Politecnico di Torino

Master's Degree in Mechatronic Engineering

Active Suspension Control for Compliant Wheeled Locomotion on Uneven Terrain

Supervisor:
Prof. Alessandro Rizzo
Prof. Nikos Tsagarakis
Dr. Arturo Laurenzi
Dr. Luca Rossini
Dr. Francesco Ruscelli

Candidate:
Riccardo Giacchino

2023/2024

Politecnico di Torino

Master's Degree in Mechatronic Engineering

Active Suspension Control for Compliant Wheeled Locomotion on Uneven Terrain

Supervisor:
Prof. Alessandro Rizzo
Prof. Nikos Tsagarakis
Dr. Arturo Laurenzi
Dr. Luca Rossini
Dr. Francesco Ruscelli

Candidate:
Riccardo Giacchino

2023/2024

Abstract

Legged robots, as mobile systems, have demonstrated remarkable potential in navigating rough and uneven surfaces. Their multi-limb structure allows for a versatile interaction with the environment, coping with terrain disturbances and deformations. However, the motion complexity could arise due to this articulated design. The robot CENTAURO employed in this thesis presents a considerable mass and dimension that reduces its agility. The introduction of wheels will ensure a simpler and less energy-consuming locomotion method. Controlling these robots requires strategies that must take into account the unstructured nature of the terrain. Position-based controllers struggle to adapt quickly and accurately to the irregular profile of the environment. Furthermore, ensuring stability through criteria that rely on the definition of a support polygon becomes challenging in non-coplanar conditions.

To expand a real-time framework developed in the laboratory, a control strategy is proposed for the lower body of the robot CENTAURO to execute wheeled locomotion in a real-life scenario. This strategy is composed of two controllers, addressing the compliance and stability of the robot on a non-coplanar surface. First, a Cartesian Impedance controller has been developed, such that the robot's legs will act as an active suspension system able to quickly react in a compliant way against non-predicted changes in the terrain. Secondly, using the information coming from proprioceptive sensors and IMU measurement, an attitude controller was developed to ensure the robot maintains its base parallel to the ground plane. This functionality guarantees stability while traversing slopes with significant altitude differences.

The controller has been vastly tested, both in simulation and in real-world trials with the robot. Experiments have shown the ability of the Cartesian impedance control to negotiate obstacles that might otherwise cause the robot to tumble with non-compliant behaviour.

Future work will focus on implementing a more sophisticated strategy that will resolve some of the simplifications made in the robot's dynamics, aiming to ensure the stability of the robot.

Keywords: Humanoid, Floating base, Impedance control, Torque controlled, Wheeled-Legged robot

Acknowledgement

I would like to express my sincere gratitude to Prof. Alessandro Rizzo, for his invaluable interest and insightful feedback throughout the entire process of completing this master's thesis. His expertise, encouragement, and patience have been instrumental in shaping the direction and quality of this research.

I am also grateful to Prof. Nikos Tsagarakis, for welcoming me inside his department and providing the opportunity to undertake this project. A big thanks go to my supervisors Dr. Arturo Laurenzi, Dr. Luca Rossini, Dr. Francesco Ruscelli, with whom this work has been conducted. Their guidance and suggestions throughout this experience, have been of immense value to me. I extend my heartfelt thanks to all the members of the HHCM.

I thank Alessandro for sharing this journey with me and for being the first and best experience of living away from my parents. Do not drink too much tea. Thanks to Brada. Even though we are far apart, with you I share the best and most precious experiences. A recognition goes to the Meccatronici, and in particular to Carlo, Alice and Alessia for being my study buddies since day one, and much more. A big thanks to my friends of Titolo Eliminato for being by my side since high school.

I am deeply indebted to my family for their unconditional love, understanding, and encouragement throughout my academic pursuits and, above all, in my life. Thanks for believing in me even when I doubted myself (so a lot of times), and for inspiring me to reach these remarkable achievements. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Legged robots have been a focus of research in the field of robot exploration given their ability to step into roles unsuitable and dangerous to humans. They are constructed to mimic both human and animal traits, enabling them to navigate a world crafted for humans, featuring inclines, stairs, and vertically oriented spaces where the use of legs is crucial for movement. These systems are composed of multiple articulated legs which are responsible for all the interaction with the environment, allowing for dexterous and independent movements. Nevertheless, achieving locomotion in such robots often necessitates intricate control strategies, involving the synchronized motion of several motors, especially in highly redundant robots with a substantial number of joints per leg. Additionally, generating locomotion solely through leg movements presents further challenges, as each leg's motion may require coordination to accommodate the dynamic effects on the entire robot.

On smooth surfaces, intricate mobility tasks can be simplified by integrating solutions like wheel locomotion. Incorporating wheels reduces the number of components that actively contribute to the planned movements, with a consequent decrease in energy consumption and controller's complexity. When deploying a wheel-on-leg system on uneven surfaces, a combined action of leg and wheels is expected. While the wheels will provide the needed propulsion, the legs can be actuated to cope with the disturbances and adapt the base attitude to the deformities coming from the terrain. However, developing a control strategy with this purpose could result challenging since these hybrid systems face great difficulties in modelling and motion generation due to their complexity. Furthermore, the real-world environments where such robots are deployed are typically characterized by unstructured terrain and irregular profiles. When navigating these environments, the robot is subjected to disturbances in the form of external forces induced by the interactions with the ground. If these disturbances are not adequately filtered

through effective control design, they can result in unstable movements, such as loss of contact with the ground or tilting of the robot.

Position-based controllers lack in successfully handle these conditions. They require precise trajectory planning and an accurate modelling of both the robot and the environment, which can be hard to achieve. Also, the non-linear profile of the terrain could raise too many disturbances in the control loop, resulting in sub-optimal control actions. On the other hand, implementing a method based on vision sensors is not guaranteed to be reliable. When moving, the robot is subject to continuous vibrations, which may introduce noise in the obtained data. Applying appropriate filters could solve this issue, but at the same time could mask the detection of small surface deformities.

To avoid tilting the robot when executing motion tasks, the stability of the system could be maintained meeting the conditions imposed by stability criteria. These stability criteria, which could be static or dynamic, typically require that a ground reference point, such as Zero Moment Point (ZMP), Center of Pressure (COP) or Center of Mass (COM) lay inside the support polygon defined by the current contact points [12]. Although, the non-coplanarity condition of an uneven terrain limits the effectiveness of these criteria since is not always possible to find a support polygon. In addition, the controller applied to these systems runs on a real-time framework capable of executing tasks with minimal latency. Then, it is mandatory to develop a strategy that reduces this parameter with ad-hoc software, allowing the robot to react as quickly as possible to any changes in state.

This thesis proposes the design of two controllers for the wheel-on-leg robot CENTAURO, developed in the laboratory of Humanoid and Human Centered Mechatronics at the Istituto Italiano di Tecnologia. The already existing real-time framework was expanded by two modules: a Cartesian impedance Controller and an attitude controller on each leg. The purpose of the former is to handle all the disturbances coming from the environment in a compliant way. When deviating from the reference values, this controller will generate a force to counteract the movement, following a mass-spring-damper dynamics. Tuning the parameter of the controller will change the behaviour of the robot to better match the condition of the terrain. The latter has been developed with the intent to dynamically change the reference values of the Cartesian impedance controller to control the Roll and Pitch angle of the base, maintaining the robot base parallel to the ground. The modified framework is then tested on both real and simulated scenarios to validate the effectiveness of the controller.

1.1 State of the art

The following section exposes a brief history of the legged robot developed by other research centers, moving towards an overview of the recent hybrid Wheel-Legged robots. Then, some relative works about the topic of navigating rough terrain with wheel locomotion are reported, explaining how other researches managed to solve this issue.

1.1.1 Legged and Legged-On-Wheel Robots

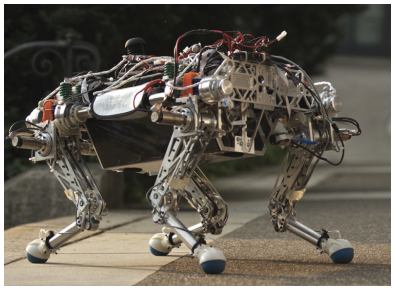
Walking machines have been an important topic of research in the field of robotics for several years [47]. Over the years, these systems were studied and developed into becoming the now called 'legged robots'. These systems, inspired by the animal environment represent, a promising solution for exploration missions in unknown and unstructured environments [3]. Their structure is typically composed of a base, in which the computational unit, sensors, and supply system are located, attached to multiple articulated chains which serve as legs of the robot, used to interact with the environments. From this common basis, several robots were developed from all over the world. IIT's Dynamic Legged System laboratory created a series of three quadruped robots HyQ, HyQmini and HyQ2Max, Figure 1.1. These robots' hydraulic actuator allows them to produce large force and transport huge loads while retaining flexible movement [42, 25, 43].

(a) HyQ (b) HyQmini (c) HyQ2Max

Figure 1.1: HyQ series developed by the Dynamic Legged System (DLS) laboratory of the Istituto Italiano di Tecnologia (IIT)

Although, the use of hydraulic actuators often leads to slower closed-loop control responses, due to their high bandwidth which is the range of frequencies over which a system can effectively operate or respond to inputs. Promising results are obtained from the application of electric motors in the robots employed in the ETH Zürich Robotic Systems Laboratory: Starl*ETH* [21] and ANYmal [20],

pictured in Figure 1.2, are quadruped robots driven by highly compliant and precisely torque controllable Series Elastic Actuators (SEA) in all joints, which is a suitable actuator system for environment interactions.



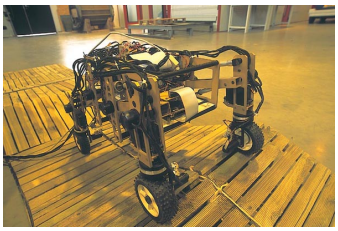
(a) StarlETH

(b) ANYmal

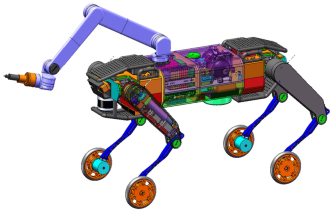
Figure 1.2: StarlETH and ANYmal of the ETH Zürich Robotic Systems Laboratory

One limitation of these robots is their exclusive reliance on leg-based locomotion methods, which necessitates a comprehensive study of gait sequences, leg motion, and various joint actuation. Additionally, this form of locomotion forces the robot to maintain only three or fewer contact points for the majority of movements, reducing the support polygon and potentially compromising stability. A simpler and more efficient solution is offered by the integration of the wheels on the robot, which reduces the complexity of the movements for simple tasks such as the forward and backward motion. The combination of the articulated legs and wheels has taken different forms: in the field of planetary exploration the wheels could be attached to an articulated multibody structure permitting a passive adaptation to the surface, as in the case of Sojourner [38], Shrimp [45], Nomad [41]. Other research shows instead a hybrid system, composed of active and passive elements. It is the case of Sherpa [8], developed by the Robotics Innovation Center of the University Bremen, or Complios [6], created in the Institut des Systèmes Intelligents et de Robotique (ISIR), where active joints are coupled with the passive action of physical springs. However, these types of robots limit the potentiality of the integration with articulated legs, reducing the adaptability when navigating uneven terrain. For this reason, completely controllable active legs have been the focus of research for active suspension mechanisms, to allow legged-on-wheel robots to complete complex locomotion tasks in unstructured environments. Great results were shown by the ISIR, with both Hylos [14] and TowrISIR [9] wheel-on-leg robots. Even the robot ANYmal has a version of legs with wheels, presented in Figure 1.4a

(a) Complios robot [6]



(b) Hylos I [14]

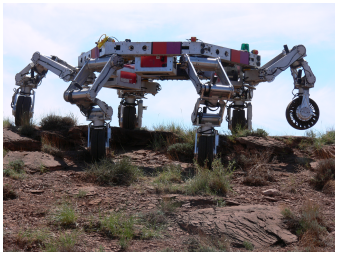


(c) TowrISIR [10]

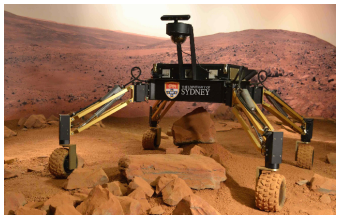
Figure 1.3: Robot developed inside the Institut des Systèmes Intelligents et de Robotique (ISIR)

Other highly articulated robots have been developed, each with its unique characteristics and intended applications. One notable example is CENTAURO, developed at the Istituto Italiano di Tecnologia's Humanoids and Human Centered Mechatronics Lab. Additionally, the robot MAMMOTH [37] and the NASA robot Athlete [17] also feature highly articulated structures with a high number of degrees of freedom (DOF). These complex structures enable these robots to achieve superior control and adaptability in their operating environments. However, the intricate nature of these highly articulated robots poses significant challenges in terms of leg motion and control. As a result, in these cases, more than others, wheeled locomotion emerges as the optimal solution to simplify locomotion tasks and improve overall efficiency.

(a) ANYmal with wheels



(b) Athlete



(c) Mammoth

Figure 1.4: Highly articulated and redundant robots

1.1.2 Related works

The advantages of hybrid wheel-legged robots were exploited in several previous works, implementing different techniques to couple the wheel locomotion with the leg actuation.

Complios [6] has a hybrid nature, composed of a physical spring to execute passive compliance, handling the horizontal motion of the wheels. In addition, a SEA controller on the vertical axis enables the implementation of force control for better load distribution management, introducing an orthogonal decomposition between the relative wheel movements. This decoupled motion is handled by two different dynamics. One is related to the vertical dynamic driven by the SEA controller, and the horizontal dynamic controlled by the wheel rotational speed to reduce internal stresses. However, the passive joint on each leg would limit its capability on unknown terrain, resulting in a swinging behaviour due to its undamped nature. In [5] the controller was updated with an obstacle negotiation technique based on reinforcement learning to better handle the distribution of the forces in different scenarios.

Instead, Hylos [14] was developed with a kinematic approach. With a total of sixteen actuated degrees of freedom between the four legs, this tiny and light robot is made up of a steerable motorized wheel for each leg and an active suspension with two degrees of freedom. A kinematic model is applied to optimize two parameters: ρ_i that is related to the traction criterion and θ_i related to the stability criterion developed in [34]. Minimizing the cost function of an optimization problem, it is possible to find the best position of the CoG such that the stability angles θ_i are maximised and the maximum friction ratio ρ_i is minimized. Then the computed optimal posture is fed into a postural control to compute the corresponding joint velocity.

At ISIR it was also developed a controller for the TowrISIR robot, Fig. 1.3c, where a wheel motion generator is proposed, that tracks the whole-body centroidal motion on unknown and uneven terrain, allowing for easier system modelling [10]. Here a wheel motion generator computes the desired velocity and acceleration of the wheels, based on the centroidal momentum and dynamic model. Then these values are fed into a multi-task Prioritized Torque controller that generates the torque of each joint through various impedance controls. In this first approach the desired leg-joint configuration is predefined and fixed, while in the more complete thesis [9] is presented a new version where given a reference motion of the Center of Mass, a whole-body motion generates the reference whole-body velocity and acceleration that servers as input for the same multi-task Torque controller. The results were very promising, with the robot able to overcome unpredicted terrain slopes, height differences, irregularities and step-like obstacles.

The author in [4] developed an optimization framework based on an MPC controller for the quadruped-on-wheel robot ANYmal, Fig. 1.4a. The main innovation of this approach was the possibility of combining the advantages of both

walking and driving. The solution to the optimization problem continuously updates the wheel and COM reference trajectory. To ensure dynamic stability, a Zero Moment Point-based inequality constraint is added to the problem formulation. As long as the ZMP [51] lies inside the support polygon, the stability is guaranteed. The operational space references are then tracked by a hierarchical whole-body controller. A series of tasks are solved via QP problem to compute the optimal generalized acceleration and contact forces, which are then converted into joint torques. The results show that with an update rate up to $200Hz$, the robot was able to overcome multiple challenging scenarios with great agility in the motion and good trajectory tracking. However, in this publication, the robot was only tested in conditions where the support polygon exists, which is not guaranteed when moving in uneven terrain.

1.2 Outline

This thesis is structured as follows:

- **Chapter 1** introduces the main challenges tackled in this thesis and provides an overview of the hybrid wheel-leg robot already developed and how they managed to solve these problems.
- **Chapter 2** briefly describes the robot dynamic, comparing fixed base and floating base systems. Then the robot CENTAURO is presented, detailing its structure, with a particular focus on the legs design. Lastly, the framework used to integrate the controllers on the robot is illustrated, together with the simulation tool employed in this thesis.
- **Chapter 3** starts with an introduction of the impedance controller and the motivation for choosing such a strategy. Then the computation of the control law for the Cartesian Impedance Controller is explained from a theoretical point of view.
- **Chapter 4** tests the efficacy of the Cartesian Impedance Controller in a simulated environment, studying the system's step response, and providing an overview of the controller's stability. Then the navigation capability of the robot is tested on two main terrains, with different settings configuration.
- in **Chapter 5**, the Attitude controller is presented, with a detailed description of the mathematical passage which led to the control law. Then the complete framework is tested in simulated environments to check its response to terrain interaction.

- **Chapter 6** shows the experiments on the real robot on a real-world terrain crafted inside the laboratory.
- In conclusion, **Chapter 7** summarizes the work presented in this thesis.

Chapter 2

Preliminaries and System Overview

Contents

Starting from the overview of the floating base dynamic, the following chapter moves on with the description of the framework developed inside the HHCM laboratory, upon which the implemented controller take place. Moreover, the robot CENTAURO employed in the tests is briefly introduced, explaining its overall structure, both for the lower and upper part. Furthermore, the simulation platform Gazebo used to test the controller is described.

2.1 Floating base robotics

Robots with floating bases, as the one in Figure 2.1, represent an improvement from conventional fixed-base robotic systems, offering enhanced mobility, adaptability, and dynamic interaction with the environment. While fixed base robots are restricted to static platforms and pre-defined trajectories, floating base robots possess six additional degrees of freedom (DOFs) allowing for translation and rotation of the base link expressed in the world frame.

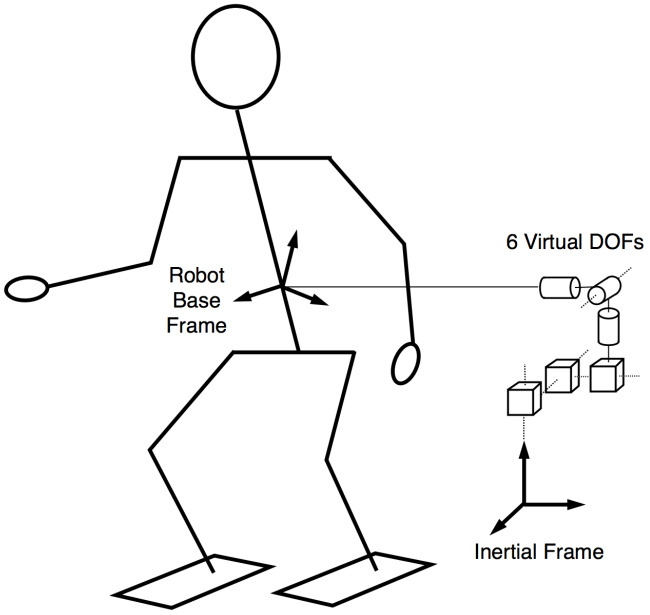


Figure 2.1: Representation of a floating base system

As detailed in [52] the new DOFs have to be considered in the description of the system. The configuration vector q, representing the position of each joint, is then composed by n_b un-actuated base joints $\boldsymbol{x_u}$ and n_j actuated joint coordinates $\boldsymbol{q_j}$:

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{x_u} \\ \boldsymbol{q_a} \end{pmatrix} \tag{2.1}$$

The base coordinates can be parameterized using different representations while maintaining a minimum number of generalized coordinates equal to $n_b = 6$. In particular, one common parametrization involves utilizing a transformation matrix $x_u \in SE(3)$ [39], where $SE(3)$ denotes the special Euclidean group in three dimensions. Alternately, is possible to divide $\boldsymbol{x_u}$ into a position vector $\boldsymbol{x_{u_p}} \in \mathbb{R}^3$ and a rotational matrix $\boldsymbol{x_{u_r}} \in SO(3)$ [11]. Other representation are the Euler angles, Angle Axis, Unit Quaternions [40].

These additional DOFs introduce various challenges in implementing a system with a floating base. The base's position and orientation can only be estimated and not directly measured. Various studies have developed techniques that utilize external sensors such as cameras or markers, as well as onboard sensors like the IMU, to measure the pose of the base. Since the virtual joints of the base are not actuated, the state of the robot can only be changed by the action of external

forces resulting from contact points between the robot and a surface. This problem will be further investigated in the following sections.

Dynamic

Robot dynamics provide us with a relation that describes how forces, both external and internal, affect the motion and the behaviour of the robot. This relation is expressed through the equation of motion, which is mainly obtained from the *Newtonian* and *Lagrangian* method, both leading to the same model. In this thesis will be used both methods.

In fixed-base robot, a complete equation of motion in joint space using the Lagrangian method is typically written as

$$\boldsymbol{M(q)\ddot{q} + C(q,\dot{q}) + g(q) = \tau - J^T h} \tag{2.2}$$

where $\boldsymbol{q, \dot{q}, \ddot{q}}$ are respectively the joint position, velocity and acceleration, $\boldsymbol{M(q)} \in \mathbb{R}^{n_j \times n_j}$ is the inertial matrix, dependent on the configuration of the robot, $\boldsymbol{C(q,\dot{q})} \in \mathbb{R}^{n_j}$ accounts for the Coriolis and centrifugal effect. $\boldsymbol{\tau}$ are the torque applied by the joint and $\boldsymbol{J^T h}$ describe the effect of external disturbances $\boldsymbol{h}$ on each joint through the Jacobian $\boldsymbol{J^T}$.

