Design, Modelling and Control of a Rotorcraft Landing Gear for Uneven Ground Conditions

by

Daniel Melia Boix

A thesis submitted in partial fulfilment of the requirements

of Edinburgh Napier University, for the award of

Doctor of Philosophy

School of engineering and the built environment

Edinburgh Napier University

August 2019

Abstract

The ability to perform vertical take-off and landing, hovering and lateral flight provides rotorcrafts crucial advantages over other aircrafts and land vehicles for operations in remote areas. However, a major limitation of rotorcrafts is the requirement of a flat surface to land, increasing the difficulty and risk of landing operations on rough terrain or unstable surfaces. This limitation is mainly due to the use of conventional landing gear like skids or wheels. The growing use of Unmanned Aerial Vehicles (UAVs) also increases the necessity for more landing autonomy of these systems.

This thesis presents the investigation into the development of an adaptive robotic landing gear for a small UAV that enhances the landing capabilities of current rotorcrafts. This landing gear consists in a legged system that is able to sense and adapt the position of its legs to the terrain conditions.

This research covers the development of effective tools for the design and testing of the control system using software and hardware platforms. Mathematical models using multibody system dynamics are developed and implemented in software simulations. A hardware robot is designed and built to validate the simulation results.

The system proposed in this thesis consists in a landing gear with four robotic legs that uses an Inertial Measurement Unit (IMU) to sense the body attitude, Force Sensing Resistors (FSR) to measure feet pressure and a distance sensor to detect ground approach. The actuators used are position-controlled servo motors that also provide angular position feedback. The control strategy provides position commands to coordinate the motion of all joints based on attitude and foot pressure information. It offers the advantage of being position-controlled, so it is easier to be implemented in hardware systems using low-cost components, and at the same time, the feet force-control and leg design add compliance to the system.

Through software simulations and laboratory experiments the system successfully landed on a 20° slope surface, substantially increasing the current slope landing limit.

Acknowledgements

First of all, I would like to thank my director of studies, Dr Keng Goh for giving me the opportunity to study a PhD at Napier University. He has been an important source of advice, guidance and support throughout my doctoral studies while giving me freedom to work in my own way. I would also like to thank my co-supervisor, James McWhinnie, for his support and advice throughout the course of this project. Previously I was lucky to work with them during my MSC in Automation and Control where they introduced me to the field of robotics and control engineering.

Special thanks to all Napier staff who has helped me in many ways during this time, especially to Bill for his help with the 3D printers and for allowing us to invade his space in the lab for the practical tests.

I would also like to acknowledge the School of Engineering and Built Environment for the support and funding without which this project wouldn't have been possible.

Finally, I would like to thank my family and friend whose support and guidance throughout my life has enabled me to be here.

Declaration

I declare that the work submitted in this dissertation is the result of my own investigation, except where otherwise stated, and notion taken from other sources of information have been properly quoted as such. Also, I declare that this dissertation has not previously been presented in identical or similar form to any other degree of professional qualification.

The thesis work was conducted from January 2016 to August 2019 under the supervision of Dr. Keng Goh and James McWhinnie at the School of Engineering and Built Environment of Edinburgh Napier University.

Edinburgh, August 2019

Daniel Melia

Contents

List of Figures

List of Tables

Acronyms

CAD	Computer Aided Design
CoM	Centre of Mass
DoF	Degree of Freedom
FSR	Force Sensing Resistor
HOSM	Higher Order Sliding Mode
IDE	Integral Development Environment
IK	Inverse Kinematics
IMU	Inertial Measurement Unit
PD	Proportional Derivative
PID	Proportional Integral Derivative
SMC	Sliding Mode Control
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing

1 Introduction

1.1 Background and Motivation

A rotorcraft or rotatory-wing aircraft is a flying machine that produces lift by means of rotating blades. The main advantage of rotorcrafts with respect to fixed-wing aircrafts is their ability to provide lift without the need of moving forward, allowing them to perform vertical take-off and landing (VTOL) operations, thus avoiding the use of runways [1]. Rotorcrafts can also perform hover in one area and fly forward/backward and laterally. All these characteristics make rotorcrafts the best option for specific operations that are not possible to be carried out by other types of aircrafts or are inefficient to conduct by ground. For these reasons they are widely used to transport goods or people in hard-to-reach areas like mountains, disaster areas, offshore platforms or ships.

The most popular type of rotorcraft is the helicopter since, during the 1940's, the first large-scale models began to be mass-produced. Since then, many attempts have been made to reduce its size and realize autonomous flights, giving birth to the first miniature rotorcraft Unmanned Aerial Vehicles (UAV) [2]. During the 70's and 80's the industry of radio-controlled hobby helicopters developed and gave way to more sophisticated and powerful avionic systems integrating processing units, sensors and wireless communication. During the 90's the appearance of modern rotorcraft UAVs attracted the attention of many research projects and academic communities, and they developed aerial robots capable of self-stabilizing, self-navigating and interacting with its surroundings. In more recent years, the use of UAVs has increased rapidly, not only in research and military projects, but also for civilian and industrial applications. Nowadays, they are being used for tasks such as surveying areas and assessing damage after a natural disaster, in monitoring and inspection tasks in industrial sectors like agriculture, mining, oil & gas, and construction, for deliveries to remote locations and in military operations [3]. In the near future, this application range is expected to

increase continually and UAVs to migrate from passive tasks like inspection or surveillance, into active tasks like grasping and manipulation [4].

Although, the use of rotorcraft vehicles has improved the performance and efficiency of many tasks, it still has one major limitation: they require a reasonably flat surface to land. In many environments where rotorcrafts operate, stable flat surfaces are often unavailable. This can lead to dangerous situations where a helicopter has to land on an unstable ship deck, a rough terrain, or a sloped hillside with the risk of entering dynamic rollover, a condition where the helicopter starts to pivot around a skid without sufficient control authority to recover stability [1]. In other cases, landing is not possible and the helicopter has to hover, which increases the power consumption and limits the time and range of the task that the rotorcraft can perform.

The limitation on the landing capabilities is mainly due to the conventional landing gear construction which can be divided in two main categories: skids or wheels. Skids are used in light helicopters to save weight and cost and for its simplicity, while heavier helicopters use wheels as they offer better ground handling.

In the case of the skids, the landing surface is composed of two parallel lines while the wheel type consists of 3 points on the same plane. They are stable when the landing surface is flat but have difficulties to find a stable solution on rough terrains. In the case of slope landings, for most helicopters a maximum slope of 5° is considered for normal operations, and between 5° and 8°, depending on the model, there is a risk to enter into dynamic rollover [1]. For dynamic rollover to occur, some factor has to make the helicopter roll or pivot around a skid or wheel. This is common in slope operations as the helicopter pivots around the uphill skid until the landing or take-off is completed. Some factors like crosswinds, hovering sideward at the moment of ground contact, or part of the landing gear being stuck, can produce an excessive rolling moment. If critical rollover angle is reached, the pilot doesn’t have enough control authority to recover and the helicopter would fall over. Slope landings and takeoffs are complicated manoeuvres and a coordinated action of controls must be used to decrease the lift force while tilting the main rotor disk in the direction of the slope [1].

A considerable amount of research has been carried out in the area of Autonomous Landing Systems, although most of this research focuses on finding the right place to land rather than adapting the landing gear to the ground conditions [5] [6]. These

systems provide the vehicles with different instrumentation such as vision-based or LIDAR systems, and control systems to give them the capability of trajectory generation, path planning and obstacle avoidance, and terrain identification in order to identify a suitable landing area [7].

The other approach to the problem of safe landing which has been less explored, is to make the helicopter adaptable to different kind of terrains and uneven surfaces by modifying the conventional landing gear system. Several early landing gear systems capable to tilt laterally were suggested, but this ideas were never fully developed, and only very recently this area has attracted the interest of some research work and several new designs are being developed. These systems are reviewed in Chapter 2.

1.2 Aim and Objectives

The aim of this thesis is to investigate how robotic landing gear systems can improve the landing capabilities and extend the operational range of current rotorcrafts, allowing such vehicles to land on uneven terrains, including irregular surfaces with obstacles, steps or slopes where they could not land using conventional landing gear.

To achieve this aim, the following research objectives have been set:

- To investigate the limitations of current landing gear systems.
- To develop mathematical models for implementation in software simulation to analyse the behaviour of the system and to test control algorithms before implementing them into a physical prototype.
- Investigate the development of appropriate control algorithms to perform landings on uneven terrains in a safe way, maintaining the main body of the vehicle in a stable position. Investigate the necessary sensing and controller design to achieve this goal.
- The design and construction of a robotic landing gear prototype that can sense and adapt the position of its legs to the terrain conditions using available sensing information. The design includes the mechanical system and leg design, electrical system and sensors, and development of algorithms and control system.
- Development of effective tools for testing and analysing the performance of the hardware platform in the laboratory environment.

1.3 Approach

Two main methodologies are used to achieve this task, which are the use of software simulations and the design of a physical prototype to test the proposed design and to develop and implement the control algorithms.

1.3.1 Modelling and software simulations

Multibody system dynamics are used to develop a mathematical model of the system to implement it in a software environment. Software simulations serve as a simplified environment to test the core ideas in a simpler way before building a hardware robot, thus reducing the number of redesigns and potential damage due to unexpected system behaviour.

Mathematic modelling techniques of robotic legged systems are reviewed. Dynamic and Kinematic models are developed using Newton-Euler and Lagrange methodologies, which are the most common techniques to obtain the equations of motion of the system. Although both methods produce the same equations, the formulation and development of these equations is quite different. While with Newton-Euler, the final equations are more tedious to construct and there are more steps involved, the Lagrange method involves solving challenging differential-algebraic equations [8]. The equations of motion of multibody systems are complex to solve because of the internal constraints, and the selection of one or other method will depend on the system to solve.

Ground contact models to simulate the ground interaction forces are also reviewed and developed. Models are integrated and implemented in Simulink for simulation purposes and to serve as a platform to design and test the control system. SimMechanics, a package for multibody dynamics simulations, has also been used to validate the results of the final models.

1.3.2 Physical prototype

A hardware robot is used to validate the simulation results. While software simulations provide more flexibility to test different control strategies without the risk of damaging physical components, hardware prototype testing provides the opportunity to validate these control algorithms in real situations. However, there are several limitations and challenges when implementing the control solutions in a prototype, like limitations in

the actuators specifications and functionality, availability of sensory information, sensors imprecisions and noise, computational power or robot's weight. These constraints imposed by the hardware system are taken into account when designing the control architecture.

One crucial factor on the controller design is the actuator technology. The prototype designed during this thesis consists in a robotic landing gear for a small UAV. Size and budget limitations conditioned the use of position-controlled motors, as torque-controlled motors and accurate torque sensors are expensive and more difficult to integrate in small systems. Besides, torque-control strategies are usually more complex and require high computational power.

From the control point of view, the system is not intended to be an autonomous landing system. The operation of the rotorcraft is assumed to be performed by an independent controller or a pilot who will control parameters like the orientation, direction or landing velocity of the rotorcraft. The robotic landing gear control will sense and adapt its legs to the landing surface and will work completely autonomously without any input from the helicopter control system or the pilot.

For the landing gear to be able to level the aircraft, first it needs to be able to sense its orientation. The sensor selected for that purpose is an Inertial Measurement Unit (IMU) as it is commonly used, low-cost sensor with well-known algorithms to calculate the orientation of a body. Feet force sensors were selected to be able not only to detect ground contact, but also to make sure that the pressure supported by each leg within an accepted range and well distributed. To detect when the helicopter is approaching to land, a distance sensor has been added so the landing gear can prepare for landing position.

For the mechanical design, prototyping techniques like CAD design and 3D printing manufacture have been used as they provide a quick and economic way to design and manufacture system components. The microcontroller platform used during the project has been the Arduino as it provides an open-source hardware and software with a large user community and compatible sensors and actuators. The prototype is used to run extensive tests to assess the robot's capabilities and performance.

1.4 Hypothesis and Contributions

The hypothesis of this thesis is that robotic landing gear can be used to increase the landing capabilities of rotorcraft vehicles. The literature review in Chapter 2 provides evidence of the research gap in this field as there are a small number of systems that use robotic landing gear. Also these works are very recent and detailed study of the performance of these systems is limited.

This thesis contributes to cover this research gap by analysing the limitations of current systems, proposing new control algorithms and designs, and developing tools to model and test the performance of the system.

The main contributions of this thesis are the following:

- Design and construction of a robotic legged landing gear that allows rotorcrafts to land on uneven terrains and overcome objects, steps and slopes in the ground. A robotic platform has been built for a small UAV using off-the-shelf and 3D-printed materials to replace the conventional skid-type landing gear.
- Development and implementation of posture control algorithms based on position and force control, to keep the helicopter body level during the whole landing operation. The control approach takes into account the hardware limitations and uses as little sensory information and computational power as possible.
- Development of laboratory test to assess the robot's performance. The platform is tested in our laboratory facilities to simulate landings on different ground configurations, with different control algorithms and results are analysed.
- Development of mathematical models of the system and landing scenario, including ground contact models. Software implementation of these models that allows for testing of different configurations and control algorithms. This model built in Simulink provides a quick and safe way to test the control algorithms that later will be validated on the robot platform.

1.5 Publications

- K. Goh, D. Melia Boix, J. McWhinnie and G. Smith, “Control of Rotorcraft Landing Gear on Different Ground Conditions”, in IEEE International Conference on Mechatronics and Automation (ICMA), Harbin, China, 2016.
- D. Melia Boix, K. Goh and J. McWhinnie, “Modelling and control of helicopter robotic landing gear for uneven ground conditions”, in Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linkoping, Sweden, 2017.
- D. Melia Boix, K. Goh and J. McWhinnie, “Helicopter Lands on Uneven Terrain by means of Articulated Robotic Legs-Modelling, Simulation and Control Approach”, in IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand, 2018.

1.6 Outline of Thesis

This thesis is organized as follows: Chapter 1 presents the motivation, objectives and contribution of this research work. Chapter 2 reviews the state of the art of adaptive landing systems and legged robots and of modelling and control techniques. Chapter 3 presents the methodology used to model the dynamics of the system as a floating base multi-body system. This chapter also presents the model for all the external forces including ground reaction forces and helicopter thrust force. Chapter 4 describes the prototype design including the mechanical design, the sensory system, actuators and electrical equipment, and the control software. Chapter 5 discusses the control system for the robotic landing gear which can be divided into low-level joint controllers and high-level posture controllers. The high level controller uses combined foot pressure and body attitude information to provide position commands to each joint controller. In this chapter a PD feedback law is used for the high-level controller and simulation results are presented. An alternative controller using sliding mode control it’s also proposed and tested in simulations. Chapter 6 presents the results of the laboratory experiments using the robotic landing gear platform with different control algorithms in different scenarios. Chapter 7 and 8 end this thesis presenting the discussion, conclusions and future work.

2 Literature Review

2.1 Adaptive Landing Gear Systems

The question of providing rotorcrafts with an adaptable landing gear that enhances the vehicle's landing capabilities is not new and several patents and designs already exist as an alternative to the conventional landing gear designs.

The most common landing gear is the skid type because of their design simplicity, they are light and require little maintenance. On larger, heavier helicopters, however, wheels can be preferable when utility and convenience are more important than weight savings, as wheels offer better ground handling capabilities like moving the helicopter on the ground. The possibility to add retractable wheels also reduces air drag, allowing for greater speeds and fuel savings for long distances, at the cost of adding additional equipment and complexity to the design. The landing gear with wheels usually have better damping properties reducing the impact forces on the helicopter fuselage during hard landings.

Another important aspect to take into consideration is the capacity of the system to absorb impacts and to deal with hard landing mitigation which can cause from simple passenger discomfort to serious vehicle and cargo damage, injuries and even possible loss of life [9]. To deal with this problem, several hard landing mitigation technologies have been developed, including redesign of aircraft seats, subfloor and landing gear to attenuate the impact force. In the case of the landing gear, these technologies include the use of materials with elasto-plastic properties that can dissipate impact energy through the plastic deformation, or the implementation of supplementary devices such oleo-pneumatic shock absorbers. Additionally, several impact mitigation methods have been deployed including external airbags, collapsible plastic and metallic structures and supplementary systems to add to the conventional landing

gear, but these solutions are difficult to implement, not reusable, add significant weight to the rotorcraft, and offer no additional benefits apart from improved crash dynamics.

To cope with the problem of landing on sloped surfaces, several patents were registered as early as in the 1960's. These designs proposed the use of different hydraulic or mechanical systems to adapt the position of the landing gear to the irregularities of the terrain. In [10] and [11], the principle of operation of the mechanism consists in a hydraulic system (pistons, valves, oil/fluid) connecting both sides of the landing gear as shown in Figure 2-1. When one of the legs/struts touches the ground, the increasing pressure pushes the fluid out from the up-slope leg reducing the extension of that leg. The fluid is pushed into the down-slope leg which increases its extension due to the pressure. That way the main body of the rotorcraft is maintained in the right attitude.

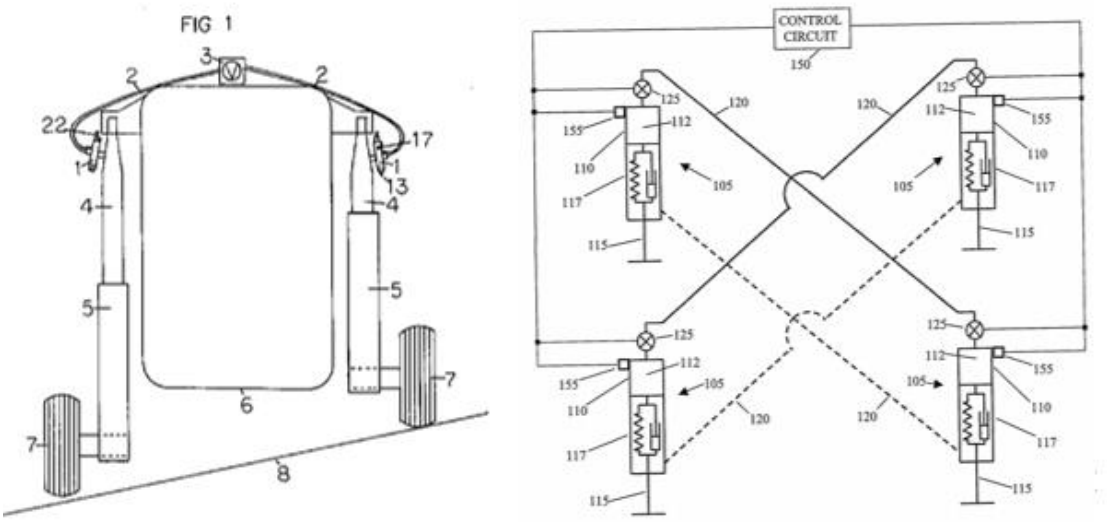

Figure 2-1 Slope landing compensator systems in [10] (left) and [11] (right)

In [12], a self-levelling landing mechanism is presented consisting of two curved track members lying in parallel planes normal to the longitudinal axis of the helicopter and passing through a base frame of the helicopter. This base frame incorporates guides that allow the movement of the curved tracks through the guides and varies the distance of their opposite ends from the ground. The suggested system is shown in Figure 2-2 .The system locks the position of the curved tracks in the base frame after landing of the helicopter on uneven terrain. While landing, the curbed tracks are free to move so the pilot will simply have to maintain the helicopter level, and the far ends of the curved tracks and skids will adjust to the level of the terrain.

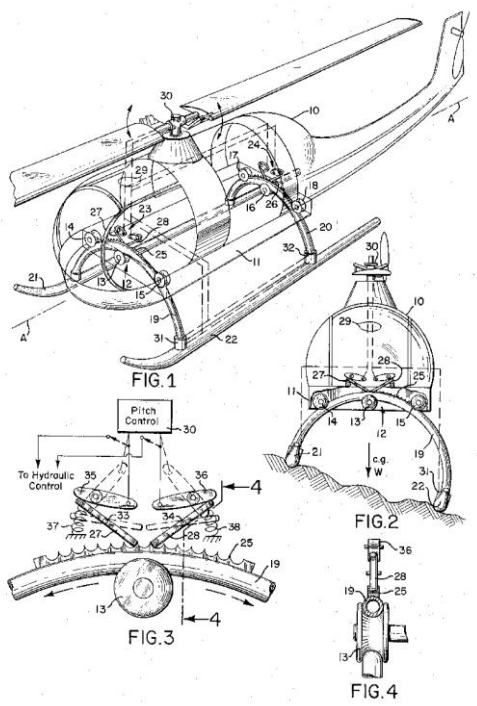

Figure 2-2 Helicopter self-levelling landing gear [12]

Another type of adjustable landing mechanism focuses on the design of telescopic legs for helicopters [13] and UAVs [14], as shown in Figure 2-3 respectively. In these systems, the legs can extend/retract by sliding one section into the other and can be driven by hydraulic, pneumatic or electric actuators. A central processing unit would be used to control the extension of each and different kind of sensors may be used to provide data to the central processing unit.

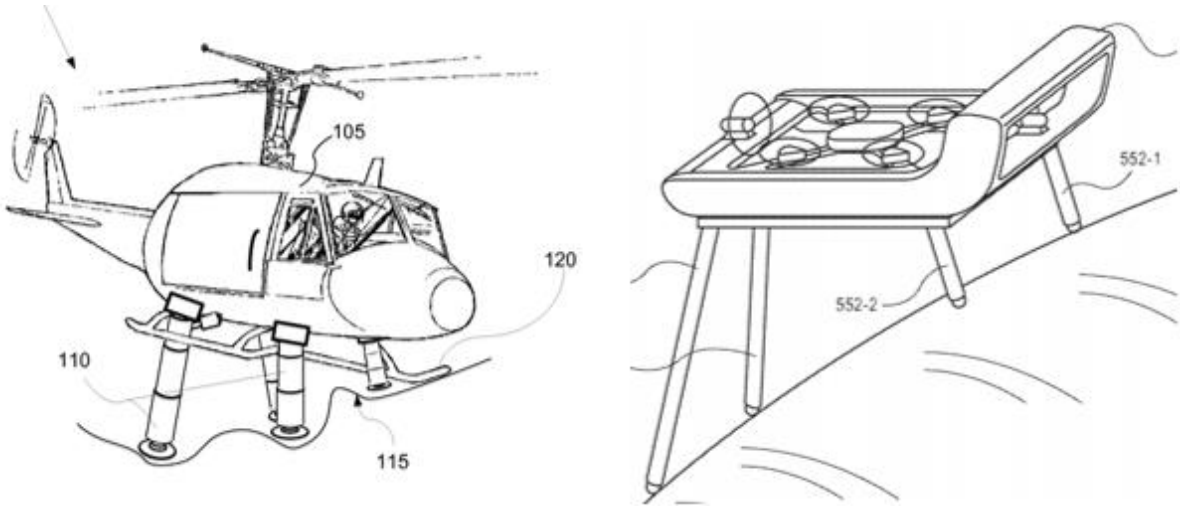

Figure 2-3 Telescopic landing gear system for helicopter [13] (left) and UAV [14] (right)

All these systems are patented but none of them are known to have been constructed or tested.

The most relevant work that has been reported on this field is a robotic landing gear developed by the Georgia Institute of Technology under funding from DARPA [15]. This landing gear consists of four articulated robotic legs with two joints on each leg actuated by one electric motor at each joint, as shown in Figure 2-4. Pressure pads are included at each foot to detect ground contact and to sense how much pressure they are exerting on the ground. A microprocessor uses information from the pressure pads to command the joints of each leg to bend to the angle needed to keep the helicopter's fuselage and rotor level. The robotic legs have been attached to a 113 kg Rotor Buzz 2 unmanned helicopter and a series of test flights were performed from 2013 to 2015. Multibody dynamic simulations have also been carried out and several publications have been made to present the results of this software simulations.

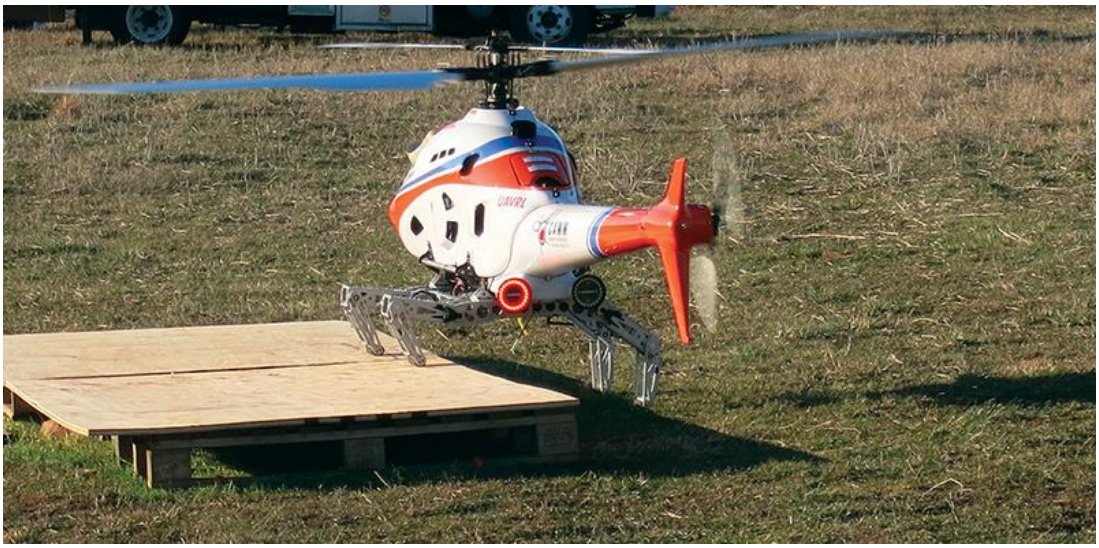

Figure 2-4 Georgia Tech robotic landing gear on Robot Buzz 2 helicopter during test flight [15]

In the first publication [16], a series of software simulations were run to model the system landing on a sloped terrain. Here the joints are modelled as rotational spring-damper systems (shown in Figure 2-5), and the slope landing controller resets the joints stiffness zero-load points and sets the damping coefficients to zero if a leg touches the ground. This causes the leg to freely retract after touchdown. Once all legs are in contact with the ground, all joint damping coefficients are reset to its initial value and the stiffness zero-load points remain at its previous value, causing the legs to block its position in a compliant way. An additional proportional-derivative controller corrects the remaining tilt of the body after landing, by measuring the tilt angle and calculating the required moment applied to each hip to level the fuselage. In the

simulations, the system is able to land successfully at slopes of 20° with the robotic landing gear.

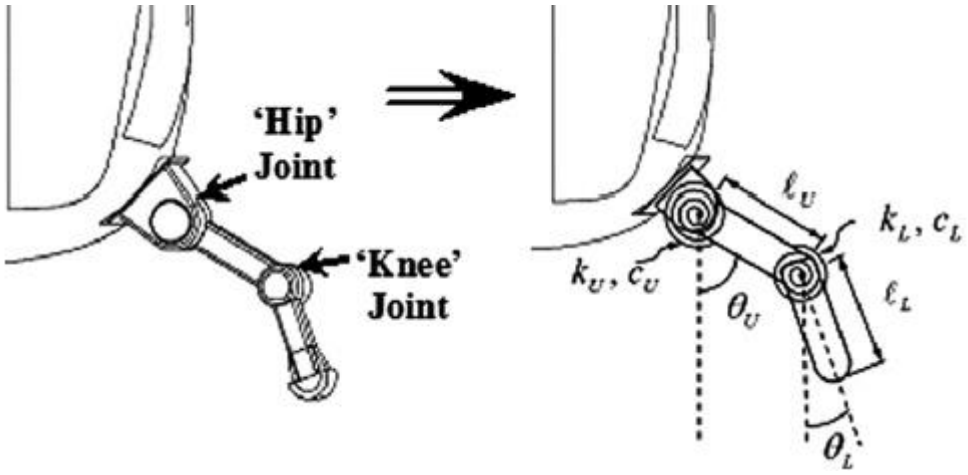

Figure 2-5 Joints modelled as rotational spring-damper systems [9]

In another publication [9], the system is tested for hard landing mitigation. In this case, as the aircraft descends, the controller also controls the spring and damper coefficients of the joints to freely retract if a leg makes ground contact. Once all legs are in contact with the ground, the spring and damper coefficients are calculated in order to control the deceleration of the fuselage until this is brought to rest.

The system is also tested on a simulation of a landing on a moving shipboard [17]. Here the algorithm that controls the position of the legs uses a Virtual Model Control (VMC) technique. This methodology was developed by [18] for control of legged robots and consists in introducing a set of “virtual” elements like springs and dampers, into a real physical systems to create a desired dynamics. Then, the forces that this virtual system would produce on the real system are calculated and converted into the joint space as joint torque commands. Figure 2-6 shows examples of applications of virtual components used for locomotion.

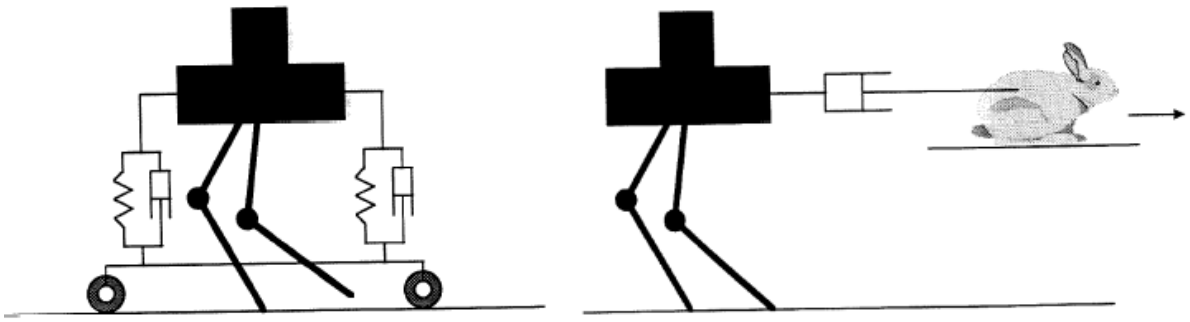

Figure 2-6 Two examples of VMC [18]. The Virtual Granny Walker (left) to control the altitude and the Bunny Mechanism (right) to make the robot move forward.

Although these publications show the application of different control techniques on software simulations, to our knowledge there haven't been any published results on the practical flying tests. According to [15] the team at Georgia Tech has claimed that their system is able to land and take off from slopes up to 20° and their system would take up to about 7% of the maximum payload on a medium sized helicopter.

In [19], the authors present a different design and control approach for an articulated landing gear. Their design uses the principle of mechanical differential to passively control the attitude of the system when landing on uneven surfaces. The system, shown in Figure 2-7, consists of four legs connected by a series of springs that are able to transfer loading from one leg to the others, and to evenly distribute the loading between its four legs when landing on uneven ground and to settle to a stable horizontal position. The authors presented a theoretical design and analysis, software simulations and physical implementation. A series of landing tests at 0, 10 and 20 degrees slope were performed and the results were presented. Unlike other designs, this system does not require any power supply, active control systems or sensors.

Figure 2-7 Passive legged landing gear [19]

In more recent years, several designs of actively controlled articulated landing gears have been developed as shown in Figure 2-8. In [20] the authors published a letter presenting DroneGear, a landing gear aimed for multicopters. Their system consists of four compliant robotic legs of 2-DoF each embedded with optical torque sensors integrated into the knee joints and Inertial Measurement Units (IMU) into passive footpads for landing zone profile estimation. According to the authors, the optical torque sensors add active-passive compliance to the leg structure and provides the whole lower leg link with sensitivity to external perturbations, unlike foot pressure pads

which provide sensitivity only in the tip. The control system of the DroneGear is out of the scope of the letter and a series of tests are conducted with the system landing on 3 and 11 degrees slopes, on step, and on flat surface.

In [21], the authors present a legged landing gear that uses a 4-bar linkage to restrict the motion of the legs in the vertical direction only. A platform is built using plastic 3D printed components. This system incorporates one servo motor per leg, foot pressure sensors, an Inertial Measurement Unit to calculate the attitude of the system, an ultrasonic range finder to measure the distance from the system to the ground and an on-board controller. The paper describes a 3-DoF model of the system to control the roll, pitch and altitude dynamics. For the body attitude control, a cascade PID controller is used to calculate the actuator torque. For the body altitude control, a cascade PID determines the total support force of all four legs, and a force optimization algorithm is used to calculate the distribution of the forces on each leg.

Figure 2-8 Adaptive landing gear systems in [20] (left) and [21] (right)

Recently, the ETH Zürich created ATHLAS, a project in which a group of 12 engineering students has developed an adaptive landing system for helicopters (shown in Figure 2-9). They constructed a prototype of the landing system and implemented it into a model helicopter of 50 kg and a rotor diameter of 3.2 m [22].

Figure 2-9 ATHLAS landing gear during flight test [22]

The system consists of four individually controllable legs and each leg uses a brushless EC flat motor to drive a ball screw gear. According to their website, the controller adapts the position of each leg to the terrain using force control. By logging the current induced in the motors, they can detect ground contact and raise the legs until all legs are in contact without any additional sensory feedback. Then, the orientation of the helicopter is regulated using an IMU. A conference paper was published in 2018 [23], where the authors give details of the mechanical structure of the legs and the interface with the fuselage, the electronics and control software used. It also describes how the system was tested and future work.

The systems found in the literature review are mostly patents that haven't been developed or very recent and ongoing research projects with little technical information published. In the Darpa system, all the result analysis come from software simulations. While tests with a physical platform have been reported, to the author's knowledge, there is not any scientific publication that analyses the performance of an adaptive landing gear during practical tests and the dynamics of the system during landing. Table 2-1 shows a comparison between the adaptive landing gear systems that have been built.

Table 2-1 Comparison of existing robotic landing gears

System	Design	Controller	Results	Critique
DARPA [9] [16] [17]	4 2-DoF legs Feet force sensors Electric motors with brake system Medium-size helicopter platform	Legs in ground contact retract using force control Joint torque-control adjusts joint stiffness and damping. Virtual Model Control	3 publications based on simulation results using multibody systems modelling Landings on 20° slopes moving ship deck, and hard landing mitigations Video of practical test	No result analysis on practical tests. No information on how the controller is implemented in the physical prototype. Only legs in ground contact move, and attitude control is applied after force control, increasing the time until the landing finishes.
ETH Zürich, ATHLAS [22]	4 1-DoF legs Brushless EC motors Ball Screw Gear Medium-size helicopter platform	Position control and force regulation No specific details	Claims landings on 20° slopes and 50cm steps 1 publication shows the system description including mechanical structure, electronics and control software. Video of practical test	Little result analysis on publications Only legs in ground contact move, and attitude control is applied after force control, increasing the time until the landing finishes.
Drone Gear [20]	4 2-DoF legs Custom-made knee optical torques sensors Feet IMU for relief profile estimation Small-size hexacopter platform	Legs in ground contact retract using joint torque feedback Three-level controller Position-based control Very little details	Practical tests on 1-axis slope of 3° and 11° Flight test with hexacopter on 35 mm step	Control system out of the scope of the letter No simulations performed. Practical tests at a maximum slope of only 11°.
UAV landing gear [21]	4 1-DoF legs Feet force sensors Body IMU Distance sensor Small-size helicopter platform	Altitude and attitude cascade PID control plus force optimization algorithm	Simulation on 3-DoF model (altitude, roll, pitch)	No result analysis on practical tests. Very little simulation analysis.
Passive landing gear [19]	4 1-DoF legs No sensing or active control elements Small-size multi-rotor platform	Passive control Mechanical differential principle Legs in contact retract while the opposite extends due to load sharing	Simulations and practical tests Drop tests with 0°, 10° and 20° slopes	No feedback signal to confirm landing state Less controllability as there is no active control Passive control cannot drive the system to completely level position

2.2 Legged Robots

The preceding section focuses on adaptive landing systems. This section focuses on legged robots where significant progress has been done in recent years in the field of balance controllers for quadruped and biped locomotion on rough terrain. Although designed for a different purpose, the concepts applied for walking robots can be adapted to landing control in terms of leg design, sensors, actuators or controller design. In this section, a review of some of the most relevant quadrupeds for rough terrain locomotion is presented.

One of the most important advances in legged robotic technology in recent years has been the development of BigDog by Boston Dynamics [24], because it addresses the two main limitations of previous robots, it includes on-board power supply, and is capable of walking in rough, slippery and sloping terrain. Although detailed information has not been found, some of its characteristics are known.

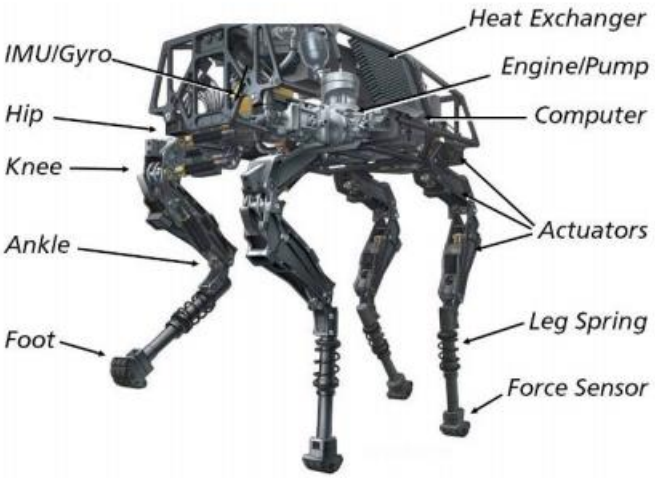

Figure 2-10 BigDog quadruped walking robot [24]

Figure 2-10 shows the BigDog which includes on-board systems that provide power, actuation, sensing, controls and communications. Each actuator has sensors for joint position and force and each leg has three joints: hip abduction/adduction, and hip and knee flexion/extension. It also has a spring in the lower leg and a rubber foot that makes ground contact compliant. It has about 50 sensors. Inertial sensors measure the attitude and acceleration of the body, while joint sensors measure motion and force of the actuators working at the joints.

The on-board control system includes a low-level control that regulates servos positions and forces at the joints and a high-level control system that coordinates the behaviour of the legs to regulate the velocity, attitude and altitude of the body during locomotion. For balancing, it uses an estimate of its lateral velocity and acceleration, determined from the sensed behaviour of the legs during stance combined with the inertial sensors, and while walking, the control system coordinates the kinematics and ground reaction forces at the feet while responding to postural commands. At the individual leg level, the control system uses joint sensor information to determine when feet are in contact with the ground and to determine the desired load on each leg and actuator.

Another significant platform is the HyQ quadruped robot developed by the Italian Institute of Technology, as shown in Figure 2-11 [25]. From the design point of view, each leg of the HyQ incorporates one electrically actuated joint for the hip, two hydraulically actuated joints at the hip and knee and a passive joint that connects the lower leg with the foot via a spring. The overall compliance of the leg is dealt with a combination of active control of the leg stiffness by means of the torque-controlled joints, and the compliant passive joint that absorbs initial ground impacts, to reach a trade-off between compliance and tracking performance.

On the control side, HyQ is equipped with an on-board computer for low and medium control tasks, while high-level control is executed by an external computer that communicates via Ethernet. At low-level, joint controllers are executed at 1 kHz, and they control joint position and torque. High-level control, generates leg trajectories at 200 Hz. The robot is equipped with over 50 sensors for control, system state monitoring and diagnostics. Among them there is an IMU with 9-DOF for robot balance, encoders at each joint for position feedback, load cells mounted on each hydraulic joint for force/torque feedback, and a potentiometer at the passive joint between the lower leg and the foot for estimation of the ground contact force by measuring the spring compression.

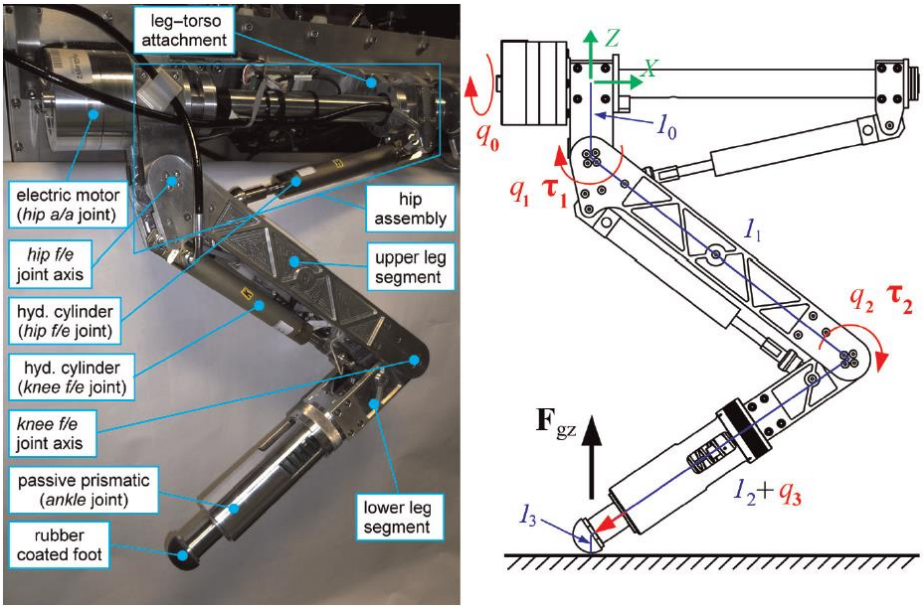

Figure 2-11 Detail of HyQ leg with component description and sketch [25]

RoboCat-1 is an electrically actuated quadruped developed by the Toyota Technological Institute [26] that has two actuated degrees of freedom per leg at the hip and knee in the sagittal plane. The robot is powered via the torque-controlled Harmonic Drive systems FHA-8C series AC servo motors. The compliance is actively -controlled without any passive element. As for the sensors, it incorporates two single axis gyroscopes in the torso to measure roll and pitch and one force sensing resistor at each foot to measure ground reaction forces. The operating system is the MATLAB xPC Target and the sampling frequency is 1 kHz. The RoboCat-1 system is shown in Figure 2-12.

The control system is divided into high-level and low-level controllers. In the high-level the CoM and foot trajectories are generated using polynomials and inverse kinematics equations are used to generate joint position commands. The low-level controller has three components [27]: friction and gravity compensation using friction hysteresis identification and inverse dynamics; active compliance control to generate joint displacements based on ground reaction force errors; and angular momentum control that uses gyro sensor information to calculate the compensating torque around the centre of mass.

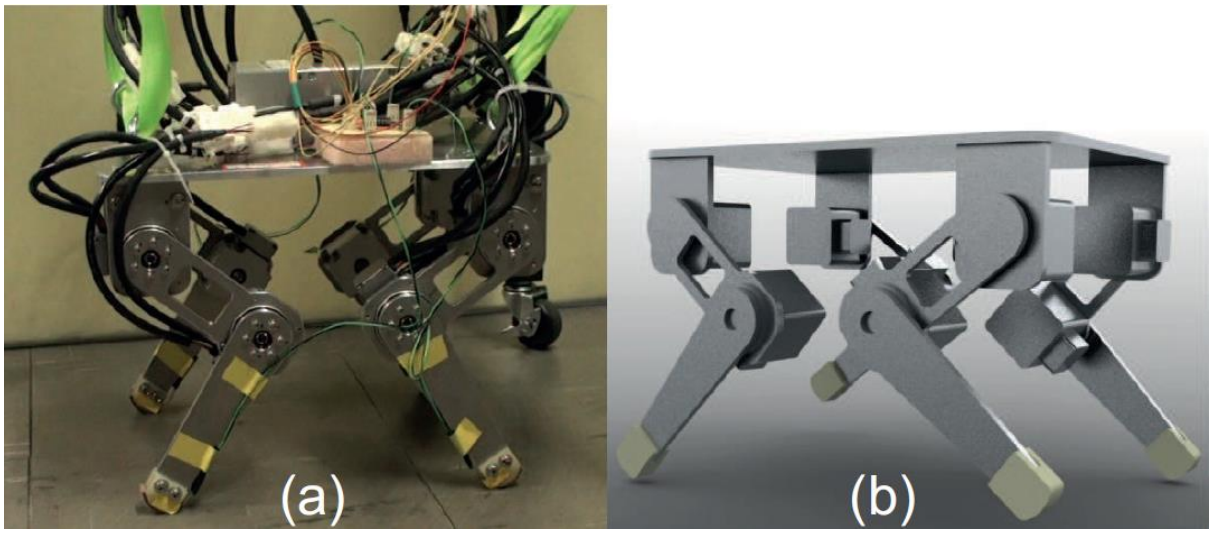

Figure 2-12 RoboCat-1 quadruped robot prototype (a) and CAD model (b) [26]

Most quadruped robots are torque-controlled systems, but there are also some examples of position-controlled robots like the Oncilla robot. This robot uses a bio-inspired design with the size and weight of a house cat [28]. It features three segmented legs with the middle link being a four-bar pantograph mechanism with a diagonal passive spring mechanism. Three actuators operate each leg. One actuator control the upper link angle. A second actuator acts flexing the two mid-joints by a cable mechanism, while the extension is driven by the passive spring, and a third actuator is responsible for the adduction/abduction. The simulation and hardware example of the system are shown in Figure 2-13.

In [29], a control architecture is designed to be implemented on the Oncilla platform based on a Central Pattern Generator (CPG) that creates synchronized rhythmic patterns for locomotion. Sensory feedback is added to implement reflexes fast corrections to add extra flexion to the leg in case of a collision of the foot with an obstacle, or to add extra extension in case of a missed contact and until the contact is sensed. Posture control feedbacks are also added to the CPG output based on the robot's body orientation. This architecture uses minimal sensory information and low-cost hardware and software.

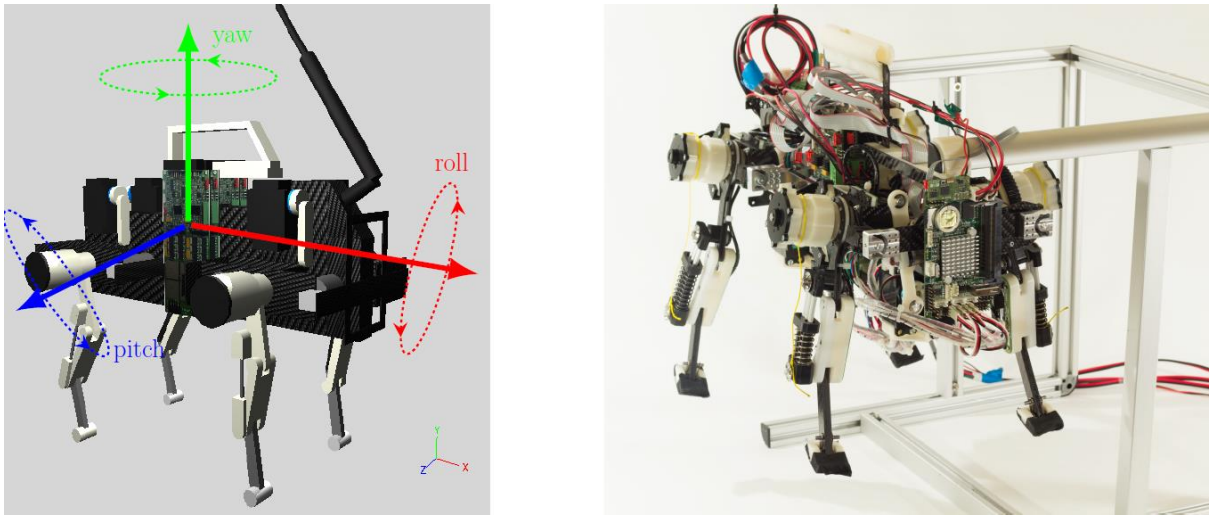

Figure 2-13 Oncilla platform simulation (left) and hardware robot (right) [29]

2.3 Mathematical Modelling

A robot is usually a mechanical system composed of a number of links connected by articulated joints. In this context, robot dynamics refer to the equations that explicitly relate the forces and torques applied to the robot system with the motion produced [30]. This relationship can be expressed as a set of second order, nonlinear ordinary differential equations called equations of motion which depend on the kinematic and inertial properties of the robot. System modelling is used to extract information of its behaviour without the necessity of building a physical prototype and to design controllers.

2.3.1 Lagrangian and Newton-Euler formulation

In general there are two main methodologies to obtain the equations of motion of a system: the Newton-Euler and the Euler-Lagrange equations [31] [32] [33].

The Newton-Euler formulation to describe rigid body dynamics applies the principles that the rate of change of the linear momentum of a body equals the total force applied to this body, and the rate of change of the angular momentum of a body equals the total torque applied to this body [30] [34]. The expressions for the dynamics are given in equations 2.1 and 2.2 respectively.

$$m\ddot{\mathbf{r}}_{\mathbf{cm}} = \mathbf{f} \tag{2.1}$$

$$\mathbf{I}\dot{\mathbf{w}} + \mathbf{w} \times (\mathbf{I}\mathbf{w}) = \boldsymbol{\tau} \tag{2.2}$$

where *m* is the mass of the body and it's constant, $\ddot{\mathbf{r}}_{\mathbf{cm}}$ is the linear acceleration of the Centre of Mass (CoM), **f** is the resultant or total force acting on the body, **I** is the inertia matrix, **w** is the angular velocity and **τ** is the total moment acting on the body.

The Euler-Lagrange formulation is derived from Newton's second law but it describes the dynamics of the system in terms of work and energy. The Lagrangian function ($\mathcal{L}$) of a system is defined as the difference between its kinetic energy (K) and its potential energy (P).

$$\mathcal{L}(q, \dot{q}) = K(q, \dot{q}) - P(q) \tag{2.3}$$

Kinetic and potential energy are expressed in terms of the generalised coordinates q_i and generalised velocities $\dot{q}_i$. The dynamic equations of motion are obtained using the Euler-Lagrange equation for each generalised coordinate.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{2.4}$$

where τ_i is the generalised torque associated with q_i. *i* = 1,…,n, and *n* is the number of degrees of freedom.

Although both methods have differences, they generate equivalent sets of equations. The main difference between both methods arise from the coordinate systems they use. Newton-Euler method uses Cartesian coordinates, therefore, for each spatial rigid body, six coordinates are used to represent the body position and orientation. The connectivity between different bodies is defined by introducing constraint equations. In constrained multibody systems, this leads to a set of redundant coordinates, and the resulting equations are expressed in terms of dependant coordinates as well as constraint forces [35]. Lagrange method uses generalised coordinates, which are defined as the smallest set of independent variables that completely describe the system configuration. For a robot manipulator, these generalised coordinates are usually the joint angles. Lagrange method reduces the number of equations to the number of degrees of freedom of the system and provides a closed form expression in terms of the joint torques and joint displacements. In Newton-Euler method, additional arithmetic operations are required to eliminate constraint terms from the equations of motion and obtain explicit relations between the joint torques and joint displacements in a closed form expression.

2.3.2 Spatial and planar models

Equations 2.1 and 2.2 are expressed in vector quantities. If the linear acceleration of a point *i* is expressed as

$$\ddot{\mathbf{r}}_{\mathbf{i}} = \ddot{\mathbf{r}}_{\mathbf{i-1}} + \dot{\mathbf{w}}_{\mathbf{i}} \times \mathbf{r}_{\mathbf{i}} + \mathbf{w}_{\mathbf{i}} \times (\mathbf{w}_{\mathbf{i}} \times \mathbf{r}_{\mathbf{i}}) \tag{2.5}$$

where $\ddot{\mathbf{r}}_{\mathbf{i-1}}$ is the linear acceleration of the previous point, $\dot{\mathbf{w}}_{\mathbf{i}}$ and $\mathbf{w}_{\mathbf{i}}$ are the angular acceleration and velocity of point *i*, and $\mathbf{r}_{\mathbf{i}}$ the vector from the previous point to point *i*.

