University of Genova

PhD Program in Bioengineering and Robotics

Online Control of Humanoid Robot Locomotion

by

Giulio Romualdi

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

August 2022

Daniele Pucci	Supervisor
Stefano Dafarra	Co-Advisor
Giorgio Cannata	Head of the PhD program

Thesis Jury and Reviewers:*

Marco Hutter*	Prof. Dr., ETH, Zürich, Switzerland
Jerry Pratt*	Senior Research Scientist, IHMC, Pensacola, Florida
Johannes Englsberger	Dr. Ing., DLR, Oberpfaffenhofen, Germany
Alexander Leonessa	Prof. Dr., Virginia Tech, Blacksburg, Virginia
Giorgio Cannata	Prof., Università degli Studi di Genova, Genoa, Italy
Lorenzo Rosasco	Prof., Università degli Studi di Genova, Genoa, Italy


iit
AMI
ISTITUTO ITALIANO
DI TECNOLOGIA
ARTIFICIAL AND MECHANICAL
INTELLIGENCE


Dibris
Dipartimento di Informatica, Bioingegneria,
Robotica e Ingegneria dei Sistemi

To my beloved family

Declaration

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

Giulio Romualdi
August 2022

Acknowledgements

> If I have seen further it is by
> standing on the shoulders of Giants.
>
> *Isaac Newton*

I would like to express my deepest appreciation to my supervisor, Daniele Pucci, for believing in me and for giving me the opportunity to conduct my research in the *Artificial and Mechanical Intelligence* laboratory. His guidance and advice have allowed me to grow more and more every day.

I am extremely grateful to Stefano Dafarra for all the time he spent discussing the research topics that are presented in this thesis.

I could never reach the end of my Ph.D. without Silvio. I sincerely appreciate all of your time and work in advising me on several subjects. Every day, I am struck by your unwavering drive to assist all members of the community.

Olivier Stasse deserves special recognition for welcoming me to the Gepetto team in Toulouse. Thank you Olivier for giving me this wonderful opportunity.

I cannot thank Nahuel enough. I thoroughly enjoyed every minute of our discussion.

My sincere gratitude goes to the reviewers, Marco Hutter and Jerry Pratt, for taking the time to read and review my thesis. I must thank Lorenzo Natale, Marco Maggiali, Ugo Pattacini, Vadim Tikhanoff, Luca Fiorio, and Alberto Parmigiani for their insightful comments during my Ph.D. research evaluations.

Words cannot explain how grateful I am to everyone on the *Baciotti* team. Thank you, Giuseppe, Ines, Paolo, Riccardo, and Antonello for proving to be real friends. I never expected to meet folks like you.

A special thank goes to Prashanth! We embarked on this journey together and, unexpectedly, we survived. Thank you for all our chats! I have always admired your approach to research and you have always been an inspiration to me.

It is now Nicola's turn. Thank you for being my favorite housemate and adventure buddy. This thesis would not have been possible without your consistent encouragement to improve.

I would also like to thank all the guys and girls from LAAS, especially Gabriele, Gianluca, Fanny, and Côme. You made me feel at home even if I was abroad.

Thanks should also go to current and past members of our lab: Yue, Carlotta, Claudia, Gianluca, Gabriele, Yeshasvi, Hosam, Lorenzo, Cisco, Punith, Fabio B., Diego, Kourosh, Raffaello, Gabriele, Italo, Affaf, Enrico, Tong, Milad, Anqing, Prajval, Venus, Valentino, Ahmad and Mohamed, with whom I have shared many unforgivable and great experiences.

I am sincerely thankful for the administrative support from Marta Caracalli, Lucia Betrò, and Valentina Scanarotti.

Thank you for everything, Olivia, the sole genuine love of my life. Thank you for always being with me. Without you, I would be lost.

Special thanks also go to Stefano and Carmine, old friends who are living evidence that real friendship can transcend enormous distances.

Grazie Mamma, Babbo ed Eli. Non esistono parole per esprimere la mia gratitudine per tutto quello che avete fatto e che costantemente fate. Sarete per sempre una fonte di ispirazione che mi guiderà ogni giorno per il resto della mia vita.

Giulio

Abstract

The complexity of robot dynamics and contact model are only a few of the challenges that increase the online threat to the locomotion problem. During the DARPA Robotics Challenge, a typical strategy to solve the humanoid movement challenge was to construct hierarchical systems made of numerous layers linked in cascade. Each layer computes its output taking into account the information received from the outer layer, the environment, the robot data, and a specific model of the robot and its interaction with the environment.

This thesis investigates several model-based controllers for time-critical humanoid robot motion control. Taking into account the layered control architecture, we vary the control models in a crescendo of complexity. Having in mind the importance of designing an online architecture for locomotion, we suggest a framework composed of three layers. The inner layer takes into account the entire robot model, whether kinematic or dynamic. The intermediate and outer layers take into account simpler or reduced models.

Given the inner layer, we first develop a controller that takes into account the entire rigid robot dynamical model in the situation of rigid contact with the environment. Second, we remove the rigid contact assumption and design a controller that accounts for compliant walking surfaces. Finally, we eliminate the rigid body hypothesis in some of the robot linkages and propose a controller that takes into account the robot's mechanical flexibility.

Considering the outer layers, we first describe a controller that assumes the robot behaves as a simplified model. Then, we seek to eliminate these simplifications while keeping the problem manageable online, by designing a controller that considers only a subset of the robot dynamics.

The proposed strategies are tested on real and simulated humanoid robots: the iCub and the TALOS humanoid robots.

Table of contents

List of figures

List of tables

Prologue

When the Czech writer Karel Čapek coined the term *robot* in 1920, he had no idea that a century later robots would be a fundamental component of modern civilization. Čapek's conception of robots, however, was quite different than ours. Čapek's robots were artificially assembled biological organisms that may be confused with humans. They were designed to free mankind from the shackles of physical fatigue. Nowadays, the word *robot* refers to mechanical artificial devices capable of completing tasks, whether ordinary or extraordinary. The robots were initially developed on a large commercial scale by *Unimation*, starting from 1960, to reduce human labor along the assembly lines. One decade later, the *Waseda University* presented the first humanoid robot. Its name is *WABOT-1*. Although initially, robots' objective was to replace unskilled labor in assembly lines, nowadays robots are designed also to assist humans in their everyday tasks. In the future, robotics will enable a human being to have a real-time sensation of being in a place, and being able to interact with the remote environment. This idea is often known as *telexistence*. Thanks to telexistence, a robot might instantaneously provide the sensation of human presence and care to anybody, regardless of distance. A robot, or rather a physical avatar, might deliver essential life-saving abilities in real-time to remote, disaster-stricken locations too hazardous for a worker. To promote the use of robots in disaster scenarios, in 2015, the US Defense Advanced Research Projects Agency funded the DARPA Robotics Challenge (DRC) where humanoid robots had to perform several tasks in a nuclear crisis scenario. The tasks to be completed included things like driving a utility vehicle, walking through a door, and manipulating a tool to cut a hole in a wall. Many of the biped robots fell during the testing, highlighting the technology's immaturity. On March 12, 2018, All Nippon Airways (ANA), Japan's largest airline, announced the $10M ANA Avatar XPRIZE. The competition aims to create an avatar system that can transport human presence to a remote location in real-time. In this context, online walking capabilities are pivotal. Moreover, when cooperating with a human, the

robot must guarantee a safe interaction while maintaining balance. As a consequence, researchers are driven to design accurate models of the physical interaction between the robot and its surroundings, attempting to make the problem of locomotion tractable online.

One of the most important inheritances that DRC left us with was the definition of a model-based hierarchical control architecture for humanoid robot locomotion. Each layer provides references for the inner layer by processing inputs from the robot, the environment, and the outputs of the outer layer while considering the robot's model. Assumptions and simplifications may be necessary to maintain the locomotion problem tractable online, at the risk of compromising the model's descriptiveness. As a result, the humanoid robot's behavior varies depending on which model is considered at each layer of the control architecture.

In this thesis, we investigate different model-based controllers for time-critical humanoid robot motion control. Considering the above hierarchical control architecture, we vary the considered models from simplified to complete dynamics depending on the desired task. In particular, considering the importance of designing an online architecture for locomotion, we propose a control architecture composed of three layers. From top to bottom, we denote these layers as: *trajectory optimization, simplified model control*, and *whole-body control*. We study the performance of the *simplified model control* and *whole-body control* layers when the models under consideration change. Indeed, we believe that various tasks may be performed while preserving a cascade control structure and modifying the models considered in the specific layer. We show that the locomotion task in different scenarios can be accomplished while keeping the cascade control structure and changing the models considered in the specific layers.

This thesis is divided into three parts and its structure reflects the cascade architecture.

Part I: Background & Fundamentals

This part introduces a background about the concepts exploited in the thesis

- Chapter 1 introduces the content of the thesis, along with some food for thought on art and literature. It also briefly introduces the underlying technologies used to implement the algorithms presented in this thesis.
- Chapter 2 introduces the rotation and the roto-translation groups. We also present the dynamics of a rigid body system.

- Chapter 3 presents the model of a floating base multi body system.
- Chapter 4 describes the simplified model considered to describe the locomotion of a bipedal robot.
- Chapter 5 gives the reader some notion about optimal control and non-linear optimization.
- Chapter 6 provides the literature review and defines the thesis context.

Part II: Whole-Body Controllers

In this part, we present the design of three whole-body controllers for humanoid robot locomotion.

- Chapter 7 compares whole-body controllers for locomotion on rigid surfaces. A kinematics-based and a dynamics-based whole-body controllers are proposed. The former considers the robot's kinematics to generate the desired joint positions or velocities. The latter, on the other hand, is based on the full dynamics of the robot. Due to the modularity of the two controllers, the two approaches may be interchanged depending on the low-level controller currently accessible on the robot. The experiments are performed on the Humanoid Robot iCub.
- Chapter 8 attempts to loosen the stiff contact assumption introduced in Chapter 7 and it contributes towards the modeling of compliant contacts for robot motion control. The chapter, more specifically, proposes a contact model that describes the mechanical characteristics of a visco-elastic carpet. The whole-body controller then exploits the model to compute viable joint torques, allowing the robot to accomplish a locomotion task. The architecture is validated in a simulated version of the Humanoid Robot iCub.
- Chapter 9 proposes an extension of the dynamics-based whole-body controller, presented in Chapter 7, in the case of a robot affected by inner link flexibility. We model the link flexibility as a passive under-actuated joints and consider their dynamics in the whole-body control layer. The approach is validated on the simulated torque-controlled Humanoid Robot TALOS.

Part III: From Simplified to Reduced Models Controllers

This part discusses the design of the trajectories and the simplified model control layers. Starting from simplified models' assumptions, we decided to move towards reduce-models to compute, online, the desired trajectories for the whole-body control layer.

- Chapter 10 presents and compares several simplified model-based implementations of the kinematic-based controller architecture. In particular, given a desired footsteps location and timing, the simplified model control layer implements two controllers for the tracking of the robot's center of mass: an instantaneous and a predictive controller. We compare the two control strategies on the Humanoid Robot iCub. Moreover, we show that one of the proposed implementations allows the iCub robot to reach the highest walking velocity ever achieved on such a robot.
- Chapter 11 discusses the design of a non-linear Model Predictive Controller (MPC) that aims at generating online feasible contact locations and forces for humanoid robot locomotion. More precisely, we moved from the simplified models exploited in Chapter 10 to a reduced description of the humanoid robot. Thanks to this choice, we consider the contact location adjustment directly in the dynamics stabilization problem. The proposed approach is validated on the new version of the iCub Humanoid Robot.

This research work has been carried out during my tenure as Ph.D. candidate in the *Artificial and Mechanical Intelligence* laboratory at the *Istituto Italiano di Tecnologia* in Genoa, Italy. My Ph.D. secondment has been carried out at the *Gepetto* laboratory at *LAAS-CNRS Laboratory for Analysis and Architecture of Systems* in Toulouse, France. The doctoral program has been carried out in accordance with the requirements of *University of Genoa, Italy* in order to obtain a Ph.D. title.

Summary of publications

The results of the research conducted for this thesis have been (or will be) published in peer-reviewed research publications. We also include additional material, such as a video presentation and a software repository, for each published paper.

The content of Chapter 7 and Chapter 10 appears in:

Romualdi, G., Dafarra, S., Hu, Y., and Pucci, D. (2018). A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE

Romualdi, G., Dafarra, S., Hu, Y., Ramadoss, P., Chavez, F. J. A., Traversaro, S., and Pucci, D. (2020). A Benchmarking of DCM-Based Architectures for Position, Velocity and Torque-Controlled Humanoid Robots. *International Journal of Humanoid Robotics*, 17(01):1950034

Video	`https://www.youtube.com/watch?v=FIqwAO71Fc4`
GitHub	`robotology/walking-controllers`

The content of Chapter 8 appears in:

Romualdi, G., Dafarra, S., and Pucci, D. (2021). Modeling of Visco-Elastic Environments for Humanoid Robot Motion Control. *IEEE Robotics and Automation Letters*, 6(3):4289–4296

Video	`https://www.youtube.com/watch?v=7XKQ5ZWJvYU`
GitHub	`ami-iit/romualdi-2021-ral-soft_terrain_walking`

The content of Chapter 9 will eventually appear in:

Romualdi, G., Villa, N., Dafarra, S., Pucci, D., and Stasse, O. (2022b). Control and Estimation of Link Flexibility for Humanoid Robot Motion Control. *IEEE-RAS International Conference on Humanoid Robots (Humanoids) (Submitted)*

The content of Chapter 11 appears in:

Romualdi, G., Dafarra, S., L'Erario, G., Sorrentino, I., Traversaro, S., and Pucci, D. (2022a). Online Non-linear Centroidal MPC for Humanoid Robot Locomotion with Step Adjustment. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10412–10419. IEEE

Video `https://www.youtube.com/watch?v=u7vCgE2w_vY9`
GitHub `ami-iit/paper_romualdi_2022_icra_centroidal-mpc-walking`

Aside from publications directly relevant to the main contribution of this thesis, we now list additional contributions closely related to the thesis research aim.

The following manuscript presents a computationally efficient method for online planning of bipedal walking trajectories with push recovery based on Simplified Models for locomotion. I have contributed to this dissemination by developing part of the push recovery algorithm and integrating it with the walking architecture described in [Romualdi et al., 2020].

Shafiee, M., Romualdi, G., Dafarra, S., Chavez, F. J. A., and Pucci, D. (2019). Online dcm trajectory generation for push recovery of torque-controlled humanoid robots. *IEEE-RAS International Conference on Humanoid Robots*, 2019-October:671–678

Video `https://www.youtube.com/watch?v=DyNG8S6zznI`

The following paper presents a framework for the teleoperation of humanoid robots using a novel approach for motion retargeting through inverse kinematics over the robot model. The proposed method enhances scalability for retargeting, i.e., it allows teleoperating different robots by different human users with minimal changes to the proposed system. My contribution towards this publication was concerned with the development of an interface on the walking controller to handle the information computed by the retargeting application. Furthermore, I supported the first author in the experimental procedures for the proposed architectures.

Darvish, K., Tirupachuri, Y., Romualdi, G., Rapetti, L., Ferigo, D., Chavez, F. J. A., and Pucci, D. (2019). Whole-Body Geometric Retargeting for Humanoid Robots. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 679–686. IEEE

Video `https://www.youtube.com/watch?v=yELyMYkCyNE`
GitHub `robotology/walking-teleoperation`

The journal publication mentioned below proposes an architecture for achieving telexistence and teleoperation of humanoid robots. The architecture combines several technological set-ups, methodologies, locomotion, and manipulation algorithms in a novel manner, thus building upon and extending works available in the literature. I have contributed to this dissemination by aiding the first author with the validation, software testing, and experimental analysis of the proposed architecture.

Elobaid, M., Hu, Y., Romualdi, G., Dafarra, S., Babic, J., and Pucci, D. (2020a). Telexistence and Teleoperation for Walking Humanoid Robots. pages 1106–1121

Video `https://www.youtube.com/watch?v=jemGKRxdAM8`

The following conference paper introduces a planner capable of generating walking trajectories using the centroidal dynamics and the full kinematics of a humanoid robot model. The interaction between the robot and the walking surface is modeled explicitly through a novel contact parametrization. I supported the first author in the experimental procedures for the proposed architectures.

Dafarra, S., Romualdi, G., Metta, G., and Pucci, D. (2020). Whole-Body Walking Generation using Contact Parametrization: A Non-Linear Trajectory Optimization Approach. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1511–1517

GitHub `ami-iit/dynamical-planner`

The conference publication mentioned in the following presents a contact-aided inertial-kinematic floating-base estimation for humanoid robots considering an evolution of the state and observations over matrix Lie groups. I supported the first author in the experimental validation of the proposed estimator.

> Ramadoss, P., Romualdi, G., Dafarra, S., Andrade Chavez, F. J., Traversaro, S., and Pucci, D. (2021). DILIGENT-KIO: A Proprioceptive Base Estimator for Humanoid Robots using Extended Kalman Filtering on Matrix Lie Groups. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2904–2910. IEEE
>
> Video `https://www.youtube.com/watch?v=CaEZvbR9ZcA`
> GitHub `ami-iit/paper_ramadoss_2021_icra_proprioceptive-base-estimator`

The following journal manuscript presents ADHERENT, a system architecture that integrates machine learning methods used in computer graphics with whole-body control methods employed in robotics to generate and stabilize human-like trajectories for humanoid robots. My contribution towards this dissemination was concerned with the development of the whole-body controllers to stabilize the trajectories provided by the machine learning algorithm. Furthermore, I supported the first author in the experimental procedures for the proposed architectures.

> Viceconte, P. M., Camoriano, R., Romualdi, G., Ferigo, D., Dafarra, S., Traversaro, S., Oriolo, G., Rosasco, L., and Pucci, D. (2022). ADHERENT: Learning Human-like Trajectory Generators for Whole-body Control of Humanoid Robots. *IEEE Robotics and Automation Letters*, 7(2):2779–2786
>
> Video `https://www.youtube.com/watch?v=s7-pML0ojK8`
> GitHub `ami-iit/paper_viceconte_2021_ral_adherent`

The following journal paper presents a planner to generate walking trajectories by using the centroidal dynamics and the full kinematics of a humanoid robot. The paper

extends and encompasses the authors' previous work [Dafarra et al., 2020]. Indeed, the introduced contact parametrization [Dafarra et al., 2020] is now considered as a dynamic complementary condition (DCC) and compared with other state-of-the-art methods. I supported the first author in the experimental validation for the proposed architectures.

> Dafarra, S., Romualdi, G., and Pucci, D. (2022). Dynamic Complementary Conditions and Whole-Body Trajectory Optimization for Humanoid Robot Locomotion. *IEEE Transactions on Robotics*
>
> GitHub `ami-iit/paper_dafarra_2022_tro_dcc-planner`

Code developed during the Ph.D.

The results of the research conducted for this thesis have been obtained thanks to a suit of libraries I developed and I am maintaining.

`osqp-eigen` is a simple `Eigen` wrapper for the `osqp` library [Stellato et al., 2018a]. The goal of `osqp-eigen` is to make it easier to describe a quadratic programming (QP) problem in `C++`. The library is open-source released under the *BSD-3-Clause license* and it is available at `https://github.com/robotology/osqp-eigen`.

`lie-group-controllers` is a header-only `C++` library containing controllers designed for Lie groups. The aim of the library is to hide the complexity of the design of a proportional and proportional and derivative controller for a general Lie group. For this reason the controllers implemented in `lie-group-controllers` are not restricted to SO(3) and SE(3). The library is open-source released under the *LGPL-2.1 license* and it is available at `https://github.com/ami-iit/lie-group-controllers`.

`bipedal-locomotion-framework` is a suite of libraries to achieve bipedal locomotion in humanoid robots. Many of the algorithms and the models presented in Part II and III have been implemented within this framework. `bipedal-locomotion-framework` implements also an efficient floating base inverse kinematics and dynamics. The project provides also `python` bindings. The content of the `bipedal-locomotion-framework`

library is the subject of a publication to be submitted. The library is open-source released under the *BSD-3-Clause license* and it is available at `https://github.com/ami-iit/bipedal-locomotion-framework`.

Part I

Background & Fundamentals

Chapter 1

Introduction

Humans have been fascinated by the possibility of replicating their images into an artificial system since ancient Greece. Some Greek myths, for example, tell of artificial entities who aid, through extraordinary actions, the men in combat. According to a legend, the hero Cadmus (Κάδμος), after founding Thebes, planted dragon teeth, which became artificial soldiers – Figure 1.1a. Another noteworthy example can be found in the *Argonautica* (Ἀργοναυτικά), a poem written by *Apollonius of Rhodes* (Ἀπολλώνιος Ῥόδιος). The author narrates the myth of Talos (Τάλως), an invulnerable artificial giant made of bronze. According to the myth, Talos was in charge to guard Crete against pirates and invaders and did not hesitate to plunge himself into the flames and heat it up to a very high degree before crashing into and burning his foes. These synthetic beings were not just employed in war; in fact, examples of a companion automaton may be found in Latin literature. In this regard, Pūblius Ovidius Nāsō, known as Ovid, writes about the myth of Pygmalion who fell in love with a statue, *Galatea*, he had crafted with his own hands – Figure 1.1b. In response to his prayers, the goddess Aphrodite brought Galatea to life.

It takes a long time to go from folklore to a documented project of an autonomous system. Given that the first one dates around 1495 when *Leonardo da Vinci* designed the *mechanical knight*, known as *Leonardo's robot*. The project was probably based on Leonardo's concept of the ideal human body proportions represented in the *Virtuvian Man* drawing. The first working android was built only about three centuries later by *Jacques de Vaucanson* in 1738. Jacques de Vaucanson's work can be seen as a bridge from an age in which artificial creatures were exclusively restricted to legends or projects, to another in which the ideal of developing artificial mechanical systems had become a reality. This was noticed by the Italian writer *Ippolito Nevio* who states in

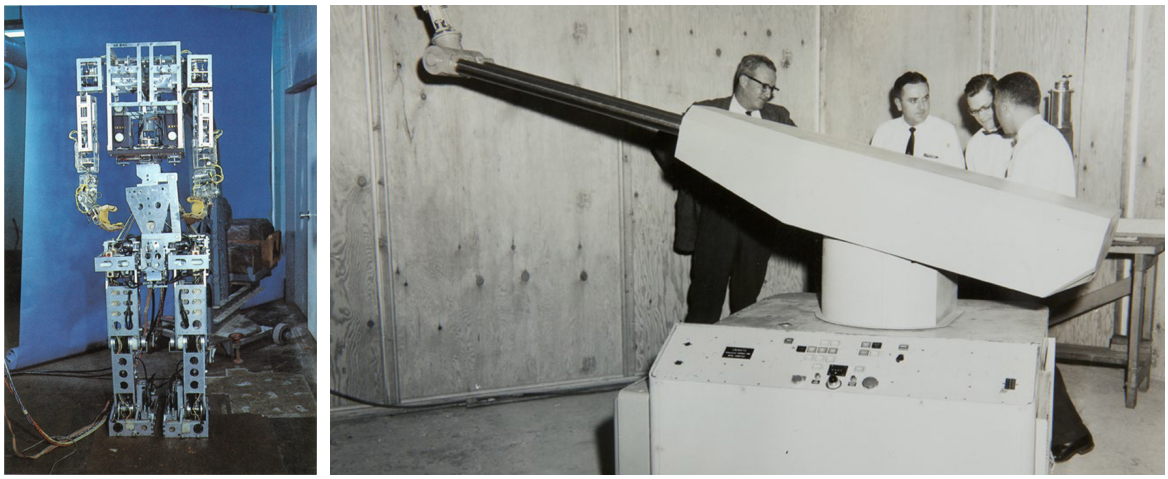
(a) (b)

Figure 1.1 (a) Jacob Jordaens (1593–1678) - *Cadmus and Minerva* (date not known). (b) Jean Raoux (1677-1734) – *Pygmalion adoring his statue* (1717).

Storia Filosofica dei Secoli Futuri that the invention of automata is the most important goal reached in the history of mankind.

The term robot was coined by the Czech writer Karel Čapek, who used it for the first time in his play *The Universal Robots of Rossum* to characterize an artificial worker. The worldwide success of Čapek's play helped the term *robot* to gain a vogue. Nowadays, the word robot has been incorporated into practically every language.

It is interesting to note that robots have had a human aspect since antiquity; yet, the first contemporary robots were used in industries and were not humanoid. The first industrial robot, *unimate* (Figure 1.2b), was constructed in 1959, while the first machine capable of human-like walking arrived only 14 years later in 1973. Its name was WABOT-1 (WAseda roBOT) [Takanishi, 2019] – Figure 1.2a. WABOT-1 weighs around 130 kg and is powered by 11 hydraulically driven joints. Walking is accomplished by segmenting the motion and storing the associated joint references in the robot's *computer*. An analog circuit attempts to match the reference joint position with the one measured by a potentiometer. Despite more than five decades since the first walking humanoid robot, bipedal locomotion remains an open problem. The complexity of the robot dynamics, the unpredictability of its surrounding environment, and the low efficiency of the robot actuation system are only a few problems that complexify the achievement of robust robot locomotion. As a result, to cope with the difficulty, researchers were driven to design simpler models [Englsberger et al., 2011; Kajita et al., 2001; Pratt et al., 2006; Vukobratović et al., 2004] that are only valid



(a) WABOT-1 (b) Unimate

Figure 1.2 (a) The WABOT1 humanoid robot (image from `www.humanoid.waseda.ac.jp/booklet/kato_2.html`). (b) A picture of the Unimate robot (image taken from `https://latestnews.plus/the-film-like-story-of-the-first-real-robot-unimate-in-history/`).

under strict assumptions. In this context, the definition of a hierarchical architecture composed of several layers was a common approach to achieve online humanoid robot control during the DARPA Robotics Challenge [Spenko et al., 2018]. In this context, each layer assumes a particular robot model, and it provides the references for the inner layer by processing the inputs from the robot, the environment, and the outputs of the outer layer.

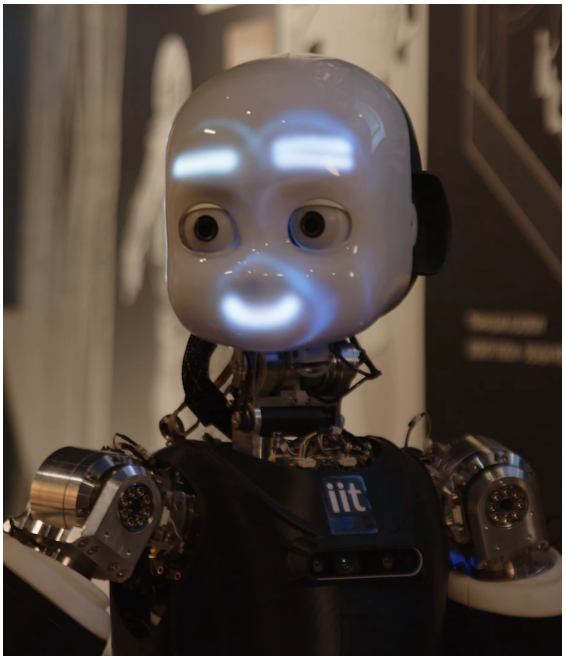
This thesis explores the hierarchical control architecture and focuses on various model-based controllers for time-critical humanoid robot motion control. Depending on the desired objective, we vary the considered models from simplified to complete robot dynamics.

In Sections 1.1 and 1.2, we briefly describe the technological resources used for the experimental validation of the thesis developments: *iCub* and *TALOS humanoid robots*. Section 1.3 introduces the notation used in the thesis

1.1 The iCub Humanoid Robot

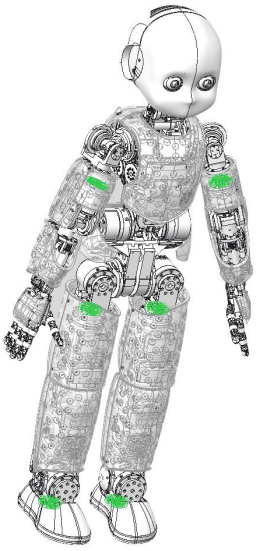
The iCub Humanoid Robot is an open source state-of-the-art robotic platform created as part of the European project RobotCub [Metta et al., 2010; Natale et al., 2017; Parmiggiani et al., 2012; Tsagarakis et al., 2007] at the Italian Institute of Technology.

(a) The iCub v2.7 robot

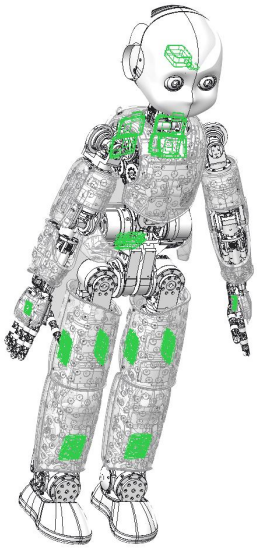


(b) The iCub v3 robot

Figure 1.3 The two versions of the iCub humanoid robot.



(a) Distribution of the six-axis force-torque



(b) Distribution of the inertial sensors

Figure 1.4 Distribution of the six embedded six-axis force-torque sensors (a) and of the inertial sensors (b) on iCub v2.7.

The iCub has been regularly updated with upgrades and new features since its first release in 2006. More than 40 partnering institutions in Europe, Asia, and the United States have received copies of iCub.

Because innovations are constantly issued and integrated into the many iCubs, all robot copies have distinct features based on their release date, the maintenance upgrades conducted over the years, and the individual customization of each iCub. The algorithms discussed in the thesis have been tested on two versions of the iCub robots, namely iCub v2.7 and iCub v3. Section 1.1.1 presents the characteristics of iCub v2.7, while Section 1.1.2 introduces iCub v3.

1.1.1 The iCub v2.7 robot

Figure 1.3a depicts the iCub humanoid robot v2.7, which is 104 cm tall and weighs 33 kg. It has 54 degrees of freedom in total, including those in the hands and eyes. Only 23 joints are used for locomotion and are distributed as follows: 4 joints in the arm, 3 of which in the shoulder and one in the elbow, 3 joints in the torso, and 6 joints in each leg. The torso and shoulder joints are mechanically coupled and driven by tendon mechanisms. All 23 joints are powered by brushless electric motors equipped with Harmonic Drive transmissions with a reduction ratio of 1/100.

A series of electronic boards known as 2FOC, EMS, and MC4Plus operate the iCub motors. The 2FOC boards use an incremental optical encoder positioned on the motor shaft to regulate the magnetic flux of a brushless motor by setting a reference PWM (Pulse Width Modulation). The EMS boards, on the other hand, are linked to the 2FOC boards and implement three control strategies, namely position, velocity, and torque control. The electronic boards are connected through an Ethernet network in daisy chain.

A three-degree-of-freedom accelerometer and three-degree-of-freedom gyroscopes are included on each motor control board. In addition, the robot's head is equipped with a full-fledged Inertial Measurement Unit, which includes a 3 DOF magnetometer, accelerometer, and gyroscope. Figure 1.4b shows how these designs offer the iCub v2.7 with a large amount of distributed inertial sensing, which has been exploited for precise calibration in [Guedelha et al., 2016].

Differently from other state-of-the-art robots [Englsberger et al., 2015b; Stasse et al., 2017], the iCub humanoid robot v2.7 does not mount pure torque sensors on the joints. As a consequence, the internal joint torques cannot be directly measured. However the robot is equipped with 6 inertial six-axis force-torque sensors - Figure 1.4a.

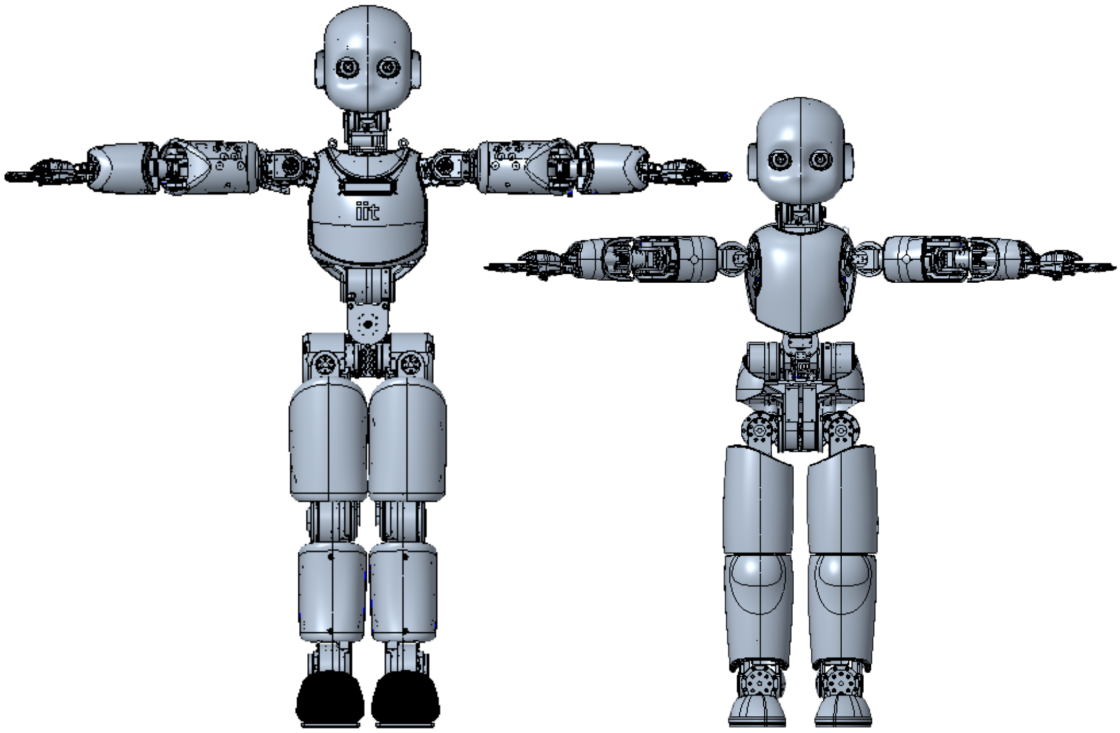


Figure 1.5 The iCub3 robot side to side to the classical iCub v2.7.

Four of them are attached to the base of each limb, while two are mounted on the robot's ankles. The location of these sensors enables the estimation of internal joint torques and external force-torque, as described in [Fumagalli et al., 2012] for a single limb and in [Traversaro, 2017, Chapter 4] for the whole-body case.

Finally, the robot head is equipped with a 4^{th} generation Intel® Core i7@1.7GHz and 8GB of RAM running Ubuntu Linux. Finally, the connection to the robot can be established through an Ethernet cable or thought a standard 5GHz Wi-Fi network.

1.1.2 The iCub v3 robot

The iCub v3 humanoid robot, depicted in Figure 1.3b, is a state-of-the-art robotic platform developed at the Italian Institute of Technology and can be seen as an evolution of the iCub v2.7 presented in Section 1.1.1.

The iCub v3 humanoid robot is larger than a traditional iCub v2.7 platform, standing 25 cm higher and weighing 22 kg heavier. The robot is 125 cm tall and weighs 52 kg. Figure 1.5 shows the different dimensions of the two platforms. The greater weight necessitates more powerful leg motors. As a consequence of the increased size

of the actuators, a new approach to the knee and ankle pitch joints was necessary. Specifically, instead of being on the same axis, the motor and actuator are separated and linked by belts. Similarly to iCub v2.7, the iCub v3 robot possesses in total 54 degrees of freedom, including those in the hands and the eyes, and only 23 joints are used for locomotion. However, unlike iCub v2.7, the torso and shoulder joints are not tendon-driven. In fact, the joints are directly connected to the motor shaft throughout the serial direct mechanisms. This provides for a larger range of motion and mechanical toughness. All 23 joints used for locomotion are powered by brushless three-phase electric motors equipped with Harmonic Drive transmissions. Each foot is made up of two rectangular parts, each measuring 25 cm in length and 10 cm in width.

Similarly to iCub v2.7, iCub v3 is equipped with a vast array of sensors, including accelerometers, gyroscopes, and force/torque sensors. Six six-axis force/torque (F/T) sensors are included in the iCub. Two are placed on each shoulder and two are attached to each foot, linking the two portions of the feet to the ankle assembly. The readouts of the F / T sensors are used to estimate the internal joint torques and the external force-torque, as presented in [Traversaro, 2017, Chapter 4].

1.1.3 Software infrastructure

A computer infrastructure is required to control the robot. To this end, the Yet Another Robot Platform (YARP) middleware [Metta et al., 2006] is exploited. YARP is an open-source multi-platform middleware whose main purpose is to allow communication between the applications (modules), which can run on different computers. More specifically, YARP is a set of libraries, protocols, and tools to keep modules and devices cleanly decoupled. Indeed, it provides an abstraction layer to interact with physical devices, such as joint encoders and F/T sensors, independently of their actual implementation. Moreover, sensor acquisition and motor controllers are provided through YARP interfaces.

Along with the YARP middleware, we took advantage of the iDynTree library [Nori et al., 2015] to design the controllers. iDynTree is a library of robot dynamics algorithms for control, estimation, and simulation. It is specifically designed for free-floating robots, but it is also possible to use it with fixed-base robots. It is written in `C++`, with Python and MATLAB interfaces. iDynTree contains support for reading and writing `URDF` files, making it possible to use it with any type of robot described by an `URDF`.

Rapid prototyping is achieved thanks to iDyntree interfaces, which can be easily integrated into off-the-shelf Python libraries and Simulink and MATLAB toolbox.

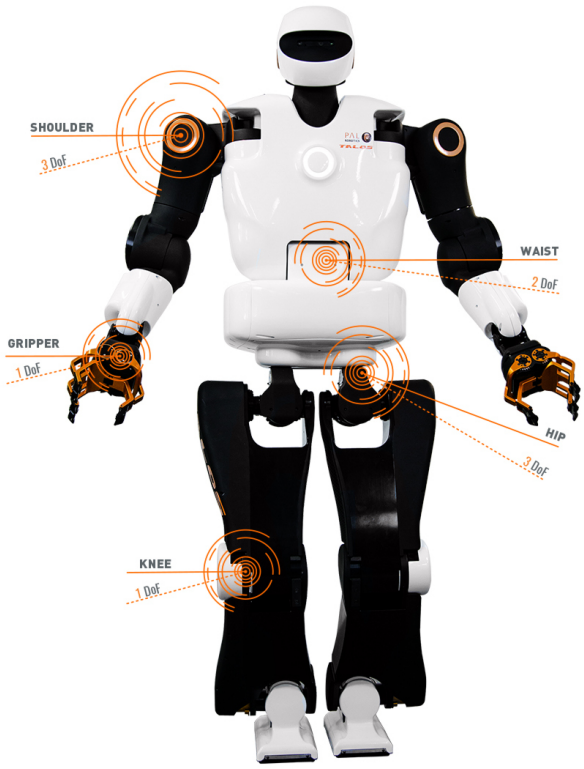


Figure 1.6 TALOS humanoid robot. Image taken from `https://pal-robotics.com/robots/talos/`

Controller prototyping also takes advantage of the Gazebo simulation environment [Koenig and Howard, 2004]. Gazebo is an open source simulator that can efficiently simulate complex multibody systems. The interface between the controller algorithms and the simulated version of the robot is handled through the corresponding YARP plugins [Mingo Hoffman et al., 2014]. The plugins make the algorithm implementation transparent. In fact, they allow testing of the very same software on both the simulator and the real robot.

1.2 The TALOS Humanoid Robot

TALOS is a state-of-the-art humanoid robot that integrates the latest cutting-edge robotics technology [Stasse et al., 2017] designed and developed by PAL-Robotics – Figure 1.6. By construction, the robot is capable of interacting with a human environment and targeting industrial applications. Since its release, several versions

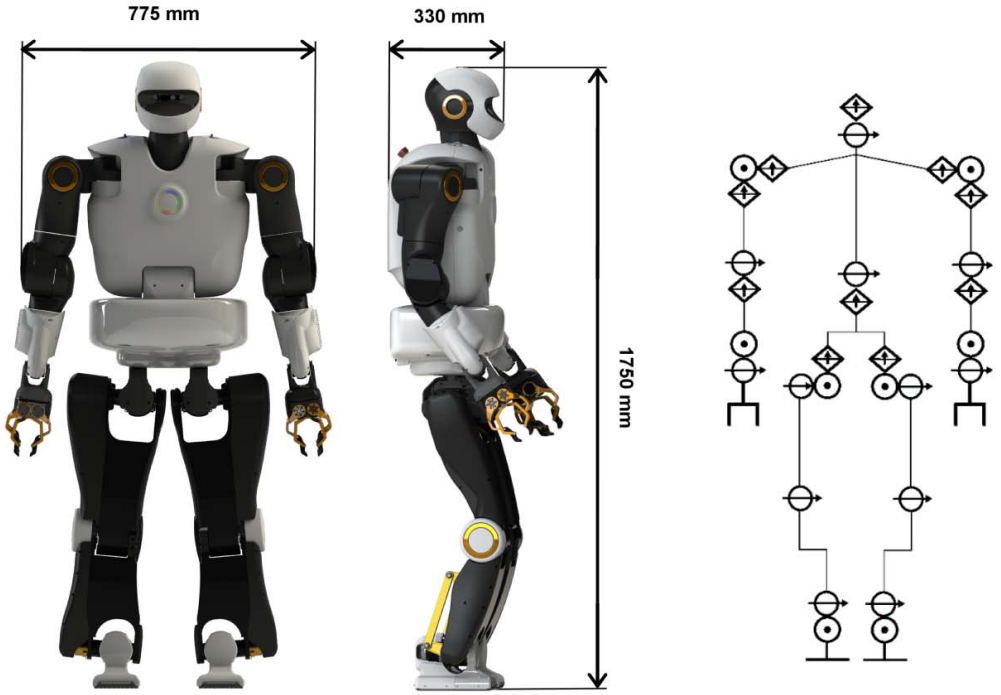


Figure 1.7 Kinematics of Pyrène robot. Image taken from [Stasse et al., 2017]

of TALOS have been produced, the following is the detailed description of *Pyrène*, the first robot in the TALOS series built by PAL-Robotics, currently available and maintained by the Gepetto group at LAAS-CNRS in Toulouse. The content of this section is inspired by the official presentation of the robot available in Stasse et al. [2017].

Figure 1.7 presents the kinematics of Pyrène, which weights 95 kg and is 1.75 m tall. It has 32 actuated degrees of freedom in total and they are distributed as follows: 2 joints in the head, 7 joints in the arm, 3 of which in the shoulder, 2 in the elbow, and 2 in the wrist, and 1 joint in the gripper. 2 joints allow controlling the pitch and yaw torso motion. Each leg has 6 joints.

Pyrène is equipped with Cobalt-Nickel-Manganese (Li-CNM) batteries that are capable of providing 75 V with a capacity of 15 A h. In the case a more power-consuming motion needs to be performed, the batteries can deliver current peaks of 150 A.

Figure 1.8 presents the actuator structure of the TALOS robot. Each brushless DC motor is connected to a harmonic drive, which is attached to a torque sensor. The torque sensor is connected to a link. Unlike iCub, each joint motor assembly is equipped with a torque sensor that can directly measure the torque applied on the

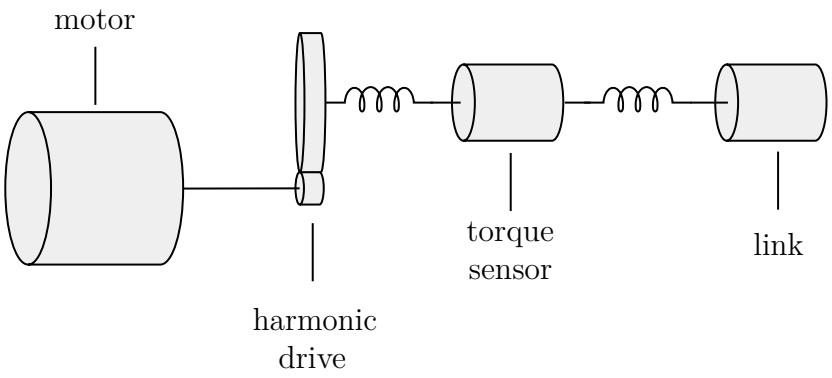


Figure 1.8 Structure of TALOS actuator

load side. Furthermore, two high-precision encoders (19 bits each) measure the motor and joint positions. Pyrène mounts an IMU at the level of its waist. This is involved in the estimation of the robot's base position and orientation. Similar to iCub, Pyrène is equipped with 6-axis force/torque sensors at the level of the hands and the feet. These sensors are often exploited to measure the contact force that the robot acts on the surrounding environment.

The robot comes with two processors, each with a dual i7 CPU running at 2.8 GHz. Each CPU contains two cores and is hyperthreaded, giving the machine a total of eight cores. The communication between the motors and sensors boards and the computers is implemented with an EtherCAT network [IEC 61158-1, 2019]. Because of TALOS' EtherCAT communication network, control loops can run at 2 kHz and up to 5 kHz, allowing for extremely reactive and dynamic motions.

Pyrène runs Ubuntu 18.04 LTS as its operating system. The ROS control system was used to develop the robot's low-level system. The multimedia system uses stacks to provide navigation and map building. Furthermore, a complete Gazebo [Koenig and Howard, 2004] simulation environment is provided.