The equation of motion describing the dynamic of a floating base system is similar to the fixed base in (2.2), with the exception that the joint configuration of the robot is expressed through the generalized coordinates in (2.1). The new equation of motion may be written as

$$\begin{bmatrix} \boldsymbol{M_u(q)} & \boldsymbol{M_{ua}(q)} \\ \boldsymbol{M_{au}(q)} & \boldsymbol{M_a(q)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\ddot{x}_u} \\ \boldsymbol{\ddot{q}_a} \end{bmatrix} + \begin{bmatrix} \boldsymbol{C_u(q,\dot{q})} \\ \boldsymbol{C_a(q,\dot{q})} \end{bmatrix} + \begin{bmatrix} \boldsymbol{g_u(q)} \\ \boldsymbol{g_a(q)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0_{6\times 1}} \\ \boldsymbol{\tau} \end{bmatrix} + \boldsymbol{J_c^T \lambda} + \boldsymbol{J^T F_{ext}} \tag{2.3}$$

Appendix A.1 details each term in the previous equation. Focusing on the upper part of the equation 2.3, it is clear that the un-actuated terms are not affected by the action of the torque $\boldsymbol{\tau}$:

$$\boldsymbol{M_u(q)\ddot{x}_u + M_u a(q)\ddot{q}_a + C_u(q,\dot{q}) + g_u(q) = J_c^T \lambda + J^T F_{ext}} \tag{2.4}$$

The equation (2.4) has to be satisfied only through the action of the contact forces $\boldsymbol{\lambda}$, which are subject to restrictions imposed by the dynamics of contact

forces [23]. Furthermore, these forces have to respect limitations related to friction, along with the condition of *unilaterality* of contact, which restricts the interaction to a pushing action, since two contacts generally can not pull one another.

Separating the contact force into tangential and normal components, $\boldsymbol{\lambda_t}$ and $\boldsymbol{\lambda_n}$, the former condition of *unilaterality* impose that

$$\boldsymbol{\lambda_n} \geq 0 \tag{2.5}$$

while modelling the friction through Coulomb's law generates a limit that can be interpreted as a friction cone

$$||\boldsymbol{\lambda_t}|| \leq \mu_0 \boldsymbol{\lambda_n} \tag{2.6}$$

As a result, for a system to be able to realize a motion $\boldsymbol{q(t)}$ it is required to find a contact force $\boldsymbol{\lambda}$ such that all the condition in (2.4), (2.5), (2.6) are satisfied.

Contacts and Constraints

The contact constraints introduced in the previous section as a set of dynamic conditions that must be satisfied can also be expressed in function of the generalized velocity and accelerations $\dot{q}$ and $\ddot{q}$ through kinematic equations for each contact point C_i [40]

$$\boldsymbol{J_{C_i}\dot{q}} = 0, \quad \boldsymbol{J_{C_i}\ddot{q}} + \boldsymbol{\dot{J}_{C_i}\dot{q}} = 0. \tag{2.7}$$

where J_{C_i} is the constrained Jacobian, mapping the motion of the joints into the motion of the contact point C_i, which is by definition still. In the case of multiple contacts n_c, the Jacobian matrices are stacked into

$$\boldsymbol{J_c} = \begin{bmatrix} \boldsymbol{J_{C_1}} \\ \vdots \\ \boldsymbol{J_{C_{n_c}}} \end{bmatrix} \in \mathbb{R}^{3n_c \times n_{tot}} \tag{2.8}$$

and the number of independent contact constraints is given by its $rank(\boldsymbol{J_c})$. Considering the case of a floating base system, the constrained Jacobian can be divided into two components, $\boldsymbol{J_c^{x_u}}$ and $\boldsymbol{J_c^{q_a}}$, which relate the contact constraints to the base motion and joint motion respectively. From a control point of view, it is possible to distinguish different cases based on the number of contacts n_c, as reported in Figure 2.2:

- If $n_c \leq 2$ the system is said to be *under-actuated*. Analyzing the case of only two legs in contact with the ground, even though the $rank(\boldsymbol{J_c}) = 6$, the

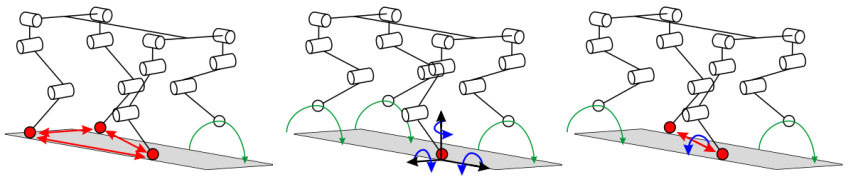


Figure 2.2: Three contact conditions of a floating base robot: in the first on the left the robot has three contact points; the center figure show one contact point, while the other legs are raised with respect to the ground; in the last figure on the right, the robot maintain contact with the ground in two opposing points [40].

$rank(\boldsymbol{J}_{\boldsymbol{c}}^{\boldsymbol{x_u}}) = 5$ because the base is free to move around the axis connecting the two contact points and no actuated joint is able to control this movement around that axis. The reaming $rank(\boldsymbol{J}_{\boldsymbol{c}}^{\boldsymbol{q_a}}) = 1$ is the kinematic internal constraint between the two legs, that cannot be moved one with respect to the other.

- If $n_c = 3$ the system is said to be *fully-actuated* since the $\boldsymbol{J}_{\boldsymbol{c}}^{\boldsymbol{x_u}}$ is full rank, so the action of only the actuated joints is enough to change the position and orientation of the body, included the base.
- If $n_c = 4$ then the system is *over-constrained*, meaning that there are an infinite number of torque combinations to achieve the motion of the robot.

2.2 CENTAURO

In this thesis, the designed controllers were tested on the robot CENTAURO, developed in the Humanoid Human Centered Mechatronic (HHCM) laboratory of the research institute Istituto Italiano di Tecnologia (IIT), in Genova. The idea behind this robot was to implement a general-purpose platform with human-size dimensions, able to perform demanding manipulation tasks, heavy payload transportation with resilience to intense physical interaction. Its design integrates both wheeled and legged locomotion, providing substantial mobility and making it suitable for deployment in hazardous environments, such as disaster sites [24].

The complete weight of the robot is about $118kg$, with a total of 40 controllable joints. The structure is divided into two main components: an upper body whose conformation is similar to the one of a human, in size and form, and a lower body primarily in charge of maintaining balance while crossing a wide variety of terrain.

Figure 2.3: Picture of CENTAURO robot

The first part is mainly applied to perform manipulation tasks through two articulated arms, which try to guarantee similar capabilities as the one of a human. Each arm is actuated in seven-DOF providing the robot with large dexterity in the interactions, with one degree of redundancy to overcome constraints that otherwise would limit the arm movements. In particular, the DOF are distributed in three-DOF shoulder complex, an elbow joint and a three-DOF wrist module. A torso element then connects all these modules. It is mounted on the pelvis edge through a joint permitting the rotation of all upper modules about the yaw axis. The position of the torso, in addition to the inertia of the upper body components, results in a COM moved about 10 cm along the X-axis. Furthermore, the torso carries a wireless communication router and two computation units. The wireless communication router chosen for the system is the Netgear Nighthawk X10 R900. For computation purposes, the system incorporates two main units. Firstly, a COM Express conga-TS170 embedded computer, featuring an Intel Core i7-6820EQ CPU running XENOMAI RT, is utilized to implement the real-time control architecture of the robot. Secondly, a ZOTAC-EN1070K PC equipped with a high-performance GPU, focuses solely on processing perception data. The technical information and the structure design are reported from the publication [24].

The lower body, instead, is a quadruped-on-wheels floating base system. It presents a pelvis in which are located the robot battery, when mounted, the power distribution electronics, and a computational unit based on a ZOTAC-EN1070K PC, performing system high-level control and motion planning. In a complex

system like this one, the most appropriate choice is to divide the weight between four legs, also obtaining better control of the stabilization and orientation of the base. Each leg is divided into a hip-knee-ankle configuration, that resembles the structure of a human's one, with a total of six actuators that allow for 5-DOF kinematics. The first motor is located in the pelvis and controls the hip yaw. Following on the actual leg there are the hip pitch, knee pitch and ankle pitch actuators, ending in an ankle yaw joint to enable active steering of the wheel that provides omni-directional wheeled locomotion. The wheel is then actuated by a motor mounted in the ankle complex, which however does not contribute in any way to the DOFs of the leg [24]. With a total of 6 motors and five DOFs, the kinematic of the leg is underactuated with respect to the Cartesian space. This problem will be further analyzed and addressed in the implementation of the Cartesian impedance controller. It is more evident when studying the kinematic chain of each leg: starting from the base link, attached to the pelvis, and proceeding to the frame of the contact point between the wheel and the ground, the wheel link is not considered as reported in Figure 2.4.

2.3 Framework

The following section provides an overview of the framework developed inside the Humanoids and Human Centered Mechatronics Laboratory, which serves as the foundation upon which the controller proposed in this thesis is built. First, it is presented a real-time multi-thread middleware framework for robotic applications named *XBot2*, a state of the art solution for motion control tasks of complex multi-joint systems such as CENTAURO, with real-time performances, widely described in [31, 27], and a ROS-Based real-time framework to implement online Cartesian control on multi-legged robots, called *CarteI/O*, introduced in the work [28, 18].

2.3.1 XBot2

The software developed in the *XBot2* framework represents a bridge that allows for custom control algorithm to run close to the hardware level of the robotic system with the benefits of low latency in accessing underlying field-bus and a strong focus on modularity and reusability of components. It is written in the programming language C++ and is organized into components and modules. The main ones are the control plugin objects, where the algorithm implemented by the user is contained, and the hardware abstraction layer (HAL), a module that connects the control plugin to a hardware or simulating system.

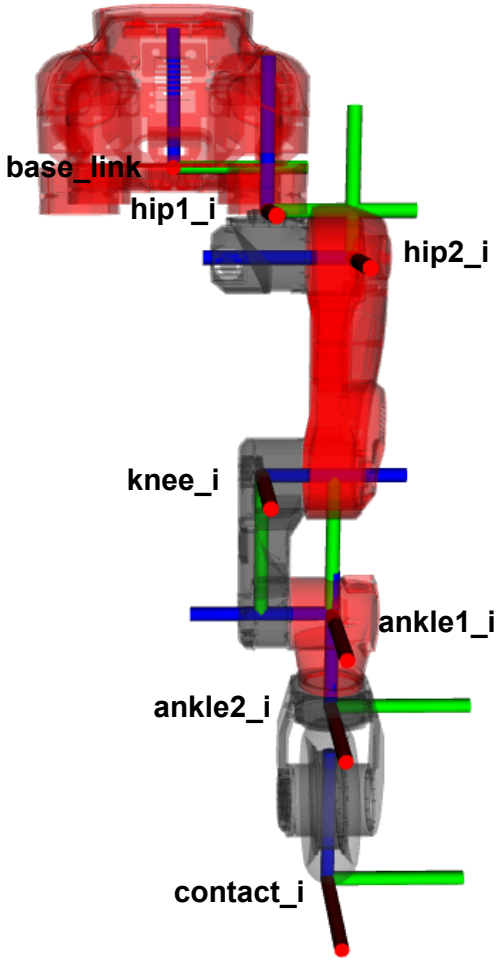


Figure 2.4: Front left leg of CENTAURO obtained from the RVIZ software. The other parts have been removed for a better understanding of the leg structure. On the software, the transparency parameters (alpha) have been set to 0.4.

The Hardware Abstraction Layer (HAL) serves as an interface software, providing users with a streamlined and intuitive means to communicate with external devices, both in real-world and simulated environments. While the user must provide the control plugin to implement control tasks, the objective of the HAL is to establish a connection with the hardware components and to handle the communication between the algorithms and the devices. It has been developed with a client-server approach, where a component called *DeviceDriver* for each device is in charge of the connection and communication with the hardware, acting as the server side, and the second element, the *DeviceClient*, that provides the user

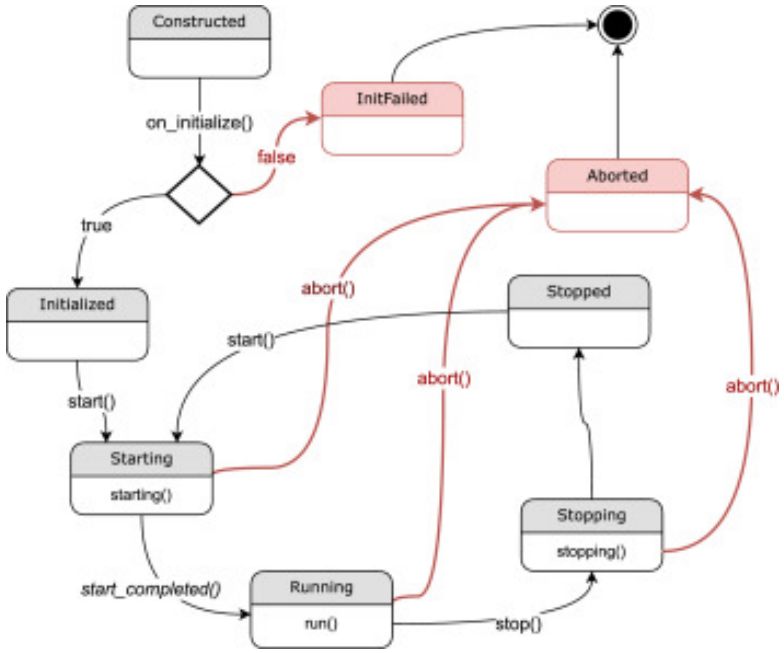


Figure 2.5: Lifecycle finite state machine for *XBot2*'s ControlPlugin

with the proper API.

On the other hand, *XBot2* allows the user to load custom modules to realise desired tasks on the robot at run-time. Each module called *ControlPlugin*, is composed of a finite state machine describing the lifecycle of the algorithm. The set of states and allowed transition is shown in Fig. 2.5. In particular, the construction and initialization are executed before the real-time phase starts and, therefore, are mainly used to allocate all required resources. Instead, the Run state is a periodic task executed at every clock tick and typically holds the repetitive instructions of the control task, like the control laws. Each plugin also contains an instance of the *RobotInterface* object, the bridge to access all the robot components specified in the HAL. The *RobotInterface* class is derived from the *XbotInterface* class, where the robot description and state are stored. Furthermore, the *XbotInterface* class take the information about the structure of the robot from the Universal Robot Description Format (URDF) and the Semantic Robot Description Format (SRDF). In the former are specified the robot kinematic as a tree structure, whereas the latter identifies meaningful branches of the given tree that carry semantic meaning with them.

The *XBot2* framework provides the user with a set of tools to integrate inside a Robotic Operating System (ROS) network. First, a library called *RosSupport*

was implemented to expose information coming from the control plugin in a real-time safe way, for monitoring and logging purposes. Other than such a library, two ready-to-use plugins serve as a ROS API on the robot side: the `ros_io` plugin broadcasts the robot states, and execution statistics to specific topics, while the plugin `ros_crtl` receives commands on appropriate topics, that are forwarded as new joint-state references to the *RobotInterface*.

To facilitate user configuration of *XBot2*, the author has introduced a YAML-based file format. Within this file, users are required to specify several mandatory fields, including:

- The paths to the SRDF and URDF files;
- A list of periodic threads to be initiated by the system, each defined by its name, scheduling policy, priority, and period;
- A list of device types to be dynamically loaded by the system, along with configuration parameters that can be stored in separate files;
- A list of control plugins to be dynamically loaded by the system, comprising its type, name, parameters, and the type of thread on which it will operate.

This structured approach simplifies the configuration process, allowing users to tailor the framework to their specific requirements. In Appendix A.3 is shown the configuration file used in this thesis

During runtime, users must have the capability to quickly initiate or halt the execution of control plugins, while simultaneously monitoring the robot's state data. While the previously described ROS tools offer a solid foundation for interacting with robots, executing commands or reading data via the terminal can still be cumbersome. To resolve this issue, a *Xbot-GUI* was developed, providing an intuitive interface to effortlessly fulfil these functions. Through this GUI, is possible to plot and check information about joint position, velocity, torque and temperature, interact with the plugins and command new references to each joint on the kinematic tree. Figure 2.6 shows an example of the GUI used during the implementation of the control strategies.

2.3.2 CartesIO

If from one side *Xbot2* offers a robust structure to interface the user with the hardware mounted on a robot, it does not include any functions related to the actual motion control. Therefore, the framework *CarteI/O* has been developed with the intent to allows untrained users to perform complex motion tasks in

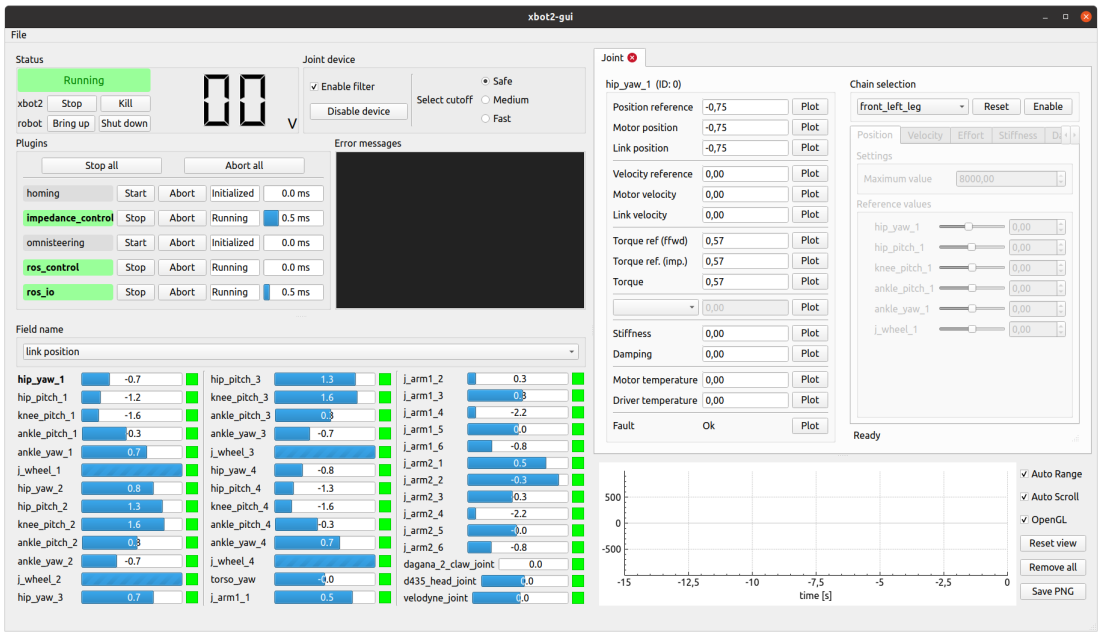


Figure 2.6: Screenshot of the Xbot GUI

Cartesian space on a highly redundant, floating base robot. This framework is divided into several components:

- the *solver* component, whose objective is computing the matrices and vectors that define the optimization problem and subsequently solving a single instance of such a problem. It is based on a Quadratic Program (QP) solver.

- a modelling language that is used to construct the mathematical problem in a higher-level way. The chosen language for this framework is the *OpenSoT* library, a whole-body inverse kinematic engine developed in C++, at the velocity level. Through the use of simple syntax called *Math Of Task*, is possible to formulate a problem composed of several tasks with different priorities, constraints and solvers. In this case, the configuration of the tasks is provided through YAML files in which a stack of problems is specified.

- A base class called *CartesianInterface* which handles the low-level and complex work of interacting and managing the tasks, through a ROS API and a set of methods available to the user, who can choose to override these functions in order to implement their own Cartesian controller. Here the state of the robot is stored inside the *ModelInterface* class, from the *XbotInterface* package, via kinematic and dynamic quantities updated by the solution of the mathematical problems.

- a middleware interface that allows the output of the *CartesianInterface*, so the solution of the inverse kinematic and dynamic problem, to be sent as new references to the actual robot hardware, maintaining real-time performances. This middleware will be the previously described *XBot2* framework, making use of the *RobotInterface* to communicate the robot state and set points to the control PC mounted directly on the robot.

2.4 Gazebo

Gazebo is an Open Source robotics simulator developed by the Open Source Robotics Foundation, that allows to simulate of the dynamical behaviour of complex robotic systems with high fidelity. Through this tool is possible to create a detailed virtual model of the robot and the environment where it is going to operate, providing valuable insight into the controller behaviour across different scenarios. It is composed of two parts: the `gzserver` executable, which runs the physics update-loop and sensor data generation, and the `gzclient` executable which provides the graphical interface to interact with the simulation. Moreover, in Gazebo the modelling of the interaction between the robots and the surroundings, as well as all the graphical rendering, are handled by the physics engines. At the moment there are four different physics engines supported by Gazebo: ODE (the default one), DART, Buller and Simbody. In this thesis, it is kept as the default engine. The simulation containing both the robots and the environment takes place inside a so called *world*, whose configuration is specified by an XML file, with the extension '.world'. Inside this file is possible to customize the parameters of the physical engine, and load models such as robots, ground planes, objects and more. Other than the world file, the graphical interface allows for convenient upload in the simulation of external models defined as SDF files. Each model is described in this file through parameters about the aspect and more important ones like the collision geometry. With this parameter is possible to change the physical behaviour of the object when undergoing an interaction of any kind. This model can be defined from elementary forms (box, cylinder, sphere) or more complex shapes crafted with 3D modelling tools. The *XBot2* framework allows for an integration of the control plugin into Gazebo thanks to the HAL system that serves as a client with respect to the simulator.

Chapter 3

Cartesian Impedance controller

Contents

The following chapter will describe the overall implementation of the Cartesian Impedance controller. First, an introduction with the motivation of the choice for this controller is presented, moving to a more detailed explanation of the theory behind the implementation. From the dynamical law governing impedance control, it is found the control law in the operational space that computes the torque to apply on the motors. Then a complete description of the algorithm implementation is presented.

