ITALIAN INSTITUTE OF TECHNOLOGY, AND
UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Hopping, Landing, and Balancing with Springs

by

Juan David Gamba Camacho

Thesis submitted for the degree of *Doctor of Philosophy* (34° cycle)

March 2022

Roy Featherstone	Supervisor
Darwin Caldwell	Supervisor
Giogio Cannata	Head of the PhD program

Thesis Jury:

Dr. Patrick Wensing, *University of Notre Dame*	External examiner
Dr. Leonardo Lanari, *Università di Roma "La Sapienza"*	External examiner

Advanced Robotics Department (iit), and
Department of Informatics, Bioengineering, Robotics and Systems Engineering (UniGe)

I would like to dedicate this thesis to my wife, parents and family.

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.

Juan David Gamba Camacho
March 2022

Acknowledgements

I would like to thank my supervisor, Dr. Roy Featherstone, for his pieces of advice and guidance along this journey. I would also like to thank my wife and parents for their unconditional support over these years and the pandemic. I also thank the Skippy team for their help and friendship.

Abstract

This work investigates the interaction of a planar double pendulum robot and springs, where the lower body (the leg) has been modified to include a spring-loaded passive prismatic joint. The thesis explores the mechanical advantage of adding a spring to the robot in hopping, landing, and balancing activities by formulating the motion problem as a boundary value problem; and also provides a control strategy for such scenarios. It also analyses the robustness of the developed controller to uncertain spring parameters, and an observer solution is provided to estimate these parameters while the robot is performing a tracking task. Finally, it shows a study of how well IMUs perform in bouncing conditions, which is critical for the proper operation of a hopping robot or a running-legged one.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

BIBO	Bounded-Input, Bounded-Output
BVP	Boundary Value Problem
CoM	Center of Mass
DAE	differential-algeabric equation
DMS	Direct Multiple Shooting
DOC	Direct Orthogonal Collocation
LV	Linear Variant
NLP	Non-linear Programming
ODE	Ordinary-Differential Equations
ODE	ordinaty differential equation
TRT	Time-Reversed Technique

Chapter 1

Introduction

For acrobats and athletes, the success or failure of a maneuver depends on precise and accurate movements. Different studies have been carried out on gymnastic and acrobatic motions performed by humans. In sports like *Artistic Gymnastics*, Aleksic-Veljkovic et al. (2014) analyzed the balance performance (based on the athlete's age) of young gymnasts competing at the international level on the balance beam. Hars et al. (2005) also studied the balance performance of human athletes at a basic back walkover by analyzing the dynamics of reaction forces acting during different supporting phases. Kim et al. (2012) performed a kinetic analysis of a standing back tuck on the balance beam (figure 1.1) with eight female

Figure 1.1 Tang Xijing of China competes during the artistic gymnastics women's balance beam final at the Tokyo 2020 Olympic Games in Tokyo, Japan, Aug. 3, 2021. (Xinhua, 2021)

gymnasts. The study used six eagle cameras and one force plate to capture the body's motion, velocity, acceleration, and ground reaction forces at jumping (around twice the athlete's weight). Motions of this kind are nearly planar activities that require precise balance control at one-support movements, the ability to produce and correctly modify high ground reaction forces, and the necessary skill to execute the actions accurately. So it makes sense to reduce the study's complexity by using a planar model of the motion.

1.1 Motivation

This thesis is part of a series on the design and construction of an experimental high-performance monopedal robot, called Skippy (Featherstone, 2021) (figure 1.2), which uses springs to help it achieve higher maximum speed at launch; shock reduction on landing; and recycling mechanical energy from one hop to another.

The main idea of the project is to demonstrate that *"it is easier to increase the complexity of a high-performance robot than to increase the performance of a highly complex robot"* (Featherstone, 2021). In this sense, a technology-inspired approach (instead of a bio-inspired one) is the basis of the robot's design to reach the maximum possible physical performance with today's robotics technology.

In figure 1.2, it is possible to see that the robot is merely planar except for the crossbar, which allows it to have three degrees of freedom. The mechanism has two revolute joints at the ankle and the crossbar and a 4-bar linkage at the hip to rotate the leg relative to the torso. The robot has:

- Two actuators at the crossbar and hip.
- Two regressive springs at the main motor and the ankle. Physically both are leaf springs made of fiberglass curved into a circular arc.
- A ring crew mechanism (Featherstone, 2022) between the main motor and the 4-bar linkage.

In this context, this thesis merges some challenges related to the Skippy robot, like:

- The trajectory optimization and design of a controller for executing complex motions like hops and summersaults.
- The study of advantages and disadvantages of using linear and nonlinear springs.
- The balancing problem on spring-loaded mechanisms.

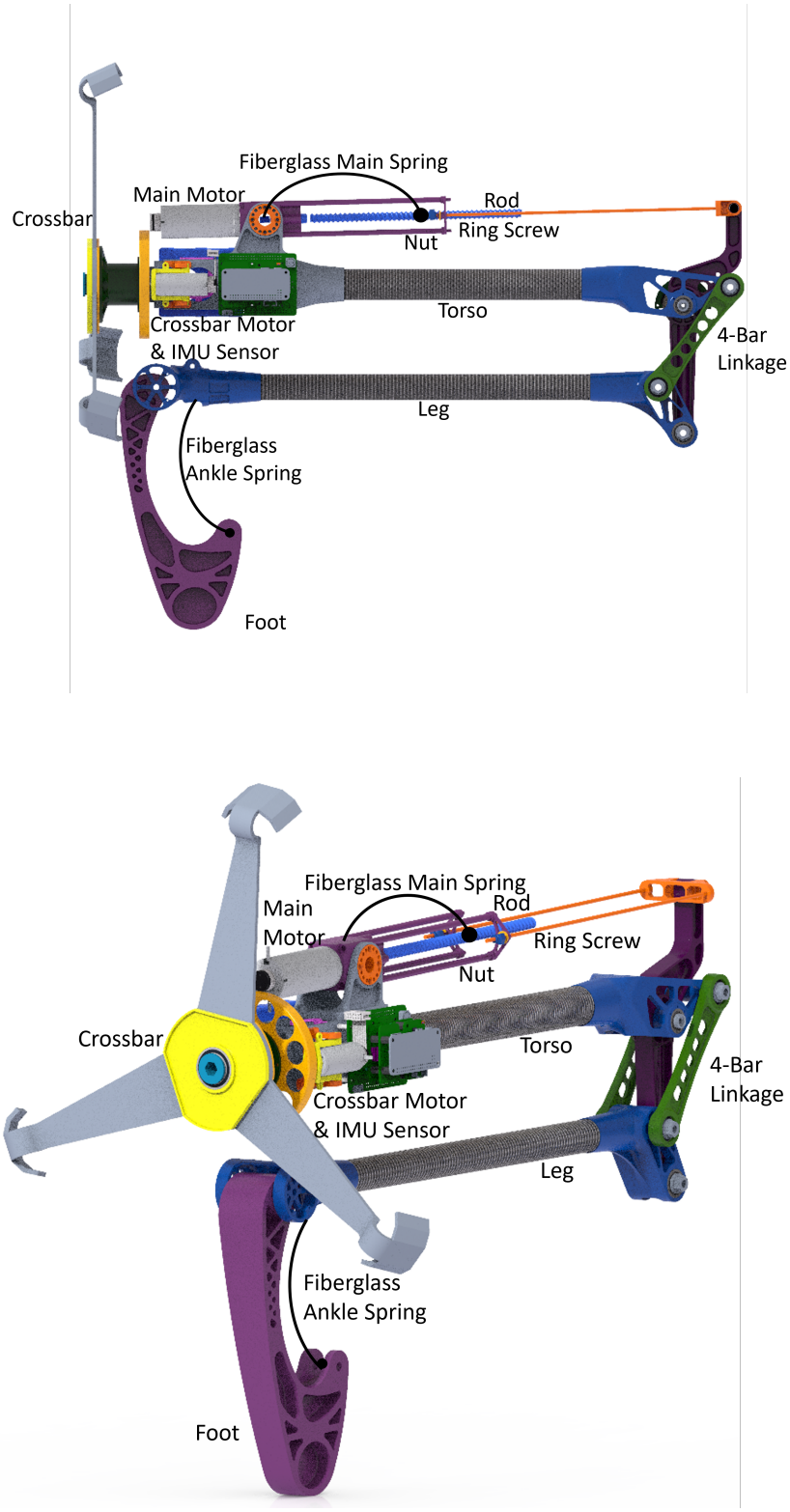

Figure 1.2 Skippy Robot.

- Online estimators to deal with uncertain spring parameters.
- Performance test of IMU sensors subject to continuous bouncing tasks.

A new balance controller has already been developed for Skippy, which can cope with the high-speed motions that Skippy is intended to exhibit (Featherstone, 2017, 2018). The controller has already been tested on different legged robots to balance (Driessen et al., 2019), hop (Yim et al., 2020), and even walk along a line (ninja walk) (González et al., 2020). However, the interaction between the balance controller and the springs has not yet been investigated, and that is one of the topics of this thesis.

Spring-loaded mechanisms like the spring-loaded inverted pendulum (SLIP) have also been widely applied in the biomechanical analysis of lower limb prostheses since it reproduces the kinetics of ground force loading during running (Kajita and Ott, 2016). In addition, the study on modeling and control of legged robots allow us to understand how such motions can be modelled, their stability analysis, and how dynamic motions are generated and commanded to avoid loss of balance and falls (Wieber et al., 2016) while performing a task.

Consequently, the presence of a spring in the leg improves the machine's performance for walking or running activities by providing the ability to store and release elastic energy. This allows energy to be recycled efficiently from one step to the next and increases the instantaneous peak power available. The disadvantage is that it makes the control problem more complicated. In locomotion applications, the Spring-Loaded Inverted Pendulum (SLIP) model has proven itself to be very useful (Ankaralı et al., 2010). Many studies using the SLIP model have addressed the problem of controlling the transition from flight to stance and from stance back to flight for performing walking or running motions (Hereid et al., 2014; Wensing and Orin, 2013); but this is different from the task of stopping, where there is a need to remove kinetic energy quickly. An approach consisting of an online controller that can drive the robot to the desired trajectory and an offline or semi-online strategy to find the desired trajectory (to be followed) for achieving certain motions seems to be a suitable strategy to exploit the execution of complex movements.

Regarding the system hardware, obtaining accurate models to describe the robot structure is usually accomplished by using offline calibration techniques involving measurement tools or very accurate design-build processes. Although, situations where the robot structure suffers breakage or malfunctions leading the system to instability while execution cannot be covered by pre-calibrating the entire system and require an online estimation scheme, which is another topic of this work.

The feasibility of these motions on real robots relies on the behavior of the existing hardware when executing this kind of task (Allione et al., 2021). In contrast to rovers and drones, a monopod robot needs to produce leaps and hops for moving along the ground. Consequently, the study of how well the robot's sensors perform in such bouncing conditions is critical for the proper operation of the robot. In this study, we also developed a test rig to explore the behavior of three IMU sensors subject to continuous bouncing situations.

1.2 Review of The State Of The Art

Robots walking, running, balancing, and performing complex motions is becoming more common nowadays. Technological improvements like higher computational power, powerful algorithms, and more sophisticated mechanisms are significantly transforming the classic idea of "slow robots" into intelligent and agile machines. There have been more than 30 years since this trend started taking its first steps with the hopping robot introduced by Raibert which consist of a telescopic springy leg connected to a torso by a revolute joint (Raibert, 1986).

1.2.1 Balancing, Tracking and Hopping

Hopping involves fast motions, so a balance controller must be able to make fast motions while maintaining, recovering, or deliberately losing its balance (Nassiraei et al., 2006; Tokur et al., 2020). These motions (hops) are essentially planar and can be explored using planar systems.

The problem of robotic balancing on a point is known to be a solved problem, but most existing solutions, such as Berkemeier and Fearing (1999); Grizzle et al. (2005); Spong (1995), are not fast enough for our application. More recent results, such as Gajamohan et al. (2012); Murata Manufacturing Co., Ltd. (2020), show better performance, but require the robot to have special features, such as reaction wheels. Cho and Jung (2003) presents a neural network control of an inverted pendulum (cart pole). Decentralized neural networks are used to balance and track the cart position. Unluturk et al. (2013) presents a two-wheeled autonomous balance robot with a simple PID control strategy to only balance the robot. Zhang et al. (2020) proposes a PD control strategy to balance a two-wheel hopping robot in move-forward and hopping modes. Wai and Chang (2006) exhibits an adaptive sliding-mode controller to control a dual-axis inverted-pendulum mechanism that is driven by synchronous motors. The authors start by adopting the energy conservation principle to

build a mathematical model of the motor-mechanism-coupled system. Next, an adaptive sliding-mode control system is developed for stabilizing and performing tracking control of the system. Chanchareon et al. (2006) introduces a trajectory-following controller for a balanced rod inverted pendulum. The authors have used a "computed feedback linearization" technique to stabilize the system at any desired position. The nonlinear system is controlled using a linear approximation by keeping the closed-loop system's roots fixed by nonlinear state feedback. This work implements a "balance strategy" based on Featherstone (2017), to achieve high-performance balancing in spring-loaded robots.

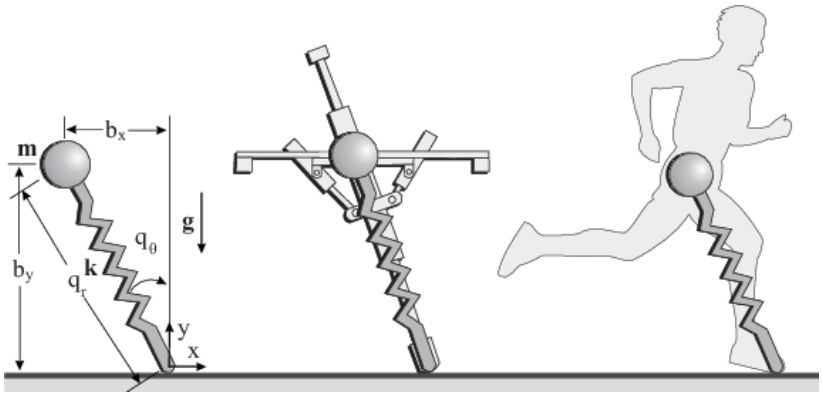

Figure 1.3 (a) The SLIP model, (b) Raibert's hopper, (c) A human runner. (Arslan et al., 2009).

The presence of a spring in the leg improves the machine's performance for walking or running activities by providing the ability to store and release elastic energy. This allows energy to be recycled efficiently from one step to the next, and also increases the instantaneous peak power available. The disadvantage is that it makes the control problem more complicated.

Raibert's original hopping machines (Raibert, 1986) had an actuated prismatic leg that was springy because of the use of a pneumatic actuator. Similarly, Terry et al. (2016) describes modeling and control techniques for a Raibert-like robot called "FRANK" based on high-order partial feedback linearization. Batts et al. (2017) presents another single-leg hopping robot that uses an actuated prismatic joint called a linear elastic actuator in parallel (LEAP) and modified version of Raibert's hopping strategy. The LEAP mechanism includes a voice coil actuator parallel with two compression springs and enables the robot to perform 19 simultaneous hops (7 seconds). These machines could only hop, so balancing (without hopping) was not an issue.

In this context, other studies have implement control strategies based on virtual springs to gain robustness and performance. Jin et al. (2017) models each leg of a quadruped robot

as two sets of virtual spring and damper systems to add robustness to the locomotion control when the legs are subject to collisions with the ground during walking. Martin et al. (2015) implements a spring-mass model for a running robot called ATRIAS. The approach led to very robust running over the unobserved ground in a high fidelity simulation.

In locomotion applications, the Spring-Loaded Inverted Pendulum (SLIP) model (figure 1.3) has proven itself to be very useful (Ankaralı et al., 2010). McDonald-Maier et al. (2000) proposes different neuro controllers to perform fast spring-legged locomotion using the classic SLIP model to run at untrained speeds over flat surfaces and uneven terrains. Piovan and Byl (2013); Xue et al. (2017) comments that the good performance of humans and animals at running and hopping compared to robots, is due to the leg's capacity to regulate energy production and removal by varying the muscle stiffnesses. Therefore the classic SLIP system with fixed stiffness is inappropriate to reproduce these motions. The authors proposed a variable stiffness SLIP system to mimic this good performance, and show via simulations the high tracking accuracy and rapid convergences to the desired motion. Many studies using the SLIP model have addressed the problem of controlling the transition from flight to stance and from stance back to flight for performing walking or running motions (Hereid et al., 2014; Piovan and Byl, 2012; Riese and Seyfarth, 2012; Wensing and Orin, 2013); but this differs from the task of stopping, where there is a need to remove kinetic energy quickly.

Applications involving physical springs in locomotion had been also explored. Ma et al. (2017) introduces a control strategy to experimentally implement a sustainable running of a spring-loaded biped robot called DURUS-2D. The authors first used a large-scale optimization to generate an energy-efficient running gait, subject to hybrid zero dynamics conditions and feasibility constraints which incorporate practical limitations of the robot model based on physical conditions. Then, a state stability-based control law is used to implement the optimized gait.

On the other hand, there has been also developed robots to perform motions while flying, Stickman is a simple two-degree of freedom robot that produces somersaulting stunts from an initial non-zero angular momentum launching. The robot uses various sensors like an IMU and a laser rangefinder to estimate and control its mid-flight kinematic configuration, this to perform certain in-flight motions and reach a desired landing position (Pope et al., 2018).

A more recent robot, Salto (Haldane et al., 2016), resembles a miniature, electrically actuated Raibert-style hopping machine having an explicit series elastic element to provide the springyness in the leg. Salto has demonstrated its ability to balance as well as hop (Yim et al., 2020) using the balance controller in Driessen et al. (2019); Featherstone (2017).

However, Salto's upper link is a reaction wheel, which means that the action of balancing has negligible effect on the spring, and vice versa.

The problem of hopping with an asymmetric upper link was studied from a control perspective by Poulakakis and Grizzle (2007); while Batts et al. (2017) presents a 3D hopping machine with a parallel spring, which is more controllable than a series spring. Their robot was able to perform continuous hops for about seven seconds before losing its balance. Xiong and Ames introduced a hopping and landing solution for a bipedal robot named Cassie (Xiong and Ames, 2018). They described the mechanical compliance of the robot leg by a virtual SLIP mechanism with nonlinear stiffness (the nonlinearity being a function of the kinematics of the leg). Then the prismatic motions of the virtual model are optimized to produce the desired hop.

The study of combined balancing and hopping on a general planar double pendulum was pioneered by Berkemeier (Berkemeier and Fearing, 1998), and this was the inspiration for Azad's later work (Azad, 2014; Azad and Featherstone, 2013). These works assume a rigid leg. However, in an experiment that was never published, Azad repeated the work on single hops (Azad and Featherstone, 2013), but with the rigid leg replaced by a springy one. Although the work was never published, an animation of his result can be viewed at Featherstone (2021). It can be seen that Azad's controller works very well with a springy leg.

The study in this thesis goes beyond Azad's work by using a faster, more responsive balance controller, and by investigating the spring's effect on the controller's ability to execute large and fast movements after the landing transient has died away. The main objective of having such a high-performance controller is to ensure the proper execution of motions that drive a spring-loaded monopod robot to take-off stance, and landing phases.

1.2.2 Parameter Identification and High Order Nonlinear Observers

The parameter identification theory deals with the problem of efficiently extracting data about the system dynamics from its measurements. Most of these strategies involve mainly optimal parameter estimation (Jovic et al., 2016; Manns et al., 2017; Rizzello et al., 2016), least-squares methods (Briot and Gautier, 2013; McFarland and Whitcomb, 2012; Riva et al., 2017), Bayes methods (Green et al., 2015; Kreucher and Lakshmanan, 1999; Zhang et al., 2015), Kalman filter extensions (Bolzon et al., 2002; Corigliano and Mariani, 2004; Iwasaki and Kataoka, 1989), among others.

Parameter identification techniques for underactuated nonlinear systems have been applied to systems like the mobile wheeled pendulum using Kalman filters (Roffel and

Narasimhan, 2014), the Furuta pendulum (Garćıa-Alarćon et al., 2012) using the least squared algorithm with Volterra polynomials (Ronquillo-Lomeli et al., 2016), and the single and double pendulum based on harmonic balance-based algorithms (Liang and Feeny, 2006, 2008).

On the other hand, high-order sliding mode finite-time algorithms are used widely due to their attractive features over other kinds of observers (Benallegue et al., 2006):

- Insensitivity (more than robustness!) to unknown inputs;
- Possibilities to use the values of the equivalent output injection for the unknown inputs or parameter identification;
- Finite-time convergence to exact values of the state vectors.

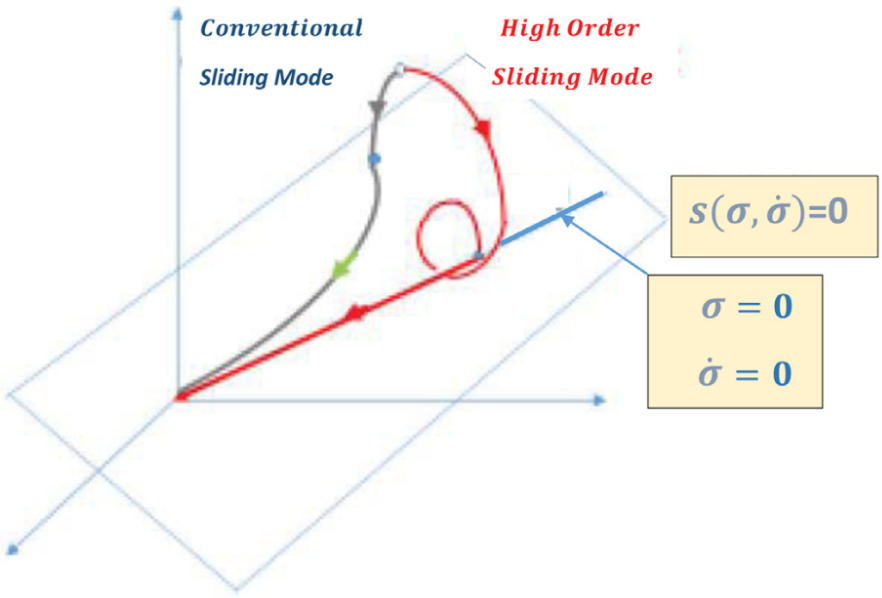

Figure 1.4 Conventional and High Order Sliding Mode (HOSM) state space (Utkin et al., 2020).

Figure 1.4 shows the trajectories driven by a conventional sliding mode (gray solid line with a green arrow) and by a high-order sliding mode (red solid line) controllers with the vector variable $\sigma, \dot{\sigma}$. The high-order sliding mode controller reaches the manifold $\sigma = \dot{\sigma} = 0$ faster than the conventional one.

HOSM observers have demonstrated their effectiveness by identifying mechanical parameters in combination with a recursive LS algorithm where the design of the non-linear injection terms is based on the generalized super-twisting algorithm (STA) (Moreno and

Osorio, 2012), leading to finite-time convergence (Davila et al., 2006; M'sirdi et al., 2006). In Adetola and Guay (2008) the authors use STA and a non-recursive LS algorithm to identify constant parameters in nonlinear systems (Boubaker and Iriarte, 2017). Benallegue et al. (2006) applies a feedback linearization controller using a high-order sliding mode observer for a quadrotor subject to parameter uncertainties and external disturbances. Through simulations, they demonstrated that the observer easily overcomes the system's nonlinearities by estimating the external disturbances, which are used as desired stability and robustness properties in the global closed-loop system. Na et al. (2013) presents an adaptive control framework for nonaffine pure-feedback nonlinear systems. Instead of using the classic backstepping approach, they employ a high-order sliding mode observer with finite-time error convergence to estimate unknown states. Moreover, the observer worked with two adaptive controllers to achieve tracking control and guarantee closed-loop system stability. Efimov et al. (2012) improves the estimation accuracy of an interval observer applied to linear-parameter varying systems by employing a high-order sliding mode method to an observable subsystem (the main system is supposed to have non-detectable or non-observable parts) to estimate the states and their derivatives. Ríos et al. (2012) presents different observers based on high-order sliding mode techniques to reconstruct the continuous and discrete states of a switched system in the presence of unknown inputs. Pilloni et al. (2015) uses a high-order sliding mode observer with an injected term of the second and third orders of the sliding mode controller to regulate the oxygen excess ratio of a proton exchange membrane fuel cell to a suitable setpoint.

In general, the finite-time convergence is based on the adaptive control theory, requiring to solve matrix-valued ordinary differential equations and check the invertibility property of a matrix online (Polyakov and Fridman, 2014). This scheme allows the reconstruction of the unknown parameters in finite-time (FT) provided that a given persistence of excitation (PE) condition holds (Polyakov and Fridman, 2014). A well-known approach used to ensure the PE condition in adaptive controllers is to add a bounded perturbation signal to the set-point or trajectory, or even use it as the reference input, which in contrast may degrade the specified regulation or tracking performance (Adetola and Guay, 2008).

The balance controller introduced in this work (Gamba and Featherstone, 2021) has already demonstrated its capacity to control a spring-loaded monopod robot through fast and large motions. Nevertheless, the controller relies on the continuous knowledge of the force exerted by the spring located at the leg of the robot, and uncertainties on this estimate can lead to poor performance and instability. In this regard, we demonstrate the feasibility

of executing a high-performance task under uncertain or unknown spring parameters by combining a HOSM observer with an injection term (Gamba et al., 2021).

1.2.3 Trajectory Optimization

The dynamic behavior required for achieving actions applied to body motions (Fang and Pollard, 2003; Lee et al., 2002) has been studied to animate characters in video games (Menache, 1999). The necessary trajectories that make up the motion can be obtained using kinematic and biological models (that incorporate muscle models and other natural considerations)(Anand et al., 2019; Jiang et al., 2019; Leiva et al., 2017), control models based on energy minimization (Quinn and Zhai, 2018; Todorov and Jordan, 1998), and data-driven models leveraging neuronal networks (Alsharif et al., 2015; Frintrop et al., 2010; Itti et al., 2003). Majkowska and Faloutsos (2007) presents a method for creating complex, multi-flip ballistic motions from simple, single-flip jumps. The authors commented that the developed method uses a computational optimization algorithm to produce physically valid results. Wooten and Hodgins (2000) describes a technique for generating motion transitions for jumps, leaps, and somersaults by parametrizing individual basis behaviors. In this sense, they designed a control system to connect the parametrized individual basis.

This behavior problem also exists when controlling robots at complex tasks (Schultz and Mombaur, 2010), specifically at trajectory optimization (Toussaint, 2009a). In this regard, the optimizer can control the system directly (Dierks and Jagannathan, 2011, 2012; Vamvoudakis and Lewis, 2010) or obtain a command signal to be executed by a low-level controller (Ibrahim et al., 2020; Lampariello et al., 2018; Quintero et al., 2013). Lin et al. (2021) compares offline the performance of deep-reinforcement learning with model predictive control in an adaptive cruise control design for car-following scenarios. The results demonstrated that both solutions performed similarly when the prediction horizon of the model predictive controller was sufficiently long to find a global solution. Kloeser et al. (2020) presents a real-time control of 1:43 scale autonomous race cars using nonlinear predictive control. The online controller allows the car to avoid obstacles, perform stop-and-go maneuvers and run at low and high speeds.

Optimization strategies had also demonstrated its applicability in legged robots applications. Di Carlo et al. (2018) presents an implementation of a model predictive controller to determine ground reaction forces for a torque-controlled quadruped robot. They performed a convex optimization by simplifying the robot's dynamics, finding a solution in one millisecond. Although the optimal solution comes from a simplified model, the robot demonstrated

robust locomotion at different gaits and three-legged gaits at a speed of three meters per second. Nguyen et al. (2019) presents a methodology for implementing an optimized jumping behavior on quadruped robots. The strategy starts by obtaining an optimized trajectory. Then, the motion is followed by using a high-frequency tracking controller and a robust landing controller to complete the jump. Tian et al. (2021) presents an optimization framework for upward jumping motion using quadratic programming. The scheme covers the offline trajectory optimization and online tracking controller to produce the act. The Simulation results show a three-link robot of 43.5 kg performing a jump of 16.4 cm height. Fan et al. (2020) addresses the motion planning problem of a free three-link robot during the in-air motion by using a homotopy method that deforms an arbitrary path connecting the initial and final states. The path deformation is performed using geometric heat flow, and it's defined by a Riemannian metric encoding the system dynamics and state constraints. Xiong and Ames (2020) presents a sequential motion planning method for generating somersaults on bipedal robots equipped with a flywheel mechanism. The authors use the bipedal robot "Cassie" with a flywheel mounted at the top of its pelvis to increase the inertia at its upper body and increase the capability to control the momentum.

In this sense, different works have introduced software solutions for obtaining optimal trajectories. Chen et al. (2020) introduces an optimal control scheme for a hybrid SLIP model to be applied in rapid trajectory optimization like walking or running. Hereid and Ames (2017) presents a software package called "FROST" to model dynamic behaviors using the hybrid zero dynamics framework (Ma et al., 2016). The software has been used in different robots to produce walking and running behaviors (Hereid et al., 2016b; Reher et al., 2016). Koenemann et al. (2019) presents a software framework for modeling and solving offline optimal control problems. Fevre et al. (2020) show's a CasADI (Andersson et al., 2018) based optimization package to efficiently solve gait optimization problems. Verschueren et al. (2018) presents "acados", a software package for model predictive control focusing on computational efficiency by using algorithms written in C.

Most of the commented optimization strategies have demonstrated their success in finding motion profiles. Nevertheless, finding motions (hops, leaps, somersaults, etc.) involving motion trajectories to be tracked by a controller in a real robot requires an optimization framework with the following features:

- Allow event-based optimization. Time-based optimization approaches are not the best choice when finding a motion conditioned by discrete actions (contact and non-contact with the ground), similar to hops. The event-based feature allows us to determine the

end of the maneuver to be determined by an event rather than a specific time duration. In this kind of application, the task's duration is unknown as the motion profile.

- Accurate solutions. As previously commented, many optimization frameworks successfully generate motions for gaming applications that don't need to be very realistic. In robotics, other approaches use an ordinary-differential equation solver with the optimizer to ensure that the solution complies with the system dynamics. Solutions of this kind demand a lot of execution time. In this sense, orthogonal direct collocation methods minimize the execution time by making the problem's formulation more sparse, which is exploited by a solver.

- Computationally cheap solution. Many approaches require lots of computational power and or execution time for solving problems of this kind. The ideal approach should exploit the system's mathematical model to find the solution rather than treating the optimization problem as a black box.

- Scalability. As commented, by using the mathematical model of the system, it is possible to obtain a cheaper solution in terms of computational resources. Now, finding this full mathematical description can be difficult and time-consuming for complex mechanisms. In this sense, the appropriate scheme should be able to deduce this full mathematical model by providing a simpler representation of the dynamics and kinematics of the robot.

- Solve Multi-phase optimizations. Solvers like IPOPT (Wächter and Biegler, 2006) require twice continuously differentiable functions for describing the problem's objective and constraint functions. In this regard, hybrid trajectory problems relate to applications with discontinuities like transitions from flight to stance and are generally implemented in a single optimization problem with multiple phases.

In this work, we use an optimization framework that meets the commented requirements for finding motion trajectories.

1.2.4 Inertia Measurement Unit (IMU) sensors and Legged Systems

As commented in section 1.1, this work is related to the development and control of the Skippy robot. In this regard, testing the capability of different IMU sensors available on today's market is necessary to explore their performance in hopping activities.

IMUs are electronic devices containing accelerometers, gyroscopes, and sometimes magnetometers. The accelerometer measures the apparent acceleration, which is the sum of the actual acceleration and the effect of the gravitational field. In theory, it is impossible to separate these two effects. However, It becomes possible by making certain assumptions about the motion. If these assumptions fail, the IMU's estimate of which way is 'up' becomes inaccurate. In this regard, the accelerometer's measurements can be used to compute attitude, the gyroscope measurements (angular velocity) can be used to calculate the change of orientation using numerical strap-down integration (Nazarahari and Rouhani, 2021). Although the devices that only use accelerometers and gyroscopes are unaffected by external magnetic fields around the apparatus, this sensor's combination may not be sufficient to increase the measurements accuracy due to the sensor's noise and gyro's well-known drift issue (Ahmad et al., 2013). In a magnetically neutral setup, the magnetometer measures the geomagnetic field, which is used to estimate the yaw angle.

In this sense, Attitude and Heading Reference Systems (AHRS) uses the IMU measurements and combines them with different sensor fusion algorithms that have been proposed in the literature (mainly based on the Kalman filter) to obtain accurate and reliable estimations (Caron et al., 2006; Li and Wang, 2013; Mirzaei and Roumeliotis, 2008). Their history began in the 1930s, constrained to large-scale applications due to their size, cost, and power consumption. IMU's used in robotics are commonly based on micro-electromechanical system technology, and unfortunately, low-cost units are usually affected by non-accurate scaling and zero biases (Tedaldi et al., 2014). In the literature, the Attitude and Heading Reference Systems are generally known as IMUs, and we'll also refer to them as IMUs.

IMUs are present in several applications in quadcopters, unmanned vehicle navigation, robotics, human motion tracking tasks, and medical prostheses; by their capability of estimating a body's specific orientation; and angular rate. As previously commented, the IMU's internal processor must make assumptions about the nature of the motion to estimate the gravitational field, and therefore which way is 'up'. These assumptions are not always available for the public, and therefore, the performance of the IMU subject to hopping/running motion is unknown. In contrast to rovers and drones, legged robots are subject to a series of support phases and flight phases in order to jog, trot, hop or run; specifically, a monopod robot experiences more intensive accelerations at these contact and non-contact phases (bouncing) by producing leaps and hops for moving along the ground. Consequently, the study of how well these sensors perform in such bouncing conditions is critical for the proper operation of robots of this kind. Nevertheless, the bouncing behavior is not limited to only robotics applications, Zhao et al. (2016) presents a systematic methodology combining multidomain hybrid

systems and optimization-based controllers to achieve human-like multi-contact prosthetic walking on prostheses. The authors used the IMU system to capture the human locomotion response to be reproduced by the prosthetic device. Schmutz et al. (2020) shows a model to estimate the horse speed per stride using only one IMU mounted in the saddle pommel (the rounded knob on a horse's saddle that a rider grips with one hand) close to the horse's withers (the ridge between the shoulder blades of an animal) without using a sensor at the limb to reset the error at each cycle. The author claimed that the proposed solution overcomes the setup constraint imposed by GPSs or 3D optical motion capture (badly influenced by the presence of obstacles and cannot be used indoors, although the optical motion capture can be used indoors. Benson et al. (2019) explores the possibility of establishing a stable gait pattern for twelve different human runners by using a single IMU attached to the runner's wrist. The authors emphasize that these running patterns may provide real-world indications of alterations in running biomechanics due to factors such as fatigue, performance, and injury status.

In robotics, IMU acceleration estimates are frequently corrupted by noise and drift. Such drifts can often be corrected by performing a zero-velocity update, which is unfeasible in some applications (Lew et al., 2019). Li et al. (2014) presents a three-dimensional model of a quadruped robot with six degrees of freedom at the torso and five degrees of freedom at each leg executing a 3D trotting gait. The IMU (mounted on the robot's torso) experienced a severe drift at the yaw signal during the experiments. The authors believe that this drift was caused by the magnetic field excited by the motor in the treadmill. 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. Kuindersma et al. (2015) describes different optimization strategies and a state estimator to execute walking over non-flat terrain with the Atlas humanoid. The authors used an IMU mounted on the pelvis to obtain the pelvis's pose and twist. To reduce the drift of the robot measurements, they used an inertial and kinematic estimator but, it was unsuitable for accurate walking over tens of meters. They finally added a LIDAR to achieve the task.

1.3 Goals and Objectives

This work's main objective is to explore the feasibility of hopping, landing, and balancing with a springy leg monopod in the presence of uncertainties at the spring parameters. The key idea is to develop a balance controller capable of accurately controlling the robot in executing fast trajectory tracking; and an optimization tool for trajectory optimization applications. The

scientific objectives regarding the theoretical aspects in short- and long-term scenarios, are the following:

- Develop a balance controller able to control the motion of spring-loaded mopods while balancing.
- Develop a strategy to deal with uncertainties at the spring parameters applied to motion tracking without compromising the balancing performance.
- Develop an approach to explore the feasibility of launching into and landing from high hop motions with such robot.
- Investigate the IMUs orientation estimation when subject to continuous bouncing applications.

1.3.1 Methodology

The used methodology consisted in identifying possible ways to achieve the commented objectives and reviewing the state-of-the-art to find the most suitable strategies to be used. Then, the most relevant approaches were chosen to be improved and provide a better solution.

The balance strategy that demonstrated the best performance consisted in neglecting the spring's force at the controller's plant. This force is only considered when calculating the actuation torque to be sent to the robot. This setup showed a certain kind of robustness to uncertain spring parameters (commented in subsection 4.4.3). Subsequently, the nonlinear parameter identification approach demonstrated to be a more suitable solution for this specific scenario than adding a robustness term to the control law by guaranteeing the persistent excitation condition by performing balance and trajectory tracking tasks.

The trajectory optimization strategy started with a semi-optimized version of the reverse-time technique employed by Azad and Featherstone (2013). This was followed by a multiple shooting approach, which demonstrated some significant improvements but was computationally expensive. It was necessary to develop a software package to compute an symbolic description of the robot's dynamics to be used by the optimizer (IPOPT). The symbolic representation was then used to perform an event-based optimization, rather than a time-based one, such as performing a leap, in which, the take-off center of mass (CoM) velocity is known, but the instant of take-off and the necessary motion to achieve it are unknown. The final version of the trajectory optimization algorithm supports these features by employing orthogonal collocation approaches to find solutions more quickly.

The general workplan suffered various modifications given that it was not possible to have the Skippy robot in time to achieve the objectives imposed at the beginning of this study. Then, the scope moved to explore the scenario of using a passive springy leg to balance and execute hops, which demonstrated some challenges that were not anticipated.

1.4 Contribution

In this work, a high-performance balance controller is developed to control a spring-loaded monopod robot while balancing and tracking a fast motion. Moreover, the controller can follow a trajectory referenced to:

- The actuated joint's position.
- The absolute torso's angle with respect to the ground (x-axis).
- The velocity of the robot's CoM to produce a hop or a leap.

This new controller proves that the balance controller in Featherstone (2017, 2018) still works when the leg is springy, although the performance is not as good as with a rigid leg, but the position and velocity variables of the passive joint have to be taken into account when calculating the state variables needed by the balance controller. There is an important distinction between the work presented here and that presented in Featherstone (2017, 2018) on the topic of balancing in the presence of other motions. In the earlier work, the other motions were assumed to be known in advance, and executed accurately by a motion controller, so that the balance controller could be told in advance what the movements would be, and hence make the robot lean in anticipation of the balance disturbances that these motions were expected to cause. In contrast, a passive springy joint is considered here, which moves in response to the actions of the balance controller, and therefore causes unanticipated disturbances. The balance controller presented here is also able to to bring improved balancing, in-flight motion and landing capabilities to the Skippy Project, in order to cope with highly athletic tasks like performing four-meter hops, robust 3D balancing and recovering from an unexpected breakage at landing (Gamba and Featherstone, 2021).

Another contribution of this thesis concerns the design of a scheme to estimate the spring parameters. Such a problem can arise when the robot is landing, and sufferers spring failure due to the action of an excessive force when the foot touches the ground. In this scenario, the spring parameters are uncertain, and the strategy needs to achieve the parameter identification without compromising the controller's performance in terms of balancing and

tracking a motion. Moreover, the robustness and stability of the balance controller described in Featherstone (2017) are evaluated by performing a fast sequence of motions where the legged robot is capable of tracking a trajectory while balancing around a support point.

This thesis also extends and generalizes the application of the work presented in Azad and Featherstone (2013) for producing hops and balance with a monopod robot. This study uses a modified version of the dynamic software package developed in Featherstone (2008) to obtain an symbolic representation of the equations of motion for any robot to be used for nonlinear trajectory optimization applications with casADI software (Andersson et al., 2018). The strategy formulates the problem as a nonlinear programming (NLP) problem, where the motion profile is obtained by minimizing a given objective and respecting the imposed linear and nonlinear constraints. Next, the whole motion is reproduced on a dynamic simulation by using the commented balance controller (Featherstone, 2017; Gamba and Featherstone, 2021). Performing the complete act by using a launch controller allows us to incorporate extra estimation and robust algorithms to ensure the motion success, like the spring estimation strategy previously presented (Gamba et al., 2021).

