot while continuously hopping.

2.3 Balance Controllers

All the robots described in Section 2.1 rely on *ad hoc* balance controllers to maintain stability at unstable equilibrium points. Typically, the control algorithms consist of a set of symbolic equations that express force variables using the robot's kinematic and inertia parameters

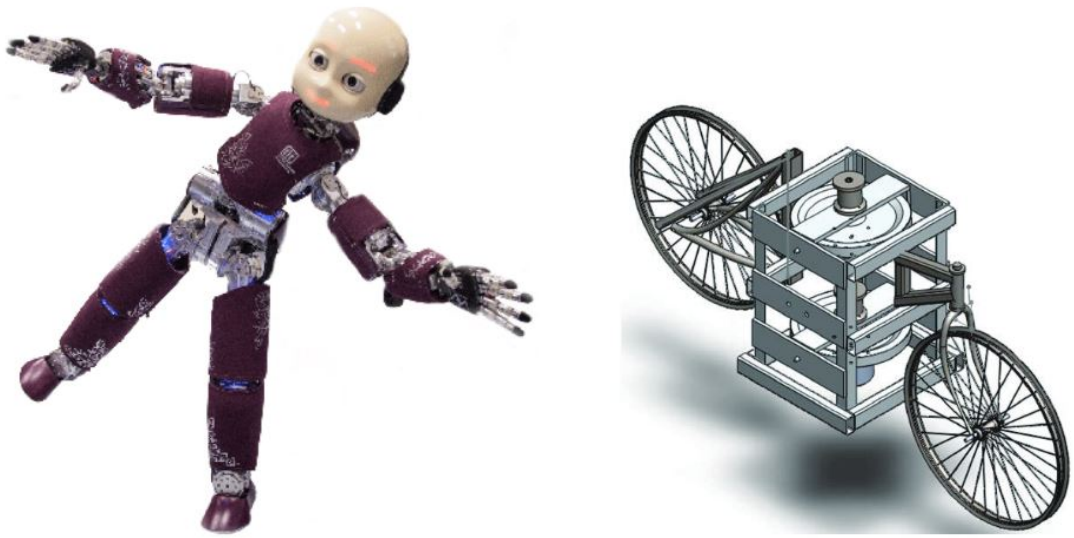
as well as state and acceleration variables. A feedback control law formula is then derived from this set of equations. Since these controllers are based on the closed-form equation of motion of the robot mechanism, they are specific to that particular device and cannot be utilized on different machines. Additionally, they are generally limited to stabilizing [9; 90] the mechanism and cannot enable the actuated joint used for balancing to do any other task, such as tracking a desired trajectory.

2.3.1 Dynamic Balancing

Dynamic balancing is the robot's ability to maintain its centre of mass above its support point over which it may not have direct or complete control while performing other tasks. Dynamic, or active, balancing requires the robot's actuators to work continuously to keep the robot balancing without falling; meaning that the robot cannot balance in such configurations when the motors are powered off, contrary to static or passive balancing, where it can balance on a stable support point without needing any motion of its actuated joints. Dynamic balancing can be achieved using balance controllers relying on the robot's angular momentum, such as Featherstone balance controller [18], which is the one that will be used to control Skippy in this thesis.

Balance controllers based on angular momentum are not a new invention in the robot control field. Their first implementation goes back in time, and it has various applications. In [89], the authors model a humanoid robot trying to balance on a single leg as an inverted double pendulum where only the ankle joint is actuated. This approach has been improved and deployed on iCub in [8], where the humanoid can balance on one foot while performing other activities with the other limbs, see Figure 2.7a. Angular momentum balance controllers are also used to balance robots on unstable supports; in [9], the authors balance and steer a bicycle, changing the angular momentum of two coaxial flywheels assembled on the bike (Figure 2.7b). In [90], the authors modify the angular momentum-based balance controller developed in [91] to stabilize a planar three-link robot composed of three joints (two of which passive). The ground is connected with a passive joint to the first link which is connected to the second link with the other passive joint. The only active joint connects the second link to the third one.

The most famous balancing machines, such as Cubli [48], Pendubot [52], or Acrobot [49], achieve balance on fixed single contact point—typically a shaft for Pendubot and Acrobot, or a cube edge for Cubli—and are not designed to move while balancing in space (although Cubli can "walk" by combining consecutive balancing and controlled falling).



(a) (b)

Figure 2.7 (a) IIT iCub balancing on a leg [8]; (b) Bicycle with flywheels [9].

In the context of legged robotics, quadrupeds can perform balancing on two legs. Many quadruped robots are typically equipped with ball feet. While these feet are relatively small compared to the overall size of the robot and provide a reasonable approximation of point feet, these robots are not specifically designed for maintaining balance on two feet. It has been demonstrated that the balance controller proposed in [18] can be applied to real quadrupeds like HyQ [92]. However, due to its complex mechanical structure, achieving balance has proven to be a challenging task, and the robot was only able to maintain static balance on two feet without actively controlling the other joints not involved in balancing. Another example of balance in quadruped robots can be found in [93], where the authors demonstrate balancing the MIT Mini Cheetah on two feet using a variational-based linearization technique to solve an unconstrained linear quadratic regulator problem that takes into account the orientation of the robot's torso.

Another category of quadruped robots combines wheeled locomotion on even surfaces with legged locomotion on uneven or unstructured terrains. In these robots, the foot is replaced by actuated wheels, allowing certain robots (e.g. ANYmal with wheels [94]) to maintain balance on their hind legs with wheels. The authors employed a whole-body Model Predictive Controller (MPC) to achieve highly dynamic motions, and in [95], they utilize reinforcement learning techniques to enable the robot to balance on its wheeled hind legs and navigate in space. This behaviour resembles an inverted pendulum, similar to the

well-known Segway [96] (also called Self Balancing Robot (SBR)), which has been the focus of numerous control strategies such as linear-quadratic regulation [97; 98], intrinsic geometric PID controller [99], or optimal control [100]. MPC can also stabilize and reject disturbances while balancing this type of robot. In [101] the authors propose a MPC solution to stabilize a SBR in simulation. In [102], the authors use MPC to stabilize an unstable heavy SBR developed in their laboratory. Due to the high computational effort required by the model predictive approach, the authors were forced to use a simplified plant model to allow the control to run online on the robot's control unit. Neither [101] nor [102] attempt any trajectory tracking, limiting the behaviour to balancing and disturbance rejection. The Ballbot [103] provides an example of an inverted pendulum that balances in 3D, with a stick balancing on a ball. The ball is actuated using an inverse mouse-ball drive, and the balance controller consists of two independent PID controllers, one for each of the vertical planes [104]. In [105], the authors employ a sliding-mode controller with a single-loop control system to track and balance the Ballbot.

2.3.2 Skippy's Controller Novelty

This thesis intends to implement and further develop the angular momentum balance controller proposed by Featherstone in [18], which has its roots in [50; 91]. This controller enables a robot to balance on a non-actuated fixed contact point while simultaneously tracking a desired trajectory for the actuated joint responsible for balancing. One of the goals of this thesis is to expand Featherstone's 2D balance controller from a fixed single contact support point to a rolling contact support point description. The newly developed controller will then be used to stabilize and move a stick-on-a-wheel robot, which is similar to a bidimensional Segway [106].

2.4 Thesis Contributions

This thesis aims to give the reader a better understanding of the manufacturing and controlling process of Skippy. The design of the mechanical components is out of the scope of this work, and individual parts or mechanisms will be described only in selected circumstances.

This thesis proposes self-implemented testing techniques to evaluate the performance of the sensorimotor system of a robot in extreme or unexpected working conditions. Knowing the limits and constraints of the sensorimotor system is vital in designing highly dynamic robots and their control system.

Once confirmed that sensors and actuators are fit for Skippy, this thesis describes the intermediate steps taken in the building process, starting from the most straightforward configuration, an inverted reaction wheel pendulum balancing in 2D, to the most complex, an inverted double pendulum with a springy leg balancing and hopping in 2D. This work lays the foundation to fully control Skippy in 3D, which will be the objective of future works.

Chapter 3

Sensors Testing

Mobile robots strive to achieve athleticism and dexterity to navigate through rough terrain, overcome obstacles, and perform agile movements. These actions generate impulsive dynamics that produce shocks, which can cause negative effects on the robot's sensors and mechanical components. Therefore, minimising the amount of shock propagating to the robot's torso is essential. Although cancelling shocks completely using the Centre of Percussion concept is ideal [76], it may not be practical in some robot leg designs. Hence, estimating shocks in terms of deceleration values, represented as g values, is crucial for designing appropriate damping and passive shock-absorption methods to ensure safe and reliable sensor operation, such as position sensors, IMUs, or cameras, within the robot.

MEMS, such as IMU, are embedded in almost all mobile robots thanks to the variety of measured signals they provide. They contain on-board processing systems called Attitude and Heading Reference Systems (AHRS), which combine the output of several sensors, such as a 3-axis accelerometer, a 3-axis magnetometer, and a 3-axis gyroscope to estimate a system's attitude, using a variety of algorithms, the details of which are usually proprietary. One particular computed output is the robot's orientation, which must be accurate up to a certain degree for the robot to operate as expected. Inaccurate estimation of orientation can cause a robot to veer off course or lose its balance and fall. In this thesis, these systems are generically referred to as IMU because it is in this way that the larger audience usually calls them.

The continuation of this chapter will analyse in depth the effects of high acceleration mechanical shock tests on absolute and quadrature encoders and the effect on a selection of IMUs of substantial oscillations in vertical acceleration caused by the alternation between flight phase and stance phase during fast legged locomotion. This systematic approach is essential to guarantee all the sensors' correct and expected behaviour during Skippy's

activities. The athletic and highly dynamic manoeuvres of Skippy put under stress all the electronic components, being subject to continuous vertical accelerations in case of continuous hopping behaviour or high decelerations due to falls while hopping. This testing approach evaluates each type of sensor individually, allowing the designer to select the most suitable sensor for the desired task. Furthermore, the control system must be aware of possible sensors' malfunctioning in advance to compensate for or mitigate them with an appropriate control strategy.

The Skippy team has developed content of this chapter and its outcome has already been published. Specifically, the results obtained in Section 3.1 have been published in [107] (© 2023 IEEE, reprinted with permission), and all the figures and tables in Section 3.1 are taken from it. My contribution to Section 3.1 consisted of

- the definition of the experimental procedures,
- the execution of the experiments,
- the data collection, and
- the analysis and presentation of the collected data.

The work proposed in Section 3.2, instead, has already been published in [108], and all the figures and tables in Section 3.2 are taken from it. My contribution was to

- design and build the testing apparatus,
- write the acquisition software for the BNO055 IMU,
- integrate the acquisition software of the 3DM-GX5-15 IMU,
- perform the experiments,
- post-processing of the collected data, and
- analyse and present the obtained results.

3.1 Encoders Testing

This section investigates the behaviour of Skippy's encoders when subject to high accelerations caused by a mechanical shock to determine if it causes any malfunctions and whether the encoders can recover from the impact. A situation like this can happen during a crash

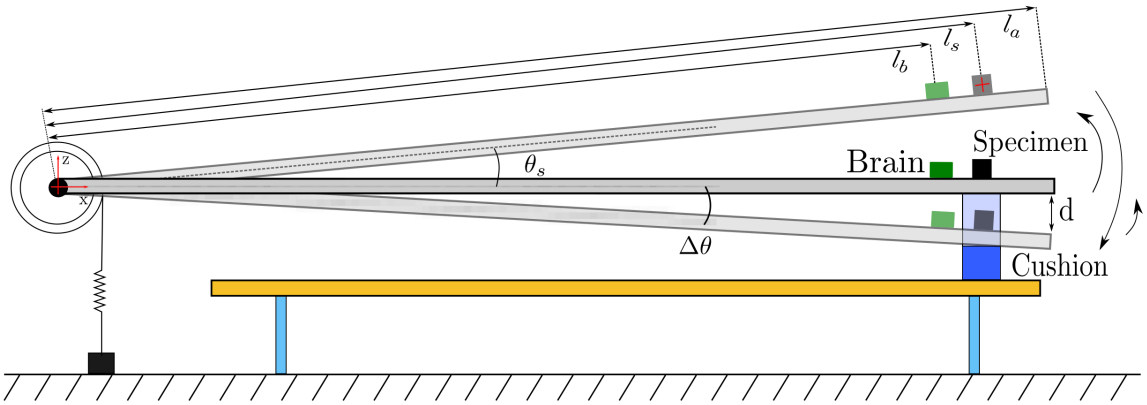


Figure 3.1 Catapult qualitative representation.

landing and when the foot hits the ground during normal and high hopping. Additionally, this study aims to provide users with more detailed information about shock resistance than what is typically provided by vendors, and a methodology for testing self-made electronics, such as the data acquisition system. While some data sheets, such as that of the AksIM-2 rotary absolute encoder [109], guarantee to function during shocks up to 100 g (6 ms, half-sine, according to EN 60068-2-27:2009), they do not provide information about recovery from more significant or different types of shocks. Other data sheets, like that of the Maxon ENX 16 EASY incremental encoder [110], do not mention shock resistance at all. As a result of this experiment, a simple methodology for testing such devices has been developed, which evaluates the impact without using expensive or fragile accelerometers or IMUs typically found in robotics laboratories, which have a saturation limit lower than the amplitude of the shock impact. The experimental procedure is simple and effective. It consists of slamming an actuator, a set of rotary sensors and their computational unit against a cushion using the spring-loaded catapult presented in [141§ 6.3.3]. During the initial impact and subsequent rebounds, all information received from the sensors is logged.

3.1.1 Experimental Setup

The equipment required for this experiment includes the following components: a DC motor, the slamming catapult, a cushion, the computational unit of Skippy (referred to as the ‘Brain’), and the two encoders under test, see Figure 3.2.

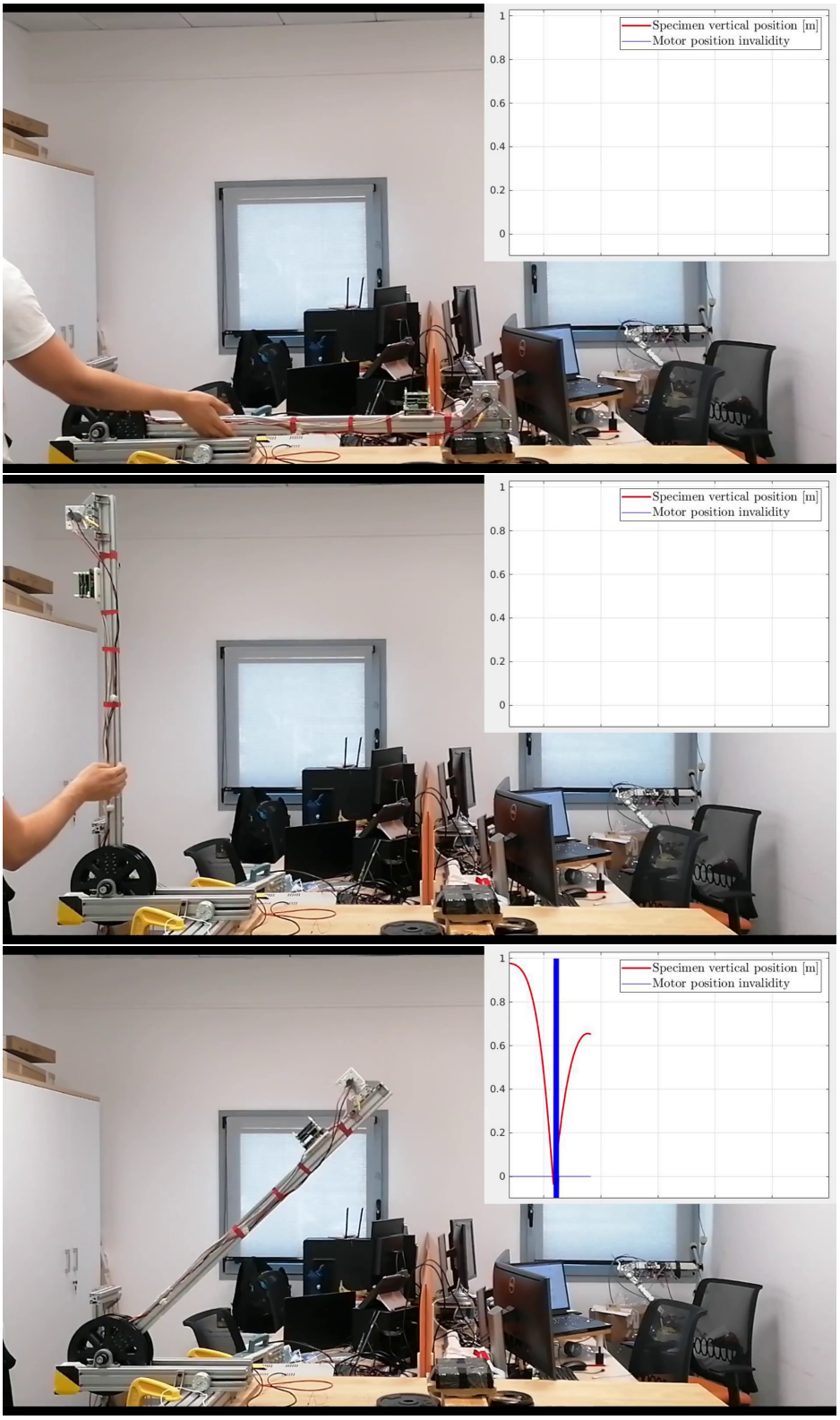


Figure 3.2 Experimental setup. Catapult at rest (top), catapult lifted (middle), and catapult bouncing (bottom).

Length of the catapult arm [m]	l_a	1.03
Distance of the specimen [m]	l_s	0.98
Distance of the brain [m]	l_b	0.8
Mass of arm and specimen at the tip [kg]	m	1.4
Pulley radius [m]	l_r	0.120
Spring stiffness [N/m]	k	147
Number of springs	n	4

Table 3.1 Catapult specifications.

Catapult

The spring-loaded catapult is the tool used in this experiment for shock-testing the specimen (Figure 3.1, Table 3.1). Instead of the typical catapult mechanism where the specimen is released, our set-up involves securing it firmly to the aluminium arm [111] and slamming it into a cushion. We utilize four linear springs that connect the pulley of the catapult to a fixed base. The springs are connected to the pulley through a rope that rotates the pulley. The arm of the catapult is directly connected to the same rotary joint as the pulley. We measure this angle using an absolute rotary encoder (more details in Section 3.1.2), an integral component of the catapult itself. The specimen is placed at the tip of the catapult's arm.

The catapult uses the previously mentioned springs to store and release energy for slamming objects into the cushion. An operator manually lifts the arm to an angle that will result in the desired impact conditions (e.g. a given velocity at impact) and releases it. At the moment of impact, all the springs have already returned to their rest length, no longer storing potential energy. The catapult arm is assumed to be perfectly rigid.

Cushion

The cushion consists of two layers of 2 cm thick self-adhesive high-density foam on a wood block. The thickness of the block is chosen such that the arm of the catapult rests horizontally on the cushion. The cushion is fixed on the workbench and compresses when the catapult hits it, simulating the effect of a crash landing. The material used in this experiment is similar to the protective cushioning employed to reduce the impact of a crash landing on Skippy.

Brain

It is the computational unit used in this experiment and in Skippy. It is responsible for sampling and logging all the sensors through all the slams. A complete description of the Brain can be found Section 4.4.4 and in [112].

Specimen

At the end of the catapult arm, there is a frame that carries the motor and both encoders, which is the specimen used in this experiment. The DC motor utilised in the experiment is a Maxon DCX22S GB KL 24V [110], which is powered by a constant voltage from a bench power supply to maintain a constant speed of 7700 rpm. This speed was chosen to avoid exciting any resonances in the catapult arm. The motor is equipped with an ENX 16 EASY incremental encoder, which is one of the two encoders being tested. Additionally, the motor's output shaft is connected through a coupling to a shaft that carries the magnetic disc of an AkSim-II rotary absolute encoder [109], which is the second encoder under test. Since both encoders measure the same angle, their measurements should agree. A detailed description of the components of the specimen can be found in Section 3.1.2. The vertical position of the specimen varies depending on the angle of the catapult and it is defined as follows:

$$z_s = l_s \, sin(\theta_s) \tag{3.1}$$

where z_s is the position of the specimen with respect to the z axis and θ_s is the angle between the x axis and the catapult arm (positive in the counterclockwise direction), see Figure 3.1.

During the whole experimental procedure, the whole mechanical structure vibrates when the motor rotates. This is due to a slight misalignment between the two shafts in the specimen. As a result, both encoders are exposed to these vibrations, leading to testing of the encoders under the influence of combined shock and vibration, even though it was not the initial objective of the experiment.

3.1.2 Data Acquisition System

The data acquisition system comprises a computational unit (the Brain), two absolute rotary position encoders and one incremental position encoder. The sampling frequency for the experiment is 5 kHz, which is the same frequency that Skippy will use in it's control loop. The measured angles are unwrapped by the signal processing software to produce measurements of the total amount of rotation since the start of the experiment. This step facilitates the

comparison of encoder readings and allows us to spot cumulative errors, if present. The Brain provides power to all the encoders. The incremental encoder and the motor absolute encoder form the specimen that undergoes shock testing.

Absolute rotary encoders

The two absolute rotary encoders are RLS AksIM-2 [109] with different sizes and configurations of magnetic rings and read-heads and they are mounted on the motor and the pulley, respectively. In the continuation of this section, such encoders will be referred to as

- Motor encoder: it is employed for directly measuring the angular position of the shaft that is connected to the motor. It has a magnetic ring of 29 mm diameter and an angular position resolution of 18 bits, and its hardware configuration, as defined by the manufacturer, is as MB029DCC18BFNT00 (see [109]).
- Catapult encoder: it is employed to determine the angular position of the arm of the catapult, θ_s, in relation to the x axis, as depicted in Figure 3.1. It has a magnetic ring of 39 mm diameter, an angular position resolution of 17 bits and the capability to track the number of revolutions. Its hardware configuration, as defined by the manufacturer, is as MB039DCC17MEDT00 (see [109]).

They are both sampled by the Brain via the BiSS-C interface, with a clock frequency of 3.8 MHz. The data from the encoders are converted into radians and then logged.

Incremental encoder

The incremental position encoder is a Maxon ENX 16 EASY [110]. The incremental interface generates 4096 steps per turn. The encoder is sampled by the Brain's Enhanced Quadrature Encoder Pulse (eQEP) module [113]. The eQEP register is read together with the absolute encoders, and its value is converted into radians before being logged.

3.1.3 Experimental and Analysis procedure

The experiment presented in this section aims to investigate the behaviour of the encoders intended to be used in Skippy. We designed an experiment to investigate the behaviour of the encoders during and after mechanical shocks, comparable to those expected during controlled high hopping and accidents, such as crash landings, which Skippy is expected to survive. Given the hardware's limitations, the analysis complies with the standard 60068-2-27 [114]

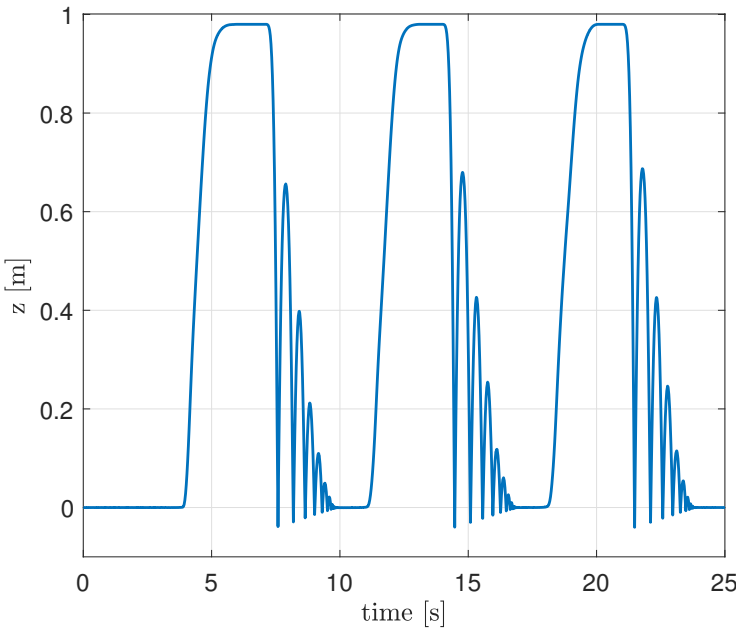


Figure 3.3 Specimen's vertical position during multiple slams. The red line is 10 times the measured angular position of the catapult, the blue line is the estimated velocity of the specimen and the yellow line is the specimen's interpolated velocity during the rebound.

as closely as possible as described in the following section. The experiment also tests the robustness of the Brain against shocks.

Experimental Procedure

At the beginning of the experiment, the catapult arm is in a resting position and is aligned with the x axis, as illustrated in Figure 3.2. The operator manually lifts the arm until it reaches the physical end-stop, causing the arm to be parallel to the z axis, Figure 3.3. The arm is then released, resulting in a rotation around the revolving axis of the pulley (y axis) and a subsequent impact with the foam. To comply with the IEC international standard 60068-2-27 [114], three consecutive slams should be performed with the same initial conditions, and the specimen should be shock-tested in each direction. However, due to hardware limitations, the specimen is only tested with shocks perpendicular to the rotation axis of the motor. Since the system is symmetric with respect to the rotation axis of the motor, only one shock direction is physically tested. The experiment can be viewed in the accompanying video of [107].

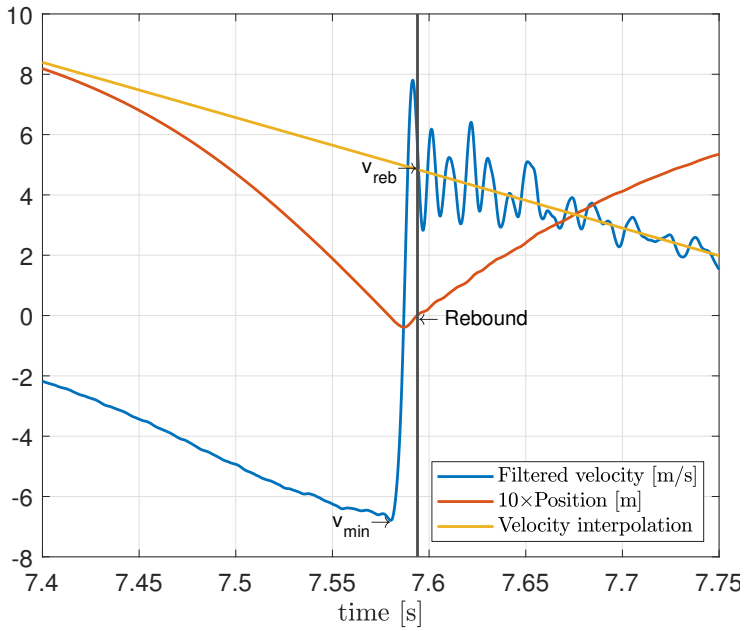


Figure 3.4 Velocity of the specimen before and after the impact.

Analysis Procedure

The analysis of the data collected during the shock experiments is split into two stages. Firstly, the peak impact is estimated and then the consistency of the measurements is examined throughout the multiple impacts.

Shock Estimation

The estimation of the vertical impact suffered by the specimen has been carried out starting at the point when the catapult is released from a vertical position. To determine the velocity of the specimen, the change in measured position is differentiated with respect to the time difference between two successive samples. To filter out undesired high-frequency noise, a third-order low-pass Butterworth filter is applied. The velocity signal is first transformed to the frequency domain using a Fast Fourier Transform (FFT) and filtered using a cutoff frequency of $F_c = 115\,\text{Hz}$, which complies with the 60068-2-27 standard. According to the standard, the lower limit of the cutoff frequency is $f_g = \frac{1.5}{D}$, where D is the duration of the shock, which in this experiment is $D = 14.6\,\text{ms}$, hence $f_g = 103\,\text{Hz}$. To ensure accuracy, only the interval of time containing the shocks is differentiated and filtered (Figure 3.4). Each experiment starts with a velocity of $6.8\,\text{m/s}$ just before impact, which is equivalent to a crash landing of a body falling from a height of $2.36\,\text{m}$.

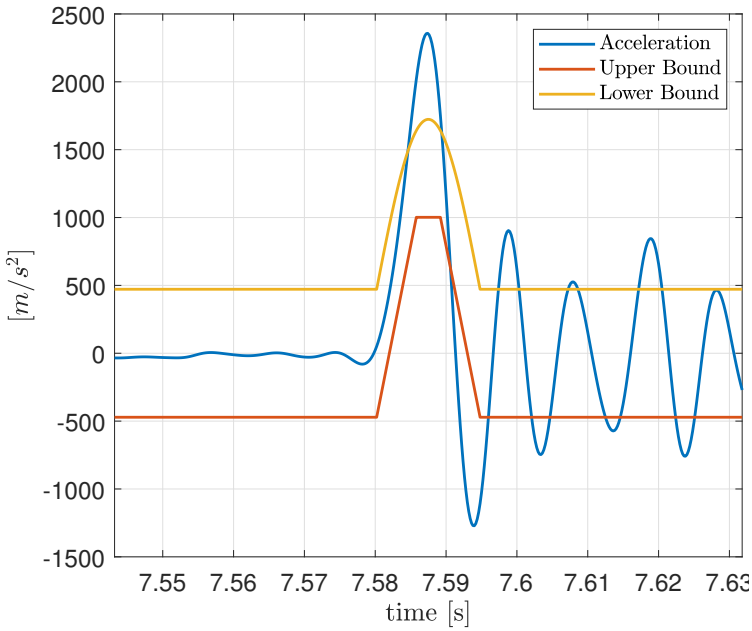


Figure 3.5 Acceleration peak and half sine shape mask.

The velocity signal obtained from the rebound of the specimen showed unexpected oscillations at a frequency of 95 Hz, which were found to be caused by the bending of the catapult arm. This led to an overestimation of the rebound velocity. To compensate for this, the rebound velocity was determined as the intersection of the least-squares fit yellow straight line, which interpolates the curve (see Figure 3.4), and the moment when the specimen loses contact with the foam (intersection of vertical line and slope of velocity in Figure 3.4). It was observed that the initial assumption of the catapult arm as a perfectly rigid body was incorrect, as the arm bends upon hitting the cushion and then vibrates, causing an overestimation of the peak acceleration. As per the standard, the acceleration should be enclosed in the 'Half Sine Shape', which defines the nominal value of the amplitude of the shock A (Figure 3.5) as

$$A = \frac{\pi \Delta V}{2D} = \frac{\pi (v_{reb} - v_{min})}{2D} = 128\,g \tag{3.2}$$

where ΔV is the velocity variation, v_{reb} is the rebound velocity, v_{min} is the impact velocity and D is the duration of the shock.

Catapult Arm Bending

To confirm the bending of the catapult arm, a simple experiment was conducted. A small and lightweight peg was placed in a hole next to the foam, which had a frictional fit. The compression of the foam was measured by observing how far the catapult arm pushed the

peg into the hole. The difference between the initial position of the peg (which was at 0 on the z axis) and the final position after being hit by the catapult was used to determine the actual compression of the foam. The experiment resulted in a measured compression of 3.2 cm, while the value calculated from the measured angle of the catapult was 4 cm. This difference of 0.8 cm implies a bending of 0.4 cm of the catapult arm at its centre, causing significant oscillations, and hence making it incompatible with the shock-testing standard. Consequently, the shock information on the AksIM-2 datasheet [109] doesn't apply to our experimental setup.

Motor Position

Both encoders of the specimen measure the same angular position of the rotating motor. Both motor angular positions are unwrapped; in this way, the signals lose the discontinuity typical of the angle of a motor rotating always in the same direction, and it is possible to compare them. The angular position of the motor has a constant velocity at 7700 rpm due to the steady power supply, resulting in a linear trend.

In this thesis, the difference between the measurement of the motor's absolute encoder and the incremental encoder is defined as the motor position error, or simply the error. This error is assessed at various intervals, both before and after the shock. At the start of each interval, the unwrapped position of each encoder is reset to zero, to account for any initial offsets caused by the assembly of the encoders. When the motor begins to rotate, the incremental encoder begins counting from zero, whereas the absolute encoder starts from the position of the magnetic ring, which is determined by the physical angle of the ring.

The error is not assessed during the impact because the data received from the motor's absolute encoder are not valid. During the shock period, most of the data packets received from the encoder have an incorrect Cyclic Redundancy Check (CRC) [115], indicating a communication error. As shown in Figure 3.6, the data packets received from the motor's absolute encoder are not reliable during the contact phase of the shocks. This pattern is observed in each slam and the subsequent three rebounds. Table 3.2 presents the average duration of the communication inconsistencies, through three slams, relative to the duration of the shock and the peak acceleration.

The quality of the signal from the incremental encoder is verified by recording the latch value of the eQEP register. This register stores the count value every time the index signal is high and is sampled synchronously. If the data acquisition system has not missed any incremental count, the latch value should be 4095 (or 0 depending on the motor's direction of rotation). Although the sampled values are accurate during the shock, some

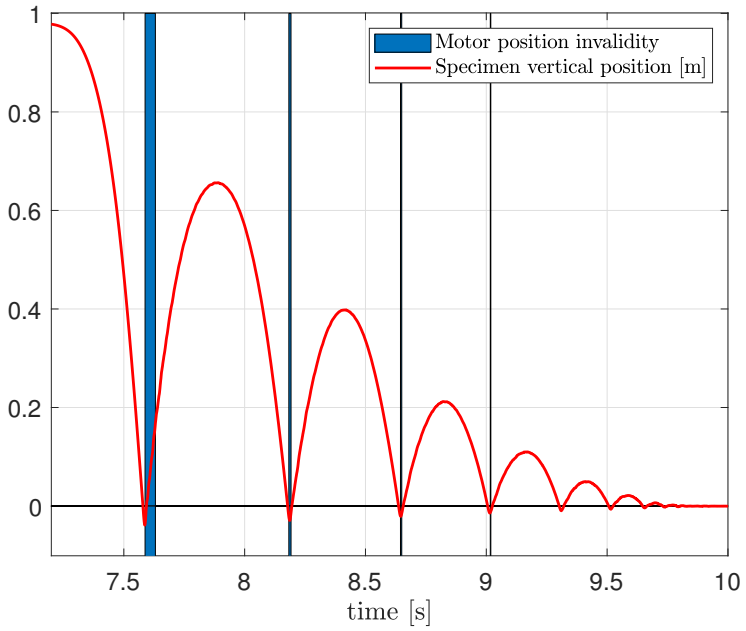


Figure 3.6 Validity of the motor absolute encoder measured position with respect to the specimen vertical position. The blue bar highlights when most of the data received from the motor encoder is not valid.

Shock N.	Shock Dur.	Peak Accel. [g]	CRC Error Dur.
1	14.6	128	45.6
2	17.7	93	9.7
3	19.1	65	7.5
4	19.4	45	4.8
5	18.5	30	0

Table 3.2 Average values over three consecutive slams. Each slam causes four subsequent rebounds where information is lost. Shock Duration (Dur.) and CRC Error Duration are in milliseconds.

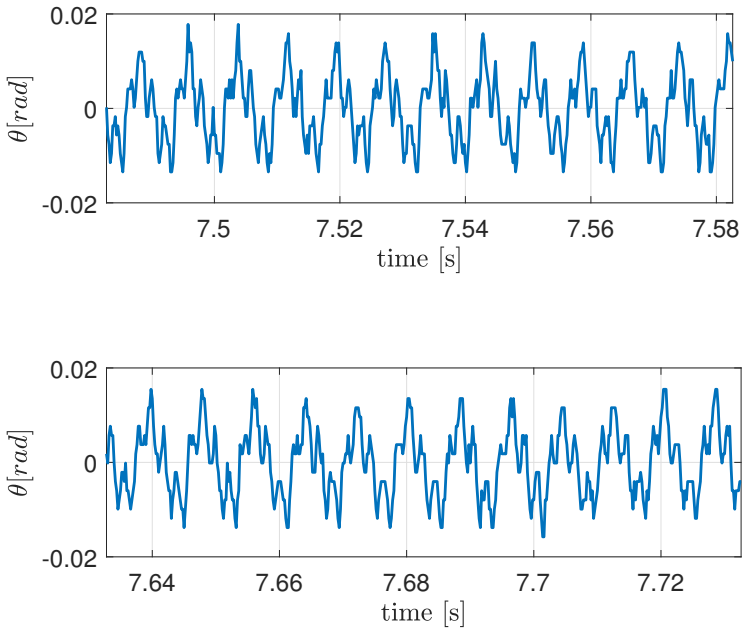


Figure 3.7 Position error θ between absolute and incremental encoder. Two intervals of time are reported: before (graph above) and after (graph below) the shock. During the shock, data are unreliable, so they are not presented.

malfunctioning of the encoder latched value has been observed. To determine the frequency of this malfunctioning, the motor was spun at a constant speed of 7700 rpm for one minute without any shock. The sampling frequency was 5 kHz. Out of 7700 turns, the latched value was not as expected (4091 instead of 4095) on 12 occasions, which amounts to 0.16 % of the total cases.

The error was examined before and after the impact of the first slam (as shown in Figure 3.7). It is evident that the shock does not affect the behaviour of the position error. The primary component of the periodic trend of the error is attributed to a slight misalignment of the rotating shaft concerning the motor and its holder, which causes vibrations at a frequency of approximately 128 Hz. The root mean square error (RMSE) and peak-to-peak error (PTPE) were assessed over two 0.5 second periods, one before and one after the impact. The values obtained (shown in Table 3.3) and a segment of these time intervals (as illustrated in Figure 3.7) demonstrate the reliability of the position measurement even after the impact. The error's magnitude is quite high, and it is a result of the vibrations generated by the rotating motor and its holder, along with the misalignment between the magnetic ring of the absolute encoder and the motor's rotor. However, the shocks do not influence the error, indicating that the encoders can survive the shocks or recover from communication problems, such as CRC errors.

Before Impact		After Impact	
RMSE [rad]	PTPE [rad]	RMSE [rad]	PTPE [rad]
0.0073	0.033	0.0072	0.035

Table 3.3 RMSE and PTPE of the position error over two different periods of time T=0.5 s.

3.1.4 Conclusions

The main contribution of this work is to demonstrate the importance of testing sensorimotor systems and their computational units in failure scenarios (e.g. robot falls and crash lands). These tests can assist in developing more robust robots and provide helpful knowledge that can be integrated into the control system, such as the presence and duration of unreliable sensor information during and after a crash.

The findings indicate that data is lost during the initial impact (at 6.8 m/s) and three subsequent rebounds, either due to communication or encoder hardware errors. The system functions normally during shocks with peak acceleration up to 30 g, which provides guidance for designing robots that do not exceed this value during normal operations. This approach can lead to more robust robots. The study tested the system under an extreme crash landing scenario, where temporary errors are acceptable, but consistency is required during normal operation. The experimental results show that information is lost from the absolute encoder due to communication or hardware problems during the initial shock and the subsequent rebounds. However, the data acquisition unit operated consistently in all situations.

To the best of the author's knowledge, there are no high-impact crash-landing tests in the current robotics literature because most robots are designed under the assumption that they will not experience severe shocks. However, since accidents are likely to occur when robots are used in real-world situations, it is critical to be aware of sensory limitations and carefully consider them in the design process to ensure that the robot can survive such crashes and continue to operate with minimal harm.

3.2 IMU Testing

Even though several legged robots have already demonstrated motions involving substantial repetitive vertical accelerations [30; 116], the effects of these motions on IMUs have not been properly investigated. To achieve consistent and reliable behaviours, legged robots must be aware of their electronics' limitations. This issue can be addressed at the design stage by choosing an IMU that is accurate enough under the expected operating conditions, or it

can be addressed at the control stage by adapting the robot's motion planning and control to make allowance for the IMU's limitations. Either way, the first step is to identify and measure those limitations.

This section investigates the effect on a selection of IMUs of substantial oscillations in vertical acceleration caused by the alternation between the flight and stance phases during fast-legged locomotion. It does this using a 'bounce test' apparatus, as described in Section 3.2.1. This thesis refers to these oscillations as 'low-acceleration impacts' to distinguish them from the much higher accelerations experienced during accidental collisions and falls of Section 3.1. The motivation for this research is the lack of claims, both in the manufacturers' and research literature, that these devices have been designed and tested for use under these conditions. Specifically, the experiments reported in this section investigate the following items: drift in the IMU's orientation estimates during approximately vertical bouncing motion; the timescale on which this drift occurs; and the time it takes the IMU to recover after bouncing ceases.

3.2.1 Experimental Setup

The experiment investigates the effects of prolonged, continuous vertical bouncing motion on the accuracy of orientation estimation on a selection of MEMS IMUs. The apparatus is shown in Figure 3.8, and consists of a long rod with a rotary actuator at one end and the IMUs at the other. Strictly speaking, this means that the IMUs travel along a circular arc. However, the radius of the circle is large enough for it to be a reasonable approximation to vertical motion.

Actuation System

The actuation system consists of a Magnet Schultz Rotary Solenoid Type G-DR-075X20A61-S1 [117] powered by a Pololu G2 High-Power Motor Driver [118]. This solenoid produces pure rotational motion and offers controllable bidirectional torque, making it easier to measure the rotation angle and control it compared to most of the other so-called rotary solenoids, which produce helical motion and do not allow controllable bidirectional torque. The solenoid directly actuates one end of a 1 m carbon fibre tube of 20 mm external diameter and 0.5 mm wall thickness. The tested specimen is placed on the other end of the tube, opposite to the solenoid. The angular position of the tube is measured with an AksIM-2 absolute position encoder [119], sampled at 500 Hz by the National Instrument Single Board sbRIO-9637 Field Programmable Gate Array (FPGA) [120], which also generates control

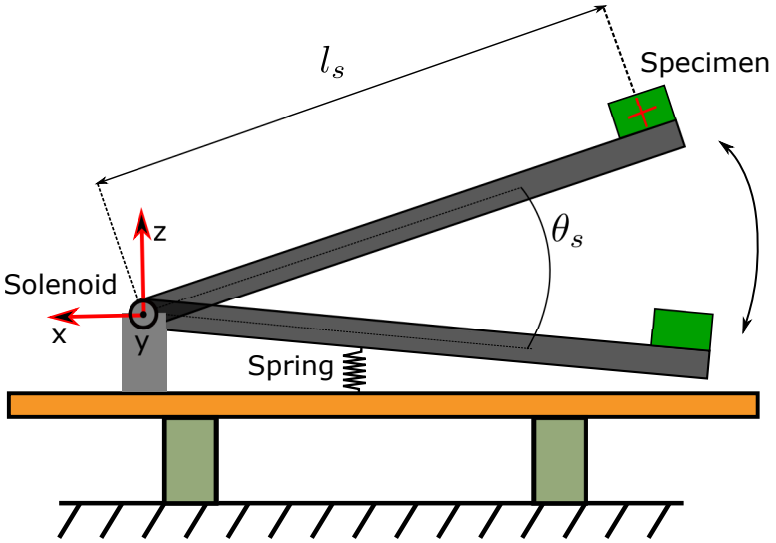


Figure 3.8 Qualitative representation of the experimental setup. The spring is made with rubber bands.

signals for the Pololu driver. The sampling frequency has been chosen to match the one used to sample both the VN100 and the 3DM-GX5-15 IMUs. The spring shown in Figure 3.8 is physically realized by means of rubber bands.

Specimen

The specimen consists of three independent IMUs provided by three different vendors; each IMU has a different communication interface. They are mounted on a custom-made 3D-printed support to allow them to have the y-axis aligned with the rotation axis of the solenoid (see Figure 3.9).

VN100

The Vectornav VN100 [121] is the IMU that can measure the highest linear acceleration (up to 16 g). It is sampled at 500 Hz, which is the maximum orientation estimate update rate configurable for the device, by the sbRIO-9637 board using the SPI communication protocol. The IMU chip is mounted on a custom-made PCB, and is the one used in Skippy and in all the balancing machines in this thesis.

3DM-GX5-15

The Lord MicroStrain 3DM-GX5-15 [122] can measure linear accelerations up to 8 g. It communicates through a USB interface with the software SensorConnect [123] provided

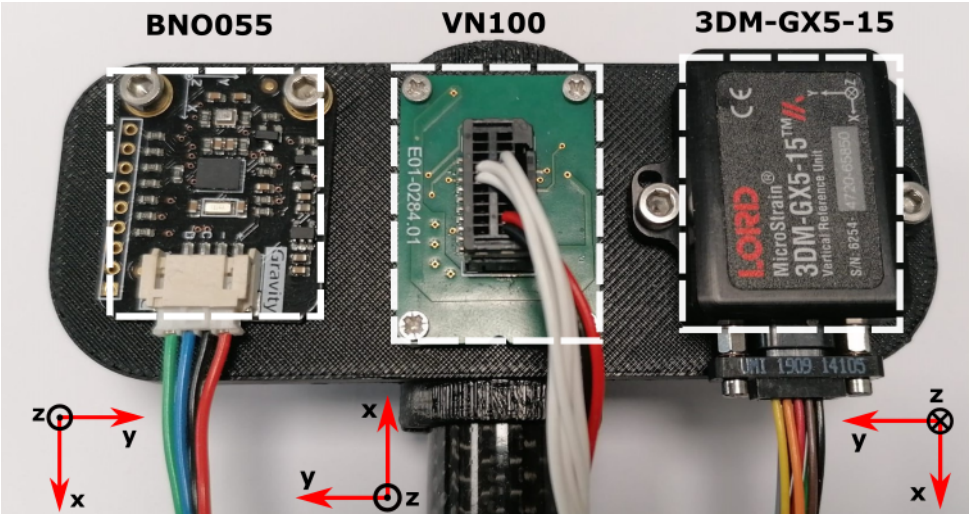


Figure 3.9 Top view of the IMUs mounted at the end of the carbon fibre rod and their reference systems. The perspective distortion in this photo makes the BNO055 and 3DM-GX5-15 appear to be at an angle when in reality, they are aligned.

by the vendor. The IMU is configured to stream data at 500 Hz continuously and is directly connected to the host PC using a USB cable. The 3DM-GX5-15 is the only tested IMU not equipped with a magnetometer.