Then, the Newton equation for linear momentum for a spatial body can be expressed in its expanded form as

$$m\begin{bmatrix} \ddot{r}_{x_{i-1}} + \dot{w}_y r_z - \dot{w}_z r_y + w_y w_x r_y - w_y^2 r_x - w_z^2 r_x + w_z w_x r_z \\ \ddot{r}_{y_{i-1}} + \dot{w}_z r_x - \dot{w}_x r_z + w_z w_y r_z - w_z^2 r_y - w_x^2 r_y + w_x w_y r_x \\ \ddot{r}_{z_{i-1}} + \dot{w}_x r_y - \dot{w}_y r_x + w_x w_z r_x - w_x^2 r_z - w_y^2 r_z + w_y w_z r_y \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{2.6}$$

The rotational characteristics of a spatial rigid body are determined by its inertia tensor

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{2.7}$$

where the diagonal elements are called the principal moments of inertia, and the off-diagonal terms are the cross products of inertia.

Then, the Euler equation for spatial rigid body rotational motion in expanded form is

$$\begin{bmatrix} I_{xx}\dot{w}_x + I_{xy}\dot{w}_y + I_{xz}\dot{w}_z - w_y w_z(I_{yy} - I_{zz}) - I_{yz}(w_z^2 - w_y^2) - w_x(w_z I_{xy} - w_y I_{xz}) \\ I_{yx}\dot{w}_x + I_{yy}\dot{w}_y + I_{yz}\dot{w}_z - w_z w_x(I_{zz} - I_{xx}) - I_{xz}(w_x^2 - w_z^2) - w_y(w_x I_{yz} - w_z I_{xy}) \\ I_{zx}\dot{w}_x + I_{zy}\dot{w}_y + I_{zz}\dot{w}_z - w_x w_y(I_{xx} - I_{yy}) - I_{xy}(w_y^2 - w_x^2) - w_z(w_y I_{xz} - w_x I_{yz}) \end{bmatrix} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{2.8}$$

The Newton-Euler equations are significantly simplified for the case of planar motion. Considering a body that can move in the YZ plane and rotate around the X-axis, then $w_y = w_z = 0$, and the inertia tensor becomes a scalar value I_{xx}

$$\begin{bmatrix} I_{xx}\dot{w}_x \\ m(\ddot{r}_{y_{i-1}} - \dot{w}_x r_z - w_x^2 r_y) \\ m(\ddot{r}_{z_{i-1}} + \dot{w}_x r_y - w_x^2 r_z) \end{bmatrix} = \begin{bmatrix} \tau_x \\ f_y \\ f_z \end{bmatrix} \tag{2.9}$$

2.4 Ground Contact Models

The modelling of contact forces is an important part of the robot dynamics and there are different standard approaches. Concerning friction forces, tangential to the contact surface, a Coulomb friction model is usually considered. Concerning the normal force to the surface, two options are the most used: a compliant or a rigid model [33].

Rigid models don't allow the contact points to penetrate below the surface. Impacts are modelled as instantaneous events between 'no contact' and 'contact' states, and the foot velocity at the moment of contact changes instantaneously to zero. This produces switching dynamics, as the equations of motion (and DoF) of the robot are different for different contact situations [36].

By contrast, compliant models add additional forces acting upon the foot, instead of geometric constraints, hence, the equations of motion and the DoF of the system don't change. These forces are simulated using spring-damper elements to take into account the viscoelastic properties of the materials in contact, and are a result of a penetration of the contact point below the contact surface. The most typical configuration to simulate contact forces is the linear spring-damper system [37]

$$f_n = -k \cdot z - b \cdot \dot{z} \tag{2.10}$$

where f_n is the normal contact force, z is the penetration depth and k and b are the spring and damper coefficients.

This model has two main drawbacks, namely it introduces a discontinuity at the moment of impact if the velocity is not zero, and permits not only forces due to the compression of the contact surfaces, but also forces that tend to hold the objects together. The authors in [37] and [38] propose several alternatives to solve these problems. A common and easy to implement approach to avoid negative 'sticking' forces is to saturate these forces by setting them to zero when they become negative. To avoid the discontinuity at the moment of contact a usual approach is to add a nonlinearity in the damping element. Thus, equation 2.10 becomes

$$f_n = -k \cdot z - b \cdot z \cdot \dot{z} \tag{2.11}$$

The dependence of the damping term on the penetration depth causes the force to build up from zero and avoids the discontinuity.

An important question is how to parameterise the contact model. In, [39] the author presents a methodology to choose the appropriate spring-damper coefficients. The spring force is proportional to the amount of ground penetration and the spring coefficient, *k*. The desired amount of ground penetration at rest is used to choose the appropriate value for the spring constant using the equation

$$k = \frac{mg}{h_{eq}} \tag{2.12}$$

where h_{eq} is the ground penetration at rest and *m* is the mass supported by each leg.

The damping force is proportional to the rate of penetration and the damping coefficient is defined as:

$$b = 2d\sqrt{mk} \tag{2.13}$$

where *d* is the damping ratio and it defines the damping properties of the ground. A damping ratio of 1 will result in a critically damped system. If it's less than 1 the system will be underdamped, and if it's greater than 1 it will be overdamped.

2.4.1 Friction Force

Regarding the friction force models, there are also different approaches in the literature that use variations of the compliant and Coulomb models.

A common approach is to model the friction force as being proportional to the normal force and friction coefficient, and in opposite direction to sliding motion [39].

$$f_t = -\mu \cdot f_n \cdot sgn(v_t) \tag{2.14}$$

where f_t is the tangential force, μ is the friction coefficient and v_t is the tangential component of the foot velocity.

Friction models have two regions usually named as 'sticking' and 'sliding' regions, so in order to produce relative motion or sliding between two surfaces in contact, an external force has to overcome the initial 'stickiness'. The switching between regions is usually produced by the change of the velocity from zero to non-zero value or because the friction force value exceeds the maximum static friction force.

Equation 2.14 provides the expression for the friction force in the 'sticking' region with μ being the static friction coefficient. In the 'sliding' region usually the value of μ changes to the dynamic friction coefficient or the model is saturated.

Different Coulomb and viscous models are explained in [40]. Figure 2-14 illustrates the implementation if these models, the transitions between sticking and sliding regions, and the friction force against the sliding speed.

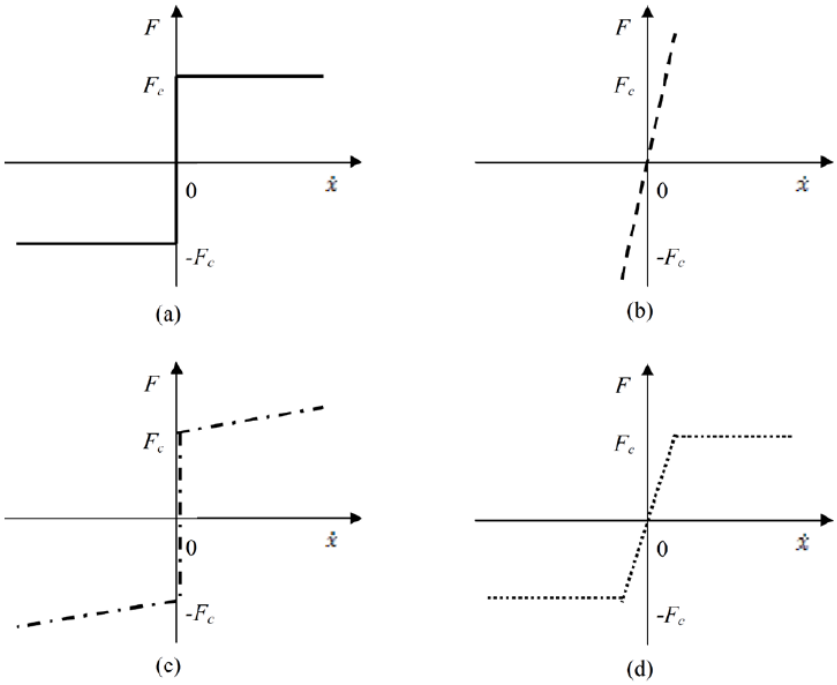

Figure 2-14 Comparison of different Coulomb friction models [40].

In [38], the authors present a comparison between different ground contact models that have been found in the literature for modelling of walking, running and jumping robots. In [16] and [17], the authors present two different implementations of compliant friction models for the simulation of rotorcraft landing. Compliant friction models apply the same spring-damper forces in the tangential directions as in the normal direction. Switching between regions occur if the value of the force exceeds the static friction force.

2.5 Joint Controllers

Depending on the application they are designed for, there are commonly two main approaches to design joint controllers for robotic manipulators: Classical or Independent Joint Control, and Model-Based Control.

A simple type of controllers that often gives good results in practice is referred in the literature as classical joint control, independent joint control on non-model-based control [41] [42] [43]. Its implementation is relatively easy because the control input depends only on locally measured variables, like joint position and velocity, and it doesn't depend on the variables of other robot joints. Thus, they can be implemented in local joint processors without the necessity of communication among the joint controllers at each joints. The term non-model-based control refers to the fact that they don't require knowledge of the model parameters and structure.

Proportional-Derivative or PD is the simplest controller of this type and the computed joint torque depends only on the position and velocity of that particular joint.

$$\tau = K_P e + K_D \dot{e} \tag{2.15}$$

where K_P and K_D are the proportional and derivative gains and e is the position error defined as the difference between the desired joint position and the actual position.

However, PD control cannot guarantee that the position error will converge to zero, and the precision of the controller will depend of the gains. In theory, increasing the gains will reduce the steady-state error, but these gains are limited in practice by the measurement noise and other unmodeled dynamics [33].

A common approach to eliminate the steady-state error and improve the disturbance rejection capabilities is to add an integral part to the PD controller. This controller is the PID and nowadays is used in most industrial robots controllers.

$$\tau = K_P e + K_D \dot{e} + K_I \int e dt \tag{2.16}$$

where K_I is the integral gain.

The integral part of the controller is proportional to the magnitude and to the duration of the error. Thus, if the error is accumulated over time the value of the integral term increases. This way, PID controllers can eliminate the steady state error using low controller gains. The integral term also accelerates the system response. However, if not properly tuned it can lead to overshoot and higher settling time.

When adding integral term to the controller it is important to be aware of the possibility of integrator windup due to actuator saturation limits [42]. This is a phenomena that appears specially when an error persists over a long period of time and causes the integral term of the PID to grow over the limits of an actuator. Then, a negative error is needed during a long time before the control signal returns to its normal operation range provoking that the controller gives an incorrect control input for a long period [44]. Anti-windup methods usually include a saturation model of the actuator to keep the output of the integral term within desired limits.

PID joint controllers perform well in a wide variety of applications, and are easy to implement as the computed torques don't depend on variables from other joints. Also, computational loads are low and do not involve solving complicated nonlinear inverse dynamics, thus, they can be implemented using low-cost hardware [33].

Robot manipulators are nonlinear systems that change their position over time. Due to coupling effects, the position of each joint affects the torques on the other joints, and it is not possible to select a joint controller with fixed gains that will give optimal response for all robot positions. Additionally, for application that require high-precision trajectory tracking or high-speed operations, a controller that takes into account the manipulator dynamics should be designed. For this purpose, the model-based control strategies were developed.

Possibly the simplest model-based controller is the PD with gravity compensation which includes the gravitational terms of the dynamic model, *G(q)*, in the control law.

$$\tau = K_P e + K_D \dot{e} + G(q) \tag{2.17}$$

This controller compensates for the joint torques created by the gravitational forces and, though still simple, it required knowledge of the gravity components of the model and model parameters. Unlike PD and PID controllers, independent joint controllers cannot be implemented as position information of all other joints is needed to compute the torque for any given joint.

Although many different model-based controllers have been proposed, most of them are variations of the so-called Computed-Torque Control, also referred as Inverse Dynamics control, which applies a control law that includes the inverse model of the system being modelled.

$$\tau = M(q)u + C(q,\dot{q})\dot{q} + G(q) \tag{2.18}$$

Where *M* and *C* are the Inertia and Coriolis matrices respectively. Equation 2.19 represents the inverse dynamics of the system but replacing the joint acceleration $\ddot{q}$ by the control input *u*. This control input is typically designed using a state feedback like PD or PID control

$$u = \ddot{q}_d + K_P e + K_D \dot{e} \tag{2.19}$$

where $\ddot{q}_d$ is the desired joint acceleration.

The controller design problem is divided into an inner loop that cancels the nonlinear dynamics and an outer feedback loop that corrects any error in the desired trajectory. Because the control input is multiplied by *M*, the controller gains are not constant, but varying with the actuator's position.

The strong point of this approach is that it uses a nonlinear control law to cancel out the nonlinearities of the dynamic model, so that the overall closed-loop system is linear. The result is that the controller can be much faster and accurate compared to a pure linear feedback control, like PD. On the other side, it is difficult to implement because it requires a good knowledge of the full model structure and parameters like masses and inertias, and because it involves complex and time-consuming computations, resulting in longer sampling times.

A proposed model-based controller to reduce the computation time is the PD with feedforward control. This method computes the torques in terms of the desired dynamics ($q_d, \dot{q}_d, \ddot{q}_d$) instead of the measured ones, thus, if the desired path is known in advance, this values can be computed offline. A feedback PD control law is added afterwards.

$$\tau = M(q_d)\ddot{q}_d + C(q_d,\dot{q}_d)\dot{q}_d + G(q_d) + K_P e + K_D \dot{e} \tag{2.20}$$

2.6 Introduction to Sliding Mode Control (SMC)

SMC belongs to the area of robust control which is an approach to controller design that aims to achieve system stability and performance even in the presence of model imprecisions, like uncertainty of model parameters or unmodelled dynamics. With this technique, the controller is designed to drive the system states and then constrain

them into a particular surface, known as the sliding surface, by using a control law based on a high-speed switching function. Here, the controller design consists in two parts: first, the design of the sliding surface that satisfies the control goal, and second the design of a control law that will drive the system into the sliding surface and keep it there for all subsequent time [45].

For the sliding surface design, usually, a function of the tracking error and some of its derivatives is selected, in such a way that its zeroing represents a linear differential equation which will drive the system error to zero.

Consider the nonlinear single-input dynamic system:

$$x^n(t) = f(\mathbf{x}, t) + u(t) \tag{2.21}$$

where the scalar *x* represents the output of interest, for the system order *n*, the scalar *u* represents the control input, $\mathbf{x} = [x\ \dot{x}\ \ldots\ x^{n-1}]^T$ is the state vector, and the function *f*(**x**,*t*) are the nonlinear modelled system dynamics and are not exactly known.

The control problem is to get the state vector to track a desired time-varying state, $\mathbf{x_d} = [x_d\ \dot{x}_d\ \ldots\ x_d^{n-1}]^T$, in the presence of disturbances or modelling imprecisions.

If the tracking error of the variable *x* is defined as $\tilde{x} = x - x_d$, then a typical choice for the sliding surface is

$$s(\mathbf{x}; t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} \tag{2.22}$$

where λ is a positive constant.

An important condition derived from equation 2.22, is that when choosing the sliding surface, this should have relative degree one with respect to the control input, thus, the first time derivative of *s* should be a function of *u*. In the case of n=2 for example, then

$$s = \dot{\tilde{x}} + \lambda\tilde{x} \tag{2.23}$$

The method of equivalent control is used to design the control law that restricts the motion of the system onto the sliding surface. To ensure that there's no motion from the sliding surface once the system reaches sliding mode, the dynamics are written as [46]

$$\dot{s} = 0 \tag{2.24}$$

For instance, consider the second order system

$$\ddot{x} = f + u \tag{2.25}$$

In order to have $x(t)=x_d(t)$, we set $\dot{s} = 0$, then using equations 2.23 and 2.25 we have

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} = f + u - \ddot{x}_d + \lambda\dot{\tilde{x}} \tag{2.26}$$

Because of uncertainties or unmodelled dynamics, the value of f is not exactly known, but estimated as $\hat{f}$. Then, $\hat{u}$, is the best approximation of a continuous control law to get $\dot{s} = 0$

$$\hat{u} = -\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} \tag{2.27}$$

$\hat{u}$ is the equivalent control, which can be interpreted as the continuous law that would maintain $\dot{s} = 0$ if f was exactly known. To deal with model uncertainties or disturbances, we add a discontinuous or switching law

$$u = \hat{u} + k \cdot sgn(s) \tag{2.28}$$

The system behaviour in Sliding Mode Control can be divided in two parts. During the time until the system trajectory reaches the sliding surface, the system is said to be in reaching mode. Here the controller acts pushing the system towards the sliding surface. When the sliding manifold is reached, the system is said to be in sliding regime or sliding mode. In this mode, the tracking error will converge asymptotically to zero following the equation

$$s(\mathbf{x}; t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} = 0 \tag{2.29}$$

Thus, the dynamic behaviour of the system in sliding mode can be tailored by the choice of the sliding surface [47].

Figure 2-15 shows the typical behaviour of a system with n=2 in sliding mode control showing the reaching and sliding mode phases. The sliding surface is a line with slope $-\lambda$ containing the point $\mathbf{x_d}$.

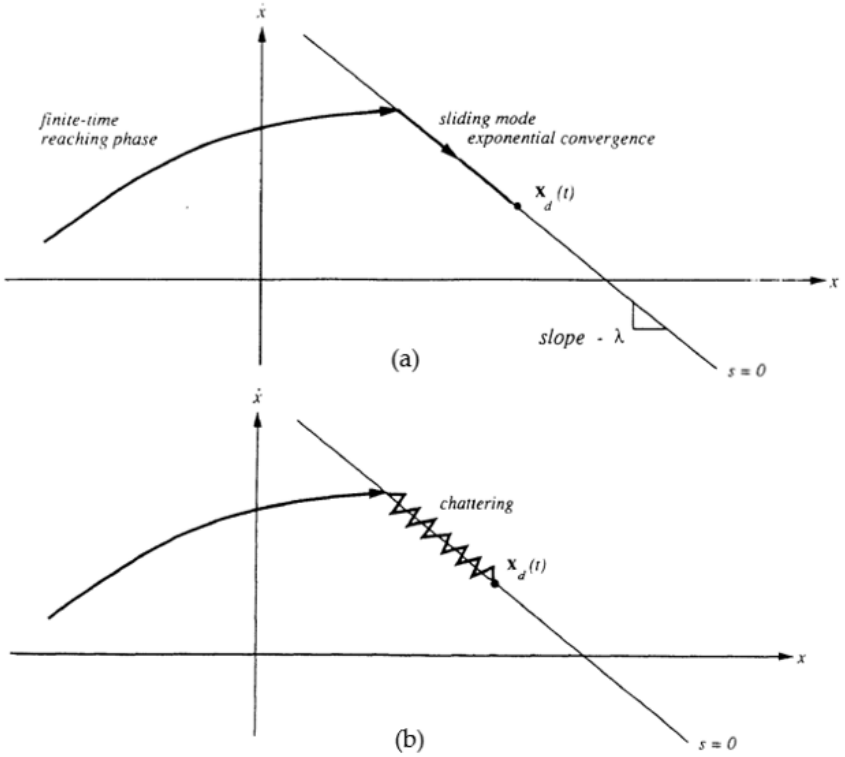

Figure 2-15 Graphical interpretation of system pushed into sliding surface (a) and representation of chattering effect (b) [46]

Figure 2-15(a) shows an ideal SMC where the switching frequency is supposed to be almost infinite, however, due to the discrete-time nature of digital computer implementation, in practice the sign function yields a “zig-zag” motion around the sliding surface, known as chattering (Figure 2-15(b)) [48].

In practice, it is important to avoid chattering since it involves high control activity and may excite high frequency neglected dynamics. A usual approach to solve this problem is to approximate the discontinuous control function in equation 2.28, in order to obtain a smooth/continuous control action, while keeping robustness and tracking precision.

A proposed method is to replace the sign function by a “sigmoid function”

$$sgn(s) \approx \frac{s}{|s| + \varepsilon} \tag{2.30}$$

where ε is a small positive scalar.

Another common solution is to constrain the system inside a boundary layer around the switching surface by using an algorithm that smooths out the control discontinuity within this layer

$$sat(s,\delta) = \begin{cases} sgn(s), & |s| > \delta \\ s/\delta\,, & |s| \le \delta \end{cases} \tag{2.31}$$

where δ is the boundary layer thickness.

Figure 2-16 shows the output of the three algorithms using sign function without smoothing out (a), sigmoid function (b) and boundary layer (c).

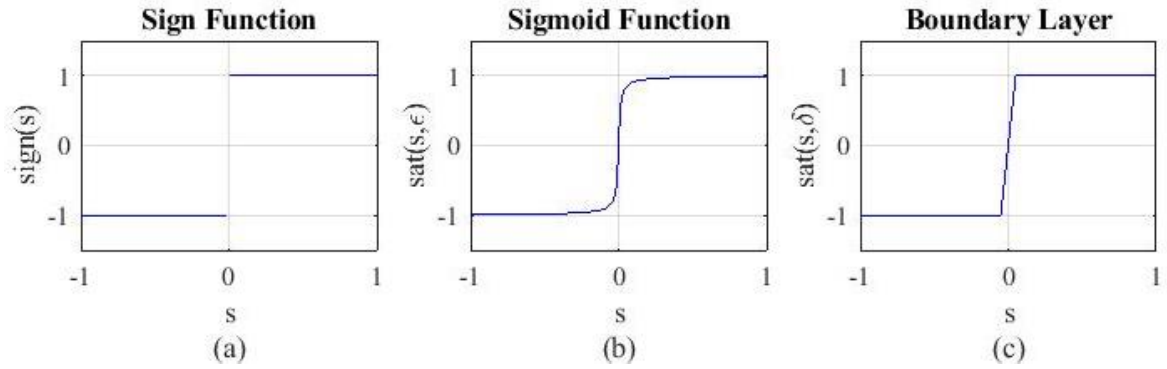

Figure 2-16 Matlab simulation of sign function (a), sigmoid (b) and boundary layer algorithm (c)

The previous methods are approximations used to obtain a smooth control law without chattering. However, some degree of tracking performance is lost. Moreover, the design of the sliding variable is constrained to be of relative degree one.

Higher Order Sliding Modes (HOSM) are an alternative for the reduction or even complete elimination of chattering, without compromising the robustness of the standard sliding mode. They are characterised by applying the discontinuous/switching action on the higher-order time derivatives of the sliding variable instead of the first one, as conventional SMC [49], in such a way that in a n^{th}-order SMC, the discontinuity acts on s^n, and the controller can drive $s = \dot{s} = \ldots = s^{n-1}$ to zero.

In [50], some of the most common 2nd order SMC algorithms are presented. These include the Twisting, Sub-Optimal, Super-Twisting or Drift algorithms. Among them, the most popular seems to be the Super-Twisting algorithm for its versatility, simplicity of implementation and because, unlike other algorithms, it doesn't require information of any of the time-derivatives of the sliding variable.

The Super-Twisting algorithm is a special case in the sense that it only has relative degree one (u appears in the first derivative of s), but it's second order as it can enforce $s = \dot{s} = 0$. The algorithm consists of two parts: one term is a continuous function of the

sliding variable, while the other is an integral of a discontinuous term, hence, the output is continuous as the discontinuity is hidden under the integral.

A general form of the Super-Twisting algorithm is defined by [47]

$$u_{st} = u_1 + u_2 \tag{2.32}$$

$$\dot{u}_1 = \begin{cases} -u_{st} & if\ |u_{st}| > U \\ -Wsign(s) & if\ |u_{st}| \leq U \end{cases} \tag{2.33}$$

$$u_2 = \begin{cases} -\lambda |s_0|^{\rho} sgn(s) & if\ |s| > s_0 \\ -\lambda |s|^{\rho} sgn(s) & if\ |s| \leq s_0 \end{cases} \tag{2.34}$$

where W and λ are positive constants, $0<\rho\leq 0.5$, U is the maximum magnitude of the control output and s_0 is a boundary layer around s.

Simplified versions of the algorithm neglect the effect of the bound of the control, U, and the boundary layer, s_0, and consider $\rho=0.5$, so the algorithm can be expressed as [45]

$$\begin{cases} u_{st} = -\lambda\sqrt{|s|}sign(s) + u_1 \\ \dot{u}_1 = -Wsign(s) \end{cases} \tag{2.35}$$

A single-parameter tuning method is proposed in [45] and [48], by defining a positive constant, C, and then select $\lambda = \sqrt{C}$ or $\lambda = 1.5\sqrt{C}$, and $W = 1.1C$. There are also methods to select this constants as a function of system parameters, but in practice it is usual to tune the system heuristically [49] [50].

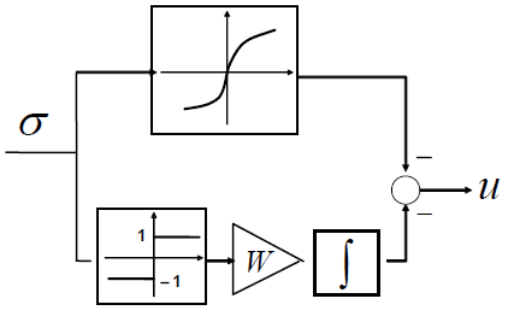

Figure 2-17 Block diagram of Super-Twisting algorithm where σ is the sliding surface and u the controller output [45].

2.7 Summary

This chapter provides a literature review of the state of the art of adaptive landing gear systems, quadruped robots, modelling techniques for legged robots, contact models and joint controller design.

The comparison between adaptive landing gear systems ranges from small to medium size, for helicopters and multicopters, with 1-DoF and 2-DoF legs, and using a variety of sensor and controller designs based on position and torque control.

From a control perspective, systems can be broadly classified as position or torque controlled. In position-controlled systems, a controller provides position commands to the joint actuators, while in the torque-controlled systems, the controller provides torque commands. The later require that the actuators have torque-control capability which is not common in small and low-cost servo motors. In some of these systems there's limited information about the control system, or this is used only in simulations and it's not clear how it would be implemented in a physical robot.

Sensor wise, the preference is to use feet force sensors rather than joint torque sensors to detect ground contact as they are easier to integrate, and usually less expensive. The control logic of some of these systems acts only in the legs that are in ground contact by retracting them, while the rest of the legs remain immobile until they touch the ground too.

There's a research gap as most of these systems are still under development and they offer limited information about their operation and control system. Some publications analyse the results of software simulations. Although some real flights have been reported, to the author's knowledge, there are no publications that analyse the performance of an adaptive landing gear on a real system.

The review of quadruped robots also reveals that most platforms share some characteristics like a hierarchic control structure with two or more levels, with different modules to control different tasks like balancing, centre of mass trajectory or leg swing trajectory. They all include some kind of inertial position and force or torque sensing, and achieve system compliance by active or passive methods. The overall system behaviour is designed to reach a trade-off between compliance and tracking performance.

From a modelling perspective, the application of the two main methodologies, Newton-Euler and Euler-Lagrange formulations, is reviewed in the field of legged robots. Rigid and compliant ground contact models are analysed and the main approaches to joint controller design, namely classical and model-based control, are also introduced. Finally, the main concepts of Sliding Mode Control are introduced.

3 System Modelling

This chapter presents the methodology to model the system dynamics. The system formed by the rotorcraft plus the landing gear can be considered as a “flying legged robot” and thus, similar modelling approaches to those used for legged robots can be applied to model the system. Legged robots can be classified as floating base systems as they are not connected to a rigid support. They have an unactuated base and fully actuated limbs. The motion of the limbs is completely defined by the joint torques, while the motion of the base is determined by the interaction forces between the limbs and the ground.

Two different models are presented. The first represents a landing gear with two robotic legs (one on each side) with a skid on each foot. In this case, the system motion can be represented with a planar model. The second, represents a landing gear with four legs (two on each side). In this case, the base can move in 6 DoF and a three dimensional model is used. This chapter also presents the model for all the external forces including ground reaction forces and helicopter thrust force.

3.1 Modelling of legged robots

In general, legged robots can be classified as floating base systems because they are not connected to a rigid support like a robotic arm. Instead, they have an unactuated (or underactuated) base connected to some actuated limbs, and the system uses the interaction forces between the ground and the limbs to produce motion on the base.

The configuration space of legged robots is composed of 6+*n* degrees of freedom (DoF), where n is the number of joints, and the other 6 DoF typically correspond to the position and orientation of the central body [33].

$$\begin{bmatrix}\mathbf{M_u}(\mathbf{q})\\ \mathbf{M_a}(\mathbf{q})\end{bmatrix}\ddot{\mathbf{q}}+\begin{bmatrix}\mathbf{C_u}(\mathbf{q},\dot{\mathbf{q}})\\ \mathbf{C_a}(\mathbf{q},\dot{\mathbf{q}})\end{bmatrix}\dot{\mathbf{q}}+\begin{bmatrix}\mathbf{G_u}(\mathbf{q})\\ \mathbf{G_a}(\mathbf{q})\end{bmatrix}=\begin{bmatrix}0_6\\ \boldsymbol{\tau}\end{bmatrix}+\begin{bmatrix}\mathbf{J_u^T}(\mathbf{q})\\ \mathbf{J_a^T}(\mathbf{q})\end{bmatrix}\mathbf{f} \tag{3.1}$$

where the subscrips **a** and **u** stand for the actuated and underactuated parts, **M** is the inertia matrix, **C** is the matrix of centrifugal and Coriolis terms, **G** is the vector of gravity terms, $\boldsymbol{\tau}$ is the vector of joint actuator torques, $\mathbf{J^T}$ is the transpose of the Jacobian matrix that maps ground reaction forces **f**, into generalised forces and torques. The vector of state variables is $\mathbf{q} = [\mathbf{q_b^T}\ \mathbf{q_n^T}]^\mathbf{T}$, where $\mathbf{q_b}$ is a 6x1 vector representing the position and orientation of the main body, and $\mathbf{q_n}$ is an nx1 vector representing each joint angle.

The structure of this model can be decoupled into base and manipulator dynamics. The first 6 rows are written in terms of Cartesian coordinates and Euler angles and coincides with the Newton-Euler equations of motion for the base. The n last rows correspond to the dynamics of a robot manipulator making contacts with its environment [51], and are usually obtained using Lagrangian formulation as it provides the closed-form equations for the manipulator.

There are different approaches in the literature to model the dynamics of legged robots. In [52], the author classifies dynamic models depending on how many assumptions they use to simplify the system dynamics. A common approach is to use simplified dynamics models like the Linear Inverted Pendulum. These models reduce considerably the complexity of the system dynamics but its use is restricted to basic environments and cannot deal with more complex scenarios. On the other hand, full-rigid-body models, fully describe the system in terms of every link and derive the relation between each joint torque and the corresponding motion. These models are very accurate but can become too complex, computationally expensive and intractable for complex robots. In a middle ground between these extremes, there are models like Single Rigid Body or Centroidal dynamics which make use of the Centroidal Momentum Matrix (CMM) to map the momentum of each individual link into a common reference frame, expressed at the system's CoM. The idea is to divide the problem into two sub-problems. The Centroidal model is used to find the correct motion of the unactuated base, then the fully-actuated manipulator dynamics are used to compute joint torques and joint motion. These methods are an exact projection of the full-body dynamics while reducing the dimensionality and complexity of the problem.

3.2 Two-legged planar model

This model represents a system formed by a rotorcraft and landing gear with two articulated robotic legs and a skid at the tip of each leg. This system can allow helicopters to safely land sideways on a slope or in terrains with two levels. It is divided in two main components: the main body and the legs. The first one includes the rotorcraft and the base of the landing gear, it contains most of the system's mass and provides a link to connect both legs, one on each side. The position and attitude of the body are controlled with the action of the legs which also transmit ground contact forces and moments to the body. The legs are attached to the main body at their respective hips and consist of two links connected by two revolute joints at the hip and the knee with its axes of rotation perpendicular to the YZ plane. A sketch of the landing gear is shown in Figure 3-1, where the main body is defined by its mass, m_B, and inertia, I_B. The distances D_x and D_y represent the horizontal and vertical distance from the CoM to the hips. The position of the CoM is defined by the vector $\mathbf{r_{cm}} = [y_B, z_B]$ and the roll angle θ. The upper leg segments have a mass, m_U, inertia, I_U, and length, l_U. The lower leg segments have a mass, m_L, inertia, I_L, and length, l_L. The joint angles are represented by q_{H_l}, q_{K_l}, q_{H_r} and q_{K_r} respectively. The vectors $\mathbf{r_{ch}}$ and $\mathbf{r_{hf}}$ represent the distances between the CoM and each hip, and between each hip and its respective foot. The forces acting on the system are the ground reaction forces, $\mathbf{f_i} = [f_y, f_z]$, the helicopter thrust force, $\mathbf{f_{th}}$, and the gravity, $\mathbf{g}$.

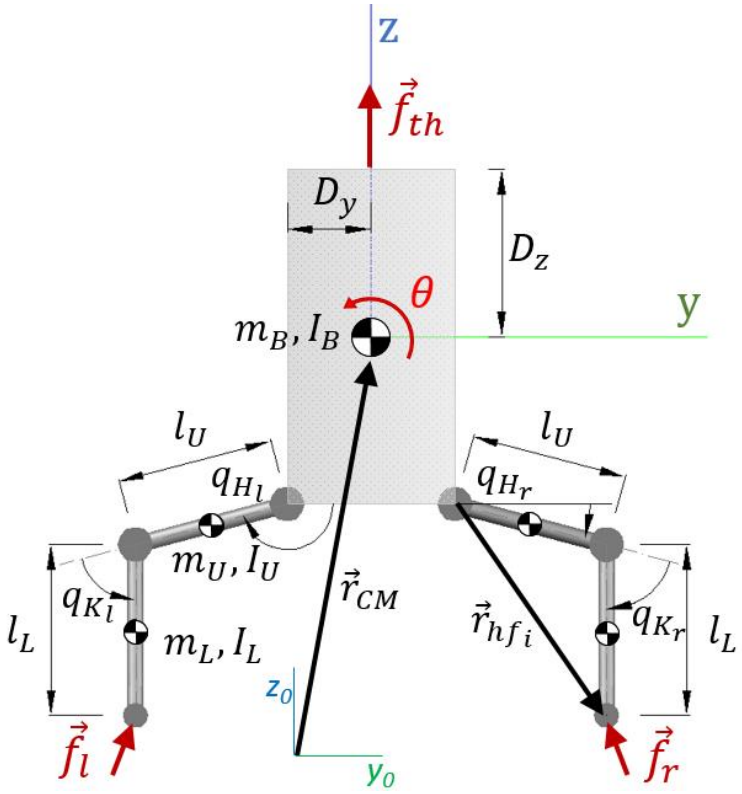

Figure 3-1 Sketch of the planar landing gear model.

To obtain the equations of motion of this planar model, a full-model approach has been adopted using Newtonian mechanics [53]. In this method, each link is analysed separately and there are three equation for each link: summation of forces in the horizontal and vertical directions and summation of torques about each link's CoM. The result is a system of 15 equations that express the acceleration of each link as a function of its position and velocity, the joint torques, external forces and internal constraints. Once the internal constraints are eliminated this gives a system of seven equations in terms of the state vector of generalised coordinates consisting of linear and angular acceleration of the main body and the four joint angles.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} + \mathbf{J}^{\mathbf{T}}(\mathbf{q})\mathbf{f} \tag{3.2}$$

where **q** is the generalised coordinates vector, $\mathbf{q} = [y_B, z_B, \theta, q_{H_l}, q_{K_l}, q_{H_r}, q_{K_r}]^T$, **M** is the inertia matrix, **C** is the matrix of centrifugal and Coriolis terms, **G** is the vector of gravity terms, $\boldsymbol{\tau} = [0, 0, 0, \tau_{H_l}, \tau_{K_l}, \tau_{H_r}, \tau_{K_r}]^T$ is the vector of joint actuator torques, $\mathbf{J}^{\mathbf{T}}$ is the transpose of the Jacobian matrix, and **f** is the vector of external forces.

Detailed below are the free body diagrams for each link of the system, and the derivation of the rigid body dynamic equations using Newton and Euler laws for linear and angular momentum.

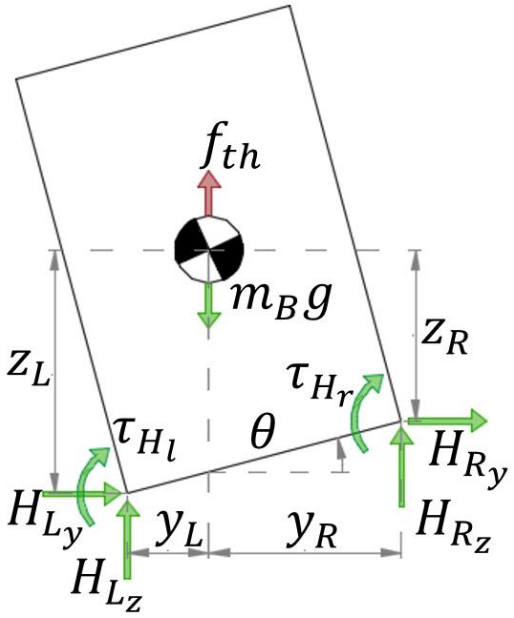

Figure 3-2 Free body diagram for the main body.

The Newton-Euler equations for the main body are:

$$m_B \ddot{y}_B = H_{L_y} + H_{R_y} + f_{th_y} \tag{3.3}$$

$$m_B \ddot{z}_B = H_{L_z} + H_{R_z} + f_{th_z} - m_B g \tag{3.4}$$

$$I_B \ddot{\theta} = H_{R_z} y_R - H_{L_z} y_L + H_{R_y} z_R + H_{L_y} z_L - \tau_{H_l} - \tau_{H_r} \tag{3.5}$$

where $\mathbf{H_L} = [H_{L_y}, H_{L_z}]$ and $\mathbf{H_R} = [H_{R_y}, H_{R_z}]$ are the reaction forces at the left and right hips respectively.

The distances from the CoM to the hips are defined by the vectors:

$$\mathbf{r_{ch_L}} = [y_L\ , z_L\ ] = [D_y c\theta - D_z s\theta\ , D_z c\theta + D_y s\theta] \tag{3.6}$$

$$\mathbf{r_{ch_r}} = [y_R\ , z_R\ ] = [D_z s\theta + D_y c\theta\ , D_z c\theta - D_y s\theta] \tag{3.7}$$

The abbreviations $s\theta$ and $c\theta$ are used to refer to the sine and cosine of the roll angle and analogue abbreviations will be used from now on to refer to the sine and cosine of all angles.

The accelerations at the hips are:

$$\ddot{y}_{H_R} = \ddot{y}_B + \left(D_z c\theta - D_y s\theta\right)\ddot{\theta} - \left(D_z s\theta + D_y c\theta\right)\dot{\theta}^2 \tag{3.8}$$

$$\ddot{z}_{H_R} = \ddot{z}_B + \left(D_z s\theta + D_y c\theta\right)\ddot{\theta} + \left(D_z c\theta - D_y s\theta\right)\dot{\theta}^2 \tag{3.9}$$

$$\ddot{y}_{H_L} = \ddot{y}_B + \left(D_z c\theta + D_y s\theta\right)\ddot{\theta} + \left(D_y c\theta - D_z s\theta\right)\dot{\theta}^2 \tag{3.10}$$

$$\ddot{z}_{H_L} = \ddot{z}_B + \left(D_z s\theta - D_y c\theta\right)\ddot{\theta} + (D_z c\theta + D_y s\theta)\dot{\theta}^2 \tag{3.11}$$

The Newton-Euler equations for the right leg are:

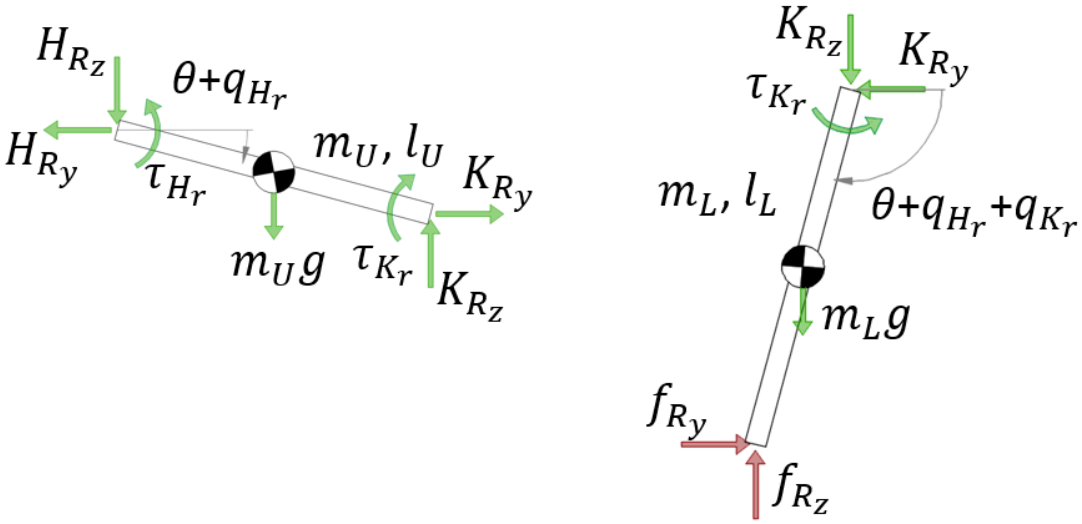

Figure 3-3 Free body diagram for the upper (left) and lower (right) right leg.

$$K_{R_y} - m_U \ddot{y}_{U_r} = H_{R_y} \tag{3.12}$$

$$K_{R_z} - m_U g - m_U \ddot{z}_{U_r} = H_{R_z} \tag{3.13}$$

$$I_U(\ddot{\theta} + \ddot{q}_{H_r}) = \tau_{H_r} - \tau_{K_r} + \left(K_{R_z} + H_{R_z}\right) l_{C_U} c\beta_{H_r} - \left(K_{R_y} + H_{R_y}\right) l_{C_U} s\beta_{H_r} \tag{3.14}$$

$$f_{R_y} - m_L \ddot{y}_{L_r} = K_{R_y} \tag{3.15}$$

$$f_{R_z} - m_L g - m_L \ddot{z}_{L_r} = K_{R_z} \tag{3.16}$$

$$I_L(\ddot{\theta} + \ddot{q}_{H_r} + \ddot{q}_{K_r}) = \tau_{K_r} + \left(f_{R_z} + K_{R_z}\right) l_{C_L} c\beta_{K_r} - \left(f_{R_y} + K_{R_y}\right) l_{C_L} s\beta_{K_r} \tag{3.17}$$

where $\mathbf{K_R} = [K_{R_y}, K_{R_z}]$ is the reaction force at the right knee. The positions of the upper and lower link CoM are defined by $\mathbf{r_{U_r}} = [y_{U_r}, z_{U_r}]$ and $\mathbf{r_{L_r}} = [y_{L_r}, z_{L_r}]$, the joint angles

are expressed as $\beta_{H_r} = \theta + q_{H_r}$ and $\beta_{K_r} = \theta + q_{H_r} + q_{K_r}$, and the distance from the extreme of a link to its CoM as $l_{C_i} = l_i/2$.

The accelerations at the CoM of the upper and lower links are:

$$\ddot{y}_{U_r} = \ddot{y}_{H_R} - l_{C_U} c\beta_{H_r} \dot{\beta}_{H_r}^{\ 2} - l_{C_U} s\beta_{H_r} \ddot{\beta}_{H_r} \tag{3.18}$$

$$\ddot{z}_{U_r} = \ddot{z}_{H_R} - l_{C_U} s\beta_{H_r} \dot{\beta}_{H_r}^{\ 2} + l_{C_U} c\beta_{H_r} \ddot{\beta}_{H_r} \tag{3.19}$$

$$\ddot{y}_{L_r} = \ddot{y}_{H_R} - l_u c\beta_{H_r} \dot{\beta}_{H_r}^{\ 2} - l_u s\beta_{H_r} \ddot{\beta}_{H_r} - l_{C_L} c\beta_{K_r} \dot{\beta}_{K_r}^{\ 2} - l_{C_L} s\beta_{K_r} \ddot{\beta}_{K_r} \tag{3.20}$$

$$\ddot{z}_{L_r} = \ddot{z}_{H_R} - l_u s\beta_{H_r} \dot{\beta}_{H_r}^{\ 2} + l_u c\beta_{H_r} \ddot{\beta}_{H_r} - l_{C_L} s\beta_{K_r} \dot{\beta}_{K_r}^{\ 2} + l_{C_L} c\beta_{K_r} \ddot{\beta}_{K_r} \tag{3.21}$$

The Newton-Euler equations for the left leg are:

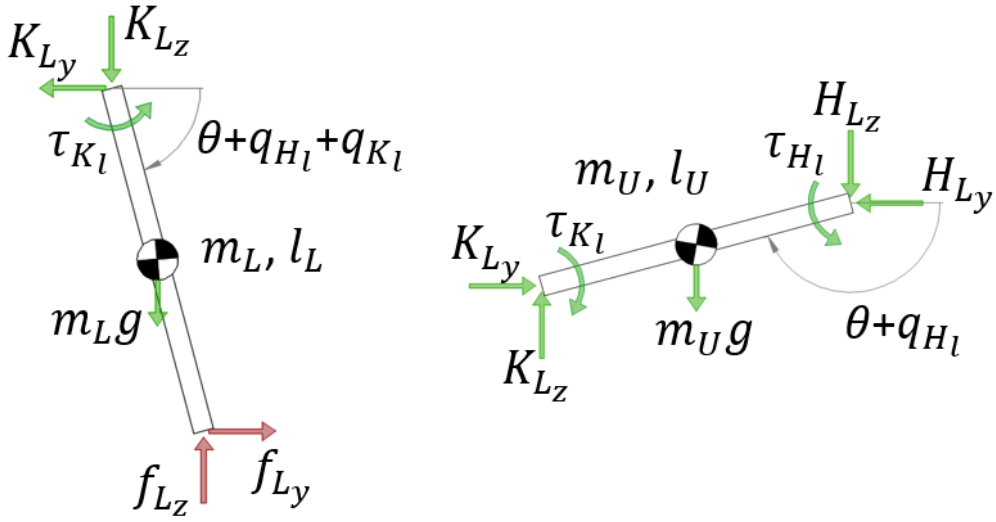

Figure 3-4 Free body diagram for the upper (right) and lower (left) left leg.

$$K_{L_y} - m_U \ddot{y}_{U_l} = H_{L_y} \tag{3.22}$$

$$K_{L_z} - m_U g - m_U \ddot{z}_{U_l} = H_{L_z} \tag{3.23}$$

$$I_U(\ddot{\theta} + \ddot{q}_{H_l}) = \tau_{H_l} - \tau_{K_l} + \left(K_{L_z} + H_{L_z}\right) l_{C_U} c\beta_{H_l} - \left(K_{L_y} + H_{L_y}\right) l_{C_U} s\beta_{H_l} \tag{3.24}$$

$$f_{L_y} - m_L \ddot{y}_{L_l} = K_{L_y} \tag{3.25}$$

$$f_{L_z} - m_L g - m_L \ddot{z}_{L_l} = K_{L_z} \tag{3.26}$$

$$I_L(\ddot{\theta} + \ddot{q}_{H_l} + \ddot{q}_{K_l}) = \tau_{K_l} + \left(f_{L_z} + K_{L_z}\right) l_{C_L} c\beta_{K_l} - \left(f_{L_y} + K_{L_y}\right) l_{C_L} s\beta_{K_l} \tag{3.27}$$

where $\mathbf{K_L} = [K_{L_y}, K_{L_z}]$ is the reaction force at the left knee. The positions of the upper and lower link CoM are defined by $\mathbf{r_{U_l}} = [y_{U_l}, z_{U_l}]$ and $\mathbf{r_{L_l}} = [y_{L_l}, z_{L_l}]$, and the joint angles are expressed as $\beta_{H_l} = \theta + q_{H_l}$ and $\beta_{K_l} = \theta + q_{H_l} + q_{K_l}$.

The accelerations at the CoM of the upper and lower links are:

$$\ddot{y}_{U_l} = \ddot{y}_{H_L} - l_{C_U} c\beta_{H_l} \dot{\beta}_{H_l}^{\ 2} - l_{C_U} s\beta_{H_l} \ddot{\beta}_{H_l} \tag{3.28}$$

$$\ddot{z}_{U_l} = \ddot{z}_{H_L} - l_{C_U} s\beta_{H_l} \dot{\beta}_{H_l}^{\ 2} + l_{C_U} c\beta_{H_l} \ddot{\beta}_{H_l} \tag{3.29}$$

$$\ddot{y}_{L_l} = \ddot{y}_{H_L} - l_u c\beta_{H_l} \dot{\beta}_{H_l}^{\ 2} - l_u s\beta_{H_l} \ddot{\beta}_{H_l} - l_{C_L} c\beta_{K_l} \dot{\beta}_{K_l}^{\ 2} - l_{C_L} s\beta_{K_l} \ddot{\beta}_{K_l} \tag{3.30}$$

$$\ddot{z}_{L_l} = \ddot{z}_{H_L} - l_u s\beta_{H_l} \dot{\beta}_{H_l}^{\ 2} + l_u c\beta_{H_l} \ddot{\beta}_{H_l} - l_{C_L} s\beta_{K_l} \dot{\beta}_{K_l}^{\ 2} + l_{C_L} c\beta_{K_l} \ddot{\beta}_{K_l} \tag{3.31}$$

The Newton-Euler equations are expressed in terms of the acceleration of the CoM of each link. By introducing the expressions of the accelerations into the Newton-Euler equations and eliminating the constraints, the full model is obtained. The detailed full model equations are presented in Appendix A.

The full-body model approach fully describes the relationship between the applied forces and moments and the resulting motion of the system. It is a suitable approach for relatively simple systems like planar models or systems with a small number of links. However, the complexity of the model and the number and length of the equations grows rapidly with the number of links and the degrees of freedom of the system. For more complex systems, it is preferable to use other approaches like Centroidal Dynamics. The methodology used has been the Newton-Euler as it presents an intuitive way to construct the equations of motion of the whole system link by link.

3.3 Four-legged model

This model represents a system formed by a rotorcraft and landing gear with four articulated robotic legs. This system can allow helicopters to land on more complex terrains than the two-legged version, including 2-axis slopes and irregular terrains with more than 2 levels. The main body includes the rotorcraft and the base of the landing gear, it contains most of the system's mass and provides a link to connect all four legs, two on each side.

As shown in Figure 3-5, the position and orientation of the main body is defined by the vector $\mathbf{r_{cm}} = [x_{cm}, y_{cm}, z_{cm}]$ that describes the position of the system's CoM with respect to the inertial frame and the roll (θ), pitch (φ) and yaw (ϕ) angles. Each leg is attached to the main body at its respective hip and consists of two links and two revolute joints at the hip and knee with its axes of rotation perpendicular to the YZ plane of the reference frame attached to the main body. The relative position of each hip with respect to the CoM is defined by the vector $\mathbf{r_{ch}}$, and the distance between each hip and its respective foot is given by the vector $\mathbf{r_{hf}}$. The forces acting on the system are the ground reaction forces at each foot, $\mathbf{f_i}$, the helicopter thrust force, $\mathbf{f_{th}}$, and the gravity $\mathbf{g}$.

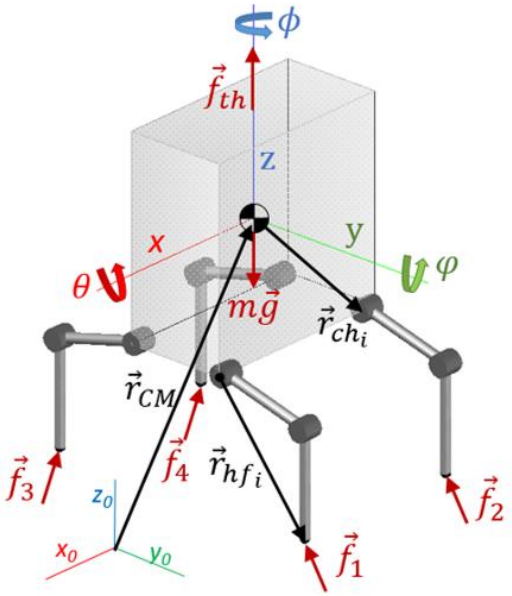

Figure 3-5 Sketch of the system with coordinate frames, important position vectors and external forces.