3.1 Introduction

When the robot moves on unstructured terrain, it is subjected to forces which act on the contact points between the robot and the environment, due to the presence of obstacles and deformities in the surface. To address these forces, it is required an interaction control to comply with these forces. This type of control can be achieved through a passive or active action, or a hybrid combination of both. In the first case, physical elements, like springs and dampers, are mounted into the components which interact with the forces, resulting in passive compliance. However, this method poses limitations on adaptability due to fixed stiffness and damping settings, and it may require complicated structural design to accommodate these physical elements. On the other hand, active compliance employs

virtual systems comprising virtual springs and dampers, offering the flexibility to adjust parameters in real-time to adapt the system's response to external interactions. Control strategies for achieving active compliance involve modelling the dynamics of such systems and integrating them into the robot's dynamics. There exist several types of controllers used to enable active compliance: impedance controller [19], admittance controller [30], hybrid position/force controller [36]. The impedance control falls into the category of indirect force control as explained in [50]. This classification implies that the control input is not a force feedback, but rather the control is achieved through motion sensing. In particular, if the interaction with the environment causes a manipulator to deviate from its desired position, it will generate a force proportional to the deviation, with the intent to restore the planned motion. This force-motion relationship mirrors that of a mass-spring-damper system, which is a typical second-order ordinary differential equation (ODE), where the parameters can be tuned to change the response behaviour. The theory of such a system is detailed in Appendix A.2. Admittance control serves as the complement to impedance control [1]. It typically focuses on regulating the motion of the robot's end-effector in response to external forces. Here, the external force is translated into new reference positions for the robot through a mass-spring-damper virtual model. These new references are then fed to a position controller, typically a Proportional-Derivative (PD) controller, which subsequently moves the robot to the desired position [30, 33]. The hybrid position/force control action is based on a relation linking the desired force to the actual one resulting from the interaction. This relation is typically an equation which explicitly models the interactions between the end-effector and environment, which in this case is not straightforward to obtain due to the high number of disturbances coming from rough terrain. Even though the already developed framework was equipped with a tool to estimate the contact force via the torque sensors mounted on the motors, a first strategy for leg control could be a simpler impedance control, that offers a more practical solution to regulate the robot's compliance with the environment, thereby mitigating the challenges posed by uncertain terrain conditions.

In the case of controlling the legs of quadruped-on-wheel platforms navigating rugged terrain, the theoretical distinctions between joint space impedance control and Cartesian impedance control play a crucial role in determining the effectiveness of the control approach. Joint space impedance control facilitates the manipulation of stiffness and damping properties at the individual joint level, affording precise control over each joint's motion. However, this method may encounter difficulties in coordinating movements across multiple joints concurrently, thereby posing challenges in maintaining stability and adaptability over uneven surfaces. Conversely, Cartesian impedance control operates directly within the

robot's task space, regulating interaction forces of the contact points. This approach facilitates coordination across all degrees of freedom, allowing the robot to dynamically adjust its posture and compliance to suit the terrain. Additionally, Cartesian impedance control offers a more intuitive framework for implementing intricate motion trajectories and navigating obstacles, thereby enhancing the quadruped robot's overall performance and resilience on challenging terrains.

3.2 Theoretical Basis of the Controller

Let's start by reporting the equation of motion expressed in Eq. (2.3) in a compact form

$$\boldsymbol{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - J^T F_{ext} + J_c^T F_c} \tag{3.1}$$

where $\boldsymbol{J^T}$ and $\boldsymbol{F_{ext}}$ are composite matrices which define a stack of Jacobian and the corresponding wrench acting on the system. A possible way to define these matrices is

$$\boldsymbol{J^T F_{ext}} = \begin{bmatrix} \boldsymbol{J_1^T} & \boldsymbol{J_2^T} & \boldsymbol{J_3^T} & \boldsymbol{J_4^T} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{F_{ext_1}} \\ \boldsymbol{F_{ext_2}} \\ \boldsymbol{F_{ext_3}} \\ \boldsymbol{F_{ext_4}} \end{bmatrix} \tag{3.2}$$

In this case, the force $\boldsymbol{F_{ext_i}}$ is considered to be the wrench acting on the end-effector of the i-th leg, while $\boldsymbol{J_i^T}$ refers to the *Relative Jacobian*, defined in $\mathbb{R}^{(n_b+n_j)\times 6}$. It represents a Jacobian matrix where only the rows relative to the joints of the i-th legs are considered. In particular, as detailed in the previous chapter, each leg is defined by a chain starting from the base link frame and concluding in the contact link frame. The joints within this chain are related to the motion of the end-effector through the relative Jacobian. Consequently, all rows corresponding to joints outside of this chain are set to zero, as they are not part of the considered kinematic structure. Thus, $\boldsymbol{J_i^T}$ accounts for the effect of external forces solely on the end effector of that chain. Similarly, $\boldsymbol{J_c^T F_c}$ contain a stack of constraint Jacobian $\boldsymbol{J_{c_i}^T}$, introduced in Section 2.1, which follows the same principle. It considers only the effect of constraints on the joints within the chain, ensuring that external constraints are accounted for appropriately. It is possible to define a new variable $\boldsymbol{y} \in \mathbb{R}^{(n_b+n_j)\times 1}$ as

$$\boldsymbol{y = \tau + J_c^T F_c - g(q)} \tag{3.3}$$

In this way, the new torque vector $\boldsymbol{y}$ will contain an offset value corresponding to the contribution of the contact forces, allowing the robot in static condition

to exert on the joints of the legs a certain amount of torque able to counteract the gravity acceleration imposed on the robot weight. Therefore, the previous equation of motion becomes

$$\boldsymbol{M(q)\ddot{q} + C(q,\dot{q})\dot{q} = y - J^T F_{ext}} \tag{3.4}$$

The resulting torque vector $\boldsymbol{y}$ consists of 46 elements, comprising the $n_b = 6$ joints of the base, which are set to zero as they are non-actuated, and the $n_j = 40$ active joints of the system.

Now, considering the contact as the end-effector of the i-th leg chain, its motion can be expressed with respect to the base frame using the spatial vector notation $\boldsymbol{x_i} = [\boldsymbol{p}^T\ \boldsymbol{\theta}^T]^T \in \mathbb{R}^{6\times 1}$, which are the linear position and orientation of the frame, accounting a total of six-DOF. The mapping of the Cartesian velocity and acceleration into the corresponding generalized joint variables $\boldsymbol{q}$, introduced in Section 2.1, can be computed using the relative Jacobian matrix $\boldsymbol{J_i^T(q)} \in \mathbb{R}^{6\times(n_b+n_j)}$ as

$$\begin{aligned} \boldsymbol{\dot{x}_i} &= \boldsymbol{J_i\dot{q}}, \\ \boldsymbol{\ddot{x}_i} &= \boldsymbol{J_i\ddot{q} + \dot{J}_i\dot{q}}. \end{aligned} \tag{3.5}$$

The same spatial vector notation is used in the second-order dynamic system that describes the controller behaviour as

$$\boldsymbol{\Lambda_d\ddot{\tilde{x}} + D\dot{\tilde{x}} + K\tilde{x} = F_{ext}} \tag{3.6}$$

where $\ddot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{x}}}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{6\times 1}$ are respectively the error between the actual and desired acceleration, velocity, and position, expressed as $\tilde{\boldsymbol{x}} = (\boldsymbol{x}_{desired} - \boldsymbol{x}_{real})$ in spatial vector notation, while $\boldsymbol{\Lambda_d} \in \mathbb{R}^{6\times 6}$ is the desired inertia matrix in the Cartesian space. $\boldsymbol{D, K} \in \mathbb{R}^{6\times 6}$ are diagonal matrices expressing the damping and the stiffness coefficient in each one of the axis directions. $\boldsymbol{F}_{ext} \in \mathbb{R}^{6\times 1}$ is the wrench of the external force acting on the end-effector and causing the deviation from its planned motion.

To link the equation of motion (3.4) with the dynamic in Eq. (3.6), it is clear the need for the corresponding operational space equation of motion that introduces a relation between the end-effector motion and the torque to be applied at each joint. Using a similar approach as the one used in [44], it is possible to isolate the generalized acceleration $\boldsymbol{\ddot{q}}$ from the joint space equation of motion (3.4)

$$\boldsymbol{\ddot{q} = M^{-1}\left(y - J^T F_{ext} - C(q,\dot{q})\dot{q}\right)} \tag{3.7}$$

where the inertial matrix $\boldsymbol{M}$ is symmetric and positive definite, so invertible. Substituting Eq. (3.7) into the second equation of (3.5) is possible to introduce the end-effector acceleration, obtaining

$$\ddot{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{J}_i \boldsymbol{M}^{-1} \left(\boldsymbol{y} - \boldsymbol{J}^{\boldsymbol{T}} \boldsymbol{F}_{\boldsymbol{ext}} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} \right) + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} \tag{3.8}$$

where $\boldsymbol{J_i}$ is the same *Relative Jacobian* introduces earlier. Analysing the terms reported in Eq. (3.8), it is possible to compute some simplification. In particular, expanding the term $\boldsymbol{J_i M^{-1} J^T}$ using the Eq. (3.2), it can be written as

$$\begin{bmatrix} \boldsymbol{J_i M^{-1} J_1^T} & \boldsymbol{J_i M^{-1} J_2^T} & \boldsymbol{J_i M^{-1} J_3^T} & \boldsymbol{J_i M^{-1} J_4^T} \end{bmatrix}^T \tag{3.9}$$

It can be demonstrated through simple numerical example, that the term $\boldsymbol{J_i M^{-1} J_j^T}$, with $\vee j \in 1, 2, 3, 4$, is null for every $j \neq i$. With this consideration, the Eq. (3.8) can be simplified with only the contribution of the external force $\boldsymbol{F_{ext_i}}$ acting on the end-effector of the i-th leg. The new equation is

$$\ddot{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{J}_i \boldsymbol{M}^{-1} \left(\boldsymbol{y} - \boldsymbol{J}_i^{\boldsymbol{T}} \boldsymbol{F}_{\boldsymbol{ext}_i} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} \right) + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} \tag{3.10}$$

After rearranging the Eq. (3.10) by multiplying the terms inside the main parenthesis, the modified equation will expose the operational space inertia matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{6\times6}$, computed as

$$\boldsymbol{\Lambda} = \left(\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{J}_i^{\boldsymbol{T}} \right)^{-1} \tag{3.11}$$

Eq. (3.10) then become

$$\boldsymbol{\Lambda} \ddot{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{\Lambda J}_i \boldsymbol{M}^{-1} \boldsymbol{y} - \boldsymbol{F}_{\boldsymbol{ext}_i} - \boldsymbol{\Lambda J}_i \boldsymbol{M}^{-1} \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} + \boldsymbol{\Lambda} \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} \tag{3.12}$$

where the dependence on $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ can be omitted with the following change in variables

$$\boldsymbol{C_{op}}\ \dot{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{\Lambda J}_i \boldsymbol{M}^{-1} \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} + \boldsymbol{\Lambda} \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}} \in \mathbb{R}^{6\times1} \tag{3.13}$$

which map the Coriolis and gravitational contributions, introduces in the joint space equation of motion (2.3), into the corresponding effect on the end-effector motion. Additionally, from Eq. (3.12), it is possible to define the dynamically consistent right pseudo-inverse of Jacobian $\overline{\boldsymbol{J}_{\boldsymbol{i}}}^{\boldsymbol{T}}$ as

$$\overline{\boldsymbol{J}_{\boldsymbol{i}}}^{\boldsymbol{T}} = \boldsymbol{\Lambda J}_i \boldsymbol{M}^{-1} \tag{3.14}$$

Substituting Eq. (3.13) and (3.14) into Eq. (3.12), the new equation will be

$$\begin{aligned} \left(\boldsymbol{\Lambda} \ddot{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{C_{op}}\ \dot{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{F_{ext}} \right) &= \overline{\boldsymbol{J}_{\boldsymbol{i}}}^{\boldsymbol{T}} \boldsymbol{y} \\ &= \boldsymbol{F_y} \end{aligned} \tag{3.15}$$

where $\boldsymbol{F_y}$ is the wrench of the force exerted by the end-effector, corresponding to the torque $\boldsymbol{y}$ at each joint of the considered chain.

As pointed out by [26], the Jacobian pseudo-inverse $\overline{\boldsymbol{J_i}}^{\boldsymbol{T}}$ offers a solution to the inverse kinematic problem for redundant manipulators, which are robots with a number of joints higher than the dimension of the space where they operate. It is used to define a mapping to the null-space of $\overline{\boldsymbol{J}}$, ensuring that vectors mapped within this space do not induce any changes in the position and orientation of the end-effector. On the other hand, [32] investigates the generalized pseudo-inverse in the case of singularities. When the first equation of Eq. 3.5 is used to solve the inverse kinematic problem, finding the joint motion related to the end-effector displacement, the inverse of the Jacobian that appears in the equation exists only if there are no singular points. Otherwise, in the presence of singularities which cause $rank(\boldsymbol{J}) = 0$, the pseudo-inverse is used to provide an approximated solution, which corresponds to performing a least-squares regression problem of the type

$$\min_{\delta \boldsymbol{q}} ||\delta \boldsymbol{x} - \boldsymbol{J}(\boldsymbol{q})\delta \boldsymbol{q}|| \tag{3.16}$$

which minimizes the error between the desired end-effector displacement, $\delta \boldsymbol{x}$, and the one caused by joint motion $\delta \boldsymbol{q}$. Transitioning into the dynamic domain, the objective is to solve the inverse dynamic problem in operational space, therefore computing the joint torque $\boldsymbol{\tau}$ that is needed to generate a specific motion at the end-effector, due to a force applied at the end-effector. In the case of a highly redundant robot, such as CENTAURO, the number of possible torque combinations which solve this problem is infinite. It is required to select a single set of torques based on certain criteria. Applying the solution described in [44], the inverse dynamic problem is solved as

$$\boldsymbol{y} = \boldsymbol{J_i^T}(\boldsymbol{q})\boldsymbol{F_y} + \left(\boldsymbol{I_n} + \boldsymbol{J_i^T}(\boldsymbol{q})\overline{\boldsymbol{J_i}}^{\boldsymbol{T}}(\boldsymbol{q})\right)\boldsymbol{\tau_0} \tag{3.17}$$

where $\overline{\boldsymbol{J_i}}^{\boldsymbol{T}}$ is considered the right pseudo-inverse of $\boldsymbol{J_i^T}$ weighted by $\boldsymbol{M}(\boldsymbol{q})$, and the operator $\left(\boldsymbol{I_n} + \boldsymbol{J_i^T}(\boldsymbol{q})\overline{\boldsymbol{J_i}}^{\boldsymbol{T}}(\boldsymbol{q})\right)$ maps all the torques $\boldsymbol{\tau_0}$ into the null space of $\overline{\boldsymbol{J_i}}^{\boldsymbol{T}}$, not causing any motion at the end-effector. Not considering the torque $\boldsymbol{\tau}_0$, the corresponding command vector $\boldsymbol{y_i}$ produces a manipulated joint motion that minimizes the instantaneous kinetic energy of the system. Therefore, substituting Eq. (3.15) into Eq. (3.17), the torques vector is computed based on the contribution of all the forces acting on the system as

$$\boldsymbol{y} = \boldsymbol{J_i^T}\left[\boldsymbol{\Lambda}\ddot{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{C_{op}}\,\dot{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{F_{ext_i}}\right] \tag{3.18}$$

The resulting torque vector $\boldsymbol{y} \in \mathbb{R}^{(n_b+n_j \times 1}$, due to the presence of the relative Jacobian $\boldsymbol{J_i^T}$, will hold the information about the torque to be applied at the joints of the i-th leg. Therefore, all the other terms inside that vector are send to zero. Note that, for the seek of understanding, this vector will be called $\boldsymbol{y_i}$ from now on. Since the objective is to force the dynamic of the leg to act as a mass-spring-damper system, the external forces $\boldsymbol{F_{ext}}$ in Eq. (3.18) will be considered the same as the forces that cause the deviation in Eq. (3.6). Substituting the two equation

$$\boldsymbol{y_i} = \boldsymbol{J_i^T} \left[\boldsymbol{\Lambda \ddot{x}_i} - (\boldsymbol{\Lambda_d \ddot{x}_r} - \boldsymbol{\Lambda_d \ddot{x}_d} + \boldsymbol{D\tilde{\dot{x}}} + \boldsymbol{K\tilde{x}}) + \boldsymbol{C_{op}}\ \boldsymbol{\dot{x}} \right] \tag{3.19}$$

Assuming to not perform inertia shaping on the system, the matrix $\boldsymbol{\Lambda_d}$ will be equal to $\boldsymbol{\Lambda}$, so two terms regarding the end-effector acceleration can cancel each other, losing the dependency on the acceleration. Also, substituting the external force component (3.6) into (3.18) shows that the actuation torques will not depend upon the value of the interaction force. The final closed-loop control law is computed as

$$\boldsymbol{y_i} = \boldsymbol{J_i^T} \left[\boldsymbol{\Lambda_d \ddot{x}_d} - (\boldsymbol{D\tilde{\dot{x}}} + \boldsymbol{K\tilde{x}}) + \boldsymbol{C_{op}}\ \boldsymbol{\dot{x}} \right] \tag{3.20}$$

This control law therefore will compute the torques for each joint of the i-th leg. To create the complete controller, the motion of each leg has to be dictated by a singular Eq. (3.20), controlling the movements of the corresponding end-effector. Summing the torque contribution of each $\boldsymbol{y_i}$, is possible to obtain the complete torque vector $\boldsymbol{y}$ which mimic the dynamic of the mass-spring-damper system for each leg. Plugging $\boldsymbol{y}$ inside Eq. (3.3) allows to compute the resulting torque $\boldsymbol{\tau}$ that will be send to the robot as

$$\boldsymbol{\tau} = \boldsymbol{g(q)} - \boldsymbol{J_c^T F_c} + \sum_{i=1}^{4} \boldsymbol{y_i} \tag{3.21}$$

Computation of operational space matrix $\boldsymbol{\Lambda}$

As reported in Section 2.2 a robot leg operates with 5 degrees of freedom (DOF) while moving in a Cartesian space with 6 DOF. Consequently, the Jacobian matrix presents an entire null row corresponding to the joint linking the ankle yaw frame with the contact link. This linearly dependent row in the Jacobian causes the matrix rank to be not full and, therefore, non-invertible [49]. In the Eq. 3.11, the multiplication of $\boldsymbol{M}$, which is symmetric, positive definite and invertible, times the singular Jacobian $\boldsymbol{J_i}$ will result in a term $\boldsymbol{J_i M^{-1} J_I^T}$ that is not invertible and possesses an eigenvalue approximately equal to zero. Given the necessity to compute the inverse of $\boldsymbol{J_i M^{-1} J_i^T}$ to derive the operation space matrix, one

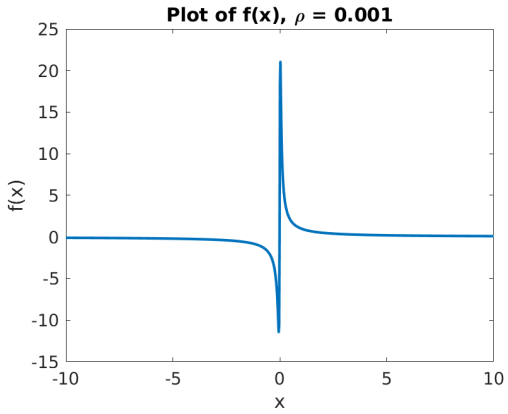


Figure 3.1: Function $f(x)$ used to compute the pseudo-inverse matrix $S^{\dagger}$, with $\rho = 0.001$

viable approach is to employ Singular Value Decomposition (SVD) to compute the inverse of this matrix [48]. Defining the matrix $\boldsymbol{L} = \boldsymbol{J_i M^{-1} J_i^T}$, it can be factored as

$$\boldsymbol{L} = \boldsymbol{U S V^T} \tag{3.22}$$

where U and V are orthogonal matrices called the left singular vectors and right singular vectors respectively, while S is a diagonal matrix whose diagonal elements are the eigenvalues of the matrix $\boldsymbol{L}$. The pseudo-inverse matrix is then computed as

$$\boldsymbol{L^{\dagger}} = \boldsymbol{V S^{\dagger} U^T} \tag{3.23}$$

where each element of the matrix $S^{\dagger}$ diagonal has been computed with the function

$$f(x) = \frac{(x + 0.01)}{\rho + x^2} \tag{3.24}$$

which is designed to approach zero as x tends to zero, while approximating $\frac{1}{x}$ for $x \neq 0$. Figure 3.1 plots the graph of Eq. 3.24. Forging the pseudo inverse with this function will ensure the resulting matrix possesses positive eigenvalues, guaranteeing positive definiteness. Also, it can be demonstrated that $\boldsymbol{L}$ will maintain the property of symmetry of the matrix $\boldsymbol{M}$. Furthermore, it can be demonstrated that $\boldsymbol{L}$ maintains the symmetry property of matrix $\boldsymbol{M}$. The computed pseudo-inverse matrix, $\boldsymbol{L^{\dagger}}$, is then assigned to the operation space matrix $\boldsymbol{\Lambda}$, inheriting its symmetric and positive definite properties.