BNO055

The Bosch BNO055 [124] is assembled on a DFRobot breakout [125]. Despite the discontinuation of the DFRobot breakout module, the BNO055 chip can be found in alternative breakout modules offered by different suppliers, such as Adafruit [126]. It is the cheapest among the tested IMUs (it can be bought for just a few tens of euros). The BNO055 provides orientation data only when configured in 'fusion mode'; such a configuration provides a 4 g linear acceleration range and an output data rate of 100 Hz. The BNO055 does not have any non-volatile memory to store the calibration data; hence every time it is powered on, it has to be calibrated [124]. The gyroscope is calibrated by holding the IMU still for some seconds, and moving the rod manually up and down proved to be sufficient to calibrate the accelerometer. The magnetometer, instead, requires a 3D change in the orientation of the device to get calibrated. Such movements are incompatible with our setup; consequently, the IMU is never fully calibrated. An Arduino UNO board [127] samples the device at 100 Hz by using the I2C communication protocol. This selection is used to compare professional and expensive acquisition systems with a hobbyist and cheap one.

Experiment Description

The experiment consists of continuously bouncing the tube against the rubber bands, which act as a spring and push the rod back up. The solenoid's torque (which is small compared to the forces exerted by gravity and the rubber bands) is actuated with a constant voltage in the same direction as the rod's motion, and it is activated only when the rod's angle, measured by the encoder, is within a specified range (Figure 3.10). The lower bound of this range is when the rod is about to touch the rubber bands, and the upper bound depends on the desired maximum acceleration at the bounce: the higher the bounce, the higher the acceleration. The system is tuned to have the bounce apex always above the upper threshold of the applied torque range. The strategy of applying a constant torque over a constant (angular) stroke implies a constant energy injection per bounce so that the apparatus will converge and settle to a steady state in which the bounce height is such that energy losses match energy inputs. The idea is to have a smooth transition between the stance and flight phases, which are the two components of the continuous bouncing behaviour of a hopping robot.

The experiment starts by turning on the apparatus, and then a sufficient amount of time is allowed to let all the sensors power on correctly. At this point, the rod is manually moved up and down until the BNO055 is partially calibrated. The following phase starts by logging the sensors with the rod at rest position for about one minute. The data from all the IMUs are recorded. Then, the rod is manually pushed downwards against the rubber bands to start the bouncing motion (Figures 3.12, 3.18 and 3.24), which continues for about three minutes. Finally, the solenoid is switched off so that the bouncing ceases, and the system continues logging the sensors for another minute with the rod remaining still at its rest position (Figures 3.13, 3.19 and 3.25).

The data collected from the IMUs are the compensated linear accelerations, the compensated angular velocities and the quaternions. The measured quaternions are then converted offline into Euler angles allowing the rotation around the y-axis of the IMUs to be compared with the angular position of the tube. The experiment aims to discover the magnitude of drift in the absolute orientation of the IMUs while continuously bouncing, the time it takes for this drift to emerge, and the time required to recover afterwards.

The experiment consists of three parts with increasing linear acceleration values. For each acceleration value, the investigation is performed three times. Each IMU is tested at an acceleration close to, but strictly below, its accelerometer's saturation limit and the collected data is checked at the end of the experiment to ensure that the accelerometer never saturated. Any trial in which the accelerometer reaches its saturation limit is rejected and replaced with a new one.

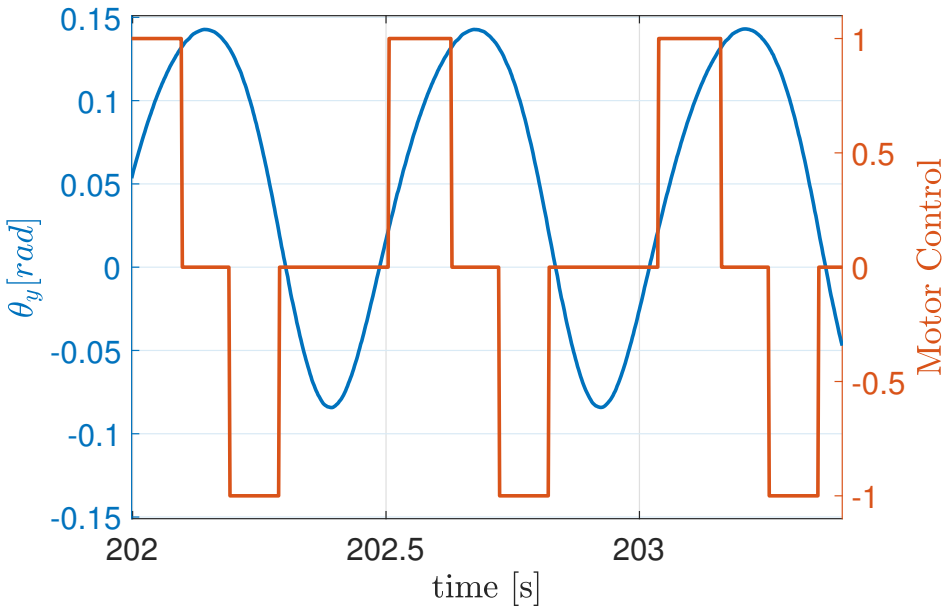


Figure 3.10 Motor control signal with respect to the angular position of the tube during the continuous bouncing of the 4 g experiment. The torque is applied only in the direction of motion and when the tube is above the rubber bands and below the bounce apex. The zero line represents the rest position of the rod. Motor control goes from -1 (100% duty cycle and negative voltage) to $+1$ (100% duty cycle and positive voltage), allowing a bidirectional control of the solenoid.

Since the three IMUs have the z-axis vertical, the experiments are classified based on the maximum acceleration on the z-axis (4 g, 8 g and 16 g). 4 g and 8 g accelerations are comparable to those experienced by the human ankle during running at the moment when the foot touches the ground (Table I of [128] and Figure 7 of [129]).

Table 3.4 reports the accuracy reported by the datasheet for each IMU and for which experiments they have been tested. The different bouncing acceleration values are obtained by changing the rubber bands' distance from the axis of rotation of the tube or increasing the actuated range of motion of the solenoid, or varying the number of rubber bands.

3.2.2 Data Acquisition System

Each IMU has a different communication interface, and therefore it requires its own independent data acquisition system; the three processes are synchronized with a dedicated software interface running on the host PC.

IMU	Roll / Pitch		Yaw		Experiments
	deg	rad	deg	rad	
BNO055	-	-	-	-	4 g
3DM-GX5-15	0.25	0.0044	-	-	4 g, 8 g
VN100	1.00	0.0175	2.0	0.035	4 g, 8 g, 16 g

Table 3.4 IMUs accuracy as stated in the manufacturer's datasheets. The last column reports the experiments in which each IMU was tested. Short dashes denote that data are not provided by the manufacturer.

A different delay in the received data from each IMU is observed due to multiple interfaces and acquisition systems. As a consequence, all the recorded data are synchronized offline once the experiment is completed before analyzing the results.

National Instrument sbRIO-9637

The National Instrument sbRIO-9637 board [120] has analog and digital inputs / outputs, a dual-core CPU and a programmable FPGA. The board is configured based on a Supervisory Control and Data Acquisition (SCADA) architecture, where the FPGA oversees performing the SPI communication and controlling the solenoid. The sbRIO-9637 is used to sample the position encoder and the VN100 using two different SPI channels, both of them at 500 Hz. The board's CPU uses these measurements to control the solenoid and sends them to the host using network streams. Finally, the host runs a software interface to monitor online the acquired data.

Arduino UNO

The Arduino UNO board [127] is the acquisition system for the Bosch BNO055. The board communicates with the IMU with an I2C interface; it samples the data at 100 Hz and then it sends them to the PC through a serial interface.

Host Computer

The data acquisition system for the Lord MicroStrain 3DM-GX5-15 [122] is the software SensorConnect [123] provided by the vendor. The sensor is sampled at 500 Hz.

A LabVIEW Virtual Instrument (VI) activates the three independent data acquisition systems simultaneously. When the start button is pressed in the host's VI, the sbRIO-9637 FPGA starts logging the VN100 and sets to high a digital output connected to the Arduino Uno to start sampling. The signal returns to low when the user interrupts the logging at the

host's VI. The host's VI also provides the logging timestamp used in the SensorConnect software to save the captured data of the appropriate time interval.

3.2.3 Results

The main objective of this work is to investigate how the estimation of the absolute orientation is affected by continuous low-intensity impacts, simulating the shocks that a running/hopping robot may experience. The experimental apparatus is designed to have the y-axis of the tested IMUs aligned with the rotation axis of the solenoid, whose angle is directly measured with the absolute position encoder. The physical imperfections of the system mean that each IMU's y-axis is not perfectly aligned with that of the solenoid. To compensate for any possible misalignment, the conversion of the quaternions into Euler angles is performed in a coordinate system that is as close as possible to the IMU's internal coordinate system but has its y-axis accurately aligned with that of the solenoid. The transformation from internal to aligned coordinate system is obtained separately for each IMU using its motion data from the first few bounces. The measured angular position of the encoder, θ_{y_e}, and the corresponding angle measured by the IMUs, θ_{y_I}, are referred to as θ_y when mentioned together. To have comparable measurements, the average of the measured values with the rod in its rest position is defined to be the reference orientation for the encoder angle and each IMU, and both the encoder angle and the IMU orientation estimates are measured relative to this orientation.

One of the outcomes of this experiment is the difference between the angular position measured by the encoder and the angular rotation about the y-axis estimated by the IMUs. This value, E_{Pitch}, will be referred to as orientation error or simply error. During the experiments, the rod presents a minor bending while it presses into the rubber bands, see Figure 3.8. This bending behaviour causes a slightly bigger rotation at the IMUs compared with encoder measurement (see the responses of the IMUs in Figures 3.12, 3.18 and 3.24). The error in the two remaining rotation axes (referred to as E_{Roll} about the x-axis and E_{Yaw} about the z-axis) is the difference between the average value of the orientation during the initial rest period and current orientations since the sensors are not expected to move about those axes during the experiment. The continuous bouncing of the specimen causes a periodic component (at around 1-2 Hz) in E_{Roll}, E_{Pitch} and E_{Yaw}. This component has been removed from the error graphs by a 6^{th} order zero-phase low pass filter consisting of a Butterworth filter with a cut-off frequency of 0.2 Hz.

The results of the three experiments are discussed in the following paragraphs. The solid horizontal lines (where present) in Figures 3.14, 3.15, 3.16, 3.20, 3.21, 3.22,3.26,

Figure 3.11 The picture explains the meanings of the different line styles used in the error graphs for E_{Roll}, E_{Pitch} and E_{Yaw}. The legends in those graphs identify each IMU with a different colour. The accuracy line, when present, shows the datasheet figure for maximum error so that it can easily be seen whether or not the actual error exceeds the datasheet figure.

3.27, and 3.28 indicate the accuracy of the IMU as stated in the datasheets provided by the manufacturers. Each experiment is performed three times because having three sets of data allows one to get an approximate idea of how much of the measured signal is random and how much is a repeatable effect caused (we assume) by the disturbing effect of the bouncing motion on each IMU. Table 3.5 reports the maximum and the average peak acceleration experienced by each IMU to show that none of the IMUs saturated during the experiment. The average linear velocities of the IMUs at landing during the 4g, 8g, and 16g experiments are 1.9m/s, 3.5m/s and 5.8m/s, respectively. These correspond to drop heights of 0.18m, 0.62m and 1.71m, respectively. However, the actual drop heights of the IMUs in the bounce tester are smaller because the IMUs follow a circular arc and are accelerated by the solenoid as well as gravity. These figures suggest that a legged robot would not experience accelerations as high as 16g during normal locomotion. However, it should be understood that the rubber bands give the rod a soft landing and a stiffer landing would produce a larger acceleration for a given velocity. Such accelerations can arise if the robot is mechanically stiff and/or pounds the ground with its feet. This type of motion causes acceleration spikes when the (relatively hard) foot strikes the (hard) ground due to mechanical shock propagating up the leg into the torso, but this effect is absent in the bounce test experiments.

For each experiment, the behaviour of the Roll, Pitch and Yaw angles is analyzed separately. Figure 3.11 describes all the lines in the error graphs.

Experiment 1: 4 g Maximum Linear Acceleration

In this experiment, the maximum value of linear acceleration along the z-axis is approximately 4 g and is close to but strictly below the saturation limit of the BNO055 (Table 3.5). All three IMUs have been tested under the same conditions. Figure 3.12 clearly shows how the three IMUs signals are all aligned with the encoder at the beginning of the experiment. Figure 3.13 displays the drift that both the VN100 and the BNO055 have accumulated during the motion, with the latter being higher in magnitude than the former. Once the actuation stops, it takes around 5-6 seconds for the apparatus to stop. In this time interval, the VN100 recovers from the drift and converges to the same value measured by the absolute encoder. The BNO055,

Experiment	4 g					
IMU	First Trial		Second Trial		Third Trial	
VN100	39.42	38.76	39.02	38.68	38.81	38.13
3DM-GX5-15	39.47	38.79	39.23	38.83	39.10	38.21
BNO055	38.37	37.88	38.24	37.65	37.98	37.27

Experiment	8 g					
IMU	First Trial		Second Trial		Third Trial	
VN100	76.64	73.83	75.54	74.07	76.56	75.41
3DM-GX5-15	77.71	75.96	76.78	74.87	77.11	75.86
BNO055	-	-	-	-	-	-

Experiment	16 g					
IMU	First Trial		Second Trial		Third Trial	
VN100	153.84	153.80	153.86	153.80	153.85	152.95
3DM-GX5-15	-	-	-	-	-	-
BNO055	-	-	-	-	-	-

Table 3.5 Every experiment is performed three times. For each experiment trial, the table reports the highest recorded value of the linear acceleration on the z-axis (left column) and the average value of the peak acceleration on each bounce (right column) on the z-axis. The maximum linear acceleration is reached when the specimen reaches the bottom of the bounce and is about to be pushed back by the rubber bands. The saturation values for the three IMUs are measured experimentally and are 155.84 for the VN100, 79.66 for the 3DM-GX5-15 and 39.22 for the BNO055. All the acceleration values are expressed in $[m/s^2]$.

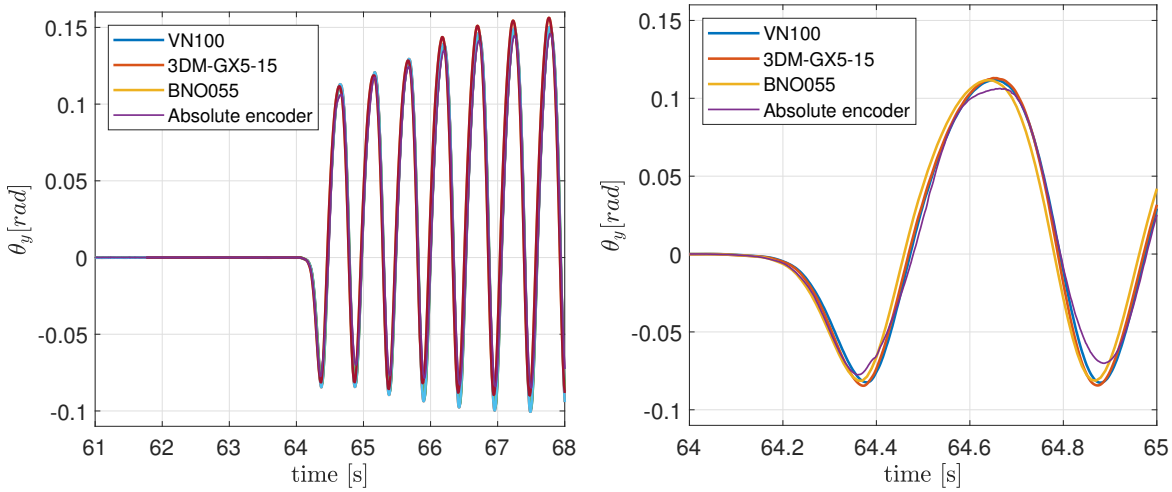


Figure 3.12 Synchronized measurements of pitch obtained from the encoder and the IMUs at the beginning of the 4 g experiment.

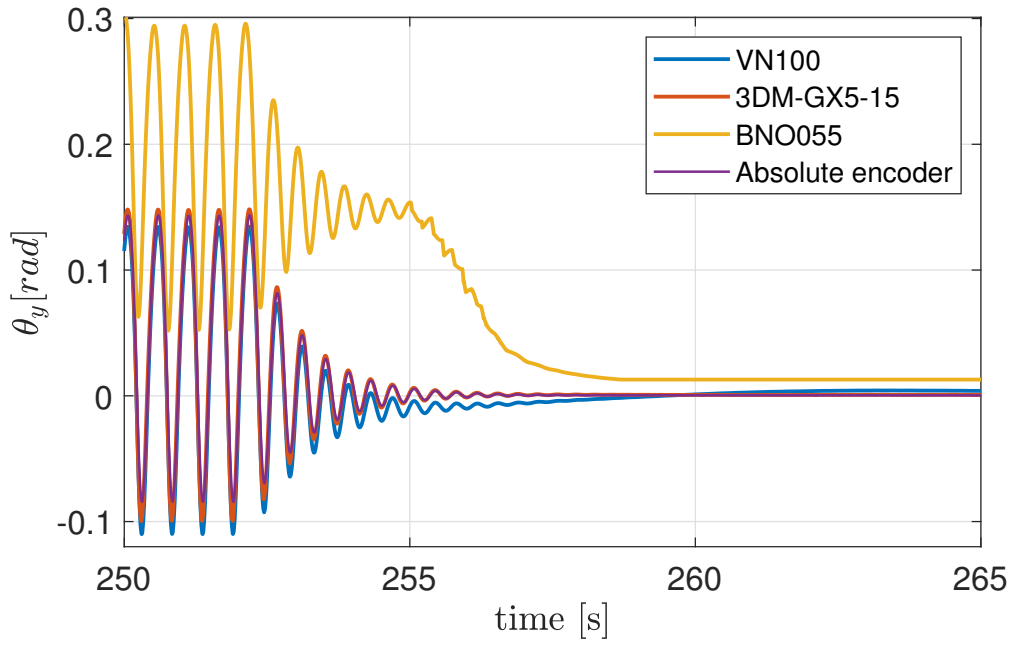


Figure 3.13 Synchronized measurements of pitch obtained from the encoder and the IMUs at the end of the 4 g experiment.

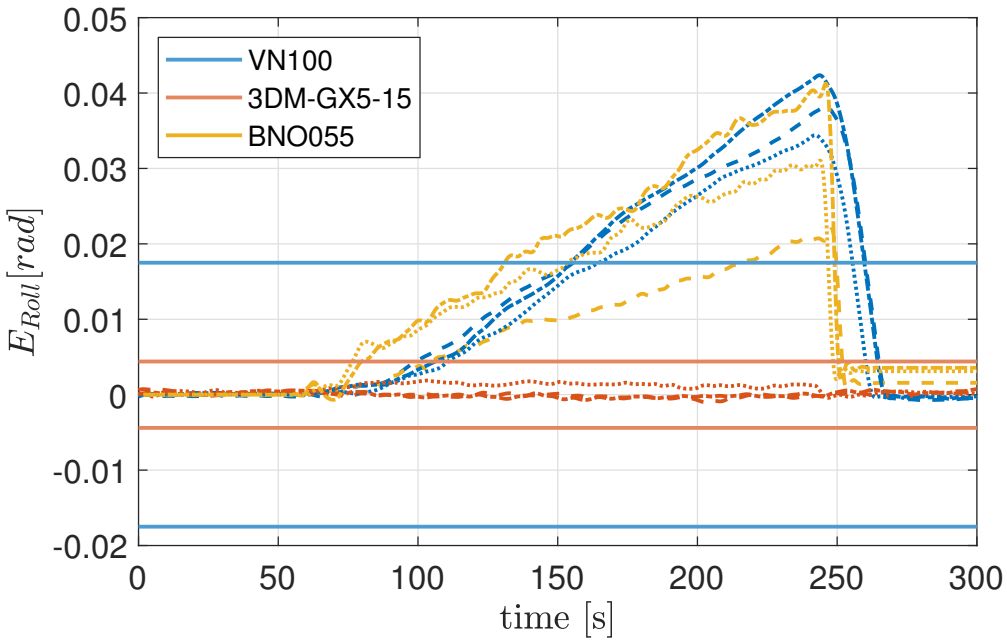


Figure 3.14 Filtered orientation error on x axis during the 4 g experiment. See Figure 3.11 for line interpretation.

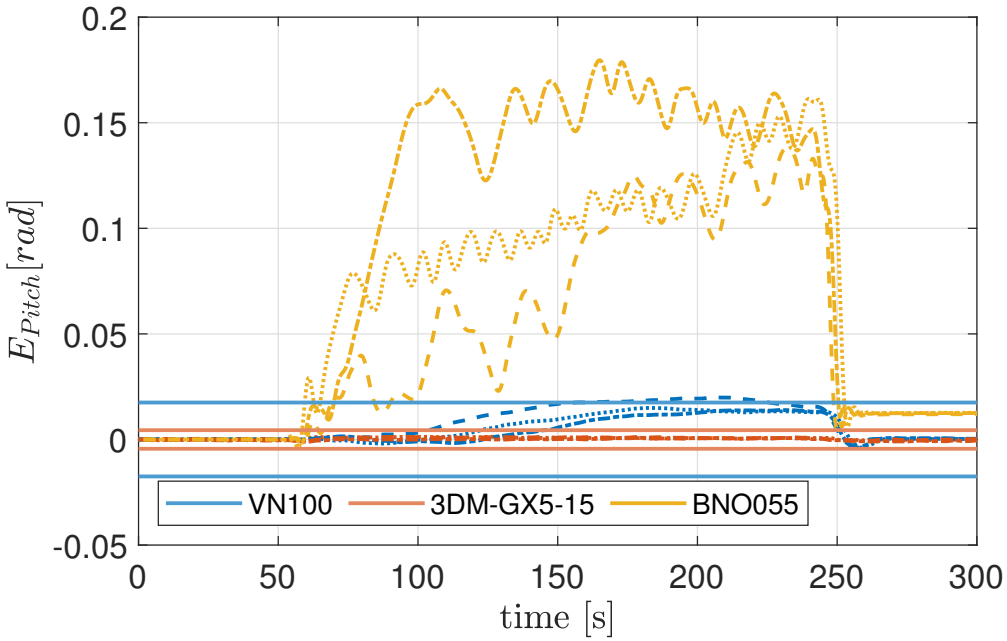


Figure 3.15 Filtered orientation error on y axis during the 4 g experiment. See Figure 3.11 for lines interpretation.

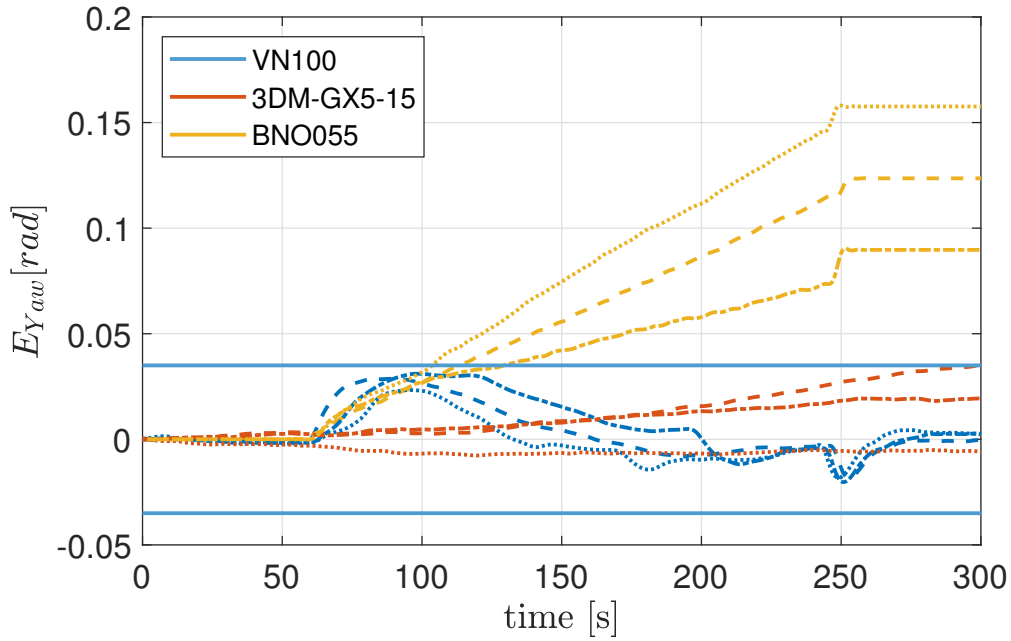


Figure 3.16 Filtered orientation error on z axis during 4 g experiment. See Figure 3.11 for lines interpretation.

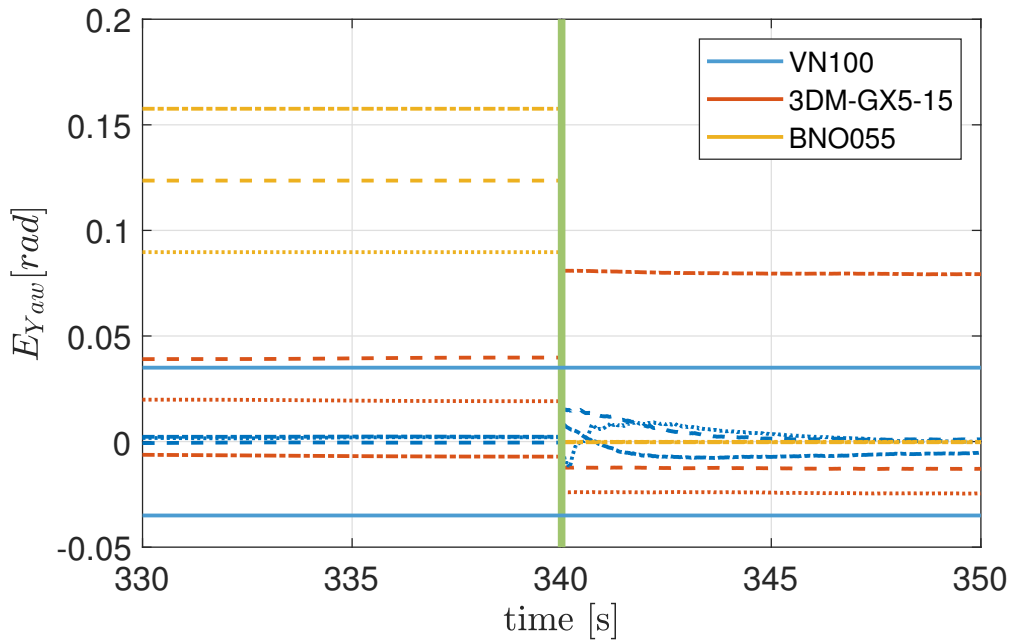


Figure 3.17 Filtered orientation error on z axis after the 4 g experiment. The green vertical line indicates when the system is temporarily powered off. See Figure 3.11 for lines interpretation.

instead, is still affected by the accumulated drift for 3 seconds after the actuation ceases. Once the bouncing motion amplitude is significantly reduced (peak-to-peak smaller than 0.01 rad), also the BNO055 partially recovers from the drift in 3 seconds, leaving a residual drift. The 3DM-GX5-15 instead proved to be consistent in the estimation of the Pitch angle throughout the experiment.

Roll (Figure 3.14):
Surprisingly, the BNO055 performs better than the VN100 in the estimation of the Roll angle. The latter shows some non-negligible drift effects in the estimation of the orientation, eventually exceeding the datasheet maximum error by more than a factor of 2. It takes the VN100 around 80 seconds to exceed the datasheet error (i.e. go out of spec) after the motion starts and 20 seconds to return inside the interval once the bouncing stops. The 3DM-GX5-15 proved to be robust to this kind of motion.

Pitch (Figure 3.15):
The VN100 shows a 0.0025 rad drift in the Pitch angle measurements, but only in one of the three trials, E_{Pitch} slightly exceeds the accuracy range. The BNO055 absolute orientation estimation drifts significantly, making its values unreliable and impossible to be used in these circumstances. Also, in this case, the 3DM-GX5-15 gives reliable results, never going out of spec.

Yaw (Figure 3.16):
The VN100 accurately measures the Yaw angle. As a matter of fact, the E_{Yaw} is never outside the accuracy interval. The 3DM-GX5-15 and the BNO055 show a drift in the Yaw angle estimation. It can be observed that in both cases, the E_{Yaw} increases during the bouncing, and it never returns to zero, even when the motion ceases. This behaviour, I think, is because these two IMUs cannot rely on the compensation of the magnetometer; the 3DM-GX5-15 does not have it, while the BNO055 cannot be calibrated due to the experimental set-up.

Power Cycle (Figure 3.17):
All of the three IMUs have been powered off and on (power cycle) after every trial to see the effects on the estimation of the Yaw angle. The VN100 is still within the accuracy range as expected. The 3DM-GX5-15 starts evaluating the Yaw angle again with an offset different from the initial one due to the lack of magnetometer data. The E_{Yaw} for this IMU never

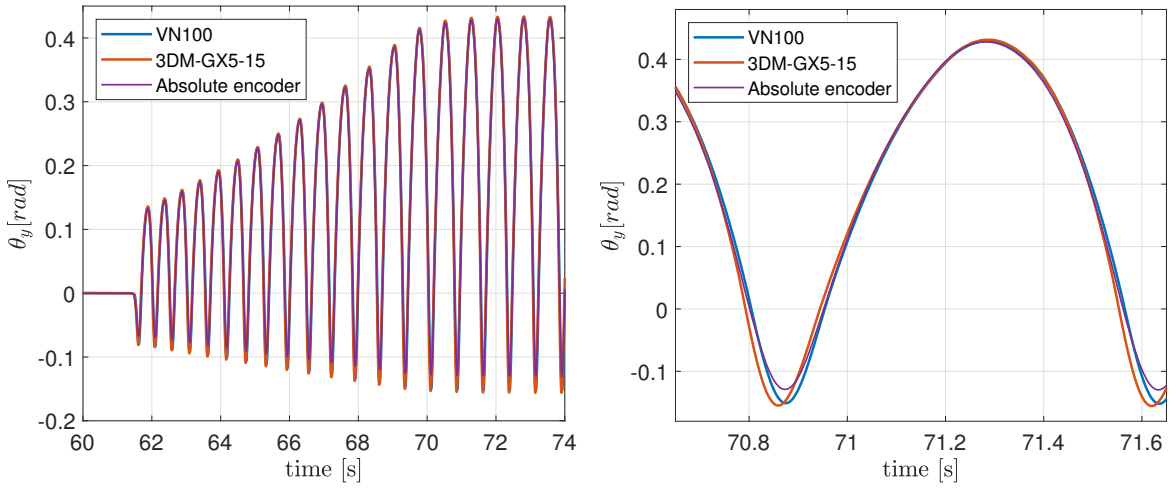


Figure 3.18 Synchronized measurements of pitch obtained from the encoder and the IMUs at the beginning of the 8 g experiment and at steady state.

returns to the initial value. The BNO055, instead, restarts from the initial value, showing an almost null error.

Experiment 2: 8 g Maximum Linear Acceleration

This experiment intends to test the VN100 together with the 3DM-GX5-15 with a linear acceleration as close as possible to 8 g, which is the acceleration saturation limit of the 3DM-GX5-15 (Table 3.5). Also, in this case, the two IMUs have the y-axis aligned with the solenoid, and the signals are synchronized, see Figure 3.18. Like in the previous experiment, the 3DM-GX5-15 proved to be a better estimator of θ_y than the VN100, which drifts significantly during the motion, Figure 3.19. Once the motion stops, it takes more than 25 s for the VN100 to eliminate the drift disturbance and give the correct angle measurement.

Roll (Figure 3.20) and Pitch (Figure 3.21):
The VN100 shows evident drift in estimating the Roll and the Pitch angles. It takes approximately 40 seconds for the E_{Roll} to exceed the accuracy limit and 55 seconds for the E_{Pitch}. These errors return inside the accuracy interval around 25 seconds after the motion ceases. The VN100 appears to have a constant recovery rate, with a recovery time depending on the error size. In both cases, the 3DM-GX5-15 shows a negligible error.

Yaw (Figure 3.22):
Interesting results are shown in the Yaw angle estimation, where the VN100 shows a significant and persistent drift. E_{Yaw} crosses the accuracy threshold after 40 seconds and persists

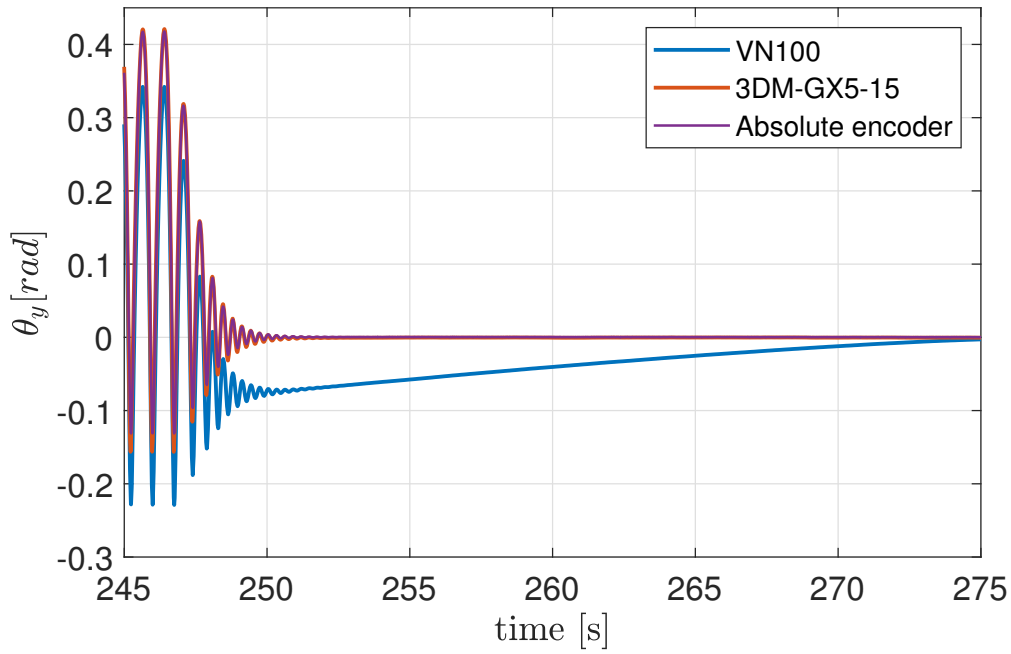


Figure 3.19 Synchronized measurements of pitch obtained from the encoder and the IMUs at the end of the 8 g experiment.

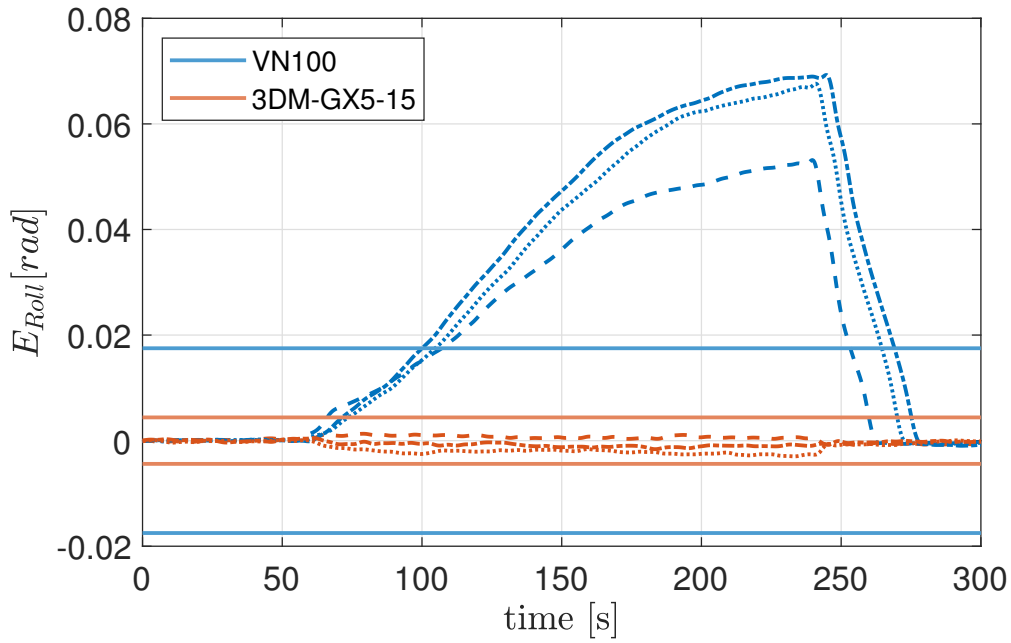


Figure 3.20 Filtered orientation error on x axis during the 8 g experiment. See Figure 3.11 for lines interpretation.

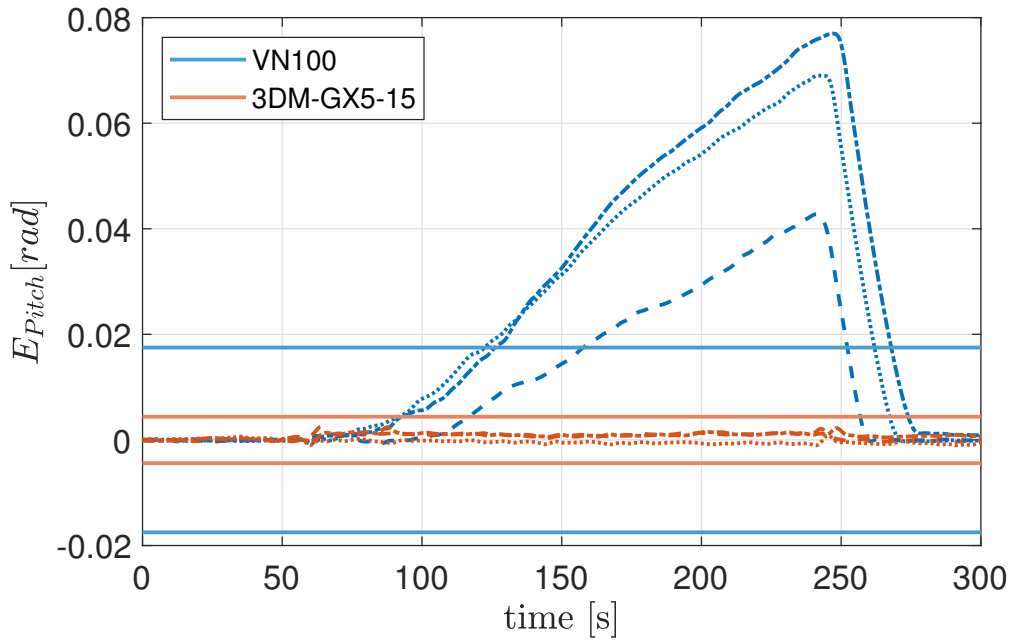


Figure 3.21 Filtered orientation error on y axis during the 8 g experiment. See Figure 3.11 for lines interpretation.

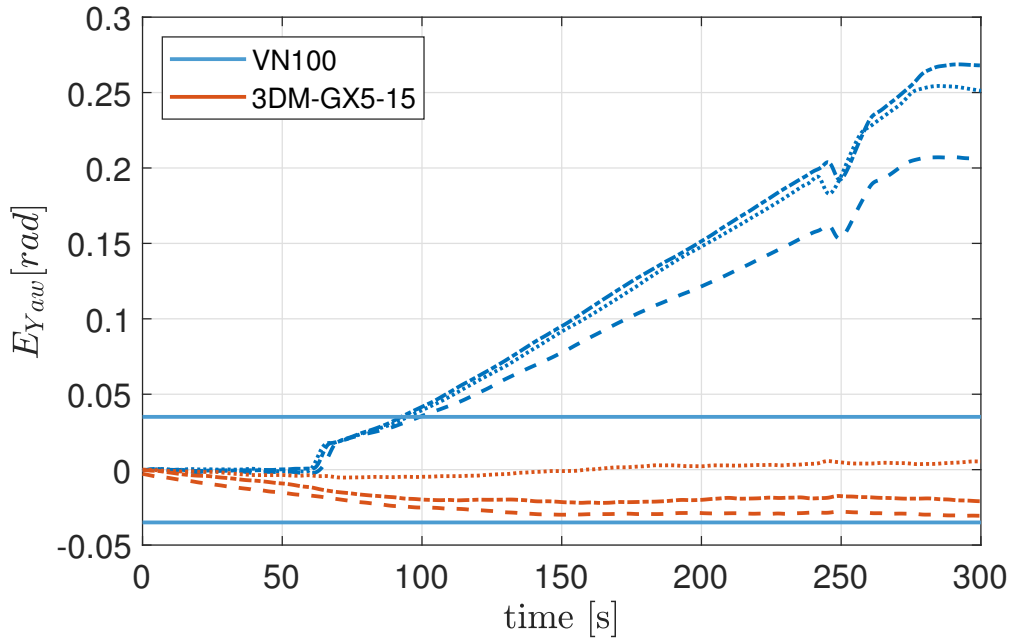


Figure 3.22 Filtered orientation error on z axis during the 8 g experiment. See Figure 3.11 for lines interpretation.

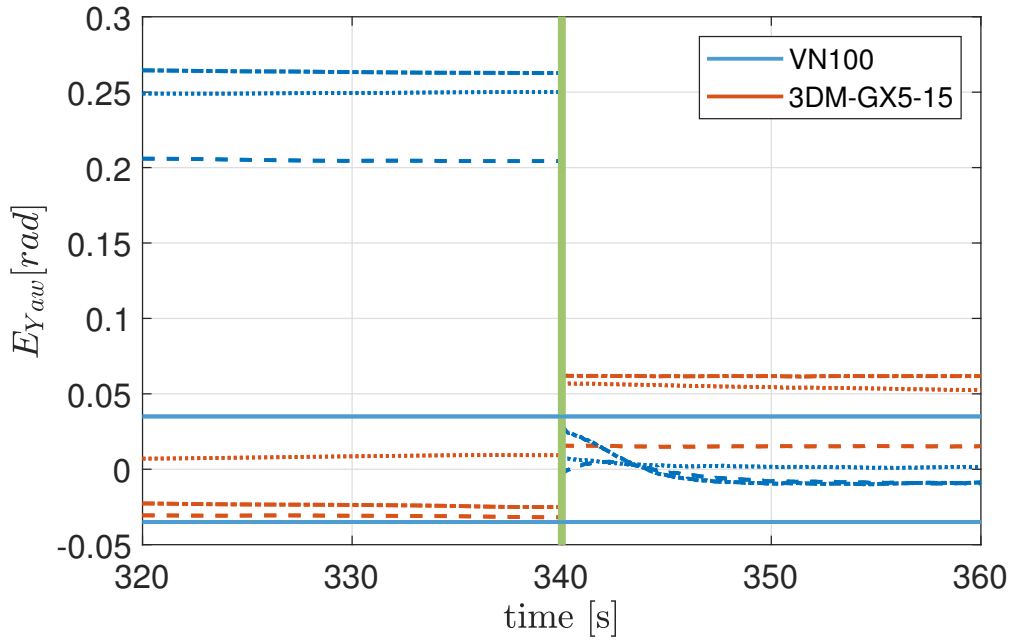


Figure 3.23 Filtered orientation error on z axis after the 8 g experiment. The green vertical line indicates when the system is temporarily powered off. See Figure 3.11 for lines interpretation.

after the bouncing stops, with a magnitude of approximately 20 degrees. On the other hand, the 3DM-GX5-15 shows a small drift in magnitude but with sign not constant through the trials (it is positive in one trial and negative in the other two).

Power Cycle (Figure 3.23):
The Yaw error of the VN100 does not return inside the accuracy interval after the bouncing stops. It requires a power cycle to make E_{Yaw} return between the boundaries of such interval. The power cycle forces the 3DM-GX5-15 to restart estimating the Yaw angle from a value which is different both from the initial one and the one before the power cycle.

Experiment 3: 16 g Maximum Linear Acceleration

The last experiment analyzes the behaviour of the VN100 near its acceleration saturation limit at 16 g (Table 3.5). Figure 3.24 shows that the IMU's measurement is aligned with the encoder's one at the beginning of the motion, while Figure 3.25 clearly shows that the two signals are not aligned anymore when the motion stops.

Roll (Figure 3.26) and Pitch (Figure 3.27):
The drift on the Roll and Pitch angles is significant and not negligible, but it does not affect

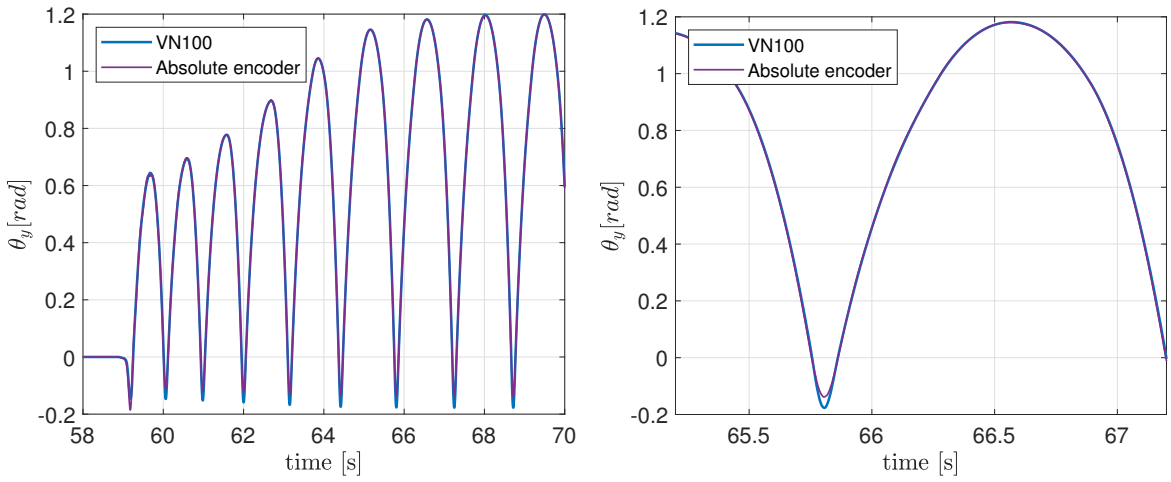


Figure 3.24 Synchronized measurements of pitch obtained from the encoder and the IMUs at the beginning of the 16 g experiment and at steady state.

the absolute estimation once the bouncing finishes; both E_{Roll} and E_{Pitch} return inside the accuracy interval after 25 seconds. E_{Roll} crosses the accuracy threshold after 20 seconds from the start of the motion, while E_{Pitch} requires 70 to 130 seconds to reach the limit, depending on the trial.

Yaw (Figure 3.28):
On the other hand, the drift on the estimation of the Yaw angle is significant; it exceeds the threshold after 20-40 seconds (depending on the trial) and persists once the IMU returns to the initial rest position. E_{Yaw} varies from 5 to 10 degrees, depending on the trial.