3.3.1 Whole Body Motion

The dynamic model used to generate the system's equations of motion is the *Centroidal Dynamics* or *Single Rigid-Body Dynamics* which has been used in quadruped locomotion [54], [55], [56]. This approach divides the system into two coupled dynamics equations, one for the floating-base and one for the *n* rigid bodies attached to it, in this case the four legs. First, the mass of each link is combined at the CoM and the inertia of each individual link is projected and expressed around the CoM to create a Centroidal Momentum Matrix (CMM). Thus, this centroid has equivalent mass and inertia properties to those of the whole body. Then, the robot Kinematic model is used to determine geometric parameters like the position of the end-effector, and the interaction between the robot and the environment, like determining the points where the external forces are applied with respect to the CoM.

This allows to obtain a set of equations that computes the motion of the centroid (and therefore the base) as a function of the external forces and doesn't depend on the links angular accelerations, resulting in a much simpler set of equations than the whole rigid body model. Second, once the trajectory of the centroid is known, the legs motion is calculated using the dynamic model of a single leg.

According to Single Rigid Body Dynamics, all the different bodies that form the system are reduced to a single point that has a mass and inertia that emulates those of the whole system in its nominal joint configuration and it's located at the CoM. Then, Newton-Euler equations for linear and angular motion are applied to this point.

$$m\ddot{\mathbf{r}}_{\mathbf{cm}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathbf{f_i} + \mathbf{f_{th}} - m\mathbf{g} \tag{3.32}$$

$$\mathbf{I}\dot{\mathbf{w}} + \mathbf{w} \times (\mathbf{I}\mathbf{w}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathbf{f_i}\, \mathbf{r}_{\mathbf{cf_i}} \tag{3.33}$$

where *m* is the combined mass of the whole system, *n* is the number of feet, **I** is the combined inertia of the main body and all legs in its initial joint configuration, expressed with respect to a coordinate frame situated at the system CoM and parallel to the inertial frame, **w** is the angular velocity of the system and $\mathbf{r}_{\mathbf{cf_i}}$ is the distance between the CoM and the foot *i* with respect to the inertial frame.

By expressing the inertia in the initial joint configuration, it is assumed that the mass of the legs is small compared with that of the main body and they do not move significantly from their initial position. This way we can express the dynamics of the robot in Cartesian coordinates only and the inertia matrix doesn't depend on the joint configuration.

The distance between the CoM and the foot *i* with respect to the inertial frame can be expressed as

$$\mathbf{r_{cf_i}} = \mathbf{R}(\mathbf{r_{ch_i}} + \mathbf{r_{hf_i}}) \tag{3.34}$$

where $\mathbf{r_{ch_i}}$ and $\boldsymbol{r_{hf_i}}$ are the CoM-hip and hip-foot distance vectors expressed in the body-fixed coordinate frame. **R** is the rotation matrix that transforms a point from the main body fixed coordinate frame to the inertial coordinate frame and it is expressed in terms of the roll (θ), pitch (φ) and yaw (ϕ) angles [27]:

$$\begin{aligned}\mathbf{R} = \mathbf{R_x}(\theta)\,\mathbf{R_y}(\varphi)\mathbf{R_z}(\phi) &= \begin{pmatrix}1 & 0 & 0\\ 0 & c\theta & -s\theta\\ 0 & s\theta & c\theta\end{pmatrix}\begin{pmatrix}c\varphi & 0 & s\varphi\\ 0 & 1 & 0\\ -s\varphi & 0 & c\varphi\end{pmatrix}\begin{pmatrix}c\phi & -s\phi & 0\\ s\phi & c\phi & 0\\ 0 & 0 & 1\end{pmatrix}\\ &= \begin{pmatrix}c\varphi c\phi & -c\varphi s\phi & s\varphi\\ s\theta s\varphi c\phi + c\theta s\phi & -s\theta s\varphi s\phi + c\theta c\phi & -s\theta c\varphi\\ -c\theta s\varphi c\phi + s\theta s\phi & c\theta s\varphi s\phi + s\theta c\phi & c\theta c\varphi\end{pmatrix}\end{aligned} \tag{3.35}$$

To obtain the orientation of the main body from the angular motion equation 3.32, the angular velocity **w** has to be expressed in terms of the Euler angles as well:

$$\mathbf{w} = \begin{pmatrix}w_x\\ w_y\\ w_z\end{pmatrix} = \begin{pmatrix}\dot{\theta}\\ 0\\ 0\end{pmatrix} + \begin{pmatrix}0\\ \dot{\varphi}\\ 0\end{pmatrix}\mathbf{R_x}(\theta) + \begin{pmatrix}0\\ 0\\ \dot{\phi}\end{pmatrix}\mathbf{R_x}(\theta)\mathbf{R_y}(\varphi) = \begin{pmatrix}\dot{\theta} + s\varphi\dot{\phi}\\ c\theta\dot{\varphi} - s\theta c\varphi\dot{\phi}\\ s\theta\dot{\varphi} + c\theta c\varphi\dot{\phi}\end{pmatrix} \tag{3.36}$$

To calculate the CMM, first the inertia of each individual link *i* with respect to the main body fixed-frame coordinates is calculated [57]. For this purpose, the principal moments of inertia of each link are first aligned with the axes of the main body fixed-frame and then the parallel axis theorem is applied to express the moments of inertia around the CoM.

$$\mathbf{I_i} = \mathbf{R_x}(q_i)\begin{pmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{pmatrix}\mathbf{R_x^T}(q_i) + m_i[\mathbf{d_i}\mathbf{d_i}\mathbf{1_3} - \mathbf{d_i}\otimes\mathbf{d_i}] \tag{3.37}$$

where I_{xx}, I_{yy} and I_{zz} are the principal moments of inertia of the link i, $\mathbf{R_x}(q_i)$ is the rotation matrix around the joint angle q_i, m_i is the mass of the link, d_i is the distance from the link's CoM to the system's CoM, $\mathbf{1_3}$ is a 3x3 identity matrix and $\otimes$ is the outer product.

The total inertia for each link expressed in the body-fixed coordinate frame is

$$\mathbf{I_i} = \begin{pmatrix} I_{xx} + m_i(d_y^2 + d_z^2) & -m_i d_x d_y & -m_i d_x d_z \\ -m_i d_x d_y & I_{yy}c^2 q_i I_{zz} s^2 q_i + m_i(d_y^2 + d_z^2) & (I_{yy} - I_{zz})cq_i sq_i - m_i d_y d_z \\ -m_i d_x d_z & (I_{yy} - I_{zz})cq_i sq_i - m_i d_y d_z & I_{yy}s^2 q_i + I_{zz}c^2 q_i + m_i(d_y^2 + d_z^2) \end{pmatrix} \quad (3.38)$$

The total inertia of the system around its CoM expressed in the body-fixed coordinate frame, $\mathbf{I_t}$, is obtained as the sum of the main body inertia, $\mathbf{I_b}$, and the inertia of each link, $\mathbf{I_i}$.

$$\mathbf{I_t} = \mathbf{I_b} + \sum_{i=1}^{i=8} \mathbf{I_i} \quad (3.39)$$

To use the inertia in equation 3.33 it needs to be converted into the inertial frame coordinates.

$$\mathbf{I} = \boldsymbol{R I_t R^T} \quad (3.40)$$

3.3.2 Single Leg Dynamics

In the previous section, the motion of the main body is computed using as inputs the external forces and the forward kinematics relating the distance between the foot and the system's CoM. In this section, the dynamic model of a single-leg is presented, and the inverted dynamics are used to calculate the joint torques needed to follow the desired foot trajectory considering the effect of the external forces.

Figure 3-6 shows a sketch of a robotic leg. Since the axes of rotation of the knee and hip joints are parallel and are in the same plane, the single-leg can be represented by a 2-DoF planar model using the ZY coordinate system at the hip. The joint angles at the hip and knee are represented by q_h and q_k respectively and the joint torques by τ_h and τ_k. All are positive in the anti-clockwise direction. l_U and l_L represent the length of the upper and lower links, and m_U, I_U and m_L, I_L are their respective masses and inertias. The ground contact forces are represented by $\mathbf{f_i}$.

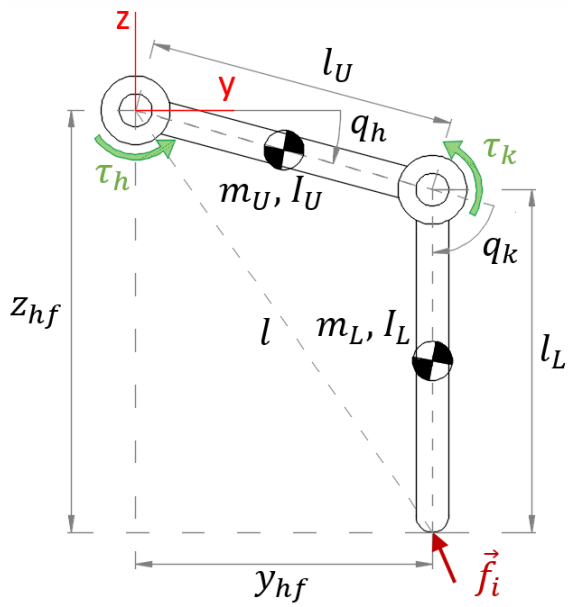

Figure 3-6 Sketch of a single leg with relevant parameters

To obtain the equations of motion of the single leg, the Lagrange methodology is used as it provides the equations of motion of the 2-DoF leg in its closed-form [58]. First, the positions of the masses in the main body coordinates are calculated assuming they are at the center of each link.

$$\mathbf{r_{m_U}} = \begin{bmatrix} y_{m_U} \\ z_{m_U} \end{bmatrix} = \begin{bmatrix} l_{C_U} c q_h \\ l_{C_U} s q_h \end{bmatrix} \tag{3.41}$$

$$\mathbf{r_{m_L}} = \begin{bmatrix} y_{m_L} \\ z_{m_L} \end{bmatrix} = \begin{bmatrix} l_U c q_h + l_{C_L} c(q_h + q_k) \\ l_U s q_h + l_{C_L} s(q_h + q_k) \end{bmatrix} \tag{3.42}$$

The velocities of m_U and m_L are calculated through the derivative of their positions, and the Lagrangian of the system is obtained through the kinetic and potential energy of each link as explained in Section 2.3.1.

$$\mathcal{L} = K - P = \sum \left(\frac{1}{2} m_i v_i^2 + \frac{1}{2} I_i \dot{q}_i^2 \right) - \sum (g m_i z_i) \tag{3.43}$$

Then, the Euler-Lagrange Equation (Equation 2.4.) is solved and the equations of motion are obtained and expressed in its canonical form in joint-space coordinates

$$\mathbf{M_i(q)} \begin{pmatrix} \ddot{q}_h \\ \ddot{q}_k \end{pmatrix} + \mathbf{C_i(q, \dot{q})} \begin{pmatrix} \dot{q}_h \\ \dot{q}_k \end{pmatrix} + \mathbf{G_i(q)} = \begin{pmatrix} \tau_h \\ \tau_k \end{pmatrix} + \mathbf{J^T(q) f} \tag{3.44}$$

with

$$\mathbf{M_i}(\mathbf{q}) = \begin{pmatrix} I_U + I_L + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L} c q_k\right) + m_U l_{C_U}^2 & I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L} c q_k\right) \\ I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L} c q_k\right) & I_L + m_L l_{C_L}^2 \end{pmatrix} \quad (3.45)$$

$$\mathbf{C_i}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{pmatrix} 0 & -m_L l_U l_{C_L} s q_k (2\dot{q}_h + \dot{q}_k) \\ m_L l_U l_{C_L} s q_k \dot{q}_h & 0 \end{pmatrix} \quad (3.46)$$

$$\mathbf{G_i}(\mathbf{q}) = \begin{pmatrix} (0.5 m_U + m_L) g l_U c q_h + m_L g l_{C_L} c(q_h + q_k) \\ m_L g l_{C_L} c(q_h + q_k) \end{pmatrix} \quad (3.47)$$

The term $\mathbf{J^T}(\mathbf{q})\mathbf{f}$ maps the effect of the ground reaction forces ($\mathbf{f} = [f_y \quad f_z]$) into the joint torques and the Jacobian $\mathbf{J}(\mathbf{q})$ is obtained through the partial derivatives of the forward kinematic equations for the foot.

$$\mathbf{r_{hf}} = \begin{bmatrix} y_{hf} \\ z_{hf} \end{bmatrix} = \begin{bmatrix} l_U c q_h + l_L c(q_h + q_k) \\ l_U s q_h + l_L s(q_h + q_k) \end{bmatrix} \quad (3.48)$$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial y_{hf}}{\partial q_h} & \frac{\partial y_{hf}}{\partial q_k} \\ \frac{\partial z_{hf}}{\partial q_h} & \frac{\partial z_{hf}}{\partial q_k} \end{bmatrix} = \begin{bmatrix} -l_U s q_h - l_L s(q_h + q_k) & -l_L s(q_h + q_k) \\ l_U c q_h + l_L c(q_h + q_k) & l_L c(q_h + q_k) \end{bmatrix} \quad (3.49)$$

The single-leg model represents the equations of motion of a 2-DoF planar manipulator with a fixed coordinate frame situated at the hip joint. In reality, the hip reference frame is attached to the main body and it moves with it, but here it is considered as a fixed reference frame, hence the velocity at the hip is zero. Because the velocity of the main body is relatively slow and is considered to be near level position at all times, it is assumed that the motion and orientation of the main body doesn't affect to the joint torques calculations.

3.4 Leg Inverse Kinematics

Leg inverse kinematics are used to convert the position commands from the controller into joint coordinates. The main controller operates in the Cartesian space as it is more intuitive to specify end-effector motion in that way [33].

The motion of each leg is confined inside a parallel plane to the YZ plane of the main body fixed-frame, so the distance between one hip and its respective foot expressed in the main body fixed-frame will be $\mathbf{r_{hf}} = [0, y_{hf}, z_{hf}]$. To maintain stability, the lateral hip-foot distance y_{hf} is maintained constant during all the landing process while only

the vertical distance z_{hf} is adjusted by the controller. Ignoring the x-coordinate as it is equal to 0, the hip-foot distance is expressed as

$$\mathbf{r_{hf}} = [y_{hf}, z_{hf}] = [y_{hf_0}, \quad z_{hf_0} + \Delta z_{hf}] \tag{3.50}$$

where y_{hf_0} and z_{hf_0} are the leg coordinates of the initial landing position and Δz_{hf} is the adjustment of the leg height due to the action of the controller as a function of the force at its respective foot and the attitude of the main body.

Control actions are performed in the joint space, so the hip-foot coordinates need to be converted into joint angles before sending them as inputs to the joint controllers or low-level controllers by using inverse kinematics equations for a 2-link planar manipulator as described in [30]

$$cq_k = \frac{{y_{hf}}^2 + {z_{hf}}^2 - l_U^2 - l_L^2}{2 l_U l_L} \tag{3.51}$$

$$sq_k = \pm\sqrt{1 - {cq_k}^2} \tag{3.52}$$

$$q_k = atan2(sq_k, cq_k) \tag{3.53}$$

$$q_h = atan2(z_{hf}, y_{hf}) - atan2(l_L sq_k, l_U + l_L cq_k) \tag{3.54}$$

These equations provide a straightforward way to compute the joint angles by knowing the Cartesian coordinates. By selecting the positive or negative sign in equation 3.52, the angles are solved for the elbow-up or elbow-down configurations.

3.5 External forces

The external forces that determine the whole body motion and the leg motion and joint torques are the ground reaction forces and the helicopter thrust force.

The thrust controller is a simplified version of the one used in [16] where the only aim is to control the descent rate of the helicopter

$$\mathbf{f_{th}} = \mathbf{R}[C \cdot (\dot{z}_{CM_d} - \dot{z}_{CM}) + m\mathbf{g}] \tag{3.55}$$

where *C* is a constant and $\dot{z}_{CM_d}$ is the desired descent rate.

The ground contact model simulates the interaction forces between the feet of the landing gear and the ground. By using the centroidal dynamics model, the position of

the system's CoM is known, and using whole body kinematics the position of each feet in the inertial frame coordinates, $\boldsymbol{r}_{f_i}$, is determined.

$$\mathbf{r_{f_i}} = \mathbf{r_{cm}} + \mathbf{r_{cf_i}} = \mathbf{r_{cm}} + \mathbf{R}(\mathbf{r_{ch_i}} + \mathbf{r_{hf_i}}) \tag{3.56}$$

In order to calculate the ground reaction forces, first a ground surface has to be defined. In this project, two types of irregular landing surfaces have been considered. One is a slope, and it´s defined by the slope (α_{SL}) and azimuth angles (α_{AZ}), and another one composed of several flat surfaces at different ground levels. In the second case, the surface is defined by the elevation of each surface.

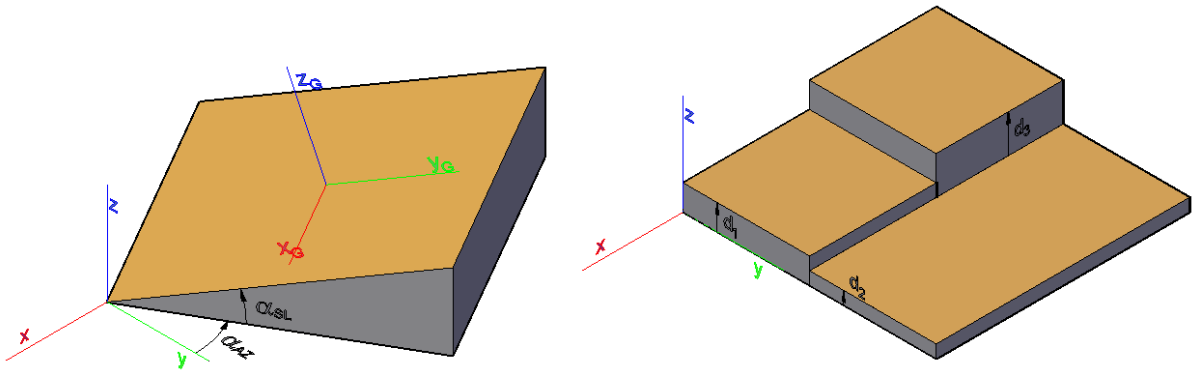

Figure 3-7 Irregular terrains: sloped surface (left) and multi-level ground (right)

Second, the foot position has to be defined in the ground coordinate system. In the stepped surface (Figure 3-7, right), the distance from each feet to the ground is defined by the z distance to the surface below each foot. In the slopped surface (Figure 3-7, left), the distance from each feet to the ground below is defined in the normal direction to the surface (z_G). In this case, the foot position in ground coordinate system is defined by

$$\mathbf{r_{f_i}^G} = \mathbf{R_G}\mathbf{r_{f_i}} \tag{3.57}$$

where $\mathbf{r_{f_i}^G}$ is the position of the i_{th} foot in the ground coordinates and $\mathbf{R_G}$ is the rotation matrix around the slope angles.

The initial ground contact points are defined as the points where the distance to the ground of each foot (in the z_G direction) changes from positive to zero: $\mathbf{p_0} = (x_0, y_0, 0)$.

In this project a compliant contact model is used with a non-linear spring-damping model to generate forces that tend to hold the foot in the initial contact position and simulates the deformation of the contact bodies.

The normal force (in the z-axis of the ground coordinate system) is given by the equation

$$f_z^G \begin{cases} -z_f^G \cdot k_z - \dot{z}_f^G \cdot b_z(z_f^G) & if\ z_f^G \leq 0 \\ 0 & if\ z_f^G > 0 \end{cases} \quad (3.58)$$

where f_z^G is the value of the ground reaction force in the z-axis of the ground coordinate frame and k_z and b_z are the spring and damper coefficients in the z direction.

To avoid the instantaneous damping force at the initial moment of touchdown, the damping coefficient is a function of ground penetration as introduced in [59]

$$b_z(z_f^G) \begin{cases} \left(z_f^G / h_{max}\right) \cdot b_{max} & if\ z_f^G < h_{max} \\ b_{max} & if\ z_f^G > h_{max} \end{cases} \quad (3.59)$$

where $b_z(z_f^G)$ varies linearly from 0 to the maximum damping coefficient b_{max} as the penetration depth increases from 0 to the maximum penetration depth h_{max}.

Thus, normal force is proportional to the amount of ground penetration (spring component) and velocity (damper component) of the foot during touchdown. It is 0 before the moment of touchdown ($z_f^G = 0$) and builds up as the foot penetration increases. To avoid “sticking forces”, it can only have positive values. In simulations, a saturation block is used to avoid negative forces that try to stick the foot to the ground, which would be physically incorrect.

This model is straightforward to implement and it’s a good representation of the reality as it avoids sticking forces and discontinuities at the moments when the foot makes and loses contact. The spring and damper coefficients are calculated using equations 2.12 and 2.13.

Friction force is modelled on the x and y-axes of the ground coordinate frame and it opposes the sliding of the feet on the slopped surface. The friction force has a “stick” region where it is modelled as a spring-damper and a “slip” region where the value of the force exceeds the maximum static friction force. If the model switches to “slip” mode, the friction force is equal to the value of the dynamic friction force.

$$f_x^G \begin{cases} -(x_f^G - x_{f_0}^G) \cdot k_x - \dot{x}_f^G \cdot b_x & if\ f_x^G < \mu_s f_z^G \\ \mu_D f_z^G & if\ f_x^G \geq \mu_s f_z^G \end{cases} \quad (3.60)$$

$$f_y^G \begin{cases} -(y_f^G - y_{f_0}^G) \cdot k_y - \dot{y}_f^G \cdot b_y & if\ f_y^G < \mu_s f_z^G \\ \mu_D f_z^G & if\ f_y^G \geq \mu_s f_z^G \end{cases} \quad (3.61)$$

Friction force is 0 if $z_f^G > 0$, and $x_{f_0}^G$ and $y_{f_0}^G$ are the x and y coordinates at the moment of touchdown. Thus, friction force acts like two spring-dampers opposing movement on the sloped surface. μ_s and μ_D are the static and dynamic friction coefficients and k_x, k_y, b_x and b_y are the respective spring and damper coefficients on each direction.

The reason to use a compliant spring-damper model to simulate friction force is because in most Coulomb models the friction force is 0 when the sliding velocity is 0. When modelling a sloped landing, the normal force (perpendicular to the slope) has a vertical and horizontal component, so if the friction force is 0, the horizontal component of the normal force will accelerate the system downslope until the sliding velocity makes the friction force large enough to stop it, thus, the system would be constantly accelerating and stopping. The spring-damper system depends on the sliding velocity and the distance between the current foot position and the initial contact point, thus, even when the sliding speed is zero the force will be non-zero.

To apply the ground reaction forces into the system's dynamic equations, they have to be converted to the inertial frame coordinate system.

$$\mathbf{f_i} = \mathbf{R_G^T f_i^G} \quad (3.62)$$

where $\mathbf{f_i} = [f_{x_i};\ f_{y_i};\ f_{z_i}]$ and $\mathbf{f_i^G} = [f_{x_i}^G;\ f_{y_i}^G;\ f_{z_i}^G]$.

3.6 System Model

Figure 3-8 shows a view of the implementation of the whole 4-legged model into Simulink. The main blocks that compose the whole model are: one block that simulates the motion of the main body, one block to simulate the motion of each of the legs, one block to simulate the ground reaction forces at each foot, and one block that includes the controller.

In the model also exist different coordinate frames, namely the inertial frame, main body coordinate frame, the legs coordinate frame, and the ground coordinate frame.

When sending data from one block to another appropriate coordinate transformations will be applied using rotation matrices.

A brief explanation of the model implementation is given in this section, although the Matlab/Simulink files for the models are included in the CD attached to this thesis.

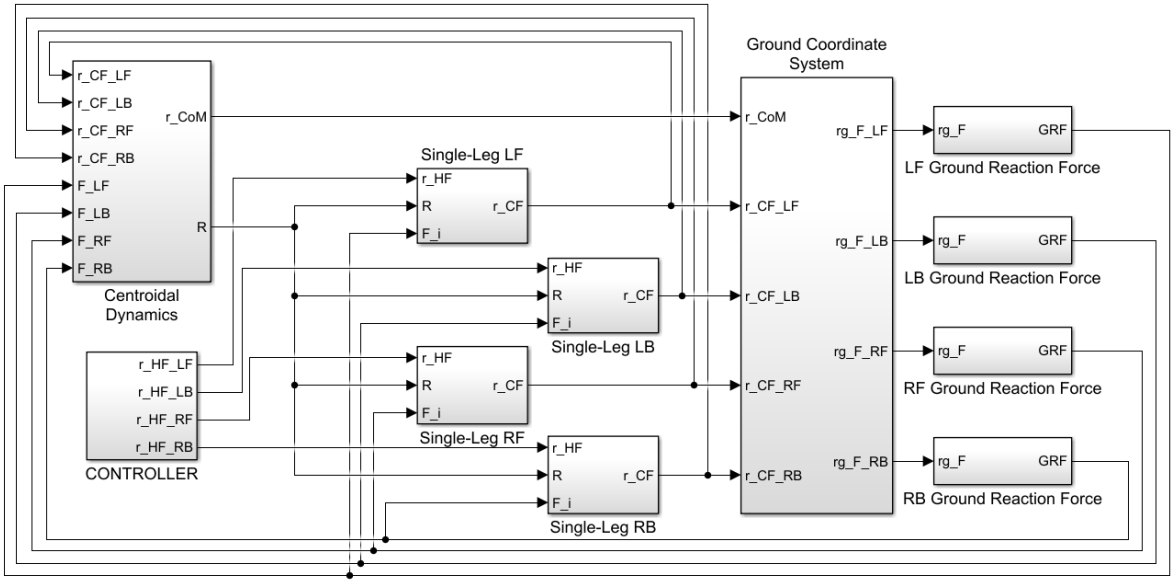

Figure 3-8 Simulink view of the whole system integration

3.6.1 Main body motion block (Centroidal Dynamics)

This block calculates the motion of the main body of the landing gear. It takes the ground reaction forces and the position vector from the CoM to each foot as the inputs and returns the position and orientation of the CoM.

In the 4-legged model, this block includes the equations for the Centroidal Dynamics, described in section 3.3.1, as shown in Figure 3-9. The CMM is calculated offline using a Matlab script (see Appendix B). The Newton-Euler equations provide 6 equations to calculate the 3 linear accelerations and 3 angular accelerations of the system, and the Euler angles are obtained using equation 3.36. It also includes equation 3.55 for the thrust force to regulate the descending velocity.

In the 2-legged model, the motion of the main body is modelled by implementing equations A.2-A.4 (See Appendix A)

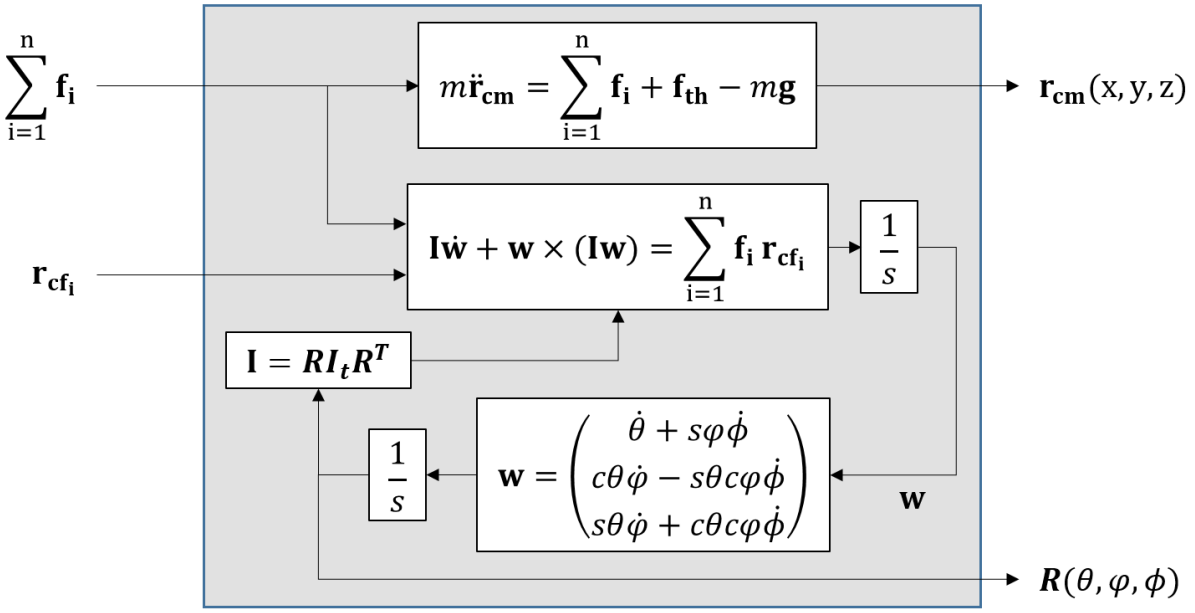

Figure 3-9 Centroidal Dynamics Block Diagram

3.6.2 Legs motion blocks

The legs models are used to calculate the required joint torques to move each joint to the desired position as shown in Figure 3-10. The inputs to the block are the ground reaction forces and the desired leg position from the controller. The block returns the actual leg position.

In the 4-legged model, this block includes the equations for the Single-Leg Dynamics, described in section 3.3.2. In the 2-legged model, this block includes equations A.5-A.8 (See Appendix A). There is one block per leg.

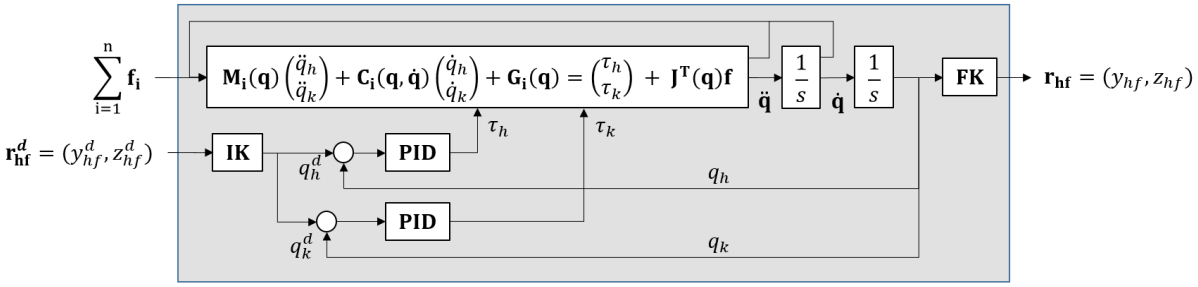

Figure 3-10 Single-leg Block Diagram. The abbreviations IK and FK refer to Inverse and Forward Kinematics respectively

3.6.3 Ground Reaction Forces Blocks

The ground reaction force block implements the equations for the ground contact model described in section 3.5 as shown in Figure 3-11. The input to the block is the

position vector of the foot which is previously converted to the ground coordinate frame. The output of the block is the vector of resultant ground reaction forces.

In the case of the 4-legged model, there will be 4 blocks (one per leg) with 3 forces per foot (in the X, Y, and Z directions). In the case of the 2-legged model, there will be 2 blocks (one per leg) with 2 forces per foot (in the X, and Y directions).

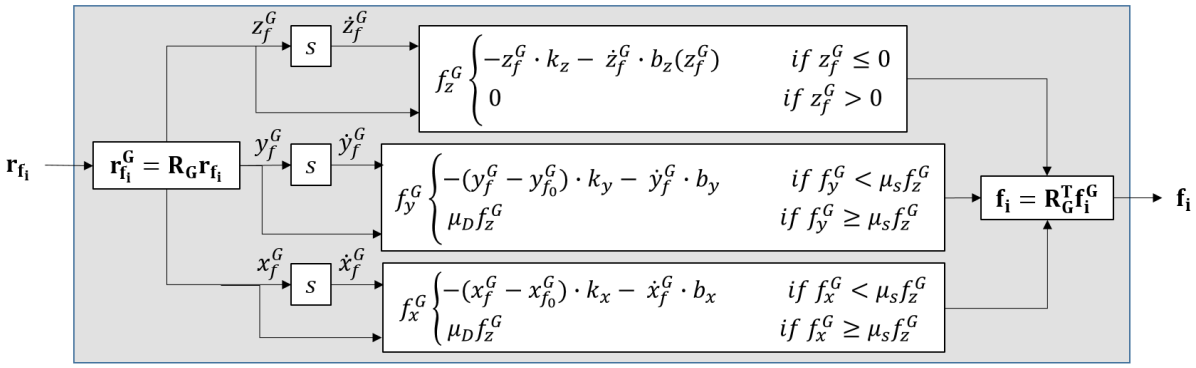

Figure 3-11 Ground Contact Model Block Diagram and transformations between ground and inertial coordinate frames.

3.6.4 Controller Block

The controller block includes the control system, which will be introduced in chapter 5. The controller block takes the Euler angles and feet pressure as inputs and sends position commands to the legs in the form of Cartesian coordinates.

3.7 Conclusions

This chapter presented the methodology to obtain the dynamic equations of the system. First a planar model of the rotorcraft plus landing gear was obtained using a full-body model, and then a three-dimensional model of the system was obtained by using two decoupled models of the centroidal dynamics and the single leg dynamics. The external forces, consisting in the rotor thrust and ground contact models are also modelled. Finally, an explanation of the implementation of the model in Matlab/Simulink is presented.

4 Prototype Design

This chapter presents the robotic landing gear design, including the mechanical design of the main body, legs and feet, the sensory system, actuators and electrical equipment, and the control software. Figure 4-1 shows a top view of the landing gear (a) and a view of the system attached to a model helicopter Align T-Rex 500.

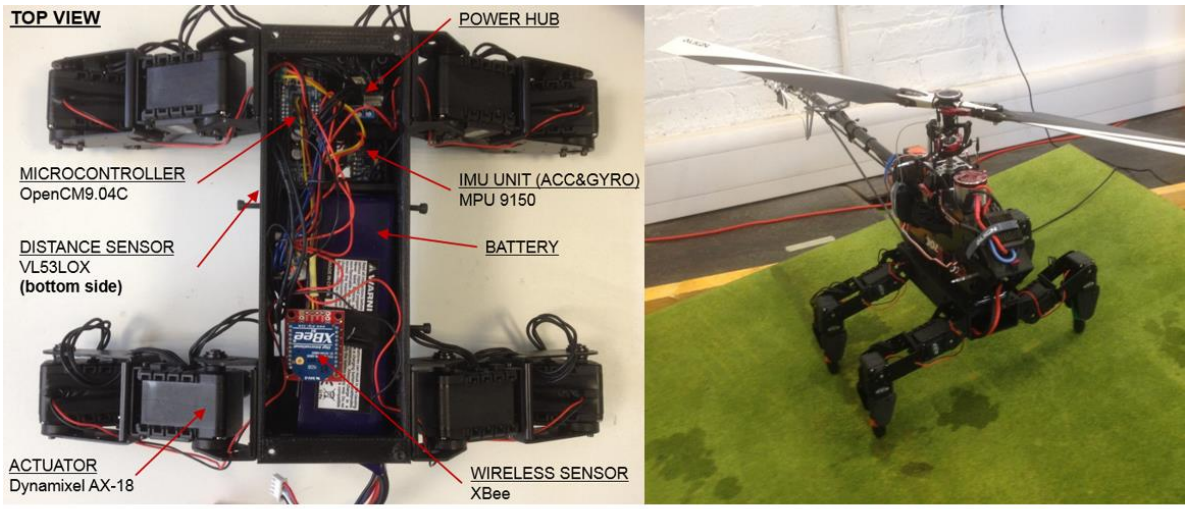

Figure 4-1 (a) top view of the landing gear with its main components and (b) view of the landing gear attached to the model helicopter on a slope landing.

4.1 System Overview

The robotic landing gear consists of four legs with two electrically actuated joints each. Its dimensions are designed proportionately to the model helicopter Align T-Rex 500, which is used as a platform for testing, and its mechanical structure is built as a combination of 3D-printed parts and aluminium frames. An On-board microprocessor controls the motion of all joints and uses feedback from force sensors, an inertial measurement unit (IMU) and a distance sensor to stabilise the system during landing. Table 4.1. shows an overview of the main system specifications. More details and justification of the components and parameters is given over chapters 4 and 6.

Table 4-1 General System specifications

Property	Value
Dimensions (fully stretched legs)	505x205x38 mm
Weight	1050g
Active DOF	8; 2 per leg
Actuators	Dynamixel AX-18 servo motors
Onboard sensors	IMU, force, distance
Onboard controller	Robotis OpenCM9.04
Control frequency	20 Hz

4.2 Mechanical Design

The mechanical structure of the system uses some off-the-shelf components and some custom-made parts. After a survey of available products in the market, the team decided to use the Dynamixel series servo motors and all its product family of controllers, brackets and other mechanical and electrical accessories. Compatible brackets have been used where possible, but some parts had to be designed and built to meet the specific requirements of the project. Figure 4-2 shows the top view of the mechanical structure of the system with its legs stretched. It shows the four identical legs with two servo motors each, where one servo motor at the hip joint connects the main body with the upper leg and another servo motor at the knee joint connects the upper and lower leg segments. All legs are attached to the main body through its respective hip and are identified as left front (LF), left back (LB), right front (RF) and right back (RB) legs respectively.

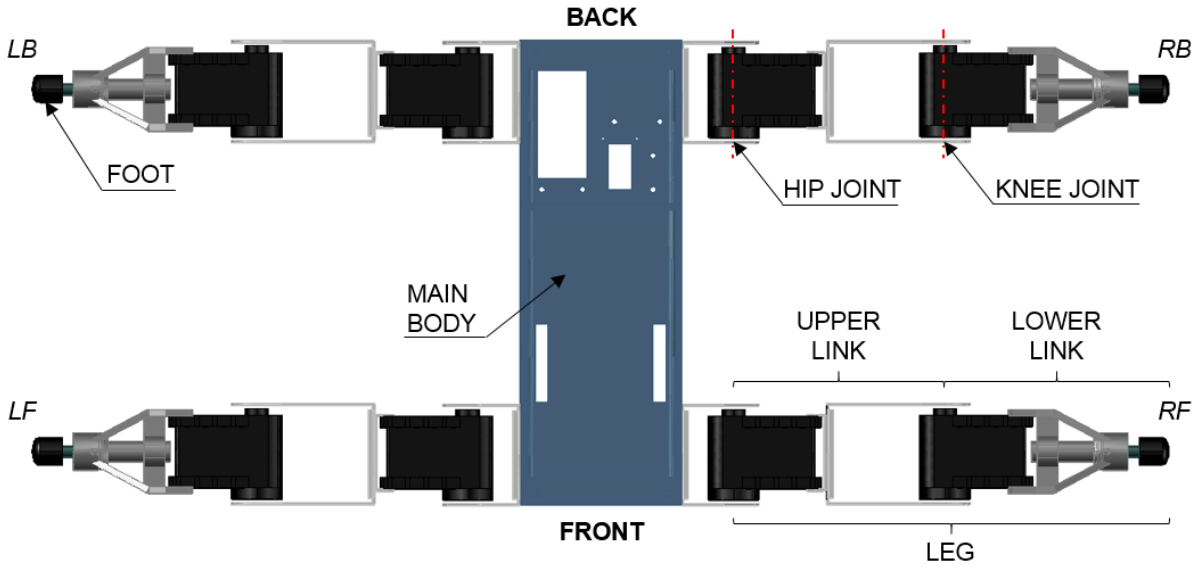

Figure 4-2 CAD drawing of the mechanical structure of the system

4.2.1 Main Body

The principal function of the main body is to carry most of the robot components as well as providing a rigid support for the legs. It is custom-made through a rapid prototyping process, using a CAD software to design the part and a 3D printer to manufacture it. The base provides enough space to accommodate the battery, microcontroller, IMU, distance sensor, wireless transmitter, power hub and all the circuitry. Most of these components are rigidly attached by means of screws. The legs are attached to the lateral sides through the aluminium brackets F2 (shown in Figure 4-5). A lid is mounted on top of the base to protect the electronic components (Figure 4-3). The landing gear is attached to the helicopter by means of screws on the lid. The main body is lightweight, helping to keep the total weight low and strong enough to resist impacts and protect the internal components.

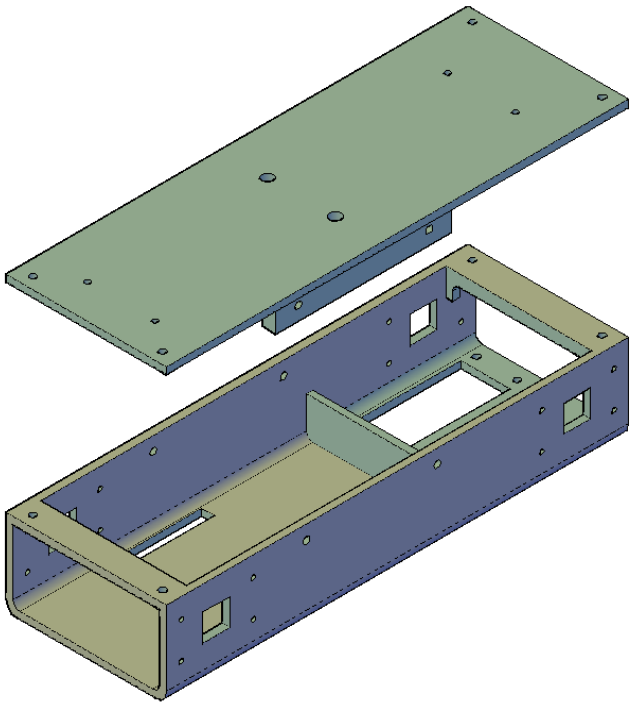

Figure 4-3 CAD view of the main body including the base and the lid

4.2.2 Robotic Legs

The main function of the legs is to keep the helicopter level during landing and take-off operations, plus they need to fold up during flight. In order to meet this requirements, while keeping the design simple, the adopted solution has been the one with two joints at the hip and the knee moving in the same plane, providing 2 DOF to each leg.

The upper link servo motor is attached through its moving horn to the bracket F2 (shown in Figure 4-5), which connects it to the main body. The body of the servo motor is assembled to the brackets F1 and F4 to connect it to the lower link servo motor horn. The lower link uses a custom-made 3D printed part assembled to the servo motor body to accommodate the foot. Both servo motors are part of the leg structure (see Figure 4-4).

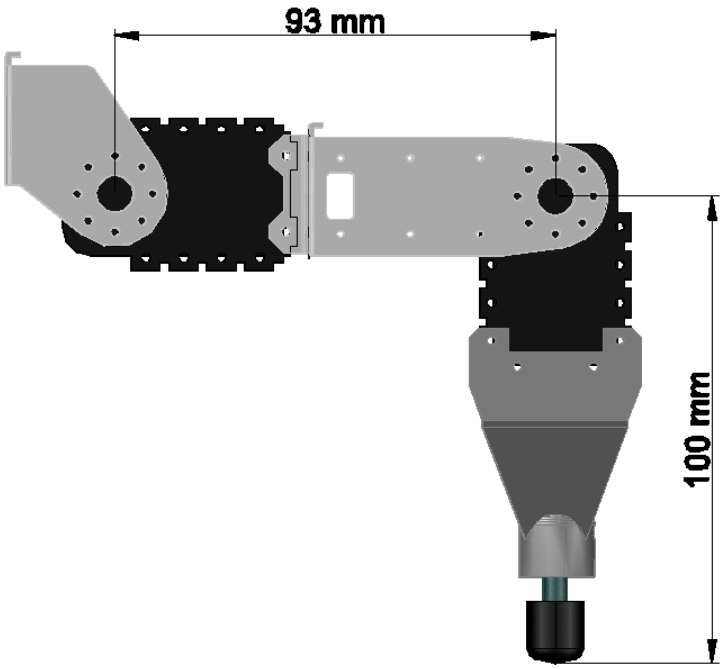

Figure 4-4 CAD drawing of a single leg.

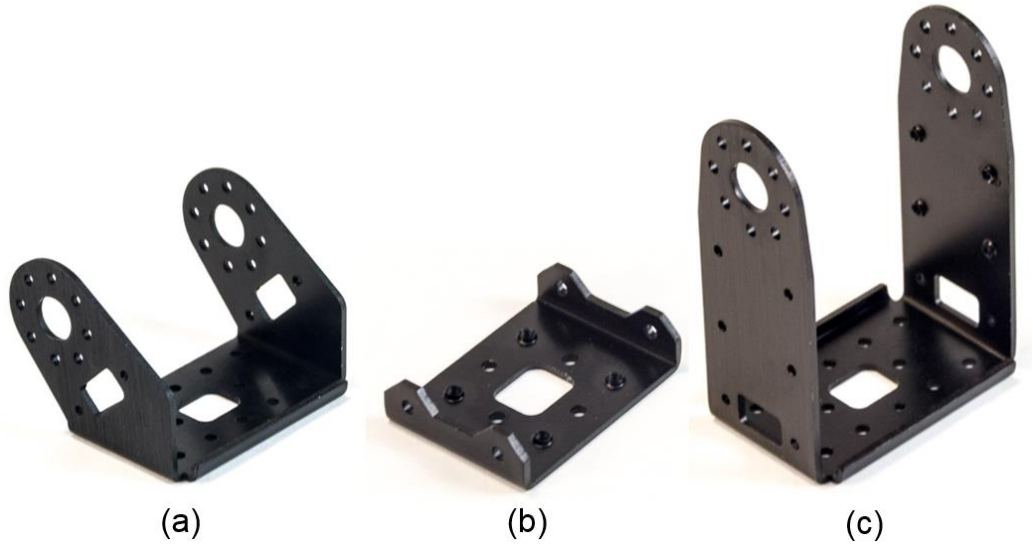

Figure 4-5 BIOLOID AX compatible metal brackets F1 (a), F2 (b) and F4 (c) [60]

4.2.3 Lower leg and Feet

The foot is an important part of the robot since it's where it happens the interaction with the ground. It needs to provide good contact with the ground, avoiding slipping and accommodate the force sensors to provide feedback to the system.

The force feedback is crucial for the good operation of the system. It must provide not only the status of the leg with the ground (contact or not contact) but also an analog value that is proportional to the weight supported by the leg. This way, the controller

can give different response depending on the measured force and a minimum force threshold can be set to detect ground contact.

The first solution proposed for the force sensing was to use the internal torque sensing feature of the servo motors, but this option was discarded due to the poor reliability of the torque signal. As the manufacturer points out in its online manual [61], this parameter is not a measured value and should be only used to predict the direction of the force applied to the motor. It is also not clear how the parameter is calculated. It was noticed that this feedback signal when the motor was in a fixed position, was somehow proportional to the torque that the motor was holding. However, when the servo is moving, this correlation is lost, and the feedback signal becomes unusable.

The adopted solution instead was inspired by Lynxmotion, a manufacturer of robot kits which provides solutions for hexapod foot contact sensors [62] [63]. The first requirement is met by using a rubber cap to cover the metal foot, which provides a good grip with the surface.

For the force sensor, a Force Sensing Resistor (FSR) is used [64]. In a first design, the foot was formed by a FSR placed on top a metal tube, and then covered with a rubber bumper and a rubber cap as shown in Figure 4-6. The metal tube then was attached to a 3D printed lower leg and this assembled to the servo motor.

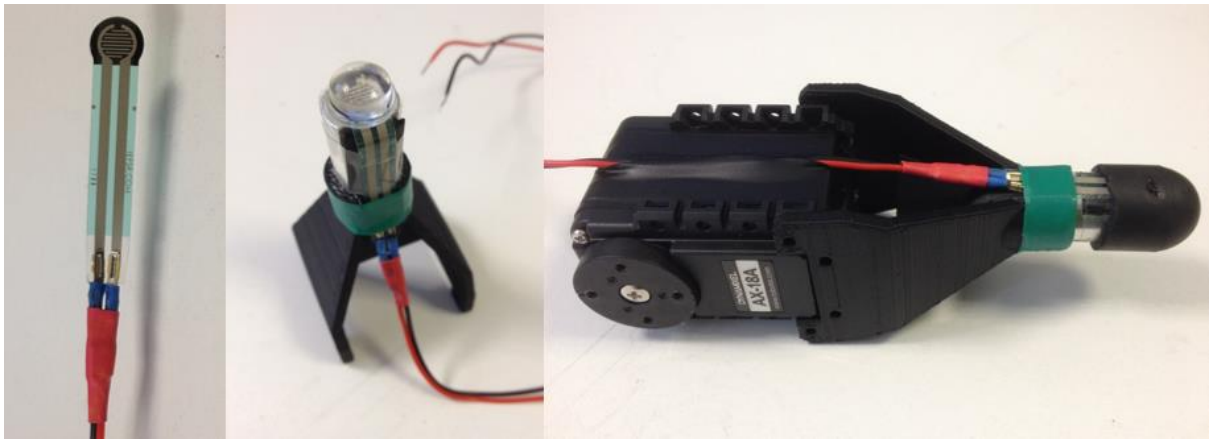

Figure 4-6 First foot design.

After initial testing, it was observed that the sensors were too exposed and got damaged frequently, needing replacement. Also, the readings of the FSR were affected by the way each foot was assembled and the pressure exerted by the rubber bumper and cap, which provided inconsistent readings. To solve this problems, the lower leg and foot design was improved. In the new version, the FSR is placed on the

bottom side of servo motor, more protected. When the foot touches the ground, it acts as a plunger, pushing a moving cylinder that slides through the lower leg body and presses the sensor. As shown in Figure 4-7, a spring connecting the plunger with the cylinder dampens the force transmitted to the sensor. A rubber bumper is placed between the cylinder and the force sensor. The tests done with the second design have given better results so far. The sensors didn't suffer any deterioration and the readings are more consistent and repeatable. The addition of the spring also provides some natural compliance to the legs.

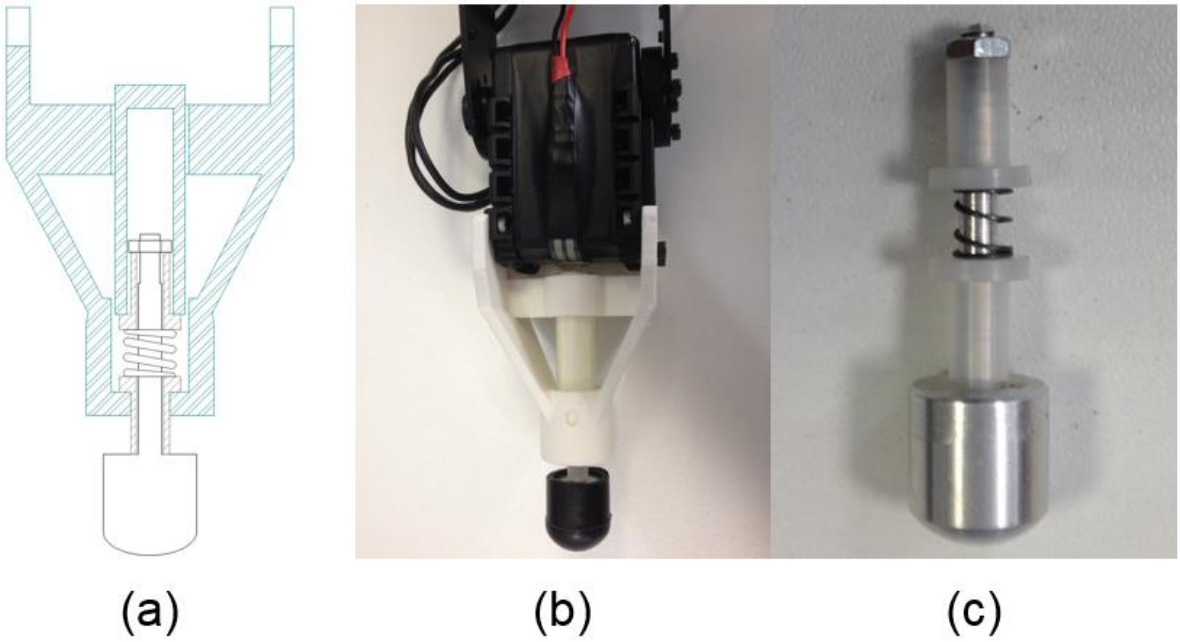

Figure 4-7 CAD drawing (a) and picture (b) of the second foot and lower leg design and spring element (c).