Estimation of contact forces

When the robot is in static conditions, the gravitational acceleration generates a force with a magnitude proportional to the mass of the robot, directed toward the ground along the Z-axis of the inertial frame, and applied to all the system components. This force is then distributed across all the contact points depending on the COM position. If not correctly compensated in the overall dynamic, these forces push the robot to fall freely toward the ground. In this case, the only opposite reaction is done by the forces generated due to the virtual spring displacement of each leg's Cartesian impedance controller. The static condition is reached when the generated wrench is equal and opposite to each contact force. This equilibrium point is different based on the spring stiffness, the lower the stiffness, the higher the displacement. A solution is then to compute the contact wrenches exerted by the ground at the contact point of each leg, which compensates for the gravity acceleration without the contribution of the Cartesian controller, as reported in Chapter 5.2 of [29]. A possible solution is presented by [7]

Starting from Eq. 2.3, in static conditions it can be simplified into

$$\boldsymbol{g(q)} = \begin{bmatrix} \mathbf{0_{6\times 1}} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J_{c_1}}^T & \boldsymbol{J_{c_2}}^T & \boldsymbol{J_{c_3}}^T & \boldsymbol{J_{c_4}}^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{F_{c_1}} \\ \boldsymbol{F_{c_2}} \\ \boldsymbol{F_{c_3}} \\ \boldsymbol{F_{c_4}} \end{bmatrix} \tag{3.25}$$

where the contact Jacobian $\boldsymbol{J_{c_i}^T} \in \mathbb{R}^{46\times 6}$ map the contact wrench $\boldsymbol{F_{c_i}} \in \mathbb{R}^{6\times 1}$, expressed in world frame, into the corresponding joint torque needed to generate the reaction forces and $\boldsymbol{g(q)} \in \mathbb{R}^{46\times 1}$ are the gravitational torque acting on the system. The Eq. 3.25 can be split into two parts, the first one regarding the floating base joints and the second about the actuated joints. The former part, composed of a total of six equations conveniently writes in the following form

$$\boldsymbol{g_{fb}(x_u)} = \begin{bmatrix} \boldsymbol{J_{c_1}}_{fb}^T & \boldsymbol{J_{c_2}}_{fb}^T & \boldsymbol{J_{c_3}}_{fb}^T & \boldsymbol{J_{c_4}}_{fb}^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{F_{c_1}} \\ \boldsymbol{F_{c_2}} \\ \boldsymbol{F_{c_3}} \\ \boldsymbol{F_{c_4}} \end{bmatrix} \tag{3.26}$$

where $\boldsymbol{g_{fb}(x_u)}\mathbb{R}^{6\times 1}$ is the vector that holds the information about the gravitational torque exerted on the floating base joints. In a conventional fixed-base system, an element $g_i(q)$ represents the torques generated at the axis of joint i in the current configuration, caused by the inertia of all the consecutive joints and links that follow in the kinematic chain. In the case of a floating base, the virtual joints connecting the base to the inertia frame are affected by the inertia of all the bodies in the system. For instance, the configuration of the vector $\boldsymbol{g_{fb}(x_u)}$

in static conditions, with the pelvis parallel to the ground, is expected to be zero for all the values except for the linear joint along Z-axis, whose value is about the mass of the robot times the gravity acceleration, and the rotational joint about Y-axis due to the decentralized COM. However, other configurations will result in non-zero values on other vector components. Lastly, the term ${\boldsymbol{J}_{\boldsymbol{c_1}_{\boldsymbol{fb}}}}^T \in \mathbb{R}^{6\times6}$ consists of the first six rows of the contact Jacobian matrix referring to the first six joints of the floating base. The wrench $\boldsymbol{F_{c_i}}$ represent the force exchanged between the i-th contact point and the ground, expressed with respect to the world frame. Its magnitude depends on both the number of contacts and the distribution of the robot's weight, being positive to adhere to the unilateral constraint. On flat terrain, the resulting direction aligns along the Z-axis, rendering all terms of the contact wrench zero except for the third element, representing the vertical direction. However, on non-flat surfaces, the contact force may consist of a normal component, perpendicular to the surface's normal, alongside a tangential component accounting for friction. These components adhere to the friction cone relation. The resulting contact force isn't purely vertical and will exhibit components along the X and Y axes. To simplify the computation of these forces, one plausible assumption is to assume the surface's normal vector is always parallel to the vertical direction of the inertia frame. Consequently, the contributions of the longitudinal and lateral components of the contact force can be disregarded. This assumption streamlines the estimation of the contact force, eliminating the need for sensors (e.g., cameras) to estimate the surface normal and the friction coefficient to study the limitations imposed by the friction cone. The solution to the problem then corresponds to minimizing the norm of the contact forces. Starting from this assumption, the new system of equations is

$$\boldsymbol{g_{fb}(x_u)} = \boldsymbol{J_{c_{fb}}}(z)\boldsymbol{F_c}(z) \tag{3.27}$$

where $\boldsymbol{J_{c_{fb}}}(z) \in \mathbb{R}^{6\times4}$ is a matrix containing four columns, each one corresponding to the third column of the world contact Jacobian ${\boldsymbol{J_{c_i}}_{\boldsymbol{fb}}}^T$ that refer only to the z-component $\boldsymbol{F_{c_i}}(z)$ of the contact wrench. From the information about the state of the model is possible to obtain the values of the Jacobians and the gravity compensation torque. The unknown variable can be computed by solving the linear system for $\boldsymbol{F_c}(z)$, which requires the definition of a pseudo-inverse since $\boldsymbol{J_{c_{fb}}}(z)$ is rectangular and not invertible. The resulting system is

$$\boldsymbol{F_c}(z) = {\boldsymbol{J_{c_{fb}}}}^{\dagger}(z)\boldsymbol{g_{fb}(x_u)} \tag{3.28}$$

where the pseudo-inverse serves to find the vector $\boldsymbol{F_c}(z)$ that minimizes the squared differences between the actual output $\boldsymbol{g_{fb}(x_u)}$ and the product of $\boldsymbol{J_{c_{fb}}}(z)$ with $\boldsymbol{F_c}(z)$. The expected result will be an uneven distribution of the contact forces, higher for the two front legs, and consequently lower on the back. These

forces are then plugged inside the control law 3.20 to be converted into torque at each joint.

Computation of damping matrix $\boldsymbol{D}$

The publication [2] related to the development of a Cartesian Impedance control for lightweight manipulators has studied the stability of the control law 3.20, demonstrating that the damping matrix $\boldsymbol{D}$ does not contribute to the stability of the system, although it is involved in the energy dissipation. Therefore is possible to find a relation to compute a variable damping matrix depending on the desired stiffness and the configuration of the system, supported by the theory in Appendix A.2.

The authors of [2] propose an approach based on the resolution of a generalized eigenvalue problem [15]:

> Given a symmetric positive definite $n \times n$ matrix $\boldsymbol{\Lambda}$ and a symmetric $n \times n$ matrix $\boldsymbol{K}$, it exists a non-singular $n \times n$ matrix $\boldsymbol{Q}$ such that $\boldsymbol{\Lambda} = \boldsymbol{Q}\boldsymbol{Q}^T$ and $\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{K}_\omega\boldsymbol{Q}^T$ for some diagonal $n \times n$ matrix $\boldsymbol{K}_\omega$.

The matrix $\boldsymbol{\Lambda}$ refers to the actual operational space matrix and $\boldsymbol{K}$ is the diagonal matrix of the stiffness coefficients that can be chosen arbitrarily. To determine the matrix $\boldsymbol{Q}$, it is necessary to convert the two relations introduced in the problem definition into the conventional form

$$\boldsymbol{A\Theta} = \boldsymbol{B\Theta D} \tag{3.29}$$

where $\boldsymbol{\Theta}$ and $\boldsymbol{D}$ are respectively the matrix of generalized eigenvectors and the diagonal matrix containing the generalized eigenvalues. For this purpose, let's introduce a new matrix $\boldsymbol{X} = (\boldsymbol{Q}^T)^{-1}$. Since $\boldsymbol{Q}$ is non-singular by definition, it is invertible. Substituting the new matrix inside the

$$\begin{aligned} \boldsymbol{\Lambda} &= \boldsymbol{Q}\boldsymbol{Q}^T, & \boldsymbol{K} &= \boldsymbol{Q}\boldsymbol{K}_\omega\boldsymbol{Q}^T \\ \boldsymbol{\Lambda}\boldsymbol{X} &= \boldsymbol{Q}, & \boldsymbol{K}\boldsymbol{X} &= \boldsymbol{Q}\boldsymbol{K}_\omega \\ \boldsymbol{\Lambda}\boldsymbol{X} &= \boldsymbol{Q}, & \boldsymbol{K}\boldsymbol{X} &= \boldsymbol{\Lambda}\boldsymbol{X}\boldsymbol{K}_\omega \\ \boldsymbol{X}^T\boldsymbol{\Lambda}\boldsymbol{X} &= \boldsymbol{I} & \boldsymbol{K}\boldsymbol{X} &= \boldsymbol{\Lambda}\boldsymbol{X}\boldsymbol{K}_\omega \end{aligned} \tag{3.30}$$

where the first equation is the demonstration of the orthogonality property of each vector $\boldsymbol{x}_i$ of the matrix $\boldsymbol{X}$, with respect to the inner product $\boldsymbol{x}_i\boldsymbol{\Lambda}\boldsymbol{y}$. The second equation represents the generalized eigenvalue problem, that can be solved to obtain the matrix $\boldsymbol{X}$ and $\boldsymbol{K}_\omega$, corresponding to the matrix of generalized eigenvectors and diagonal matrix of eigenvalues. The solution has been implemented

based on the detailed instruction specified in [13]. Once these matrices are computed, it can be chosen to define the matrix $\boldsymbol{D}$ as

$$\boldsymbol{D} = 2\boldsymbol{Q}\boldsymbol{D}_{\zeta}\sqrt{\boldsymbol{K}_{\omega}}\boldsymbol{Q}^{T} \tag{3.31}$$

where $\boldsymbol{D}_{\zeta} \in \mathbb{R}^{6\times 6}$ is a diagonal matrix specifying the damping ratio in each Cartesian direction.

Computation of errors in the mass-spring-damper dynamic

As previously mentioned, in the mass-spring-damper dynamics described in Eq. (3.6), the variables $\tilde{\boldsymbol{x}}$, $\dot{\tilde{\boldsymbol{x}}}$, and $\ddot{\tilde{\boldsymbol{x}}}$ represent the error between the desired motion and the actual motion of the i-th leg's end-effector. Given their spatial notation, each error can be decomposed into two distinct components

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_{\boldsymbol{P}} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{O}} \end{bmatrix}, \qquad \dot{\tilde{\boldsymbol{x}}} = \begin{bmatrix} \dot{\tilde{\boldsymbol{x}}}_{\boldsymbol{P}} \\ \dot{\tilde{\boldsymbol{x}}}_{\boldsymbol{O}} \end{bmatrix}, \tag{3.32}$$

Here, the notation **P** denotes the position component, while **O** refers to the orientation component. It's worth noting that the acceleration error is not considered, as it is simplified in Eq. (3.20). Regarding the position error, computation is straightforward as both position and its time derivative can be obtained by subtracting the desired and actual end-effector positions. The same applies to the velocity error. However, computing the orientation error, and consequently, its time derivative, requires additional steps depending on the representation of the end-effector orientation, which can be in Euler angles, angle and axis representation, or unit quaternions [44]. In our case, since the orientation of the contact link is expressed in the rotational matrix at the coding level, the simpler solution was to use the angle-axis method to compute the error, reducing the required computations. Considering the real rotational matrix $\boldsymbol{R}_{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{n}_{\boldsymbol{r}} & \boldsymbol{s}_{\boldsymbol{r}} & \boldsymbol{a}_{\boldsymbol{r}} \end{bmatrix}$, where $\boldsymbol{n}_{\boldsymbol{r}}, \boldsymbol{s}_{\boldsymbol{r}}, \boldsymbol{a}_{\boldsymbol{r}}$ are the column vectors reported using a conventional representation of the end-effector frame axis (normal, sliding, and approach axis, respectively), the orientation error can be computed starting from the equivalent rotation.

$$\boldsymbol{R}(\theta, \boldsymbol{r}) = \boldsymbol{R}_{d}\boldsymbol{R}_{r}^{T} \tag{3.33}$$

which represents the required rotation starting from the configuration described by $\boldsymbol{R}_{d}$ to align with the actual rotation $\boldsymbol{R}_{r}$. The matrix $\boldsymbol{R}(\theta, \boldsymbol{r})$ can be expressed then as rotation about a certain axis $\boldsymbol{r} \in \mathbb{R}^{3\times 1}$ of a certain angle θ through the angle-axis representation. The error is then computed as

$$\tilde{\boldsymbol{x}}_{\boldsymbol{O}} = \boldsymbol{r}\sin\theta \tag{3.34}$$

From the computation detailed in [44], the corresponding error between the desired and actual angular velocity can be obtained from

$$\dot{\tilde{\boldsymbol{x}}}_{\boldsymbol{O}} = \boldsymbol{L}^{\boldsymbol{T}}\boldsymbol{\omega}_d - \boldsymbol{L}\boldsymbol{\omega}_r \tag{3.35}$$

with

$$\boldsymbol{L} = -\frac{1}{2}\left(\boldsymbol{S}(\boldsymbol{n_d})\boldsymbol{S}(\boldsymbol{n_r}) + \boldsymbol{S}(\boldsymbol{s_d})\boldsymbol{S}(\boldsymbol{s_r}) + \boldsymbol{S}(\boldsymbol{a_d})\boldsymbol{S}(\boldsymbol{a_r})\right) \tag{3.36}$$

where the term $\boldsymbol{S}(\boldsymbol{v})$ correspond to the Skew symmetric matrix of the vector $\boldsymbol{v}$, and $\boldsymbol{L}$ is a $\mathbb{R}^{3\times3}$ matrix. When this controller is used to test the ability of the robot to navigate rough terrain, the desired velocity and acceleration are set to zero, since the objective of the controller is to maintain a certain fixed reference position. Moreover, for this reason, the desired position and orientation remain constant during the experiments. However, the analysis of the error computation will be used in the following chapter when the Attitude Controller will be introduced.

3.3 Controller implementation

Based on the theoretical formulation described in the previous chapter, the controller has been developed in `C++` to seamlessly integrate into the existing frameworks. Emphasis was placed on implementing a code capable of independent execution regardless of the chosen kinematic link to control. Moreover, some guidelines were followed to ensure real-time execution and minimize computational overhead per cycle. The additional framework is composed of three classes: *ControlManager*, *CartesianImpedanceSolver* and *CartesianImpedanceController*. Following is a bottom-up description of each class.

The *ControlManager* represent the control plugin that is integrated into *Xbot2*. As described in 2.3 it is a finite state machine, composed of four different states. Its purpose is to coordinate the computation of the torque values from internal functions and external classes and send these values to the actual robot, both in real and simulation scenarios through the *RobotInterface*. Additionally, the *ControlManager* class is in charge of sharing a pointer to a *ModelInterface* object, which is constantly updated and synchronized with the *RobotInterface* object. The four states are:

- `on_initialized()` where the resources are allocated and the objects related to the other classes are initialized. Here is also created an instance of `ModelInterface` which is then synchronized with the robot detected by the `Xbot2` framework;

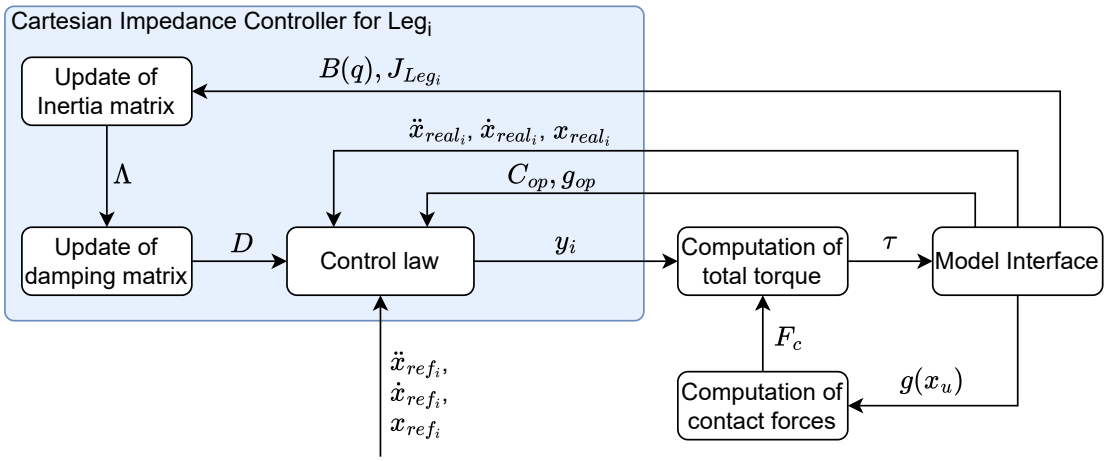


Figure 3.2: Diagram of the Cartesian Impedance Controller. Each block represents a part of the controller explained in the previous section.

- `on_start()`;
- `run()` is the periodic task in which the torques to be sent to the robot are computed at each iteration based on the control law in Eq. (3.20);
- `on_stop()`.

The *CartesianImpedanceSolver* on the other hand is the connection to the *CartesI/O* framework. It is derived from the *CartesianInterface* class, therefore inheriting all its methods. Inside this class, the solver identifies all the tasks specified within the stack of problem files and formulates the inverse kinematic problem accordingly. Using the API provided by its parent class, the solver generates pointers to each task, allowing to manage and access them as needed. The last class is the *CartesianImpedanceController*, which contains all the functions to compute the forces according to Eq. 3.6.

Each leg motion is governed by a single Cartesian Impedance Controller, which adjusts the torque of the joints between the base link frame and the leg's end effector. During each controller cycle, the torque values are updated for each leg based on the solution of an inverse dynamic problem. The process starts by updating the inertia matrix, denoted as $\mathbf{\Lambda}$, computing its pseudo-inverse through the Singular Value Decomposition (SVD) as detailed in Eq. (3.23). Here, the relative Jacobian $\boldsymbol{J_i}$ and the joint space inertia matrix $\boldsymbol{B}$ are obtained from the *ModelInterface*. Subsequently, the updated value of $\mathbf{\Lambda}$ is employed to derive the new damping matrix, as described in Eq. (3.31), achieved by solving the associated Generalized

Eigenvalue Problem. This resultant damping matrix $\boldsymbol{D}$ is then applied to calculate the forces $\boldsymbol{F_{ext}}$ resulting from the mass-spring-damper dynamics, as depicted in Eq. (A.2). These computed forces are introduced into Eq. (3.18), which is the solution of the inverse dynamic problem, excluding the acceleration term since inertia shaping is not performed. Consequently, the torque vector resulting from Eq. (3.20) contains the required torque values for the joints in the i-th leg chain, thus enabling the leg to behave as a mass-spring-damper system. These torques of each leg are then aggregated into a single vector $\boldsymbol{y}$. The additional contribution of the contact force is determined by solving a least-square minimization problem using the pseudo-inverse, as elucidated in Eq. (3.28). Subsequently, all computed forces are integrated into Eq. (3.21), alongside the gravitational terms, to ascertain the corresponding torque values $\boldsymbol{\tau}$ to be transmitted to the robot, encompassing the gravity compensation terms and the imposed dynamics.

Chapter 4

Cartesian Impedance Controller Simulation on Rough Terrains

Contents

After the implementation of the controller, the next step is to test the effectiveness of this strategy on the robot, initially through simulation and subsequently in real-life scenarios. The upcoming chapter details a series of experiments to confirm the controller's proper operation on rough terrain. Initially, the step response of the controller is examined on one of the robot's legs. Then the robot's behaviour is studied on a self-implemented terrain, composed of small but frequent deformities, with different configurations of the Cartesian Impedance Control to analyse how the stiffness values affect the ability of the robot to overcome the obstacles.

4.1 Step response study

The first experiment aims to study the step response characteristics of the controller to verify whether it behaves according to the design specification. The test is computed on three different configurations, with different stiffness on the third component, corresponding to the linear motion along the Z-axis, comparing low, medium, and high stiffness settings, as detailed in the Tables 4.1, 4.2, 4.3. It is worth noticing that in this section as well as in the next ones, in the terrain where

the robot is tested the majority of the shock-absorbing action happens along the Z-axis, therefore the deviations from the desired position and orientation about other Cartesian axes are minimized by setting constant high values of stiffness.

Legs	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**	**D. R.**
Leg 1	10000	10000	500	2000	2000	2000	0.7
Leg 2	10000	10000	500	2000	2000	2000	0.7
Leg 3	10000	10000	500	2000	2000	2000	0.7
Leg 4	10000	10000	500	2000	2000	2000	0.7

Table 4.1: Low stiffness values. Note that the stiffness referring to the linear part is expressed in N/m, while the torsional stiffness is expressed in Nm/rad. The damping ratio (D.R.) is a unitless measure.

Legs	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**	**D. R.**
Leg 1	10000	10000	3000	2000	2000	2000	0.7
Leg 2	10000	10000	3000	2000	2000	2000	0.7
Leg 3	10000	10000	3000	2000	2000	2000	0.7
Leg 4	10000	10000	3000	2000	2000	2000	0.7

Table 4.2: Medium stiffness values. Note that the stiffness referring to the linear part is expressed in N/m, while the torsional stiffness is expressed in Nm/rad. The damping ratio (D.R.) is a unitless measure.

Legs	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**	**D. R.**
Leg 1	10000	10000	7000	2000	2000	2000	0.7
Leg 2	10000	10000	7000	2000	2000	2000	0.7
Leg 3	10000	10000	7000	2000	2000	2000	0.7
Leg 4	10000	10000	7000	2000	2000	2000	0.7

Table 4.3: High stiffness values. Note that the stiffness referring to the linear part is expressed in N/m, while the torsional stiffness is expressed in Nm/rad. The damping ratio (D.R.) is a unitless measure.