Power Cycle (Figure 3.29):
Like in the 8 g experiment, a power cycle proved to be necessary to bring the E_{Yaw} back inside the accuracy interval.

3.2.4 Discussion

The IMU that displayed the worst performance in the 4 g experiment is the BNO055, which was the cheapest out of the three IMUs. The drift in the estimation of the Yaw angle was more than 4 degrees and the BNO055 was the only IMU that could not recover the initial orientation after the bouncing motion had stopped. A simple power cycle proved to be partially effective: it brings back E_{Yaw} to its original value but it leaves the IMU not calibrated.

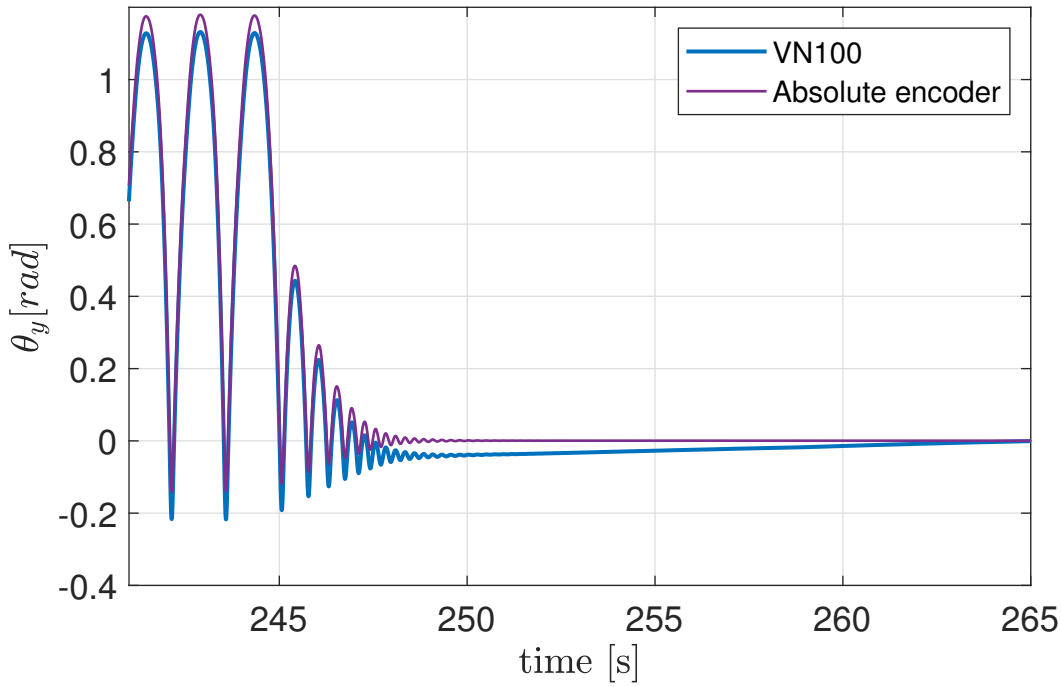


Figure 3.25 Synchronized measurements of pitch obtained from the encoder and the IMU at the end of the 16 g experiment.

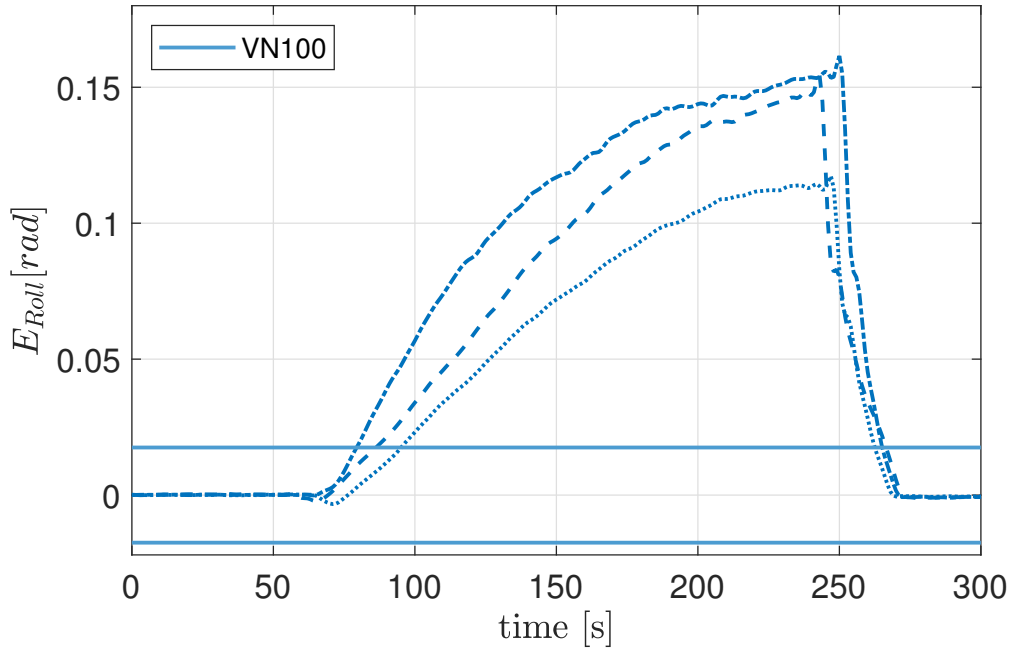


Figure 3.26 Filtered orientation error on x axis during 16 g experiment. See Figure 3.11 for lines interpretation.

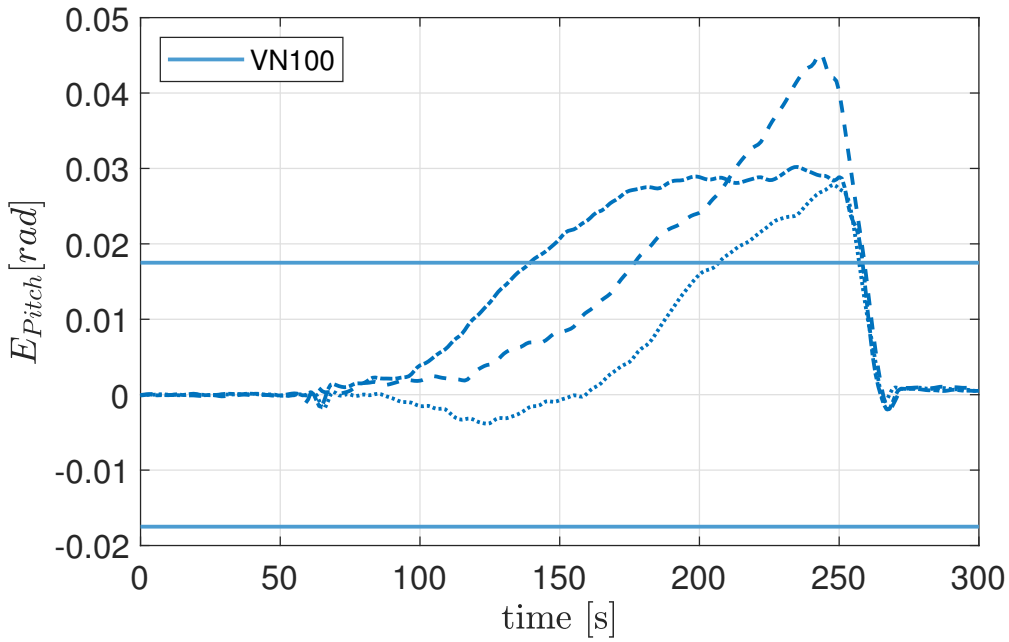


Figure 3.27 Filtered orientation error on y axis during 16 g experiment. See Figure 3.11 for lines interpretation.

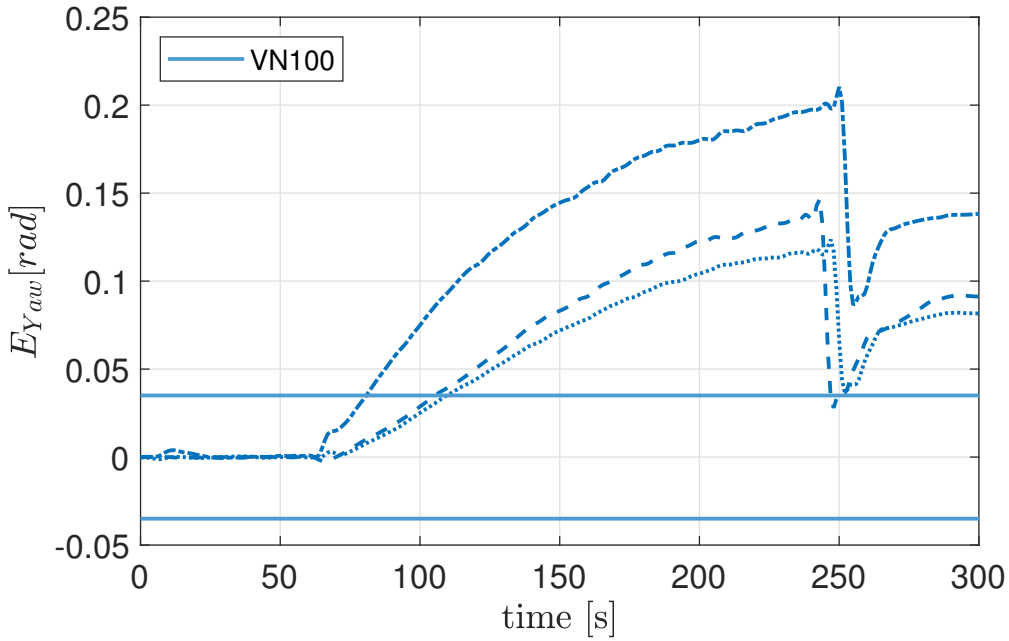


Figure 3.28 Filtered orientation error on z axis during 16 g experiment. See Figure 3.11 for lines interpretation.

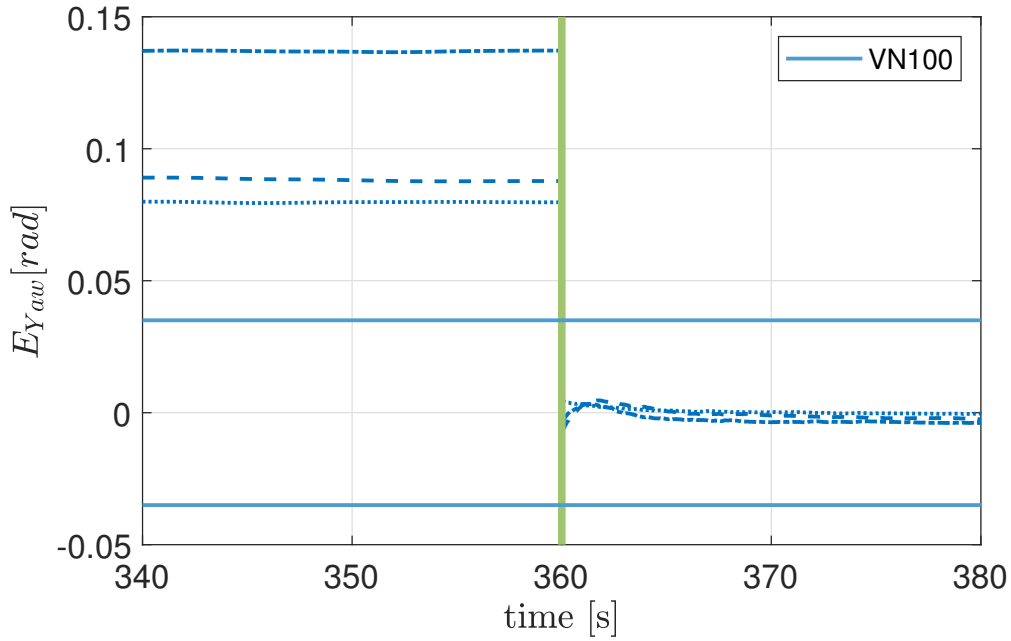


Figure 3.29 Filtered orientation error on z axis after 16 g experiment. The green vertical line indicates when the system is temporarily powered off. See Figure 3.11 for lines interpretation.

The 3DM-GX5-15 achieved the best performance in all the conditions where it was tested. It is the only evaluated IMU that does not have a drift in estimating the Roll and Pitch angles both in the 4 g and 8 g experiments. Instead, the drift on the Yaw angle is due to the absence of the magnetometer and cannot be compensated. In this case, a power cycle proved to be ineffective.

The VN100 is the only IMU tested in all three experiments. It presents a drift in estimating the Roll angle in every experiment, and in the 4 g experiment, it behaves worse than the BNO055. In all the experiments, both E_{Roll} and E_{Pitch} return inside the accuracy range once the motion stops. During the 4 g experiment the IMU does not show any drift in the Yaw angle estimation. Such behaviour is not present in the 8 g and 16 g experiments, where a power cycle turned out to be necessary to bring back the E_{Yaw} into the accuracy range.

3.2.5 Conclusions

This work is intended to give the reader a better understanding of the behaviour of MEMS IMUs in an environment for which they have not been specifically designed. The experiments reported in this section investigated the orientation estimation performance of a selection of IMUs, subject to continuous low-intensity impacts, which aim to reproduce impacts

experienced during hopping or running. According to the results, all three IMUs suffered a certain amount of drift during the experiments.

The continuous bouncing of the specimen resembles the motion of a bipedal or quadruped robot running. Therefore using these MEMS IMUs as a balancing reference point may lead to unexpected results, making it difficult to control the robot. The evident drift in the absolute orientation estimation cannot be left out in the design process of the control system, which has to be robust enough either to withstand or to compensate for it. Alternatively, having this knowledge, the simplest thing to do would be for the robot to rest until the drift gets eliminated, similar to humans that stop when they are disoriented.

In this work, a drift in the IMU's data means an increasing error in the estimation of the vertical, and practical consequences that could be experienced differ from the type of robot considered. In the case of a balancing machine, such as in Chapters 4, 5 and 6, the robot would start oscillating while trying to find a balanced configuration, and eventually, it will become impossible to balance. On the other hand, for a hopping robot like in Chapter 7, a wrong estimation of the vertical means that the robot would jump with an angle that is not the desired one, causing an undesired behaviour and possibly making it impossible to recover from such a hop.

3.3 General Comments and Limitations

In the case of such an athletic and dynamic robot like Skippy, whose performance is deliberately trying to reach the limit of the hardware capabilities, it is essential to know the behaviour of the sensors in such conditions. Thanks to the tests performed in this chapter, the control system can be aware of the sensors' limits and how to deal with them to achieve the desired performance. Furthermore, a more robust robot design can be achieved, ensuring the correct sensors' behaviour in all the desired working conditions. As a consequence, a simple methodology for testing encoders and IMUs is presented.

Both experiments can be improved to remove (or at least mitigate) some of the design flaws. The results obtained in Section 3.1 can be improved by designing a stiffer catapult arm, reducing the motor-shaft misalignment, building a more solid platform for the foam and putting the encoder at the centre of percussion of the catapult. The new testing apparatuses will reduce the undesired vibrations and the mechanical shock experienced by the encoder, allowing for more precise measurements. The work of Section 3.2 can be further developed with more and newer IMUs, which allow for a full calibration before the experiment; increasing the number of trials will allow a statistical analysis of the performance.

Chapter 4

Reaction Wheel Pendulum

This chapter presents the balance controller for an inverted reaction wheel pendulum robot. The chapter is organized as follows. First, a brief introduction of the general balance control theory, then a simplification for the case of a reaction wheel pendulum, then the robot used for the experiments is described, followed by the experimental results.

4.1 General Balance Control Theory

This section presents a brief outline, and only for the case of balancing in a plane, of the balancing theory proposed by Featherstone and described in detail in [18; 130]. The theory has then also been extended to control the absolute motion of a 2D triple pendulum in [131]. The basic idea is to close a control loop around the plant shown in Figure 4.1, which describes the exact dynamics of balancing for any planar robot, having any number of bodies and joints. The robot is balancing on a stationary point that can be modelled as a passive revolute joint, and a single actuated joint q_{a} responsible for balancing. Such a robot can be modelled as the inverted double pendulum of Figure 4.2. Since the robot is balancing on a passive revolute joint, the general equation of motion

$$H\ddot{q} + C = \tau \tag{4.1}$$

can be written as

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix} \tag{4.2}$$

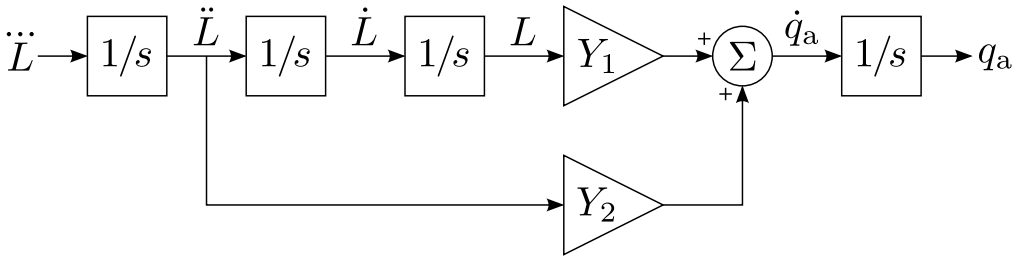


Figure 4.1 Plant model of the balance dynamics for any planar robot [5].

where H_{ij} are the elements of the joint-space inertia matrix and C_i are the elements of the bias vector containing Coriolis, centrifugal and gravitational terms, $\ddot{q}_i$ are the joint accelerations, and τ_2 is the torque at the only actuated joint.

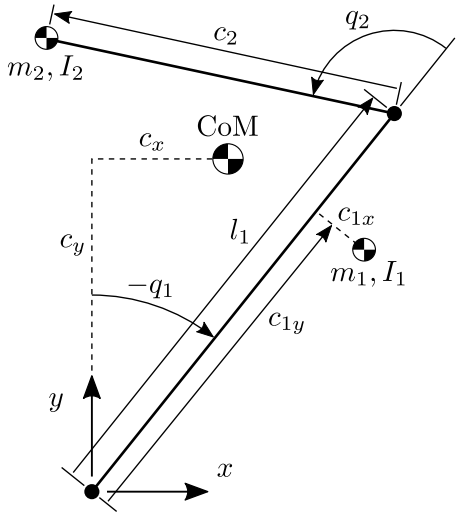


Figure 4.2 Dynamic model of the planar double pendulum [5].

The first step consists in defining all the quantities in Figure 4.1. Let us define L to be the angular momentum of the robot in Figure 4.2 about the support point. From classical mechanics, we have that the derivative of the angular momentum about the support point and its first two time derivatives are

$$\dot{L} = -mgc_x \tag{4.3}$$

$$\ddot{L} = -mg\dot{c}_x \tag{4.4}$$

$$\dddot{L} = -mg\ddot{c}_x \tag{4.5}$$

where m is the mass of the robot and g is the magnitude of the gravitational acceleration; c_x, $\dot{c}_x$ and $\ddot{c}_x$ are the position, velocity, and acceleration in the x direction of the centre of mass. Following a special property of joint-space momentum that is proved in Appendix B of [18],

the angular momentum of the robot about joint 1 is

$$L = p_1 = H_{11}\dot{q}_1 + H_{12}\dot{q}_2 \tag{4.6}$$

where p_i is the generalized momentum variable corresponding to velocity variable $\dot{q}_i$. The vector p equals the product $H\dot{q}$, so

$$p_i = \sum_{j=1}^{i} H_{ij}\dot{q}_j \ . \tag{4.7}$$

Both L and $\ddot{L}$ depend linearly on the robot's velocity, implying that $L = \ddot{L} = 0$ is equivalent to $\dot{q}_1 = \dot{q}_2 = 0$ since they are linearly independent. $\dot{L}$ is a constant multiple of c_x. As shown in [18] and [132], any controller that makes $L = \dot{L} = \ddot{L} = 0$ will make the robot balance but will not drive the actuated joint in the desired position. Let us add an extra fictitious prismatic joint acting in the x direction between the joint 1 of the robot and the ground. The extra joint is called joint 0 to preserve the numbering of the existing joints. The extra joint does not move, and its purpose is to increase the number of coefficients in the equation of motion so that standard dynamics functions can be used to calculate the quantities needed by the balance controller. The equation of motion now reads

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ \tau_2 \end{bmatrix} . \tag{4.8}$$

The values of position and velocity for joint 0 are set to zero, while τ_0 takes the necessary value to ensure that $\ddot{q}_0 = 0$ always, which implies that $\tau_0 = m\ddot{c}_x$ is the x component of the ground reaction force acting on the robot. The special property of the joint-space momentum used before to define $p_1 = L$ can be used in this case to define p_0 as the linear momentum of the whole robot in the x direction, so

$$p_0 = m\dot{c}_x = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 \ . \tag{4.9}$$

Combining Equations 4.4 and 4.9 we get

$$\ddot{L} = -g(H_{01}\dot{q}_1 + H_{02}\dot{q}_2) \tag{4.10}$$

and combining Equations 4.5 and 4.8 we get

$$-\dddot{L}/g = H_{01}\ddot{q}_1 + H_{02}\ddot{q}_2 + C_0 \,. \tag{4.11}$$

We now have an equation relating $\dddot{L}$ to the two independent joint accelerations and a pair of linear equations relating L and $\ddot{L}$ to the two independent joint velocities

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{4.12}$$

Solving this equation for $\dot{q}_2$ gives

$$\dot{q}_2 = Y_1 L + Y_2 \ddot{L} \tag{4.13}$$

where

$$Y_1 = \frac{H_{01}}{D} \quad , \quad Y_2 = \frac{H_{11}}{gD} \tag{4.14}$$

and

$$D = H_{01}H_{12} - H_{11}H_{02} \,. \tag{4.15}$$

We now have defined all the quantities in Figure 4.1, which are summarised here: L is the angular momentum of the whole robot about the support point; $\dot{L}$, $\ddot{L}$ and $\dddot{L}$ are its first three time derivatives; q_a and $\dot{q}_\mathrm{a}$ are the position and velocity variables of the actuated joint (which is joint 2 in Figure 4.2); and Y_1 and Y_2 are the two configuration-dependent gains that describe the robot's balancing dynamics. $\dddot{L}$ is the input, q_a is the output, and L, $\dot{L}$, $\ddot{L}$ and q_a are the state variables.

The next step consists in defining $\dddot{L}$. To maintain balance, the balance controller's task is to determine an appropriate value for $\dddot{L}$ that will enable q_a to track a specified command signal q_c. A suitable control law to accomplish this is

$$\dddot{L} = k_{dd}\ddot{L} + k_d\dot{L} + k_L L + k_q(q_\mathrm{a} - u) \,, \tag{4.16}$$

where u is the input to the controller (see Equation (4.19)). The feedback gains are obtained via pole placement as:

$$\begin{aligned} k_{dd} &= -a_3 & k_d &= -a_2 + a_0 Y_2/Y_1 \\ k_L &= -a_1 & k_q &= -a_0/Y_1 \,, \end{aligned} \tag{4.17}$$

where

$$\begin{aligned} a_0 &= \lambda_1\lambda_2\lambda_3\lambda_4 \\ a_1 &= -\lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_2\lambda_4 - \lambda_1\lambda_3\lambda_4 - \lambda_2\lambda_3\lambda_4 \\ a_2 &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 + \lambda_2\lambda_4 + \lambda_3\lambda_4 \\ a_3 &= -\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4\,, \end{aligned} \tag{4.18}$$

and $\lambda_1, \ldots, \lambda_4$ are the chosen values of the poles. Let us define the time constant for toppling, denoted as T_c, as the rate at which the robot starts to fall when there is no movement in the actuated joint. It is a physical property of the robot and varies depending on the robot's structure and configuration. A good initial strategy to select the first three poles is $\lambda_1 = \lambda_2 = -1/T_\mathrm{c}^*$, where T_c^* is the robot's natural time constant of toppling in a balanced configuration (e.g. $q_1 = q_2 = 0$ for the robot in Figure 4.2), and $\lambda_3 = T_\mathrm{c}$, where T_c is the robot's natural time constant of toppling in the current configuration. The fourth pole is the one that determines the closed-loop bandwidth and has to be selected carefully not to make the system unstable.

When using pole placement, it is assumed that the two gains, Y_1 and Y_2, in Figure 4.1 are constants, which implies that the plant is linear. However, in reality, these gains vary with the robot's configuration, which means that the plant is not perfectly linear. Despite this, the variation in the gains is small enough that it does not greatly impact the performance of the balance controller.

The input to the controller, u, is computed from the filtered command signal, q_f, according to

$$u = q_\mathrm{f} + \alpha_1\dot{q}_\mathrm{f} + \alpha_2\ddot{q}_\mathrm{f}, \tag{4.19}$$

where $q_\mathrm{f} = \mathrm{AF}(q_\mathrm{c})$, $\dot{q}_\mathrm{f} = \mathrm{AF}(\dot{q}_\mathrm{c})$ and $\ddot{q}_\mathrm{f} = \mathrm{AF}(\ddot{q}_\mathrm{c})$, and q_c, $\dot{q}_\mathrm{c}$ and $\ddot{q}_\mathrm{c}$ are the desired position, velocity and acceleration of the actuated joint. α_1 and α_2 are feedforward gains that introduce two zeros into the transfer function at $-1/T_\mathrm{c}^*$ which cancel the poles λ_2 and λ_3. AF is an acausal filter consisting of a first-order low-pass filter with a time constant T_f that runs backwards in time from a point sufficiently far in the future back to the present. Its reverse-time transfer function is $1/(1+T_\mathrm{f}s)$, which corresponds to a forward-time transfer function of $1/(1-T_\mathrm{f}s)$. The practical effect of this filter is to minimize the robot's tracking error by causing it to lean ahead of time in anticipation of balance disturbances caused by the motion command signal q_c. More information about this filter can be found in [18 § 4.3]. In the experiments presented in this thesis, the complete signal $u(t)$ is calculated in advance, setting $T_\mathrm{f} = T_\mathrm{c}^*$. However, in a practical implementation, it would be enough to give the controller $3\,T_\mathrm{c}^*$ advance notice of the command signal [18].

Given the control law in (4.16), with gains as in (4.17) and (4.18), and the input signal as in (4.19), it can be shown that the complete transfer function from q_c to q_a would be

$$q_\mathrm{a}(s) = \frac{a_0(1+T_\mathrm{c}s)(1+\alpha_1 s+\alpha_2 s^2)}{s^4+a_3s^3+a_2s^2+a_1s+a_0} q_c(s), \tag{4.20}$$

which simplifies to

$$q_\mathrm{a}(s) = \frac{1}{1+s/(-\lambda_4)}\, q_c(s) \tag{4.21}$$

after the cancellation of poles and zeros, if it were really true that Y_1, Y_2 and T_c were constants [18]. This expression is the theoretical transfer function of the balance controller, and it will be compared to the experimental results obtained in this thesis.

The controller's output must be either an acceleration or a torque for the actuated joint; that is, either $\ddot{q}_2$ or τ_2. Combining Equations 4.8, and 4.11 we obtain

$$\begin{bmatrix} 0 & H_{01} & H_{02} \\ 0 & H_{11} & H_{12} \\ -1 & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \tau_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -\ddot{L}/g - C_0 \\ -C_1 \\ -C_2 \end{bmatrix} \tag{4.22}$$

which can be solved for both $\ddot{q}_2$ and τ_2. The main point to understand is that the elements H_{ij} can be computed using any typical computer program for calculating joint-space inertia matrices, while the elements C_i can be computed using standard code for joint-space bias vectors, which involves the sum of all velocity and gravitational terms. The other variables required by the balance controller are the robot's total mass, m, the gravitational acceleration magnitude, g, which are constants, and the x coordinates of the robot's CoM position and velocity, c_x and $\dot{c}_x$. By taking advantage of traditional dynamics software in this manner, it is possible to minimize the amount of new code required to implement the balance controller.

4.2 RWP Special Case

In the case of the RWP, thanks to its mechanical symmetry (the centre of mass of the reaction wheel coincides with its rotation axis), the balance controller can be simplified. This analysis has already been presented in [4] and is briefly summarized here since it is the first controller that will be experimentally tested in this thesis.

The motions of the robot shown in Figure 4.3 can be described by the two joint angles q_1 and q_2, and due to the symmetry of the reaction wheel, the robot balances when $q_1 = 0$,

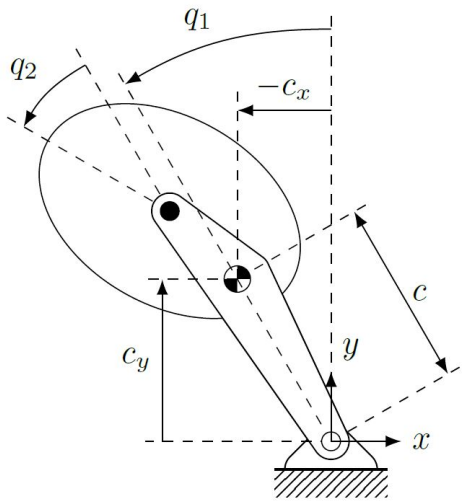


Figure 4.3 Reaction wheel pendulum model [4].

regardless of the value of the only actuated joint q_2. The equation of motion of the RWP is

$$\begin{aligned} H_{11}\ddot{q}_1 + H_{12}\ddot{q}_2 &= mcg\sin(q_1) \\ H_{21}\ddot{q}_1 + H_{22}\ddot{q}_2 &= \tau_2 \end{aligned} \tag{4.23}$$

where $\ddot{q}_i$ are the joint acceleration variables, H_{ij} are the elements of the joint-space inertia matrix, τ_2 is the torque of the actuated joint 2, m is the total mass of the robot, c is the distance of the CoM from the origin, and g is the magnitude of the gravitational acceleration. For this RWP, the joint-space inertia matrix terms can be rewritten as

$$H_{11} = mcgT_c^2 \quad \text{and} \quad H_{12} = H_{21} = H_{22} = -G_\omega H_{11} \tag{4.24}$$

where T_c is the natural time constant of the toppling of the robot when $q_1 = 0$ considering the robot as a single rigid body, and G_ω is the angular velocity gain of the robot [133]. Equations 2 and 3 of [4] relate T_c and G_ω to the dynamic parameters of the robot and not to its configuration.

$$T_c^2 = \frac{H_{11}}{mcg} = \frac{mc^2 + I_1 + I_2}{mcg} \tag{4.25}$$

$$G_\omega = \frac{\Delta\dot{q}_1}{\Delta\dot{q}_2} = -\frac{I_2}{mc^2 + I_1 + I_2} \tag{4.26}$$

where I_1 and I_2 are the rotational inertias of the two bodies about their CoM respectively. The equations are exact only when $c_y = c$ meaning $q_1 = 0$. The state variables required by the balance controller and presented in Section 4.1 can be simplified by replacing the

configuration-dependent joint-space inertia matrix terms with the constant values of T_c and G_ω (as shown in Equations 9 to 12 of [4]), and now they read

$$M = T_c^2(\dot{q}_1 - G_\omega \dot{q}_2), \tag{4.27}$$

$$\dot{M} = q_1, \quad \ddot{M} = \dot{q}_1, \quad \dddot{M} = \ddot{q}_1, \tag{4.28}$$

$$Y_1 = \frac{-1}{T_c^2 G_\omega} \quad \text{and} \quad Y_2 = \frac{1}{G_\omega} \tag{4.29}$$

where $M = L/(mcg)$ and after the assumption of small variations of q_1 and a negligible value of $\dot{q}_1^2$. The control law presented in 4.16 can be adapted to the specific case of the RWP and now reads

$$\dddot{M} = k_{dd}\ddot{M} + k_d\dot{M} + k_M M + k_q(q_2 - u) \tag{4.30}$$

where the gains are those described in Equation 4.17, $k_M = k_L$, and the input signal u is the same as the one described in Equation 4.19. The output of the balance controller can be converted into a torque or acceleration command for joint 2

$$\begin{aligned} \ddot{q}_2 &= (mcg\sin(q_1) - H_{11}\dddot{M})/H_{12} \\ \tau_2 &= H_{21}\dddot{M} + H_{22}\ddot{q}_2 \end{aligned} \tag{4.31}$$

where the terms H_{ij} are constant and defined in Equation 4.24. Thanks to the specific structure of the RWP, the simplified version of the balance controller allows balancing the robot without using any dynamics software to calculate the matrix H and the vector C. In this manner, the amount of code required to implement the balance controller is at its minimum.

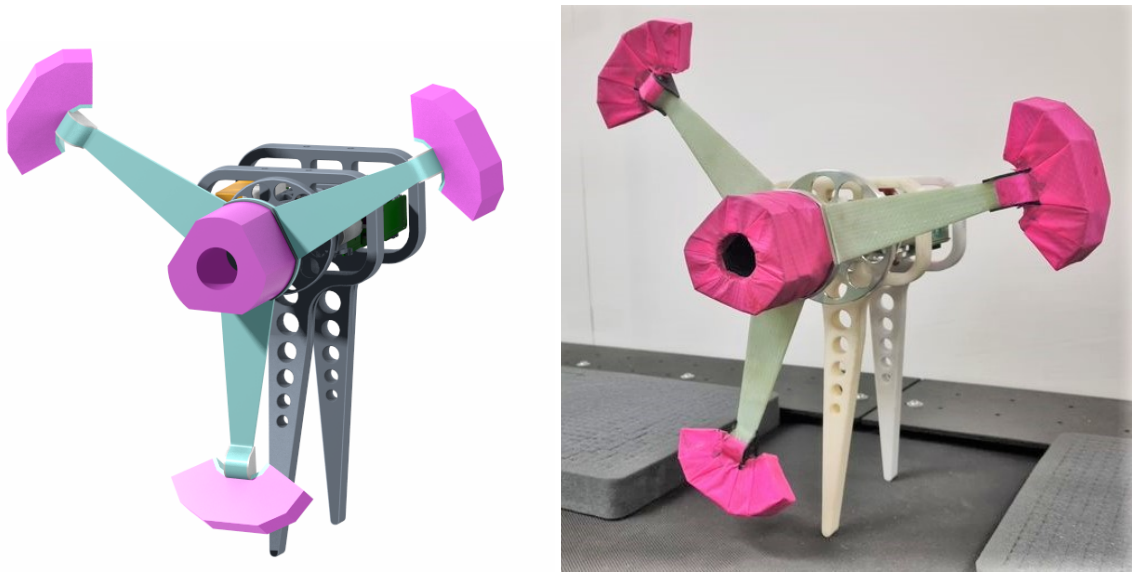
4.3 Balance Offset Observer

The content of this section has already been presented in [4 § 4], and it is reported here for completeness since all the experiments presented in this thesis rely on this technique.

The reaction wheel pendulum described before balances when $q_1 = 0$, and it has been proved that the balance controller is quite sensitive to errors in the estimate of the balanced configuration [50; 91]. This error is specifically crucial for Skippy, which uses as a reference the angle measured by the IMU. This value can be affected by drift or offset due to the physical assembly. Consequently, a method to compensate for such errors has to be used.

Let us define

$$q_1 = \hat{q}_1 - q_o \tag{4.32}$$



(a) (b)

Figure 4.4 (a): CAD model; (b): Skippy's Head as RWP

where $\hat{q}_1$ is the measured value of q_1, and q_o is the unknown balance offset error, which is assumed to vary so slowly that $\dot{q}_o$ can be neglected with respect to $\dot{\hat{q}}_1$ and $\dot{q}_1$. There are now two ways to calculate $\dot{M}$

$$\dot{M} = \hat{q}_1 - q_o \tag{4.33}$$

and

$$\dot{M} = \frac{\mathrm{d}M}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t} T_\mathrm{c}^2 (\dot{q}_1 - G_\omega \dot{q}_2) \tag{4.34}$$

where $\mathrm{d}/\mathrm{d}t$ is the numerical differentiation. Combining these two equations, we can estimate q_o as

$$q_o = \mathrm{LPF}(\hat{q}_1 - \frac{\mathrm{d}M}{\mathrm{d}t}) \tag{4.35}$$

where 'LPF' is a low pass filter with a cutoff frequency of $1\,\mathrm{rad/s}$. The expression $\hat{q}_1 - q_o$ is used in place of q_1 in all the equations of Sections 4.1 and 4.2.

4.4 Experimental Setup

Figure 4.4b presents the robot used for this experiment. It consists of Skippy's 'Head', containing the control unit (referred as the 'Brain'), sensors and motor, mounted on a pair of stilts, and the reaction wheel, which we usually refer to as the symmetric crossbar, in contrast to the asymmetric crossbar, which will be presented in the upcoming chapters. The

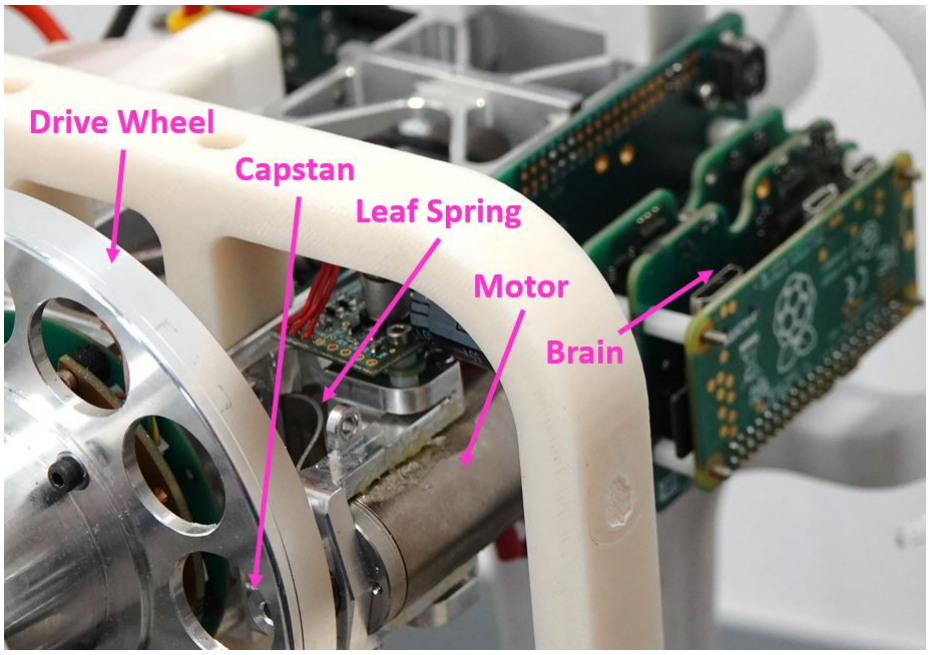


Figure 4.5 Capstan-drive mechanism and Brain [5].

Head, together with the stilts, forms body 1, and the crossbar is body 2 in the model of the reaction wheel pendulum shown in Figure 4.3. The dynamic parameters are initially taken both from the CAD model of the robot and from the manual measurements (see Appendix A), and then manually tuned to compensate for minor inaccuracies in the CAD model and the measurements. The stilts touch the ground at two points, making a two-point contact foot aligned with the rotation axis of the crossbar, making the robot behave as a planar RWP.

4.4.1 Actuation System

The actuation system depicted in Figure 4.5 comprises a Maxon DC motor DCX22L-GB-KL-18V [110], which actuates a custom-made spring-loaded friction capstan-drive mechanism. The motor shaft is attached to an aluminium cylindrical capstan, which is pressed against the inner part of an aluminium cylindrical drive wheel by a titanium alloy leaf spring. This generates a pressing force of 120 N at the capstan, which allows the transmission of up to 3 Nm of torque to the drive wheel without slipping. The balance controller and the motor servo ensure that this torque limit is never exceeded.

The mechanism is designed for Skippy, which is intended to hop up to 3 m and survive falls from that height without damaging the drive system. In fact, the drive wheel slips on the capstan when high accelerations due to crash landings are applied to the crossbar, not damaging the rest of the drive system. The rolling motion between the capstan and the wheel

provides minimal backlash and a reduced stiction coefficient compared to traditional geared drives. However, the system is susceptible to wear and may be damaged by dust and debris. The system has a gear ratio $G = 12.5$. The motor is controlled by a Pololu G2 24v13 motor driver [118], which enables the Brain to regulate the motor via a Pulse Width Modulated (PWM) signal.

4.4.2 Reaction Wheel

The balancing machine, as will do Skippy, uses a three-blade reaction wheel to balance itself. Each blade is made of fibreglass of a specific custom shape, and it can withstand shocks due to crash landings without breaking. Glued brass weights are at the edge of each blade to increase the wheel's inertia. The extremities of the wheel are protected with shock-absorbing foam covered with an anti-tear pink fabric.

4.4.3 Sensors

To maintain balance, the robot needs to know the angle between body 1 and the vertical (q_1) and the angle between the crossbar and body 1 (q_2). Additionally, the robot needs to determine the angular position of the motor (q_m) to calculate its speed, which serves as input to the motor servo. All the sensors have been thoroughly tested in Chapter 3.

IMU

The Inertia Measurement Unit mounted on the robot is the VN-100 by Vectornav [121]. It has the y-axis aligned with the rotation axis of the crossbar. In the configuration used for this experiment, it is used to evaluate the angle q_1 and its velocity $\dot{q}_1$. It is connected to the Brain with a dedicated SPI channel and sampled at 1 kHz. The IMU's latency is 4 - 5.6 ms.

Absolute Position Encoder

Referring to Figure 4.3, the AksIM-2 absolute magnetic rotary encoder is utilized to measure the angle q_2 of the crossbar with respect to the lower link of the robot. The specific model used is the MB049DCC19BDNT00, which has a magnetic ring diameter of 49 mm and provides a resolution of 19 bits for angular position. The Brain samples the encoder at a rate of 5 kHz, utilizing the BiSS-C interface operating at a frequency of 3.8 MHz.

Parameter	Measured Value	CAD Value	Tuned Value
Velocity Gain G_ω	-0.068	-0.072	-0.071
Toppling Constant T_c [s]	0.182	0.176	0.182
First Moment of Mass mc [kg m]	0.53	0.54	0.55

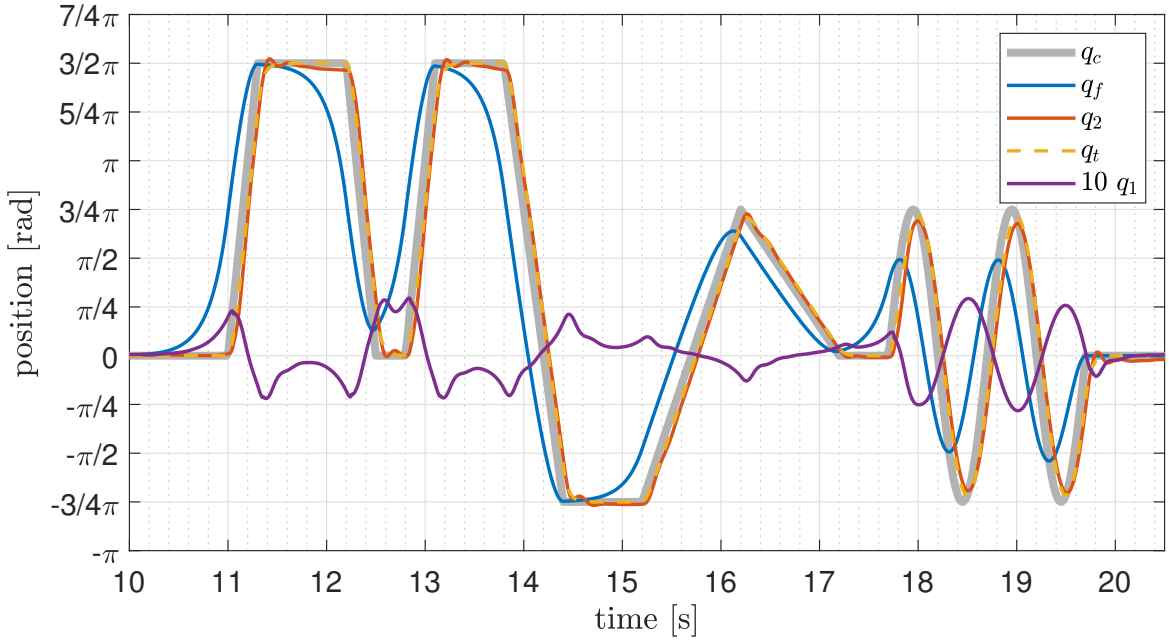
Table 4.1 Dynamic parameters of the reaction wheel pendulum.

Incremental Encoder

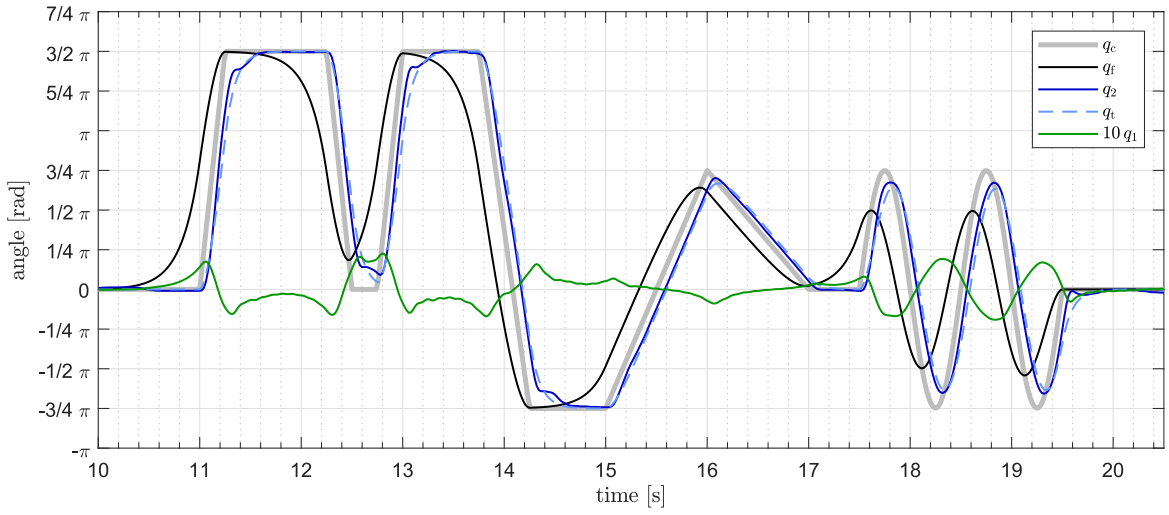
To determine the angular position of the motor, a Maxon ENX 16 EASY incremental position encoder is used. The encoder generates 4096 steps per turn through its incremental interface. The Brain samples the encoder at a rate of 5 kHz using the eQEP module, and the register is read synchronously with the absolute encoder. The quadrature encoder's position values, denoted by q_m, are then used to approximate the motor's angular velocity, which is subsequently used by the motor servo. A numerical differentiation method and a second-order discrete-time Butterworth filter with a cutoff frequency of 400 rad/s are employed to estimate the motor's angular velocity, denoted as $\dot{q}_m$.