Finally, we also explore the orientation response of IMUs subject to continuous hopping activities. These kinds of motions are mainly present in legged robots at running, walking, and trotting gaits.

1.5 Outline

This work is organized according to the following chapters:

- **Chapter 2:** Introduces the optimization framework developed used for finding motion profiles. The development of the framework is compared with multiple shooting and collocation methods. The chapter also presents a launch controller applied to a two-degrees of freedom robot and compares the results with those obtained in Azad and Featherstone (2013).

- **Chapter 3:** Presents a trajectory optimization study based on the optimization framework developed in chapter two to explore the landing capabilities of a spring-loaded monopod robot with an actuator consisting of a DC motor and frictionless gear. The study presents the motion response of using a linear and nonlinear spring at the robot's leg.

- **Chapter 4:** Exhibits the design of the balance control theory commented in Featherstone (2017) applied to spring-loaded robots. The chapter expands the results presented

in Gamba and Featherstone (2021) and introduces the necessary modifications to control the robot's torso absolute orientation rather than controlling the actuated joint position. Finally, the chapter presents a sensitivity analysis of the launch controller when a spring-loaded robot is subject to uncertain spring parameters.

- **Chapter 5:** Introduces an estimation scheme based on a high-order sliding mode observer to estimate the spring parameters after a partial failure. The chapter presents a mathematical stability proof, and its performance is validated through an absolute motion-tracking task.

- **Chapter 6:** Presents the apparatus built to perform the IMU's bouncing test. The chapter also presents the results obtained from the commented three IMU's subject to continuous bouncing with a maximum acceleration of 4 Gs for around three minutes. The results demonstrated that orientation obtained from the IMU's drifts during the experiment.

Chapter 2

Hopping

For acrobats and athletes, the success or failure of a maneuver depends on precise and accurate movements. In this sense, strategies for finding optimized ways of executing a task and approaches to implement them in real systems are still an open problem in robotics. Launching into a hop without considering the take-off velocities can cause head landings or crashes, as commented in Pope et al. (2018). Xiong and Ames (2020) present a sequential motion planning method for generating somersaults on bipedal robots; the authors modified the robot Cassie by adding a flywheel at the top of its pelvis to amplify its capability to control the momentum while flying. Additionally, Kollarčík (2021) used a sequential optimization approach for finding a launching motion for a vertical hop with a biped wheeled robot and concluded that by employing this strategy, it is necessary to manually adjust the motion duration T until finding the best instant, which is highly inefficient.

This chapter presents a study about the launching (take-off) motion before a leap. Azad and Featherstone (2013) accomplishes the launching problem by finding a motion profile with a Time-Reversal Technique (TRT) and designing a controller capable of repeating this motion to perform a launching motion in a real robot. They assume that the motion needed to bring the robot to a still balanced configuration from a landing (supposing a plastic collision between the ground and the robot's foot) can be reversed and used to produce a take-off. Here this strategy is improved by formulating the motion profile as a Boundary Value Problem (BVP) and optimizing it as a Non-linear programming problem (NLP). The balance controller is improved based on the balance theory presented in Featherstone (2017).

Section 2.1 introduces a detailed description of the robot model used for computing the launching motion with the new method and the strategy presented by Azad and Featherstone (2013).

Section 2.2 explains how the conservation of momentum law (while the robot is flying after taking off the ground) needs to be considered to obtain the launching configuration. Moreover, the problem is treated as a BVP and implemented as an NLP problem to exploit globalization strategies developed for NLP algorithms simplifying the incorporation of dynamic equality constraints, like

$$z(i) = z(i-1) + \int_{t_{i-1}}^{t_i} \dot{z} dt, \tag{2.1}$$

where z is the system's state at a time step i; and mechanical inequality constraints, as

$$z_m \leq x \leq z_M \tag{2.2}$$

where the system state is bounded by a region between z_m and z_M limits.

Section 2.3 provides a brief introduction of the balance problem properties and design of the controllers presented in Azad and Featherstone (2013) and Featherstone (2017). The latter gives the basis of the launch controller to produce the take-off.

Section 2.4 presents the design of the launching controller, and it's compared with Azad's controller in terms of accuracy and performance.

Our main objective is to reproduce the results obtained by Azad and Featherstone (2013) for producing a launching motion and compared them with the strategy proposed in this chapter in terms of accuracy and applicability.

2.1 Robot Model

The robot model consists of a simplified monopod two-link mechanism as shown in Fig. 2.1. Links 1 and 2 are the leg and torso, respectively. Joint 1 is a passive revolute joint that models the contact between the foot and ground, and joint 2 is the actuated joint. Table 2.1 shows the links' mass and length parameters used in Azad and Featherstone (2013); and the symbols m_i, l_i, and I_i appearing below denote the mass, length, and rotational inertia about the center of mass (CoM), respectively, of link i. The joint variables are q_1 and q_2. When all joints

Link (i)	Mass (kg)	Length (m)	CoM (m)	Inertia at CoM (kgm^2)
1	2	0.5	0	0
2	14	0.75	0.375	1.96875

Table 2.1 Length and inertia parameters of the robot shown in Fig.2.1

.

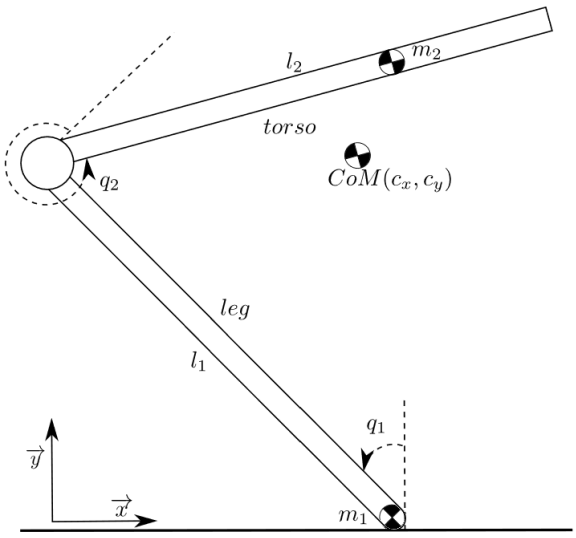

Figure 2.1 Robot model. q_2 is negative in this configuration, and has been drawn as $q_2 + 2\pi$.

are zero, the leg is vertical, and the torso is horizontal out to the right. Positive motion of a revolute joint i rotates link i counter-clockwise relative to link $i-1$; In Fig. 2.1, q_1 is positive and q_2 is negative.

The robot's dynamics is described by the equation:

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{12} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix}, \tag{2.3}$$

where H_{ij} are the elements of the joint-space inertia matrix, $\ddot{q}_1$ and $\ddot{q}_2$ are the joint acceleration variables, C_1 and C_2 elements containing gravity, Coriolis and centrifugal forces, and τ_2 is the torque at the actuated joint. The matrix is symmetric, which is why H_{12} appears twice. The elements of the joint-space inertia matrix and C vector can be obtained by

$$\begin{aligned}
H_{11} &= m_1 l_{c1}^2 + I_1 + m_2 \left(l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos(q_2)\right) + I_2, \\
H_{12} &= m_2 l_1 l_{c2} \cos(q_2) + m_2 l_{c2}^2 + I_2, \\
H_{22} &= m_2 l_{c2}^2 + I_2, \\
C_1 &= -m_2 l_1 l_{c2} \sin(q_2) \left(\dot{q}_2^2 + 2\dot{q}_1 \dot{q}_2\right) + G_1, \\
C_2 &= m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1^2 + G_2, \\
G_1 &= (m_1 l_{c1} g + m_2 g l_1) \sin(q_1) + m_2 g l_{c2} \sin(q_1 + q_2), \\
G_2 &= m_2 l_{c2} g \sin(q_1 + q_2).
\end{aligned}$$

where g is the gravitational force, l_{c1} and l_{c2} are the positions of the link's CoM to its respective link's frame, and G_1 and G_2 denote the effect of g in links one and two, respectively.

2.2 Launching Trajectory Search

Optimization strategies had been extensively used by Dinev et al. (2020); Fevre et al. (2020); Hereid et al. (2016a); Kloeser et al. (2020); Kuindersma et al. (2016); Pugh and Martinoli (2007); Tian et al. (2021); Toussaint (2009b); Zhang et al. (2013) for solving motion planning problems. In this sense, the motion search exercise is solved by employing the system dynamics commented in the previous section as BVP of a differential-algebraic equation (DAE).

Differential equations commonly model the evolution of the dynamics of physical systems over time. In cases where the system's states are also subject to limitations (kinematic and dynamic), the mathematical model also contains algebraic equations to express these restrictions. In this regard, DAE equations involve an unknown function and its derivative in the general form:

$$f(t,z,\dot{z}) = 0, \qquad t_0 \leq t \leq t_f, \tag{2.4}$$

where $z = z(t)$, the unknown function, and $f = f(t,z,\dot{z})$ have N components denoted by z_i and f_i for $i = 1,2,3,...,N$. For an initial value problem, the solution needs also to satisfy the condition $z(t_0) = z_0$, and for a boundary value problem, the solution needs to satisfy a set of constraints in the form of $b(z(t)) = 0$. The DAE's solution method will depend on the modeled structure.

In this section, we'll use a class of DAE named the semi-explicit DAE or ordinary system equation (ODE) with constraints. In this sense, the dynamic model explained in equation (2.3) takes the form:

$$\ddot{q} = H^{-1}(\tau - C) = f(q,\dot{q},\tau_2) \qquad 0 = b(t,q,\dot{q},\tau_2), \tag{2.5}$$

where the model is discretized over a time $t \in (t_{k-1}, t_k]$ in time periods $k = 1,...,N_T$. Where a certain control profile is found by evaluating the ODE system and satisfying the imposed boundaries (Kunkel et al., 2006).

Consequently, the problem is implemented as an NLP problem to facilitate the addition of the following constraints:

- To guarantee the system continuity over the optimization, we introduce the following nonlinear dynamic equality at every step

$$\dot{q}(i) = \dot{q}(i-1) + \int_{t_{i-1}}^{t_i} f(q, \dot{q}, \tau_2) dt, \qquad q(i) = q(i-1) + \int_{t_{i-1}}^{t_i} \dot{q} dt; \tag{2.6}$$

- To bound the operating space of the joints positions due to the robot's kinematics, we implement mechanical inequalities $q_{min} \leq q \leq q_{max}$. In presence of motors at the joint's axis, the joint's velocity can also be restricted to a maximum value $|\dot{q}| \leq \dot{q}_{max}$;

- Initial state equalities/inequalities are used to ensure that the optimal solution starts at a specific value or under a specified region;

- Like initial equalities/inequalities, terminal state equalities/inequalities are used to specify that the solution should finish in a specific value or under specific region.

ODE systems can generally be solved using ***direct*** and ***indirect*** techniques. ***Direct*** approaches are usually used given because they are simpler to set up and solve (von Stryk and Bulirsch, 1992) than ***indirect*** approaches, where it is necessary to construct the adjoint equations and their gradients to obtain a more accurate metric for the solution (Cervantes and Biegler, 2009). ***Direct*** methods split into *sequential* and *simultaneous* methods. *Sequential* or *direct single shooting* methods solve the ODE model in an inner-loop (simulation), and the discretized control input τ_k is updated out of the simulation loop using an NLP solver. The outer-loop control update can use an analytical (direct sensitivity equations, the integration of adjoint sensitivity equations) or numeric sensitivity analysis of the ODE model to calculate the objective function gradient concerning the control input (Biegler, 2010). These kinds of strategies are moderately easy to construct but require multiple integrations of the ODE system, which can be computationally costly and inefficient (Kollarčík, 2021).

Simultaneous methods deal with a full discretization of the state, control profiles, and state equations. This discretization can be done using collocation or a Runge–Kutta method. This transcription doesn't require any calculation with ODE solvers, and leads to a large NLP, and needs to be handled with large-scale NLP solvers (Biegler, 2010).

Direct Multiple Shooting (DMS)

Between *sequential* and *simultaneous* methods, *direct multiple shooting* (DMS) approaches are capable of handling unstable DAE systems (not guaranteed by sequential schemes (Biegler et al., 2002)). In this context, the motion search is first achieved by employing a

DMS method, where the control policy and states are discretized in N_T periods. The system's dynamics are satisfied by imposing the commented constraints and integrating the system dynamics over each step k. The solution accuracy of

$$z(t_i) = z(t_{i-1}) + \int_{t_{i-1}}^{t_i} f(z(t),t)dt, \tag{2.7}$$

will depend of the robustness of the integration method implemented. Easier integration techniques like Euler (Hindmarsh et al., 1984) clearly demand less computational effort than more complex methods like Runge-Kutta (Hairer and Wanner, 1996) or a DAE solver by compromising the solution's accuracy. Relaxing the system's dynamic constraints by using simpler solvers to obtain an "initial guess" is a common optimization trick that may help the solver avoiding getting stuck in the local minima. A more elaborate integration method is later used to redefine the "initial guess" (Biegler, 2010).

Collocation Methods

Direct Collocation Methods (DCMs) also known as *full-simultaneous* methods, approximate the continuous functions of the optimization problem in polynomials. In this way, this approach substitutes the integration method used in DMS by algebraic equality constraints enforced at the collocation points. In other words, it uses an implicit solver that satisfies the system's dynamics at the collocation points instead of an explicit integrator. Trapezoidal or Hermite-Simpsons methods with relativity low-order polynomials are often used to implement the *direct collocation method* implicit integrator. As expected, the new extra collocation constraints increase the NLP problem dimension and its sparsity. Modern NLP algorithms exploit the problem's sparsity to reach solutions faster. Typical *direct collocation method*s cannot guarantee the system's accuracy (eq. (2.7)) as using an *Ordinary-Differential Equations* (ODE) solver (like in DMS). A more complex variation of *direct collocation method*s are *Direct Orthogonal Collocation* (DOC) methods (Finlayson, 1980) which employ high-order polynomials. By considering the correct sets of orthogonal polynomials, it is possible to increase the accuracy's order of the collocation scheme (the result is still an approximation to the true ODE solution) (Biegler, 2010). A more detailed explanation of orthogonal polynomials is found in Hale and Townsend (2013); Trefethen (2012).

Section Organization

Subsection 2.2.1 briefly discusses the calculation of the launch configuration at the instant when the robot lifts off the ground. Subsection 2.2.2 comments about the formulation of the BVP as an NLP problem and how this NLP is solved using DMS and DOC methods. Finally, subsection 2.2.3 presents a comparison between the implemented schemes and the TRT in terms of dimension, iterations, and convergence time.

2.2.1 The Launching Instant

The robot configuration at the instant when the robot takes off the ground is named the launching state. This state is correlated to the CoM's linear velocity with respect to the coordinate frame and the centroidal angular momentum ${}^{c}L$. The desired take-off linear velocities along the x and y axes are denoted by $\dot{c}_{xd}$ and $\dot{c}_{yd}$, respectively, and can be obtained using parabolic-based physics equations,

$$\dot{c}_{xd} = \frac{h_x}{T_f} \quad \dot{c}_{yd} = \frac{T_f\, g}{2} \qquad h_y = \frac{\dot{c}_{yd}^2}{2\, g} \tag{2.8}$$

where h_x is the hop length h_x, h_y the maximum height, and T_f the flight time. These equations are simplified by using a polar notation, where the launch velocity is denoted by $v = \sqrt{\dot{c}_x^2 + \dot{c}_y^2}$ with an angle $0 \leq \theta \leq \pi/2$ in rad with respect to the ground (x-axis). In this sense, an angle of $\theta = \pi/2$ rad will result in a vertical hop with zero velocity along the x-axis. During the launching motion, small values of $\theta \leq \pi/4$ rad may cause a high reaction force along the x-axis that exceeds the friction cone resulting in a slip. To produce a traveling hop $h_x \neq 0$, the launching state is next rewritten in terms of the hop length h_x

$$v_d = \sqrt{\frac{h_x\, g}{\sin(2\,\theta_d)}}, \qquad \begin{bmatrix} \dot{c}_{xd} \\ \dot{c}_{yd} \end{bmatrix} = v_d \begin{bmatrix} \cos\theta_d \\ \sin\theta_d \end{bmatrix}, \tag{2.9}$$

where v_d is the desired launching velocity at an angle θ_d. In this context, the new equation does not required to know in advance the flight duration T_f.

The desired centroidal angular momentum is more complex to obtain than the linear velocities because it is necessary to consider the robot's motion during the flight and the desired landing configuration. During the flight phase, based on the conservation of momentum's law, the robot can modify its rotational velocity by varying the mass distribution about the CoM (given that the rotational momentum is constant during this phase). In this context, the

landing configuration depends on the motion and the change of rotational velocity during the flight. Now, assuming that the launching and landing configurations are barely similar and in both scenarios the robot has the leg extended, that is the configuration where the robot has less rotational velocity, a flight controller can execute smooth motions to correct the robot pose prior to landing. In this sense, the solution will consist in obtaining the motion that produces less rotation, so the robot can shrink during the flight to adjusts itself to the desired touchdown configuration.

As the scope of this chapter is to compare the hopping strategy with the results obtained by Azad and Featherstone (2013), we use the same robot and same launching states presented in the paper.

The following subsection explains how to obtain the hopping motion profile by solving an NLP problem. The problem is solved using the DMS approach and using two different polynomials with the DOC method.

2.2.2 NLP Problem Formulation

The launching motion is implemented as an NLP problem, where the constrained parameter optimization problem has nonlinear terms in its objective or constraints functions. In this sense, it is necessary to define the problem constraints and the decision variables to control during the optimization. This launching NLP problem implementation is done using Matlab with the software package CasADI (Andersson et al., 2018) and the solver Interior Point OPTimizer (IPOPT) (Wächter and Biegler, 2006) to minimize a given cost function J by controlling the decision variables and satisfying the imposed constraints.

Once the problem is properly described and discretized in $N_T = 1500$ steps to have a small sampling time $T_s = 0.001\,\text{s}$ according to the motion's duration obtained in Azad and Featherstone (2013) (1.525 s). The NLP problem is formulated in initial constraints to describe the starting state and conditions of the task to solve, loop constraints containing a set of conditions evaluated every iteration of the optimization, and terminal constraints

specifying the final requirements to achieve at the end of the task.

$$\min_{\tau_2,T} \quad J = \int_0^T w_r\,\tau_2(t)^2\,dt + w_t\,T \tag{2.10}$$

subject to:

$$0.5\,\mathrm{s} \leq T \leq 2\,\mathrm{s}, \tag{2.11}$$

initial constraints

$$q_0 = \begin{bmatrix} 0 & \pi/2 \end{bmatrix}^\top, \quad \dot{q}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top, \tag{2.12}$$

loop constraints

$$\begin{bmatrix} -\pi/2 & -\pi/2 \end{bmatrix}^\top \leq q(k) \leq \begin{bmatrix} \pi/2 & \pi \end{bmatrix}^\top, \quad |\dot{q}(k)| \leq \infty, \tag{2.13}$$

$$|\tau_2(t)| \leq \tau_{max}, \tag{2.14}$$

$$\dot{q}(k+1) = \dot{q}(k) + \int_{t_k}^{t_{k+1}} f(q,\dot{q},\tau_2)dt, \qquad q(k+1) = q(k) + \int_{t_k}^{t_{k+1}} \dot{q}dt, \tag{2.15}$$

terminal constraints

$$\left| \begin{bmatrix} \dot{c}_x & \dot{c}_y \end{bmatrix}^\top - \begin{bmatrix} 1.1510 & 1.7157 \end{bmatrix}^\top \right| \leq 1 \times 10^{-3}, \quad \frac{|^cL + 12.0995|}{m} \leq 0.17. \tag{2.16}$$

where the objective function (2.10) is in Bolza form combining a Lagrange term where there isn't a terminal cost and the second is the Mayer term, in which there is no running cost (Kelly, 2017). The gains are $w_r = 0.25$ and $w_t = 1000$.

At the beginning of the optimization, the motion duration T is added as a control variable and bounded in (2.11), where the maximum time is set a little bit bigger than the motion duration obtained in Azad and Featherstone (2013) (1.525 s). Then, (2.12) defines the initial states of the system, where the robot starts in an upright position with zero velocity.

At the loop constraints, we define the operating regions for the positions and velocities for the joints and the actuation torque. The positions and velocities of the joints are described in equation (2.13) in rad and rad/s, respectively. The actuation torque is delimited by a parameter τ_{max} in equation (2.14) and its value will be discussed in next section. An extra equation inserts the system's dynamics commented in (2.3) as a constraint in (2.15) to ensure the system's continuity during the optimization. The following subsection will discuss the employable methods to achieve this integration at every step k.

Finally, the terminal constraint in equation (2.16) specifies the desired launching conditions that need to be accomplished at the end of the motion to take-off. At the equation, ${}^{c}L$ denotes the centroidal angular momentum of the whole robot, and $m = 16\,\text{Kg}$ denotes the total mass of the robot.

The subsequent subsection introduces a scheme to speed up the NLP solution by employing collocation methods combined with orthogonal polynomials. This modification makes the formulation bigger by increasing the number of loop constraints and sparsity obtaining an optimal solution faster.

2.2.3 Optimization Results

	TRT	DMS	DOC-LG	DOC-LGR
Number of Decision Variables	-	7505	25505	25505
Number of Constraints	-	6007	24007	24007
Number of Solver Iterations	-	45	166	43
Cost Function Value	-	1.923e+03	1.923e+03	1.923e+03
Execution Time (s)	0.65	58.91	70.98	20.16
Time per Iteration(s/it)	-	1.3091	0.4276	0.4688
Launching Instant (s)	1.5250	0.8728	0.8728	0.8728

Table 2.2 Comparison table between the following methods: (i) TRT, (ii) DMS, (iii) DOC with Legendre-Gauss polynomial (DOC-LG) and (iv) DOC with Legendre-Gauss-Radau polynomial (DOC-LGR).

The previously commented methods (DMS, DOC, and TRT) are used to produce a leap of 0.5 m with launching velocity $v_0 = 2.0657$ m/s at $\theta = 0.5916$ rads; with the robot described in section 2.1. DMS procedure executes the state integration of equation (2.15) by using an explicit 4th order Runge Kutta integration method. While the DOC approach uses an implicit Runge-Kutta with an orthogonal polynomial like Legendre-Gauss (LG), Legendre-Gauss-Radau (LGR), and Legendre–Gauss–Lobatto (LGL) polynomials. Garg et al. (2010) presents a study using the commented orthogonal polynomials (Legendre-Gauss, Legendre-Gauss-Radau, and Legendre-Gauss–Lobatto) for trajectory optimization problems. The authors noticed that using Legendre-Gauss–Lobatto polynomials adds constant oscillations about the true solution in contrast with Legendre-Gauss and Legendre-Gauss-Radau polynomials which converge faster without oscillations.

In this sense, TRT, DMS, DOC-Legendre-Gauss, and DOC-Legendre-Gauss-Radau approaches are selected to obtain the launching motion with a maximum torque of $\tau_{max} =$

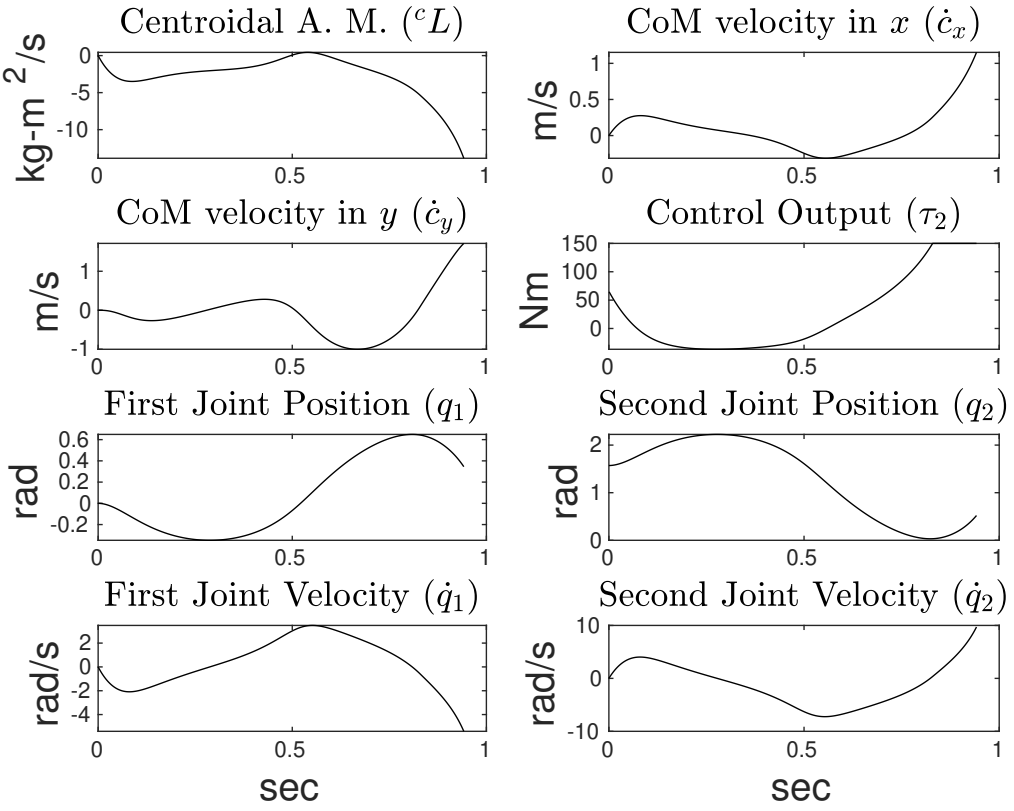

Figure 2.2 Optimized launching motion with a torque τ_2 saturation limit at 150 Nm.

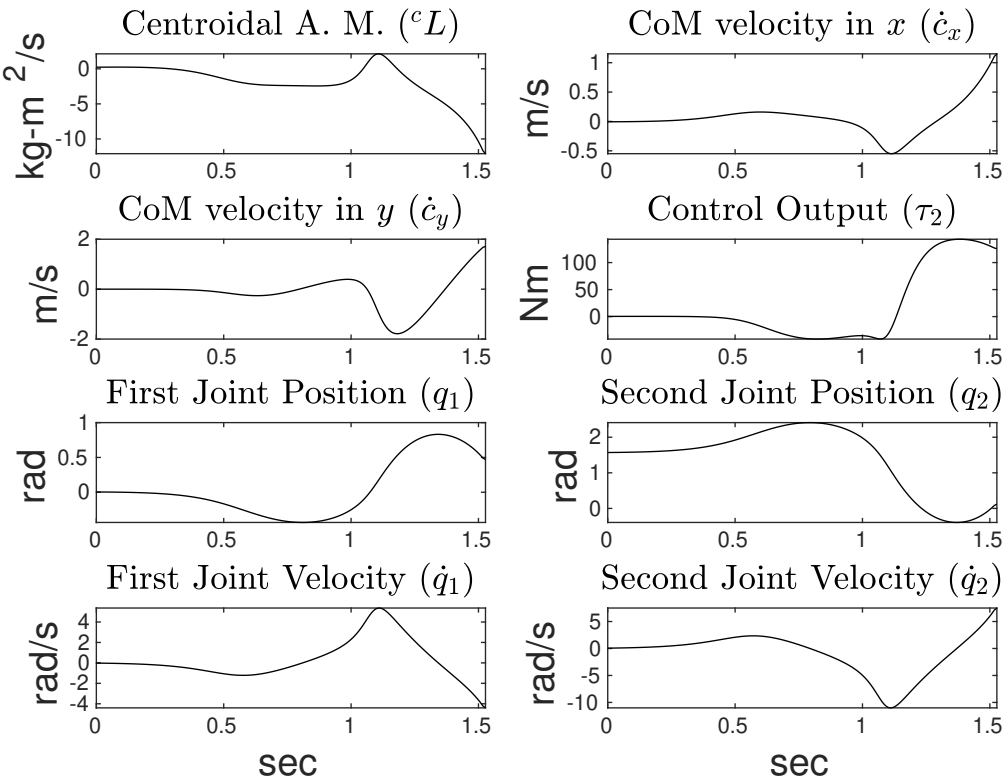

Figure 2.3 Launching motion obtained with the TRT introduced by Azad and Featherstone (2013)

300 N. Table 2.2 compares the four strategies in terms of optimality and computer efficiency on a Lenovo laptop with a Core i7-6500U CPU @ 2.5GHz and eight gigabytes of RAM. The table demonstrates that the number of decision variables in the DOC methods increased more than three times, and the number of constraints increased almost four times compared with the DMS approach. As expected, the DOC approaches increased the NLP problem size by implementing an implicit solver.

The DMS, DOC-Legendre-Gauss, and DOC-Legendre-Gauss-Radau approaches found the same launching motion and minimized cost function at the end of the optimization. DMS strategy took less time for finding the optimal solution than DOC-Legendre-Gauss, but the DOC-Legendre-Gauss took three times more iterations than the DMS approach. On average, each DOC-Legendre-Gauss iteration took three times less than the DMS iteration. According to Biegler (2010), the Legendre-Gauss-Radau polynomial has better stability properties than Legendre-Gauss, solving the problem with fewer iterations.

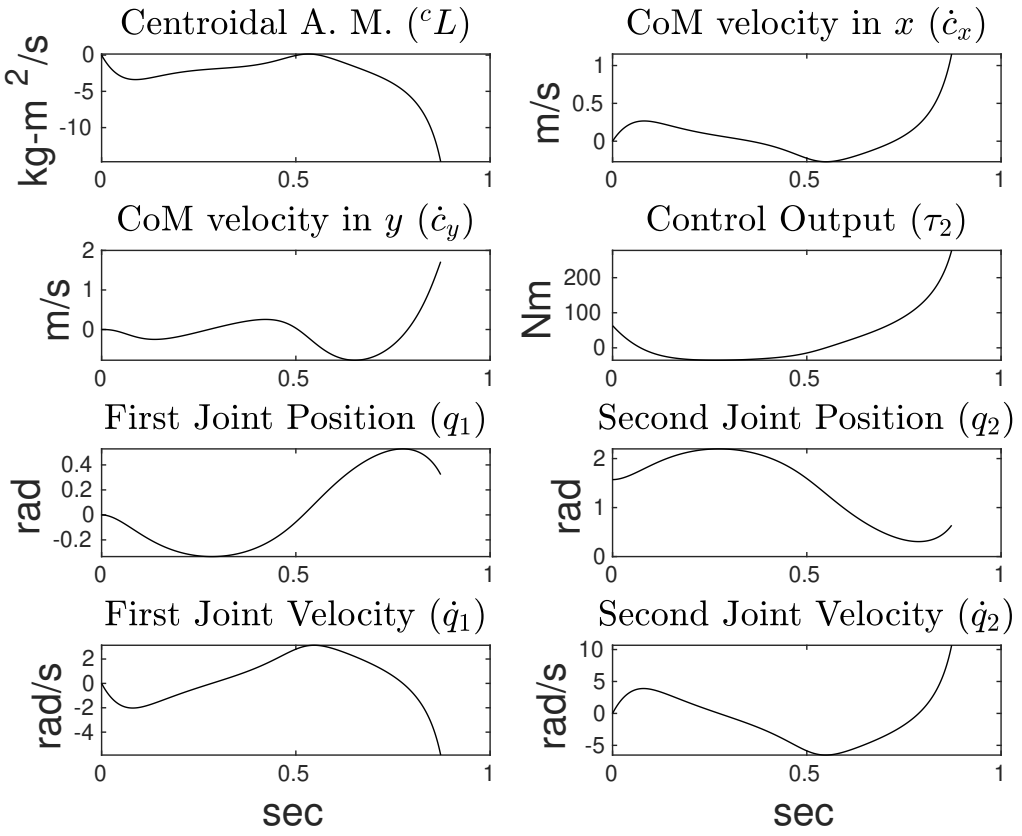

Figure 2.4 Optimized launching motion with a torque τ_2 saturation limit at 300 Nm.

DMS, DOC-Legendre-Gauss and DOC-Legendre-Gauss-Radau motion is presented in figures 2.6 and 2.4. The optimized motion achieved the following values at the launching instant: $L = -14.7072$ $\dot{c}_x = 1.1520$ $\dot{c}_y = 1.7147$. Figures 2.5 and 2.3 present the launching motion obtained using TRT which turned out to take longer 1.525 s than the optimized ones 0.8728 s. Considering how TRT is constructed, the launching state is the same as the desired, whereas, in the optimization approaches, the launching state is into the offset imposed at the NLP formulation (equation (2.16)).

The optimized torque profile is smother than the obtained by TRT but required bigger limits τ_{max} to achive the motion. To achive a motion within the same torque range, the torque limit is modified to $\tau_{max} = 150\,\text{Nm}$ and the new optimized motion takes $T = 0.9418\,\text{s}$ and it's presented in figures 2.2 and 2.7. The new torque profile now saturates to 150 Nm at 0.83 s and it achieved the launching state ${}^cL = -13.8648\,\text{Kg}\,\text{m}^2/\text{s}$ $\dot{c}_x = 1.1520$ $\dot{c}_y = 1.7147$.

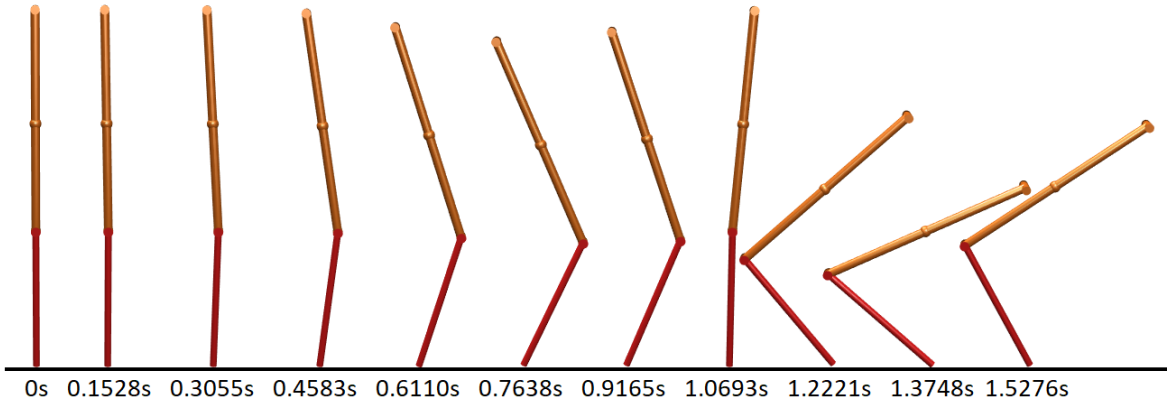

Figure 2.5 Launching motion obtained with the TRT introduced by Azad and Featherstone (2013)

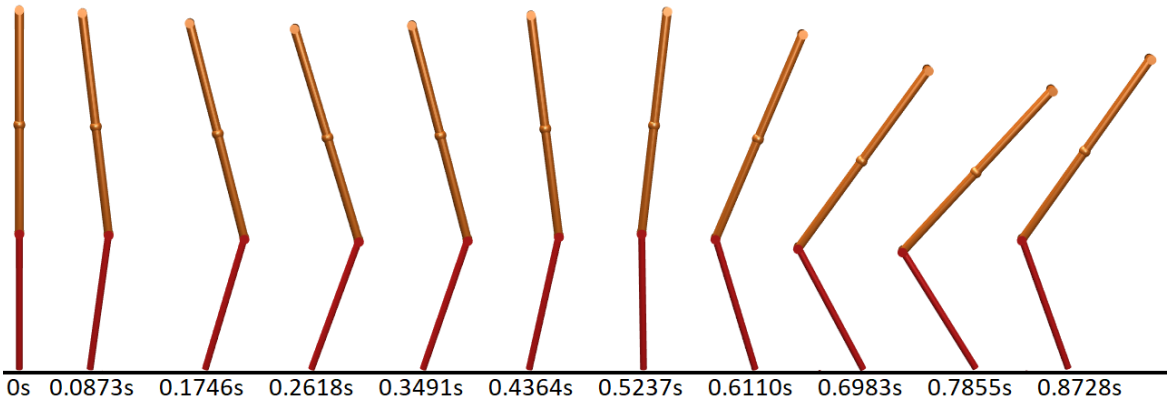

Figure 2.6 Optimized launching motion with a torque τ_2 saturation limit at 300 Nm.

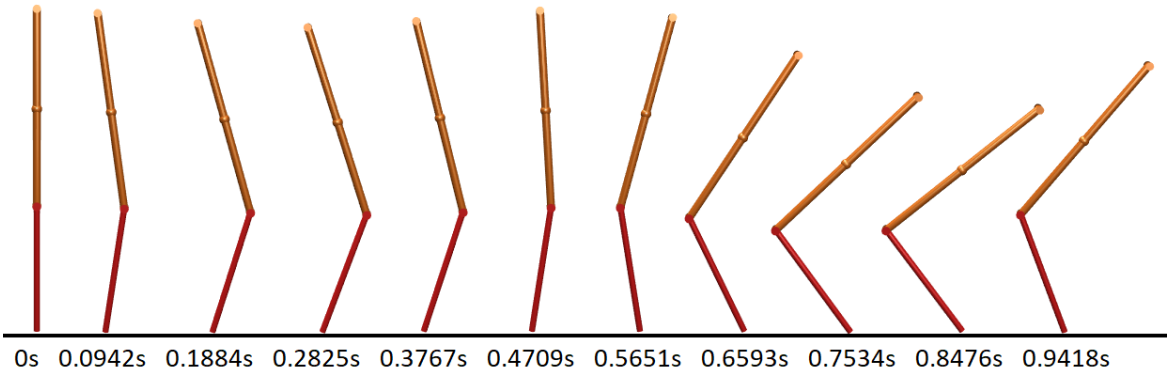

Figure 2.7 Launching motion obtained with a torque τ_2 saturation limit at 150 Nm.

2.3 Balance Controller

This section presents a brief outline of a balance control theory that is explained in detail in Featherstone (2017, 2018), and which has its roots in earlier work in Azad (2014); Azad and Featherstone (2013). This theory was developed with the objective of achieving balance quickly and maintaining it in the presence of large fast motions. Moreover, it has been demonstrated experimentally on a reaction-wheel pendulum (Driessen et al., 2019), on a single-leg hopping robot (Yim et al., 2020), and it has been adapted to work on a quadruped (González et al., 2020).

2.3.1 Balancing States

Assuming that the foot never slips, it is possible to design a momentum-based controller to maintain the robot's balance. Consequently, the center of mass of the robot along the x axis is:

$$c_x = -\frac{1}{m}(m_1 l_{c1} \sin(q_1) + m_2 (l_1 \sin(q_1) + l_{c2} \sin(q_1 + q_2 - \pi/2))) \tag{2.17}$$

As joint one q_1 is un-actuated, gravity is the only force capable of producing a moment about the support point, modifying the angular momentum of the robot about this point. If we define L to be the angular momentum of the whole robot about the support point, then we have:

$$\dot{L} = -m g c_x, \tag{2.18}$$

where m is the total mass of the robot, g is the acceleration due to gravity (a positive number), and c_x is the x coordinate of the robot's center of mass (CoM). The expression $-m g c_x$ is the moment of gravity about the support. The equation that follows directly from (2.18) is

$$\ddot{L} = -m g \dot{c}_x, \tag{2.19}$$

and the angular momentum is described as:

$$L = H_{11} \dot{q}_1 + H_{12} \dot{q}_2, \tag{2.20}$$

which follows from a special property of joint-space momentum that is proved in the appendix B of Featherstone (2017).

The following two subsections explain two different approaches to solve the balance problem for the monopod robot introduced in section 2.1 with the schemes presented in Azad and Featherstone (2013) and Featherstone (2017), respectively.

2.3.2 PID Momentum Balance Controller

Azad and Featherstone (2013) balance controller uses the previously commented momentum states in a PID controller with an additional term τ^d to control the q_2. The controller calculates the actuation torque without any extra calculations or dynamic compensations. The system's output is:

$$\tau_2 = k_{dd}\,\ddot{L} + k_d\,\dot{L} + k_L\,L + \tau^d \tag{2.21}$$

where k_{dd},k_d and k_L are controller gains, and τ_d is the necessary holding torque at the actuated joint for a desired balanced configuration. The effect of τ_d is to make this configuration an equilibrium point of the closed-loop system. If q_1^d and q_2^d are the joint angles in the desired configuration, τ_d is calculated as:

$$\tau^d = m_2\,l_{c2}\cos(q_1^d + q_2^d). \tag{2.22}$$

The controller's gains calculation and stability proof can be found in Azad and Featherstone (2013).