1.3 Notation

Throughout the thesis we will use the following notation.

- The i_{th} component of a vector x is denoted as x_i.
- The transpose operator is denoted by $(\cdot)^\top$.

- Given a function of time $f(t)$ the dot notation denotes the time derivative, i.e. $\dot{f} := \frac{\mathrm{d}f}{\mathrm{d}t}$. Higher-order derivatives are denoted by a corresponding amount of dots.
- $I_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix of dimension n.
- $0_{n\times n} \in \mathbb{R}^{n\times n}$ denotes a zero matrix, while $0_n = 0_{n\times 1}$ is a zero column vector of size n.
- e_i is the canonical base in $\mathbb{R}^n$, i.e., $e_i = [0, 0, \ldots, 1, 0, \ldots, 0]^\top \in \mathbb{R}^n$, where the only unitary element is in position i. Throughout the thesis, n will be 3 or 6 depending on the context.
- The operator $\times$ defines the cross product in $\mathbb{R}^3$.
- The weighted L2-norm of a vector $v \in \mathbb{R}^n$ is denoted by $\|v\|_\Gamma$, where $\Gamma \in \mathbb{R}^{n\times n}$ is a positive define matrix.
- $\mathcal{I} = (o_\mathcal{I}, [\mathcal{I}])$ is a fixed inertial frame with respect to (w.r.t.) which the robot's absolute pose is measured. Its z axis is supposed to point against gravity, while the x direction defines the forward direction. $o_\mathcal{I}$ denotes the origin of the frame and $[\mathcal{I}]$ its orientation.
- Given the inertial frame $\mathcal{I} = (o_\mathcal{I}, [\mathcal{I}])$ and a frame $B = (o_B, [B])$, we define $B[\mathcal{I}] = (o_B, [\mathcal{I}])$ as the frame having its origin in o_B and the orientation as the inertial frame.
- ${}^A R_B \in SO(3)$ and ${}^A H_B \in SE(3)$ denote the rotation and transformation matrices that transform a vector expressed in the B frame, ${}^B x$, into a vector expressed in the A frame, ${}^A x$.
- ${}^D \mathrm{v}_{A,D} \in \mathbb{R}^6$ is the relative velocity between frame A and D, whose coordinates are expressed in frame D.
- ${}_D \mathrm{f} \in \mathbb{R}^6$ is the 6D force applied in D, whose coordinates are expressed in D.
- $x_{\mathrm{CoM}} \in \mathbb{R}^3$ is the position of the center of mass relative to $\mathcal{I}$.

Chapter 2

Rigid Body System Modeling

In this chapter, we introduce the rotation group SO(3) and the rototranslation (Euclidean) group SE(3). For each group. These two entities play a crucial role in modeling the dynamics of a rigid body system. For each group, we present the definition and analyze its properties. The chapter also presents the dynamics of a rigid body system subject to an external force. The chapter is organized as follows. Section 2.1 introduces the rotation group, Section 2.2 presents the rototranslation group SE(3) and the dynamics of the rigid body system. Finally, in Section 2.4, we guide the reader through the similarity between the rotation and the Euclidean groups. We then introduce the *Lie Group* as a generalization of the rotation and roto-translation sets. A more rigorous introduction of *Lie Group* theory is discussed in Appendix A.

2.1 The Rotation group

The set of rotation matrices SO(3) represents the set of $\mathbb{R}^{3\times3}$ orthogonal matrices with determinant equal to one, namely

$$\mathrm{SO}(3) := \{\, R \in \mathbb{R}^{3\times3} \mid R^\top R = I_3,\ \det(R) = 1 \,\}. \tag{2.1.1}$$

It is worth noting that SO(3) is a *group* under the product operator, i.e., $(\mathrm{SO}(3), \cdot)$. Indeed, the product of two rotation matrices is still a rotation matrix. Its identity element is the 3×3 identity matrix I_3 and for each element $R \in \mathrm{SO}(3)$ there exists the inverse of R, i.e., $R^{-1} = R^\top \in \mathrm{SO}(3)$.

Given two frames A and B, an element ${}^{A}R_{B} \in \mathrm{SO}(3)$ denotes the coordinate transformation from frame B to A. ${}^{A}R_{B}$ only depends on the relative orientation between the frame orientations $[A]$ and $[B]$.

Given a 3D vector ${}^{B}p$ whose coordinates are expressed in the frame B, applying ${}^{A}R_{B}$ to ${}^{B}p$ results in the coordinate transformation from the frame B to the frame A, namely ${}^{A}p = {}^{A}R_{B}{}^{B}p$ – a generalization of this concept is described in Appendix A.2.

2.1.1 Angular velocity

Given a smooth trajectory $R(t) \in \mathrm{SO}(3)$ with $t \in \mathbb{R}$ such that $R(0) = I_3$, we define the time derivative of the rotation matrix as $\dot{R}(t)$. Given the SO(3) orthogonality condition, $R(t)R(t)^{\top} = I_3$, the time derivative writes as

$$\dot{R}(t)R(t)^{\top} + R(t)\dot{R}(t)^{\top} = 0_{3\times 3}, \tag{2.1.2}$$

which can be rearranged as $\dot{R}(t)R(t)^{\top} = -\left(\dot{R}(t)R(t)^{\top}\right)^{\top}$. It is worth noting that $\dot{R}(t)R(t)^{\top}$ is a skew-symmetric matrix, and thus (2.1.2) can be rewritten as

$$\dot{R}(t) = (\omega(t)\times)R(t), \tag{2.1.3}$$

where $\omega(t)\times$ belongs is a skew-symmetric matrix of the form

$$\omega\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2.1.4}$$

If $R(t) = I_3$, Equation (2.1.3) becomes $\dot{R}(t) = \omega\times$. Hereafter we denote the set of the 3D skew-symmetric matrices with $\mathfrak{so}(3)$:

$$\mathfrak{so}(3) := \left\{ S \in \mathbb{R}^{3\times 3} \mid S^T = -S \right\}. \tag{2.1.5}$$

We notice that an element of $\mathfrak{so}(3)$ is uniquely identified by a 3D vector – see Equation (2.1.4). We now introduce *hat* as

$$\omega^{\wedge} := \omega \times . \tag{2.1.6}$$

The inverse of the *hat* map is denoted as *vee* and it is defined as

$$(\omega\times)^{\vee} := \omega. \tag{2.1.7}$$

A more rigorous definition of the *hat* and *vee* operators is provided in Equation (A.3.3).

Given a smooth trajectory ${}^{A}R(t)_{B} \in \mathrm{SO}(3)$ representing a time-varying coordinate transformation from the frame B to the frame A.

We define *right trivialized angular velocity*, denoted by ${}^{A}\omega_{A,B}\times \in \mathfrak{so}(3)$ as

$${}^{A}\omega_{A,B}\times := {}^{A}\dot{R}_{B}\,{}^{A}R_{B}^{\top}. \tag{2.1.8}$$

The *right trivialized angular velocity* is equal to the angular velocity of the frame A with respect to the frame B whose coordinates are expressed in A.

We define *left trivialized angular velocity*, denoted with ${}^{B}\omega_{A,B}\times \in \mathfrak{so}(3)$ as

$${}^{B}\omega_{A,B}\times := {}^{A}R_{B}^{\top}\,{}^{A}\dot{R}_{B}. \tag{2.1.9}$$

The *left trivialized angular velocity* is equal to the angular velocity of the frame A with respect to the frame B whose coordinates are expressed in B.

It is worth recalling that the right and left trivialized angular velocities can also be computed by time differentiating the orthogonality conditions of the form ${}^{A}R_{B}\,{}^{A}R_{B}^{\top} = I_3$ and ${}^{A}R_{B}^{\top}\,{}^{A}R_{B} = I_3$, respectively.

2.1.2 Exponential and Logarithmic map

Given a unit vector[1] $u \in \mathbb{R}^3$ and an angle $\theta \in \mathbb{R}$ the associated rotation matrix is given by the exponential map as

$$R = \mathrm{Exp}(\theta u) := I_3 + \sin(\theta)(u\times) + (1-\cos(\theta))(u\times)^2, \tag{2.1.10}$$

where (2.1.10) is the well-known *Rodrigues' Rotation Formula* [Murray et al., 1994].

Similarly, the logarithmic map is given by

$$\theta u = \mathrm{Log}(R) := \frac{\theta(R-R^{\top})^{\vee}}{2\sin(\theta)}; \quad \theta = \cos^{1}\left(\frac{\mathrm{tr}(R)-1}{2}\right). \tag{2.1.11}$$

[1]Given a vector $v \in \mathbb{R}^n$ we call it unit vector if its norm is equal to 1. i.e., $v^{\top}v = 1$.

Where $\mathrm{tr}(R)$ represents the trace of the matrix R. It is worth noting that the Log map applied on a rotation matrix returns the axis times the angle of the axis-angle representation of the rotation. A detailed and more rigorous definition of the Exponential and Logarithmic operators is discussed in Appendix A.6

2.1.3 The adjoint representation

Considering the left and right trivialized angular velocities ${}^{B}\omega_{A,B}\times \in \mathfrak{so}(3)$ and ${}^{A}\omega_{A,B}\times \in \mathfrak{so}(3)$ introduced in Equation (2.1.9) and (2.1.8), respectively. We introduce the adjoint representation as $\mathrm{Ad}_{{}^{A}R_{B}}\left({}^{B}\omega_{A,B}\times\right)$

$$ {}^{A}\omega_{A,B}\times = \mathrm{Ad}_{{}^{A}R_{B}}\left({}^{B}\omega_{A,B}\times\right) = {}^{A}R_{B}\left({}^{B}\omega_{A,B}\times\right){}^{A}R_{B}^{\top}. \tag{2.1.12} $$

A more general definition of the adjoint representation is presented in Equation (A.7.1), In other words, the adjoint representation $\mathrm{Ad}_{{}^{A}R_{B}}\left({}^{B}\omega_{A,B}\times\right)$ maps a left trivialized angular velocity into a right trivialized one, i.e., a body frame velocity into an inertial frame velocity. As discussed in detail in Appendix A.7, it is possible to show that the adjoint representation is a linear transformation, and as a consequence it can be uniquely represented by a matrix. We define such a matrix as *adjoint matrix* and we denote it with $\mathbf{Ad}_{{}^{A}R_{B}}$:

$$ \mathbf{Ad}_{{}^{A}R_{B}} = {}^{A}R_{B}. \tag{2.1.13} $$

Indeed, we can prove the identity $R(\omega\times)R^{\top} = (R\omega)\times$ by letting the right-hand side term act upon an arbitrary vector $v \in \mathbb{R}^3$

$$ \begin{aligned} \left[(R\omega)\times\right] v &= (R\omega) \times v && \text{(2.1.14a)} \\ &= (R\omega) \times (RR^{\top}v) && \text{(2.1.14b)} \\ &= R\left[\omega \times (R^{\top}v)\right] && \text{(2.1.14c)} \\ &= R(\omega\times)R^{\top}v, && \text{(2.1.14d)} \end{aligned} $$

where we use the fact that for any rotation matrix R and vectors a and b we have $(Ra) \times (Rb) = R(a \times b)$.

2.2 The Euclidean group

The Euclidean set SE(3) is defined as

$$\mathrm{SE}(3) := \left\{ \begin{bmatrix} R & p \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \mid R \in \mathrm{SO}(3), p \in \mathbb{R}^3 \right\}. \tag{2.2.1}$$

SE(3) is a group under the product operator. The identity element of the $(\mathrm{SE}(3), \cdot)$ group is the 4×4 identity matrix I_4. Given two frames A and B, an element of ${}^{A}H_{B} \in \mathrm{SE}(3)$ describes the position and orientation of the frame B with respect to the other frame A and it writes as

$${}^{A}H_{B} := \begin{bmatrix} {}^{A}R_{B} & {}^{A}o_{B} \\ 0_{1\times 3} & 1 \end{bmatrix}. \tag{2.2.2}$$

Given a 3D vector ${}^{B}p$ whose coordinates are written in B and the same vector ${}^{A}p$ whose coordinates are written in A. We define the the *homogeneous representation* of ${}^{A}p$ and ${}^{B}p$ as ${}^{A}\bar{p} := ({}^{A}p; 1) \in \mathbb{R}^4$ and likewise for ${}^{B}\bar{p}$. Then

$${}^{A}\bar{p} = {}^{A}H_{B}\,{}^{B}\bar{p}, \tag{2.2.3}$$

which is the matrix form of ${}^{A}p = {}^{A}R_{B}\,{}^{B}p + {}^{A}o_{B}$. In this context ${}^{A}H_{B}$ is often called *homogeneous transformation*.

2.2.1 6D spatial velocity

Given a smooth trajectory ${}^{A}H_{B}(t) \in \mathrm{SE}(3)$ we define the time derivative of the homogeneous transformation as ${}^{A}\dot{H}_{B}$. A more *compact* representation of ${}^{A}\dot{H}_{B}$ can be obtained multiplying it by the inverse of ${}^{A}H_{B}$ on the left or on the right. In both cases, the result element is the so called *6D spatial velocity* and it is an element of $\mathfrak{se}(3)$ Where the set $\mathfrak{se}(3)$ is defined as

$$\mathfrak{se}(3) := \left\{ \begin{bmatrix} \omega\times & v \\ 0_{1\times 3} & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \mid \omega\times \in \mathfrak{so}(3), v \in \mathbb{R}^3 \right\}. \tag{2.2.4}$$

Hereafter the *6D spatial velocity* is often denoted also a *twist*. We notice that an element of $\mathfrak{se}(3)$ is uniquely identified by a 6D vector. Similar to what we discussed for the angular velocity (Section 2.1.1), we now introduce the *hat* operator. Given

$\mathrm{v} = \left[v^\top, \omega^\top\right]^\top \in \mathbb{R}^6$. We define the *hat* operator as

$$\mathrm{v}^\wedge := \begin{bmatrix} v \\ \omega \end{bmatrix}^\wedge = \begin{bmatrix} \omega\times & v \\ 0_{1\times 3} & 0 \end{bmatrix}. \tag{2.2.5}$$

The inverse of the *hat* map is denoted as *vee* and it is defined as

$$\begin{bmatrix} \omega\times & v \\ 0_{1\times 3} & 0 \end{bmatrix}^\vee := \mathrm{v}. \tag{2.2.6}$$

A more rigorous definition of the *hat* and *vee* operators is provided in Equation (A.3.3).

Considering a smooth curve ${}^A H_B(t) \in \mathrm{SE}(3)$ representing a time-varying homogeneous transformation from the frame B to the frame A. We define the *right trivialized spatial velocity*, denoted with ${}^A\mathrm{v}^\wedge_{A,B} \in \mathfrak{se}(3)$ as

$${}^A\mathrm{v}^\wedge_{A,B} := {}^A\dot{H}_B\, {}^A H_B^{-1} = \begin{bmatrix} {}^A\dot{R}_B & {}^A\dot{o}_B \\ 0_{1\times 3} & 0 \end{bmatrix} \begin{bmatrix} {}^A R_B^\top & -{}^A R_B^\top {}^A o_B \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{2.2.7a}$$

$$= \begin{bmatrix} {}^A\dot{R}_B\, {}^A R_B^\top & {}^A\dot{o}_B - {}^A\dot{R}_B\, {}^A R_B^\top {}^A o_B \\ 0_{1\times 3} & 0 \end{bmatrix}. \tag{2.2.7b}$$

Note that ${}^A\dot{R}_B\, {}^A R_B^\top$ appearing on the right hand side of (2.2.7) is skew symmetric and it is equal to (2.1.8). We now define ${}^B v_{A,B}$ and ${}^B\omega_{A,B} \in \mathbb{R}^3$ as

$${}^A v_{A,B} := {}^A\dot{o}_B - {}^A\dot{R}_B\, {}^A R_B^\top {}^A o_B \tag{2.2.8a}$$

$${}^A\omega^\wedge_{A,B} := {}^A\dot{R}_B\, {}^A R_B^\top. \tag{2.2.8b}$$

The *right trivialized* velocity of frame B with respect to the frame A is then given by

$${}^A\mathrm{v}_{A,B} := \begin{bmatrix} {}^A v_{A,B} \\ {}^A\omega_{A,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.2.9}$$

If the frame A is the inertial frame and B a frame rigidly attached to the body, the right trivialized spatial velocity is equivalent to the so-called *inertial velocity*. The angular term of ${}^A\mathrm{v}_{A,B}$, ${}^B\omega_{A,B}$, is the angular velocity of the frame B with respect to the frame A, whose coordinates are written in A. The linear component, ${}^A v_{A,B}$ is the velocity, expressed in A, of a point belonging to the rigid body that is instantaneously coincident with the origin o_A. To give the reader a better understanding, ${}^A v_{A,B}$ is the

velocity of a *rigid extension* of the rigid body that extends up to the origin of the inertial frame A – see Figure 2.1.

We define the *left trivialized spatial velocity*, denoted with ${}^{B}\mathrm{v}^{\wedge}_{A,B} \in \mathfrak{se}(3)$, as

$$
\begin{aligned}
{}^{B}\mathrm{v}^{\wedge}_{A,B} := {}^{A}H_{B}^{-1\,A}\dot{H}_{B} &= \begin{bmatrix} {}^{A}R_{B}^{T} & -{}^{A}R_{B}^{T\,A}o_{B} \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}\dot{R}_{B} & {}^{A}\dot{o}_{B} \\ 0_{1\times 3} & 0 \end{bmatrix} && \text{(2.2.10a)} \\
&= \begin{bmatrix} {}^{A}R_{B}^{T\,A}\dot{R}_{B} & {}^{A}R_{B}^{T\,A}\dot{o}_{B} \\ 0_{1\times 3} & 0 \end{bmatrix}. && \text{(2.2.10b)}
\end{aligned}
$$

Note that ${}^{A}R_{B}^{T\,A}\dot{R}_{B}$ appearing on the right hand side of (2.2.10) is skew symmetric and it is equal to (2.1.9). Let us now define ${}^{B}v_{A,B}$ and ${}^{B}\omega_{A,B} \in \mathbb{R}^3$ so that

$$
{}^{B}v_{A,B} := {}^{A}R_{B}^{T\,A}\dot{o}_{B}, \tag{2.2.11a}
$$

$$
{}^{B}\omega^{\wedge}_{A,B} := {}^{A}R_{B}^{T\,A}\dot{R}_{B}. \tag{2.2.11b}
$$

The *left trivialized* velocity of frame B with respect to frame A is

$$
{}^{B}\mathrm{v}_{A,B} := \begin{bmatrix} {}^{B}v_{A,B} \\ {}^{B}\omega_{A,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.2.12}
$$

If the frame A is the inertial frame and B a frame rigidly attached to the body, the right trivialized spatial velocity is equivalent to the so-called *body velocity*. ${}^{B}\omega_{A,B}$ is the angular velocity of the frame B with respect to the frame A, whose coordinates are written in B. The linear component, ${}^{B}v_{A,B}$ is the velocity of the origin of B, o_B with respect the frame A whose coordinates are written in B – see Figure 2.1.

2.2.2 6D spatial force

Given a rigid body, we introduced the spatial force as a 6D vector containing a pure force a torque applied to the body [2] [Traversaro et al., 2017, Chapter 2.5]. Similar to the spatial velocity, also the spatial force admits a left and right trivialization. Given a body with a frame, denoted with B, rigidly attached to it and a frame A fixed in the space, we define *right trivialized 6D-force*, denoted ${}_{B}\mathrm{f} \in \mathfrak{se}(3)^*$ as

$$
{}_{B}\mathrm{f} := \begin{bmatrix} {}_{B}f \\ {}_{B}\mu \end{bmatrix} \in \mathbb{R}^6. \tag{2.2.13}
$$

[2] A more rigorous approach is presented in Appendix A.4

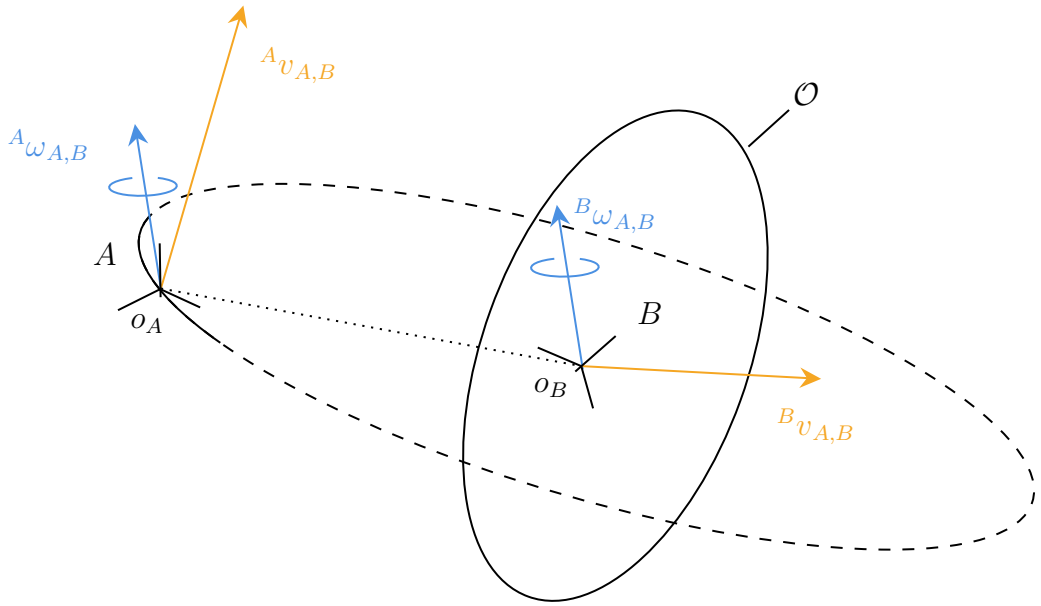


Figure 2.1 Spatial velocity of a rigid-body. ${}^{A}\omega_{A,B}$ and ${}^{B}\omega_{A,B}$ are the angular velocity of the rigid body $\mathcal{O}$ expressed in A and B, respectively. ${}^{A}v_{A,B}$ is the linear velocity, expressed in A, of a point belonging to the *rigid extension* of the rigid body $\mathcal{O}$ coincident with the origin o_A. ${}^{B}v_{A,B}$ is the linear velocity, expressed in B, of the point o_B.

where $\mathfrak{se}(3)^*$ is the dual space of $\mathfrak{se}(3)$[3] and ${}_{B}f$ represents the force acting on the rigid body and ${}_{B}\mu$ is the pure torque about the origin o_B – see Figure 2.2. We define *left trivialized spatial force*, ${}_{A}\mathrm{f} \in \mathfrak{se}(3)^*$ as

$$ {}_{A}\mathrm{f} := \begin{bmatrix} {}_{A}f \\ {}_{A}\mu \end{bmatrix} \in \mathbb{R}^6. \tag{2.2.14} $$

Here ${}_{A}f$ represents the force acting on the rigid body and ${}_{A}\mu$ is the pure torque about the origin o_A – Figure 2.2. To give the reader a better understanding, ${}_{A}\mu$ is the torque about a *rigid extension* of the rigid body that extends up the origin of the inertial frame A – see Figure 2.2.

[3] A rigorous definition of the dual space is provided in Appendix A.4

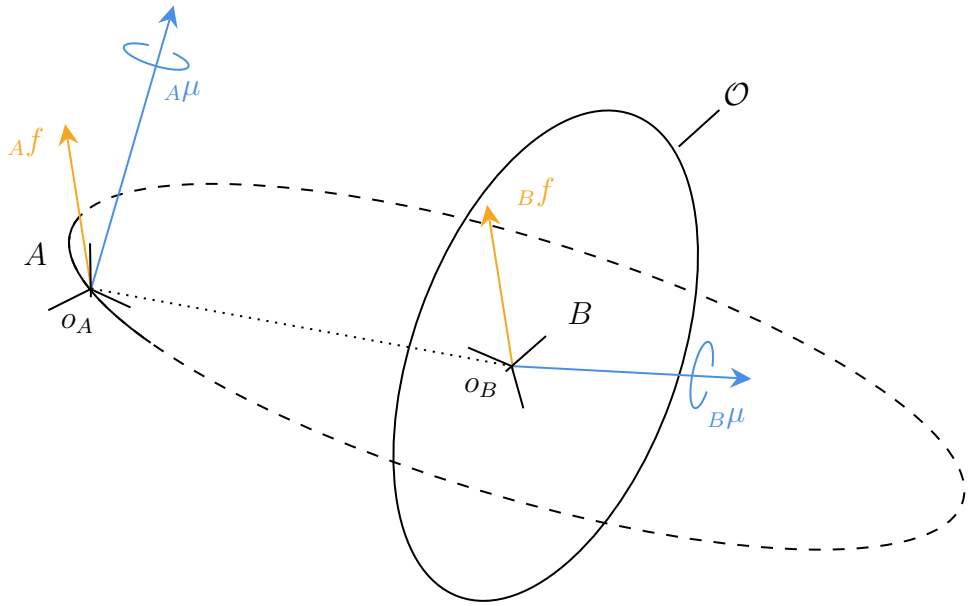


Figure 2.2 Spatial force of a rigid-body. ${}_Af$ and ${}_Bf$ are the force acting on the rigid body $\mathcal{O}$ expressed in A and B, respectively. ${}_A\mu$ is the torque, expressed in A, of a point belonging to the *rigid extension* of the rigid body $\mathcal{O}$ coincident with the origin o_A. ${}_B\mu$ is the torque, expressed in B, of the point o_B.

2.2.3 Exponential and Logarithmic map

Given a small increment $\mathrm{v}^\wedge \in \mathfrak{se}(3)$ such that

$$\mathrm{v}^\wedge = \begin{bmatrix} \theta\times & \rho \\ 0_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3), \quad \mathrm{v} = \begin{bmatrix} \rho \\ \theta \end{bmatrix} \in \mathbb{R}^6, \tag{2.2.15}$$

where $\rho \in \mathbb{R}^3$ and $\theta \in \mathbb{R}^3$ are, respectively, the linear and the angular terms of the increment. The Exp map writes as [Solà et al., 2018]:

$$H = \mathrm{Exp}(\mathrm{v}) := \begin{bmatrix} \mathrm{Exp}(\theta) & V(\theta)\rho \\ 0_{3\times 1} & 1 \end{bmatrix}, \tag{2.2.16}$$

where $V(\theta)$ is given by

$$V(\theta) = I_3 + \frac{1-\cos\|\theta\|}{\|\theta\|^2}(\theta\times) + \frac{\|\theta\| - \sin\|\theta\|}{\|\theta\|^3}(\theta\times)^2. \tag{2.2.17}$$

Given an element of the 3D rigid motion group $H \in \mathrm{SE}(3)$ such that

$$H = \begin{bmatrix} R & p \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3), \tag{2.2.18}$$

the Log map is

$$\mathrm{v} = \mathrm{Log}(H) := \begin{bmatrix} V^{-1}\left(\mathrm{Log}(R)\right) p \\ \mathrm{Log}(R) \end{bmatrix}. \tag{2.2.19}$$

2.2.4 The adjoint representation

Considering ${}^{A}H_{B} \in \mathrm{SE}(3)$ and a spatial velocity element ${}^{B}\mathrm{v}^{\wedge}_{A,B} \in \mathfrak{se}(3)$, the adjoint representation maps a left trivialized spatial velocity into a right trivialized one as:

$${}^{A}\mathrm{v}^{\wedge}_{A,B} = \mathrm{Ad}_{{}^{A}H_{B}}\left({}^{B}\mathrm{v}^{\wedge}_{A,B}\right) = {}^{A}H_{B}\,{}^{B}\mathrm{v}^{\wedge}_{A,B}\,{}^{A}H_{B}^{-1}. \tag{2.2.20}$$

Since the adjoint representation is a linear mapping, there exists the inverse of $\mathrm{Ad}_{{}^{A}H_{B}}$, hereafter written as $\mathrm{Ad}^{-1}_{{}^{A}H_{B}} = \mathrm{Ad}_{{}^{A}H_{B}^{-1}}$. $\mathrm{Ad}^{-1}_{{}^{A}H_{B}}$ maps a right trivialized spatial velocity into a left trivialized one, i.e.,

$${}^{B}\mathrm{v}^{\wedge}_{A,B} = \mathrm{Ad}^{-1}_{{}^{A}H_{B}}\left({}^{A}\mathrm{v}^{\wedge}_{A,B}\right) = \mathrm{Ad}_{{}^{A}H_{B}^{-1}}\left({}^{A}\mathrm{v}^{\wedge}_{A,B}\right) = {}^{A}H_{B}^{-1}\,{}^{A}\mathrm{v}^{\wedge}_{A,B}\,{}^{A}H_{B}. \tag{2.2.21}$$

Given the adjoint representation $\mathrm{Ad}_{{}^{A}H_{B}}$, the associated adjoint matrix $\mathbf{Ad}_{{}^{A}H_{B}}$ is

$$\mathbf{Ad}_{{}^{A}H_{B}} = \begin{bmatrix} {}^{A}R_{B} & \left({}^{A}o_{B}\times\right){}^{A}R_{B} \\ 0_{3\times 3} & {}^{A}R_{B} \end{bmatrix}. \tag{2.2.22}$$

The adjoint matrix $\mathbf{Ad}_{{}^{A}H_{B}}$ is identified by developing the adjoint representation in (2.2.20) as

$$\begin{aligned}
\mathbf{Ad}_{H}\,\mathrm{v} &= \left[\mathrm{Ad}_{H}\left(\mathrm{v}^{\wedge}\right)\right]^{\vee} = \left(H\mathrm{v}H^{-1}\right)^{\vee} && \text{(2.2.23a)}\\
&= \begin{bmatrix} R(\omega\times)R^{\top} & -R(\omega\times)R^{\top}o + Rv \\ 0_{1\times 3} & 0 \end{bmatrix}^{\vee} && \text{(2.2.23b)}\\
&= \begin{bmatrix} (R\omega)\times & (o\times)R\omega + Rv \\ 0_{1\times 3} & 0 \end{bmatrix}^{\vee} && \text{(2.2.23c)}\\
&= \begin{bmatrix} (o\times)R\omega + Rv \\ R\omega \end{bmatrix} = \begin{bmatrix} R & (o\times)\,R \\ 0_{3\times 3} & R \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. && \text{(2.2.23d)}
\end{aligned}$$

Where, to improve the readability all the prefix and postfix have been removed. Hereafter the Adjoint matrix $\mathbf{Ad}_{{}_AH_B}$ is denoted with ${}^AX_B$. The linear mapping between the ${}^B\mathrm{v}_{A,B}$ and ${}^A\mathrm{v}_{A,B}$ writes as

$$ {}^A\mathrm{v}_{A,B} = {}^AX_B\,{}^B\mathrm{v}_{A,B}. \tag{2.2.24} $$

A rigorous definition of the Adjoint representation is provided in Equation (A.7.1).

2.2.5 The co-adjoint representation

Similarly to what we discussed for a 6D velocities, we can define a linear map to change the coordinates of a 6D force from a frame B to another frame A. This coordinate transformation is indicated with ${}_AX^B$ and written as

$$ {}_A\mathrm{f} = {}_AX^B\,{}_B\mathrm{f} \tag{2.2.25} $$

Sometimes, ${}_AX^B$ is also denoted with $\mathbf{Ad}^*_{{}_AH_B}$, see Appendix A.7.

The matrix ${}_AX^B$ is induced by the velocity transformation (2.2.22) and it is related to ${}^BX_A$ as

$$ \mathbf{Ad}^*_{{}_AH_B} = {}_AX^B = {}^BX_A^\top = \mathbf{Ad}^\top_{{}_AH_B^{-1}}\,. \tag{2.2.26} $$

In particular

$$ \mathbf{Ad}^*_{{}_AH_B} = \begin{bmatrix} {}^AR_B & 0_{3\times3} \\ \left({}^Ao_B\times\right){}^AR_B & {}^AR_B \end{bmatrix}. \tag{2.2.27} $$

2.2.6 The adjoint representation of $\mathfrak{se}(3)$

Considering ${}^Bu(t)^\wedge \in \mathfrak{se}(3)$ and ${}^Au(t)^\wedge \in \mathfrak{se}(3)$ such that

$$ {}^Au(t) = {}^AX_B(t)\,{}^Bu(t). \tag{2.2.28} $$

The time derivative of we may evaluate the time of Equation (2.2.28) is given by

$$ \frac{\mathrm{d}}{\mathrm{d}\,t}{}^Au = \frac{\mathrm{d}}{\mathrm{d}\,t}\left({}^AX_B\,{}^Bu\right) \tag{2.2.29a} $$

$$ = {}^AX_B\left(\frac{\mathrm{d}}{\mathrm{d}\,t}{}^Bu + {}^B\mathrm{v}_{A,B}\times{}^Bu\right), \tag{2.2.29b} $$

where ${}^{B}\mathrm{v}_{A,B}\times$ is defined as

$$ {}^{B}\mathrm{v}_{A,B}\times := \begin{bmatrix} {}^{B}\omega_{A,B}\times & {}^{B}v_{A,B}\times \\ 0_{3\times 3} & {}^{B}\omega_{A,B}\times \end{bmatrix}. \tag{2.2.30} $$

The ${}^{B}\mathrm{v}_{A,B}\times$ matrix is often denoted as $\mathbf{ad}_{{}^{B}\mathrm{v}^{\wedge}_{A,B}}$, a rigorous explanation of this choice is given by Equation (A.8.5).

2.2.7 The co-adjoint representation of $\mathfrak{se}(3)$

Similarly to what we discussed for a 6D velocities, we can compute the time derivative of the co-adjoint linear map (2.2.25) as:

$$ \frac{\mathrm{d}}{\mathrm{d}\,t}{}^{A}\mathrm{f} = \frac{\mathrm{d}}{\mathrm{d}\,t}\left({}_{A}X^{B}\,{}_{B}\mathrm{f}\right) \tag{2.2.31a} $$

$$ = {}_{A}X^{B}\left(\frac{\mathrm{d}}{\mathrm{d}\,t}{}_{B}\mathrm{f} + {}^{B}\mathrm{v}_{A,B}\times^{*}{}_{B}\mathrm{f}\right), \tag{2.2.31b} $$

where ${}^{B}\mathrm{v}_{A,B}\times^{*}$ is defined as

$$ {}^{B}\mathrm{v}_{A,B}\times^{*} := \begin{bmatrix} {}^{B}\omega_{A,B}\times & 0_{3\times 3} \\ {}^{B}v_{A,B}\times & {}^{B}\omega_{A,B}\times . \end{bmatrix} \tag{2.2.32} $$

The ${}^{B}\mathrm{v}_{A,B}\times^{*}$ matrix is often denoted as $\mathbf{ad}^{*}_{{}^{B}\mathrm{v}^{\wedge}_{A,B}}$, a rigorous explanation of this choice is given by Equation (A.8.13).

2.2.8 Mixed spatial velocity

In some applications, it may be helpful to define the 6D velocity as

$$ \begin{bmatrix} {}^{A}\dot{o}_{B} \\ {}^{A}\omega_{A,B} \end{bmatrix}. \tag{2.2.33} $$

This representation is often called *hybrid* [Murray et al., 1994] or *mixed* [Traversaro et al., 2017, Chapter 2.3] velocity representation. To avoid confusion with hybrid systems theory, we will call it as *mixed* velocity of frame B with respect to frame A. The mixed representation is particularly helpful when designing and implementing controllers that aim to follow a desired Cartesian trajectory. To simplify the description of the algorithms and concepts presented in this thesis, we need a way to express this

quantity coherently with the rest of the concepts introduced until know. Given two frame A and B with a given origin and orientation, hereafter indicated as o_A, o_B and $[A]$ and $[B]$, we introduce the frame $B[A] := (o_B,\ [A])$ as a frame having the origin on o_B and oriented as A. We formally define the mixed velocity ${}^{B[A]}\mathrm{v}_{A,B}$ as

$$ {}^{B[A]}\mathrm{v}_{A,B} := {}^{B[A]}X_B\, {}^{B}\mathrm{v}_{A,B} = \begin{bmatrix} {}^{A}R_B & 0_{3\times 3} \\ 0_{3\times 3} & {}^{A}R_B \end{bmatrix} \begin{bmatrix} {}^{A}R_B^{\top}\, {}^{A}\dot{o}_B \\ {}^{B}\omega_{A,B} \end{bmatrix} = \begin{bmatrix} {}^{A}\dot{o}_B \\ {}^{A}\omega_{A,B} \end{bmatrix}. \tag{2.2.34} $$

It is important to notice that ${}^{B[A]}\mathrm{v}_{A,B}$ is not isomorphic to any element of $\mathfrak{se}(3)$, i.e., ${}^{B[A]}\mathrm{v}^{\wedge}_{A,B} \notin \mathfrak{se}(3)$. On the other hand, ${}^{B[A]}\mathrm{v}_{A,B}$ can be seen as the row concatenation of the left trivialized linear velocity whose coordinate are expressed in A and the right trivialized angular velocity. In other words ${}^{B[A]}\mathrm{v}_{A,B}$ will belong to the Cartesian product of $\mathbb{R}^3$ and $\mathfrak{so}(3)$, i.e., ${}^{B[A]}\mathrm{v}^{\wedge}_{A,B} \in \mathbb{R}^3 \times \mathfrak{so}(3)$.

2.2.9 Mixed spatial force

Smilar to the mixed spatial velocity, in some applications, it may be helpful to define the 6D force as

$$ {}^{B[A]}\mathrm{f} = \begin{bmatrix} {}_{A}f \\ {}_{B[A]}\mu \end{bmatrix}. \tag{2.2.35} $$

This representation is often called *mixed* [Traversaro et al., 2017, Chapter 2.3] force representation. Given two frames A and B with a given origin and orientation, hereafter indicated as o_A, o_B and $[A]$ and $[B]$, we introduce the frame $B[A] := (o_B,\ [A])$ as a frame having the origin on o_B and oriented as A. We formally define the mixed spatial force ${}_{B[A]}\mathrm{f}$ as

$$ {}_{B[A]}\mathrm{f} := \mathbf{Ad}^{*}_{B[A]H_B}\, {}_{B}\mathrm{f} = {}_{B[A]}X^{B}\, {}_{B}\mathrm{f} = \begin{bmatrix} {}^{A}R_B & 0_{3\times 3} \\ 0_{3\times 3} & {}^{A}R_B \end{bmatrix} \begin{bmatrix} {}_{B}f \\ {}_{B}\mu \end{bmatrix} = \begin{bmatrix} {}_{A}f \\ {}_{B[A]}\mu \end{bmatrix}. \tag{2.2.36} $$

To give the reader a better comprehension, we may think that a mixed spatial force is a 6D vector whose linear component is the force acting on the rigid body whose entries are written with respect to the frame A, while the angular component is the torque about o_B whose coordinate are written in A.

The mixed spatial force will play a crucial role in the design of the control systems presented in the following parts.

2.3 Rigid body dynamics

Let a rigid body $\mathcal{O}$ lying in a uniform gravitational filed and given a frame B rigidly attached to $\mathcal{O}$. Given an inertial frame $\mathcal{I}$, the configuration of $\mathcal{O}$ is completely determined by the homogeneous transformation ${}^{\mathcal{I}}H_B \in \mathrm{SE}(3)$. We denote ${}^{B}\mathrm{v}_{\mathcal{I},B} \in \mathfrak{se}(3)$ the left trivialized spatial velocity of $\mathcal{O}$. Let m the mass of the rigid body, $\mathbb{I} \in \mathbb{R}^{3\times3}$ the 3D inertial matrix of the body expressed in B and ${}^{B}p_c$ the position of the center of mass (CoM) expressed in B, we introduce the 6D inertia matrix of the body as

$$ {}_{B}\mathbb{M}_B = \begin{bmatrix} m & -m{}^{B}p_c\times \\ m{}^{B}p_c\times & \mathbb{I} \end{bmatrix}. \tag{2.3.1} $$

It is important to recall that ${}_{B}\mathbb{M}_B$ is a constant matrix.

We now introduce the *left-trivialized Lagrangian of the rigid body*, denoted with $\mathcal{L}({}^{\mathcal{I}}H_B, {}^{B}\mathrm{v}_{\mathcal{I},B})$, as

$$ \mathcal{L}({}^{\mathcal{I}}H_B, {}^{B}\mathrm{v}_{\mathcal{I},B}) = \mathcal{K}({}^{B}\mathrm{v}_{\mathcal{I},B}) - \mathcal{U}({}^{\mathcal{I}}H_B) \tag{2.3.2a} $$

$$ \frac{1}{2}{}^{B}\mathrm{v}_{\mathcal{I},B}^{\top}\ {}_{B}\mathbb{M}_B\ {}^{B}\mathrm{v}_{\mathcal{I},B} + \begin{bmatrix} g^{\top} & 0_{1\times3} \end{bmatrix} {}^{B}H_{\mathcal{I}} \begin{bmatrix} m{}^{B}p_c \\ m \end{bmatrix}, \tag{2.3.2b} $$

where $\mathcal{K}({}^{B}\mathrm{v}_{\mathcal{I},B})$ is the *kinetic energy*, while $\mathcal{U}({}^{\mathcal{I}}H_B)$ is the *potential energy*. $g \in \mathbb{R}^3$ is the gravitational acceleration vector. By applying *Hamilton's Variational Principle*, it is possible to prove that given a time interval $[t_0, t_f]$, the trajectory ${}^{B}H_{\mathcal{I}}$ of the rigid body $\mathcal{O}$ is the one that minimizes the action [Traversaro, 2017, Theorem 2.1]:

$$ \mathfrak{G} = \int_{t_0}^{t_f} \mathcal{L}({}^{\mathcal{I}}H_B, {}^{B}\mathrm{v}_{\mathcal{I},B})\,\mathrm{d}\,t, \tag{2.3.3} $$

where $\mathcal{L}({}^{\mathcal{I}}H_B, {}^{B}\mathrm{v}_{\mathcal{I},B})$ is given by (2.3.2).

In particular the resulting equation of motions are the one that satisfies the following differential equation

$$ {}_{B}\mathbb{M}_B{}^{B}\dot{\mathrm{v}}_{\mathcal{I},B} + {}^{B}\mathrm{v}_{\mathcal{I},B} \times^{*} {}_{B}\mathbb{M}_B{}^{B}\mathrm{v}_{\mathcal{I},B} = {}_{B}\mathbb{M}_B \begin{bmatrix} {}^{\mathcal{I}}R_B^{\top}g \\ 0_{3\times1} \end{bmatrix}. \tag{2.3.4} $$

The reader can find a brief introduction of *Hamilton's Variational Principle* in Appendix D.1.

In the case of a non-conservative spatial force acting on the body, Equation (2.3.4) becomes

$$ {}_B\mathbb{M}_B {}^B\dot{\mathrm{v}}_{\mathcal{I},B} + {}^B\mathrm{v}_{\mathcal{I},B} \times^* {}_B\mathbb{M}_B {}^B\mathrm{v}_{\mathcal{I},B} = {}_B\mathbb{M}_B \begin{bmatrix} {}^{\mathcal{I}}R_B^\top g \\ 0_{3\times 1} \end{bmatrix} + {}_B\mathrm{f}. \tag{2.3.5} $$

Equation (2.3.5) can be expressed also with respect to the inertial frame $\mathcal{I}$

$$ {}_{\mathcal{I}}\mathbb{M}_B {}^B\dot{\mathrm{v}}_{\mathcal{I},B} + {}^{\mathcal{I}}\mathrm{v}_{\mathcal{I},B} \times^* {}_{\mathcal{I}}\mathbb{M}_B {}^{\mathcal{I}}\mathrm{v}_{\mathcal{I},B} = {}_{\mathcal{I}}\mathbb{M}_B \begin{bmatrix} g \\ 0_{3\times 1} \end{bmatrix} + {}_{\mathcal{I}}\mathrm{f}. \tag{2.3.6} $$

Even if (2.3.6) seems similar to (2.3.5), it is important to underline that ${}_B\mathbb{M}_B$ is a constant matrix depending only on the geometry of the rigid body, while ${}_{\mathcal{I}}\mathbb{M}_B$ is a time-varying quantity that depends on the position of the rigid-body - i.e., ${}_{\mathcal{I}}\mathbb{M}_B = {}_{\mathcal{I}}X^B {}_B\mathbb{M}_{B\mathcal{I}}X^B$.

We finally introduce the 6D spatial momentum of the body $\mathcal{O}$ with respect to a frame $\mathcal{I}$ expressed in B and denoted with ${}_Bh_{\mathcal{I},B}$ as a 6D vector isomorphic to $\mathfrak{se}(3)^*$, i.e., ${}_Bh_{\mathcal{I},B} \in \mathbb{R}^6 \cong \mathfrak{se}(3)^*$ and it writes as

$$ {}_Bh_{\mathcal{I},B} := \mathbf{Ad}^*_{B_{H_{\mathcal{I}}}} {}_B\mathbb{M}_B {}^B\mathrm{v}_{\mathcal{I},B}. \tag{2.3.7} $$

By means of the 6D spatial momentum, Equation (2.3.6) can be rewritten as [Featherstone, 2014].

$$ {}_{\mathcal{I}}\dot{h}_{\mathcal{I},B} = {}_{\mathcal{I}}X^B {}_B\mathbb{M}_B \begin{bmatrix} {}^{\mathcal{I}}R_B^\top g \\ 0_{3\times 1} \end{bmatrix} + {}_{\mathcal{I}}\mathrm{f}. \tag{2.3.8} $$

Equation (2.3.8) is also known as *Newton-Euler equations* for a rigid body.