4.4.4 Control Unit: The Brain

The balancing machine, like Skippy, is completely untethered. All sensing and computations are conducted in real-time onboard by the Brain's microcontroller, the TMS320F28377S [134], to which all the sensors are directly connected. The control system is divided into two layers, with the balance controller operating at 1 kHz and the motor servo tracking the output of the first layer at 5 kHz. The acceleration of the crossbar $\ddot{q}_2$ is the output of the balance controller, which is then numerically integrated to obtain the crossbar velocity ($\dot{q}_2$), which is then multiplied by the gear ratio G to transform it into motor velocity ($\dot{q}_m$). The motor velocity is used as input to the motor velocity servo, which is a simple proportional controller with a gain of $K_p = 0.008\,\mathrm{V}/(\mathrm{rad/s})$. Additionally, a feed-forward signal is included to account for the back electromotive force (τ_{emf}), which is added to the motor servo output as an additional term $\tau_{emf} = \tau_c \dot{q}_m$, where τ_c is the torque constant of the motor and $\dot{q}_m$ is the motor velocity. The low-level controller output consists of the proportional PWM signal and the motor's direction of rotation.



(a)



(b)

Figure 4.6 (a) Skippy and (b) Tippy [4] tracking performance. q_c is the command signal, q_f is the filtered command signal, q_2 is the angle of the crossbar, q_t is the theoretical response, and 10 q_1 is ten times the angle of the robot with respect to the vertical.

4.5 Experimental Results

The experiment evaluates the tracking performance of the RWP implementation of the balancing controller. The robot used for the experiment is the one presented in Section 4.4, and the dynamic parameters of the simplified RWP balance controller are reported in Table 4.1. Such parameters are initially both measured experimentally (see Appendix A) and derived from the CAD model (Equations 4.25 and 4.26). The obtained values agree and are fine-tuned manually during the experimental procedure. The robot tracks the desired command signal, which is the angular position of the crossbar $q_2 = q_\mathrm{a}$. The command signal q_c has been transformed using the leaning in anticipation technique described in Section 4.1. Although T_c varies with the robot's configuration, the filter time constant T_f is assumed to be constant with the tuned value of $T_\mathrm{c} = 0.182\,\mathrm{s}$. In the following experiment, the poles of the balance controller are chosen as follows: $\lambda_1 = \lambda_2 = \lambda_3 = -1/T_\mathrm{c}$ and $\lambda_4 = -20$, leading to a theoretical closed-loop transfer function

$$q_\mathrm{a}(s) = \frac{1}{1 + s/(-\lambda_4)}\, q_c(s) = \frac{1}{1 + 0.05\,s}\, q_c(s) \tag{4.36}$$

which is used as a reference for the actual response in the experimental results.

4.5.1 Tracking Performance

Figure 4.6 compares the tracking performance obtained with this robot with the one obtained with Tippy in [4]. Thanks to a stiffer mechanical design, it was possible to have the pole $\lambda_4 = -20$, which is two times higher than in [4] where $\lambda_4 = -10$. The higher pole and a better mechanical design enable faster and more precise tracking, with the obtained results clearly outperforming those obtained in [4]. Skippy's Head is not fixed to the ground; instead, the stilts simply touch the ground. Consequently, the reaction wheel's energetic and fast motions cause the stilts to slip. The controller can compensate for such an unexpected disturbance at the price of a small overshoot. This effect is visible at 11.5 s (see Figure 4.6a), where a high deceleration to stop the crossbar causes the stilts to slip, with a consequent slight overshoot in the trajectory tracking. Further discussion on this topic is present in Chapter 5, where the foot slips significantly, causing a more significant compensating overshoot.

4.6 Conclusion

This chapter presented the results obtained with Skippy's Head configured as a reaction wheel pendulum. This proved the efficiency of part of the sensorimotor system that Skippy will use to perform its duties. It also confirmed that the controller described in [18] could be used for high-performance control using an IMU to estimate the angle with respect to the vertical, despite the delay in the sensor's measurements. This is a new result compared to Tippy [4], where the vertical was measured with an encoder. Furthermore, this experiment proved the ability of the controller to achieve high balancing performance with a robot not anchored to the ground (opposite of Tippy, which pivots on a shaft fixed to the ground), in contrast to the results obtained in balancing HyQ [92] where it was impossible to achieve precise and fast tracking due to its hardware design not specifically intended for these tasks.

Chapter 5

General Inverted Double Pendulum

This chapter presents the first implementation of the general balance controller presented in Section 4.1 on a inverted double pendulum. The results shown in the upcoming sections have already been published in [5] (© 2023 IEEE, partially reproduced here with permission) and are reported here because they are significant for the development of Skippy and are part of the novelty of this thesis. The figures and tables in this chapter are taken from [5].

5.1 Experimental Setup

The experimental setup for this experiment is the same as the one described in Section 4.4 with the asymmetric crossbar replacing the reaction wheel, and is shown in Figure 5.1. The asymmetric crossbar is a 3D-printed beam connected at one end to the drive mechanism and with extra weight at the other end to increase its inertia. The robot in this configuration can be described as a inverted double pendulum balancing in 2D, as shown in Figure 4.2, with the dynamic and kinematic parameters described in Table 5.1.

5.2 Control System

The simplifying assumptions made for the RWP in Section 4.2 are not valid anymore; hence the general balance controller is applied. The control system at each iteration evaluates the matrix H and the vector C described in Section 4.1. The balance controller uses such information to calculate the desired output for the actuated joint $q_\mathrm{a} = q_2$. In the experiments reported here, the poles are chosen as follows: $\lambda_1 = -1/T_\mathrm{c}$, $\lambda_2 = \lambda_3 = -1/T_\mathrm{c}^*$, and $\lambda_4 = -20$, where T_c is the robot's natural time constant of toppling in its current configuration, and T_c^*

Dynamic parameter	CAD value	Tuned value
$m_1\ [kg]$	1.41	1.41
$I_1\ [kg\,m^2]$	0.0073	0.007
$c_{1x}\ [m]$	-0.004	-0.009
$c_{1y}\ [m]$	0.255	0.254
$l_1\ [m]$	0.28	0.28
$m_2\ [kg]$	0.44	0.44
$I_2\ [kg\,m^2]$	0.0083	0.008
$c_2\ [m]$	0.110	0.109

Table 5.1 Dynamic parameters of the robot required by the model of Figure 4.2, derived from the CAD model (central column) and manually adjusted via experimentation on the real robot (right column).

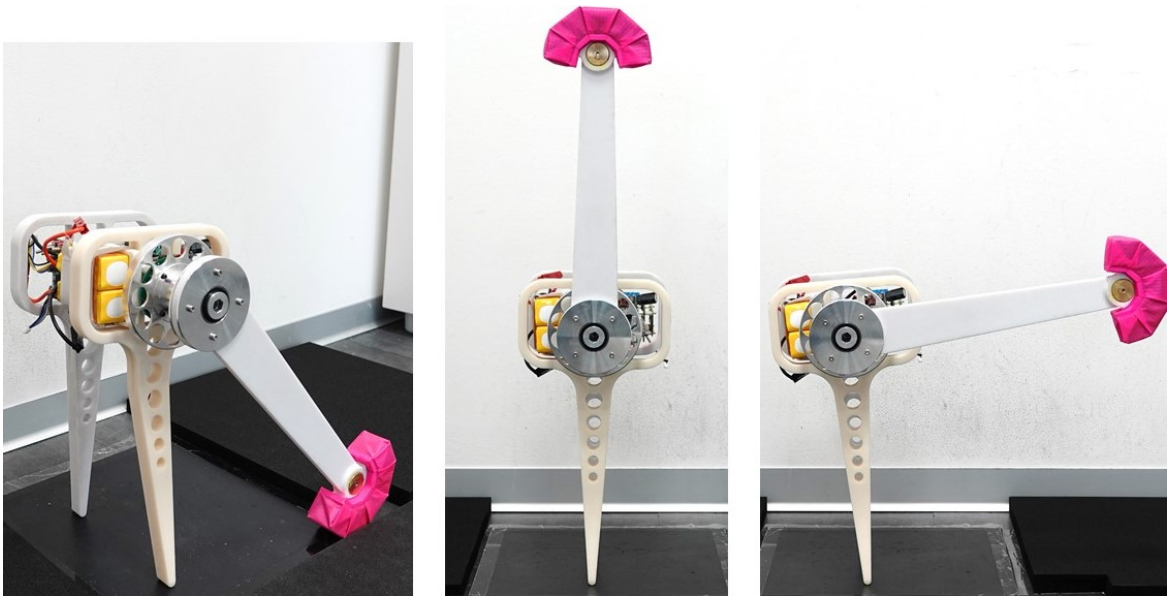


Figure 5.1 Photos of the balancing machine. In the left picture, the robot is not operating and rests by touching the ground with the upper link and the two-point contact foot. In the central picture, it is balancing with the upper link upright. In the right picture, it is balancing with the upper link at an angle of $-\pi/2$ with respect to the lower link.

is a constant value equal to T_c when the robot is balancing with the upper link upright (i.e., $q_1 = q_2 = 0$). The fourth pole, which is the one that determines the closed-loop bandwidth, is four times higher than that in [59] and 2 times higher than [4], allowing faster and more responsive behaviour. The input command signal q_c is filtered as described in Equation 4.19, and the acausal filter time constant used in this experiment is $T_f = T_c^*$. The complete transfer function is

$$q_a(s) = \frac{a_0(1+T_c s)(1+\alpha_1 s+\alpha_2 s^2)}{s^4+a_3 s^3+a_2 s^2+a_1 s+a_0} q_c(s), \tag{5.1}$$

which simplifies to the theoretical response

$$q_a(s) = \frac{1}{1+s/(-\lambda_4)}\, q_c(s) = \frac{1}{1+0.05\,s}\, q_c(s) \tag{5.2}$$

5.3 Experimental Results

This is the first experimental evaluation of the tracking performance of the general implementation of the balancing controller. The results are shown alongside the theoretical results in Figures 5.2 and 5.3. The robot follows a desired command signal that represents the angular position of the asymmetric crossbar, which has been transformed using the leaning in anticipation technique discussed in Section 4.1. Figure 5.4 shows the tracking error through the experiment after the robot self-balanced itself. Although T_c varies with the configuration of the robot, in the range 0.170 s to 0.182 s, the filter time constant T_f is assumed to be constant with the value of $T_c^* = 0.182$ s, which is the value of T_c when the robot is in an upright configuration.

The experiment begins with the robot in a rest position, with no actuation and the crossbar touching the ground, as depicted in Figure 5.1. The first step of the robot is to autonomously balance itself by gently pushing the crossbar towards the foot until the machine begins to tip towards the left. At this point, the balancing controller is activated, and the crossbar is raised to an upright position, reaching $q_2 = 0$. As the crossbar nears the vertical position, the balance offset observer, explained in Section 4.3, is turned on to estimate, track, and compensate for the difference between the IMU's estimation of the vertical and the actual balanced vertical position of the robot. The robot then commences tracking a trajectory at time $t = 5$ s in Figure 5.2. The trajectory starts with a slow sine wave that covers nearly the entire range of motion of the upper link (otherwise, it will collide with the ground). Next, there is a fast sine wave that is centred on the upright configuration. This is followed by

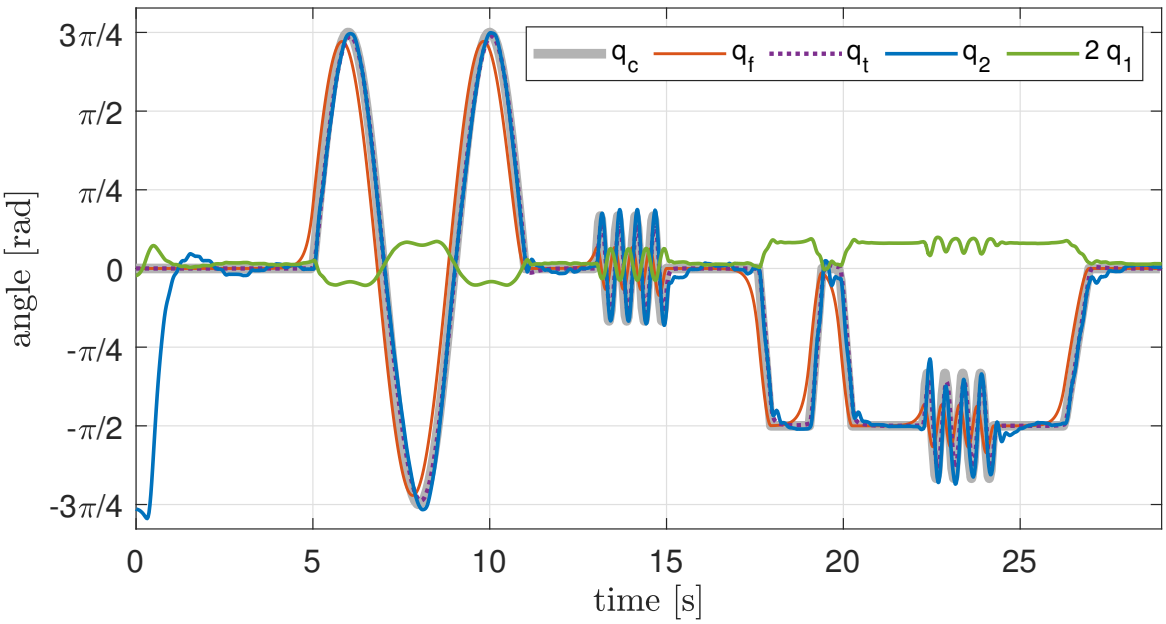
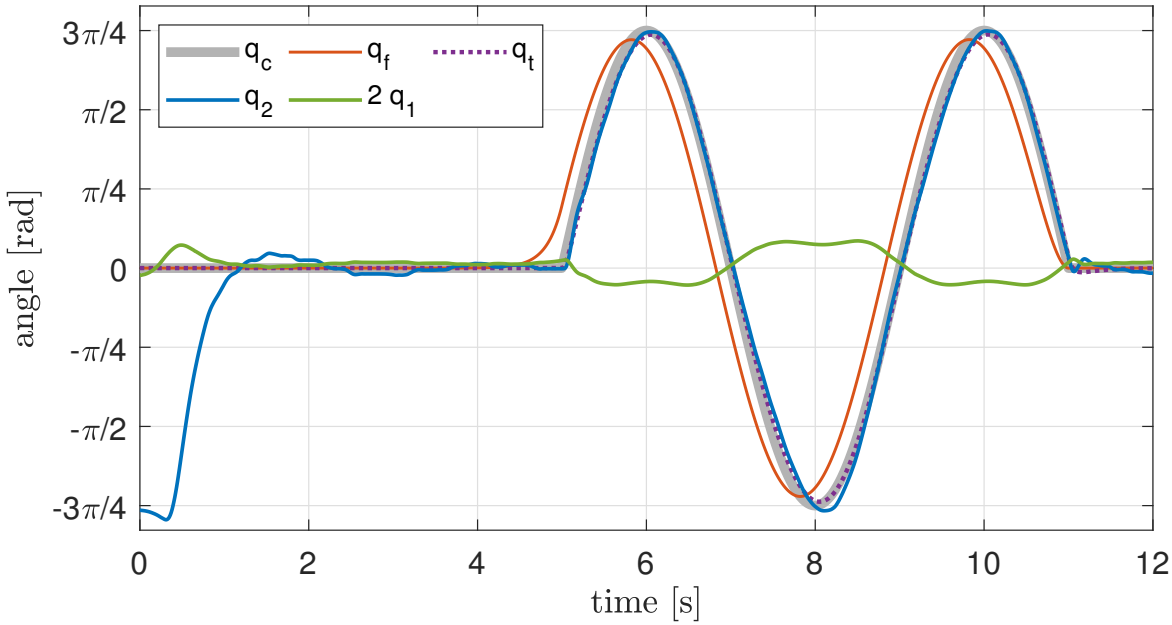


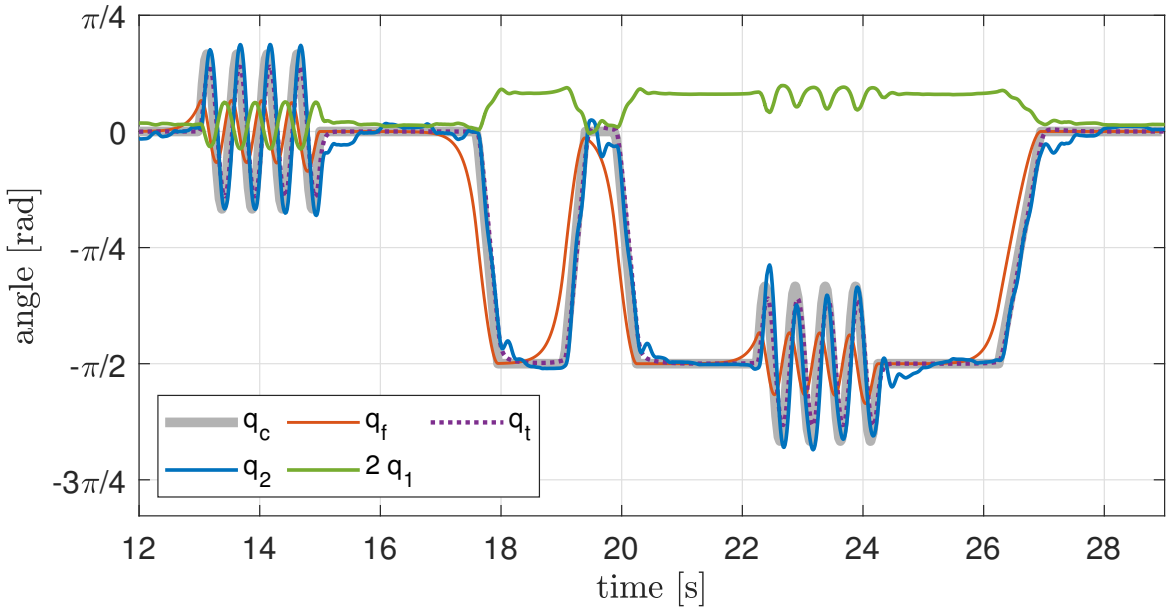
Figure 5.2 Tracking of motion command q_c. q_f is the filtered input signal, q_t is the theoretical response while q_2 is the measured value. $2q_1$ is two times the vertical angle.

three fast ramps that each have a magnitude of $\pi/2$ rad and last for 0.35 s. After this, there is another fast sine wave that is centred on the $-\pi/2$ rad bent configuration. Finally, the trajectory finishes with a slow ramp that takes 0.7 s to return to the upright configuration. The two sets of fast sine waves featured in the experiment have an amplitude of $\pi/6$ rad and a frequency of 2 Hz, which is more than three times higher than the frequency of the waves tracked by Acrobot (0.63 Hz [59]) and twice the frequency of the waves tracked by Tippy (1 Hz [4]). The tracking delay of the system (between q_t and q_2) is less than half the delay found in the aforementioned studies, with Tippy having a tracking delay of $0.15\,T_c$, Acrobot of $0.18\,T_c$, and the new balancing machine of $0.077\,T_c$, where T_c is different for each robot, being a physical property of the machine. Furthermore, the delay introduced by the IMU does not impact the performance of the controller, which is consistent with the simulation outcomes presented in [18].

The front foot, which is the contact point closer to the camera, bears most of the robot's weight, while the rear contact is prone to slipping when the upper link undergoes rapid motions involving large accelerations and forces. This slipping causes a yawing motion, with q_y representing the yaw angle. Although the planar controller with only one actuated joint cannot control these out-of-plane motions, the fact that they do not significantly affect the controller's performance demonstrates its robustness. Throughout the experiment, these small turns result in a cumulative rotation of about 6.3°. The controller responds to such



(a)



(b)

Figure 5.3 Enlarged views of Figure 5.2.

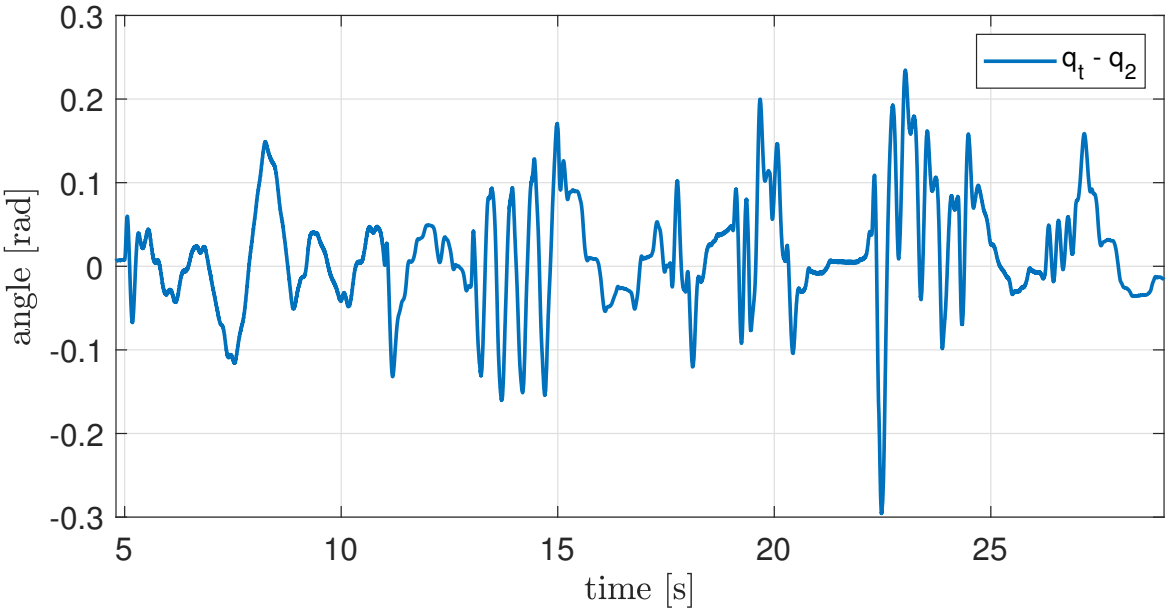


Figure 5.4 Tracking error through the experiment after the robot self-balanced itself.

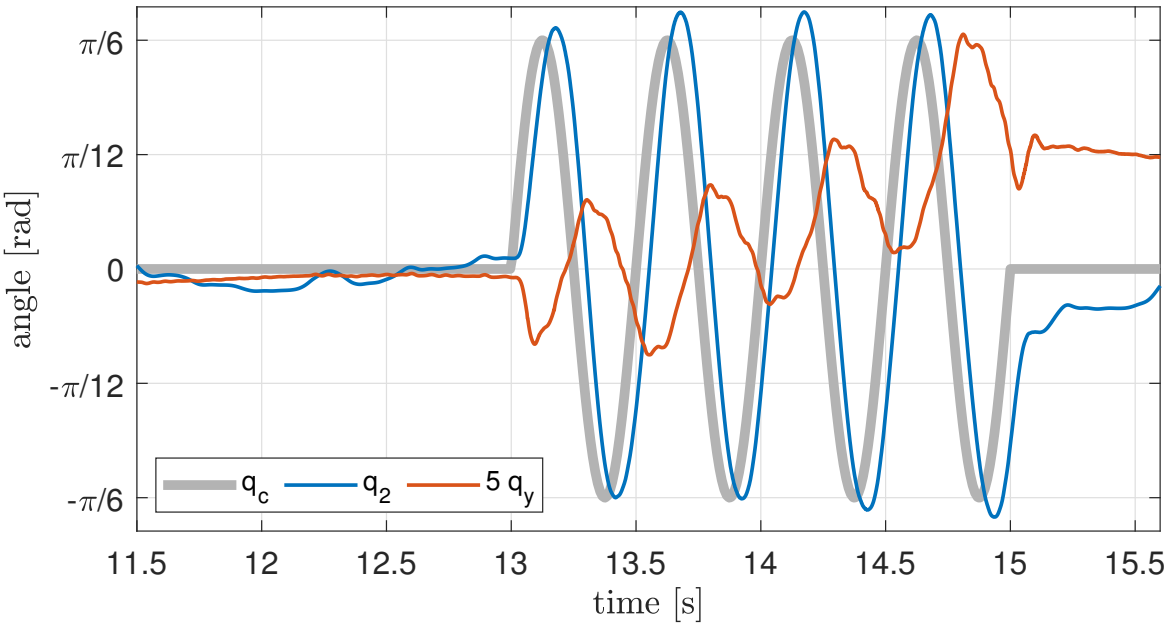


Figure 5.5 The back foot slips during the second sine wave. The controller is able to respond to a 0.1 rad slip with an overshoot of 0.065 rad without losing its balance. $5\,q_y$ is five times the yaw angle.

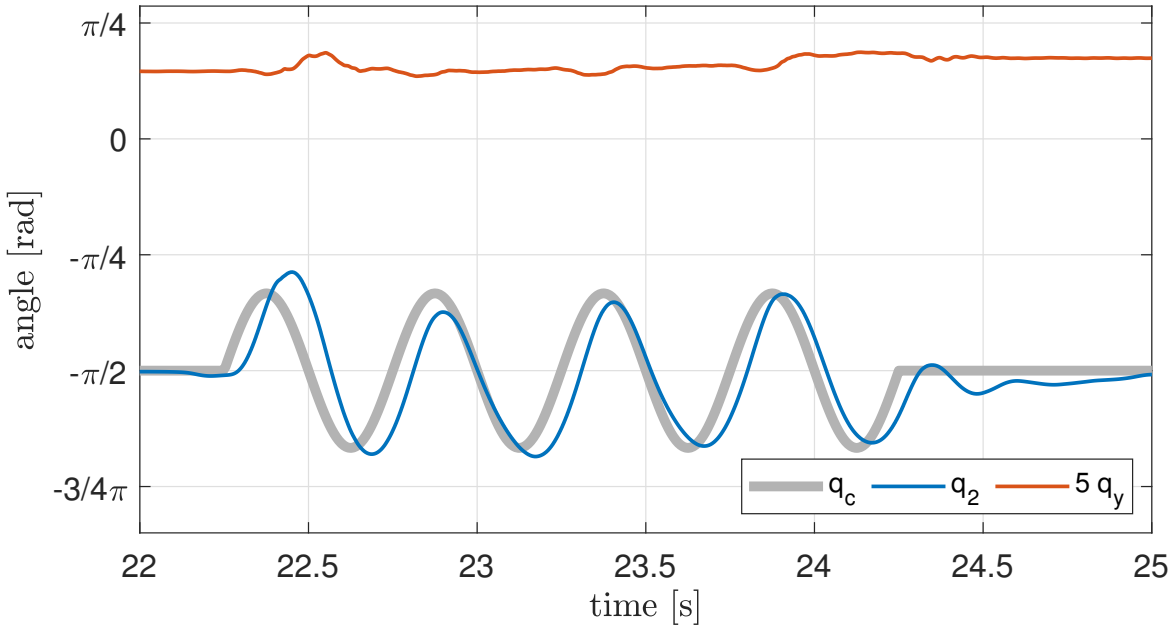


Figure 5.6 The back foot slips during the third sine wave, causing the robot to rotate in the yaw direction. Although the magnitude of the slip is small when compared to Figure 5.5, the unbalanced configuration of the robot causes an initial overshoot during the first sine cycle. $5\,q_y$ is five times the yaw angle.

events with a small overshoot in the trajectory tracking while ensuring system stability and controllability. Figure 5.5 shows the overshoot during the tracking of the first set of fast sine waves, where the back foot of the robot slips back and forth on every cycle, causing q_y to vary in a sinusoidal way. The accompanying video of [5] provides a clear illustration of this effect. The second set of fast sine waves has the same magnitude and frequency as the first one but has a mean value of $-\pi/2$ rad. In this specific configuration, even a slight slip results in significant overshoot, as seen in Figure 5.6. Nonetheless, the controller responds effectively to the overshoot and recovers good tracking after one sine cycle. Figure 5.2 shows a small overshoot at time 19 s followed by a compensation undershoot; this behaviour is the result of the foot slipping during the tracking of fast ramps with $\pi/2$ rad amplitude and 0.35 s duration.

Other evidence of the robustness of the controller is presented in the accompanying video of [5]. The video shows the robot being intentionally disturbed by the operator, who hits it with a tennis ball and pushes it with a rod. Despite these unexpected disturbances, the robot is able to maintain its balance. Figure 5.7 and the accompanying video of [5] show the effects of the tennis ball.

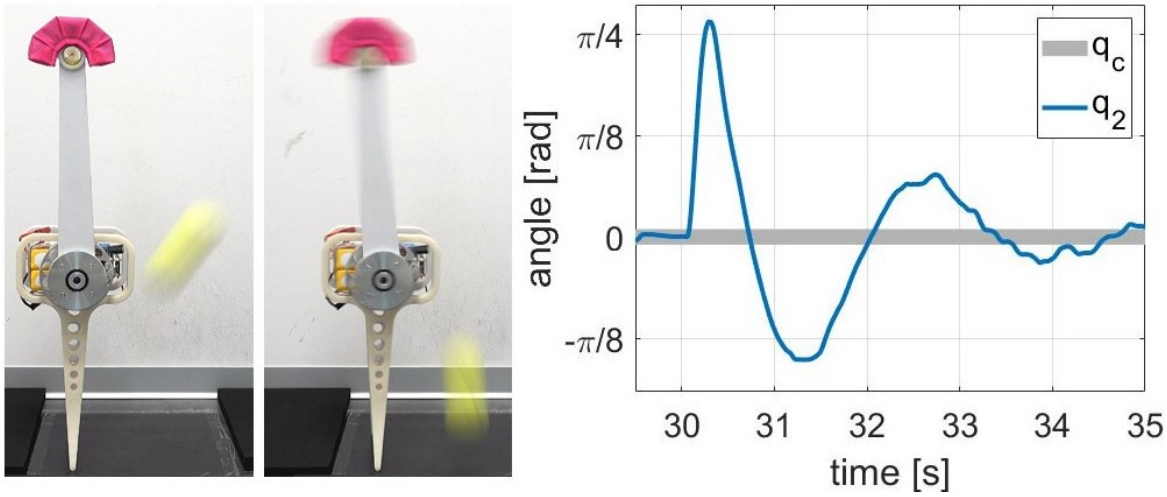


Figure 5.7 In the left picture, the robot is about to be hit by a tennis ball. In the central picture, the robot reacts to the external disturbance by making an excursion of the upper link to maintain its balance. The graph on the right shows the angular position of the upper link after being hit by the ball.

5.4 Conclusion

This chapter presented the first successful experimental demonstration of the general balance control theory on a floating base robot that functions as an inverted double pendulum. The results of the experiment showcase the controller's capacity to accurately track slow and fast ramps and sinusoids while being able to respond to unforeseen circumstances, such as slipping and external disturbances, with only an acceptable level of overshoot. The robot could stand up autonomously and balance indefinitely without falling over, proving the reliability of the hardware and the balance controller. Furthermore, even with the more complicated dynamics of a general double pendulum, all calculations (except for the filtered input signal) are carried on on-board by a 16-bit micro-controller running at a servo rate of 1 kHz proving the light weight of the control algorithm.

Chapter 6

Balancing on a Rolling Contact

This chapter presents an extension of the general balance controller described in Chapter 4.1. This extension replaces the previous point-foot assumption with a circular-foot assumption, enabling the controller to effectively govern robots that maintain balance and move on a rolling contact. Section 6.1 presents the new controller under the assumption of a robot balancing on flat horizontal terrain. The controller is tested and validated in simulation and on a real robot. Section 6.2 extends the controller's range of applicability to robots balancing on slopes and validates it through simulations only. The contents of Section 6.1 have already been accepted for publication in [135] (© 2023 IEEE, reproduced with permission) and are reported here because they represent a significant theoretical contribution of this thesis, which is then validated with practical experiments. The figures and tables in Section 6.1 are taken from [135].

6.1 Balancing on a Horizontal Surface

The theory presented in this section is applicable to general planar robots that balance on a rolling contact on flat, horizontal ground. However, it will be developed for the special case of an inverted double pendulum. The extension to the general case follows the same procedure as described in [18 § 6].

6.1.1 Robot Model

The model we shall use is shown in Figure 6.2, and we shall call it a rolling double pendulum. It consists of an upper link (Body 2) which is connected to a lower link (Body 1) via an actuated revolute joint (Joint 2) with joint variable q_2. The lower link rolls without slipping

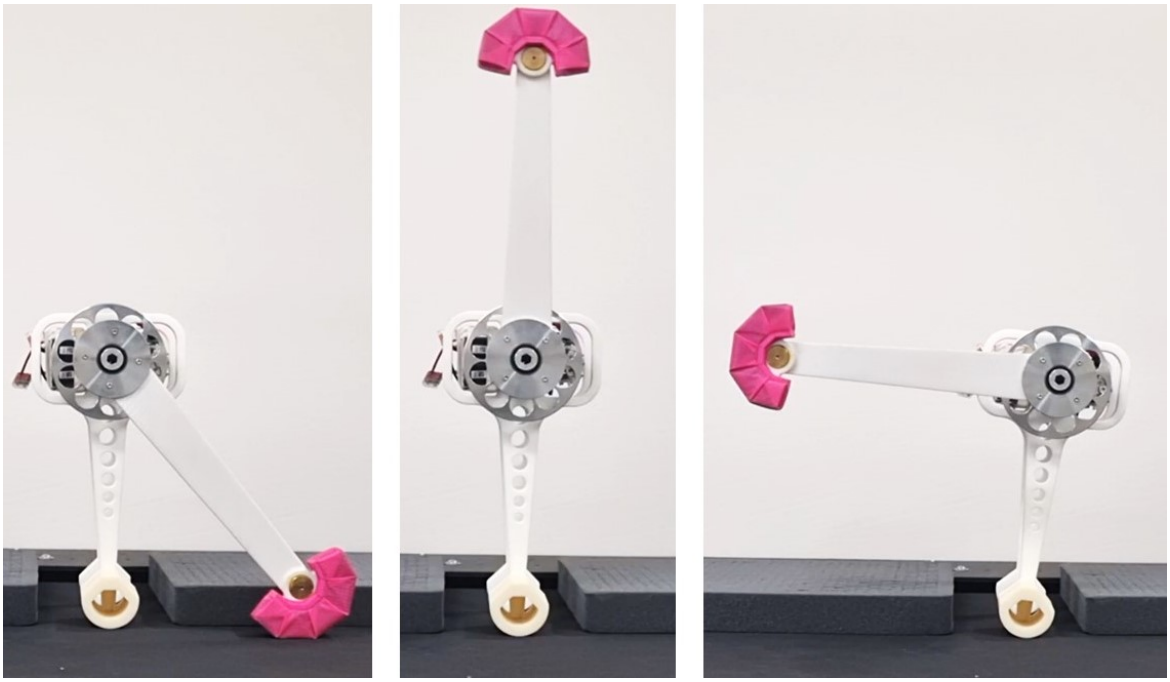


Figure 6.1 Photos of the balancing machine used for the experiments. On the left, the robot is in a resting position, not operating. In the other pictures it is actively balancing with the upper link upright (centre) and with an angle of $\pi/2$ with respect to the lower body (right).

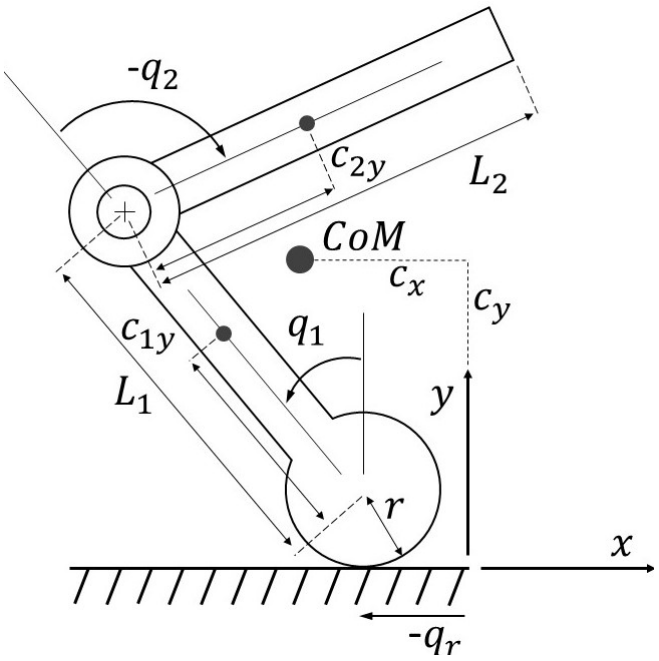


Figure 6.2 Schematic model of the rolling double pendulum.

Body i	m [kg]	c_x [m]	c_y [m]	L [m]	I [kg m^2]
r	0	0	0	r	0
1	1.28	0	0.25-r	0.28-r	0.0099
2	0.705	0	0.114	0.3	0.0121

Table 6.1 Considering body i: m is the mass, c_x and c_y are the coordinates of the centre of mass with respect to the body reference frame, L is the length of the body, I is the inertia of the body at the centre of mass. r is the radius of the rolling contact and varies among the experiments.

over a supporting surface (the ground) which is assumed to be flat and horizontal; and the portion of the lower link that makes contact with the ground is a circle of radius r. The lower link may therefore be a rounded foot, as shown in Figure 6.2; but it could also be a wheel, and this will be discussed in Section 6.1.7. The special case $r = 0$ is allowed, and in this case the model simplifies to an inverted double pendulum on a point foot.

The rolling contact is modelled as a combination of a revolute joint located at the centre of the circle and a prismatic joint in the horizontal direction. The former has a joint variable q_1, which is chosen as the independent variable of the rolling contact, and which is measured from the vertical as shown in Figure 6.2. The latter has a joint variable q_r, which is a dependent variable constrained to have the value $q_\mathrm{r} = -rq_1$. We define the vector $q = [q_\mathrm{r}\, q_1\, q_2]^\mathrm{T}$ to be the vector of all joint variables, and $\bar{q} = [q_1\, q_2]^\mathrm{T}$ to be the vector of independent joint variables.

Table 6.1 presents the numeric values of the kinematic and inertia parameters of the robot used in the simulation experiment in Section 6.1.4, considering a range of radii. The robot used for the physical experiment in Section 6.1.6 has $r = 0.03\,\mathrm{m}$ and its kinematic and inertia parameters are shown in Table 6.3.

Under the assumption that the robot never slips or loses contact with the ground, the equation of motion of the unconstrained robot is

$$\begin{bmatrix} H_{\mathrm{rr}} & H_{\mathrm{r1}} & H_{\mathrm{r2}} \\ H_{1\mathrm{r}} & H_{11} & H_{12} \\ H_{2\mathrm{r}} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_\mathrm{r} \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_\mathrm{r} \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \tau_2 \end{bmatrix} \tag{6.1}$$

where H_{ij} are the elements of the joint-space inertia matrix, $\ddot{q}_i$ are the joint accelerations, C_i are the elements of the bias vector containing Coriolis, centrifugal and gravitational terms, and τ_2 is the joint torque acting on joint 2.

Since q_r is dependent on q_1, it is necessary to explicitly express the motion constraints [136] to apply the general balance controller described in Chapter 4. The explicit constraints map the independent variables' position, velocity and acceleration to the other joint variables. Having defined q as the vector of all joint variables, $\bar{q}$ at the vector of independent joint variables, and r as the radius of the contact surface, Equations 6.2 and 6.3 describe respectively velocity and acceleration constraints.

$$\dot{q} = G\dot{\bar{q}} \quad ; \quad \begin{bmatrix} \dot{q}_\mathrm{r} \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} -r & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{6.2}$$

$$\ddot{q} = G\ddot{\bar{q}} + \dot{G}\dot{\bar{q}} = G\ddot{\bar{q}} + g = G\ddot{\bar{q}} \tag{6.3}$$

since the matrix G is not time varying $\dot{G} = 0$ and therefore the vector $g = 0$ always.

Given the matrix G and vector g, the number of variables of the system can be reduced by applying the motion constraints. The constrained system is a function only of the independent variables $\bar{q}$ [136]. If the unconstrained virtual robot can be described by Equation 6.1, then the equation of motion for the constrained system is

$$H_G\ddot{\bar{q}} + C_G = u \tag{6.4}$$

where

$$H_G = G^T H G \quad , \quad C_G = G^T (C + Hg) \quad \text{and} \quad u = G^T \tau = [0 \ \tau_2]^\mathrm{T} \tag{6.5}$$

6.1.2 Tracking Error

This section demonstrates the need for a rolling-contact extension to the balance controller in [18] by showing in simulation that tracking accuracy declines substantially if a round foot is approximated with a point foot. The simulation was performed in Simulink R2020b using the integrator ode45 with relative tolerance set to 10^{-6} and other parameters at their default values.

The results are shown in Figure 6.3. In this graph, 'cmd' is the motion command signal for the actuated joint, and the job of the balance controller is to make this joint follow the command signal while simultaneously maintaining the robot's balance. The signal q_t is the theoretical response as explained in Section 4.1. It is the response that the balance controller

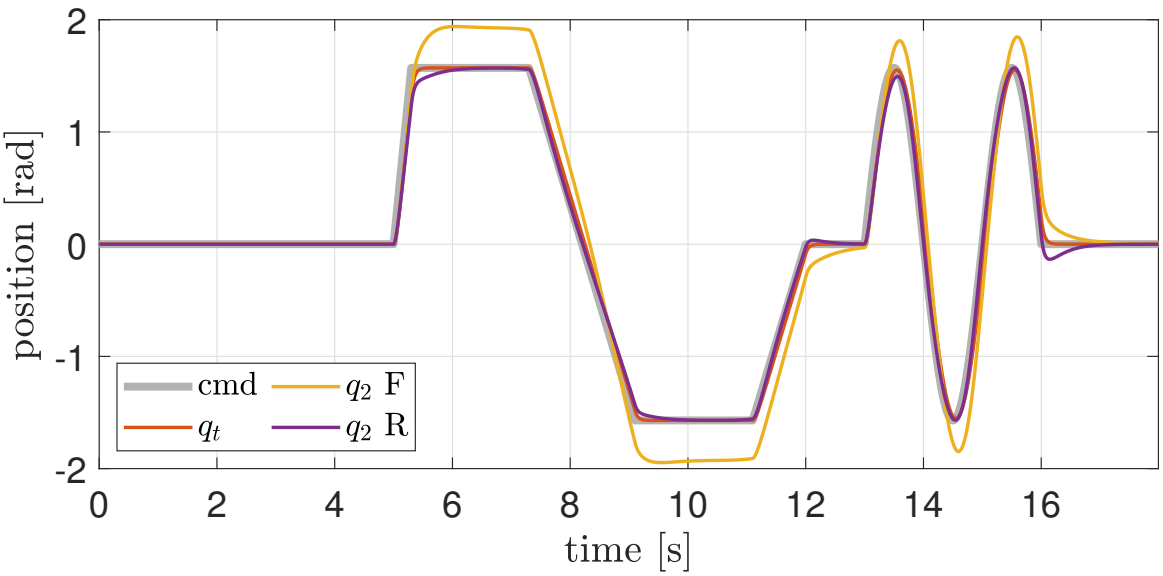


Figure 6.3 Tracking position for joint q_2 for a rolling double pendulum with $r = 2\,\text{cm}$. cmd is the desired trajectory, q_t is the theoretical response, $q_2\text{F}$ is the tracking of the balance controller assuming a fixed contact point, and $q_2\text{R}$ is the tracking of the balance controller assuming a rolling contact point.

is programmed to produce, and it is defined as

$$q_{\text{t}}(s) = \frac{1}{1 + s/(-\lambda_4)}\,\text{cmd}(s) = \frac{1}{1 + 0.05\,s}\,\text{cmd}(s) \tag{6.6}$$

where λ_4 is the fourth pole of the balance controller described in Chapter 4. Tracking error is therefore defined to be the difference between the theoretical and actual response. The two signals $q_2\text{F}$ and $q_2\text{R}$ are the actual responses of the point-foot and round-foot balance controllers, respectively. Although both controllers do a good job of maintaining the robot's balance, it can be seen that the tracking accuracy of the point-foot balance controller is substantially worse than that of the round-foot balance controller.

To obtain these results, the simulator models the dynamics using the parameters in Table 6.1 with $r = 2\,\text{cm}$; the round-foot controller uses the same parameters; and the point-foot controller uses the same parameters but with $r = 0$. Both controllers have their poles and zeros set as follows: one pole is set at $-1/T_{\text{c}}$ in its current configuration, as calculated by the controller using its own model; two more poles are set to a constant value equal to $-1/T_{\text{c}}^*$ where $T_{\text{c}}^* = T_{\text{c}}$ when the robot is in its upright vertical configuration; the two zeros are set to cancel these two poles; and the fourth pole, which is the one that determines the theoretical response, is set at $-20\,\text{rad/s}$ in both cases. Also, both controllers employ the

acausal filter described in Section 4.1 to filter the input signal q_c, which implements leaning in anticipation. Note that T_c depends on r, so the two controllers are using different values of T_c both in the feedback loop and in the acausal filter.

6.1.3 New Balance Controller

The planar balance controller presented in Chapter 4 and then experimentally demonstrated in [4], Section 4.5 and Chapter 5 assumes the robot to be balancing on a single fixed point in 2D or a knife edge in 3D. In [4], the robot behaves as a reaction wheel pendulum whose base is fixed to the ground via a revolute joint, and the balancing point coincides with the rotation axis of the joint. In Section 4.5 and Chapter 5, instead, the robot is an inverted pendulum which is not fixed to the ground. It has two contact points with the floor aligned with the upper link's rotation axis, which mimics a knife edge contact. In every case, the controller assumes the contact point is fixed while the robot balances.