4.3 Actuators

The Dynamixel servo motors from the company Robotis have been chosen for its good performance and extended use among similar research projects, low cost, and because they offer a full product family of compatible accessories. The model used in this project is the Dynamixel AX-18A, which incorporates in a single package a gear reducer, a precision DC motor and a control circuitry with networking functionality. The design of the motor is robust and, despite its compact size, it can produce high torques. They come with an internal controller that allows for accurate position and speed control with a resolution of 10 bits and provides feedback for angular position, angular velocity and load torque. It can also measure internal temperature and voltage and has an alarm system to prevent damage.

Table 4-2 Dynamixel AX-18A general specifications [65]

Item	Specifications
Baud Rate	7843 bps ~ 1 Mbps
Resolution	0.29°x1024 (0~300°)
Weight	55.9g
Dimensions (W x H x D)	32mm x 50mm x 40mm
Stall Torque	1.8 N*m (at 12V, 2.2A)
Input Voltage	9.0 ~ 12.0V (Recommended : 11.1V)
Protocol Type	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	TTL Level Multi Drop Bus
ID	0 ~ 253
Feedback	Angular position, angular velocity, load torque, temperature and input voltage
Material	Engineering Plastic

The physical communication of the servos is a 3-pin TTL connection and a multidrop bus, so many devices can be controlled using a single bus.

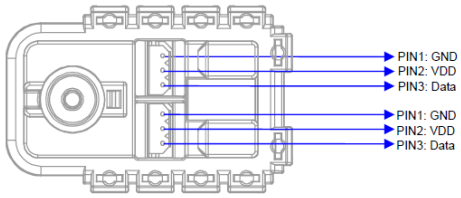

Figure 4-8 Dynamixel AX series connection [65]

The servos can be daisy chained, reducing the amount of wiring required and each servo connected to one bus need to have a different ID value so the controller can select which device is operating. The communication protocol is made through half duplex asynchronous serial communication with 8 bit, stop bit and no parity.

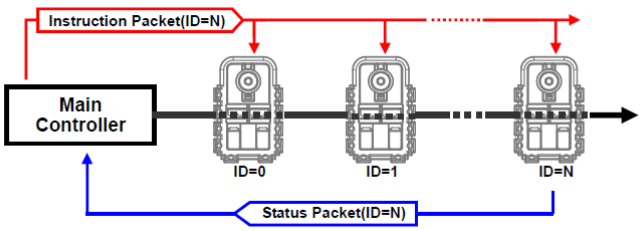

Figure 4-9 Multiple Dynamixel network [65]

AX-series controllers are compatible with Matlab, Labview, VB.NET, C#, Python and Java and there are available many libraries and code examples.

4.3.1 Dynamixel Control Table

Dynamixel AX-12/18 smart servos incorporate an internal control circuitry to control its operation by writing/reading data via Instruction Packets into specific addresses of the control table. The control table is the area of the memory that contains information on the status and operation of the servos. As seen in Figure 4-10, data can be stored in RAM or EEPROM memory and each memory address or register contains information relative to different parameters.

	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	12(0x0C)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	1(0x01)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
	7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
	8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
	9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
	15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	Alarm LED	RD,WR	4(0x04)
	18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Down Calibration(L)	RD	?
	21(0X15)	Down Calibration(H)	RD	?
	22(0X16)	Up Calibration(L)	RD	?
	23(0X17)	Up Calibration(H)	RD	?
RAM Area	24(0X18)	Torque Enable	RD,WR	0(0x00)
	25(0X19)	LED	RD,WR	0(0x00)
	26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
	27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
	28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
	29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
	30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
	31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
	32(0X20)	Moving Speed(L)	RD,WR	0
	33(0X21)	Moving Speed(H)	RD,WR	0
	34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
	35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
	36(0X24)	Present Position(L)	RD	?
	37(0X25)	Present Position(H)	RD	?
	38(0X26)	Present Speed(L)	RD	?
	39(0X27)	Present Speed(H)	RD	?
	40(0X28)	Present Load(L)	RD	?
	41(0X29)	Present Load(H)	RD	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46[0x2E]	Moving	RD	0(0x00)
	47[0x2F]	Lock	RD,WR	0(0x00)
	48[0x30]	Punch(L)	RD,WR	32(0x20)
	49[0x31]	Punch(H)	RD,WR	0(0x00)

Figure 4-10 Dynamixel control table [65]

In this section, the most used parameters are explained.

ID

Each servo connected to the system has a unique ID number, so the microcontroller can send instructions to each servo individually by specifying its ID. The ID number is

stored in register 3 and can be in the range of 0 to 252 and, the number 254 is used to send an instruction to all the Dynamixels.

Goal Position and Present Position

This register is used to command the servo to move to a specified angular position. The goal position can be in the range of 0 to 1023 and it's a 2-byte register, being 511 the central position at 150°, and 0 and 1023, the minimum and maximum limits at 0° and 300° respectively. The area from 300° to 360° is a dead zone when the servo is in joint mode. The present position register is used to read the current angular position of the servo.

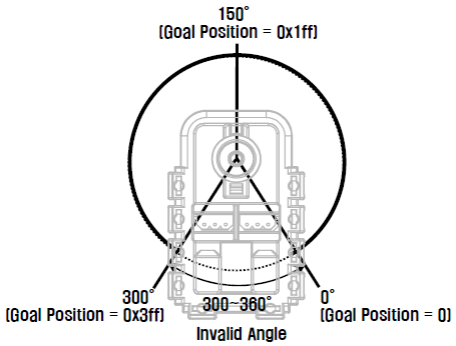

Figure 4-11 Servo angular positions [65]

Clockwise/Counter-clockwise (CW/CCW) Angle limit

This registers are used to limit the operating angle range of the servo, so the goal position always needs to be within the range between the clockwise angle limit and counter-clockwise angle limit.

Moving speed and present speed

The moving speed register sets the angular velocity at which the actuator is moving to its goal position. The velocity range goes from 0 to 1023, being 1 the minimum speed and 1023 being 114 rpm. When set to 0, the velocity is the largest possible without applying any velocity control. The present speed register is used to read the current angular velocity of the servo.

Present load

It represents the magnitude of the load applied to the Dynamixel actuator. It's a 10-bit value and its range is from 0 to 2047, being the values between 0-1023 loads applied

in the CCW direction and 1024-2047 loads applied in the CW direction. Thus, the 10th bit of the register represents the sign and the value 1024 represents a load of value 0.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Figure 4-12 Load register binary representation [65]

According to the manufacturer, the load value is an estimation, not a direct torque measurement. For this reason, it cannot be used to measure weight or torque, but only to detect in which direction the force works.

Compliance

The compliance defines the flexibility of the motor and how the output shaft absorbs shocks. Figure 4-13 shows the relationship between output torque and servo position.

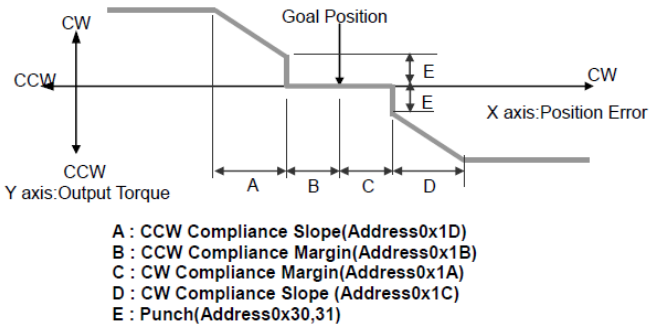

Figure 4-13 Dynamixel compliance setting [65]

The compliance margin is the error allowed between the present position and the goal position before the motor starts applying torque. The slope defines the rate of increase of torque as the present position moves away from the goal position. The punch is the minimum current supplied to the motor during operation.

Torque enable

This register enables/disables the generation of torque. When set to 0, the power to the motor is interrupted.

4.4 Range of Motion and Safety Considerations

The range of motion of all joints has to be defined to be large enough to fulfil the design requirements of the system but also the motion has to be limited for safety considerations. The main task of the robotic landing gear is to adapt the position of the legs to the geometry of the ground to allow the system to land on uneven terrains

maintaining the attitude of the main body in a level position and ensuring a firm ground contact for all four legs. For this purpose, the motion of each leg is coordinated to move the feet up and down only, while maintaining the horizontal hip-foot distance constant, because any lateral motion by the feet could tip the helicopter.

For a given leg configuration, the horizontal distance between the right and left feet will define the maximum distance that the legs can extend/retract vertically and therefore, the maximum slope that the system can land on. Increasing this lateral distance, increases the resistance of the system against lateral tilting, giving more stability, but reduces the maximum landing slope, and vice versa. Figure 4-14 shows the system with the right leg in the initial landing position. The upper leg link is in horizontal position to maximise the distance between the right and left feet, to give more lateral stability. The lower link is in vertical position to provide a good foot-ground interaction.

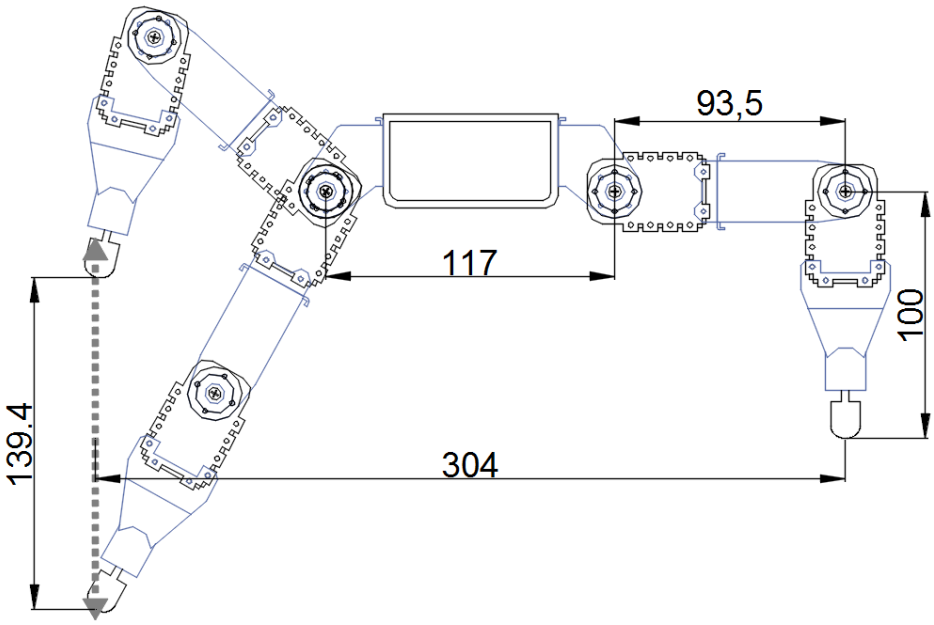

Figure 4-14 Legs in landing position (right) and fully retracted and fully extended positions (left).

The left leg shows the completely extended and completely retracted positions. If the horizontal hip-foot distance is kept constant at 93.5 mm and the total length of the leg is L = 93.5 +100 = 193.5 mm, then maximum vertical hip-foot distance can be calculated applying trigonometry:

$$z_{max} = \sqrt{L^2 - y^2} = \sqrt{193.5^2 - 93.5^2} = 169.4\, mm \tag{4.1}$$

For safety reasons, the minimum vertical hip-foot distance is limited to 30 mm to avoid that the main body could hit the ground. Therefore the difference between a fully retracted leg and a fully extended one gives an available foot stroke of 139.4 mm.

Thus the maximum landing slope will be:

$$\alpha = \tan^{-1}\left(\frac{z}{y}\right) = \tan^{-1}\left(\frac{139.4}{304}\right) = 24.6° \tag{4.2}$$

Figure 4-15 shows the landing gear with the left leg in fully retracted position and the right leg fully extended. Another factor to consider when landing on a big slope is the risk of the rotor hitting the ground on the upslope side.

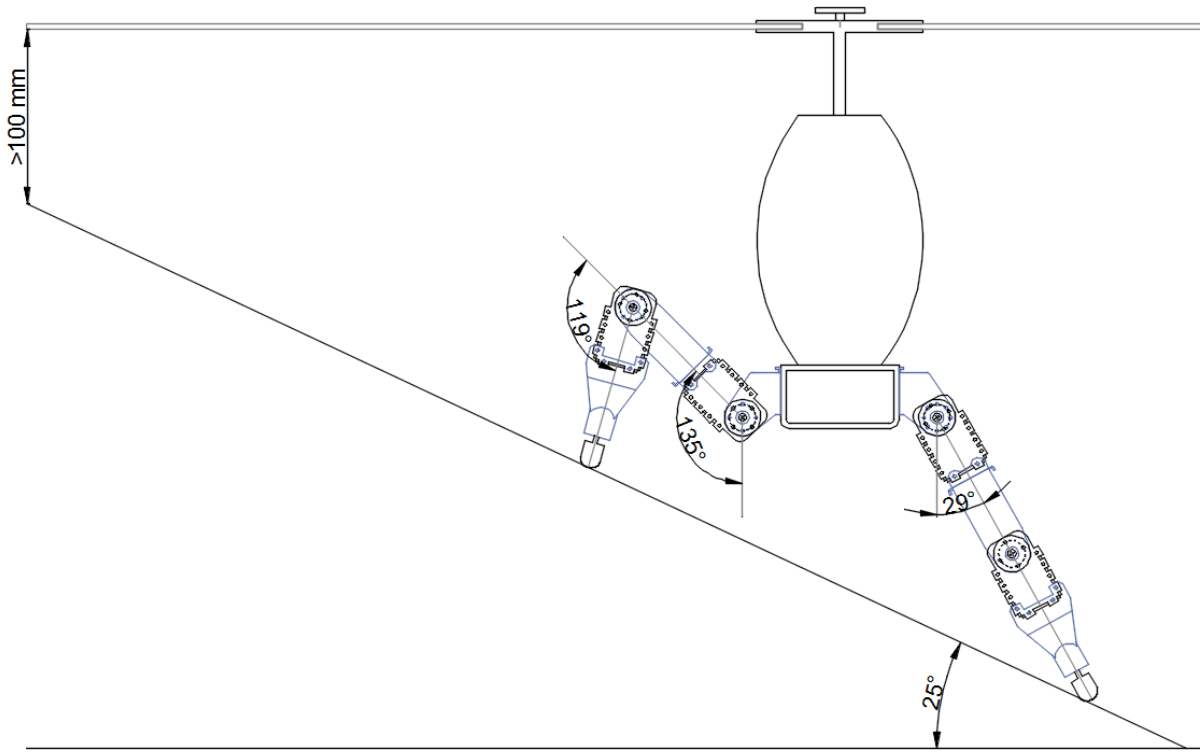

Figure 4-15 Fully extended/retracted leg angles and maximum landing slope.

To avoid damage due to collisions between different moving parts of the robotic landing gear and the helicopter, the movement of the servo motors is restricted to the area between the fully retracted and fully extended leg positions.

Figure 4-16 and Table 4-3 summarise the range of motion of all the servo motors. The motion on the back legs is equal to the front legs.

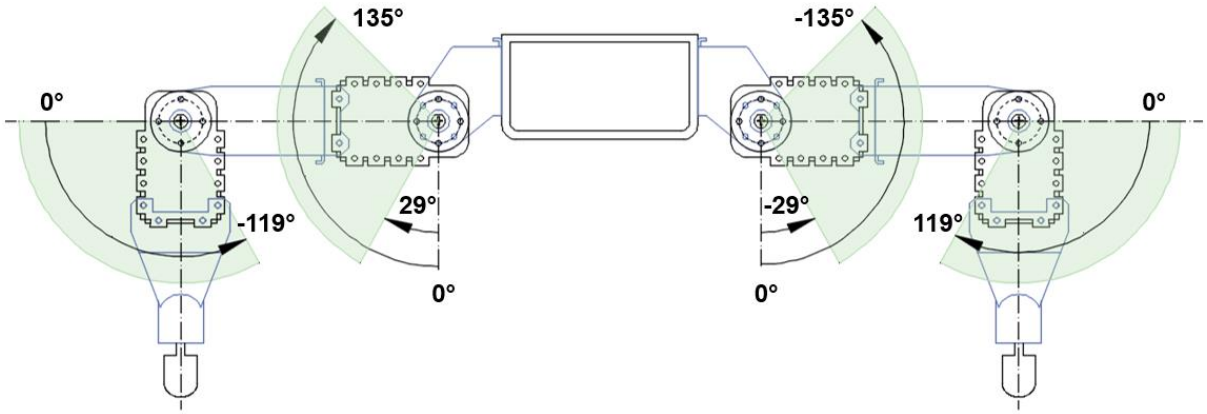

Figure 4-16 Joint servos motion range.

Table 4-3 Joint angle limits in degrees and equivalent 10-bit value for the CW/CCW angle limit registers in the AX-18 control table.

Servo	Lower Limit (°)	Lower Limit (AX-18 register)	Higher Limit (°)	Higher Limit (AX-18 register)
LH	29	611	135	973
LK	-119	106	0	512
RH	-135	51	-29	413
RK	0	512	119	918

4.5 Sensors

4.5.1 Inertial Measurement Unit

The motion sensor MPU-9150 combines a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer in a single package. It contains 16-bits analog to digital conversion hardware for each channel and captures the x, y and z channels at the same time. The sensor uses the I2C-bus to interface with the controller [66].

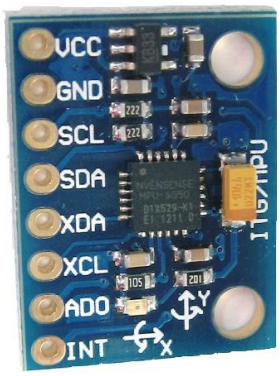

Figure 4-17 MPU 9150 Gyro + Accelerometer + Magnetometer [66]

The IMU is used to obtain the attitude of the main body. According to [67], the roll (θ) and pitch (φ) angles can be obtained from an accelerometer using the following expressions.

$$\theta = atan2(a_y, a_z) \tag{4.3}$$

$$\varphi = atan2(-a_x, a_y^2 + a_z^2) \tag{4.4}$$

assuming that the axis of the IMU unit match the system's axes convention. a_x, a_y and a_z are the readings of the accelerometer in the x, y and z directions.

The gyroscope provides the rate of rotation of the main body and is given by:

$$\dot{\theta}(°/s) = \frac{g}{131} \tag{4.5}$$

where g is the signal from the gyroscope in a given axis. The raw signal is divided by 131, as the accuracy of the gyroscope is of 131 steps per 1 °/s.

The gyroscope signal provides accurate results in the short period but it drifts in the long term producing an error. By contrast, the accelerometer is noisy and imprecise in the short term but provides a stable signal over a long period. For this reason, a filter needs to be used.

Initially, a complementary filter was designed to combine the readings from both sensors, but the final versions of the prototype use a Kalman filter as it provided better results during the tests.

The algorithm of the Kalman filter provides an efficient way to calculate the estimate of the state of a system based upon measurements and predictions based on the statistical noise from the measurement and the process. The Kalman filter is widely used in many research fields and applications. For an introduction and full derivation of the equations of the filter refer to [68].

For this project, the Kalman filter Arduino library described in [69] has been used.

4.5.2 Distance Sensor

The VL53L0X is a *Time of Flight* distance sensor that uses a small laser source and measures how long the light has taken to bounce back to the sensor. It can handle

about 50 - 1200 mm of range distance. The input voltage of the breakout board is 3-5V and communication is done over I2C [70].

The distance sensor is used in the project to detect when the aircraft is approaching to land and to start the landing operation. It is attached at the bottom side of the main body, pointing to the ground, and the distance measured determines if the legs fold/unfold switching between flight and landing modes. A low-pass filter is used to smooth the signal:

$$y_i = \lambda x_i + (1 - \lambda) y_{i-1} \tag{4.6}$$

where y_i is the filter output, x_i is the sensor measurement, y_{i-1} is the previous filter output and λ is the filter gain.

4.5.3 Force Sensor

The force sensors are used to measure the force applied to each foot and to detect when a foot makes ground contact. The sensors used are the Interlink Electronics Force Sensing Resistors (FSR) 400 and they are placed one at each leg. These sensors contain a polymer film that exhibits a decrease in resistance with any increase in the force applied to the surface of the sensor [64]. Sensing range of the sensor is ~0.2N – 20N and have a diameter of 5mm.

To measure the output of the sensor, this is tied to a pull-down 10kΩ resistor to form a voltage divider. The pin between the FSR and the fixed resistor is connected to an analog pin of the controller board. As with the distance sensor, a low-pass filter is used to smooth the signal.

4.5.4 Angular position

As discussed in section 4.3, the actuators provide angular position feedback with a 10-bit resolution, that is, with a range from 0 to 1023. The mapping from the present position register output to the angular position in degrees was shown in Figure 4-11. The following function is used to map the present position register output into the angular position in radians with a range of ± 2.618 rad, being 0 rad the servo centred position.

$$Angle\ (rad) = (Encoder(int) - 512) \cdot \frac{150°}{512\ units} \cdot \frac{\pi}{180°} \tag{4.7}$$

4.6 Control System

4.6.1 On-board Controller

The on-board controller is the OpenCM9.04 A-type, an open-source microcontroller from ROBOTIS, compatible with the Dynamixel 3-pin TTL communication protocol which incorporates 3 serial ports, analog and digital pins, I2C and SPI protocols. The board can be programmed using the Arduino Integrated Development Environment (IDE). cations of the microcontroller.

Table 4-4 summarises the most relevant specifications of the microcontroller.

Table 4-4 OpenCM9.04C Specifications [61].

Item	Description
CPU	STM32F103CB (ARM Cortex-M3)
Operating Voltage	5V~16V
I/O	GPIO x 26
Analog Input	10 (12 bit)
Clock	72MHz
USART	3
SPI	2
I2C	2
TTL Bus 3-pin	4

4.6.2 System Architecture

Figure 4-18 shows the connections map of the system. The IMU and distance sensor are connected to the OpenCM9.04 through the pins D24 and D25 using one of the I2C channels. The FSRs are connected using 4 analog inputs. All sensors are connected to the 5V power supply. The microcontroller can transmit data to a PC by using a USB cable or through a wireless transmitter connected to one of the serial ports. The TTL ports are used to connect the Dynamixel AX-18 motors and to provide power to the board. When connected to the PC, the USB port can power the controller and the sensors but needs additional power for the motors. The board is connected to a power hub using one of the TTL ports and this is connected to a 12V DC power source using an adapter or an on-board Li-Po battery.

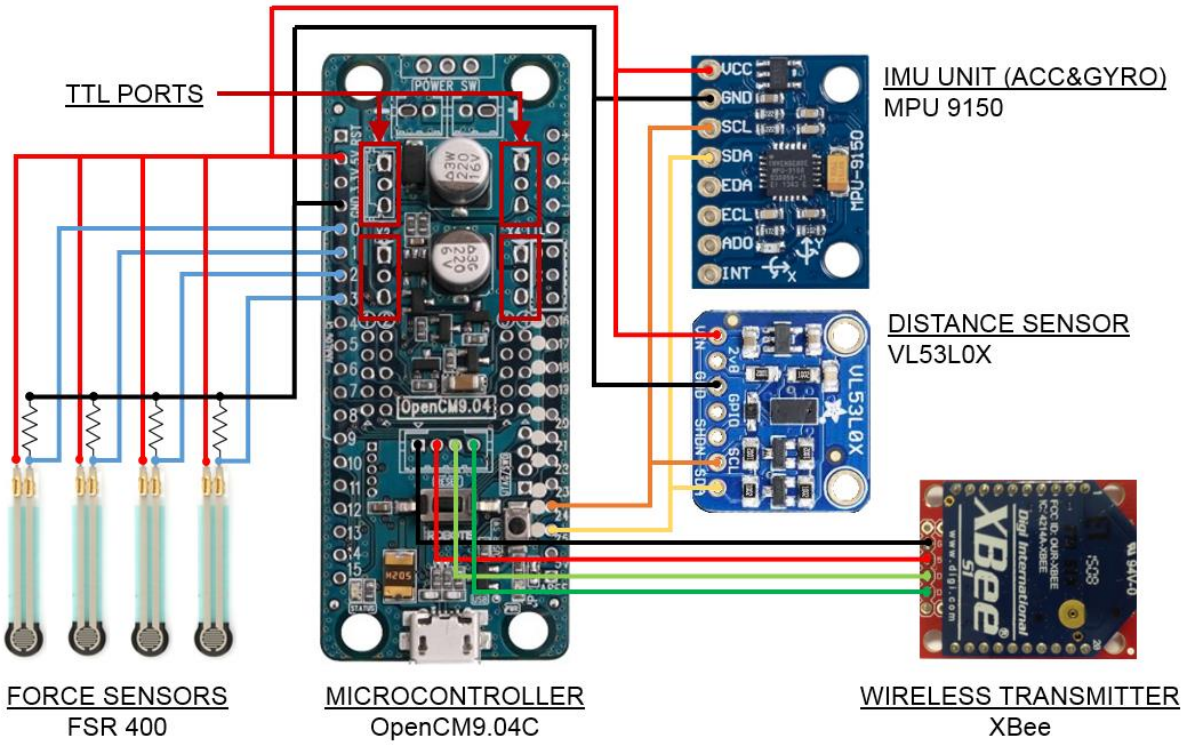

Figure 4-18 Control system Architecture.

With this setup, the code developed to control the system can run at a maximum frequency of 20 Hz, or a 50 milliseconds cycle time.

4.6.3 Software

The Arduino IDE is used to write and upload code to the controller, and to interface with the motors and sensors. One of the main advantages of using Arduino is the existence of a vast collection of libraries that allow users to quickly implement hardware solutions. Libraries are useful because they can perform tedious tasks at instruction packet level compiling and sending physical bytes of data, and the programmer can focus on the high level programming tasks which makes programming easier and more intuitive [71].

Manufacturers usually provide code libraries to interface with their hardware. Libraries used in this project to control sensors and motors include:

- *Dynamixel Workbench*. This is a library developed by Robotis to control any type of Dynamixel servo motors [72].

- *I2C Devace Library (i2cdevlib)*. It's a collection of uniform and well-documented classes to provide simple and intuitive interfaces to I2C devices including the MPU-9150 IMU [73].
- *VL53L0X*. It's a library developed by Adafruit Industries to control the Adafruit VL53L0X time-of-flight sensor [74].
- *Kalman Filter*. This is a library to implement a Kalman filter in most microcontrollers [69].

4.7 Conclusions

This chapter presented the mechanical and electrical design of the robotic landing gear. First an overview and general specifications of the system are described, and then the mechanical structure of the system is described in detail, including the leg and foot design and the functional and safety considerations to define the joints range of motion. Next, the actuators and sensors used are described, followed by the on-board controller, control architecture and control software. Detailed description and specifications of every component are included.

5 Control System

This chapter presents the control system for the robotic landing gear. The goal of the control system is to adapt the position of the legs to the geometry of the ground to allow the system to land on uneven terrains. This includes maintaining the attitude of the main body in a stable horizontal position and ensure that all four legs are in firm contact with the ground. The control hierarchy of the system can be divided into two levels: a High-level controller which uses combined foot pressure and body attitude information to provide position commands to each joint, and a Low-level controller which is in charge of converting these position commands into torque commands for the joint motors. For the high-level controller, a controller is designed using a PD feedback law.

5.1 Control System Overview

Figure 5-1 summarises the overall landing gear control scheme for one leg, where the High-level controller computes the Cartesian trajectory for each foot, The Inverse Kinematics produce the angle trajectories for the hip and knee joints and the Low-Level controller computes the torque commands. In this block diagram, *i* stands for the leg number (*i*=1, 2, 3, 4) and *j* defines if the joint is either the hip or knee. The position vector $\mathbf{r_{hf}}$ represents the hip-foot distance for feet *i*, while q_j and $q_{d\,j}$ represent the current and desired values for each joint position respectively. The main body pitch and roll angles are represented by φ and θ. $\mathbf{f_i}$ represent the vector of contact forces at foot *i*, and τ_j are the joint torques.

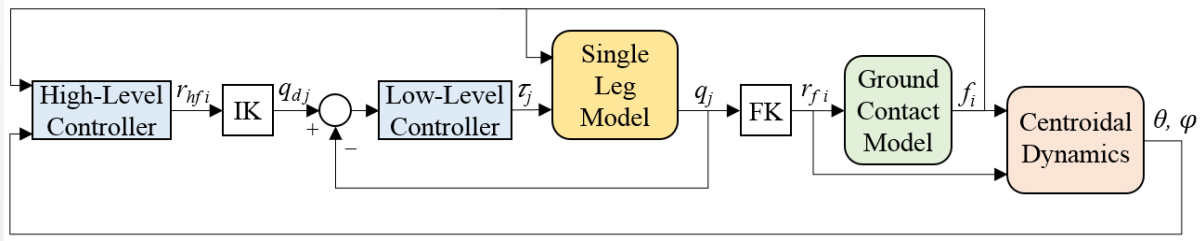

Figure 5-1 Control block-diagram

The blocks Single-Leg model, Leg Forward and Inverse Kinematics (FK/IK), Ground Contact Model and Centroidal Dynamics have already been explained in chapter 3, while this chapter will focus on the high and low level controllers.

5.2 Low-Level controller

The goal of the low level controller is to compute the joint torques needed to move the feet to the desired position. The desired trajectory of the feet is provided by the high level controller and the leg inverse kinematics model.

In section 2.5 most common types of joint controllers were reviewed and were broadly classified between model-based and non-model-based techniques. Developments in hardware, sensor and actuator technology has led to the development of torque controllable robots. For example, in the case of walking robots and legged locomotion research, where several prototypes have been developed using model-based techniques like computed-torque control. Some examples are the HyQ [75], StarlEHT [76] and NUDT [77] robots, which use joint controllers based on inverse dynamics control, or RoboCat-1 [26] that uses friction and gravity compensation.

However, there are several limitations to the implementation of model-based controllers. The first one, is the availability of the torque-control capability of the actuators. Most low-cost servo motors don’t have this feature and can only be used in position-control mode. Another added limitation is the difficulty of estimating precisely model parameters like the body inertias, and the expense of the computer power needed to compute the complicated inverse dynamics control equations. All this reasons make that in applications where a precise trajectory-following is not required, model-based approaches are not used, and classical controllers are preferred as they are easier to implement in hardware robots. An example of a low-cost quadruped robot that uses a position-control system can be found in [29].

In the case of the servo motors used in our physical prototype, the torque control capability is not available, and the motors incorporate their own internal joint controllers which are controlled by means of position setpoints. For the purpose of software simulations, only conventional joint controllers like PD and PID will be used for their simplicity and because of the limited range of motion of the legs doesn’t require precise trajectory following.

5.3 PD High-Level controller

The control system of the robotic landing gear must fulfil the following requirements:

- The system has to keep the legs in a retracted position during flight to reduce air drag, and be able to detect approach to ground and move the legs to landing position.
- The controller has to be activated when the landing operation starts and has to keep the helicopter body in a level position during the whole operation.
- To conclude the landing operation, the helicopter body has to be level and all legs have to be in contact with the ground.
- Once the landing operation is concluded, the legs are locked in its current position at that moment.

For the first requirement, the system uses a time-of-flight sensor to detect the distance to the ground. For the second and third requirements, foot force sensors (FSR) and an IMU are used. Encoders in the servo motors are used to obtain the servo motors positions.

The design of the controller has to meet these requirements with the available sensing, with the additional constraints that it cannot depend on the torque control capability of the servo motors and it should not be computationally heavy in order to be implemented in our prototype and send position commands to the joint controllers.

The proposed solution here is a whole-body posture controller with the body velocities as controller outputs. Just as an example, Figure 5-2 shows a 2D representation of the landing gear system landed in a tilted position. In order to produce a rolling moment that moves the main body to a level position, a pair of hip velocities can be generated by producing the opposite velocity at the respective foot relative to the ground. This velocity vector will be parallel to the z-axis of the body-fixed frame, its direction will depend on the direction of the body rotation, and its magnitude will be a function of the body attitude and feet pressure feedback. Hence, the foot velocity in the y-axis will be zero.

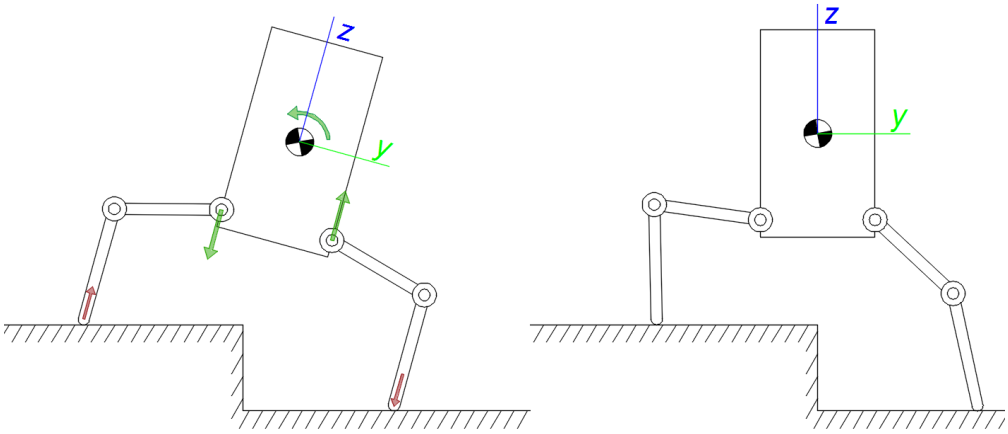

Figure 5-2 Posture control system. If the body is tilted to the right side, a positive velocity in the right hip in the direction of the z-axis, and negative in the left hip will produce a posture correction motion. This is done by pushing each foot against the ground with the opposite of its hip velocity.

The feet pressure and body attitude controllers will be two separate modules of the high-level controller that work independently and even in the absence of the other. The controller is activated whenever any leg touches the ground during landing.

The force controller computes the component of the foot velocity due to the foot pressure feedback, $\dot{z}_{hf_F}$. When the controller is activated, if a leg is in contact with the ground ($f_z > 0$), it will retract following a PD control law. Otherwise, if the leg is not in contact, it will extend at a fixed rate. Once all four legs are in contact with the ground, the force control is switched off.

$$\dot{z}_{hf_F} \begin{cases} e_{F_i} \cdot k_{P_F} + \dot{e}_{F_i} \cdot k_{D_F} & if\ f_{z_i} > 0 \\ -k_{ext} & if\ f_{z_i} \leq 0 \end{cases} \tag{5.1}$$

where the force error is the difference between the desired and measured force in the z direction $e_F = f_{z_d} - f_{z_i}$, and k_{P_F} and k_{D_F} are the proportional and derivative gains. k_{ext} is a fixed rate of extension.

Figure 5-3 shows a graphical representation of the force controller.

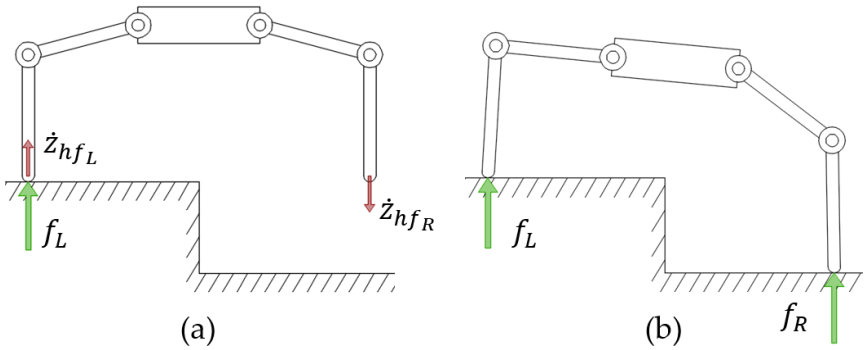

Figure 5-3 Force controller. In (a), the left leg retracts because the left foot is in ground contact, while the right extends until it touches the ground (b). At this moment, the force controller is switched off.

According to equation 5.1, if a leg in ground contact retracts too fast, it can lose contact momentarily and would start to extend until it makes ground contact again. When implementing the force controller, this situation where a leg switches continuously between extension and retraction has to be avoided as it would lead to abrupt changes in the ground reaction force introducing disturbances into the helicopter body orientation. For this reason, once a leg touches the ground for first time, this information is stored, and this leg is only allowed to retract if the controller output is positive, or to stop moving if it loses contact and the controller output becomes negative, but not to extend. If the landing operation is interrupted and the landing gear goes above a certain height, the state of each leg is restarted.

The attitude controller is activated at the same time that the force controller and coordinates the motion of all four legs as a function of the roll and pitch angles of the main body. The component of the foot velocity due to the roll, $\dot{z}_{hf_\theta}$, and pitch, $\dot{z}_{hf_\varphi}$, it's also computed using a PD control law.

$$\begin{bmatrix} \dot{z}_{hf_\theta} \\ \dot{z}_{hf_\varphi} \end{bmatrix} = \begin{bmatrix} k_{P_\theta} e_\theta + k_{D_\theta} \dot{e}_\theta \\ k_{P_\varphi} e_\varphi + k_{D_\varphi} \dot{e}_\varphi \end{bmatrix} \tag{5.2}$$

where e, k_P and k_D are the angle error in degrees, and proportional and derivative gains respectively for the roll and pitch controller.

The output of the roll controller is added to the legs at one side of the x-axis of the main body and subtracted to the others depending on the sign of the roll angle to generate a rolling motion around the x-axis (Figure 5-4(a)). In a similar way, the output of the pitch controller is added to the legs on one side of the y-axis and subtracted to the others (Figure 5-4(b)).

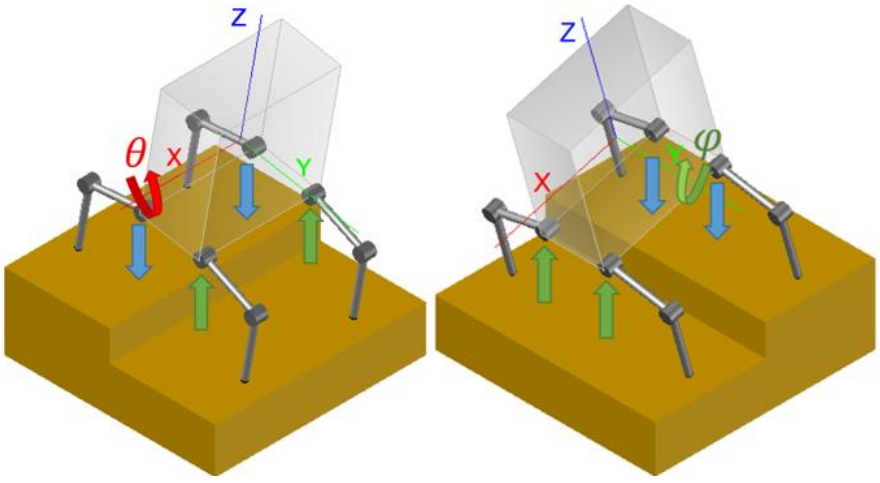

Figure 5-4 Attitude controller around the x-axis (a) and y-axis (b)

The total rate of extension/retraction of each leg is calculated as the sum of the three controllers and then integrated to obtain the total adjustment of each leg.

$$\Delta z_{hf} = \int (\dot{z}_{hf_F} \pm \dot{z}_{hf_\theta} \pm \dot{z}_{hf_\varphi}) \tag{5.3}$$

The desired hip-foot Cartesian coordinates are given by the equation 3.50, then the joint angles for the 8 motors are obtained using the inverse kinematics equations and the position commands are sent to the joint controllers as explained in section 3.4. Kinematic constraints are introduced to limit the maximum leg extension and retraction.

Overall, the landing process has two stages. A first one starting at the moment that a ground contact is detected, where force and attitude controllers are activated, and a second stage, starting when all legs have already made ground contact. At this moment, the force controller is switched off and the attitude controller alone corrects the remaining tilt of the helicopter body. The implementation of the control system in the hardware prototype is presented in chapter 6. Equation 5.8 represents the controller function in stage 1. In stage 2, the function will be the same but without the force controller term:

$$\Delta z_{hf} = \int (\pm \dot{z}_{hf_\theta} \pm \dot{z}_{hf_\varphi}) \tag{5.4}$$

The force controller guarantees that all four legs stay in contact with the ground while the attitude controller forces the roll and pitch angles to converge to zero. The combination of both reduces the time from the moment that the first leg touches the ground until the landing finishes with all four legs landed and the main body in a level position improving stability of the system during the landing process and the overall performance of the controller.

5.4 Software Simulations with the 2-legged model

The Dynamic model and the control system of the landing gear are implemented in the software environment Matlab/Simulink for testing and for controller parameters tuning. The purpose of the simulations is twofold. In one hand it is used to validate the dynamic models and assess if they behave in a coherent way. On the other hand, they serve as a measure of the performance of the control system. As described in chapter 3, two different models were developed: a two-legged and a four-legged version. A

thrust model was also developed to control the descending speed during landing, and different types of terrain to simulate different ground reaction forces during landing.

In this Section, the 2-legged model is tested in two simulations on flat terrain and in a terrain with a step. This model is a planar model and therefore it can only move in the planar space (two directions in the YZ plane and one rotation about the X axis).

5.4.1 Landing simulation on flat terrain

In the first simulation, the two-legged model is used in a landing simulation on even terrain. In this experiment the high-level controller is not activated and the system is tested using different sets of parameters for the low-level controllers. The physical parameters, summarised in Table 5-1, like the mass of the system or the length of each link are set to match as close as possible those of the system prototype described in chapter 4, so the results of the simulations can be comparable with the results of the laboratory test.

Spring and damper coefficients for the normal ground force model are 1500 kg/s^2 and 40 kg/s respectively.

Table 5-1 Principal simulation parameters.

Parameter	Value
Upper leg mass	0.1 kg
Lower leg mass	0.15 kg
Upper leg length	0.0935 m
Lower leg length	0.1045 m
Total mass (Landing gear + Helicopter)	3 kg
Motor max torque	18 kfg.cm / 1.76 Nm
Main body dimensions	0.1x0.2 m

During a typical landing manoeuvre, as the helicopter approaches the surface the skids should be level, with no forward movement and with the descent rate approaching to zero [1]. The landing descent rate in this simulation is set at -0.1 m/s. The low-level PID controllers are tuned manually by trial and error to obtain the desired performance. Different sets of parameters are used to see the correlation between the PID constants in the joint controllers and the behaviour of the system. For this test a more compliant PID (PID1) with lower gains is compared with one with higher gains (PID2) as shown in Table 5-2.

Table 5-2 Different controller's settings

Controller Gain	PID1	PID2
K_P	5	50
K_I	7	30
K_D	0.4	4.5

Figure 5-5 shows how the helicopter starts the descent from 0.5 m (bottom left) and the thrust controller stabilises the descent rate at -0.1 m/s (top). Touchdown occurs at 3.1 s, with the position of the CoM at 0.2 m above the ground. The roll angle of the main body and the lateral displacement of the CoM are zero (bottom right side). The thrust force (second row) simulates the pilot reducing the lift force gradually after touchdown to provide a smooth settling down of the aircraft. The velocity graph shows how the system with the compliant joint controllers (PID1) settles down in a smoother way while the one with higher values (PID2) leads to more oscillations.

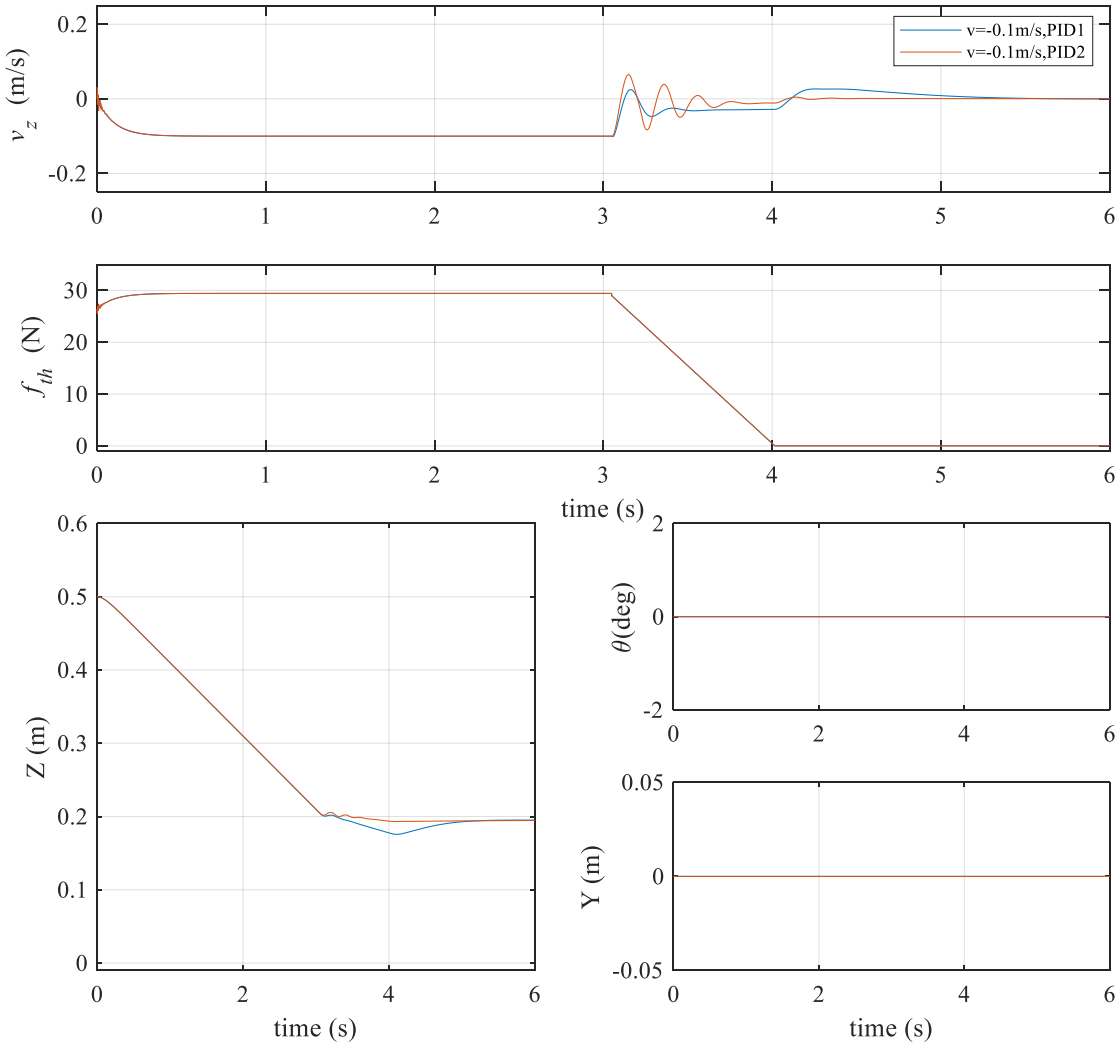

Figure 5-5 Landing velocity (top), thrust force (second row), z-CoM position (left side), main body roll angle (top right side) and y-CoM position (bottom right side).

The ground reaction forces and joint torques produced during landing are shown in Figure 5-6. The ground reaction forces (top row), increase gradually after touchdown as the weight of the helicopter is transferred to the ground, and the same occurs with the hip torques (middle row) and knee torques (bottom row). It can be seen that with the PID1 settings this load transfer is smoother while the PID2 settings produce more fluctuations, even leading to the feet losing ground contact momentarily after touchdown (ground reaction force going back to zero). Joint torques are at every moment below the maximum motor torques (1.76 Nm) and with a large safety margin.

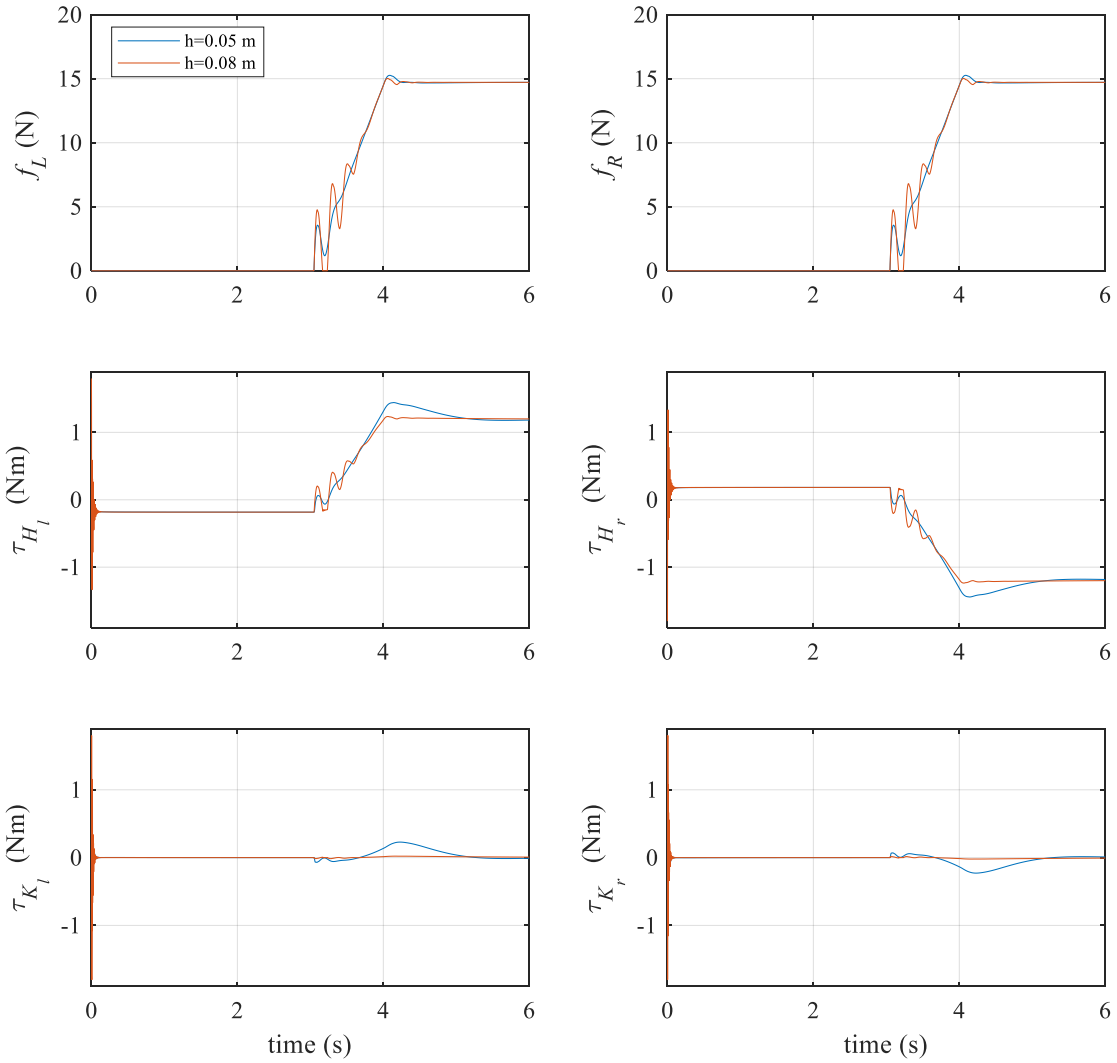

Figure 5-6 Ground reaction forces (top row) on the left/right foot, hip torques (middle row) and knee torques (bottom row) on the left/right leg during landing.

Finally, Figure 5-7 shows the joint angle deflections during landing. The previous figures show how the PID1 settings absorb the impact energy with the deflection of all four joints, while with the PID2 settings the joints are stiffer, transferring more of the

impact force to the helicopter body in the form of fluctuations in the settling velocity graph (Figure 5-5).