2.4 The rotation and euclidean groups: a Lie groups prospective

In Sections 2.1 and 2.2 we present the rotation and Euclidian groups. One may have noticed some similarities between the two sets. Indeed, we have

- Both sets are groups under the matrix product operator.
- Both groups admit a left and a right velocity
- Both groups admit an Exponential and a Logarithm map.

This similarity is not accidental. Indeed SO(3) and SE(3) are two examples of *Matrix Lie Group*. In this context we say that the angular velocity ω and the spatial 6D velocity v belong to the *Lie Algebra* of SO(3) and SE(3), denoted respectively by $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$. In Appendix A, we generalize the concepts presented in this chapter to a generic Matrix Lie Group.

Chapter 3

Modeling of Floating Base Multi-Body Systems

In the previous chapter, we introduced the rotation and the rototranslation matrix Lie Group and the associated rigid body dynamics. We now exploit these concepts to describe the kinematics and the dynamics of a floating base system. What is presented in this chapter is crucial to fully understand the design of the controllers and the estimators presented in Part II and Part III. The chapter is organized as follows. In Section 3.1, we describe a multi-body system in terms of interaction between links and joints. Such a description is often denoted *system model* [Featherstone, 2014]. Section 3.2 discusses the kinematics of the multi-body system. Here, we present the *minimum set of coordinates* q as an element of a Lie group $\mathcal{Q}$. We also introduce the *velocity of the multi-body system* ν as an element of the Lie algebra $\mathfrak{q}$. Section 3.2 also contains the definition of the *forward kinematics* function and the relation between the link velocity and the velocity of the multi-body system ν. Section 3.3 presents the dynamics equation of the multi-body system, since the multi-body configuration space is not a vector space, the dynamics is obtained by extending the Euler-Poincaré presented in Section 2.3 to case of multi-body systems [Marsden and Ratiu, 2010]. Finally, Section 3.4 discusses the centroidal dynamics [Orin et al., 2013] as the aggregate linear and angular momentum of each link of the robot referred to the center of mass of the robot.

3.1 Floating base multi-body system modeling

A *rigid-body system* is an assembly of component parts, namely: n_b *rigid bodies* and n_j *joints* which are responsible for the kinematic constraints in the system. In this section, we present a formalism used to describe rigid-body systems in terms of graphs. In particular, we seek to describe the interaction between bodies and joints as an element of nodes and the arch of a graph. We call such a description *system model* with respect to an inertial frame $\mathcal{I}$.

A system model is often expressed in the form of a *connectivity graph* having the following properties:

- Each node represents one rigid body in the dynamical system. We assume that at least one frame is rigidly attached to each body;
- Each arc represents a joint;
- There exists one node, i.e. a rigid body, which is called *base* of the multi-body system. The base is indicated by the frame B;
- The graph is undirected;
- The graph is connected, i.e., there exists at least one path between any two nodes.

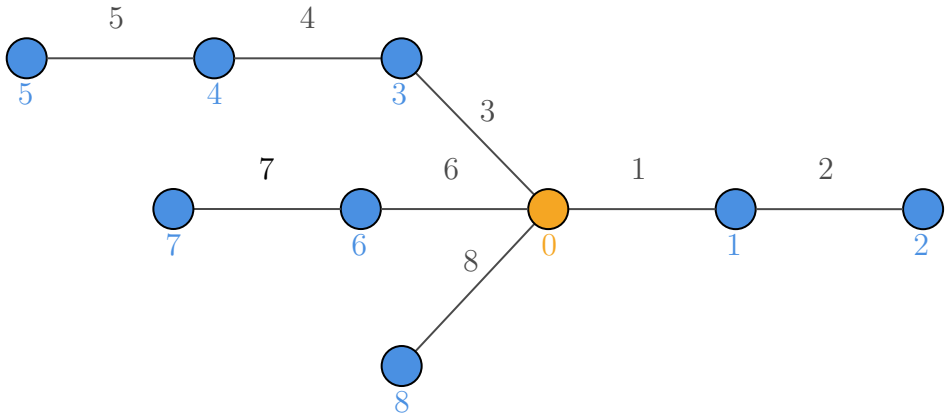
If the base of the multi-body system does not have an *a priori* fixed pose with respect to an inertial frame, $\mathcal{I}$, the associated body is called *floating base*. In this case, the pose (position and orientation) of the floating base B with respect to the inertial frame $\mathcal{I}$ is given by the element of the SE(3) and it writes as:

$$ {}^{\mathcal{I}}H_B = \begin{bmatrix} {}^{\mathcal{I}}R_B & {}^{\mathcal{I}}p_B \\ 0_{1\times 3} & 1 \end{bmatrix}. \tag{3.1.1} $$

A graph is a topological tree if there exists exactly one path between any two nodes in the graph. If the connectivity graph of a rigid-body system is a topological tree, then we call the system itself a *kinematic tree*. If the rigid-body system is represented by a kinematic tree, there exists a well-defined relation between the number of joints n_j, and the number of rigid bodies n_b, that is,

$$ n_b = n_j + 1. \tag{3.1.2} $$

(a) Regular numbering applied to an unbranched kinematic tree



(b) Regular numbering applied to a kinematic tree

Figure 3.1 Schematic representation of a multi-body structure. The links are represented by the graph nodes, while joints are the graph's arcs. The links and joints are named by applying the Regular numbering principle.

From now on, we consider only floating base multi-body systems characterized by a kinematic tree.

Once we describe the rigid-body system as a graph, we aim to number the nodes and arcs following the *regular numbering* scheme [Featherstone, 2014]. We now present the scheme for the topological tree. The generic scheme that applies to the generic graph is described in [Featherstone, 2014, Section 4.1.2]. Given a kinematic tree G, we denote the elements as follows:

- The base is represented by the number 0;
- The remaining $n_b - 1$ nodes are numbered in any order such that each node has a higher number than its parent in G ;
- The arcs are identified with a number for 1 to n_j such that the arc i connects between node i and its parent.

If these principles are applied to an unbranched kinematic tree (one with no more than one child), the bodies and joints are numbered sequentially from the base, as shown in Figure 3.1a. On the other hand, if the tree is composed of multiple branches, the numbering is not unique. Figure 3.1b represents one of the possible examples.

A connectivity graph is fully determined by a pair of arrays named p and s, such that the element i in p is the vector of the predecessor of the joint i while the element i in s contains the successor of the joint i. Taking as an example the unbranched kinematic tree in Figure 3.1a the arrays s and p are given by

$$p = \begin{bmatrix} 0 & 1 & \dots & n_j - 2 & n_j - 1 \end{bmatrix}, \quad s = \begin{bmatrix} 1 & 2 & \dots & n_j - 1 & n_j \end{bmatrix}. \tag{3.1.3}$$

On the other hand, the tree in Figure 3.1b is described by:

$$p = \begin{bmatrix} 0 & 1 & 0 & 3 & 4 & 0 & 0 \end{bmatrix}, \quad s = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}. \tag{3.1.4}$$

The *parent array* λ identifies the parent of each body. $\lambda(i)$ is the parent of the body i. We finally introduce for any body i three sets, namely: $\kappa(i)$, $\mu(i)$ and $\nu(i)$. $\kappa(i)$ is the set of all joints on the path between body i and base. If $i = 0$, i.e. the body is the base, the set is empty. $\mu(i)$ is the set of the children of body i, and $\nu(i)$ is the set of bodies in the subtree starting at body i. $\kappa(i)$, $\mu(i)$ and $\nu(i)$ are called the *support*, *child* and *subtree* sets, respectively. Given the tree in Figure 3.1b the *support*, *child* and *subtree* sets are given by

$$\kappa(0) = \emptyset \qquad \mu(0) = \{1, 3, 6, 8\} \qquad \nu(0) = \{0, 1, 2, 3, 4, 5, 6, 7, 8\} \tag{3.1.5a}$$
$$\kappa(1) = \{1\} \qquad \mu(1) = \{2\} \qquad \nu(1) = \{1, 2\} \tag{3.1.5b}$$
$$\kappa(2) = \{1, 2\} \qquad \mu(2) = \emptyset \qquad \nu(2) = \{2\} \tag{3.1.5c}$$
$$\kappa(3) = \{3\} \qquad \mu(3) = \{4\} \qquad \nu(3) = \{3, 4, 5\} \tag{3.1.5d}$$
$$\kappa(4) = \{3, 4\} \qquad \mu(4) = \{5\} \qquad \nu(4) = \{4, 5\} \tag{3.1.5e}$$
$$\kappa(5) = \{3, 4, 5\} \qquad \mu(5) = \emptyset \qquad \nu(5) = \{5\} \tag{3.1.5f}$$
$$\kappa(6) = \{6\} \qquad \mu(6) = \{7\} \qquad \nu(6) = \{6, 7\} \tag{3.1.5g}$$
$$\kappa(7) = \{6, 7\} \qquad \mu(7) = \emptyset \qquad \nu(7) = \{7\} \tag{3.1.5h}$$
$$\kappa(8) = \{8\} \qquad \mu(8) = \emptyset \qquad \nu(8) = \{8\}. \tag{3.1.5i}$$

When a joint connects two bodies, its relative motion is limited. The mobility allowed by the joint and the placement of the joint relative to each body are necessary for a thorough description of the restriction. A joint model explains the former. The geometry of the system determines the latter.

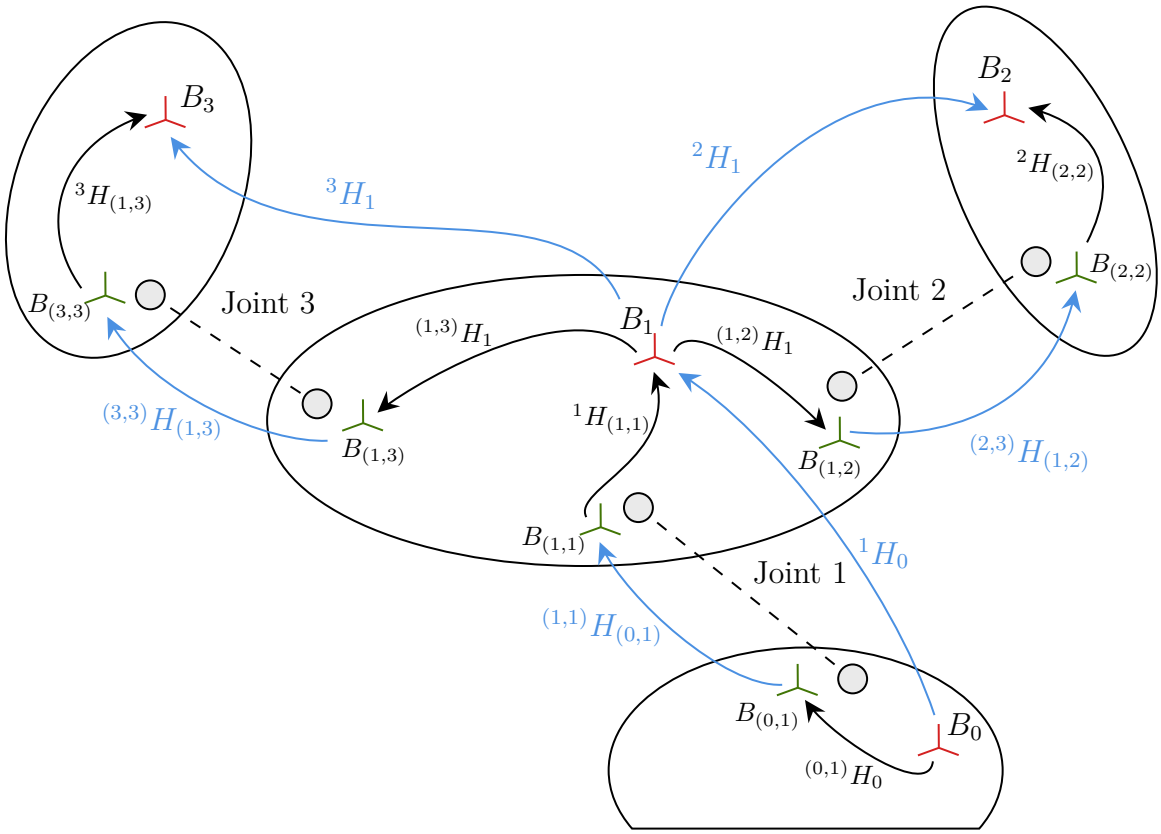


Figure 3.2 Geometric model of a rigid-body system. The red frames are the *link frames*. The green frames are the *joint frames*. Each arrow represents a homogeneous transformation. A black arrow indicates a constant transformation, while a blue one a joint state-dependent transformation.

The joint model In this context, a joint may be thought of as a motion constraint between two Cartesian frames, one of which is embedded in the body i and the other in its parent, $\lambda(i)$. The kind of mobility that a joint allows is determined by its type of joint. A *revolute* joint, for example, provides for pure rotation along a single axis, but a spherical joint allows for unrestricted rotation around a single point. Consequently, we should introduce the admissible motions for each type of joint. However, since in this thesis we assume that the floating base system is composed only of one-degree-of-freedom revolute joints, we analyze the joint characteristics only for this kind of joint. A revolute joint j is completely characterized by an axis ${}_ja \in \mathcal{S}^2$ and an angle s_j, where $\mathcal{S}^2$ is the set of 3D vectors with norm equal to 1, i.e., $\mathcal{S}^2 = \left\{x \in \mathbb{R}^3 | x^\top x = 1\right\}$ and $s_j \in \mathbb{R}$.

Geometric model The geometric model describes the location of each joint in each rigid body. In detail, each body i is characterized by a set of frames. One, known as *link frame*, identifies the position of the link. The others are placed in the joints, j. Figure 3.2 represents the geometry model of a rigid-body system. Here, each rigid body has a frame embedded in it, which defines a local coordinate system for that body. In Figure 3.2 these frames are labeled as B_i where i represents the body index. Figure 3.2 also presents several frames with names of the form $B_{(i,j)}$ where i refers to a body and j to the associated joint. The position and the orientation of the frame $B_{(i,j)}$ with respect to B_i are constant and are given by the transformation ${}^{(i,j)}H_i$. Finally, each joint describes a joint transform labeled ${}^{(i,k)}H_{(j,k)}$ where i refers to the parent body, j to the child body, and k to the joint. ${}^{(i,k)}H_{(j,k)}$ depends only on the joint position. From now on we assume that, given two rigid bodies i and j connected by a joint k, the transformation ${}^{(i,k)}H_i$ and ${}^{(j,k)}H_j$ are chosen so that ${}^{(i,k)}H_{(j,k)} = I_4$ only if the joint position state is equal to 0, i.e., ${}^{(i,k)}H_{(j,k)}(s_j) = {}^{(i,k)}H_{(j,k)}(0) = I_4$.

3.2 Multi-body kinematics

Once the floating base multi-body system has been described in the form of a connectivity graph, the position and orientation of the link i can be reconstruct as follows:

$$ {}^{\mathcal{I}}H_i = {}^{\mathcal{I}}H_0 \prod_{j \in \kappa(i)} {}^{\lambda(j)}H_j, \tag{3.2.1} $$

where in our case the link 0 is the floating base of the robot, that is, $0 \equiv B$ and ${}^{\mathcal{I}}H_0 \equiv {}^{\mathcal{I}}H_B$ – see Equation (3.1.1). The transformation matrix ${}^{\lambda(j)}H_j$ depends upon the position of the joint j, henceforth indicated with s_j. To give the reader a better comprehension of Equation (3.2.1), we can imagine that ${}^{\mathcal{I}}H_i$ is given by the concatenation of SE(3) elements that express the link's frame to the parent. Equation (3.2.1) can be obtained by exploring the connectivity graph from the link i to the base link. For example, considering the tree in Figure 3.1b and the link 7, ${}^{\mathcal{I}}H_7$ is given by ${}^{\mathcal{I}}H_7 = {}^{\mathcal{I}}H_0 {}^0H_6(s_6) {}^6H_7(s_7)$.

Following the geometric model in Figure 3.2, ${}^{\lambda(j)}H_j$ transformation can be seen as a composition of three transformations. ${}^{\lambda(j)}H_j$ writes

$$ {}^{\lambda(j)}H_j = {}^{\lambda(j)}H_{(\lambda(j),j)} {}^{(\lambda(j),j)}H_{(j,j)} {}^{(j,j)}H_j, \tag{3.2.2} $$

where ${}^{\lambda(j)}H_{(\lambda(j),j)}$ and ${}^{(j,j)}H_j$ give the position and orientation of the link frames with respect to the frame associated with the joint. ${}^{(\lambda(j),j)}H_{(j,j)}$ depends on the joint's position value. For a revolute joint characterized by the axis ${}_ja \in \mathcal{S}^2$ and an angle $s_j \in \mathbb{R}$, ${}^{(\lambda(j),j)}H_{(j,j)}$ is given by

$$ {}^{(\lambda(j),j)}H_{(j,j)} = \mathrm{Exp}\left(\begin{bmatrix} 0_{3\times1} \\ s_j\ {}_ja \end{bmatrix}\right) = \exp\left(\begin{bmatrix} (s_j\ {}_ja)\times & 0_{3\times1} \\ 0_{1\times3} & 0 \end{bmatrix}\right) \tag{3.2.3} $$

Recalling that the exponential operator for a skew-symmetric matrix given by the well-known Rodriguerz formula – see Section 2.1.2 [Murray et al., 1994], ${}^{(\lambda(j),j)}H_{(j,j)}$ writes as

$$ \begin{aligned} {}^{(\lambda(j),j)}H_{(j,j)}(s_j) &= \begin{bmatrix} {}^{(\lambda(j),j)}R_{(j,j)}(s_j) & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \\ {}^{(\lambda(j),j)}R_{(j,j)}(s_j) &= I_3 + \cos(s_j)({}_ja\times) + \sin(s_j)({}_ja\times)^2. \end{aligned} \tag{3.2.4} $$

We now define the *forward kinematics* function of a floating base system described as a *kinematic tree* composed of $n_j + 1$ rigid bodes connected with n_j joints as:

$$ \Pi : \mathrm{SE}(3) \times \mathbb{R}^{n_j} \longrightarrow \mathrm{SE}(3); \quad {}^{\mathcal{I}}H_i = \Pi\left({}^{\mathcal{I}}H_B, s\right) = {}^{\mathcal{I}}H_0 \prod_{j\in\kappa(i)} {}^{\lambda(j)}H_j. \tag{3.2.5} $$

We define the center of mass (CoM) of the multi-body system, denoted by x_{CoM}, as the weighted average of all the links CoM positions:

$$ x_{\mathrm{CoM}} := {}^{\mathcal{I}}H_B \sum_i \frac{m_i}{m} {}^{B}H_i\ {}^{i}p_{\mathrm{CoM}}, \tag{3.2.6} $$

where ${}^{i}p_{\mathrm{CoM}} \in \mathbb{R}^3$ is the (constant) CoM position of link i expressed in i coordinates. $m, m_i \in \mathbb{R}$ are respectively the robot total mass and the i-th link mass.

Since every link pose can be computed as a function of the base pose ${}^{\mathrm{I}}H_B$ and the joint position values s, we define q as the minimum set of coordinates as

$$ q_{\mathrm{SE}(3)} := ({}^{\mathcal{I}}H_B, s) \in \mathrm{SE}(3) \times \mathbb{R}^n = \mathcal{Q}_{\mathrm{SE}(3)}. \tag{3.2.7} $$

$\mathcal{Q}_{\mathrm{SE}(3)}$ is a composition of Lie groups, as a consequence, it is the Lie group itself. Consequently, there exists a Lie algebra $\mathfrak{q}_{\mathrm{SE}(3)} = \mathfrak{se}(3) \times \mathbb{R}^n$ associated with $\mathcal{Q}_{\mathrm{SE}(3)}$. The element of the Lie algebra is the spatial base velocity and the joint velocity. Since

in the rest of the thesis we often use the mixed spatial velocity representation, we often indicate q as

$$q_{\mathbb{R}^3\times\mathrm{SO}(3)} := ({}^{A}o_B, {}^{\mathcal{I}}R_B, s) \in \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^n = \mathcal{Q}_{\mathbb{R}^3\times\mathrm{SO}(3)}. \tag{3.2.8}$$

Similar to before $\mathcal{Q}_{\mathbb{R}^3\times\mathrm{SO}(3)}$ is a composition of Lie groups, so it admits a Lie algebra $\mathfrak{q}_{\mathbb{R}^3\times\mathrm{SO}(3)} \in \mathbb{R}^3 \times \mathfrak{so}(3) \times \mathbb{R}^n$. The elements of the Lie algebra are the spatial base velocity in mixed representation and the joints velocity.

Given a link L, we now aim to write a relation between its spatial velocity and the elements of $\mathfrak{q}_{\mathrm{SE}(3)}$. The following results are obtained by describing the link velocity by means of the left trivialized spatial velocity; however, similar results hold also if the velocity is expressed in right trivialization or mixed representation. Indeed, a spatial velocity in body frame, i.e., left trivialized, can be converted in inertial frame and in mixed representation thanks to Equations (2.2.24) and (2.2.34), respectively. Let us now define the velocity of a link L as ${}^{L}\mathrm{v}_{\mathcal{I},L}$. ${}^{L}\mathrm{v}_{\mathcal{I},L}$ can be decomposed into two terms as

$${}^{L}\mathrm{v}_{\mathcal{I},L} = {}^{L}X_{B}\,{}^{B}\mathrm{v}_{\mathcal{I},B} + {}^{L}\mathrm{v}_{B,L}, \tag{3.2.9}$$

where ${}^{B}\mathrm{v}_{\mathcal{I},B}$ is the left trivialized base velocity, while ${}^{L}\mathrm{v}_{B,L}$ is a function of the position of the joints $s \in \mathbb{R}^{n_j}$ and the velocity $\dot{s} \in \mathbb{R}^{n_j}$. In fact ${}^{L}\mathrm{v}_{B,L}$ is given by

$$\begin{aligned}
{}^{L}\mathrm{v}_{B,L}^{\wedge} &= {}^{B}H_{L}^{-1}\,{}^{B}\dot{H}_{L} && \text{(3.2.10a)}\\
&= \sum_{j\in\kappa(L)} \left({}^{L}H_{\lambda(j)} \frac{\partial}{\partial s_j} \left({}^{\lambda(j)}H_j \right) {}^{j}H_{L} \right) \dot{s}_j && \text{(3.2.10b)}\\
&= \sum_{j\in\kappa(L)} \left({}^{L}H_{j}\,{}^{\lambda(j)}H_{j}^{-1} \frac{\partial}{\partial s_j} \left({}^{\lambda(j)}H_j \right) {}^{j}H_{L} \right) \dot{s}_j && \text{(3.2.10c)}\\
&= \sum_{j\in\kappa(L)} \left[{}^{L}X_j \left({}^{\lambda(j)}H_{j}^{-1} \frac{\partial}{\partial s_j} \left({}^{\lambda(j)}H_j \right) \right)^{\vee} \right]^{\wedge} \dot{s}_j, && \text{(3.2.10d)}
\end{aligned}$$

where from (3.2.10c) to (3.2.10d) we used the fact that the adjoint action is a linear transformation so it can be associated to the adjoint matrix, i.e.

$$
\begin{aligned}
{}^{L}H_j\,{}^{\lambda(j)}H_j^{-1}\frac{\partial}{\partial s_j}\left({}^{\lambda(j)}H_j\right){}^{j}H_L &= \mathrm{Ad}_{{}^{L}H_{\lambda(j)}}\left({}^{\lambda(j)}H_j^{-1}\frac{\partial}{\partial s_j}\left({}^{\lambda(j)}H_j\right)\right) && \text{(3.2.11a)}\\
&= \left[\mathbf{Ad}_{{}^{L}H_{\lambda(j)}}\left({}^{\lambda(j)}H_j^{-1}\frac{\partial}{\partial s_j}\left({}^{\lambda(j)}H_j\right)\right)^{\vee}\right]^{\wedge} && \text{(3.2.11b)}\\
&= \left[{}^{L}X_j\left({}^{\lambda(j)}H_j^{-1}\frac{\partial}{\partial s_j}\left({}^{\lambda(j)}H_j\right)\right)^{\vee}\right]^{\wedge}. && \text{(3.2.11c)}
\end{aligned}
$$

Let us now introduce the *left-trivialized joint motion subspace* ${}^{j}\mathbf{s}$ as

$${}^{j}\mathbf{s} = \left({}^{\lambda(j)}H_j^{-1}\frac{\partial}{\partial s_j}\left({}^{\lambda(j)}H_j\right)\right)^{\vee}, \tag{3.2.12}$$

equation (3.2.10) becomes

$${}^{L}\mathrm{v}_{B,L} = \sum_{j\in\kappa L}{}^{L}X_j\,{}^{j}\mathbf{s}\,\dot{s}_j. \tag{3.2.13}$$

We observe that, in the case of a revolute joint, ${}^{j}\mathbf{s}$ is constant and it depends on the revolute axis [Traversaro et al., 2017].

Since ${}^{L}\mathrm{v}_{B,L}$ is an affine function of joints velocity $\dot{s}\in\mathbb{R}^{n_j}$, Equation (3.2.13) can be written as

$${}^{L}\mathrm{v}_{B,L} = {}^{L}J^{\dot{s}}_{B,L}(s)\dot{s}, \tag{3.2.14}$$

In summary, the left trivialized link velocity is as follows:

$${}^{L}\mathrm{v}_{\mathcal{I},L} = \begin{bmatrix}{}^{L}X_B & {}^{L}J^{\dot{s}}_{B,L}\end{bmatrix}\begin{bmatrix}{}^{B}\mathrm{v}_{\mathcal{I},B}\\ \dot{s}\end{bmatrix} = {}^{L}J^{B}_{\mathcal{I},L}\,{}^{B}\nu. \tag{3.2.15}$$

${}^{L}J^{B}_{\mathcal{I},L}\in\mathbb{R}^{6\times(6+n_j)}$ is the *left-trivialized Jacobian* of the link L, ${}^{B}\nu\in\mathbb{R}^{6+n_j}$ is the left trivialized multi-body system velocity and ${}^{B}\nu^{\wedge}\in\mathfrak{q}_{\mathrm{SE}(3)}$:

$${}^{B}\nu = \begin{bmatrix}{}^{B}\mathrm{v}_{\mathcal{I},B}\\ \dot{s}\end{bmatrix}. \tag{3.2.16}$$

Given a left trivialized link velocity ${}^{L}\mathrm{v}_{\mathcal{I},L}$, we obtain the associated mixed representation as

$$
\begin{aligned}
{}^{L[\mathcal{I}]}\mathrm{v}_{\mathcal{I},L} &= {}^{L[\mathcal{I}]}X_L \begin{bmatrix} {}^{L}X_B & {}^{L}J^{\dot{s}}_{B,L} \end{bmatrix} \begin{bmatrix} {}^{B}X_{B[\mathcal{I}]} & 0_{6\times n_j} \\ 0_{n_j\times 6} & I_{n_j} \end{bmatrix} \begin{bmatrix} {}^{B[\mathcal{I}]}\mathrm{v}_{\mathcal{I},B} \\ \dot{s} \end{bmatrix} && \text{(3.2.17a)} \\
&= {}^{L[\mathcal{I}]}J^{B[\mathcal{I}]}_{\mathcal{I},L}\, {}^{B[\mathcal{I}]}\nu. && \text{(3.2.17b)}
\end{aligned}
$$

where ${}^{L[\mathcal{I}]}J^{B[\mathcal{I}]}_{\mathcal{I},L} \in \mathbb{R}^{6\times(6+n_j)}$ is the *mixed velocity Jacobian* of link L and ${}^{B[\mathcal{I}]}\nu \in \mathbb{R}^{6+n_j}$ is the mixed multi-body system velocity

$$
{}^{B[\mathcal{I}]}\nu = \begin{bmatrix} {}^{B[\mathcal{I}]}\mathrm{v}_{\mathcal{I},B} \\ \dot{s} \end{bmatrix}. \tag{3.2.18}
$$

To simplify the notation, we will simply assume that we use the mixed representation, i.e. ${}^{L[\mathcal{I}]}J^{B[\mathcal{I}]}_{\mathcal{I},L}$ will be simply indicated J_L. Similarly, ${}^{B[\mathcal{I}]}\nu$ will become ν.

3.3 Multi-body dynamics

As mentioned in Section 3.2, the robot configuration space is characterized by the *position* and the *orientation* of the base frame B with respect to the inertial frame $\mathcal{I}$, and the joints' position coordinates s. Hereafter, we indicate the robot configuration with the following triplet $q = ({}^{\mathcal{I}}p_B, {}^{\mathcal{I}}R_B, s)$. q is an element of a Lie group, $\mathcal{Q} = \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^n$. The associated Lie algebra is denoted as $\mathfrak{q} = \mathbb{R}^3 \times \mathfrak{so}(3) \times \mathbb{R}^n$, where $\mathfrak{q}$ is isomorphic to $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n$, i.e. $\mathfrak{q} \approx \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n$. The *velocity of the multi-body system* belongs to $\mathfrak{q}$. We define it as the following triplet $\nu = \left({}^{\mathcal{I}}\dot{p}_B, {}^{B[\mathcal{I}]}\dot{\omega}_{\mathcal{I},B}, \dot{s}\right)$. From here on, we assume that the multi-body system interacts with the environment, exchanging n_c distinct 6D spatial forces – see Section 2.2.2. Since the configuration space is not a vector space, we cannot apply the classical Euler-Lagrange equations. This is solved by employing the Euler-Poincaré formalism [Marsden and Ratiu, 2010, Chapter 13.5], obtaining as a final result:

$$
M(q)\dot{\nu} + C(q,\nu)\nu + G(q) = \begin{bmatrix} 0_{6\times n} \\ I_n \end{bmatrix} \tau_s + \sum_{k=1}^{n_c} J_{\mathcal{C}_k}^{\top}\, {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k. \tag{3.3.1}
$$

Here $M \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix, $C \in \mathbb{R}^{(n+6)\times(n+6)}$ accounts for Coriolis and centrifugal effects, and $G \in \mathbb{R}^{n+6}$ contains the gravity term. $\tau_s \in \mathbb{R}^n$ is a vector representing the internal actuation torques, ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k \in \mathbb{R}^6$ denotes the k-th external

wrench applied by the environment on the robot expressed in mixed representation. The Jacobian $J_{\mathcal{C}_k}$ is the mixed velocity Jacobian of the contact frame $\mathcal{C}_k$. Equation (3.3.1) is strictly related to the rigid body dynamics presented in Section 2.3. In fact, we introduce the *left-tirvialized Lagrangian of the multi-body system* $\mathcal{L}(q, \nu)$ as

$$\mathcal{L}(q, \nu) = \mathcal{K}(\nu) - \mathcal{U}(q) \tag{3.3.2}$$

where $\mathcal{K}$ and $\mathcal{U}$ represent the *kinetic* and *potential* energy of the multi-body system and they are described in [Traversaro, 2017, Section 3.5]. It is possible to prove that Equation (3.3.1) is the solution of *Hamilton's Variational Principle* considering the Lagrangian function (3.3.2).

By stacking all the Jacobians and contact wrenches, we can rewrite Equation (3.3.1) as follows:

$$M(q)\dot{\nu} + C(q, \nu)\nu + G(q) = \begin{bmatrix} 0_{6\times n} \\ I_n \end{bmatrix} \tau_s + J_{\mathcal{C}}^{\top} \mathrm{f}, \tag{3.3.3}$$

where

$$J_{\mathcal{C}}(q) = \begin{bmatrix} J_{\mathcal{C}_1}(q) \\ \vdots \\ J_{\mathcal{C}_{n_c}}(q) \end{bmatrix}, \quad \mathrm{f} = \begin{bmatrix} {}_{\mathcal{C}_1[\mathcal{I}]}\mathrm{f}_1 \\ \vdots \\ {}_{\mathcal{C}_{n_c}[\mathcal{I}]}\mathrm{f}_{n_c} \end{bmatrix}. \tag{3.3.4}$$

In the case of rigid contacts between the robot and environment, we consider a set of holonomic constraints represented as

$${}^{\mathcal{I}}H_{\mathcal{C}_i} \equiv {}^{\mathcal{I}}\bar{H}_{\mathcal{C}_i}. \tag{3.3.5}$$

In other words, we require that the position and orientation of the frame associated with each contact link to be constant and equal to ${}^{\mathcal{I}}\bar{H}_{\mathcal{C}_i}$. By time differentiating Equation (3.3.5) and recalling that $\mathrm{v} = J_{C_i}\nu$ (3.2.17), we rewrite the rigid contact constraint in the *Pfaffian form* [Lynch and Park, 2017, Section 8.7] as

$$J_{\mathcal{C}_i}(q)\nu = 0. \tag{3.3.6}$$

Here, we prescribe the spatial velocity associated with each link in contact equal to zero. Since the Jacobian matrix is a smooth mapping from $\mathcal{Q}$ to $\mathbb{R}^{6\times(6+n)}$, Equation (3.3.6) can be differentiated

$$J_{\mathcal{C}_i}\dot{\nu} + \dot{J}_{\mathcal{C}_i}\nu = 0, \tag{3.3.7}$$

obtaining a dependency on $\dot{\nu}$. Equations (3.3.3) and (3.3.7) together are the dynamical equations that describe the motion of a floating-base system that instantiates rigid contacts with the environment.

The dynamics (3.3.1) is often expressed by separating the first 6 rows, referring to the non-actuated floating base, from the last n rows referring to the actuated joints as:

$$M_\nu(q)\dot{\nu} + h_\nu(q,\nu) = \sum_{k=1}^{n_c} J_{\mathcal{C}_{k_\nu}}^\top(q)\ {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k, \tag{3.3.8a}$$

$$M_s(q)\dot{\nu} + h_s(q,\nu) = \tau_s + \sum_{k=1}^{n_c} J_{\mathcal{C}_{k_s}}^\top(q)\ {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k, \tag{3.3.8b}$$

where $h(q,\nu) = C(q,\nu) + G(q)$, the subscript ν refers to the first 6 rows of the matrix, while s refers to the last n rows.

3.4 Centroidal dynamics

Let us introduce the *articulated body momentum* as the sum of the 6D spatial momentum (2.3.7) of each rigid body of the floating base system, i.e.

$${}_B h = \sum_{j=1}^{n_c} \mathbf{Ad}^*_{B H_j}\ {}_j\mathbb{M}_j^j \mathrm{v}_{\mathcal{I},j}. \tag{3.4.1}$$

In some cases, it is convenient to introduce the *articulated body centroidal momentum*. Let $\bar{G}$ be a frame whose origin is located on the CoM of the multi-body system, while the orientation is parallel to the inertial frame $\mathcal{I}$. The *articulated body centroidal momentum*, often shortened to *centroidal momentum* and denoted with ${}_{\bar{G}}h \in \mathbb{R}^6$ is given by

$${}_{\bar{G}}h = \begin{bmatrix} {}_{\bar{G}}h^p \\ {}_{\bar{G}}h^\omega \end{bmatrix} := \mathbf{Ad}^*_{\bar{G} H_B}\ {}_B h \tag{3.4.2a}$$

$$= \mathbf{Ad}^*_{\bar{G} H_B} \sum_j \mathbf{Ad}^*_{B H_j}\ {}_j\mathbb{M}_j^j \mathrm{v}_{\mathcal{I},j} \tag{3.4.2b}$$

$$= {}_{\bar{G}}X^B \sum_j {}_B X^i\ {}_j\mathbb{M}_j^j \mathrm{v}_{\mathcal{I},j}. \tag{3.4.2c}$$

It is worth recalling that the linear component of the centroidal momentum ${}_{\bar{G}}h^p$ depends linearly on the CoM velocity

$${}_{\bar{G}}h^p = m\dot{x}_{\mathrm{CoM}}. \tag{3.4.3}$$

Seeing that the link velocity linearly depends on the velocity of the multi-body system ν (3.2.17), Equation (3.4.2) can be factorized as follows

$$ {}_{\bar{G}}h = J_{\text{CMM}}\nu \tag{3.4.4} $$

where $J_{\text{CMM}} \in \mathbb{R}^{6\times n}$ is the *Centroidal Momentum Matrix* (CMM) [Orin and Goswami, 2008].

The centroidal momentum rate of change balances the external wrenches applied to the robot:

$$ {}_{\bar{G}}\dot{h} = \sum_{j=1}^{n_c} {}_{\bar{G}}X^{\mathcal{C}_j}\, {}_{\mathcal{C}_j}\mathrm{f}_j + m\bar{g} \tag{3.4.5a} $$

$$ = \sum_{j=1}^{n_c} \begin{bmatrix} {}^{\mathcal{I}}R_{\mathcal{C}_j} & 0_{3\times 3} \\ \left[\left({}^{\mathcal{I}}p_j - x_{\text{CoM}}\right)\times\right] {}^{\mathcal{I}}R_{\mathcal{C}_j} & {}^{\mathcal{I}}R_{\mathcal{C}_j} \end{bmatrix} {}_{\mathcal{C}_j}\mathrm{f}_j + m\bar{g} \tag{3.4.5b} $$

The adjoint matrix ${}_{\bar{G}}X^{\mathcal{C}_j} \in \mathbb{R}^{6\times 6}$ transforms the contact wrench from the application frame (located in ${}^{\mathcal{I}}p_j$ with orientation ${}^{\mathcal{I}}R_j$) to $\bar{G}$. Finally, $\bar{g} = \begin{bmatrix} 0 & 0 & -g & 0 & 0 & 0 \end{bmatrix}^\top$ is the 6D gravity acceleration vector.

If the external wrenches are expressed in mixed representation, the centroidal momentum dynamics (3.4.5) becomes

$$ {}_{\bar{G}}\dot{h} = \sum_{j=1}^{n_c} {}_{\bar{G}}X^{\mathcal{C}_j[\mathcal{I}]}\, {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f} + m\bar{g} \tag{3.4.6a} $$

$$ = \sum_{j=1}^{n_c} \begin{bmatrix} I_3 & 0_{3\times 3} \\ \left({}^{\mathcal{I}}p_j - x_{\text{CoM}}\right)\times & I_3 \end{bmatrix} {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j + m\bar{g}. \tag{3.4.6b} $$

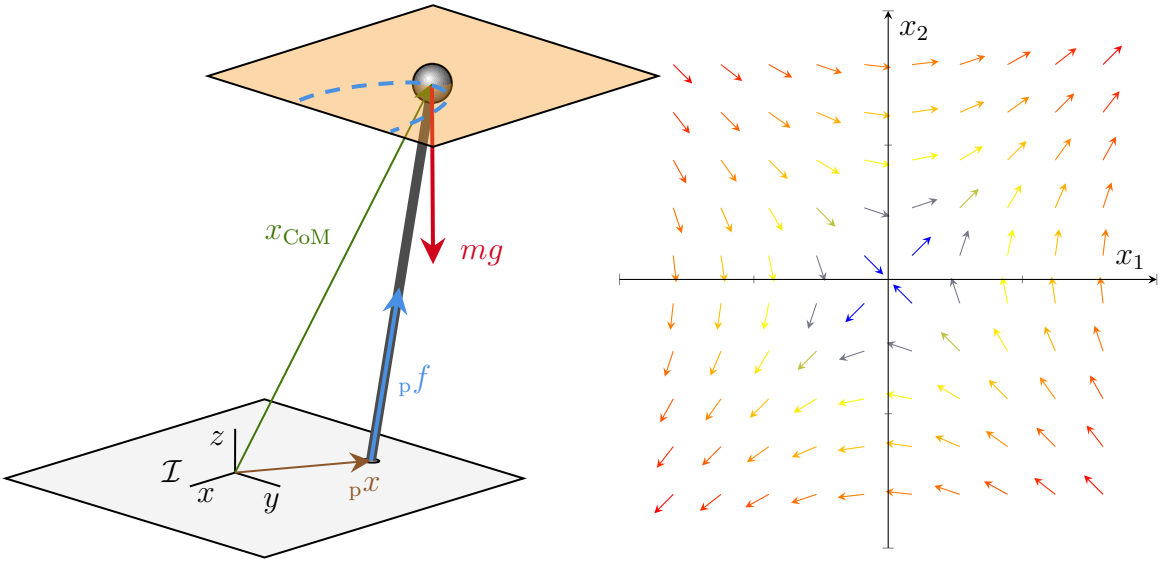
As will become evident later on in the manuscript, the centroidal momentum dynamics will play a crucial role in the design of the controller presented in Chapter 11.

Chapter 4

Simplified Models for Locomotion

In Chapter 3 we present the kinematics and the dynamics of a floating base system. In this chapter, we present some approximations of the centroidal dynamics presented in Section 3.4 by assuming some simplifying hypotheses. The tools introduced in this chapter will be considered in the design of the simplified model controllers presented in Chapter 10.

Assuming that a robot makes a single contact with the environment while keeping a constant center of mass height, the robot's centroidal dynamics can be approximated to the well-known Linear Inverted Pendulum (LIPM) [Kajita et al., 2003, 2001] and the Capture Point model [Pratt et al., 2006]. These two models assume constant angular momentum and approximate the center of mass dynamics with a linear time-invariant dynamical system, making the study of a feasible trajectory for the center of mass simpler. The LIPM and the CP became very popular along with the Zero Moment Point (ZMP) as a contact feasibility criterion [Vukobratovic and Juricic, 1969]. Similar to the ZMP, the centroidal moment pivot (CMP) has a widespread diffusion in the modeling and control of floating base system [Hopkins et al., 2014; Li et al., 2020; Popovic et al., 2005; Seyde et al., 2018; Shafiee-Ashtiani et al., 2017a]. The definition of the CMP is pivotal in considering a variable centroidal angular momentum while generating a feasible CoM trajectory. Indeed, when the CMP corresponds to the ZMP the centroidal angular momentum remains constant. Englsberger et al. [2015a, 2011] extends the Capture Point in the 3D scenario, defining the *Divergent Component of Motion* (DCM). The DCM is a linear combination of the center of mass position and velocity. By definition, the CoM can be indirectly stabilized by tracking an appropriate DCM trajectory. The DCM model assumes a constant natural frequency of the LIPM for trajectory planning. As a consequence, the ZMP can deviate from the desired



(a) Model representation (b) Phase space representation

Figure 4.1 The linear inverted pendulum approximates a humanoid robot walking on a planar ground while keeping a constant CoM height and zero centroidal angular momentum rate of change.

reference when the vertical CoM dynamics does not match the time-invariant LIPM dynamics. Hopkins et al. [2014] attempt at loosening this assumption by extending the DCM to consider a time-varying natural frequency.

In this chapter, we present a short overview of the simplified models that are often exploited by the robot locomotion community. In particular, Section 4.1 introduces the LIPM. Section 4.2 and 4.3 present the ZMP and CMP, respectively. We introduce the DCM in Section 4.4. Finally, Section 4.5 extends the definition of DCM to the time-varying case.

4.1 The linear inverted pendulum

When a biped robot supports its body on one leg, its dynamics can be approximated by the Linear Inverted Pendulum Model (LIPM). This model approximates floating base dynamics (3.3.1) simply considering the center of mass, which is represented as a point mass on top of a pendulum with negligible inertia and one contact, i.e., one stance foot, which is rigidly in contact with the ground – Figure 4.1a.

Let us assume an inertial frame $\mathcal{I}$ whose z is pointing against the gravity vector g and assuming:

1. a pure force ${}_pf \in \mathbb{R}^3$ located at the point ${}_px$ on the ground acts on the system;

2. the contact point ${}_px$ is weakly stable, i.e., ${}_p\dot{x} = 0$ and the contact force ${}_pf$ belongs to the friction cone [1], that is, ${}_pf \in \mathcal{Q}^3_\mu$ where

$$\mathcal{Q}^3_\mu = \left\{ {}_pf \in \mathbb{R}^3 \mid {}_pf_z\mu \geq \sqrt{{}_pf_x^2 + {}_pf_y^2} \right\} \tag{4.1.1}$$

3. the z component of the distance between contact location ${}_px$ and the CoM x_{CoM} is kept constant during the motion i.e., $e_3^\top({}_px - x_{\text{CoM}}) = h$, $e_3^\top \dot{x}_{\text{CoM}} = 0$, $e_3^\top \ddot{x}_{\text{CoM}} = 0$;

4. the centroidal angular momentum is constantly zero, i.e., ${}_{\bar{G}}h^\omega = 0$ and ${}_{\bar{G}}\dot{h}^\omega = 0$.