In contrast, the rolling double pendulum's contact point does move, and, as the results of Section 6.1.2 have just shown, it is important that the balance controller takes this into account. We therefore proceed to develop an extension of the theory in [18] to take this movement into account. Let L be the angular momentum of the whole robot about the support point at the current instant. From elementary mechanics, $\dot{L}$ must equal the sum of the moments about the support point of each external force acting on the robot; but the only external force with a nonzero moment is gravity, and so we have

$$\dot{L} = -mg(c_x - q_\mathrm{r}) + m \begin{bmatrix} \dot{q}_\mathrm{r} \\ 0 \end{bmatrix} \times \begin{bmatrix} \dot{c}_x \\ \dot{c}_y \end{bmatrix} \tag{6.7}$$

where m is the mass of the robot and g is the magnitude of the gravitational acceleration; c_x is the position, and $\dot{c}_x$ and $\dot{c}_y$ are the velocity in the x and y direction of the CoM with respect to the reference frame fixed at the origin; q_r and $\dot{q}_\mathrm{r}$ are position and velocity in the x direction of the contact point. The first element of Equation 6.7 takes into account the position of the rolling contact, while the second its velocity. Since its contribution is two order of magnitude below the other elements, the second term of Equation 6.7 can be neglected. Therefore, Equation 6.7 (and its derivatives) can be simplified to

$$\dot{L} = -mg(c_x - q_\mathrm{r}) = -mg(c_x + rq_1) \tag{6.8}$$

$$\ddot{L} = -mg(\dot{c}_x - \dot{q}_\mathrm{r}) = -mg(\dot{c}_x + r\dot{q}_1) \tag{6.9}$$

$$\ddot{L} = -mg(\ddot{c}_x - \ddot{q}_\mathrm{r}) = -mg(\ddot{c}_x + r\ddot{q}_1) \tag{6.10}$$

where $\dot{c}_x$ and $\ddot{c}_x$ are the velocity and acceleration in the x direction of the centre of mass with respect to the reference frame fixed at the origin; and $\dot{q}_\mathrm{r}$ and $\ddot{q}_\mathrm{r}$ are the velocity and acceleration in the x direction of the contact point.

As the independent variable q_1 is an angle, and as the rolling contact is a rotation about the support point, we can say that the angular momentum of the constrained robot about the contact point is

$$L = p_1 = H_{G_{11}}\dot{q}_1 + H_{G_{12}}\dot{q}_2 \tag{6.11}$$

which follows from a special property of joint-space momentum that is proved in Appendix B of [18] and presented in Section 4.1.

All the considerations made in [18] about the relationship between joint position and velocity variables, and L, $\dot{L}$ and $\ddot{L}$ are valid also in case of rolling contact. Both L and $\ddot{L}$ depend linearly on the robot's velocity, implying that $L = \ddot{L} = 0$ is equivalent to $\dot{q}_1 = \dot{q}_2 = 0$ except in special circumstances when the robot is physically unable to balance (see Section 6.1.5). Also, $\dot{L}$ is a constant multiple of $(c_x + rq_1)$. As shown in [18] and [132], any controller that makes

$$L = 0\ ,\ \dot{L} = 0 \text{ and } \ddot{L} = 0 \tag{6.12}$$

will make the robot balance but will not drive the actuated joint to the desired position.

To allow the robot to track a desired trajectory for the actuated joint, we need an expression for $\dot{c}_x$ in terms of the joint velocity variables. This can be obtained by adding an extra fictitious prismatic joint acting in the x direction between the joint q_1 of the constrained robot and the ground. The extra joint is called joint 0 to preserve the numbering of the existing joints. The extra joint does not move, and therefore does not affect the dynamics. Its purpose is to increase the number of coefficients in the constrained equation of motion, which becomes

$$\begin{bmatrix} H_{G_{00}} & H_{G_{01}} & H_{G_{02}} \\ H_{G_{10}} & H_{G_{11}} & H_{G_{12}} \\ H_{G_{20}} & H_{G_{21}} & H_{G_{22}} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_{G_0} \\ C_{G_1} \\ C_{G_2} \end{bmatrix} = \begin{bmatrix} u_0 \\ 0 \\ u_2 \end{bmatrix} . \tag{6.13}$$

The special property of the joint-space momentum used before to define $p_1 = L$ can be used in this case to define p_0 as the linear momentum of the whole robot in the x direction, so

$$p_0 = m\dot{c}_x = H_{G_{01}}\dot{q}_1 + H_{G_{02}}\dot{q}_2 \ . \tag{6.14}$$

Combining Equations 6.9 and 6.14 we get

$$\ddot{L} = -g(p_0 + mr\dot{q}_1) = -g(H_{G_{01}} + mr)\dot{q}_1 - gH_{G_{02}}\dot{q}_2 \tag{6.15}$$

and combining Equations 6.13 and 6.10 we get

$$-\dddot{L}/g = u_0 + mr\ddot{q}_1 = (H_{G_{01}} + mr)\ddot{q}_1 + H_{G_{02}}\ddot{q}_2 + C_{G_0} \tag{6.16}$$

where the term $u_0 = m\ddot{c}_x$ is the x component of the ground reaction force acting on the robot. So, also in the case of rolling contact, there is an equation relating $\dddot{L}$ to the two independent joint accelerations and a pair of linear equations relating L and $\ddot{L}$ to the two independent joint velocities

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{G_{11}} & H_{G_{12}} \\ -g(H_{G_{01}} + mr) & -gH_{G_{02}} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{6.17}$$

To simplify the notation, we can then define a new matrix called H_R, defined as

$$H_{R_{ij}} = \begin{cases} (H_{G_{ij}} + mr) & \text{if } (i,j) = (1,0) \text{ or } (i,j) = (0,1) \\ H_{G_{ij}} & \text{otherwise .} \end{cases} \tag{6.18}$$

Equation 6.16 becomes

$$-\dddot{L}/g = H_{R_{01}}\ddot{q}_1 + H_{R_{02}}\ddot{q}_2 + C_{G_0} \tag{6.19}$$

and Equation 6.17 now reads

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{R_{11}} & H_{R_{12}} \\ -gH_{R_{01}} & -gH_{R_{02}} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{6.20}$$

Solving this equation for $\dot{q}_2$ gives

$$\dot{q}_2 = Y_1 L + Y_2 \ddot{L} \tag{6.21}$$

where

$$Y_1 = \frac{H_{R_{01}}}{D} \quad , \quad Y_2 = \frac{H_{R_{11}}}{gD} \tag{6.22}$$

and

$$D = H_{R_{01}} H_{R_{12}} - H_{R_{11}} H_{R_{02}} . \tag{6.23}$$

The obtained equations are the same as those described in [18] with the only difference that the H matrix is the one of the constrained rolling robot and not the one of the original robot. Nevertheless, it is possible to use the same control law described in Chapter 4, which is

$$\dddot{L} = k_{dd}\ddot{L} + k_d\dot{L} + k_L L + k_q(q_\mathrm{a} - q_d)\,, \tag{6.24}$$

where $q_\mathrm{a} = q_2$ is the only actuated joint and q_d is the input filtered command for such joint, and L, $\dot{L}$ and $\ddot{L}$ are defined respectively by Equations 6.11, 6.8 and 6.9. Also, the definition of the feedback gains obtained via pole placement remains unchanged

$$\begin{aligned} k_{dd} &= -a_3 \qquad & k_d &= -a_2 + a_0 Y_2/Y_1 \\ k_L &= -a_1 \qquad & k_q &= -a_0/Y_1\,, \end{aligned} \tag{6.25}$$

where

$$\begin{aligned} a_0 &= \lambda_1\lambda_2\lambda_3\lambda_4 \\ a_1 &= -\lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_2\lambda_4 - \lambda_1\lambda_3\lambda_4 - \lambda_2\lambda_3\lambda_4 \\ a_2 &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 + \lambda_2\lambda_4 + \lambda_3\lambda_4 \\ a_3 &= -\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4\,, \end{aligned} \tag{6.26}$$

and $\lambda_1, \ldots, \lambda_4$ are the chosen values of the poles.

The controller's output must be either an acceleration or a torque for the actuated joint; that is, either $\ddot{q}_2$ or u_2. Combining Equations 6.13, 6.18 and 6.19 we obtain

$$\begin{bmatrix} 0 & H_{R_{01}} & H_{R_{02}} \\ 0 & H_{R_{11}} & H_{R_{12}} \\ -1 & H_{R_{21}} & H_{R_{22}} \end{bmatrix} \begin{bmatrix} u_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -\ddot{L}/g - C_{G_0} \\ -C_{G_1} \\ -C_{G_2} \end{bmatrix} \tag{6.27}$$

which can be solved for both $\ddot{q}_2$ and u_2; and $u_2 = \tau_2$ as shown in Equation 6.5.

6.1.4 Simulation Experiments

In this section are reported the results of simulation experiments where the robot starts in a vertical position ($q_\mathrm{r} = q_1 = q_2 = 0$) and then tracks a desired trajectory for joint 2 while balancing on a rolling contact. The robot model, the initial conditions, the zeros and the poles of the balance controller are the same as Section 6.1.2. The experiment is performed every time with a different radius of the contact surface. The initially tested radii are 0, 2, 4, 6, 8, and 10 cm. Although the reference trajectory is the same for all the experiments, the filtered command signal produced by the acausal filter is specific for each experiment. This

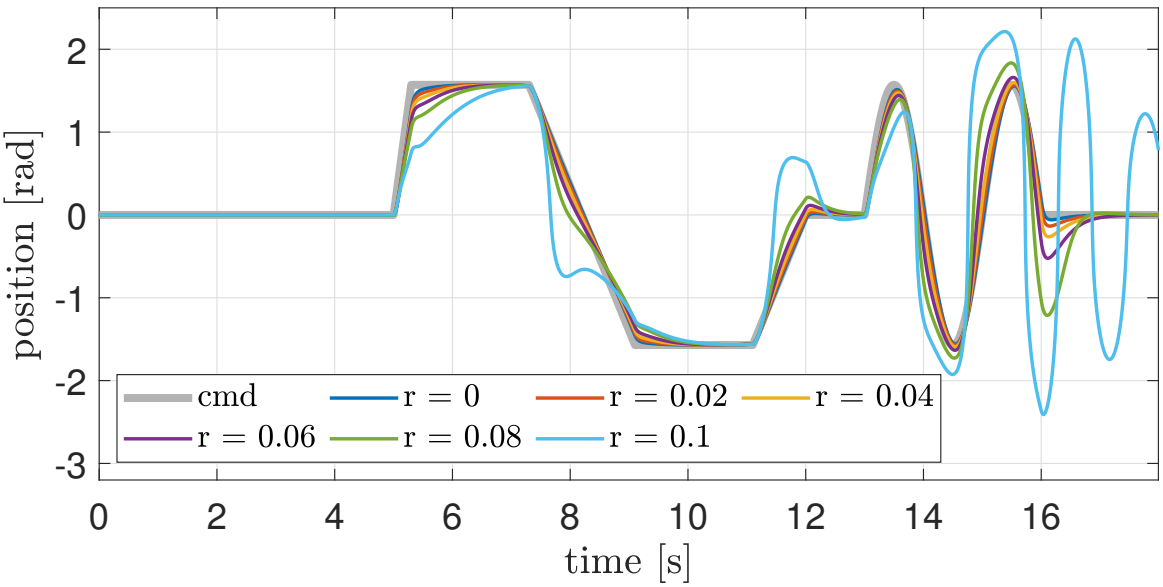


Figure 6.4 Tracking position for joint 2 with varying radius [m]

is because the time constant of the acausal filter is T_c, and T_c varies with r. Some example values of T_c are reported in Table 6.2.

Figure 6.4 reports the tracking position of joint 2. The best performance is obtained when the radius is zero, meaning that the robot is balancing on a fixed point and not on a rolling contact. The tracking accuracy with $r = 0.02\,\text{m}$ is almost as good as with $r = 0\,\text{m}$ and then gradually decreases as r increases, until $r = 0.08\,\text{m}$. The increment of the radius to $r = 0.1\,\text{m}$ causes a large reduction in the tracking performance. The reason behind this behaviour is the topic of Section 6.1.5.

6.1.5 Linear Velocity Gain

A robot's performance at balancing is limited by its physical ability to balance, which is a property of the robot itself, not the control system. Linear velocity gain, as defined in [133], provides a quantitative measure of a robot's physical ability to balance. It is defined as the ratio of the change in the horizontal velocity of the CoM to the change in velocity of the joint used to balance the robot when both changes are caused by an impulse at that joint. For the robot used in these experiments, the velocity gain is

$$G_v = \frac{\Delta \dot{c}_x}{\Delta \dot{q}_2} = \frac{-D}{mH_{R_{11}}} \tag{6.28}$$

r [m]	0	0.02	0.04	0.06	0.08	0.10	0.20	0.225	0.25
G_v[m]	0.0218	0.0177	0.0135	0.0094	0.0053	0.0011	-0.0196	-0.0247	-0.0299
T_c [s]	0.1899	0.1965	0.2040	0.2123	0.2216	0.2324	0.3277	0.3777	0.4609

Table 6.2 Velocity gain G_v and toppling time constant T_c with the robot in its vertical position according to the radius value r.

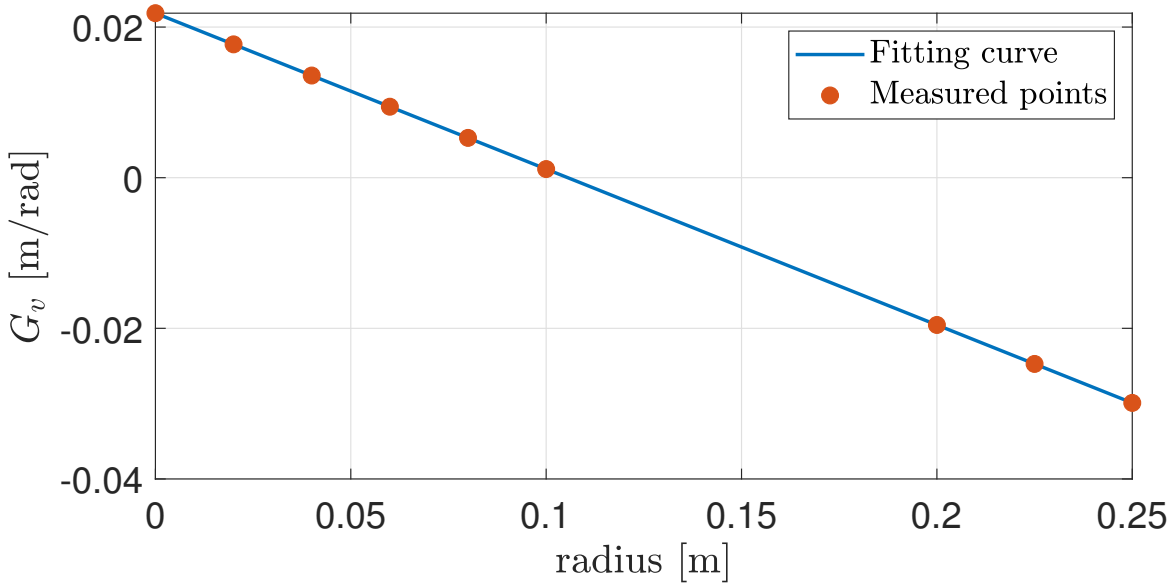


Figure 6.5 Velocity gain with the robot in vertical position ($q_r = q_1 = q_2 = 0$), where $G_v = -0.207\,r + 0.0218$

where D and $H_{R_{11}}$ are respectively described in Equations 6.23 and 6.18, and m is the total mass of the robot. A robot's physical ability to balance is proportional to $|G_v|$, and it is physically impossible for a robot to balance in any configuration where $G_v = 0$.

The performance deterioration shown in Section 6.1.4 is directly related to the reduction of the magnitude of the linear velocity gain G_v with increasing radius. Figure 6.5 and Table 6.2 clearly show a negative linear relationship between the velocity gain and the radius of the rolling contact. With a rolling double pendulum with dynamic parameters described in Table 6.1, it is possible to calculate the velocity gain as

$$G_v = -0.207\,r + 0.0218 \tag{6.29}$$

when the robot is in a vertical position, $q_r = q_1 = q_2 = 0$. As the radius increases, the velocity gain decreases, reducing the robot's physical ability to balance. A radius of 0.1055 m makes the velocity gain equal to zero, making the robot physically unable to balance.

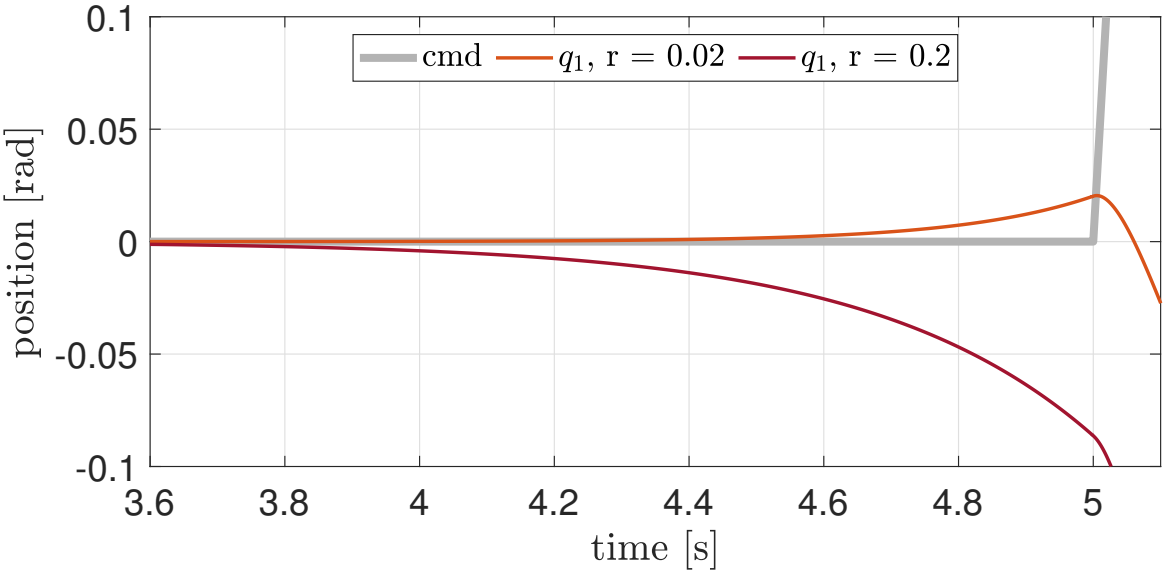


Figure 6.6 Leaning in anticipation effects in case of G_v positive $r = 0.02$ and negative $r = 0.2$.

Any robot requires the magnitude of the velocity gain to be as high as possible to balance better, being physically unable to balance when $G_v = 0$. This implies $D = 0$ via Equation 6.28, and, as a consequence, the impossibility of evaluating the gains Y_1 and Y_2 with Equation 6.22. Figure 6.5 shows that the linear relation between the radius and the velocity gain continues and becomes negative with the increasing radius. The change of sign in the velocity gain implies that the controller moves the upper link in the opposite direction to maintain the balance with these new radii compared to the radii tested in Section 6.1.4. This effect can be observed in Figure 6.6, where the robot leans in anticipation in the opposite direction to compensate for the disturbance introduced by tracking the same trajectory. If the linear velocity gain is positive, there is a positive variation of q_1, negative otherwise.

The three red dots in the bottom right corner of Figure 6.5 represent three radii of the rolling contact that makes the robot controllable even with a negative linear velocity gain. The tested radii are 20, 22.5, and 25 cm, with the last value implying that the lower link's centre of mass coincides with the rolling contact's rotation axis. The simulation results are shown in Figure 6.7. The tracking is accurate, with the accuracy increasing with the length of the radius. Such behaviour is due to the increase in the magnitude of the velocity gain, as shown in Table 6.2. This data demonstrates that it is the decrease in the robot's physical ability to balance, rather than some defect in the balance controller, that accounts for the decline in tracking accuracy observed in Figure 6.4.

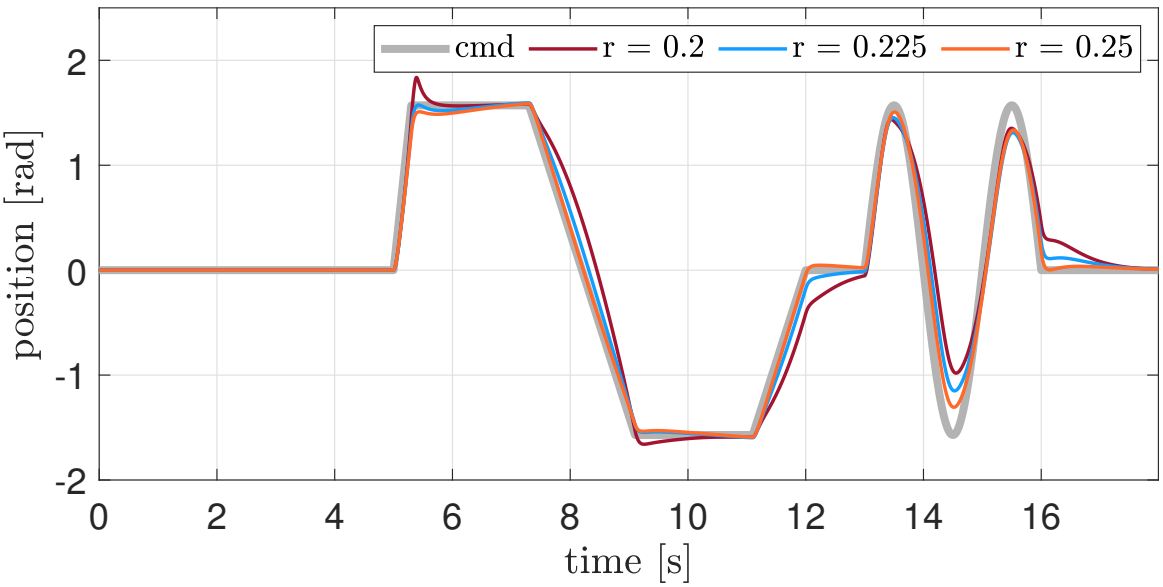


Figure 6.7 Tracking position for joint 2 with varying radius [m]

It is worth noting that the velocity gain of the robot changes with the configuration of the robot. This means it is not constant throughout the experiment, as shown in Figure 6.8. A robot with a rolling contact with a radius between 10 and 20 cm is physically unable to track the desired trajectory since the velocity gain will cross the zero line while moving. This is because the velocity gain becomes more positive when the upper link has an angle with respect to the lower link, meaning that it increases the performance of a robot with a radius smaller than 10 cm and deteriorates those of a robot with a radius above the threshold. A radius of 20 cm is the smallest one that allows the robot to safely track the desired trajectory.

6.1.6 Physical Experiment

This section describes an experiment in which the tracking performance of the general implementation of the balancing controller is compared with the new rolling-contact balance controller on a real robot. The obtained results are presented together with the simulated and theoretical results.

Robot Description

The robot used for this experiment is shown in Figure 6.1, and it is essentially the underactuated inverted double pendulum described in Chapter 5. Electronics and actuator are the same, but minor mechanical changes are present, which are a gear ratio $G = 12$ and a bigger

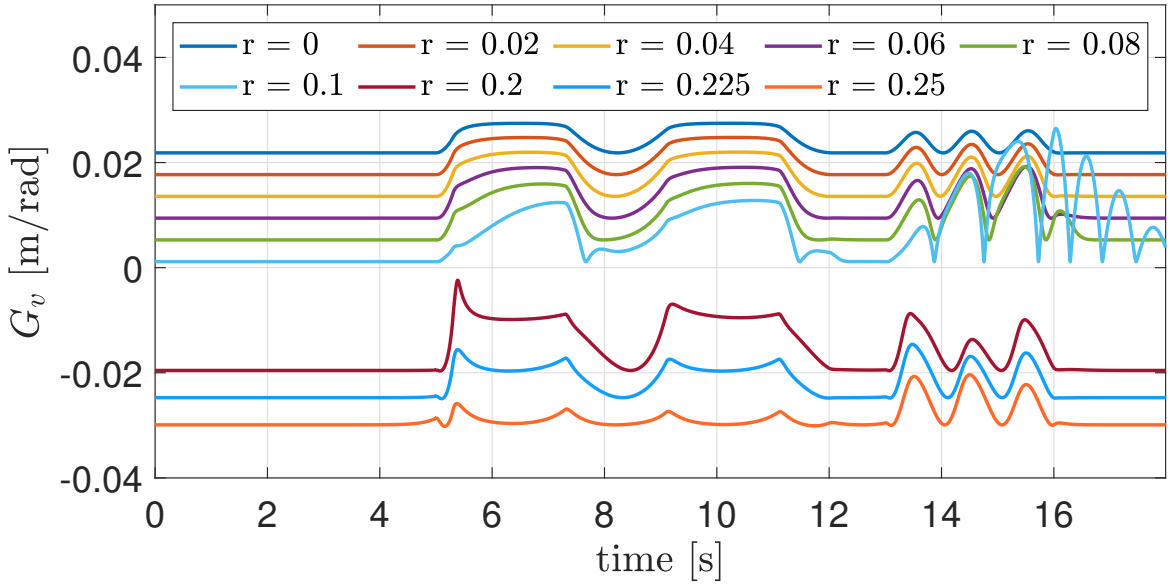


Figure 6.8 Robot velocity gain during the experiment with varying radius [m].

Body i	m [kg]	c_x [m]	c_y [m]	L [m]	I [kg m^2]
1	0	0	0	0.03	0
2	1.41	0	0.22	0.25	0.0083
3	0.44	0	0.105	0.3	0.0078

Table 6.3 Dynamic and kinematic parameters of the robot used for the physical experiment modelled as in Figure 6.2. Considering body i: m is the mass, c_x and c_y are the coordinates of the centre of mass with respect to the body reference frame, L is the length of the body, I is the inertia of the body at the centre of mass.

drive wheel to reduce the wear effects experienced in Chapter 5. The main difference is the contact point with the ground. In Chapter 5, the robot balances on a line segment formed by two-point contacts at the tip of its two sharp stilts and the ground. In this work, instead, the tip of the stilts is a round surface with a radius $r = 0.03$ m. The choice of the radius size is not casual. In fact, it is the same radius that Skippy will have in its foot.

With the joint variables as in Figure 6.2, where q_r describes the horizontal motion of the rolling contact, q_1 is the angle of the lower body with respect to the vertical, and q_2 is the angle of the upper link with respect to the lower, the kinematic and dynamic parameters of both the physical robot and its simulated model are reported in Table 6.3.

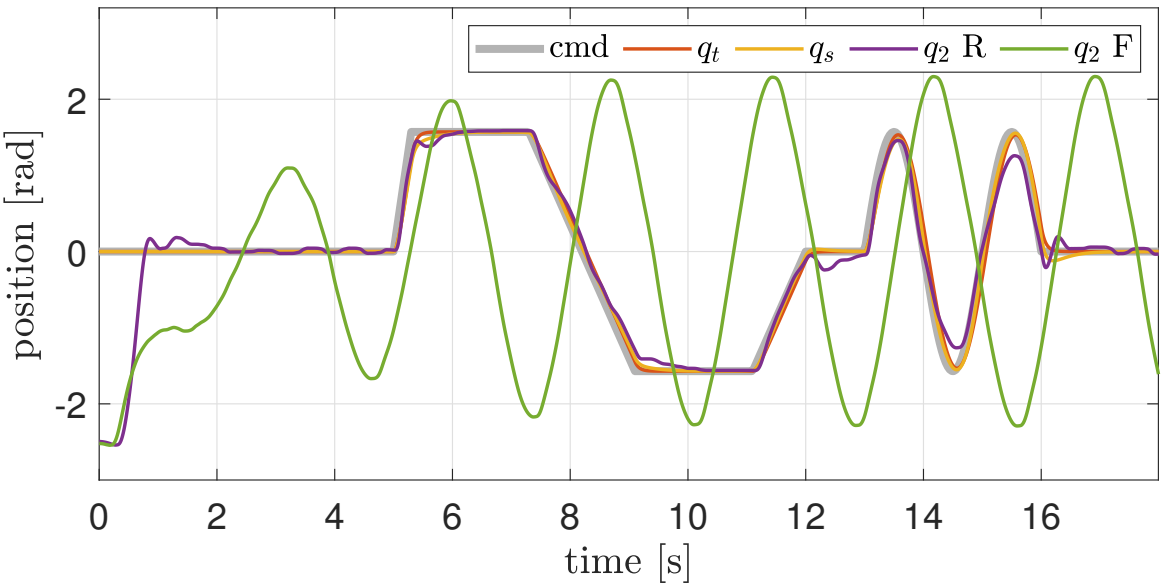


Figure 6.9 Tracking position for joint 2. cmd is the desired trajectory, q_t is the theoretical response, q_s is the simulated output of the rolling balance controller, q_2F is the tracking of the balance controller assuming a fixed contact point, and q_2R is the tracking of the balance controller assuming a rolling contact point. q_2F and q_2R refer to the physical experiment.

Experiment Description

The experiment starts with the robot in a resting position, with the upper link touching the ground. Then the robot balances itself in less than 2.5 seconds and starts tracking the same trajectory as Sections 6.1.2 and 6.1.4. The trajectory consists of ramps and a sine wave with an amplitude of $\pi/2$. The controller has its poles and zeros set with the same strategy used in simulation but with different values. One pole is set at $-1/T_c$, where T_c is the time constant of toppling of the robot in its current configuration, as calculated by the controller using its own model; two more poles are set to a constant value equal to $-1/T_c^* = -1/0.192$, where $T_c^* = T_c$ in the robot's upright vertical configuration; the two zeros are set to cancel these two poles; and the fourth pole, which is the one that determines the theoretical response, set at $-14\,\mathrm{rad/s}$. The input signal is filtered with the acausal filter described in [18] with a time constant T_f equal to the robot's time constant of toppling in the upright vertical configuration.

Experimental Results

The experimental results are shown in Figure 6.9. The first experiment (green line) tests the controller used in Chapter 5 with the new mechanical setup. The robot can barely balance itself, and it is completely unable to track the desired trajectory. Then the new controller is

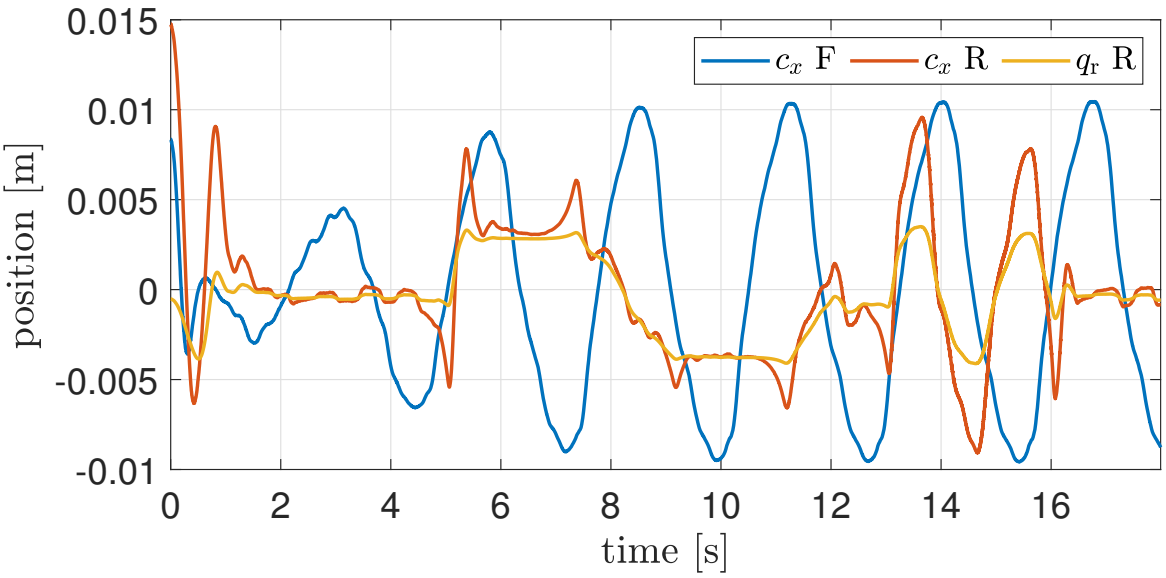


Figure 6.10 Estimated position of the centre of mass throughout the experiment. c_x F is the CoM position when the balance controller assumes a fixed contact point, and c_x R is when the balance controller assumes a rolling contact point.

tested (purple line), and now not only can the robot balance itself in less than 2.5 seconds, but it can also accurately track the desired trajectory while balancing (see accompanying video of [135]). The reason for this behaviour can be found in Figure 6.10, where it can be seen that while balancing with the new controller, the position of the centre of mass is very close to the value of q_r. The CoM position c_x is not directly measured, but it is evaluated in real-time by the robot control unit from the robot's kinematic model given the measured joint angle values. In this situation, the centre of mass is aligned with the contact point; hence the robot is balanced. The old balance controller, though, is unaware that the contact point with the ground has moved and tries to move the centre of mass to be above where it thinks that the contact point is. As a consequence of this motion, the robot starts oscillating, making this controller unusable for balancing this type of robot. Figure 6.10 also shows the effect of the leaning in anticipation caused by the acausal filter, with the robot leaning in the opposite direction to compensate for future disturbance introduced by the trajectory tracking. For example, before the first ramp at $t = 5\,\mathrm{s}$, the centre of mass of the robot controlled with the rolling balance controller, c_xR line of Figure 6.10, moves in the opposite direction before the ramp begins. Similar movements can be seen just before the second and third ramps.

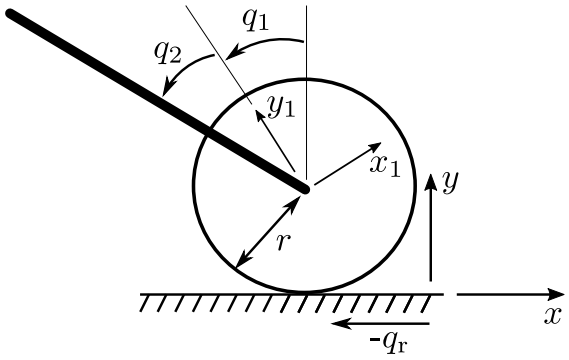


Figure 6.11 Schematic model of the robot balancing on a wheel

6.1.7 Balancing on a Wheel

The results obtained in Section 6.1.4 suggest that this controller can also be used to balance a stick on a wheel, as shown in Figure 6.11. With respect to the dynamic parameters described in Table 6.1, only some minor changes to the definition of Body 1 have to be made, which are $L_1 = 0$ and $c_{y_1} = 0$. The new robot is essentially made of a wheel of radius $r = 0.2\,\mathrm{m}$ and a stick connected to the centre of the wheel by means of an actuated revolute joint. The wheel can rotate without slipping; hence the robot can move horizontally in the plane.

The modelling technique used in Section 6.1.4 is also valid for this robot. Referring to Figure 6.11, q_r is the rolling distance, q_1 is the angle of the coordinate frame of the wheel $x_1 - y_1$ with respect to the vertical, and q_2 is the angle of the actuated joint with respect to the coordinate frame of the wheel. The body coordinate frame $x_1 - y_1$ is located at the centre of the wheel, and it rotates with the wheel itself; the axis y_1 is aligned with the vertical when $q_1 = 0$.

The system can track the desired trajectory for joint 2 very efficiently due to the high-velocity gain $G_v = -0.031$. In a balanced configuration, $q_1 + q_2 = 0$, so by making q_2 track a desired trajectory, q_1 is essentially equal to $-q_2$. As a consequence, the robot rolls significantly while tracking the desired trajectory (which is the same as Section 6.1.4) as shown in Figure 6.12 and in the accompanying video of [135].

The controller can be adapted to make the robot travel at a commanded velocity. All that needs to be done is to set the position command for q_2 equal to the integral of $1/r$ times the velocity command. This works because the stick will be upright when the robot is travelling at a constant velocity, so $\dot{q}_1 = -\dot{q}_2$, hence $\dot{q}_r = r\dot{q}_2$. The example reported in Figure 6.13, shows how the stick's lean angle $(q_1 + q_2)$ varies while the robot travels at a commanded velocity.

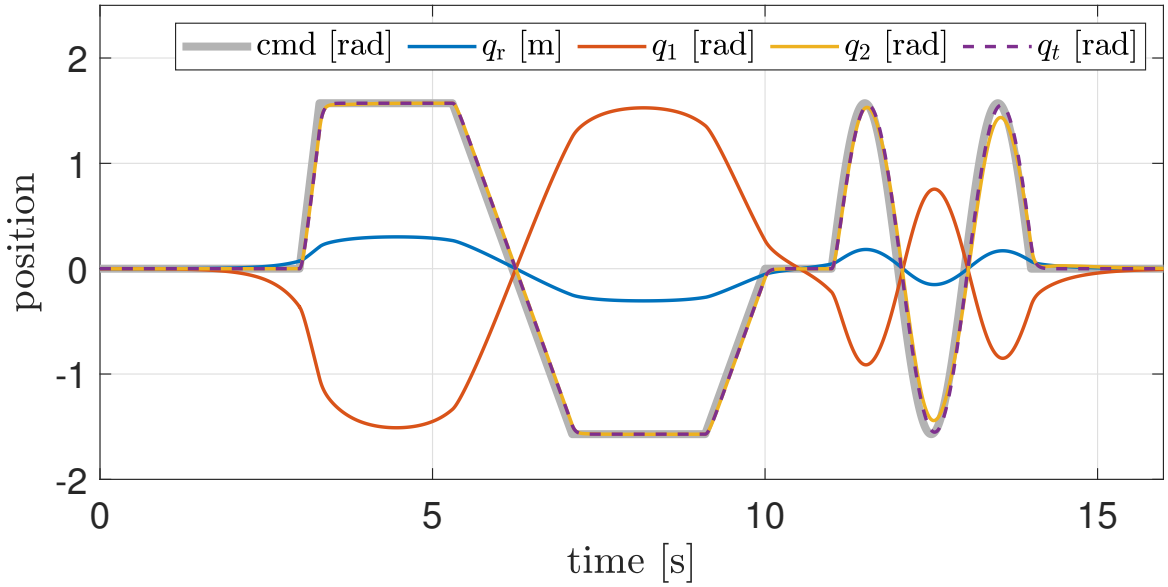


Figure 6.12 Tracking position for joint 2 with radius of the wheel $r = 0.2$ m. cmd is the desired trajectory, q_t is the theoretical response, q_2 is the tracking of the balance controller, q_r and q_1 are the other two joints positions.

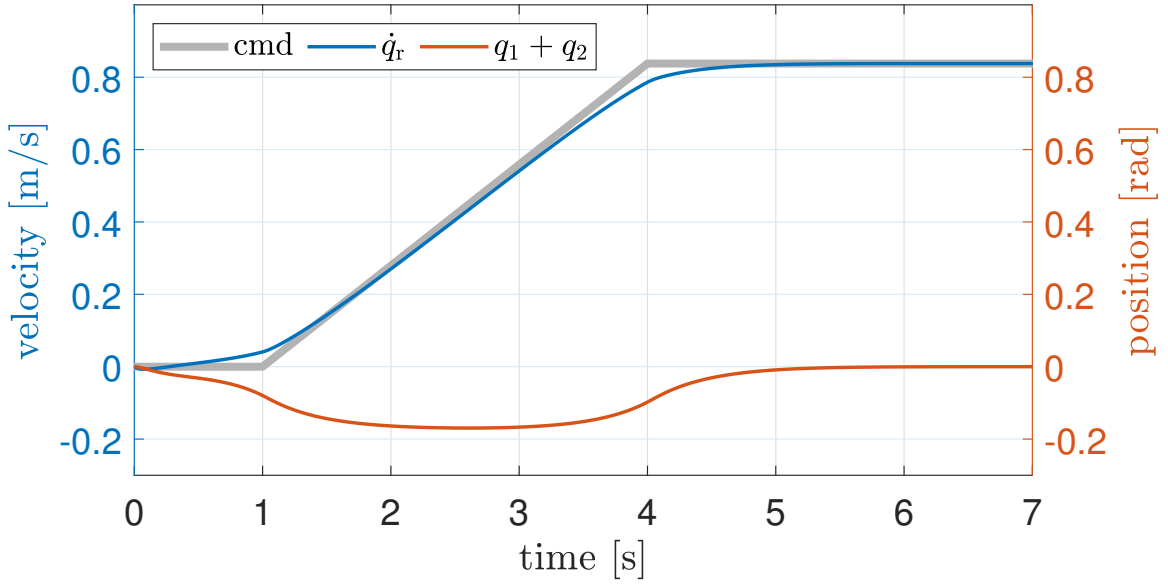
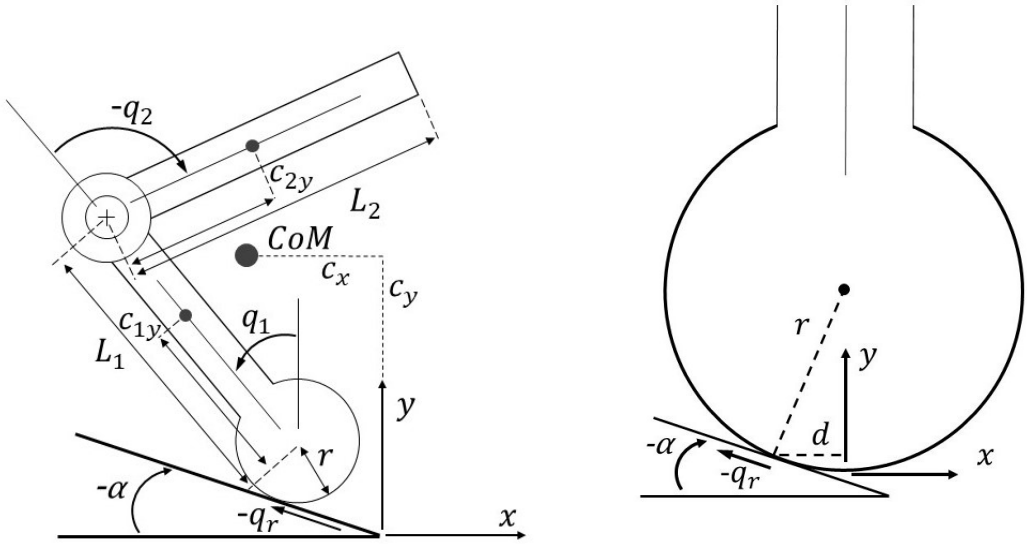


Figure 6.13 Tracking velocity for joint q_r. cmd is the desired travelling velocity, $\dot{q}_r$ is the travelling velocity, and $(q_1 + q_2)$ is the stick lean angle.



(a) (b)

Figure 6.14 (a) Schematic model of the rolling double pendulum balancing on a slope; (b) detail showing that the location of the global reference frame depends and the contact point when $q_1 = q_r = 0$. $d = r\sin(\alpha)$ is the horizontal offset.

6.2 Balancing on a Slope

This section extends the work presented in Section 6.1 by removing the constraint of the robot balancing on a flat horizontal surface and replacing it with a slope whose magnitude is known in advance. This section is organized as follows. First Section 6.2.2 shows the need of an extension of the Rolling Contact balance controller. Then Section 6.2.3 presents the new 'Rolling Slope' balance controller and Section 6.2.4 displays the results of simulated experiments.

6.2.1 Model Description

The robot used in this section is shown in Figure 6.14. It is modelled in the same way as in Section 6.1 with the only difference being that the prismatic joint modelling the ground contact q_r moves on a slope with an angle α with respect to the horizontal; there is no variation in the definition of q_1 and q_2, nor in the definition of the matrix G and the vector g. The location of the global reference frame varies depending on the slope and coincides with the contact point when $q_1 = q_r = 0$, as shown in Figure 6.14b.

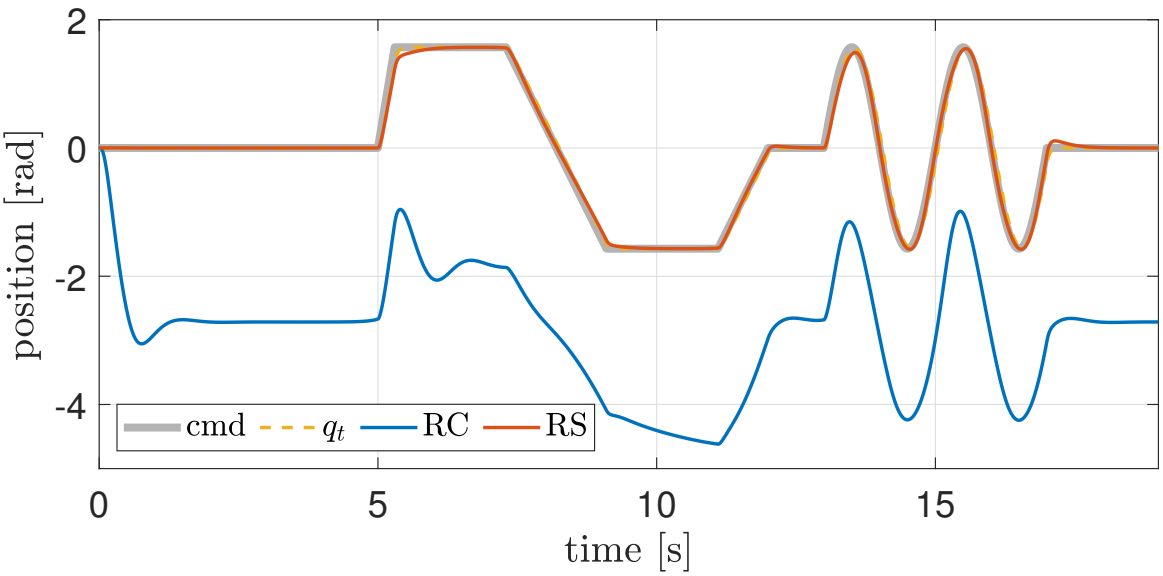


Figure 6.15 Tracking position for joint 2 for a rolling double pendulum balancing on a 22.5° slope. cmd is the desired trajectory, q_t is the theoretical response, RC is the tracking of the balance controller assuming a flat horizontal surface, and RS is the tracking of the balance controller assuming the robot balancing on a known slope.

6.2.2 Tracking Error

This section demonstrates the need of an extension to the Rolling Contact balance controller when the robot balances on a slope. The need of a new controller is showed in simulation, where the tracking accuracy decreases in the presence of a slope. The simulation was performed in Simulink R2020b using the integrator ode45 with relative tolerance set to 10^{-6} and other parameters at their default values. The reference trajectory and its filter, and the balance controller are configured in the same way as in Section 6.1.4. The time constant of the acausal filter T_f is constant and it is equal to the value of the constant of toppling of the robot in the balanced vertical configuration ($q_2 = 0$, and q_1 varies with the angle α). In this experiment the slope has an angle $\alpha = -22.5°$ and $T_\mathrm{f} = 0.187\,\mathrm{s}$. The controller has its poles and zeros set as follows: one pole is set at $-1/T_\mathrm{c}$ in its current configuration, two more poles are set to a constant value equal to $-1/T_\mathrm{c}^*$, where $T_\mathrm{c}^* = T_\mathrm{c}$ in the robot's vertical configuration; the two zeros are set to cancel these two poles; and the fourth pole, which determines the theoretical response, is set to $-20\,\mathrm{rad/s}$. The results are shown in Figure 6.15. The 'Rolling Contact' controller is able to balance the robot but the tracking is not accurate with a slope angle $\alpha = -\pi/8\,\mathrm{rad} = -22.5°$.