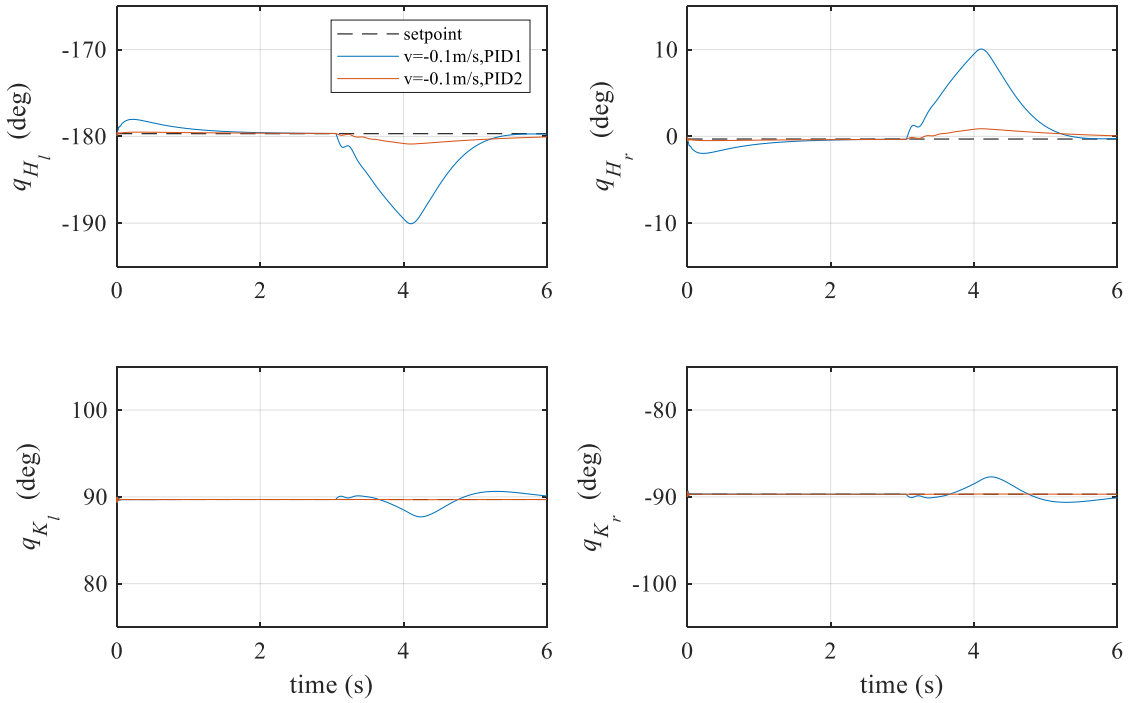

Figure 5-7 Joint angle deflection during landing in all four joints: left hip, right hip, left knee and right knee.

The transient oscillations in the impact forces and joint torques increase when the simulations are done with a higher landing velocity and the impact forces are better absorbed with the compliant joints. Figure 5-8 shows a graphical representation of the joints deflection during the landing process.

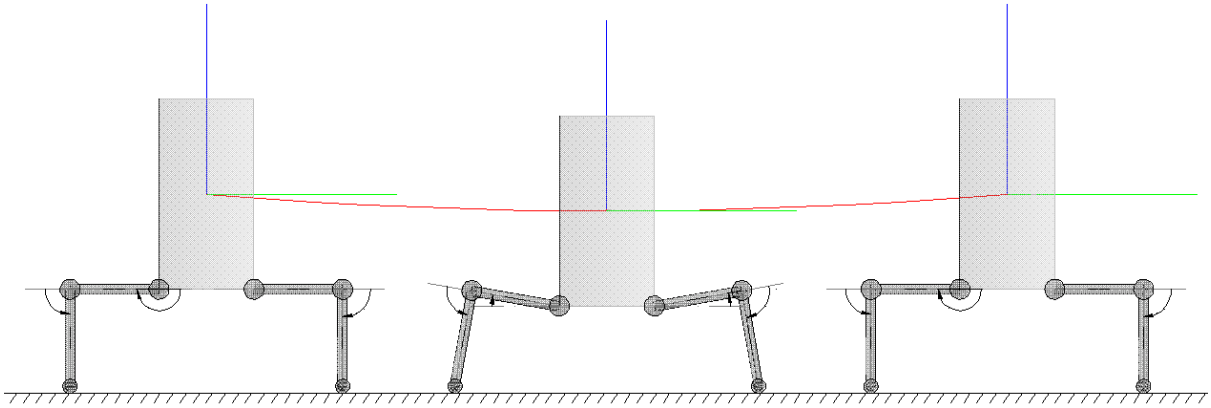

Figure 5-8 Snapshots of system configuration at the moment of touchdown (left), maximum deflection (middle) and final position (right) with compliant joint controllers. Red line shows CoM height at each moment.

As shown in this section, the compliant PID will respond to the disturbance created by the impact force by deflecting the joints angles from its target position, while the rigid

PID will allow less joint deflection. As a consequence, the compliant PID will absorb better this impact force, transmitting less forces and moments to the main body.

The task of the controller is not to achieve an accurate trajectory tracking for each individual joint, but rather to maintain the body attitude at a levelled position. For this purpose, the compliant behaviour is preferable as the system settles down with fewer oscillations.

5.4.2 Landing simulation on uneven terrain

In this simulation, the two-legged model is used to land on uneven terrain. For this test, a step of variable height is introduced in the ground similar to that in Figure 5-2. The high-level controller is activated to try to keep the helicopter body in a level position during the landing operation. Because of the planar nature of the model, the main body can only rotate in the roll direction, thus the level controller acts only in the roll angle while the pitch angle is constrained to be zero.

The controllers' parameters for the High-level attitude and force controllers are tuned manually starting with small values and increasing them until the desired response is achieved producing smooth leg motion without oscillations on the helicopter body. The controllers' parameters are indicated in the Table 5-3 below:

Table 5-3 High and Low level controller's settings.

Controller Gain	Joint Controllers	High-Level Attitude Control (Roll)	High-Level Force Control
K_P	5	0.05	0.15
K_I	7	--	--
K_D	0.4	0.0001	0.001

The step height is set at 8, 11 and 14 cm, which would be the equivalent of a slope of 15°, 20° and 25° respectively. The rest of the parameters are the same as in the previous test.

Figure 5-9 shows a similar behaviour to the flat landing test in terms of descending rate and thrust force. Due to the step, the right foot will make ground contact first. This will make the robot to tend to tilt sideways and the force and attitude controllers to act trying to correct the position of the main body. The peak inclination of the helicopter

body is maintained within 1.5-2° for all scenarios and there's a final lateral *CoM* displacement after landing of 2 cm in the worst case.

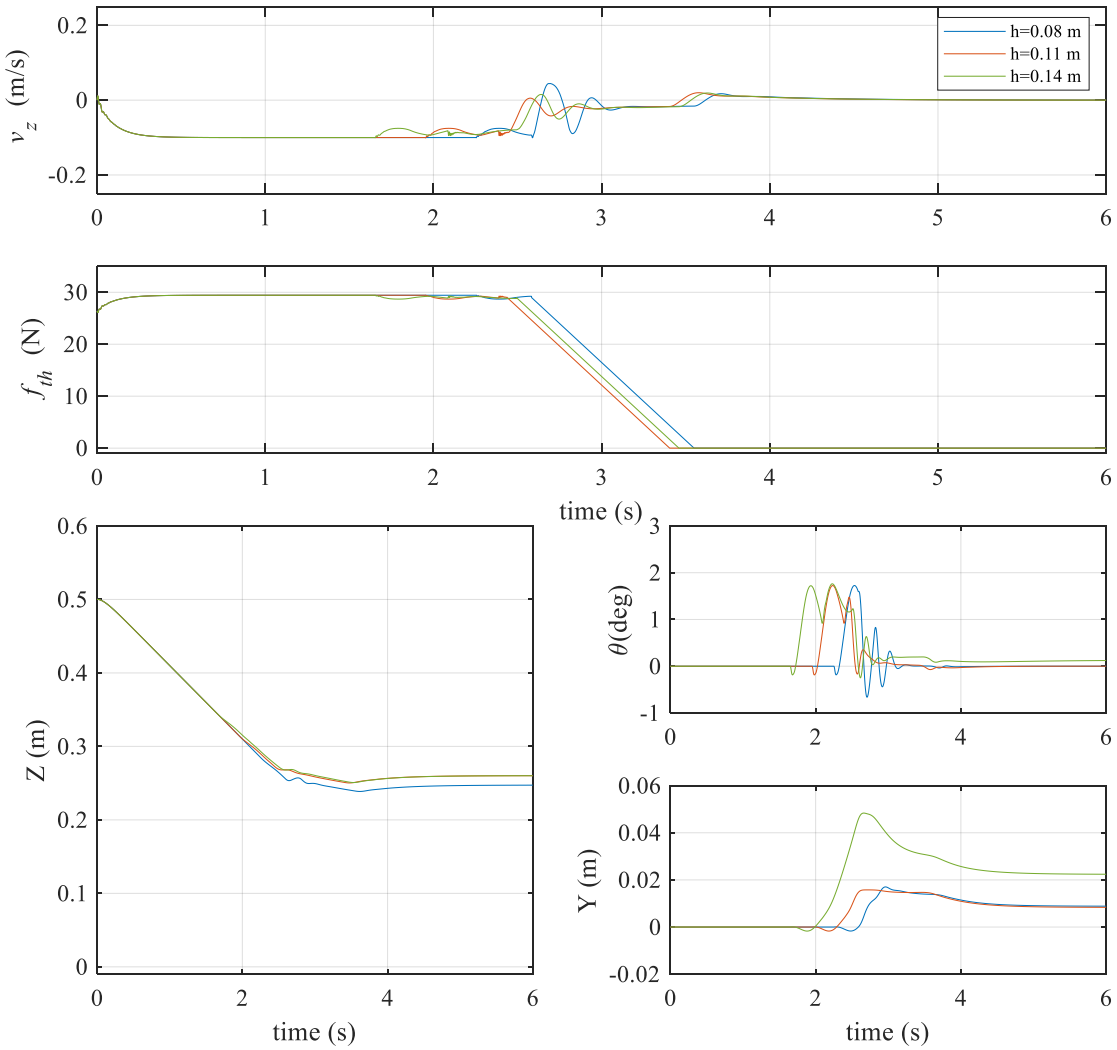

Figure 5-9 Landing velocity (top), thrust force (second row), z-CoM position (bottom left side), main body roll angle (top right side) and y-CoM position (bottom right side).

Ground reaction forces (Figure 5-10, top row) show how the right foot makes ground contact first with a small impact force as this leg starts retracting. The left side makes ground contact later with a higher impact force as this leg has already started to extend and the foot velocity at touchdown is higher. The joint torques (Figure 5-10, middle and bottom rows) present small fluctuations at the moments where the legs start/stop moving and when ground contact is made. After touchdown the weight of the helicopter is transferred to the joints gradually and the loading is distributed equally in both legs.

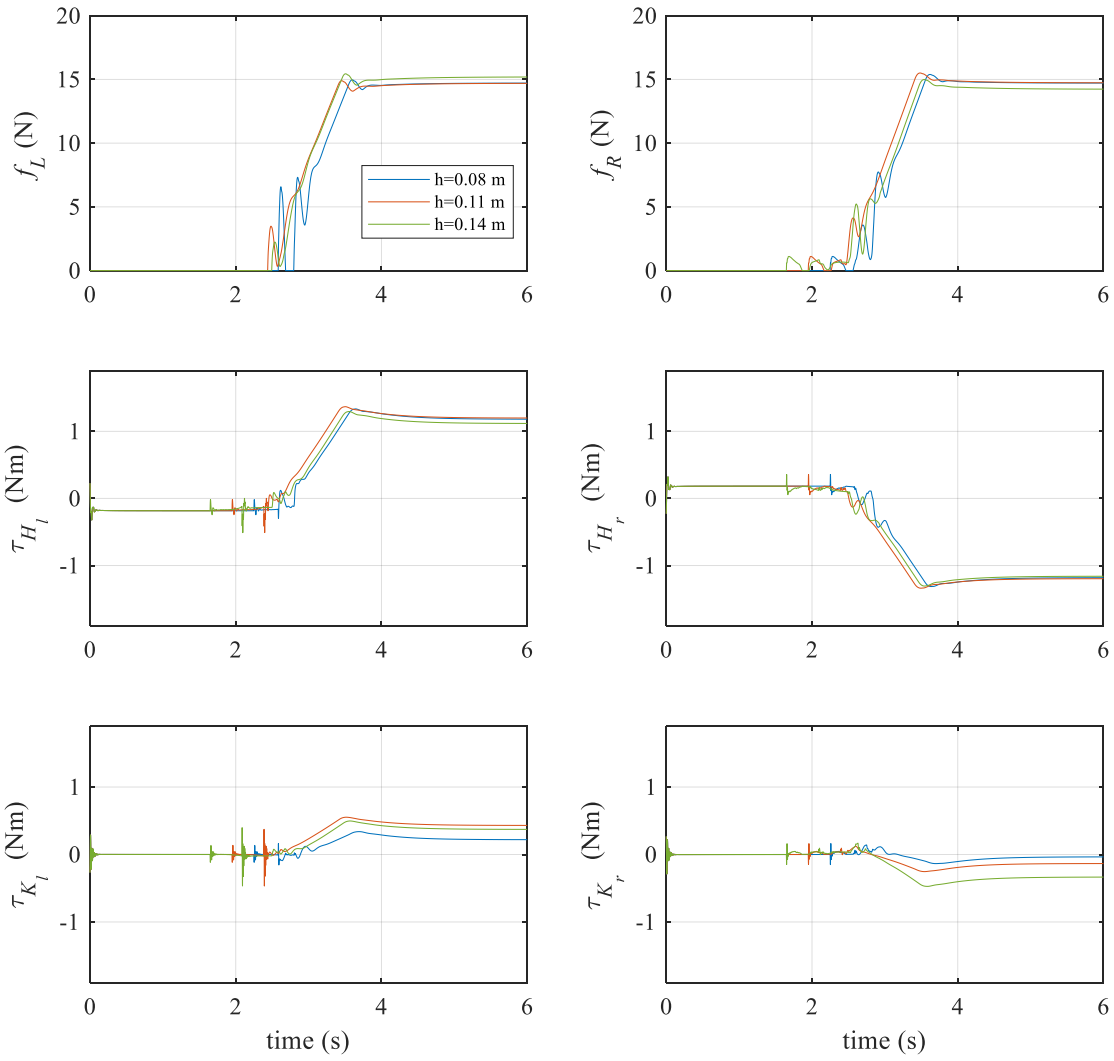

Figure 5-10 Ground reaction forces (top row) on the left/right foot, hip torques (middle row) and knee torques (bottom row) on the left/right leg during landing.

Figure 5-11 shows the legs' motion during landing. Both legs start in the same position but after the first leg touches the ground, the right leg starts to compress and the left leg extends until it touches the ground. In the case of the 11 and 14 cm step, the left leg reaches the maximum extension before it touches the ground. From this point, it doesn't extend anymore, and the rest of the step is overcome with the retraction of the right leg alone.

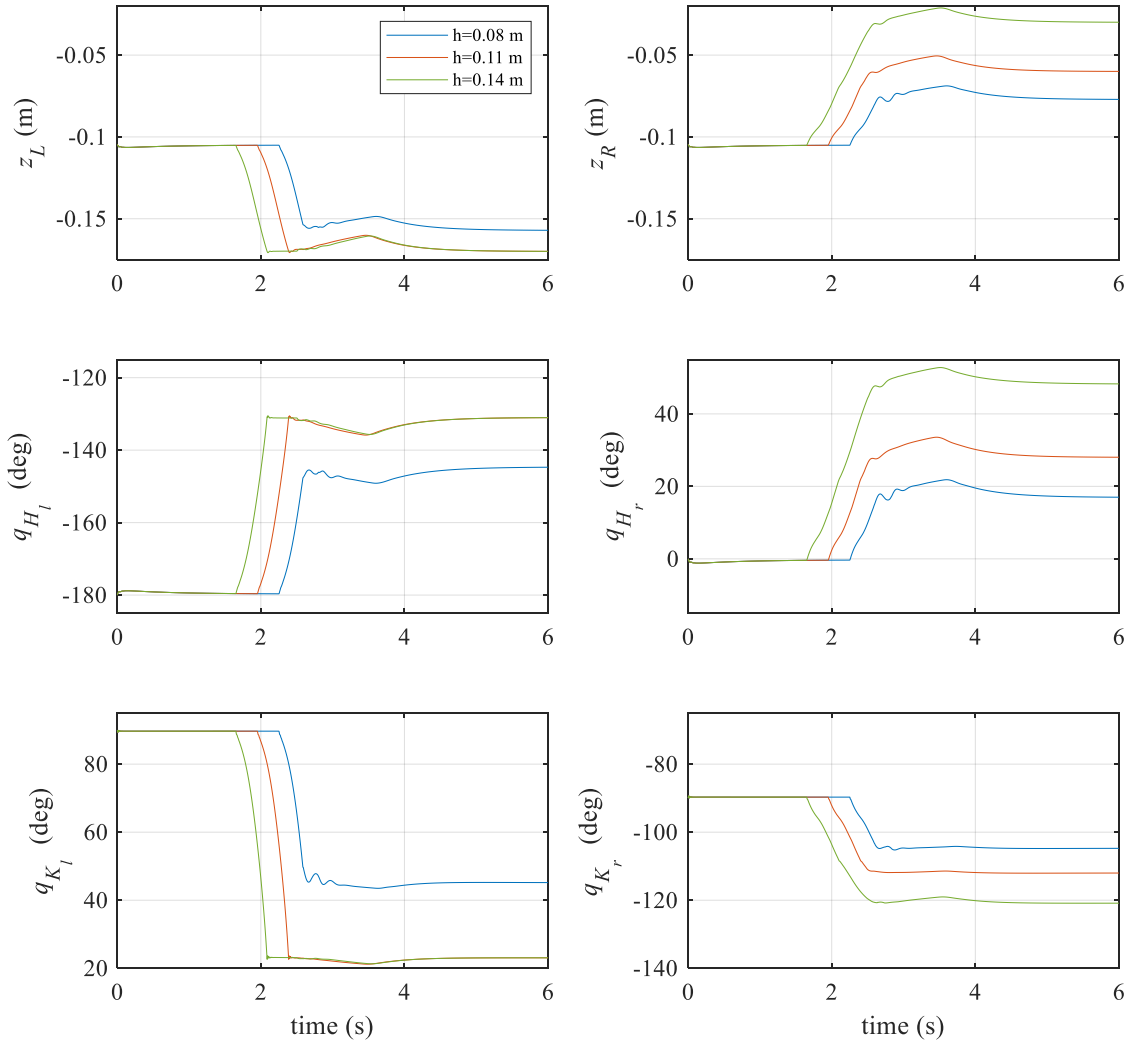

Figure 5-11 Vertical hip-foot distance at the left and right legs (top row). Joint angle deflection during landing in all four joints: Left Hip, Right Hip, Left Knee and Right Knee (middle and bottom rows).

5.5 Software Simulations with 4-legged model

In this Section, the simulations are done with the 4-legged model in two landing scenarios, a slopped terrain and an irregular terrain with surfaces at different levels. In contrast with the 2-legged model, the 4-legged one is a spatial model so the position of its CoM can move in three Cartesian directions (X-Y-Z) and can rotate in the 3-Euler angle direction (Roll-Pitch-Yaw).

When performing slope landings the pilot usually orients the helicopter across the slope rather than with the slope, as facing the helicopter uphill or downhill could result in the tail or the rotor hitting the ground [1]. Figure 5-12 (left) shows the first landing scenario where the helicopter lands on a 20° slope and has an offset of 10° with the direction of the slope. Figure 5-12 (right) shows a landing scenario with an irregular terrain with surfaces at different levels with a maximum step height of 0.12 m.

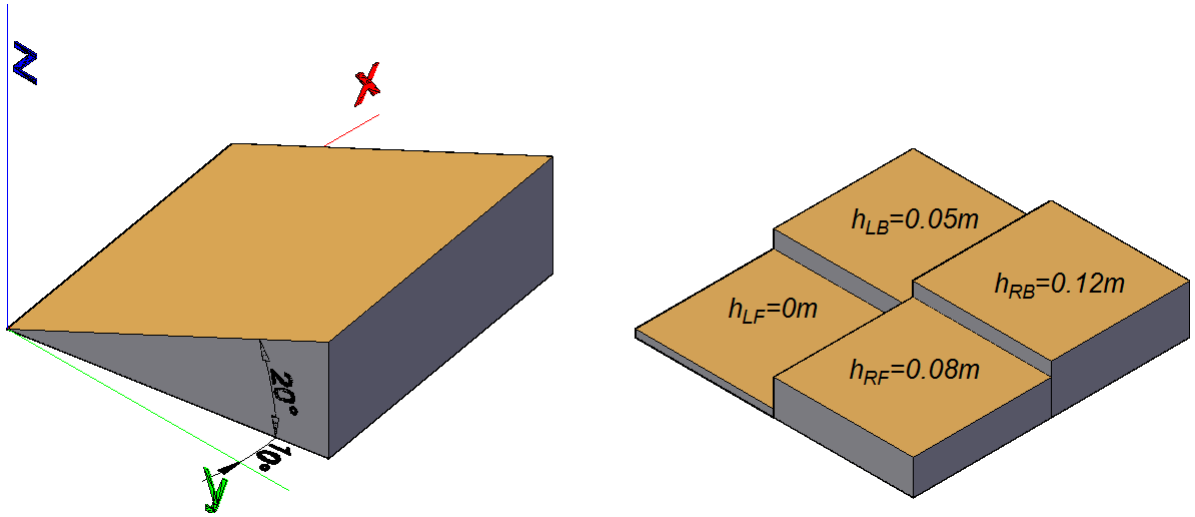

Figure 5-12 Ground model for slope landing simulation (left) and multi-level surface (right)

5.5.1 Landing simulation on a slopped surface

In this simulation the helicopter lands on the ground surface showed in Figure 5-12 (left). Because of the misalignment across the slope, the helicopter body will naturally tilt in the roll and pitch directions and the level controller will tend to level the body and keep both angles at zero.

The legs and actuators properties are the same that in the previous simulations. The main body dimensions are 0.2x0.1x0.2m and the total mass of the system is 3.5 kg. Because the mass supported by each leg is smaller than in the 2-legged case, the ground force coefficients will also be smaller. The spring and damper normal ground force coefficients are set to 900 kg/s^2 and 25 kg/s respectively.

The controllers' parameters for the joint controllers and high-level attitude and force controllers are indicated in the Table 5-4 below:

Table 5-4 High and Low level controller's settings.

Controller Gain	Joint Controllers	High-Level Attitude Control (Roll)	High-Level Attitude Control (Pitch)	High-Level Force Control
K_P	10	0.03	0.03	0.15
K_I	14	--	--	--
K_D	0.4	0.006	0.006	0.002

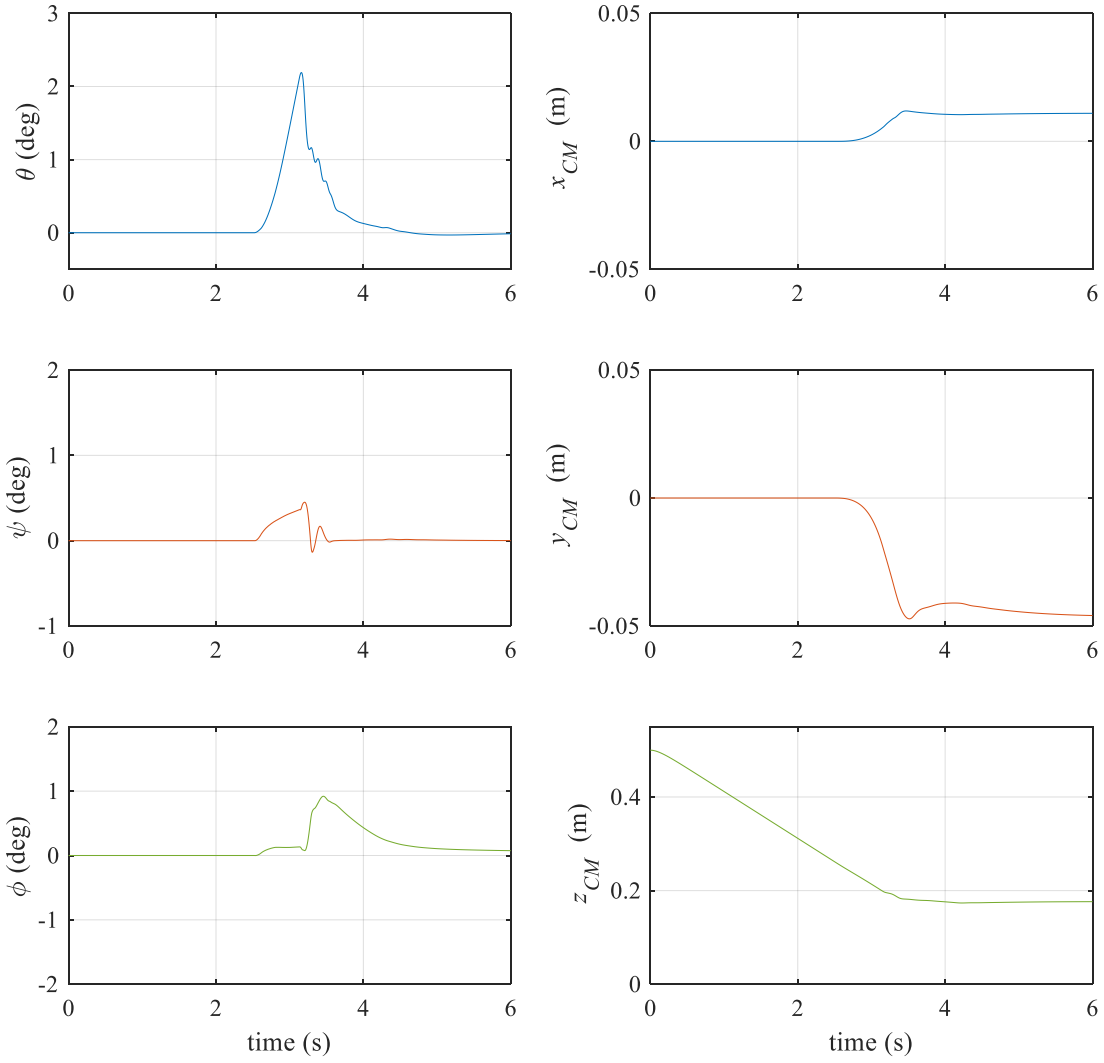

Figure 5-13 CoM angular and linear displacement for slope landing with robotic landing gear.

Figure 5-13 shows the linear and angular displacement of the main body CoM during the landing operation. The z-coordinate shows how the system descends from a starting point of 0.5m at -0.1m/s and finishes with its CoM around 0.2m above the ground. The system tilts 2° around the roll axis at the moment of touchdown while the pitch angle is kept below 1°. Both angles recover the horizontal position quickly. The system turns around 1° in the yaw direction during the landing operation. There is a small lateral displacement of 5cm and 1cm in the X and Y directions due to the compliance of the joints and because the friction force model allows for a small amount of sliding downslope.

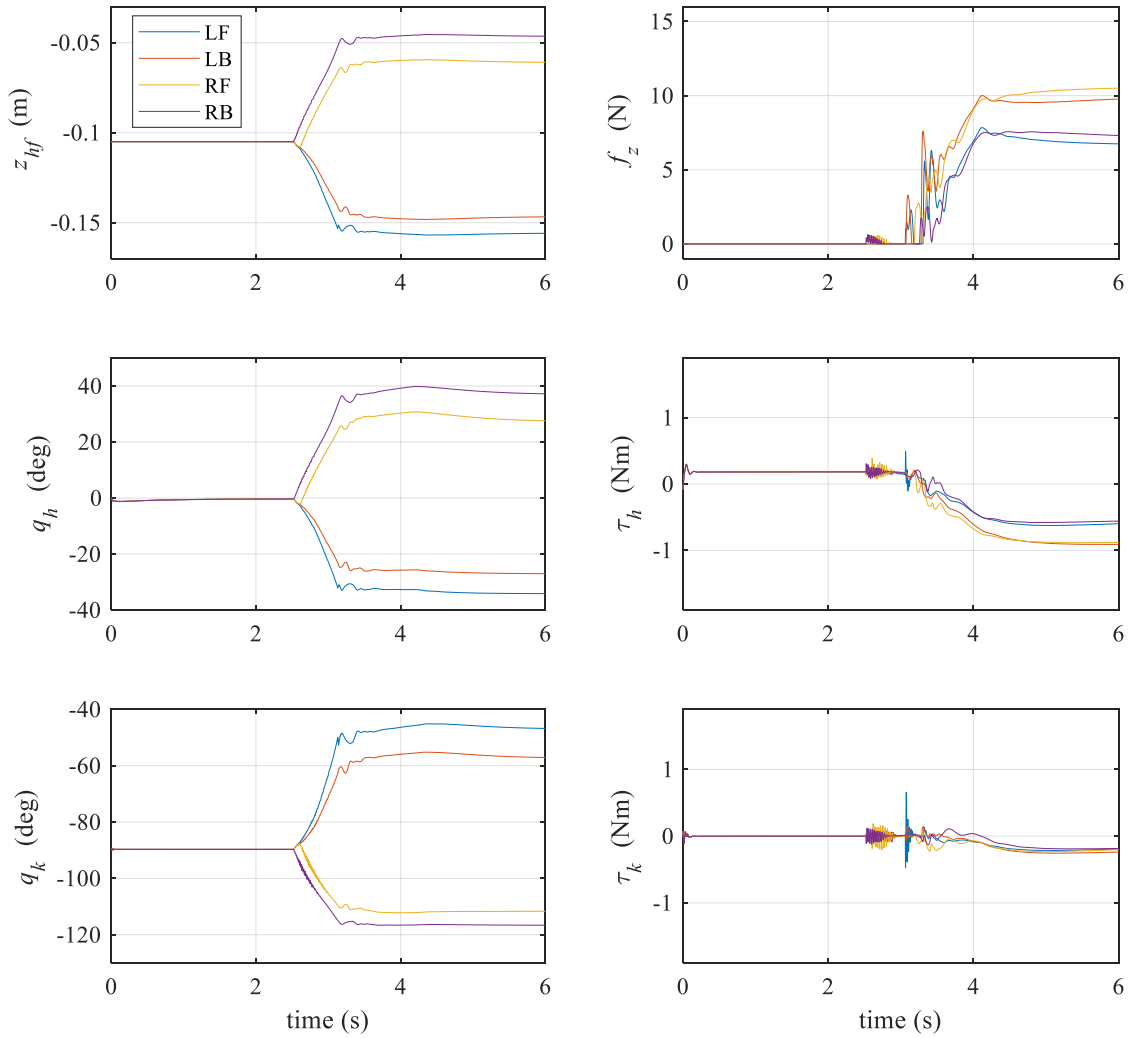

Figure 5-14 Left side from top to bottom: legs hip-foot z-distance, hip angles and knee angles. Right side from top to bottom: normal ground reaction forces, hip torques and knee torques. The nomenclature for the legs is Left Front (LF), Left Back (LB), Right Front (RF) and Right Back (RB).

Figure 5-14 shows the adjustment of the hip-foot distance of each leg and the motion of the joint angles at the hips and knees (left side, top to bottom). On the right side, it shows the ground reaction forces, hip torques and knee torques. All legs start in the same position, but when the right back (RB) touches the ground first, this one starts to retract while all the others extend. Right front leg (RF) touches shortly after, stops extending and begins retracting. Both left legs touch ground a bit later almost at the same time and at this point the force controller is switched off and the attitude controller corrects the remaining tilt. After a short transient all joints reach its final position. The forces and torques plots show small perturbations when the legs start/stop moving and when they touch the ground, and also how the weight of the helicopter is transmitted progressively to the joints as the rotor stops creating lift. It can also be

seen how the diagonal legs RF-LB support more weight than the other two, but in all cases the weight supported by each leg is smaller than with the 2-legged landing gear.

5.5.2 Landing simulation on irregular terrain at different levels

In this simulation the helicopter lands on the multi-level surface showed in Figure 5-2 (right) where the steps between the surfaces below each leg will make the helicopter body to tilt in both the pitch and roll directions. In this simulation, the system is tested at a higher descending rate (-0.5m/s) to see how this would affect to the performance of the controllers.

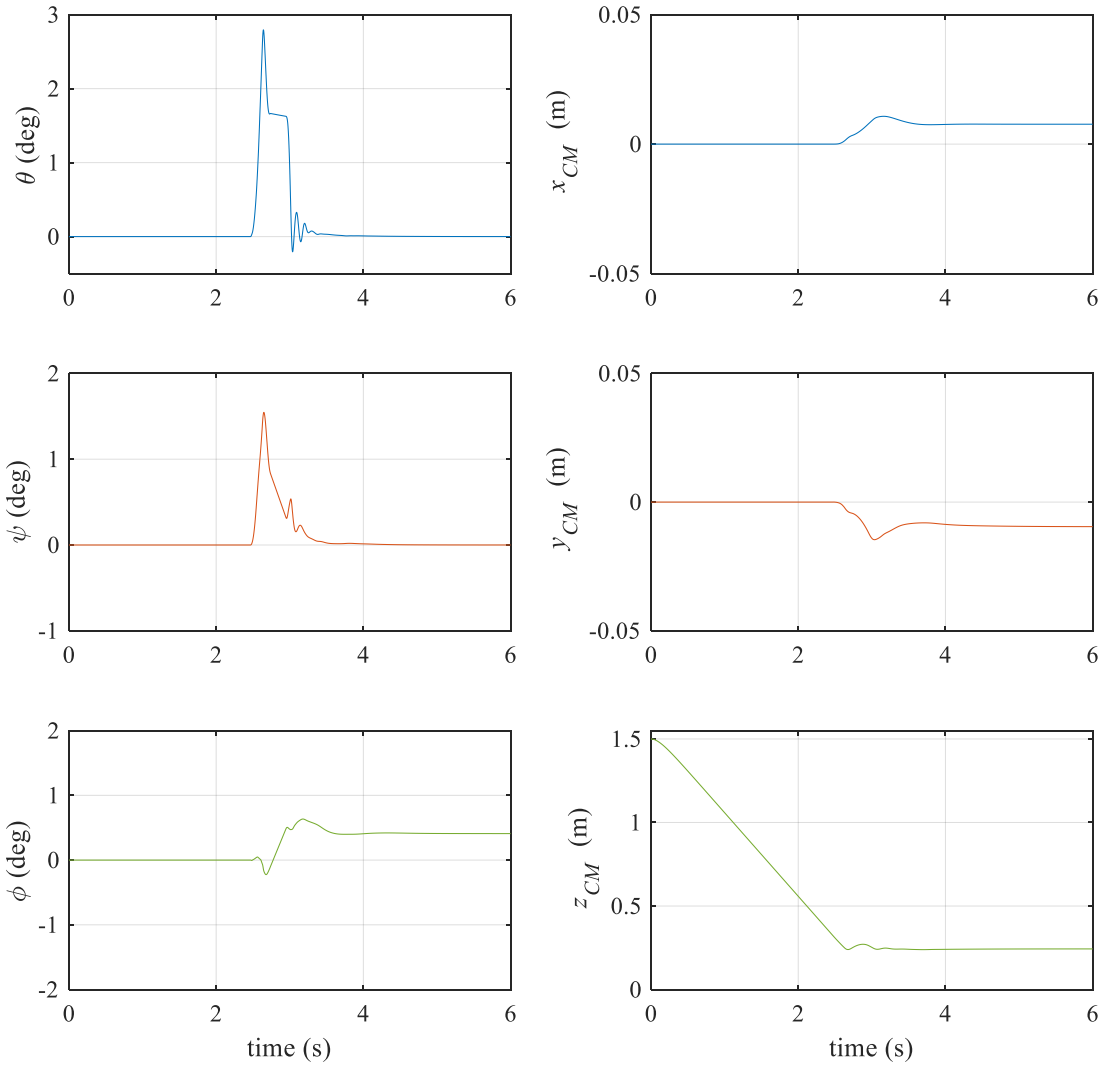

Figure 5-15 CoM angular and linear displacement.

Figure 5-15 shows how maximum inclination of the helicopter body is kept at similar levels than in previous simulations but the transient response is shorter as the time between the touchdown of the first and last leg decreases.

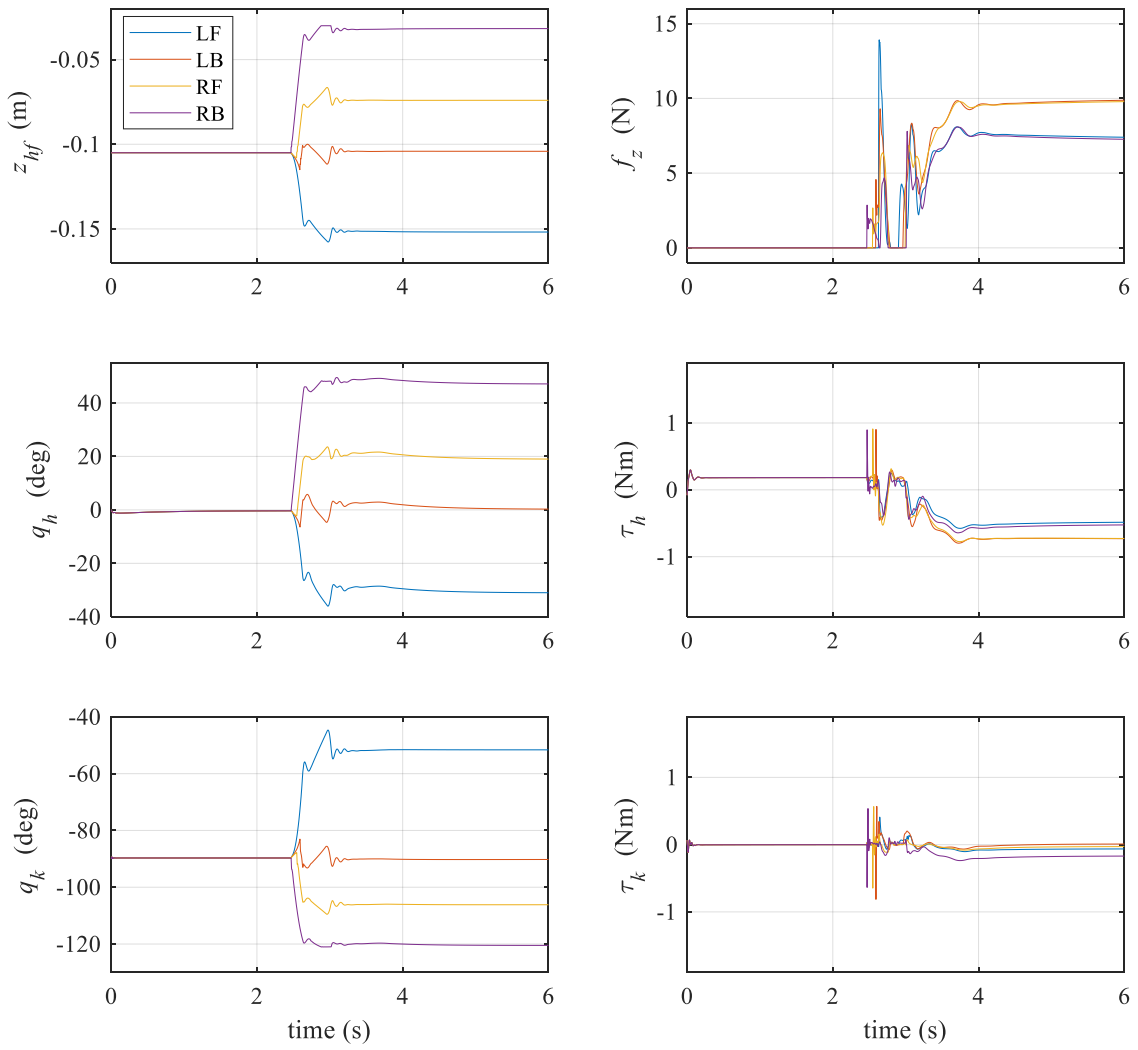

Figure 5-16 Left side from top to bottom: legs hip-foot z-distance, hip angles and knee angles. Right side from top to bottom: normal ground reaction forces, hip torques and knee torques.

A similar pattern can be observed in the forces and torques (Figure 5-16) where the transient response is shorter, although the first impact force peaks are higher as the descending velocity is higher.

5.6 SMC landing gear control

In addition to the control scheme presented in the previous sections, an alternative controller using Sliding Mode Control technique has been proposed to compare the system performance. In this section the High-level attitude controller is replaced with a number of sliding mode control (SMC) algorithms. Simulations are done in similar environments and terrains as in the previous scenarios.

The implementation of the SMC in our system is done by replacing the PD attitude controller by one of the algorithms presented in Section 2.6. Two algorithms are tested,

one with conventional SMC with boundary layer, and the HOSM Super-Twisting controller.

Because the goal of the controller is to keep the roll and pitch angles at zero, the switching function, s, is selected to be the difference between the measured angle and the desired one for the roll, θ , and for the pitch, φ.

$$s_\theta = \tilde{\theta} = \theta_d - \theta \tag{5.5}$$

$$s_\varphi = \tilde{\varphi} = \varphi_d - \varphi \tag{5.6}$$

where θ_d and φ_d are the desired angle values.

The output of the attitude controller generates a pair of hip velocities, and therefore it regulates the rate of change of the angular position of the helicopter body

$$u_\theta = f(\dot{\theta}) \tag{5.7}$$

$$u_\varphi = f(\dot{\varphi}) \tag{5.8}$$

Hence, the system is relative degree one since the control input appears in the first derivative of s. Moreover, this controller doesn't require to calculate the time derivative of s or to measure the rate of change of the error.

5.7 Slope landing simulation with boundary layer SMC

In this section, a SMC is applied into our control system, by modifying the high-level attitude controller, and replacing the PD algorithm with a SMC algorithm with boundary layer control. This controller is implemented in the 4-legged dynamic model and is tested in the same scenario as in section 5.5.1., simulating a landing at a 20° slope with a deviation of 10° with respect to the direction of the slope to compare the results. The SMC is not implemented into the force controller as it only acts during the short time between the first and the last leg touching the ground.

The control signal of the boundary layer controller is given by:

$$u = K_D sat(s, \delta) \tag{5.9}$$

where K_D is the controller gain and the function $sat(s, \delta)$ is given by the equation 2.31.

The parameters of the controller are tuned manually by trial and error. In boundary layer Sliding Mode, the parameter selection is rather intuitive. The controller gain represents the maximum absolute value of the leg velocity due to the attitude controller

and it's selected at K_D=0.06 m/s, while the boundary layer thickness is selected to keep the sliding variable within δ=0.08 rad. These parameters are tuned manually.

Figure 5-17 shows the control signal *u*, of the boundary layer controller for different values of the sliding surface, *s* (Similar to Figure 2-16(c)). It can be seen how the output of the controller is equal to the maximum output value for large values of s, and it decreases linearly as s approaches to zero. The smoothing of the output near the zero error zone, prevents the controller to "push" the system too hard and avoids or reduces the chattering effect. The slope of the output value is controlled by the appropriate selection of K_D and δ.

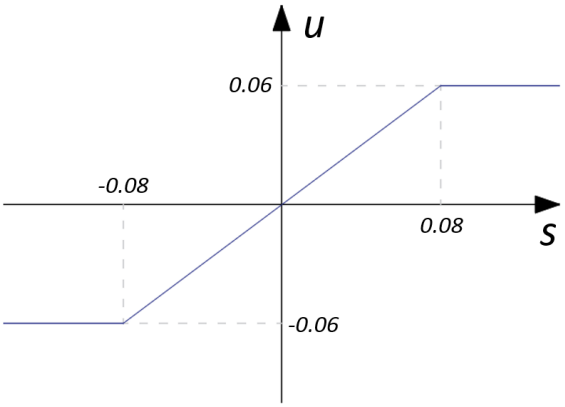

Figure 5-17 SMC output as a function of *s*.

The parameters for the pitch controller are K_D=0.02 m/s and δ=0.08 rad. The rest of the simulation parameters are the same as in section 5.5.1.

Error! Reference source not found. shows the results of the simulations. The top row shows the orientation of the main body (roll, pitch and yaw angles) where the inclination peaks at 3.5° in the roll direction. Both the pitch and yaw angles are kept within less than 1°. The left side of the graph shows the evolution of the legs position (hip-foot distance in the z-direction and joint angles) after touchdown and until the end of the landing operation. The right side shows the ground reaction forces and joint torques.

The motion of the legs is smooth and the inclination of the main body presents slightly higher peak values but with similar performance with respect to the PD algorithm.

The joint controllers used for this simulation are the same that the ones in the previous section with the same PID parameters indicated in Table 5-4 (P:10, I:14, D:0.4). Here

we can see that the joint torques present a more discontinuous behaviour with high-frequency oscillations.

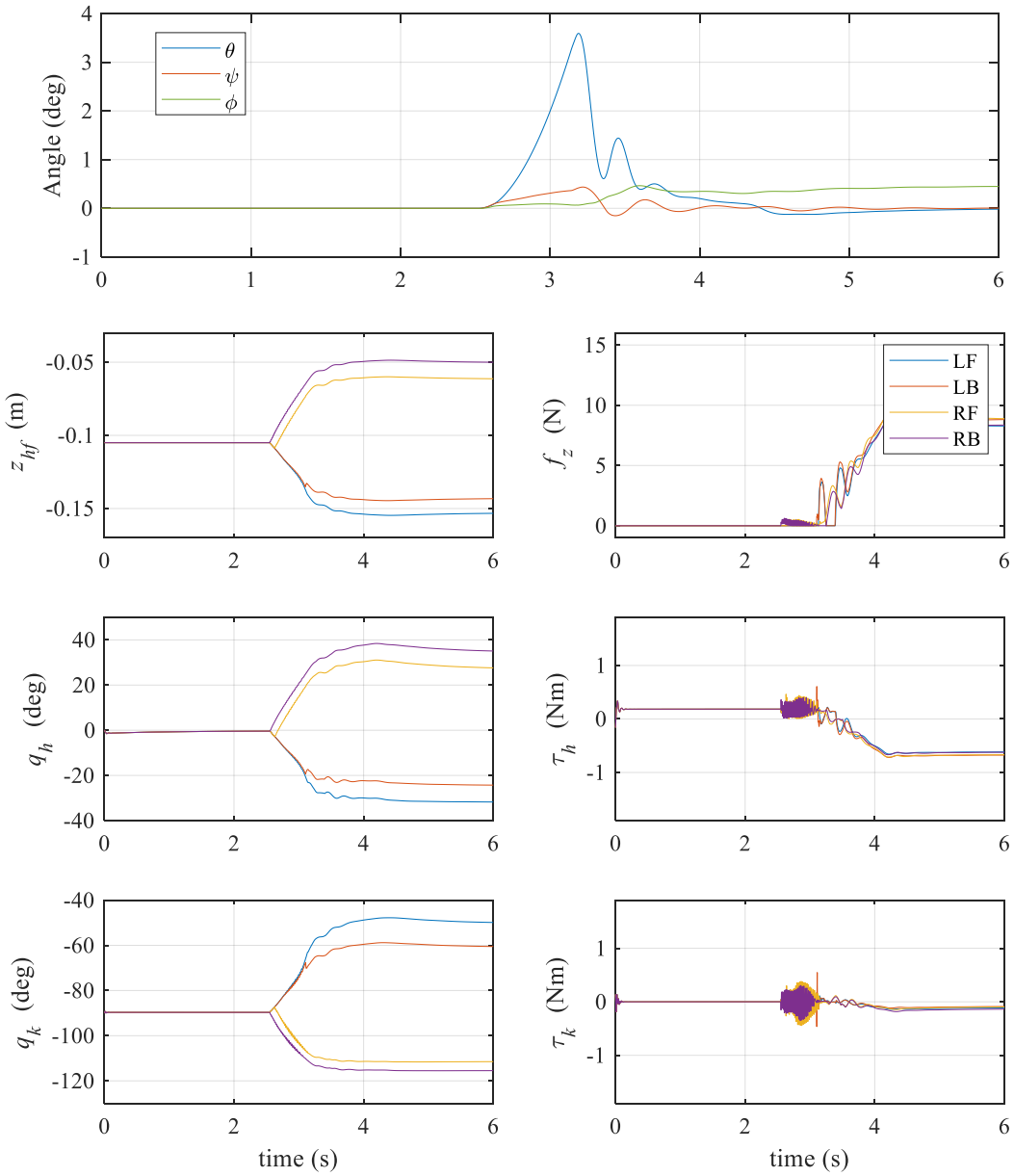

Figure 5-18 Top row shows main body's Roll-Pitch-Yaw angles. Next rows on the left side from top to bottom: legs hip-foot z-distance, hip angles and knee angles. Right side from top to bottom: normal ground reaction forces, hip torques and knee torques.

Figure 5-19 shows the result of another simulations with lower PID parameters (P:5, I:7, D:0.4). In this case, the discontinuities are eliminated from the joint torques, but the performance of the control system degrades and oscillations in the main body orientation and legs motion start to appear.

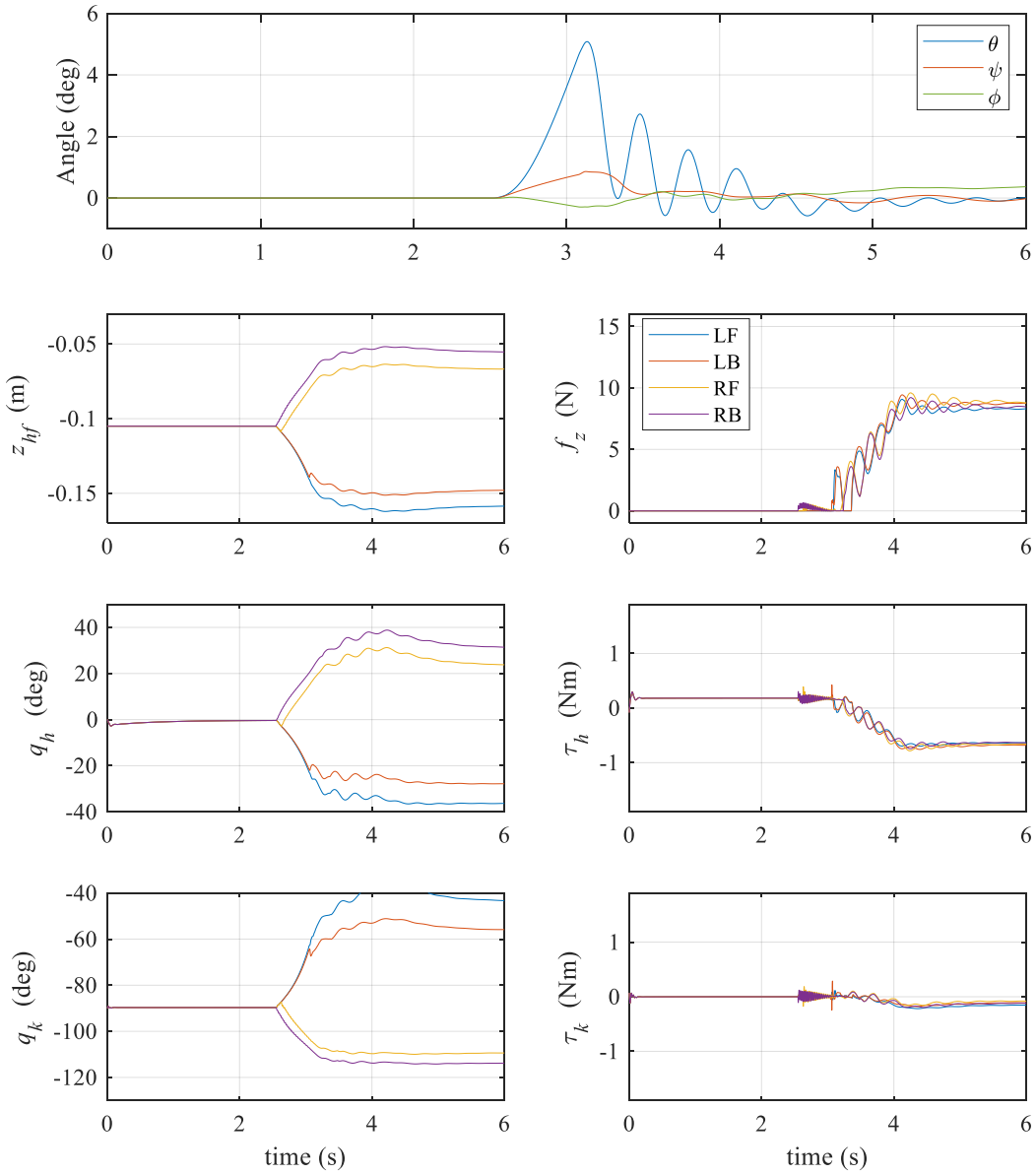

Figure 5-19 Top row shows main body's Roll-Pitch-Yaw angles. Next rows on the left side from top to bottom: legs hip-foot z-distance, hip angles and knee angles. Right side from top to bottom: normal ground reaction forces, hip torques and knee torques.