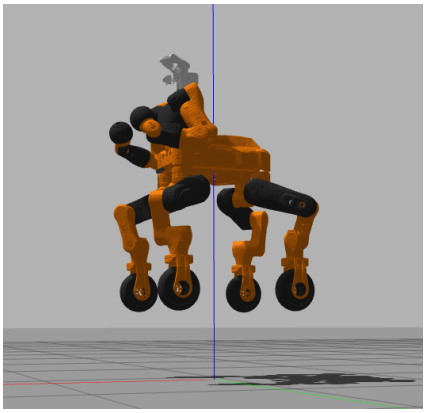
To properly study the step response of only the modelled mass-spring-damping system, the robot is lifted and locked in the air to cancel the effects of the floating base, Figure 4.1a. The computation of the contact forces is therefore neglected since the weight of the robot does not have to be compensated, while the other terms in the control law are maintained. The signal plotted in Figure 4.1b confirms that the controller responds to a change in reference position (of about

$10cm$) according to the theory in Appendix A.2: a higher gain on the position error results in a quicker settling time, while the non-critical damping cause some oscillation before reaching the steady state. Of course, these conditions are far from the ones where the robot is expected to move, and the plots in Figure 4.1b do not indicate the real response of the system, however, it serves as a reliable indicator of response stability under ideal conditions.

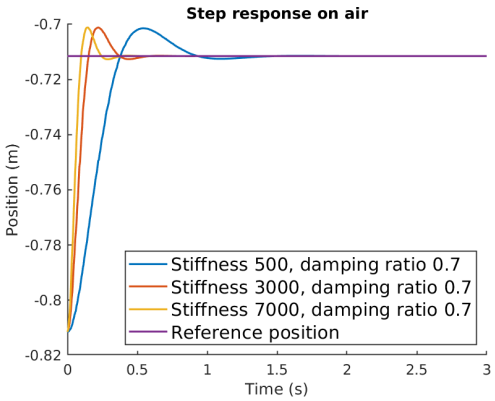
On the other hand, when the robot is in contact with the ground and behaves as a proper floating base system, all the robot movements are affected by the inertia of the base, and cannot be neglected. In this case, all the contributions of the control law are maintained according to the theory of the previous section. In Figure 4.2b the responses show differences during the settling phase between the two cases, suggesting a discrepancy between the idealized linear model employed in our Cartesian impedance controller and the true nonlinear dynamics of the system. Specifically, as the controller operates between the base link frame and the end effector of the leg, it neglects the inertial effects exerted by the entire body. Consequently, each motion of the legs due to external disturbances or reference changes, could generate moments on the base which are not explicitly addressed, as would be in the case of a whole-body dynamic control strategy. The observed oscillations are a result of this simplified approach since the controller takes additional time to counteract these dynamics. Additionally, since the computed contact forces represent an estimation of the real ones, there could be residual forces exerted by the robot, that the Cartesian controller has to counteract, maintaining a certain error on the position to generate the proportional action. The result is visible in a steady state error which increases when the set stiffness values get lower, as expected from the Eq. 3.6. Still, these tests demonstrate the stability of the complete system

4.2 Rough Terrain

In this scenario, the same three configurations of the Cartesian Impedance controller are tested on terrain with persistent small disturbances but no variation on the slopes, which tries to simulate one of the possible conditions where the robot could be required to operate, such as surfaces with cobblestone or gravel. As existing libraries lack models of ground planes with these characteristics, a custom terrain was generated by placing 100 primitive Gazebo boxes with a random position and orientation in an area of 10 x 4 meters, with a specific placement along the Z-axis such that the portion of the box exiting from the ground will never be higher than the radius of the robot wheel, of 0.124 meters. This will guarantee that the robot will always be able to overcome the obstacle. An example of the

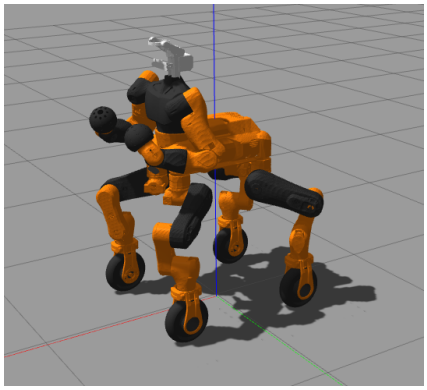


(a) The pelvis link of CENTAURO has been moved and locked in the air. The robot now behaves as a fixed base system.

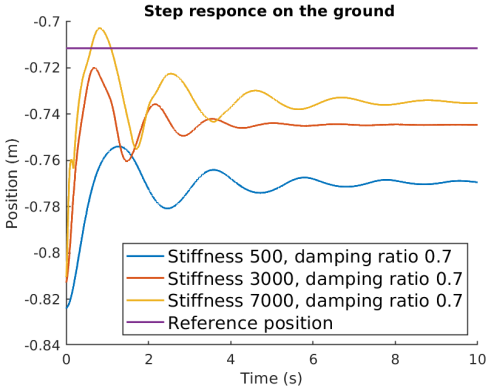


(b) Step response along the Z-axis of the front left leg, with three different configurations of the Cartesian impedance controller.

Figure 4.1: Step response of the front left leg when the robot is lifted in air



(a) CENTAURO is in contact with the ground



(b) Step response along the Z-axis of the front left leg, with three different configurations of the Cartesian impedance controller.

Figure 4.2: Step response of the front left leg when the robot is in contact with the ground

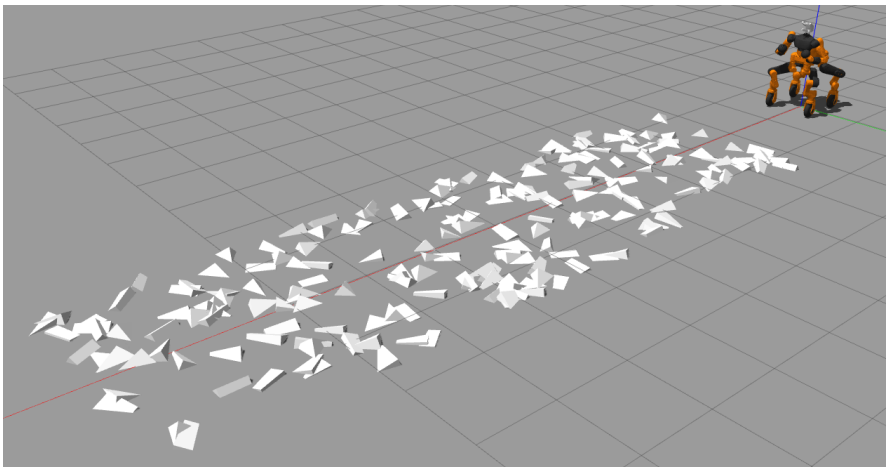


Figure 4.3: Rough terrain generated through a Python script to place 100 primitive boxes in a random position and orientation. It represents a custom model imported into the Gazebo environment for easy deployment in the simulation world.

model can be found in Figure 4.3.

The motion of the robot is achieved through wheeled locomotion. This movement is controlled by a *Xbox One X* joystick, which converts the position of the stick into the corresponding velocity along the Cartesian axis. These values are then transformed into torque for the ankle complex by the control plugin *omnisteering*, allowing for both steering and wheel rotation. Note that when this plugin and the Cartesian plugin are activated together, there could be some conflict between the torque references sent to the ankle yaw joints. From one side, there is a reaction torque to maintain the ankle with a certain orientation, while, on the other side, the omnisteering actuates the joints to compute a steering action. To solve this issue, the contribution of the Cartesian impedance controller is disabled, so the wheels are free to move. In all the experiments the maximum velocity is set to 1 m/s. The choice will be better explained in the following section.

To properly assess that a control strategy could increase the ability of the robot to overcome this scenario, the robot is tested first without the Cartesian Impedance Controller. In this case, the robot position is maintained by an impedance control at each joint implemented in the framework, with high values of stiffness and damping, resulting in a very rigid behaviour. Several tests show that this non-compliant configuration renders the system unable to adapt to irregularities in the terrain. The rigidity of the system prevents effective absorption or mitigation of sudden impacts generated by these deformities, ultimately leading

to destabilization, as described by [46]. Additionally, the loss of contact with the terrain alters the support polygon, progressively reducing its size with each contact loss, thereby narrowing the stability margin and increasing the risk of tipping or tumbling, particularly during rapid shifts in the COM position. Furthermore, the rigid structure of the robot, coupled with its elevated weight, amplifies impact forces, further compromising stability. Taking into consideration a successful case, the occurrence of this phenomenon is supported by Figure 4.4, where the module of the contact force on each leg is estimated by a tool integrated within the *Xbot2* framework. The Blue lines illustrate both the occurrence of contact loss, when the force diminishes to zero, and the elevated magnitude of the estimated contact forces along the vertical direction, resulting from the impact of the wheels on the terrain. Testing the robot in three other configurations reveals the effectiveness of the Cartesian Impedance Controller in improving its overall ability to traverse challenging terrain, despite the initial rigidity and instability observed without the controller. In particular, both the high and medium stiffness configurations show reduced instances of contact loss during terrain traversal compared to the non-compliant scenario, indicating the controller's capacity to mitigate destabilization. However, it is the configuration with a stiffness set at 500 on the Z-axis that showcases the most substantial enhancement. In this configuration, the robot exhibits remarkable adaptability to changes in terrain, maintaining continuous contact and navigating the terrain with greater stability with respect to the other setting. The Figure 4.4 report in Orange the estimated contact forces on the four legs.

The proposed active suspension mechanism aims to absorb shocks, dampen vibrations, and minimise the oscillation of the base, by effectively mitigating the distribution of forces across the robot's structure. In this context, the roll angle serves as a direct indicator of the system's ability to resist tilting or tipping over in response to lateral forces or uneven terrain profiles and can be easily measured through the IMU sensor mounted on the base of the robot. The box plot analysis in Figure 4.5a reports the roll angle deviations in the three configurations of the controller as well as the case without it. The figure indicates a trend towards reducing both the maximum and minimum roll angle values, as depicted by the decreased variability of the data points, excluding outliers. In configurations with high stiffness, the response in terms of roll angle remains similar to the scenario without the controller. This observation is evident from the marginal differences between the 100^{th} and 0^{th} percentiles, which differ by only 0.02 radians and 0.03 radians, respectively. Instead, configurations with lower stiffness exhibit a more compliant behaviour, effectively reducing oscillations at the base, as evidenced by the data. This result is further supported by the box plot of the angular velocity about the longitudinal axis, in Figure 4.5b, which is directly related to the generated angular moment of the base [16]. Here the reduction of variability is

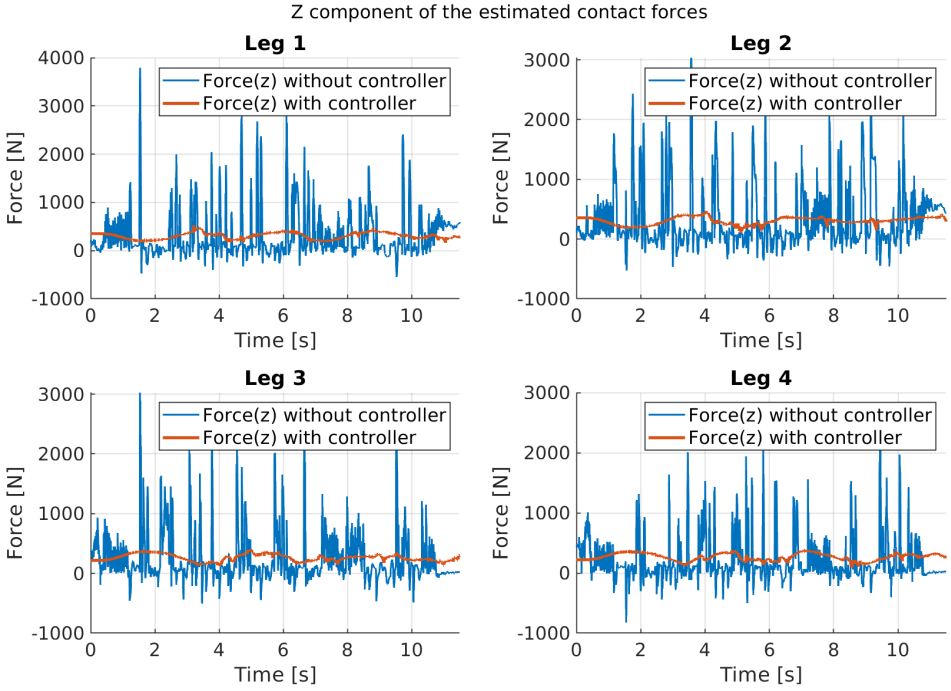
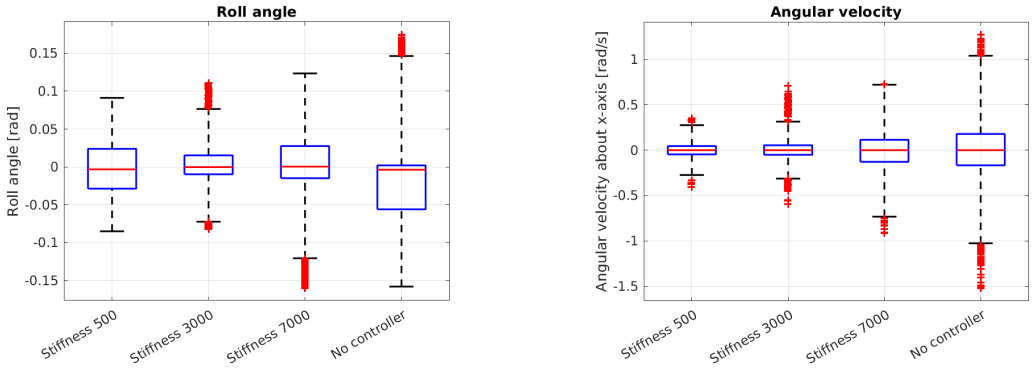


Figure 4.4: In this Figure, only the vertical component of the contact wrench for each leg is plotted. These forces are expressed with respect to the local ankle yaw link frame. The blue line refers to the case without the controller, while the orange line reports the estimated forces when the controller is set in the 4.1 configuration



(a) (b)

Figure 4.5: Values distribution of Roll angle (a) and angular velocity (b) resulting from experiments in a simulated environment. The reported stiffness setting refers to the value of the stiffness gain along the Z-axis motion of the Cartesian space set in the Cartesian Impedance Controller. The stiffness settings in the other directions are constant across the tests, as detailed in the Tables 4.3, 4.2, 4.1

even more evident and reaches its minimum in the most compliant configuration. Furthermore, in Figure 4.5a is possible to notice a centred distribution of roll angle deviations around zero in the scenario where the controller is active. This suggests a more balanced and stable configuration, where the wrenches are acting more symmetrically.

In addition to the observed improvements brought by the implementation of the controller, the tests highlighted the impact of inertia on the robot's dynamics. As the robot accelerates, legs with lower inertia exhibit greater responsiveness compared to the base, which possesses a larger mass and consequently higher inertia. In forward acceleration the front legs tend to extend forward, inducing a dragging effect on the base, while the rear legs tend to compress backwards, exerting a pushing force on the base. During deceleration, the reverse phenomenon occurs, with the front legs compressing and the rear legs extending. This behaviour is a consequence of the inertia of the robot's body and is independent of the stiffness settings of the Cartesian impedance control on each leg. However, the stiffness settings can influence the size to which the legs extend or compress, as well as how quickly they respond to the changes in motion. In the case of low stiffness, this effect is visible in the first 3 sec of the estimated forces plot in Figure 4.4, where the backward rotation of the base, results in higher forces on the back legs and a consequent reduction of the forces on the front. To reduce this effect, the velocity of the robot was limited to $1m/s$ to minimize the acceleration and

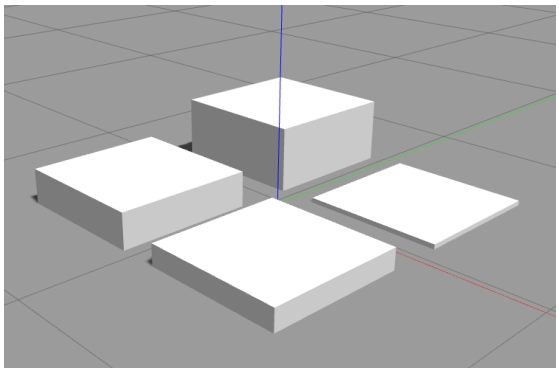


Figure 4.6: Example of boxes motion of Gazebo

delivered manually as a ramp through the joystick.

4.3 Terrain with slopes

The experiments in the previous scenario demonstrated the ability of the controller's capability to mitigate minor disturbances on mostly flat terrain. However, it remains untested how effectively the controller performs on uneven terrain where slopes necessitate the legs to adjust to different altitudes. To simulate such environments in Gazebo, one option involves utilizing pre-existing models that already incorporate terrain features with slopes. Alternatively, a custom model can be created using height maps, which automatically extract the terrain profile from greyscale images. However, both approaches come with their challenges. Detailed environments can significantly slow down simulations due to increased graphical rendering complexity. Additionally, there's a risk of encountering issues with collision configurations, potentially reducing the accurate representation of the terrain's interaction with the robot model. Since the objective of these simulations is to provide vertical displacement of the contact point on different and not constant altitudes, A simpler approach involves employing primitive boxes positioned beneath each contact point of the robot and controlling their movement via the ROS topic provided by Gazebo. This method allows for real-time manipulation of the blocks' positions, offering the possibility to control the vertical position of each box independently. The configuration of this setup is depicted in Figure 4.6. It's important to recognize that while this approach provides a practical means of simulating changes in altitude, it does not model forces acting along the x or y axes, which may occur in situations where the normal vector of the contact forces is not vertical. However, modelling such forces would extend beyond the scope of this thesis.

In this scenario, all the controller configurations will be tested to assess its capability to respond to higher-magnitude disturbances, in terms of vertical displacement of the contact points from the initial position. The box motion, which represents the source of the disturbances, is defined by a sinusoidal function of the type

$$g(t) = A \sin(2\pi f t - f_{offset}) - p_{offset} \tag{4.1}$$

where A represent the amplitude of the wave, f is its frequency, t is the current time of the simulation, while f_{offset} and p_{offset} are respectively the phase offset and the position offset on the vertical axis. Here, the setting of the waves is such that the motion starts from the ground level, and reaches a maximum altitude of 25 cm, with a frequency of 10 Hz. The phase offset instead is used to singularly delay the motion of each box.

In the presented simulations, without any controller to mitigate the disturbances, the robot's stability was severely compromised. As the terrain shifted, the robot struggled to maintain balance and, ultimately, fell and tumbled over due to the inability to adapt to a changing surface. Setting the controller to high or medium stiffness the robot still exhibited a rigid response, which, in addition to the system's movements caused by the base inertia, led the robot to lose multiple contacts, and therefore lose stability. With the controller set to lower stiffness, as in the case of Tab 4.2 and 4.1, the robot demonstrated better adaptability to the changing terrain compared to the other cases. The compliant nature of these settings allowed the legs to adjust more easily to the shifting terrain, resulting in improved stability. However, testing the controller with different settings of Eq. 4.1 demonstrated that despite initial adaptability, the robot eventually diverged and fell as the disturbances persisted. This is an indication of the controller's limitations in maintaining stability over persistent terrain changes, demonstrating that the success of some simulations depends primarily on the environment setup, instead of the action of the controller.

Fundamentally, the Cartesian Impedance Controller is designed to regulate the robot's interaction forces with the environment, maintaining a specific position and stiffness profile. However, in dynamic environments such as terrains with slopes, this fixed reference point becomes a limitation rather than an advantage. As described in the experiments, the robot struggled to adapt to the changing surface, ultimately leading to instability and falls. One of the primary reasons behind this behaviour lies in the controller's inability to dynamically adjust its reference points in response to variations in the terrain. In particular, in the case of medium and high stiffness, the rise time presents a more rapid response and as

a result, the robot struggles to reject disturbances with low frequencies.

4.4 Conclusion

The experiments conducted aimed to evaluate the effectiveness of a Cartesian Impedance controller in improving the performance of CENTAURO on traversing challenging terrains. The results obtained from the experiments demonstrated that the controller provides a stable response to the interaction with external stimuli. Even though whole-body dynamics were not considered in the implementation, the controller allowed the robot to cross rough terrain with a balanced distribution of the forces and compliant behaviour, adapting to the surface and maintaining contact with the ground in a more compliant configuration. The constant contacts, in addition to lower oscillations to which the base is subject while traversing these environments, represent also an improvement in the stability of the robot, although is not possible to guarantee it due to the absence of any stability criteria. Furthermore, experiments on terrains with slopes revealed the controller's limitations in responding to higher-magnitude disturbances. The rigidity imposed by higher stiffness values forced the robot to maintain its initial *homing* configuration, which led to unstable positions in the longitudinal and lateral directions, while the controller with low stiffness setting demonstrated better adaptability over terrain changes. However, these results show how the Cartesian Impedance Controller alone is not able to adapt to all terrain configurations, particularly the ones with slopes. An additional component is therefore required to update in real-time the references of the Cartesian Impedance controller to maintain the base as stable as possible.

Chapter 5

Attitude Controller

Contents

In this chapter, it is explained the implementation of an Attitude Controller used to control the orientation of the robot's base. First, a theoretical overview is proposed, with all the mathematical passages used to derive the control law. Then, the integration with the Cartesian Impedance Controller framework, followed by a series of experiments to validate the effectiveness of the controller.