α [rad]	$-\pi/4$	$-\pi/8$	0	$\pi/4$	$\pi/8$
q_1 [rad]	0.0842	0.0456	0.000	-0.0456	-0.0842
T_c [s]	0.187	0.191	0.192	0.191	0.197

Table 6.4 Leaning angle q_1 and toppling time constant T_c with the robot balancing with $q_2 = 0$ according to the slope value α.

6.2.3 Rolling Slope Controller

The rolling double pendulum's contact point moves while balancing, and when the balancing surface is not horizontal, it moves both horizontally and vertically. As a consequence, the horizontal distance of the centre of mass with respect to the contact point is affected by the angle of the slope. This definition takes into account the position of the rolling contact point but not its velocity, as in Section 6.1.3. Also in this case the only external force acting on the robot is the one due to gravity, therefore the derivative of the angular momentum and its first two derivatives are

$$\dot{L} = -mg(c_x - q_r\cos(\alpha)) = -mg(c_x + rq_1\cos(\alpha)) \tag{6.30}$$

$$\ddot{L} = -mg(\dot{c}_x - \dot{q}_r\cos(\alpha)) = -mg(\dot{c}_x + r\dot{q}_1\cos(\alpha)) \tag{6.31}$$

$$\dddot{L} = -mg(\ddot{c}_x - \ddot{q}_r\cos(\alpha)) = -mg(\ddot{c}_x + r\ddot{q}_1\cos(\alpha)) \tag{6.32}$$

All the dynamic considerations made in Section 6.1.3 are still valid and after replacing Equations 6.9 and 6.10 with Equations 6.31 and 6.32 we can now define a new matrix H_α as

$$H_{\alpha_{ij}} = \begin{cases} (H_{G_{ij}} + mr\cos(\alpha)) & \text{if } (i,j) = (1,0) \text{ or } (i,j) = (0,1) \\ H_{G_{ij}} & \text{otherwise .} \end{cases} \tag{6.33}$$

All the other considerations made for balancing a rolling inverted pendulum on a flat horizontal surface are valid also in case of balancing on a slope, given that the matrix H_R of Section 6.1.3 is replaced with the newly defined matrix H_α. The control law and its gains, and the output mapping are unchanged.

6.2.4 Simulation Experiments

This section reports the results of simulation experiments. The experiments start with the robot in an upright and balanced position, where $q_2 = 0$, and q_1 and q_r depend on the angle of the slope. Then the robot tracks a desired trajectory for joint 2 while balancing on a

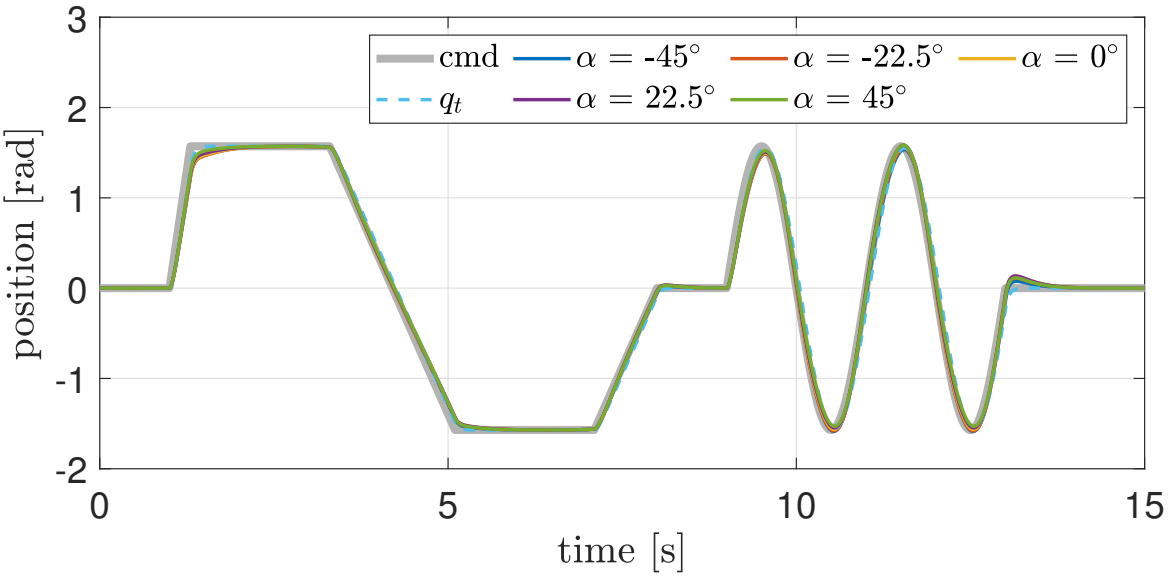


Figure 6.16 Tracking position for joint 2 with varying slope.

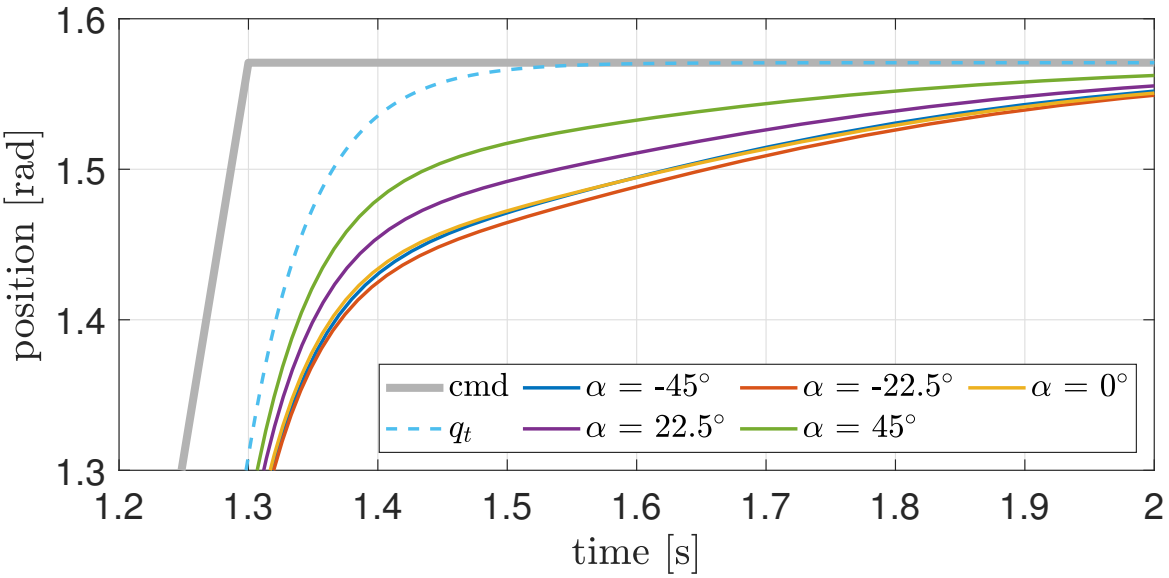


Figure 6.17 Detail of tracking position for joint 2 with varying slope.

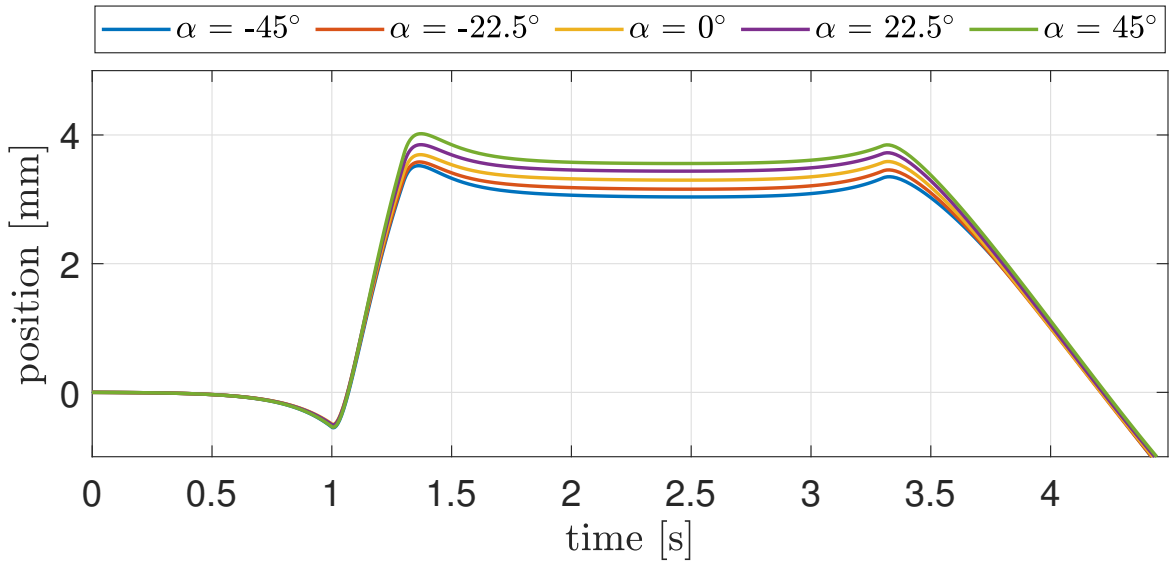


Figure 6.18 Motion of the rolling contact q_r with varying slope.

slope. The experiment is performed every time with a different angle of the slope, ranging from $-\pi/4$ to $\pi/4$. Although the reference trajectory is the same for all the experiments, the filtered command signal produced by the acausal filter is specific for each experiment. This is because the time constant of the acausal filter it T_c, and T_c varies with α, as shown in Table 6.4 To see if the tracking accuracy is affected when the upper link goes uphill or downhill, the same trajectory has been tested both with a positive and a negative slope angle α.

The tracking accuracy is shown in Figure 6.16. It is slightly affected by the slope angle, with better performance when the angle q_2 has the same sign as the slope, as shown in Figure 6.17. The author attributes this behaviour to the simplifying assumption introduced in Equation 6.7. When balancing on a slope, the y component of the CoM position and velocity are affected by the slope angle. Nevertheless, the inaccuracy introduced in the estimation of $\dot{L}$ is so minimal that the feedback control loop compensates for it without a significant variation in the tracking performance (see Figure 6.18).

6.3 Conclusion

This chapter presented an extension to Featherstone's 2D balance controller in which the point-foot assumption is replaced with a circular-foot assumption. This extension increases the generality of the controller so that it can be applied to legged robots with round feet and to robots that balance and travel on a pair of wheels (i.e., Segway-like robots).

Section 6.1 first demonstrated the need for such an extension, then developed the new theory, and then demonstrated the effectiveness of the new controller both in simulation and on a real robot. The newly developed controller has then been extended, and tested in simulation, to the case of a rolling double pendulum balancing on a slope in Section 6.2. The presence of a slope doesn't affect robots balancing on a pointed foot because the foot doesn't move while balancing. On the other hand, the slope affects the motion of robots balancing on a circular foot since the contact point with the ground moves while the robot balances.

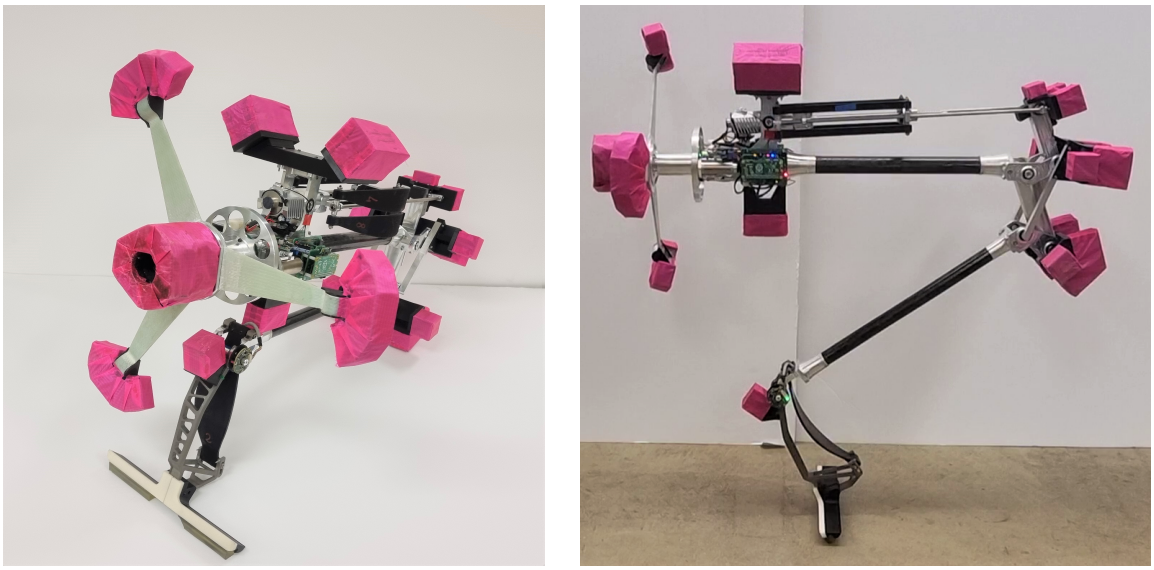
Chapter 7

Balancing and Hopping on a Springy Leg

This chapter presents the first implementation of the general balance controller on a robot balancing and hopping on a springy leg. The robot used for this experiment is Skippy (see Figure 7.1), fitted with a knife-edge shoe that constrains its motions to a vertical plane. In this configuration, the robot behaves as an inverted double pendulum. The difference with Chapter 5, apart from the springy leg, is that now the motor has to compensate for the weight of the upper link continuously. Due to mechanical constraints, there is not a balanced position where the robot's structure compensates for gravity (e.g. $q_1 = q_2 = 0$ in Chapter 5).

7.1 The Robot

The robot in this configuration follows the same modelling strategy proposed by Gkikakis in [1 § 3]. His study represents a preliminary design work for Skippy, particularly the co-optimization of the various mechanical design parameters and the desired hopping and balancing behaviours. Skippy can be modelled in this configuration by 10 bodies and 10 joints as in Figure 7.2 and Table 7.1, $q = [q_1, ..., q_{10}]^{\mathrm{T}}$. The first two joint variables, q_1 and q_2, are the contact point's x and y coordinates with respect to the ground, and are used only in simulation ($q_1 = q_2 = 0$ on the real robot). q_3 describes the rolling contact motion of the foot as $q_3 = -\mathrm{rtoe}\, q_4$ where rtoe is the radius of the foot as in Figure 7.2, and the rolling contact is modelled in the same way as in Section 6.1.1. q_4 is the angle of the foot with respect to the vertical and q_5 is the ankle-leg angle. q_6 is the leg-lever angle and q_7 is the lever-torso angle. q_8 is the torso-follower angle and q_9 is the torso-rocker angle. q_{10} is the nut displacement (towards the motor). Bodies 1 to 3 are fictitious massless bodies that allow the foot to lift off the ground and roll in simulation. $q_6 + q_7 = q_{67}$ is called the hip angle, representing the angle between the carbon fibre tube of the torso and the one of the leg. Figure 7.2 shows Skippy's



(a) (b)

Figure 7.1 Skippy with its springy leg and knife-edge shoe at rest (a) and balancing (b).

body and all the joint variables measured in the real robot. The connection between bodies 6 and 10 is made by springs, called 'Main' springs (described later in this chapter), allowing a greater thrust (than what is achievable with rigid links) at the price of a more complex control. In the simplified representation of Figure 7.2 the Main springs act in tension, when in reality the act in compression (see Appendix B Figure B.2 for a complete kinematic diagram).

Two simplifying assumptions are made in this chapter: rigid links are in place of the Main springs, hence q_{67} and q_{10} measurements agree, and the radius of the tip of the foot (called toe) rtoe $= 0$, hence the robot balances on a fixed point without rolling and $q_3 = 0$ always.

The rest of this section describes the physical bodies of Skippy and all the real components. Bodies 1 to 3 are not presented, since they are fictitious massless bodies used only in simulation. Section 7.2 presents the kinematic model.

7.1.1 Foot

It corresponds to body 4 of the kinematic model. It is made of a 3D printed titanium alloy (Ti6AL4V), and it is connected to the leg through a spring-loaded revolute joint called the 'Ankle', Figure 7.3. The spring is a fiber glass regressive leaf spring and it is called 'Ankle' spring, more details in Section 7.1.10. A 3D printed ABS 'Shoe' is added to the titanium piece to constrain its motion in 2D. The shoe makes the contact point a sharp knife edge that

Bodies	Joint Variable
B_1 – fictitious massless body	q_1 – x coordinate of the contact point
B_2 – fictitious massless body	q_2 – y coordinate of the contact point
B_3 – fictitious massless body	q_3 – x coordinate of the rolling contact
B_4 – foot	q_4 – foot angle
B_5 – leg	q_5 – ankle angle
B_6 – lever	q_6 – leg-lever angle
B_7 – torso	q_7 – lever-torso angle
B_8 – follower	q_8 – torso-follower angle
B_9 – rocker+main motor+screw rod	q_9 – torso-rocker angle
B_{10} – nut+push rod+main spring+ coupler	q_{10} – nut displacement (towards motor)

Table 7.1 Left: list of bodies; right: list of joint variables.

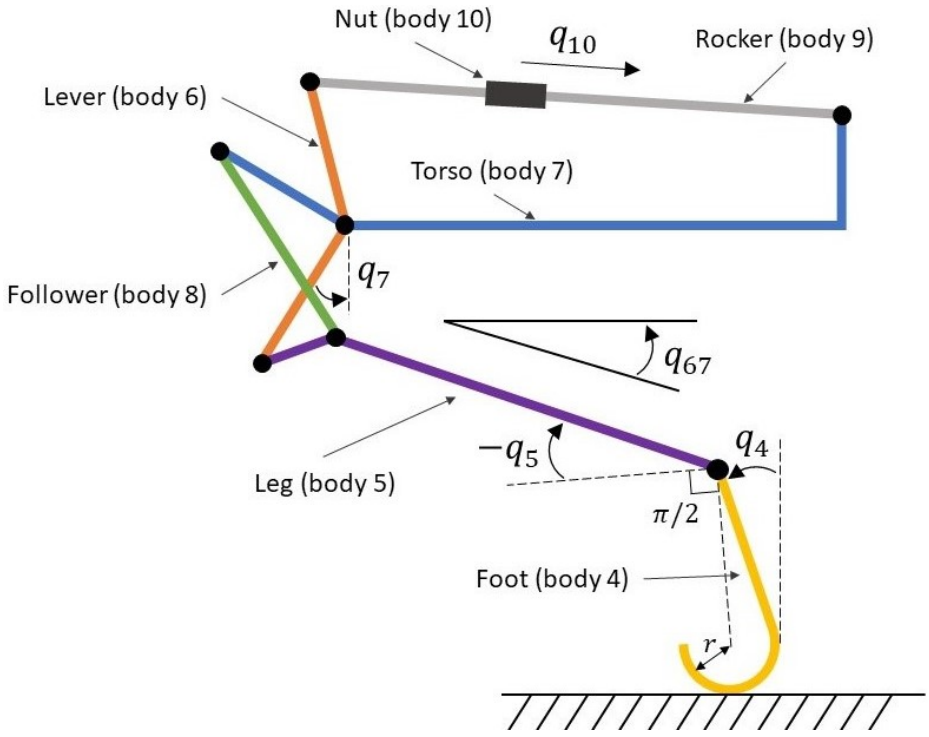
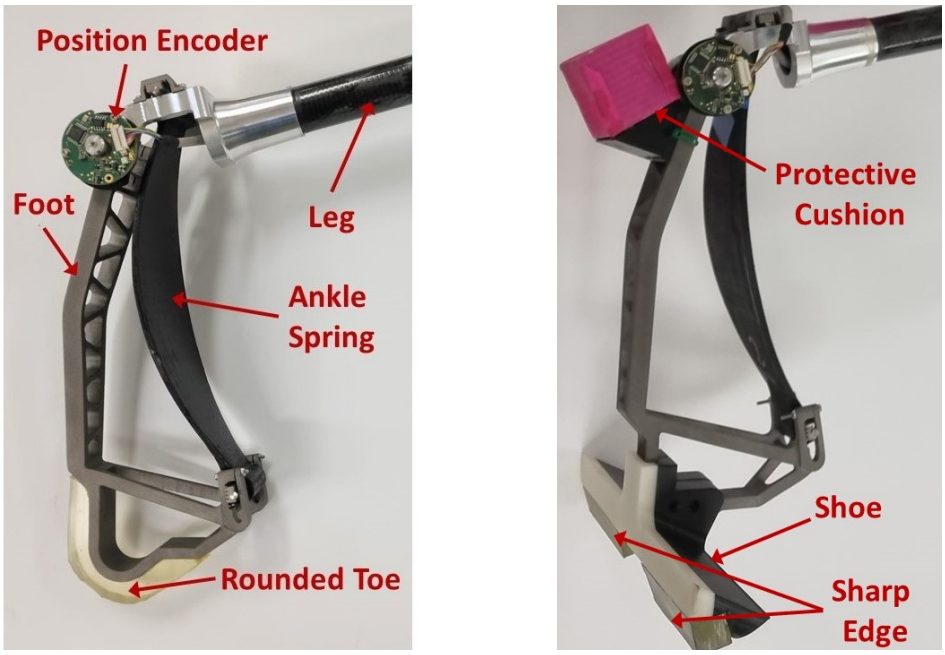


Figure 7.2 Skippy's bodies and joints used to control the real robot.



(a) (b)

Figure 7.3 (a) Foot assembly and (b) foot assembly constrained.

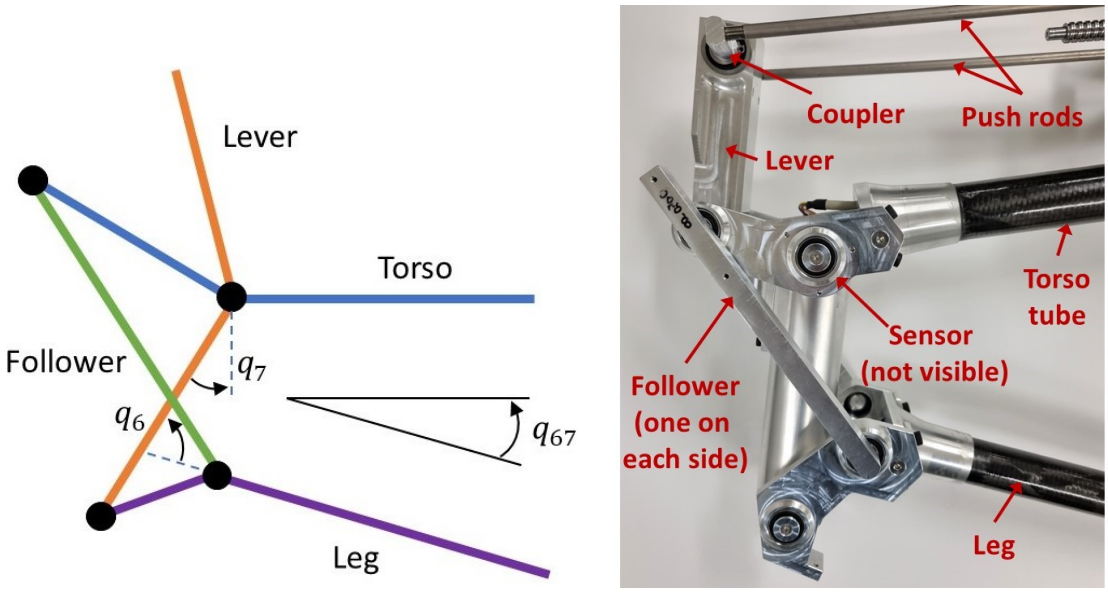
can be considered a single contact point in 2D, in agreement with the simplifying assumption of a toe radius equal to zero. The angle of the foot with respect to the leg is measured with an AksIM-2 absolute magnetic rotary encoder as described in Section 7.1.9.

7.1.2 Leg

It is body 5 of the model. It is a carbon fibre tube which connects the ankle (see Figure 7.3) to the 4-bar mechanism (see Figure 7.4b).

7.1.3 4-bar Mechanism

The 4-bar mechanism is the type of joint used for Skippy's hip, details in Figure 7.4. It connects the leg to the torso and amplifies the lever's motion so that the lever needs to turn only 90° in order to make the leg turn 180°, which is its full range of motion. The lever (body 6) and the follower (body 8) are machined aluminium parts. The hip AksIM-2 absolute magnetic rotary encoder measures the angle q_7, which is the input to the kinematic model of the 4-bar mechanism. The output is the hip angle q_{67}, also called the 'Leg' angle. It is designed to have a gear ratio between the input-output angle not linear. The non-linear relationship allows a higher torque transmission when the leg is folded and a higher velocity



(a) (b)

Figure 7.4 (a) Schematic representation of the 4-bar mechanism; (b): Physical representation of the 4-bar mechanism.

transmission when the leg is stretched. The behaviour is required for hopping, where a high torque transmission rate is necessary to begin the stance phase when the robot is folded, but a high speed rate is needed when the leg is stretched to hop high.

7.1.4 Torso

It is body 7 of the model, see Figure 7.5. It consists of a carbon fibre tube with an aluminium fork at the rear (hip) end and the Head frame at the front (crossbar) end. Attached to the Head, as described in Section 4.4.1, there is Skippy's Brain, an absolute encoder and the motors. Consequently, it requires protection from impacts. This is achieved by combining the protective cushions mounted on the Head and the crossbar.

7.1.5 Crossbar

The symmetric crossbar, or simply crossbar, is one of the two actuated links of Skippy. Two crossbars have been used throughout the multiple experiments reported in this thesis: the symmetric crossbar and the asymmetric crossbar. Both crossbars are driven in the same way (see Section 4.4.1), and they can be exchanged to balance different robots, such as in Chapter 4 where the robot balances with the symmetric crossbar or Chapters 5 and 6, where it

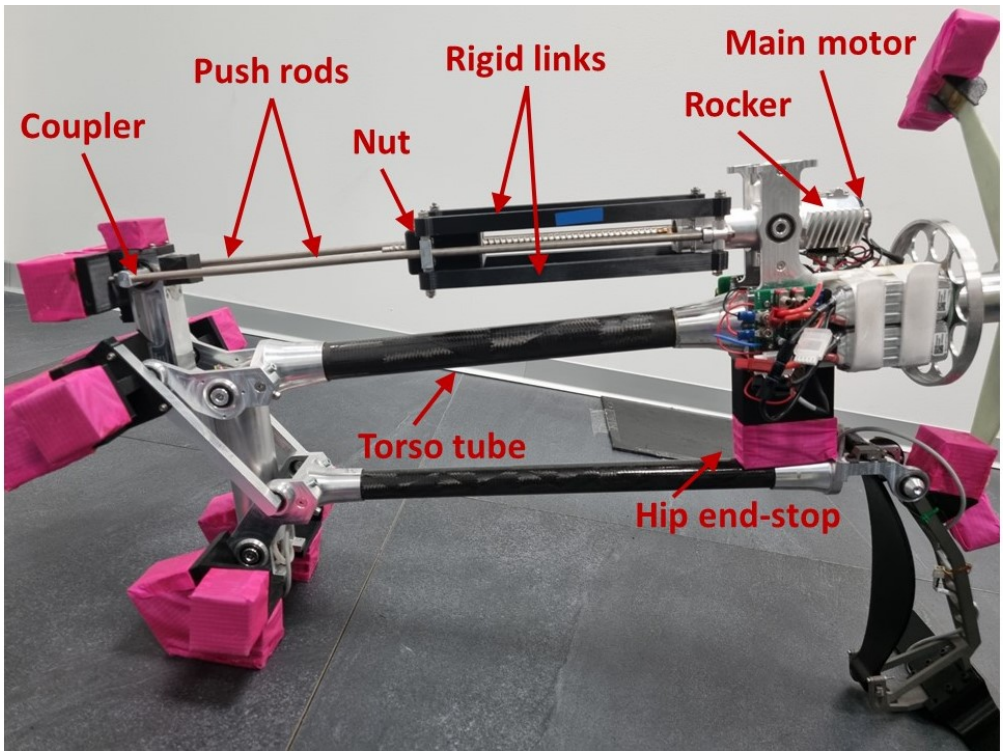


Figure 7.5 Details of Skippy's bodies.

Figure 7.6 Skippy's symmetric crossbar.

uses the asymmetric crossbar. Skippy's symmetric crossbar consists of a three-blade reaction wheel, Figure 7.6. Each blade is made of fibreglass in a specific custom shape, and it can withstand shocks due to crash landings without breaking. Glued brass weights are at the edge of each blade to increase the wheel's inertia. The extremities of the wheel are protected with shock-absorbing foam covered with an anti-tear pink fabric. The flywheel has a dual function: it makes the robot balance and protects the Head in case of a crash landing. In this configuration, the crossbar is not actuated because the robot's balancing plane is kept vertical by the physical constraint of the foot (i.e., the Shoe); its only purpose is to protect the Head in case of falls. Consequently, it is modelled as a mass being part of the torso.

7.1.6 Rocker

It is body 9 in the model, and it is shown in Figure 7.5. It is mounted on top of the Head, and its primary function is holding the Main motor. On top of the Rocker is shock-absorbing foam, whose purpose is to protect the Head in case of falls (see Figure 7.1). The Rocker allows the motor to rotate on an axis parallel to the rotation axis of the hip. This motion enables the screw rod and push rods to be aligned with the imaginary line connecting the coupler and the Rocker, more details in Section 7.2, which changes angle with respect to the torso depending on the position of the lever. Although the body is called Rocker, it also includes the screw rod and the main motor.

7.1.7 Nut

It is body 10 in the model, and it is shown in Figure 7.5. Although it is called Nut, it also includes the coupler, push rods and the rigid links that take the place of the Main springs.

7.1.8 Actuation System

The actuation system is shown in Figure 7.7. It comprises a Maxon DC motor DCX32L-GB-KL-24V which actuates a Misumi BSSC1004-256 ball screw mechanism. The motor shaft is connected to the screw rod through a Misumi XGT2-19C-6-6 elastic coupler to cope with the slight misalignment that may arise in the assembly. The ball screw mechanism transforms the rotational motion of the motor into linear motion of the nut, with a pitch of 4 mm. The motor is controlled by a Pololu G2 24v21 motor driver, enabling the Brain to regulate it via a PWM signal like the Crossbar motor. The nut, as shown in Figure 7.5, pushes and pulls the lever of the 4-bar mechanism, opening and closing the Leg angle. The rigid links connecting

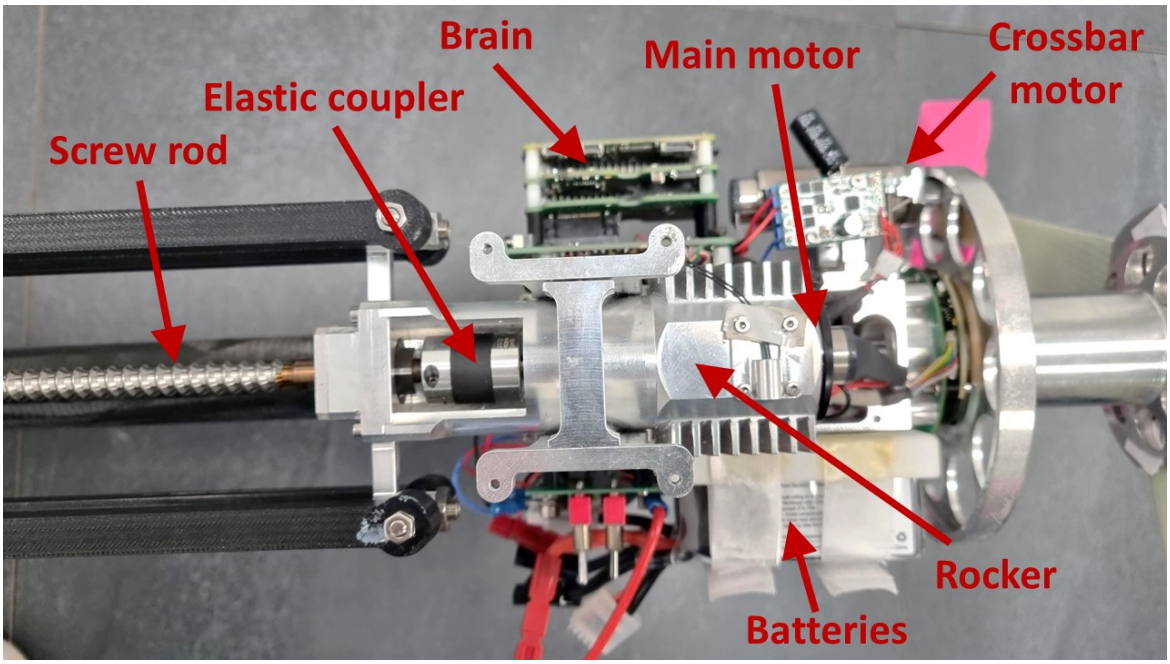


Figure 7.7 Skippy's hip actuation system.

the nut to the push-rods will be replaced by the Main springs in future works to increase Skippy's jumping capabilities.

7.1.9 Sensors

Skippy, in its 2D springy leg configuration, has four independent variables measured using four sensors.

- q_5 and q_7 are measured with two AksIM-2 MB029DCC18BFNT00 absolute magnetic rotary encoders, with a magnetic ring diameter of 29mm and a resolution of 18 bit,
- q_4 is a combination of the torso absolute orientation measured via IMU, q_5 and q_{67}, and
- q_{10} is linearly converted from the angular position of the Main motor and the screw's pitch. The angular position of the motor is measured using a Maxon ENX 16 EASY incremental position encoder with the same characteristics as the Crossbar motor encoder, described in Section 4.4.2.

Spring	L_R [mm]	θ [rad]	F_{25} [N]	F_H [%]
Main	195	1.885	550	1.02
Ankle	200	$\pi/3$	955	1.32

Table 7.2 Measured parameters used to model the behaviour of the springs. Skippy's main spring consists of four physical springs in parallel, and F_{25} refers to a single physical spring.

7.1.10 Skippy's Springs

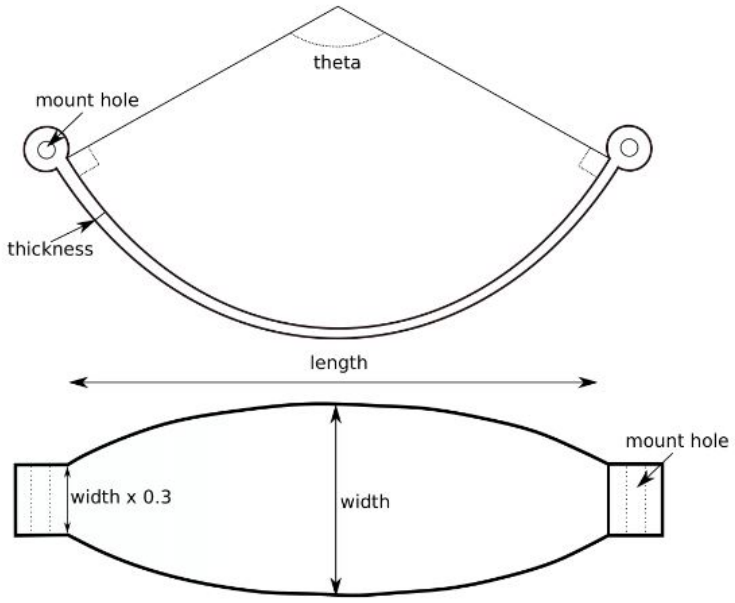
Balancing and hopping are distinct activities. Hopping entails energetic and vigorous manoeuvres while balancing necessitates precise and gentle motions. Skippy is equipped with fibreglass curved leaf springs to accomplish both objectives simultaneously. The decision to employ these springs, as described in the aforementioned work [1 § 5.3], was based on three primary factors: (1) the requirement for a high ratio of elastic energy to weight, (2) the necessity for consistent spring-like behaviour, and (3) the desire for a force profile that diminishes with increasing compression (regressiveness). Steel (for reason (1)) and rubber (for reason (2)) were excluded as materials due to these considerations. The regressive nature of the springs is pivotal for Skippy's performance as it enables greater storage of elastic energy for a given stroke at a maximum force and increased stiffness at lower force levels. Consequently, regressive springs supply Skippy with sufficient energy for hopping while simultaneously providing the stiffness necessary for the robot to achieve rapid and precise balance.

The springs are fibreglass curved leaf springs with constant thickness and varying width at rest state, as shown in Figure 7.8. The behaviour of these springs can be accurately described using three parameters: their resting length L_R, the angle θ of the curved arc when at rest, and the force required to compress them by 25% F_{25}, as shown in Table 7.2. A fourth parameter, force hysteresis F_H, also models the energy loss that occurs in real springs.

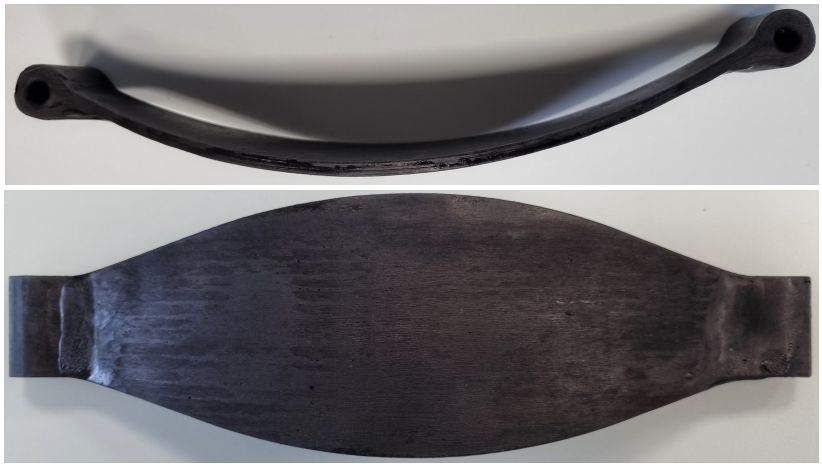
I have personally tested the springs with a tensile and compression machine to evaluate their real behaviour and how accurately the model can describe them, while the Skippy team realized the model itself and analysed the collected data; the results are shown in Figure 7.9. Because the Ankle spring always operates in compression, it is tested only for compression, while the Main spring is tested mainly for compression and slight tension. Both models accurately describe the behaviour of the springs, except for a slight deviation at the cycle's edges.

Both springs have also been tested to their compression limit. Figure 7.10 shows the results of the failure tests, with both springs failing way beyond the working conditions. During normal operations, the springs behave consistently, as shown in Figure 7.11, guaranteeing

the same behaviour over time. These results allow the robot to be used for robot learning, where the same mechanical behaviour is necessary through the different iterations.

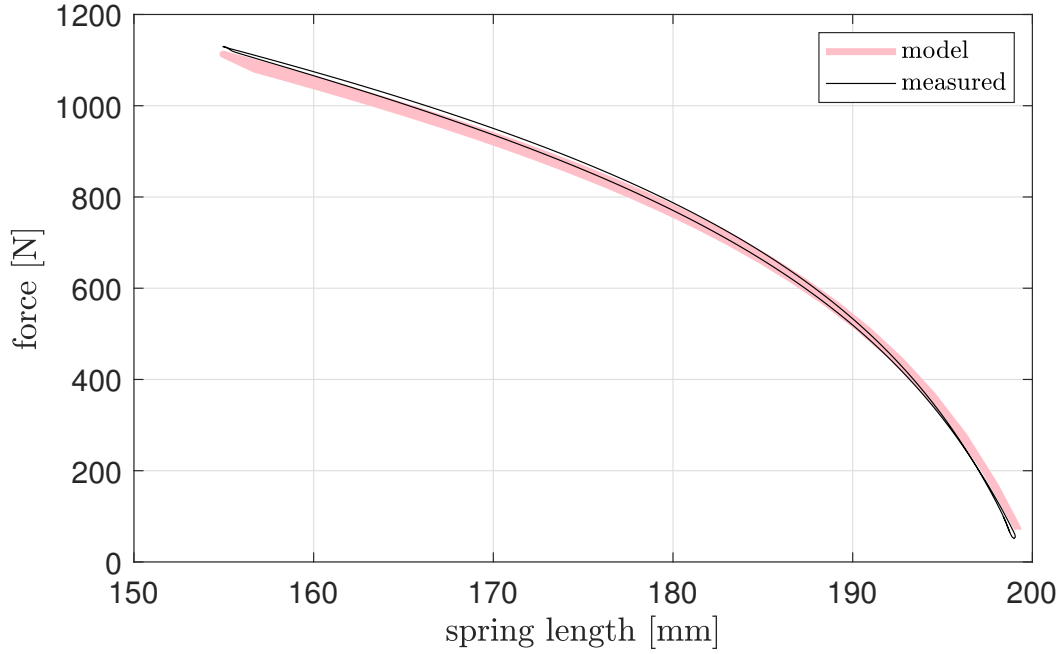


(a)

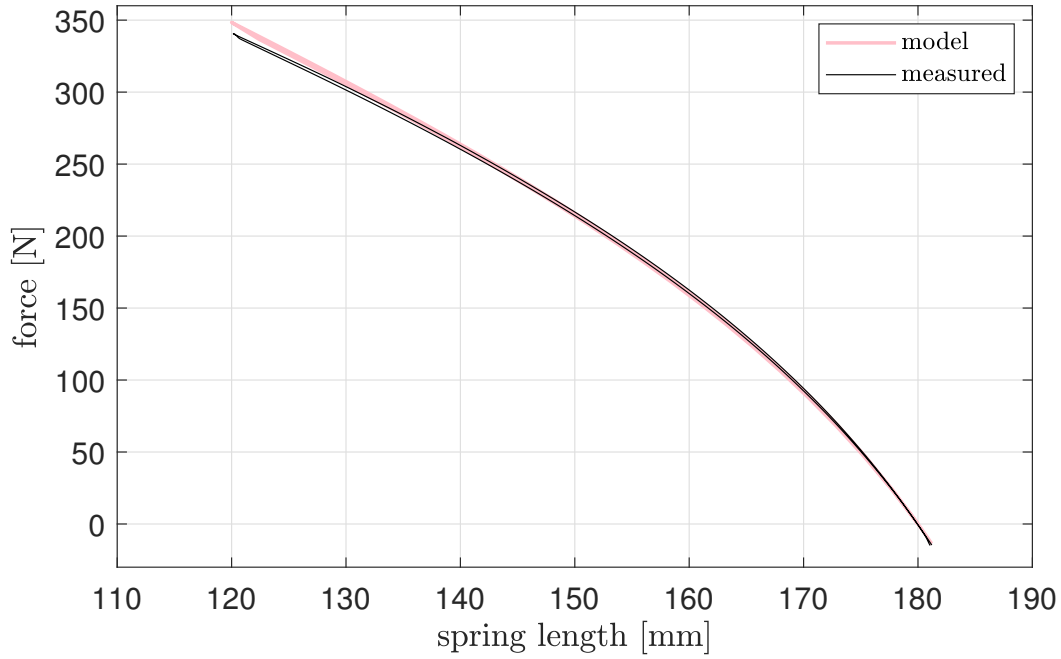


(b)

Figure 7.8 (a): Geometry of the tapered fibreglass springs [1]; (b): Physical realization of the Ankle spring



(a)



(b)

Figure 7.9 Accuracy of model fit to measured data for (a) Ankle spring and (b) Main spring.

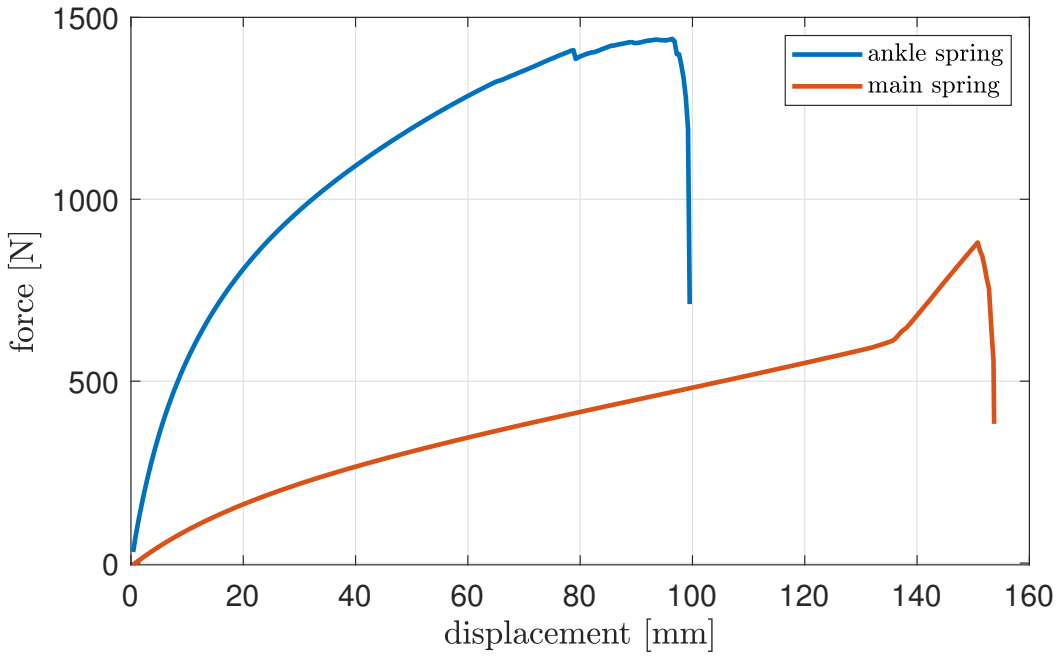


Figure 7.10 Fracture limit for the springs in compression.