In the control system of the landing gear, the interaction with the ground is dealt at two levels. The high-level controller reacts to the changes in body orientation and feet pressure and sends appropriate commands to the joint controllers. A possible cause for the lower performance is that the SMC controller might not have the same robustness that the PD controller to respond to these changes, and because of this, more efforts are transmitted to the joint controllers which leads to the appearance of the high-frequency oscillations.

5.8 Slope landing simulation with super-twisting SMC

In this section, the system is tested with a higher order sliding mode. As in the previous section, the high-level attitude controller is replaced this time with a Super-Twisting algorithm. The landing gear is tested in the same conditions than the previous tests to analyse and compare the performance of the controller.

The tuning of the controller is done using the simplified version of the Super-Twisting algorithm, considering that there is no bound in the control or any boundary layer. The single-parameter tuning methods proposed in [45] and [48] are used for simplicity. Figure 5-20 plots the value of W versus λ, using Method 1 ($\lambda = 1.5\sqrt{C}$) and Method 2 ($\lambda = \sqrt{C}$) for different values of C.

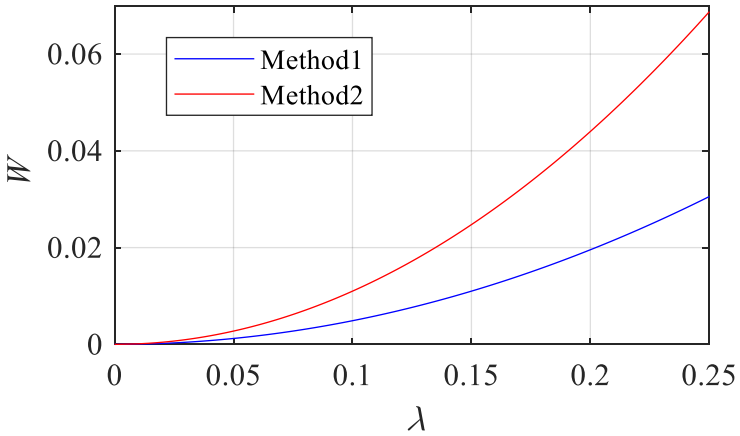

Figure 5-20 Super-Twisting parameters W versus λ using tuning method 1 ([48]) and 2 ([45]).

The results of the slope landing simulation with the 4-legged model are shown in Figure 5-21. Here, the roll controller is tuned using a value of λ=0.1, and W_1=0.005 (Method 1), and W_2=0.011 (Method 2). The parameters for the pitch controller are λ=0.025, and W_1=0.0003, and W_2=0.0006. Method 1 (left side) has a higher peak inclination than Method 2. In both cases chattering starts to appear near the sliding surface. Increasing any of the parameters degrades the performance of the controller, while reducing them also decreases the chattering but increases the peak inclination.

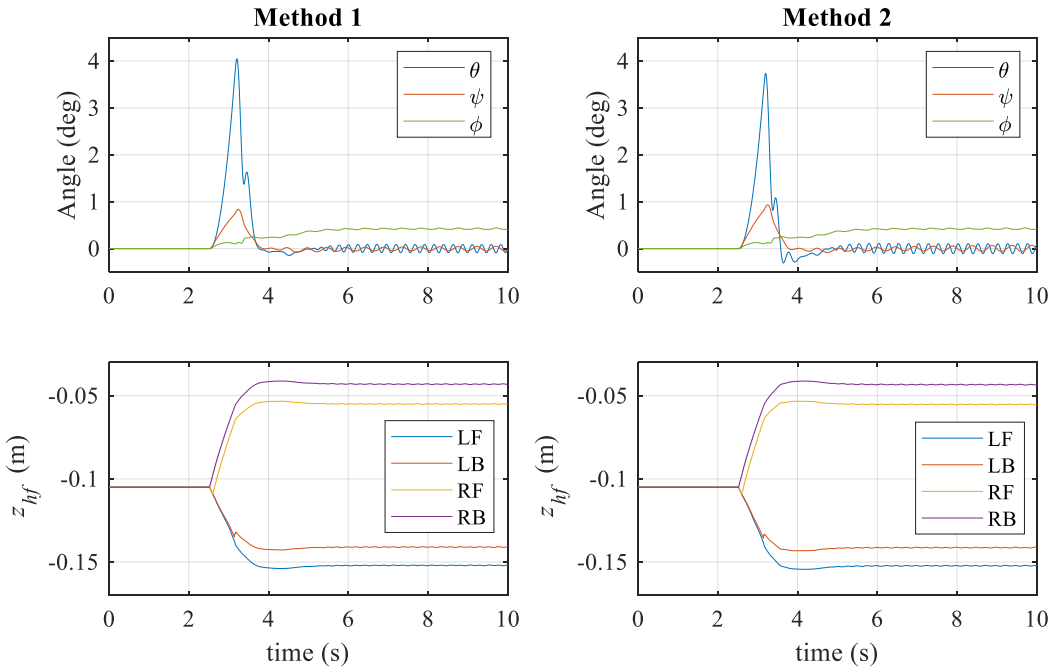

Figure 5-21 Main body's Roll-Pitch-Yaw angles (top) and legs' hip-foot z-distance (bottom) using tuning Method 1 ([48]) and 2 ([45]).

5.9 Simulations with the 2-legged model

To further analyse the performance of the SMC with boundary layer and Super-Twisting algorithms, these controllers are also implemented in the 2-legged planar model of the landing gear and are tested on the same conditions: with a descending rate of -0.1 m/s and terrain slope of 20°. Figure 5-22 shows that both controllers perform better in this case than with the 4-legged model. In this simulations, the controller parameters could be set to higher values without resulting in increased chattering, thus, reducing the peak inclination during the landing. The parameters selected were K_D=0.3 m/s and δ=0.08 rad for the boundary layer algorithm and λ=0.5, and W=0.1 for the Super-Twisting.

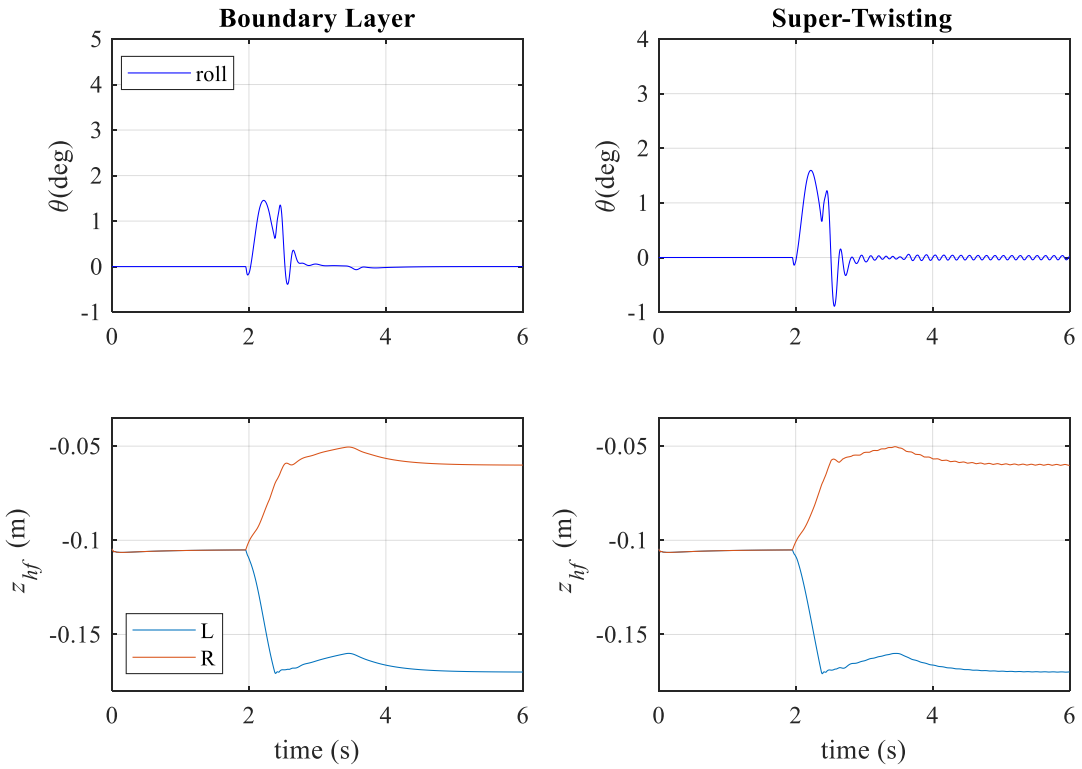

Figure 5-22 Roll angle and legs hip-foot distance (right and left leg) using the boundary layer and Super-Twisting SMC in slope landing.

5.10 Conclusions

This chapter presented the control system of the robotic landing gear, composed of the low-level joint controllers and the high-level attitude and force controller. When landing on uneven terrain, the high-level controller adjusts the position of the legs to adapt them to the ground conditions and keeping the helicopter levelled. Proportional-Derivative controllers have been designed that use foot pressure and body attitude feedback to calculate the appropriate feet positions. Feet positions are then converted from Cartesian coordinates into joint position commands using inverse kinematics equations. Low-level joint controllers use this joint position commands to calculate the appropriate joint torques to produce the desired motion. Two alternative controllers to the PD attitude controller have been proposed and implemented using a boundary layer SMC and Super-Twisting algorithms.

The performance of the control system has been tested in software simulations using the mathematical models presented in Chapter 3. The 2-legged and 4-legged models have been used to perform landing simulations on flat terrain, slopped grounds and multi-level surfaces, using different controller configurations and simulation parameters. The controller showed good capability to absorb impact forces produced

during landing and to keep the helicopter body within acceptable inclination levels. The simulation results in this chapter represent a source of information to evaluate the validity of the mathematical models and assess the performance of the controllers.

On the modelling side, the 2-legged model was obtained using a full-rigid-body approach, applying Newton-Euler to every link of the system, while the 4-legged one was obtained by using a model decomposition. Here, the motion of the main body and the ground reaction forces are calculated using a centroidal dynamics model and the joint torques and leg motion are calculated using four single-leg models. The full-rigid-body approach fully-describes the relationship of the torques and forces acting on the system and the corresponding motion of each link, thus, it is reasonable to say that is more accurate, but it is also more complex and difficult to obtain. The model decomposition, introduces a series of assumptions and simplifications to reduce the complexity of the model, but it can lead to a loss of some degree of accuracy. The 2-legged model is a planar model and, thus, it has limitations and its motion is more constrained, while the 4-legged model allows motion in the 3 dimensional space and is more representative of the reality.

Both models work well in the environments where they were tested, and all the control algorithms were successfully implemented, however, the 4-legged model proved to be more sensible to model parameters variations, while the 2-legged model is less affected by them.

Regarding the algorithms that have been used for the attitude controller, the PD and the SMC with boundary layer produced similar results in terms of body orientation. The SMC produces slightly higher peak inclinations at the moment of touchdown, but the motion of the legs is more linear with less oscillations than the PD. Both controllers are easy to tune and require little sensory information, but the PD requires to calculate the derivative of the error, which can generate problems in the case of measurement noise. The SMC however, transmits more high-frequency oscillations to the low-level controllers. In the case of the Super-Twisting algorithm, the peak angles obtained were higher and the controller showed difficulty the settle down, oscillating around the level position, in some scenarios. The results of the PD controller are more consistent through all simulations done with both models and with all different joint controllers' configurations.

6 Laboratory Experiments

In previous chapters, mathematical models and simulations were carried out to test the proposed control architecture. This chapter uses the prototype system presented in chapter 4 to validate the simulation results on a hardware platform.

First the experiment setup is explained, then the controller digital implementation on the robot is described, and finally the results obtained from the hardware tests are shown and analysed.

6.1 Experiment setup

The robotic landing gear prototype explained in chapter 4 has been designed to fit on the model helicopter Align T-Rex 500L Dominator which has been used for the laboratory tests. The main dimensions and weight are specified in the Table 6-1 and Figure 6-1 below:

Table 6-1T-REX 500L Dominator specifications [78]

Property	Value
Length	863 mm
Height	285 mm
Main Rotor Diameter	978 mm
Weight (without battery)	Approx. 1500 g

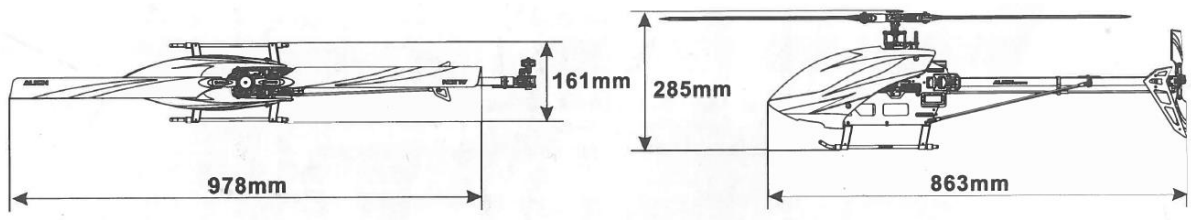

Figure 6-1 T-REX 500L Dominator main dimensions [78]

The robotic landing gear is attached to the helicopter replacing the original skid-type landing gear as shown in Figure 4-1(b). The total weight of the model helicopter, with the battery and the robotic landing gear is about 3 kg. The helicopter and the landing gear use separate batteries as the first one operates at 24V while the servo motors on the landing gear need a 12V power supply.

During the course of this project, the model helicopter was assembled, set-up and calibrated for flight, although flying tests haven't been performed due to the lack of access to large and un-transited outdoor space where this can be done. Unlike modern drones, model helicopters are more unstable and difficult to control and it is not safe to fly them indoors. For this reasons, the laboratory tests have been carried out using a pulley system with a descending rate controlled by a servo motor which allow test data to be collected.

The uneven terrain recreation is made up of wooden pieces and boxes of different heights. A wooden platform tilted at different angles of inclination is used to simulate a slope landing. Tests are done using several configurations of these obstacles on the ground.

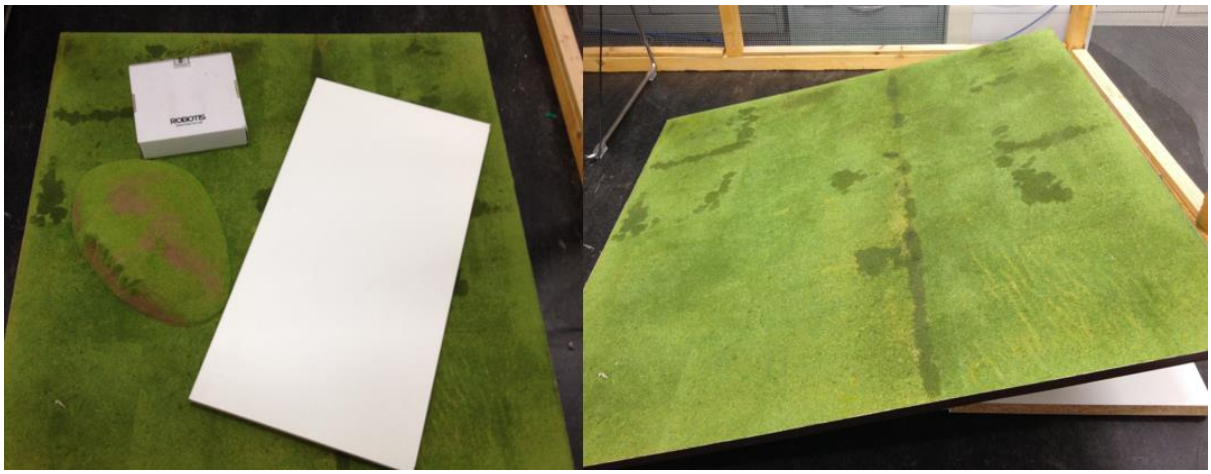

Figure 6-2 Elements used to recreate uneven terrain

6.2 Limitations of laboratory experiments and deviations from simulations

There exist a number of differences between simulating a landing with a pulley and doing it in a real flight. During a real flight, the orientation of the helicopter in space during the descent is totally controlled by the forces produced by the main and tail rotors, but with the pulley descent, only the descent rate is controlled, while the

orientation of the body will oscillate freely due to the irregular mass distribution of the rotorcraft. With the pulley system, the descent rate is constant from the beginning until the rotorcraft has completely landed, but in a real flight the pilot normally has the control authority to increase or reduce the speed according to the situation, i.e., it can reduce the landing speed if the rotorcraft body tilts too much. The pilot can also correct the body orientation with the helicopter controls. These elements give a better controllability of the situation with a real flight compared to the laboratory pulley test, however, the main rotor operation during real flight can introduce vibration and noise that might affect to the performance of the landing gear sensors. The effects of the noise can be attenuated by increasing the filtering of the sensor signals and, even that this can introduce delays, it shouldn't affect the performance of the system given that the descent rates are relatively small. An attempt to simulate a real "controlled" flight was done using a vertical slider with a horizontal rod attached to the helicopter (as shown in Figure 6-3), however this solution was discarded as it constrained too much the motion of the system and amplified the vibration produced by the rotor.

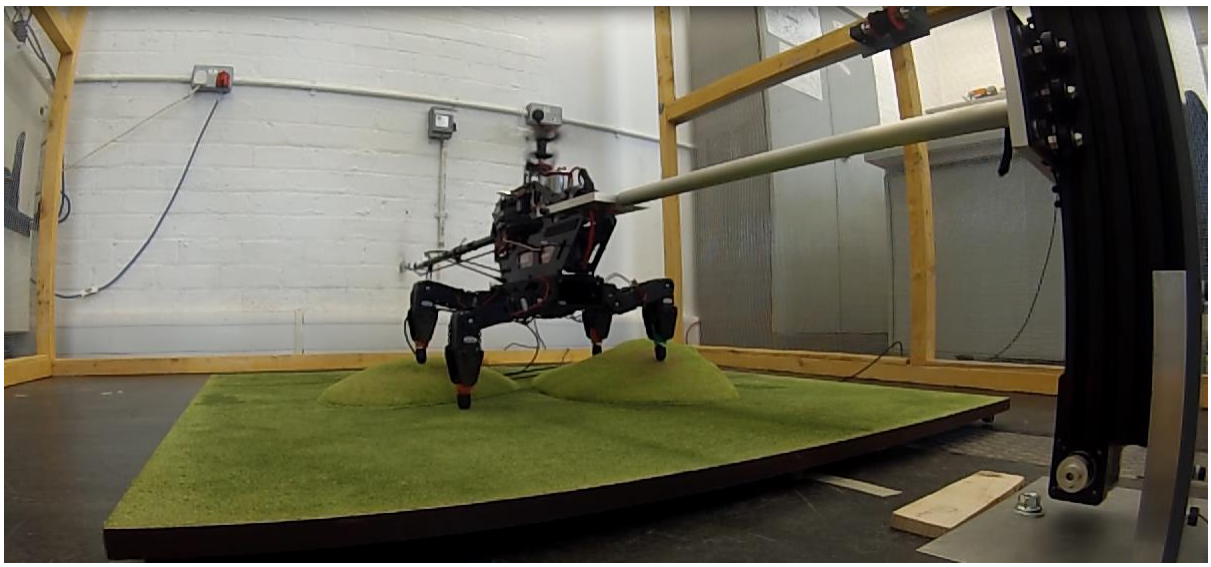

Figure 6-3 Landing experiment with vertical slider

With respect to the simulations, one of the main differences is that it's not possible to implement the joint controllers from the simulations into the physical prototype, as the servo motors incorporate its own built-in controllers. The internal motor controllers are position-based and don't allow for the implementation of PID control, thus, the goal of the experimental tests is to validate the performance of the high-level controller, and show that it can be implemented in different systems independently of the joint controllers. Some of the parameters used in the simulations like the body inertias, are

difficult to estimate due to the irregular mass distribution and the placement of the components in the model helicopter. In the simulations the inertia of the helicopter body is calculated as if it was a rectangular box. This will affect the performance of the roll and pitch controllers which will need to be fine-tuned, especially the pitch controller, due to the presence of the tail in the longitudinal axis of the helicopter.

Other sources of divergences will be the control frequency which was set at 1 kHz in the simulations, but in the laboratory tests it is limited to 20 Hz due to hardware limitations and code size. This is expected to affect the reaction time of the controller. Physical properties limitations can also have an impact like the maximum motor speed or position resolution, and filters used in the IMU and FSR signals can be also contribute to the overall delay.

6.3 Controller Implementation

As previously explained in chapter 4, the control system of the robotic landing gear is implemented into the on-board microcontroller using the Arduino IDE. This includes the definition of the logic that will drive the control flow of the program and the digital implementation of the control algorithms defined in chapter 5.

Figure 6-4 and Figure 6-5 show the control flow diagram of the logic implemented for the landing simulation experiments. The program starts with the legs retracted for flight mode while the distance sensor measures the distance to the ground. When the system descends to landing-initiation distance, the legs extend to landing position and the foot pressure sensors start to check for ground contact. If a sensor detects an increase in the measured pressure that reaches a set threshold, it activates the control system. At this moment, both the force and attitude controllers are activated, and the legs that are in contact with the ground retract and the ones not in contact extend, while at the same time the attitude controller will adapt the legs if the system is tilting in any direction. Once all four feet are in contact with the ground, the force controller is deactivated, and the attitude controller continues to be in action until the system is stabilized at a level position. A counter is started at this time. If, during one second, the angles are kept within a range of ±0.01 rad around the setpoint and all legs are making ground contact, the system locks the position of the servo motors. As the flowchart shows, if the landing operation is interrupted at any point, the control system can go back to the initial conditions.

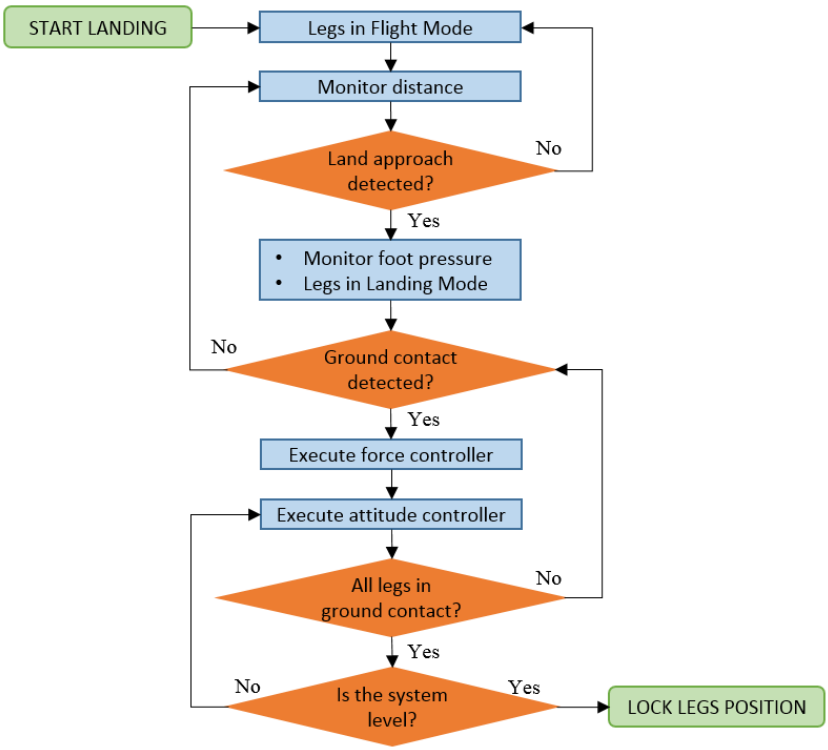

Figure 6-4 Landing decision flowchart

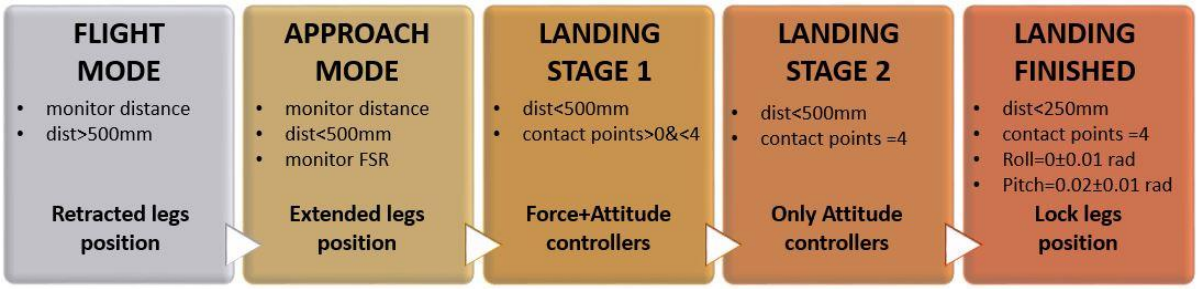

Figure 6-5 Sequence of stages during normal landing operation

During the landing operation, the control system executes the force and attitude controllers. These consist of three PD controllers that use force feedback from the FSR sensors, and the roll and pitch angles form the IMU as defined in chapter 5 (equation 5.3). The output of each PD, controls the velocity at which each foot extends or retracts

$$\dot{z}_{hf} = k_P e + k_D \dot{e} \tag{7.1}$$

This output is then integrated in order to send it to the joint controller as a foot position command.

$$\Delta z_{hf} = \int \dot{z}_{hf} = \int (k_P e + k_D \dot{e}) \tag{7.2}$$

The high-level controller output is the sum of the three controllers' outputs. To convert these continuous-time algorithms into discrete-time form for computer digital implementation the z-transform is used.

For each of the controllers, first we consider the Laplace transform of equation 7.2

$$\frac{U(s)}{E(s)} = (k_p + \ k_d \cdot s)\frac{1}{s} \tag{7.3}$$

where *U* is the controller output and *E* is the input error. k_p and k_d are the controller parameters.

And then the s-transform function is converted into a z-transfer function using the substitution [79]

$$s \rightarrow \frac{1 - z^{-1}}{T_S} \tag{7.4}$$

where T_S is the sample time.

Resulting in

$$\frac{U(s)}{E(s)} = (K_P + K_D s)\frac{1}{s} \ \rightarrow \ \frac{U(Z)}{E(Z)} = \left(K_P + K_D \frac{1 - z^{-1}}{T_S}\right)\frac{T_S}{1 - z^{-1}} \tag{7.5}$$

$$U_z = U_{z-1} + [K_P e_z T_S + K_D(e_z - e_{z-1})] \tag{7.6}$$

where U_z is the current controller output and U_{z-1} is the previous output.

Looking at equation 7.6, it can be observed that the PD controller after the integration becomes a PI controller.

Regarding the low-level controllers, each servo motor has a built-in controller, hence, for the practical tests, only the high-level controller is implemented while the motors internal controllers have been used as joint controllers.

Table 6-2 Domain Transforms.

Time-Domain	**s-Domain**	**z-Domain**
$u(t) = \int (e \cdot k_p + \ \dot{e} \cdot k_d)\, dt$	$\frac{U(s)}{E(s)} = (k_p + \ k_d \cdot s)\frac{1}{s}$	$U_z = U_{z-1} + [K_P e_z T_S + K_D(e_z - e_{z-1})]$

The same procedure applies for the sliding mode control algorithms.

6.4 Experiments with PD controller

Extensive laboratory experiments have been done with the landing gear prototype to validate the simulation results and to assist tuning the controllers. The tests are carried out following the experimental setup described in the previous section using the pulley system and the uneven terrain recreation. In this section, the results of three different tests are shown, where the system lands on a flat surface, a slopped terrain at 20°, and an uneven terrain with obstacles of up to 12 cm height.

6.4.1 Controller Tuning

During the initial experiments the gains of the force and attitude controllers were tuned separately (see Figure 6-6). The force controller was tuned implementing it in an individual leg and placing an obstacle under that leg. The landing gear was descended manually at different speeds and the controller gains were tuned (starting from low values and increasing) so that when the leg touched the obstacle it produced a smooth leg retraction, moving with the ground to don't lose contact, and without exerting too much force against it, that could disturb the helicopter attitude. The rate of extension of the other legs was easy to tune as it is a constant.

The attitude controller was tuned placing the landing gear on a slope. First the roll controller was tuned placing the longitudinal axis of the helicopter across the slope and increasing the controller gains until the legs motion would drive the roll angle to zero fast enough but not too aggressively to produce oscillations. The same procedure was followed for the pitch angle, placing the transversal axis of the helicopter across the slope. The ideal performance of the pitch controller was obtained with much smaller gains than those for the roll controller. This was expected, because of the inertia added by the mass of the tail far from the CoM, which would create instability if the moment created by the legs' motion is too high.

Once all the controllers are tuned individually, they are added together and fine-tuned. The values used for the simulations in this chapter are k_p =0.00025, k_d =0.000025 for the force controllers. For the attitude controllers, two sets of values are used. In the initial stage between the first and the last leg touchdown (Stage 1 in Figure 6-5), the values are k_p =0. 25, k_d =0. 025 for the roll controller and k_p =0. 0375, k_d =0. 00375 for the pitch controller. From this point until the end of the landing operation (Stage 2 in Figure 6-5) the values are k_p =0. 5, k_d =0. 05 for the roll controller and k_p =0. 075, k_d

=0. 0075 for the pitch controller. This way, during the first stage, the output of the attitude controller is decreased to give more importance to the force controllers. During the second stage the force controller is switched off and the control action is executed by the attitude controller alone.

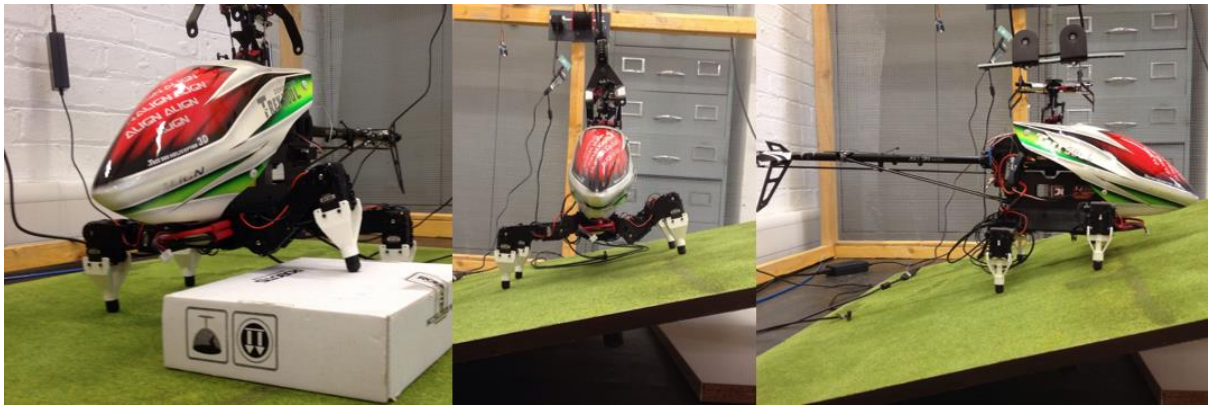

Figure 6-6 Landing gear during tuning of the force (left), roll (middle) and pitch (right) controllers.

6.4.2 Flat landing test

In this test, the helicopter with the landing gear is descended at -0.1 m/s onto a flat surface. The goal of the experiment is to check how the sensors and the control system react in this kind of terrain. The first graph in Figure 6-7 shows the reading of the distance sensor, where the distance from the bottom of the landing gear to the ground starts at around 850 mm, descends at a constant rate and finishes at 80 mm from the ground. The second graph shows the computation of the roll and pitch angles from the Inertial Measurement Unit and the kalman filter, and the evolution of the helicopter inclination during the landing. Before the descent, the roll and pitch angles are at a constant value. At 3 seconds after the start of the test, the servo motor starts to actuate the pulley system and the helicopter starts to descend, with the roll and pith angles oscillating around the zero value, especially the pitch due to the effect of the tail. The three vertical marks in the graph represent respectively the moments where the first leg touches the ground, when the last leg touches the ground and when the landing operation finishes and the position of the legs is locked. The time difference between the first and second mark is very little as almost all legs touch the ground at the same instant. After touchdown, both angles are quickly driven to their respective setpoints, which are 0° for the roll angle and 1° for the pitch. The selection of the pitch setpoint aims to tilt the helicopter slightly forwards to compensate for the tail weight.

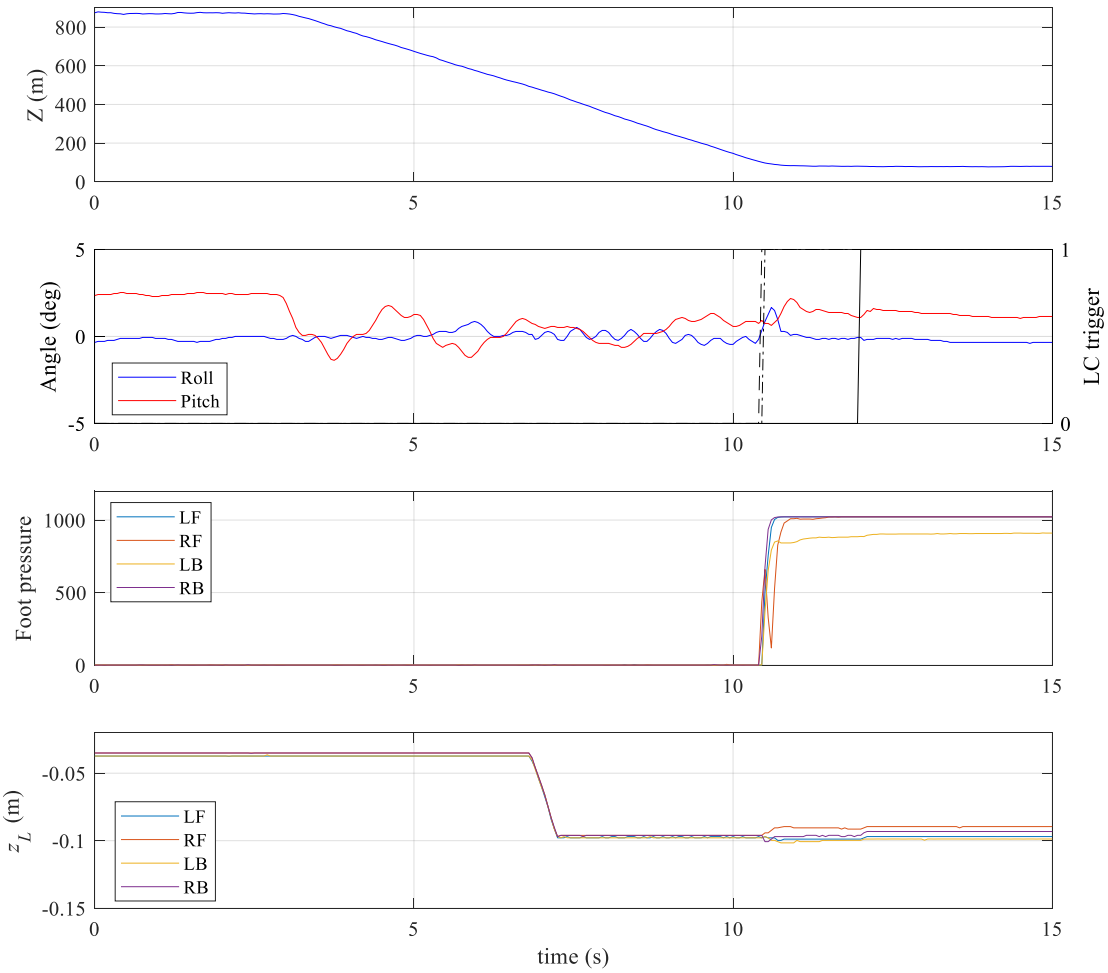

Figure 6-7 Flat landing plots. From top to bottom: Distance sensor, helicopter attitude and level controllers trigger, foot pressure readings and hip-foot z-distance

The third graph shows the readings from the feet Force Sensing Resistors which are passed through a low-pass filter. Here, the raw measurements are used and no conversion to force units is done as it is not needed for the operation of the control system. The sensor readings go from zero to its final value with almost no transient. The last of the graphs shows the computation of the hip-foot z-distance on each leg based on the angular position reading from the servo motors encoders. The graph starts with the legs in the retracted position until the sensor distance measuring reaches the target of 500 mm. At this point, the legs extend to landing position. After touchdown very little adjustment is done as the landing surface is flat.

6.4.3 Slope terrain test

Another scenario tested on the simulations has been a slope landing of 20° with a misalignment of the helicopter's longitudinal axis across the slope. This situation is also recreated in the laboratory experiments for comparison and validation with the simulations. Figure 6-8 shows the results of the slope landing test. The helicopter

starts descending as in the previous experiment and makes the first ground contact at time 7s with the right back leg and right front leg almost at the same time, which start to retract. Left legs start to extend until time 8 s where they make ground contact. At this point the force controller is switched off. The peak roll and pitch inclination is about 3° at this point, when the attitude controller drives both angles to their respective setpoints. Force sensors show a transient between the first and last leg ground contact and then settle down at similar levels. After all legs made ground contact, it takes around 3 seconds to drive the pitch angle within the desired range and then trigger the locking signal for the legs.

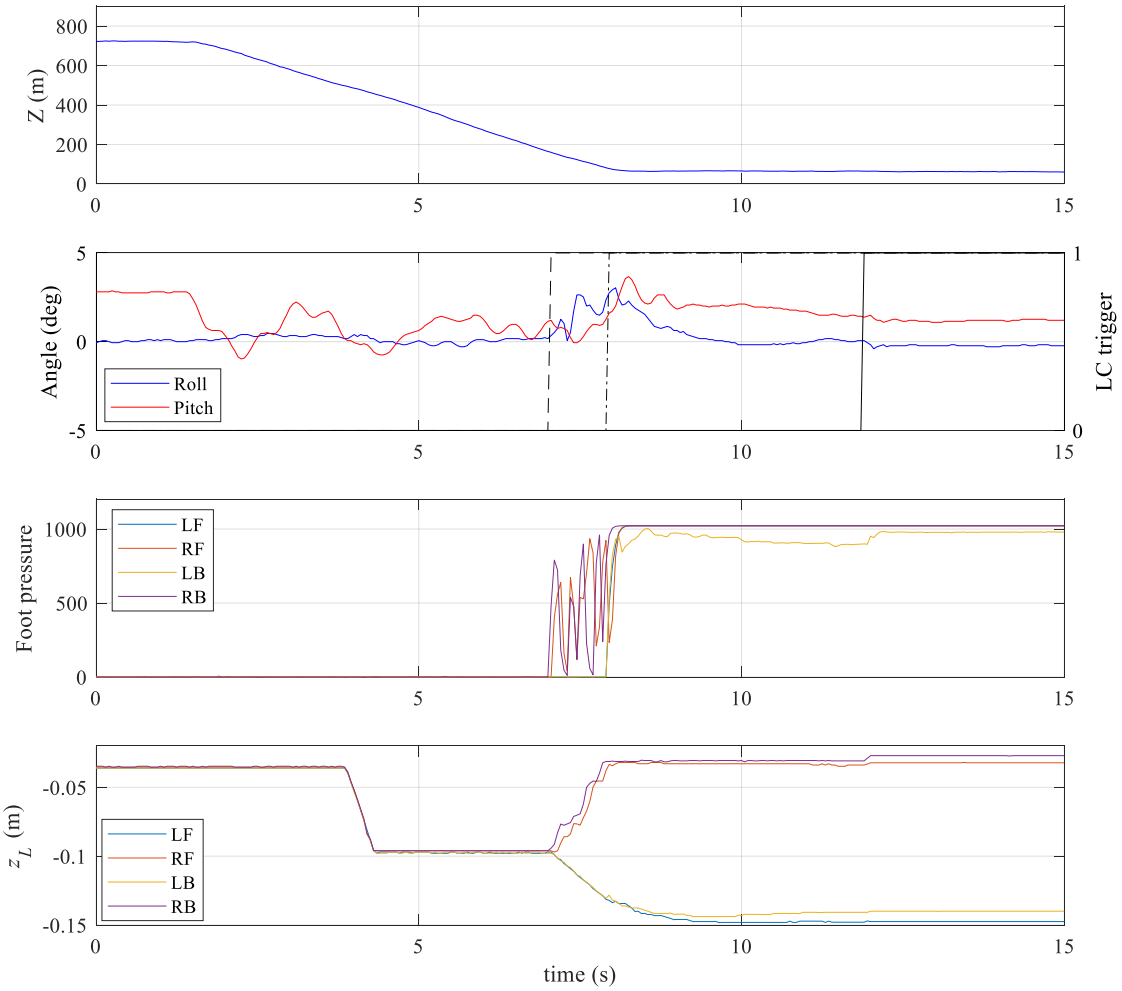

Figure 6-8 Slope landing plots. From top to bottom: Distance sensor, helicopter attitude and level controllers trigger, foot pressure readings and hip-foot z-distance

Figure 6-9 Robotic landing gear during slope landing

6.4.4 Uneven terrain

In this experiment, an uneven ground is recreated placing obstacles on the ground (Figure 6-10) with a maximum step height of 12 cm. As seen in Figure 6-11, the right front leg is the first to make ground contact followed by the right back, left back and left front. The time between the first and last leg ground contact is around 1 s and during this time each leg extends or retracts according to its state. The roll angle reaches a peak inclination of less than 3° and the pitch oscillates between 2.5° and -2° before being driven to its setpoint. As in the previous test, the roll angle is driven quickly near the 0° setpoint, while the pitch takes longer to reach the 1° target. This delays the trigger for the signal to lock the legs.

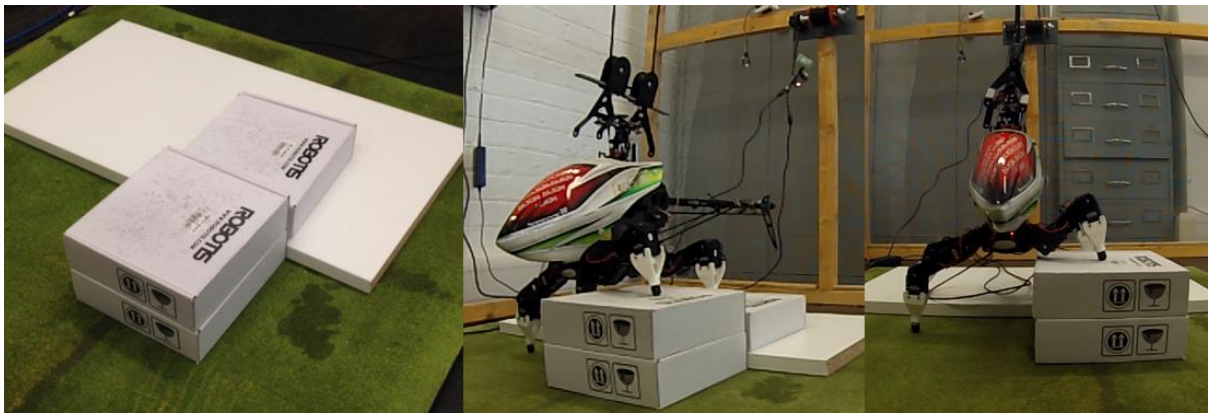

Figure 6-10 Robotic landing gear during uneven terrain landing

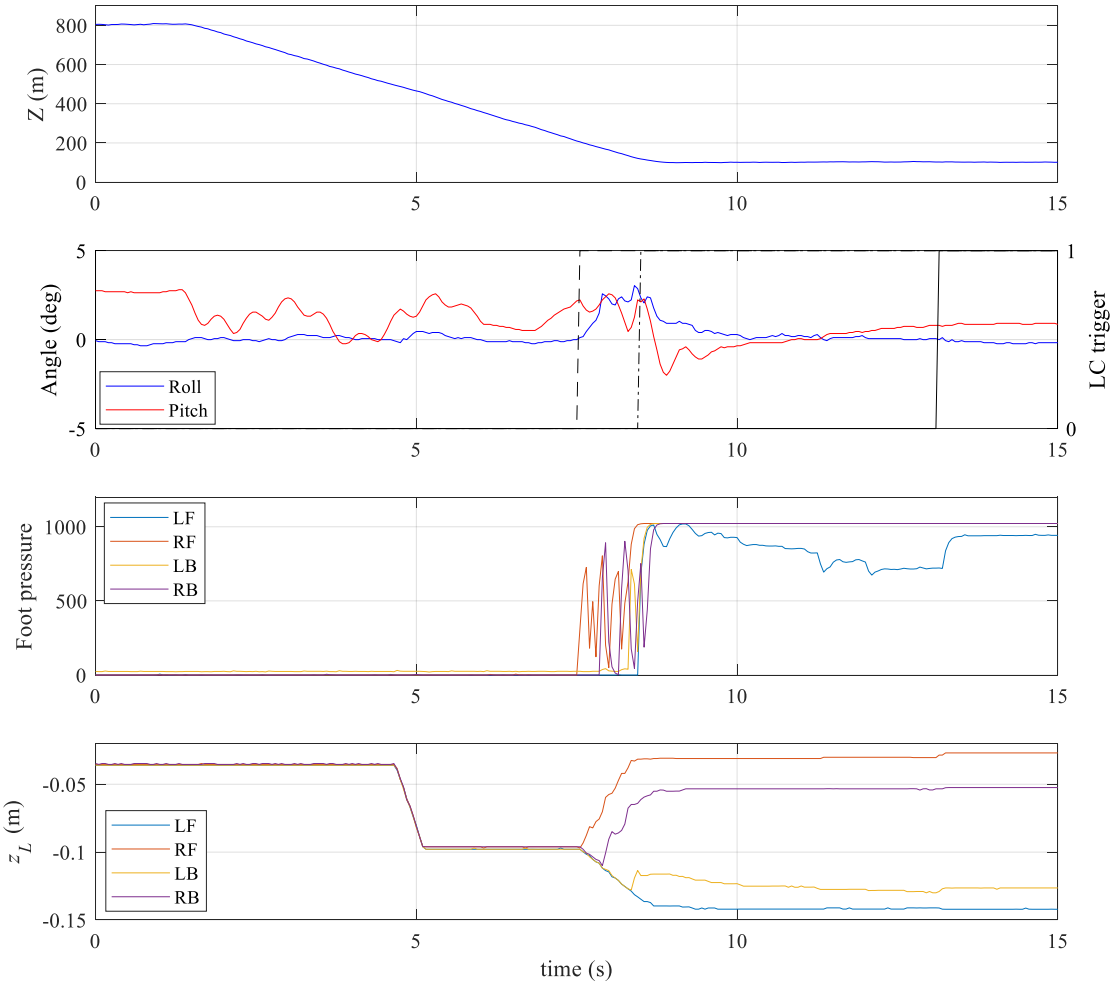

Figure 6-11 Uneven terrain landing plots. From top to bottom: Distance sensor, helicopter attitude and level controllers trigger, foot pressure readings and hip-foot z-distance

6.5 Experiments with SMC controllers

In Chapter 5, two SMC algorithms have been proposed to be implemented in the landing gear control system in substitution of the PD attitude controller. In this section the results of implementing these controllers in the physical platform are presented. The controllers implemented are the conventional SMC with boundary layer, and HOSM Super-Twisting algorithm. The tuning procedure is done manually, starting with similar values to the ones from the software simulations, and increasing/decreasing the gains depending on the performance. For the conventional SMC controller, the values used in these experiments are, K_D=0.1 and δ=0.2 for the roll angle, and K_D=0.035 and δ=0.2 for the pitch angle. For the Super-Twisting SMC, the values are, λ=0.1 and W=0.005 for the roll and λ=0.025 and W=0.0003 for the pitch.

6.5.1 SMC with boundary layer

Figure 6-12 shows the results of a slope landing test at 20° using the conventional SMC controller. The inclination of the helicopter peaks at 3° after touchdown and is quickly driven to zero. The performance of the controller is very similar to the PD controller in the same scenario. In this case, the pitch angle reaches its target faster than the roll angle. The signal to lock the legs is triggered 3 seconds after the last leg makes ground contact.

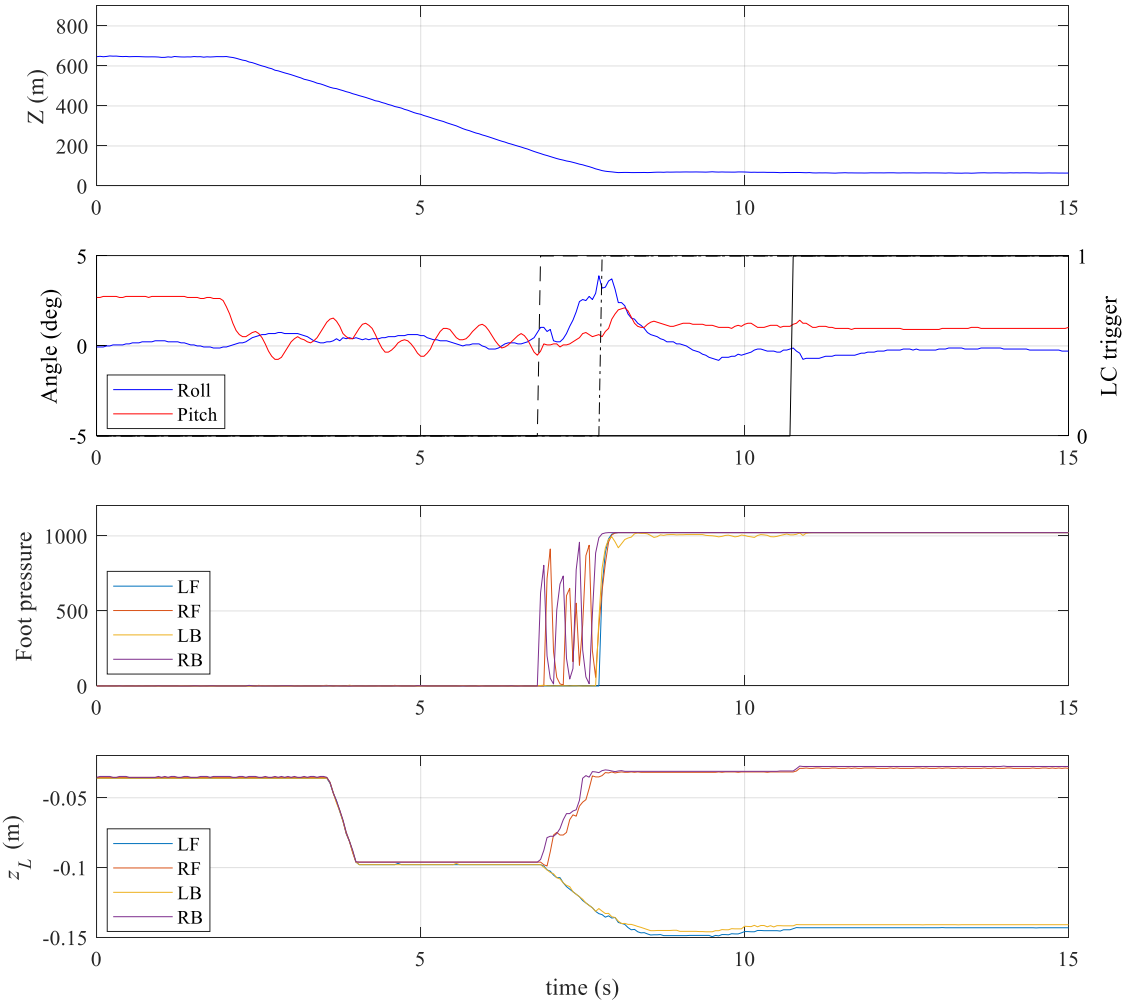

Figure 6-12 Slope landing plots with conventional SMC. From top to bottom: Distance sensor, helicopter attitude and level controllers trigger, foot pressure readings and hip-foot z-distance

6.5.2 Super Twisting HOSM

Figure 6-13 shows the Super-Twisting controller in the same landing scenario. In this case, the inclination of the aircraft peaks at 4° and it is driven to zero. The controller, however, produces more oscillations around the target position before the signal to lock the legs is triggered.