Taking into account the assumptions, the centroidal momentum dynamics (3.4.5) becomes

$${}_{\bar{G}}h = \begin{bmatrix} {}_{\bar{G}}h^p \\ {}_{\bar{G}}h^\omega \end{bmatrix} = \begin{bmatrix} {}_pf - mg \\ ({}_px - x_{\text{CoM}})^\wedge {}_pf \end{bmatrix} \tag{4.1.2a}$$

$$({}_px - x_{\text{CoM}})^\wedge {}_pf = 0. \tag{4.1.2b}$$

Given the hypothesis 3 and the constraint (4.1.2b), the components ${}_pf$ writes as

$${}_pf_x = m\zeta^2 e_1^\top \left(x_{\text{CoM}} - {}_{\text{p}}x\right), \tag{4.1.3a}$$

$${}_pf_y = m\zeta^2 e_2^\top \left(x_{\text{CoM}} - {}_{\text{p}}x\right), \tag{4.1.3b}$$

$${}_pf_z = -m|g|, \tag{4.1.3c}$$

where $\zeta = \sqrt{\frac{|g|}{h}}$ is the time constant of the LIPM. ${}_pf_x$, ${}_pf_y$, and ${}_pf_z$ represent the components x, y, and z of the vector ${}_pf$. By substituting the contact force values (4.1.3)

[1] As discussed in Section 5.1.2, the friction cone is an example of a second-order cone. Where a second-order cone, also known as Lorentz cone, is given by

$$\begin{aligned} \mathcal{Q}^{n+1} &= \left\{ \begin{bmatrix} x \\ t \end{bmatrix} \in \mathbb{R}^{n+1} \;\middle|\; \|x\| \leq t \right\} \\ &= \left\{ \begin{bmatrix} x \\ t \end{bmatrix} \in \mathbb{R}^{n+1} \;\middle|\; \begin{bmatrix} x \\ t \end{bmatrix}^\top \begin{bmatrix} I_n & 0_{n\times 1} \\ 0_{1\times n} & -1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq 0, t \geq 0 \right\}. \end{aligned}$$

into the dynamics (4.1.2), we obtain the following:

$$\ddot{x}_{\text{CoM}} = \begin{bmatrix} e_1^\top \\ e_2^\top \\ 0_{3\times 1} \end{bmatrix} \zeta^2 \left(x_{\text{CoM}} - {}_{\text{p}}x \right).$$

Since the dynamics is different from zero only in the planar coordinates, we can define

$$x_{\text{LIP}} = \begin{bmatrix} e_1^\top \\ e_2^\top \end{bmatrix} x_{\text{CoM}}, \quad x_{\text{LIP}} \in \mathbb{R}^2, \tag{4.1.4}$$

and, without loss of generality, we consider ${}_{\text{p}}x \in \mathbb{R}^2$. We then obtain the LIP dynamic equation,

$$\ddot{x}_{\text{LIP}} = \zeta^2 \left(x_{\text{LIP}} - {}_{\text{p}}x \right), \tag{4.1.5}$$

The LIP model (4.1.5) is described by a second-order linear dynamical system. It is worth noticing that the dynamics of the x and y coordinates of the LIPM are completely decoupled; as a consequence it is possible to focus our analysis on only one component, then the same observations will hold also for the other. Without loss of generality, let us define $x_1 = x_{\text{LIP}_x}$ and $x_2 = \dot{x}_{\text{LIP}_x}$, the state-space representation of the (4.1.5) writes as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \zeta^2 & 0_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\zeta^2 \end{bmatrix} {}_p x_x = A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + B \, {}_p x_x \tag{4.1.6}$$

The system has one single equilibrium point given by

$$\begin{bmatrix} x^e_{\text{LIP}_x} \\ \dot{x}^e_{\text{LIP}_x} \end{bmatrix} := \begin{bmatrix} {}_p x_x \\ 0 \end{bmatrix}. \tag{4.1.7}$$

Since the eigenvalues of A are equal to $\lambda_{1,2} = \pm\zeta$, the equilibrium point is unstable. Figure 4.1b shows the state-space representation of the dynamics (4.1.6). It is worth noting that the dynamical system is completely controllable, in fact, the rank of the controllability matrix $\mathcal{R}$ is equal to the dimension of the system. As a consequence, it is possible to design a control system such that the dynamics (4.1.6) is asymptotically stabilized.

The LIP model can be extended considering also a finite-sized-foot [Kajita et al., 2001; Koolen et al., 2012]. In this case, it can be shown that ${}_{\text{p}}x$ corresponds to the

position of the zero moment point x_{ZMP} [Vukobratović et al., 2004], instead of the position of the foot.

4.2 The zero moment point

Consider a rigid body that makes a contact with a surface and assume that:

1. there exists an inertial frame $\mathcal{I}$;
2. there exist a frame B rigidly attached to the body and we denote o_B the origin of the frame and $[B]$ its orientation;
3. there exists a contact domain $\Omega \in \mathbb{R}^3$, we denote with ${}^B x$ a point in the contact surface expressed in the the frame B;
4. $\forall\, {}^B x \in \Omega$ there exists a continuous pure force distribution that depends on the point location, i.e.,
$$\rho : \mathbb{R}^3 \longrightarrow \mathbb{R}^3. \tag{4.2.1}$$

Given the above assumption, the contact torque distribution about a point p_B, $\sigma_{o_B} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ writes
$$\sigma_{o_B} = {}^B x \times \rho({}^B x). \tag{4.2.2}$$

Once the pure force and torque distribution are defined, then the equivalent left trivialized contact 6D force, i.e., in body frame writes as [Caron et al., 2015]
$${}_B\mathrm{f} = \begin{bmatrix} {}_B f \\ {}_B \mu \end{bmatrix} = \begin{bmatrix} \int_\Omega \rho \, \mathrm{d}\, \Omega \\ \int_\Omega \sigma_{o_B} \, \mathrm{d}\, \Omega \end{bmatrix}. \tag{4.2.3}$$

Let us introduce another frame $F[B] := (o_F, [B])$ placed in the contact domain Ω as a frame that has its origin in o_F and is oriented as B. Now we aim to find the origin o_F such that the tangential component of the angular term of ${}_{F[B]}\mathrm{f}$ is equal to zero, i.e., $e_4^\top \, {}_{F[B]}\mathrm{f} = e_5^\top \, {}_{F[B]}\mathrm{f} = 0$. The location of the origin o_F is crucial in the study of the contact dynamic balance, which is achieved by ensuring that the contact area Ω remains invariant. If o_F exists and belongs to the contact domain, i.e., $o_F \in \Omega$ the contact dynamic balance is ensured and o_F is equivalent to the Zero Moment Point (ZMP), often denoted by x_{ZMP}. Otherwise, if $o_F \notin \Omega$, then x_{ZMP} is not defined and the rigid body will rotate. This statement is generally denoted by the ZMP condition [Arakawa and Fukuda, 1997], also known as the ZMP stability criterion [Li et al., 1998].

Given a 6D force ${}_B\mathrm{f}$, if the ZMP exists, it is given by [Vukobratović et al., 2004]:

$$ {}^B x_{\mathrm{ZMP}} = \begin{bmatrix} -\frac{{}_B\mu_y}{{}_Bf_z} \\ \frac{{}_B\mu_x}{{}_Bf_z} \end{bmatrix}. \tag{4.2.4} $$

The concept of ZMP has been frequently used in robot control [Hirai et al., 1998; Kajita et al., 2003, 2010; Shih, 1996] as a criterion of postural stability, however, we observe that:

1. the term *zero* moment point is misleading since, in general, only two of the three moments components are zero;
2. even if the ZMP criterion is satisfied, the contact may not be *weak contact stable* [2]. In fact, the ZMP can be defined, but the pure force ${}_{F[B]}\mathrm{f}$ may not belong to the friction cone.

Despite these weaknesses of the ZMP condition, the criterion is widely used with the LIPM where weakly stability of the contact is considered as a hypothesis. Combining the definition of the ZMP with the LIP the CoM dynamics becomes

$$ \ddot{x}_{\mathrm{LIP}} = \zeta^2 \left(x_{\mathrm{LIP}} - {}^{\mathcal{I}}H_B \, {}^B x_{\mathrm{ZMP}} \right). \tag{4.2.5} $$

4.2.1 Connection between the ZMP and the centroidal momentum dynamics

Given a multi-body system that makes a contact with the environment. Consider a rigidly attached frame to the contact surface $B := (o_B, [B])$ and a frame placed on the CoM, $G[B] := (x_{\mathrm{CoM}}, [B])$, that has its origin in the center of mass x_{CoM} and is oriented as B. The centroidal dynamics of the system (3.4.5) is given by

$$ {}_{G[B]}\dot{h} = \begin{bmatrix} {}_{G[B]}\dot{h}^p \\ {}_{G[B]}\dot{h}^\omega \end{bmatrix} = {}_{G[B]}X^B \, {}_B\mathrm{f} + m\bar{g}. \tag{4.2.6} $$

[2] A contact is weakly stable if an only if [Caron et al., 2015]: i)the relative velocity and acceleration of the contact are zero, ii) the 6D wrench belongs to the wrench cone.

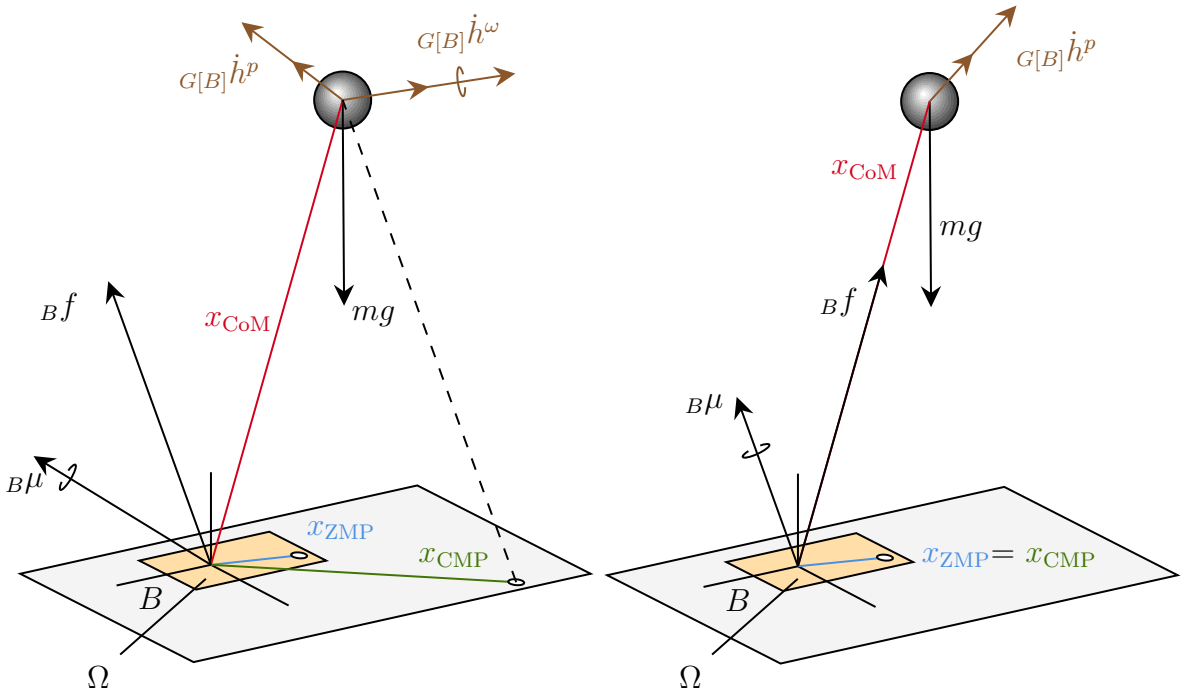


Figure 4.2 The CMP is the point at which the ground reaction force must act to maintain the horizontal component of the centroidal angular momentum constant. The CMP corresponds with the ZMP when the rate of change of the centroidal angular momentum is zero - 4.2b. When the centroidal angular momentum is not constant, the CMP and the ZMP are two different points.

Substituting ${}_{B}\text{f}$ from (4.2.6) into (4.2.4), we write a relationship between the ZMP and the centroidal momentum as

$$ {}^{B}x_{\text{ZMP}} = \begin{bmatrix} {}^{B}x_{\text{CoM}_x} - \frac{{}_{G[B]}\dot{h}^{\omega}_{y}}{{}_{G[B]}\dot{h}^{p}_{z}+m\|\bar{g}\|} - {}^{B}x_{\text{CoM}_z}\frac{{}_{G[B]}\dot{h}^{p}_{x}}{{}_{G[B]}\dot{h}^{p}_{z}+m\|\bar{g}\|} \\ {}^{B}x_{\text{CoM}_y} + \frac{{}_{G[B]}\dot{h}^{\omega}_{x}}{{}_{G[B]}\dot{h}^{p}_{z}+m\|\bar{g}\|} - {}^{B}x_{\text{CoM}_z}\frac{{}_{G[B]}\dot{h}^{p}_{y}}{{}_{G[B]}\dot{h}^{p}_{z}+m\|\bar{g}\|} \end{bmatrix}. \tag{4.2.7} $$

4.3 The centroidal moment pivot

Given a multi-body system that interacts with its surroundings. Consider a frame that is rigidly coupled to the contact surface $B := (o_B, [B])$ and a frame that is positioned on the CoM. The origin of $G[B] := (x_{\text{CoM}}, [B])$ is at the center of mass x_{CoM}, and is oriented as B. The Centroidal Moment Pivot (CMP) [Popovic et al., 2005] is defined as the point where a line parallel to the contact force, passing through the CoM, intersects

the contact surface. More formally, we define the CMP, denoted as x_{CMP}, as

$$x_{\text{CMP}} \in \mathbb{R}^3 \quad \text{such that } ({}^B x_{\text{CMP}} - {}^B x_{\text{CoM}}) \times {}_B f = 0,\ e_3^\top {}^B x_{\text{CMP}} = 0. \tag{4.3.1}$$

Expanding Equation (4.3.1), the components ${}^B x_{\text{CMP}}$ write as

$${}^B x_{\text{CMP}_x} = {}^B x_{\text{CoM}_x} - \frac{{}_B f_x}{{}_B f_z} x_{\text{CoM}_z}, \tag{4.3.2a}$$

$${}^B x_{\text{CMP}_y} = {}^B x_{\text{CoM}_y} - \frac{{}_B f_y}{{}_B f_z} x_{\text{CoM}_z}, \tag{4.3.2b}$$

$${}^B x_{\text{CMP}_z} = 0. \tag{4.3.2c}$$

Combining Equation (4.2.7) with the CMP definition (4.3.2) we can express the CMP in terms of ZMP location, rate of change of the centroidal angular momentum, and the ground reaction force as

$${}^B x_{\text{CMP}_x} = {}^B x_{\text{ZMP}_x} + \frac{{}_{G[B]}\dot{h}^\omega_y}{{}_B f_z}, \tag{4.3.3a}$$

$${}^B x_{\text{CMP}_y} = {}^B x_{\text{ZMP}_y} - \frac{{}_{G[B]}\dot{h}^\omega_x}{{}_B f_z}, \tag{4.3.3b}$$

$${}^B x_{\text{CMP}_z} = 0. \tag{4.3.3c}$$

To give the reader a better comprehension, we can image the CMP as the point where the ground reaction force would have to act to keep the horizontal component of the whole-body angular momentum constant. When the centroidal angular momentum is constant, i.e., ${}_{G[B]}\dot{h}^\omega = 0_{3\times 1}$ the CMP coincides with the ZMP – Figure 4.2. Moreover, while by definition the ZMP cannot leave the contact domain Ω, the CMP, in the case of ${}_{G[B]}\dot{h}^\omega_x \neq 0$ or ${}_{G[B]}\dot{h}^\omega_y \neq 0$, can. Let us assume that the motion of the multi-body system is approximated by the LIPM, i.e., the hypothesis in Section 4.1 holds, then it is possible to prove that the position of the CMP coincides with the ZMP. To prove this statement, it is worth noticing that when the hypotheses of the LIPM are satisfied the centroidal angular momentum is kept constant – Section 4.1, hence, substituting ${}_{G[B]}\dot{h}^\omega$ into Equation (4.3.3), we can conclude that if the system is approximated by the LIPM, then the CMP and ZMP coincide.

4.4 The divergent component of motion

Given a multi-body system making n_c contacts with the environment and having a mass equal to m. Let us consider an inertial frame $\mathcal{I} := (o_{\mathcal{I}}, [\mathcal{I}])$. Let $\bar{G}; = (x_{\text{CoM}}, [\mathcal{I}])$ a frame whose origin is located on the CoM of the multi-body system, while the orientation is parallel to the inertial frame $\mathcal{I}$. The CoM acceleration is given by:

$$\ddot{x}_{\text{CoM}} = mg + {}_{\bar{G}}f, \tag{4.4.1}$$

where ${}_{\bar{G}}f$ is the sum of all the external forces acting on the robot CoM. The divergent component of motion (DCM) [Englsberger et al., 2013, 2015a, 2011], often denoted with ξ, is a linear transformation of the Center of Mass state:

$$\xi = x_{\text{CoM}} + b\dot{x}_{\text{CoM}}, \tag{4.4.2}$$

where $b \in \mathbb{R}_+$ is a strictly positive constant.

By reordering Equation (4.4.2) the CoM dynamics can be derived as

$$\dot{x}_{\text{CoM}} = -\frac{1}{b}(x_{\text{CoM}} - \xi). \tag{4.4.3}$$

The CoM position is governed by a stable first-order system. It is worth noticing that given a desired DCM set-point $\xi(t) = \bar{\xi}$, the CoM position will converge to it, Indeed let us define the error e as $e = x_{\text{CoM}} - \xi$, the error dynamics is $\dot{e} = -e/b$ which is an asymptotically stable system.

By differentiating (4.4.3) and combining it with (4.4.2) and (4.4.1), the DCM dynamics holds:

$$\dot{\xi} = -\frac{1}{b}(x_{\text{CoM}} - \xi) + b\ddot{x}_{\text{CoM}} \tag{4.4.4a}$$

$$= -\frac{1}{b}(x_{\text{CoM}} - \xi) + \frac{b}{m}({}_{\bar{G}}f + mg). \tag{4.4.4b}$$

The main idea is to design the external forces to be appropriate for the robot's walking task while the feasibility constraints are satisfied (i.e., the center of pressure inside the support polygon). For the sake of simplicity, a force-to-point transformation is used to express the external forces:

$${}_{\bar{G}}f = \gamma(x_{\text{CoM}} - x_{\text{eCMP}}), \tag{4.4.5}$$

where γ is a positive constant and eCMP is the enhanced centroidal moment pivot point [Englsberger et al., 2013, 2015a]. It is important to point out that the eCMP is related to the CMP [Popovic et al., 2005]. The first one could not belong to the ground plane. The second is the intersection point between the ground surface and the line between the CoM and the eCMP. Combining Equation (4.4.4) with (4.4.5), we rewrite the DCM as:

$$\dot{\xi} = \left(\frac{b\gamma}{m} - \frac{1}{b}\right) x_{\text{CoM}} + \frac{1}{b}\xi - \frac{b\gamma}{m} x_{\text{eCMP}} + bg, \tag{4.4.6}$$

this shows that the states x_{CoM} and ξ are in general coupled, however choosing the parameter γ equal to m/b^2 the DCM dynamics (4.4.6) becomes independent of the CoM position

$$\dot{\xi} = \frac{1}{b}\xi - \frac{1}{b} x_{\text{eCMP}} + bg. \tag{4.4.7}$$

Let us introduce the virtual repellent point (VPR) as

$$x_{\text{VRP}} = x_{\text{eCMP}} + b^2 g, \tag{4.4.8}$$

the DCM dynamics can be simplified as:

$$\dot{\xi} = \frac{1}{b}(\xi - x_{\text{VRP}}). \tag{4.4.9}$$

This shows that the 3D-DCM dynamic equation is a first-order unstable dynamic system $\forall b > 0$.

Combining Equation (4.3.2) with (4.4.5), we can express the CMP in terms of the eCMP and the ground reaction force as:

$$^{B}x_{\text{CMP}_x} = {}^{B}x_{\text{eCMP}_x} + \frac{b^2}{m} {}_{G[B]}f_x - \frac{{}_{G[B]}f_x}{{}_{G[B]}f_z}\left(x_{\text{eCMP}_z} + \frac{b^2}{m} {}_{G[B]}f_z\right), \tag{4.4.10a}$$

$$^{B}x_{\text{CMP}_y} = {}^{B}x_{\text{eCMP}_y} + \frac{b^2}{m} {}_{G[B]}f_y - \frac{{}_{G[B]}f_y}{{}_{G[B]}f_z}\left(x_{\text{eCMP}_z} + \frac{b^2}{m} {}_{G[B]}f_z\right), \tag{4.4.10b}$$

$$^{B}x_{\text{CMP}_z} = 0. \tag{4.4.10c}$$

Finally, it is worth to notice that while moving the z coordinate of the eCMP will change, indeed combining (4.3.2) with the CoM dynamics (4.4.1) and projecting it on

the z coordinate we obtain:

$$x_{\mathrm{eCMP}_z} = x_{\mathrm{CoM}_z} - b^2 \left(\ddot{x}_{\mathrm{CoM}_z} + |g|\right). \tag{4.4.11}$$

In other words, if the height of the CoM varies, the height of the eCMP will change accordingly.

4.4.1 Connection between the DCM and the LIPM

Let us assume that the motion of the multi-body system is approximated by the LIPM, i.e., the hypothesis in Section 4.1 holds, and b is equal to the inverse of the LIPM time constant, that is, $b = 1/\zeta$, then it is possible to prove that:

1. the position of the eCMP coincides with the CMP and the ZMP;
2. the z coordinate of the VRP and the DCM coincides with the CoM height.

To prove the first statement, it is worth noticing that we should prove that the eCMP and the CMP coincide, indeed in Section 4.3, we already show that when the model is approximated by the LIPM the CMP and ZMP coincide. In the LIPM regime, the z component of the CoM acceleration is forced to be zero. So subsisting the assumption in (4.4.11) and making explicit b we obtain:

$$x_{\mathrm{eCMP}_z} = x_{\mathrm{CoM}_z} - \frac{|g|}{\zeta} = x_{\mathrm{CoM}_z} - \frac{|g|}{|g|} x_{\mathrm{CoM}_z} = 0. \tag{4.4.12}$$

Now, substituting (4.4.12) into (4.4.10), we obtain that ${}^{B}x_{\mathrm{CMP}} = {}^{B}x_{\mathrm{eCMP}}$.

To prove the second statement we can consider the CoM dynamics (4.4.3) projected on the z coordinate, since $\dot{x}_{\mathrm{CoM}_z} = 0$, it becomes evident that $\xi_z = x_{\mathrm{CoM}_z} = h$. By projecting (4.4.9) on the z coordinate, we have $\dot{\xi}_z = \frac{1}{b}(\xi_z - x_{\mathrm{VRP}_z}) = 0$, hence $\xi_z = x_{\mathrm{VRP}_z}$.

Since the DCM dynamics is different from zero only in the planar coordinates, we define

$$\xi_{\mathrm{LIP}} = \begin{bmatrix} e_1^\top \\ e_2^\top \end{bmatrix} \xi, \quad \xi_{\mathrm{LIP}} \in \mathbb{R}^2, \tag{4.4.13}$$

We then obtain the CoM and DCM dynamics if the LIP hypotheses are satisfied:

$$\dot{x}_{\mathrm{LIP}} = -\zeta \left(x_{\mathrm{LIP}} - \xi_{\mathrm{LIP}}\right), \tag{4.4.14a}$$

$$\dot{\xi}_{\mathrm{LIP}} = \zeta \left(\xi_{\mathrm{LIP}} - x_{\mathrm{ZMP}}\right). \tag{4.4.14b}$$

4.5 The time-varying DCM

Given a multi-body system making n_c contacts with the environment and having a mass equal to m, an inertial frame $\mathcal{I} := (o_{\mathcal{I}}, [\mathcal{I}])$ and $\bar{G} := (x_{\text{CoM}}, [\mathcal{I}])$. We previously show that the DCM dynamics is defined by (4.4.2), where the parameter b is a constant positive number. The time-varying DCM [Hopkins et al., 2014] extends the DCM definition [Englsberger et al., 2015a] to account for a natural frequency that changes over time. Hopkins et al. [2014] define the time-varying DCM as

$$\xi = x_{\text{CoM}} + \eta(t)\dot{x}_{\text{CoM}}. \tag{4.5.1}$$

where $\eta(t)$ is a strictly positive function such that $\eta(t) > \bar{\eta} > 0$ By reordering Equation (4.5.1) the CoM dynamics can be derived as

$$\dot{x}_{\text{CoM}} = -\frac{1}{\eta(t)}(x_{\text{CoM}} - \xi). \tag{4.5.2}$$

The CoM position is governed by an asymptotically stable first-order nonlinear system [Hopkins et al., 2014].

By differentiating (4.5.2) and combining it with (4.5.1) and (4.4.1), the Time-Varying DCM dynamics holds:

$$\dot{\xi} = -\frac{1}{\eta}(x_{\text{CoM}} - \xi) - \frac{\dot{\eta}}{\eta}(x_{\text{CoM}} - \xi) + \eta\ddot{x}_{\text{CoM}} \tag{4.5.3a}$$

$$= \left(\frac{1+\dot{\eta}}{\eta}\right)(\xi - x_{\text{CoM}}) + \frac{\eta}{m}({}_{\bar{G}}f + mg) \tag{4.5.3b}$$

Following the same approach as in Section 4.4, we ask for an external force ${}_{\bar{G}}f$ equal to

$${}_{\bar{G}}f = \frac{m(1+\dot{\eta})}{\eta^2}(x_{\text{CoM}} - x_{\text{eCMP}}), \tag{4.5.4}$$

we simply select the time-varying DCM dynamics (4.5.3) as:

$$\dot{\xi} = \frac{\dot{\eta}+1}{\eta}(\xi - x_{\text{eCMP}}) + \eta g. \tag{4.5.5}$$

Let us reformulate the virtual repellent point (VPR) (4.4.8) as

$$x_{\text{VRP}} := x_{\text{eCMP}} - \frac{\eta^2}{\dot{\eta}+1}g, \tag{4.5.6}$$

the DCM dynamics can be simplified as:

$$\dot{\xi} = \frac{\dot{\eta} + 1}{\eta}(\xi - x_{\text{VRP}}). \tag{4.5.7}$$

Note that for a constant parameter $\eta(t) = b$ and $\dot{\eta}(t) = b$, the time-varying DCM dynamics (4.5.7) the CoM dynamics (4.5.2) and the VRP definition (4.5.6) are equivalent to the time-invariant equations (4.4.9) (4.4.3) and (4.4.8), respectively.

We remark that if there exists an instant t^* such that $\eta(t^*) = 0$ or $\dot{\eta}(t^*) = -1$, the dynamics (4.5.2) is uncontrollable. If $\eta(t^*) = 0$, Equations (4.5.2), (4.5.5), and (4.5.4) are not defined. On the other hand, if $\dot{\eta}(t^*) = -1$ the sum of the external force ${}_{\bar{G}}f$ becomes zero – Equation (4.5.4). As a consequence, we have to enforce $\eta(t) \geq \epsilon_\eta$ and $\dot{\eta}(t) \geq \epsilon_{\dot{\eta}} - 1$ where ϵ_η $\epsilon_{\dot{\eta}}$ are any small positive number. As a consequence, the aforementioned constraints restrict the application of the time-varying DCM to tasks that do not consider states where the sum of external force ${}_{\bar{G}}f$ is equal to zero, i.e., all the tasks composed by a flight phase.

We finally highlight that for a choice of $\eta(t) = 1/\omega(t)$ and consequently $\dot{\eta}(t) = -\dot{\omega}(t)/\omega(t)^2$, Equations (4.5.2), (4.5.5) and (4.5.4) are equivalent to the equation introduced by Hopkins et al. [2014].

Chapter 5

Optimal Control and Non-Linear Optimization Basics

In previous chapters, we investigated the tools for modeling a floating base system that establishes a set of contacts with the environment. This chapter introduces the basics and terminology of *non-linear programming* and *optimal control*. We then take advantage of the technique presented here to design the controllers analyzed in Part II and Part III. Non-linear programming is the mathematical process of finding a set of variables such that a non-linear function is minimized (or maximized). Once the theory of *non-linear programming* is introduced, we apply it to the resolution of the *optimal control* problems. The *optimal control theory* is a branch of control theory that aims at finding a control for a dynamic system over a period of time such that an objective function is optimized.

The chapter will take an *utilitaristic* approach to describe such frameworks, to this concern we decided to avoid focusing on the rigorous proofs. The reader who wants a more rigorous understanding of non-linear optimization theory should consult the extensive literature. Here, it is worth mentioning [Boyd and Vandenberghe, 2004; Chachuat, 2007; Diehl et al., 2009] from which parts of this chapter took inspiration. On the other hand, complete and rigorous manuscripts on optimal control are [Allgöwer et al., 1999; Bemporad et al., 2002; Biral et al., 2016; Boyd and Vandenberghe, 2004; Qin and Badgwell, 2000].

The chapter is organized as follows. Sections 5.1 and 5.2 give an overview of convex sets and convex functions, respectively. Section 5.3 introduces the optimization problem. Section 5.4 presents the Quadratic Programming problem as a special case of the optimization problem. The Quadratic Programming problem will be extensively

exploited in the design of the whole-body controllers presented in Part II. Section 5.5 and 5.6 introduce some of the basics of Optimal Control and Model Predictive Control. The content of Section 5.6 will be crucial to fully comprehend the design of the controllers detailed in Part III.

5.1 Convex set

5.1.1 Affine and convex sets

Given two points, x_1, x_2 in a set $\mathbb{R}^n$ such that $x_1 \neq x_2$, we define $y \in \mathbb{R}^n$ the *line* passing through x_1 and x_2 as

$$y = \theta x_1 + (1 - \theta)x_2, \tag{5.1.1}$$

where $\theta \in \mathbb{R}$. When $\theta = 1$, y coincides with x_1, while if $\theta = 0$, $y = x_2$. A set C is affine if given two distinct points in C the line connecting them belongs to C, i.e., for any $x_1, x_2 \in C$, and $theta = [0, 1]$ then $\theta x_1 + (1 - \theta)x_2 \in C$. Given a set of points $x_1, x_2, \ldots, x_n$, we define the *affine combination of* $x_1, x_2, \ldots, x_n$, $\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ where $\theta_1 + \theta_2 + \cdots + \theta_n = 1$ and $\theta_i \geq 0$ for $i = 1, \ldots, n$.

Convex sets

A set C is said to be *convex* if the line segment between two points in C belongs to C. More formally, given any $x_1, x_2 \in C$ and any θ such that $0 \leq \theta \leq 1$ $\theta x_1 + (1-\theta)x_2 \in C$. To give the reader a better understanding, a set is convex if every point in the set can be connected with an unobstructed straight path between them. Figure 5.1a illustrates an example of a convex set, while Figure 5.1b presents an example of a nonconvex set.

Given n points, $x_1, x_2, \ldots, x_n$, and $\theta_1, \theta_2, \ldots, \theta_n \in \mathbb{R}$ such that $\theta_1 + \theta_2 + \cdots + \theta_n = 1$ and $\theta_i \geq 0$ for $i = 1, \ldots, n$, the *convex combination* of $x_1, x_2, \ldots, x_n$ is given by $\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$.

The *convex hull* of a set C, denoted with $\mathbf{conv}\, C$, is the set of all the convex combinations of the points in C and it is written as:

$$\mathbf{conv}\, C = \left\{ \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n | x_i \in C, \theta_i \geq 0, i = 1, \ldots, n, \sum_{i=1}^{n} \theta_i = 1 \right\}. \tag{5.1.2}$$

Here we underline that the convex hull of C is a convex set

(a) Convex set. (b) Nonconvex set.

Figure 5.1 Examples of convex and nonconvex sets. (a) The oval shape is a convex set (b) The *'s'* shaped set is not convex, since the red line segment between x_1 and x_2 in not fully contained in the set.

Convex cones

A set C is *cone* if for every point $x \in C$ and a real positive parameter $\theta \geq 0$, $\theta x \in C$. Given a cone C, if it is convex, we say that C is a *convex cone*, thus for any $x_1, x_2 \in C$ and $\theta_1, \theta_2 \geq 0$ the affine combination of x_1, x_2 belongs to C, i.e $\theta_1 x_1 + \theta_2 x_2 \in C$.

Given n points $x_1, x_2, \ldots, x_n$, and $\theta_1, \theta_2, \ldots, \theta_n \in \mathbb{R}$ such that $\theta_i \geq 0$ for $i = 1, \ldots, n$, the *conic combination* of $x_1, x_2, \ldots, x_n$ is given by $\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$. Here, it is worth recalling that if x_i is in a convex cone C, then every conic combination of x_i belongs in C The *conic hull* of a set C, denoted by $\mathbf{conic}\, C$, is the set of all the conic combinations of the points in C and it writes as:

$$\mathbf{conic}\, C = \{\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n | x_i \in C, \theta_i \geq 0, i = 1, \ldots, n\}\,. \tag{5.1.3}$$

5.1.2 Convex set examples

In this section, we recall some important examples of convex sets that we will encounter throughout the rest of the manuscript. We first introduce the hyperplanes, halfspaces and then second-order cones, also known as Lorentz cones. We conclude the section by presenting the polyhedra and the associated *Minkowski-Weyl* theorem.

Hyperplanes and halfspaces

Given a set H, we say that H is an *Hyperplane* if it can be represented in the form

$$H = \{x \in \mathbb{R}^n | a^\top x = b\}, \tag{5.1.4}$$

where $a \in \mathbb{R}^n$ and b is a real number. The hyperplane representation can be seen as the set of points with a constant scalar product with a vector a. Similarly, a can be seen as the *normal vector* of the plane. b is the offset of the plane from the origin. Given any point x_0 in the hyperplane, Equation (5.1.4) can be rewritten as $\{x \in \mathbb{R}^n | a^\top (x - x_0) = 0\}$

A hyperplane splits the space into two *halfspaces*. We define a *closed* halfspace as the convex set $\{x \in \mathbb{R}^n | a^\top x \leq b\}$. We note that the halspace $a^\top x \geq b$ is the halfspace extending in the direction of the vector a, while $a^\top x \leq b$ contains $-a$ – see Figure 5.3a.

Second order cone

Given a vector $x \in \mathbb{R}^n$, the Euclidean norm in $\mathbb{R}^n \parallel \ \parallel$ and a positive scalar t, we define the *second order cone* as

$$\mathcal{Q}^{n+1} = \left\{ \begin{bmatrix} x \\ t \end{bmatrix} \in \mathbb{R}^{n+1} \;\middle|\; \|x\| \leq t \right\} \tag{5.1.5a}$$

$$= \left\{ \begin{bmatrix} x \\ t \end{bmatrix} \in \mathbb{R}^{n+1} \;\middle|\; \begin{bmatrix} x \\ t \end{bmatrix}^\top \begin{bmatrix} I_n & 0_{n\times 1} \\ 0_{1\times n} & -1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq 0, t \geq 0 \right\}. \tag{5.1.5b}$$

The second-order cone, often named *Lorentz cone*, is a convex cone.

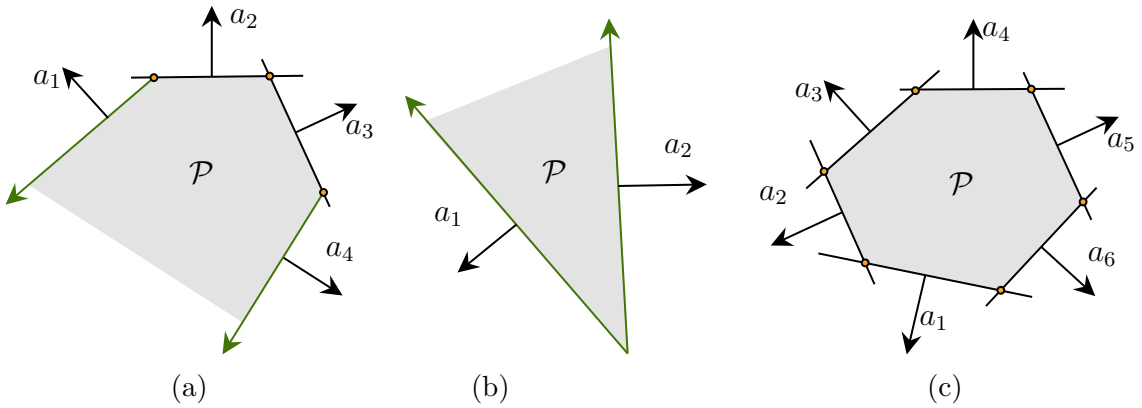
We notice that, in the context of rigid contact modeling, the friction cone (Equation (4.1.1)) is a Lorentz cone. Indeed, given the point in contact with the environment and a contact force $f \in \mathbb{R}^3$, we say that f belongs to the friction cone if and only if

$$\sqrt{f_1^2 + f_2^2} \leq \mu f_3, \tag{5.1.6}$$

where μ is the static friction coefficient, or in other words,

$$f \in \left\{ f \in \mathbb{R}^3 \mid \sqrt{f_1^2 + f_2^2} \leq \mu f_3 \right\}. \tag{5.1.7}$$

Setting $\mu f_3 = t$ and $x^\top = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$, Equation (5.1.7) is equivalent to (5.1.5). Thus, we can conclude that the friction cone is an example of the Lorenz cone.



(a) (b) (c)

Figure 5.2 Examples of polyherda. The orange circle denotes the vertices of the polyhedra, while the green arrows the rays. (a) The $\mathcal{V}$-*rep* of a generic polyhedron is composed by vertices and rays. (b) A polyhedral cone is described by rays. (c) The $\mathcal{V}$-*rep* of a polytope consist in vertices only

Polyhedra

We define a *polyhedron*, denoted as $\mathcal{P}$, as the solution set of a finite number of linear inequalities and equalities such that:

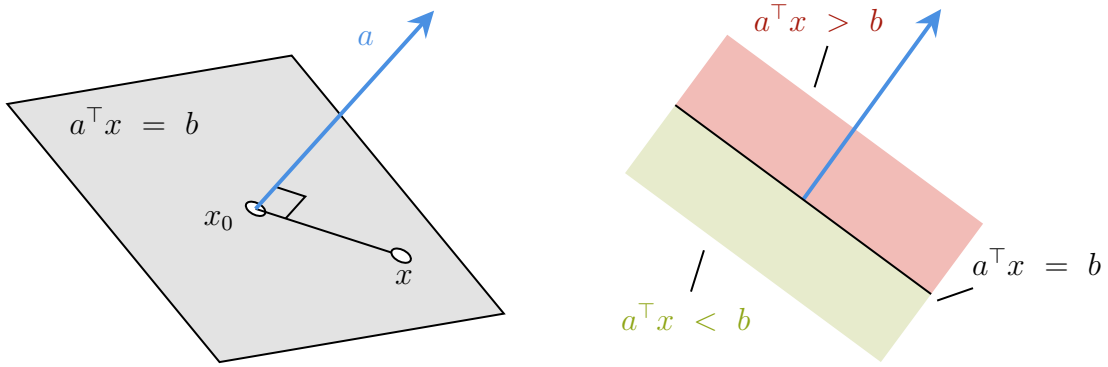
$$\mathcal{P} = \{x \mid a_i^\top x \leq b_i,\ i = 1, \dots, m,\ c_j^\top x = d_j\ j = 1, \dots, n\}. \tag{5.1.8}$$

Often the equalities $c_j^\top x = d_j$ are not considered in the polyhedron formulation. In fact, equality can always be replaced with two inequalities, for instance $c_j^\top x = d_j$ is equivalent to $c_j^\top x \leq d_j$ and $-c_j^\top x \leq -d_j$. Considering this, in the following we will remove the equality terms from the polyhedron definition. Thus we write (5.1.8) as

$$\mathcal{P} = \{x \mid a_i^\top x \leq b_i,\ i = 1, \dots, k\}, \tag{5.1.9}$$

where $k = m + 2n$. It is worth noting that a polyhedron can be seen as the intersection of finite halfspaces and hyperplanes. Hereafter, we call a bounded polyhedron *polytope*. Figure 5.2a illustrates an example of a polyhedron, while Figure 5.2c a polytope. Equation (5.1.9) is often written in a more compact from as

$$\mathcal{P} = \{x \mid Ax \preceq b\}, \tag{5.1.10}$$



(a) Geometric representation of a hyperplane. (b) Halfspaces generated by a hyperplane.

Figure 5.3 Geometric representation of a hyperplane and the associated halfspaces. (a) An hyperplane in $\mathbb{R}^3$. The plane is uniquelly determinated by a vector a normal to the plane and a point x_0, for ant point x in the plane different from x_0, $x - x_0$ is orthogonal to a. (b) The halfspace determinated by $a^\top x \geq b$ contains the vector a, while the halfspace described by $a^\top x \leq b$ extends in the direction $-a$

where $b = \begin{bmatrix} b_1 & \dots & b_m \end{bmatrix}^\top$. While A is given by

$$A = \begin{bmatrix} a_1^\top \\ \vdots \\ a_m^\top \end{bmatrix}. \tag{5.1.11}$$

In Equation (5.1.10), $\preceq$ denotes the *component-wise inequality* in $\mathbb{R}^m$, i.e., $x \preceq y$ if and only if $x_i \leq y_i$ for each $i = 1, \dots, m$. If the polyhedron $\mathcal{P}$ is described by a null vector b, the set is called *polyhedral cone* – Figure 5.2b. The linear approximation of a friction cone is an example of a polyhedral cone. Equation (5.1.10) is often denoted as *halfspace representation*, or shortly $\mathcal{H}$*-rep* of a polyhedron. Given a polyhedron, the halfspace representation is not unique. For a $\mathcal{H}$*-rep* polyhedron $\mathcal{P} = \{x \mid Ax \preceq b\}$ an i-th inequality is said to be *redundant for* $\mathcal{P}$ if its removal preserves the polyhedron [Bemporad et al., 2001]

$$\mathcal{P} = \{x \mid Ax \preceq b\} = \{x \mid A_j x \leq b_j,\ \forall j \neq i\}. \tag{5.1.12}$$

If none of the inequalities is redundant, then $\mathcal{H}$*-rep* is said to be *minimal halfspace representation*.

We can describe a polyhedron $\mathcal{P}$ in terms of points, denoted *vertices* and generating vectors, often named *rays*. This description is often called *vertex representation*, or

$\mathcal{V}$*-rep*. It is worth mentioning that if a polyhedron is described only by vertices it is a polytope, while if only rays are required, the polyhedron is a *polyhedral cone*. Formally, the *vertex representation* of a polyhedron $\mathcal{P}$ writes as

$$\mathcal{P} = \left\{ \theta_1 v_1 + \cdots + \theta_m v_m + \cdots + \theta_k v_k \;\middle|\; \sum_{i=1}^{m} \theta_i = 1,\ \theta_i \geq 0\ i = 1, \ldots, k \right\} \tag{5.1.13a}$$

$$= \mathbf{conv}\{v_1, \ldots, v_m\} \oplus \mathbf{conic}\{v_{m+1}, \ldots, v_k\} \tag{5.1.13b}$$

where $\oplus$ indicates the *Minkowski sum*.

Figure 5.2 illustrates the vertex representation in case of polyhedron 5.2a, cone 5.2b and polytope 5.2c.

Minkowski-Weyl theorem We now state, without proving, one of the most important results of the convex polyhedral theory, the *Minkowski-Weyl theorem*. Given a set $\mathcal{P} \subseteq \mathbb{R}^n$. Then the following are equivalent:

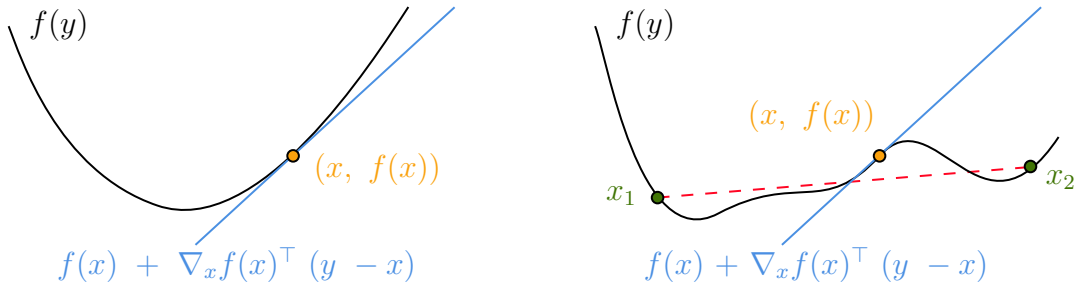
1. $\mathcal{H}$*-rep*: There exist a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^n$ such that $\mathcal{P} = \{x \mid Ax \preceq b\}$.

2. $\mathcal{V}$*-rep*: There exist two finite sets $V, R \subseteq \mathbb{R}^n$ such that $\mathcal{P} = \mathbf{conv}\, V \oplus \mathbf{conic}\, R$.

To give the reader a better understanding of the implications of this theorem, we recall some advantages of the $\mathcal{H}$*-rep* and $\mathcal{V}$*-rep* representations. For example, testing if a vector belongs to a polyhedron is trivial if $\mathcal{P}$ is expressed in the $\mathcal{H}$*-rep*, while it becomes more complex in the $\mathcal{V}$*-rep*. On the other hand, we can exploit the $\mathcal{V}$*-rep* when we want to compute the linear combination of a vector $x \in \mathcal{P}$. For instance, given $x \in \mathcal{P}$ and $y = Ax$, then y belongs to a polyhedron, $y \in \mathcal{P}_A$ if and only if $x \in \mathcal{P}$, if x is expressed with the $\mathcal{V}$*-rep* description, i.e $x = \theta_1^* v_1 + \cdots + \theta_m^* v_m + \cdots + \theta_k^* v_k = \mathcal{V}\Theta^*$, then y is equal to $y = A\mathcal{V}\Theta^* \in \mathcal{P}_A$. We name *vertex enumeration problem* the conversion from the $\mathcal{H}$*-rep* to the $\mathcal{V}$*-rep*. The dual transformation problem of a $\mathcal{V}$*-rep* to a minimal $\mathcal{H}$*-rep* is often called *facet enumeration problem* or *(convex) hull problem*. Both problems can be solved by applying the *double description method* [Fukuda and Prodon, 1995; Motzkin et al., 1953].