5.1 Introduction

In scenarios where robots operate across varied terrain with slopes or ramps, the adaptability of a system relying solely on a Cartesian Impedance Controller is found to be limited, as demonstrated in preceding chapters. While increasing leg compliance can enhance adaptability, it often introduces excessive oscillations that risk destabilizing the robot. Conversely, rigid behaviour tends to result in the system tumbling. Both issues derive from the nature of the Cartesian Impedance controller, which aims to maintain a fixed configuration during navigation, thus

sticking to predetermined reference values of position, velocity and acceleration. Recognizing the need to inject new reference values into the Cartesian Impedance Controller to enhance adaptability, a strategy is required to dynamically adjust the leg positions, thereby controlling the rotations of the robot's base, while the Cartesian Impedance Controller is responsible for the filtering action of the disturbances resulting from the interactions with the environment. This strategy could leverage all available information from the robot's sensors. Notably, the robot is equipped with an IMU (Inertial Measurement Unit) sensor mounted on its floating base. This sensor provides estimates of angular velocity and linear acceleration relative to the world or inertial frame, along with the rotation matrix R_B^W representing the Roll-Pitch-Yaw (RPY) angles between the two frames. The IMU sensor provides crucial information about the magnitude of variation in the robot's orientation in response to changes in terrain elevation. When the robot tilts on one side, adjusting the legs's motion can mitigate this deviation in orientation, by applying new references on the Cartesian Impedance Controller, which will then follow these new values, with a behaviour depending on the stiffness settings. Leveraging data from the IMU sensor, an impedance control mechanism is suggested for managing both roll and pitch angles between the base and the inertial frame. The primary objective is to maintain these angles as close to zero as possible, maintaining the base parallel to the ground plane.

5.2 Theoretical Basis

The control strategy implemented for both the pitch and roll angle is based on the theory of Appendix A.2, therefore developing an impedance controller which receives as input the values of the two angles, and their rate of change over time, corresponding to the angular velocity. These parameters are obtained from the IMU sensor mounted on the base of the robot, providing real-time information about the robot's orientation with respect to the inertial frame. Using these inputs, the controller computes the commanded acceleration necessary to counteract the error concerning the reference values which, in this case, are both set to zero for the angular velocity and the two angles. The control laws governing this process are

$$\begin{aligned} \ddot{\alpha_{CMD}}_{Roll} &= K_v \left(\omega_{x_{ref}} - \omega_{x_{real}}\right) + K_p \left(\theta_{Roll_{ref}} - \theta_{Roll_{real}}\right) \\ \ddot{\alpha_{CMD}}_{Pitch} &= K_v \left(\omega_{y_{ref}} - \omega_{y_{real}}\right) + K_p \left(\theta_{Pitch_{ref}} - \theta_{Pitch_{real}}\right) \end{aligned} \tag{5.1}$$

where K_v and K_p are control gains that influence the system's response, respectively the derivative and proportional gain. Adjusting these gains allows for tuning the controller's behaviour to meet specific performance requirements. $\ddot{\theta}_{CMD}$ is the commanded acceleration of the roll and pitch angle to maintain the

real values as close as possible to their references, while θ_{Roll} and θ_{pitch} are the roll and pitch angle. The parameters ω_x and ω_y are respectively the angular velocity about the X and Y axis, carrying the information about the change over time of the roll and pitch angles. The notation *ref* refers to the reference value, here set to zero, while *real* to the actual value computed by the IMU sensor. By integrating over time the commanded acceleration $\ddot{\alpha}_{CMD_{Roll}}$ and $\ddot{\alpha}_{CMD_{Pitch}}$, the corresponding velocity and angle $\dot{\alpha}_{CMD_{Roll}}, \dot{\alpha}_{CMD_{Pitch}}, \alpha_{CMD_{Roll}}, \alpha_{CMD_{Pitch}}$ can be computed. Based on the theory of Appendix A.2, given a value of K_p the corresponding K_v is computed as:

$$K_v = 2\zeta\sqrt{K_p} \tag{5.2}$$

where ζ is the chosen damping factor.

Since CENTAURO is categorized as a floating base system, the base cannot be directly actuated like the other joints in the system, as explained in Section 2.1 about the floating base theory. To adjust the attitude of the robot in order to maintain the base parallel to the ground, it is required to find a relationship between the roll and pitch angle and the motion of the other robot components. A possible solution can be deduced from a geometrical perspective, which relates the contact point on each leg with the base angles and velocities. Let's simplify the problem and take into consideration only the movement about the longitudinal axis (i.e., the roll angle). By placing the wheels of the right portion of the robot on a higher level as in Figure 5.1 it is possible to generate a roll angle between the base frame and the vector normal to the ground. This angle equals the angle between a surface parallel to the XY plane of the inertial frame and the segment connecting one wheel to the wheel on the opposite side of the robot. With this geometrical insight, using a straightforward trigonometric equation is possible to compute the height difference between the left and right wheels

$$\Delta z_{Roll} = \frac{d}{2} \cdot \sin(\theta_{Roll}) \tag{5.3}$$

where d, in this case, represents the distance between the contact points on the right side and the contact points on the left side, expressed with respect to the local contact link frames. Furthermore, Δz_{Roll} is half of the altitude difference between the two sides of the robot.

The geometrical relation introduced with Eq. (5.3), offers a way to convert the output of the control law (5.1) into movements of the legs. By replacing in Eq. (5.3) the current angle θ_{Roll} with the computed value of $\alpha_{CMD_{Roll}}$ derived from Eq. (5.1), the resulting Δz_{Roll} denotes the vertical displacement required to actuate the control action. Notably, the commanded displacement is symmetrically

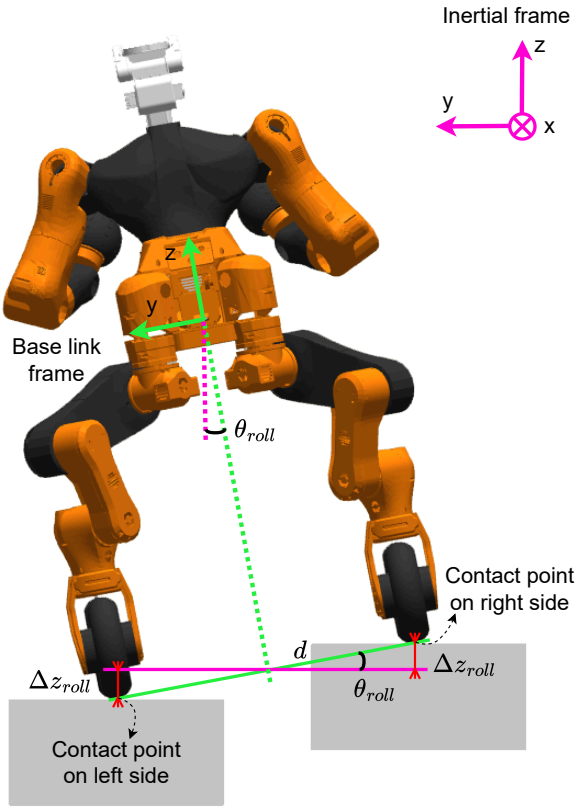


Figure 5.1: Drawing representation of the mapping between the roll angle θ_{roll} obtained from the IMU sensor and the corresponding vertical displacement Δz_{roll} of the contact point. The value d represents the distance between the contact points on the left side and the contact points on the right side.

distributed between the robot's sides, with half allocated to the right side and the other half to the left side. Subsequently, by derivation over time of Δz_{Roll}, is possible to compute the corresponding velocity and acceleration

$$\begin{aligned} \Delta z_{Roll} &= \frac{d}{2} sin\left(\alpha_{CMD_{Roll}}\right) \\ \dot{\Delta} z_{Roll} &= \frac{d}{2} cos\left(\alpha_{CMD_{roll}}\right) \dot{\alpha_{CMD_{roll}}} \\ \ddot{\Delta} z_{Roll} &= \frac{d}{2}\left[-\sin\left(\alpha_{CMD_{roll}}\right) \dot{\alpha}^2_{CMD_{roll}} + cos\left(\alpha_{CMD_{roll}}\right) \ddot{\alpha}_{CMD_{roll}}\right] \end{aligned} \tag{5.4}$$

These equations establish a direct mapping from a torsional motion into a linear one, converting a control action for the roll angle into the vertical displacement

of the contact points, in terms of variations in position, velocity and acceleration of the leg's end effector. In this way, the system can control the attitude of the base through the motion of the legs.

The same consideration can be obtained along the lateral axis, deriving the needed equations to control the pitch angle

$$\begin{aligned} \Delta z_{Pitch} &= \frac{d}{2} sin\left(\alpha_{CMD_{Pitch}}\right) \\ \dot{\Delta} z_{Pitch} &= \frac{d}{2} cos\left(\alpha_{CMD_{Pitch}}\right) \dot{\alpha_{CMD_{Pitch}}} \\ \ddot{\Delta} z_{Pitch} &= \frac{d}{2}\left[-\sin\left(\alpha_{CMD_{Pitch}}\right) \dot{\alpha}^2_{CMD_{Pitch}} + cos\left(\alpha_{CMD_{Pitch}}\right) \ddot{\alpha}_{CMD_{Pitch}}\right] \end{aligned} \tag{5.5}$$

The contribution coming from (5.4) and (5.5) are then summed to obtain the total resulting contribution

$$\begin{aligned} \Delta z_{tot} &= \Delta z_{Roll} + \Delta z_{Pitch} \\ \dot{\Delta} z_{tot} &= \dot{\Delta} z_{Roll} + \dot{\Delta} z_{Pitch} \\ \ddot{\Delta} z_{tot} &= \ddot{\Delta} z_{Roll} + \ddot{\Delta} z_{Pitch} \end{aligned} \tag{5.6}$$

As a result, the constant computation in real-time of new values of Eq. (5.6), allows the Attitude controller to effectively map the terrain profile and adjust the leg motion accordingly. This adaptive approach should ensure that the base remains parallel to the ground plane, mitigating slopes and differences in altitudes between the legs. Then, once the newly determined position, velocity and acceleration references of each leg's end-effector are computed, they serve as inputs to the Cartesian Impedance Controller. In particular, the displacement Δz_{tot} first needs to be summed to the starting position of the robot, computing, therefore, an integrating action. In this capacity, the Cartesian Impedance Controller functions as a position controller, tasked with tracking the trajectory dictated by the Attitude controller, while rejecting at the same time all the disturbances arising from the interaction with the environment. This integration of controllers should provide robust performance and more adaptability to challenging terrain conditions, consisting of both high and low-frequency disturbances.

The new system is composed of multiple interconnected virtual spring-damper components: first, an impedance controller at the base, subdivided into two torsional springs, controlling the motion along the lateral and longitudinal directions. Given the floating base nature of the robot, additional components are required

for the torsional springs to receive forces from interactions with the environment. As a result, these torsional springs are connected to four supplementary virtual systems, directly linked to the end-effectors of the legs. This configuration yields a complex system where the effect of each component is coupled from a purely theoretical point of view. When an end-effector undergoes displacement due to terrain interaction, this movement triggers a counteracting response from the overall virtual system, in such a way that the motion of each component is affected by the motion of the others. Achieving a decoupled effect within such a system requires precise tuning of stiffness and damping parameters for each component. Specifically, a decoupled effect could be assumed from a study of the system's response: considering an ideal scenario, the settling time of the Attitude controller can be computed from Eq. 5.1 as

$$t_{s,1\%} = \frac{4.6}{\zeta\sqrt{K_p}} = \frac{4.6}{\zeta\omega_n} \tag{5.7}$$

while the settling time of the Cartesian Impedance Controller in ideal conditions can be easily obtained from the step response plot in Figure 4.1b. It is reasonable to consider the two systems are decoupled if their settling times differ by a factor of 10, indicating that the natural frequency of the Attitude Controller is ten times greater than that of the Cartesian Impedance Controller. This assumption is equivalent to selecting a low stiffness for the torsional spring and high stiffness values in the Cartesian impedance springs. In this configuration, the latter exhibits a rigid behaviour with minimal deformation, thereby reducing its impact on the compliant behaviour of the torsional springs. Consequently, it becomes plausible to approximate a decoupled effect within the system, where ideally, the action of one subsystem remains unaffected by the actions of others.

5.3 Controller Implementation

To implement the controller based on the theory explained in the previous section and integrate it into the computational process explained in Section 3.3, a real-time safe class *AttitudeController* was created. This class inherits the instance of the *ModelInterface* initialized into the *ControlManager* object, as well as the pointer to the tasks defined in the stack of problem file, while the information about the damping ratio and proportional gain values, for both pitch and roll controller, is defined in a custom YAML file, as the one reported in Appendix A.5

During a single computational cycle, first, the current base orientation is obtained from the IMU sensor through the *ModelInterface* API. With this data, the new reference values for each kinematic chain are computed, in this case, one for

each of the four legs. First, the commanded accelerations are obtained from Eq. (5.1). Then the torsional action is translated into vertical motion of the contact points through Eq. (5.5) and (5.4). These values are then set as the new references of the corresponding Cartesian Impedance Controller. Subsequently, the corresponding torques are computed as explained in the section 3.3. Figure 5.2 reports the diagram of the complete framework for a better understanding of how each block communicates.

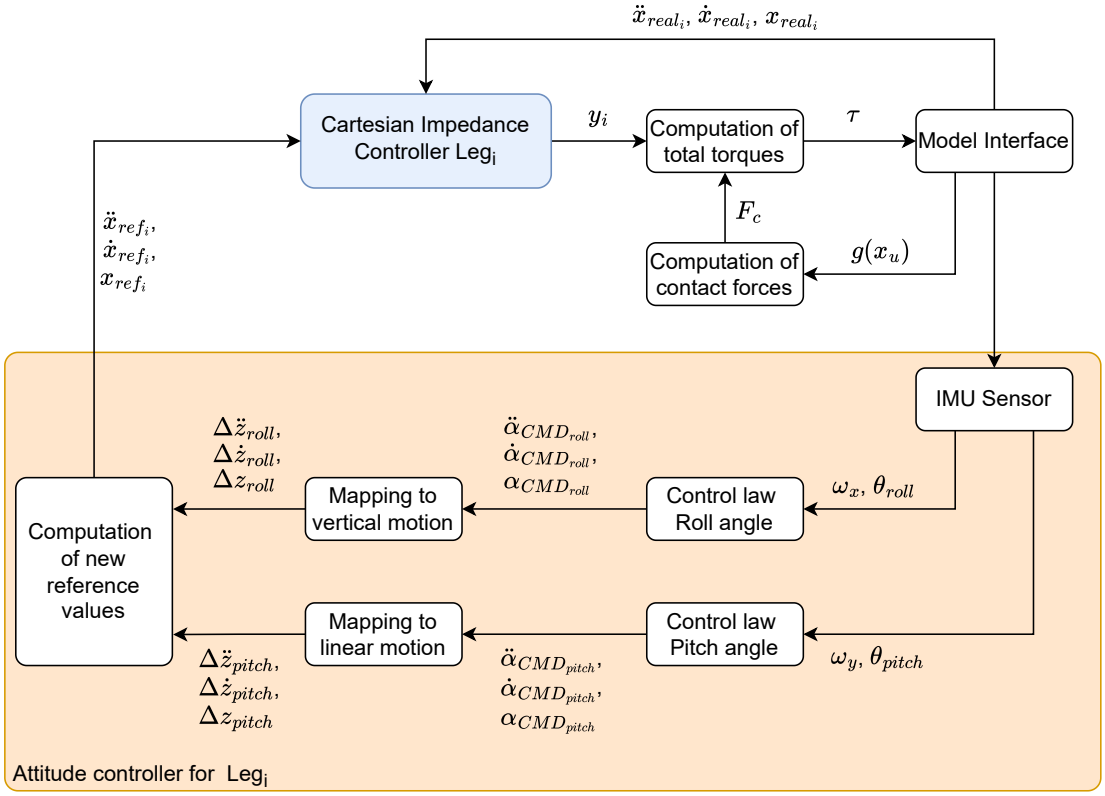
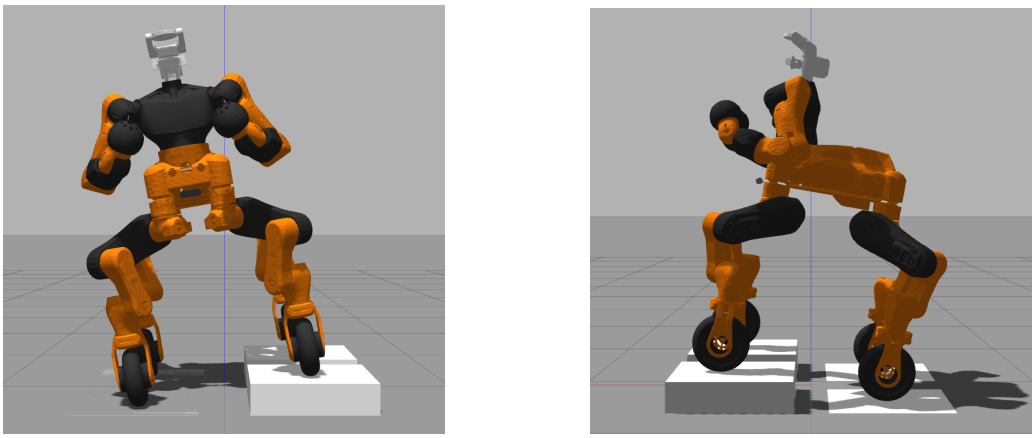


Figure 5.2: Diagram of the complete framework, detailing the Attitude Controller and reporting the Cartesian Impedance Controller as a "black box" which receives the reference position and outputs the torque values. Refer to Figure 3.2 for a precise description of the Cartesian Impedance Controller.

5.4 Experiments on simulated environment

In this section, some experiments will be conducted to validate the correct integration of the Attitude Controller inside the Cartesian Impedance Controller framework. First, the step response is studied both for pitch and roll compensation,



(a) (b)

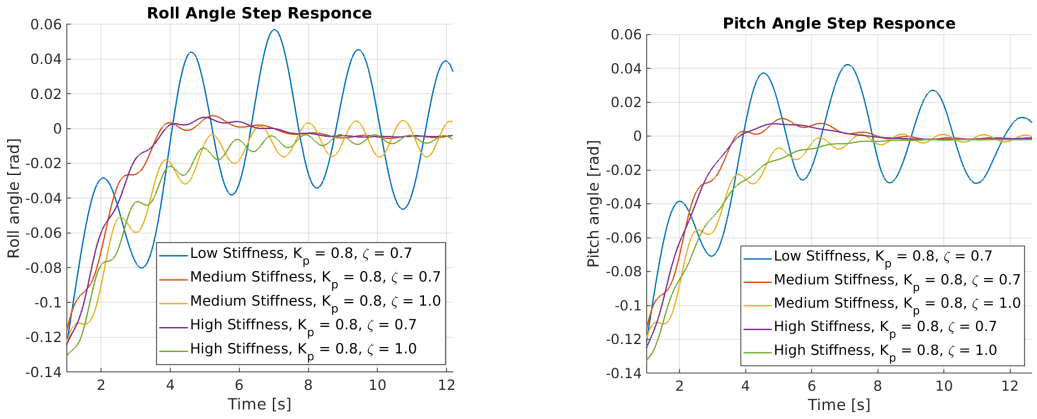
Figure 5.3: Environment setups for the test on the step response of the Attitude Controller. (a) is used to test the roll angle, and (b) to test the pitch angle.

to verify the stability of the system with different configurations of proportional gain, on the same Cartesian Impedance Controller setting used in the previous simulations. Then, based on the conclusion from the step response, the Attitude Controller is tested in a simulated scenario with slopes, which should provide an overview of the improved performances of the whole framework.

5.4.1 Step Response

Here the boxes structure implemented in Section 4.3 is used to test the step response on the system by positioning one side of the robot on a raised plane of constant height. This altitude difference will create an error between the actual angle and the reference value, which is set to zero, and it will be compensated through the action of the control law. The test is conducted separately on the roll angle and pitch angle, even though the obtained results should be similar in the two cases.

First, the error about the roll angle was tested by setting the boxes at a height of about 0.125 m from the ground, as shown in Figure 5.3a. After a considerable number of simulations, the configuration of the Attitude Controller was chosen arbitrarily, incorporating a proportional gain of $K_{p_{Roll}} = 0.8$ and two damping factor variations: $\zeta_{Roll} = 0.7$ and critical damping $\zeta_{Roll} = 1$. The resulting roll angle curves from these simulations, as illustrated in Figure 5.4a, highlight variations in the system behaviour across different configurations. In particular, when the Cartesian Impedance Controller is set with low stiffness, 4.1, the Atti-



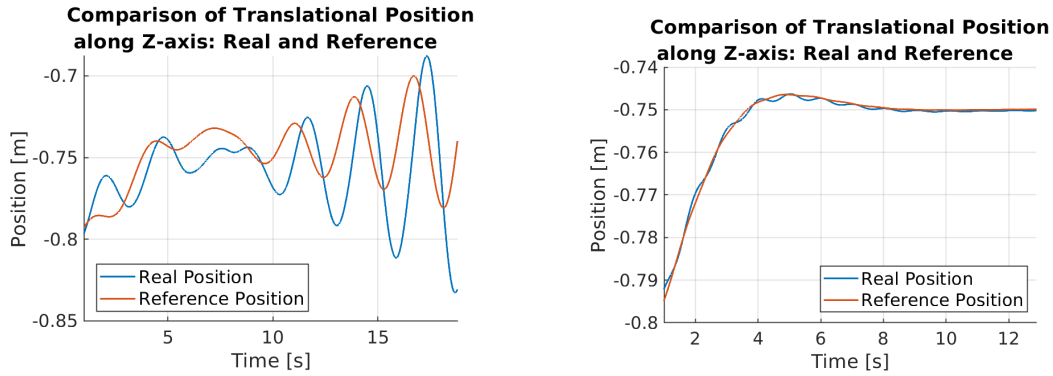
(a) (b)

Figure 5.4: Plots of the Roll and Pitch angle resulting from the step response test. In (a) the curves refer to the setup reported in Figure 5.3a, while (b) shows the plot in configuration 5.3b. In the legend, the low, medium and high stiffness refers to the configuration of the Cartesian Impedance Controller, detailed in Table 4.1, 4.2, 4.3 respectively.

tude Controller tend to diverge, exhibiting increasing oscillations, especially when the damping factor is set to the critical one. However, as the robot stiffness is increased, the system tends towards stability, settling into a steady state with several oscillations of the angles. The best results in terms of stability of the response are achieved through a higher stiffness setting in the Cartesian Impedance Controller, coupled with non-critical damping in the Attitude Controller. Similar results are obtained from the test on the pitch angle, Figure 5.4b, where the most rigid configuration demonstrated the ability to reach a steady state with fewer oscillations and a cleaner response, while the system's stability in relation to the roll angle simulations remained consistent across different configurations.