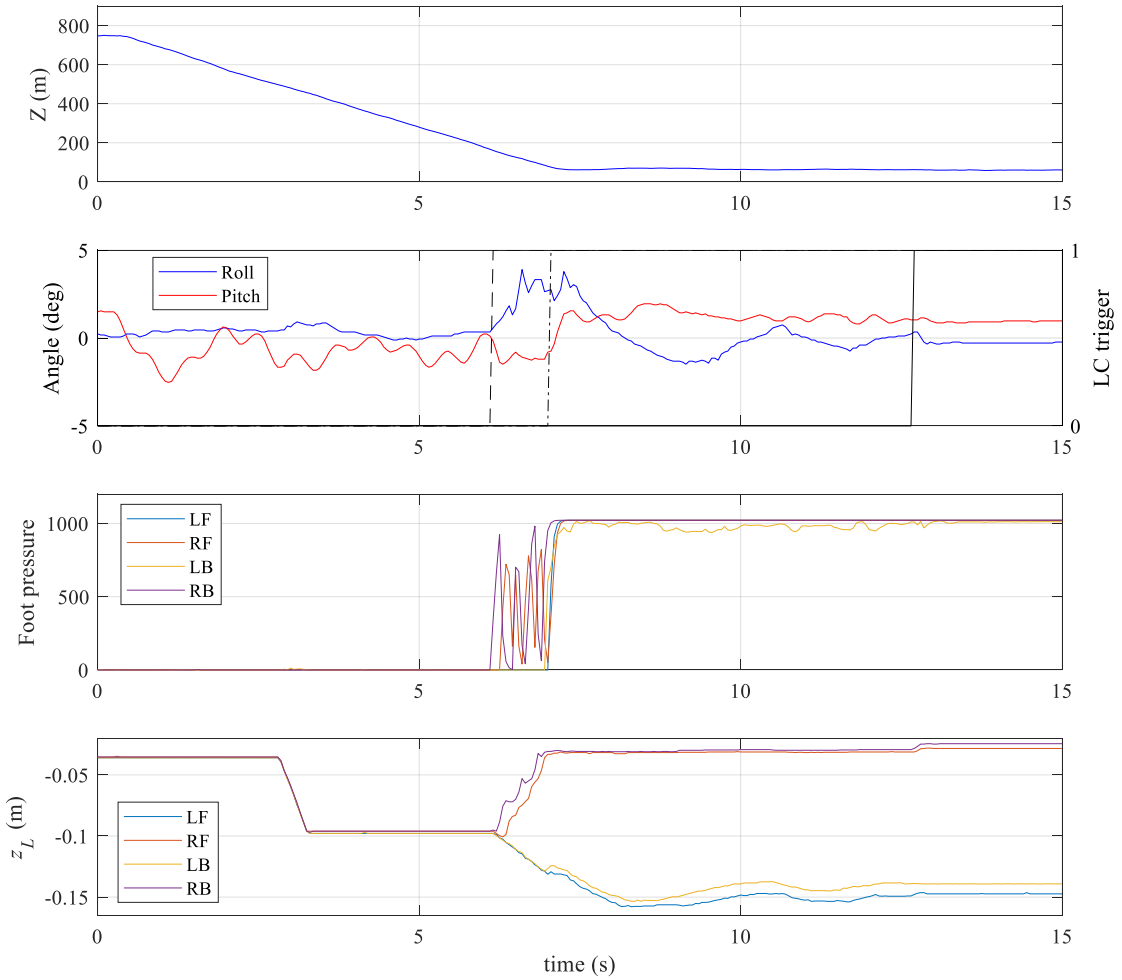

Figure 6-13 Slope landing plots with Super-Twisting SMC. From top to bottom: Distance sensor, helicopter attitude and level controllers trigger, foot pressure readings and hip-foot z-distance

To test the capability of the three controllers to keep the body attitude within the accepted range for a longer period, additional tests were done increasing the time before the system locks the legs' position. In this case, the controller would have to drive the system to the target angles and keep it for 3 seconds before the system would lock. As shown in Figure 6-14, the PD and conventional SMC controllers were able to accomplish this goal. However, the Super-Twisting controller had problems to make the system settle down, and the aircraft would start to oscillate around the target angles and become unstable. Because the controller is unable to keep the angles within the desired range for 3 seconds, the signal to lock the legs is not triggered. Additionally, the peak inclination with this controller is higher than with the other two. The controller that triggers the signal in less time is the conventional SMC.

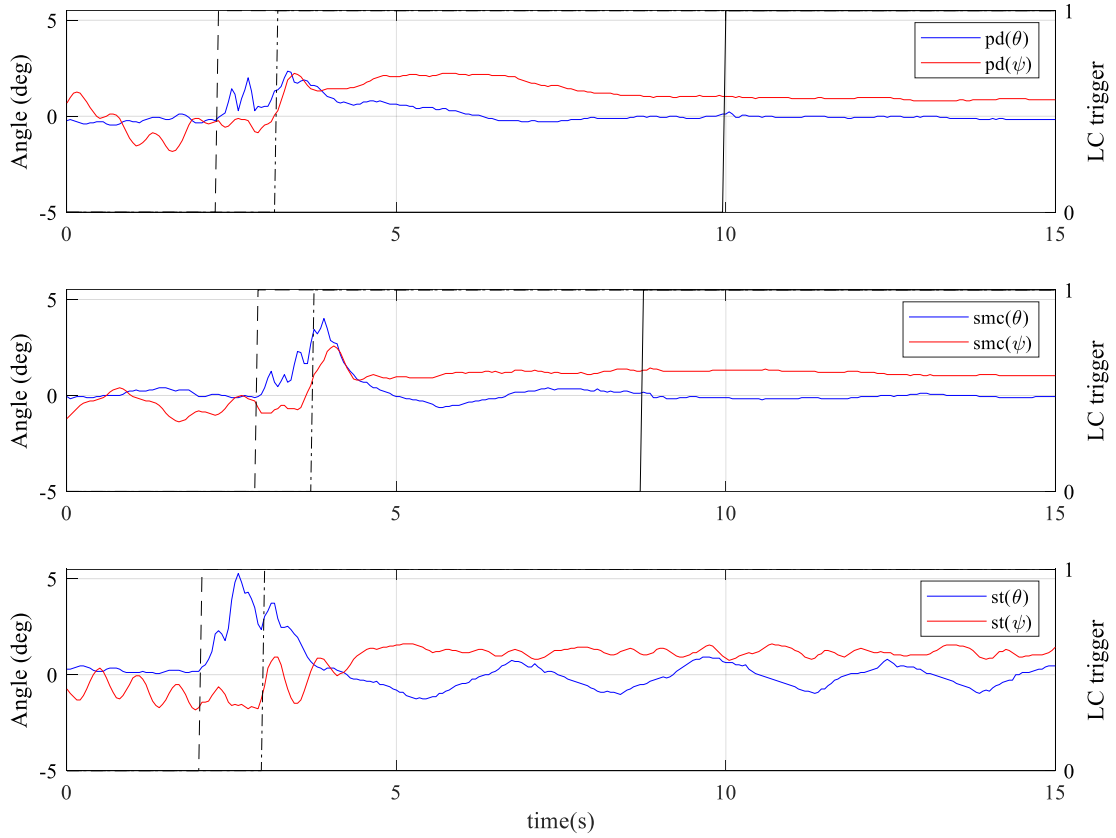

Figure 6-14 Slope landing plots of the roll and pitch using PD (top), conventional SMC (middle) and Super-Twisting (bottom) controllers.

6.6 Conclusions

This chapter presented the implementation of the control system into the hardware prototype and the laboratory experiments conducted. For the experiments, the landing gear is attached to a model helicopter with a total weight of 3 kg. The system has its own battery and all the computation is done in an on-board microcontroller. The experiments consisted in using a pulley system to descend the prototype into a recreation of an uneven surface to simulate the landing conditions on rough terrains. The descent velocity used in all the experiments has been -0.1 m/s and the tests included landing on a flat area, on a 20° slope, and on a surface with different height levels with a maximum step size of 12 cm.

The implementation of the controller into the prototype only covers the high-level controller, as the servo motors used in the landing gear, have their own built-in controllers. Thus, the experiments only evaluate the performance of the high-level control system. The controller tuning procedure is also explained.

Three different algorithms have been implemented and tested for the attitude controller including a proportional-derivative, a conventional sliding mode controller with boundary layer, and super-twisting higher order sliding mode controller. The results of the experiments show the versatility of the system being able to land in different kinds of terrains, keeping the attitude of the helicopter within safe margins at every moment, reducing the risk of entering dynamic rollover.

The comparison of the three controllers shows that the peak inclination of the helicopter in all the environments tested is within the range of 3-4°. However, the Super-Twisting controller has more difficulty to settle down the system, and oscillates around the target angles. PD and conventional SMC controller show similar performance, however, in the presence of sensor noise, the simplicity of the SMC can be an advantage as it doesn't need to calculate the derivative of the error.

From the mechanical design point of view, the leg springs loading/unloading add natural compliance to the legs and helps smoothing out the readings from the foot pressure sensors, improving the performance of the force controllers.

The results from the practical tests show a close correlation with the results from the software simulations.

7 Discussion

The work carried out during this project can be categorized into three different areas, the development of mathematical models for use in software simulations, the design of a control system, and the building and testing of a hardware prototype.

7.1.1 Modelling

The mathematical modelling has been an important part of this research work, and during this project, several methodologies and approaches have been applied to obtain the equations of motion of the legged robotic landing gear. Here, the two main methodologies, namely the Newton-Euler and Lagrange formulations, together with different approaches to robotic systems modelling, like full-rigid-body approach and the Centroidal dynamics and model decomposition, have been used.

The first model developed was a planar model of a landing gear with 2 legs. This model was obtained using a full-rigid-body model approach, applying Newton-Euler equations to every link of the system. The equations of motion obtained with this approach fully-describe the relationship of the torques and forces acting on the system and the corresponding motion. The interaction forces between the connected joints appear explicitly in the model equations and there are dependencies between the states of all generalised coordinates. This model is very accurate, but, because of its planar nature, it can only describe the motion of the system in a 2-dimensional space. The planar model was used for initial controller implementation where only the roll angle was controlled, and to asses landing on simple geometries like a single-axis slope or a terrain with a step.

The second model developed consisted in a spatial model of the landing gear with four legs. Here, the degrees of freedom of the main body are increased from 3 to 6, and 4 more DoF are added with the increase from 2 to 4 legs. Apart from that, the Newton-Euler equations are applied in its full spatial from. Thus, the complexity of the full-rigid-body approach in this case was too high, and a Centroidal dynamics approach has

been adopted, by decomposing the model into the main body model and four single-leg models.

In doing this, various assumptions and simplifications were made. The first one is that the inertia of the Centroidal body is calculated as the sum of the inertia of all the system links in its initial position. Thus, when the legs move from its initial position the real inertia of the system will be slightly different that the one used in the model. This assumption is acceptable if the weight of the main body is much higher than the mass of the legs, or if these move at a slow velocity and stay near the initial position, so the full-body inertia remains similar to the one in nominal joint position at any time.

The second assumption is a result of the model decomposition. The Centroidal model uses a full-body kinematic model to determine the position of the feet with respect to the CoM, and the points where the ground contact forces are applied. These allows to calculate the moments and forces produced in the CoM and the motion of the main body. The joint torques at each leg are calculated separately, using a single-leg model. This model consists in a fixed-frame 2 DoF planar robotic leg. Because of this model decomposition, the interaction forces between the main body and the legs at the hips are not explicitly present in the equations of motion, and the dynamic model of the single-leg ignores the effects of the whole system motion. These effects are not significant during flight phase as the system simply descends at a constant rate without relative motion between the base and the legs, and there are no external forces. During the landing phase, however, the motion of the CoM will affect the feet position, the foot-ground interaction forces, and therefore the joint torques. These ground reaction forces, which are directly related with the motion of the CoM, are introduced as external forces into the fixed-frame leg model, thus, the effects of the whole system motion is reflected in the joint torques calculation in the form of external forces.

However, legged robots are floating-base systems, which means that the reference frame of its legs is not attached to a fixed point, but to the main body, and its position and orientation change over time. In this system, if the helicopter body tilts in any direction, the position of the legs with respect to the world coordinate frame will also change. Then, the torques calculated with the leg dynamic model won’t correspond exactly with the torques needed in reality.

This is possible to correct in the roll direction, as the leg motion is contained within the YZ plane. Thus, if the main body rotates around its X axis, the reference frame of the leg model can be rotated, in order to calculate the joint torques with the right joint angles. If the body tilts on the pitch direction however, this correction is not possible as the single-leg model cannot move outside the YZ plane, and there will be a mismatch between the calculated torques, and the torques corresponding to that body position. Thus, this model is valid when the main body attitude is around the level position.

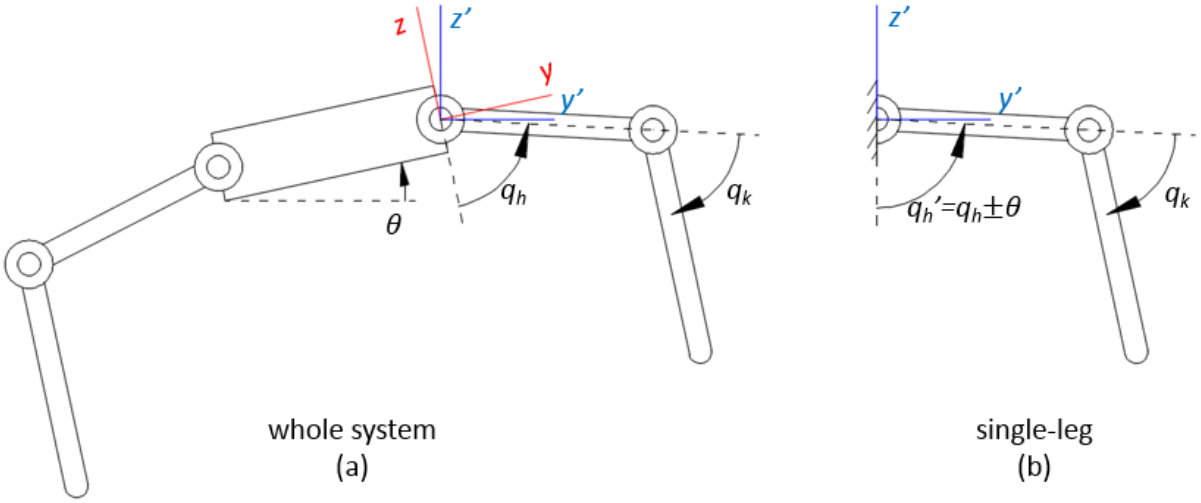

Figure 7-1 Change in the single-leg coordinate system to match body positon.

This model it's a representation of the motion of the system in a 3-dimensional space, and allowed to implement the attitude controllers to control the roll and pitch angles, and to perform landing simulations in more complex terrains, like 2-axis slopes or multi-level surfaces.

7.1.2 Control system

The developing of mathematical models allowed to use software simulations to design a control system, implement it and test it before building a hardware prototype.

The main concepts of the high-level control system were inspired by looking at balance controllers for legged robots and floating-base systems, were this systems control the position and attitude of its base by controlling the interaction forces between the legs and the ground.

The first approach to the controller design was to use the planar model to control just 1 DoF (the roll angle of the helicopter body) through the adjustment of the legs positions. When landing on uneven surfaces, if a leg makes ground contact before the

other, the system will start to tilt. At this moment, the controller starts to retract the leg in ground contact and extend the other as a function of the angle error. To control the leg motion in a more intuitive way, the leg kinematics were applied, so the position commands were given in terms of the coordinates of the end effector, instead of joint angles. This system, thus, uses a position-based control approach.

The next step was to use the 4-legged spatial model to apply control of the pitch and roll angle by combining the output of both controllers. When testing this system on landings on more complex terrains, like surfaces with 4 different ground levels, it was realized that the attitude controller alone was not enough to assure a stable landing. As there was not foot pressure feedback, the attitude controller couldn't guarantee that the landing would conclude with all four legs on the ground, and in some cases, the landing gear would end in a position of "*marginal-stability*" with only three supporting legs. This situation was avoided by adding a force controller that would measure the foot pressure and would retract the legs that were in ground contact and extend the ones that were not. The force controller ensures that all four legs make ground contact, giving more stability to the helicopter.

As only the legs that are in contact with the ground can generate ground forces to control the base attitude, it is important to land all the legs with the minimum time possible without generating disturbing impact forces. Thus, the force controller, also improves the performance of the attitude controller by reducing the time until all four legs have landed. The combination of the force and attitude controllers reduce both, the landing time and the peak body inclinations during the landing process, and because it's a position-based controller, it can be implemented more easily into hardware robots.

At the low-level, conventional PID controllers have been used for simulations instead of model-based controllers. Model-based techniques allow to design controllers with lower gains, adding compliance to the joints, without compromising tracking performance. However, the goal of the control system is not to achieve an accurate trajectory tracking for the motion of each individual joint, but rather to maintain the body attitude at a levelled position, with the coordination of all joints' action. Thus, the position tracking performance at the individual joint level is not crucial for the overall system performance. On the other hand, model-based techniques require computing complex dynamics equations. They are also more difficult to implement in hardware

systems as they increase the need of computational power and require motors with torque-control capability.

7.1.3 Hardware prototype

The hardware prototype was built to test and validate the results obtained with software simulations. It was designed using off-the-shelf and 3D printed parts to integrate all the components needed including servo motors, on-board processor and sensors. After mechanical and electrical integration, the code was developed to communicate with system components and to implement the control system using C programming language. In this part of the project, only the high-level control is used, as the servo motors already have built-in controllers.

The laboratory experiments confirmed a good performance and effective landing control of the proposed control architecture. It is easy to implement, using low-cost actuators and sensors, and with low computational power. Due to hardware limitations, the control frequency was reduced from 1 kHz in the simulations to 20 Hz in the prototype, without affecting significantly to the overall performance. The tests also showed that the high-level force and attitude controllers can effectively be combined with different low-level controllers, like the PID joint controllers in the simulations or the built-in motor controllers in the practical tests.

In quadruped robots, compliance is a highly desired property in order to avoid high impact forces that can disturb the equilibrium of the robot [80] [81]. Compliant behaviour can be achieved either passively, by adding elastic elements in the joints or leg structure, or actively, through the controller design like using model-based or force control techniques. In the landing gear system, the force controller adds active compliance to the leg by controlling the force at the foot, but the motors cannot react immediately to an external impact force. The addition of the leg springs, provides the system with passive compliance that can react faster to external disturbances.

The FSR sensors offer obvious advantages like their compact dimensions and they are easy to integrate into the system, however they have lower accuracy than other types of sensors [82]. The position of the spring between the foot and the force sensor, also improves the performance of the force controllers as it acts as a natural filter, smoothing out the readings from the sensors. Despite the relatively low accuracy of the FSR, the system doesn’t rely too much on the sensitivity of the sensors. This is

done by filtering the signal and preventing the leg to switch between extension and retraction mode. Once a leg has made ground contact, the output of the force controller can only make the leg retract or to stop moving, but not extend.

7.1.4 Maximum landing slope

The two main factors that are considered to limit the maximum landing slope are the geometry of the legs and the friction with the terrain. The geometry of the legs is determined by the length of the leg segments and the lateral distance between the feet. By increasing the length of the leg segments, higher slopes can be overcome but additional weight is added to the structure. For the same length of the leg segments, higher available foot stroke can be achieved by reducing the lateral foot distance, and thus higher achievable slopes. However, this reduces the support area between the feet, and increases the risk of lateral tilt. The foot design is another important factor, as it needs to provide sufficient friction to avoid slippage on large slopes. During the laboratory tests, the feet were covered with a rubber cap that provided sufficient friction on landing tests at slopes higher than 20°. However, for different kind of terrains different foot materials and designs can be used to optimise the friction between surfaces.

The optimal system design needs to find a balance between all these parameters. The landing gear has to be able to land on sufficiently large slopes to meet the system requirements, but without adding to much weight to the system and without compromising stability and safety.

8 Conclusions and Future Works

8.1 Conclusions

Although rotorcrafts are highly manoeuvrable machines, they are limited when it comes to landing on unprepared sites on rough terrains. This limitation is mainly due to the use of conventional landing gear like skids or wheels.

Adaptive landing gear for helicopters it's a new field, and only very recently a few prototypes have been designed and developed. These systems offer many potential benefits including extended operational range of current vehicles and increased safety on rough terrain landings. Adaptive landing gear can be used to assist pilots to land on different situations like search-and-rescue missions, landing on ships or sea platforms, or in mountain environments. Additionally, the growth in the use of unmanned vehicles, like UAV or drones, increases the demand for automated landing systems. Its main barriers for commercial applications are the increase in the weight of the system and its complexity compared to conventional landing gear systems.

Other proposed systems found in literature review are in early development stage or don't offer much technical information regarding the control system. Only the Darpa adaptive landing gear has several publications. However, these published works are all based on the results of software simulations, not on hardware implementation results. Different controllers are proposed for this system involving some kind of torque-control at the joint level.

In this thesis, a new design and control system for a robotic landing gear has been developed. The solution presented consisted in a legged system that can sense and adapt the legs' positions to the ground conditions, providing a safe landing as the attitude of the vehicle is maintained in a level position during the whole operation. Such a system increases the landing capacities of rotorcraft vehicles allowing them to land on higher slopes and uneven terrains with obstacles or steps.

The control solution proposed in this thesis offers the advantage of being position-controlled, so it is easier to implement it in hardware systems using low-cost components. However, position-controlled systems are usually stiffer. The solution proposed incorporates foot pressure sensors, which fulfil a double purpose. First, they are used to detect ground contact and to activate the landing control. Second, they are used to add compliant behaviour to the system by regulating the contact force at each foot. The leg design also incorporates a compliant element in the form of a spring to provide instant compliance, protect the force sensors and smooth the foot pressure reading from the sensors.

The force and attitude controller complement each other by producing the necessary motion to correct the rotorcraft inclination, adapt the feet to the terrain, and reduce the time until all four legs have landed. Thus the system benefits from the simplicity and easy implementation of a position-controlled system that regulates the attitude of the helicopter body and a force controller and leg design that incorporate active and passive compliance to the system for better handling of the ground reaction forces.

This thesis also presented the development of mathematical models for testing the system in software simulations using Matlab/Simulnk. These models include the dynamic model of the multi-body system, the ground contact model, and the control algorithms. Extensive simulations have been carried out and presented in this thesis to validate the mathematical models and to develop control algorithms.

The prototype system designed in this project served as a platform to test the control algorithms previously developed in the simulations. The prototype design included a mechanical and electrical design and manufacture, the sensor placement and integration and the development of code for the digital implementation of the controllers. The tests results presented in this thesis also provide evidence of the performance of the system by measuring important system parameters like the feet pressure and body attitude. Slope landings of 20° and landings with obstacles up to 12 cm were performed with peak body inclinations within 3-4°.

The main objective of this project was to contribute on the development of a control architecture and system design to adapt the position of the legs to the ground

conditions, in a manner that allows a safe and stable landing on uneven terrains. In summary, the research contributions of this thesis includes:

- The development of mathematical models for its implementation and testing in software simulations in Matlab/Simulink
- The development of a control strategy based on position and force control to adapt the legs position to the terrain and to add compliance to the system
- The design, building and testing of a physical prototype to validate the findings and the performance of the control algorithms.
- The development of the Arduino code to implement the control strategy in the prototype robot.
- The publications mentioned in section 1.5.

8.2 Future work

During this project, a prototype of the robotic landing gear has been designed and built. This first version has been tested in laboratory experiments, collecting data and improving the design. However, there are many areas in which the system can be developed further.

- The software simulations and laboratory pulley tests have allowed to implement several control techniques on the landing gear, to collect data in order to analyse the system performance, and improve the sensor integration and leg design. During these tests, the system was plugged to the mains and to a lab PC, although it is equipped with on-board batteries and wireless system to transmit data. The next logic step would be to test the system in a real outdoors flight. The most crucial factor here is safety, as the system needs to be able to land even in the event of a power failure on the landing gear. There are examples of experimental setups to do so in a small-medium size model helicopter using an auxiliary structure for safety like in [22] (see Figure 2-9) or [83]. Flight tests can provide important information on how the control system and the sensors perform in the presence of vibration introduced by the rotor of the helicopter.
- On the practical side, a necessary feature that needs to be implemented for further development of the project, is a safety mechanism that allows the

helicopter to land safely even in the case of a power failure in the robotic landing gear. This can be implemented by adding a mechanical blocking mechanism of brake system to the legs design. This brake system would also allow to cut the power of the servo motors, and maintain the position after landing finishes.

- Very recently, several systems have appeared with a leg design that uses a four-bar linkage [22] [21]. By modifying the leg design to mechanically constrain the motion in the vertical direction, the number of motors in the system could be reduced to one per leg, potentially reducing the weight of the system, its, complexity, and the probability of electrical failure.
- During a flight, it is not always easy for the pilot to assess the inclination or possible obstacles on the terrain. As an additional feature, a slope detection system could be implemented using range sensors, without increasing significantly the complexity and cost of the system. By using an array of 4 distance sensors, one on each leg, instead of one, the distance to the ground and the slope can be known before landing. This would allow to pre-adjust the legs before landing, or to inform the pilot if the slope is too high. The feet can also be replaced by wheels to increase the ground handling capability of the system.
- On the modelling side, the effort was placed on developing the mathematical model of the legged system, while the helicopter model was reduced to a thrust force model to control the descent rate. An area of improvement can be the development of more realistic helicopter models that better describe the dynamics and behaviour of the system, and combine them with the landing gear model. Although some examples were found in the literature review like in [84] [83], these area was left out of the scope of this project.
- On the control side, it would be interesting to try a different actuator technology on future versions of the prototype. This would allow to implement different control approaches like model-based control or virtual model control, and to compare the results with the actual control system.

Appendix A Planar Model

As described in section 3.1., the full-body planar model of the system is obtained by applying Newton-Euler equations to each individual link. There are three equations of motion for each of the five links: summation of horizontal forces, summation of vertical forces, and summation of torques about the centre of mass. By eliminating the internal reaction forces the system can be reduced to seven equations.

Equations 3.3-3.5, 3.12-3.17 and 3.22-3.27, describe the dynamics of each link on each degree of freedom. Equations 3.8-3.11, 3.18-3.21, and 3.28-3.31, describe the kinematic relations between the accelerations of the links and the generalised coordinates. By introducing the kinematic equations into the dynamic equations, the equations of motion can be expressed in terms of the generalised coordinates only. This allows to calculate the joint torques if the external forces and robot's kinematics are known.

The equations are expressed in matrix form as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} + \mathbf{J}^{\mathbf{T}}(\mathbf{q})\mathbf{f} \tag{A.1}$$

where q is the generalised coordinates vector, $\mathbf{q} = [y_B, z_B, \theta, q_1, q_2, q_3, q_4]^T$, **M** is the inertia matrix, **C** is the matrix of centrifugal and Coriolis terms, **G** is the vector of gravity terms, $\boldsymbol{\tau} = [0, 0, 0, \tau_1, \tau_2, \tau_3, \tau_4]^T$ is the vector of joint actuator torques, $\mathbf{J^T}$ is the transpose of the Jacobian matrix, and **f** is the vector of external forces. The sub-indices H_r, K_r, H_l, and K_l, used in the equations in chapter 3 have been substituted in the appendix for 1, 2, 3, and 4 respectively for more clarity of the equations.

The fully extended equations are given below:

Equation 1: angular acceleration of the main body

$$
\begin{aligned}
&\left[2(m_U+m_L)D_z c\theta - m l_{C_U}(s\beta_1+s\beta_3) - m_L l_{C_L}(s\beta_2+s\beta_4)\right]\ddot{y}_B \\
&\quad + \left[2(m_U+m_L)D_z s\theta + m l_{C_U}(c\beta_1+c\beta_3) + m_L l_{C_L}(c\beta_2+c\beta_4)\right]\ddot{z}_B \\
&\quad + \Big[I_B + 2I_U + 2I_L + (m_U+m_L)\left(2D_z^2+2D_y^2\right) \\
&\quad + 2m_L\left(l_U^2 + l_{C_L}^2 + l_U l_{C_L}(cq_2+cq_4)\right) + 2m_U l_{C_U}^2 \\
&\quad + m_L l_L\left(D_z\big(s(\theta-\beta_2)+s(\theta-\beta_4)\big) + D_y\big(c(\theta-\beta_2)-c(\theta-\beta_4)\big)\right) \\
&\quad + m l_u\left(D_y(cq_1-cq_3) - D_z(sq_1+sq_3)\right)\Big]\ddot{\theta} \\
&\quad + \left[I_U + I_L + m_U l_{C_U}^2 + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L} cq_2\right)\right. \\
&\quad \left. + m l_{C_U}(R_1 c\beta_1 - R_2 s\beta_1) + m_L l_{C_L}(R_1 c\beta_2 - R_2 s\beta_2)\right]\ddot{q}_1 \\
&\quad + \left[I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L} cq_2\right) + m_L l_{C_L}(R_1 c\beta_2 - R_2 s\beta_2)\right]\ddot{q}_2 \\
&\quad + \left[I_U + I_L + m_U l_{C_U}^2 + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L} cq_4\right)\right. \\
&\quad \left. + m l_{C_U}(R_3 c\beta_3 - R_4 s\beta_3) + m_L l_{C_L}(R_3 c\beta_4 - R_4 s\beta_4)\right]\ddot{q}_3 \\
&\quad + \left[I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L} cq_4\right) + m_L l_{C_L}(R_3 c\beta_4 - R_4 s\beta_4)\right]\ddot{q}_4 \\
&\quad - [m l_u(R_1 s\beta_1 + R_2 c\beta_1)\dot{q}_1 + m_L l_L(R_1 s\beta_2 + R_2 c\beta_2)(\dot{q}_1+\dot{q}_2) \\
&\quad + m_L l_U l_L s\beta_2 \dot{q}_2 + m l_u(R_3 s\beta_3 + R_4 c\beta_3)\dot{q}_3 \\
&\quad + m_L l_L(R_3 s\beta_4 + R_4 c\beta_4)(\dot{q}_3+\dot{q}_4) + m_L l_U l_L s\beta_4 \dot{q}_4]\dot{\theta} \\
&\quad - [m l_{C_U}(R_1 s\beta_1 + R_2 c\beta_1) + m_L l_{C_L}(R_1 s\beta_2 + R_2 c\beta_2)]\dot{q}_1^2 \\
&\quad - [m_L l_{C_L}(l_u sq_2 + R_1 s\beta_2 + R_2 c\beta_2)](2\dot{q}_1+\dot{q}_2)\dot{q}_2 \\
&\quad - [m l_{C_U}(R_3 s\beta_3 + R_4 c\beta_3) + m_L l_{C_L}(R_3 s\beta_4 + R_4 c\beta_4)]\dot{q}_3^2 \\
&\quad - [m_L l_{C_L}(l_u sq_4 + R_3 s\beta_4 + R_4 c\beta_4)](2\dot{q}_3+\dot{q}_4)\dot{q}_4 + 2(m_U+m_L)g D_z s\theta \\
&\quad + m g l_{C_U}(cq_1+cq_3) + m_L g l_{C_L}(cq_2+cq_4) \\
&\quad = (D_z c\theta - D_y s\theta - l_U s\beta_1 - l_L s\beta_2)F_{x_R} + (D_z c\theta + D_y s\theta - l_U s\beta_3 \\
&\quad - l_L s\beta_4)F_{x_L} + (D_z s\theta + D_y c\theta + l_U c\beta_1 + l_L c\beta_2)F_{y_R} + (D_z s\theta - D_y c\theta \\
&\quad + l_U c\beta_3 + l_L c\beta_4)F_{y_L}
\end{aligned}
\tag{A.2}
$$

where

$$R_1 = D_z s\theta + D_y c\theta \;\; ; \;\; R_2 = D_z c\theta - D_y s\theta \;\; ; \;\; R_3 = D_z s\theta - D_y c\theta \;\; ; \;\; R_4 = D_z c\theta + D_y s\theta$$

Equation 2: horizontal linear acceleration of the main body

$$
\begin{aligned}
&[m_B + 2m_U + 2m_L]\ddot{y}_B \\
&\quad + \left[2(m_U + m_L)D_z c\theta + ml_{C_U}(s\beta_1 + s\beta_3) + m_L l_{C_L}(s\beta_2 + s\beta_4)\right]\ddot{\theta} \\
&\quad - \left[ml_{C_U}s\beta_1 + m_L l_{C_L}s\beta_2\right]\ddot{q}_1 - m_L l_{C_L}s\beta_2\ddot{q}_2 - \left[ml_{C_U}s\beta_3 + m_L l_{C_L}s\beta_4\right]\ddot{q}_3 \\
&\quad - m_L l_{C_L}s\beta_4\ddot{q}_4 \\
&\quad + \left[2(m_U + m_L)D_z s\alpha\dot{\theta} - m_L l_L c\beta_2\dot{q}_2\right. \\
&\quad - \left(ml_{C_U}c\beta_1 + m_L l_{C_L}c\beta_2\right)\left(\dot{\theta} + 2\dot{q}_1\right) - m_L l_L c\beta_4\dot{q}_4 \\
&\quad \left. - \left(ml_{C_U}c\beta_3 + m_L l_{C_L}c\beta_4\right)\left(\dot{\theta} + 2\dot{q}_3\right)\right]\dot{\theta} - \left[ml_{C_U}c\beta_1 + m_L l_{C_L}c\beta_2\right]\dot{q}_1^2 \\
&\quad - m_L l_{C_L}c\beta_2(\dot{q}_2 + 2\dot{q}_1)\dot{q}_2 - \left[ml_{C_U}c\beta_3 + m_L l_{C_L}c\beta_4\right]\dot{q}_3^2 \\
&\quad - m_L l_{C_L}c\beta_4(\dot{q}_4 + 2\dot{q}_3)\dot{q}_4 = F_{x_R} + F_{x_L}
\end{aligned}
\tag{A.3}
$$

Equation 3: vertical linear acceleration of the main body

$$
\begin{aligned}
&[m_B + 2m_U + 2m_L]\ddot{z}_B \\
&\quad + \left[2(m_U + m_L)D_z s\theta + ml_{C_U}(c\beta_1 + c\beta_3) + m_L l_{C_L}(c\beta_2 + c\beta_4)\right]\ddot{\theta} \\
&\quad + \left[ml_{C_U}c\beta_1 + m_L l_{C_L}c\beta_2\right]\ddot{q}_1 + m_L l_{C_L}c\beta_2\ddot{q}_2 + \left[ml_{C_U}c\beta_3 + m_L l_{C_L}c\beta_4\right]\ddot{q}_3 \\
&\quad + m_L l_{C_L}c\beta_4\ddot{q}_4 \\
&\quad + \left[2(m_U + m_L)D_z c\theta\dot{\theta} - m_L l_L s\beta_2\dot{q}_2\right. \\
&\quad - \left(ml_{C_U}s\beta_1 + m_L l_{C_L}s\beta_2\right)\left(\dot{\theta} + 2\dot{q}_1\right) - m_L l_L s\beta_4\dot{q}_4 \\
&\quad \left. - \left(ml_{C_U}s\beta_3 + m_L l_{C_L}s\beta_4\right)\left(\dot{\theta} + 2\dot{q}_3\right)\right]\dot{\theta} - \left[ml_{C_U}s\beta_1 + m_L l_{C_L}s\beta_2\right]\dot{q}_1^2 \\
&\quad - m_L l_{C_L}s\beta_2(\dot{q}_2 + 2\dot{q}_1)\dot{q}_2 - \left[ml_{C_U}s\beta_3 + m_L l_{C_L}s\beta_4\right]\dot{q}_3^2 - m_L l_{C_L}s\beta_4(\dot{q}_4 \\
&\quad + 2\dot{q}_3)\dot{q}_4 + (m_B + 2m_U + 2m_L)g = F_{y_R} + F_{y_L} + F_{th}
\end{aligned}
\tag{A.4}
$$

Equation 4: right hip joint angle

$$
\begin{aligned}
&-\left[ml_{C_U}s\beta_1 + m_L l_{C_L}s\beta_2\right]\ddot{y}_B + \left[ml_{C_U}c\beta_1 + m_L l_{C_L}c\beta_2\right]\ddot{z}_B \\
&\quad + \left[I_U + I_L + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L}cq_2\right) + m_U l_{C_U}^2\right. \\
&\quad + ml_{C_U}(D_y cq_1 - D_z sq_1) + m_L l_{C_L}(D_z s(\theta-\beta_2) + D_y c(\theta-\beta_2))\Big]\ddot{\theta} \\
&\quad + \left[I_U + I_L + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L}cq_2\right) + m_U l_{C_U}^2\right]\ddot{q}_1 \\
&\quad + \left[I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L}cq_2\right)\right]\ddot{q}_2 \\
&\quad + \left[\left[m_L l_{C_L}(D_z c(\theta-\beta_2) - D_y s(\theta-\beta_2)) + ml_{C_U}(D_z cq_1 + D_y sq_1)\right]\dot{\theta}\right. \\
&\quad + m_L l_U l_L sq_2\dot{q}_2\Big]\dot{\theta} - m_L l_U l_{C_L} sq_2(2\dot{q}_1 + \dot{q}_2)\dot{q}_2 + mgl_U c\beta_1 \\
&\quad + m_L g l_{C_L} c\beta_2 = \tau_1 - (l_U s\beta_1 + l_L s\beta_2)F_{x_R} + (l_U c\beta_1 + l_L c\beta_2)F_{y_R}
\end{aligned}
\tag{A.5}
$$

Equation 5: right knee joint angle

$$
\begin{aligned}
&-m_L l_{C_L} s\beta_2\ddot{y}_B + m_L l_{C_L} c\beta_2\ddot{z}_B \\
&\quad + \left[I_L + m_L l_{C_L}(D_z s(\theta-\beta_2) + D_y c(\theta-\beta_2) + l_U cq_2 + l_{C_L})\right]\ddot{\theta} \\
&\quad + \left[I_L + m_L\left(l_{C_L}^2 + l_{C_L} l_U cq_2\right)\right]\ddot{q}_1 + \left[I_L + m_L l_{C_L}^2\right]\ddot{q}_2 \\
&\quad + \left[\left[m_L l_{C_L}(D_z c(\theta-\beta_2) - D_y s(\theta-\beta_2))\right]\dot{\theta} + m_L l_{C_L} l_L sq_2\left(\dot{\theta} + 2\dot{q}_1\right)\right]\dot{\theta} \\
&\quad + m_L l_U l_{C_L} sq_2\dot{q}_1^2 + m_L g l_{C_L} c\beta_2 = \tau_2 - l_L s\beta_2 F_{x_R} + l_L c\beta_2 F_{y_R}
\end{aligned}
\tag{A.6}
$$

Equation 6: left hip joint angle

$$
\begin{aligned}
&-\left[ml_{C_U}s\beta_3 + m_L l_{C_L}s\beta_4\right]\ddot{y}_B + \left[ml_{C_U}c\beta_3 + m_L l_{C_L}c\beta_4\right]\ddot{z}_B \\
&\quad + \left[I_U + I_L + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L}cq_4\right) + m_U l_{C_U}^2\right. \\
&\quad + ml_{C_U}(D_y cq_3 - D_z sq_3) + m_L l_{C_L}(D_z s(\theta-\beta_4) - D_y c(\theta-\beta_4))\Big]\ddot{\theta} \\
&\quad + \left[I_U + I_L + m_L\left(l_U^2 + l_{C_L}^2 + 2l_U l_{C_L}cq_4\right) + 2m_U l_{C_U}^2\right]\ddot{q}_3 \\
&\quad + \left[I_L + m_L\left(l_{C_L}^2 + l_U l_{C_L}cq_4\right)\right]\ddot{q}_4 \\
&\quad + \left[\left[m_L l_{C_L}(D_z c(\theta-\beta_4) + D_y s(\theta-\beta_4)) + ml_{C_U}(D_z cq_3 - D_y sq_3)\right]\dot{\theta}\right. \\
&\quad + m_L l_U l_L s\theta_4\dot{q}_4\Big]\dot{\theta} - m_L l_U l_{C_L} sq_4(2\dot{q}_3 + \dot{q}_4)\dot{q}_4 + mgl_U c\beta_3 \\
&\quad + m_L g l_{C_L} c\beta_4 = \tau_3 - (l_U s\beta_3 + l_L s\beta_4)F_{x_R} + (l_U c\beta_3 + l_L c\beta_4)F_{y_R}
\end{aligned}
\tag{A.7}
$$

Equation 7: left knee joint angle

$$
\begin{aligned}
&-m_L l_{C_L} s\beta_4 \ddot{y}_B + m_L l_{C_L} c\beta_4 \ddot{z}_B \\
&\quad + \left[I_L + m_L l_{C_L}(D_z s(\theta - \beta_4) - D_y c(\theta - \beta_4) + l_U c q_4 + l_{C_L})\right]\ddot{\theta} \\
&\quad + \left[I_L + m_L(l_{C_L}^2 + l_{C_L} l_U c q_4)\right]\ddot{q}_3 + \left[I_L + m_L l_{C_L}^2\right]\ddot{q}_4 \\
&\quad + \left[\left[m_L l_{C_L}(D_z c(\theta - \beta_4) + D_y s(\theta - \beta_4))\right]\dot{\theta} + m_L l_{C_L} l_L s q_4 \left(\dot{\theta} + 2\dot{q}_3\right)\right]\dot{\theta} \\
&\quad + m_L l_U l_{C_L} s q_4 \dot{q}_3^2 + m_L g l_{C_L} c\beta_4 = \tau_4 - l_L s\beta_4 F_{x_R} + l_L c\beta_4 F_{y_R}
\end{aligned}
\tag{A.8}
$$

Appendix B Centroidal Momentum Matrix

This Appendix includes the Matlab script used to calculate the Centroidal Momentum Matrix that maps the inertia of each individual link into the system's CoM. First, the location of the CoM in the z-axis is computed. Then, the inertia tensor of the whole system is calculated by aligning the principal moments of inertia of each link with the axes of the main body fixed frame, and then applying the parallel axis theorem to express the moments of inertia around the CoM.

```
%% PRELIMINARIES
%This section defines the main system parameters
% Main body dimensions (m)
Dx=100/1000; %x dimension
Dy=50/1000;  %y dimension
Dz=100/1000; %z dimension!
% Links length (m)
l_u=0.0935;
l_l=0.1045;
% Masses (kg)
mb=2.5;     %main body (base)
mu=0.1;     %upper link
ml=0.15;    %lower link
thick=0.03; %Links thickness
%Main rotor parameters
%https://www.align-trex.co.uk/425-carbon-fiber-blades-hd420f.html
m_mr=0.12; %mass of rotor blades (120g/set)
r_mr=0.489; %main rotor radius (diameter of Trex500L=978mm)
%Total system mass
m=mb+4*mu+4*ml+m_mr;
%Joint angles for different legs positions
%Landing position
th_u_L=-pi;
th_l_L=-pi/2;
th_u_R=0;
th_l_R=-pi/2;

%Left retracted - Right extended
% th_u_L=-3.979;
% th_l_L=-1.867;
% th_u_R=-0.75;
% th_l_R=-1.326;
%Right retracted - Left extended
% th_u_L=-2.39;
% th_l_L=-1.815;
% th_u_R=0.837;
% th_l_R=-1.274;
```

```
%%------------------------ CoM LOCATION-------------------------------------
%This section computes the location of the system CoM in the z-direction
taking as a reference the CoM of the base link

z_b=0;% z-ccordinate of the base link
z_mr=Dz;% z-ccordinate of the main rotor
z_u_R=-Dz+(l_u/2)*sin(th_u_R); % z-coordinate of the upper link(Right Side)
z_u_L=-Dz+(l_u/2)*sin(th_u_L); % z-coordinate of the upper link(Left Side)
z_l_L=-Dz+l_u*sin(th_u_L)+(l_l/2)*sin(th_l_L); % z-coordinate of the lower
link(Left Side)
z_l_R=-Dz+l_u*sin(th_u_R)+(l_l/2)*sin(th_l_R);  % z-coordinate of the lower
link(Right Side)

%Distance from the base link CoM to the system's CoM
d=-(mb*z_b+m_mr*z_mr+mu*(2*z_u_R+2*z_u_L)+ml*(2*z_l_L+2*z_l_R))/m;

%% ----------------INERTIA MOMENTS OF THE MAIN BODY-----------------------
%Inertia wrt principal axes
Ixx_b=mb*((2*Dy)^2 + (2*Dz)^2)/12;
Iyy_b=mb*((2*Dz)^2 + (2*Dx)^2)/12;
Izz_b=mb*((2*Dy)^2 + (2*Dx)^2)/12;
% Distance base-CoM
xg_b=0;
yg_b=0;
zg_b=d;
% Moments of Inertia referred to the base CoM
Ixxg_b=Ixx_b + mb*(xg_b^2 + zg_b^2);
Iyyg_b=Iyy_b + mb*(zg_b^2 + yg_b^2);
Izzg_b=Izz_b + mb*(xg_b^2 + yg_b^2);
% Inertia Tensor
I_b=[Ixxg_b 0 0;0 Iyyg_b 0;0 0 Izzg_b];

%% ----------------INERTIA MOMENTS OF THE MAIN ROTOR----------------------
%Inertia wrt principal axes
Ixx_mr=Izz_mr/2;
Iyy_mr=Izz_mr/2;
Izz_mr=(m_mr*r_mr^2)/3;
% Distance mr-CoM
xg_mr=0;
yg_mr=0;
zg_mr=Dz+d;
% Moments of Inertia referred to the base CoM
Ixxg_mr=Ixx_mr + m_mr*(yg_mr^2 + zg_mr^2);
Iyyg_mr=Iyy_mr + m_mr*(xg_mr^2 + zg_mr^2);
Izzg_mr=Izz_mr + m_mr*(xg_mr^2 + yg_mr^2);
% Inertia Tensor
I_mr=[Ixxg_mr 0 0;0 Iyyg_mr 0;0 0 Izzg_mr];

%%-------------INERTIA MOMENTS OF THE UPPER LEG SEGMENTS-----------------
%Inertia wrt principal axes
Ixx_u=mu*((thick)^2 + (l_u)^2)/12;
Iyy_u=mu*((thick)^2 + (thick)^2)/12;
Izz_u=mu*((thick)^2 + (l_u)^2)/12;

% Distance u_R-CoM (Right Side)
xg_u_RF=Dx; % Right front leg
xg_u_RB=-Dx;% Right back leg
yg_u_R=Dy+(l_u/2)*cos(th_u_R);
zg_u_R=-(Dz-d)+(l_u/2)*sin(th_u_R);
```

```
% Distance u_L-CoM (Left Side)
xg_u_LF=Dx;
xg_u_LB=-Dx;
yg_u_L=Dy+(l_u/2)*cos(th_u_L);
zg_u_L=-(Dz-d)+(l_u/2)*sin(th_u_L);

% Rotated Moments of Inertia (Right Side)
Ixx_Ru=Ixx_u;
Iyy_Ru=Iyy_u*cos(th_u_R)^2+Izz_u*sin(th_u_R)^2;
Izz_Ru=Iyy_u*sin(th_u_R)^2+Izz_u*cos(th_u_R)^2;
Iyz_Ru=(Iyy_u-Izz_u)*cos(th_u_R)*sin(th_u_R);
Izy_Ru=Iyz_Ru;
% Rotated Moments of Inertia (Left Side)
Ixx_Lu=Ixx_u;
Iyy_Lu=Iyy_u*cos(th_u_L)^2+Izz_u*sin(th_u_L)^2;
Izz_Lu=Iyy_u*sin(th_u_L)^2+Izz_u*cos(th_u_L)^2;
Iyz_Lu=(Iyy_u-Izz_u)*cos(th_u_L)*sin(th_u_L);
Izy_Lu=Iyz_Lu;

% Moments of Inertia referred to the CoM (Right Front Leg)
Ixxg_RFu=Ixx_Ru + mu*(yg_u_R^2 + zg_u_R^2);
Iyyg_RFu=Iyy_Ru + mu*(xg_u_RF^2 + zg_u_R^2);
Izzg_RFu=Izz_Ru + mu*(xg_u_RF^2 + yg_u_R^2);
Ixyg_RFu=-mu*xg_u_RF*yg_u_R;
Iyxg_RFu=Ixyg_RFu;
Ixzg_RFu=-mu*xg_u_RF*zg_u_R;
Izxg_RFu=Ixzg_RFu;
Iyzg_RFu=Iyz_Ru- mu*yg_u_R*zg_u_R;
Izyg_RFu=Iyzg_RFu;
% Inertia Tensor
I_RFu=[Ixxg_RFu Ixyg_RFu Ixzg_RFu;Iyxg_RFu Iyyg_RFu Iyzg_RFu;Izxg_RFu
Izyg_RFu Izzg_RFu];

% Moments of Inertia referred to the CoM (Right Back Leg)
Ixxg_RBu=Ixx_Ru + mu*(yg_u_R^2 + zg_u_R^2);
Iyyg_RBu=Iyy_Ru + mu*(xg_u_RB^2 + zg_u_R^2);
Izzg_RBu=Izz_Ru + mu*(xg_u_RB^2 + yg_u_R^2);
Ixyg_RBu=-mu*xg_u_RB*yg_u_R;
Iyxg_RBu=Ixyg_RBu;
Ixzg_RBu=-mu*xg_u_RB*zg_u_R;
Izxg_RBu=Ixzg_RBu;
Iyzg_RBu=Iyz_Ru- mu*yg_u_R*zg_u_R;
Izyg_RBu=Iyzg_RBu;
% Inertia Tensor
I_RBu=[Ixxg_RBu Ixyg_RBu Ixzg_RBu;Iyxg_RBu Iyyg_RBu Iyzg_RBu;Izxg_RBu
Izyg_RBu Izzg_RBu];

% Moments of Inertia referred to the base CoM (Left Front Leg)
Ixxg_LFu=Ixx_Lu + mu*(yg_u_L^2 + zg_u_L^2);
Iyyg_LFu=Iyy_Lu + mu*(xg_u_LF^2 + zg_u_L^2);
Izzg_LFu=Izz_Lu + mu*(xg_u_LF^2 + yg_u_L^2);
Ixyg_LFu=-mu*xg_u_LF*yg_u_L;
Iyxg_LFu=Ixyg_LFu;
Ixzg_LFu=-mu*xg_u_LF*zg_u_L;
Izxg_LFu=Ixzg_LFu;
Iyzg_LFu=Iyz_Lu- mu*yg_u_L*zg_u_L;
Izyg_LFu=Iyzg_LFu;
% Inertia Tensor
I_LFu=[Ixxg_LFu Ixyg_LFu Ixzg_LFu;Iyxg_LFu Iyyg_LFu Iyzg_LFu;Izxg_LFu
Izyg_LFu Izzg_LFu];
```

```
% Moments of Inertia referred to the base CoM (Left Back Leg)
Ixxg_LBu=Ixx_Lu + mu*(yg_u_L^2 + zg_u_L^2);
Iyyg_LBu=Iyy_Lu + mu*(xg_u_LB^2 + zg_u_L^2);
Izzg_LBu=Izz_Lu + mu*(xg_u_LB^2 + yg_u_L^2);
Ixyg_LBu=-mu*xg_u_LB*yg_u_L;
Iyxg_LBu=Ixyg_LBu;
Ixzg_LBu=-mu*xg_u_LB*zg_u_L;
Izxg_LBu=Ixzg_LBu;
Iyzg_LBu=Iyz_Lu- mu*yg_u_L*zg_u_L;
Izyg_LBu=Iyzg_LBu;
% Inertia Tensor
I_LBu=[Ixxg_LBu Ixyg_LBu Ixzg_LBu;Iyxg_LBu Iyyg_LBu Iyzg_LBu;Izxg_LBu
Izyg_LBu Izzg_LBu];

%% ------------INERTIA MOMENTS OF THE LOWER LEG SEGMENTS-------------------
%Inertia wrt principal axes
Ixx_l=ml*((thick)^2 + (l_l)^2)/12;
Iyy_l=ml*((thick)^2 + (thick)^2)/12;
Izz_l=ml*((thick)^2 + (l_l)^2)/12;

% Distance u_R-CoM (Right Side)
xg_l_RF=Dx;
xg_l_RB=-Dx;
yg_l_R=Dy+(l_l/2)*cos(th_l_R);
zg_l_R=-(Dz-d)+(l_l/2)*sin(th_l_R);
% Distance l_L-CoM (Left Side)
xg_l_LF=Dx;
xg_l_LB=-Dx;
yg_l_L=Dy+(l_l/2)*cos(th_l_L);
zg_l_L=-(Dz-d)+(l_l/2)*sin(th_l_L);

% Rotated Moments of Inertia (Right Side)
Ixx_Rl=Ixx_l;
Iyy_Rl=Iyy_l*cos(th_l_R)^2+Izz_l*sin(th_l_R)^2;
Izz_Rl=Iyy_l*sin(th_l_R)^2+Izz_l*cos(th_l_R)^2;
Iyz_Rl=(Iyy_l-Izz_l)*cos(th_l_R)*sin(th_l_R);
Izy_Rl=Iyz_Rl;
% Rotated Moments of Inertia (Left Side)
Ixx_Ll=Ixx_l;
Iyy_Ll=Iyy_l*cos(th_l_L)^2+Izz_l*sin(th_l_L)^2;
Izz_Ll=Iyy_l*sin(th_l_L)^2+Izz_l*cos(th_l_L)^2;
Iyz_Ll=(Iyy_l-Izz_l)*cos(th_l_L)*sin(th_l_L);
Izy_Ll=Iyz_Ll;


% Moments of Inertia referred to the base CoM (Right Front Leg)
Ixxg_RFl=Ixx_Rl + ml*(yg_l_R^2 + zg_l_R^2);
Iyyg_RFl=Iyy_Rl + ml*(xg_l_RF^2 + zg_l_R^2);
Izzg_RFl=Izz_Rl + ml*(xg_l_RF^2 + yg_l_R^2);
Ixyg_RFl=-ml*xg_l_RF*yg_l_R;
Iyxg_RFl=Ixyg_RFl;
Ixzg_RFl=-ml*xg_l_RF*zg_l_R;
Izxg_RFl=Ixzg_RFl;
Iyzg_RFl=Iyz_Rl- ml*yg_l_R*zg_l_R;
Izyg_RFl=Iyzg_RFl;
% Inertia Tensor
I_RFl=[Ixxg_RFl Ixyg_RFl Ixzg_RFl;Iyxg_RFl Iyyg_RFl Iyzg_RFl;Izxg_RFl
Izyg_RFl Izzg_RFl];
```

```
% Moments of Inertia referred to the base CoM (Right Back Leg)
Ixxg_RBl=Ixx_Rl + ml*(yg_l_R^2 + zg_l_R^2);
Iyyg_RBl=Iyy_Rl + ml*(xg_l_RB^2 + zg_l_R^2);
Izzg_RBl=Izz_Rl + ml*(xg_l_RB^2 + yg_l_R^2);
Ixyg_RBl=-ml*xg_l_RB*yg_l_R;
Iyxg_RBl=Ixyg_RBl;
Ixzg_RBl=-ml*xg_l_RB*zg_l_R;
Izxg_RBl=Ixzg_RBl;
Iyzg_RBl=Iyz_Rl- ml*yg_l_R*zg_l_R;
Izyg_RBl=Iyzg_RBl;
% Inertia Tensor
I_RBl=[Ixxg_RBl Ixyg_RBl Ixzg_RBl;Iyxg_RBl Iyyg_RBl Iyzg_RBl;Izxg_RBl
Izyg_RBl Izzg_RBl];

% Moments of Inertia referred to the base CoM (Left Front Leg)
Ixxg_LFl=Ixx_Ll + ml*(yg_l_L^2 + zg_l_L^2);
Iyyg_LFl=Iyy_Ll + ml*(xg_l_LF^2 + zg_l_L^2);
Izzg_LFl=Izz_Ll + ml*(xg_l_LF^2 + yg_l_L^2);
Ixyg_LFl=-ml*xg_l_LF*yg_l_L;
Iyxg_LFl=Ixyg_LFl;
Ixzg_LFl=-ml*xg_l_LF*zg_l_L;
Izxg_LFl=Ixzg_LFl;
Iyzg_LFl=Iyz_Ll- ml*yg_l_L*zg_l_L;
Izyg_LFl=Iyzg_LFl;
% Inertia Tensor
I_LFl=[Ixxg_LFl Ixyg_LFl Ixzg_LFl;Iyxg_LFl Iyyg_LFl Iyzg_LFl;Izxg_LFl
Izyg_LFl Izzg_LFl];

% Moments of Inertia referred to the base CoM (Left Back Leg)
Ixxg_LBl=Ixx_Ll + ml*(yg_l_L^2 + zg_l_L^2);
Iyyg_LBl=Iyy_Ll + ml*(xg_l_LB^2 + zg_l_L^2);
Izzg_LBl=Izz_Ll + ml*(xg_l_LB^2 + yg_l_L^2);
Ixyg_LBl=-ml*xg_l_LB*yg_l_L;
Iyxg_LBl=Ixyg_LBl;
Ixzg_LBl=-ml*xg_l_LB*zg_l_L;
Izxg_LBl=Ixzg_LBl;
Iyzg_LBl=Iyz_Ll- ml*yg_l_L*zg_l_L;
Izyg_LBl=Iyzg_LBl;
% Inertia Tensor
I_LBl=[Ixxg_LBl Ixyg_LBl Ixzg_LBl;Iyxg_LBl Iyyg_LBl Iyzg_LBl;Izxg_LBl
Izyg_LBl Izzg_LBl];

%%--------------------TOTAL MOMENT OF INERTIA-------------------------------

Ir=I_b+I_mr+I_RFu+I_RBu+I_LFu+I_LBu+I_RFl+I_RBl+I_LFl+I_LBl;
```