5.2 Convex function

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, we say that f is *convex* if



(a) Convex function. (b) Nonconvex function.

Figure 5.4 Examples of convex and nonconvex functions. (a) Plot of a convex function. $f(y)$ lies above the first order approximation of f at x. (b) Graph of a nonconvex function. The chord between x_1 and x_2 intersects the plot. Furthermore, the linear approximation of f intersects the graph.

1. the domain of f, denoted with $\mathbf{dom}(f)$ is a convex set;
2. for all $x_1, x_2 \in \mathbf{dom}(f)$ and $0 \leq \theta \leq 1$ we have

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2). \tag{5.2.1}$$

To give the reader a better understanding, we can imagine drawing a chord from any x_1 to y_1, if it lies above the graph of f, f is convex. A function f is *strictly convex*, if for $x_1 \neq x_2$ and $0 < \theta < 1$, we have $f(\theta x_1 + (1-\theta)x_2) < \theta f(x_1) + (1-\theta)f(x_2)$. If $-f$ is (strictly) convex, then f is said to be (strictly) concave. Figure 5.4 illustrates an example of convex and nonconvex functions.

5.2.1 First and second order conditions for the convexity

Let assume a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, and let us define $\nabla_x f(x)$ as the gradient of f

$$\nabla_x f(x)^\top = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n}, \end{bmatrix} \tag{5.2.2}$$

then the *first-order condition for the convexity* state that if f is convex if and only if $\mathbf{dom}(f)$ is convex and $f(y) \geq f(x) + \nabla_x f(x)^\top (y - x)$ for any x and y in the domain of f. Figure 5.4a illustrates the geometrical representation of the condition. Given a first-order Taylor approximation of the function f at x, then the function is convex only if its value is always greater than the approximation. It is worth noticing that if $\nabla_x f(x) = 0_{n\times 1}$ and f are convex, for the first-order condition we have that for any $y \in \mathbf{dom}(f)$ $f(y) \geq f(x)$, consequently x is a global minimizer of f. In the case where

f is strictly convex, it is possible to prove that $\nabla_x f(x) = 0_{n\times 1}$ implies that x is the only global minimizer of f.

Let us now assume that f is twice differentiable and given *Hessian* $\nabla_x^2 f(x)$ as

$$\nabla_x^2 f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1^2} & \frac{\partial f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2^2} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_2} & \cdots & \frac{\partial f}{\partial x_n^2} \end{bmatrix}. \tag{5.2.3}$$

Then f is convex if and only if the domain of f is convex and the Hessian is a positive semidefinite matrix, commonly denoted with $\nabla_x^2 f(x) \succeq 0$. This condition is often called *second-order condition for the convexity*. Similarly, f is concave if and only if the domain of f is convex and $\nabla_x^2 f(x) \preceq 0$, the Hessian is a negative semidefinite matrix. Here, it is important to recall that even if $\nabla_x^2 f(x) \succ 0$ implies that f is strictly convex, the converse does not hold.

Due to the second-order condition, we can easily check if a quadratic function is convex. The quadratic functions play an important role in the optimization, indeed as shown in the next chapters, the cost functions considered in this thesis are often quadratic. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, we say that f is quadratic if it can be expressed as:

$$f(x) = x^\top P x + q^\top x. \tag{5.2.4}$$

It is easy to prove that the Hessian of f is P while the gradient is q. If $P \succeq 0$, then f is said to be a convex quadratic function. As shown in Section 5.4, the minimization of a quadratic function leads to a huge class of optimization problems called Quadratic Programming (QP) problems.

5.3 Optimization problem

Given a set Z called *optimization problem domain*, and a set of *admissible variables* denoted with $S \subseteq Z$, such that the *decision variables* x belongs to S, i.e. $x \in S$, we define the *cost* function $f : Z \to \mathbb{R}$ such that each decision variable x has a given *cost*. We formulate a *mathematical optimization problem*, or just *optimization problem* as

$$\underset{x}{\text{minimize}} \quad f(x) \tag{5.3.1a}$$

$$\text{subj. to } x \in S \subseteq Z. \tag{5.3.1b}$$

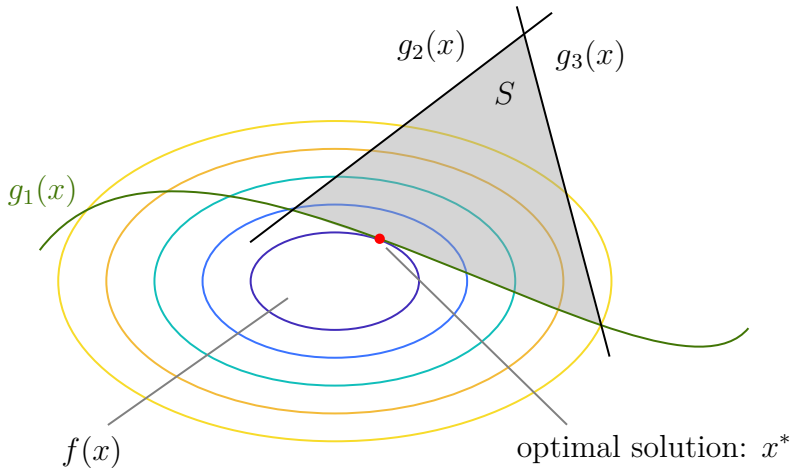


Figure 5.5 Graphic representation of an optimization problem. The ellipses represent the level sets of the cost function $f(x)$. $g_1(x)$, $g_2(x)$ and $g_3(x)$ are the inequalities constraints that define the feasible set S. x^* is the optimal solution of the problem. Since x^* belongs to the curve $g_1(x) = 0$ we can conclude that the first constraint is active at the optimal solution.

If the optimization problem (5.3.1) has a solution, we define the *primal optimal value* of the problem (5.3.1), denoted as f^* as the least possible cost as

$$f^* = \inf_{x \in S} f(x), \tag{5.3.2}$$

such that for all $x \in S$, $f(x) \geq f^*$. If $f^* \to -\infty$, we say that the problem is *unbounded below*. If $S = Z$ the problem is said to be *unconstrained*. If S is empty, the problem is *infeasible*.

The *optimal solution* x^* is defined as the decision variable whose cost is associated with the optimal value f^*, i.e., $x^* \in S$ with $f(x^*) = f^*$. If x^* exists, then we can rewrite Equation (5.3.2) as

$$f^* = \min_{x \in S} f(x) = f(x^*), \tag{5.3.3}$$

x^* is often called *global optimizer* or *optimal solution* of the optimization problem.

In this manuscript, we consider only optimization problems whose domain Z is a subset of the finite-dimensional Euclidean vector space $\mathbb{R}^n$. Consequently, Equation (5.3.1)

can be rewritten as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) && \text{(5.3.4a)} \\ \text{subj. to} \quad & g(x) \preceq 0_{m\times 1} && \text{(5.3.4b)} \\ & h(x) = 0_{p\times 1} && \text{(5.3.4c)} \\ & x \in Z, && \text{(5.3.4d)} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the cost function, $g : \mathbb{R}^n \to \mathbb{R}^m$ represents the inequality constraints and $h : \mathbb{R}^n \to \mathbb{R}^p$ the equalities. Z is given by the intersection of the domains of f, g and h.

Consider a feasible point $\bar{x}$. We say that the i-th inequality constraint $g_i(x) \leq 0$ is *active* if $g_i(\bar{x}) = 0$, on the other hand, if $g_i(\bar{z}) < 0$ the constraint is said to be *inactive*. It is worth noting that an equality constraint is always active for all feasible points. Similarly to what is discussed regarding the polyhedral in Section 5.1.2, a set of constraints is redundant if by removing one of the constraints, the feasible set S remains invariant.

Often the inequalities $h(x) = 0$ are not considered in the optimization problem formulation; indeed, an equality constraint can always be replaced by two inequalities; otherwise, another possibility is to parameterize the solution of the equality constraint. More details can be found in [Borrelli et al., 2017, Chapter 1.1.1].

5.3.1 The optimality conditions for unconstrained problems

Given an unconstrained problem of the form

$$\underset{x}{\text{minimize}} \quad f(x). \tag{5.3.5}$$

If $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, then it is possible to prove that if x^* is a local minimizer, then the gradient of f at x^* is equal to zero, i.e., $\nabla_x f(x^*) = 0_{n\times 1}$ and the Hessian matrix is positive semidefinite $\nabla_x^2 f(x^*) \succeq 0$. If the function f is convex, then x^* is a the global minimizer if and only if $\nabla_x f(x^*) = 0_{n\times 1}$.

This optimality condition plays an important role in solving the least square problems. Given $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^m$, the objective of a least square problem is to find the optimal solution of $Ax = b$ such that the square norm of $Ax - b$ is minimized.

The problem can be framed within the optimization framework with the aim of solving

$$\underset{x}{\text{minimize}} \quad (Ax-b)^\top (Ax-b). \tag{5.3.6}$$

Since the squared norm is a convex function, finding the minimum can be achieved by setting the gradient of $(Ax-b)^\top(Ax-b)$ to zero:

$$\nabla_x \left[(Ax-b)^\top (Ax-b)\right] = -2A^\top b + 2A^\top Ax = 0, \tag{5.3.7}$$

whose solution is given by $x^* = (A^\top A)^{-1}A^\top b$.

5.3.2 Lagrange duality theory

Lagrangian function Given an optimization problem of the form (5.3.4) we define the *Lagrangian function* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \lambda^\top g(x) - \mu^\top h(x), \tag{5.3.8}$$

where $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p$ are the *Langrange multipliers* or *dual variables*. In this context, the optimization problem (5.3.4) is often denoted as *primal optimization problem*, or shortly *primal problem* We recall that if $\bar{x}$ is a feasible point for the primal problem (5.3.4) and $\mu \succeq 0$, then $\mathcal{L}(\bar{x}, \lambda, \mu) \leq f(\bar{X})$.

Lagrange dual function Given the Lagrangian function $\mathcal{L}$, we define the *Lagrange dual function*, or simply *dual function*, as the unconstrained infimum of the Lagrangian over x, for fixed multipliers λ and μ and it writes as

$$g(\lambda, \mu) = \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda, \mu). \tag{5.3.9}$$

When the Lagrangian is unbounded below in x, the dual function takes the value $-\infty$, in this case, we say that the pair (λ, μ) is *dual infeasible*. In the other case, the pair (λ, μ) is *dual feasible*. We now state two important properties of the dual function

1. the dual function is concave even when the original problem is not convex [Boyd and Vandenberghe, 2004, Chapter 5.1.2];

2. the dual function $g(\lambda, \mu)$ is a lower bound on the optimal value p^* of the primal problem (5.3.4), indeed, for any μ and $\lambda \succeq 0$ we have

$$g(\lambda, \mu) \leq p^*. \tag{5.3.10}$$

When the pair (λ, μ) is dual feasible, the latest statement gives a non-trivial lower bound on p^*.

Lagrangian dual problem Equation (5.3.10) states that the Lagrangian dual function is a lower bound on the optimal problem p^*. Now we aim to find the greatest value of the upper bound. This problem is often called *Lagrangian dual problem* associated with the optimization problem (5.3.4), and is written as

$$\underset{\lambda,\mu}{\text{maximize}} \quad g(\lambda, \mu) \tag{5.3.11a}$$
$$\text{subj. to} \quad \lambda \succeq 0_{m\times 1}. \tag{5.3.11b}$$

The pair (λ, μ) is dual feasible if $\lambda \succeq 0_{m\times 1}$ and $g(\lambda, \mu) \geq -\infty$. We call *dual optimal* or *optimal Lagrange multipliers* and denoted with (λ^*, μ^*) the optimal solution of (5.3.11). We finally denoted with d^* the optimal value of the Lagrange dual problem (5.3.11) as

$$d^* = g(\lambda^*, \mu^*) = \inf_x \left(f(x) + {\lambda^*}^\top g(x) + {\mu^*}^\top h(x) \right). \tag{5.3.12}$$

Weak duality Given d^* and p^* the dual and primal optimal value, respectively, then we have

$$d^* \leq f^*. \tag{5.3.13}$$

This condition is called *weak duality*. The difference $f^* - d^*$ is called *optimal duality gap* or *duality gap*.

Strong duality If $d^* = f^*$, the duality gap is zero, and we can conclude that the *strong duality* holds. Strong duality does not hold in general, however, there exist conditions on the primal problem which imply the strong duality. Here, it is worth mentioning *Slater's condition* [Boyd and Vandenberghe, 2004, Chapter 5.2.7].

Certificate of optimiality Given the primal optimal problem (5.3.4) and its dual (5.3.11), the cost function f evaluated at any feasible point $\hat{x}$ is an upper bound

of the primal optimal value p^*, indeed we have $g(\lambda, \mu) \leq d^* \leq p^* \leq f(\hat{x})$. We say that $\hat{x}$ is ϵ-*suboptimal* with $\epsilon = f(\hat{x}) - g(\lambda, \mu)$. In this scenario, the pair (λ, μ) is also called a *certificate* since it proves the suboptimality of $\bar{x}$. Several optimization algorithms iteratively solve the primal and dual optimization problem until a given value of ϵ is reached.

Complementary slackness Let us consider the primal optimal problem (5.3.4) and its dual (5.3.11) and assume that the string duality holds. Let x^* the primal optimal value and (λ^*, μ^*) the dual optimal point. This means that

$$\begin{aligned} f(x^*) &= g(\lambda^*, \mu^*) && \text{(5.3.14a)} \\ &= \inf_x (f_0(x) + {\lambda^*}^\top g(x) + {\mu^*}^\top h(x) && \text{(5.3.14b)} \\ &\leq f(x^*) + {\lambda^*}^\top g(x^*) + {\mu^*}^\top h(x^*) && \text{(5.3.14c)} \\ &\leq f(x^*). && \text{(5.3.14d)} \end{aligned}$$

Equation (5.3.14a) consists of the strong duality property, while (5.3.14b) is the definition of the dual optimal function (5.3.12). If the inf of the optimal function is equal to the inequality (5.3.14c) is actually an equality if a feasible x^* exists. We can now conclude that

$$f(x^*) = f(x^*) + {\lambda^*}^\top g(x^*) + {\mu^*}^\top h(x^*). \tag{5.3.15}$$

Since x^* its a feasible value, $h(x^*) = 0$, as a consequence μ^* is different from zero. On the other hand, Equation (5.3.15), implies ${\lambda^*}^\top g(x^*) = 0$. This condition is known as *complementary slackness* and it holds for any primal optimal value x^* and any dual optimal pair (λ^*, μ^*). Given an inequality constraint $g_i(x^*)$ evaluated at the optimal value and its associated dual optimal variable λ_i^*, the complementary slackness is resumed in these two following implications

1. if $\lambda_i^* > 0$ then $g_i(x^*) = 0$;
2. if $g_i(x^*) < 0$ then $\lambda_i^* = 0$.

In other words, the i-th optimal Lagrange multiplier is zero unless the i-th constraint is active at the optimum, i.e., for $x = x^*$.

5.3.3 Karush-Kuhn-Tucker Conditions

Let us assume that the cost function f, the equality constraints h and the inequality constraints g functions are differentiable, Equation (5.3.12) is satisfied only if the gradient of the Lagrange function $\mathcal{L}(x, \lambda^*, \mu^*)$ must be zero at x^*:

$$\nabla_x f(x^*) + {\lambda^*}^\top \nabla_x g(x^*) + {\mu^*}^\top \nabla_x h(x^*) = 0. \tag{5.3.16}$$

We can summarize the condition (5.3.16) with the consideration done in the above set of inequalities and equalities

$$\nabla_x f(x^*) + {\lambda^*}^\top \nabla_x g(x^*) + {\mu^*}^\top \nabla_x h(x^*) = 0 \tag{5.3.17a}$$
$${\lambda^*}^\top g(x^*) = 0 \tag{5.3.17b}$$
$$\lambda^* \succeq 0_{m\times 1} \tag{5.3.17c}$$
$$g(x^*) \preceq 0_{m\times 1} \tag{5.3.17d}$$
$$h(x^*) = 0_{m\times 1}. \tag{5.3.17e}$$

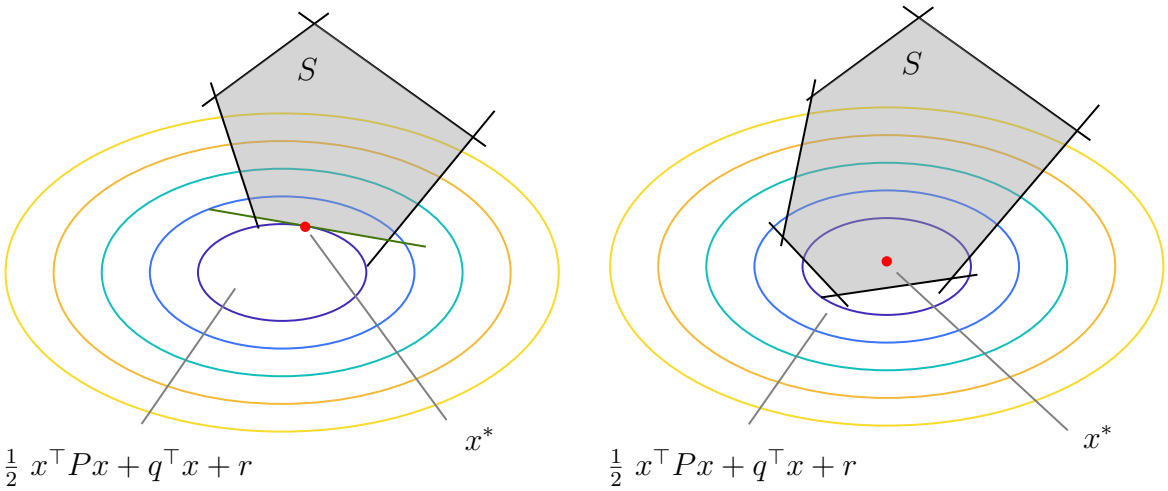
The conditions in (5.3.17) are called Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are necessary conditions for any primal-dual optimal pair if strong duality holds and the cost and constraints are differentiable. If the primal problem is also convex, then the KKT conditions are also sufficient.

Linear Independence Constraint Qualification

Consider an optimization problem of the form (5.3.4) and let $I_{ac}(\bar{x})$ the set of active constraints. Then, the *Linear Independence Constraint Qualification* (LICQ) is satisfied at $\bar{x}$ if the set of the active constraint gradients is linearly independent. We notice that several optimization algorithms solve the problem by relying on the linear independence constraint qualification (LICQ) [Betts, 2010]. Indeed, the LICQ allows for the characterization of the set of all possible directions required to solve a constraint nonlinear optimization problem.

5.4 Quadratic Programming

The optimization problem (5.3.4) is called a *quadratic program* (QP) if the objective is convex quadratic and the constrain functions are affine. We express a quadratic



(a) Optimizer on boundary of S.

(b) Optimizer in interior of S.

Figure 5.6 Geometric interpretation of the QP solution. (a) The solution belongs to the boundary of P. One of the constraint is active (green line). (b) The solution belongs to the interior of P. In this case, $x^* = -P^{-1}q$.

program problem in the following form:

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}x^\top Px + q^\top x + r \tag{5.4.1a}$$

$$\text{subj. to} \quad Gx \preceq h. \tag{5.4.1b}$$

$x \in \mathbb{R}^n$, $P \succ 0$ is a positive definite matrix, $q \in \mathbb{R}^n$, $G \in \mathbb{R}^{m\times n}$, $h \in \mathbb{R}^m$ and $r \in \mathbb{R}$. If the set $S = \{x | Gx \preceq h\}$ is not empty, the solution exists and is unique. Figure (5.6) illustrates the solution of a QP problem. In Figure (5.6a) the optimal solution belongs to the boundary of the feasible set S, while in Figure (5.6b) the solution belongs to the interior of S. In this case, the optimization problem (5.4.1) is equivalent to an unconstrained optimization problem. Since the cost function is quadratic and hence convex, the optimal solution is given by setting the gradient of $\frac{1}{2}x^\top Px + q^\top x + r$ to zero:

$$\nabla_x \left[\frac{1}{2}x^\top Px + q^\top x + r\right] = 0, \tag{5.4.2}$$

whose solution is given by $x^* = -P^{-1}q$.

Dual of QP Consider (5.4.1) the Lagrange function (5.3.8) is given by

$$\mathcal{L}(x, \lambda) = x^\top Px + q^\top x - \lambda^\top \left[Gx - h\right], \tag{5.4.3}$$

The dual cost (5.3.9) is obtained by minimizing the Lagrange function (5.4.3) with respect to x

$$g(\lambda) = \min_x \left\{ x^\top P x + q^\top x - \lambda^\top \left[Gx - h \right] \right\}, \tag{5.4.4}$$

For a given λ, the Lagrange function (5.4.3) is convex, consequently the dual cost (5.4.4) is obtained by setting the gradient of (5.4.3) equal to zero. We then have

$$x = -P^{-1} \left(q + G^\top \lambda \right). \tag{5.4.5}$$

Finally, the dual problem (5.3.11) is given by substituting (5.4.5) into (5.4.4) and computing the maximum for $\lambda \succeq 0_{m\times 1}$ as

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \frac{1}{2}\lambda^\top (GP^{-1}G^\top)\lambda + \lambda^\top (h - GP^{-1}q) + \frac{1}{2} q^\top H^{-1} q && \text{(5.4.6a)} \\ \text{subj. to} \quad & \lambda \succeq 0_{m\times 1}. && \text{(5.4.6b)} \end{aligned}$$

The dual of a QP problem is a QP problem itself.

KKT of QP Given a QP problem (5.4.3) the KKT conditions (5.3.17) become

$$\begin{aligned} Px + q + G^\top \lambda &= 0 && \text{(5.4.7a)} \\ {\lambda^*}^\top (Gx - h) &= 0 && \text{(5.4.7b)} \\ \lambda^* &\succeq 0_{m\times 1} && \text{(5.4.7c)} \\ Gx - h &\preceq 0_{m\times 1}. && \text{(5.4.7d)} \end{aligned}$$

5.5 Optimal control

Let us consider a dynamical system

$$\dot{x}(t) = f(x(t), u(t), t), \tag{5.5.1}$$

where $x(t) \in \mathbb{R}^n$ is the state of the dynamical system and $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. Our aim is to determine the evolution of the system given an initial value $x(0) = x_0$. Such a problem is often known as *Initial Value Problem*, denoted as IVP. If the terminal condition is also specified, $x(t_f) = x_f$, we name the problem *Boundary Value Problem* (BVT).

If the control input $u(t)$ is known and is sufficiently regular, the IVP has an unique solution that depends on the control history $u(t)$.

In more complex scenarios, the dynamical system (5.5.1) is extended with a set of algebraic constraints of the form

$$c(x(t), u(t), t) \leq 0_{n_c \times 1}. \tag{5.5.2}$$

The dynamical system (5.5.1) together with (5.5.2) define a *Differential Algebraic Equation* (DAE).

Let us now introduce a cost function whose purpose is to evaluate the performance index of a control input sequence u in a closed interval $t \in [t_0, t_f]$. We define a *performance functional index*, or simply *cost function*, denoted as $\mathcal{J}(x_0, u(.), t)$ as:

$$\mathcal{J}\left(x_0, u(.), t\right) = \beta(x(t_f), t_f) + \int_{t_0}^{t_f} \ell\left(x(\tau), u(\tau), \tau\right) \mathrm{d}\,\tau, \tag{5.5.3}$$

where $\beta(x(t_f), t_f)$ is called *Mayer* term and weighs the terminal state. While $\ell\left(x(\tau), u(\tau), \tau\right)$ is the *Lagrange* term. The Mayer term is also known as *terminal cost*, and weighs the state of the system at $t = t_f$. On the other hand, the Lagrange term, also known as *running cost*, weighs the path to reach the final state.

Combining the dynamical system (5.5.1), the algebraic constraint (5.5.2), and the cost function (5.5.3) we define the optimal control problem (OCP) as follows

$$\begin{aligned}
\underset{u \in \mathcal{U}}{\text{minimize}} \quad & \beta(x(t_f), t_f) + \int_{t_0}^{t_f} \ell\left(x(\tau), u(\tau), \tau\right) \mathrm{d}\,\tau && && \text{(5.5.4a)}\\
\text{subj. to} \quad & \dot{x}(t) = f(x(t), u(t), t) && t \in [t_0, t_f] && \text{(5.5.4b)}\\
& c(x(t), u(t), t) \leq 0_{n_c \times 1} && t \in [t_0, t_f] && \text{(5.5.4c)}\\
& x(t_0) = x_0. && && \text{(5.5.4d)}
\end{aligned}$$

The optimal control problem (5.5.4) is known as *Bolza* optimization problem. We recall that an OCP can be written in three forms, namely: *Bolza*, *Mayer* and *Lagrange* forms. They apparently differ in the formulation of the functional to be optimized, but it is worth noticing that they are equivalent and that it is possible to convert each problem into the other two forms. If the integral term in the cost function (5.5.4a) is set to zero, the problem is said to be written in *Mayer* form. On the other hand, if the terminal cost in (5.5.4a) is set to zero, the problem is in Lagrange form.

The OCP (5.5.4) is normally solved by applying three different strategies. The *dynamic programming* [Bellman, 1952; Tawiah and Song, 2021], the *indirect methods* [Bertolazzi et al., 2005; Liberzon, 2012; Pontriagin, 1962] or the *direct methods* [Betts, 2010].

Dynamic programming The dynamic programming (DP) methodology attempts to analytically find the optimal control strategy by breaking the OCP into smaller subproblems applying the *Bellman's principle of optimality* [Bellman, 1952; Dreyfus, 2002; Gross, 2016]:

> *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Since we do not exploit the *dynamic programming* techniques for the rest of the thesis, we avoid further discussing these methods.

Indirect methods The indirect methods are another class of analytical optimization methods. Unlike DP, they rely on the *Pontryagin's minimum principle* [Kirk, 1970, Section 5.3] to compute the optimality conditions. The methods exploit the Hamiltonian function introduced for the Hamilton-Jacobi-Bellman equation to reduce the solution of the OCP to the solution of a $2n$ equations given in the form of a two-point boundary value problem. Since we do not exploit *Indirect methods*for the rest of the thesis, we will not further discuss them.

Direct methods Finally, in the direct method approach, the OCP is discretized, *transcribed* into a nonlinear programming problem, and finally solved numerically.

In this thesis, we approach the OCP only by applying direct methods. Section 5.5.1 presents the common integration methods exploited to convert the continuous dynamics (5.5.1) into a discrete one of the form

$$x(t + \delta t) = \Gamma(x(t), u(t), t). \tag{5.5.5}$$

5.5.1 Direct methods

Given an optimization problem (5.5.4), our objective is to transcribe it into a nonlinear programming problem of the form (5.3.1). Even if the formulation in (5.5.4) seems to

be similar to (5.3.1), it is worth recalling some main differences. In an OCP, we seek a control strategy $u(t)$ where t belongs to a close interval. $u(t)$ is in general a function that may depend on the state of the system. On the other hand, the outcome of a nonlinear optimization problem (5.3.1) is just a sequence of control actions. Furthermore, the nonlinear optimization problem (5.3.1) is completely agnostic to the concept of time evolution of the dynamical system (5.5.1). These two main differences are overcome by discretizing the dynamics of the continuous system.

Discretization of the dynamical system

We call the solution of the dynamics equation (5.5.1), the *state transition function*

$$x(t) = \phi(x_0, t_0, u). \tag{5.5.6}$$

We want to transform it into discrete difference equations, suitable for numerical computing. We now introduce the notation x_k as the value of a variable x evaluated at $t_0 + k\,\mathrm{d}\,t$, i.e., $x_k = x(t_0 + k\,\mathrm{d}\,t)$. We approximate the state evolution of (5.5.1) with

$$x_{k+1} = \Gamma\left(x_k, u_k, k\right). \tag{5.5.7}$$

where $\Gamma : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{N} \to \mathbb{R}^n$. It is worth noting that, by recursively applying (5.5.7) from $k = 0$ to $k = N$, (5.5.7) is the discrete approximation of (5.5.6). $N = (t_f - t_0)/\,\mathrm{d}\,t$ is often denoted as *time horizon*.

Forward Euler The most common *single-step* method is the *forward Euler* integration scheme:

$$x_{k+1} = x_k + \mathrm{d}\,t f\left(x_k, u_k, t_k\right). \tag{5.5.8}$$

It is worth noting that, for sufficiently high $\mathrm{d}\,t$, the forward Euler method is subject to numerical instability.

Backward Euler Given a dynamical system (5.5.1) the *backward Euler* method gives the following approximation

$$x_{k+1} = x_k + \mathrm{d}\,t f\left(x_{k+1}, u_{k+1}, t_{k+1}\right). \tag{5.5.9}$$

The backward Euler is an *implicit single step* method. As a consequence, it is numerically stable independently from the chosen time step [Ascher et al., 1997]. Given

the presence of x_{k+1} in both sizes of (5.5.9) it would be necessary to numerically solve (5.5.9) to find an explicit expression of x_{k+1} as a function of x_k, u_k, and t_k.

Tustin integration method The *Tustin integration* also known *trapezoidal* method is another implicit method of the form:

$$x_{k+1} = x_k + \frac{\mathrm{d}\,t}{2}\left[f\left(k_k, u_k, t_k\right) + f\left(x_{k+1}, u_{k+1}, t_{k+1}\right)\right]. \tag{5.5.10}$$

Unlike Euler's methods, the trapezoidal method considers the dynamics at two time instants, k and $k+1$. For this reason, the trapezoidal method is considered a *multiple collocations* method. We finally recall that the trapezoidal method (5.5.10) gives a better approximation of the Euler methods (5.5.8) (5.5.9).

Zero order hold Let us now assume that the dynamical system (5.5.1) is described by linear time-invariant ordinary differential equations of the form

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{5.5.11}$$

Where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ and $x(t_0) = x_0$. The system (5.5.11) admits a closed form solution (5.5.6) of the form

$$x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)\,\mathrm{d}\,\tau \tag{5.5.12}$$

Let us now assume that the control input is kept constant in an interval $\mathrm{d}\,t$, i.e., $u(t) = \bar{u}$ for $t \in [t_0, t_0 + \mathrm{d}\,t]$. Then Equation (5.5.12) can be rewritten as

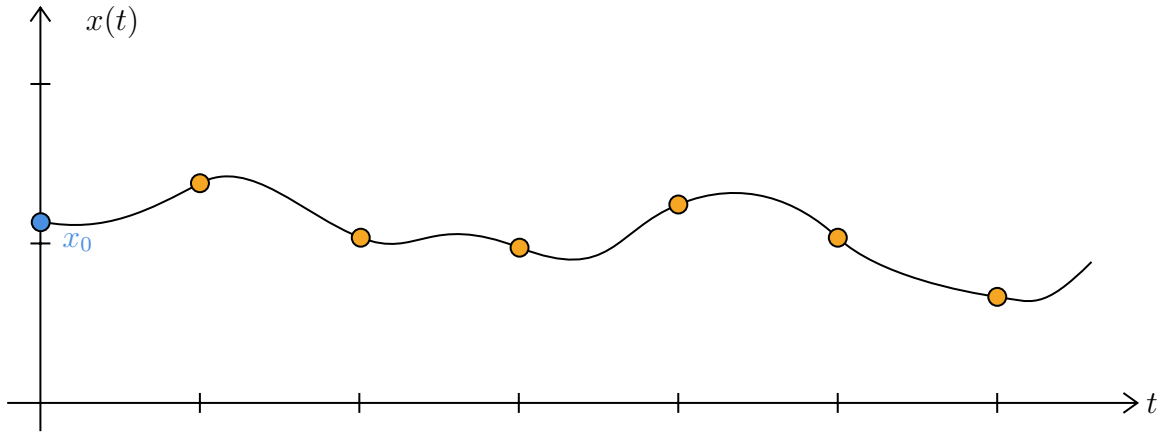
$$x(t_0 + \mathrm{d}\,t) = e^{A(\mathrm{d}\,t)}x_0 + \int_{0}^{\mathrm{d}\,t} e^{A\tau}\,\mathrm{d}\,\tau B\bar{u} \tag{5.5.13}$$

Assume that the control input is kept constant during every sampling period $\mathrm{d}\,t$, and that its value is equal to $u(t_0 + k\,\mathrm{d}\,t) = u_k$. Then Equation (5.5.13) can be rewritten as

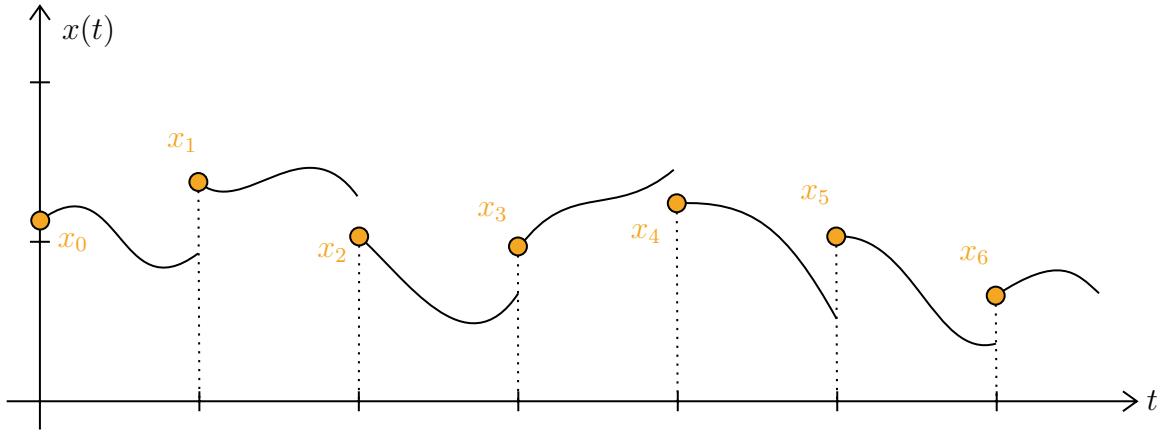
$$x_{k+1} = e^{A\,\mathrm{d}\,t}x_k + \int_{0}^{\mathrm{d}\,t} e^{A\tau}\,\mathrm{d}\,\tau B u_k \tag{5.5.14a}$$

$$= A_d x_k + B_d u_k. \tag{5.5.14b}$$

This method is known as the *zero order hold* integration method since the input is kept constant (i.e., it is held) during the sampling period.



(a) Single shooting.



(b) Multiple shooting.

Figure 5.7 Single and Multiple shooting.

The presented integration methods can be exploited recursively to determine the state evolution from t_0 to t_f. Consequently, it would be possible to convert the continuous dynamical system into a set of equality constraints that can be embedded in the optimization problem (5.3.1). This technique is often denoted as *shooting*.

5.5.2 Shooting methods

Single shooting

Let us consider the discretized dynamical system (5.5.7). We notice that a given state x_{k+1} can be written as a function of the initial state x_0 and the input sequence u_j

where $j \in [0, k]$. In fact,

$$x_1 = \Gamma(x_0, u_0), \tag{5.5.15a}$$
$$x_2 = \Gamma(x_1, u_1) = \Gamma\left(\Gamma(x_0, u_0), u_1\right), \tag{5.5.15b}$$
$$x_4 = \Gamma(x_2, u_2) = \Gamma\left(\Gamma\left(\Gamma(x_0, u_0), u_1\right), u_2\right), \tag{5.5.15c}$$
$$\vdots \tag{5.5.15d}$$
$$x_{k+1} = \Gamma(x_k, u_k) = \Gamma\left(\ldots\left(\Gamma(x_0, u_0), \ldots\right), u_k\right). \tag{5.5.15e}$$

Therefore, with a *single shooting* method, it is possible to obtain the terminal state starting from the initial one, without having to consider all the intermediate states. In an optimization framework, the set of optimization variables would consist only of the control inputs applied at each step.

Multiple shooting

If the system is nonlinear, the composition of function Γ N times may result in a very complex expression. In addition, it is difficult to specify *path* constraints, i.e., those involving intermediate states. These problems, typical of the method just introduced, are solved using a *multiple shooting* approach [Bock and Plitt, 1984; Diehl et al., 2006], where all the intermediate state variables are also optimization variables. This has the clear disadvantage of including more variables and constraints in the optimization problem. In fact, it is necessary to add a constraint for each pair of optimization variables of the form

$$x_1 - \Gamma(x_0, u_0) = 0 \tag{5.5.16a}$$
$$x_2 - \Gamma(x_1, u_2) = 0 \tag{5.5.16b}$$
$$x_3 - \Gamma(x_2, u_2) = 0 \tag{5.5.16c}$$
$$\vdots \tag{5.5.16d}$$
$$x_N - \Gamma(x_{N-1}, u_{N-1}) = 0. \tag{5.5.16e}$$

Where each constraint i depends only on the state i , $i + 1$ and the control input i. Seeing that the dynamics is discretized, the cost function (5.5.4a) becomes

$$\mathcal{J}\left(x_0, \ldots, x_N, u_0, \ldots, u_{N-1}\right) = \beta\left(x_N\right) + \sum_{i}^{N-1} \ell\left(x_i, u_i\right) \mathrm{d}\, t. \tag{5.5.17}$$

Following the same approach, the algebraic constraints (5.5.4c), become a list of constraints

$$c(x_0, u_0) \leq 0_{n_c \times 1} \tag{5.5.18a}$$
$$c(x_1, u_1) \leq 0_{n_c \times 1} \tag{5.5.18b}$$
$$c(x_2, u_2) \leq 0_{n_c \times 1} \tag{5.5.18c}$$
$$\vdots \tag{5.5.18d}$$
$$c(x_{N-1}, u_{N-1}) \leq 0_{n_c \times 1}. \tag{5.5.18e}$$

Substituting the discretized dynamics (5.5.16), the cost function (5.5.17) and algebraic constraints (5.5.18) into the Bolza problem (5.5.4), we obtain the final form of the optimal control problem solved with a direct multiple shooting method

$$\underset{x_0,\ldots,x_N,u_0,\ldots,u_{N-1}}{\text{minimize}} \quad \beta\left(x_N\right) + \sum_{k}^{N-1} \ell\left(x_k, u_k\right) \mathrm{d}\, t \tag{5.5.19a}$$
$$\text{subj. to} \quad x_{k+1} - \Gamma(x_k, u_k) = 0 \qquad k = 0 \ldots N-1 \tag{5.5.19b}$$
$$c(x_k, u_k) \leq 0_{n_c \times 1} \qquad k = 0 \ldots N-1 \tag{5.5.19c}$$
$$x_0 = x(t_0). \tag{5.5.19d}$$

5.6 Model predictive control

When solving the optimal control problem (5.5.4) using a direct method, the output is a sequence of control input u_i. Indeed, given x_0, we obtain u_k, with k from 0 to $N-1$. The application of all these control inputs would result in an *open-loop* control. Alternatively, we can apply u_0 only and discard all other control inputs. Let the system evolve to retrieve a new feedback. The time horizon is shifted by an amount equal to $\mathrm{d}\, t$ and the optimal control problem (5.5.19) is solved again with a different initial condition. This method of discarding the control values except the first and shifting the time horizon is called *receding horizon* principle [Mayne and Michalska, 1990; Shahriar et al., 2013]. It provides the basis for the so-called *model predictive control* (MPC) [Bemporad et al., 2002; García et al., 1989].

We identify with $x_{k|i}$ the state vector at time $k+i$ predicted at time i by starting from the current state $x_{0|i} = x(t_0 + i\, \mathrm{d}\, t)$. Similarly, $u_{k|i}$ is the optimal control strategy

that should be applied at $k+i$ and computed at time i. Given the optimal control sequence, denoted as $\mathcal{U}_i^* = \{u_{0|i}^*, u_{1|i}^*, \ldots, u_{N|i}^*\}$. We apply only the first element of $\mathcal{U}_i^*$ to the dynamical system (5.5.1)

$$u(t) = u_{0|i}^*. \tag{5.6.1}$$

Then the optimization problem is solved at time $i+1$ considering the new state $x_{0|i+1}$ – Figure 5.8. In the end, applying the receding horizon principle, we can formulate the optimal control problem (5.5.19) as follows:

$$\begin{aligned}
\underset{x_{0|i},\ldots,x_{N|i},u_{0|i},\ldots,u_{N-1|i}}{\text{minimize}} \quad & \beta\left(x_{N|i}\right) + \sum_{k}^{N-1} \ell\left(x_{k|i}, u_{k|i}\right) \mathrm{d}\, t && && \text{(5.6.2a)}\\
\text{subj. to} \quad & x_{k+1|i} - \Gamma(x_{k|i}, u_{k|i}) = 0 && k = 0 \ldots N-1 && \text{(5.6.2b)}\\
& c(x_{k|i}, u_{k|i}) \leq 0_{n_c \times 1} && k = 0 \ldots N-1 && \text{(5.6.2c)}\\
& x_{0|i} = x(t_0 + i\, \mathrm{d}\, t). && && \text{(5.6.2d)}
\end{aligned}$$

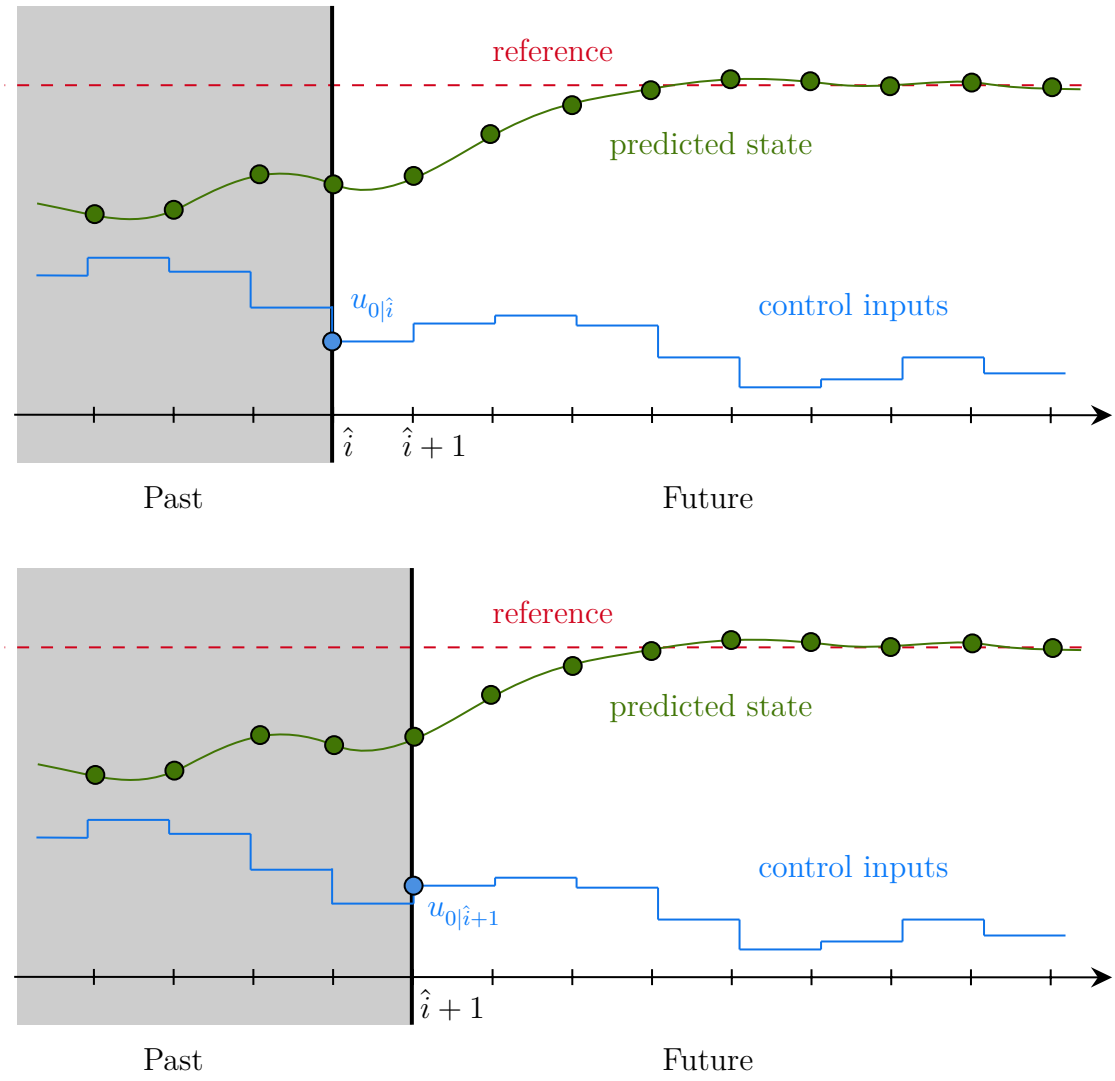


Figure 5.8 Receding horizon principle. At each sampling time, starting at the current state, an open-loop optimal control problem is solved over a finite horizon. The computed optimal manipulated input signal is applied to the process only during the following sampling interval $i, i+1$. At the next time step $i+1$, a new optimal control problem based on new measurements of the state is solved over a shifted horizon.