2.3.3 Balance Theory

The methodology starts by adding a fictitious prismatic joint q_0 joint (in the x-direction) between joint one and the ground. Joint q_0 never moves and does not affect the robot's dynamics. However, it increases the size of the equation of motion to:

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{01} & H_{11} & H_{12} \\ H_{02} & H_{12} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ \tau_2 \end{bmatrix} \tag{2.23}$$

and the extra terms are denoted by:

$$\begin{aligned} H_{00} &= m_1 + m_2, \\ H_{01} &= -m_1\,l_{c1}\cos(q_1) - m_2\,(l_1\cos(q_1) + l_{c2}\sin(q_1)), \\ H_{02} &= -m_2\,l_{c2}\cos(q_1 + q_2 - \pi/2), \\ C_0 &= \sin(q_1)\,(m_1\,l_{c1} + m_2\,l_1)\,\dot{q}_1^2 + m_2\,l_{c2}\sin(q_1 + q_2 - \pi/2)\,(\dot{q}_1 + \dot{q}_2)^2. \end{aligned}$$

By adding the extra joint q_0, it is possible to recall the special property of joint-space momentum used in equation (2.20) to obtain equations linking the joint-space dynamics with

the motion of the CoM:

$$m\dot{c}_x = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 = -\dot{L}/g, \tag{2.24}$$

then, the ground reaction force acting on the robot along the x-axis is $m\ddot{c}_x$, which is

$$\tau_0 = m\ddot{c}_x = -\ddot{L}/g. \tag{2.25}$$

Next, combining (2.19), (2.20) and (2.24) gives

$$\begin{bmatrix} L \\ \dot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{2.26}$$

which gives a direct mapping between the robot's joint velocities and the momentum states L and $\dot{L}$; by assuming that the matrix is invertible (which it will be if the robot is physically capable of balancing), equation (2.26) is solved into:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \frac{1}{gD} \begin{bmatrix} -gH_{02} & -H_{12} \\ gH_{01} & H_{11} \end{bmatrix} \begin{bmatrix} L \\ \dot{L} \end{bmatrix} \tag{2.27}$$

where

$$D = H_{01}H_{12} - H_{11}H_{02}. \tag{2.28}$$

Consequently $\dot{q}_2$ can be expressed as

$$\dot{q}_2 = Y_1 L + Y_2 \dot{L} \tag{2.29}$$

where

$$Y_1 = \frac{H_{01}}{D}, \qquad Y_2 = \frac{H_{11}}{gD} \tag{2.30}$$

Y_1 and Y_2 vary depending on the robot's configurations q, and can be expressed as simple functions of two physical properties of the mechanism: its time constant of toppling, T_c, which measures how quickly the robot falls if the controller does nothing, and its velocity gain (Featherstone, 2015, 2016), which measures the effect on centre of mass (CoM) velocity of a unit change in the velocity of the actuated joint. The formulae are

$$Y_1 = \frac{1}{mgT_c^2 G_v} \qquad Y_2 = -\frac{1}{mgG_v} \tag{2.31}$$

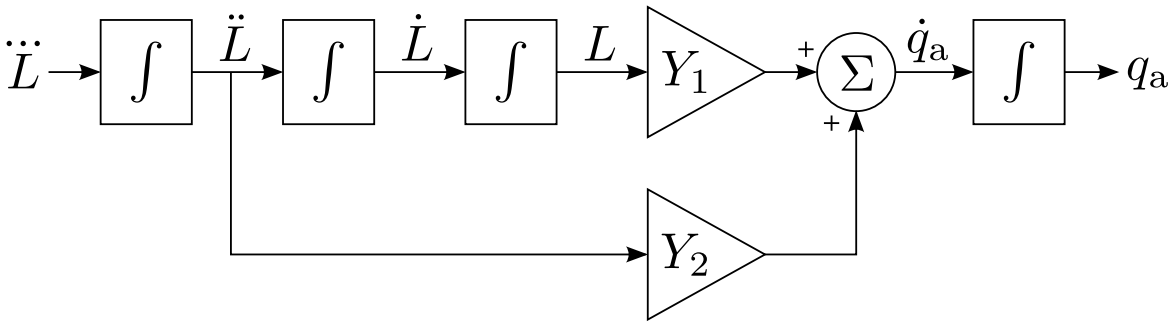

Figure 2.8 Plant describing the dynamics of balancing. q_a is the actuated joint variable, which is q_2 in Fig. 2.1.

where G_v is the linear velocity gain as defined in Featherstone (2016). T_c appears again in the system's open-loop transfer function and it's need to improve the controller's tracking performance which is commented later.

Controller Design

Bringing the open-loop system to the formalism:

$$\dot{x} = Ax + Bu \qquad y = Cx \tag{2.32}$$

we have:

$$\begin{bmatrix} \dddot{L} \\ \ddot{L} \\ \dot{L} \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2 & 0 & Y_1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u \qquad y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} \tag{2.33}$$

The system can be classified as Linear Variant (LV) system because Y_1 and Y_2 vary according to q. Then, it is possible to analyze the system's internal stability or *Bounded-Input, Bounded-Output* (BIBO) stability. BIBO stability is the system property that any bounded input yields a restricted output. In other words, as long as the input signal has an absolute value less than some constant, it is possible to guarantee an output with an absolute value less than some other constant (Chen, 1998).

Calculating the sytem's transfer function by

$$y(s) = C(sI - A)^{-1}B \tag{2.34}$$

it is obtained,

$$q_2(s) = \frac{Y_2\,(s^2 + 1/T_c^2)}{s^4} u(s); \tag{2.35}$$

And it is possible to see that the system has two zeros at $\pm 1/T_c$ and one pole at 0, which means that the system is not stable in open-loop. The end of this section comments on the strategy used by Featherstone (2016) to deal with the zero at the right semi plane $1/T_c$.

Then, to know the controller's capability of moving the system around in its entire configuration space, it is necessary to examine the rank of the controllability matrix C_{ont}, calculated as:

$$C_{ont} = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & Y_2 & 0 & Y_1 \end{bmatrix} \tag{2.36}$$

The system is controllable only if $Y_1 \neq 0$, which means that the robot needs to be above the ground ($c_y \neq 0$) and to have the capability of controlling the robot CoM. Then, the following control law is introduced:

$$u = -kx + k_q r \tag{2.37}$$

where r is a reference signal, k is a gain vector

$$k = \begin{bmatrix} k_{dd} & k_d & k_L & k_q \end{bmatrix} \tag{2.38}$$

The system closed-loop form $\dot{x} = (A - Bk)x - Br$ becomes:

$$\dot{x} = \begin{bmatrix} -k_{dd} & -k_d & -k_L & -k_q \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2(q) & 0 & Y_1(q) & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} + \begin{bmatrix} k_q \\ 0 \\ 0 \\ 0 \end{bmatrix} r \tag{2.39}$$

Subsequently, the system transfer function is obtained:

$$q_2(s) = \frac{k_q Y_2(q)\,(s^2 - 1/T_c^2)}{s^4 + k_{dd}\,s^3 + (Y_2(q)\,k_q + k_d)\,s^2 + k_L\,s + Y_1(q)\,k_q}\, q_c(s) \tag{2.40}$$

The control law introduced in eq. (2.37) is good enough for controlling the robot's balance and q_2 to a fixed position. For a tracking application, Featherstone (2017) uses an acausal filter to add a positive zero into the tracking signal and improve the tracking performance.

2.4 Trajectory Execution

The controller employed to track the optimized launching motion is based on the balance theory introduced in Featherstone (2017). The controller is modified to control the CoM velocity along the x-axis c_x by driving $\ddot{L}$ through a tracking task. Simulation results are compared with Azad and Featherstone (2013) to demonstrate the effectiveness of the new launching strategy.

2.4.1 Launch Controller

The open-loop LV system introduced in eq. (2.33) is modified by changing the system's output to:

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} \tag{2.41}$$

in this sense, we use $\ddot{L}$ as output instead of q_2 given that we are interested in controlling the robot's CoM instead of only q_2. After modifying the system's plant, the open-loop transfer function is

$$y(s) = \frac{1}{s} u(s) \tag{2.42}$$

in this form, the system does not present any zeros in the left semi-plane as commented in eq (2.35). The system behaves like a simple integrator, and to ensure that the tracking error $e_L = \ddot{L}_c - \ddot{L}$ goes to zero during the take-off motion, the control law u is computed as:

$$u = -k_{dd}(\ddot{L} - \ddot{L}_c) + \dddot{L}_c, \tag{2.43}$$

where $\ddot{L}_c$ denotes the desired signal to be tracked, and as the launching motion is known, it is possible to obtain $\dddot{L}_c$ by differentiating $\ddot{L}_c$. Then, the closed-loop system looks like:

$$\dot{x} = (A - Bk)x - Br = \begin{bmatrix} -k_{dd} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2(q) & 0 & Y_1(q) & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} + \begin{bmatrix} k_{dd} \\ 0 \\ 0 \\ 0 \end{bmatrix} (\ddot{L}_c + \dddot{L}_c / k_{dd}) \tag{2.44}$$

and the the transfer function of the closed-loop system is:

$$\ddot{L}(s) = \frac{k_{dd}(1+s/k_{dd})}{k_{dd}+s}\ddot{L}_c = \ddot{L}_c \tag{2.45}$$

which demonstrates that by only imposing a positive gain k_{dd} it is possible to drive $\ddot{L} \to \ddot{L}_c$.

Launch Motion Comparison

Here we use the optimized launching motion obtained in the last section (fig. (2.2)), which limits the robot's torque $\tau_{max} = 150\,\text{Nm}$ to demonstrate that it is possible to satisfy the imposed mechanical constraints imposed during the NLP optimization with the launch controller. The controllers presented in this section (the new controller) and Azad and Featherstone (2013) are compared in terms of achieving the launching state obtained from the NLP optimization with $\tau_{max} = 150\,\text{Nm}$. Figure 2.9 shows the performance of both controllers to track a given motion (gray line). In this case, the trajectory used is to perform a take-off. The blue and black lines demonstrate the tracking execution of the Azad and Featherstone (2013) and the new controller. In contrast with the new controller, Azad and Featherstone (2013) controller presented some irregularities when tracking the angular momentum L, and the torque output τ_2 did not obey the imposed constraint in the NLP optimization. Figure 2.10 demonstrates a scenario where the controller's output is bounded to a maximum torque τ_{max} by imposing a saturation limit at 150 Nm similar to having an actuator with limited power. Table 2.3 shows the launching states obtained in both scenarios (fig. 2.9 and fig. 2.10) with the Azad and Featherstone (2013) and the new launch controllers. The average error is calculated by:

$$Av.Error = \left(\left| \frac{L_d - L}{|L_d|} \right| + \left| \frac{\dot{c}_{dx} - \dot{c}_x}{|\dot{c}_{dx}|} \right| + \left| \frac{\dot{c}_{dy} - \dot{c}_y}{|\dot{c}_{dy}|} \right| \right) \times \frac{100\%}{3} \tag{2.46}$$

where $L_d, \dot{c}_{dx}$ and $\dot{c}_{dy}$ denote the desired parameters obtained from the NLP optimization. In the table 2.3, the Azad and Featherstone (2013) controller with and without limited torque demonstrated larger errors in the linear velocities along the x and y axes ($\dot{c}_x$ and $\dot{c}_y$) which are critical for achieving an accurate motion.

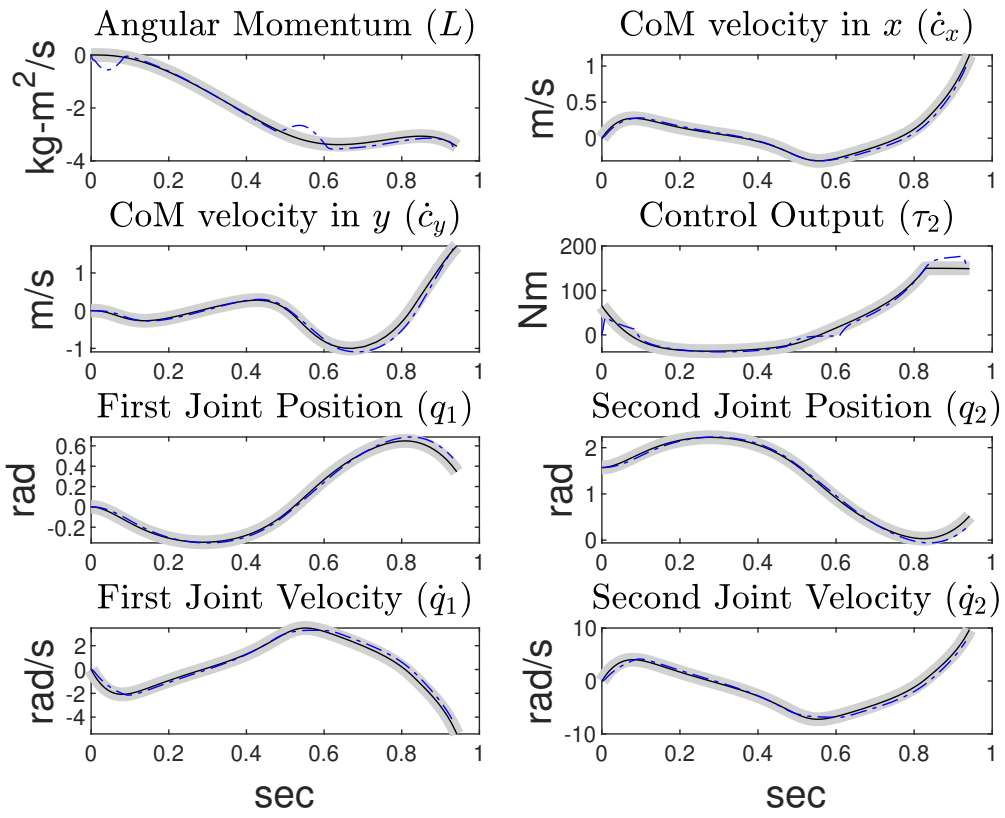

Figure 2.9 Performance comparison of both controllers without a torque τ_2 saturation limit. The gray line denotes the reference signal, the blue line shows the performance obtained with the Azad and Featherstone (2013) controller, and the black line is the tracking obtained with the new launch controller.

2.5 Conclusion

This chapter presented a strategy to accurately control a monoped robot during the launching motion to produce a leap. A DOC-Legendre-Gauss-Radau method was developed to obtain this motion profile. Then, a launch controller was designed based on the balance theory presented in Featherstone (2017) to produce the optimized motion on the robot. The presented strategy was compared with the approach introduced by Azad and Featherstone (2013) in terms of optimality and accuracy. In this context, it was demonstrated that by formulating the BVP as an NLP problem, it is possible to find a quicker and smoother motion than the TRT used in Azad and Featherstone (2013). And that it is also possible to design a controller for driving the robot accurately during the launching.

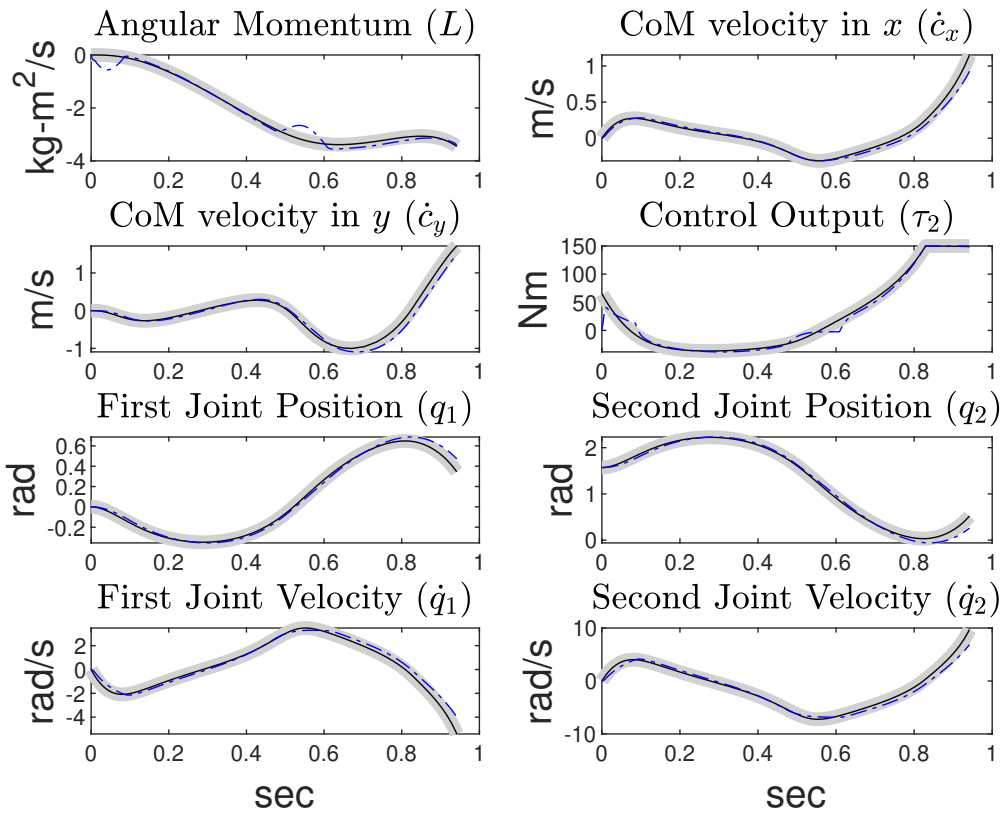

Figure 2.10 Performance comparison of both controllers with a torque τ_2 saturation limit at 150 Nm. The gray line denotes the reference signal, the blue line shows the performance obtained with the Azad and Featherstone (2013) controller, and the black line is the tracking obtained with the new launch controller.

At the optimization phase, *direct multiple shooting* and *direct orthogonal collocation methods* were implemented in search of a computationally cheaper and accurate approach to solve the NLP problem. Initially, with the *direct multiple shooting* method, it was possible to find a feasible and optimal trajectory using an explicit integrator (4th order Rung-Kutta). Collocation methods give another alternative to solve the optimization faster by replacing the explicit integrator with an implicit one. Then, formulating the collocation integration employing orthogonal polynomial instead of low-order ones improves the integration accuracy by the proper choice of the number of collocation points and polynomial type (Legendre-Gauss, Legendre-Gauss-Radau, and Legendre-Gauss-Lobatto). After implementing the NLP problem with Legendre-Gauss polynomial, it was possible to reduce the computational time per iteration more than three times compared to the *direct multiple shooting* method. The

	$L\,(\mathrm{kg\,m^2/s})$	$\dot{c}_x\,(\mathrm{m/s})$	$\dot{c}_y\,(\mathrm{m/s})$	Av.Error
Desired launching state	-3.4448	1.152	1.7147	-
The new launch controller	-3.4528	1.1528	1.7128	0.13%
Previous controller	-3.4784	0.9873	1.6447	6.45%
Previous controller with limited torque	-3.4368	0.9280	1.4989	10.76%

Table 2.3 Comparison table between the new launch controller and Azad and Featherstone (2013) controller with and without limited torque($|\tau| \leq 150\,\mathrm{N}$); for the results presented in figures 2.9 and 2.10.

direct multiple shooting method still solved the optimization faster, using four times fewer iterations. The DOC approach with an Legendre-Gauss-Radau polynomial instead of an Legendre-Gauss obtained the best performance. This approach takes nearly the same time per iteration as the DOC-Legendre-Gauss, and two iterations less than the DMS approach. The DOC-Legendre-Gauss-Radau (DOC-LGR) solved the entire NLP problem 2.9 times faster than the *direct multiple shooting* and 3.5 times faster than the DOC-Legendre-Gauss on a Lenovo laptop with a Core i7-6500U CPU @ 2.5GHz and eight gigabytes of RAM.

Next, to obtain a controller to accurately tracking the motion profile obtained from the commented optimization, the controller introduced by Azad and Featherstone (2013) was compared with a designed launch controller in terms of accuracy. The Azad and Featherstone (2013) controller repeated the desired motion with some significant drifts during the tracking. It also missed the take-off state by 6.45% without obeying the imposed torque limit at the optimization. Consequently, a saturation function was employed to force the controller to follow the constrained torque, but the performance notably degraded by missing the take-off state by 10.76%. On the other hand, the proposed controller reproduced the desired motion satisfying the mechanical and torque constraints imposed during the launching optimization with a small error (0.13%) at the take-off instant.

The next chapter introduces another monopod robot with a passive spring between its actuated and un-actuated joints. The new passive spring enables the robot to store and release elastic energy as needed. The following chapter explores the capability of spring-loaded robots to achieve less impact at landing from a vertical hop. The optimization approach used in this chapter (DOC-LGR method) is employed to accomplish this study.

Chapter 3

Landing

Like hopping, landing requires fast motions to recover the robot's balance by dissipating the system's kinetic energy from the touchdown instant. Different studies have addressed the problem of controlling the transition from flight to stance and from stance back to flight for performing walking or running motions (Hereid et al., 2014; Wensing and Orin, 2013); but this differs from the task of stopping, where there is a need to remove kinetic energy quickly and achieve a fixed configuration.

Spring-loaded mechanisms present shock reduction capability at the touchdown but also the ability to recycle energy during a motion which increases the instantaneous peak power available in the system. Behaviors of this kind are reached by more complex control strategies.

With this study, we are interested in finding a motion profile to drive the robot from a vertical landing velocity of $-6\,\mathrm{m/s}$ to a fixed balance position experiencing the minimum ground contact force. The chapter also extends the trajectory optimization problem by comparing the performance of using a linear and a nonlinear spring in terms of power consumption at the actuator and maximum vertical force.

The robot is equipped with a DC motor and a frictionless reduction driver to produce torque at the actuated joint, he landing motion is obtained by employing the trajectory optimization method introduced in chapter two with a pair of extra variables to find the optimal spring parameters and a soft constraint to reduce the maximum vertical force during the landing. In this context, we find the required voltage profile V by solving an NLP problem, where we formulate the system dynamics, constraints, initial, loop, and terminal conditions.

Our main objective is to explore the applicability of the optimization method commented in the previous chapter for finding an optimal motion profile and the spring parameters for executing a successful landing with the robot. We are also interested in comparing the

Link (i)	Mass (kg)	Length (m)	CoM (m)	Inertia at CoM (kgm^2)
1	0.2	0.2	0.1	0.001
2	0.3	0.3	0.15	0.003
3	2	0.5	0.33	0.08

Table 3.1 Length and inertia parameters of the robot shown in Fig.3.1.

benefits and disadvantages of employing a nonlinear spring over a linear one in terms of control complexity, performance, and electrical energy consumption.

3.1 Robot Model

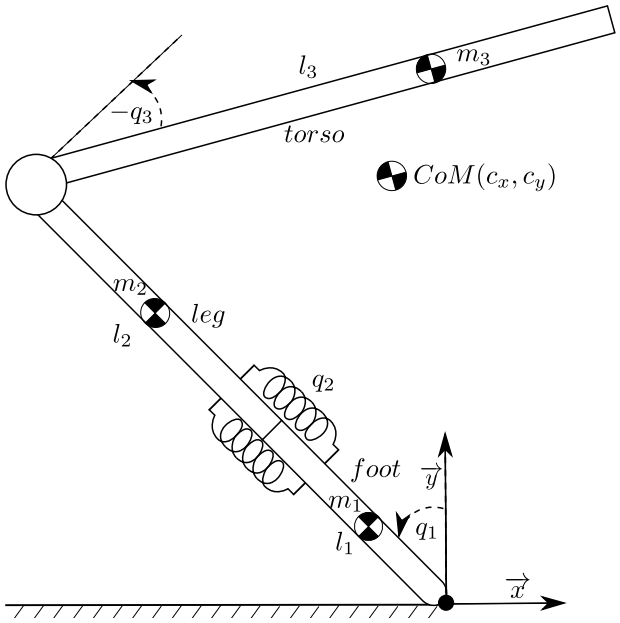

Figure 3.1 Spring loaded monoped robot.

The springy-leg robot shown in Fig. 3.1 is a planar, three-link mechanism in which the links 1, 2, and 3 are the foot, the leg, and the torso, respectively. Joint 1 is a passive revolute joint that models the contact between the foot and ground; joint 2 is a spring-loaded passive prismatic joint, and joint 3 is the actuated joint. Table 3.1 shows the links' mass and length parameters. The symbols m_i, l_i, and I_i appearing below denote the mass, length, and rotational inertia about the center of mass (CoM), respectively, of link i. The joint variables are q_1, q_2 and q_3. When all joints are zero, the leg is vertical, and the torso is horizontal out

to the right. Positive motion of a revolute joint i rotates link i counter-clockwise relative to link $i-1$; and positive motion of joint 2 extends the leg (so the actual length of the leg is $l_1 + l_2 + q_2$). In Fig. 3.1, q_1 is positive, q_2 is zero and q_3 is negative. The whole structure is clamped to the ground considering the gravity acceleration $g = 9.8\,\mathrm{m/s^2}$.

We now introduce two fictitious extra joints along the x and y axes between joint one and the base. These imaginary joints are named "joint x" and "joint y". They never move, and therefore never affect the robot's dynamics. Their purpose is to increase the number of coefficients in the robot's equation of motion, which now reads

$$\begin{bmatrix} H_{xx} & H_{xy} & H_{x1} & H_{x2} & H_{x3} \\ H_{xy} & H_{yy} & H_{y1} & H_{y2} & H_{y3} \\ H_{x1} & H_{y1} & H_{11} & H_{12} & H_{13} \\ H_{x2} & H_{y2} & H_{12} & H_{22} & H_{23} \\ H_{x3} & H_{y3} & H_{13} & H_{23} & H_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ 0 \\ F_s \\ \tau_3 \end{bmatrix}, \tag{3.1}$$

where H_{ij} are the elements of the joint-space inertia matrix, $\ddot{q}_1$, $\ddot{q}_2$ and $\ddot{q}_3$ are the joint acceleration variables, C_x, C_y, C_1, C_2 and C_3 elements contain gravity, Coriolis and centrifugal forces. Here, F_x and F_y are the necessary forces along the x and y axes to produce zero acceleration at joint x and joint y; τ_3 is the torque command at the actuated joint (Joint 3); and F_s is the force produced by the spring. This spring force can be calculated based on a linear and a nonlinear model, described by

$$F_s = -K_s \left(q_2 + G\, q_2 \frac{q_2 + l_1}{l_1} \right), \tag{3.2}$$

where K_s the stiffness coefficient, and $-0.9 \leq G \leq 0.9$ is a bounded extra parameter that let us switch between a linear spring $G = 0$ and a nonlinear one in regressive $0 < G \leq 0.9$ or progressive $-0.9 \leq G < 0$ modes (Fig.3.2). The spring model does not consider any damping coefficient. We will demonstrate the robot's actuator capability of working also as a damper during the landing.

Given that our model assumes that the robot is clamped to the ground and only rotations are allowed around the support point q_1, the initial joints velocities $\dot{q}_i$ are calculated as

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{22} & H_{23} \\ H_{13} & H_{23} & H_{33} \end{bmatrix}^{-1} \begin{bmatrix} H_{x1} & H_{y1} \\ H_{x2} & H_{y2} \\ H_{x3} & H_{y3} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{3.3}$$

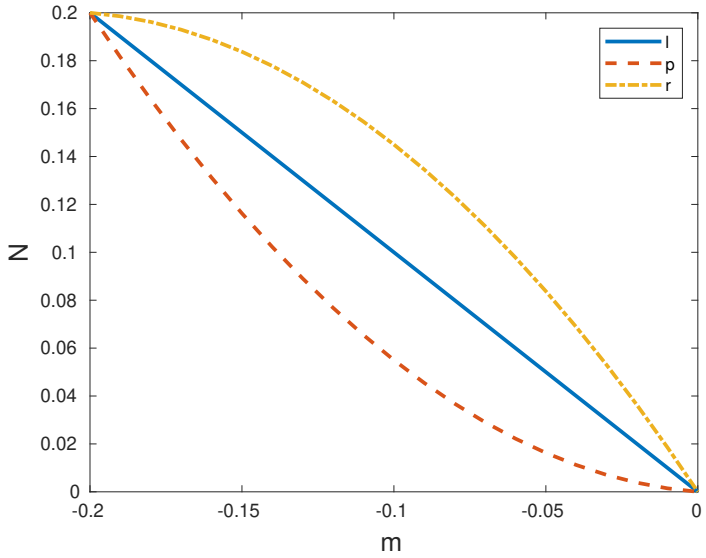

Figure 3.2 Spring profile for different G values with a hypothetic stiffness of $1\,\mathrm{N/m}$. "l" indicates the linear model $G = 0$, "p" and "r" denote the nonlinear model in progressive $G = -0.9$ and regressive $G = 0.9$ modes.

where u_x and u_y mean the robot's center of mass (CoM) velocity at the touchdown along the x and y axes. This equation solves the impulsive dynamics of the robot (Featherstone, 2008) assuming a plastic collision between the foot and the ground.

To guarantee the feasibility of the motion profile for a robot with a mobile base and a high-friction solid rubber foot on a hard floor, it is necessary to ensure that the force F_x is bounded during the landing. F_y also needs to be limited to avoid hypothetic mechanical damages due to high impact forces. The forces and joints accelerations are obtained by solving

$$\begin{bmatrix} -1 & 0 & H_{x1} & H_{x2} & H_{x3} \\ 0 & -1 & H_{y1} & H_{y2} & H_{y3} \\ 0 & 0 & H_{11} & H_{12} & H_{13} \\ 0 & 0 & H_{12} & H_{22} & H_{23} \\ 0 & 0 & H_{13} & H_{23} & H_{33} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ F_s \\ \tau_3 \end{bmatrix}, \tag{3.4}$$

and the magnitude of the ground reaction force is,

$$F = \sqrt{F_x(t)^2 + F_y(t)^2}. \tag{3.5}$$

The actuation torque is produced by a 24 Volts DC motor with a frictionless reduction driver of $1:40$ at joint three, where it is necessary to control the motor voltage to produce

MAXON DCX 32 L 24 V	Data-sheet Units	Implemented Units
Torque Constant(k_{tv})	$27.3\,\mathrm{mNm/A}$	$0.0273\,\mathrm{Nm/A} \times N_d$
Speed Constant (k_{tv})	$350\,\mathrm{rpm/V}$	$0.0273\,\mathrm{V/(rad/s)} \times N_d$
Inductance (L)	$0.103\,\mathrm{mH}$	$0.000103\,\mathrm{H}$
Resistance (R)	$0.331\,\Omega$	$0.331\,\Omega$
No Load Current (I_{nl})	$164\,\mathrm{mA}$	$0.164\,\mathrm{A}$
Rotor Inertia	$72.8\,\mathrm{gcm^2}$	$7.28\,\mathrm{kgm^2} \times 10^{-6} \times N_d^2$

Table 3.2 DC motor parameters. The implemented torque and speed constants already consider the ratio of the reduction driver (N_d).

torque in the system. In general, The DC motor's voltage can be controlled using pulse width modulation (PWM). In this sense, the joint three torque τ_3 depends on the amount of current i present in the motor, the motor's torque constant times the ratio of the reduction drive $N_d = 40$. The current is modified depending on the voltage V applied to the engine and the joint's velocity $\dot{q}_3$. The employed motor is similar to the DC motor MAXON DCX 32 L 24V, and its parameters are presented in table 3.2, where the implemented torque and speed constants already consider the ratio of the reduction driver.

The joint torque τ_3 is obtained by

$$\tau_3 = k_{tv}\, i - k_{tv}\, I_{nl} \tanh\left(\dot{q}_3\, N_d\right), \tag{3.6}$$

the first term on the right-hand side denotes the torque produced by the amount of current i in the motor. The second term denotes an approximation to the Coulomb friction in the motor, where k_{tv} denotes the torque and speed constants, I_{nl} is the no-load current, and N_d is the reduction driver ratio.

As the electrical time constant is a fraction of a millisecond, which is way faster than the mechanical dynamics that it could be regarded as instantaneous. We use the following equation to describe the motor's current,

$$i = \frac{V - k_{tv}\, \dot{q}_3}{R}, \tag{3.7}$$

where the input is the voltage V and R denotes the motor's resistance.

3.2 Optimization

This subsection introduces the NLP formulation required to bring the robot from a landing velocity of $-6\,\mathrm{m/s}$ to a fixed balance configuration. The motion is discretized in 500 steps and is allowed to be solved in maximum 2.5 s. The optimizer uses the DOC method and the LGR polynomial described in the previous chapter to perform the intrinsic integration at the three collocation points in every step.

The NLP problem is formulated in initial constraints to describe the starting state and conditions of the task to solve, loop constraints containing a set of conditions evaluated every iteration of the optimization, and terminal constraints specifying the final requirements to achieve at the end of the task.

$$\min_{V(t)} \quad J = \int_0^T F_x^2 + F_y^2\, dt + w_f F_{max}^2 \tag{3.8}$$

subject to:

$$0.1\,\mathrm{s} \leq T \leq 1\,\mathrm{s}, \tag{3.9}$$

$$1\,\mathrm{N/m} \leq K \leq 5\times 10^4\,\mathrm{N/m} \quad -0.9 \leq G \leq 0.9 \tag{3.10}$$

$$1\,\mathrm{N} \leq F_{max} \leq 400\,\mathrm{N}, \tag{3.11}$$

initial constraints

$$q_2 = 0\,\mathrm{m}, \quad \mathfrak{u}_x = 0\,\mathrm{m/s}, \quad \mathfrak{u}_y = -6\,\mathrm{m/s}, \tag{3.12}$$

$$i_0 = 0\,\mathrm{Amp}, \tag{3.13}$$

loop constraints

$$\begin{bmatrix} -\pi/2 & -0.2 & -5\pi/12 \end{bmatrix}^\top \leq q(t) \leq \begin{bmatrix} \pi/2 & 0 & 5\pi/12 \end{bmatrix}^\top \tag{3.14}$$

$$|\dot{q}_3(t)\, N_d| \leq 1183.3332\,\mathrm{rad/s}, \tag{3.15}$$

$$|V(t)| \leq 24\,\mathrm{Volts}, \tag{3.16}$$

$$|F_x(t)| \leq 30\,\mathrm{N}, \quad 0\,\mathrm{N} < F_y(t), \tag{3.17}$$

$$0 \leq \frac{F}{F_{max}} \leq 1, \tag{3.18}$$

terminal constraints

$$\left| \begin{bmatrix} c_x & \dot{q} & F_s & \tau_3 \end{bmatrix}^\top - \begin{bmatrix} 0 & 0 & 0 & 0 & C_2 & C_3 \end{bmatrix}^\top \right| \leq 1\times 10^{-9}. \tag{3.19}$$

The objective function (3.8) has a Lagrange term that minimizes F_x and F_y, and a Mayer term that penalizes the maximum force reached during the landing (Kelly, 2017). The maximum force is obtained by creating a control variable F_{max} (eq. (3.11)), and imposing the constraint (3.18), where the force magnitude is divided by F_{max} and limited between zero and one. In this sense, the optimizer sets F_{max} to a value not bigger than the maximum force because of the existent penalty at the end of the optimization. With a reasonable penalty's magnitude $w_f = 10$, it is possible to minimize the maximum force during the landing. In this way, the objective function is smooth enough to allow the optimizer to converge to a solution with the smallest maximum force.

Then, other parameters aside from the actuation signal (voltage) that need to be optimized are introduced from equations (3.9) to (3.11). The total task execution is bounded in eq. (3.9); the spring parameters are defined in eq. (3.10), this spring model does not consider any damping coefficient; finally, the maximum force variable is defined in eq. (3.10).

The motion starts at the touchdown instant (eq.(3.12)), where the spring is in a rest position, the initial current at the motor is zero and the robot's joint velocities $\dot{q}$ are zero. The landing velocity $\mathfrak{u}$ is transmitted to the robot joints using the equation (3.3) because the dynamic model used in the optimization does not consider any movement nor velocity along joints x and y.

During the landing, the joints position q are delimited based on the physical limits of the robot (eq. (3.14)) in $\begin{bmatrix} \text{rad} & \text{m} & \text{rad} \end{bmatrix}^\top$ and the joint velocities $\dot{q}_1$ and $\dot{q}_2$ are assumed to be unbounded, $\dot{q}_3$ is bounded in equation 3.15 based on the motor's maximum speed. The forces along the x and y axes are limited in equation (3.17). The output voltage is limited to 24 Volts (eq. (3.16)), and the motor's current is initialized at zero amps (eq. (3.13)). In addition to the DC motor, a friction less reduction driver of $N_d = 40$ is used to connect joint three with the motor. The motor's inertia multiplied by N_d^2 is also added to the diagonal element of the joint-space inertia matrix corresponding to joint 3 (i.e., H_{33} in eq.(3.1)). We also added an extra constraint to the problem related to the motor's maximum speed (3.15).

Equation (3.19) presents the terminal constraints, where the robot needs to be in balance, with zero velocity. Moreover, the spring and actuation torque need to produce only the necessary force for the robot to continue in a fixed position.

3.3 Results

Figures 3.3 and 3.4 show the obtained motion using a linear ($G = 0$) and a nonlinear ($G \neq 0$) spring. In the graphs displayed from figure 3.5 to 3.12, the blue and red lines represent the optimized solution for the linear and nonlinear springs, respectively.

Both motion profiles found using the linear and nonlinear springs achieve the fixed position in 2.5 s. The spring stiffness found in the linear model is $480.7420\,\mathrm{N/m}$, and the motion profile is shown in figure 3.3. By allowing G to take a value different than zero the

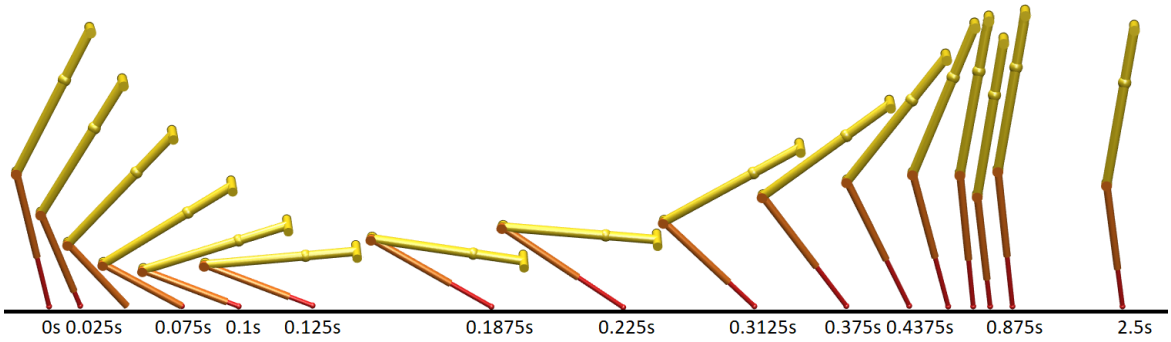

Figure 3.3 Robot's motion during landing using the linear spring.

the spring parameters found are $K = 546.1554\,\mathrm{N/m}$, $G = -0.3134$, and the motion profile is shown in figure 3.4.

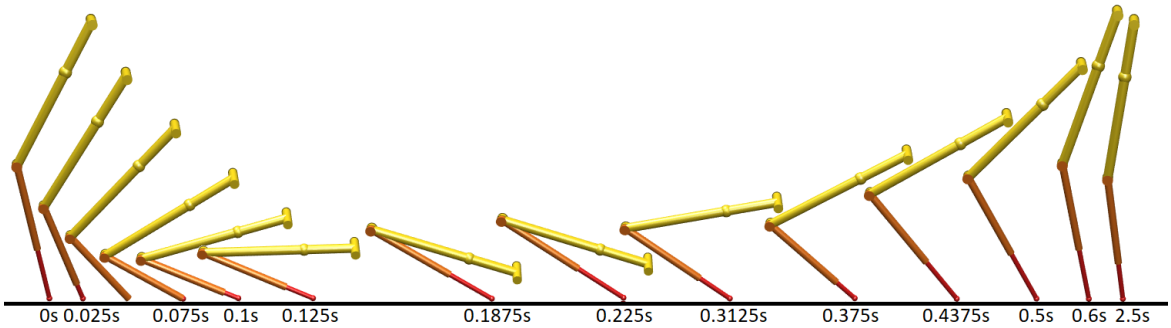

Figure 3.4 Robot's motion during landing using the nonlinear spring.