From the reported result, it is clear that the response of the two controllers is not as decoupled as expected. In configurations with lower stiffness, the Cartesian Impedance Controller exhibits a longer response time to track changes in reference values, compared to more rigid setups, as demonstrated in Figure 5.5a. This delay results in higher errors between real and desired values, attributed to increased compliance, thereby causing the two control actions to interfere with each other. Moreover, the dynamic nature of the entire body introduces moments around the robot's base, inducing angular velocities and changes in angles unaddressed by either controller. Coupled with the system's non-linearity, it results in an unstable system which tends to diverge. Conversely, in configurations with higher stiffness

the legs respond nearly instantaneously to new reference values with minimal errors, Figure 5.5b. This produces a more decoupled effect that, while preventing instability, does not guarantee the robustness of the response. Indeed, even a slight increase in proportional gain to $K_p = 1.5$ often leads to divergence across settings. On the other hand, choosing a lower proportional gain could mitigate the coupled effect on the system, especially in the case of the Cartesian Impedance Controller set with 500 N/m stiffness on the vertical motion, since the actual assumption of the settling time is not guaranteed in this configuration. However, it would reduce the Attitude controller's efficacy in rejecting low-frequency disturbances, due to a longer settling time.



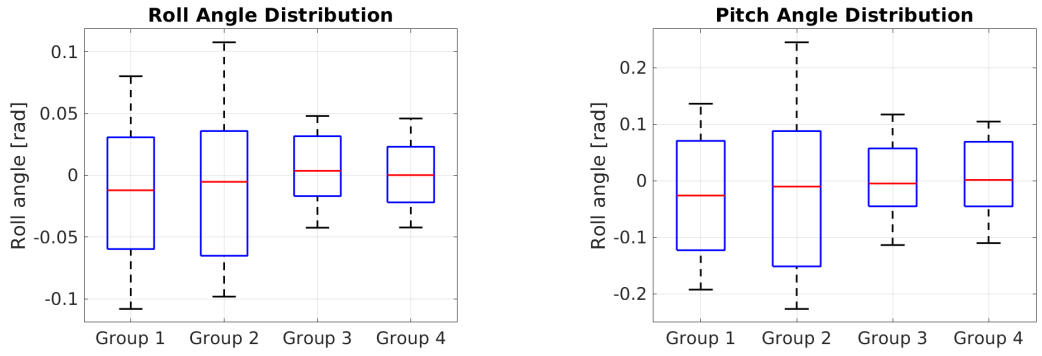
(a) (b)

Figure 5.5: It shows the comparison between the end-effector real position and the reference position of only the vertical component along the Z-axis of the Cartesian space. In (a) the Cartesian Impedance Controller is set with as in Table 4.1, instead in (b) the setting is reported in 4.3.

5.4.2 Terrain with Slopes

In this experiment, the scenario described in Section 4.3 is replicated to assess the ability of the Attitude Controller to keep the base parallel to the ground. Its parameters will be set with a proportional gain $K_{p_{Roll}} = K_{p_{Pitch}} = 0.8$ and the damping ratio to $\zeta_{Roll} = \zeta_{Pitch} = 0.7$, which is demonstrated to be the configuration with the more stable response.

In the following results, the adaptability of the Attitude Controller will be compared with the robot's behaviour when controlled only by the Cartesian Impedance Controller. In Figure 5.6, the distribution of Roll and Pitch angles is reported in multiple cases: without the Attitude Controller and with the legs



(a) (b)

Figure 5.6: Resulting Roll and Pitch angle distribution from simulated experiments. **Group 1**: Cartesian Impedance Controller set with lower stiffness (Table 4.1), Attitude Controller not enabled; **Group 2**: Cartesian Impedance Controller configured as 4.2, Attitude Controller not enabled; **Group 3**: Cartesian Impedance Controller set with stiffness 4.2, Attitude Controller enabled; **Group 4**: Cartesian Impedance Controller set with stiffness 4.3, Attitude Controller enabled.

stiffest set to the values indicated in Tables 4.1 and 4.2; with Attitude control enabled and more rigid legs. It is worth noticing that the configuration with lower stiffness on the Cartesian Impedance Controller and the Attitude controller enabled has not been tested, as its tendency to diverge was demonstrated in the previous section. Figure 5.6 shows a smaller variability of the Roll and Pitch angles compared to the cases without the Attitude Controller, indicating a reduction of the base rotation thanks to the control action of the implemented controller. Furthermore, additional simulations with higher frequencies on Eq. 4.1 are conducted for completeness. In these cases, the Attitude controller remains effective in preventing the robot from tumbling. However, significant oscillations are observed, addressed to both the simplification of the robot dynamics and the ability of the Attitude controller to mitigate disturbances which cause rapid shifts in the end-effector position. This outcome serves to reinforce the controller's intended function of rejecting disturbances with lower frequencies.

5.4.3 Conclusion

The simulations conducted in this study have provided valuable insights into the integration and performance of the Attitude Controller within the Cartesian Impedance Controller framework. Through testing, it was observed that the Atti-

tude Controller plays a crucial role in controlling the base orientation, particularly in smooth terrain with changing altitude. Results indicated that configurations with higher stiffness in the Cartesian Impedance Controller tended to exhibit more stable response, particularly when coupled with non-critical damping in the Attitude Controller. Conversely, lower stiffness settings led to increased oscillations and instability, underscoring the intricate relationship between system dynamics and controller design. Moreover, the experiments conducted on terrains further demonstrated the effectiveness of the Attitude Controller in maintaining the base parallel to the ground. In this way, even though the stability is not guaranteed by specific strategies, the multiple contacts maintained by the Cartesian Impedance Controller, coupled with reduced oscillations, present a robust approach to ensure the Center of Mass (COM) remains within the support polygon region, thereby improving the overall stability of the robot.

Chapter 6

Experiments on the Real Robot

Contents

After implementing the controllers, it was time to test the framework on the real robot, navigating through a complex path with rough terrain and various disturbances. Initially, the Cartesian Impedance Controller is tested in its most compliant configuration, which demonstrated the best performance, showcasing its ability to actively respond to terrain deformations. Following this, the Cartesian Impedance Controller is tested alongside the Attitude controller to demonstrate its capability to adapt the robot's legs to the terrain profile.

6.1 Experiments Setup

The experiments took place inside the laboratory of the Humanoids and Human Centered Mechatronics department, where the robot CENTAURO is kept, which is equipped with several materials that can be used to craft terrains. To ensure the safety of both the user and the robot, inside the laboratory has been installed a crane structure to support the robot when it is not in operation and as a safety measure in case of faults during testing. On several occasions, it saved the robot from falling, except one. Thanks to its design, the crane is capable of moving in any direction within the laboratory area, enabling it to closely follow the robot's movements for added protection. An additional layer of protection is added through a wireless emergency mushroom button, which can instantly cut

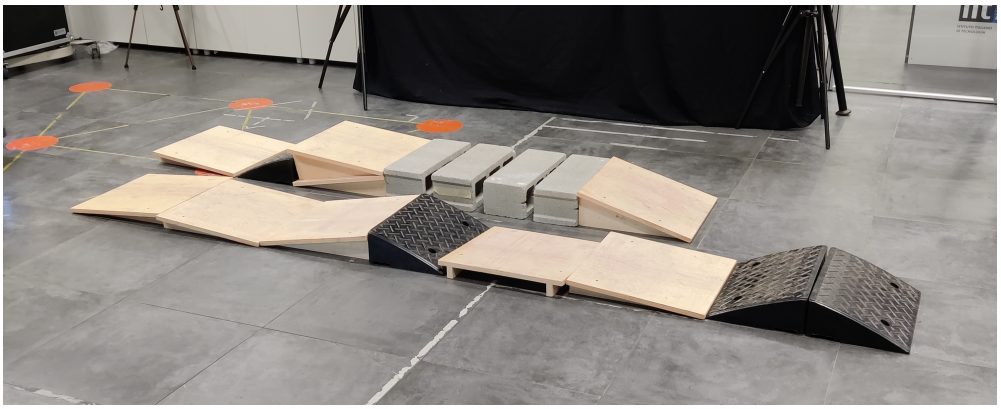


Figure 6.1: Real terrain crafted inside the laboratory to test the controllers in real conditions.

the power supply to the robot at any given moment. By doing so, it effectively disables the motors, preventing the occurrence of dangerous movements that could potentially damage the robot.

For these experiments, a custom terrain was crafted specifically for the testing environment. Within the laboratory, several triangular-shaped blocks were available for use. However, it's worth noting that there were no components with an actual rough surface that could accurately replicate disturbances encountered in the experiment described in Section 4.2. Nevertheless, the blocks were strategically arranged to create a non-coplanar surface with disturbances of different entities. In particular, the terrain is divided into two separate lines, one for the right side and one for the left side of the robot, creating a completely dissimilar profile. The path starts on the left lane with a bump of significant dimension, moving into a ramp on the right side which converges on a series of bricks. These bricks serve a dual purpose: first, they create an altitude difference between the contact points, and second, the empty spaces between them introduce disturbances for the Cartesian Impedance Controller. Meanwhile, the left side of the robot remains engaged with other bumps. The path concludes with a succession of two minor jumps on the right side, simulating high-intensity impacts on the end-effector. Figure 6.1 displays a terrain picture.

The robot's motion is still controlled through the joystick, as described in Section 4.2, activating the additional control plugin *omnisteering*. However, in this case, the robot velocity was limited to a max of 0.5 m/s, due to the limitation imposed by the supporting crane and to safely test the controllers without any risk of damaging the robot.

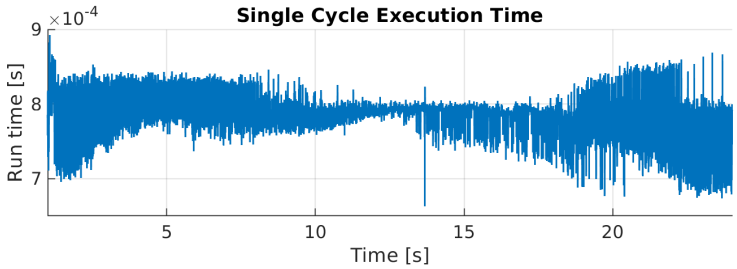


Figure 6.2: It represents the time required by the control plugin to execute one cycle of the `Run()` state, during the time it was enabled.

6.2 Cartesian Impedance Controller on Real Robot

As specified in the implementation process, the additional framework developed is designed to be real-time safe, ensuring that the computational time required to execute instructions remains low. Upon integration into the *XBot2* framework, the controller's code operates on the real-time thread, with a rate of 2 ms, as specified in the configuration file, Appendix A.3. This means that the computational time needed to execute the code must fall within this time frame. To verify the actual time required by each control plugin to execute a complete cycle, the *Xbot2* framework provides a ROS topic for monitoring. By following specific rules during the code implementation, a relatively stable computational time of approximately 0.8 ms was achieved. This was confirmed by observing the output of the topic concerning the computational time of the Cartesian Impedance Controller, as displayed in Figure 6.2

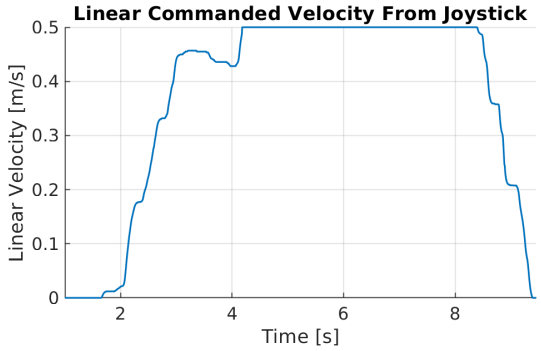
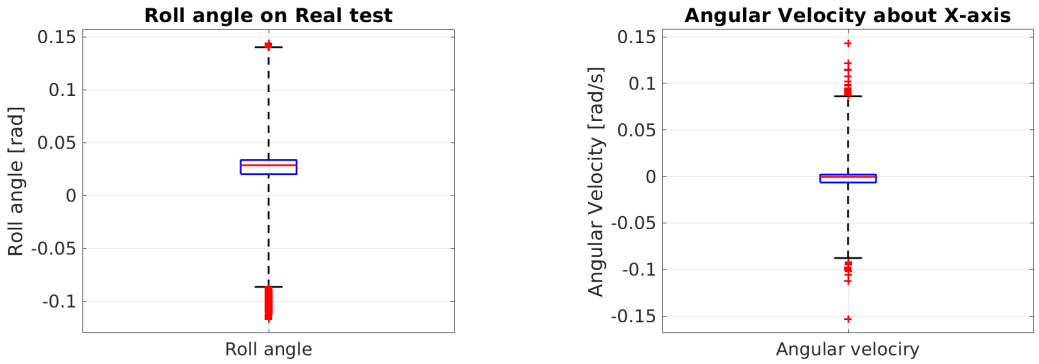


Figure 6.3: Linear velocity which is the output of the joystick command. It represents the velocity on the longitudinal axis of the Cartesian space.

In the initial attempt to navigate the terrain, the robot is equipped only with the Cartesian Impedance Controller, configured as detailed in Table 4.1. Begin-



(a) (b)

Figure 6.4: Roll angle and angular velocity distribution resulting from the test of the Cartesian Impedance Controller on the real environment. The angular velocity reported refers to the only component along the X-axis of the Cartesian space.

ning from a stationary position, the robot's velocity is gradually increased manually, following a ramp-like curve, until it reaches maximum speed, to mitigate the tumbling effect described in Section 4. This process is illustrated in Figure 6.3. Throughout the experiment, the robot demonstrated a remarkable ability to cross the proposed terrain at a relatively high speed, reacting in a compliant way to the obstacles of the environment. The Cartesian Impedance Controller was able to actively modify the leg position to better absorb the impact with both bumps and jumps. Furthermore, the compliant behaviour facilitated by the lower stiffness setting effectively reduces oscillations about the base, obtaining results consistent with the simulation, as evidenced by the box plot of the roll angle and angular velocity in Figure 6.4. In addition, Figure 6.5 shows that the four legs remain always in contact with the ground, except for one spike on the rear right leg during the in-air phase of the jump following the brick section.

6.3 Attitude Controller on the Real Robot

In this scenario, the Attitude Controller was tested on the real robot in the following configuration: Cartesian Impedance Controller set as in Tab. 4.3, with high stiffness on the translational Z-axis motion, while the Attitude controller has a proportional gain $K_p = 0.8$ and a damping factor of $\zeta = 0.7$. In this case, due to the instability of the response demonstrated in the simulation experiments, the forward motion of the robot was limited to a lower value, and slightly increased throughout the experiments to avoid any possible damage to the robot.

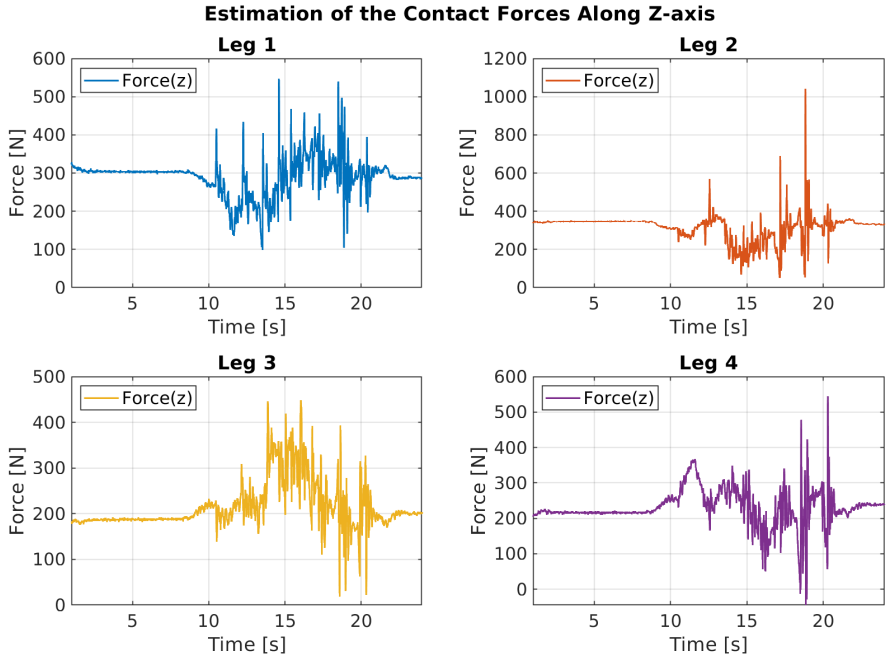


Figure 6.5: Contact force estimation in the experiment on the real terrain. In this Figure, only the vertical component of the contact wrench for each leg is plotted. These forces are expressed with respect to the local ankle yaw link frame.

The initial experiment conducted on the custom terrain demonstrated the effectiveness of the Attitude controller in accurately mapping the terrain profile. This mapping capability enables the controller to dynamically adjust the end-effector references based on the terrain profile, to maintain the robot's base parallel to the ground. In Figure 6.6 are reported the values of both pitch and roll angle, where is possible to see that the rotations of the base during the experiment remain limited thanks to the control action. The reference positions computed by the Attitude controller are then tracked by the Cartesian Impedance Controller, which allows the end-effector to follow the imposed trajectory to better adapt to the terrain, as shown in Figure 6.7. However, due to the controller's settings, the system demonstrates reduced capability to act as an active suspension in response to terrain disturbances. This limitation becomes more apparent in the following experiments with increased maximum speeds. At higher speeds, disturbances in the terrain, which were previously perceived as slow changes in terrain conformation, manifest now as disturbances at higher frequencies. The system's less compliant behaviour, as dictated by the Cartesian Impedance Controller, prevents effective filtering of these higher-frequency disturbances, leading to severe oscillations that the Attitude Controller attempts to counteract by continuously

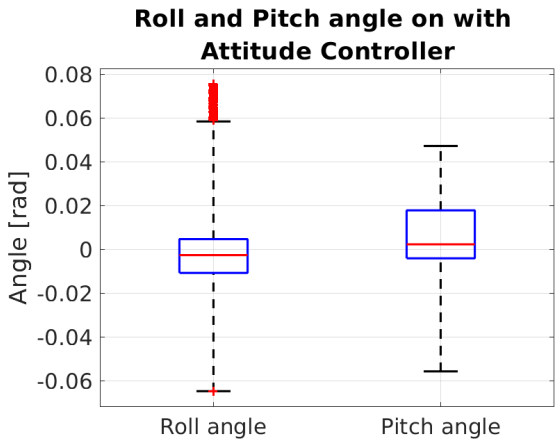


Figure 6.6: Distribution of the Roll and Pitch angle of the robot base, collected from the IMU sensor during the experiment of the Attitude Controller.

increasing opposing control actions, which results in system divergence. This observation underscores that the controller is primarily designed to address slow changes in terrain profile rather than high-frequency disturbances, in alignment with its intended purpose. However, the complex integration with the Cartesian Impedance Controller effectively mitigates the response to rapid terrain fluctuations.

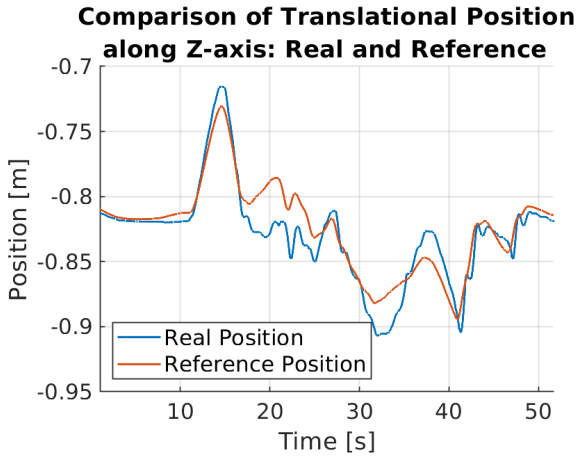


Figure 6.7: It is shown the comparison between the end-effector real position and the reference position of only the vertical component along the Z-axis of the Cartesian space.

Chapter 7

Conclusion

The objective of this thesis is to develop an active suspension mechanism for the lower body of the redundant quadruped-on-wheel robot CENTAURO, enabling it to navigate unknown and rough terrain while employing wheeled locomotion. To achieve this, a Cartesian Impedance Controller was developed to map each leg of the robot as a virtual mass-spring-damper system and control the end-effector motion across all six directions of Cartesian space. The controller is implemented to be integrated into the existing real-time frameworks developed by the Humanoids and Human Centered Mechatronics department, facilitating direct communication with the robot.

The controller underwent comprehensive testing, both in simulation and on the physical robot, providing an evaluation of its performance from both quantitative and qualitative perspectives. The tests revealed that a more compliant behaviour enables the robot's legs to effectively absorb disturbances arising from interactions, thereby reducing oscillation on the robot's base and ensuring constant contact with the terrain for improved stability. However, it was also observed that a more rigid configuration can be advantageous on smoother terrain with fewer disturbances. Nonetheless, the tests highlighted the limitation of this configuration in effectively adapting to terrain with significant slopes or ramps, which results in altitude differences across the robot.

To address such scenarios, an Attitude controller was implemented. to control the base orientation through an impedance controller regarding the roll and pitch angles of the base. By converting rotational motion into end-effector movements, the Attitude Controller employs the Cartesian Impedance Controller as a position controller to track a desired trajectory for the legs. The tests on the robot in real and simulated environments demonstrated the framework's ability to adjust leg positions according to the terrain profile, effectively mitigating the effects

of changing environments. Overall, the developed framework exhibits promising capabilities for improving the robot's navigation performance across various terrains.