Chapter 6

State of the Art and Thesis Context

This chapter presents the bipedal locomotion state of the art, focusing on the definition of the different layers of the walking architecture. The chapter contains two main sections, and it is organized as follows. Section 6.1 contains the state of the art. Section 6.2 presents the context of the thesis contributions.

6.1 State of the Art

Despite the efforts of researchers in the field of humanoid robotics, bipedal locomotion remains an open problem. The complexity of the robot dynamics, the unpredictability of its surrounding environment, and the low efficiency of the robot actuation system are only few problems that complexify the achievement of robust robot locomotion. In the wide variety of robot controllers for bipedal locomotion, the Divergent-Component-of-Motion (DCM) is a ubiquitous concept used for generating walking patterns. Unlike quadrupeds [Poulakakis et al., 2005] or wheeled robots [Borst et al., 2009], maintaining the upright position is a complex task for a bipedal robot due to several factors, including, but not limited to the instability due to the bipedal posture, complexity in modeling, control in real time, and the actuation system to guarantee fast and efficient motions [Kaneko et al., 2011; Tsagarakis et al., 2017]. To guarantee stable bipedal locomotion, feedback control plays an important role, where common approaches are based on simplified models to achieve fast feedback.

During the DARPA Robotics Challenge, a common approach to humanoid robot control consisted of defining a hierarchical architecture composed of several layers [Feng

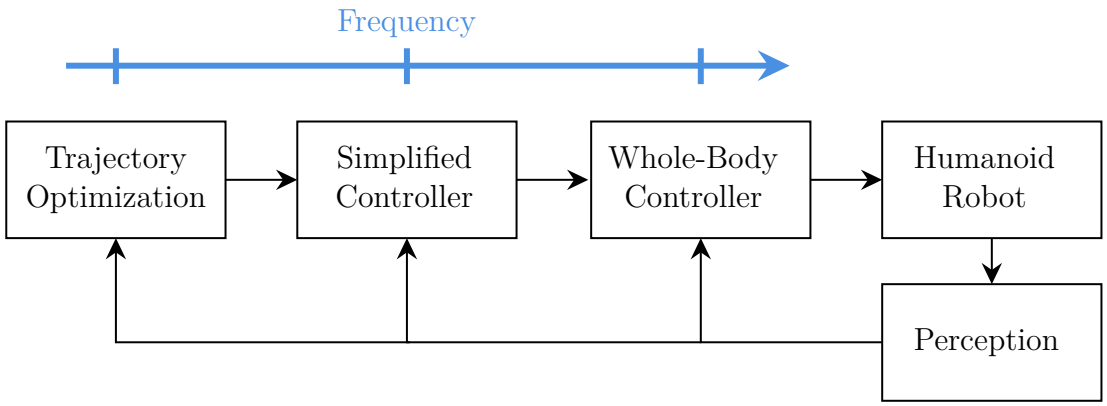


Figure 6.1 The three layer controller architecture. Inner the loop, the higher the frequency. Each layer gathers the outcome of the outer layer, the information from the robot through the perception block and generates the references for the inner layer.

et al., 2015a]. Each layer generates references for the layer below by processing inputs from the robot, the environment, and the output of the previous layer. The inner layer, the shorter the time horizon used to evaluate the output. In addition, lower layers usually employ more complex models to evaluate output, but a shorter time horizon often results in faster computations to obtain these outputs. From top to bottom, these layers are here called: *trajectory optimization*, *simplified model control*, and *whole-body quadratic programming (QP) control* – Figure 6.1. When an external disturbance acts on the robot, it may be necessary to consider the robot's state up to the trajectory optimization layer, recomputing the contact location to avoid the robot from falling.

6.1.1 Trajectory optimization layer

The *trajectory optimization* layer aims to compute a sequence of contacts' location and timings while considering the robot state. This layer often exploits optimization techniques to consider the feasibility of the contact location. To do so, both kinematic and dynamical robot models can be considered [Dai et al., 2014; Herzog et al., 2015]. More generally, when planning locomotion trajectories, the choice of the robot description plays a crucial role. In fact, the simpler the model, the simpler the problem. However, on the other hand, too simple models prevent the robot from performing highly dynamic motions. For instance, flat terrain allows one to model the robot as a simple unicycle [Flavigne et al., 2010; Morin and Samson, 2008] which enables fast solutions to the optimization problem for the walking pattern generation [Dafarra et al., 2018].

When an external disturbance acts on the robot, the trajectory optimization layer should be in charge of recompiling the contact location to avoid the robot from falling. In recent years, several attempts have been made in this direction. The new contact locations can be optimized assuming a constant step duration [Feng et al., 2016; Shafiee-Ashtiani et al., 2017b; Stephens and Atkeson, 2010b] or adaptive step timing [Griffin et al., 2017; Khadiv et al., 2016; Scianca et al., 2020; Shafiee et al., 2019]. Approximating the robot with a simplified model allows for solving the footsteps adjustment problem online. However, simplified model-based controller architectures often treat the contact adjustment strategy separately from the main control loop [Griffin et al., 2016; Mesesan et al., 2021; Shafiee et al., 2019] or apply heuristic-based strategies [Di Carlo et al., 2018]. Furthermore, approaches based on simplified models require hand-crafted models for the task at hand [Englsberger et al., 2015a; Kajita et al., 2001; Poulakakis and Grizzle, 2009], thus making the transition between different tasks, i.e., from locomotion to running, often very complex.

At the planning level, several attempts have been made to consider the centroidal dynamics [Orin et al., 2013] and robot kinematics to plan the desired motion trajectories [Dafarra et al., 2020, 2022; Dai et al., 2014; Fernbach et al., 2018; Herzog et al., 2015]. These methods can be split into five categories depending on the amount of a priori information on the contact location and timings they require. From the more general (and also computationally greedy) to the more contact information dependent, we have: *Complementary-free methods*, *Mixed-integer methods*, *predefined contact sequence*, *predefined contact location and timings* and *Offline library-based methods*.

Complementary-free methods These methods consider the definition of contacts explicitly within the planner. Hence, contact location, timing, and sequence are decided directly by the planner, allowing to generate complex motions [Dafarra et al., 2020, 2022; Dai et al., 2014]. With this approach, no prior knowledge is injected into the system to generate walking trajectories (i.e., the definition of the contact is explicitly considered inside the planner), but the whole-body motions result from a particular choice of the cost function. Although providing enhanced planning capabilities, given its complexity, a whole-body planner may take several minutes to compute a feasible solution. This prevents their usage online.

Mixed-integer methods These approaches attempt to reduce the computational burden of the *complementary-free methods* by using an integer variable to determine

where a contact should be established [Deits and Tedrake, 2015; Mason et al., 2018] and in which instant [Aceituno-Cabezas et al., 2017; Ibanez et al., 2014]. These approaches generally require mixed integer programming tools [Axehill and Hansson, 2006; Gurobi Optimization, 2022; Stellato et al., 2018b]. Mixed-integer programming methods provide enhanced modeling capabilities. However, the exploitation of integer variables strongly affects the computational performances, especially in case several contacts can be established.

Predefined contact sequence A common approach to reducing the computational demand is to assume a predefined contact sequence [Caron and Pham, 2017; Carpentier et al., 2016; Winkler et al., 2018] while keeping the contact location and timings as the output of the planner. For instance, for a bipedal robot, we can assume that a contact with the right foot will be followed by another one with the left foot. Even if such a choice simplifies the planning problem, the computational effort still prevents the use of the planner in a closed-loop controller.

Predefined contact location and timings Taking into account a predefined contact location and timings, the planning problem can finally be solved online. Since the contact sequence is predefined, this kind of planner generally aims to generate only the centroidal quantities [Orin et al., 2013] and [Traversaro, 2017, Section 3.9.3]. To keep the problem treatable online, the non-linear non-convex angular momentum dynamics (see Section 3.4) is often neglected. For instance, Caron et al. [2016] consider only the CoM dynamics to generate feasible locomotion patterns. Ponton et al. [2016] propose a convex relaxation of the angular momentum dynamics that allowed to efficiently compute momentum trajectories and contact forces while considering the angular momentum minimization. This relaxation is then extended to include an explicit target momentum in the cost function [Ponton et al., 2018].

Differential Dynamic Programming (DDP) techniques have also been exploited to generate feasible whole-body trajectories given a set of predefined contact locations and timings. Budhiraja et al. [2018] design a whole-body trajectory planner based on Differential Dynamic Programming (DDP). The proposed method produces locomotion motions by exploiting the centroidal angular momentum. Recently, [Dantec et al., 2021] has taken advantage of the DDP framework to design a whole-body model predictive control with state feedback on a torque-controlled humanoid robot.

These methods need to rely on external contact planners. However, in some applications, where multiple contacts can be established in several regions, this approach may be the most viable solution. Furthermore, since the optimization problem can be treated online, the contact location and timings can be adjusted to avoid the robot from falling in case of unstructured interaction with the environment.

Offline library-based methods Finally, another common approach to reduce the computational complexity is to use an offline constructed gait library [Guo et al., 2021; Nguyen et al., 2020]. This approach simplifies the problem and allows the planner to run online. However, the offline gait can generate only a finite set of motions, making the task of adding a new behavior to the library of motions more complex.

6.1.2 Simplified model control layer

The *simplified model control* layer is responsible for finding feasible robot center-of-mass (CoM) trajectories. However, the computational burden of finding feasibility regions usually calls for simplified models to characterize the robot dynamics. Indeed, assuming a constant height of the center of mass while walking and a constant angular momentum, it is possible to design a simple control strategy based on the well-known Linear Inverted Pendulum Model (LIPM) [Kajita et al., 2001] – See Section 4.1. Taking into account the LIPM dynamics, several authors propose the idea of decomposing the CoM dynamics into a stable and an unstable component [Englsberger et al., 2011; Hof, 2008; Koolen et al., 2012; Pratt et al., 2006, 2012; Takenaka et al., 2009]. Hof [2008] denotes the unstable part of the dynamics as *Extrapolated Center of Mass*. While, authors of [Pratt et al., 2012] and [Koolen et al., 2012] name it *(instantaneous) Capture Point* (ICP). The term *Divergent Component of Motion* (DCM) was finally coined by Takenaka et al. [2009]. Initially presented for a 2D scenario, the DCM has been extended in the 3D case, too [Englsberger et al., 2013, 2015a]. By definition, the DCM model assumes a constant natural frequency of the LIPM.

The LIPM and the DCM models have a widespread diffusion in both position-based control robots [Kamioka et al., 2018; Leng et al., 2020; Ramuzat et al., 2021; Shafiee-Ashtiani et al., 2017b] and torque-controlled robots [Dafarra et al., 2016; Englsberger et al., 2018a,b; Griffin and Leonessa, 2016; Hopkins et al., 2014; Koolen et al., 2012; Pratt et al., 2012; Stephens and Atkeson, 2010b]. Both the LIPM and the DCM are linear models; their linearity is based on the assumption of constant CoM height and constant centroidal angular momentum. The model time constant depends on

the desired CoM height. Englsberger et al. [2013] shows that the linearity of the DCM model can be preserved even in the case of varying the CoM height. However, assuming a constant natural frequency, the desired DCM reference can deviate from the time-invariant LIPM dynamics when the vertical CoM varies. Hopkins et al. [2014] attempt at loosening this assumption by extending the DCM to consider a time-varying natural frequency. By varying the natural frequency of the DCM, the authors of [Hopkins et al., 2014] can achieve generic CoM height trajectories during stepping. By relaxing the hypothesis of constant CoM height, the LIPM dynamics can be extended to a *variable-height inverted pendulum* (VHIP) model [Koolen et al., 2016]. The author of [Caron, 2020] proposes a simplified model controller based on the linear feedback of the VHIP. This approach is based on a dynamical extension of the DCM. Here, the VHIP time parameter is considered in the DCM state.

The simplified models have become very popular thanks to the combination with the Zero Moment Point (ZMP) as a contact feasibility criterion [Vukobratovic and Juricic, 1969] in the case of complanar contact scenarios and indefinitely high friction between the feet and the environment. Caron et al. [2017] generalize the ZMP contact stability in the case of multiple non-complanar contacts while considering frictional constraints.

By exploiting these simplified models, several instantaneous controllers [Englsberger et al., 2018a,b, 2015a; Hopkins et al., 2014] and on-line model predictive controller (MPC) [Bombile and Billard, 2017; Dafarra et al., 2018; Diedam et al., 2008; Griffin and Leonessa, 2016; Naveau et al., 2017; Wieber, 2006] have been designed. Englsberger et al. [2015a] propose a simple proportional controller to stabilize the unstable linear DCM dynamics. Similarly Hopkins et al. [2014] present a proportional-integral controller that guarantees the tracking of the desired time-varying DCM trajectory. On the other hand, MPC frameworks often provide references for the footstep locations [Diedam et al., 2008; Feng et al., 2016; Joe and Oh, 2018; Shafiee-Ashtiani et al., 2017b] and timings [Griffin et al., 2017; Khadiv et al., 2016] in the form of small adjustments with respect to desired values.

Finally, using the LIPM model, it is also possible to derive MPC schemes that are guaranteed to produce *stable* CoM trajectories [Scianca et al., 2016, 2020]

6.1.3 Whole-Body control layer

The *whole-body control* layer generates robot positions, velocities, or torques depending on the available control modes of the underlying robot. These outputs aim at stabilizing

the references generated by the previous layers. It considers the whole-body kinematic or dynamical models and very often instantaneous optimization techniques: no MPC methods are here employed. Furthermore, the associated optimization problem is often framed as a hierarchical stack-of-tasks, with strict or weighted hierarchies [Nava et al., 2016; Stephens and Atkeson, 2010b]. The optimization problem depends linearly on the decision variables. As a consequence, the whole-body control layer often implements a quadratic programming problem (see Section 5.4). The authors of [Herzog et al., 2016] solve the whole-body control problem by designing hierarchical inverse dynamics that consider the centroidal momentum of the robot. The authors transcribe the problem in a cascade of Quadratic Programming problems.

In recent years, due to the increased development of humanoid robots exposing the torque control interface [Englsberger et al., 2015b; Kaneko et al., 2019; Stasse et al., 2017], the scientific community has been interested in the possibility of using torque control-based algorithms to perform locomotion tasks [Englsberger et al., 2018b; Feng et al., 2015b; Koolen et al., 2012; Kuindersma et al., 2016; Lee et al., 2016; Ramuzat et al., 2022, 2021; Stephens and Atkeson, 2010a]. Indeed, torque-controlled robots have several advantages over position or velocity-controlled ones. A torque-controlled robot is, in fact, intrinsically compliant in the case of unexpected external interactions [Fahmi et al., 2019; Henze et al., 2016; Mesesan et al., 2019], and thus it can be used to perform cooperative tasks with humans [Romano et al., 2018; Sheridan, 2016; Tirupachuri et al., 2020]. In the whole-body QP control layer, the interaction between the environment and the robot is often modeled using the *rigid contact* assumption [Herzog et al., 2016; Hopkins et al., 2015; Nava et al., 2016]. Under this hypothesis, the controller can instantly change the contact forces to guarantee the tracking of desired quantities. If the robot interacts on a *visco-elastic* surface, the *rigid contact* assumption no longer holds, and the whole-body QP controller cannot arbitrarily change the contact forces.

At the modeling level, when a robot interacts with a *visco-elastic* surface, it is pivotal to design models that characterize the interaction properties, such as compliance and damping. Remark that the term *contact* describes situations where two bodies come in touch with each other at specific locations [Gilardi and Sharf, 2002]. Then, contact models can be classified into two main categories: *rigid* and *compliant*. When the contact is rigid, the bodies' mechanical structure does not change substantially, and the velocities of the system are often subject to discontinuities [Whittaker and McCrae, 1988]. Although they enable instantaneous feedback controllers for a large variety of robots [Englsberger et al., 2018b], rigid contact models may lead to poorly reproducible

simulation results in the presence of static frictions and multi-body systems [Mason and Wang, 1988; Stronge, 1991]. Therefore, the use of *compliant* contact models is a valid alternative to solve the limitations due to the rigid contact formulation.

The contact between a rigid body and a compliant environment can be defined as a linear or a non-linear function. The *Kelvin-Voigt* model is a linear model that describes the contact with a purely viscous damper and a purely elastic spring connected in parallel [Hajikarimi and Moghadas Nejad, 2021]. The *Maxwell* model assumes a purely viscous damper and a purely elastic spring connected in series [Hajikarimi and Moghadas Nejad, 2021]. Both the *Kelvin-Voigt* and the *Maxwell* models are characterized by only two parameters that depend on the mechanical properties of contact surfaces.

On the other hand, the literature on the non-linear modeling of compliant contacts is extensive. Originally, compliant contact models considered the contact between two spheres or between a sphere and a flat plate [Falcon et al., 1998; Hunt and Crossley, 1975; Lankarani and Nikravesh, 1990; Marhefka and Orin, 1999]. For instance, Hunt and Crossley [1975] characterize the ground as a non-linear spring-damper pair whose intrinsic parameters are chosen to satisfy Hertz's theory [Johnson, 1985, Chapter 4]. This model has become well known in the robotics community and is often denoted as the *Hunt-Crossley* model. Lankarani and Nikravesh [1990] extends the *Hunt-Crossley* model to consider the impact within multi-body systems.

On the other hand, the authors of [Azad and Featherstone, 2014] slightly modify the *Hunt-Crossley* model to consider the coefficient of restitution between spheres and plates of various materials.

At the control level, when the contact between a robot and its surrounding environment is *sufficiently* stiff, controllers based on the *rigid contact* assumption may still lead to *acceptable* robot performances.

In this case, a possibility is to design contact-model-free passivity-based control strategies [Henze et al., 2016; Mesesan et al., 2019]. When contact compliance impairs robot performance, it is pivotal to design contact models that take into account contact stiffness and damping, which can then be incorporated into feedback controllers.

Pure stiffness (no damping) linear lumped models of the soft contact can, for instance, be considered to design whole-body controllers in the presence of visco-elastic contact surfaces [Catalano et al., 2020; Flayols et al., 2020; Raibert and Craig, 1981].

Damping components can also be added to the contact model [Azad et al., 2016; Chiaverini et al., 1994; Fahmi et al., 2020], but the main assumption remains that the robot makes contact with the environment at isolated points, not on surfaces.

Still using lumped parameters, another approach to modeling soft contact surfaces consists in assuming that the contact interaction can be characterized by equivalent springs and rotational dampers, which can then be employed for robot control [Li et al., 2019; Sygulla and Rixen, 2020]. This approach leads to the problem of giving a physical meaning to the contact parameters associated with the torque in the contact wrench. Furthermore, springs and rotational dampers are inherently ill-posed: they make use of the three-angle SO(3) minimal representation to model foot rotations (e.g. the roll-pitch-yaw convention). At the planning level, the finite element method (FEM) can also be used to model the static equilibrium of a body in contact with a soft environment [Bouyarmane and Kheddar, 2011]. While providing enhanced modeling capabilities, FEM methods demand heavy computational time, which usually forbids their use in time-critical feedback control applications.

6.2 Thesis Context

Taking into account the three-layer controller architecture of locomotion – Figure 6.1, a natural question arises about the choice of the specific implementation of each block. Indeed, we humbly believe that different tasks can be accomplished while keeping a cascade control structure and changing the models considered in the specific layer.

Having in mind the research question, we now detail the context of the thesis contributions. Considering the three-layer controller architecture – Figure 6.1 we split the contributions into two parts. Part II is related to the whole-body-control layer. While Part III concerns the trajectory optimization and the simplified model control layer. The first chapter of each part presents a benchmarking of the associated state-of-the-art algorithms. The remaining chapters attempt to loosen some of the hypotheses and limitations of the state-of-the-art controllers.

6.2.1 Part II: Whole-Body Controllers

This part presents the design of three whole-body controllers for humanoid robot locomotion. The content of each chapter follows.

Chapter 7: Benchmarking of Whole-Body Controllers for Locomotion on Rigid Environment

In this chapter we present a comparison of whole-body controllers for locomotion on rigid contact. Assuming the three-layer architecture in Figure 6.1, we propose a kinematics-based and a dynamics-based whole-body controller. The former considers the kinematics of the robot to generate the desired joint positions/velocities. The latter is based on the entire robot dynamics and its output is the desired joint torques. Thanks to the modularity of the two control problems, it is possible to exchange the two implementations depending on the low-level controller currently available on the robot, namely *position*, *velocity* or *torque* controller.

We compare the two whole-body controllers in the three-layer architecture. In particular, the *trajectory optimization* layer is kept fixed with a unicycle-based planner that generates the desired DCM and foot trajectories, and the *simplified model control* layer, instead, implements an instantaneous controller for the DCM tracking. The several combinations of the control architecture are tested on the iCub humanoid robot v2.7 – see Section 1.1.1. While discussing the results, we underline the strengths and weaknesses of the two approaches.

Chapter 8: Whole-Body Controller on Visco Elastic Environment

This chapter contributes towards the modeling of compliant contacts for robot motion control. More precisely, the main contributions follow. *i*) A new contact model that characterizes the stiffness and damping properties of a visco-elastic material that exerts forces and torques on rigid surfaces in contact with it. Differently from classical state-of-the-art models used for robot control [Fahmi et al., 2020; Flayols et al., 2020; Li et al., 2019], the presented model considers the environment as a continuum of spring-damper systems, thus allowing us to compute the equivalent contact force and torque by computing surface integrals over the contact surface. As a consequence, we avoid using linear rotational springs and dampers to describe the interaction between robot and environment. We show that the model we propose extends and encompasses these linear models. *ii*) A whole-body controller that allows humanoid robots to walk on visco-elastic floors, which are modeled using the proposed compliant contact model. The robot controller estimates the model parameters on-line, so there is no prior knowledge of the contact parameters. *iii*) A validation of the approach on the simulated torque-controlled humanoid robot iCub v2.7 (Section 1.1.1), with an extensive

comparison between the proposed methods and classical state of the art techniques. Furthermore, we analyze the robustness capability of the presented controller with respect to non-parametric uncertainty in the contact model. Finally, we present the contact parameter estimation performance in the case of an anisotropic environment.

Chapter 9: Whole-Body Control of Humanoid Robots with Link Flexibility

In this chapter, we propose an extension of the state-of-the-art whole-body torque controller in the case of the robot affected by inner link flexibility. More precisely, the main contributions are threefold. *i*) A whole-body controller that allows humanoid robots affected by undesired link deflection to walk on rigid floors. Similarly to the Authors of [Villa et al., 2022] we model the link flexibility with undeactuated passive joints. Our controller implicitly considers the joint deformation and, differently from [Villa et al., 2022], our design does not perform any local compensation for the deflections. *ii*) An observer whose objective is to estimate the state of the flexible joint, specifically the position, the velocity and the torque, using only the measured contact force and the actuated joint state *iii*) The approach was validated on the simulated torque-controlled humanoid robot TALOS (Section 1.2), with extensive comparisons between the proposed methods and classical state-of-the-art techniques.

6.2.2 Part III: From Simplified to Reduced Models Controllers

This part discusses the design of the trajectories and the simplified model control layers. We also try to move from the simplified to reduced models to compute the desired trajectories for the whole-body control layer.

Chapter 10: Benchmarking of Simplified-Model Controllers for Locomotion

This chapter presents and compares several DCM based implementations of the kinematic-based three-layer controller architecture 6.1 In particular, the *trajectory optimization* layer is kept fixed with a unicycle-based planner that generates the desired DCM and foot trajectories. The *simplified model control* layer, instead, implements two controllers for the DCM tracking: an *instantaneous* and a *MPC controller*. In the same layer, we also present a controller that ensures the tracking of the CoM and the ZMP, which exploits 6-axes Force Torque sensors (F/T). Finally, the *whole-body QP control* ensures the tracking of the desired CoM and feet trajectories.

We test and compare the two implementations of the simplified model control layer on the iCub humanoid robot v2.7 – see Section 1.1.1. We show that one of the proposed implementations allows the iCub robot to achieve a walking velocity of 0.3372 meters per second, which is the highest walking velocity ever achieved by the iCub robot.

Chapter 11: Non-Linear Centroidal Model Predictive Controller

This chapter presents the design of a non-linear Model Predictive Controller (MPC) that aims at generating online feasible contact locations and wrenches for humanoid robot locomotion. More precisely, different from classical control architectures based on simplified models, the presented approach considers the reduced centroidal dynamics model while keeping the problem still treatable online. By modeling the system using a reduced model instead of a simplified one, we achieve highly dynamic robot motions to be performed online. The contact location adjustment is considered in the centroidal dynamics stabilization problem; thus, it is not required to design an ad-hoc block for this feature. Furthermore, unlike existing work [Jeong et al., 2019; Shafiee-Ashtiani et al., 2017b], the controller we propose automatically considers double support phases; thus the contact adaptation feature is continuously active during both single and double support phases. On the other hand, considering only the reduced centroidal dynamics model keeps the problem at a low complexity and allows us to test the controller in a close-loop architecture. This, except in a few cases [Dantec et al., 2021, 2022], is almost impossible if the complete robot model is taken into account [Dafarra et al., 2016; Fernbach et al., 2018; Herzog et al., 2015].

We validate the proposed control strategy on a simulation of one-leg and two-leg systems performing jumping and running tasks, respectively. Results show that, differently from the simplified model controllers, the proposed centroidal MPC generalizes on the number of contacts and the adjustment is automatically performed by the controller in the case of external disturbance acting on the system. Furthermore, we embed the MPC controller into the three-layer controller architecture (Figure 6.1) as a reduced model control layer. The entire approach is also validated on the position-controlled Humanoid Robot iCub v3 – see Section 1.1.2. We show that the proposed strategy prevents the robot from falling while walking and being pushed with external forces up to 40 Newton lasting 1 second when applied to the robot's arm.

Part II

Whole-Body Controllers

Chapter 7

Benchmarking of Whole-Body Controllers for Locomotion on Rigid Environment

In Part I we introduced the background and the literature review of the thesis. Instead, this chapter presents the first contribution of the manuscript. Here, we present and compare several whole-body controllers for bipedal locomotion in a rigid environment. In particular, we specify the three-layer controller architecture presented in Figure 6.1, as shown in Figure 7.1. The *trajectory optimization* layer is kept fixed with a unicycle-based planner [Dafarra et al., 2018] that generates the desired DCM and foot trajectories. The *simplified model control layer*, instead, implements two types of controllers for the tracking of the DCM: an instantaneous and an MPC one. The content of the *trajectory optimization* and *simplified model control layer* is detailed in Chapter 10. Finally, the *whole-body QP control* ensures the tracking of the desired CoM and feet trajectories by considering the complete robot models. In this context, we first present a kinematics-based whole-body controller, and then we extend the framework to consider the robot dynamics. Thanks to the modularity of the two problems, it is possible to exchange the two implementations depending on the low-level control interfaces available on the robot. The several combinations of the control architecture along with the simplified model controllers presented in Chapter 10 are compared on the iCub humanoid robot v2.7 – see Section 1.1.1.

The chapter is organized as follows. Section 7.1 presents the kinematics-based whole-body QP control layer. Section 7.2 details the dynamics-based whole-body controller. Section 7.3 presents the experimental validation of the proposed approach

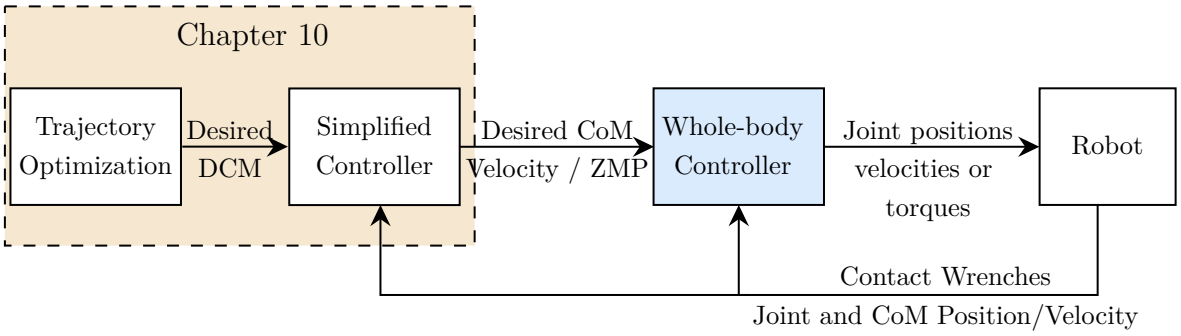


Figure 7.1 The control architecture is composed of three layers: the *trajectory optimization*, the *simplified model control*, and the *whole-body control*. The middle and the other layers are described in Chapter 10.

and shows an explanatory table comparing the different control approaches. Finally, Section 7.4 concludes the chapter.

The content of this chapter appears partially in:

> Romualdi, G., Dafarra, S., Hu, Y., and Pucci, D. (2018). A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE
>
> Romualdi, G., Dafarra, S., Hu, Y., Ramadoss, P., Chavez, F. J. A., Traversaro, S., and Pucci, D. (2020). A Benchmarking of DCM-Based Architectures for Position, Velocity and Torque-Controlled Humanoid Robots. *International Journal of Humanoid Robotics*, 17(01):1950034
>
> Video `https://www.youtube.com/watch?v=FIqwAO71Fc4`
> GitHub `robotology/walking-controllers`

7.1 Kinematics based whole-body QP control layer

The goal of the kinematics-based whole-body QP control layer is to ensure the tracking of a set of kinematic quantities considering the robot's kinematics. The proposed controller computes the desired robot generalized mixed representation velocity ${}^{B[\mathcal{I}]}\boldsymbol{\nu}$ (3.2.18), where $B[\mathcal{I}] = (o_B, [\mathcal{I}])$ is a frame placed on the robot base and oriented as the inertial

frame $\mathcal{I}$. We formulate the control problem using the stack of tasks approach. We achieve the control objective by framing the controller as a constrained optimization problem where the low priority tasks are embedded in the cost function, while the high priority tasks are treated as constraints. A similar approach has also been presented in [Kanoun et al., 2011; Khudher and Powell, 2016; Rapetti et al., 2020], it is important to recall that other strategies are available to solve the kinematics control problem, in this context it is worth mentioning [Buss and Kim, 2005; Goldenberg et al., 1985; Sciavicco and Siciliano, 1988].

In the next section, we present the set of low and high priority tasks. For the sake of clarity, we denote the generalized mixed robot velocity ${}^{B[\mathcal{I}]}\nu$ as ν.

7.1.1 Low and high priority tasks

What follows presents the tasks required to evaluate the desired generalized robot velocity, ν. We denote by Ψ the equality task and by Φ the inequality task.

Centroidal momentum task

The centroidal momentum, denoted with ${}_{\bar{G}}h \in \mathbb{R}^6$, is the aggregate linear and angular momentum of each robot link referred to the center of mass (CoM) of the robot ${}_{\bar{G}}h^\top = \begin{bmatrix} {}_{\bar{G}}h^{p^\top} & {}_{\bar{G}}h^{\omega^\top} \end{bmatrix}$ – Section 3.4. It is worth recalling that the centroidal momentum can be factorized as follows (3.4.4)

$$ {}_{\bar{G}}h = J_{\mathrm{CMM}}\nu, \tag{7.1.1} $$

where J_{CMM} is the centroidal momentum matrix [Orin and Goswami, 2008; Orin et al., 2013]. In order to set a desired centroidal momentum trajectory, we specify the following task:

$$ \Psi_h = {}_{\bar{G}}h^* - J_{\mathrm{CMM}}\nu, \tag{7.1.2} $$

where the desired linear centroidal momentum is often chosen to guarantee the tracking of the desired center of mass trajectory $x^{\mathrm{ref}}_{\mathrm{CoM}}(t)$

$$ {}_{\bar{G}}h^{p^*} = m\left[\dot{x}^{\mathrm{ref}}_{\mathrm{CoM}} + K_{\mathrm{CoM}}(x^{\mathrm{ref}}_{\mathrm{CoM}} - x_{\mathrm{CoM}})\right]. \tag{7.1.3} $$

Here K_{CoM} is a positive matrix. The desired angular momentum ${}_{\bar{G}}h^{\omega^*}$ is often set equal to zero, i.e., ${}_{\bar{G}}h^{\omega^*} = 0_{3\times 1}$.

The centroidal momentum task is often split into the linear and angular centroidal momentum tasks. The linear centroidal momentum task, denoted with Ψ_{CoM}, is:

$$\Psi_{\text{CoM}} = \begin{bmatrix} I_3 & 0_{3\times 3} \end{bmatrix} \Psi_h \tag{7.1.4a}$$
$$= {}_{\bar{G}}h^{p^*} - \begin{bmatrix} I_3 & 0_{3\times 3} \end{bmatrix} J_{\text{CMM}}\nu \tag{7.1.4b}$$
$$= {}_{\bar{G}}h^{p^*} - J_{\text{CMM}_p}\nu \tag{7.1.4c}$$

where ${}_{\bar{G}}h^{p^*}$ is chosen as (7.1.3) and J_{CMM_p} is the matrix composed of the first three rows of J_{CMM}. Similarly, we introduce the angular momentum task:

$$\Psi_{h^\omega} = \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} \Psi_h \tag{7.1.5a}$$
$$= {}_{\bar{G}}h^{\omega^*} - \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} J_{\text{CMM}}\nu \tag{7.1.5b}$$
$$= {}_{\bar{G}}h^{\omega^*} - J_{\text{CMM}_\omega}\nu, \tag{7.1.5c}$$

where J_{CMM_ω} is the matrix composed of the last three rows of J_{CMM}.

Cartesian task

While walking, we often require some of the robot link frames to have a specific position and orientation with respect to the inertial frame. To accomplish this task, we recall that given a frame L, its velocity expressed in mixed representation, denoted as ${}^{L[\mathcal{I}]}\text{v}_{\mathcal{I},L}$, is given by

$${}^{L[\mathcal{I}]}\text{v}_{\mathcal{I},L} = J_L\nu \tag{7.1.6}$$

where J_L is the mixed velocity Jacobian of the link L (3.2.17). To ask for a desired Cartesian trajectory ${}^{\mathcal{I}}H_L^{\text{ref}} = (p_L^{\text{ref}}, {}^{\mathcal{I}}R_L^{\text{ref}}) \in \mathbb{R}^3 \times \text{SO}(3)$, we specify the following task

$$\Psi_{L_{\text{SE}(3)}} = {}^{L[\mathcal{I}]}\text{v}^*_{\mathcal{I},L} - J_L\nu \tag{7.1.7}$$

where ${}^{L[\mathcal{I}]}\text{v}^*_{\mathcal{I},L} = \begin{bmatrix} \dot{p}_L^{*\top} & {}^{\mathcal{I}}\omega_{\mathcal{I},L}^{*\top} \end{bmatrix}^\top$ is chosen as

$$\dot{p}_L^* = \dot{p}_L^{\text{ref}} + K_{L_p}(p_L^{\text{ref}} - p_L) \tag{7.1.8a}$$
$${}^{\mathcal{I}}\omega^*_{\mathcal{I},L} = {}^{\mathcal{I}}\omega^{\text{ref}}_{\mathcal{I},L} + K_{L_\omega}\,\text{Log}\left({}^{\mathcal{I}}R_L^{\text{ref}}\ {}^{\mathcal{I}}R_L^\top\right). \tag{7.1.8b}$$

Here K_{L_p} and K_{L_ω} are two positive matrices. By such a particular choice of the desired velocity (7.1.8b), it is possible to guarantee almost-global stability and convergence of ${}^{\mathcal{I}}R_L$ to ${}^{\mathcal{I}}R_L^{\text{ref}}$ [Olfati-Saber, 2001].

Starting from the definition of the SE(3) task (7.1.7), we introduce the positional and rotational tasks for the frame L, respectively, denoted as $\Psi_{L_{\mathbb{R}^3}}$ and $\Psi_{L_{\mathrm{SO}(3)}}$. The positional task $\Psi_{L_{\mathbb{R}^3}}$ is just a projection of the SE(3) task $\Psi_{L_{\mathrm{SE}(3)}}$ as:

$$\begin{aligned}\Psi_{L_{\mathbb{R}^3}} &= \begin{bmatrix} I_3 & 0_{3\times 3}\end{bmatrix} \Psi_{L_{\mathrm{SE}(3)}} && (7.1.9a)\\ &= \dot{p}^*_L - \begin{bmatrix} I_3 & 0_{3\times 3}\end{bmatrix} J_L \nu && (7.1.9b)\\ &= \dot{p}^*_L - J_{L_p}\nu, && (7.1.9c)\end{aligned}$$

where $\dot{p}^*_L$ is set as (7.1.8a) and J_{L_p} is the matrix composed of the first three rows of J_L. Similarly, we define the SO(3) task as follows:

$$\begin{aligned}\Psi_{L_{\mathrm{SO}(3)}} &= \begin{bmatrix} 0_{3\times 3} & I_3\end{bmatrix} \Psi_{L_{\mathrm{SE}(3)}} && (7.1.10a)\\ &= {}^{\mathcal{I}}\omega^*_{\mathcal{I},L} - \begin{bmatrix} 0_{3\times 3} & I_3\end{bmatrix} J_L \nu && (7.1.10b)\\ &= {}^{\mathcal{I}}\omega^*_{\mathcal{I},L} - J_{L_\omega}\nu. && (7.1.10c)\end{aligned}$$

${}^{\mathcal{I}}\omega^*_{\mathcal{I},L}$ is set as (7.1.8b) and J_{L_ω} is the matrix composed of the last three rows of J_L.

Joint regularization task

In order to prevent the controller from providing solutions with huge joint variations, we introduce a regularization task for the joint variables. The task is achieved by asking for a desired joint velocity that depends on the error between the desired and measured joint values, such as:

$$\Psi_s = \dot{s}^* - \begin{bmatrix} 0_{n\times 6} & I_n\end{bmatrix} \nu, \tag{7.1.11}$$

where n is equal to the robot actuated degrees of freedom and $\dot{s}^*$ is set equal to

$$\dot{s}^* = \dot{s}^{\mathrm{ref}} + K_s(s^{\mathrm{ref}} - s). \tag{7.1.12}$$

Here s^{ref} is the desired joint configuration. K_s is a positive define diagonal matrix.

Joint limits task

The feasibility of the joint position s is generally guaranteed by means of a set of inequalities of the form

$$s^- \leq s \leq s^+, \tag{7.1.13}$$

where s^- and s^+ are respectively the lower and upper joints position limits. In the case of the kinematics-based whole-body controller, the joint values s cannot be arbitrarily chosen. To overcome this issue, we substitute the joint position s in (7.1.13) with its discrete dynamics computed at time using the forward Euler method $t_0 + k\,\mathrm{d}\,t$, i.e., $s(t_0 + (k+1)\,\mathrm{d}\,t) = s_{k+1} = s_k + \mathrm{d}\,t\dot{s}_k$:

$$s^- \leq s_k + \mathrm{d}\,t\dot{s}_k \leq s^+. \tag{7.1.14}$$

Rearranging (7.1.14) we obtain the final formulation of the joint limits task

$$\Phi_s : \quad \frac{s^- - s}{\mathrm{d}\,t} \leq \dot{s} \leq \frac{s^+ - s}{\mathrm{d}\,t}. \tag{7.1.15}$$

where we remove the time dependency k.

7.1.2 Quadratic programming problem

The control objective is achieved by transcribing the control problem as a constrained optimization problem considering the tasks presented in Section 7.1.1. We want to underline that given an equality task Ψ, it is always possible to consider it as a low or high priority task, or, in another word, as a term of the cost function or as an equality constraint. Consequently, this makes the kinematics based whole-body problem modular[1]. A different choice of the considered tasks and priorities allows one to obtain completely different robot behaviors. Given the above observation, we decide to present the specific implementation we consider in the experimental results – Section 7.3.

The tracking of the left and right foot poses are considered as high priority SE(3) tasks (7.1.7) and are denoted respectively as $\Psi_{L_{\mathrm{SE(3)}}}$ and $\Psi_{R_{\mathrm{SE(3)}}}$. We take into account the CoM tracking as a high priority task (7.1.4). The torso orientation is considered as a low priority task SO(3) task (7.1.10) and we denote it with $\Psi_{T_{\mathrm{SO(3)}}}$. Furthermore, the joint postural condition (7.1.11) is also added as a low priority task. Finally, we ask for a joint velocity $\dot{s}$ such that the inequality joint limits task (7.1.14) is satisfied.

[1]The `QPInverseKinematics` class implemented in our framework exploits this modularity to customize the control problem: `https://github.com/ami-iit/bipedal-locomotion-framework/tree/v0.6.0/src/IK`

The above hierarchical control objectives can be cast into a whole-body optimization problem:

$$\begin{aligned}
\underset{\nu}{\text{minimize}} \quad & \Psi_{T_{\mathrm{SO(3)}}}^{\top} \Lambda_T \Psi_{T_{\mathrm{SO(3)}}} + \Psi_s^{\top} \Lambda_s \Psi_s && \text{(7.1.16a)} \\
\text{subj. to} \quad & \Psi_{L_{\mathrm{SE(3)}}} = 0 && \text{(7.1.16b)} \\
& \Psi_{R_{\mathrm{SE(3)}}} = 0 && \text{(7.1.16c)} \\
& \Psi_{\mathrm{CoM}} = 0 && \text{(7.1.16d)} \\
& \Phi_s. && \text{(7.1.16e)}
\end{aligned}$$

Since the decision variable is the robot velocity ν and the tasks depend linearly on ν, we transcribe the optimization problem (7.1.16) into a quadratic programming problem (Section 5.4) of the form:

$$\begin{aligned}
\underset{\nu}{\text{minimize}} \quad & \nu^{\top} H \nu + 2 g^{\top} \nu && \text{(7.1.17a)} \\
\text{subj. to} \quad & A\nu \preceq b && \text{(7.1.17b)}
\end{aligned}$$

The Hessian matrix H and the gradient vector g are evaluated from (7.1.16a). The constraint matrix and vector A and b are obtained from (7.1.16b), (7.1.16c), (7.1.16d) and (7.1.16e). Using this formulation, the optimization problem can be solved using a standard numerical QP solver.

7.1.3 Position and velocity controlled robot

It is important to notice that the outcome of (7.1.16) is (also) the robot joint velocity. When a robot velocity controller is available, one can set these joint velocities to the low-level robot controller. In this case, the * quantities in the tasks (Section 7.1.1) can be evaluated using the feedback from the robot sensors, and the robot is said to be *velocity controlled*. On the other hand, if the robot velocity control is not available, one may integrate the outcome of (7.1.16) to obtain desired joint position to be set to a low-level robot position controller. In this case, the * quantities in (7.1.16) can be evaluated using the desired integrated quantities instead of the sensor feedback, and (7.1.16) behaves as an inverse kinematics module. Consequently, the robot is said to be *position controlled*.

7.2 Dynamics-based whole-body QP control layer

The Dynamics-based whole-body QP control layer considers the dynamic model of the system to ensure the tracking of the desired trajectories. The proposed controller computes the desired robot joint torque τ, the generalized mixed representation acceleration ${}^{B[\mathcal{I}]}\dot{\nu}$, and a set of desired spatial contact forces expressed in the mixed representation ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$. Here $B[\mathcal{I}] = (o_B, [\mathcal{I}])$ is a frame placed on the base of the robot and oriented as the inertial frame $\mathcal{I}$. On the other hand, $\mathcal{C}_j[\mathcal{I}] = (p_j, \mathcal{I})$ is the frame associated with the admissible contact (e.g. the foot) where $j \in \mathbb{N}$ such that $1 \leq j \leq n_c$, with n_c is the number of admissible contacts. The control problem is formulated using the stack of tasks approach. Similar to the kinematics based whole-body control layer (Section 7.1), the objective is achieved by transcribing the problem as a constrained optimization problem. We define the low priority tasks as terms of the cost function. While the high priority tasks are treated as constraints.

An approach similar to the one presented in this section has been also described in [Dean-Leon et al., 2019; Del Prete et al., 2016; Englsberger et al., 2018b; Henze et al., 2016; Mesesan et al., 2019; Nava et al., 2016; Ramuzat et al., 2022]

In the next section, we present the set of low and high priority tasks. For the sake of clarity, we simplify the notation of the generalized mixed robot velocity and acceleration by dropping the superscript $B[\mathcal{I}]$. Furthermore the contact wrenches are stacked into a vector $\mathrm{f} = \begin{bmatrix} {}_{\mathcal{C}_1[\mathcal{I}]}\mathrm{f}_1^\top & {}_{\mathcal{C}_2[\mathcal{I}]}\mathrm{f}_2^\top & \dots & {}_{\mathcal{C}_{n_c}[\mathcal{I}]}\mathrm{f}_{n_c}^\top \end{bmatrix}^\top \in \mathbb{R}^{6n_c}$

7.2.1 Low and high priority tasks

What follows presents the tasks required to define the control objectives. We denote by Ψ the equality task and by Φ the inequality task.