Both motions demonstrate that it is necessary to produce fast movements during the first 0.18 s. During this time frame, the robot is trying to reduce its downward momentum as gradually as possible so as to minimize the forces required, but this implies maximizing the downward travel of the CoM from the moment of impact to the bottom point in the landing motion.

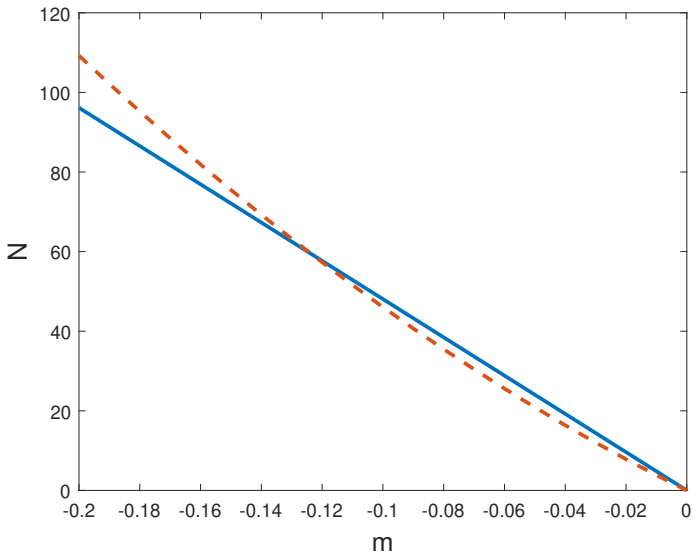

Figure 3.5 Spring profiles obtained from the motion's optimization. The solid blue line indicates the linear model with stiffness of 480.7420 N/m, $G = 0$, and the dashed red line shows the nonlinear progressive model $G = -0.3134$ with stiffness of 546.1554 N/m.

Then, the robot performs a motion to smoothly release the elastic energy stored at the spring during the impact.

In figure 3.5, it is possible to see that the nonlinear spring required 1.14 times more stiffness than the linear one to complete the landing, and both springs deliver a very similar force when it is compressed to around -0.12 m. According to the results, the nonlinear progressive spring ($-0.9 \leq G < 0$) seems to be the best option for landing.

Figure 3.6 shows the position response of the CoM along the x and y axes for the two different springs.

- With the linear spring, the CoM's position along the x-axis c_x starts at 25.7 mm and finishes at 0.1075 nm, and reaches a minimum of -9.37 cm. The CoM's position along the y-axis c_y starts at 67.91 cm and finishes at 66.73 cm, it reaches a minimum of 15.7 cm.

- With the nonlinear spring, the CoM's position along the x-axis c_x starts at 27 mm and finishes at 8.2116 nm, and reaches a minimum of -10.86 cm. The CoM's position along the y-axis c_y starts at 67.87 cm and finishes at 66.17 cm, it reaches a minimum of 16 cm.

Figure 3.7 shows the velocity response of the CoM along the x and y axes for the two different springs.

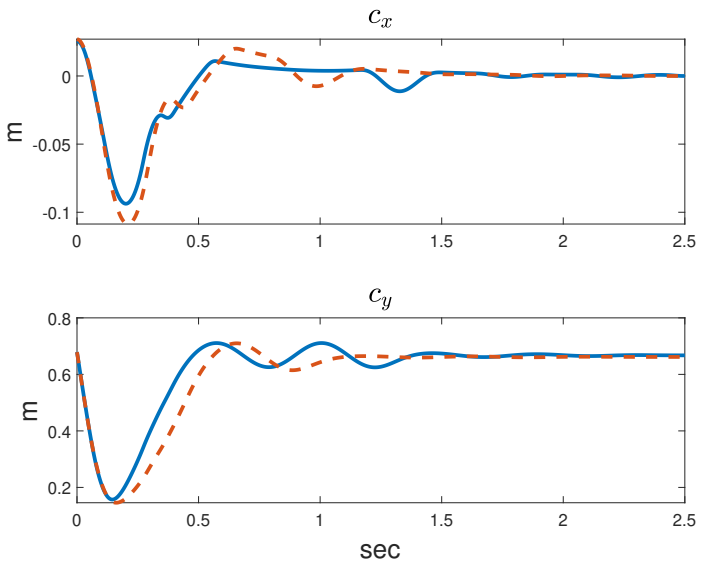

Figure 3.6 CoM position c response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

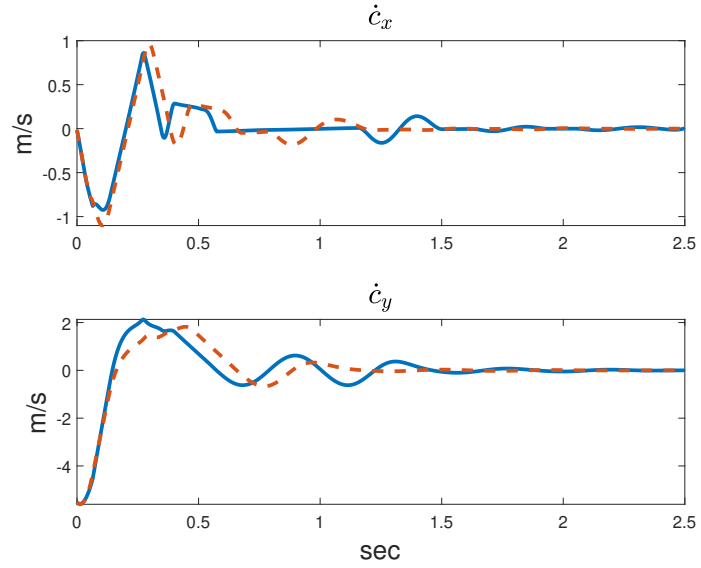

Figure 3.7 CoM velocity $\dot{c}$ response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

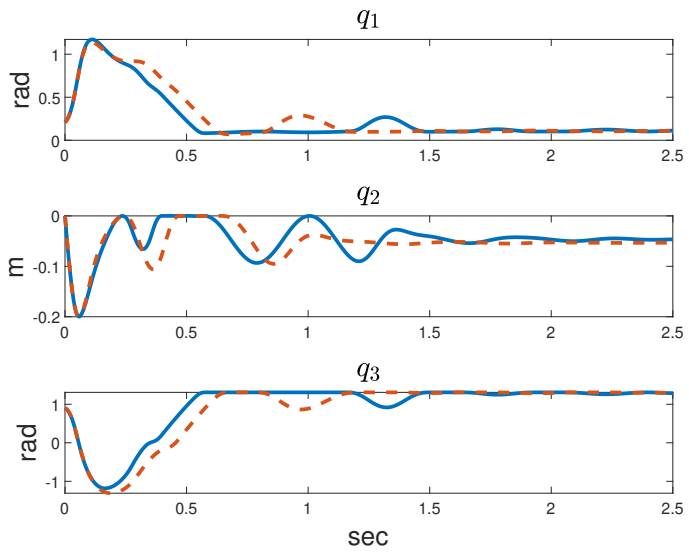

Figure 3.8 Joints position q_i response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

- With the linear spring, the CoM's velocity along the x-axis $\dot{c}_x$ starts at $-1.7\,\mathrm{cm/s}$ and finishes with a velocity of $-8.2734\,\mathrm{nm/s}$. At the beginning of the motion, it moves towards $-92.42\,\mathrm{cm/s}$, and then reaches a maximum positive velocity of $86.28\,\mathrm{cm/s}$. The CoM's velocity along the y-axis $\dot{c}_y$ starts at $-5.5236\,\mathrm{m/s}$, it reaches a maximum positive velocity of $2.1323\,\mathrm{m/s}$ and finishes at $-8.9906\,\mathrm{nm/s}$.

- With the nonlinear spring, the CoM's velocity along the x-axis $\dot{c}_x$ starts at $-1.69\,\mathrm{cm/s}$ and finishes with a velocity of $-3.1463\,\mathrm{nm/s}$. At the beginning of the motion, it moves towards $-1.1018\,\mathrm{m/s}$, and then reaches a maximum positive velocity of $96.30\,\mathrm{cm/s}$. The CoM's velocity along the y-axis $\dot{c}_y$ starts at $-5.5236\,\mathrm{m/s}$, it reaches a maximum positive velocity of $1.8236\,\mathrm{m/s}$ and finishes at $-9.5050\,\mathrm{nm/s}$.

Figure 3.8 shows the evolution of the joint positions q_i during the impact.

- With the linear spring, joint one q_1 moves from $0.2081\,\mathrm{rad}$ to $0.1116\,\mathrm{rad}$ and reaches a maximum of $1.1717\,\mathrm{rad}$. Joint two q_2 starts at zero and finishes at $-46.6\,\mathrm{mm}$, reaching a maximum displacement of $-19.97\,\mathrm{cm}$, and stores a maximum of $9.5907\,\mathrm{J}$ in elastic energy. Joint three q_3 moves from $0.9017\,\mathrm{rad}$ to $1.2884\,\mathrm{rad}$, reaching a maximum displacement of $-1.182\,\mathrm{rad}$.

- With the nonlinear spring, joint one q_1 moves from $0.2074\,\mathrm{rad}$ to $0.1084\,\mathrm{rad}$ and reaches a maximum of $1.1395\,\mathrm{rad}$. Joint two q_2 starts at zero and finishes at $-53.3\,\mathrm{mm}$,

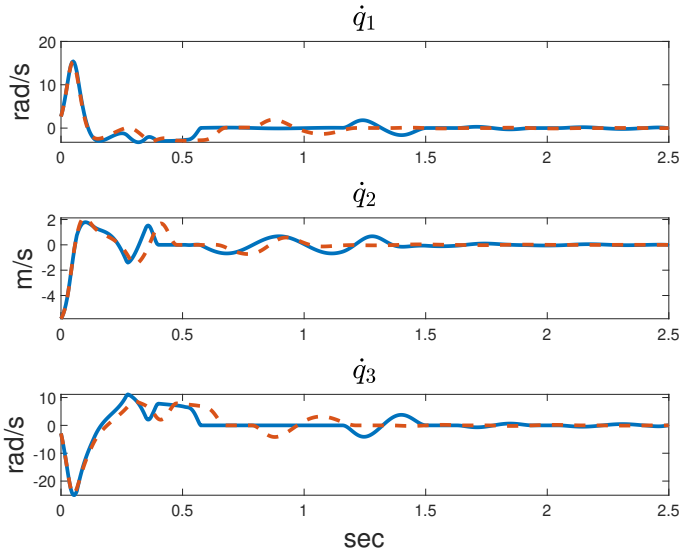

Figure 3.9 Joints velocities $\dot{q}_i$ response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

reaching a maximum displacement of $-19.97\,\text{cm}$, and stores a maximum of $10.8839\,\text{J}$ in elastic energy. Joint three q_3 moves from $0.8980\,\text{rad}$ to $1.299\,\text{rad}$, reaching a maximum displacement of $-1.309\,\text{rad}$.

Figure 3.9 shows the performance of the joint velocities $\dot{q}_i$ during the impact.

- With the linear spring, joint one $\dot{q}_1$ moves from $2.6694\,\text{rad/s}$ to $10.307\,\text{rad/s} \times 10^{-9}$, reaching a maximum velocity of $15.412\,\text{rad/s}$. Joint two $\dot{q}_2$ moves from $-5.8332\,\text{m/s}$ to $10.296\,\text{nm/s}$, reaching a maximum positive velocity of $1.7885\,\text{rad/s}$. Joint three moves from $-2.877\,\text{rad/s}$ to $9.3227\,\text{rad/s} \times 10^{-9}$, reaching a maximum velocity of $-25.0751\,\text{rad/s}$.

- With the nonlinear spring, joint one $\dot{q}_1$ moves from $2.6598\,\text{rad/s}$ to $6.3893\,\text{rad/s} \times 10^{-9}$, reaching a maximum velocity of $15.2171\,\text{rad/s}$. Joint two $\dot{q}_2$ moves from $-5.8342\,\text{m/s}$ to $10.384\,\text{nm/s}$, reaching a maximum positive velocity of $2.1276\,\text{rad/s}$. Joint three moves from $-2.8661\,\text{rad/s}$ to $-1.818\,\text{rad/s} \times 10^{-10}$, reaching a maximum velocity of $-24.7437\,\text{rad/s}$.

Figure 3.10 shows the ground forces along the x axis F_x and the total magnitude F.

- With the linear spring, F_x moves from $620.5\,\text{mN}$ to $0.64\,\mu\text{N}$ complying with the imposed constraint (eq.(3.17)). F moves from $2.0438\,\text{N}$ to $24.525\,\text{N}$, reaching a maximum force of $159.2430\,\text{N}$ at $t = 90\,\text{ms}$.

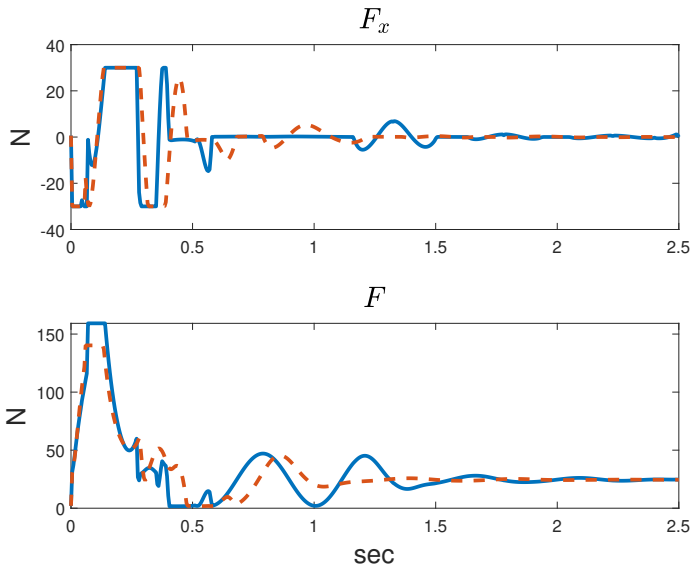

Figure 3.10 Vertical force F and torque profile τ_3 response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

- With the nonlinear spring, F_x moves from 619.7 mN to 0.49 μN complying with the imposed constraint (eq.(3.17)). F moves from 2.044 N to 24.525 N, reaching a maximum force of 140.2414 N at $t = 75$ ms.

Figure 3.11 shows the spring force F_s and the actuation torque τ_3.

- With the linear spring, F_s moves from zero to 22.4227 N, achieving a maximum force of 96.0277 N. τ_3 moves from zero to 1.2631 Nm, reaching a maximum torque of 70.8505 Nm at $t = 0.14$ s.
- With the nonlinear spring, F_s moves from zero to 22.4305 N, achieving a maximum force of 109.0452 N. τ_3 moves from zero to 1.209 Nm, reaching a maximum torque of 62.1355 Nm at $t = 0.135$ s.

Figure 3.12 shows the current i and the voltage V during the impact for the two different springs.

- With the linear spring, i moves from zero to 1.1574 A, achieving a peak current of 64.9205 A. V moves from -9.68 V to 0.3741 V, reaching a peak of -24 V at $t = 0.04$ s.
- With the nonlinear spring, i moves from zero to 1.1078 A, achieving a peak current of 56.9346 A. V moves from -9.77 V to 0.3669 V, reaching a peak of -23.4245 V at $t = 0.04$ s.

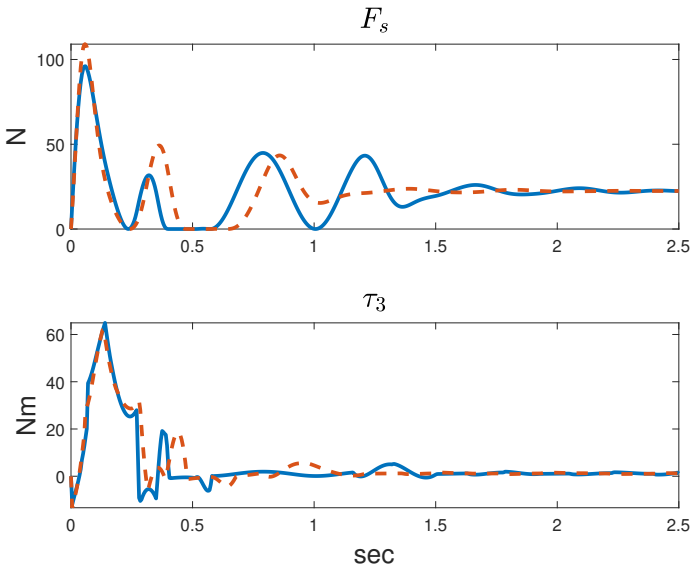

Figure 3.11 Vertical force F and torque profile τ_3 response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

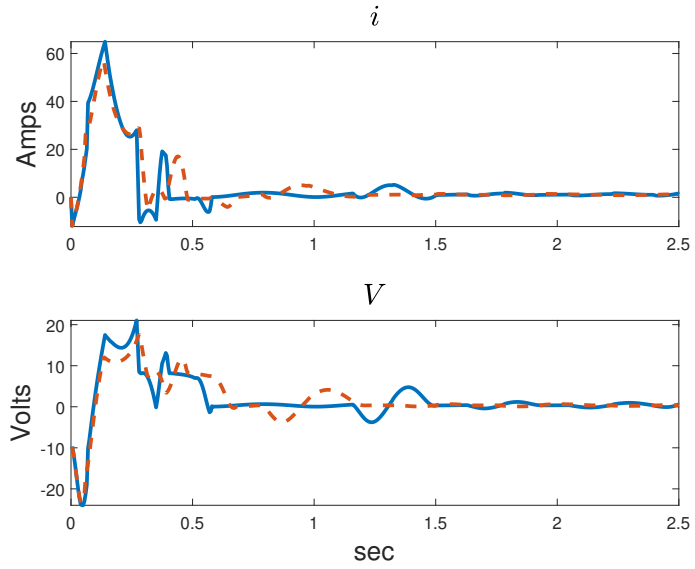

Figure 3.12 Voltage V and current i response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

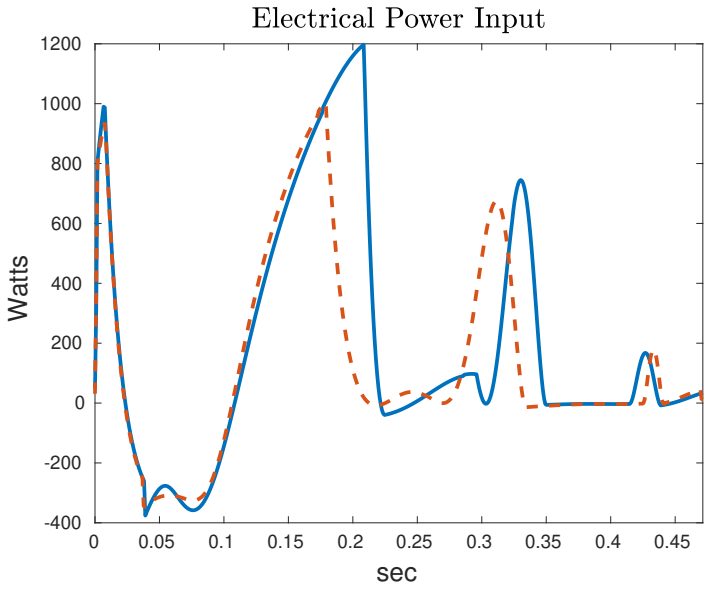

Figure 3.13 Electrical power input response of Fig.3.1 system with a linear (solid blue line) and a nonlinear (dashed red line) spring.

Figure 3.13 shows the electrical power input during the impact for the two different springs.

- With the linear spring, the motor demanded a maximum electrical power of 1197.3 watts at $t = 0.2084$ s.
- With the nonlinear spring, the motor demanded a maximum electrical power of 1004.8 watts at $t = 0.1782$ s.

3.3.1 Discussion

As shown in figure 3.5, the capability of the nonlinear spring to deliver more force at maximum compression seems to have an impact on the movement of the robot's CoM along the x-axis after the touchdown. The nonlinear spring setup experienced a larger movement. The robot's CoM motion along the y-axis shows a more damped response related to the linear spring.

Because of the robot's structure, it is impossible to drive the CoM along only one axis while keeping zero velocity along the other axis, as shown in the graph of $\dot{c}_x$ where the CoM moves in the x direction during the impact. Figures 3.6 and 3.7 also show that the motion with the nonlinear spring had a smaller reaction velocity along the y-axis but a bigger movement along the x-axis, which may demand more effort to keep the robot's balance

during the landing. The nonlinear spring improves the deceleration of the CoM along the y axis, requiring extra movement along the x-axis, increasing the control complexity to keep the robot's balance while reaching a stable configuration.

The nonlinear spring added more movement of the CoM along the x-axis but presented a smother response along the y-axis. It also experienced bigger peak transients at the CoM's velocity along the x-axis. At y-axis, the nonlinear spring had a smaller reaction velocity than the linear one, the response is also smother and delayed from the linear one.

In figure 3.8, it is also possible to observe a similar response at the joint level for both springs. Specifically, both motions have a very similar profile before the springs had shrunk to their maximum at $t = 60\,\text{ms}$, then, the response of the nonlinear spring seems to be delayed in about 80 ms respect to the linear one. The nonlinear spring reaches the nominal position (position required to keep the system in a fixed configuration) quicker than the linear spring. On the other hand, the linear spring setup presented more transients after the maximum displacement, exhibiting the task's complexity of controlling the flow of elastic energy that goes in and out of the spring during the landing. Notably, this flow of elastic energy needs to be minimized during the task to reach a fixed configuration. The nonlinear spring shows an extra damping capability which settles the system in around 0.3 seconds faster.

Figure 3.10 shows how the extra damping capability helps the nonlinear spring to experience fewer reaction forces and transients along the x-axis after the minimum peak force (-30N). It is also possible to observe that the motion with the nonlinear spring produced an 11% lower maximum force F. This is a significant reduction and shows that a nonlinear spring can be advantageous for reducing peak forces on landing. In this particular example, the nonlinear spring also reduced the peak torque needed at the actuated joint by about 12%.

Figure 3.12 shows that the nonlinear spring setup required less power to achieve the task. It demanded a power peak 16.1% smaller.

3.4 Conclusion

This chapter introduced a numerical strategy to find the motion profile and spring parameters for a spring-loaded monopod robot that minimizes the peak force at landing.

In this context, we introduced the spring-loaded monopod robot to exploit the robot's ability to vary its stiffness by modifying its configuration. Since the robot is supposed to have a rubber foot to make contact with the ground and not a mechanism to clamp it to the ground, we have constrained the maximum force along the x-axis to ensure that there are no slips during the motion. Moreover, considering gravity implies that the angular momentum

of the robot about the support point (the foot) can be modified according to the position of the CoM along the x-axis, consequently losing its balance. In this regard, the motion profile successfully minimized the maximum impact force while keeping the robot in balance even when the spring shrunk to its maximum. In this trajectory optimization study, we also added the spring's parameters as control variables to the NLP formulation for finding the optimal values for a linear and nonlinear spring. We did not consider any damping coefficient at the spring in order to explore the control profiles capability of serving as a damper for the whole system. Finally, we modeled the robot's actuator as a DC motor in conjunction with a frictionless reduction driver to obtain a motion profile that could be replicated in reality. In this connection, we added the necessary constraints to the NLP formulation according to the mechanical limitations of the motor.

After running the optimization, the optimizer found the progressive nonlinear spring optimal for this task rather than the regressive one. The nonlinear spring reached a slightly higher spring force, but the ground reaction force and actuator torque were lower than the linear one.

The results also demonstrated that the robot couldn't decelerate the CoM along only the y-axis while keeping zero velocity along the x-axis because of its structure. Furthermore, this disturbance along the x-axis is correlated to the type of springs and the parameters chosen; the linear and nonlinear springs minimized the maximum landing force by almost 17% and 28%, respectively, compared with the spring-mass-damper system with gravity. In addition, before the spring reached its maximum compression, both robots had a similar response. The nonlinear setup showed a delayed reaction with fewer vibrations than the linear setup after the peak force at the spring.

The results in this chapter have demonstrated the complexity of the landing task for a spring-loaded monopd robot. The results pointed that a nonlinear progressive spring can increase the robot's ability to perform a soft landing with less electrical power. It's also evidenced the possibility of solving trajectory optimization and parameter optimization problems by using the orthogonal collocation methods implemented in conjunction with the Rigid Body Dynamics Algorithms (Featherstone, 2008).

The next chapter introduces a modified version of the balance controller presented in Featherstone (2017) applied to the spring-loaded monopod used in this chapter. The controller keeps the robot in balance while it tracks the desired motion described by the third joint. Moreover, with a modified version of the new controller, the robot can control the absolute angle of the hip defined by joints three and one.

Chapter 4

Balancing with a Springy Leg

This chapter presents a simulation study of the problem of balancing a planar double pendulum in which the lower body (the leg) has been modified to include a spring-loaded passive prismatic joint. Robots of this kind can travel by hopping and can also stand and balance on a single point. The purpose of this study is to investigate the degree to which a balance controller can cope with the large and rapidly changing forces from the spring. It is shown that good performance can be achieved using an existing balance controller if the spring-loaded joint is instrumented. So, its position and velocity are considered when calculating the state variables needed by the balance controller.

The main result of this chapter is that the balance controller in Featherstone (2017, 2018) still works when the leg is springy, although the performance is not as good as with a rigid one. Nevertheless, we can perform high-speed motions without losing balance, like directly controlling the position of the actuated joint. By modifying the plant of the system, we also demonstrate the possibility of indirectly controlling the angular position between the foot and the ground, relaxing the usual assumption of controlling the robot's movements based only on the motor-actuated joints.

There is an important distinction between the study presented here and those presented in Featherstone (2016, 2017) on the topic of balancing in the presence of other motions. In the earlier works, it was assumed that the other motions were produced by a motion controller that was executing a prescribed motion command that was independent of the actions of the balance controller, and that was known in advance. Then, the robot could lean in anticipation of balance disturbances that would be caused by executing the motion command. This implies that the other motions are actuated, controlled and known in advance. In contrast, we consider here a passive springy joint, which is neither actuated nor controlled, and which moves in response to the actions of the balance controller.

This chapter also explores the response of the launching controller subject to uncertain spring parameters. Precision at launching is critical for executing a successful hopping motion.

The rest of this chapter is organized as follows: first, the robot model; then the theory of balancing used; then the balance controller; and finally the simulation experiments, their results and an the controller's response subject to uncertain spring parameters.

4.1 General Setup

We consider two robots in this chapter: one with a springy leg and one with a rigid leg. The springy-leg robot is shown in Fig. 4.1. The rigid-leg robot is obtained from the springy-leg robot by locking joint 2 (the prismatic joint) in the position it takes when the robot is balanced, the torso is at right angles to the leg, and the spring is holding the weight of the upper leg and torso against gravity.

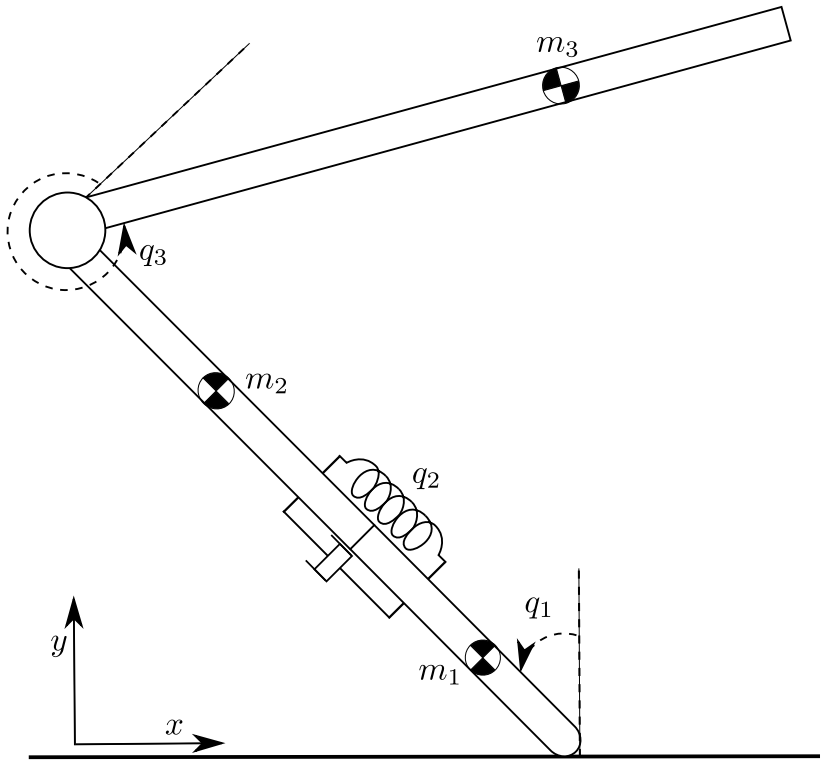

Figure 4.1 Robot model. q_3 is negative in this configuration, and has been drawn as $q_3 + 2\pi$.

The springy-leg robot is similar to the robot introduced in the previous chapter in which links 1, 2, and 3 are the lower leg (or foot), upper leg, and torso, respectively. Joint 1 is

Link (i)	Mass (kg)	Length (m)	CoM (m)	Inertia at CoM (kgm^2)
1	0.2	0.2	0.1	0.001
2	0.3	0.3	0.15	0.003
3	2	0.5	0.33	0.08

Table 4.1 length and inertia parameters of the robot shown in Fig.4.1

.

a passive revolute joint that models the contact between the foot and ground; joint 2 is a spring-loaded passive prismatic joint, and joint 3 is the actuated joint. The links' mass and length parameters are shown in Table 4.1. The symbols m_i, l_i and I_i appearing below denote the mass, length and rotational inertia about the centre of mass (CoM), respectively, of link i.

The joint variables are q_1, q_2 and q_3. When all three are zero, the leg is vertical, the leg's length is $l_1 + l_2$, and the torso is horizontal out to the right. Positive motion of a revolute joint i rotates link i counter-clockwise relative to link $i-1$; and positive motion of joint 2 extends the leg (so the actual length of the leg is $l_1 + l_2 + q_2$). In Fig. 4.1, q_1 is positive and q_3 is negative.

The stiffness of the spring is 2000 N/m, and the damping coefficient is 15 Ns/m, which results in an under-damped system. The stiffness is appropriate for a robot able to make small hops of around 1 m with a spring compression significantly less than l_1, yet is soft enough that joint 2 moves significantly during landings and fast balancing movements. In other words, the stiffness is low enough to interfere significantly with the actions of the balance controller.

4.2 Balance Theory

In this section, the analysis presented in Featherstone (2017) is modified for the planar robot mechanism shown in Fig. 4.1. As previously commented, The upper joint (joint 3) is actuated by the controller, the middle joint (joint 2) is actuated by a spring, and the lower joint (joint 1) represents the contact between the bottom of the lower link (the foot) and the ground and is consequently un-actuated. By representing the contact with a revolute joint, the controller assumes that the foot neither slips nor loses contact with the ground and that the movement

of the contact point as the foot rotates is negligible. The equation of motion of this robot is

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{22} & H_{23} \\ H_{13} & H_{23} & H_{33} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 0 \\ F_s \\ \tau_3 \end{bmatrix} \tag{4.1}$$

where H_{ij} are elements of the joint-space inertia matrix, $\ddot{q}_1$, $\ddot{q}_2$ and $\ddot{q}_3$ are the joint acceleration variables, C_1, C_2 and C_3 contain gravity and velocity terms, $F_s = -K_s q_2 - D_s \dot{q}_2$, K_s and D_s denote the stiffness and damping coefficients of the spring, and τ_3 is the torque at the actuated joint.

As joint one is un-actuated, it follows that the force of gravity is the only force capable of exerting a moment about the support point and change the angular momentum of the robot about this point. If we define L to be the angular momentum of the whole robot about the support point then we have

$$\dot{L} = -m\,g\,c_x \tag{4.2}$$

where m is the total mass of the robot, g is the acceleration due to gravity (a positive number), and c_x is the x coordinate of the robot's center of mass (CoM). The expression $-m\,g\,c_x$ is the moment of gravity about the support. The equation that follows directly from (4.2) is

$$\ddot{L} = -m\,g\,\dot{c}_x \tag{4.3}$$

and the expression for L is

$$L = H_{11}\,\dot{q}_1 + H_{12}\,\dot{q}_2 + H_{13}\,\dot{q}_3 \tag{4.4}$$

which is proved in the appendix of Featherstone (2017).

The next step is to add a fictitious joint between joint one and the ground, which is a prismatic joint in the x-direction. We can call it joint zero so as not to disturb the numbering of the other joints. This joint never moves, so it does not affect the robot's dynamics. However, it does increase the size of the equation of motion, which now reads

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} & H_{03} \\ H_{01} & H_{11} & H_{12} & H_{13} \\ H_{02} & H_{12} & H_{22} & H_{23} \\ H_{03} & H_{13} & H_{23} & H_{33} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ F_s \\ \tau_3 \end{bmatrix} \tag{4.5}$$

The reason for this extra joint is that it provides us with two equations linking the joint-space dynamics with the motion of the CoM:

$$m\dot{c}_x = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 + H_{03}\dot{q}_3, \tag{4.6}$$

which is proved in Featherstone (2017), and

$$\tau_0 = m\ddot{c}_x = -\dddot{L}/g \tag{4.7}$$

Actuated Joint Tracking

Let q_a denote the controlled variable. To control the motion of the actuated joint we set $q_\mathrm{a} = q_3$. In this case we combine (4.3), (4.4) and (4.6)

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{13} \\ -gH_{01} & -gH_{03} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_3 \end{bmatrix} + \begin{bmatrix} H_{12} \\ -gH_{02} \end{bmatrix} \dot{q}_2. \tag{4.8}$$

We omit the last term (the one involving $\dot{q}_2$) in order to make point that the resulting control system does not need to consider the dynamics introduced by the spring. The simplified equation can then be solved to give

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_\mathrm{a} \end{bmatrix} = \frac{1}{gD} \begin{bmatrix} -gH_{03} & -H_{13} \\ gH_{01} & H_{11} \end{bmatrix} \begin{bmatrix} L \\ \ddot{L} \end{bmatrix} \tag{4.9}$$

where

$$D = H_{01}H_{13} - H_{11}H_{03} \tag{4.10}$$

assuming that the matrix is invertible (which it will be if the robot is physically capable of balancing (Featherstone, 2017)). Consequently $\dot{q}_\mathrm{a}$ can be expressed as

$$\dot{q}_\mathrm{a} = Y_1 L + Y_2 \ddot{L} \tag{4.11}$$

where

$$Y_1 = \frac{H_{01}}{D}, \qquad Y_2 = \frac{H_{11}}{gD} \tag{4.12}$$

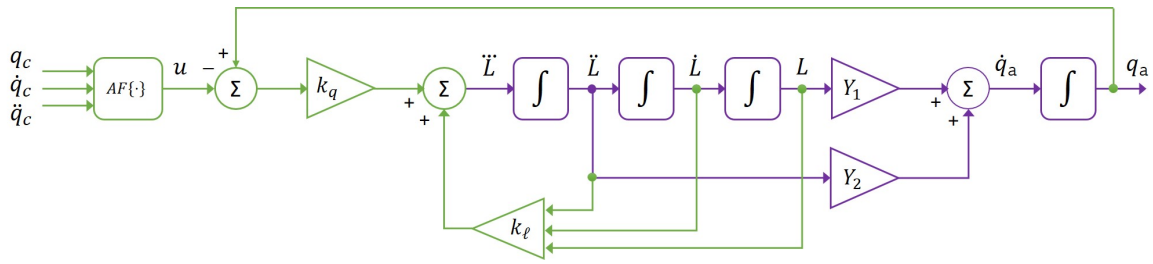

Figure 4.2 Block diagram for the balancing control problem of the springy-leg robot, with $k_\ell = \text{diag}(k_{dd}, k_d, k_L)$. The green lines indicate the balance controller, and the purple lines denote the plant. q_a denotes the controlled joint variable, which can be q_3 or $q_1 + q_3$

Absolute Orientation Tracking

To control the absolute angular position we redefine q_a as $q_a = q_1 + q_3$ and use the following mapping

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_a \end{bmatrix}, \tag{4.13}$$

and the linear relationship (4.8) can be solved as:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_a \end{bmatrix} = \frac{1}{gD} \begin{bmatrix} -gH_{03} & -H_{13} \\ g(H_{01} - H_{03}) & H_{11} - H_{13} \end{bmatrix} \begin{bmatrix} L \\ \ddot{L} \end{bmatrix}, \tag{4.14}$$

where $D = H_{13}H_{01} - H_{11}H_{03}$. This equation requires $D \neq 0$, which holds in any configuration in which the robot is physically capable of balancing itself Featherstone (2017). So, $\dot{q}_a$ can be expressed as:

$$\dot{q}_a = Y_1 L + Y_2 \ddot{L}, \tag{4.15}$$

where

$$Y_1 = (H_{01} - H_{03})/D, \qquad Y_2 = (H_{11} - H_{13})/gD. \tag{4.16}$$

It is worth noticing that Y_1 and Y_2 vary with configuration, and can be expressed as simple functions of two physical properties of the mechanism: its time constant of toppling, T_c, which measures how quickly the robot falls if the controller does nothing, and its velocity gain (Featherstone, 2015, 2016), which measures the effect on centre of mass (CoM) velocity of a unit change in the velocity of the actuated joint. The formulae are

$$Y_1 = \frac{1}{mg\,T_c^2\,G_v} \qquad Y_2 = -\frac{1}{mg\,G_v} \tag{4.17}$$

where G_v is the linear velocity gain as defined in Featherstone (2016). T_c appears again in the acausal filter mentioned in the next section.

4.3 Controller

The balance controller works by controlling the plant shown in Fig. 4.2, as explained in Featherstone (2017, 2018). The job of the balance controller is to calculate a value for $\dddot{L}$ to make q_a follow a given command signal, q_c, without losing balance. Denoting $q_\mathrm{a} := q_3$, a suitable control law to accomplish this is

$$\dddot{L} := k_{dd}\ddot{L} + k_d\dot{L} + k_L L + k_q(q_\mathrm{a} - q_d)\,, \tag{4.18}$$

where q_d is the input to the control law (see below). The feedback gains are obtained via pole placement as

$$\begin{aligned} k_{dd} &= -a_3 \qquad & k_d &= -a_2 + a_0 Y_2/Y_1 \\ k_L &= -a_1 \qquad & k_q &= -a_0/Y_1\,, \end{aligned} \tag{4.19}$$

where

$$\begin{aligned} a_0 &= \lambda_1\lambda_2\lambda_3\lambda_4 \\ a_1 &= -\lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_2\lambda_4 - \lambda_1\lambda_3\lambda_4 - \lambda_2\lambda_3\lambda_4 \\ a_2 &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 + \lambda_2\lambda_4 + \lambda_3\lambda_4 \\ a_3 &= -\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4 \end{aligned} \tag{4.20}$$

and $\lambda_1, \ldots, \lambda_4$ are the chosen values of the poles. We set λ_1 to the closed-loop bandwidth that we want the controller to achieve, while λ_2 and λ_3 are destined to be cancelled by two introduced zeros, mentioned below, and λ_4 is set to $-1/T_\mathrm{c}$ in order to cancel a natural zero in the transfer function. The use of pole placement involves an assumption that Y_1 and Y_2 are constants, implying that the plant is linear. This assumption is justified by the observation that in practice Y_1 and Y_2 vary slowly with configuration.

The input to the control law, q_d, is computed from the command signal and its derivatives, q_c, $\dot{q}_\mathrm{c}$ and $\ddot{q}_\mathrm{c}$, according to

$$q_d = \mathrm{AF}\left(q_\mathrm{c} - \left(\frac{1}{\lambda_2} + \frac{1}{\lambda_3}\right)\dot{q}_\mathrm{c} + \frac{1}{\lambda_2\lambda_3}\ddot{q}_\mathrm{c}\right). \tag{4.21}$$

This formula has two effects. First, it introduces two zeros into the transfer function, at λ_2 and λ_3, which cancel the corresponding poles. Second, it applies an acausal filter, AF,

which makes the robot lean in anticipation of the balance disturbances that will be caused by future commanded motions (Featherstone, 2017). Specifically, AF is a first-order low-pass filter with time constant T_c, which runs backwards in time from a point sufficiently far in the future back to the present. To implement this filter, the controller needs to know the expected short-term future value of q_c. Information of this kind can be found in the robot's high-level controller, which usually knows what movement it intends to make next.