Appendix C Arduino Code

This Appendix includes the full code used in the prototype during the tests. The code is divided in four sections. In the first one, the library files are included and the variables are defined. The second one includes the code for initialization of the serial port, sensors and servo motors. The third section consists in the main program, which includes the program logic and decision flowchart. The last section includes all the functions that have been created and used in the main program.

```
/*INCLUDE LIBRARIES AND VARIABLES DEFINITION*/


//Include library files
#include <Wire.h>
#include <VL53L0X.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Kalman.h"

//Define timers and time variables
uint32_t start, finish, timer1, currMillis, prevMillis=0;
double duration, t_delay, Ts=0.05;

//Distance sensor variables
VL53L0X sensor;
uint16_t sensorRange;
double filter, prevFilter, gain=0.25;
boolean approach=0;

//Force sensors and force controller variables
int sensorPinLF=1, sensorPinLB=7, sensorPinRF=2, sensorPinRB=6, tot,
sensorValueLF = 0, sensorValueLB = 0, sensorValueRF = 0, sensorValueRB = 0;
double e_f, e1_lf, e1_rf, e1_lb, e1_rb, f_pd, incr=0.002, incr2=0.0002;
double kp_lf=0.00025, kd_lf=0.000025, kp_rf=0.00025, kd_rf=0.000025,
kp_lb=0.00025, kd_lb=0.000025, kp_rb=0.00025, kd_rb=0.000025, alpha1=0.75;
boolean contactLF, contactLB, contactRF, contactRB, touchedLF, touchedLB,
touchedRF, touchedRB;

//PID variables
double e1_roll, delta_u_roll=0, e1_pitch, delta_u_pitch=0, e_roll, sp_r=0,
e_pitch, sp_p=0.02;
//SMC variables
double s_r, s_p, psi_r=0.2, psi_p=0.2, sat_s, u_sat=0, u_vel_r=0,
u_vel_p=0, Kd_r=0.1, Kd_p=0.035;
double c2r=0, c1r=0.2, c2p=0, c1p=0.05, wr, wp, u_st;
//Attitude controller Output
double vz_lf, vz_rf, vz_lb, vz_rb;
```

```
//Inertial Measurement Unit
MPU6050 imu; //Define sensor
int16_t ax, ay, az;
int16_t gx, gy, gz;
Kalman kalmanX, kalmanY; //Define Kalman object
double gyroXrate, gyroYrate, gyroX, gyroY, roll, pitch, roll2, pitch2;
double accX, accY, accZ, alpha=0.5, dt;

//Angles Initial Positions
float th_LFh_0=-3.14/2 + 1.571, th_LFk_0=-1.571, th_RFh_0=-3.14/2 + 1.571,
th_RFk_0=-1.571;
float th_LBh_0=-3.14/2 + 1.571, th_LBk_0=-1.571, th_RBh_0=-3.14/2 + 1.571,
th_RBk_0=-1.571;
double th_LFh, th_LFk, th_RFh, th_RFk, th_LBh, th_LBk, th_RBh, th_RBk;
double th_LFh1, th_LFk1, th_RFh1, th_RFk1, th_LBh1, th_LBk1, th_RBh1,
th_RBk1;
//Legs Initial Position
double Y_max=-0.16, Y_min=-0.035, L1=0.093, L2=0.096;
double x_lf_0=-(L1*cos(th_LFh_0)+L2*cos(th_LFh_0+th_LFk_0)),
y_lf_0=L1*sin(th_LFh_0)+L2*sin(th_LFh_0+th_LFk_0), x_lf=x_lf_0,
y_lf=y_lf_0;
double x_rf_0=L1*cos(th_RFh_0)+L2*cos(th_RFh_0+th_RFk_0),
y_rf_0=L1*sin(th_RFh_0)+L2*sin(th_RFh_0+th_RFk_0), x_rf=x_rf_0,
y_rf=y_rf_0;
double x_lb_0=-(L1*cos(th_LBh_0)+L2*cos(th_LBh_0+th_LBk_0)),
y_lb_0=L1*sin(th_LBh_0)+L2*sin(th_LBh_0+th_LBk_0), x_lb=x_lb_0,
y_lb=y_lb_0;
double x_rb_0=L1*cos(th_RBh_0)+L2*cos(th_RBh_0+th_RBk_0),
y_rb_0=L1*sin(th_RBh_0)+L2*sin(th_RBh_0+th_RBk_0), x_rb=x_rb_0,
y_rb=y_rb_0;

//Encoders positions
int enc_LFh, enc_LFk, enc_LBh, enc_LBk, enc_RFh, enc_RFk, enc_RBh, enc_RBk;

boolean landed, ledState=0, appr=0;
double incrLF, incrRF, incrLB, incrRB;
int balanced, led_pin = 14, i=20;

//Define dynamixel motors variables
#include <DynamixelWorkbench.h>
#define DXL_BUS_SERIAL1 "1"    //Dynamixel on Serial1(USART1)  <-OpenCM9.04
#define BAUDRATE  1000000
#define GOAL_SPEED1    150
#define GOAL_SPEED2    300
//Motors ID's
#define LF_h 1
#define LF_k 2
#define LB_h 5
#define LB_k 6
#define RF_h 3
#define RF_k 4
#define RB_h 7
#define RB_k 8

DynamixelWorkbench dxl_wb;



/*INITIALIZATION*/
```

```
void setup() {
  // Initialize serial port, sensors and motors
  Serial.begin(BAUDRATE);
  //Serial2.begin(115200); //(Wireless)
  Wire.begin();
  imu.initialize();

  sensor.init();               //Distance sensor
  sensor.setTimeout(500);
  sensor.startContinuous();

  dxl_wb.begin(DXL_BUS_SERIAL1, BAUDRATE);  //Servo motors
  servosInit();                //Initialize motors
  servosSpeed(GOAL_SPEED1);    //Set motors operational speed
  servosLimits();              //Set motors position limits
}

/*MAIN PROGRAM*/

void loop() {

  start=millis();      // read start time

  distanceSensor();    // Execute sensors sub-routines
  footSensors();
  KalmanAngle();

  // If distance>500mm put legs in flying position
  if (filter>500){
    flightPose();
    goalAngles();
    setPosition();
    resetvalues();
  }
  // Below 500mm move legs to landing position
  else if (filter<500 && filter>300){
    landingPose();
    goalAngles();
    setPosition();
    if (appr==1){
      servosSpeed(GOAL_SPEED1); //Set motors speed
      appr=0;
    }
  }
  // If distance>300mm initialize landing procedure
  else if (filter<300){
    if (appr==0){
      appr=1;
      servosSpeed(GOAL_SPEED2); //Set motors speed
    }
    //Check how many legs are in ground contact?
    if (tot<4){
      tot=touchedLF + touchedLB + touchedRF + touchedRB;
    }
    //Check body attitude. Is it balanced?
    if (roll>-0.01 && roll<0.01 && pitch>0.01 && pitch<0.03){
      balanced=balanced + 1; //Increase counter if it's balanced
    }
    else {balanced=0;} //Restart counter if not
    //Landing started. Execute controllers. Stage 1
```

```
    if (tot>=1 && tot<4 && landed==0){
     led_blink();      //LED blinking indicates landing started
     levelcontroller1(); //PD attitude controller
     //SMC();  //SMC controller
     //STSMC();//Super-Twisting controller
     landing();
    }
    else if (tot==4 && landed==0){
        if (balanced>=i){      //If all condition met --> landing finished
          fixPosition();       //fix legs to the current position
          landed=1;
          led_on();    //LED on indicates landing finished
          servosSpeed(GOAL_SPEED1);
        }
        else if (balanced<i){  //keep executing level controller. Stage 2
          led_blink();
          levelcontroller2();//PD attitude controller
          //SMC();     //SMC controller
          //STSMC();   //Super-Twisting controller
          landing2();
        }
      }
      else if (filter>250){
      landingPose();
      resetvalues();
      led_off();
      }
      if (landed==0){  //while landing is not finished
       goalFeetXY();   //execute leg kinematics
       goalAngles();
       setPosition();  //and convert to motors position commands

  }
  readmotors();//read motors positions

  finish=millis();     // read finish time
  //Calculate cycle time duration
  duration=(double)finish - (double)start;
  /*To obtain a fixed sample time, the cycle time duration is subtracted
  from the sample time, and the difference is added to the code as a delay*/
  t_delay=Ts*1000-duration;
  if (t_delay>0){delay(t_delay);}
}

/*FUNCTIONS*/

//This function provides the feet Cartesian coordinates for flight position
void flightPose(){

  y_lf=Y_min;
  y_rf=Y_min;
  y_lb=Y_min;
  y_rb=Y_min;
  x_lf=-0.067;
  x_rf=0.067;
  x_lb=-0.067;
  x_rb=0.067;

}

//This function provides feet Cartesian coordinates for landing position
```

```
void landingPose(){

  y_lf=y_lf_0;
  y_rf=y_rf_0;
  y_lb=y_lb_0;
  y_rb=y_rb_0;
  x_lf=x_lf_0;
  x_rf=x_rf_0;
  x_lb=x_lb_0;
  x_rb=x_rb_0;
}
//This function initialises the 8 servo motors
void servosInit(){
  dxl_wb.ping(LF_h);
  dxl_wb.ping(LF_k);
  dxl_wb.ping(RF_h);
  dxl_wb.ping(RF_k);
  dxl_wb.ping(LB_h);
  dxl_wb.ping(LB_k);
  dxl_wb.ping(RB_h);
  dxl_wb.ping(RB_k);

  dxl_wb.jointMode(LF_h);
  dxl_wb.jointMode(LF_k);
  dxl_wb.jointMode(RF_h);
  dxl_wb.jointMode(RF_k);
  dxl_wb.jointMode(LB_h);
  dxl_wb.jointMode(LB_k);
  dxl_wb.jointMode(RB_h);
  dxl_wb.jointMode(RB_k);
}
/*This function reads the encoder of the 8 servo motors and returns the
angular position in radians*/
void readmotors(){

  th_LFh1=enc2angl(dxl_wb.itemRead(LF_h, "Present_Position"));
  th_LFk1=enc2angl(dxl_wb.itemRead(LF_k, "Present_Position"));
  th_LBh1=enc2angl(dxl_wb.itemRead(LB_h, "Present_Position"));
  th_LBk1=enc2angl(dxl_wb.itemRead(LB_k, "Present_Position"));
  th_RFh1=enc2angl(dxl_wb.itemRead(RF_h, "Present_Position"));
  th_RFk1=enc2angl(dxl_wb.itemRead(RF_k, "Present_Position"));
  th_RBh1=enc2angl(dxl_wb.itemRead(RB_h, "Present_Position"));
  th_RBk1=enc2angl(dxl_wb.itemRead(RB_k, "Present_Position"));

}
/*This function sets the moving speed to the 8 servo motors*/
void servosSpeed(int vel){
  dxl_wb.itemWrite(LF_h, "Moving_Speed", vel);
  dxl_wb.itemWrite(LF_k, "Moving_Speed", vel);
  dxl_wb.itemWrite(RF_h, "Moving_Speed", vel);
  dxl_wb.itemWrite(RF_k, "Moving_Speed", vel);
  dxl_wb.itemWrite(LB_h, "Moving_Speed", vel);
  dxl_wb.itemWrite(LB_k, "Moving_Speed", vel);
  dxl_wb.itemWrite(RB_h, "Moving_Speed", vel);
  dxl_wb.itemWrite(RB_k, "Moving_Speed", vel);
}



/*This function sets the upper and lower position limits of the 8 servo
motors*/
```

```
void servosLimits(){
  //check EXCEL file: MX_AX_AngleLimits.xlsx
  dxl_wb.itemWrite(LF_h, "CW_Angle_Limit", 611);
  dxl_wb.itemWrite(LF_h, "CCW_Angle_Limit", 963);
  dxl_wb.itemWrite(LF_k, "CW_Angle_Limit", 51);
  dxl_wb.itemWrite(LF_k, "CCW_Angle_Limit", 512);
  dxl_wb.itemWrite(RF_h, "CW_Angle_Limit", 61);
  dxl_wb.itemWrite(RF_h, "CCW_Angle_Limit", 413);
  dxl_wb.itemWrite(RF_k, "CW_Angle_Limit", 512);
  dxl_wb.itemWrite(RF_k, "CCW_Angle_Limit", 973);

  dxl_wb.itemWrite(LB_h, "CW_Angle_Limit", 611);
  dxl_wb.itemWrite(LB_h, "CCW_Angle_Limit", 963);
  dxl_wb.itemWrite(LB_k, "CW_Angle_Limit", 51);
  dxl_wb.itemWrite(LB_k, "CCW_Angle_Limit", 512);
  dxl_wb.itemWrite(RB_h, "CW_Angle_Limit", 61);
  dxl_wb.itemWrite(RB_h, "CCW_Angle_Limit", 413);
  dxl_wb.itemWrite(RB_k, "CW_Angle_Limit", 512);
  dxl_wb.itemWrite(RB_k, "CCW_Angle_Limit", 973);
}

/*This function reads the pressure from the force sensors and determines the
state of each foot.
For each foot, the "touched" flag is set to high if at any moment during the
landing the pressure at this foot reaches the threshold value.
The "contact" flag indicates wether at the present moment, a leg is on contact
or not.*/
void footSensors(){

  sensorValueLF = analogRead(sensorPinLF)*alpha1+sensorValueLF*(1-alpha1);
    if (touchedLF==0){if (sensorValueLF>300){touchedLF=1;}}
    else if (touchedLF==1){
      if (contactLF==0){if (sensorValueLF>300){contactLF=1;}}
      else {if (sensorValueLF<300){contactLF=0; e1_lf=0;}}
    }

  sensorValueRB = analogRead(sensorPinRB)*alpha1+sensorValueRB*(1-alpha1);
    if (touchedRB==0){if (sensorValueRB>300){touchedRB=1;}}
    else if (touchedRB==1){
      if (contactRB==0){if (sensorValueRB>300){contactRB=1;}}
      else {if (sensorValueRB<300){contactRB=0; e1_rb=0;}}
    }

  sensorValueLB = analogRead(sensorPinLB)*alpha1+sensorValueLB*(1-alpha1);
    if (touchedLB==0){if (sensorValueLB>300){touchedLB=1;}}
    else if (touchedLB==1){
      if (contactLB==0){if (sensorValueLB>300){contactLB=1;}}
      else {if (sensorValueLB<300){contactLB=0; e1_lb=0;}}
    }

  sensorValueRF = analogRead(sensorPinRF)*alpha1+sensorValueRF*(1-alpha1);
    if (touchedRF==0){if (sensorValueRF>300){touchedRF=1;}}
    else if (touchedRF==1){
      if (contactRF==0){if (sensorValueRF>300){contactRF=1;}}
      else {if (sensorValueRF<300){contactRF=0; e1_rf=0;}}
    }
}


/*This function is the digital implementation of the PD force controller*/
```

```
double PD_FS(int fs_value, int setpoint, double &e_prev, double kp, double
kd){ //&-->call by reference
    e_f=setpoint-fs_value;
    f_pd=kp*e_f+kd*(e_f-e_prev)/Ts;
    e_prev=e_f;
    return f_pd;
}

/*This function is the digital implementation of the PD attitude controller
during Stage 1*/
void levelcontroller1(){
  //Roll controller
  e_roll=sp_r-roll;    //Roll error
  delta_u_roll=0.25*e_roll+0.025*(e_roll-e1_roll)/Ts; //PD output
  e1_roll=e_roll;      //Previous error update
  //Pitch controller
  e_pitch=sp_p-pitch;
  delta_u_pitch=0.07*e_pitch+0.00375*(e_pitch-e1_pitch)/Ts;
  e1_pitch=e_pitch;
  //Feet velocity due to attitude controller
  vz_lf=delta_u_roll+delta_u_pitch;
  vz_rf=-delta_u_roll+delta_u_pitch;
  vz_lb=delta_u_roll-delta_u_pitch;
  vz_rb=-delta_u_roll-delta_u_pitch;

}
/*This function is the digital implementation of the PD attitude controller
during Stage 2*/
void levelcontroller2(){
  //Roll controller
  e_roll=sp_r-roll;
  delta_u_roll=0.5*e_roll+0.05*(e_roll-e1_roll)/Ts;
  e1_roll=e_roll;
  //Pitch controller
  e_pitch=sp_p-pitch;
  delta_u_pitch=0.075*e_pitch+0.0075*(e_pitch-e1_pitch)/Ts;
  e1_pitch=e_pitch;
  //Feet velocity due to attitude controller
  vz_lf=delta_u_roll+delta_u_pitch;
  vz_rf=-delta_u_roll+delta_u_pitch;
  vz_lb=delta_u_roll-delta_u_pitch;
  vz_rb=-delta_u_roll-delta_u_pitch;

}

/*This function is the digital implementation of the SMC attitude controller
with boundary layer*/
void SMC(){
  //Roll controller
  s_r=sp_r-roll;       //Roll error
  if (abs(s_r)>psi_r) {sat_s=s_r/abs(s_r);}   //sat function
  else {sat_s=s_r/psi_r;}
  u_vel_r =-Kd_r*sat_s;        //SMC output
  e1_roll=e_roll;              //Previous error update
  //Pitch controller
  s_p=sp_p-pitch;      //Pitch error
  if (abs(s_p)>psi_p) {sat_s=s_p/abs(s_p);}
  else {sat_s=s_p/psi_p;}
  u_vel_p =-Kd_p*sat_s;
  e1_pitch=e_pitch;
  //Feet velocity due to attitude controller
```

```
  vz_lf=-u_vel_r-u_vel_p;
  vz_rf=u_vel_r-u_vel_p;
  vz_lb=-u_vel_r+u_vel_p;
  vz_rb=u_vel_r+u_vel_p;
}

/*This function is the digital implementation of the Super-Twisting SMC
attitude controller*/
void STSMC(){
  //Roll controller
  s_r=sp_r-roll;
  wr=wr+(-c2r*s_r/abs(s_r))*Ts;
  //u_st=-c1r*sqrt(abs(s_r))*(s_r/abs(s_r))+wr;
  u_st=-c1r*pow(abs(s_r),0.75)*(s_r/abs(s_r))+wr;
  u_vel_r=u_st;
  e1_roll=e_roll;
  //Pitch controller
  s_p=sp_p-pitch;
  wp=wp+(-c2p*s_p/abs(s_p))*Ts;
  //u_st=-c1p*sqrt(abs(s_p))*(s_p/abs(s_p))+wp;
  u_st=-c1p*pow(abs(s_p),0.75)*(s_p/abs(s_p))+wp;
  u_vel_p=u_st;
  e1_pitch=e_pitch;
  //Feet velocity due to attitude controller
  vz_lf=-u_vel_r-u_vel_p;
  vz_rf=u_vel_r-u_vel_p;
  vz_lb=-u_vel_r+u_vel_p;
  vz_rb=u_vel_r+u_vel_p;
}


/*This function defines the increase of leg length during landing Stage 1.
If a leg hasn't touched the the ground it will retract at a fixed rate.
If it has already touched the ground, and it's in contact at the moment it
will retract according to the controller output.
Otherwise it will stay as it is.*/
void landing(){
  if (touchedLF==0){
    incrLF=incrLF - incr; if (incrLF <= Y_max) {incrLF=Y_max;}}
  else {
    if (contactLF==1){
      incrLF=incrLF-PD_FS(sensorValueLF,400,e1_lf,kp_lf,kd_lf)*Ts+
      vz_lf*Ts;
      if (incrLF >= 0.065) {incrLF=0.065;}}
    else {incrLF=incrLF;}
  }

  if (touchedLB==0){
    incrLB=incrLB - incr; if (incrLB <= Y_max) {incrLB=Y_max;}}
  else {
    if (contactLB==1){
      incrLB=incrLB-PD_FS(sensorValueLB,400,e1_lb,kp_lb,kd_lb)*Ts+
      vz_lb*Ts;
      if (incrLB >= 0.065) {incrLB=0.065;}}
    else {incrLB=incrLB;}
  }

  if (touchedRF==0){
    incrRF=incrRF - incr; if (incrRF <= Y_max) {incrRF=Y_max;}}
  else {
    if (contactRF==1){
```

```
      incrRF=incrRF-PD_FS(sensorValueRF,400,e1_rf,kp_rf,kd_rf)*Ts+
      vz_rf*Ts;
      if (incrRF >= 0.065) {incrRF=0.065;}}
    else {incrRF=incrRF;}
  }

  if (touchedRB==0){
    incrRB=incrRB - incr; if (incrRB <= Y_max) {incrRB=Y_max;}}
  else {
    if (contactRB==1){
      incrRB=incrRB-PD_FS(sensorValueRB,400,e1_rb,kp_rb,kd_rb)*Ts+
      vz_rb*Ts;
      if (incrRB >= 0.065) {incrRB=0.065;}}
    else {incrRB=incrRB;}
  }
}


/*This function defines the increase of leg length during landing Stage 2
when all legs have touched the ground. If a leg is in contact it will adjust
according to the attitude controller only*/
void landing2(){
  if (contactLF==0)
    {incrLF=incrLF-incr2+vz_lf*Ts; if (incrLF <= Y_max) {incrLF=Y_max;}}
  else {incrLF=incrLF + vz_lf*Ts;}

  if (contactRF==0)
    {incrRF=incrRF-incr2+vz_rf*Ts; if (incrRF <= Y_max) {incrRF=Y_max;}}
  else {incrRF=incrRF + vz_rf*Ts;}

  if (contactLB==0)
    {incrLB=incrLB-incr2+vz_lb*Ts; if (incrLB <= Y_max) {incrLB=Y_max;}}
  else {incrLB=incrLB + vz_lb*Ts;}

  if (contactRB==0)
    {incrRB=incrRB-incr2+vz_rb*Ts; if (incrRB <= Y_max) {incrRB=Y_max;}}
  else {incrRB=incrRB + vz_rb*Ts;}
}

/*This function reads and filters the output from the distance sensor*/
void distanceSensor(){

  sensorRange=sensor.readRangeContinuousMillimeters();
  filter=prevFilter+gain*(sensorRange-prevFilter);
  prevFilter=filter;
}

/*This function reads the output from the IMU. It combines the output of the
3-axis accelerometer and 3-axis gyroscope using a Kalman filter*/
void KalmanAngle(){
  //Take accelerometor and gyroscope readings in x, y, and z axis
  imu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);       //Read sensor
  //Compute roll and pitch angles from accelerometer
  accX=ax*alpha + (accX*(1-alpha));    //Pre-filter
  accY=ay*alpha + (accY*(1-alpha));
  accZ=az*alpha + (accZ*(1-alpha));
  roll2=atan2(accX, sqrt(accZ*accZ + accY*accY))+0.17; //Roll
  pitch2=atan2(accY, accZ);    //Pitch
  //Compute roll and pitch angles from gyroscope
  gyroX=gy*alpha + (gyroX*(1-alpha)); //Pre-filter
  gyroY=gx*alpha + (gyroY*(1-alpha));
```

```
  gyroXrate=-(gyroX/131.0*DEG_TO_RAD); //x angular velocity
  gyroYrate=(gyroY/131.0*DEG_TO_RAD);  //y angular velocity
  dt=(millis()-timer1)*0.001;
  //Use Kalman filter to combine readings from both sensors
  roll = kalmanX.getAngle(roll2, gyroXrate, dt);
  pitch = kalmanY.getAngle(pitch2, gyroYrate, dt);
  timer1=millis();
}

/*This function restarts all system states or accumulative values (integrals)
if after or during the landing a height threshold is reached, so the gear is
set at initial state.*/
void resetvalues(){
  landed=0;
  incrLF=0;
  incrLB=0;
  incrRF=0;
  incrRB=0;
  wr=0;
  wp=0;
  u_vel_r=0;
  u_vel_p=0;
  e1_roll=0;
  touchedLF=0;
  touchedRF=0;
  touchedLB=0;
  touchedRB=0;
  contactLF=0;
  contactRF=0;
  contactLB=0;
  contactRB=0;
  tot=0;
}

/*This function defines the LED blinking during the landing operation*/
void led_blink() {
  currMillis = millis();
  if (currMillis - prevMillis >= 150) {
    // save the last time you blinked the LED
    prevMillis = currMillis;
    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {ledState = HIGH;}
    else {ledState = LOW;}
    // set the LED with the ledState of the variable:
    digitalWrite(led_pin, ledState);
  }
}

/*This function turns the LED off*/
void led_off() {
  digitalWrite(led_pin, HIGH);  // set to as HIGH LED is turn-off
}

/*This function turns the LED on*/
void led_on() {
  digitalWrite(led_pin, LOW);   // set to as LOW LED is turn-on
}

/*This function provides the desired feet position in Cartesian coordinates.
The y-coordinate is a function of the controller output. Kinematic limits
are imposed.*/
```

```
void goalFeetXY(){
  y_lf=y_lf_0 + incrLF;
    if (y_lf < Y_max) {y_lf=Y_max;}
    else if (y_lf > Y_min) {y_lf=Y_min;}
  y_rf=y_rf_0 + incrRF;
    if (y_rf < Y_max) {y_rf=Y_max;}
    else if (y_rf > Y_min) {y_rf=Y_min;}
  y_lb=y_lb_0 + incrLB;
    if (y_lb < Y_max) {y_lb=Y_max;}
    else if (y_lb > Y_min) {y_lb=Y_min;}
  y_rb=y_rb_0 + incrRB;
    if (y_rb < Y_max) {y_rb=Y_max;}
    else if (y_rb > Y_min) {y_rb=Y_min;}
}

/*This function calcualates the hip and knee angles for each foot as a
function of the Cartesian coordinates (Inverse Kinematics). Then it converts
the angles into encoder values for the motors*/
void goalAngles(){
// Left front leg
  th_LFh = acos((-sq(L2)+sq(L1)+sq(x_lf)+sq(y_lf))/(2*L1*sqrt(sq(x_lf)+
    sq(y_lf)))) + atan(x_lf/y_lf);
  th_LFk = acos((-sq(x_lf)-sq(y_lf)+sq(L1)+sq(L2))/(2*L1*L2)) - 3.1416;
  enc_LFh = angl2enc (th_LFh);
  enc_LFk = angl2enc (th_LFk);
// Left back leg
  th_LBh = acos((-sq(L2)+sq(L1)+sq(x_lb)+sq(y_lb))/(2*L1*sqrt(sq(x_lb)+
    sq(y_lb)))) + atan(x_lb/y_lb);
  th_LBk = acos((-sq(x_lb)-sq(y_lb)+sq(L1)+sq(L2))/(2*L1*L2)) - 3.1416;
  enc_LBh = angl2enc (th_LBh);
  enc_LBk = angl2enc (th_LBk);
// Right front leg
  th_RFh = - acos((-sq(L2)+sq(L1)+sq(x_rf)+sq(y_rf))/(2*L1*sqrt(sq(x_rf)+
    sq(y_rf)))) + atan(x_rf/y_rf);
  th_RFk = - acos((-sq(x_rf)-sq(y_rf)+sq(L1)+sq(L2))/(2*L1*L2)) + 3.1416;
  enc_RFh = angl2enc (th_RFh);
  enc_RFk = angl2enc (th_RFk);
// Right back leg
  th_RBh = - acos((-sq(L2)+sq(L1)+sq(x_rb)+sq(y_rb))/(2*L1*sqrt(sq(x_rb)+
    sq(y_rb)))) + atan(x_rb/y_rb);
  th_RBk = - acos((-sq(x_rb)-sq(y_rb)+sq(L1)+sq(L2))/(2*L1*L2)) + 3.1416;
  enc_RBh = angl2enc (th_RBh);
  enc_RBk = angl2enc (th_RBk);
}

/*Function enc2angl converts encoder values returned by the servo motor into
readable angular positions in radians*/
double enc2angl (double enc){
  double angle = (enc-512)/512*150/180*3.1416;
  return angle;
}
/*Function angl2enc converts angular positions in radians into encoder values
that can be interpreted by the servo motors*/
int angl2enc (double angl){
  double enc = angl/3.1416*180/150*512+512;
  return (int) enc;
}

/*This function provides position commands to all motors. The input is the
encoder value for the desired angular position*/
void setPosition(){
```

```
  dxl_wb.itemWrite(LF_h, "Goal_Position", enc_LFh);
  dxl_wb.itemWrite(LF_k, "Goal_Position", enc_LFk);
  dxl_wb.itemWrite(RF_h, "Goal_Position", enc_RFh);
  dxl_wb.itemWrite(RF_k, "Goal_Position", enc_RFk);
  dxl_wb.itemWrite(LB_h, "Goal_Position", enc_LBh);
  dxl_wb.itemWrite(LB_k, "Goal_Position", enc_LBk);
  dxl_wb.itemWrite(RB_h, "Goal_Position", enc_RBh);
  dxl_wb.itemWrite(RB_k, "Goal_Position", enc_RBk);
}

/*This function fixes the legs in its current position by reading the current
position and command it as the goal position*/
void fixPosition(){
  dxl_wb.itemWrite(LF_h, "Goal_Position",
     dxl_wb.itemRead(LF_h,"Present_Position"));
  dxl_wb.itemWrite(LF_k, "Goal_Position",
     dxl_wb.itemRead(LF_k,"Present_Position"));
  dxl_wb.itemWrite(RF_h, "Goal_Position",
     dxl_wb.itemRead(RF_h,"Present_Position"));
  dxl_wb.itemWrite(RF_k, "Goal_Position",
     dxl_wb.itemRead(RF_k,"Present_Position"));
  dxl_wb.itemWrite(LB_h, "Goal_Position",
     dxl_wb.itemRead(LB_h,"Present_Position"));
  dxl_wb.itemWrite(LB_k, "Goal_Position",
     dxl_wb.itemRead(LB_k,"Present_Position"));
  dxl_wb.itemWrite(RB_h, "Goal_Position",
     dxl_wb.itemRead(RB_h,"Present_Position"));
  dxl_wb.itemWrite(RB_k, "Goal_Position",
     dxl_wb.itemRead(RB_k,"Present_Position"));
}
```

References

[1] US Department of Transportation, Federal Aviation Administration, Helicopter Flying Handbook, Oklahoma, 2012.

[2] G. Cai, T. H. Lee and B. M. Chen, Unmanned Rotorcraft Systems, London: Springer-Verlag, 2011.

[3] S. Flower, “Drone Make Inroads Into Industrial and Commercial Applications,” *Wireless, Design & Development,* vol. Vol. 26, no. Iss. 3, pp. 4-5, May/Jun 2018.

[4] F. Ruggiero, V. Lippiello and A. Ollero, “Aerial Manipulation: A literature Review,” *IEEE Robotics and Automation Letters,* 2018.

[5] S. Scherer, C. Lyle and S. Sanjiv, “Autonomous landing at unprepared sites by a full-scale helicopter,” *Robotics and Autonomous Systems,* vol. 60, pp. 1545-1562, 2012.

[6] P. J. Garcia-Pardo, G. S. Sukhatme and J. F. Montgomery, “Towards vision-based safe landing for an autonomous helicopter,” *Robotics and Autonomous Systems,* vol. 38, pp. 19-29, 2002.

[7] F. Kendoul, “Survey of Advances in Guidance, Navigation and Control of Unmanned Rotorcraft Systems,” *Journal of Field Robotics,* pp. 315-378, 2012.

[8] E. Leylek, M. Ward and M. Costello, “Flight Dynamic Simulation for Multibody Aircraft Configurations,” *Journal of Guidance, Control and Dynamics,* vol. 35, 2012.

[9] J. Keifer, M. Ward and M. Costello, “Rotorcraft Hard Landing Mitigation Using Robotic Landing Gear,” *Journal of Dynamic Systems, Measurement and Control,* vol. 138, 2016.

[10] D. W. Felder, “Slope Landing Compensator System”. Patent US4062507 A, 1977.

[11] R. Parks, R. Minelli, S. Haase and N. Boertlein, “Adaptive landing gear”. Patent US20070221783 A1, 2007.

[12] S. H. Mason, “Helicopter self-leveling landing gear”. United States Patent US 3857533 A, 1974.

[13] S. M. Calvert, “Telescoping landing leg system”. United States Patent US9033276 B1, 2015.

[14] N. K. Gentry, “Adjustable landing gear assembly for unmanned aerial vehicles”. United States Patent US20160272308A1, 2016.

[15] K. Button, “Robotic landing legs,” Aerospace America, July/August 2017. [Online]. Available: https://aerospaceamerica.aiaa.org/departments/robotic-landing-legs/.

[16] V. Manivannan, J. P. Langley, M. F. Costello and M. Ruzzene, “Rotorcraft Slope Landings with Articulated Landing Gear,” in *AIAA Atmospheric Flight Mechanics (AFM) Conference*, 2013.

[17] K. Dooroo and M. Costello, “Virtual Model Control of Rotorcraft with Articulated Landing Gear for Shipboard Landing,” in *AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum*, San Diego, CA, 2016.

[18] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth and G. Pratt, “Virtual Model Control: An intuitive Approach for Bipedal Locomotion,” *The International Journal of Robotics Research,* vol. 20, pp. 129-143, 2001.

[19] S. Baker, D. Soccol, A. Postula and M. V. Srinivasan, “Passive Landing Gear using Coupled Mechanical Design,” in *Australian Conference on Robotics and Automation*, Sydney, 2013.

[20] Y. S. Sarkisov, G. A. Yashin, E. V. Tsykunov and D. Tsetserukou, “DroneGear: A Novel Robotic Landing Gear With Embedded Optical Torque Sensors for Safe Multicopter Landing on an Uneven Surface,” *IEEE Robotics and Automation Letters,* Vols. Vol. 3, No. 3, 2018.

[21] D. Hu, Y. Li, M. Xu and Z. Tang, “Research on UAV Adaptive Landing Gear Control System,” in *Journal of Physics: Conference Series*, 2018.

[22] ETH Zürich, “ATHLAS All-Terrain Helicopter Landing System,” 2017. [Online]. Available: http://www.athlas.ethz.ch.

[23] B. Stolz, T. Brodermann, E. Castiello, G. Englberger, D. Erne, J. Gasser, E. Hayoz, S. Muller, L. Muhlebach, T. Low, D. Scheuer, L. Vandeventer, M. Bjelonic, F. Gunther, H. Kolvenbach, M. Hopflinger and M. Hutter, “An Adaptive Landing Gear for Extending the Operational Range of Helicopters,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, 2018.

[24] M. Raibert, K. Blankespoor, G. Nelson and R. Playter, “BigDog, the Rough-Terrain Quadruped Robot,” in *17th IFAC World Congress*, Seoul, Korea, 2008.

[25] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella and D. G. Caldwell, “Design of HyQ - a hydraulically and electrically actuated quadruped robot,” *Journal of Systems and Control Engineering,* vol. 225, no. Special Issue Paper, pp. 831-849, 2011.

[26] K. Kotaka, B. Ugurlu, M. Kawanishi and T. Narikiyo, “Prototype Development and Real-Time Trot-Running Implementation of a Quadruped Robot: RoboCat-1,” in *International Conference in Mechatronics (ICM)*, Vicenza, Italy, 2013.

[27] B. Ugulru, I. Havoutis, C. Semini, K. Kayamori, D. G. Caldwell and T. Narikiyo, “Pattern generation and compliant feedback control for quadrupedal dynamic trot-walking locomotion: experiments on RoboCat-1 and HyQ,” *Autonomous Robots,* vol. 38, no. 4, p. 415–437, 2015.

[28] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D’Haene, R. Moeckel and A. Jan Ijspeert, “Oncilla Robot—A Light-weight Bio-inspired Quadruped Robot for Fast Locomotion in Rough Terrain,” in *Symposium on Adaptive Motion of Animals and Machines*, Japan, 2011.

[29] M. Ajallooeian, S. Gay, A. Tuleu, A. Sprowitz and A. J. Ijspeert, “Modular Control of Limit Cycle Locomotion over Unpercieved Rough Terrain,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013.

[30] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Modelling and Control, John Wiley & Sons, Inc, 2006.

[31] H. Asada, Introduction to Robotics, Massachusetts Institute of Technology, 2015.

[32] R. M. Murray, Z. Li and S. S. Sastry, A mathematical Introduction to Robotic Manipulation, CRC Press, 1994.

[33] B. Siciliano and O. Khatib, Springer Handbook Of Robotics, Springer-Verlag Berlin Heidelberg, 2016.

[34] R. N. Jazar, Theory of Applied Robotics. Kinematics, Dynamics and Control, New York, USA: Springer Science+Business Media, LLC, 2007.

[35] A. A. Shabana, Dynamics of Multibody Systems, New York, USA: Cambridge Univesrity Press, 2005.

[36] M. Sobotka, Hybrid Dynamical System Methods for Legged Robot Locomotion with Variable Ground Contact, Munich: Technische Universit¨at M¨unchen, 2006.

[37] D. W. Marhefka and D. E. Orin, “Simulation of Contact Using a Nonlinear Damper Model,” in *International Conference on Robotics and Automation*, Minneapolis, 1996.

[38] A. Ehsaniseresht and M. Mohammadi Moghaddam, “A new ground contact model for the simulation of bipeds' salking running and jumping,” in *RSI International Conference on Robotics and Mechatronics*, Tehran, Iran, 2105.

[39] B. Raiszadeh and D. Way, “Ground Contact Model for Mars Science Laboratory Mission Simulations,” in *AIAA Atmospheric Flight Mechanics Conference*, Minneapolis, Minnesota, 2012.

[40] Y. Liu, J. Li, Z. Zhang, X. Hu and W. Zhang, “Experimental Comparison of five friction models on the same test-bed of the micro stick-slip motion system,” *Mechanical Sciences. Open Access,* 2015.

[41] J. J. Craig, Introduction to Robotics. Mechanics and control, USA: Pearson Prentice Hall, 2005.

[42] F. L. Lewis, D. M. Dawson and C. T. Abdallah, Robot Manipulator Control. Theory and Practice, Marcel Dekker Inc., 2004.

[43] R. Kelly, V. Santibañez and A. Loria, Control of Robot Manipulators in Joint Space, Germany: Spring-Verlag London Limited, 2005.

[44] K. J. Astrom and R. M. Murray, Feedback Systems: An introduction for Scientists and Engineers, v2.11b ed., Princeton, New Jersey: Princeton University Press, 2012.

[45] Department of Electriccal and Electronics Engineering, Università degli Studi di Cagliari, “A quick introduction to sliding mode control and its applications”.

[46] J.-J. E. Slotine and W. Li, Applied Nonlinear Control, Upper Saddle River, New Jersey: Prentice Hall Inc., 1991.

[47] S. K. Spurgeon, “Sliding mode control: a tutorial,” in *European Control Conference (ECC 2014)*, Strasbourg, France, 2014.

[48] Y. Shtessel, C. Edwards, L. Fridman and A. Levant, “Sliding Mode Control and Observation,” Birkhäuser, 2014.

[49] K. Goh, S. K. Spurgeon and B. Barrie Jones, “Higher-order sliding mode control of a diesel generator set,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineers,* vol. 217, 2003.

[50] G. Bartolini, A. Ferrara, A. Levant and E. Usai, “On second order sliding mode controllers,” in *Lecture Notes in Control and Information Sciences*, Springer, 1999.

[51] J. Carpentier and N. Mansard, “Multicontact Locomotion of Legged Robots,” *IEEE Transactions on Robotics,* vol. 34, no. 6, 2018.

[52] A. W. Winkler, “Optimization-based motion planning for legged robots. PhD Thesis,” ETH Zurich, 2018.

[53] M. H. Raibert, . H. B. Brown, M. Chepponis, E. Hastings, J. Koechling, K. N. Murphy, S. S. Murphy and A. J. Stentz, “Dynamically Stable Legged Locomotion. Progress report: October 1982-1983,” Ca rnegie-Mellon University , Pittsburgh, PA, 1983.

[54] B. U. Rehamn, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell and C. Semini, “Towards a Multi-legged Mobile Manipulator,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

[55] X. Zhang, L. Lang, J. Wang and H. Ma, “The Quadruped Robot Locomotion Based on Force Control,” in *27th Chinese Control and Decision Conference (CCDC)*, 2015.

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

[57] C. Korpela, M. Orsag and P. Oh, “Towards Valve Turning using a Dual-Arm Aerial Manipulator,” in *International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, 2014.

[58] C. Semini, “HyQ - Design and Development of a Hydraulically Actuated Quadruped Robot, PhD Thesis,” Italian Institute of Technology (IIT), 2010.

[59] M. Peasgood, E. Kubica and J. McPhee, “Stabilization of a Dynamic Walking Gait Simulation,” *Journal of Computational and Nonlinear Dynamics,* vol. 2, pp. 65-72, 2006.

[60] Trossen Robotics, 2018. [Online]. Available: http://www.trossenrobotics.com/.

[61] “ROBOTIS e-Manual,” [Online]. Available: http://emanual.robotis.com/. [Accessed 2018].

[62] R. Inc., “Lynxmotion Hexapod Foot Sensor (Pair),” 2018. [Online]. Available: https://www.robotshop.com/en/lynxmotion-hexapod-foot-sensor-pair.html.

[63] R. Inc., “Lynxmotion Tubing Foot Contact Switch Kit (Pair),” 2018. [Online]. Available: https://www.robotshop.com/uk/lynxmotion-tubing-foot-contact-switch-kit.html.

[64] Interlink Electronics, Inc., *FSR 400 Series Datra Sheet,* Westlake Village, CA, USA.

[65] ROBOTIS, *Dynamixel AX-12. User's Manual,* 2006.

[66] Arduino, 2016. [Online]. Available: https://www.arduino.cc/.

[67] M. Pedley, “Tilt Sensing Using a Three-Axis Accelerometer,” Freescale Semiconductor, Inc., 2013.

[68] K. Lauszus, “A practical approach to Kalman filter and how to implement it,” TKJ Electronics, 2012. [Online]. Available: https://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/.

[69] K. Lauszus, “Kalman Filter,” TKJ Electronics, 2012. [Online]. Available: GitHub Repository https://github.com/TKJElectronics/KalmanFilter.

[70] Adafruit, “Adafruit VL53L0X Time of Flight Micro LIDAR Distance Sensor Breakout,” 2018. [Online]. Available: https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout.

[71] D. Jacobson, *Introduction to Dynamixel Motor Control Using the Arbotix-M Robocontroller. Manual,* Cleveland, OH: Department of Mechanical and Aerospace Engineering. Case Western Reserve University, 2014.

[72] ROBOTIS, “Dynamixel Workbench,” Github, Inc., 2019. [Online]. Available: https://github.com/ROBOTIS-GIT/dynamixel-workbench.

[73] J. Rowberg, “I2C device library,” Github, Inc., 2019. [Online]. Available: https://github.com/jrowberg/i2cdevlib.

[74] Adafruit Industries, “Adafruit VL53L0X Library,” Github, Inc., 2019. [Online]. Available: https://github.com/adafruit/Adafruit_VL53L0X.

[75] T. Boaventura, C. Semini, J. Buchli, M. Frigerio, M. Focchi and D. G. Caldwell, “Dynamic Torque Control of a Hydraulic Quadruped Robot,” in *International Conference on Robotics and Automation (ICRA)*, USA, 2012.

[76] M. Hutter, M. A. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy and R. Siegwart, “Hybrid Operational Space Control for Compliant Legged Systems,” in *Robitics Science and Systems Conference VIII*, Sydney, Australia, 2012.

[77] L. Lang, J. Wang, J. Rao, H. Ma and Q. Wei, “Dynamics Stability Analysis of a Trotting Quadruped Robot Based on Switching control,” *International Journal of Advanced Robotic Systems,* vol. 12, 2015.

[78] ALIGN Corporation Limited, “T-REX 500L Dominator Instruction Manual,” Taiwan, 2015.

[79] M. S. Fadali and A. Visioli, Digital Control Engineering. Analysis and Design, Burlington, USA: Elsevier Inc., 2009.

[80] B. Ugurlu, K. Kotaka and T. Narikiyo, “Actively-Compliant Locomotion Control on Rough Terrain: Cyclic Jumping and Trotting Experiments on a Stiff-by-

Nature Quadruped," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013.

[81] T. Boaventura Cunha, "Hydraulic Compliance Control of the Quadruped Robot HyQ. PhD Thesis," Instituto Italiano di Tecnologia (IIT), Genoa, Italy, 2013.

[82] C. Lebosse, P. Renaud, B. Bayle and M. de Mathelin, "Modeling and Evaluation of Low-Cost Force Sensors," *IEEE Transactions on Robotics,* vol. 27, no. 4, 2011.

[83] K. Kondak, M. Bernard, N. Losse and G. Hommel, "Elaborated Modelling and Control for Autonomous Small Size Helicopters," in *ISR/ Robotik*, Munich, 2006.

[84] L. A. Sandino, M. Bejar and A. Ollero, "A survey on Methods for Elaborated Modeling of the Mechanics of a Small-Size Helicopter. Analysis and Comparison," *Journal Of Intelligent and Robotic Systems,* vol. 72, no. 2, p. 219–238, 2013.