Centroidal momentum task

Given a frame $\bar{G} = (x_{\mathrm{CoM}}, [\mathcal{I}])$, the centroidal momentum rate of change ${}_{\bar{G}}\dot{h}$ balances the external spatial force applied on the robot (3.4.6):

$$ {}_{\bar{G}}\dot{h} = \sum_{j=1}^{n_c} \begin{bmatrix} I_3 & 0_{3\times3} \\ (p_j - x_{\mathrm{CoM}}) \times & I_3 \end{bmatrix} {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j + m\bar{g}, \tag{7.2.1} $$

where $\bar{g} = \begin{bmatrix} 0 & 0 & -g & 0 & 0 & 0 \end{bmatrix}^\top$ is the 6D gravity acceleration vector. Rearranging Equation (7.2.1), we obtain

$$
\begin{aligned}
{}_{\bar{G}}\dot{h} &= \begin{bmatrix} I_3 & 0_{3\times3} & \dots & I_3 & 0_{3\times3} \\ (p_1 - x_{\mathrm{CoM}})\times & I_3 & \dots & (p_{n_c} - x_{\mathrm{CoM}})\times & I_3 \end{bmatrix} \mathrm{f} + m\bar{g} && \text{(7.2.2a)} \\
&= A_c \mathrm{f} + m\bar{g}. && \text{(7.2.2b)}
\end{aligned}
$$

To ask for a desired centroidal momentum trajectory, we specify the following task:

$$
\Phi_h = {}_{\bar{G}}\dot{h}^* - A_c \mathrm{f} - m\bar{g}. \tag{7.2.3}
$$

We set the desired linear centroidal momentum rate of change to guarantee the tracking of the desired center of mass trajectory $x^{\mathrm{ref}}_{\mathrm{CoM}}(t)$ as

$$
{}_{\bar{G}}\dot{h}^{p^*} = m\left[\ddot{x}^{\mathrm{ref}}_{\mathrm{CoM}} + K^d_{\mathrm{CoM}}(\dot{x}^{\mathrm{ref}}_{\mathrm{CoM}} - \dot{x}_{\mathrm{CoM}}) + K^p_{\mathrm{CoM}}(x^{\mathrm{ref}}_{\mathrm{CoM}} - x_{\mathrm{CoM}})\right]. \tag{7.2.4}
$$

Here K^p_{CoM} and K^d_{CoM} are positive define diagonal matrices. On the other hand, the desired angular momentum rate of change ${}_{\bar{G}}\dot{h}^{\omega^*}$ is given by

$$
{}_{\bar{G}}\dot{h}^{\omega^*} = {}_{\bar{G}}\dot{h}^{\omega^{\mathrm{ref}}} + K_{h^\omega}\left({}_{\bar{G}}h^{\omega^{\mathrm{ref}}} - {}_{\bar{G}}h^{\omega}\right), \tag{7.2.5}
$$

where $K_{h^\omega} > 0$. If a high-level whole-body planner is available, for instance [Carpentier et al., 2016], $h^{\omega^{\mathrm{ref}}}$ is chosen to satisfy (3.4.4)

$$
{}_{\bar{G}}h^{\omega^{\mathrm{ref}}} = J^{\mathrm{pl}}_{\mathrm{CMM}_\omega}\nu^{\mathrm{pl}}, \tag{7.2.6}
$$

where the superscript pl indicates that the quantity is computed by the high-level planner. Another common choice is to set the desired angular momentum equal to zero, i.e., $h^{\omega^{\mathrm{ref}}} = 0_{3\times1}$

Similar to the kinematics-based controller case, the centroidal momentum task is often divided into linear and angular centroidal momentum tasks. The linear task, denoted by Ψ_{CoM} is given by

$$
\begin{aligned}
\Psi_{\mathrm{CoM}} &= \begin{bmatrix} I_3 & 0_{3\times3} \end{bmatrix} \Psi_h && \text{(7.2.7a)} \\
&= {}_{\bar{G}}\dot{h}^{p^*} - \begin{bmatrix} I_3 & 0_{3\times3} \end{bmatrix} A_c \mathrm{f} - mg && \text{(7.2.7b)} \\
&= {}_{\bar{G}}\dot{h}^{p^*} - A_{c_p}\mathrm{f} - mg && \text{(7.2.7c)}
\end{aligned}
$$

where ${}_{\bar{G}}\dot{h}^{p^*}$ is chosen as (7.2.4) and A_{c_p} is the matrix composed of the first three rows of A_c. The angular momentum task Ψ_{h^ω} is

$$\begin{aligned}\Psi_{h^\omega} &= \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} \Psi_h && \text{(7.2.8a)}\\ &= {}_{\bar{G}}\dot{h}^{\omega^*} - \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} A_c \mathrm{f} && \text{(7.2.8b)}\\ &= {}_{\bar{G}}\dot{h}^{\omega^*} - A_{c_\omega} \mathrm{f}. && \text{(7.2.8c)}\end{aligned}$$

${}_{\bar{G}}\dot{h}^{\omega^*}$ is set equal to (7.2.5). A_{c_ω} is the matrix composed of the last three rows of A_c.

Cartesian task

While walking, we often require some of the robot link frames to have a specific position and orientation with respect to the inertial frame. To accomplish this task, we recall that given a frame L, its acceleration expressed in mixed representation, denoted as ${}^{L[\mathcal{I}]}\dot{\mathrm{v}}_{\mathcal{I},L}$, is given by

$${}^{L[\mathcal{I}]}\dot{\mathrm{v}}_{\mathcal{I},L} = J_L \dot{\nu} + \dot{J}_L \nu, \tag{7.2.9}$$

where J_L is the mixed velocity Jacobian of the link L (3.2.17) and $\dot{J}_L$ its derivative. To follow a desired Cartesian trajectory ${}^{\mathcal{I}}H_L^{\mathrm{ref}} = (p_L^{\mathrm{ref}}, {}^{\mathcal{I}}R_L^{\mathrm{ref}}) \in \mathbb{R}^3 \times \mathrm{SO}(3)$, we specify the following task:

$$\Psi_{L_{\mathrm{SE}(3)}} = {}^{L[\mathcal{I}]}\dot{\mathrm{v}}^*_{\mathcal{I},L} - J_L \dot{\nu} - \dot{J}_L \nu, \tag{7.2.10}$$

where ${}^{L[\mathcal{I}]}\dot{\mathrm{v}}^*_{\mathcal{I},L} = \begin{bmatrix} \ddot{p}_L^{*\top} & {}^{\mathcal{I}}\dot{\omega}_{\mathcal{I},L}^{*\top} \end{bmatrix}^\top$ is chosen as [2]

$$\begin{aligned}\ddot{p}_L^* &= \ddot{p}_L^{\mathrm{ref}} + K_{L_p}^d(\dot{p}_L^{\mathrm{ref}} - \dot{p}_L) + K_{L_p}^p(p_L^{\mathrm{ref}} - p_L) && \text{(7.2.11a)}\\ {}^{\mathcal{I}}\dot{\omega}^*_{\mathcal{I},L} &= {}^{\mathcal{I}}\dot{\omega}^{\mathrm{ref}}_{\mathcal{I},L} + K_{L_\omega}^d({}^{\mathcal{I}}\omega^{\mathrm{ref}}_{\mathcal{I},L} - {}^{\mathcal{I}}\omega_{\mathcal{I},L}) + K_{L_\omega}^p \operatorname{Log}\left({}^{\mathcal{I}}R_L^{\mathrm{ref}}\ {}^{\mathcal{I}}R_L^\top\right). && \text{(7.2.11b)}\end{aligned}$$

Here $K_{L_p}^p$, $K_{L_p}^d$, $K_{L_\omega}^p$ and $K_{L_\omega}^d$ are positive defined matrices.

Starting from the definition of the SE(3) task (7.2.10), we introduce the positional and rotational tasks for the frame L, respectively, denoted as $\Psi_{L_{\mathbb{R}^3}}$ and $\Psi_{L_{\mathrm{SO}(3)}}$:

$$\begin{aligned}\Psi_{L_{\mathbb{R}^3}} &= \begin{bmatrix} I_3 & 0_{3\times 3} \end{bmatrix} \Psi_{L_{\mathrm{SE}(3)}} && \text{(7.2.12a)}\\ &= \ddot{p}_L^* - \begin{bmatrix} I_3 & 0_{3\times 3} \end{bmatrix} \left(J_L \dot{\nu} + \dot{J}_L \nu\right) && \text{(7.2.12b)}\\ &= \ddot{p}_L^* - J_{L_p} \dot{\nu} - \dot{J}_{L_p} \nu, && \text{(7.2.12c)}\end{aligned}$$

[2] An efficient implementation of the controller is available at `https://github.com/ami-iit/lie-group-controllers`.

$$\Psi_{L_{\mathrm{SO(3)}}} = \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} \Psi_{L_{\mathrm{SE(3)}}} \tag{7.2.13a}$$

$$= {}^{\mathcal{I}}\dot{\omega}^*_{\mathcal{I},L} - \begin{bmatrix} 0_{3\times 3} & I_3 \end{bmatrix} \left(J_L \dot{\nu} + \dot{J}_L \nu \right) \tag{7.2.13b}$$

$$= {}^{\mathcal{I}}\dot{\omega}^*_{\mathcal{I},L} - J_{L_\omega}\dot{\nu} - \dot{J}_{L_\omega}\nu. \tag{7.2.13c}$$

$\ddot{p}^*_L$ is given by (7.2.11a), while ${}^{\mathcal{I}}\dot{\omega}^*_{\mathcal{I},L}$ is obtained from (7.2.11b). J_{L_p} and J_{L_ω} are, respectively, the matrices composed of the first and last three rows of J_L.

Floating base dynamics task

To whole-body control layer often considers the base (3.3.8a) and joint dynamics (3.3.8b) as a set of high priority tasks. In this context, we define the base dynamics task as

$$\Psi_{\mathrm{dyn}_\nu} = h_\nu + M_\nu \dot{\nu} - J^\top_{\mathcal{C}_\nu} \mathrm{f}. \tag{7.2.14}$$

On the other hand, the joint dynamics task is given by

$$\Psi_{\mathrm{dyn}_s} = h_s + M_s(q)\dot{\nu} - \tau - J^\top_{\mathcal{C}_s} \mathrm{f}, \tag{7.2.15}$$

where the subscript ν refers to the first 6 rows of the matrix, while s to the last n rows, with n the number of actuated degrees of freedom of the system.

Joint regularization task

To prevent the controller from providing solutions with huge joint variations, we introduce a regularization tasks for the joint variables. The task is achieved by asking for a desired joint velocity that depends on the error between the desired and measured joint values, such as:

$$\Psi_s = \ddot{s}^* - \begin{bmatrix} 0_{n\times 6} & I_n \end{bmatrix} \dot{\nu}, \tag{7.2.16}$$

$\ddot{s}^*$ is equal to

$$\ddot{s}^* = \ddot{s}^{\mathrm{ref}} + k^d_s(\dot{s}^{\mathrm{ref}} - \dot{s}) + k^p_s(s^{\mathrm{ref}} - s). \tag{7.2.17}$$

Here s^{ref} is the desired joint configuration trajectory and is often set constant, i.e. $s^{\mathrm{ref}}(t) = \bar{s}$.

Local ZMP task

Given a desired ZMP position expressed in the $\mathcal{C}_j[\mathcal{I}]$ frame, we want to define a task such that the desired contact force in $\mathcal{C}_j$ generates ${}^{\mathcal{C}_j[\mathcal{I}]}x_{\mathrm{ZMP}_j}$. We now recall that, given

a 6D force ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$, the ZMP, if exists, is given by [Vukobratović et al., 2004]

$$ {}_{\mathcal{C}_j[\mathcal{I}]}x_{\mathrm{ZMP}_j} = \begin{bmatrix} -\frac{{}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_y}}{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}} \\ \frac{{}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_x}}{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}} \end{bmatrix}. \tag{7.2.18} $$

Assuming that ${}_{\mathcal{C}_j[\mathcal{I}]}f_z \neq 0$, i.e., the contact is active, we rearrange Equation (7.2.18) as

$$ {}_{\mathcal{C}_j[\mathcal{I}]}f_{z_j}\, {}^{\mathcal{C}_j[\mathcal{I}]}x_{\mathrm{ZMP}_j} = \begin{bmatrix} -{}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_y} \\ {}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_x} \end{bmatrix}. \tag{7.2.19} $$

Considering (7.2.19), we define the local ZMP task for the contact $\mathcal{C}_j$ as

$$ \Psi_{\mathrm{ZMP}_j} = \left({}_{\mathcal{C}_j[\mathcal{I}]}x^{\mathrm{ref}}_{\mathrm{ZMP}_j} e_3^\top - \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \right) {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j. \tag{7.2.20} $$

Global ZMP task

In the case all the robot contact lies on the same plane, for example, the robot is walking on a planar surface, it is possible to ask for a desired global ZMP, denoted with ${}^{\mathcal{I}}x_{\mathrm{ZMP}}$ (please note that the suffix j is not present in this case since the ZMP is considered as a global quantity and not strictly dependent to the contact). Given n_c coplanar contacts, we define the global ZMP as [Kajita et al., 2014, Section 3.2.3]:

$$ {}^{\mathcal{I}}x_{\mathrm{ZMP}} = \sum_{j=1}^{n_c} \frac{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}}{\sum_{i=1}^{n_c} {}_{\mathcal{C}_i[\mathcal{I}]}f_{i_z}} \left(p_j + {}^{\mathcal{C}_j[\mathcal{I}]}x_{\mathrm{ZMP}_j} \right). \tag{7.2.21} $$

By combining (7.2.18) with (7.2.21) we obtain

$$ {}^{\mathcal{I}}x_{\mathrm{ZMP}} = \sum_{j=1}^{n_c} \frac{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}}{\sum_{i=1}^{n_c} {}_{\mathcal{C}_i[\mathcal{I}]}f_{i_z}} \left(p_j + \begin{bmatrix} -\frac{{}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_y}}{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}} \\ \frac{{}_{\mathcal{C}_j[\mathcal{I}]}\mu_{j_x}}{{}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z}} \end{bmatrix} \right). \tag{7.2.22} $$

Assuming that ${}_{\mathcal{C}_j[\mathcal{I}]}f_{j_z} > 0$ we define the global ZMP task as

$$ \Psi_{\mathrm{ZMP}} = \left({}^{\mathcal{I}}x^{\mathrm{ref}}_{\mathrm{ZMP}} \begin{bmatrix} e_3^\top & \dots e_3^\top \end{bmatrix} - \begin{bmatrix} \Gamma_1 & \dots \Gamma_{n_c} \end{bmatrix} \right) \mathrm{f}, \tag{7.2.23} $$

where $\Gamma_i \in \mathbb{R}^{2\times 6}$ writes as

$$\Gamma_i = \begin{bmatrix} 0 & 0 & p_{i_x} & 0 & -1 & 0 \\ 0 & 0 & p_{i_y} & 1 & 0 & 0 \end{bmatrix}. \tag{7.2.24}$$

Generic variable regularization task

In order to prevent the controller from providing solutions with huge torque τ or contact force ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$, we introduce a regularization task of the form

$$\Psi_u = u^{\mathrm{ref}} - u, \tag{7.2.25}$$

where u is either the joint torque τ or the contact force ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$.

Feasibile contact force task

The feasibility of the contact wrench ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$ is guaranteed by a set of inequality of the form:

$$\Phi_{\mathrm{f}_j}: \quad A_{\mathcal{C}_j[\mathcal{I}]}\, {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j \leq b. \tag{7.2.26}$$

where $A_{\mathcal{C}_j[\mathcal{I}]}$ is a matrix that depends on the position of the robot joints and on the base pose.

More specifically, ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$ must belong to the associated friction cone, while the position of the local CoP is constrained within the support polygon.

Joint limits task

The feasibility of the joint position s is generally guaranteed by means of a set of inequalities of the form

$$s^- \leq s \leq s^+, \tag{7.2.27}$$

where s^- and s^+ are respectively the lower and upper joints position limits. In the case of dynamics-based whole-body controller, the joint values s cannot be arbitrary chosen. To overcome this issue, we substitute the joint position s in (7.2.27) with its second-order discrete dynamics computed at time $t_0 + k\,\mathrm{d}\,t$ by applying the forward Euler method, i.e. $s(t_0 + (k+1)\,\mathrm{d}\,t) = s_{k+1} = s_k + \mathrm{d}\,t\dot{s}_k + 0.5\,\mathrm{d}\,t^2\ddot{s}_k$:

$$s^- \leq s_k + \mathrm{d}\,t\dot{s}_k + \frac{1}{2}\,\mathrm{d}\,t^2\ddot{s}_k \leq s^+. \tag{7.2.28}$$

Rearranging (7.2.28) we obtain the final formulation of the joint limits task

$$\Phi_s : \quad 2\frac{s^- - s - \mathrm{d}\, t\dot{s}}{\mathrm{d}\, t^2} \leq \dot{s} \leq 2\frac{s^+ - s - \mathrm{d}\, t\dot{s}}{\mathrm{d}\, t^2}. \tag{7.2.29}$$

where we remove the time dependency k.

Feasible joint torque

To guarantee feasible contact torque, we define the following inequality task

$$\Phi_\tau : \quad \tau^- \leq \tau \leq \tau^+, \tag{7.2.30}$$

where τ^- and τ^+ represent the maximum negative and positive torque that the joints can produce.

7.2.2 Quadratic programming problem

The control objective is achieved by transcribing the control problem as a constrained optimization problem considering the tasks presented in Section 7.2.1. Similarly to what we discussed for the kinematics-based whole-body controller in Section 7.1, an equality task Ψ can always be considered as a lower or high priority task, or, in another word, as a term of the cost function or as an equality constraint. Consequently, this makes the dynamics based whole-body problem modular[3].

From now on, we consider the following set of tasks. The tracking of the left and right foot poses are considered as high priority SE(3) tasks (7.2.10) and they are denoted respectively as $\Psi_{L_{\mathrm{SE(3)}}}$ and $\Psi_{R_{\mathrm{SE(3)}}}$. We take into account the CoM tracking as high priority task (7.2.7) and the global ZMP tracking (7.2.23). We also consider the base (7.2.14) and the joint dynamics (7.2.15) as high priority tasks. The torso orientation is implemented as a low priority task SO(3) task (7.2.13) and we denote it with $\Psi_{T_{\mathrm{SO(3)}}}$. Furthermore, the joint postural condition (7.2.16) is also added as a low priority task. To minimize the joint torques and forces, we also add two regularization tasks (7.2.25) denoted Ψ_τ and Ψ_{f}. We also ask for a joint torque τ such that the inequality joint limits task (7.2.30) is satisfied. Finally, to guarantee feasible contact

[3]The `QPTSID` class implemented in our framework exploits this modularity to customize the control problem: `https://github.com/ami-iit/bipedal-locomotion-framework/tree/v0.6.0/src/TSID`

forces for the left and right feet, we add the task (7.2.26), denoted respectively as Φ_{f_L} and Φ_{f_R}.

The above hierarchical control objectives can be cast into a whole-body optimization problem:

$$
\begin{aligned}
\underset{\dot{\nu},\,\tau,\,\mathrm{f}}{\text{minimize}} \quad & \Psi_{T_{\mathrm{SO(3)}}}^{\top} \Lambda_T \Psi_{T_{\mathrm{SO(3)}}} + \Psi_s^{\top} \Lambda_s \Psi_s + \Psi_{\mathrm{f}}^{\top} \Lambda_{\mathrm{f}} \Psi_{\mathrm{f}} + \Psi_{\tau}^{\top} \Lambda_{\tau} \Psi_{\tau} && \text{(7.2.31a)} \\
\text{subj. to} \quad & \Psi_{L_{\mathrm{SE(3)}}} = 0 && \text{(7.2.31b)} \\
& \Psi_{R_{\mathrm{SE(3)}}} = 0 && \text{(7.2.31c)} \\
& \Psi_{\mathrm{CoM}} = 0 && \text{(7.2.31d)} \\
& \Psi_{\mathrm{dyn}_{\nu}} = 0 && \text{(7.2.31e)} \\
& \Psi_{\mathrm{dyn}_{s}} = 0 && \text{(7.2.31f)} \\
& \Psi_{\mathrm{ZMP}} = 0 && \text{(7.2.31g)} \\
& \Phi_s && \text{(7.2.31h)} \\
& \Phi_{\tau} && \text{(7.2.31i)} \\
& \Phi_{\mathrm{f}_L} && \text{(7.2.31j)} \\
& \Phi_{\mathrm{f}_R} && \text{(7.2.31k)}
\end{aligned}
$$

Since the decision variables are the robot acceleration $\dot{\nu}$, joint torques τ and the contact forces ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$, and the tasks depend linearly on them, we convert the optimization problem (7.2.31) into a quadratic programming problem (Section 5.4) and we solve it via off-the-shelf solvers.

7.3 Experimental results

In this section, we present experiments obtained from several implementations of the whole-body controllers presented in Sections 7.1 and 7.2. The experimental activities are carried out with the humanoid robot iCub v2.7 [Metta et al., 2010] – see Section 1.1.1. To test the whole-body controllers, we decide to exploit the thee-layer controller architecture – Figure 6.1. In this context, the trajectory optimization layer implements the DCM planner presented in Section 10.2.1. While the swing foot trajectories are generated minimizing the spatial acceleration as discussed in Section 10.2.2. The simplified model control layer implements the instantaneous controller described in Section 10.2.3.

The control architecture runs on the iCub head's computer, namely a 4-th generation Intel® Core i7 @ 1.7 GHz. In any of its implementations, the architecture takes (on average) less than 2 ms to evaluate its outputs. The optimization problems are solved using the OSQP library [Stellato et al., 2018a].

We compare the whole-body controllers using a similar approach presented by Torricelli et al. [2015]. In all experiments, the humanoid robot walks on a horizontal ground at a constant speed[4]. In the following sections, we benchmark the different implementations of the whole-body controllers, focusing on two main aspects: *tracking* and *energy consumption* performances.

7.3.1 Tracking performances

This section presents the tracking performances analysis of the kinematics-based and the dynamics-based whole-body controllers.

Kinematics-based walking architecture

To compare the kinematics-based controller architectures, we decide to perform two main experiments in which the robot walks with two different straight velocity. Namely:

- **experiment 1** the forward robot speed is $0.1563\,\mathrm{m\,s^{-1}}$;
- **experiment 2** the forward robot speed is $0.3372\,\mathrm{m\,s^{-1}}$.

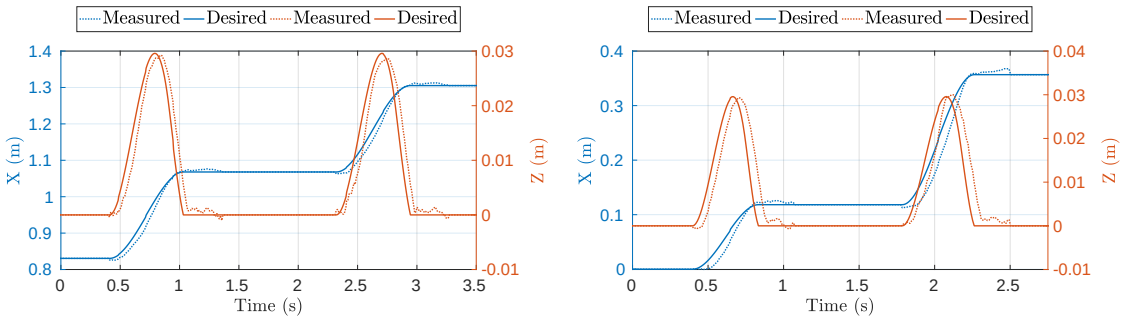
The choice of these velocities derives from the comparison between different simplified model control layers presented in Section 10.3.

Figures 7.2a and 7.3a show the tracking of the desired positions of the left foot when the robot is position and velocity controlled, respectively. The position controller ensures better tracking performance than the velocity one. One may consider increasing the gains of the controllers (7.1.16), however, increasing too much the gains induces overall oscillation in the robot.

The aforementioned foot tracking problem worsens at higher walking velocity. Figure 7.2b shows that the feet' tracking error is lower than 5 cm on the x axis and 0.5 cm on the z one for position control. Instead, the velocity control in Figure 7.3b keeps the error always lower than 6 cm and 3 cm on the the x and y components, respectively.

[4]We define the walking velocity as the ratio between the step length and the measured step duration.

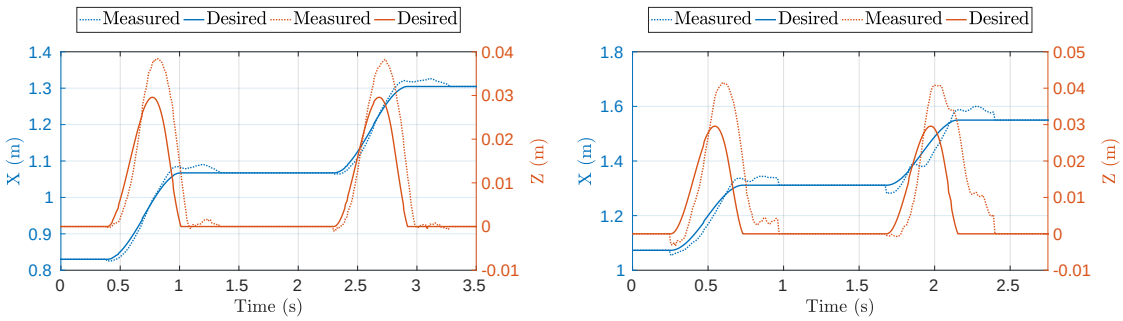
Position Control



(a) $0.1563\,\mathrm{m\,s^{-1}}$ (b) $0.3372\,\mathrm{m\,s^{-1}}$

Figure 7.2 Tracking of the left foot position using Whole-body QP control as inverse kinematics. (a) Straight velocity $0.1563\,\mathrm{m\,s^{-1}}$. (b) Straight velocity $0.3372\,\mathrm{m\,s^{-1}}$.

Velocity Control



(a) $0.1563\,\mathrm{m\,s^{-1}}$ (b) $0.3372\,\mathrm{m\,s^{-1}}$

Figure 7.3 Tracking of the left foot position using Whole-body QP control as velocity control. (a) Straight velocity $0.1563\,\mathrm{m\,s^{-1}}$. (b) Straight velocity $0.3372\,\mathrm{m\,s^{-1}}$.

Dynamics-based walking architecture

Controlling the robot using a torque controller architecture is not an easy task.

Indeed, the performance guaranteed by the position/velocity architecture is not reached because of an imperfect low-level torque control, presence of friction and model errors. For this reason, to validate the torque architecture, we also decide to present the simulation results. When the robot is torque controlled, the noise affecting the measured DCM does not allow us to use the simplified model controllers. Thus we decided to stabilize a desired CoM instead of DCM. Indeed the simplified model control, injects a (desired) ZMP that depends on the measured DCM. As the consequence, it generates undesired vibrations on the robot. We also tried to implement low pass filters

for mitigating such behavior. However, we did not find the right trade-off for obtaining overall performance improvements. Although the extensive hand-made tuning of the simplified model controllers, we were not able to close the loop on the desired DCM. Tracking down the source of the DCM noise to the measured joint velocities, we decided to stabilize a desired CoM trajectory instead. In order to maintain consistency with the previous architectures, we generate such trajectory from the LIPM dynamics (4.4.14a) starting from a desired DCM trajectory.

Table 7.1 summarizes the maximum velocities achieved using different implementations of the dynamics-based architecture. The labels *simulation* and *real robot* mean that the experiments are carried out on the Gazebo Simulator [Koenig and Howard, 2004] or the real platform, respectively. To further test the dynamics based whole-body controller we decied to perform simulation experiments with two different implementation of the Simplified model control layer, namely the instantaneous controller (Section 10.2.3) and a model predictive controller (Section 10.2.3). We denote these two types of controller as *instantaneous* and *predictive* in the Table 7.1.

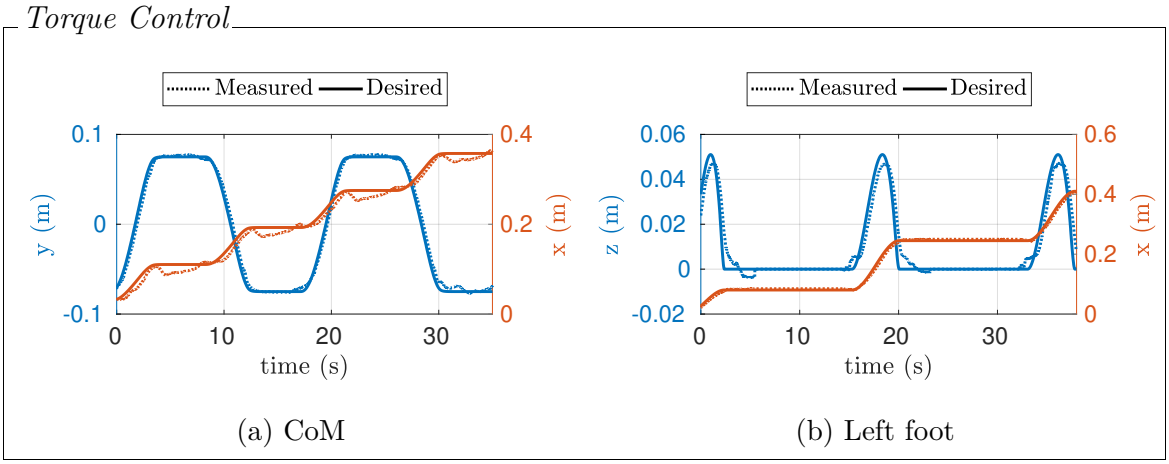
Experiments on the real robot In this section, we present the performance of the walking architecture when the robot is torque controlled. Figure 7.4a depicts the CoM tracking performances. It is important to notice that the tracking error on the x-axis is greater than the one on the y-axis. To reduce this, one may tend to increase the associated gain. However, our experience showed that increasing the CoM gain contributes to the overall vibration of the robot.

Figure 7.4b depicts the tracking of the desired left foot trajectory. Event if the walking velocity is lower than the one used for the kinematics based architecture, the dynamics based whole-body QP is not able to guarantee good performances. One may consider increasing the gains of the feet controller (7.2.31b), (7.2.31c), although the extensive hand-made tuning, we were not able to increase the robot velocity.

Such poor performances may be attributed to the low-level torque controller. Indeed, as depicted in Figure 7.5 the tracking performance of the low-level torque control

Table 7.1 Maximum forward walking velocities achieved in simulation and in a real scenario in case of a torque-controlled robot.

Platform	Simplified Model Control	Max Straight Velocity (m/s)
Real Robot	-	0.0186
Simulation	Instantaneous	0.2120
Simulation	Predictive	0.1448



(a) CoM (b) Left foot

Figure 7.4 Tracking of the CoM (a), and left foot position (b) with whole-body QP control as torque control.

is poor. One is tempted to increase the gains of the low-level torque controller to ensure better performances. However, since the iCub robot does not have joint torque sensors, joint torques are estimated by using the readouts of the force-torque sensors – Section 1.1.1. We observed that the noise due to the force-torque sensors is harmful to the estimated torque, and consequently increasing too much the gains causes undesired overall vibrations.

Experiments on the simulation scenario In this section, we present the simulation results. To simplify the analysis we decide to show only the results when the robot walks with a forward velocity of $0.1448\,\mathrm{m\,s^{-1}}$.

Figures 7.6a and 7.7a presents the tracking performance with the instantaneous and the predictive simplified model controller, respectively. Both implementations guarantee excellent performances, with a CoM error below 1 cm. Notice that when the simplified model controller layer is implemented with the instantaneous controller, the whole-body QP control layer sometimes fails to find an admissible solution. This happens because the desired ZMP, evaluated using the instantaneous controller, may exit the feet support polygon, so it may be not feasible. To face this issue we suggest projecting the desired ZMP onto the support polygon [Englsberger et al., 2011] Figures 7.6b and 7.7b depict the tracking of the desired left foot trajectory. The controller is able to guarantee a tracking error always below 1 cm.

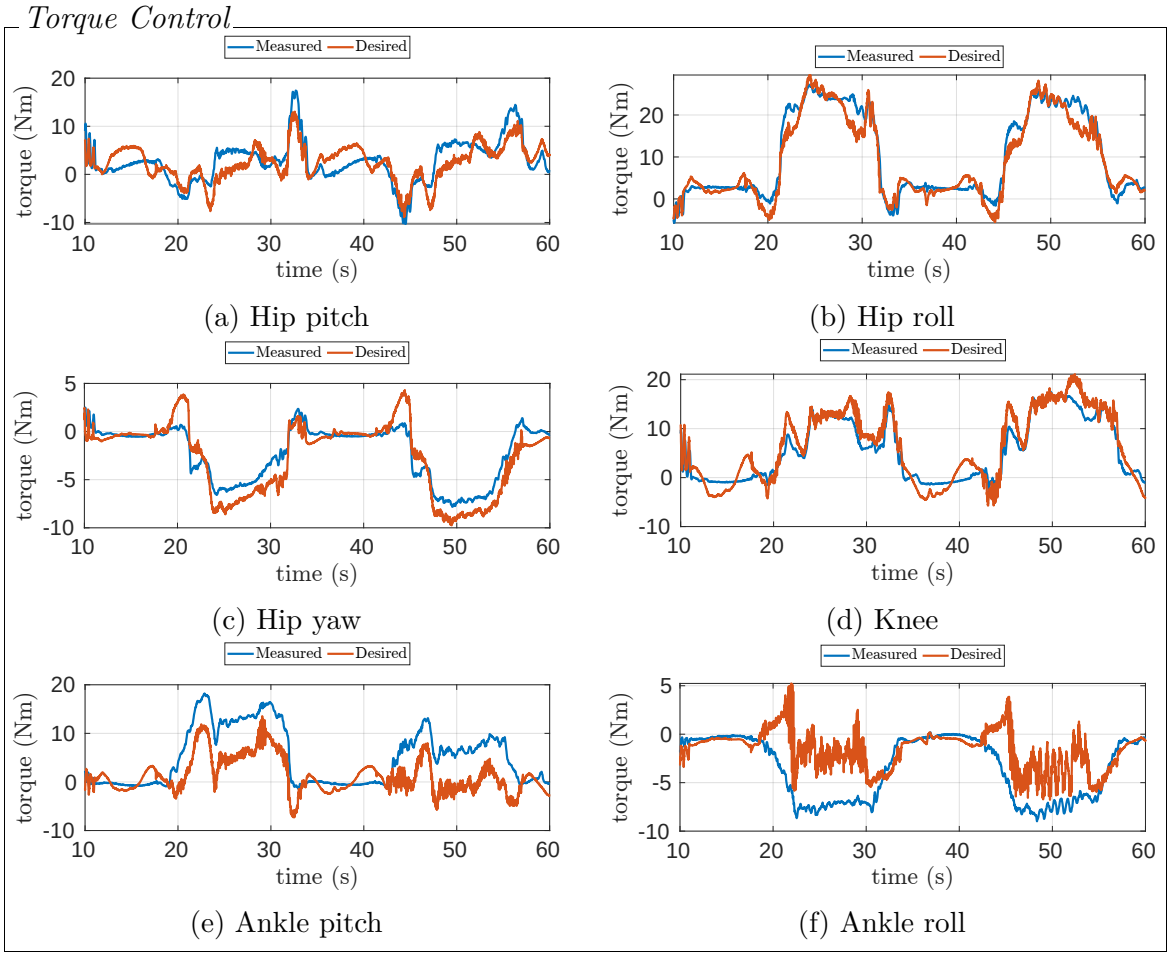


Figure 7.5 Tracking of the desired joint torques of the left leg.

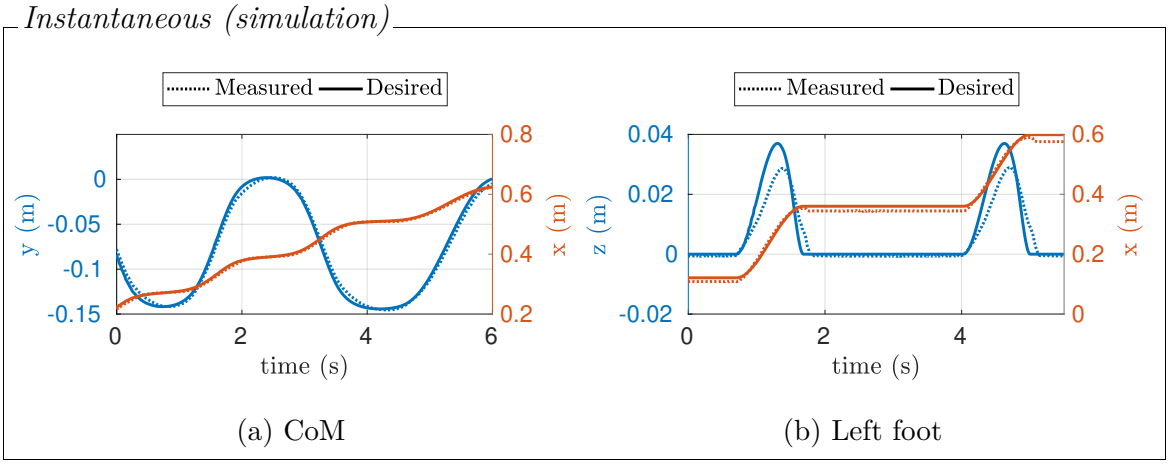
7.3.2 Energy consumption

To compare the energy efficiency of different control architecture we use the *Specific Energetic Cost*. The *Specific Energetic Cost* is defined as: [Torricelli et al., 2015]

$$c_{et} = \frac{E}{mD}, \tag{7.3.1}$$

where E is the positive mechanical work of the actuation system, m is the mass of the system, and D is the distance traveled.

Table 7.2 summarizes the Specific Energetic Cost evaluated using different implementations of the architecture. The labels *Simulation* and *Real Robot* mean that the experiments are carried out on the Gazebo Simulator or the real platform, respectively. The labels, *Position* and *Torque* control, instead, mean that the whole-body QP control



(a) CoM (b) Left foot

Figure 7.6 Instantaneous simplified controller and whole-body controller tracking in simulation. (a) CoM trajectory. (b) Swing foot trajectory. Forward velocity: $0.1448\,\mathrm{m\,s^{-1}}$.

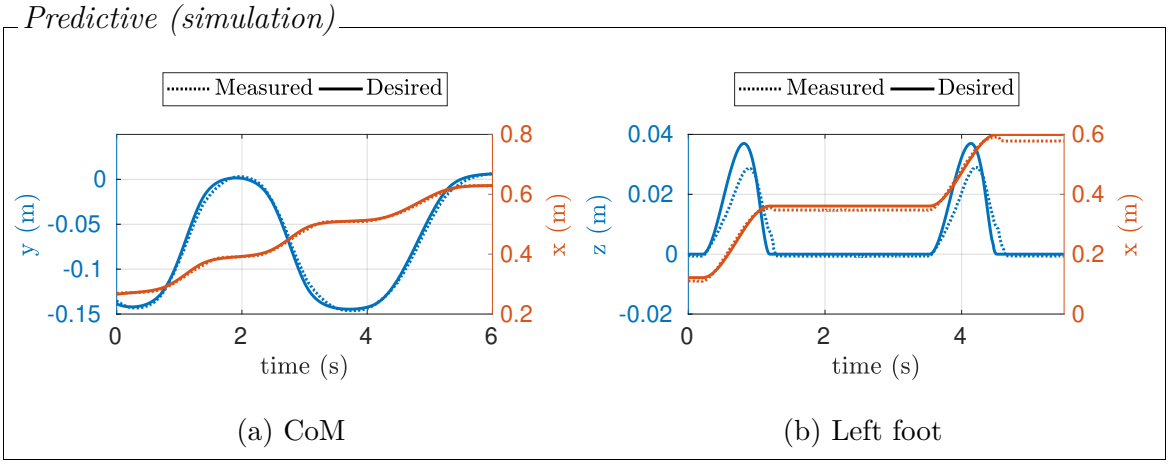
layer outputs are either desired joint positions or torque, respectively. We noticed the Dynamics-based architecture has a lower Specific Energetic Cost than the Kinematics-based architecture; the reason of this result is attributable to the minimization of the joint torque when the robot is torque controlled – see Equation (7.2.31a).

7.4 Conclusion

This chapter presents and compares several whole-body controllers for locomotion in a rigid environment, namely a kinematics-based whole-body controller and a controller that takes into consideration the dynamics of the system. Thanks to the modularity of the proposed controller, it is possible to exchange the two implementations depending on the low-level control interfaces available on the robot. The chapter also presents an extensive benchmarking of the different whole-body controller implementations in a

Table 7.2 Specific Energetic Cost evaluated in simulation and in a real scenario in case of torque and position controlled robot.

Platform	Whole-Body QP Control	Velocity (m/s)	c_{et} (J/Kg/m)
Real Robot	Position	0.0186	9.66
Real Robot	Torque	0.0186	3.85
Simulation	Position	0.2120	4.82
Simulation	Torque	0.2120	2.55



(a) CoM (b) Left foot

Figure 7.7 Predictive simplified controller and whole-body controller tracking in simulation. (a) CoM trajectory. (b) Swing foot trajectory. Forward velocity: $0.1448\,\mathrm{m\,s^{-1}}$.

three-layer controller architecture. We carry on the test on the Humanoid Robot iCub v2.7 – see Section 1.1.1.

It is worth mentioning that the presented controllers rely on the assumption of rigid contacts. In the next chapter, we will attempt to loosen this hypothesis by modeling the contact between the robot and the environment as a continuous spring-damper system. The presented contact model will then be considered in the design of a whole-body controller that allows a simulated version of the humanoid robot iCub to walk on compliant terrain.

Chapter 8

Whole-Body Controller on Visco Elastic Environment

In Chapter 7, we introduced a whole-body controller for humanoid robot locomotion in contact with a stiff environment. When the contact between a robot and its surroundings is sufficiently rigid, controllers based on the rigid contact assumption may nevertheless result in satisfactory robot performance. However, when contact compliance reduces robot performance, it is necessary to design contact models that account for contact stiffness and damping, which may subsequently be included in feedback controllers. In light of that, this chapter presents a model of compliant contacts for time-critical humanoid robot motion control. The proposed model considers the environment as a continuum of spring-damper systems, allowing us to compute the equivalent contact force and torque that the environment exerts on the contact surface. We show that the proposed model extends the linear and rotational springs and dampers, classically used to characterize soft terrains, to the case of *large* contact surface orientations. The contact model is then used for the real-time whole-body control of humanoid robots walking in visco-elastic environments. The overall approach is validated by simulating walking motions of the iCub humanoid robot. Furthermore, we compare the proposed whole-body control strategy with state-of-the-art approaches. In this respect, we investigate the terrain compliance that makes the classical approaches that assumes rigid contacts fail. We finally analyze the robustness of the presented control design with respect to non-parametric uncertainty in the contact model.

The chapter is organized as follows: Section 8.1 presents the model used to characterize the interaction between the robot and the environment. Section 8.2 details the whole-body controller for walking on compliant surfaces. The section also contains

the design of an observer to estimate the contact parameters. Section 8.3 presents the simulation results on the iCub humanoid robot v2.7 – see Section 1.1.1. Finally, Section 8.4 concludes the chapter.

The content of this chapter appears in:

> Romualdi, G., Dafarra, S., and Pucci, D. (2021). Modeling of Visco-Elastic Environments for Humanoid Robot Motion Control. *IEEE Robotics and Automation Letters*, 6(3):4289–4296
>
> Video `https://www.youtube.com/watch?v=7XKQ5ZWJvYU`
> GitHub `ami-iit/romualdi-2021-ral-soft_terrain_walking`

8.1 Modeling of visco-elastic environments

Let us now consider a rigid body that makes a contact with a visco-elastic surface, and we assume that:

1. there exists an inertial frame $\mathcal{I}$;
2. there exist a frame B rigidly attached to the body and we denote o_B the origin of the frame and $[B]$ its orientation;
3. all the point of the rigid body in contact with the environment define a set denoted with $\Omega \in \mathbb{R}^3$ and named *contact domain*, we denote with ${}^{B}x$ a point on the contact surface expressed in the frame B, while with ${}^{\mathcal{I}}x$ the very same point expressed in the inertial frame $\mathcal{I}$;
4. the environment characteristics are isotropic;
5. the rigid body moves with a 6D velocity, denoted as ${}^{B[\mathcal{I}]}\mathrm{v}$ such that

$$ {}^{B[\mathcal{I}]}\mathrm{v} = \begin{bmatrix} {}^{\mathcal{I}}\dot{o}_B \\ {}^{\mathcal{I}}\omega_{\mathcal{I},B} \end{bmatrix} \tag{8.1.1} $$

where $B[\mathcal{I}] = (o_B, [\mathcal{I}])$ is a frame having the origin in o_B and oriented as $\mathcal{I}$.

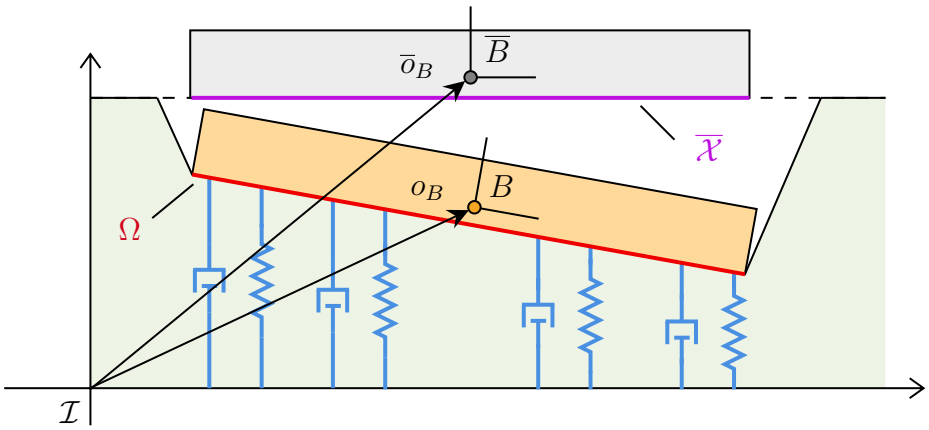


Figure 8.1 2D representation of the visco-elastic model. The gray rectangle represents the zero-force rigid body position. The orange rectangle is the body. Ω is the contact domain while $\overline{\mathcal{X}}$ is equal to Ω if the contact wrench is null ${}_{B[\mathcal{I}]}\mathrm{f} = 0$. The interaction between the rigid body and the environment can be approximate as a continuum of spring-damper systems.