7.1 Future works

The Cartesian Impedance Controller represents a foundation block of a framework that could be expanded with additional components. Throughout the conducted experiments, the stiffness settings remained static and arbitrarily chosen by the user. As highlighted in the recent research [35], dynamic adjustment of impedance settings holds the potential to minimize tracking errors, thereby enhancing the adaptability of the robotic system. A possible approach could rely on camera sensors to modify the stiffness values in real-time, based on the typology of terrain the robot has to navigate.

Moreover, some of the simplifications of the robot dynamic will be investigated to create a more accurate mathematical model of the robot, using critical points such as the Center of Mass, allowing a more precise representation of the robot's whole-body dynamics.

Ensuring the stability of the robot throughout various movements, including forward and backward acceleration, as well as during terrain navigation, is pivotal for robust performance. To achieve this, future work will rely on the implementation of stability criteria, as the ones proposed in [12, 22, 34], which leverage ground points such as the Zero Moment Point or estimations of contact force, both of which can be easily obtained from the already existing framework.

Furthermore, The complex virtual system deriving from the integration of the Cartesian Impedance Controller and the Attitude Controller will be studied to mathematically analyse how the systems are coupled with each other.

Appendix A

A.1 Detailed equation of motion of floating base dynamic

Following is a more detailed explanation of the terms introduced in the equation of motion for floating base.

$$\begin{bmatrix} \boldsymbol{M_u(q)} & \boldsymbol{M_{ua}(q)} \\ \boldsymbol{M_{au}(q)} & \boldsymbol{M_a(q)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\ddot{x}_u} \\ \boldsymbol{\ddot{q}_a} \end{bmatrix} + \begin{bmatrix} \boldsymbol{C_u(q,\dot{q})} \\ \boldsymbol{C_a(q,\dot{q})} \end{bmatrix} + \begin{bmatrix} \boldsymbol{g_u(q)} \\ \boldsymbol{g_a(q)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0_{6\times 1}} \\ \boldsymbol{\tau} \end{bmatrix} + \boldsymbol{J_c^T \lambda} + \boldsymbol{J^T F_{ext}} \tag{A.1}$$

- $\begin{bmatrix} \boldsymbol{M_u(q)} & \boldsymbol{M_{ua}(q)} \\ \boldsymbol{M_{au}(q)} & \boldsymbol{M_a(q)} \end{bmatrix} \in \mathbb{R}^{(n_b+n_j)\times(n_b+n_j)}$ is the symmetric and positive definite joint space inertia matrix. It can be sectioned into four other matrices: $\boldsymbol{M_u} \in \mathbb{R}^{n_b \times n_b}$ is the inertia matrix of the floating base, $\boldsymbol{M_a} \in \mathbb{R}^{n_j \times n_j}$ is the inertia matrix of all the actuated joints in the system and $\boldsymbol{M_{au}} = \boldsymbol{M_{ua}}^T \in \mathbb{R}^{n_j \times n_b}$ represent the coupled inertia between the base joints and the actuated joints;
- $\begin{bmatrix} \boldsymbol{\ddot{x}_u} & \boldsymbol{\ddot{q}_a} \end{bmatrix}^{\boldsymbol{T}} \in \mathbb{R}^{1\times(n_b+n_j)}$ are the generalized acceleration of the system, both floating base and joints;
- $\boldsymbol{C_u} \in \mathbb{R}^{n_b\times(n_b+n_j)}$ and $\boldsymbol{C_a} \in \mathbb{R}^{n_j\times(n_b+n_j)}$ are the floating base and actuated Coriolis matrices;
- $\begin{bmatrix} \boldsymbol{g_u(q)} & \boldsymbol{g_a(q)} \end{bmatrix}^{\boldsymbol{T}} \in \mathbb{R}^{1\times(n_b+n_j)}$ are the gravitational terms;
- $\begin{bmatrix} \boldsymbol{0_{6\times 1}} \\ \boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^{(n_b+n_j)\times 1}$ is the vector containing the torque values for the actuated joint;
- $\boldsymbol{\lambda} \in \mathbb{R}^{6n_c\times 1}$ are the contact wrenches, one for each contact point;

- $\boldsymbol{F}_{ext} \in \mathbb{R}^{6\times1}$ is the wrench of the external forces that could be applied in any point of the robot.

A.2 Mass-Spring-Damper system

A Mass-Spring-Damper system, in Fig. A.1, is a mechanical model that consists of three main components: the mass which represents the inertia of the system; a spring that connects the mass to the ground and generates a force that is proportional to the displacement of the mass from its equilibrium point; a damper model the friction of the system dissipating energy proportionally to the velocity of the system.

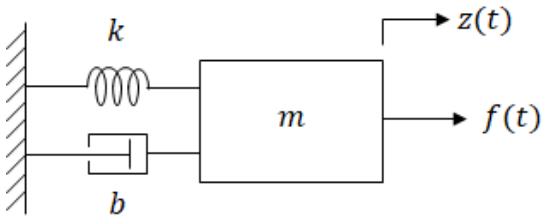


Figure A.1: Mass-Spring-Damper system

The governing equation of this system is

$$m\ddot{x}(t) + d\dot{x}(t) + kx(t) = F(t) \tag{A.2}$$

where $x(t)$ is the displacement of the mass from its equilibrium point in meters, $\dot{x}(t)$ is the velocity in (m/s), $\ddot{x}(t)$ is the acceleration in (m/s^2). The coefficients m, d, k are respectively the mass in (kg), the stiffness in (N/m) from Hooke's law, d is the damping coefficient in (Nm/s), and $F(t)$ is the external forces acting on the mass that generates the motion.

The Eq. A.2 can be written in dimensionless form as a canonical second-order differential equation as

$$\ddot{x}(t) + 2\zeta\omega_n\dot{x}(t) + \omega_n^2 x(t) = k_{dc}\omega_n^2 F(t) \tag{A.3}$$

where ω_n $(rad/s)$is defined as the natural frequency, also known as the frequency at which the system would oscillate if it were undamped. From Eq. A.2 is computed as

$$\omega_n = \sqrt{\frac{k}{m}} \tag{A.4}$$

The symbol ζ represents the damping ratio, a dimensionless quantity that characterizes the rate at which the oscillations of the system decay as a result of frictional effects. It is formulated as

$$\zeta = \frac{d}{2\sqrt{km}} \tag{A.5}$$

Tuning the parameters ζ and ω_n will modify the response of the system to any given input. A higher value of the natural frequency guarantees a quicker response of the system, while the damping ratio affects the oscillation of the system before reaching the steady state, as shown in Fig. A.2. Based on the value of ζ is possible to categorize three different responses:

- **Underdamped system**: if $0 \leq \zeta < 1$ the system will oscillate while approaching the steady state. From a frequency domain analysis, the poles defined as $s_p = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$ are complex with negative real parts, so the system is stable.
- **Critically-damped system**: if $\zeta = 1$ both the poles are real with magnitude $s_p = -\zeta\omega_n$. The response does not oscillate and will have the quickest settling time.
- **Overdamped system**: if $\zeta > 1$ the system remain stable and does not oscillates.

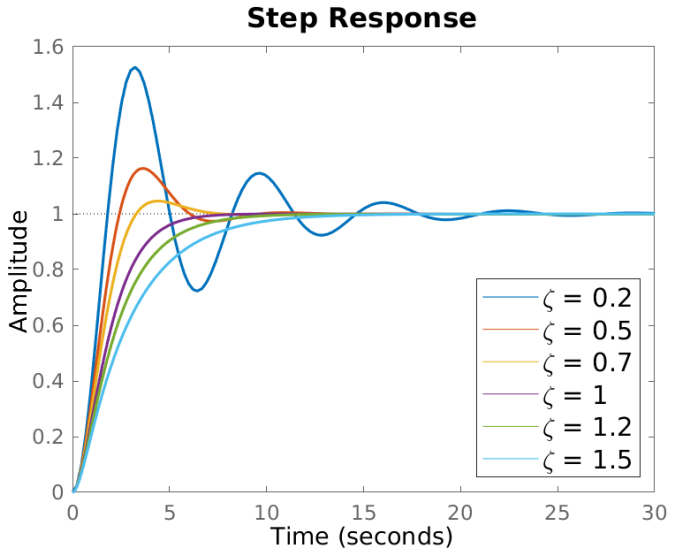


Figure A.2: Step response with different ζ value

A.3 Configuration file for the XBot2 framework

```yaml
XBotInterface:
  urdf_path: "$(rospack␣find␣centauro_urdf)/urdf/centauro.urdf"
  srdf_path: "$(rospack␣find␣centauro_srdf)/srdf/centauro.srdf"

ModelInterface:
  model_type: RBDL
  is_model_floating_base: true

# hal
xbotcore_device_configs:
  sim: "$(rospack␣find␣centauro_config)/hal/centauro_gz.yaml"
  dummy: "$(rospack␣find␣centauro_config)/hal/centauro_dummy.yaml
      "

# threads
xbotcore_threads:
  rt_main: {sched: fifo , prio: 60, period: 0.002}
  nrt_main: {sched: other, prio: 0 , period: 0.005}

# plugins
xbotcore_plugins:

  homing:
    thread: rt_main
    type: homing

  ros_io:
    thread: nrt_main
    type: ros_io

  ros_control:
    thread: nrt_main
    type: ros_control
    params: {autostart: {type: bool, value: true}}

  omnisteering:
    type: omnisteering_controller_plugin
    thread: nrt_main
    params:
```

```
        wheel_names: [wheel_1, wheel_2, wheel_3, wheel_4]
        wheel_radius: [0.124, 0.124, 0.124, 0.124]

    impedance_control:
      type: controllermanager
      thread: rt_main
      params:
        stack_path: "${PWD}/CentauroConfig/stack_problem.yaml"
        stab_controller_path: "${PWD}/CentauroConfig/
            roll_pitch_controller_config.yaml"
        stab_control_enable: 0

# global parameters
xbotcore_param:
  "/xbot/hal/joint_safety/filter_autostart": {value: true, type: bool}
  "/xbot/hal/joint_safety/filter_cutoff_hz": {value: 2.0, type: double}
  "/xbot/hal/enable_safety": {value: false, type: bool}
```

A.4 Stack of problem file

```
stack:
    - ["Leg1", "Leg2", "Leg3", "Leg4"]

constraints: ["JointLimits", "VelocityLimits"]

## Task definition ##

Leg1:
    type: "Interaction"
    name: leg_1
    active: true
    distal_link: contact_1
    base_link: base_link
    orientation_gain: 2.0
    stiffness: [10000.0, 10000.0, 500.0, 2000.0, 2000.0, 0.0]
    damping: [0.7, 0.7, 0.7, 0.7, 0.7, 0.7]

Leg2:
    type: "Interaction"
    name: leg_2
```

```
    active: true
    distal_link: contact_2
    base_link: base_link
    orientation_gain: 2.0
    stiffness: [10000.0, 10000.0, 500.0, 2000.0, 2000.0, 0.0]
    damping: [0.7, 0.7, 0.7, 0.7, 0.7, 0.7]

Leg3:
    type: "Interaction"
    name: leg_3
    active: true
    distal_link: contact_3
    base_link: base_link
    orientation_gain: 2.0
    stiffness: [10000.0, 10000.0, 500.0, 2000.0, 2000.0, 0.0]
    damping: [0.7, 0.7, 0.7, 0.7, 0.7, 0.7]

Leg4:
    type: "Interaction"
    name: leg_4
    active: true
    distal_link: contact_4
    base_link: base_link
    orientation_gain: 2.0
    stiffness: [10000.0, 10000.0, 500.0, 2000.0, 2000.0, 0.0]
    damping: [0.7, 0.7, 0.7, 0.7, 0.7, 0.7]

## Constraint Task ##

JointLimits:
    type: "JointLimits"

VelocityLimits:
    type: "VelocityLimits"
```

A.5 Attitude Controller Configuration File

```

Leg1:
  Roll:
```

```
    gain: 0.8
    relative_leg: contact_2
    damping_factor: 0.7
  Pitch:
    gain: 0.8
    relative_leg: contact_3
    damping_factor: 0.7

Leg2:
  Roll:
    gain: 0.8
    relative_leg: contact_1
    damping_factor: 0.7
  Pitch:
    gain: 0.8
    relative_leg: contact_4
    damping_factor: 0.7

Leg3:
  Roll:
    gain: 0.8
    relative_leg: contact_4
    damping_factor: 0.7
  Pitch:
    gain: 0.8
    relative_leg: contact_1
    damping_factor: 0.7

Leg4:
  Roll:
    gain: 0.8
    relative_leg: contact_3
    damping_factor: 0.7
  Pitch:
    gain: 0.8
    relative_leg: contact_2
    damping_factor: 0.7
```

Bibliography

[1] Richard J Adams and Blake Hannaford. Stable haptic interaction with virtual environments. *IEEE Transactions on robotics and Automation*, 15(3):465–474, 1999.

[2] A. Albu-Schaffer, C. Ott, and G. Hirzinger. A passivity based cartesian impedance controller for flexible joint robots - part ii: full state feedback, impedance design and experiments. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 2666–2672 Vol.3, April 2004.

[3] C. Dario Bellicoso, Marko Bjelonic, Lorenz Wellhausen, Kai Holtmann, Fabian Günther, Marco Tranzatto, Péter Fankhauser, and Marco Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018.

[4] Marko Bjelonic, C. Dario Bellicoso, Yvain de Viragh, Dhionis Sako, F. Dante Tresoldi, Fabian Jenelten, and Marco Hutter. Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots. *IEEE Robotics and Automation Letters*, 4(2):2116–2123, 2019.

[5] Arthur Bouton, Christophe Grand, and Faïz Benamar. Obstacle negotiation learning for a compliant wheel-on-leg robot. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2420–2425, 2017.

[6] Arthur Bouton, Christophe Grand, and Faïz Benamar. Design and control of a compliant wheel-on-leg rover which conforms to uneven terrain. *IEEE/ASME Transactions on Mechatronics*, 25(5):2354–2363, 2020.

[7] Stéphane Caron. Computing torques to compensate gravity in humanoid robots — scaron.info. `https://scaron.info/robotics/computing-torques-to-compensate-gravity-in-humanoid-robots.html`. [Accessed 27-03-2024].

[8] Florian Cordes, Daniel Kuehn, Christian Oekermann, Ajish Babu, Tobias Stark, and Frank Kirchner. An active suspension system for a planetary rover. 06 2014.

[9] Wenqian Du. *Motion generation of four-limb robots using whole-body torque control*. Theses, Sorbonne Université, November 2020.

[10] Wenqian Du, Mohamed Fnadi, and Faïz Benamar. Rolling based locomotion on rough terrain for a wheeled quadruped using centroidal dynamics. *Mechanism and Machine Theory*, 153:103984, 2020.

[11] Y. Fujimoto and A. Kawamura. Robust biped walking with active interaction control between robot and environment. In *Proceedings of 4th IEEE International Workshop on Advanced Motion Control - AMC '96 - MIE*, volume 1, pages 247–252 vol.1, 1996.

[12] Elena Garcia, Joaquin Estremera, and Pablo Gonzalez de Santos. A comparative study of stability margins for walking machines. *Robotica*, 20(6):595–606, 2002.

[13] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial, 2023.

[14] Christophe Grand, Faïz Amar, Frederic Plumet, and Philippe Bidaud. Stability and traction optimization of a reconfigurable wheel-legged robot. *I. J. Robotic Res.*, 23:1041–1058, 01 2004.

[15] David A. Harville. *Eigenvalues and Eigenvectors*, pages 521–588. Springer New York, New York, NY, 1997.

[16] Hugh Herr and Marko Popovic. Angular momentum in human walking. *Journal of Experimental Biology*, 211(4):467–481, 02 2008.

[17] Matt Heverly, Jaret Matthews, Matthew Frost, and Christopher Mcquin. Development of the tri-athlete lunar vehicle prototype. 01 2010.

[18] Enrico Mingo Hoffman, Alessio Rocchi, Arturo Laurenzi, and Nikos G. Tsagarakis. Robot control for dummies: Insights and examples using opensot. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 736–741, 2017.

[19] N Hogan. Impedance control: An approach to implementation. *part I-III, Journal of dynamic systems, measurement, and control*, 107(1):1–24, 1985.

[20] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C. Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016.

[21] Marco Hutter, C Remy, Mark Hoepflinger, and Roland Siegwart. High compliant series elastic actuation for the robotic leg scarleth. pages 507–514, 01 2011.

[22] Yan Jia, Xiao Luo, Baoling Han, Guanhao Liang, Jiaheng Zhao, and Yuting Zhao. Stability criterion for dynamic gaits of quadruped robot. *Applied Sciences*, 8(12), 2018.

[23] Imin Kao, Kevin Lynch, and Joel W. Burdick. *Contact Modeling and Manipulation*, pages 647–669. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[24] Navvab Kashiri, Lorenzo Baccelliere, Luca Muratore, Arturo Laurenzi, Zeyu Ren, Enrico Mingo Hoffman, Malgorzata Kamedula, Giuseppe Francesco Rigano, Jorn Malzahn, Stefano Cordasco, et al. Centauro: A hybrid locomotion and high power resilient manipulation platform. *IEEE Robotics and Automation Letters*, 4(2):1595–1602, 2019.

[25] Hamza Khan, Satoshi Kitano, Marco Frigerio, Marco Camurri, Victor Barasuol, Roy Featherstone, Darwin G. Caldwell, and Claudio Semini. Development of the lightweight hydraulic quadruped robot — minihyq. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, 2015.

[26] O. Khatib. Motion/force redundancy of manipulators. In *Proc. of the Japan-USA Symposium on Flexible Automation*, volume 1, pages 337–342, Kyoto, Japan, July 1990.

[27] Arturo Laurenzi, Davide Antonucci, Nikos G. Tsagarakis, and Luca Muratore. The xbot2 real-time middleware for robotics. *Robotics and Autonomous Systems*, 163:104379, 2023.

[28] Arturo Laurenzi, Enrico Mingo Hoffman, Luca Muratore, and Nikos G. Tsagarakis. Cartesi/o: A ros based real-time capable cartesian control framework. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 591–596, 2019.

[29] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017.

[30] Harsh Maithani, Juan Antonio Corrales Ramon, and Y. Mezouar. Predicting human intent for cooperative physical human-robot interaction tasks. pages 1523–1528, 07 2019.

[31] Luca Muratore, Arturo Laurenzi, Enrico Mingo Hoffman, and Nikos G. Tsagarakis. The xbot real-time software framework for robotics: From the developer to the user perspective. *IEEE Robotics Automation Magazine*, 27(3):133–143, 2020.

[32] Yoshihiko Nakamura and Hideo Hanafusa. Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, 09 1986.

[33] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. Unified impedance and admittance control. In *2010 IEEE International Conference on Robotics and Automation*, pages 554–561, 2010.

[34] E.G. Papadopoulos and D.A. Rey. A new measure of tipover stability margin for mobile manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3111–3116 vol.4, 1996.

[35] Mathew Jose Pollayil, Franco Angelini, Guiyang Xin, Michael Mistry, Sethu Vijayakumar, Antonio Bicchi, and Manolo Garabini. Choosing stiffness and damping for optimal impedance planning. *IEEE Transactions on Robotics*, 39(2):1281–1300, 2023.

[36] M. H. Raibert and J. J. Craig. Hybrid Position/Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, 06 1981.

[37] William Reid, Francisco Javier Pérez-Grau, Ali Haydar Göktoğan, and Salah Sukkarieh. Actively articulated suspension for a wheel-on-leg rover operating on a martian analog surface. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5596–5602, 2016.

[38] Timothy Ohm Richard Volpe, J. Balaram and Robert Ivlev. Rocky 7: a next generation mars rover prototype. *Advanced Robotics*, 11(4):341–358, 1996.

[39] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Control of legged robots with optimal distribution of contact forces. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 318–324, 2011.

[40] ETH Zurich Robotic Systems Lab. Robot dynamics lecture notes, 2017. https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf [Accessed: 12/02/2024].

[41] E. Rollins, J. Luntz, A. Foessel, B. Shamah, and W. Whittaker. Nomad: a demonstration of the transforming chassis. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 1, pages 611–617 vol.1, 1998.

[42] Claudio Semini. Hyq - design and development of a hydraulically actuated quadruped robot. 2010.

[43] Claudio Semini, Victor Barasuol, Jake Goldsmith, Marco Frigerio, Michele Focchi, Yifu Gao, and Darwin Caldwell. Design of the hydraulically-actuated, torque-controlled quadruped robot hyq2max. *IEEE/ASME Transactions on Mechatronics*, PP:1–1, 10 2016.

[44] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[45] Roland Siegwart, Pierre Lamon, Thomas Estier, Michel Lauria, and Ralph Piguet. Innovative design for wheeled locomotion in rough terrain. *Robotics and Autonomous Systems*, 40:151–162, 08 2002.

[46] Arun Kumar Singh, Rahul Kumar Namdev, Vijay Eathakota, and K. Madhava Krishna. A novel compliant rover for rough terrain mobility. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4788–4793, 2010.

[47] Shin-Min Song and Kenneth J Waldron. *Machines that walk: the adaptive suspension vehicle*. MIT press, 1989.

[48] Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

[49] Matthew Towers. Algebra 1, 2022. https://www.ucl.ac.uk/~ucahmto/0005_2021/Ch3.S5.html [Accessed: 29/02/2024].

[50] Luigi Villani and Joris De Schutter. *Force Control*, pages 161–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[51] Miomir Vukobratovic and Branislav Borovac. Zero-moment point - thirty five years of its life. *I. J. Humanoid Robotics*, 1:157–173, 03 2004.

[52] Pierre-Brice Wieber. On the stability of walking systems. In *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, 2002.