Given the control law in (4.18), with gains as in (4.19) and (4.20), and the input signal as in (4.21), it can be shown that the complete transfer function from q_c to q_a would be

$$q_a(s) = \frac{1}{1+s/(-\lambda_1)}\, q_c(s) \tag{4.22}$$

if it were really true that Y_1 and Y_2 were constants (Featherstone, 2017). We take this expression as the theoretical transfer function of the balance controller, and compare the actual response with the theoretical one in the experimental results reported below.

Finally, the output of the control law ($\dddot{L}$) must be converted to a torque or an acceleration at the actuated joint, which can be done by solving the equations (4.5) and (4.7),

$$\begin{bmatrix} 0 & H_{01} & H_{02} & H_{0a} \\ 0 & H_{11} & H_{12} & H_{1a} \\ 0 & H_{12} & H_{22} & H_{2a} \\ -1 & H_{1a} & H_{2a} & H_{aa} \end{bmatrix} \begin{bmatrix} \tau_a \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_a \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 \\ -C_1 \\ F_s - C_2 \\ -C_a \end{bmatrix}. \tag{4.23}$$

4.4 Experiments

This section presents the experimental results for actuated joint and absolute tracking tasks; and launching study with uncertain spring parameters. In the experiments we use the ode23t integrator from MATLAB with a relative tolerance set to 10^{-6} and other parameters at their default values. The controller is implemented as a continuous-dynamics subsystem, and the sensors are assumed to be perfect. Although the balance controller employed on the experiments assumes that the foot never loses contact nor slips with the ground, in the simulation we incorporate the following contact model described in Azad and Featherstone (2013). Contact forces acting on the foot in the normal and tangent directions are

$$\begin{aligned} F_y &= \max(0, K_n z^{3/2} + D_n z^{1/2}\dot{z}), \\ F_x &= \mathrm{clip}(K_t z^{1/2} u + D_t z^{1/2}\dot{u}, -\mu F_y, \mu F_y), \end{aligned} \tag{4.24}$$

where z and u are the ground compression and shear deformation, and μ is the coefficient of friction, K_n and D_n are the normal and K_t and D_t are the tangential stiffness and damping coefficients. The function $\text{clip}(a,b,c)$ returns the value of a clipped to b and c. The parameter values used in the simulations are

$$\begin{aligned} K_t = 12.7 \times 10^6 \quad D_t = 3.1 \times 10^5 \quad \mu = 1 \\ K_n = 8.5 \times 10^6 \quad D_n = 3.1 \times 10^5 \end{aligned} \tag{4.25}$$

which are consistent with a hard floor and a high-friction hard rubber foot.

The results in both scenarios are displayed in figures 4.7 and 4.9, where the signals are: the original command signal, q_c (i.e., not q_d as in (4.21)); the theoretical response of the balance controller if the plant really were linear (labelled q_t); the actual response of the controlled joint on the springy leg robot (labelled $q_a(springy)$); and the actual response on the rigid-leg robot (labelled $q_a(rigid)$). The rigid-leg response is obtained from a separate simulation in which the initial conditions are set as close as possible to the state of the springy-leg robot at the beginning of the sequence.

4.4.1 Actuated Joint Tracking

This subsection presents the results of a simulation experiment in which the springy-leg robot starts in an upright position, tips itself forward (positive x direction), crouches, launches into a hop of 0.7 m length and 0.9920 m height (rise of the CoM from lift-off to apex), lands with two bounces, stabilizes itself in a balanced configuration with $q_a = 0$, and then executes a motion command consisting of a sequence of ramps and sinusoids.

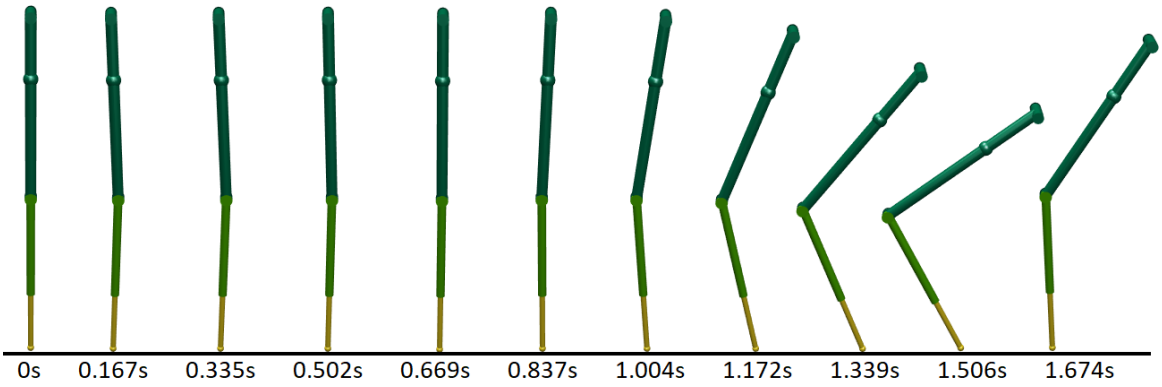

Figure 4.3 Motion profile of the launching phase.

Figures 4.3, 4.4 and 4.5 show the complete motion. The launching and the flight motion profiles were obtained using the DOC method presented in chapter two with three collocation

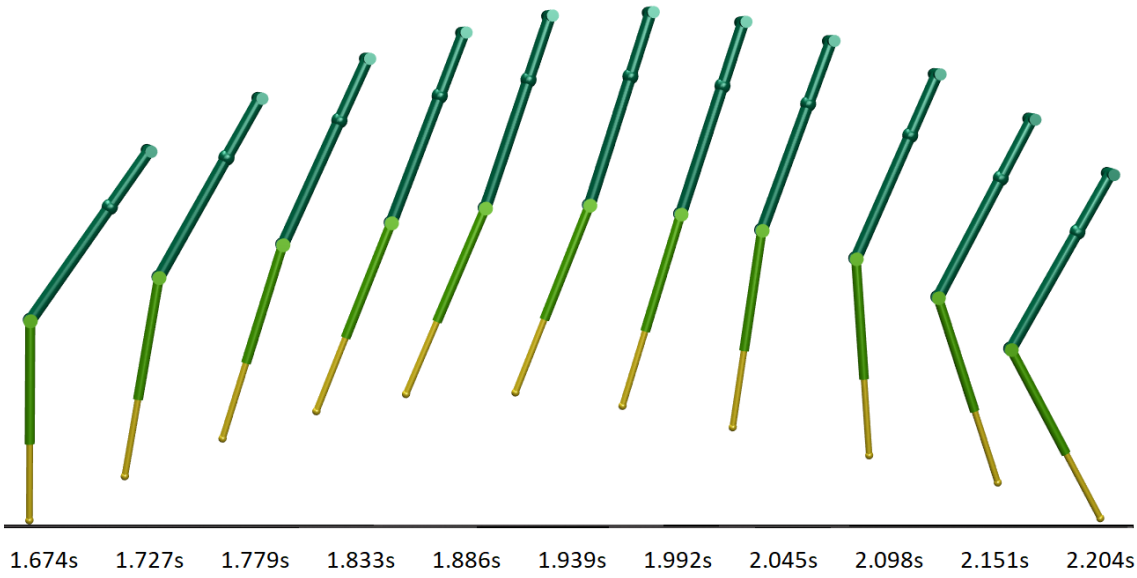

Figure 4.4 Motion profile of the flight phase.

points and discretized in 300 and 150 steps, respectively. The landing motion is obtained using the balance controller to bring the robot to balance, and at $t = 2.554\,\mathrm{s}$ the commanded angle q_d is switched to drive the robot to the ready configuration to start the tracking phase.

During the landing, the evolution of the controller's state variables is plotted in Fig. 4.6. At this phase, the poles of the controller are set to $-20,-7,-7,-1/T_c$, all in units of rad/s, and the controller is first set to recover the balance of the robot during the first 0.35 s of the landing, resulting in two bounces after the first touchdown, which demonstrates the controller's response to situations when it's assumption about the ground foot contact is not satisfied for short periods (bounces). In the graph, it can be seen that q_3 is initially pushed to about 0.39 rad by the momentum of the landing, then rises to nearly 1.37 rad during the first bounce (blue shaded area). The joint is driven close to zero after the third touchdown (end of magenta shaded area).

The robot's foot loses contact with the ground between $t = 2.3364\,\mathrm{s}$ and $t = 2.6224\,\mathrm{s}$, and $t = 2.7466\,\mathrm{s}$ and $t = 2.9198\,\mathrm{s}$ and reaches a maximum height of 9.2286 cm at the first bounce and 3.16516 cm at the second bounce. These periods are shown shaded in the graph, blue for the first bounce and magenta for the second bounce. During these periods, the balance controller does not know that the foot has left the ground and continues to use the model described in section 4.2, which assumes that the robot's foot is on the ground. It can be seen that this short period of flight does not significantly affect the controller. It can also be seen that the robot has come to rest within about 0.48 s after the third touchdown. Note that the

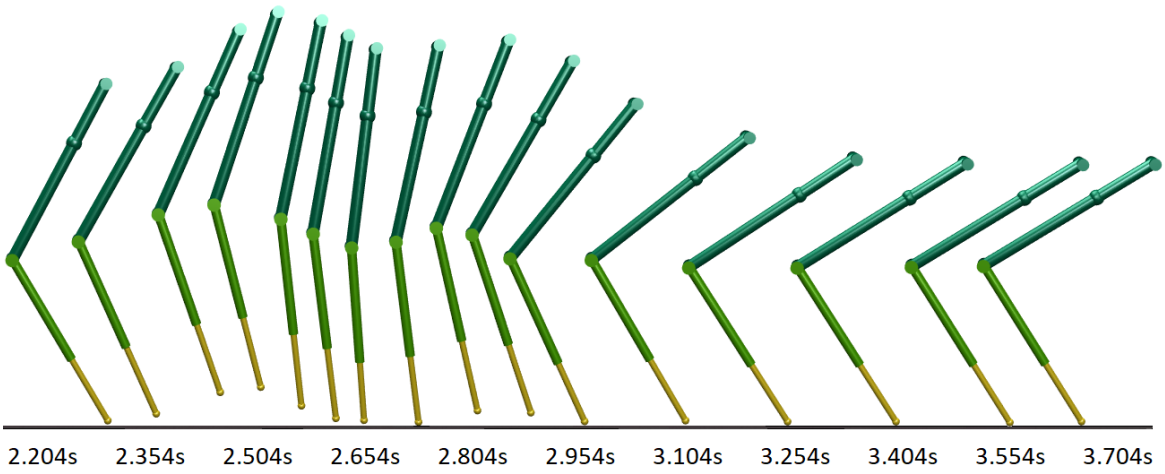

Figure 4.5 Motion profile of the landing phase, when $q_a = q_3$.

controller is assumed to know the orientation of the robot at all times, e.g. from an onboard inertial measurement unit, so it always knows the correct values of q_1 and $\dot{q}_1$.

At time 3.7037 s the command signal (q_c) switches from zero to the sequence of ramps and sine waves mentioned above. This portion of the action sequence is plotted in Fig. 4.7. The command sequence consists of three ramps between 0 and 1 rad, followed by a long ramp from 1 rad to -1 rad, all at a speed of 4 rad/s, followed by two more ramps at speeds of 2 rad/s and 1 rad/s, followed by two cycles of a sine wave at 1 Hz. Observe that this command sequence asks the robot to make large and fast movements.

During this tracking phase, the poles are set at -20, -20, -20 and $-1/T_c$, all in units of rad/s, and the two introduced zeros are at -20 rad/s. For this robot, T_c varies between 0.2112 s at $q_a = -1.0087$ rad and 0.287 s at $q_a = 1.1343$ rad in this action sequence.

Fig.4.8 shows the motion of the spring during the launch, flight (red shaded area), landing, two bounces and tracking phases. During the take-off, the spring compresses to a maximum of 61.22 mm. Then, it compresses to a maximum of 72.78 mm, 42.48 mm, and 27.27 mm at the first, second and third touchdowns after the 0.7 m leap. During the tracking, the spring compression varies in a 2 cm range, which is 4% of the leg length.

It can be seen in figure 4.7 that there are some significant tracking errors, particularly at the beginnings and ends of the ramps. There are two main contributors to these errors. The greater one is that the balance controller assumes that the plant in Fig. 4.2 is linear, when in reality it is not. The lesser one is an approximation in the way that the acausal filter works: for simplicity, it uses values of T_c calculated for a balanced configuration at each instant instead of a leaning configuration.

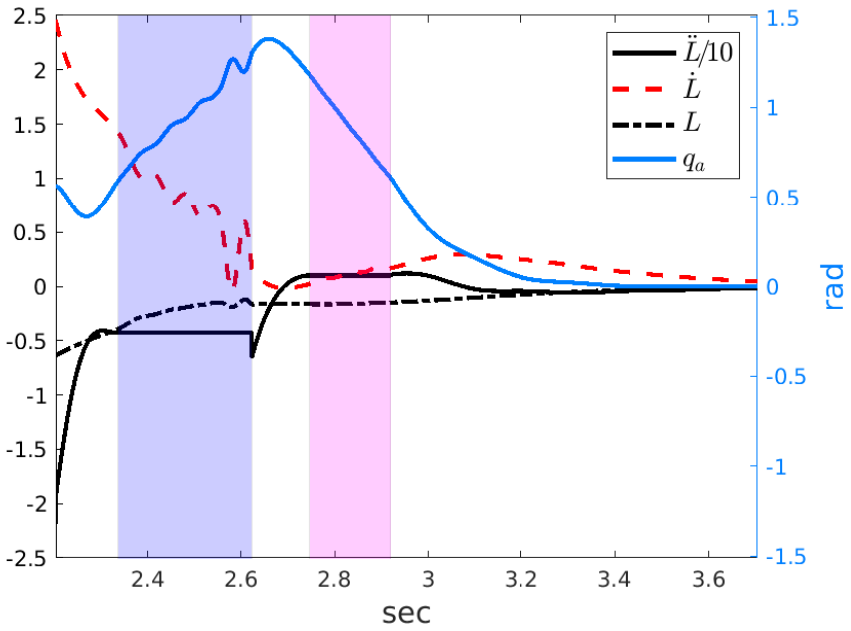

Figure 4.6 Evolution of the controller's state variables from the moment of landing until the robot has settled. The left side scale corresponds to $\ddot{L}$, $\dot{L}$ and L, and the right side scale corresponds to $q_\mathrm{a} = q_3$. The shaded zones show the periods in which the foot has lost contact with the ground because of the bounces.

It can also be seen that the springy-leg and rigid-leg responses are very nearly the same. So, the presence of the spring-loaded joint in the leg has had almost no effect on the closed-loop behavior of the robot while following large, fast motion commands; and this has happened even though the spring is relatively soft and the spring-loaded joint makes significant movements. One place where one can see a difference occurs at the end of the second ramp, where the springy-leg response shows a small amount of ringing that dies away in about 0.5 s. There is also a little bit of ringing at the end of the long ramp, and at the end of the sine wave. However, considering the under-damped nature of the spring-damper pair in the leg, there is remarkably little ringing overall, and what little there is dies away quickly. So we can conclude that in this experiment the balance controller has accomplished three tasks simultaneously using only a single actuator: balance the robot, follow the command signal, and suppress vibrations.

4.4.2 Absolute Tracking

This subsection presents the results of a simulation experiment in which the springy-leg robot starts on a balance position with $q_\mathrm{a} = q_1 + q_3 = 0$ with zero velocity and then executes an

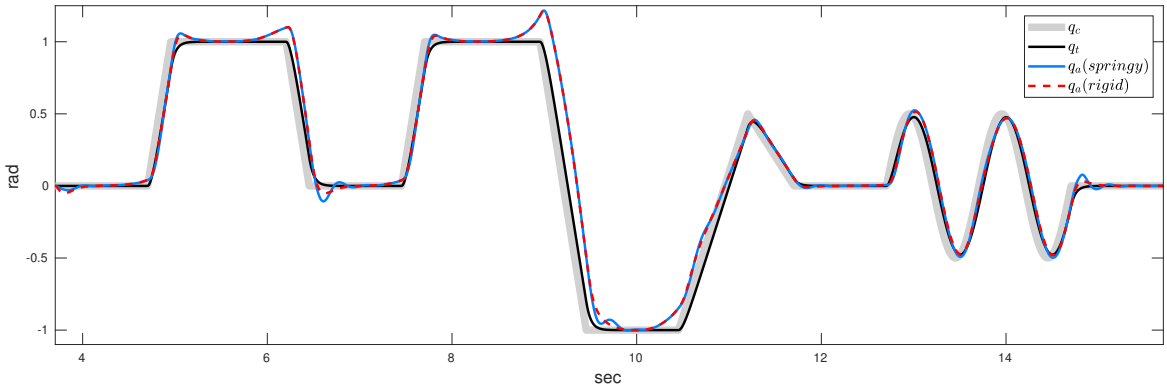

Figure 4.7 Tracking performance of the balance controller, when $q_a = q_3$.

absolute motion tracking consisting of a sequence of ramps and sinusoids plotted in Fig. 4.9. The poles of the controller were set to $\lambda_1 = -20$, $\lambda_2 = -20$, $\lambda_3 = -20$, and $\lambda_4 = -1/T_c$, all expressed $\mathrm{rad\,s^{-1}}$, and the two introduced zeros are at $-20\,\mathrm{rad/s}$.

The desired sequence consists of three ramps between 0 and 0.8 rad, all at a speed of $3\,\mathrm{rad\,s^{-1}}$, followed by a long ramp from 0.8 rad to -0.6 rad at $-2.5\,\mathrm{rads^{-1}}$, followed by two more ramps at speeds of $2.5\,\mathrm{rads^{-1}}$ and $-1\,\mathrm{rads^{-1}}$, followed by three cycles of a sine wave at 1 Hz. Notice that, such a desired sequence asks the robot to make large and fast motions. Two zeros are introduced to the desired signal at $-20\,\mathrm{rad\,s^{-1}}$, as mentioned in (4.21). For this springy-leg robot, T_c varies between 0.1562 s at $q_a = -0.6$ rad and 0.2507 s at $q_a = 0.8$ rad in this action sequence.

In Fig. 4.9, it can be seen that there are some significant tracking errors, particularly at the beginning and end of the ramps. Three main factors contribute to such errors: (i) the balance controller assumes that the plant in Fig. 4.2 is linear when in reality it is not; (ii) for simplicity, the acausal filter uses values of T_c calculated for a balanced configuration at each instant instead of a leaning configuration; (iii) at each of the balance configurations used to calculate T_c, it assumes that q_2 is constant for the whole tracking when in fact it oscillates around 10% and reaches a peak of 17% of the link's length, causing tracking inaccuracies as shown in Figures 4.9. Then, in this experimental case, we can conclude that the balance controller has accomplished balance while performing an absolute tracking task. However, it can be seen that the controller is less efficient at damping oscillations compared with the results in Section 4.4.1 (results in figure 4.10 are less damped than results in figure 4.8).

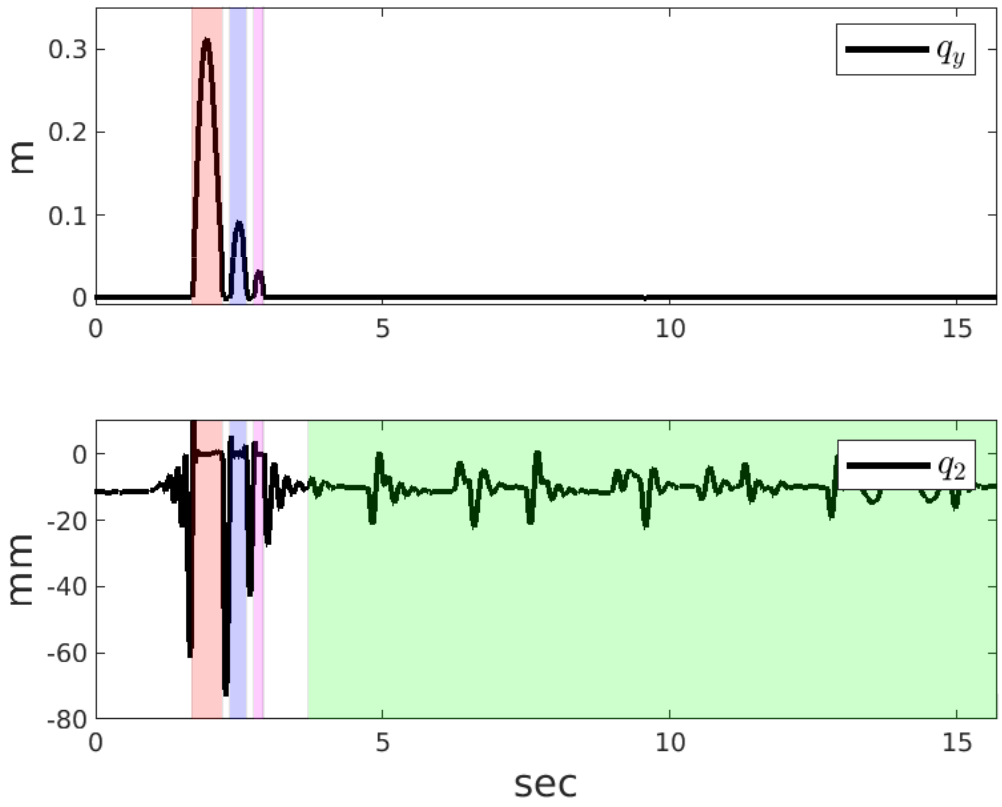

Figure 4.8 Motion of the spring during the whole motion. The shaded zones of red, blue, and magenta colors show the instant in which the foot has lost contact with the ground because of the leap and two bounces, respectively. The green-shaded zone denotes the moment when the robot is performing the tracking sequence of Fig. 4.7. The quantity q_y is the height of the foot above the ground.

4.4.3 Launching with an Uncertain Spring

This section presents a sensitivity analysis at the launch controller when the robot (model described in section 4.1) is subject to uncertainties in the spring parameters. In this sense, this section starts by obtaining a launching movement by formulating and solving it as an NLP problem. Then, the motion is tracked using the launch controller presented in section 2.4 adapted to the spring-loaded monopod. In this sense, the controller output is obtained by:

$$\dddot{L} := -k_{dd}(\ddot{L} - \ddot{L}_c) + \dddot{L}_c \tag{4.26}$$

where $\ddot{L}_c$ denotes the desired signal to be tracked, and as the launching motion is known, it is possible to obtain $\dddot{L}_c$ by differentiating $\ddot{L}_c$. The gain k_{dd} is obtained as described in equation

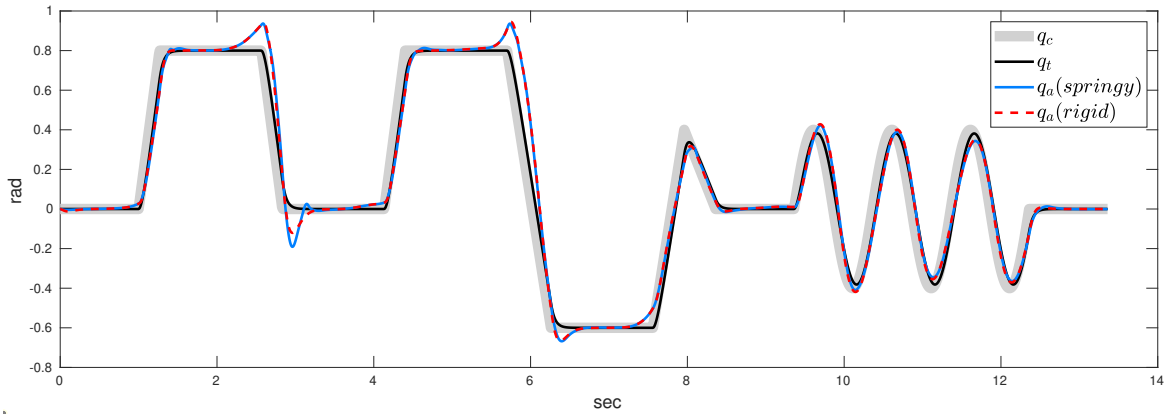

Figure 4.9 Tracking performance of the balance controller, with $q_{\mathrm{a}} = q_1 + q_3$.

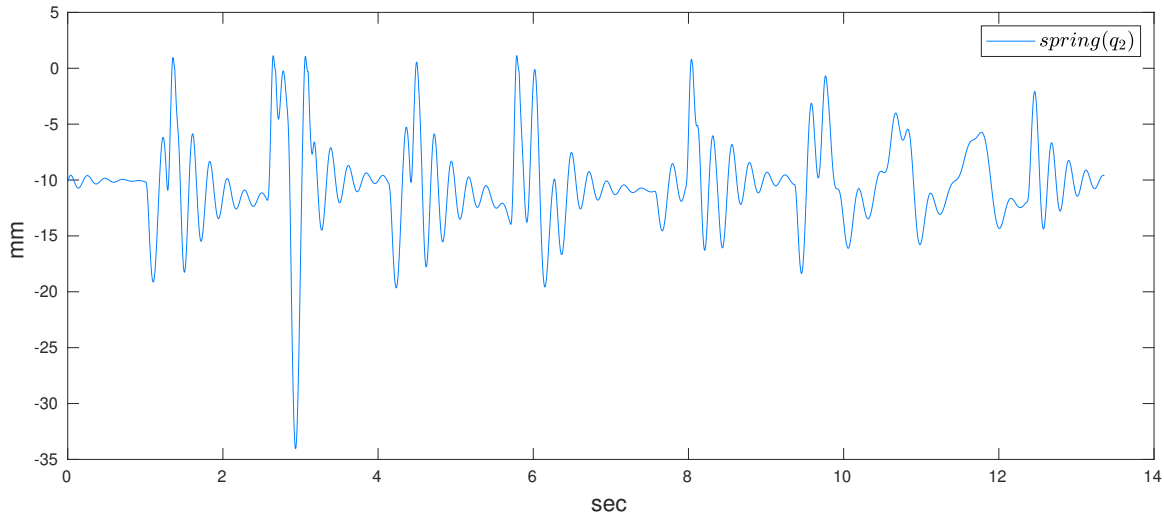

Figure 4.10 Motion of the spring during tracking the motion at figure 4.9.

(4.19). Finally, the controller's output $\dddot{L}$ is mapped to acceleration or torque at the actuated joint q_3 by:

$$\begin{bmatrix} 0 & H_{01} & H_{02} & H_{03} \\ 0 & H_{11} & H_{12} & H_{13} \\ 0 & H_{21} & H_{22} & H_{23} \\ -1 & H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} \tau_3 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 \\ -C_1 \\ \hat{F}_s - C_2 \\ -C_3 \end{bmatrix}, \tag{4.27}$$

where $\hat{F}_s$ is the estimated spring's force. In this experiment, $\hat{F}_s$ is modified to emulate uncertain spring parameters.

NLP Problem Formulation

The launching motion is implemented as an NLP problem, where the constrained parameter optimization problem has nonlinear terms in its objective or constraints functions. In this sense, it is necessary to define the problem constraints and the decision variables to control during the optimization. This launching NLP problem implementation is done using Matlab with the software package CasADI (Andersson et al., 2018) and the solver Interior Point OPTimizer (IPOPT) (Wächter and Biegler, 2006) to minimize a given cost function J by controlling the decision variables and satisfying the imposed constraints.

The problem is adequately described when discretized in $N_T = 600$ steps, implying a maximum sampling time $T_s = 0.0033\,\text{s}$. The NLP problem is formulated in initial constraints to describe the starting state and conditions of the task to solve, loop constraints containing a set of conditions evaluated every iteration of the optimization, and terminal constraints specifying the final requirements to achieve at the end of the task.

$$\min_{\tau_3} \quad J = \int_0^T w_r \, \tau_3(t)^2 \, dt \tag{4.28}$$

subject to:

$$0.5\,\mathrm{s} \leq T \leq 2\,\mathrm{s}, \tag{4.29}$$

initial constraints

$$c_x = 0, \quad F_s = C_2, \quad \dot{q}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \tag{4.30}$$

loop constraints

$$\begin{bmatrix} -\pi/2 & -0.2 & -\pi/3 \end{bmatrix}^\top \leq q(k) \leq \begin{bmatrix} \pi/2 & 0 & \pi/3 \end{bmatrix}^\top, \quad |\dot{q}(k)| \leq \infty, \tag{4.31}$$

$$|\tau_2(t)| \leq 50\,\mathrm{Nm}, \tag{4.32}$$

$$\dot{q}(k+1) = \dot{q}(k) + \int_{t_k}^{t_{k+1}} f(q, \dot{q}, \tau_2) dt, \qquad q(k+1) = q(k) + \int_{t_k}^{t_{k+1}} \dot{q} dt, \tag{4.33}$$

terminal constraints

$$\left| \begin{bmatrix} L^c & \dot{c}_x & \dot{c}_y \end{bmatrix}^\top - \begin{bmatrix} -1.189 & 0.9988 & 2.4224 \end{bmatrix}^\top \right| \leq 1 \times 10^{-3}. \tag{4.34}$$

where the objective function (4.28) has only a Lagrange term (Kelly, 2017).

At the beginning of the optimization, the motion duration T is added as a control variable and bounded in (4.29). Then, (4.30) defines the initial states of the system, where the robot starts in a balanced configuration with zero velocity.

In the loop constraints, we define the operating regions for the positions and velocities for the joints and the actuation torque. The operating regions are described in equation (4.31) in rad and rad/s, respectively. The actuation torque is limited in equation (4.32). An extra equation inserts the system's dynamics expressed as a constraint in (4.33) to ensure the system's continuity during the optimization. This continuity is ensure using the DOC-LGR method with three collocation points. The function $f(q, \dot{q}, \tau_3)$ calculates the acceleration $\ddot{q}$, as a function of the state variables and τ_3.

Finally, the terminal constraint in equation (4.34) specifies the desired launching conditions that need to be accomplished at the end of the motion to take-off. At the equation, ${}^cL$ denotes the centroidal angular momentum of the whole robot, which is the system's angular momentum obtained in equation (4.4) transformed to the CoM's frame.

Results

Several simulations were performed in which the robot was controlled according to the control law in equation (4.26) with $\dddot{L}$ converted to a torque (τ_3) using equation (4.27). In each simulation, $\hat{F}_s$ was taken to be equal to γF_s, where F_s is the correct value and γ is a constant taking values in the range 0 to 1.7. In each simulation, the robot starts in a balanced configuration with $q = [0.6475, -0.009, -0.6075]$, and the control system executes the command signal ($\ddot{L}_c$ and $\dddot{L}_c$) found by the optimization process. If the control system manages to follow the optimal trajectory perfectly, then its configuration at lift-off should be $q = [0.0275, -0.0118, 0.9568]$ in 0.391 s at time 0.391 s. The simulation finishes at the instant when the foot losses contact with the ground regardless of whether it takes longer or shorter time than the optimized motion. In case that it takes longer, the actuation torque is set to $\tau_3 = C_3$.

The average error for each scenario is calculated by:

$$Av.Error = \left(\left| \frac{{}^cL_d - {}^cL}{|{}^cL_d|} \right| + \left| \frac{\dot{c}_{dx} - \dot{c}_x}{|\dot{c}_{dx}|} \right| + \left| \frac{\dot{c}_{dy} - \dot{c}_y}{|\dot{c}_{dy}|} \right| \right) \times \frac{100\%}{3} \tag{4.35}$$

where ${}^cL_d, \dot{c}_{dx}$ and $\dot{c}_{dy}$ denote the lift-off desired parameters obtained from the NLP optimization. Table 4.2 shows the launching states results for different uncertain spring parameters. The results show that the controller presents a certain robustness to incorrect estimates of F_s. According to the table, the controller accomplished the launching with nearly 11% of error by assuming the total absence of the spring force $\hat{F}_s = 0$. The results also demonstrated that the controller presents more robustness when the estimated spring force ($\hat{F}_s$) is lower than the true spring force in the robot (F_s). Once the robot is flying, the control system loses control over the CoM trajectory and centroidal angular momentum, so errors in these quantities cannot be corrected. Therefore, achieving the desired launching states with precision is critical for ensuring the entire motion's success.

4.5 Conclusion

This chapter investigated the effect of a springy leg on the balance controller described in Featherstone (2017, 2018), which aims to achieve and maintain balance in the presence of large, fast motions. The purpose of this study was to investigate the feasibility of a robot that uses a spring in its leg to help it hop efficiently, but which also needs to balance on that leg. It was shown that the balance controller can cope with the large, fast motions that occur

	${}^{c}L\,(\mathrm{kg\,m^2/s})$	$\dot{c}_x\,(\mathrm{m/s})$	$\dot{c}_y\,(\mathrm{m/s})$	Av.Error	Task Duration (s)
NLP Solution	-1.1808	0.9918	2.4131	-	0.391
$\hat{F}_s = 0$	-0.9435	0.8715	2.4589	11.3753%	0.3968
$\hat{F}_s = 0.1F_s$	-0.9741	0.8908	2.4664	9.9641%	0.396
$\hat{F}_s = 0.2F_s$	-1.003	0.9086	2.4719	8.6259%	0.3952
$\hat{F}_s = 0.3F_s$	-1.0302	0.9247	2.4751	7.3603%	0.3945
$\hat{F}_s = 0.4F_s$	-1.00556	0.9393	2.4761	6.1693%	0.3938
$\hat{F}_s = 0.5F_s$	-1.0793	0.9521	2.4746	5.0461%	0.3931
$\hat{F}_s = 0.6F_s$	-1.1014	0.9634	2.4704	3.956%	0.3925
$\hat{F}_s = 0.7F_s$	-1.1221	0.9732	2.4631	2.9723%	0.3919
$\hat{F}_s = 0.8F_s$	-1.1416	0.9814	2.4524	1.9964%	0.3914
$\hat{F}_s = 0.9F_s$	-1.1649	0.9907	2.4389	0.8398%	0.3911
$\hat{F}_s = F_s$	-1.1893	0.9988	2.4219	0.5978%	0.391
$\hat{F}_s = 1.1F_s$	-1.2129	1.0047	2.4001	1.5238%	0.3911
$\hat{F}_s = 1.2F_s$	-1.2376	1.0088	2.3695	2.7808%	0.3914
$\hat{F}_s = 1.3F_s$	-1.2641	1.0116	2.3284	4.1891%	0.3918
$\hat{F}_s = 1.4F_s$	-1.2935	1.0138	2.2750	5.8312%	0.392
$\hat{F}_s = 1.5F_s$	-1.327	1.0158	2.2074	7.7768%	0.3921
$\hat{F}_s = 1.6F_s$	-1.3657	1.017	2.1228	10.0785%	0.3922
$\hat{F}_s = 1.7F_s$	-0.1931	0.1813	0.7481	78.1211%	0.1419

Table 4.2 Launching sensitivity related to different spring parameters.

during landing, including small bounces, and that it can track large, fast motion commands after landing with almost the same accuracy as could be achieved if the leg were rigid. And all that is necessary in order to achieve this performance is that the spring-loaded joint is instrumented, and that the measured motion of the spring is taken into account when mapping from the robot's state variables to the balance controller's state variables. This study also showed the possibility of performing an absolute motion tracking, not only controlling the angle of the actuated joint q_3 but also the angle of the passive joint q_1 by introducing a state mapping for the balance controller given as (4.13). Moreover, we also demonstrated that the varying-linear plant constructed by the controller does not need to know about the dynamics introduced by the spring, although these dynamics are taken into account when calculating the system's states and mapping the controller's output to a joint torque or acceleration command. It also showed that the balance controller could suppress vibrations caused by the spring.

This chapter also explored the response of the launching controller subject to uncertain spring parameters. The results demonstrated the controller's robustness by having a launching error below 11% for different scenarios. Although the obtained error is small, it is too big for executing a precise hopping motion.

The next chapter will present a parameter identification approach to achieve a proper landing and tracking when the robot suffers a spring rupture and loses 50% of its spring force. The strategy demonstrates the feasible application of high-order sliding mode observers to balancing applications.

Chapter 5

Non-linear Observers for balancing

The balance controller presented in Azad and Featherstone (2016) can drive a monopod robot through fast motions without compensating for the system dynamics and achieves a reasonable error when tracking a reference trajectory. This performance can be improved by considering the system dynamics when obtaining the actuated torque (Featherstone, 2017). However, the presence of parametric uncertainties degrades the system's efficiency and may cause instability. Thus, an observer is useful to estimate any given system uncertainty.

This chapter presents a modified version of the existing balance controller Featherstone (2017) to achieve high performance in balancing and absolute angular position tracking tasks (Singh, 2021) for a spring-loaded monopod in the presence of uncertainties at the spring parameters. A high-order sliding mode (HOSM) observer algorithm is used to estimate the external force generated by a viscoelastic element (spring plus dashpot) located between the robot leg and foot, identifying its elastic and viscous coefficients. The stability analysis of the overall closed-loop system is demonstrated by using the Lyapunov stability theory. Numerical simulations are included to illustrate the performance and feasibility of the proposed control methodology.

Our main objective is to present a solution for estimating the spring parameters online while the robot executes a task without compromising the controller's performance.

5.1 Parameters Identification

The parameter identification theory deals with the problem of efficiently extracting data about the system dynamics from its measurements. Most of these strategies involve mainly least-squares methods, Bayesian analysis, Kalman filter extensions, among others. Finite-time algorithms have also demonstrated their effectiveness by identifying mechanical parameters in combination with a recursive least-squares algorithm where the design of the non-linear injection terms is based on the generalized super-twisting algorithm (Moreno and Osorio, 2012), leading to finite-time convergence (Davila et al., 2006; M'sirdi et al., 2006). In Adetola and Guay (2008) the authors use the super-twisting algorithm and a non-recursive least-square algorithm to identify constant parameters in nonlinear systems (Boubaker and Iriarte, 2017). In general, the finite-time convergence is based on the adaptive control theory, requiring to solve matrix-valued ordinary differential equations and check the invertibility property of a matrix online (Polyakov and Fridman, 2014). This scheme allows the reconstruction of the unknown parameters in finite time provided that a given persistence of excitation (PE) condition holds (Polyakov and Fridman, 2014). A well-known approach used to ensure the PE condition in adaptive controllers is to add a bounded perturbation signal to the set-point or trajectory, or even use it as the reference input, which in contrast may degrade the specified regulation or tracking performance (Adetola and Guay, 2008). Here, the PE condition can be guaranteed by performing balance and trajectory tracking tasks.

5.1.1 Finite Time Algorithm

The lack of knowledge, or even the existence of uncertainties in the system's parameters, is a common obstacle in the design of a model-based controller capable of successfully executing a given task. In this subsection, we introduce the basic structure of a High-order Sliding mode (HOSM) observer.

Let us introduce the following notation:

- $\mathbb{R}$ is the set of real numbers;
- m denotes the number of joints or degrees of freedom presented at the robot;
- $H \in \mathbb{R}^{m\times m}$ and $C \in \mathbb{R}^m$ denote respectively the inertia matrix and the gravity, Coriolis and centrifugal, and friction forces already commented in previous chapters;
- positions, velocities and accelerations of the robot are described by the vectors $q \in \mathbb{R}^m$, $\dot{q} \in \mathbb{R}^m$ and $\ddot{q} \in \mathbb{R}^m$ respectively;

- $\tau \in \mathbb{R}^m$ denotes the torque signal produced by a given controller;
- n denotes the number of uncertain parameters to be estimated;
- $\rho > 0$ denotes a positive constant.
- $I \in \mathbb{R}^{n \times n}$ is an identity matrix.
- for a given matrix Q, the pseudo-inverse and transpose matrices of Q are denoted by $Q^{\dagger}$ and $Q^{\top}$ respectively, the induced norm is given by $\|Q\| = \sqrt{\lambda_{max}(Q^{\top}Q)}$, where λ_{max} is the maximum eigenvalue;
- $\lceil \cdot \rfloor^{\gamma} = |\cdot|^{\gamma} \operatorname{sgn}(\cdot)$ for $\gamma > 0$, where $\operatorname{sgn}(\cdot)$ is the sign function, and $\cdot$ denotes a real number.