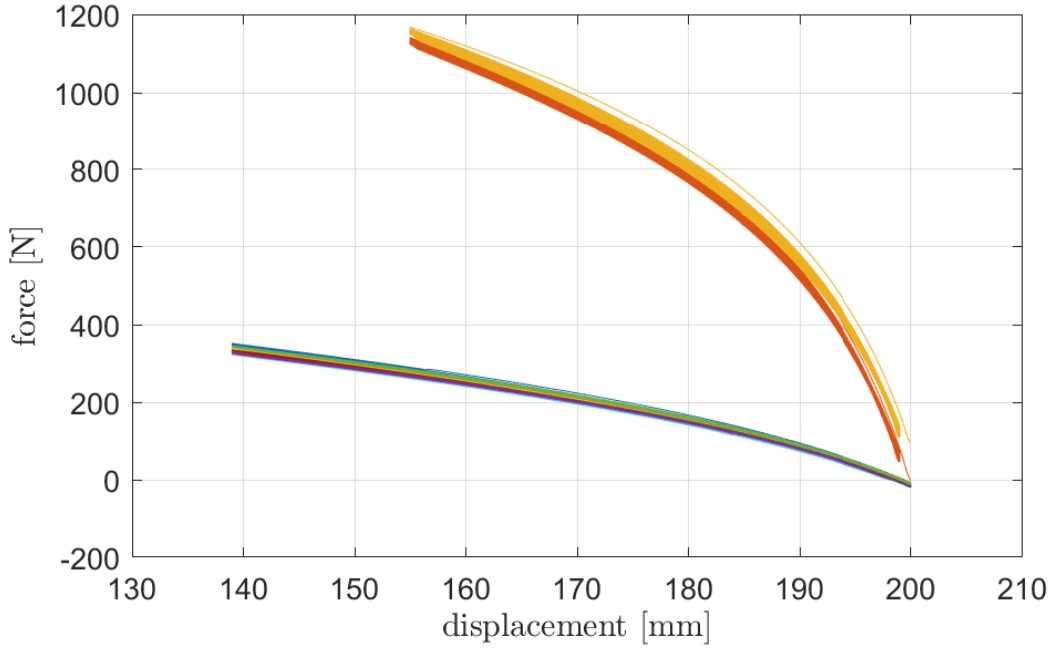


Figure 7.11 Repeatability experiments. Red and yellow lines represent the Ankle springs, the other colours the Main springs. One colour for each spring that was tested (10 Main springs and 2 Ankle springs).

7.2 Kinematic Model

This section describes Skippy's kinematic model depicted in Figure 7.12 and how all the variables are defined. A more detailed kinematic diagram is present in Appendix B Figure B.2. The diagram presented in Figure 7.12, also called 'old model', slightly differs from the one shown in Figure B.2, called 'new model'. In the new model the follower and the lower part of the lever are shown at right angles to the horizontal part of the torso while in the old model they are shown at right angles to the tilted portion of the torso. This discrepancy affects the definitions of q_6, q_7 and q_8, but not q_{67}. There are also differences in the positions of some of the body coordinate frames: in the new model, F4 is at P2, F5 is at P1, and F6 is at P0. In this chapter all the kinematic calculations are done with the old model.

Skippy's motion can be described by five independent variables q_1, q_2, q_4, q_5 and q_{10}. This section describes how the other dependent variables are derived. q_3 represents the motion of the rolling contact at the ground, and it is defined as $q_3 = -\text{rtoe}\, q_4$. Considering the triangle P0-P5-P6 in Figure 7.12, we can observe that P0-P5 and P0-P6 are constant, and P5-P6 varies with q_{10}; this kinematic triangle allows us to derive q_9 the angle between the torso and the rocker and consequently q_7 the angle between the torso and the lever. The 4-bar mechanism can be modelled as two kinematic triangles P0-P3-P4 and P0-P1-P3; having q_7 as an input, we can solve the first triangle for q_8 the torso follower angle. Solving the other triangle allows us to derive q_6, the angle between the leg and the lever and consequently q_{67}. The Skippy team developed the Matlab functions to calculate Skippy's kinematics (see Appendix B), and my personal contribution consisted of integrating these functions into Skippy's control system.

7.2.1 Model Mapping

As described in Section 6.1.1, it is necessary to explicitly express the motion constraints to apply the general balance controller. It is then necessary to define the matrix G and the vector g.

$$G^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\text{rtoe} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & \beta & \gamma & \theta & 1 \end{bmatrix}$$

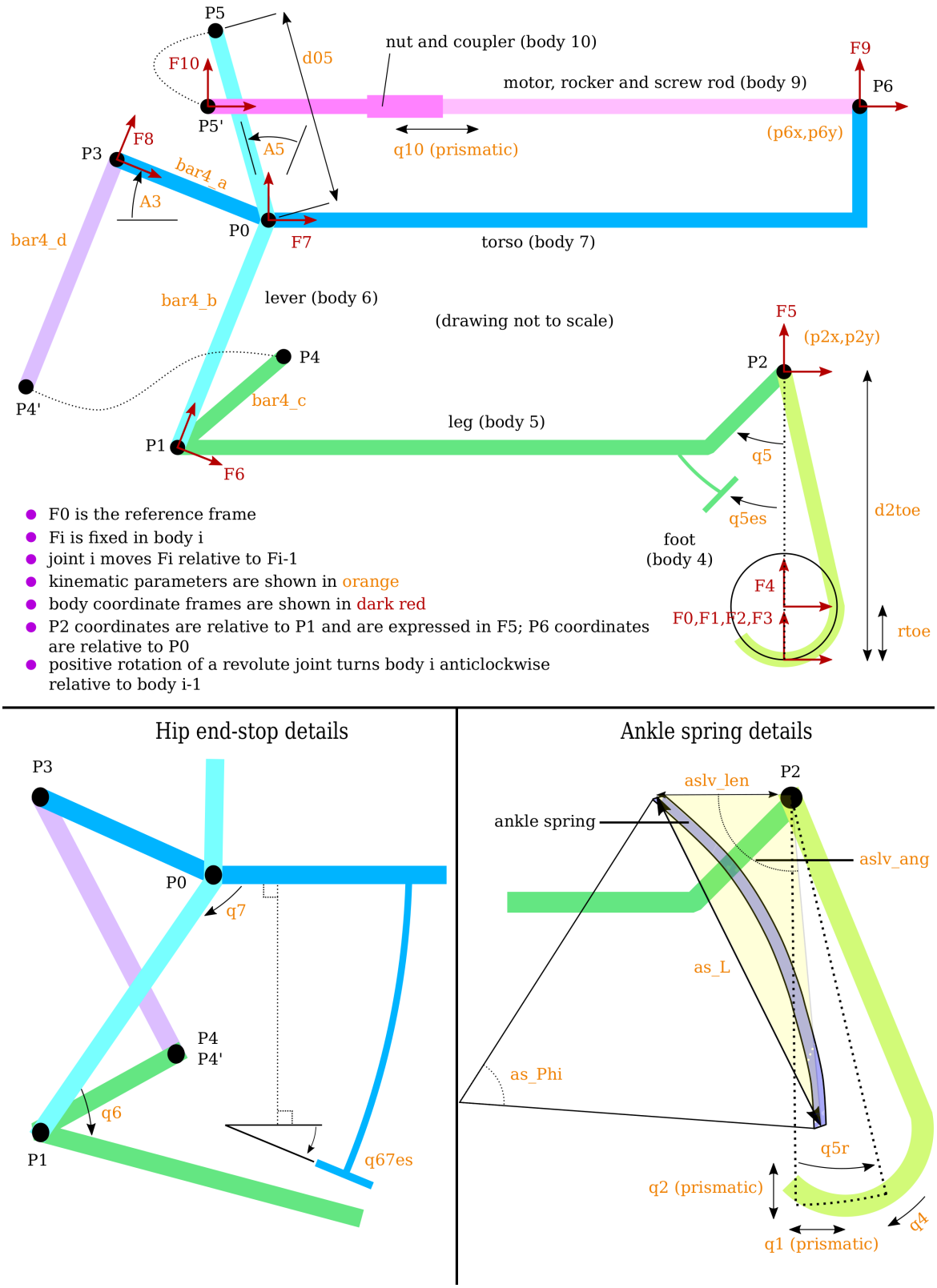


Figure 7.12 Top diagram showing a simplified Skippy's kinematic model in its open-loop zero position ($q = 0$), and details of the hip end stop (4-bar mechanism) and ankle spring. [1]. Skippy's kinematic and dynamic parameters are reported in Tables 7.3 and 7.4.

Name	Kinematic Parameter	Value
rtoe	radius of toe	0.00
d2toe	length of foot (body 4)	0.245
A3	angle of 4-bar segment AB within torso	0.3699
A5	lever angle	0.5236
bar4_a	length of 4-bar segment (in torso)	0.0536
bar4_b	length of 4-bar segment (in lever, body 6)	0.1353
bar4_c	length of 4-bar segment (in leg)	0.0553
bar4_d	length of 4-bar segment (follower, body 8)	0.1309
bar4_phi	CD offset angle at motion limits	0.1301
bar4_Bmin	4-bar angle at maximum stretch	0.3699
bar4_Bmax	4-bar angle at maximum fold	1.9407
p0x	x coordinate of P0 (lever joint in torso)	0
p0y	y coordinate of P0 (lever joint in torso)	0
p1x	x coordinate of P1 (lever joint in leg) relative to P0	0
p1y	y coordinate of P1 (lever joint in leg) relative to P0	-0.1353
p2x	x coordinate of P2 (ankle) relative to P1	0.519
p2y	y coordinate of P2 (ankle) relative to P1	0.039
p3x	x coordinate of P3 (follower joint in torso) relative to P0	-0.050
p3y	y coordiante of P4 (follower joint in torso) relative to P0	0.019
p4x	x coordinate of P4 (follower joint in leg) relative to P1	0.0265
p4y	y coordiante of P4 (follower joint in leg) relative to P1	0.0485
p5x	x coordinate of P5 (coupler) relative to P0	-0.0525
p5y	y coordinate of P5 (coupler) relative to P0	0.0909
p6x	x coordinate of P6 (rocker joint in torso) relative to P0	0.42
p6y	y coordinate of P6 (rocker joint in torso) relative to P0	0.080
p7x	x coordinate of P7 (crossbar) relative to P0	0.621
p7y	y coordinate of P7 (crossbar) relative to P0	0.01
p8x	x coordinate of P8 (ankle spring joint in leg) relative to P2, adjustable	-0.0327
p8y	y coordinate of P8 (ankle spring joint in leg) relative to P2, adjustable	0.023
p9x	x coordinate of P9 (ankle spring joint in foot) relative to P2, adjustable	-0.0647
p9y	y coordinate of P9 (ankle spring joint in foot) relative to P2, adjustable	-0.1775

Table 7.3 List of kinematic parameters of Skippy's mechanism of Figures 7.12 and B.2. Lengths are in metres and angles in radians. See Table 7.2 for spring parameters.

Name	Dynamic Parameter	Value
foot_m	mass of foot (body 4)	0.520
foot_cx	foot centre of mass x coordinate	-0.010
foot_cy	foot centre of mass y coordinate	0.138
foot_rog	foot radius of gyration	0.028
leg_m	mass of leg (body 5)	0.615
leg_cx	leg centre of mass x coordinate	-0.410
leg_cy	leg centre of mass y coordinate	-0.021
leg_rog	leg radius of gyration	0.195
lever_m	mass of lever (body 6)	0.367
lever_cx	lever centre of mass x coordinate	-0.029
lever_cy	lever centre of mass y coordinate	0.140
lever_rog	lever radius of gyration	0.084
torso_m	mass of torso+, rocker, nut (bodies 7, 9, 10)	3.970
torso_cx	torso+ centre of mass x coordinate	0.371
torso_cy	torso+ centre of mass y coordinate	0.040
torso_rog	torso radius of gyration	0.210

Table 7.4 List of dynamic parameters of Skippy's mechanism of Figures 7.12 and B.2. Lengths are in metres, masses in kilograms and angles in radians. The symbol 'torso+' means the torso plus the crossbar, treated as a single rigid body.

$$g^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dot{\alpha}\dot{q}_{10} & \dot{\beta}\dot{q}_{10} & \dot{\gamma}\dot{q}_{10} & \dot{\theta}\dot{q}_{10} & 0 \end{bmatrix}$$

where α, β, γ and θ are configuration-dependent functions relating dependent joint velocities to $\dot{q}_{10}$. Appendix B shows the Matlab functions used to calculate matrix G and vector g.

The Ankle spring has been designed to be stiff enough so that it almost doesn't compress while balancing, allowing for a simplifying assumption $\dot{q}_5 = 0$. Thanks to the regressive nature of the spring (more details in Section 7.1.10), the torque it generates while the robot balances can be neglected by the balance controller. This simplification proves the robustness of the balance controller but, at the same time, limits the motion command for the actuated joint. Fast movements of the actuated joint make the spring oscillate, reducing the tracking performance of the robot. Furthermore, the definition of $q_1 = q_2 = 0$ always, implies $\dot{q}_1 = \dot{q}_2 = 0$ always. As a consequence, the number of independent variables required to describe Skippy's motion decreases to two and the matrix G can be simplified to $\overline{G}$ by removing the first, second and fifth columns.

$$\overline{G}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & -\mathrm{rtoe} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & \beta & \gamma & \theta & 1 \end{bmatrix} \tag{7.1}$$

As described in Chapter 6.1, it is necessary to explicitly express the motion constraints to apply the general balance controller of Chapter 5. The two independent variables used to balance Skippy's leg are q_4 and q_{10}. $q_4 = y_1$ is the passive joint used as a reference for balancing, while $q_{10} = y_2$ is the actuated joint used to balance the robot, and $y = [y_1\, y_2]^{\mathrm{T}}$.

Defining the equation of motion of the unconstrained robot as

$$H_{\mathrm{u}}\ddot{q} + C_{\mathrm{u}} = \tau_{\mathrm{u}} \tag{7.2}$$

where H_{u} is the joint-space inertia matrix, C_{u} is the bias vector containing Coriolis, centrifugal and gravitational terms, and $\tau_{\mathrm{u}} = [0, ..., 0, \tau_{10}]^{\mathrm{T}}$ the vector of joint force variables. then the equation of motion of the constrained robot is

$$H\ddot{y} + C = \tau \tag{7.3}$$

where

$$H = \overline{G}^{\mathrm{T}} H_{\mathrm{u}} \overline{G} \quad , \quad C = \overline{G}^{\mathrm{T}}(C_{\mathrm{u}} + H_{\mathrm{u}} g) \quad \text{and} \quad \tau = \overline{G}^{\mathrm{T}} \tau_{\mathrm{u}} = [0 \;\; \tau_{10}]^{\mathrm{T}} \tag{7.4}$$

7.3 Centre of Mass Observer

The balance controller is very sensitive in the estimation of the position of the CoM [91]. For this reason, an observer has been proposed in Chapter 4 to compensate for inaccuracies in the estimation of the vertical through the IMU. However, the observer defined in Chapter 4 cannot be applied to Skippy. The reason lies in the definition of the observer, where it is assumed that the position of the CoM is proportional only to the robot's angle with respect to the vertical, assuming that the robot is perfectly balanced when upright. This assumption is not valid with this configuration of Skippy because the angle used for balancing is $y_1 \neq 0$ in any possible balanced configuration; furthermore, the position of the CoM is no longer only proportional to such angle, but it also depends on q_5 and q_{67}. A new observer is then proposed, intending to compensate for sensors' inaccuracies in estimating the x component of CoM position.

Given y_1 and y_2 as the independent variables used by the balance controller, from previous chapters, we have that the angular momentum of the robot about the support point is

$$L = H_{11}\dot{y}_1 + H_{12}\dot{y}_2 \tag{7.5}$$

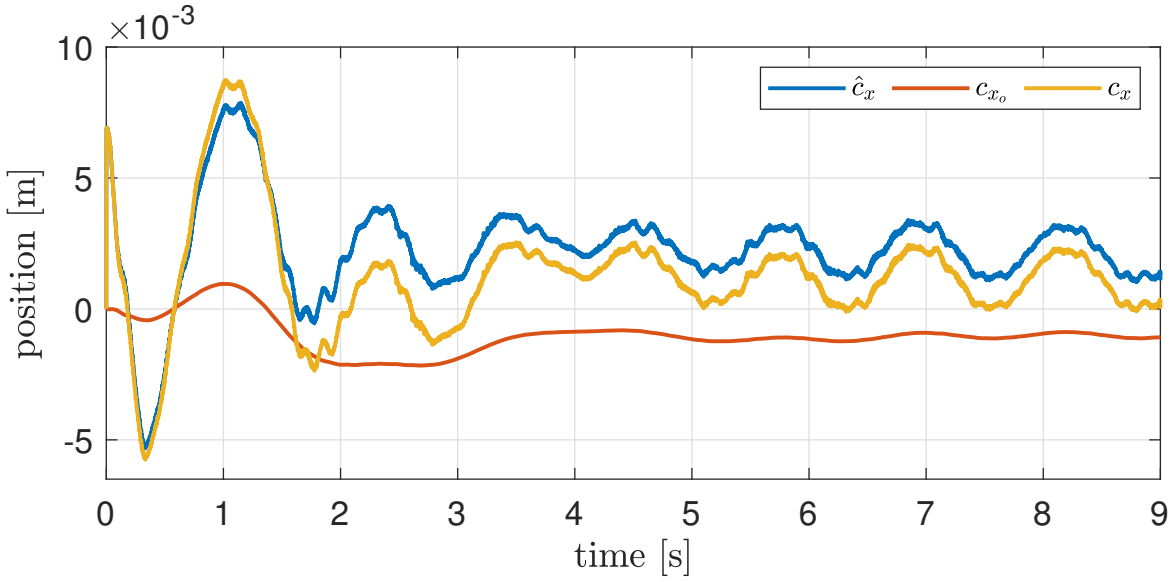


Figure 7.13 $\hat{c}_x$ is the measured position of the centre of mass, c_{x_o} is the offset error, and c_x is the corrected value.

and since no external force is applied to the robot, the derivative of the angular momentum of the robot balancing on a fixed point is

$$\dot{L} = -mgc_x \,. \tag{7.6}$$

We now have two ways to define $\dot{L}$ which are Equation 7.6 and the derivative of Equation 7.5

$$\dot{L} = \frac{dL}{dt} \,. \tag{7.7}$$

Let us define

$$c_x = \hat{c}_x - c_{x_o} \tag{7.8}$$

where $\hat{c}_x$ is the position of the CoM obtained with the sensors' data, and c_{x_o} is the unknown error in such estimate, which varies so slowly that we can approximate $\dot{c}_{x_o} = 0$. The value of the error can be obtained by combining the two definitions of $\dot{L}$ and filtering the output

$$c_{x_o} = \text{LPF}\left(\hat{c}_x + \frac{1}{mg}\frac{dL}{dt}\right) \tag{7.9}$$

where 'LPF' is a Low Pass Filter with a cutoff frequency of 1 rad/s. The expression $\hat{c}_x - c_{x_o}$ is used in place of c_x by the balance controller in this chapter.

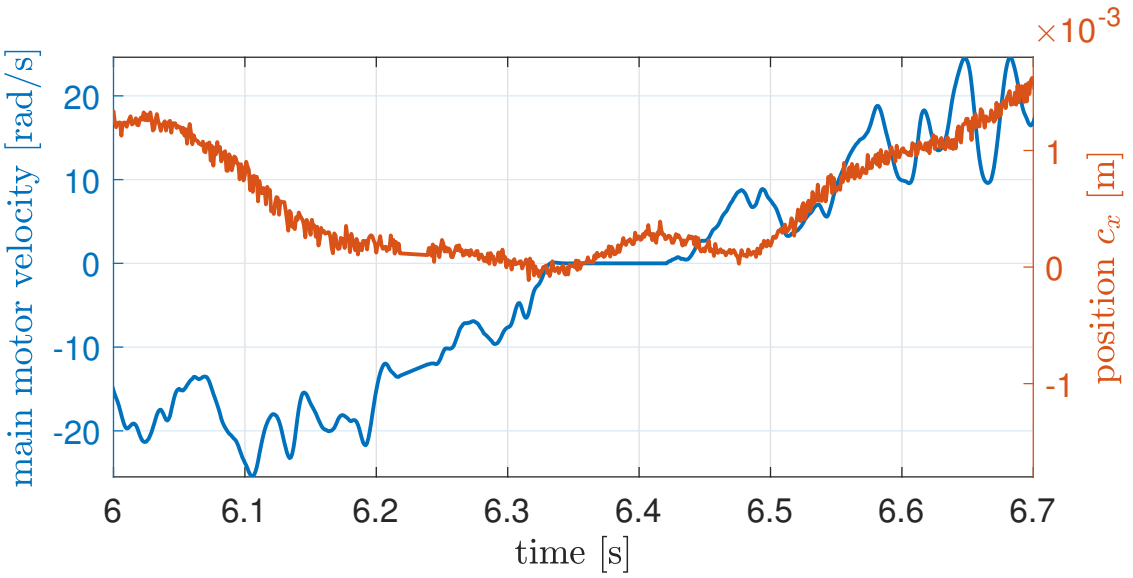


Figure 7.14 Velocity of the main motor sticks to zero when c_x reaches to zero.

Figure 7.13 shows how the CoM observer works throughout the experiment, where the robot is manually placed in a balanced configuration and is commanded to balance without moving the torso. The robot starts balancing at $t = 0\,\text{s}$; simultaneously, the observer estimates the offset error, allowing the robot to align the updated CoM position with the origin, $c_x = 0$. The moment c_x is aligned with the vertical, the balance controller asks for minimal variation in $\dot{q}_{10}$ since the robot is in a balanced configuration. Such a small command causes the motor stiction problem, as shown in Figure 7.14, where the motor is still for about 0.1 s. Consequently, the robot starts going off balance until the balance controller asks for a sufficiently high command, making the robot oscillate, making it impossible to align c_x with the vertical permanently. Although the observer is designed to compensate fully for the CoM estimation error, it compensates only for half of the error. The exact reason for this could not be found. Still, the author believes the cause can be researched in a combination of motor stiction and not perfectly precise dynamic and kinematic parameters due to manufacturing tolerance and physical assembly.

7.4 Experiment

The experiment validates Skippy as a balancing and hopping machine. The balance controller is the same as in Chapter 5, the four poles are $\lambda_1 = -1/T_\text{c}$, $\lambda_2 = \lambda_3 = -4.5$, and $\lambda_4 = -6$, where T_c is the robot's constant of toppling in the current configuration. The two zeros are

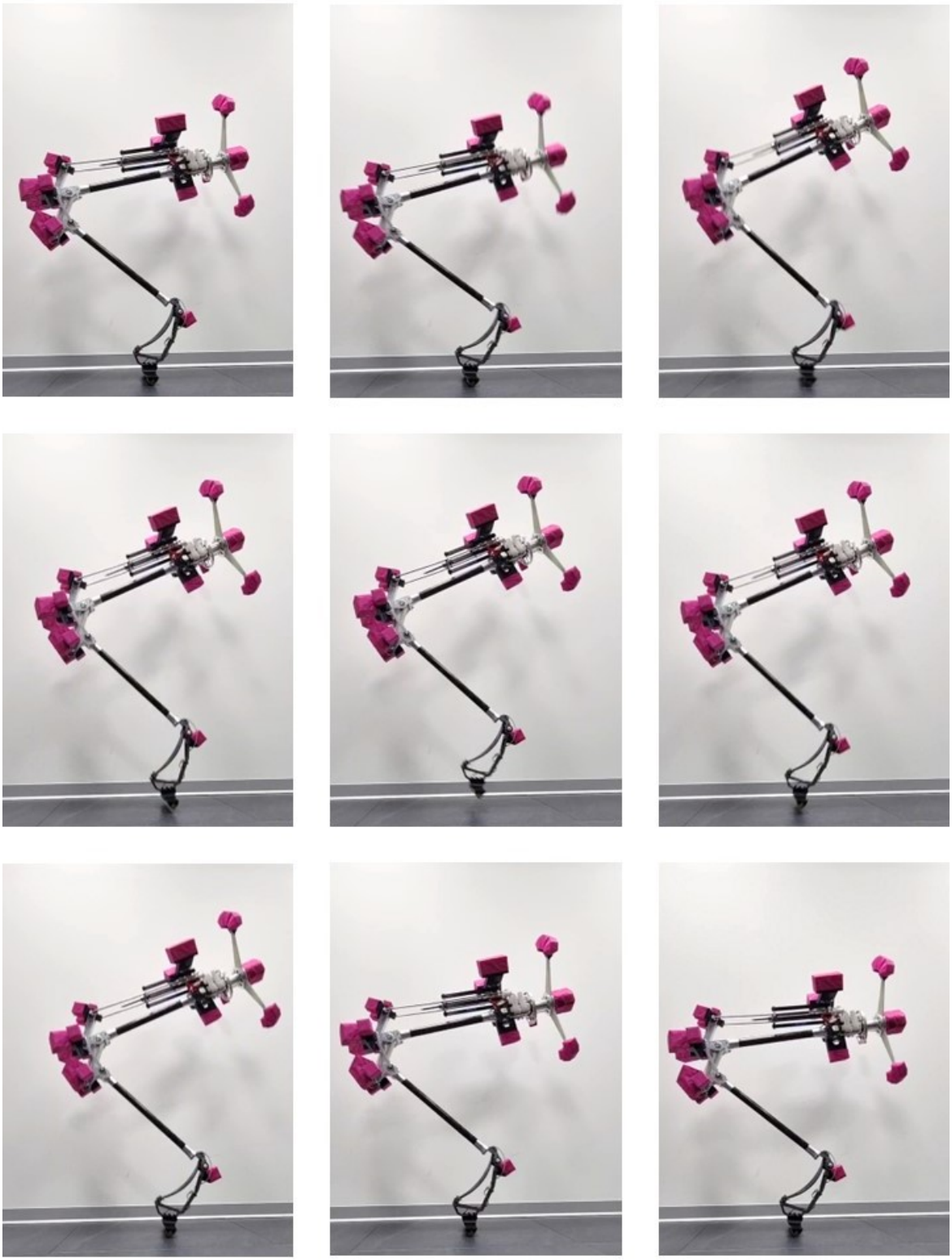


Figure 7.15 Hopping experiment.

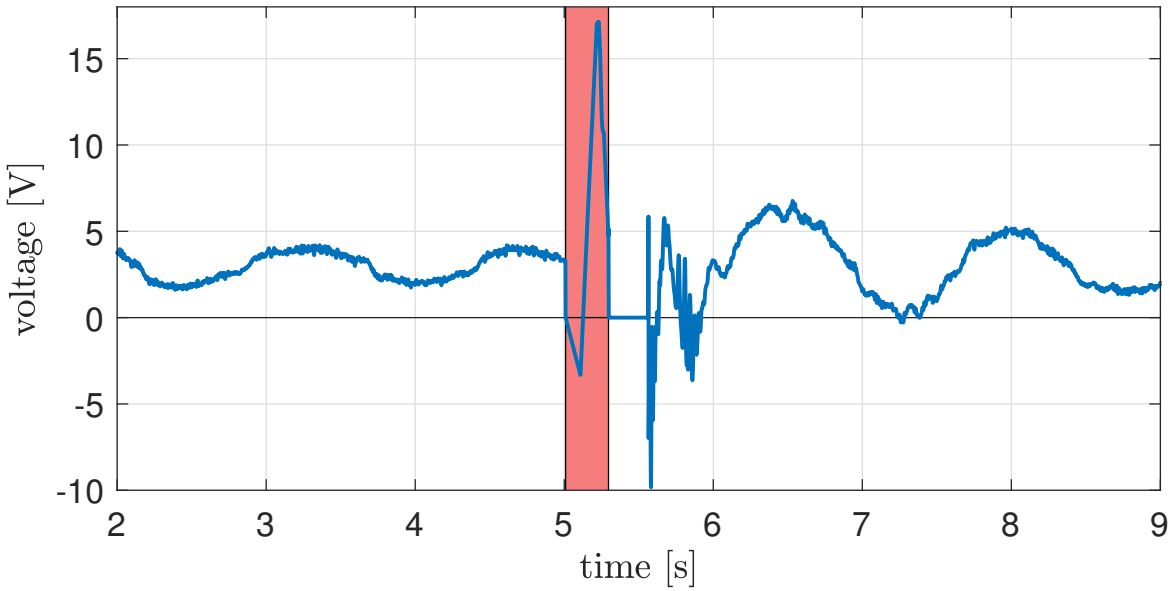


Figure 7.16 Voltage profile through the experiment. White background indicates when the voltage is the output of the control system; pink background when it is the feed forward signal for hopping.

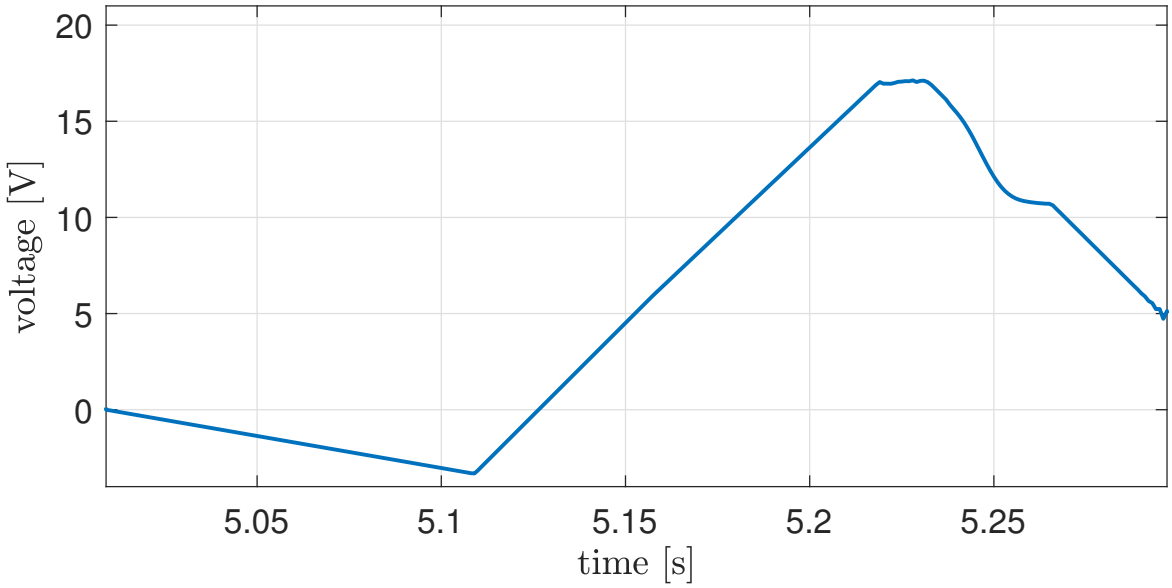


Figure 7.17 Feed forward voltage profile for hopping. The voltage goes to zero immediately after this signal.

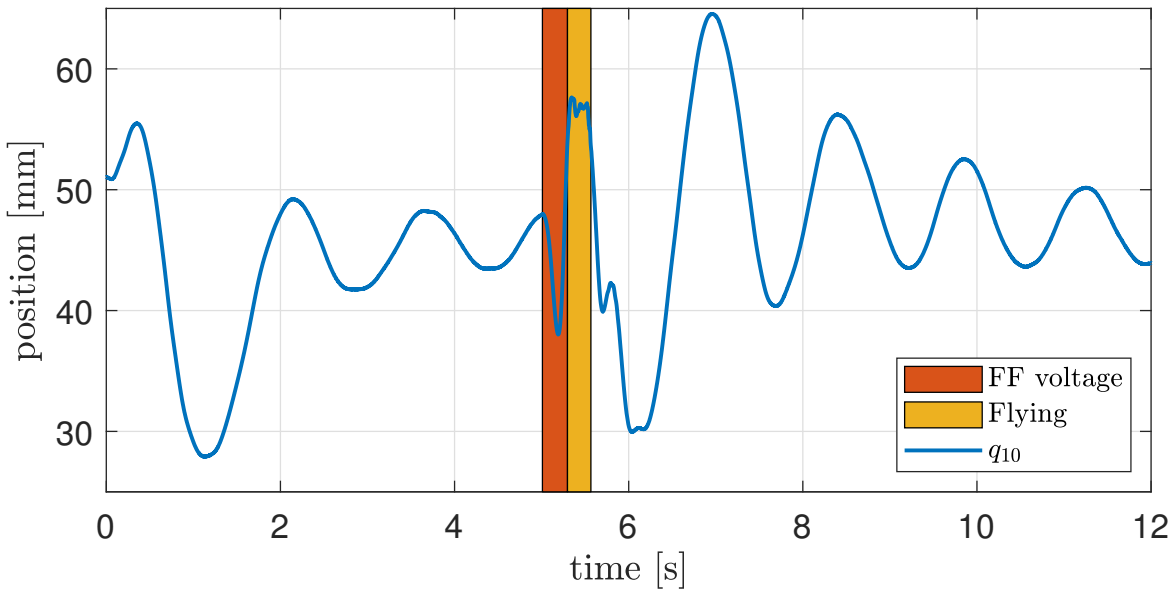


Figure 7.18 Position of joint q_{10} while balancing and hopping.

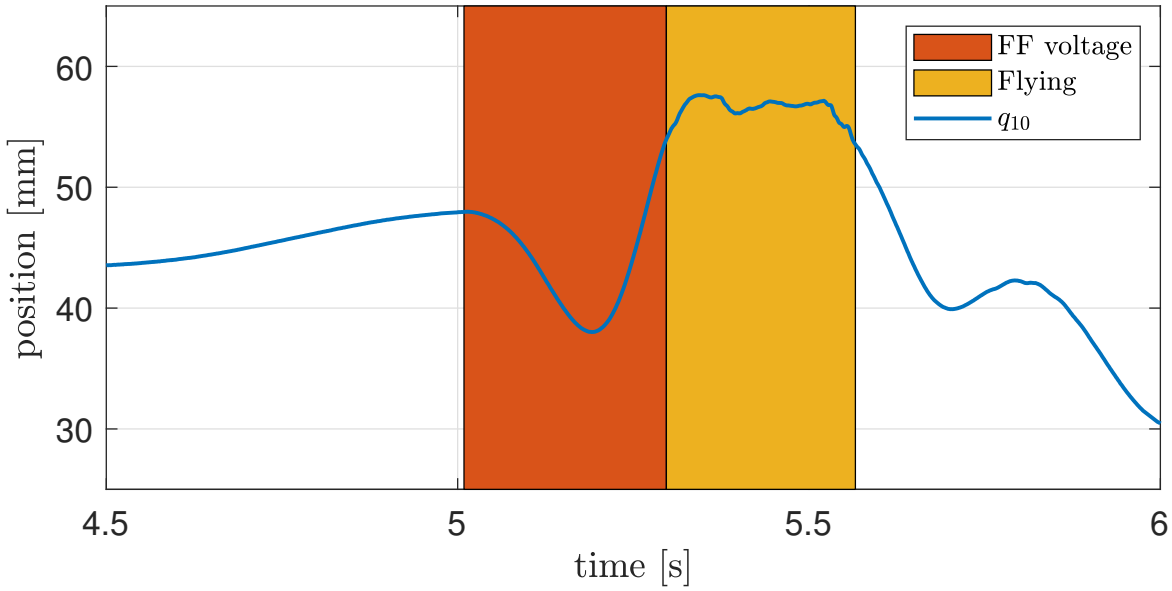


Figure 7.19 Position of joint q_{10} while balancing and hopping.

set to cancel λ_2 and λ_3. The command signal is a constant value for the actuated joint 10, $q_d = 0.048$. The experiment, shown in Figure 7.15, starts with the operator manually placing Skippy in a balanced configuration. Then, the balance controller is turned on (together with the CoM observer), and the robot starts balancing. Then, the robot opens the Leg angle to the desired position (the command signal is a constant position for q_{10}). Once reached such position, the balance controller is switched off, and an open loop voltage signal (see Figures 7.16 and 7.17) is applied to the Main motor. The voltage profile is an optimised signal allowing the robot to hop with the desired velocity (vertical and horizontal components) of the centre of mass. The signal's optimisation strategy is out of this thesis's scope and can be found in [1]. The desired vertical velocity of the CoM is $1\,\mathrm{m/s}$, corresponding to a vertical hop of $5\,\mathrm{cm}$. In this initial experiment, the balance controller is switched on again after a time interval determined experimentally since no contact sensor is placed in the foot to detect landing. Once turned on again, the balance controller allows the robot to recover the balance regardless of the disturbances caused by the small hop.

Figures 7.18 and 7.19 show the position of joint q_{10} throughout the experiment. The figure is characterized by three different background colours, each representing a state of the control system: white, the robot is actively balancing thanks to the balance controller; red, the feed-forward signal computed apriori off-line is applied to the motor to make the robot hop, and the balance controller is turned off; yellow, the robot is flying, and zero Volts are applied to the motor.

Unlike the robots used in Chapters 4, 5 and 6 where there is a configuration where the structure of the robot itself can compensate for gravity, Skippy, in its 2D configuration, cannot do so. Consequently, it is forced to always actively compensate for gravity while balancing but cannot balance with minimal movements of the actuated joint. This behaviour has multiple causes, the most significant being the main motor's stiction and the Ankle spring's presence. Once the robot reaches a balanced position, the balance controller asks for a minimal variation in the acceleration of the actuated joint to keep the balance. This variation corresponds to such a small velocity command that the motor doesn't move, as the torque generated by the motor is not enough to overcome the static friction. As a consequence, the robot starts going off balance until a sufficiently high-velocity command is given. This behaviour causes sudden changes in the motor velocity, which excites the spring at the ankle, requiring an additional movement of the actuated joint to compensate for it. The effects of this compensation can be seen in Figure 7.18, where q_{10} continuously oscillates and in Figure 7.13 where the robot's CoM moves according to the motion of q_{10}.

7.5 Conclusion

This chapter presented the first successful experimental demonstration of the balancing and hopping machine called Skippy in its 2D configuration. The experiment showed Skippy balancing, hopping, landing and balancing again without any external intervention from the operator. The results of the experiment proved the potentialities of Skippy and its successful design, which can be successfully used for both balancing and hopping experiments. Furthermore, this chapter demonstrated that the control strategy adopted in the previous chapters can be used for more complex machines, even on a 16-bit micro-controller running at 1 kHz, proving even more the light weight of the control algorithm.

Chapter 8

Conclusion

This thesis presented Skippy, a fully autonomous balancing and hopping machine. It gives the reader a better understanding of the building process of a balancing and hopping robot and the strategies adopted to make it reliable and robust. Starting from Chapter 3, purpose-built machines are built and used to test and validate both hardware (sensors) and software (balance controllers).

Chapter 3 analyses the aspects of testing the sensorimotor system of a dynamic robot. The procedure depicted in this chapter gives the reader a better understanding of the sensors' limitations and malfunctions. More specifically, it highlights the temporary malfunction of the absolute encoder acquisition system in case of mechanical shocks due to high acceleration impacts. It also proves the presence of a drift in the attitude estimation of Skippy's IMU when subject to a continuous low acceleration motion.

Chapter 4 presents Featherstone's balance controller and the application of its simplified version on a real robot. The balancing machine behaves as an inverted reaction wheel pendulum with a floating base. It is the first time high performance is achieved on a floating base balancing machine estimating the vertical with an IMU.

Chapter 5 describes the first successful implementation of Featherstone's general balance controller on a floating base inverted double pendulum. The experimental results shows the controller's capability to precisely track various signals (i.e. fast and slow ramps, sine waves) while being able to compensate for unexpected external disturbances, such as the back foot slipping or the robot being hit by a tennis ball. Furthermore, the robot could stand up and balance itself indefinitely without falling over, proving the reliability of both hardware and software.

Chapter 6 extends Featherstone's balance controller to the case of robots that balance on rounded feet or wheels, replacing Featherstone's point-foot assumption with a circular-foot assumption. It presents the new theory for the case of a robot balancing on a flat horizontal

surface. It validates it on an underactuated inverted double pendulum both in simulation and on a real robot. The controller proved its effectiveness in simulation for a Segway-like robot, making the robot balance while simultaneously travelling at a desired speed. The controller is further extended to the case of a robot balancing on a slope, and it is then validated in simulation.

Chapter 7 introduces Skippy in its final version. Although the robot is complete, in this chapter, its motion is constrained in 2D with an extension of the spring-loaded foot (contact point with the ground). The chapter presents all Skippy's components and explains how the robot's control system models them. This chapter is the first implementation of Featherstone's balance controller on a robot balancing on a springy leg. An experiment proves the effectiveness and the robustness of both the robot and balance controller, which is not aware of the presence of the spring. The robot can balance, hop, land and balance again without any external intervention.

This work leaves as a heritage a reliable and robust robot called Skippy. All the robots build in this thesis have been subject to multiple falls and crashes. Nevertheless, the robust design of Skippy (and its predecessor balancing machines) never experienced fatal damages or malfunctioning during all the tests performed. The mechanical structure combined with the shock absorbing foam carefully placed on strategic points, protected all the electronic components, making possible to use the same Brain and encoders through all the experiments. Consequently, Skippy proved to be an adequate machine for testing and developing dynamic and athletic manoeuvres, which can end up in crash landings, not being afraid of breaking the robot.

The directions of future works are many, and here some ideas are proposed.

- Experimental validation of a robot balancing on a rolling contact on a slope.
- Continuous hopping in 2D.
- Making a somersault in 2D.
- Balancing and hopping in 2D on a rolling contact.
- Integration of series elastic elements in the main actuation system.
- Implementation and experimental validation of a 3D balance controller on a robot that makes contact with the ground both on a point foot and on a rolling contact.
- Balancing and hopping in 3D.

References

[1] A. E. Gkikakis, *Mechanism and Behaviour Co-optimisation of High Performance Mobile Robots*. PhD thesis, Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS), University of Genoa, 2021. doi: 10.15167/gkikakis-antoniosemmanouil_phd2021-04-21.

[2] A. I. Roose, S. Yahya, and H. Al-Rizzo, "Fuzzy-logic control of an inverted pendulum on a cart," *Computers & Electrical Engineering*, vol. 61, pp. 31–47, 2017.

[3] J. Moreno-Valenzuela, C. Aguilar-Avelar, S. A. Puga-Guzmán, and V. Santibáñez, "Adaptive neural network control for the trajectory tracking of the furuta pendulum," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 3439–3452, 2016.

[4] J. J. M. Driessen, A. E. Gkikakis, R. Featherstone, and B. R. P. Singh, "Experimental demonstration of high-performance robotic balancing," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9459–9465, 2019.

[5] F. Allione, A. E. Gkikakis, and R. Featherstone, "Experimental demonstration of a general balancing controller on an untethered planar inverted double pendulum," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8292–8297, 2022.

[6] M. Azad *et al.*, *Balancing and hopping motion control algorithms for an under-actuated robot*. PhD thesis, Australian National University, 2014.

[7] C. Semini, N. G. Tsagarakis, E. Guglielmino, and D. G. Caldwell, "Design and experimental evaluation of the hydraulically actuated prototype leg of the HyQ robot," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3640–3645, 2010.

[8] D. Pucci, F. Romano, S. Traversaro, and F. Nori, "Highly dynamic balancing via force control," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 141–141, 2016.

[9] A. Suebsomran, "Balancing control of bicycle robot," in *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 69–73, 2012.

[10] B. Katz, J. D. Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6295–6301, 2019.

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

[12] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[13] ANYbotics, "ANYmal." https://www.anybotics.com/robotics/anymal/, accessed Sep. 2023.

[14] Boston Dynamics, "Spot." https://www.bostondynamics.com/products/spot, accessed Mar. 2023.

[15] Boston Dynamics, "Atlas." https://www.bostondynamics.com/atlas, accessed Mar. 2023.

[16] Agility Robotics, "Digit." https://agilityrobotics.com/robots, accessed Mar. 2023.

[17] TRoy Featherstone, "The Skippy Project." http://royfeatherstone.org/skippy/, accessed Sep. 2023.

[18] R. Featherstone, "A simple model of balancing in the plane and a simple preview balance controller," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1489–1507, 2017.

[19] B. R. P. Singh, *Angular Momentum based Balancing Control and Shock-proof Design of Legged Robots*. PhD thesis, University of Pisa, 2021.

[20] J. D. Gamba Camacho, *Hopping, Landing, and Balancing with Springs*. PhD thesis, Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS), University of Genoa, 2022.

[21] C. Semini, V. Barasuol, M. Focchi, C. Boelens, M. Emara, S. Casella, O. Villarreal, R. Orsolino, G. Fink, S. Fahmi, *et al.*, "Brief introduction to the quadruped robot HyQReal," *Istituto di Robotica e Macchine Intelligenti (I-RIM)*, 2019.

[22] Unitree, "Go1." https://www.unitree.com/en/go1, accessed Mar. 2023.

[23] M. Kamedula, N. Kashiri, and N. G. Tsagarakis, "On the kinematics of wheeled motion control of a hybrid wheeled-legged centauro robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2426–2433, 2018.

[24] Y. de Viragh, M. Bjelonic, C. D. Bellicoso, F. Jenelten, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots using linearized zmp constraints," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1633–1640, 2019.

[25] L. Bruzzone and P. Fanghella, "Mantis hybrid leg-wheel robot: Stability analysis and motion law synthesis for step climbing," in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pp. 1–6, 2014.

[26] H. Wang, H. Yu, J. Liu, L. He, Q. Li, and N. Zhao, "Design and analysis of a hybrid two-wheel-hopping robot," in *Proceedings of the 2013 International Conference on Advanced Mechatronic Systems*, pp. 363–368, 2013.

[27] J. Zhao, T. Zhao, N. Xi, M. W. Mutka, and L. Xiao, "MSU Tailbot: Controlling aerial maneuver of a miniature-tailed jumping robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2903–2914, 2015.

[28] F. Negrello, M. Garabini, M. Catalano, P. Kryczka, W. Choi, D. Caldwell, A. Bicchi, and N. Tsagarakis, "WALK-MAN humanoid lower body design optimization for enhanced physical performance," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1817–1824, 2016.

[29] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "TALOS: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 689–695, 2017.

[30] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[31] N. Carlési and A. Chemori, "Nonlinear model predictive running control of kangaroo robot: A one-leg planar underactuated hopping robot," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3634–3639, 2010.

[32] G. Zuo, Y. Liu, and X. Wang, "Design of hopping mechanism for a kangaroo-bionic robot," in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pp. 3312–3317, 2016.

[33] J. Zhang, G. Song, Y. Li, G. Qiao, A. Song, and A. Wang, "A bio-inspired jumping robot: Modeling, simulation, design, and experimental results," *Mechatronics*, vol. 23, p. 1123–1140, 12 2013.

[34] J. Zhang, G. Song, Z. Li, G. Qiao, H. Sun, and A. Song, "Self-righting, steering and takeoff angle adjusting for a jumping robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2089–2094, 2012.

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

[36] G. Yuntao, D. Gang, W. Ning, and Z. Dongxia, "The design and realization of a rotary inverted pendulum based on stm32," in *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*, pp. 185–188, 2015.

[37] I. Siradjuddin, Z. Amalia, B. Setiawan, R. P. Wicaksono, and E. Yudaningtyas, "Stabilising a cart inverted pendulum system using pole placement control method," in *2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering*, pp. 197–203, 2017.