Then, the dynamic equation of motion introduced in the previous chapters can be solved for a chosen joint b and written into a form:

$$\ddot{q}_b = f(q, \dot{q}, \tau) + \Gamma(q, \dot{q}, \tau)\theta \,, \tag{5.1}$$

where $f(\cdot,\cdot,\cdot)$ is a nonlinear scalar function, $\Gamma(\cdot,\cdot,\cdot)^{\top} \in \mathbb{R}^n$ is assumed to be a bounded meassurable regressor in norm i.e., $0 < \Gamma_m \leq \|\Gamma(q, \dot{q}, \tau)\| \leq \Gamma_M < \infty$ for $\forall q, \dot{q}, \tau$, where Γ_m and Γ_M are scalar bounds, and it satisfies the persistent excitation (PE) condition, i.e. there exists constants $\rho_1 > 0$ and $t_k > 0$ such that

$$\int_{t}^{(t+t_k)} \Gamma^{\top}(t_k)\Gamma^{\top}(t_k)\, dt_k \geq \rho_1 I \tag{5.2}$$

(Poznyak, 2010); and $\theta \in \mathbb{R}^n$ denotes the uncertain parameters to be identified. In this sense, the HOSM algorithm formulation takes the following form;

$$\ddot{\hat{q}}_b = -k_1\, \phi(e) + \Gamma\hat{\theta} + f(q, \dot{q}, \tau) \tag{5.3}$$

$$\dot{\hat{\theta}} = -k_2 \Gamma^{\top} \alpha(e) \tag{5.4}$$

where $k_1 > 0$ is a positive scalar gain, k_2 is either an $n \times n$ positive definite gain matrix or a scalar, $e = \dot{\hat{q}}_b - \dot{q}_b$ is the parameter estimation error where $\dot{q}_b$ is measurable, $\dot{\hat{q}}_b$ is the observer's estimation (Boubaker and Iriarte, 2017), and $\tilde{\theta} \in \mathbb{R}^n = \hat{\theta} - \theta$ is the parameter identification error to be minimized using the gradient-type adaptation law given by (5.4). Finally, $\phi(e)$ and $\alpha(e)$ are nonlinear functions of the parameter estimation error e to be

defined in what follows:

$$\phi(e) = \lceil e \rfloor^{1/2} + e\,, \tag{5.5}$$

$$\alpha(e) = \frac{1}{2}\lceil e \rfloor^{0} + \frac{3}{2}\lceil e \rfloor^{1/2} + e\,. \tag{5.6}$$

Now, we can state the following theorem about the convergence properties of the finite-time algorithm.

Theorem 1. *Consider the dynamic system described by (5.1) and assume that the regressor vector $\Gamma_l(q,\dot{q})$ is measurable and satisfies the PE condition. Then, the finite-time parameter identification algorithm (5.3)-(5.4) with (5.5)-(5.6) ensures the uniformly finite-time stability property of the system errors, and the following properties hold:*

- $\lim_{t\to\infty} e(t) = 0$;
- $\lim_{t\to\infty} \tilde{\theta}(t) = 0$.

Proof. Let the system error dynamics be written as follows: $\dot{e} = -k_1\,\phi(e) + \Gamma_l\,\tilde{\theta}$ and $\dot{\tilde{\theta}} = -k_2\,\alpha(e)\,\Gamma_l^{\top}$, and define the state vector $\zeta = [\,\phi(e)\ \theta^{\top}\,]^{\top}$. Now, consider the following Lyapunov function candidate: $V(t,e,\tilde{\theta}) = \zeta^{\top} P(t)\,\zeta$ where $P(t)$ is a symmetric, bounded and positive definite matrix satisfying $\dot{P}(t) = -\phi'(e)[P(t)A(t) + A^{\top}(t)P(t) + Q(t)]$. The function $V(t,e,\tilde{\theta})$ is continuous and continuously differentiable everywhere in $\mathbb{R}^3$, except on the set $\Omega = \{(e,\tilde{\theta}) \in \mathbb{R}^3 \,|\, e = 0\}$, where it is not Lipschitz continuous and $\dot{V}$ is not well defined for the partial derivative $\phi'(e)$. Notice that when $e = 0$ in isolated points it implies that $\alpha(e) = \phi'(e)\phi(e)$ and $\dot{\zeta} = \phi'(e)A(t)\zeta$. Then, the time-derivative of V takes the form: $\dot{V}(t,e,\tilde{\theta}) = \zeta^{\top}[\dot{P}(t) + \phi'(e)P(t)A(t) + \phi'(e)A(t)^{\top}P(t)\,]\,\zeta$, at the points where V is differentiable, which implies that $\dot{V}(t,e,\tilde{\theta}) = -\phi'(e)\zeta^{\top}Q(t)\zeta = -1/2\left(|e|^{-1/2} + 2\right)\zeta^{\top}Q(t)\zeta$. Notice that $|e_1|^{\frac{1}{2}} \leq \|\zeta\|_2$, with $\|\zeta\|_2^2 = \zeta^{\top}\zeta = |e| + 2|e|^{\frac{3}{2}} + e^2 + \|\tilde{\theta}\|_2^2$ as the Euclidean norm of ζ. Thus, $\dot{V}$ may be upper bounded as follows: $\dot{V}(t,e,\tilde{\theta}) \leq -c_3\,(1/2)(\|\zeta\|_2 + 2\|\zeta\|_2^2)$. This expression holds when the trajectories of system errors are outside of the set Ω, and it indicates that $\dot{V} < 0$ in the complement of the set Ω. To remain on the set Ω for a given time interval $t \in [t, t+T_0]$, it is necessary that $e(t) = 0$ and $\Gamma_l\,\tilde{\theta}(t) = 0$ during that interval, which violates the persistent excitation (PE) condition. Then, it follows from Zubov's theorem Poznyak (2010) that the origin is asymptotically stable. For a more detailed proof, please consult Boubaker and Iriarte (2017); Moreno and Guzman (2011). □

5.1.2 Spring Parameters Identification

Here, we address the balancing and hopping motion control problem for a spring-loaded inverted pendulum (SLIP) based robot leg, in the presence of uncertainties and external disturbances. We consider a real-world scenario where a given monopod robot suffers a failure in its embedded array of spring-damper pairs during the landing phase. Some factors that cause viscoelastic elements to fail are: fatigue, inadequate materials, poor manufacturing processes, improper service and environmental effects. Then, using the finite time algorithm, the robot can achieve a successful landing and balancing, while performing an efficient tracking control for its absolute angular position (as presented in Chapter IV), even in the presence of such failure modes.

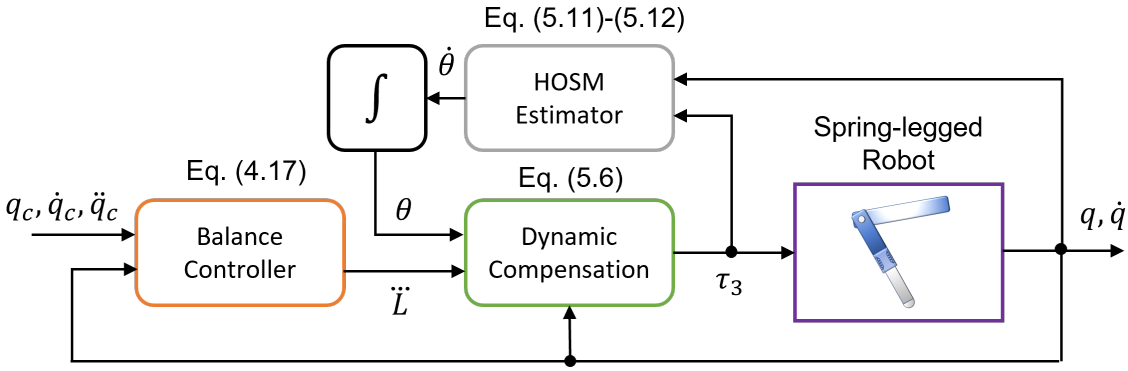

Figure 5.1 Block diagram of the proposed balance control strategy.

The control strategy is presented in Fig. 5.1, where the algorithm is used to estimate the external force generated by a viscoelastic element (spring plus dashpot), located between the robot leg and foot, identifying its elastic and viscous coefficients. The stability analysis of the overall closed-loop system is demonstrated by using the Lyapunov stability theory. Numerical simulations are included to illustrate the performance and feasibility of the proposed control methodology.

Robot Model

The employed robot is shown in Fig. 5.2, and is similar to the robot introduced in Chapter IV. The links' mass and length parameters are given as follows: $m_1 = 0.2\,\text{kg}$, $m_2 = 0.5\,\text{kg}$, $m_3 = 2.0\,\text{kg}$, $l_1 = 0.2\,\text{m}$, $l_2 = 0.3\,\text{m}$, and $l_3 = 0.5\,\text{m}$. Each link i is modelled as a uniform thin rod, which means that the CoM is half way along the rod, and the rotational inertia about the CoM is $I_i = m_i l_i^2/12$ for $i = 1,2,3$. Joint 2 is passively actuated by an array of two parallel and identical springs with stiffness and damping coefficients of $1000\,\text{N}\,\text{m}^{-1}$ and $8\,\text{N}\,\text{s}\,\text{m}^{-1}$

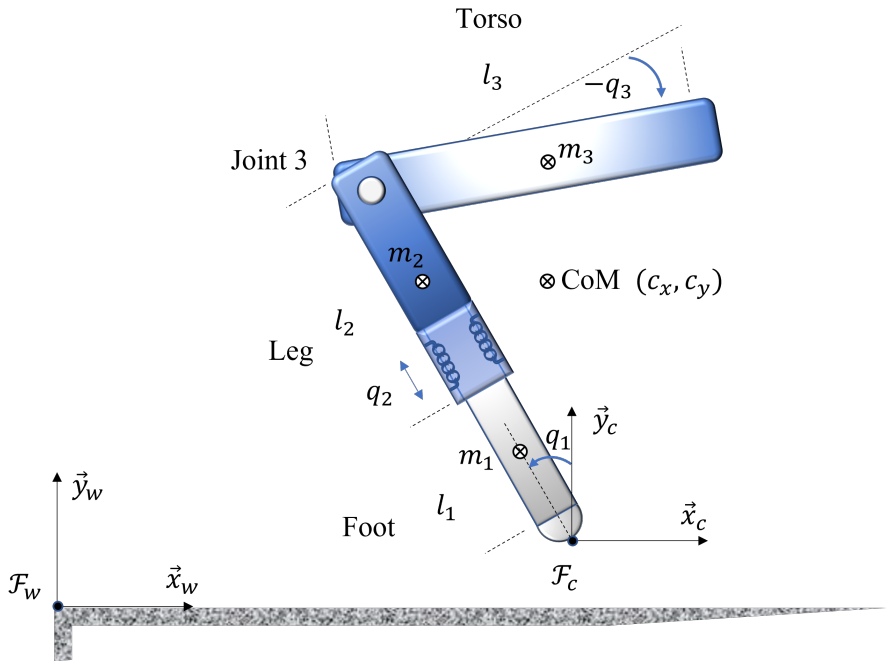

Figure 5.2 Legged robot model: joint variables, lengths and masses.

each, which results in an under-damped system. The total stiffness is appropriate for a legged robot to make small hops of around 1 m and is soft enough that Joint 2 moves significantly during landings and fast balancing movements. In other words, the stiffness is low enough to interfere significantly with the actions of the balance controller.

The tracking results are compared with rigid leg robot (similar to the results shown in Chapter IV) obtained from the springy-leg robot by locking Joint 2 in the position it takes when the robot is balanced, the torso is horizontal (so $q_a = 0$), and the spring is holding the weight of the upper leg and torso, which works out as $q_2 = -0.0282\,\text{m}$.

Balance Controller

As described in the previous chapter, the goal of the balance controller is to calculate a suitable value for $\ddot{L}$ to ensure that $q_a = q_1 + q_3$ follows a given desired signal q_c, while maintaining the robot's balance. Now, recalling equation (4.23) from previous chapter to obtain τ_3 (necessary torque to achieve the $\ddot{L}$ obtained from the controller), and assuming that

K_s and D_s are both uncertain applied to the absolute tracking problem, we have:

$$\begin{bmatrix} 0 & H_{01} & H_{02} & H_{03} \\ 0 & H_{11} & H_{12} & H_{13} \\ 0 & H_{21} & H_{22} & H_{23} \\ -1 & H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} \tau_3 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 \\ -C_1 \\ \hat{F}_s - C_2 \\ -C_3 \end{bmatrix}, \tag{5.7}$$

where $\hat{F}_s$ is the estimated spring-damper force, and the equation can be solved for both τ_3 and $\ddot{q}_3$. Next subsection introduces a state estimator (or observer) to provide an accurate estimate of the internal force F_s, according to (5.7), by identifying the stiffness and damping coefficients, K_s and D_s, of the viscoelastic element located at the Joint 2, while the springy-leg robot is balancing and tracking a given position desired signal.

HOSM Observer

Considering the practical case in which one of the pairs of the spring-damper system fails due to the rupture of the spring in the landing phase and the internal force F_s drops down (e.g., 50%), affecting the robot's ability to balance and hop. Then, we propose a high-order sliding mode (HOSM) observer to ensure a successful landing, balancing, and steering for the springy-leg robot while it estimates the stiffness and damping coefficients of the viscoelastic element. It is worth noticing that the controller introduced in Azad and Featherstone (2013) can be applied to the proposed control problem given by (5.7) for similar balancing and trajectory tracking tasks. However, less accuracy is expected since the torque is directly computed from the control law without compensating the robot dynamics.

Let us introduce the following notation:

- H and C denote respectively the inertia matrix and the gravity, Coriolis and centrifugal, and friction forces for the springy-leg robot, similar to (4.1);
- q_x and q_y are extra prismatic joints used to describe the position of the robot's foot with respect to the world frame;
- positions, velocities and accelerations of the robot are described by the vectors q, $\dot{q}$ and $\ddot{q}$ respectively, where $q = \begin{bmatrix} q_x \; q_y \; q_1 \; q_2 \; q_3 \end{bmatrix}^\top$ and so on;
- the virtual vector $y = \begin{bmatrix} q_x \; q_y \; q_1 \; q_2 \; q_\mathrm{a} \end{bmatrix}^\top$ satisfies $\dot{q} = T\,\dot{y}$ and $\ddot{q} = T\,\ddot{y}$, where T is a mapping matrix similar to the matrix defined in (4.13), and $\dot{y}$ and $\ddot{y}$ denote the velocity and accelerations of the virtual vector y;

- $\tau = \begin{bmatrix} 0 & 0 & 0 & 0 & \tau_3 \end{bmatrix}^\top$ denotes the torque produced by the balance controller;
- $F = \begin{bmatrix} 0 & 0 & 0 & F_s & 0 \end{bmatrix}^\top$ denotes the internal force generated by the spring-damper element.

Then, similarly to (4.5) and using the aforementioned notation, the motion dynamics for the springy-leg robot can be rewritten simply as Featherstone (2017):

$$\ddot{y} = (HT)^{-1} \left[\tau + F - C \right], \tag{5.8}$$

or, in a more detailed manner, as:

$$\ddot{y} = (HT)^{-1} \left[\tau - C \right] + (HT)^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -q_2 & -\dot{q}_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} K_s \\ D_s \end{bmatrix}, \tag{5.9}$$

which may be written more compactly as

$$\ddot{y} = f(q, \dot{q}, \tau_3) + \Gamma(q, \dot{q})\, \theta, \tag{5.10}$$

where $f(\cdot,\cdot,\cdot)$ is a nonlinear matrix function, $\Gamma(\cdot,\cdot)$ is the regressor matrix and $\theta = \begin{bmatrix} K_s & D_s \end{bmatrix}^\top$ denotes the vector of uncertain parameters to be identified. Here, without loss of generality, we assume that $\Gamma(\cdot,\cdot)$ is measurable and bounded in norm i.e., $0 < \Gamma_m \leq \| \Gamma(q, \dot{q}) \| \leq \Gamma_M < \infty$ for $\forall q, \dot{q}$, and satisfies the persistent excitation (PE) condition (Poznyak, 2010). Selecting the last coordinate of the acceleration vector $\ddot{y}$ in (5.10) we obtain:

$$\ddot{q}_\mathrm{a} = f_l(q, \dot{q}, \tau_3) + \Gamma_l(q, \dot{q})\, \theta, \tag{5.11}$$

where $f_l(\cdot,\cdot,\cdot)$ is a nonlinear scalar function obtained from the last row of $f(\cdot,\cdot,\cdot)$ and $\Gamma_l = \begin{bmatrix} \Gamma_{l1} & \Gamma_{l2} \end{bmatrix}$ is a regressor vector obtained from the last row of the regressor matrix $\Gamma(\cdot,\cdot)$. Here, the objective is to estimate θ using a nonlinear parameter identification algorithm provided that $\dot{q}_\mathrm{a}$ and Γ_l are both measurable from the system signals. Then, let us briefly address the high-order sliding mode (HOSM) observer approach based on the finite-time algorithm and briefly describe its convergence properties. A complete explanation for the key features of the HOSM observer approach can be found in Boubaker and Iriarte (2017); Moreno and Guzman (2011).

According to (5.11), a finite-time parameter identification algorithm can be described by:

$$\ddot{\tilde{q}}_{\mathrm{a}} = -k_1\phi(e) + \Gamma_l(q,\dot{q})\,\hat{\theta} + f_l(q,\dot{q},\tau_3)\,, \tag{5.12}$$

$$\dot{\hat{\theta}} = -\alpha(e)\begin{bmatrix} k_2 & 0 \\ 0 & k_3 \end{bmatrix}\Gamma_l^{\top}(q,\dot{q})\,, \tag{5.13}$$

where k_1, k_2, k_3 are positive gains, $e = \dot{\tilde{q}}_{\mathrm{a}} - \dot{q}_{\mathrm{a}}$ is the parameter estimation error where $\dot{q}_{\mathrm{a}}$ is obtained from the sum of joint velocities $\dot{q}_1$ and $\dot{q}_3$ (both assumed to be measurable), $\dot{\tilde{q}}_{\mathrm{a}}$ is the absolute angular velocity estimated by the super-twisting or the fixed-time observer algorithms (Boubaker and Iriarte, 2017), $\hat{\theta}$ is the vector of uncertain parameters to be identified by using the gradient-type adaptation law given by (5.13). Then, it is possible to define the parameter identification error $\tilde{\theta} = \hat{\theta} - \theta$ to be identified by using the gradient-type adaptation law given by (5.13). Finally, $\phi(e)$ and $\alpha(e)$ are nonlinear functions of the parameter estimation error e defined in equations (5.6) and (5.5).

Simulation Results

In this section, we present the simulation results of an experimental case in which the springy-leg robot lands with a velocity of $-5\,\mathrm{m\,s^{-1}}$ along the y-axis. This is similar to falling from a height of 1.3 m from the foot to the ground. At the touch-down, one of the springs breaks at $t = 0.0114\,\mathrm{s}$ when q_2 is compressed by 70 mm dropping the spring array's force by 50%. Next, q_2 reaches the maximum compression at 164 mm due to the low stiffness in the spring's array, producing a small hop after all the elastic energy is released, and lifting the robot $q_y = 0.45$ m between the foot and the ground. When the robot lands for a second time the spring compresses 62 mm, and the balance controller is able to stabilize the system, and then executes a motion command consisting of a sequence of ramps and sinusoids. The estimator and the balance controller work collectively during the whole experiment, and the dynamic simulation of the springy-leg robot can be seen in Gamba et al. (2021).

In this simulation, we use the *ode23t* solver from MATLAB with relative tolerance set to 10^{-6} and other parameters chosen at their default values. Although the balance controller assumes that the foot never loses contact nor slips with the ground, in the simulation we have included the ground-contact model presented in Chapter IV.

The balance controller and the online estimator are switched on from the beginning of the simulation. The behavior over time of the controller's state variables and estimation variables are depicted respectively in Fig. 5.3 and Fig. 5.4. The poles of the controller are $\lambda_1 = -20$, $\lambda_2 = -20$, $\lambda_3 = -20$, and $\lambda_4 = -1/T_{\mathrm{c}}$, all expressed $\mathrm{rad\,s^{-1}}$. The gains of the estimator are:

$k_1 = 10.83$, chosen so that it ensures the persistent excitation condition, $k_2 = 9.9 \times 10^3$ and $k_3 = 18.26$, which are selected according to the magnitude of θ and the desired convergence rate of $\hat{\theta}$. The estimated parameters $\tilde{\theta}$ and e are initialized to zero at the beginning of the simulation when $K_s = 2000\,\mathrm{N\,m^{-1}}$ and $D_s = 16\,\mathrm{N\,s\,m^{-1}}$ at the spring's array.

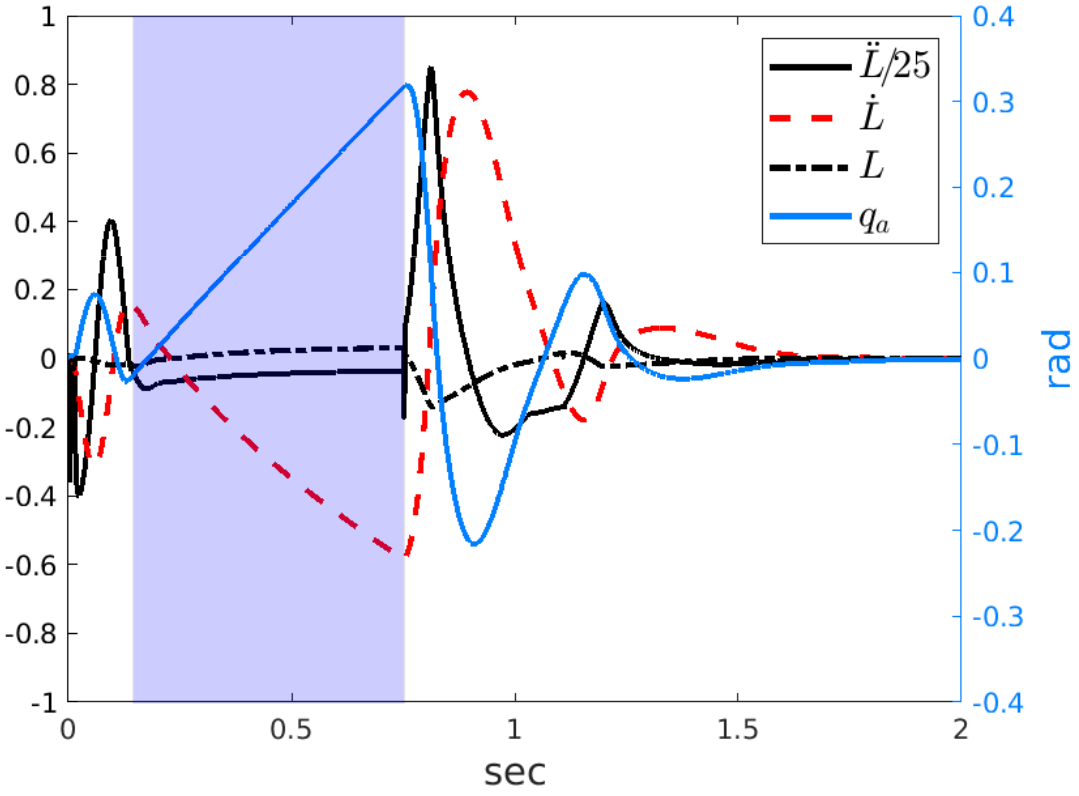

Figure 5.3 Evolution of the controller's state variables from the moment of landing until the robot has settled. The left side scale corresponds to $\ddot{L}$, $\dot{L}$ and L, and the right side scale corresponds to q_a. The shaded zone shows the time interval during which the foot has lost contact with the ground because of the small hop.

From Fig. 5.3, it can be seen that q_a is initially pushed to about 0.08 rad by the momentum of the landing before swinging slightly negative. Next, it rises to nearly 0.32 rad during the hop (shaded area) and then moves back to approximately −0.21 rad at the second landing. Finally, after a small bounce at $t = 1.085\,\mathrm{s}$, the absolute joint q_a is driven close to zero.

In figure 5.4, it is possible to observe some oscillations in $\hat{D}_s$ between 1.274, 14.93 and 10.9 $\mathrm{N\,s\,m^{-1}}$ before the hop (shaded area). This oscillatory behavior may arise due to the lack of persistent excitation of $\dot{q}_\mathrm{a}$ between the rupture point ($t = 0.0114\,\mathrm{s}$) and the hop ($t = 0.1456\,\mathrm{s}$). q_2 only shrinks and expands once during this period. Notice that $\hat{\theta}$ ($\hat{K}_s$ and $\hat{D}_s$) tends to drift away when the persistent excitation ($\dot{q}_\mathrm{a} \approx 0$) is absent or very small, and

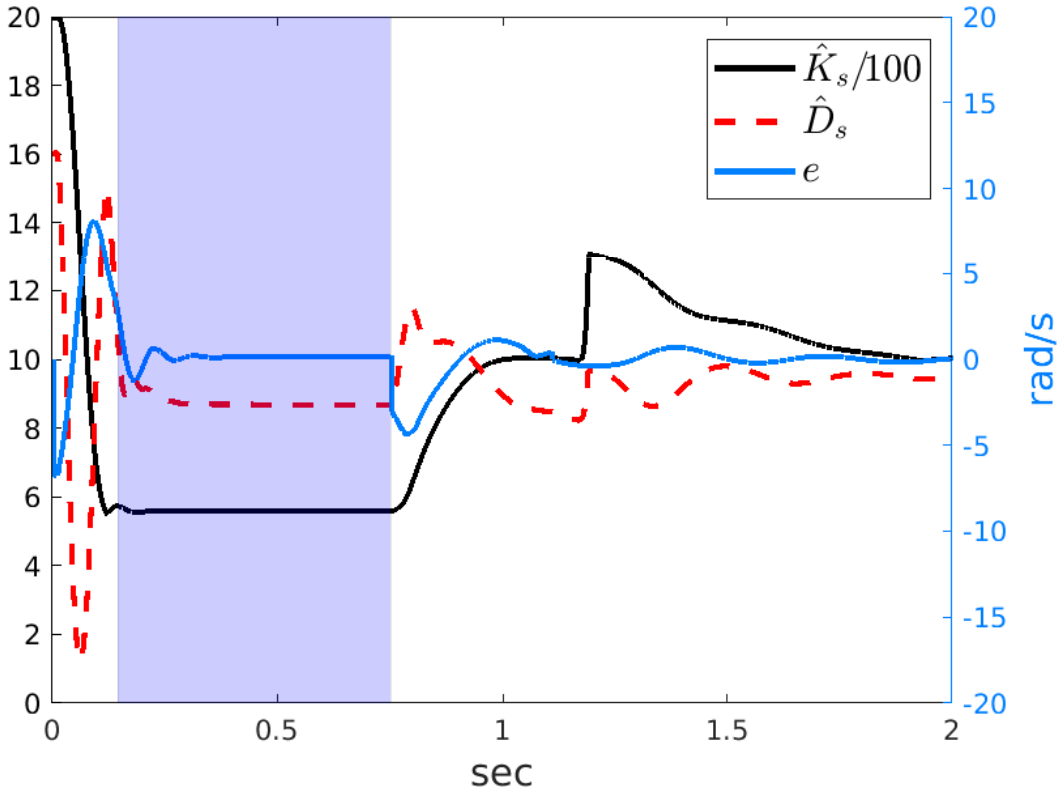

Figure 5.4 Evolution of the estimator's parameters from the moment of landing until the robot has settled. The left side scale corresponds to $\hat{K}_s$ and $\hat{D}_s$, and the right side scale corresponds to e. The shaded zone shows the time interval during which the foot has lost contact with the ground because of the small hop.

the system continues moving due to the elastic energy on the spring (spike drift between 1.15 s and 1.25 s in Fig. 5.4).

The failure occurs at $t = 0.0114$ s when the q_2 compresses to 70 mm. After the rupture, the force produced by the spring's array drops 50% to $K_s = 1000\,\mathrm{N\,m^{-1}}$ and $D_s = 8\,\mathrm{N\,s\,m^{-1}}$, the estimated $\hat{K}_s$ decreases to $550\,\mathrm{N\,m^{-1}}$ and creates a small overshoot just before the hop (shaded area).

Then, the robot's foot loses contact with the ground (small hop) between $t = 0.1456$ s and $t = 0.7510$ s, and reaches the maximum height of 0.45 m between the foot and the ground. This flight period is shown shaded in both Fig. 5.3 and Fig. 5.4. The balance controller does not know that the foot has left the ground, and continues using the model described in the previous chapter, which assumes that the robot's foot is always on the ground. On the other hand, it is assumed that the observer knows the position and velocity of the whole system in the world frame. Although this assumption is difficult to fulfill on real applications

and may need external sensors like cameras that are always watching the robot, the scope of these results is to give a benchmark of the best observer's response when all the robot states are available. During the flight period, the controller's and estimator's stability are not significantly affected by the lack of ground contact, and the robot has come to rest within about 1 s after the second landing. Notice that the balance controller is assumed to know the orientation of the robot at all times, e.g., using an onboard IMU, so it always knows the correct values of q_1 and $\dot{q}_1$.

At $t = 2.011$ s the desired signal q_c switches from zero to a sequence of ramps and sinusoids similar to the one mentioned in subsection 4.4.2 that asks the robot to make large and fast motions. The control sequence starts with three ramps between 0 and 0.8 rad at a speed of 3 rad s^{-1}, followed by a long ramp from 0.8 rad to -0.6 rad at -2.5 rads^{-1}, followed by two more ramps at speeds of 2.5 rads^{-1} and -1 rads^{-1}, followed by three cycles of a sine wave at 1 Hz. Fig. 5.5 plots this portion of the action sequence.

Two zeros are introduced to the desired signal at -20 rad s^{-1}, as mentioned in (4.21). For this springy-leg robot, T_c varies between 0.1562 s at $q_a = -0.6$ rad and 0.2507 s at $q_a = 0.8$ rad in this action sequence. The rigid-leg response is obtained from a separate simulation in which the initial conditions are set as close as possible to the actual state of the springy-leg robot at $t = 2.011$ s. The behavior of the signals is shown in Fig. 5.5.

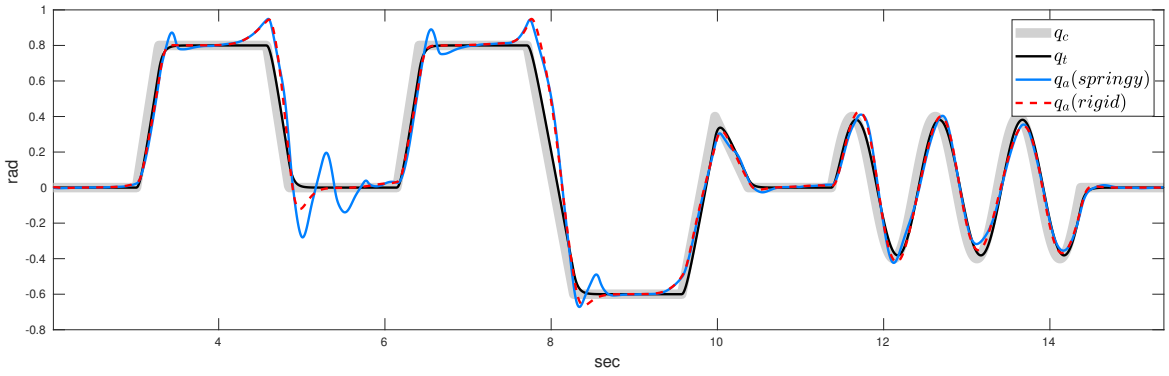

Figure 5.5 Absolute motion tracking performance of the balance controller. q_c is the original desired signal, q_t denotes the theoretical response of the balance controller if the plant was really linear, $q_a\,(springy)$ is the actual response of the absolute joint on the springy-leg robot, and $q_a\,(rigid)$ denotes the actual response on the rigid-leg robot.

In Fig. 5.5, it can be seen that there are some significant tracking errors, particularly at the beginning and end of the ramps. Three main factors contribute to such errors: (i) the balance controller assumes that the plant in Fig. 4.2 is linear when in reality it is not; (ii) for simplicity, the acausal filter uses values of T_c calculated for a balanced configuration at

each instant instead of a leaning configuration; (iii) at each of the balance configurations used to calculate T_c, it assumes that q_2 is constant for the whole tracking when in fact it oscillates around 10% and reaches a peak of 20%, causing tracking inaccuracies as shown in Figures 5.5 and 5.6. At a given moment, we can see that a difference occurs at the end of the second ramp, around 4.8 s, where the springy-leg response shows a small amount of ringing that dies away in about 0.5 s. There is also a little bit of ringing at the end of the long ramp. However, considering the under-damped nature of the spring-damper pair in the leg, there is remarkably little ringing overall, and what little there is dies away quickly.

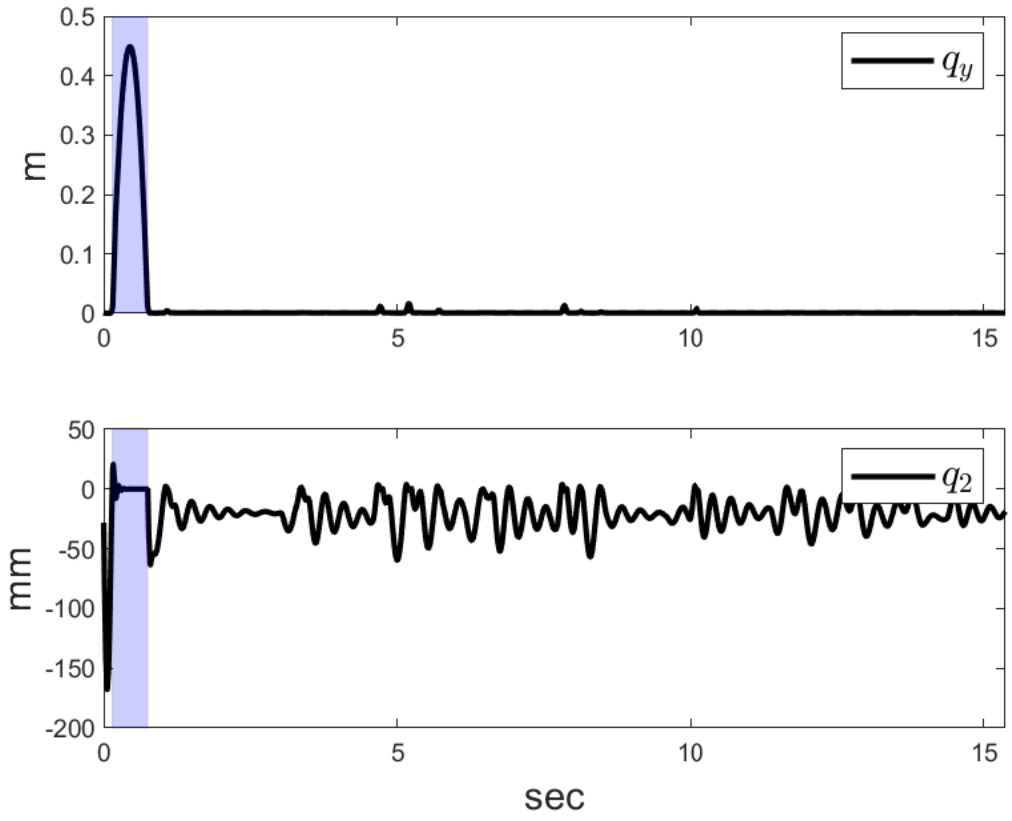

Figure 5.6 Motion of the foot and spring during landing and tracking of the desired signal in Fig. 5.5.

Figure 5.6 shows the motion of the spring and the foot along the y-axis during landing (shaded area) and tracking of the command sequence (after the shaded area). At the landing phase, the spring reached a compression peak of about 16.7 cm, which is 33.4% of the leg length. During the tracking task, the spring compression varied in a range of 5.5 cm, which is 11% of the leg length. At the foot position (along with the y-axis) graph, it is possible

to observe the hop that occurred after the spring's rupture; it also shows that the robot lost contact with the ground around seven times during the tracking phase.

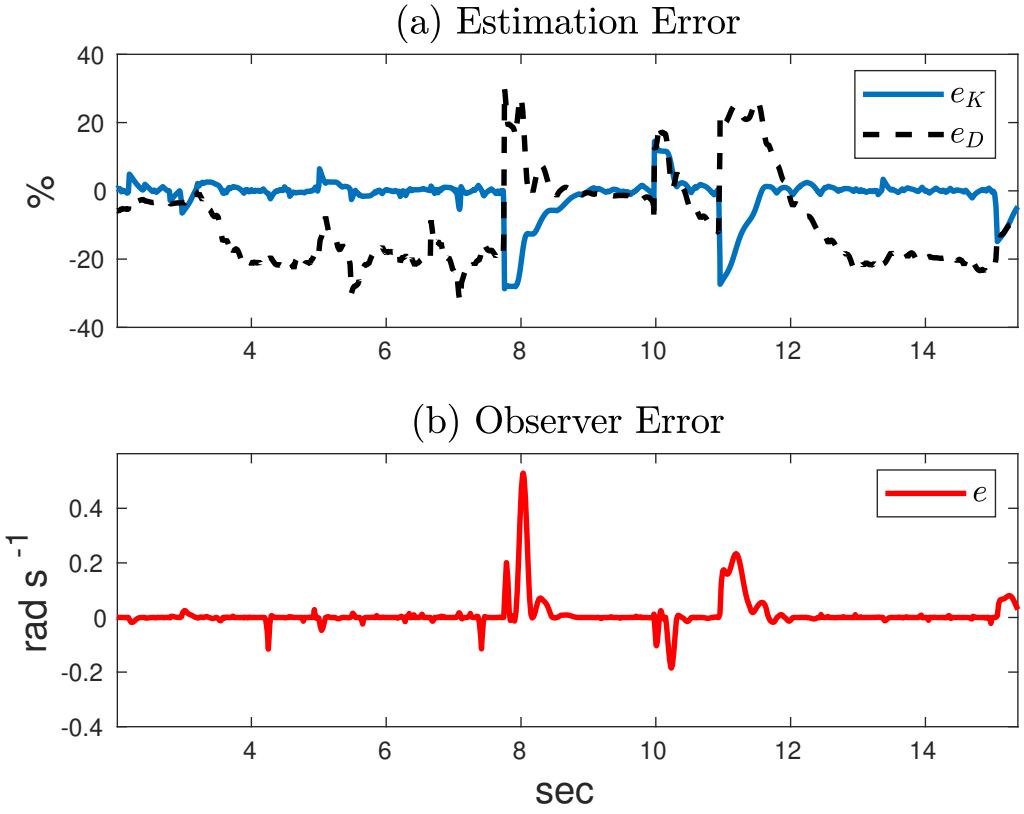

Figure 5.7 Figure *a* shows the evolution of the error related to the estimations $\tilde{\theta}$. Figure *b* shows the observer's injected term error e response. Both graphs display information from the tracking phase shown in Fig. 5.5.

In previous simulations, the balance controller has presented a certain robustness to inaccuracies at the spring force estimation below 25%. Figure 5.7 shows the evolution in time of the estimated error in percentage (e_K for the stiffness coefficient error, and e_D for the damping coefficient error), and the observer's injected term error e during the tracking phase, e_K presents two peak variations of -28.68% and -27.36% at $t = 7.7\,\text{s}$ and $t = 10.9\,\text{s}$ respectively, which coincides with the instants that present larger error at the injected term. These drifts can be correlated to the lack of PE. Moreover, as the damping coefficient contributes less to the dynamics of the system, it seems to be more difficult to have a stable estimation of it than the stiffness.

Then, in this experimental case, we can conclude that the balance controller combined with an HOSM observer has accomplished three tasks simultaneously using only a single

actuator: balance the robot, follow the commanded signal, and suppress vibrations, even in the presence of uncertainties in the stiffness and damping coefficients of the viscoelastic element.

5.2 Conclusion

This chapter has studied to what degree a springy leg may affect the performance of the balance controller presented in Gamba and Featherstone (2021), combined with a HOSM observer presented in Moreno and Guzman (2011) when a failure occurs in the spring-loaded device, which loses 50% of its stiffness due to a spring fracture. This strategy aims to keep the robot in balance in the presence of large and fast motions, as well as parametric uncertainties. It was shown that the balance controller can cope with large, fast motions that occur during landing, including small bounces (Figures 5.3 and 5.4). Moreover, it can track large, fast motion commands even with a noisy spring force estimation after landing, with almost the same accuracy that could be achieved if the leg was rigid (Fig. 5.5). The drifts at the observer estimation do not significantly reduce the balance controller performance at landing, balancing, and tracking, which gives a sense of certain robustness to inaccuracies in the spring model.

The next chapter will analyze the estimation performance of the absolute orientation acquired from three different IMUs subject to continuous low-intensity impacts (4 g maximum acceleration along the z-axis). Similar to mounting the IMU at a running robot.

Chapter 6

IMU Bouncing Test

The feasibility of highly athletic motions on real robots relies on the behavior of the existing hardware when executing this kind of task. In contrast to rovers and drones, a monopod robot needs to produce leaps and hops for moving along the ground. Consequently, the study of how well these sensors perform in such bouncing conditions is critical for the proper operation of the robot.

One of the technological objectives of the Skippy project was to measure the performance of off-the-shelf components such as encoders and IMUs when subjected to the kinds of motion that a highly athletic robot might experience, like bouncing, landing, etc. In this sense, testing and analyzing the response of different IMUs subject to continuous bouncing applications is also part of this study. The principal purpose of this test is to discover whether or not exposure to prolonged bouncing motion causes the IMU's estimate of its orientation to become inaccurate.

In the IMU bouncing test, it is necessary to find the most suitable peripherals for facilitating the mechanical and control implementation. The controller should guarantee the proper execution of the control loop and data acquisition at the specified frequency. It also needs to allow online monitoring, logging, and control during the experiment execution. Finally, at the end of the test, the application should have the option to save the logged data in a .mat file for further analysis in MATLAB.

In this chapter, we develop a test rig to explore the behavior of three different IMU sensors subject to continuous bouncing. This task was performed by the Skippy team and I contributed to the realization of the experiment by:

- Coordinating the purchase of the actuation system;

- Developing the low-level and high-level code presented in the sbRIO development board (NI, 2021b) to control the solenoid and log the measurements obtained from the optical encoder and one IMU;
- Finding the ideal filter used for post-processing the measured data for the respective analysis.
- Providing support during the experiments and data analysis.

Due to time constraints, It wasn't possible to expand the logging code for the other two IMUs on the SbRIO, and so the data was captured using different devices.

Our main objective in this chapter is to test the orientation estimation of different IMU sensors subject to simultaneous bouncing conditions, which are common in legged systems subject to running and hopping activities.

6.1 Experimental Setup

The apparatus consists of a long rod connected to a rotational actuator, and three IMUs mounted at the other end. The actuator moves the rod up and down in a vertical plane. The IMUs travel up and down in a circular arc with a large radius, imitating a hopping movement.

This experiment was performed using a one meter carbon fiber rod with a diameter of twenty millimeters and a wall thickness of 0.5 mm connected to a rotatory solenoid (Magnet-Schultz GmbH, 2021). Some 3D-printed parts and an aluminum hub were used to complete the mechanical design of the test rig. At the base, an optical encoder (RLS, 2021) is placed to obtain the angular position of the rod during the experiment. The experiment requires mounting the three different IMUs (BOSCH, 2021; LORD Sensing Microstrain, 2021a; VectorNav, 2021) at the end of the rod. The controller was implemented on the sbRIO-9637 development board from NI using the software LabVIEW (NI, 2021a). The board was configured based on a "Supervisory Control and Data Acquisition" (SCADA) architecture to monitor online the acquired data and store it if needed. Two extra devices (Arduino, 2021; LORD Sensing Microstrain, 2021b) were used to complete the logging of all the IMUs. The complete setup is shown in figure 6.1, and it is divided into the actuation system, the sensors, and the controller and data acquisition systems.

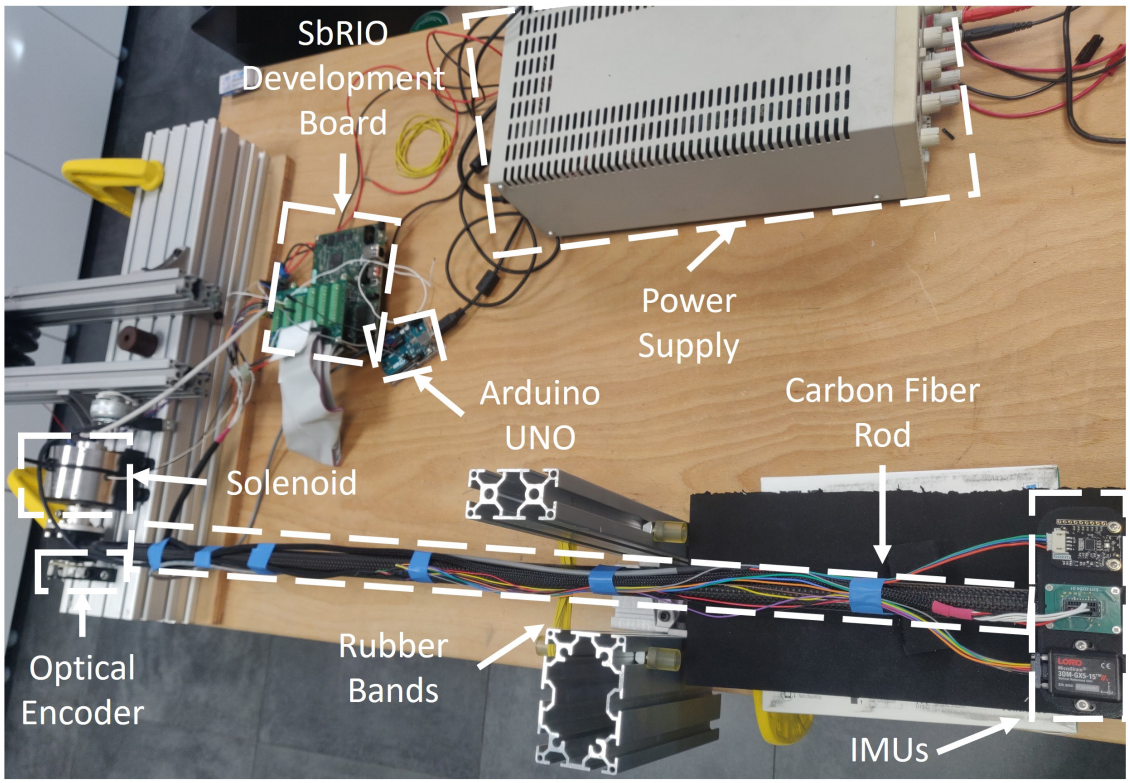

Figure 6.1 Vertical view of the experimental setup.

6.1.1 Actuation system

In the actuation system, there is a Rotary Solenoid Type G DR (GDR075X20A61-S1) (Magnet-Schultz GmbH, 2021) powered by a Pololu G2 High-Power Motor Driver (Pololu, 2021), which is controlled by the sbRIO-9637 development board (NI, 2021b). The solenoid directly actuates the carbon fiber tube as shown in figure 6.2.

6.1.2 Sensors

In the sensors system, there is one optical encoder and three different IMUs. The three IMUs are mounted on a custom-made 3D-printed support (Fig. 6.3) to allow them to have the y-axis aligned with the rotation axis of the solenoid. The data collected from the IMUs are the compensated linear accelerations, the compensated angular velocities, and the quaternions. The measured quaternions are offline converted into Euler angles.

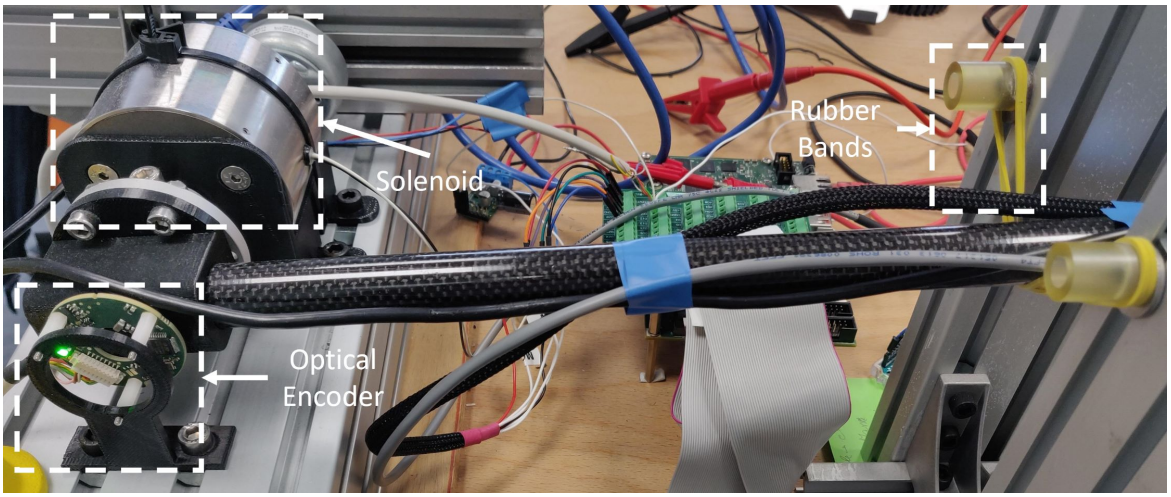

Figure 6.2 Base view of the experimental setup.

Optical Encoder

The angular position of the tube is measured employing an AksIM-2 absolute position encoder (RLS, 2021), sampled at 500 Hz by sbRIO-9637 development board (NI, 2021b).

VN-100

The Vectornav VN100 (VectorNav, 2021) is the IMU able to measure the higher linear acceleration (16 g). It is sampled at 500 Hz by the sbRIO-9637 board using the SPI communication protocol. The IMU has a custom-made PCB, and this IMU is intended to be used in the highly athletic monopedal robot called Skippy (Featherstone, 2021).

3DM-GX5-15

The Lord MicroStrain 3DM-GX5-15 (LORD Sensing Microstrain, 2021a) can measure linear accelerations up to 8 g and it is configured to stream data at 500 Hz continuously over the recommended standard 232 (RS232 (Han and Kong, 2010)) serial communication protocol to the software provided by the vendor (LORD Sensing Microstrain, 2021b). The IMU communicates using the RS232 protocol over a universal serial bus (USB) cable connected to the host computer.

BN0055

The Bosch BNO055 (BOSCH, 2021) is assembled on a DFRobot breakout (DFRobot, 2021). Although DFRobot breakout is discontinued, the BNO055 chip can be found in breakouts

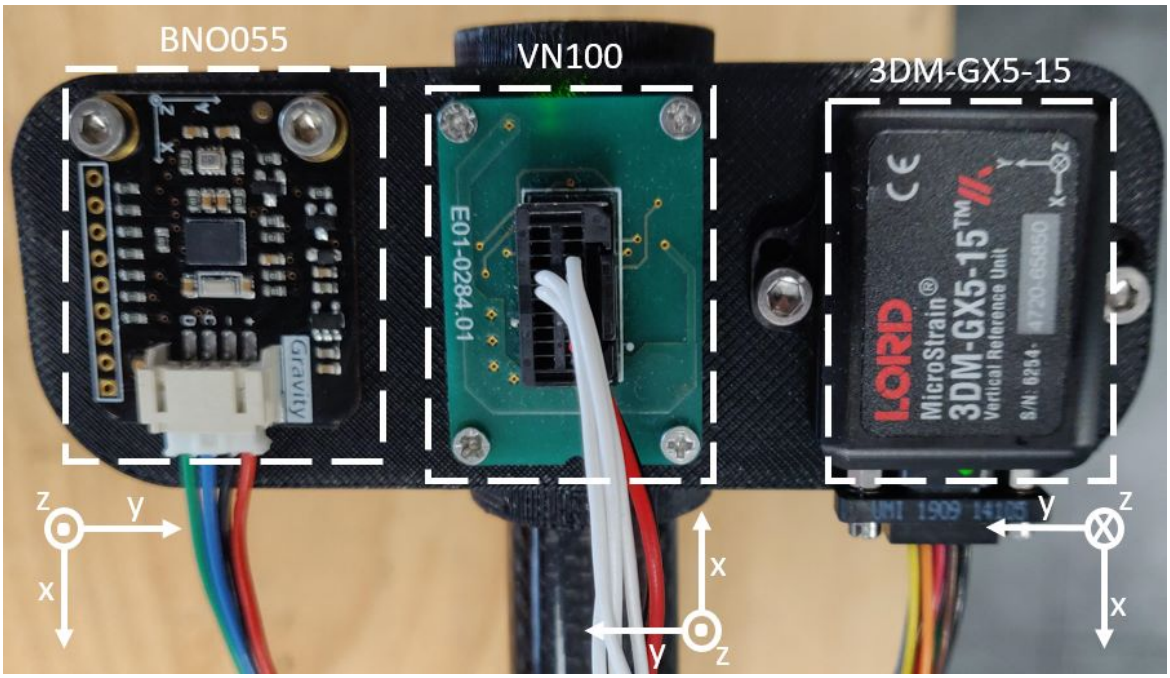

Figure 6.3 Vertical view of the IMUs mounted at the end of the carbon fiber rod.

with identical characteristics provided by other vendors, such as Adafruit (Adafruit, 2021). It is the cheapest among the tested IMUs (it costs a few tens of euros). This device has been configured at the lowest linear acceleration range (4 g) to have the highest possible resolution at the measurements. An arduino UNO board (Arduino, 2021) samples the device at 100 Hz by using the I2C communication protocol. The idea of this setup is to compare professional and expensive acquisition systems with hobbyist and cheap ones.

6.1.3 Controllers

The three independent data acquisition systems are activated simultaneously by a LabVIEW virtual instrument (VI). When the start button is pressed in the host's VI, the 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.

SbRIO 9637

The sbRIO-9637 board (NI, 2021b) has analog and digital inputs/outputs, a dual-core CPU, and a programmable FPGA (Xilinx, 2021). The board was configured based on a Supervisory Control and Data Acquisition (SCADA) architecture, where the FPGA oversees performing:

- The SPI communication;
- The PWM logic;
- The necessary signal conditioning.

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 Human-Machine-Interface (HMI) to monitor online the acquired data. The application also allows the user to clear, hold and save the logs in a .mat file for further analysis of the data collected.

Arduino UNO

The Arduino Uno board (Arduino, 2021) acquires the data from the Bosch BNO055 sensor and sends them to the PC through a serial interface. The PC then stores the data in a .csv file.

Host Computer

A host computer runs the software SensorConnect (LORD Sensing Microstrain, 2021b) provided by the vendor. The IMU Lord MicroStrain 3DM-GX5-15 communicates using the RS232 protocol over a USB cable connected to the one of the computer's port.

6.2 Experiment Description

The experiment continuously bounces the tube off the rubber bands and actuates the solenoid in the same direction of the rod's motion only when the rod's angle (measured by the encoder) is in a specified range (see Fig.6.4). 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 idea is to have a smooth transition between the touchdown and flying phases, which are the two components of the continuous bouncing behavior of a hopping robot.

The experiment starts by logging the sensors with the rod at a rest position for about one minute and nineteen seconds. Then, the rod is manually pushed downwards to start the

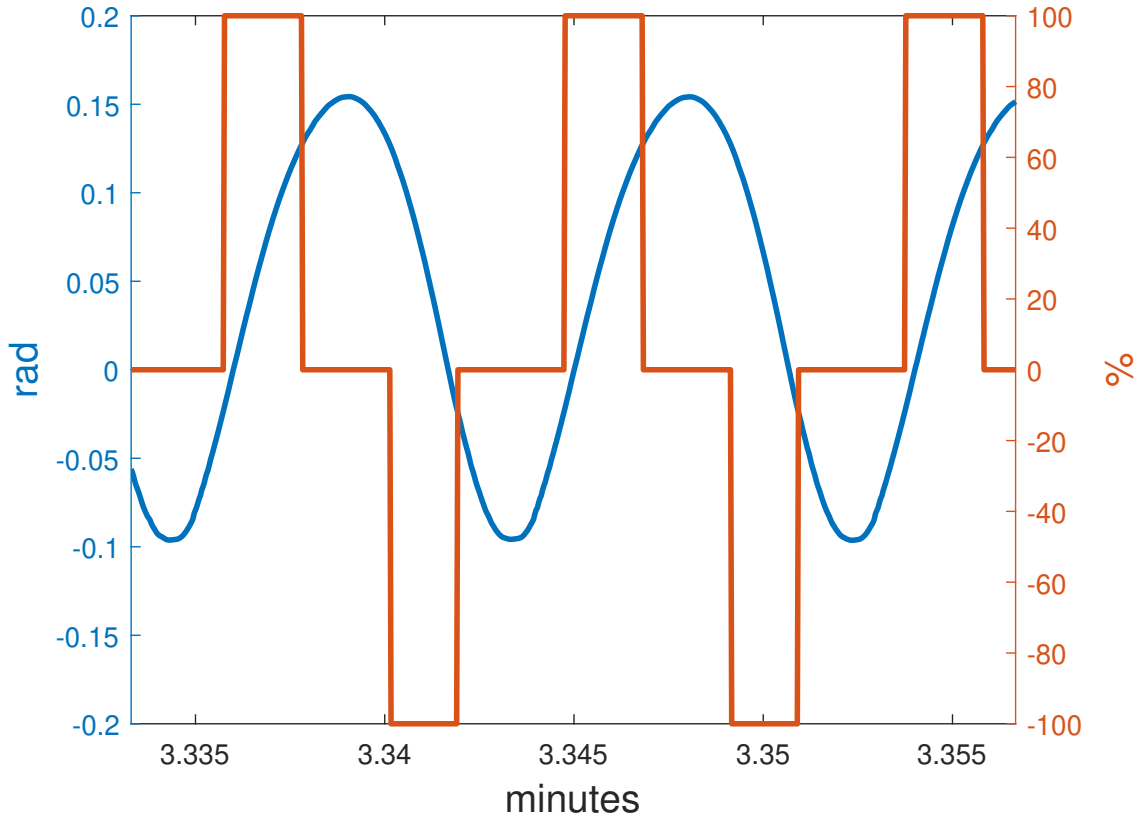

Figure 6.4 Actuation Logic. The blue line is the angle measurement obtained from the encoder, and the red line denotes the duty cycle of the PWM signal controlling the solenoid. The left-side scale applies to the encoder measurement, and the right-side scale applies to the solenoid's duty cycle.

bouncing motion, which continues for about three minutes and ten seconds. Finally, the solenoid is switched off, so that the bouncing ceases, and the system continues logging the sensors for one and a half minutes. After the data is captured, the measurements obtained from the encoder and the Euler rotations about the y-axis from the three IMUs are synchronized offline for further analysis. See figures 6.5 and 6.6.

6.3 Results

This experiment aims to analyze the estimation performance of the absolute orientation acquired from the IMUs subject to continuous low-intensity impacts, similar to mounting the IMU on a running robot. Due to the physical setup, all the tested IMUs have the y axis aligned with the rotation axis of the solenoid; this angle is directly obtained from the encoder

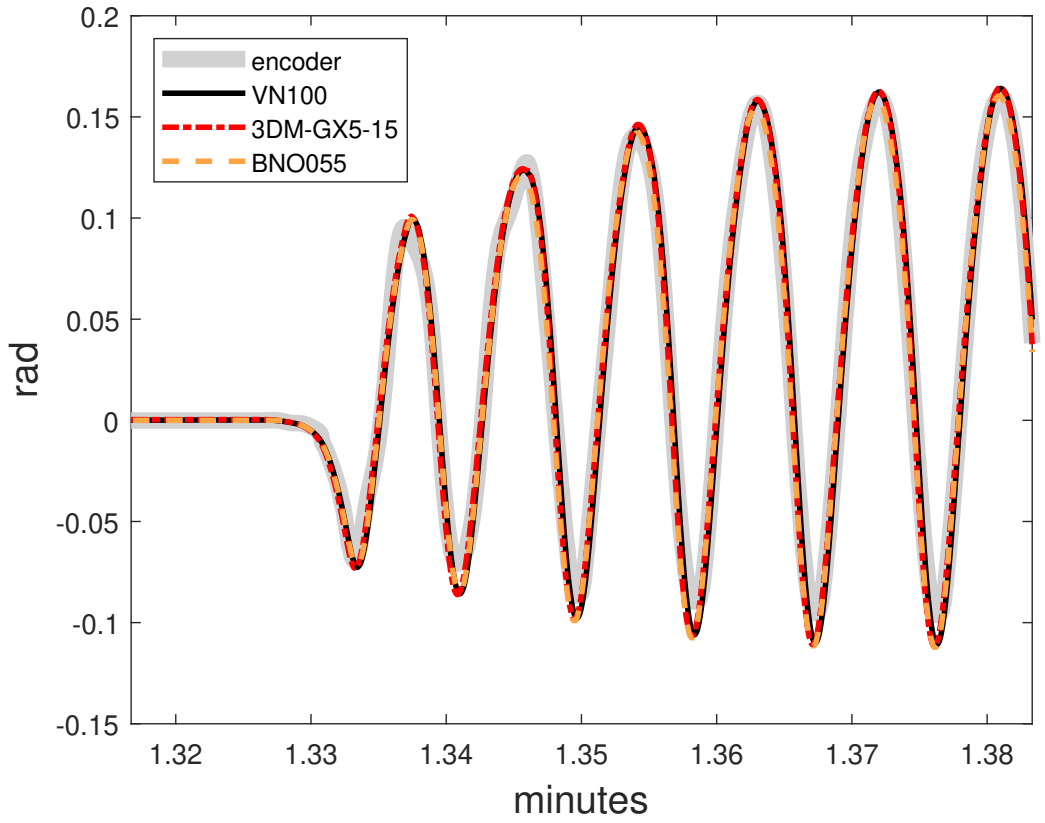

Figure 6.5 Synchronized measurements obtained from the encoder and the three IMUs at the beginning of the bouncing motion.

and is used as a reference to compute the rotation angles measured by the IMUs. Hence, the main focus is to study the difference between the angular position measured by the encoder and the angular rotation about the y-axis estimated by the IMUs. An error is calculated between the IMU's (rotation about the y-axis) and the encoder's measurements (see figure 6.7). During the experiment, the rod presented a minor bending while it hits the rubber bands. This bending behavior causes a slightly bigger rotation at the IMUs compared with encoder measurements (see the bottoms peaks of the VN-100 and GX5 IMUs responses in figure 6.6 before $t = 4.46\,\text{min}$). The error at the two remaining rotation axes (x and z axes) is the difference between the initial and final orientations because the sensors do not move about those axes during the experiment. The continuous bouncing of the specimen causes a periodic component in the y-axis errors. This component has been removed from the error graphs by a zero-phase low pass filter (the errors are displayed in figures 6.8 and 6.9 for the x and z axes). The filtered signals indicate the orientation drift of each IMU during the complete experiment (figures 6.7, 6.8 and 6.9).

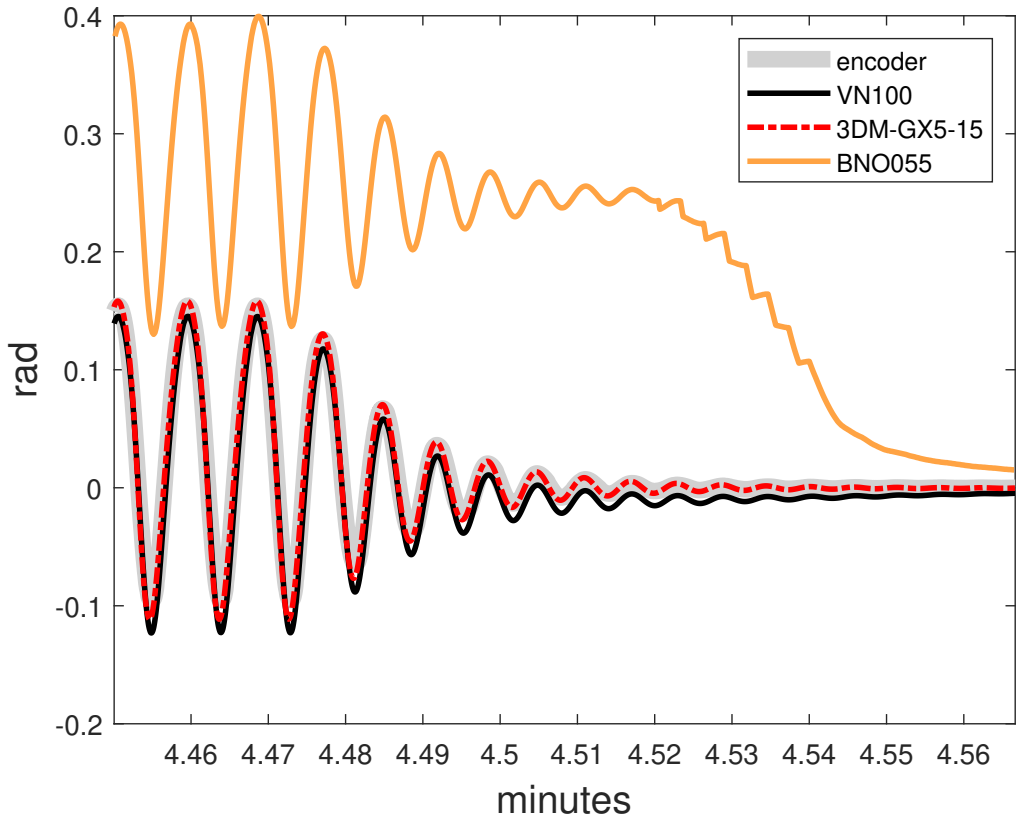

Figure 6.6 Synchronized measurements obtained from the encoder and the three IMUs at the end of the bouncing motion.

In this experiment, the maximum value of linear acceleration along the z-axis is 4 g. All three IMUs have been tested under the same conditions.

Figure 6.7 shows the filtered orientation error response about the y axis. In this graph, the three IMUs have an error very close to zero before the bouncing motion $t = 1.327$ min. During the bouncing motion, the VN-100 IMU drifts to a minimum value of -0.007 rad (-0.4011 deg) at 1.924 min, and a maximum value of 0.019 rad (1.0886 deg) at 4.239 min; the GX5 IMU drifts to a minimum value of -8.76×10^{-4} rad (-0.0502 deg) at 1.045 min (which is before the bouncing begins), and a maximum value of 2.28×10^{-3} rad (0.1306 deg) at 4.253 min; and the BNO055 IMU exhibits a maximum drift of 0.2775 rad (15.8996 deg) at 4.144 min. These drifts can also be seen in figure 6.6 at the end of the bouncing motion. At the end of the bouncing motion, at $t = 4.47$ min, the VN-100 and GX5 IMUs succeed in reducing the drift to a value close to zero; the BNO055 IMU error touches the zero value and converges to a 0.01123 rad (0.6434 deg) offset (the sensor failed to recover the right orientation).

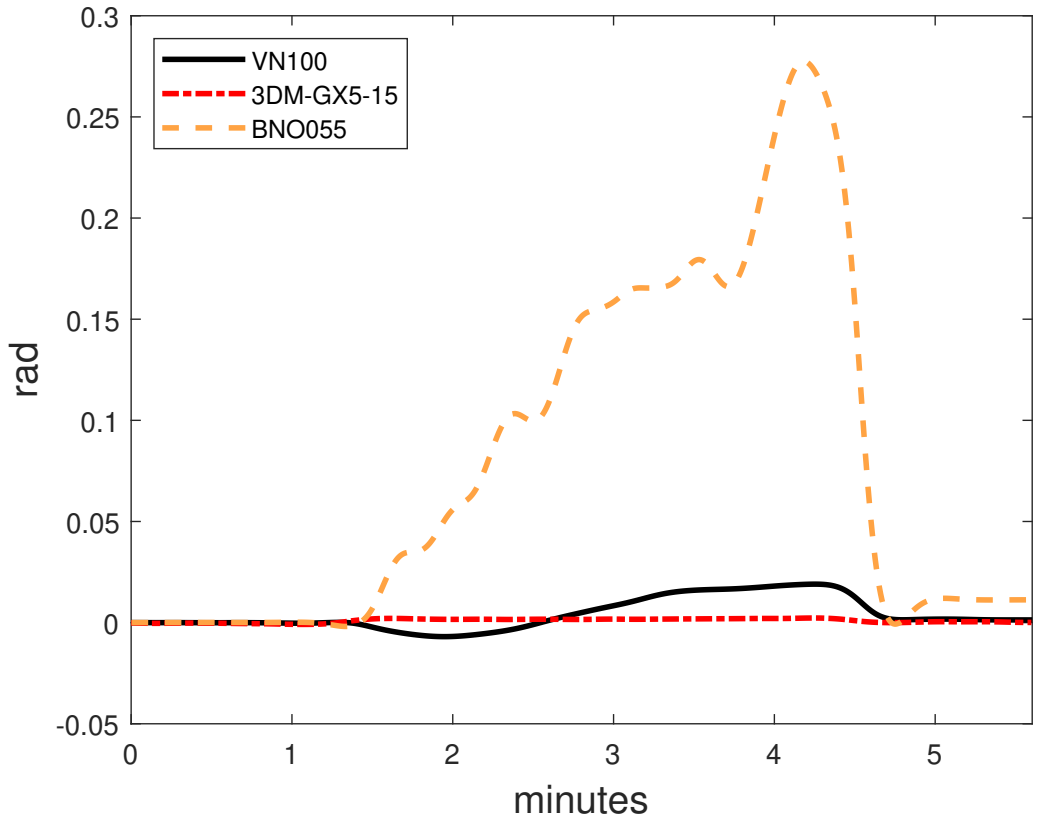

Figure 6.7 Filtered orientation error response about the y axis.

Figure 6.8 shows the filtered orientation error response about the x axis. In this chart, the GX5 IMU error varies between -6.5×10^{-4} rad (-0.0376 deg) and 3×10^{-4} rad (0.0172 deg) during the whole motion. However, during the bouncing motion, the VN-100 IMU drifts to a maximum value of 0.0035 rad (0.2005 deg) at 1.82 min and a minimum value of -0.0316 rad (-1.8105 deg) at 4.368 min; and the BNO055 IMU drifts to a minimum value of -0.1231 rad (-7.0531 deg) at 4.299 min. At the end of the bouncing motion, at $t = 4.47$ min, the VN-100 succeeds in reducing the drift to a value close to zero; the BNO055 IMU error touches the zero value and converges to a -4.537×10^{-3} rad (-0.26 deg) offset.

Figure 6.9 shows the filtered orientation error response about the z-axis. This plot shows that the three IMUs suffer a significant drift, the GX5 IMU performed better than the other two, but the final offset is still big. At this point, we do not have a technical explanation for this behavior; we believe that it could be related to the test apparatus or the environment.

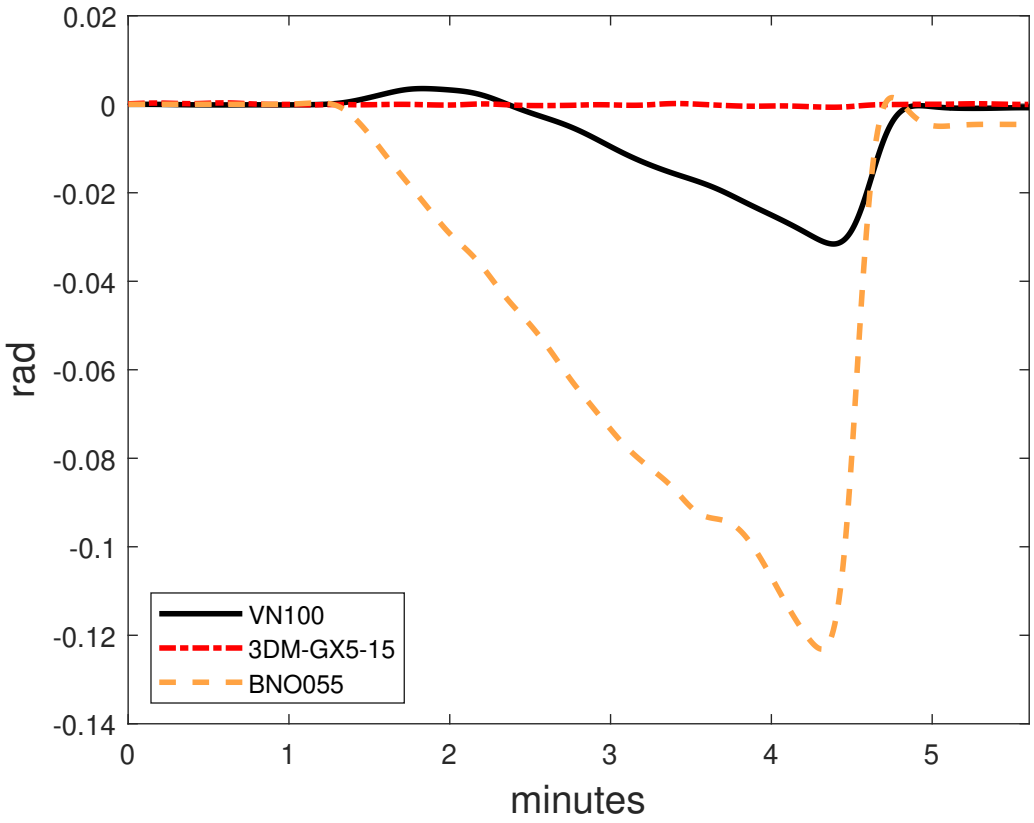

Figure 6.8 Filtered orientation error response about the x axis.

IMU	$\lvert E_x \rvert$		$\lvert E_y \rvert$		$\lvert E_z \rvert$		Resolution		Repeatability	
	(rad)	(deg)	(rad)	(deg)	(rad)	(deg)	(rad)	(deg)	(rad)	(deg)
VN100	0.0316	1.81	0.0190	1.09	0.0747	4.28	0.0009	0.05	0.0035	0.2
3DM-GX5-15	0.0007	0.04	0.0023	0.13	0.0127	0.73	0.0002	0.01	0.0035	0.2
BNO055	0.1231	7.06	0.2775	15.90	0.0606	3.47				

Table 6.1 Absolute value of the maximum error about each axis for the three IMUs. Errors in red denote that it surpassed the resolution and repeatability tolerances in the data sheet. Blank spaces denote that the information was not available.

Table 6.1 shows the absolute value of the maximum error about each axis for the three IMUs, where the values in red denote that it surpassed the resolution and repeatability tolerances given the IMU's data sheet. The minimum difference that the device can measure (resolution) and repeatability (the closeness of agreement among several consecutive measurements) tolerances were obtained from the sensor's datasheet for the VN100 and 3DM-GX5-15 IMUs. It was not possible to find these values for the BNO055 device. The results about the x and y axes show that the 3DM-GX5-15 IMU was the unique one capable

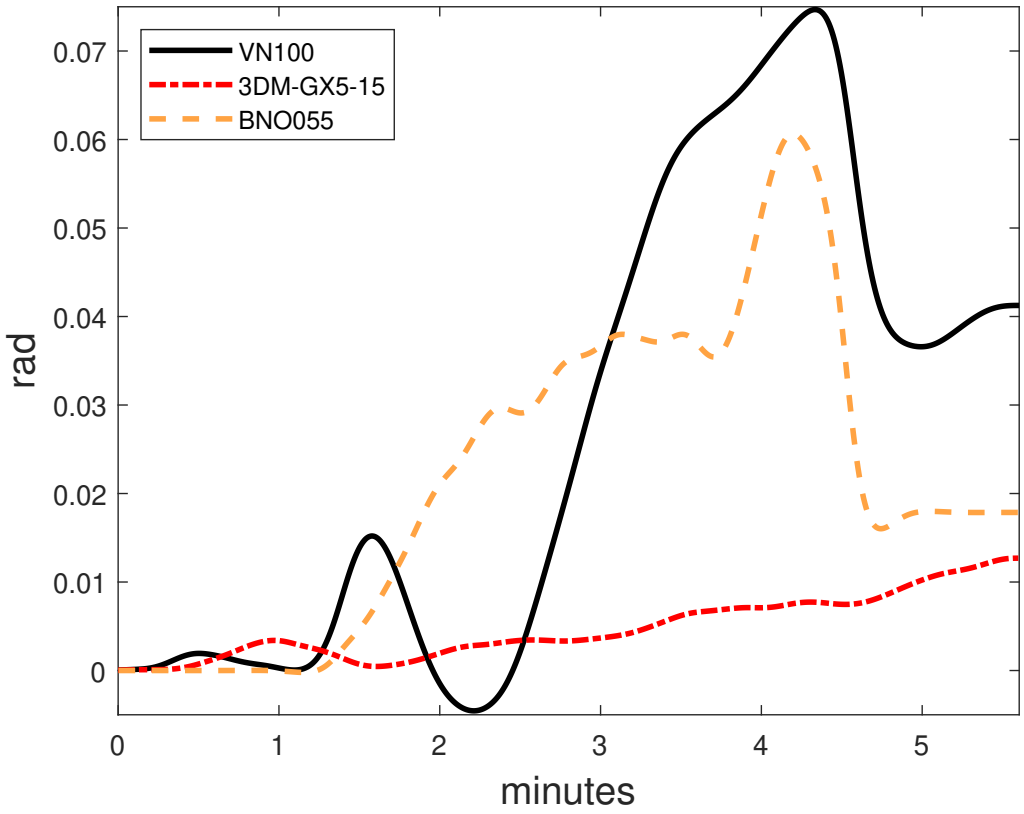

Figure 6.9 Filtered orientation error response about the z axis.

of having an absolute error below the repeatability range; the BNO055 IMU showed the largest errors.

6.4 Conclusion

This chapter aimed to analyze the estimation performance of the absolute orientation acquired from the IMUs subject to continuous low-intensity impacts (4 g maximum acceleration along the z-axis), similar to mounting the IMU at a running robot. According to the obtained results, all three IMUs suffered a certain amount of drift during the experiment. In summary, the IMU that had the worst performance was the BNO055, and it can be because of its low price, compared with the other two IMUs. On the other hand, the VN-100 demonstrated a better performance but experienced a drift during the bouncing motion. The GX5 IMU achieved the best performance during the experiment. Although, at the orientation estimation about the z-axis, we observed an unusual behavior on all the IMUs. In contrast with the other

two IMUs, the BNO055 IMU was the only one that couldn't recover the initial orientation after the bouncing motion had stopped; the maximum offset presented was about 0.5 deg that maybe into the precision range of the sensor. After doing a quick survey about the IMU prices in the market, we have found that the GX5 IMU costs about three times the price of the VN100 IMU, and the VN100 IMU costs about twenty-five times the BNO055 price. In this sense, there is a clear relationship between price and orientation precision.

Chapter 7

Conclusion

This thesis explored the feasibility of hopping, landing, and balancing with a springy leg monopod in the presence of uncertainties at the spring parameters. The key idea was to develop a balance controller capable of accurately controlling the robot in executing fast trajectory tracking; and an optimization tool for trajectory optimization applications.

Chapter two presented a strategy to accurately control a monoped robot during the launching motion to produce a leap. A DOC-Legendre-Gauss-Radau method was developed to obtain this motion profile. Then, a launch controller was designed based on the balance theory presented in Featherstone (2017) to produce the optimized motion on the robot. The presented strategy was compared with the approach introduced by Azad and Featherstone (2013) in terms of optimality and accuracy. In this context, it was demonstrated that by formulating the boundary value problem as a nonlinear programming problem, it is possible to find a quicker and smoother motion than the timed-reversal technique used in Azad and Featherstone (2013). And that it is also possible to design a controller for driving the robot accurately during the launching. The proposed controller succeeded in reproducing the desired motion satisfying the mechanical and torque constraints imposed during the launching optimization with a small error (0.13%) at the take-off instant.

The third chapter presented a numerical solution for finding motion profiles and spring parameters applied to a spring-loaded monopod robot at landing. The first section reviewed an analytical approach to obtain the spring parameters for a linear mass-spring-damper system subject to a gravityless mass impact scenario. The reviewed procedure aimed to reduce the maximum impact force considering the maximum spring displacement. The second section of this chapter introduced a numerical strategy to find the motion profile and spring parameters for a spring-loaded monopod robot that minimizes the peak force at landing; this was the main topic of this chapter. After running the optimization, the optimizer found the progressive

nonlinear spring optimal for this task rather than the regressive one. The nonlinear spring reached a slightly higher spring force, but the ground reaction force and actuator torque were lower than the linear one. The results in this chapter have demonstrated the complexity of the landing task facing a spring-loaded monoped robot. The results showed that the robot with a nonlinear progressive spring can increase the robot's ability to perform a soft landing with less electrical power. It's also showed the possibility of solving trajectory optimization and parameter optimization problems by using the orthogonal collocation methods implemented in conjunction with the Rigid Body Dynamics Algorithms (Featherstone, 2008).