[38] M. D. Ratolikar and R. P. Kumar, "Neural network control of an inverted pendulum on a two DoF cart moving in the vertical plane," in *2021 6th International Conference on Robotics and Automation Engineering (ICRAE)*, pp. 84–88, 2021.

[39] N. Patel and A. Borkar, "Hybrid control design for swing up and stabilization of cart pendulum system," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pp. 1051–1057, 2017.

[40] G. Sainzaya, F.-N. Yu, T.-L. Hsieh, and C.-Y. Yang, "LQR control with refined pid to balance rotary inverted pendulum with time-varying uncertainty," in *2017 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, pp. 1–6, 2017.

[41] E. Susanto, B. Rahmat, and M. Ishitobi, "Stabilization of rotary inverted pendulum using proportional derivative and fuzzy controls," in *2022 9th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp. 34–37, 2022.

[42] S.-H. Lee and A. Goswami, "Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 4667–4672, 2007.

[43] L. Chi-Yen, Y. Shuo, B. Benjamin, and M. Zachary, "Enhanced balance for legged robots using reaction wheels," in *2023 International Conference on Robotics and Automation (ICRA)*, 2023.

[44] C. Carignan and D. Akin, "The reaction stabilization of on-orbit robots," *IEEE Control Systems Magazine*, vol. 20, no. 6, pp. 19–33, 2000.

[45] A. M. Oluwatosin, Y. Hamam, and K. Djouani, "Attitude control of a cubesat in a circular orbit using reaction wheels," in *2013 Africon*, pp. 1–8, 2013.

[46] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the reaction wheel pendulum," *Automatica*, vol. 37, no. 11, pp. 1845–1851, 2001.

[47] F. Jepsen, A. Soborg, A. R. Pedersen, and Z. Yang, "Development and control of an inverted pendulum driven by a reaction wheel," in *2009 International Conference on Mechatronics and Automation*, pp. 2829–2834, 2009.

[48] M. Gajamohan, M. Muehlebach, T. Widmer, and R. D'Andrea, "The Cubli: A reaction wheel based 3D inverted pendulum," in *2013 European Control Conference (ECC)*, pp. 268–274, 2013.

[49] F. Xue, Z. Hou, and H. Deng, "Balance control for an acrobot," in *2011 Chinese Control and Decision Conference (CCDC)*, pp. 3426–3429, 2011.

[50] M. Azad and R. Featherstone, "Balancing control algorithm for a 3D under-actuated robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3233–3238, 2014.

[51] S. Caux, E. Mateo, and R. Zapata, "Balance of biped robots: special double-inverted pendulum," in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, vol. 4, pp. 3691–3696, 1998.

[52] M. Spong and D. Block, "The pendubot: a mechatronic system for control research and education," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1, pp. 555–556 vol.1, 1995.

[53] L. Bai, W.-j. Ge, X.-h. Chen, and X.-y. Meng, "Hopping capabilities of a bio-inspired and mininally actuated hopping robot," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, pp. 1485–1489, 2011.

[54] V. Zaitsev, O. Gvirsman, U. Ben Hanan, A. Weiss, A. Ayali, and G. Kosa, "Locust-inspired miniature jumping robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 553–558, 2015.

[55] U. Scarfogliero, C. Stefanini, and P. Dario, "Design and development of the long-jumping "grillo" mini robot," in *Proceedings 2007 IEEE International conference on robotics and automation*, pp. 467–472, IEEE, 2007.

[56] M. Noh, S.-W. Kim, S. An, J.-S. Koh, and K.-J. Cho, "Flea-inspired catapult mechanism for miniature jumping robots," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1007–1018, 2012.

[57] C. Zhang, W. Zou, L. Ma, and Z. Wang, "Biologically inspired jumping robots: A comprehensive review," *Robotics and Autonomous Systems*, vol. 124, p. 103362, 2020.

[58] J. Zhao, J. Xu, B. Gao, N. Xi, F. J. Cintrón, M. W. Mutka, and L. Xiao, "MSU Jumper: A single-motor-actuated miniature steerable jumping robot," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 602–614, 2013.

[59] M. Berkemeier and R. Fearing, "Sliding and hopping gaits for the underactuated acrobot," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 629–634, 1998.

[60] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, vol. 22, no. 11, pp. 1217–1227, 1989.

[61] T. A. McMahon and G. C. Cheng, "The mechanics of running: How does stiffness couple with speed?," *Journal of Biomechanics*, vol. 23, pp. 65–78, 1990. International Society of Biomechanics.

[62] K. Harbick and G. Sukhatme, "Controlling hopping height of a pneumatic monopod," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, pp. 3998–4003 vol.4, 2002.

[63] Z. Batts, J. Kim, and K. Yamane, "Untethered one-legged hopping in 3D using linear elastic actuator in parallel (leap)," in *2016 International Symposium on Experimental Robotics* (D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, eds.), (Cham), pp. 103–112, Springer International Publishing, 2017.

[64] P. Terry and K. Byl, "Com control for underactuated 2D hopping robots with series-elastic actuation via higher order partial feedback linearization," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 7795–7801, 2015.

[65] S. Giewont and F. Sahin, "Delta-Quad: An omnidirectional quadruped implementation using parallel jointed leg architecture," in *2017 12th System of Systems Engineering Conference (SoSE)*, pp. 1–6, 2017.

[66] D. J. Blackman, J. V. Nicholson, C. Ordonez, B. D. Miller, and J. E. Clark, "Gait development on a direct drive , quadrupedal robot," in *Unmanned Systems Technology XVIII* (R. E. Karlsen, D. W. Gage, C. M. Shoemaker, and G. R. Gerhart, eds.), vol. 9837, p. 98370I, International Society for Optics and Photonics, SPIE, 2016.

[67] D. J. Blackman, J. V. Nicholson, J. L. Pusey, M. P. Austin, C. Young, J. M. Brown, and J. E. Clark, "Leg design for running and jumping dynamics," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2617–2623, 2017.

[68] M. H. Raibert, J. H. Benjamin Brown, and M. Chepponis, "Experiments in balance with a 3D one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[69] Y. Nakata, A. Ide, Y. Nakamura, K. Hirata, and H. Ishiguro, "Hopping of a monopedal robot with a biarticular muscle driven by electromagnetic linear actuators," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3153–3160, 2012.

[70] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Task-based limb optimization for legged robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2062–2068, 2016.

[71] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3345–3351, 2017.

[72] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation," *Science Robotics*, vol. 1, no. 1, p. eaag2048, 2016.

[73] B. Brown and G. Zeglin, "The bow leg hopping robot," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 781–786, IEEE, 1998.

[74] G. Zeglin and B. Brown, "Control of a bow leg hopping robot," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, pp. 793–798, 1998.

[75] T. Tsujita, T. Kitahara, R. Tahara, *et al.*, "Drop test for evaluating effect of cushioning material and servo gain on parachute landing impact using a small one-legged robot," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2474–2479, 2017.

[76] B. R. P. Singh and R. Featherstone, "Mechanical shock propagation reduction in robot legs," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1183–1190, 2020.

[77] S. Zihajehzadeh, D. Loh, T. J. Lee, R. Hoskinson, and E. J. Park, "A cascaded Kalman filter-based GPS/MEMS-IMU integration for sports applications," *Measurement*, vol. 73, pp. 200–210, 2015.

[78] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, 07 2015.

[79] J. Liang, H. Duan, J. Li, H. Sun, X. Sha, Y. Zhao, and L. Liu, "Accurate estimation of gait altitude using one wearable IMU sensor," in *2018 IEEE 1st International Conference on Micro/Nano Sensors for AI, Healthcare, and Robotics (NSENS)*, pp. 64–67, 2018.

[80] K. Abdulrahim, C. Hide, T. Moore, and C. Hill, "Aiding MEMS IMU with building heading for indoor pedestrian navigation," in *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service*, pp. 1–6, 2010.

[81] A. Safaeifar and A. Nahvi, "Drift cancellation of an orientation tracker for a virtual reality head-mounted display," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pp. 296–301, 2015.

[82] M. Li, Z. Jiang, P. Wang, L. Sun, and S. Sam Ge, "Control of a quadruped robot with bionic springy legs in trotting gait," *Journal of Bionic Engineering*, vol. 11, no. 2, pp. 188–198, 2014.

[83] S. Zihajehzadeh, D. Loh, M. Lee, R. Hoskinson, and E. Park, "A cascaded two-step Kalman filter for estimation of human body segment orientation using MEMS-IMU," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6270–6273, 2014.

[84] T. Tan, Z. A. Strout, H. Xia, M. Orban, and P. B. Shull, "Magnetometer-free, IMU-based foot progression angle estimation for real-life walking conditions," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 282–289, 2021.

[85] S. Cardarelli, A. Mengarelli, A. Tigrini, A. Strazza, F. Di Nardo, S. Fioretti, and F. Verdini, "Single IMU displacement and orientation estimation of human center of mass: A magnetometer-free approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 8, pp. 5629–5639, 2020.

[86] D. Yang, J. Huang, X. Tu, G. Ding, T. Shen, and X. Xiao, "A wearable activity recognition device using air-pressure and IMU sensors," *IEEE access*, vol. 7, pp. 6611–6621, 2018.

[87] W. Qi and A. Aliverti, "A multimodal wearable system for continuous and real-time breathing pattern monitoring during daily activity," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 8, pp. 2199–2207, 2020.

[88] J. S. Arlotti, W. O. Carroll, Y. Afifi, P. Talegaonkar, L. Albuquerque, J. E. Ball, H. Chander, A. Petway, *et al.*, "Benefits of IMU-based wearables in sports medicine: Narrative review," *International Journal of Kinesiology and Sports Science*, vol. 10, no. 1, pp. 36–43, 2022.

[89] S. Kajita, K. Yokoi, M. Saigo, and K. Tanie, "Balancing a humanoid robot using backdrive concerned torque control and direct angular momentum feedback," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, pp. 3376–3382 vol.4, 2001.

[90] X. Xin, M. Ono, S. Izumi, T. Yamasaki, and K. Zhang, "Angular momentum based stabilizing control of underactuated multi-link planar robots with last active joint," in *2018 Annual American Control Conference (ACC)*, pp. 1939–1944, 2018.

[91] M. Azad and R. Featherstone, "Angular momentum based balance controller for an under-actuated planar robot," *Autonomous Robots*, vol. 40, no. 1, pp. 93–107, 2016.

[92] C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D. G. Caldwell, and C. Semini, "Line walking and balancing for legged robots with point feet," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3649–3656, 2020.

[93] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 8, pp. 49785–49797, 2020.

[94] M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter, "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2116–2123, 2019.

[95] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5120–5126, IEEE, 2023.

[96] L. J. Pinto, D.-H. Kim, J. Y. Lee, and C.-S. Han, "Development of a segway robot for an intelligent transport system," in *2012 IEEE/SICE International Symposium on System Integration (SII)*, pp. 710–715, 2012.

[97] S. Chantarachit, "Development and control segway by LQR adjustable gain," in *2019 International Conference on Information and Communications Technology (ICOIACT)*, pp. 649–653, 2019.

[98] H. Chen, B. Wang, Z. Hong, C. Shen, P. M. Wensing, and W. Zhang, "Underactuated motion planning and control for jumping with wheeled-bipedal robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 747–754, 2021.

[99] I. D. Basnayake, T. W. U. Madhushani, and D. H. S. Maithripala, "Intrinsic pid controller for a segway type mobile robot," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pp. 1–6, 2017.

[100] R. Babazadeh, A. G. Khiabani, and H. Azmi, "Optimal control of segway personal transporter," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, pp. 18–22, 2016.

[101] M. M. Azimi and H. R. Koofigar, "Model predictive control for a two wheeled self balancing robot," in *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pp. 152–157, 2013.

[102] M. Okulski and M. Ławryńczuk, "Development of a model predictive controller for an unstable heavy self-balancing robot," in *2018 23rd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 503–508, 2018.

[103] T. Lauwers, G. Kantor, and R. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2884–2889, 2006.

[104] U. Nagarajan, G. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, 2014.

[105] S.-M. Lee and B. S. Park, "Robust control for trajectory tracking and balancing of a ballbot," *IEEE Access*, vol. 8, pp. 159324–159330, 2020.

[106] M. U. Draz, M. S. Ali, M. Majeed, U. Ejaz, and U. Izhar, "Segway electric vehicle," in *2012 International Conference of Robotics and Artificial Intelligence*, pp. 34–39, 2012.

[107] F. Allione, B. R. P. Singh, A. E. Gkikakis, and R. Featherstone, "Mechanical shock testing of incremental and absolute position encoders," in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 52–57, 2021.

[108] F. Allione, J. D. Gamba, A. E. Gkikakis, R. Featherstone, and D. Caldwell, "Effects of repetitive low-acceleration impacts on attitude estimation with micro-electromechanical inertial measurement units," *Frontiers in Robotics and AI*, vol. 10, 2023.

[109] RLS, "RLS AksIM-2 absolute encoders." https://www.rls.si/eng/, accessed Mar. 2023.

[110] Maxon Group, "ENX 16 EASY." https://www.maxongroup.com/, accessed Mar. 2023.

[111] Misumi, "Aluminum Extrusion 5 Series/slot width 6/20x40mm, Parallel Surfacing." https://uk.misumi-ec.com/, accessed Mar. 2023.

[112] A. E. Gkikakis, D. Kanoulas, and R. Featherstone, "Autonomous real time architecture for high performance mobile robots," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 841–846, 2021.

[113] Texas Instruments, "Enhanced Quadrature Encoder Pulse (eQEP) Module." https://www.ti.com/, accessed Mar. 2023.

[114] International Electrotechnical Commission (IEC), "Environmental testing – Part 2-27: Tests – Test Ea and guidance: Shock." International Standard IEC 60068-2-27, ed. 4.0, 2008-2, 2008.

[115] W. Peterson and D. Brown, "Cyclic Codes for Error Detection," *Proc. the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[116] J. K. Yim, B. R. P. Singh, E. K. Wang, R. Featherstone, and R. S. Fearing, "Precision robotic leaping and landing using stance-phase balance," *IEEE RA-L*, vol. 5, no. 2, pp. 3422–3429, 2020.

[117] Magnet Schultz, "Proportional Rotary Solenoids Type G DR." https://www.magnet-schultz.com/en/rotary-solenoids/proportional-rotary-solenoids-type-g-dr/, accessed Mar. 2023.

[118] Pololu, "Pololu G2 High-Power Motor Driver 24v21." https://www.pololu.com/product/2995, accessed Mar. 2023.

[119] RLS, "RLS AksIM-2 absolute encoders." https://www.rls.si/eng/, accessed Mar. 2023.

[120] National Instruments, "Controller Single-Board CompactRIO FPGA." https://www.ni.com/it-it/support/model.sbrio-9637.html, accessed Mar. 2023.

[121] VECTORNAV, "VECTORNAV Inertial Measurement Unit." https://www.vectornav.com/products/detail/vn-100, accessed Mar. 2023.

[122] Lord MicroStrain, "3DMGX5-AR." https://www.microstrain.com/inertial-sensors/3dm-gx5-15, accessed Mar. 2023.

[123] Lord MicroStrain, "Desktop Sensing Software." https://www.microstrain.com/software/sensorconnect, accessed Mar. 2023.

[124] BOSCH, "Smart Sensor: BNO055." https://www.bosch-sensortec.com/products/smart-sensors/bno055/, accessed Mar. 2023.

[125] DFRobot, "Gravity: BNO055+BMP280 intelligent 10DOF AHRS." https://www.dfrobot.com/product-1793.html, accessed Mar. 2023.

[126] Adafruit Industries, "BNO055 Absolute Orientation Sensor." https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor, accessed Mar. 2023.

[127] Arduino, "Arduino Uno Board." https://store.arduino.cc/products/arduino-uno-rev3, accessed Mar. 2023.

[128] C. Koga, K. Miyase, and M. Tokui, "Analyzing running form with acceleration sensor," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–5, 2020.

[129] A. Bhattacharya, E. McCutcheon, E. Shvartz, and J. Greenleaf, "Body acceleration distribution and o2 uptake in humans during running and jumping," *Journal of applied physiology: respiratory, environmental and exercise physiology*, vol. 49, pp. 881–7, 12 1980.

[130] R. Featherstone, "A new simple model of balancing in the plane," in *Proc. Int. Symp. Robotics Research*, Sestri Levante, Italy, Sept. 12–15, 2015.

[131] R. Featherstone, "Control of absolute motion while balancing in 2D," in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 121–127, 2021.

[132] N. Miyashita, M. Kishikawa, and M. Yamakita, "3D motion control of 2 links (5 D.O.F.) underactuated manipulator named acrobox," in *2006 American Control Conference*, pp. 6 pp.–, 2006.

[133] R. Featherstone, "Quantitative measures of a robot's physical ability to balance," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1681–1696, 2016.

[134] Texas Instruments, "TMS320F28377S." https://www.ti.com/, accessed Mar. 2022.

[135] F. Allione, R. Featherstone, P. M. Wensing, and D. Caldwell, "Balancing on a rolling contact," *IEEE Robotics and Automation Letters - Early Access*, pp. 1–8, 2023. DOI: 10.1109/LRA.2023.3326696.

[136] R. Featherstone, *Rigid Body Dynamics Algorithms*, pp. 39–64. Boston, MA: Springer US, 2008.

[137] G. Zuo, X. Wang, D. Gong, and Y. Liu, "Dynamic modeling and balance control for bionic kangaroo robot during stance phase," in *2016 Chinese Control and Decision Conference (CCDC)*, pp. 5154–5157, 2016.

[138] J. Grizzle, C. Moog, and C. Chevallereau, "Nonlinear control of mechanical systems with an unactuated cyclic variable," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 559–576, 2005.

[139] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the hydraulically actuated, torque-controlled quadruped robot HyQ2Max," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 635–646, 2017.

[140] G. Zeglin, *The Bow Leg Hopping Robot*. PhD thesis, Carnegie Mellon University Pittsburgh, Pennsylvania, 1999.

[141] J. J. Driessen, *Design of high-performance legged robots: A case study on a hopping and balancing robot*. PhD thesis, Università degli Studi di Genova, 2019.

Appendix A

RWP Dynamic Parameters Estimation

This appendix describes the experimental methodology used to determine the dynamic parameters required by the Reaction Wheel Pendulum (RWP) balance controller. Such values are used as a starting point and then manually tuned to compensate for measurement inaccuracies while doing experiments with the real robot. The dynamic parameters are three and are the First Moment of Mass, the Time Constant of Toppling and the Angular Velocity Gain.

A.1 First Moment of Mass

This section presents the procedure used to estimate experimentally the first moment of mass of Skippy's Head when configured as a RWP. The first moment of mass, mc, is the product of the mass of the robot m and the distance of the robot's centre of mass c with respect to the rotation axis of the pendulum.

The experimental procedure consists in placing the whole robot on its side, as shown in Figure A.1, so that it is supported at two points having the same height and at a known distance d; one of the support point is on a scale. Get the scale reading and call it m_2. Then measure the angle between the dashed line and the horizontal (the dashed line is aligned with the vertical when the robot balances). The angle θ is obtained as the difference between two IMU readings when the robot is vertical and lying on its side. The robot is held manually to obtain the vertical measurement ($q_1 = 0$), and the IMU is read when a spirit level placed on top of the Head is perfectly horizontal. By exploiting the following equations and knowing

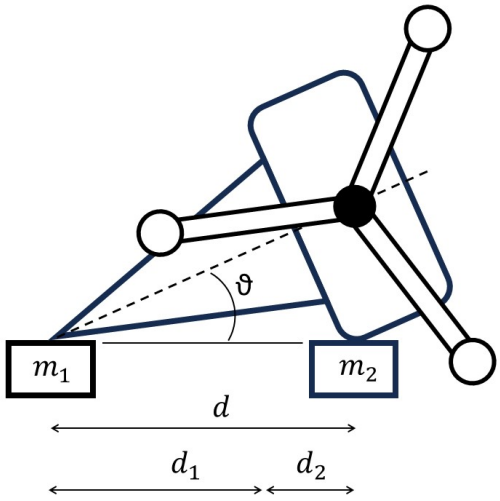


Figure A.1 Drawing for the first moment of mass mc.

the angle θ it is possible to get the value of mc.

$$\begin{aligned} m_1 + m_2 &= m \\ d_1 + d_2 &= d \\ d_1 m_1 &= d_2 m_2 \end{aligned} \tag{A.1}$$

The measured physical parameters of the robot used to estimate its first moment of mass are $m_2 = 1.75\,\text{kg}$, $d = 0.29\,\text{m}$, and $\theta = 0.31\,\text{rad}$. Then, the first moment of mass obtained after trigonometric calculations is $mc = m_2 d/\cos(\theta) = 0.53\,\text{kg m}$.

A.2 Time Constant of Toppling

This section presents the procedure used to estimate the Time Constant of Toppling T_c experimentally of the RWP balancing machine. T_c is the rate at which the robot starts to fall when there is no movement in the actuated joint. It is a physical property of the robot and varies depending on its structure and configuration. It can be measured by manually holding the robot very close to its balanced configuration, then letting the robot fall while logging its angle with respect to the vertical, q_1. The following step consists in fitting using the least squares algorithm the logged data to the curve

$$q_1 = A + B e^{-t/T_\text{c}} + C e^{t/T_\text{c}} \tag{A.2}$$

where A, B, C and T_c are all unknown parameters. A accounts for a possible error in the vertical direction, and B and C depend on the initial conditions. As this approach depends on a small-angle approximation, the robot can fall up to $q_1 = 0.1$ before discarding the remaining data. The data are obtained via IMU measurements. The experiment has been performed four times, and the average value of the time constant of toppling is $T_c = 0.182 \pm 0.005\,\mathrm{s}$.

A.3 Angular Velocity Gain

This section presents the procedure used to estimate the Angular Velocity Gain G_ω experimentally. G_ω provides a quantitative measure of a robot's physical ability to balance. It is defined as the ratio of the change in the robot's angle with respect to the vertical, $\dot{q}_1$ (measured with the IMU's gyroscope), to the change in velocity of the joint used to balance the robot, $\dot{q}_2$ (q_2 is measured with the absolute position encoder and then numerically differentiated to get $\dot{q}_2$), when both changes are caused by an impulse at that joint. The procedure consists in placing the robot in a configuration very close to balance (e.g. just 1 mm on the side of perfect balance) and manually holding it in that position. Then a short high-voltage pulse is sent to the crossbar motor in a direction that will make the robot reach the balanced position. After the experiment, the velocity ratio between the vertical $\dot{q}_1$ and the crossbar joint $\dot{q}_2$ is calculated. The experiment has been performed four times and the average of estimated velocity gain following this procedure is $G_\omega = -0.068 \pm 0.004$.

Appendix B

Skippy's Kinematics

This appendix presents the code used to perform Skippy's kinematic calculations in 2D. The Matlab function *hopper6gq*, described below, performs its calculations according to the old kinematic model as appears in Chapter 7, not the new one that appears in this appendix.

The content of this appendix is part of the outcome of the Skippy team research. More specifically, I used the already developed Matlab functions (which rely on the 4-bar mechanism kinematic model) to control the robot in simulation. Then, I converted them into C code and used them to control the real robot.

B.1 4-bar Linkage Kinematics

This section describes how to find the input/output relationship of the Skippy's 4-bar mechanism shown in Figure B.1. The input is the angle B and the output is $E+F+\pi$, with $E+F-\pi$ varying in the range $\pm\pi/2$ and the gear ratio from input to output approximately equals to 2.

The diagram in Figure B.1 can be described by the equations below.

$$
\begin{aligned}
a &= b\cos(B)+e\cos(E) & a^2 &= b^2+e^2-2be\cos(F-C)\\
a &= d\cos(A)+f\cos(G) & a^2 &= d^2+f^2-2df\cos(H-D)\\
b &= a\cos(B)+e\cos(F-C) & b^2 &= a^2+e^2-2ae\cos(E)\\
b &= c\cos(C)+f\cos(G-B) & b^2 &= c^2+f^2-2cf\cos(H)\\
c &= b\cos(C)+f\cos(H) & c^2 &= b^2+f^2-2bf\cos(G-B)\\
c &= d\cos(D)+e\cos(F) & c^2 &= d^2+e^2-2de\cos(E-A)
\end{aligned}
$$

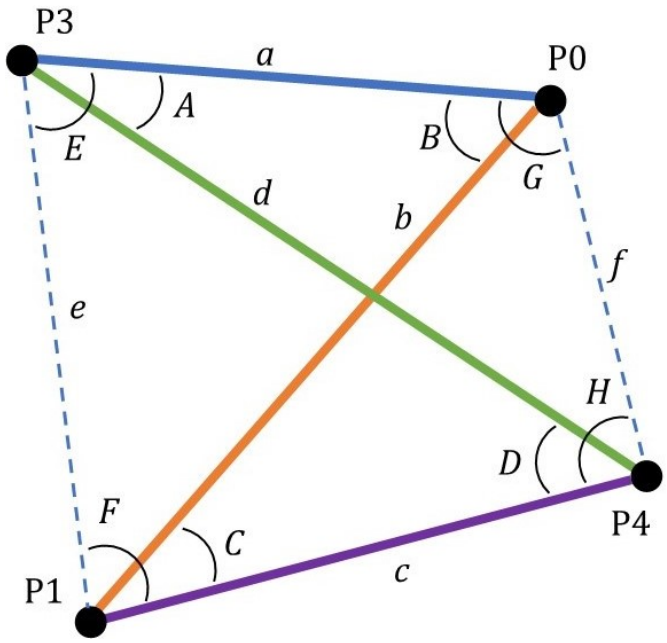


Figure B.1 Kinematic diagram of Skippy's 4-bar mechanism shown in Figure B.2. P0 and P3 are fixed in the torso, and P1 and P4 are fixed in the leg. P0-P1 is the lower part of the lever, and P3-P4 is the follower. a, b, c and d corresponds to the parameters bar4_a, bar4_b, bar4_c and bar4_d respectively.

$$
\begin{aligned}
d &= a\cos(A) + f\cos(H-D) & d^2 &= a^2 + f^2 - 2af\cos(G) \\
d &= c\cos(D) + e\cos(E-A) & d^2 &= c^2 + e^2 - 2ce\cos(F) \\
e &= c\cos(F) + d\cos(E-A) & e^2 &= a^2 + b^2 - 2ab\cos(B) \\
e &= a\cos(E) + b\cos(F-C) & e^2 &= c^2 + d^2 - 2cd\cos(D) \\
f &= a\cos(G) + d\cos(H-D) & f^2 &= a^2 + d^2 - 2ad\cos(A) \\
f &= c\cos(H) + b\cos(G-B) & f^2 &= b^2 + c^2 - 2bc\cos(C)
\end{aligned}
$$

$$
\begin{aligned}
&\sin(A)/f = \sin(G)/d = \sin(H-D)/a \\
&\sin(B)/e = \sin(E)/b = \sin(F-C)/a \\
&\sin(C)/f = \sin(H)/b = \sin(G-B)/c \\
&\sin(D)/e = \sin(F)/d = \sin(E-A)/c \\
&A + B = C + D
\end{aligned}
$$

B.1.1 Forward Kinematics

The forward kinematics is performed in three steps

1. Solve for e using $e^2 = a^2 + b^2 - 2ab\cos(B)$.
2. Solve for E using either $\cos(E) = (a - b\cos(B))/e$ or $\cos(E) = (a^2 + e^2 - b^2)/(2ae)$.
3. Solve for F using $\cos(F) = (c^2 + e^2 - d^2)/(2ce)$.

B.1.2 Inverse Kinematics

The inverse kinematics is not used in this thesis. Nevertheless, the more efficient way to obtain the inverse kinematics is through numerical approximation (i.e. using a least-squares polynomial fit as calculated by the Matlab function *polyfit*).

B.1.3 Velocity Kinematics

The equations below are used to calculate the velocity kinematics of the 4-bar mechanism.

$$\frac{d}{dt}(e^2 = a^2 + b^2 - 2ab\cos(B)) \Rightarrow 2e\dot{e} = 2ab\sin(B)\dot{B} \Rightarrow \dot{e} = \frac{ab}{e}\sin(B)\dot{B}$$

$$\frac{d}{dt}(e\cos(E) = a - b\cos(B)) \Rightarrow \dot{e}\cos(E) - e\sin(E)\dot{E} = b\sin(B)\dot{B} \Rightarrow$$

$$\Rightarrow \dot{E} = \frac{\dot{e}\cos(E) - b\sin(B)\dot{(B)}}{e\sin(E)} = \frac{\dot{e}\cos(E) - e\dot{e}/a}{e\sin(E)} = (\frac{a\cos(E) - e}{a\sin(e)})\frac{\dot{e}}{e}$$

$$\frac{d}{dt}(2ce\cos(F) = c^2 + e^2 - d^2) \Rightarrow 2c\dot{e}cos(F) - 2cesin(F)\dot{(F)} = 2e\dot{e} \Rightarrow$$

$$\Rightarrow \dot{F} = \frac{ce\cos(F) - e\dot{e}}{ce\sin(F)} = (\frac{c\cos(F) - e}{c\sin(F)})\frac{\dot{e}}{e}$$

and combining them together we get

$$\dot{E} + \dot{F} = (\frac{a\cos(E) - e}{a\sin(E)} + \frac{c\cos(F) - e}{c\sin(F)})\frac{\dot{e}}{e} = (\frac{a\cos(E) - e}{a\sin(E)} + \frac{c\cos(F) - e}{c\sin(F)})\frac{ab}{e^2}\sin(B)\dot{B}$$

B.2 Matlab Functions

Matlab functions used to calculate Skippy's kinematics.

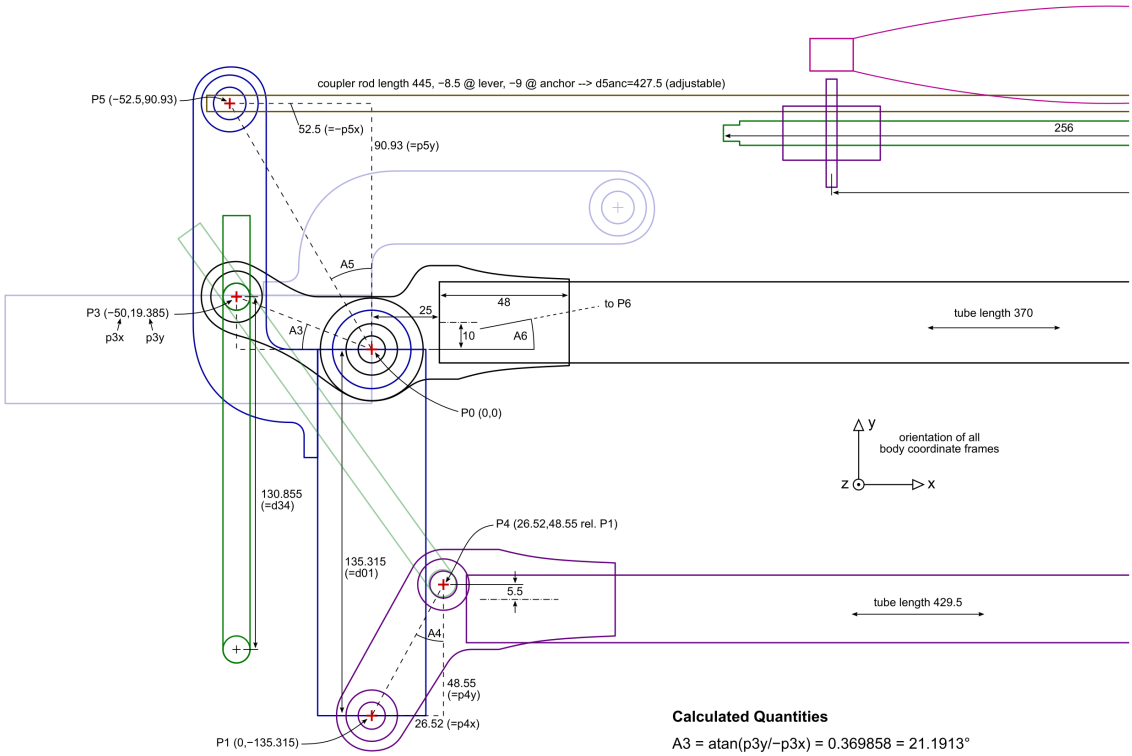
```
function  [qn,qdn,G,gs] = hopper6gq( robot, qo, qdo )
% hopper6gq  gamma_q function for hopper6
% [qn,qdn,G,gs]=hopper6gq(robot,qo,qdo)  calculates the
% kinematic constraint data for hopper6 in the format
% expected of a spatial_v2 gamma_q function.
% (See spatial_v2 documentation.)  qn and qdn are joint
```

```
% position and velocity vectors that satisfy the kinematic
% constraints exactly, and are calculated from the
% independent variables in qo and qdo, which are elements
% 1, 2, 4, 5 and 10.  G is the 10x5 Jacobian that maps
% the vector of independent velocities to the full
% velocity vector, as in qd=G*yd, where
% yd=qd([1,2,4,5,10]); and gs contains both the velocity
% product terms and the Baumgarte stabilization terms
% for use in the acceleration calculation formula
% qdd=G*ydd+gs.  See separate documentation for an
% explanation of the kinematics calculations.

qn = qo;
qdn = qdo;
G = zeros(10,5);
g = zeros(10,1);

% Independent Variables
G(1,1) = 1;
G(2,2) = 1;
G(4,3) = 1;
G(5,4) = 1;
G(10,5) = 1;

% Rolling Contact at Ground
qn(3) = qn(4) * -robot.kine.rtoe;
qdn(3) = qdn(4) * -robot.kine.rtoe;
G(3,3) = -robot.kine.rtoe;

% First Triangle (rocker, lever, torso)
a = robot.kine.d56 - qn(10);
b = robot.kine.d05;
c = robot.kine.d06;
[ABC,G1,g1] = R3Pkin( [], a, b, c, -qdn(10) );
qn(7) = -ABC(1) + pi/2 - robot.kine.a6 + robot.kine.a5 -
   robot.kine.a3;
qn(9) = -ABC(2) + robot.kine.a6;
G(7,5) = G1(1);
G(9,5) = G1(2);
qdn(7) = G(7,5) * qdn(10);
qdn(9) = G(9,5) * qdn(10);
g(7) = -g1(1);
g(9) = -g1(2);
```

```
% Second and Third Triangles (hip 4-bar linkage)
a = robot.kine.d03;
b = robot.kine.d01;
c = robot.kine.d14;
d = robot.kine.d34;
B = pi/2 - qn(7);
Bd = -qdn(7);
[eFmCE,G2,g2] = R3Pkin(B,[],a,b,Bd); % eFmCE=[e;F-C;E]
e = eFmCE(1);
ed = G2(1) * Bd;
[DEmAF,G3,g3] = R3Pkin([],e,c,d,ed); % DEmAF=[D;E-A;F]
C = DEmAF(3) - eFmCE(2);
A = eFmCE(3) - DEmAF(2);
qn(6) = C - pi/2 + robot.kine.a3 + robot.kine.a4;
qn(8) = pi/2 - A;
Cd = G3(3)*ed - G2(2)*Bd;
Ad = G2(3)*Bd - G3(2)*ed;
qdn(6) = Cd;
qdn(8) = -Ad;
G(6,5) = (G3(3)*G2(1) - G2(2)) * -G(7,5);
G(8,5) = (G3(2)*G2(1) - G2(3)) * -G(7,5);
g(6) = (G3(3)*G2(1)-G2(2))*-g(7)+G3(3)*g2(1)+g3(3)-g2(2);
g(8) = (G3(2)*G2(1)-G2(3))*-g(7)+G3(2)*g2(1)+g3(2)-g2(3);

% Stabilization terms
Tstab = 0.02;
gs = g + (qdn-qdo)*2/Tstab + (qn-qo)/Tstab^2;
end

function  [oBC,G,g] = R3Pkin( A, a, b, c, vel )
% R3Pkin  pos/vel/accn kinematics of a triangular RRRP
% mechanism. This function calculates the position,
% velocity and acceleration kinematics of a triangular
% RRRP mechanism with revolute joints at the corners
% of a triangle ABC, having sides a (=BC), b (=CA) and
% c (=AB), and a prismatic joint on side a.  In effect,
% this is the kinematics of a triangle with two constant
% and one variable side.  Two calls are possible, depending
% on whether A or a is the input variable:
%   [aBC,G,g] = R3Pkin(A,[],b,c,Ad)  in: A,  out: a, B, C
%   [ABC,G,g] = R3Pkin([],a,b,c,ad)  in: a,  out: A, B, C
% where A, B and C are interior angles; a, b and c are
% lengths; aBC=[a;B;C], ABC=[A;B;C], and Ad and ad are the
```

```
% time derivatives of A and a.  G is a 3x1 Jacobian matrix
% and g is a 3x1 bias vector.  G and g are calculated only
% if requested by the caller (i.e., nargout==2 or 3). Ad
% and ad are optional, and are needed only to calculate g.
%
% Case 1:  A-->a,B,C
% Argument A may take any value, but b and c must be
% non-negative and b+c must be positive.  Return value a
% is strictly positive, and B and C are in the range
% -pi...pi and have the same sign as sin(A).  G maps Ad
% to aBCd (=d(aBC)/dt) in the velocity equation aBCd=G*Ad;
% and g contains the velocity terms in the acceleration
% equation aBCdd=G*Add+g.  Accuracy decreases as a
% approaches zero; and the function generates an error
% if a==0, which happens if b==c and cos(A)==1.
%
% Case 2:  a-->A,B,C
% Arguments a, b and c must all be strictly positive and
% must satisfy the triangle inequality.  Return values
% A, B and C are in the range 0...pi.  G maps ad to
% ABCd (=d(ABC)/dt) in the velocity equation ABCd=G*ad;
% and g contains the velocity terms in the acceleration
% equation ABCdd=G*add+g. Accuracy decreases as the
% triangle approaches degeneracy.

if (length(a)==0) == (length(A)==0)
  error('exactly one of arguments A and a must be empty');
end
if nargout > 2 && nargin < 5
  error('velocity argument required if return value g
     requested');
end
if length(a)==0          % Case 1: A-->a,B,C
    cosA = cos(A);
    sinA = sin(A);

    if b < 0 || c < 0 || b+c <= 0 || b == c && cosA == 1
      error('case A-->a,B,C requires b,c>=0, b+c>0 and if b
         ==c then cos(A)<1');
    end
    a2 = b^2 + c^2 -2*b*c*cosA;
    a = sqrt(a2);
    B = atan2( b*sinA, c-b*cosA );
```

```
    C = atan2( c*sinA, b-c*cosA );

    oBC = [a;B;C];

    if nargout > 1      % calculate G
      Ga = b*c * sinA/a;
      GB = (c^2 - b^2 - a2) / (2*a2);
      G = [Ga;GB;-1-GB];
    end
    if nargout > 2      % calculate g (needs Ad)
      Ad2 = vel^2;
      ga = Ad2 * (b*c*cosA - Ga^2) / a;
      gB = Ad2 * Ga * (b^2-c^2) / (a*a2);
      g = [ga;gB;-gB];
    end
else    % Case 2:  a-->A,B,C
    if a <= 0 || b <= 0 || c <= 0 || a >= b+c || a <= abs(b
       -c)
      error('case a-->A,B,C requires a,b,c>0, a<b+c and a>|
         b-c|');
    end

    cosA = (b^2 + c^2 - a^2) / (2*b*c);
    cosB = (c^2 + a^2 - b^2) / (2*c*a);
    cosC = (a^2 + b^2 - c^2) / (2*a*b);

    oBC = acos( [cosA;cosB;cosC] );

    if nargout > 1      % calculate G
      sinA = sin(oBC(1));
      GA = a / (b*c*sinA);
      GB = -cosC / (c*sinA);
      G = [GA;GB;-GA-GB];
    end
    if nargout > 2      % calculate g (needs ad)
      ad2 = vel^2;
      sinB = sinA * (b/a);
      gA = ad2 * (1/(b*c) - cosA*GA^2) / sinA;
      gB = ad2 * ((b^2-c^2)/(c*a^3) - cosB*GB^2) / sinB;
      g = [gA;gB;-gA-gB];
    end
end
end
```



Given Quantities

p2x p2y p3x p3y
p4x p4y p5x p5y
p6x p6y p7x p7y
p8x p8y p9x p9y
d01 d34 d2toe rtoe
d5anc ms_L as_L

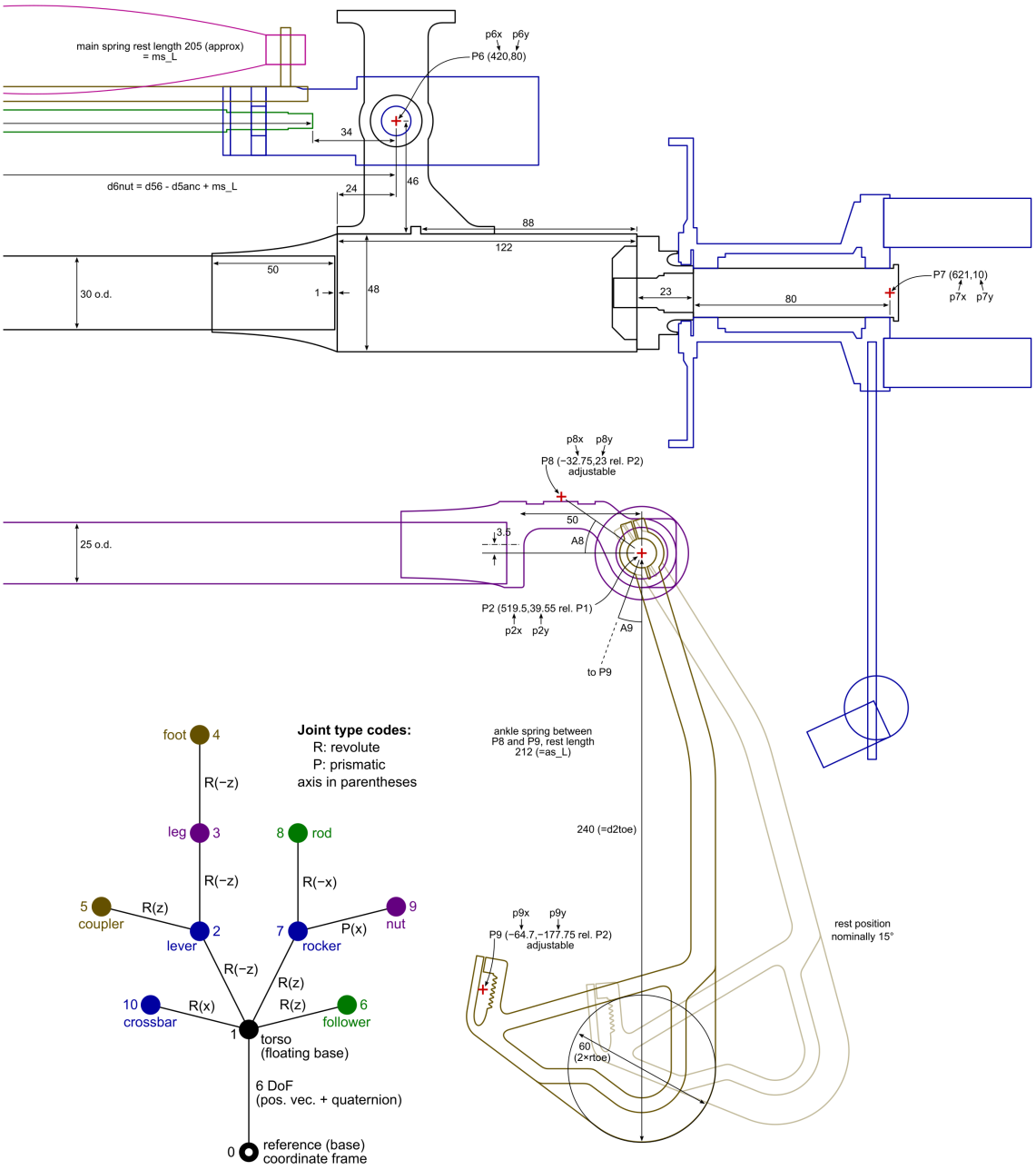
Body Coordinate Frame Origins

1	torso	P0
2	lever	P0
3	leg	P1
4	foot	P2
5	coupler	P5
6	follower	P3
7	rocker	P6
8	rod	P6
9	nut	nut centre
10	crossbar	P7

Calculated Quantities

A3 = atan(p3y/−p3x) = 0.369858 = 21.1913°
A4 = atan(p4y/p4x) = 1.070844 = 61.3548°
A5 = atan(−p5x/p5y) = 0.523611 = 30.0007°
A6 = atan(p6y/p6x) = 0.188222 = 10.7843°
A8 = atan(p8y/−p8x) = 0.612261 = 35.08°
A9 = atan(−p9x/−p9y) = 0.349087 = 20.0012°
bar4_a = sqrt($p3x^2$+$p3y^2$) = 0.0536263
bar4_b = d01 = 0.135315
bar4_c = sqrt($p4x^2$+$p4y^2$) = 0.055321
bar4_d = sqrt((p4x−p3x)^2+(d01+p3y−p4y)^2) = 0.1308554
bar4_phi = π/2 − A3 − A4 = 0.130094
bar4_Bmin = A3
bar4_Bmax = A3 + π/2
d05 = sqrt($p5x^2$+$p5y^2$) = 0.1049977
d06 = sqrt($p6x^2$+$p6y^2$) = 0.4275512
d28 = aslv_len = sqrt($p8x^2$+$p8y^2$) = 0.0400195
d29 = sqrt($p9x^2$+$p9y^2$) = 0.1891591
d56 = sqrt((p6x−p5x)^2+(p6y−p5y)^2) = 0.4726264
d6nut = d56 - d5anc + ms_L = 0.2501264

Figure B.2 New kinematic diagram of Skippy in its open-loop zero position (i.e. $q_1, ...q_{10} = 0$). Diagram created by R. Featherstone and copied with permission. In this diagram (referred to as 'new model'), the follower and the lower part of the lever are shown at right angles to the horizontal part of the torso. In contrast, in the old diagram of Figure 7.12 (referred to as



'old model'), they are shown at right angles to the tilted portion of the torso. This discrepancy affects the definitions of q_6, q_7 and q_8, but not q_{67}. There are also differences in the positions of some of the body coordinate frames: in the new model, F4 is at P2, F5 is at P1, and F6 is at P0. In this thesis, all the kinematic calculations are done with the old model.