Chapter four investigated the effect of a springy leg on the balance controller described in Featherstone (2017, 2018), which aimed to achieve and maintain balance in the presence of large, fast motions. The balance controller can cope with the large, fast motions that occur during landing, including small bounces, and tracking large, fast motion commands after landing with almost the same accuracy as could be achieved if the leg were rigid. It also showed that the balance controller could suppress vibrations caused by the spring. This chapter also showed the possibility of performing an absolute motion tracking, not only controlling the angle of the actuated joint q_3 but also the angle of the passive joint q_1 by introducing a state mapping for the balance controller given as (4.13). It also demonstrated that the varying-linear plant constructed by the controller does not need to know about the dynamics introduced by the spring. Although these dynamics are taken into account when calculating the system's states and mapping the controller's output to a joint torque or acceleration command.

The fifth chapter studied to what degree a springy leg may affect the performance of the balance controller (Gamba and Featherstone, 2021), combined with a HOSM observer (Moreno and Guzman, 2011) when a failure occurs in the spring-loaded device, which loses 50% of its stiffness due to a spring breakage. This strategy succeeds in keeping the robot's balance in the presence of large and fast motions, as well as parametric uncertainties. Some estimation drifts at the observer estimation did not significantly reduce the balance controller performance at landing, balancing, and tracking, which gives a sense of certain robustness to inaccuracies in the spring model.

Chapter six aimed to analyze the estimation performance of the absolute orientation acquired from the IMUs subject to continuous low-intensity impacts (4 g maximum acceleration along the z-axis), similar to mounting the IMU om a running robot. According to the obtained results, the three IMUs suffered a certain kind of drift during the experiment. In summary, the IMU that had the worst performance was the BNO055, which was the cheapest of the three IMUs tested. On the other hand, the VN-100 demonstrated a better

performance but experienced a drift during the bouncing motion. The GX5 IMU achieved the best performance during the experiment. Although, at the orientation estimation about the z-axis, we observed an unusual behavior on all threee IMUs. In contrast with the other two IMUs, the BNO055 IMU was the only one that couldn't recover the initial orientation after the bouncing motion had stopped; the maximum offset presented was about 0.5 deg, which may be into the precision range of the sensor.

Some proposed topics for further investigation and development involve:

- Complete the design and construction of the Skippy robot.
- Implement the balance controller in a real spring-loaded robot.
- Obtain a motion profile for a hop and execute with the launching controller on a real robot.
- Implement a parameter identification approach for estimating the plant parameters needed by the balance controllers to overcome uncertainties at dynamic parameters of the robot, like inertia, masses, and position of CoM.
- Design and validate through experiments a 3D balance controller for one-support point robots to achieve high-performance motions.
- Investigate the effect of a series elastic element in the actuated joint.
- Explore the extension of the balance controller to systems with compliant mechanisms.
- Perform the IMU hooping test for more aggressive hopping motions, maximum linear acceleration of 8 g or 16 g along the IMU's z-axis.

References

Adafruit (2021). Bno055 absolute orientation sensor. https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor. Last access. Oct. 26th 2021.

Adetola, V. and Guay, M. (2008). Finite-time parameter estimation in adaptive control of nonlinear systems. *IEEE Transactions on Automatic Control*, 53(3):807–811.

Ahmad, N., Ghazilla, R. A. B. R., Khairi, N. M., and Kasi, V. (2013). Reviews on various inertial measurement unit (imu) sensor applications. In *SiPS 2013*.

Aleksic-Veljkovic, A., Madić, D., Veličković, S., Herodek, K., and Popović, B. (2014). Balance in young gymnasts: Age-group differences. *Facta Universitatis Series Physical education and Sport*, 12:289–296.

Allione, F., A., B. R. P. S., Gkikakis, E., and Featherstone, R. (2021). Mechanical shock testing of incremental and absolute position encoders. In *2021 20th International Conference on Advanced Robotics*, pages 384–391.

Alsharif, O., Ouyang, T., Beaufays, F., Zhai, S., Breuel, T., and Schalkwyk, J. (2015). Long short term memory neural network for keyboard gesture decoding. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2076–2080.

Anand, A. S., Zhao, G., Roth, H., and Seyfarth, A. (2019). A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 537–543.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Ankaralı, M. M., Saranlı, U., and Saranlı, A. (2010). Control of Underactuated Planar Hexapedal Pronking through a Dynamically Embedded SLIP Monopod. In *2010 IEEE International Conference on Robotics and Automation*, pages 4721–4727, Anchorage AK, USA.

Arduino (2021). Arduino Uno Board. https://store.arduino.cc/products/arduino-uno-rev3. Last access. Oct. 3rd 2021.

Arslan, O., Saranli, U., and Morgul, O. (2009). An approximate stance map of the spring mass hopper with gravity correction for nonsymmetric locomotions. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2388 – 2393.

Azad, M. (2014). *Balancing and hopping motion control algorithms for an under-actuated robot*. PhD thesis, Australian National University. Research School of Engineering.

Azad, M. and Featherstone, R. (2013). Balancing and Hopping Motion of a Planar Hopper with One Actuator. In *2013 IEEE International Conference on Robotics and Automation*, pages 2027–2032, Karlsruhe, Germany.

Azad, M. and Featherstone, R. (2016). Angular Momentum based Balance Controller for an Under-actuated Planar Robot. *Autonomous Robot*, 40:93–107.

Batts, Z., Kim, J., and Yamane, K. (2017). Untethered One-Legged Hopping in 3D Using Linear Elastic Actuator in Parallel (LEAP). In *2016 International Symposium on Experimental Robotics*, pages 103–112, Tokyo, Japan.

Benallegue, A., Mokhtari, A., and Fridman, L. (2006). Feedback linearization and high order sliding mode observer for a quadrotor uav. In *International Workshop on Variable Structure Systems, 2006. VSS'06.*, pages 365–372.

Benson, L. C., Ahamed, N. U., Kobsar, D., and Ferber, R. (2019). New considerations for collecting biomechanical data using wearable sensors: Number of level runs to define a stable running pattern with a single imu. *Journal of Biomechanics*, 85:187–192.

Berkemeier, M. D. and Fearing, R. S. (1998). Sliding and Hopping Gaits for the Underactuated Acrobot. *IEEE Transactions on Robotics and Automation*, 14(4):629–634.

Berkemeier, M. D. and Fearing, R. S. (1999). Tracking Fast Inverted Trajectories of the Underactuated Acrobot. *IEEE Transactions on Robotics and Automation*, 15(4):740–750.

Biegler, L., Cervantes, A., and Wachter, A. (2002). Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57:575–593.

Biegler, L. T. (2010). *Nonlinear Programming*. Society for Industrial and Applied Mathematics.

Bolzon, G., Fedele, R., and Maier, G. (2002). Parameter identification of a cohesive crack model by kalman filter. *Computer Methods in Applied Mechanics and Engineering*, 191(25):2847–2871.

BOSCH (2021). BNO055. https://www.bosch-sensortec.com/products/smart-sensors/bno055/. Last access. Oct. 26th 2021.

Boubaker, O. and Iriarte, R. (2017). *The Inverted Pendulum in Control Theory and Robotics: From Theory to New Innovations*. Control, Robotics and Sensors. Institution of Engineering and Technology.

Briot, S. and Gautier, M. (2013). Global identification of joint drive gains and dynamic parameters of parallel robots. *Multibody System Dynamics*, 136.

Caron, F., Duflos, E., Pomorski, D., and Vanheeghe, P. (2006). Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects. *Information Fusion*, 7(2):221–230.

Cervantes, A. and Biegler, L. T. (2009). Optimization strategies for dynamic systems. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization, Second Edition*, pages 2847–2858. Springer.

Chanchareon, R., Sangveraphunsiri, V., and Chantranuwathana, S. (2006). Tracking control of an inverted pendulum using computed feedback linearization technique. In *2006 IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–6.

Chen, C.-T. (1998). *Linear System Theory and Design*. Oxford University Press, Inc., USA, 3rd edition.

Chen, H., Wensing, P. M., and Zhang, W. (2020). Optimal control of a differentially flat two-dimensional spring-loaded inverted pendulum model. *IEEE Robotics and Automation Letters*, 5(2):307–314.

Cho, H. T. and Jung, S. (2003). Balancing and position tracking control of an inverted pendulum on a x-y plane using decentralized neural networks. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 1, pages 181–186 vol.1.

Corigliano, A. and Mariani, S. (2004). Parameter identification in explicit structural dynamics: performance of the extended kalman filter. *Computer Methods in Applied Mechanics and Engineering*, 193(36):3807–3835.

Davila, J., Fridman, L., and Poznyak, A. (2006). Observation and Identification of Mechanical Systems via Second Order Sliding Modes. In *2006 International Workshop on Variable Structure Systems*, pages 232–237, Alghero, Italy.

DFRobot (2021). Gravity: Bno055+bmp280 intelligent 10dof ahrs. https://www.dfrobot.com/product-1793.html. Last access. Oct. 26th 2021.

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

Dierks, T. and Jagannathan, S. (2011). Online optimal control of nonlinear discrete-time systems using approximate dynamic programming. *Journal of Control Theory and Applications*, 9:361–369.

Dierks, T. and Jagannathan, S. (2012). Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1118–1129.

Dinev, T., Xin, S., Merkt, W., Ivan, V., and Vijayakumar, S. (2020). Modeling and control of a hybrid wheeled jumping robot. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Driessen, J. J. M., Gkikakis, A. E., Featherstone, R., and Singh, B. R. P. (2019). Experimental Demonstration of High-Performance Robotic Balancing. In *2019 International Conference on Robotics and Automation*, pages 9459–9465, Montreal, Canada.

Efimov, D., Fridman, L., Raïssi, T., Zolghadri, A., and Seydou, R. (2012). Interval estimation for lpv systems applying high order sliding mode techniques. *Automatica*, 48(9):2365–2371.

Fan, Y., Liu, S., and Belabbas, M.-A. (2020). Mid-air motion planning of robot using heat flow method with state constraints. *Mechatronics*, 66:102323.

Fang, A. C. and Pollard, N. S. (2003). Efficient synthesis of physically valid human motion. *ACM Trans. Graph.*, 22(3):417–426.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg.

Featherstone, R. (2015). Quantitative Measures of a Robot's Ability to Balance. In *Proceedings of Robotics: Science and Systems*, Rome, Italy.

Featherstone, R. (2016). Quantitative Measures of a Robot's Physical Ability to Balance. *The International Journal of Robotics Research*, 35(14):1681–1696.

Featherstone, R. (2017). A Simple Model of Balancing in the Plane and a Simple Preview Balance Controller. *The International Journal of Robotics Research*, 36(13-14):1489–1507.

Featherstone, R. (2018). A New Simple Model of Balancing in the Plane. In *Bicchi A. and Burgard W. (eds) Robotics Research (vol. 2), pp. 167–183, Springer Proceedings in Advanced Robotics, vol. 3, Springer, Cham.*

Featherstone, R. (2021). The Skippy Project. http://royfeatherstone.org/skippy. Last access: Aug. 30th 2021.

Featherstone, R. (2022). The Ring Screw Mechanism. http://royfeatherstone.org/ringscrew/index.html. Last access: Jan. 26th 2022.

Fevre, M., Wensing, P. M., and Schmiedeler, J. P. (2020). Rapid bipedal gait optimization in casadi. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 3672–3678. IEEE.

Finlayson, B. A. (1980). Orthogonal collocation on finite elements—progress and potential. *Mathematics and Computers in Simulation*, 22(1):11–17.

Frintrop, S., Rome, E., and Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. *ACM Trans. Appl. Percept.*, 7(1).

Gajamohan, M., Merz, M., Thommen, I., and D'Andrea, R. (2012). The Cubli: A Cube that can Jump up and Balance. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3722–3727, Vilamoura, Portugal.

Gamba, J. D. and Featherstone, R. (2021). Balancing on a Springy Leg. In *2021 IEEE International Conference on Robotics and Automation*, Xi'an, China.

Gamba, J. D., Leite, A. C., and Featherstone, R. (2021). Robust balancing control of a spring-legged robot based on a high-order sliding mode observer. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 384–391.

Gamba, J. D., Leite, A. C., and Featherstone, R. (2021). Robust Balancing Control of a Spring-legged Robot based on a High-order Sliding Mode Observer. https://youtu.be/OfSZMrSk-PA. Last access: Sep. 10th 2021.

Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T. (2010). Brief paper: A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851.

Garćıa-Alarćon, O., Puga-Guzman, S., and Moreno-Valenzuela, J. (2012). On parameter identification of the furuta pendulum. *Procedia Engineering*, 35:77–84. International Meeting of Electrical Engineering Research 2012.

González, C., Barasuol, V., Frigerio, M., Featherstone, R., Caldwell, D., and Semini, C. (2020). Line walking and balancing for legged robots with point feet. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3649–3656.

Green, P., Cross, E., and Worden, K. (2015). Bayesian system identification of dynamical systems using highly informative training data. *Mechanical Systems and Signal Processing*, 56-57:109–122.

Grizzle, J. W., Moog, C. H., and Chevallereau, C. (2005). Nonlinear Control of Mechanical Systems with an Unactuated Cyclic Variable. *IEEE Transactions on Automatic Control*, 50(5):559–576.

Hairer, E. and Wanner, G. (1996). *Implementation of Implicit Runge-Kutta Methods*, pages 118–130. Springer Berlin Heidelberg, Berlin, Heidelberg.

Haldane, D. W., Plecnik, M. M., Yim, J. K., and Fearing, R. (2016). Robotic Vertical Jumping Agility via Series-elastic Power Modulation. *Science Robotics*, 1:1–9.

Hale, N. and Townsend, A. (2013). Fast and accurate computation of gauss–legendre and gauss–jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 35(2):A652–A674.

Han, X. and Kong, X. (2010). The designing of serial communication based on rs232. In *2010 First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems*, pages 382–384.

Hars, M., Holvoet, P., Gillet, C., Barbier, F., and Lepoutre, F. X. (2005). Quantify dynamic balance control in balance beam: measure of 3-d forces applied by expert gymnasts to the beam. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(sup1):135–136.

Hereid, A. and Ames, A. D. (2017). Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726.

Hereid, A., Cousineau, E. A., Hubicki, C. M., and Ames, A. D. (2016a). 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In Kragic, D., Bicchi, A., and Luca, A. D., editors, *2016 IEEE*

International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016, pages 1447–1454. IEEE.

Hereid, A., Kolathaya, S., and Ames, A. D. (2016b). Online optimal gait generation for bipedal walking robots using legendre pseudospectral optimization. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6173–6179.

Hereid, A., Kolathaya, S., Jones, M. S., Van Why, J., Hurst, J. W., and Ames, A. D. (2014). Dynamic Multi-Domain Bipedal Walking with Atrias through SLIP Based Human-Inspired Control. In *17th International Conference on Hybrid Systems: Computation and Control*, pages 263–272, Berlin, Germany.

Hindmarsh, A. C., Gresho, P. M., and Griffiths, D. F. (1984). The stability of explicit euler time-integration for certain finite difference approximations of the multi-dimensional advection–diffusion equation. *International Journal for Numerical Methods in Fluids*, 4(9):853–897.

Ibrahim, M., Kallies, C., and Findeisen, R. (2020). Learning-supported approximated optimal control for autonomous vehicles in the presence of state dependent uncertainties. In *2020 European Control Conference (ECC)*, pages 338–343.

Itti, L., Dhavale, N., and Pighin, F. (2003). Realistic avatar eye and head animation using a neurobiological model of visual attention. In Bosacchi, B., Fogel, D. B., and Bezdek, J. C., editors, *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation VI*, volume 5200, pages 64 – 78. International Society for Optics and Photonics, SPIE.

Iwasaki, T. and Kataoka, T. (1989). Application of an extended kalman filter to parameter identification of an induction motor. In *Conference Record of the IEEE Industry Applications Society Annual Meeting,*, pages 248–253 vol.1.

Jiang, Y., Van Wouwe, T., De Groote, F., and Liu, C. K. (2019). Synthesis of biologically realistic human motion using joint torque actuation. *ACM Trans. Graph.*, 38(4).

Jin, B., Sun, C., Zhang, A., Liu, S., Hao, W., Deng, G., and Ma, P. (2017). Single leg compliance control for quadruped robots. In *2017 Chinese Automation Congress (CAC)*, pages 7624–7628.

Jovic, J., Escande, A., Ayusawa, K., Yoshida, E., Kheddar, A., and Venture, G. (2016). Humanoid and human inertia parameter identification using hierarchical optimization. *IEEE Transactions on Robotics*, 32(3):726–735.

Kajita, S. and Ott, C. (2016). Limbed systems. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 419–442. Springer.

Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Rev.*, 59(4):849–904.

Kim, K.-W., Ryu, Y., and Jeon, K.-K. (2012). A kinetics analysis of tucked backward salto on the balance beam. *Korean Journal of Sport Biomechanics*, 22.

Kloeser, D., Schoels, T., Sartor, T., Zanelli, A., Frison, G., and Diehl, M. (2020). NMPC for racing using a singularity-free path-parametric model with obstacle avoidance. In *Proceedings of the IFAC World Congress*.

Koenemann, J., Licitra, G., Alp, M., and Diehl, M. (2019). Openocl - open optimal control library.

Kollarčík, A. (2021). *Modeling and Control of Two-Legged Wheeled Robot*. PhD thesis, Czech Technical University in Prague. Faculty of Electrical Engineering, Department of Control Engineering.

Kreucher, C. and Lakshmanan, S. (1999). Lana: a lane extraction algorithm that uses frequency domain features. *IEEE Transactions on Robotics and Automation*, 15(2):343–350.

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

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455.

Kunkel, P., Mehrmann, V., Mehrmann, V., and Society, E. M. (2006). *Differential-algebraic Equations: Analysis and Numerical Solution*. EMS textbooks in mathematics. European Mathematical Society.

Lampariello, R., Mishra, H., Oumer, N., Schmidt, P., De Stefano, M., and Albu-Schäffer, A. (2018). Tracking control for the grasping of a tumbling satellite with a free-floating robot. *IEEE Robotics and Automation Letters*, 3(4):3638–3645.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, page 491–500, New York, NY, USA. Association for Computing Machinery.

Leiva, L. A., Martn-Albo, D., and Plamondon, R. (2017). The Kinematic Theory Produces Human-Like Stroke Gestures. *Interacting with Computers*, 29(4):552–565.

Lew, T., Emmei, T., Fan, D. D., Bartlett, T., Santamaria-Navarro, A., Thakker, R., and akbar Agha-mohammadi, A. (2019). Contact inertial odometry: Collisions are your friends.

Li, M., Jiang, Z., Wang, P., Sun, L., and Sam Ge, S. (2014). Control of a quadruped robot with bionic springy legs in trotting gait. *Journal of Bionic Engineering*, 11(2):188–198.

Li, W. and Wang, J. (2013). Effective adaptive kalman filter for mems-imu/magnetometers integrated attitude and heading reference systems. *Journal of Navigation*, 66(1):99–113.

Liang, Y. and Feeny, B. (2006). Parametric identification of a base-excited single pendulum. *Nonlinear Dynamics*, 46:17–29.

Liang, Y. and Feeny, B. (2008). Parametric identification of a chaotic base-excited double pendulum experiment. *Nonlinear Dynamics*, 52:1573–269X.

Lin, Y., McPhee, J., and Azad, N. L. (2021). Comparison of deep reinforcement learning and model predictive control for adaptive cruise control. *IEEE Transactions on Intelligent Vehicles*, 6(2):221–231.

LORD Sensing Microstrain (2021a). 3DM-GX5-15. https://www.microstrain.com/inertial-sensors/3dm-gx5-15. Last access. Oct. 26th 2021.

LORD Sensing Microstrain (2021b). Sensor Connect. https://www.microstrain.com/software/sensorconnect. Last access. Oct. 26th 2021.

Ma, W.-L., Hereid, A., Hubicki, C. M., and Ames, A. D. (2016). Efficient hzd gait generation for three-dimensional underactuated humanoid running. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5819–5825.

Ma, W.-L., Kolathaya, S., Ambrose, E. R., Hubicki, C. M., and Ames, A. D. (2017). Bipedal robotic running with durus-2d: Bridging the gap between theory and experiment. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, page 265–274, New York, NY, USA. Association for Computing Machinery.

Magnet-Schultz GmbH (2021). Proportional Rotary Solenoids Type G DR. https://www.magnet-schultz.com/en/rotary-solenoids/proportional-rotary-solenoids-type-g-dr/. Last access. Oct. 26th 2021.

Majkowska, A. and Faloutsos, P. (2007). Flipping with physics: Motion editing for acrobatics. SCA '07, page 35–44, Goslar, DEU. Eurographics Association.

Manns, P., Sreenivasa, M., Millard, M., and Mombaur, K. (2017). Motion optimization and parameter identification for a human and lower back exoskeleton model. *IEEE Robotics and Automation Letters*, 2(3):1564–1570.

Martin, W. C., Wu, A., and Geyer, H. (2015). Robust spring mass model running for a physical bipedal robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6307–6312.

McDonald-Maier, K., Glauche, V., Beckstein, C., and Blickhan, R. (2000). Controlling fast spring-legged locomotion with artificial neural networks. *Soft Computing*, 4:157–164.

McFarland, C. J. and Whitcomb, L. L. (2012). A new adaptive identifier for second-order rotational plants with applications to underwater vehicles. In *2012 Oceans*, pages 1–9.

Menache, A. (1999). *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Mirzaei, F. M. and Roumeliotis, S. I. (2008). A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143–1156.

Moreno, J. A. and Guzman, E. (2011). A New Recursive Finite-Time Convergent Parameter Estimation Algorithm. In *18th IFAC World Congress*, pages 3439–3444, Milan, Italy.

Moreno, J. A. and Osorio, M. (2012). Strict Lyapunov Functions for the Super-Twisting Algorithm. *IEEE Transactions on Automatic Control*, 57(4):1035–1040.

M'sirdi, N. K., Rabhi, A., Fridman, L., Davila, J., and Delanne, Y. (2006). Second Order sliding Mode Observer for Estimation of Velocities, Wheel Sleep, Radius and Stiffness. In *2006 American Control Conference*, pages 3316–3321, Minneapolis MN, USA.

Murata Manufacturing Co., Ltd. (2020). Murata BOY's Capabilities. http://www.murata.com/en-global/about/mboymgirl/mboy/capabilities. Last access. Oct. 3rd 2020.

Na, J., Ren, X., and Zheng, D. (2013). Adaptive control for nonlinear pure-feedback systems with high-order sliding mode observer. *IEEE Transactions on Neural Networks and Learning Systems*, 24(3):370–382.

Nassiraei, A. A., Masakado, S., Matsuo, T., Sonoda, T., Takahira, I., Fukushima, H., Murata, M., Ichikawa, K., Ishii, K., and Miki, T. (2006). Development of an artistic robot "jumping joe". In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1720–1725.

Nazarahari, M. and Rouhani, H. (2021). 40 years of sensor fusion for orientation tracking via magnetic and inertial measurement units: Methods, lessons learned, and future challenges. *Information Fusion*, 68:67–84.

Nguyen, Q., Powell, M. J., Katz, B., Carlo, J. D., and Kim, S. (2019). Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454.

NI (2021a). LabVIEW. https://www.ni.com/en-us/shop/labview.html. Last access. Oct. 26th 2021.

NI (2021b). sbRIO-9637. https://www.ni.com/en-us/support/model.sbrio-9637.html. Last access. Oct. 26th 2021.

Peters, D. A. (1997). Optimum spring-damper design for mass impact. *SIAM Review*, 39(1):118–122.

Pilloni, A., Pisano, A., and Usai, E. (2015). Observer-based air excess ratio control of a pem fuel cell system via high-order sliding mode. *IEEE Transactions on Industrial Electronics*, 62(8):5236–5246.

Piovan, G. and Byl, K. (2012). Enforced Symmetry of the Stance Phase for the Spring-Loaded Inverted Pendulum. In *2012 IEEE International Conference on Robotics and Automation*, pages 1908–1914, St. Paul, MA, USA.

Piovan, G. and Byl, K. (2013). Two-element control for the active slip model. In *2013 IEEE International Conference on Robotics and Automation*, pages 5656–5662.

Pololu (2021). Pololu G2 High-Power Motor Driver 24v21. https://www.pololu.com/product/2995. Last access. Oct. 26th 2021.

Polyakov, A. and Fridman, L. (2014). Stability Notions and Lyapunov Functions for sliding Mode Control Systems. *Journal of the Franklin Institute*, 351(4):1831–1865.

Pope, M. T., Christensen, S., Christensen, D., Simeonov, A., Imahara, G., and Niemeyer, G. (2018). Stickman: Towards a human scale acrobatic robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2134–2140.

Poulakakis, I. and Grizzle, J. W. (2007). Formal embedding of the Spring Loaded Inverted Pendulum in an Asymmetric Hopper. In *Proceedings of the 2007 European Control Conference*, pages 3159–3166, Kos, Greece.

Poznyak, A. (2010). *Advanced Mathematical Tools for Control Engineers: Volume 1: Deterministic Systems*. Advanced Mathematical Tools for Automatic Control Engineers. Elsevier Science.

Pugh, J. and Martinoli, A. (2007). Inspiring and modeling multi-robot search with particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 332–339.

Quinn, P. and Zhai, S. (2018). Modeling gesture-typing movements. *Human–Computer Interaction*, 33(3):234–280.

Quintero, S. A. P., Collins, G. E., and Hespanha, J. P. (2013). Flocking with fixed-wing uavs for distributed sensing: A stochastic optimal control approach. In *2013 American Control Conference*, pages 2025–2031.

Raibert, M. (1986). *Legged Robots That Balance*. MIT Press, Cambridge, MA, USA.

Reher, J., Hereid, A., Kolathaya, S., Hubicki, C. M., and Ames, A. (2016). Algorithmic foundations of realizing multi-contact locomotion on the humanoid robot durus. In *WAFR*.

Riese, S. and Seyfarth, A. (2012). Robustness and Efficiency of a Variable-leg-spring Hopper. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1347–1352, Rome, Italy.

Riva, M. H., Dagen, M., and Ortmaier, T. (2017). Comparison of covariance estimation using autocovariance ls method and adaptive srukf. In *2017 American Control Conference (ACC)*, pages 5780–5786.

Rizzello, G., Naso, D., Turchiano, B., and Seelecke, S. (2016). Robust position control of dielectric elastomer actuators based on lmi optimization. *IEEE Transactions on Control Systems Technology*, 24(6):1909–1921.

RLS (2021). AksIM-2™ Off-Axis Rotary Absolute Magnetic Encoder Module . https://www.rls.si/eng/aksim-2-off-axis-rotary-absolute-encoder. Last access. Oct. 26th 2021.

Roffel, A. and Narasimhan, S. (2014). Extended kalman filter for modal identification of structures equipped with a pendulum tuned mass damper. *Journal of Sound and Vibration*, 333(23):6038–6056.

Ronquillo-Lomeli, G., Ríos-Moreno, G. J., Gómez-Espinosa, A., Morales-Hernández, L. A., and Trejo-Perea, M. (2016). Nonlinear identification of inverted pendulum system using volterra polynomials. *Mechanics Based Design of Structures and Machines*, 44(1-2):5–15.

Ríos, H., Davila, J., and Fridman, L. (2012). High-order sliding mode observers for nonlinear autonomous switched systems with unknown inputs. *Journal of the Franklin Institute*, 349(10):2975–3002. The Collection of the Benjamin Franklin Laureates of 2007, 2008 and 2009.

Schmutz, A., Chèze, L., Jacques, J., and Martin, P. (2020). A method to estimate horse speed per stride from one imu with a machine learning method. *Sensors*, 20(2).

Schultz, G. and Mombaur, K. (2010). Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783–792.

Singh, B. R. P. (2021). *Angular Momentum based Balancing Control and Shock-proof Design of Legged Robots*. PhD thesis, Dept. Information Engineering, University of Pisa, Italy.

Spong, M. W. (1995). The Swing up Control Problem for the Acrobot. *IEEE Control Systems Magazine*, 15(1):49–55.

Tedaldi, D., Pretto, A., and Menegatti, E. (2014). A robust and easy to implement method for imu calibration without external equipments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3042–3049.

Terry, P., Piovan, G., and Byl, K. (2016). Towards precise control of hoppers: Using high order partial feedback linearization to control the hopping robot frank. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6669–6675.

Tian, D., Gao, J., Liu, C., and Shi, X. (2021). Simulation of upward jump control for one-legged robot based on qp optimization. *Sensors*, 21(5).

Todorov, E. and Jordan, M. I. (1998). Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80(2):696–714. PMID: 9705462.

Tokur, D., Grimmer, M., and Seyfarth, A. (2020). Review of balance recovery in response to external perturbations during daily activities. *Human Movement Science*, 69:102546.

Toussaint, M. (2009a). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1049–1056, New York, NY, USA. Association for Computing Machinery.

Toussaint, M. (2009b). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1049–1056, New York, NY, USA. Association for Computing Machinery.

Trefethen, L. N. (2012). *Approximation Theory and Approximation Practice (Other Titles in Applied Mathematics)*. Society for Industrial and Applied Mathematics, USA.

Unluturk, A., Aydogdu, O., and Guner, U. (2013). Design and pid control of two wheeled autonomous balance robot. In *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, pages 260–264.

Utkin, V., Poznyak, A., Orlov, Y., and Polyakov, A. (2020). Conventional and high order sliding mode control. *Journal of the Franklin Institute*, 357(15):10244–10261.

Vamvoudakis, K. G. and Lewis, F. L. (2010). Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888.

VectorNav (2021). VN-100. https://www.vectornav.com/products/detail/vn-100. Last access. Oct. 26th 2021.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., and Diehl, M. (2018). Towards a modular software package for embedded optimization. *IFAC-PapersOnLine*, 51(20):374–380. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

von Stryk, O. and Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Ann. Oper. Res.*, 37(1–4):357–373.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57.

Wai, R.-J. and Chang, L.-J. (2006). Adaptive stabilizing and tracking control for a nonlinear inverted-pendulum system via sliding-mode technique. *IEEE Transactions on Industrial Electronics*, 53(2):674–692.

Wensing, P. M. and Orin, D. E. (2013). High-speed Humanoid Running through Control with a 3D-SLIP Model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140, Tokyo, Japan.

Wieber, P., Tedrake, R., and Kuindersma, S. (2016). Modeling and Control of Legged Robots. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, Springer Handbooks, pages 1203–1234. Springer.

Wooten, W. and Hodgins, J. (2000). Simulating leaping, tumbling, landing and balancing humans. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 656–662 vol.1.

Xilinx (2021). ZYNQ. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Last access. Oct. 26th 2021.

Xinhua (2021). Chinese gymnasts finish 1-2 in women's balance beam at Tokyo Olympics. //www.xinhuanet.com/english/2021-08/03/c_1310105152_6.htm. Last access: Oct. 18th 2021.

Xiong, X. and Ames, A. (2018). Bipedal Hopping: Reduced-order Model Embedding via Optimization-based Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3821–3828, Madrid, Spain.

Xiong, X. and Ames, A. (2020). Sequential motion planning for bipedal somersault via flywheel slip and momentum transmission with task space control.

Xue, T., Zhao, J., and Wang, J. (2017). Motion control for variable stiffness slip model of legged robot single leg. In *2017 Chinese Automation Congress (CAC)*, pages 4711–4716.

Yim, J. K., Singh, B. R. P., Wang, E. K., Featherstone, R., and Fearing, R. S. (2020). Precision Robotic Leaping and Landing Using Stance-Phase Balance. *IEEE Robotics and Automation Letters*, 5(2):3422–3429.

Yim, J. K., Singh, B. R. P., Wang, E. K., Featherstone, R., and Fearing, R. S. (2020). Precision robotic leaping and landing using stance-phase balance. *IEEE Robotics Autom. Lett.*, 5(2):3422–3429.

Zhang, L., Ren, X., and Guo, Q. (2020). Balance control of a wheeled hopping robot. In *2020 39th Chinese Control Conference (CCC)*, pages 3801–3805.

Zhang, W., Liu, J., Cho, C., and Han, X. (2015). A hybrid parameter identification method based on bayesian approach and interval analysis for uncertain structures. *Mechanical Systems and Signal Processing*, 60-61:853–865.

Zhang, Y., wei Gong, D., and hua Zhang, J. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, 103:172–185.

Zhao, H., Horn, J., Reher, J., Paredes, V., and Ames, A. D. (2016). Multicontact locomotion on transemoral prostheses via hybrid system models and optimization-based control. *IEEE Transactions on Automation Science and Engineering*, 13(2):502–513.

Appendix A

Linear Spring Damper Design For Mass Impact

Here, we introduce a strategy to design a linear spring-damper machine capable of minimizing the maximum force at a mass impact after a plastic collision. The optimization exercise presented in Peters (1997) seeks to obtain the stiffness K and damper D coefficients of a linear spring-damper system. The author uses a dimensionless approach to optimize one dimensionless variable ζ instead of the two coefficients K and D. The author presents a setup that intentionally neglects the presence of gravity in order to construct a second-order dimensionless homogeneous differential equation. Then, he obtains the optimal results based on an analytical solution of the dimensionless equation.

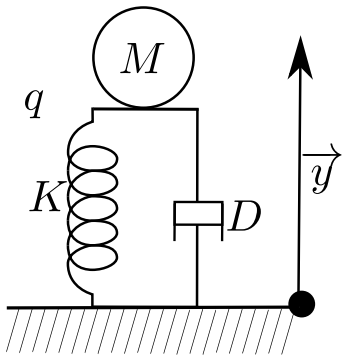

Figure A.1 Mass-spring-damper system.

The mechanism of Fig.A.1 consists of a Mass (M) of 2 kg connected to a spring-damper system with stiffness K and damping D coefficients. The whole structure is clamped to the ground without considering the gravity force g. The position of M along the y axis to the ground is denoted by q. Its velocity is $\dot{q}$ and acceleration is $\ddot{q}$, which is obtained by the

following equation:

$$\ddot{q} = -\frac{K}{M}q - \frac{D}{M}\dot{q} \tag{A.1}$$

Peters (1997) presents an approach to obtain the stiffness K and damping D coefficients from minimizing the maximum force F_{max} at the impact. With this method, it is also possible to limit the impact force F by constraining the maximum displacement $|q| < q_{max}$ of the spring. The author uses the concept of nondimensional quantities to reduce the system's dynamic equation (eq. A.1), and achieve the parameter search by only finding one variable $\zeta = \dfrac{D}{2\sqrt{KM}}$, which is nondimensional. Then, he solves the reduced system as a second-order homogeneous differential equation for $\zeta < 1, \zeta = 1$ and $\zeta > 1$, these operation regions of ζ are later analytically reduced to $0 \leq \zeta \leq 1/2$, leading to the optimal value of $\zeta = 0.40397275$.

Finally, K and D can be obtained by:

$$K = 0.36057021\,\frac{\dot{q}_0^2 M}{q_{max}^2} \qquad D = 0.48515107\,\frac{\dot{q}_0 M}{q_{max}} \tag{A.2}$$

Where $\dot{q}_0$ denotes the initial velocity, and q_{max} is the maximum spring displacement. The maximum impact force F_{max} can also be obtained by,

$$F_{max} = .52059862\,\frac{M\dot{q}_0^2}{q_{max}} \tag{A.3}$$

likewise,

$$q_{max} = .52059862\,\frac{M\dot{q}_0^2}{F_{max}}. \tag{A.4}$$

For a mass $M = 2\,\mathrm{kg}$ travelling at $\dot{q}_0 = -6\,\mathrm{m/s}$, and a maximum spring displacement of $q_{max} = 0.2\,\mathrm{m}$ we have the following spring parameters,

$$D = 29.1091\,\mathrm{Ns/m} \qquad K = 649.0264\,\mathrm{N/m} \tag{A.5}$$

Using the parameters obtained from equation (A.5), we have constructed a numerical simulation using the ODE45 solver of Matlab with its default parameters to verify the optimal parameters. Figure A.2 demonstrates the mass's position q and velocity $\dot{q}$ during the impact. In the position graph, the spring starts at position zero and finishes at $0.3223\,\mathrm{mm}$, and is moving towards zero. During the motion it reaches a minimum peak of $-0.2\,\mathrm{m}$ and positive peak of $49.9\,\mathrm{mm}$. At the velocity graph, the spring starts with a velocity of $-6\,\mathrm{m/s}$ and finishes at $-27.3521\,\mathrm{mm/s}$. During the landing it reaches a maximum velocity of $2.163\,\mathrm{m/s}$.

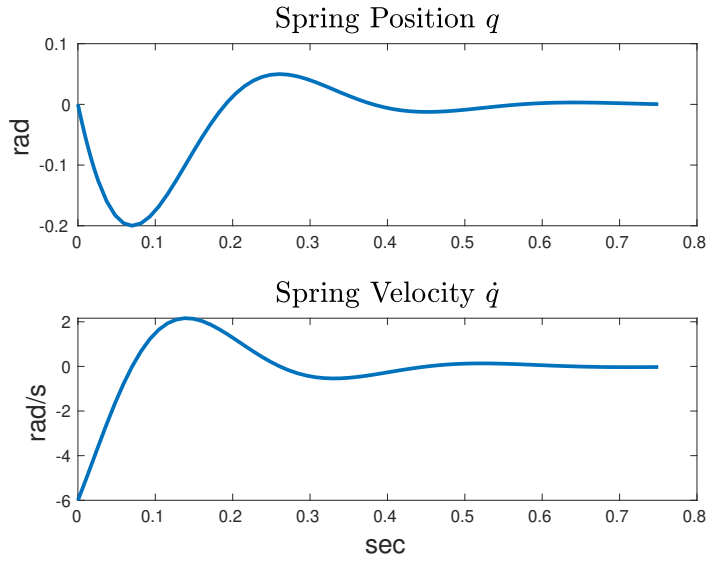

Figure A.2 q and $\dot{q}$ response of fig.A.1 system with a linear spring-damper for an impact of a mass of 2 kg travelling at $-6\,\mathrm{m/s^2}$.

As expected, there will always be a positive excursion in the absence of gravity, as shown in the position graph. After the maximum displacement q_{max} is reached, the recovery is approximately one-fourth of the initial displacement produced by the spring. Consequently, figure A.3 shows a recovering force produced by the spring after achieving the maximum force F_{max}. At the beginning of the impact the force starts at 174.6544 N and finishes at zero, due to the lack of gravity. The mass experienced a maximum force of 187.3914 N.

The results presented in this section give an idea about the minimum and maximum force that can be experienced by a mass of two kilograms subject to an impact (landing after a plastic collision) at $-6\,\mathrm{m/s}$ using an optimal spring-damper system. The next section aims to improve the obtained results by using a spring-loaded monopod robot considering the gravitational force. It will also demonstrate a more complex strategy to achieve the landing and find the optimal spring results.

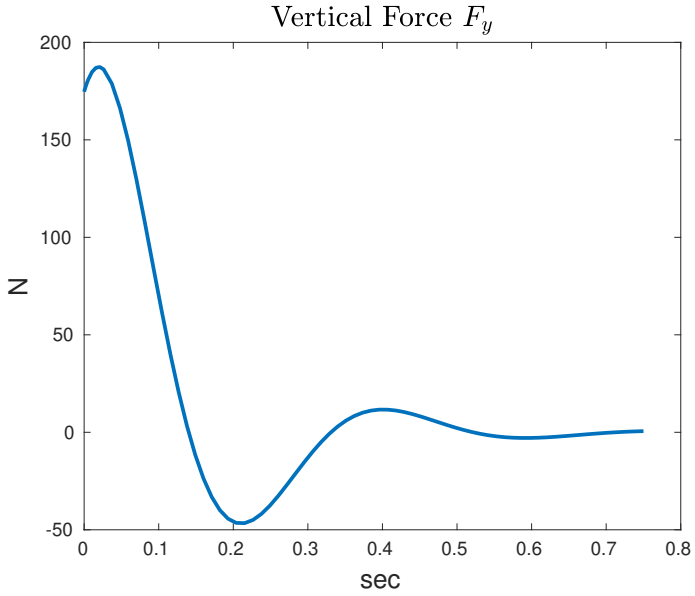

Figure A.3 Vertical force F response of fig.A.1 system with a linear spring-damper for an impact of a mass of 2 kg travelling at $-6\,\mathrm{m/s^2}$.