6. $\forall x \in \Omega$, there exists a continuous pure force distribution that depends on the point ${}^{\mathcal{I}}x$ and its velocity ${}^{\mathcal{I}}\dot{x}$, i.e.,

$$\rho_x : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3. \tag{8.1.2}$$

Figure 8.1 shows a rigid body in contact with a visco-elastic environment.

Each point of Ω may define a different function ρ_x. For the sake of simplicity, we assume that ρ_x is the same for each point in contact with the environment. Consequently, we drop the subscript x in Equation (8.1.2). Given the above assumptions, the contact torque distribution about a point $o_B \in \mathbb{R}^3$, $\sigma_{o_B} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$ writes

$$\sigma_{o_B}\left({}^{\mathcal{I}}x, {}^{\mathcal{I}}\dot{x}\right) = \left({}^{\mathcal{I}}x - o_B\right) \times \rho\left({}^{\mathcal{I}}x, {}^{\mathcal{I}}\dot{x}\right). \tag{8.1.3}$$

Once the pure force and torque distribution are defined, then the equivalent contact 6D force expressed in mixed representation is given by [Caron et al., 2015] – Section 4.2:

$${}_{B[\mathcal{I}]}\mathrm{f} = \begin{bmatrix} {}_{\mathcal{I}}f \\ {}_{B[\mathcal{I}]}\mu \end{bmatrix} = \begin{bmatrix} \int_\Omega \rho \, \mathrm{d}\,\Omega \\ \int_\Omega \sigma_{o_B} \, \mathrm{d}\,\Omega \end{bmatrix}. \tag{8.1.4}$$

We now propose a model that can be used to describe the contact between a body and a compliant environment.

Lemma 1. *Let $\bar{\mathcal{X}}$ the set of points $\bar{x} \in \mathbb{R}^3$:*

$$\bar{\mathcal{X}} = \{\bar{x} \in \mathbb{R}^3 : \rho(\bar{x}, 0) = 0\}. \tag{8.1.5}$$

Assume that: i) the contact domain Ω is a rectangle of dimensions l and w; ii) the point $o_B \in \mathbb{R}^3$ is the center of the rectangular domain; iii) the distribution ρ is given by:

$$\rho\left({}^{\mathcal{I}}x, {}^{\mathcal{I}}\dot{x}\right) = k\left({}^{\mathcal{I}}\bar{x} - {}^{\mathcal{I}}x\right) - b\,{}^{\mathcal{I}}\dot{x}, \tag{8.1.6}$$

with $k > 0$ and $b > 0$.

Then, the equivalent contact force and torque ${}_{B[\mathcal{I}]}\mathrm{f}$ (8.1.4) are given by

$$\begin{aligned}
{}_{\mathcal{I}}f &= lw|e_3^\top {}^{\mathcal{I}}R_B e_3|[k(\bar{o}_B - o_B) - b\dot{o}_B] && \textit{(8.1.7a)}\\
{}_{B[\mathcal{I}]}\mu &= \frac{lw}{12}|e_3^\top {}^{\mathcal{I}}R_B e_3| \\
&\quad \Big\{ l^2({}^{\mathcal{I}}R_B e_1) \times [b({}^{\mathcal{I}}R_B e_1) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} + k\,{}^{\mathcal{I}}\bar{R}_B e_1] && \textit{(8.1.7b)}\\
&\quad + w^2({}^{\mathcal{I}}R_B e_2) \times [b({}^{\mathcal{I}}R_B e_2) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} + k\,{}^{\mathcal{I}}\bar{R}_B e_2] \Big\},
\end{aligned}$$

where ${}^{\mathcal{I}}R_B$ is the rotation from the inertial frame $\mathcal{I}$ to a frame rigidly attached to the body B. $\dot{o}_B$ and ${}^{\mathcal{I}}\omega_{\mathcal{I},B}$ are the linear and angular velocity of the rigid body expressed in mixed representation. $\bar{o}_B$ and ${}^{\mathcal{I}}\bar{R}_B$ are the position and the rotation of the frame B such that in case of zero velocity, the 6D force is null.

The set $\bar{\mathcal{X}}$ can also be defined by considering a point P that belongs to the contact domain Ω of the rigid body B in contact with the compressible environment. Then, the point P can move to a point P_0 of the space such that the force distribution at the point P_0 is zero. In coordinates, let P be defined by the coordinate vector $x_p \in \mathbb{R}^3$ and P_0 be defined by the coordinate vector $\bar{x}_p \in \mathbb{R}^3$. Then given x_p

$$\rho(\bar{x}_p, 0) = 0. \tag{8.1.8}$$

$\bar{\mathcal{X}}$ contains all the points $\bar{x}_p$ associated with each point P belonging to the contact domain Ω. The proof of Lemma 1 is in Appendix B.

Lemma 1 shows that the 6D contact force (8.1.4) depends only on the distribution of the contact force ρ on the shape of the contact domain Ω and on the rigid body state, namely *position*, *orientation*, *linear* and *angular velocity*. As a consequence, we avoid using rotational springs and dampers to describe the interaction between the robot and

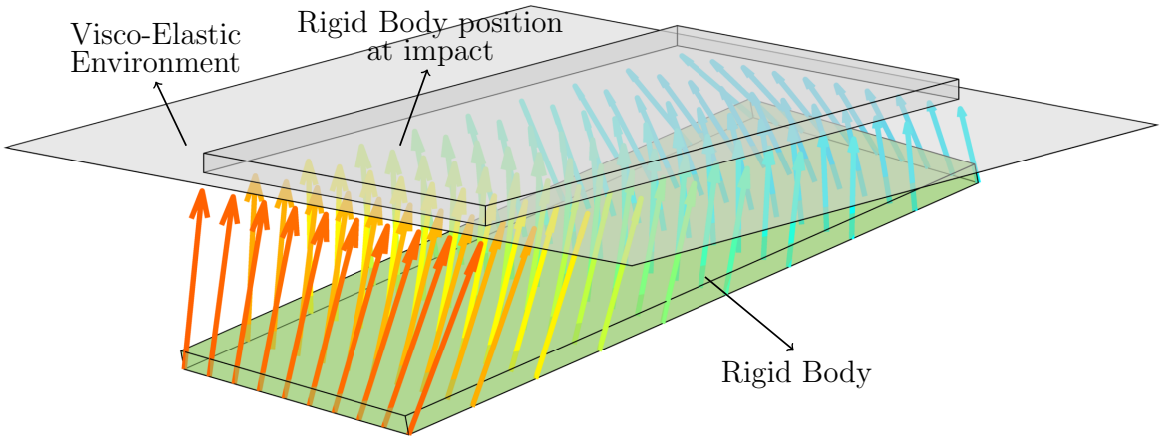


Figure 8.2 Vector field generated by the visco-elastic model – Equation (8.1.6). The grey box represents the zero-force rigid-body position. The green parallelepiped is the body. The arrows represent the forces that act on the contact surface; the lighter the color, the higher the force magnitude.

the environment. Furthermore, Lemma 1 also contains a close solution for the equivalent contact wrench (8.1.7a) (8.1.7b) in the case of a linear contact model (8.1.6) and a rectangular contact surface Ω. As a consequence, it can be exploited in hard-real-time applications such as control architectures. To give the reader a better understanding, we can imagine that the set $\bar{\mathcal{X}}$ contains the position of all the points on the foot sole at the touch down. Once the contact is established, the environment is deformed. The interaction between the foot and the environment is then approximated as a continuum of spring-damper systems – Equation (8.1.6). Each spring-damper system exerts a force on the associated point of the contact domain. Combining all forces, we can imagine that the rigid body is subject to the vector field represented in Figure 8.2. Finally, we want to recall that, since the contacts are unilateral, this model is valid as long as the normal forces are positive and the tangential component lies inside the friction cone.

8.1.1 Linear approximation of the visco-elastic model

It is worth noting that the model (8.1.7a)-(8.1.7b) also encompasses the classical linear modeling techniques for soft terrains. The following corollary shows, in fact, that linear approximations of (8.1.7a)-(8.1.7b) lead to linear and rotational springs and dampers that are usually used to model contact wrenches due to soft terrains [Sygulla and Rixen, 2020, Equation (8)].

Corollary 1. *Let ${}_{\mathcal{I}}f$ and ${}_{B[\mathcal{I}]}\mu$ be the contact force and torque given by* (8.1.7a) *and* (8.1.7b)*, respectively. Assume that ${}^{\mathcal{I}}\bar{R}_B = I_3$ and ${}^{\mathcal{I}}R_B$ is approximated with its first order of the Taylor expansion, i.e., ${}^{\mathcal{I}}R_B = I_3 + \Theta\times$, with $\Theta \in \mathbb{R}^3$. Assume that Θ represents the classical sequence of roll-pitch-yaw, namely ${}^{\mathcal{I}}R_B(\Theta) = R_z(\Theta_3)R_y(\Theta_2)R_x(\Theta_1)$. Then, the contact model* (8.1.7a)-(8.1.7b) *writes*

$$ {}_{\mathcal{I}}f_l = \mathcal{K}_l(\bar{o}_B - o_B) - \mathcal{B}_l\dot{o}_B, \tag{8.1.9a} $$

$$ {}_{B[\mathcal{I}]}\mu_l = -\mathcal{K}_a\Theta - \mathcal{B}_a\dot{\Theta}, \tag{8.1.9b} $$

with

$$ \mathcal{K}_l = lwkI_3, \quad \mathcal{K}_a = k\frac{lw}{12}\begin{bmatrix} w^2 & 0 & 0 \\ 0 & l^2 & 0 \\ 0 & 0 & l^2{+}w^2 \end{bmatrix} \tag{8.1.10} $$

$$ \mathcal{B}_l = lwbI_3, \quad \mathcal{B}_a = b\frac{lw}{12}\begin{bmatrix} w^2 & 0 & 0 \\ 0 & l^2 & 0 \\ 0 & 0 & l^2{+}w^2 \end{bmatrix} \tag{8.1.11} $$

The proof of Corollary 1 is in Appendix C. Corollary 1 thus shows that the model (8.1.7a)-(8.1.7b) extends the linear models [Sygulla and Rixen, 2020] to the case of *large* contact surface orientations.

Here, we want to underline that the classical linear approaches for modeling compliant contacts (8.1.9) – for example, rotational springs and dampers [Sygulla and Rixen, 2020] – are often valid only for *small* contact surface rotations. This is due to the minimal representation (i.e., three angles, such as *roll*, *pitch*, *yaw*) used to represent SO(3). In addition, the equivalent *rotational stiffness and dampers* values $\mathcal{K}_a$ and $\mathcal{B}_a$ are often not related to the physical parameters of the contact. On the other hand, even if the model we propose (8.1.7a)-(8.1.7b) is indeed a 6d force, it is obtained by integrating pressure and shear stresses distributions to better catch the fundamental effects of the contact physics, without having the aforementioned issues of classical rotational spring and damper models.

Let us now introduce the approximation error between the linear model (8.1.9) and the model presented (8.1.7) as $\epsilon_{\mathrm{f}} = {}_{B[\mathcal{I}]}\mathrm{f} - {}_{B[\mathcal{I}]}\mathrm{f}_l$. Figure 8.3 shows the last component of the error ϵ_{f}, i.e., $e_6^\top\epsilon_{\mathrm{f}}$, in the case of zero velocity and zero pitch and roll angles. The higher the angle, the greater the difference. Thus, the higher is the angle, the less valid the linear approximation is. A similar analysis holds also for the other components of the 6D force. To conclude, the model presented in Lemma 1 is not equivalent to a

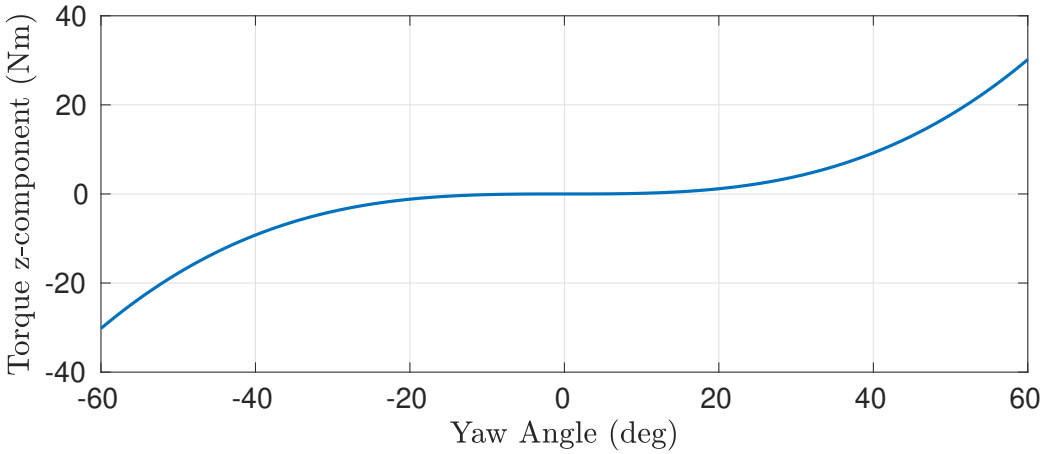


Figure 8.3 Linear approximation error for different values of yaw angle.

linear lumped contact model, but it can be seen as a non-linear generalization while keeping a low mathematical complexity.

8.2 Whole-body controller

The whole-body controller aims to track kinematic and dynamic quantities. The proposed whole-body controller computes the desired joint torques using the robot joint dynamics (3.3.8b), where the robot acceleration $\dot{\nu}$ is set to the desired *starred* quantity and the contact wrenches ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ are estimated or measured:

$$\tau^* = M_s(q)\dot{\nu}^* + h_s(q,\nu) - \sum_{k=1}^{n_c} J_{\mathcal{C}_{k\,s}}^{\top}(q)\ {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k. \tag{8.2.1}$$

The desired generalized robot acceleration $\dot{\nu}^*$ is chosen to follow the desired centroidal momentum trajectory, the torso and root orientation, and the feet pose, while considering the contact model presented in Section 8.1.

The control problem is formulated using the stack of tasks approach. The control objective is achieved by framing the controller as a constrained optimization problem where the low priority tasks are embedded in the cost function, while the high priority tasks are treated as constraints.

8.2.1 Low and high priority tasks

What follows presents the tasks required to evaluate the desired generalized robot acceleration $\dot{\nu}^*$.

Centroidal momentum task

In case of visco-elastic contacts, the contact wrench ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ cannot be arbitrarily chosen. From now on, we assume that we can control the contact wrench derivative ${}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k$. Thus, by differentiating the centroidal momentum dynamics (3.4.5), we obtain:

$$ {}_{\bar{G}}\ddot{h} = \sum_{k=1}^{n_c} \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ \left({}^{\mathcal{I}}\dot{p}_k - \dot{x}_{\mathrm{CoM}}\right)\times & 0_{3\times3} \end{bmatrix} {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k + \begin{bmatrix} I_3 & 0_{3\times3} \\ \left({}^{\mathcal{I}}p_k - x_{\mathrm{CoM}}\right)\times & I_3 \end{bmatrix} {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k \tag{8.2.2} $$

To follow the desired centroidal momentum trajectory, we minimize the weighted norm of the error between the robot centroidal momentum and the desired trajectory:

$$ \Psi_h = \frac{1}{2} \left\| {}_{\bar{G}}\ddot{h}^* - {}_{\bar{G}}\ddot{h} \right\|_{\Lambda_h}, \tag{8.2.3} $$

where Λ_h is a positive definite diagonal matrix. ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ is the estimated/measured contact wrench. ${}_{\bar{G}}\ddot{h}^*$ is the desired centroidal momentum derivative, and it is responsible for stabilizing the desired centroidal momentum dynamics:

$$ \begin{aligned} {}_{\bar{G}}\ddot{h}^* = {}_{\bar{G}}\ddot{h}^{\mathrm{ref}} + k_h^d({}_{\bar{G}}\dot{h}^{\mathrm{ref}} - {}_{\bar{G}}\dot{h}) + k_h^p({}_{\bar{G}}h^{\mathrm{ref}} - {}_{\bar{G}}h) \\ + k_h^i \int {}_{\bar{G}}h^{\mathrm{ref}} - {}_{\bar{G}}h \,\mathrm{d}\,t, \end{aligned} \tag{8.2.4} $$

where the integral of the angular momentum is computed numerically. Here, k_h^d, k_h^p, and k_h^i are three diagonal matrices. When the equality holds, the centroidal dynamics converges exponentially to the reference value if and only if k_h^p, k_h^i and $k_h^d k_h^p - k_h^i$ are positive definite.

Torso and root orientation tasks

While walking, we require the torso and the root frames to have a specific orientation with respect to the inertial frame. To accomplish this task, we minimize the norm of the

error between a desired angular acceleration and the actual frame angular acceleration:

$$\Psi_\circ = \frac{1}{2}\left\|\dot{\omega}_\circ^* - (\dot{J}_\circ \nu + J_\circ \dot{\nu})\right\|_{\Lambda_\circ}^2, \tag{8.2.5}$$

where the subscript $\circ$ indicates the frames of root R and torso T. $\Lambda_\circ$ is a positive definite matrix that weighs the contributions in different directions. $\dot{\omega}_\circ^*$ is set to guarantee almost global stability and convergence of ${}^{\mathcal{I}}R_\circ$ to ${}^{\mathcal{I}}R_\circ^{\text{ref}}$ [Olfati-Saber, 2001]:

$$\begin{aligned}\dot{\omega}_\circ^* = \dot{\omega}^{\text{ref}} &- c_0\left(\hat{\omega}\,{}^{\mathcal{I}}R_\circ\,{}^{\mathcal{I}}R_\circ^{\text{ref}^\top} - {}^{\mathcal{I}}R_\circ\,{}^{\mathcal{I}}R_\circ^{\text{ref}^\top}\hat{\omega}^{\text{ref}}\right)^\vee \\ &- c_1\left(\omega - \omega^{\text{ref}}\right) - c_2\left({}^{\mathcal{I}}R_\circ\,{}^{\mathcal{I}}R_\circ^{\text{ref}^\top}\right)^\vee.\end{aligned} \tag{8.2.6}$$

Here, c_0, c_1, and c_2 are positive numbers.

Swing foot task

Concerning the tracking of the swing foot trajectory, we minimize the following cost function

$$\Psi_F = \frac{1}{2}\left\|\dot{\mathrm{v}}_F^* - (\dot{J}_F \nu + J_F \dot{\nu})\right\|_{\Lambda_F}^2, \tag{8.2.7}$$

the angular part of $\dot{\mathrm{v}}_F^*$ is given by (8.2.6) where the subscript $\circ$ is replaced by F, while the linear part $\ddot{p}^*$ is equal to $\ddot{p}_F^* = \ddot{p}_F^{\,\text{ref}} - k_{x_f}^d(\dot{p}_F - \dot{p}_F^{\text{ref}}) - k_{x_f}^p(p_F - p_F^{\text{ref}})$. Here, the gains are again positive definite matrices.

Regularization tasks

In order to prevent the controller from providing solutions with huge joint variations, we introduce a regularization task for the joint variables. The task is achieved by asking for a desired joint acceleration that depends on the error between the desired and measured joint values, namely:

$$\Psi_s = \frac{1}{2}\left\|k_s^p(s^{\text{ref}} - s) - k_s^d\dot{s} - \ddot{s}\right\|_{\Lambda_s}^2, \tag{8.2.8}$$

where s^{ref} is a desired joint configuration, k_s^d, k_s^p and Λ_s are symmetric positive definite matrices. To reduce the amount of the contact wrench required to accomplished the

centroidal momentum tracking, the following task is considered:

$$\Psi_{\mathrm{f}_k} = \frac{1}{2} \left\| k_f^p \left({}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k^{\mathrm{ref}} - {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k \right) - {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k \right\|^2_{\Lambda_{f_k}}. \tag{8.2.9}$$

Here, Λ_{f_k} and k_f^p are positive definite matrices. ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ is the estimated/measured contact wrench and ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k^{\mathrm{ref}}$ is the desired force regularization value.

Contact wrench feasibility

The feasibility of the contact wrenches ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ is generally guaranteed by another set of inequalities of the form:

$$A_{\mathcal{C}_k[\mathcal{I}]}\ {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k \leq b. \tag{8.2.10}$$

where $A_{\mathcal{C}_k[\mathcal{I}]}$ is a matrix that depends on the position of the robot joints and the base pose.

More specifically, ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ must belong to the associated friction cone, while the position of the local CoP is restricted within the support polygon. However, in the case of visco-elastic contacts, the contact wrenches cannot be arbitrarily chosen. This limitation can be overcome by discretizing the contact wrench dynamics using the forward Euler method:

$${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k^{i+1} = {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k^{i} + {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k^{i}\,\mathrm{d}\,t, \tag{8.2.11}$$

where T is the constant integration time. We can require the contact wrench at the next instant to guarantee the inequality constraints (8.2.10). Combining (8.2.11) and (8.2.10), we can now obtain a tractable set of inequality constraints

$$A_{\mathcal{C}_k[\mathcal{I}]}\ {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k\,\mathrm{d}\,t \leq b - A_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k, \tag{8.2.12}$$

where the superscript i has been dropped, and ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ represents the measured/estimated contact wrench.

Floating-base system base dynamics

The whole-body controller considers the base system dynamics presented in Equation (3.3.8a) as:

$$M_\nu \dot{\nu} + h_\nu = \sum_{k=1}^{n_c} J_{\mathcal{C}_k}^\top {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k. \tag{8.2.13}$$

where ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k$ are the measured/estimated contact wrenches.

Contact model dynamics

The contact wrench dynamics can be obtained by differentiating equations (8.1.7a) and (8.1.7b). By computing the time derivative of f, one has the following control-affine dynamical system:

$$ {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k = h_{\mathrm{f}_k} + g_{\mathrm{f}_k}\, {}^{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{v}}_{F_k} \tag{8.2.14} $$

where $h_{\mathrm{f}_k} \in \mathbb{R}^6$ and $g_{\mathrm{f}_k} \in \mathbb{R}^{6\times 6}$ is full rank for each possible admissible state. By explicating the dependence on the robot generalized acceleration, Equation (8.2.14) writes as:

$$ {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k = h_{\mathrm{f}_k} + g_{\mathrm{f}_k}\left(J_{F_k}(q)\dot{\nu} + \dot{J}_{F_k}(q)\nu\right). \tag{8.2.15} $$

8.2.2 Quadratic programming problem

The control objective is achieved by casting the control problem as a constrained optimization problem whose conditional variables are $\dot{\nu}$ and $\dot{f}_k$ where k represents the foot in contact with the environment, namely left, right, or both. In details:

$$ \begin{aligned}
\underset{\dot{\nu},\dot{f}_k}{\text{minimize}} \quad & \Psi_h + \Psi_{\mathcal{T}} + \Psi_{\mathcal{R}} + \Psi_F + \Psi_s + \Psi_{\mathrm{f}_k} && \text{(8.2.16a)}\\
\text{subject to} \quad & A_{\mathcal{C}_k[\mathcal{I}]\ {}_{\mathcal{C}_k[\mathcal{I}]}}\dot{\mathrm{f}}_k \,\mathrm{d}\,T \le b - A_{\mathcal{C}_k[\mathcal{I}]\ {}_{\mathcal{C}_k[\mathcal{I}]}}\mathrm{f}_k && \text{(8.2.16b)}\\
& M_\nu\dot{\nu} + h_\nu = \sum_{k=1}^{n_c} J_{\mathcal{C}_k}^\top {}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k && \text{(8.2.16c)}\\
& {}_{\mathcal{C}_k[\mathcal{I}]}\dot{\mathrm{f}}_k = h_{\mathrm{f}_k} + g_{\mathrm{f}_k}\left(J_{F_k}(q)\dot{\nu} + \dot{J}_{F_k}(q)\nu\right). && \text{(8.2.16d)}
\end{aligned} $$

We notice that the system base dynamics (8.2.13) and the contact dynamics (8.2.15) depend linearly on the decision variables $\dot{\nu}$ and ${}_{C_k[\mathcal{I}]}\mathrm{f}_k$. Furthermore, the tasks presented in Section 8.2.1 depend quadratically on the decision variables. Consequently, the optimization problem in (8.2.16) can be transcribed into a quadratic programming problem (see Section 5.4) and solved via off-the-shelf solvers. Once the desired robot acceleration $\dot{\nu}^*$ is computed, the desired joint torques can be easily evaluated with Equation (8.2.1).

8.2.3 Contact parameters estimation

The optimal control problem presented in Section 8.2.1 is based on the perfect knowledge of the contact parameters k and b. In a simulated environment, the value of the

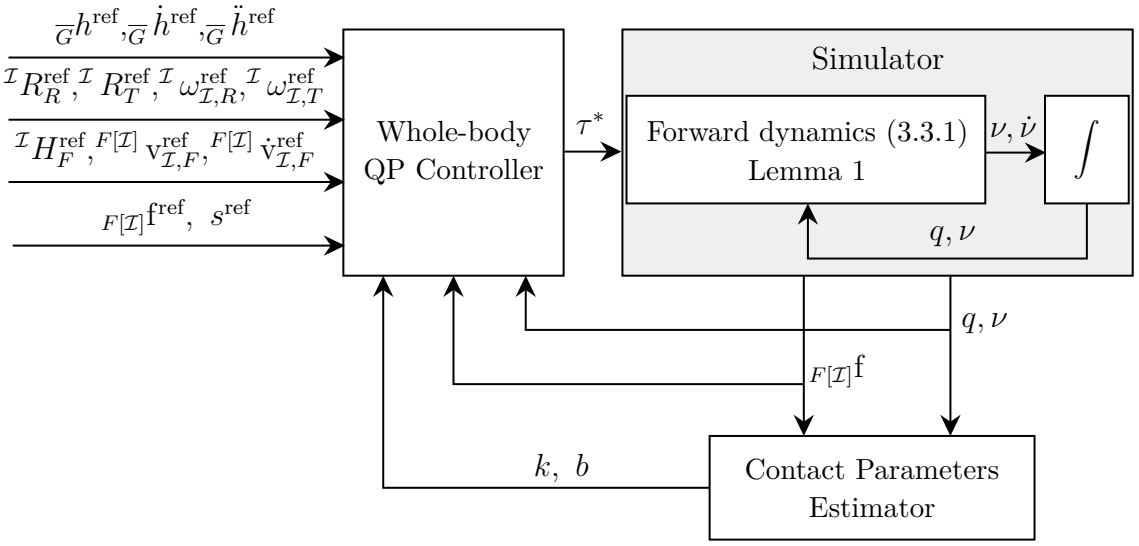


Figure 8.4 Controller architecture.

parameters is perfectly known. However, in the real scenario, an estimation algorithm is required to compute the parameters.

The contact model described by Equations (8.1.7a) and (8.1.7b) is linear with respect to the contact parameters as $_{B[\mathcal{I}]}\text{f} = \mathcal{Y}(o_B, {}^{\mathcal{I}}R_B, \dot{o}_B, {}^{\mathcal{I}}\omega_{\mathcal{I},B})\pi$, where $\mathcal{Y}(o_B, {}^{\mathcal{I}}R_B, \dot{o}_B, {}^{\mathcal{I}}\omega_{\mathcal{I},B}) \in \mathbb{R}^{6\times 2}$ is the regressor and is equal to

$$\mathcal{Y} = lw|e_3^\top Re_3| \begin{bmatrix} \bar{o}_B - o_B & -\dot{o}_B \\ \frac{l^2}{12}\left((Re_1)\times\right)\bar{R}e_1 + \frac{w^2}{12}\left((Re_2)\times\right)\bar{R}e_2 & \left[\frac{l^2}{12}\left((Re_1)\times\right)^2 + \frac{w^2}{12}\left((Re_2)\times\right)^2\right]\omega \end{bmatrix} \tag{8.2.17}$$

where, for sake of clarity, we removed all the prefixes and suffices. $\pi \in \mathbb{R}^2$ contains the spring and damper coefficients, i.e.,

$$\pi = \begin{bmatrix} k \\ b \end{bmatrix}. \tag{8.2.18}$$

We aim to estimate the contact parameters $\hat{\pi}$ so that the *least-squares* criterion is minimized. Namely, we seek for $\hat{\pi}$ such that the error between the measured contact forces and the predicted contact force is minimized over a time windows having a length of t samples:

$$\hat{\pi}_t = \arg\min_{\pi} \frac{1}{2}\sum_{i=1}^{t} \left\| {}_{B[\mathcal{I}]}\text{f}[i] - \mathcal{Y}_i\pi \right\|_{\Gamma_i}^2. \tag{8.2.19}$$

Here, $\Gamma_i \succ 0$ is a strictly positive definite matrix. ${}_{B[\mathcal{I}]}\mathrm{f}[i]$ and $\mathcal{Y}_i$ represent the forces and the regressor computed at instant i.

We solved this problem by applying the Recursive least squares (RLS) filter [Ljung, 1999, Section 11.2] as:

$$\hat{\pi}_t = \hat{\pi}_{t-1} + L_t \left[{}_{B[\mathcal{I}]}\mathrm{f}[t] - \mathcal{Y}_t \hat{\pi}_{t-1} \right], \tag{8.2.20}$$

where ${}_{B[\mathcal{I}]}\mathrm{f}[t] - \mathcal{Y}_t \hat{\pi}_{t-1}$ is the innovation at time t. The gain $L_t \in \mathbb{R}^{2\times 6}$ is given by:

$$L_t = P_{t-1} \mathcal{Y}_t^\top \left[\Gamma_t + \mathcal{Y}_t P_{t-1} \mathcal{Y}_t^\top \right]^{-1}, \tag{8.2.21}$$

P_{t-1} is the estimation error covariance at the instant $t-1$:

$$P_t = \left[I_2 - L_t \mathcal{Y}_t \right] P_{t-1} \left[I_2 - L_t \mathcal{Y}_t \right]^\top + L_t \Gamma_t L_t^\top. \tag{8.2.22}$$

At every time step, Equations (8.2.20)-(8.2.21)-(8.2.22) are used to estimate the contact parameters k and b.

In the case of zero linear and angular velocity, the last three columns of $\mathcal{Y}$ are zero, so the damper parameter is not observable when the contact surface does not move. Walking simulations we performed tend to show that the foot velocity is almost always different from zero when in contact with the ground. So, the contact parameters are, in practice, observable during robot walking.

Equations (8.2.20)-(8.2.21)-(8.2.22) are historically derived from the recursive least square theory [Hayes, 1996, Section 9.3] and [Ljung, 1999, Section 11.2]. However, we notice that the very same results can be obtained by designing a Kalman filter [Kalman, 1960][1] considering the following dynamical system

$$\pi_{t+1} = \pi_t, \tag{8.2.23}$$

[1]Considering a Linear Time Invariant discrete (LTI) dynamical system of the form

$$\begin{aligned} x_{k+1} &= F x_k + G u_k + w_k \\ z_k &= H x_k + v_k \end{aligned}$$

where x_k is the state of the system, u_k is the exogenous input, z_k is the measurement vector. w_k is the process noise vector that is assumed to be zero-mean Gaussian with covariance Q, i.e.,, $w_k \sim \mathcal{N}(0, Q)$. v_k is the measurement noise vector. It is assumed to be zero-mean Gaussian with covariance R, i.e.,, $v_k \sim \mathcal{N}(0, R)$. The Kalman filter [Kalman, 1960] is a recursive algorithm that estimates the state x_k starting from a series of measurements. It is possible to prove that the Kalman filter is the optimal observer in the case of linear dynamics and Gaussian noises w_k and v_k.

having as a measurement equation

$$_{B[\mathcal{I}]}\mathrm{f}[t] = \mathcal{Y}_t \pi_t + \psi_t, \tag{8.2.24}$$

where ψ is a white Gaussian noise having zero mean and covariance $\mathrm{E}\left[\psi_t \psi_t^\top\right] = \Gamma_t$.

8.3 Results

In this section, we present the simulation tests of the control strategy presented in Section 8.2. The proposed strategy is also compared with the task-based inverse dynamics algorithm that considers rigid contacts presented in Section 7.2. From now on, the proposed control approach is called *TSID-Compliant*, and the controller of Section 7.2 *TSID-Rigid*. The experiments are carried out on a simulated version of the humanoid robot iCub v2.7 [Metta et al., 2010] – Section 1.1.1. The architecture takes (on average) less than 1 ms to evaluate its outputs. The OSQP [Stellato et al., 2018a] library is used to solve the optimization problems. The code is open source it is available at `https://github.com/dic-iit/Romualdi-2021-RAL-soft_terrain_walking`. Simulations are obtained by integrating the robot forward dynamics (FD) obtained from (3.3.1). Figure 8.4 shows the connection between the whole-body controller the contact parameter estimator and the simulator. The FD is evaluated with the contact model in Lemma 1 perturbed with a zero-mean Gaussian noise.

To validate the performance of the proposed architecture, we present three main experiments. First, we compare the performances of the TSID-Compliant and the TSID-Rigid controllers in case of different contact parameters. Second, we analyze the robustness of the TSID-Compliant in case of non-parametric uncertainty in the contact model. Finally, we show the contact parameter estimation performances in case of anisotropic environment. In all scenarios, the robot walks straight and the maximum velocity is $0.17\,\mathrm{m\,s^{-1}}$.

8.3.1 Comparison between TSID-Compliant and TSID-Rigid

Table 8.1 summarizes the outcome of the control strategies for different contact parameters. Labels *success* and *failure* mean that the associated controller is able or not to ensure that the robot is balanced while walking.

To compare the two controllers, we decided to perform three main experiments. In the first experiment, we choose a set of contact parameters such that both whole-body

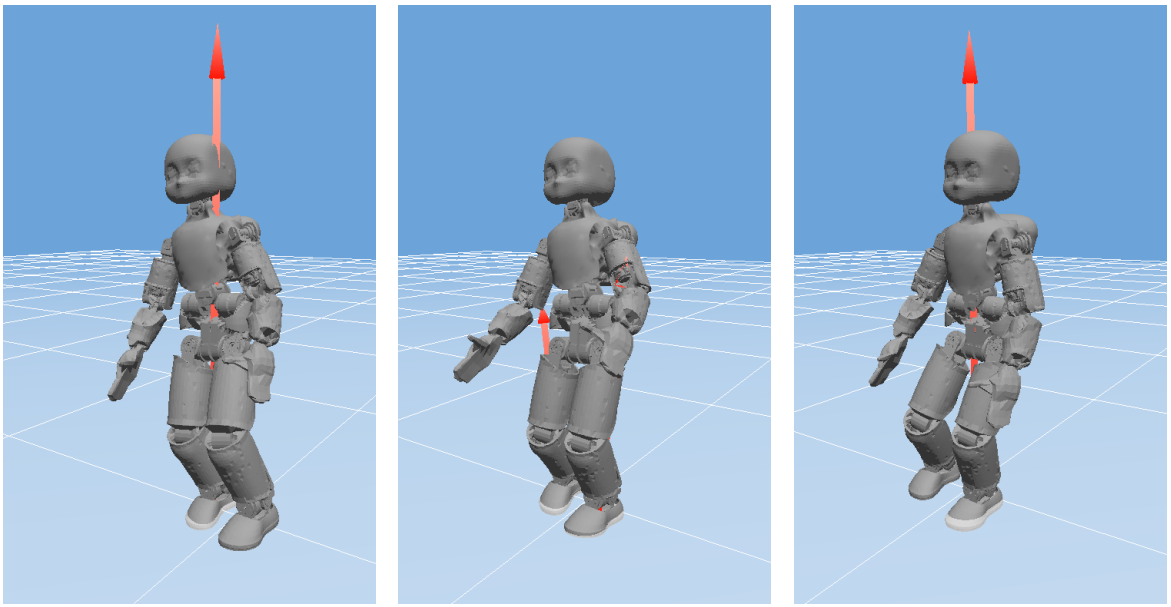


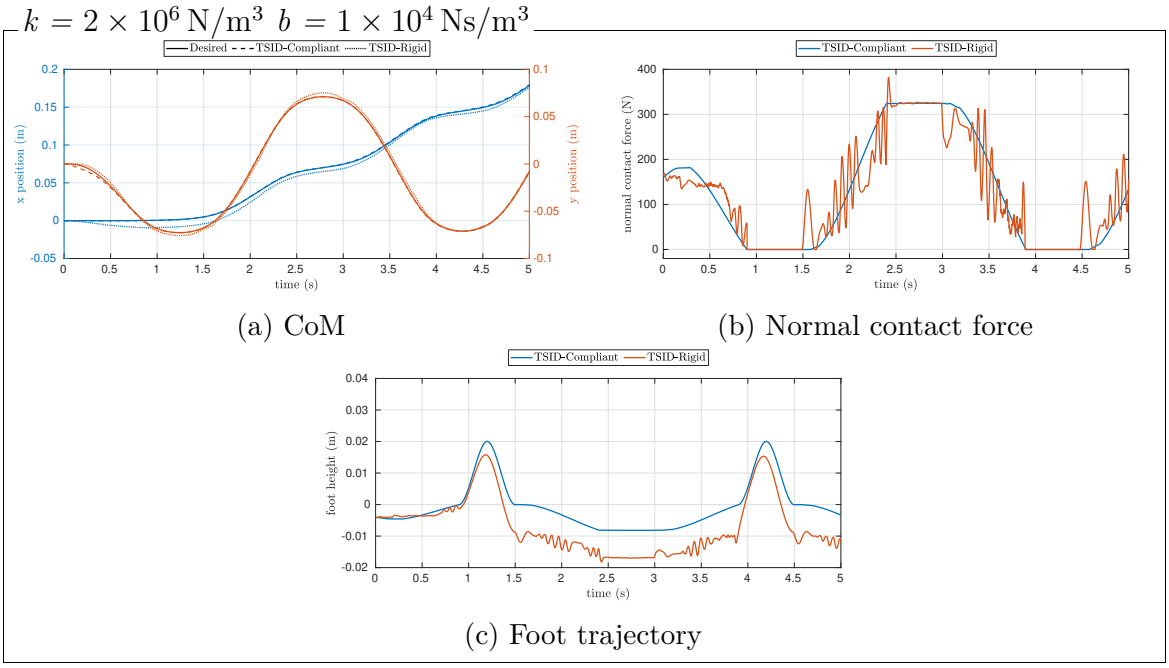
Figure 8.5 A simulation of the iCub robot walks with the TSID-Compliant controller.

controllers guarantee balance while walking. In the second, we keep the damper coefficient b constant, and we decrease the spring coefficient k. Finally, in the third experiment, we keep k constant and decrease b. Namely:

- **Experiment 1** $k = 2 \times 10^6\,\mathrm{N/m^3}$, $b = 1 \times 10^4\,\mathrm{Ns/m^3}$;
- **Experiment 2** $k = 1 \times 10^6\,\mathrm{N/m^3}$, $b = 1 \times 10^4\,\mathrm{Ns/m^3}$;

Table 8.1 Outcomes of whole-body controllers implementation on compliant terrain walking.

TSID Type	**k (N/m³)**	**b (Ns/m³)**	**Outcome**
Compliant	8×10^5	1×10^4	Success
Compliant	1×10^6	1×10^4	Success
Compliant	2×10^6	1×10^4	Success
Compliant	8×10^5	1×10^3	Success
Compliant	2×10^6	1×10^3	Success
Compliant	1×10^6	1×10^3	Success
Rigid	8×10^5	1×10^4	Failure
Rigid	1×10^6	1×10^4	Failure
Rigid	2×10^6	1×10^4	Success
Rigid	8×10^5	1×10^3	Failure
Rigid	1×10^6	1×10^3	Failure
Rigid	2×10^6	1×10^3	Failure



(a) CoM

(b) Normal contact force

(c) Foot trajectory

Figure 8.6 Comparison between TSID-Rigid and TSID-Compliant.

- **Experiment 3** $k = 2 \times 10^6\,\mathrm{N/m^3}$, $b = 1 \times 10^3\,\mathrm{Ns/m^3}$.

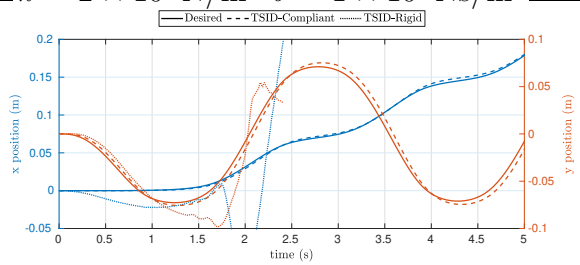
Experiment 1

Figure 8.6a depicts the CoM tracking performance obtained with the TSID-Compliant and the TSID-Rigid. Both controllers seem to show good tracking performances, and the CoM error is kept below 1 cm in both cases. Note that the TSID-Rigid controller induces faster variations in the measured contact wrenches – Figure 8.6b. This contributes to the overall higher vibrations of the robot. One reason for this behavior is that the TSID-Rigid assumes full control on the desired contact wrenches. This assumption is generally valid for stiff contacts, but it does not hold if the environment is compliant. Figure 8.6c presents the left foot trajectory when the whole-body controller is either TSID-Compliant or TSID-Rigid. The TSID-Compliant ensures a smoother foot motion when it is in contact with the environment ($1.5\,\mathrm{s} < t < 3.5\,\mathrm{s}$).

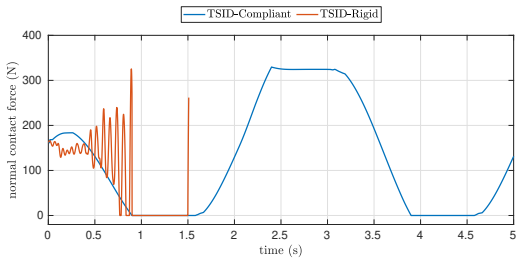
Experiment 2

There is no significant difference between the CoM tracking obtained with the two implementations of the whole-body controller – Figure 8.7a for $t < 1\,\mathrm{s}$. However,

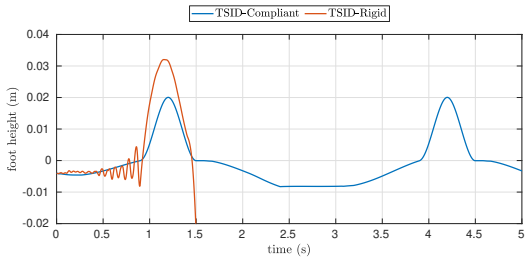
$k = 2 \times 10^6\,\mathrm{N/m^3}$ $b = 1 \times 10^3\,\mathrm{Ns/m^3}$



(a) CoM



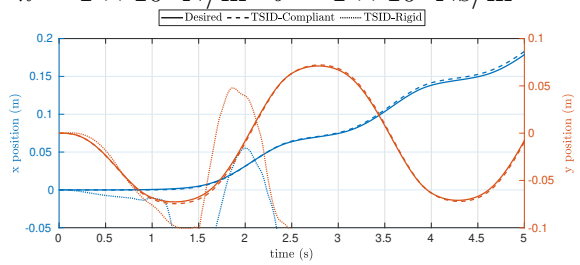
(b) Normal contact force



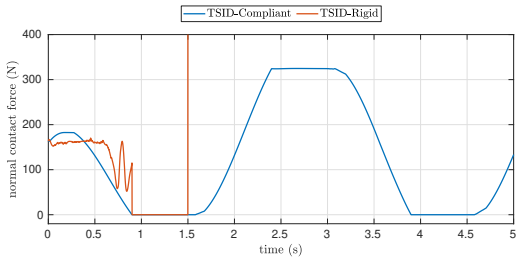
(c) Foot trajectory

Figure 8.7 Comparison between TSID-Rigid and TSID-Compliant. At $t \approx 1.75\,\mathrm{s}$, the TSID-Rigid makes the robot fall down.

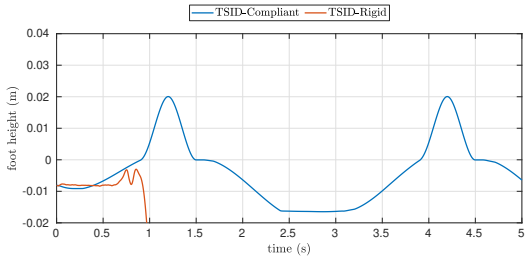
$k = 1 \times 10^6\,\mathrm{N/m^3}$ $b = 1 \times 10^4\,\mathrm{Ns/m^3}$



(a) CoM



(b) Normal contact force



(c) Foot trajectory

Figure 8.8 Comparison between TSID-Rigid and TSID-Compliant. At $t \approx 0.9\,\mathrm{s}$, the TSID-Rigid makes the robot fall down.

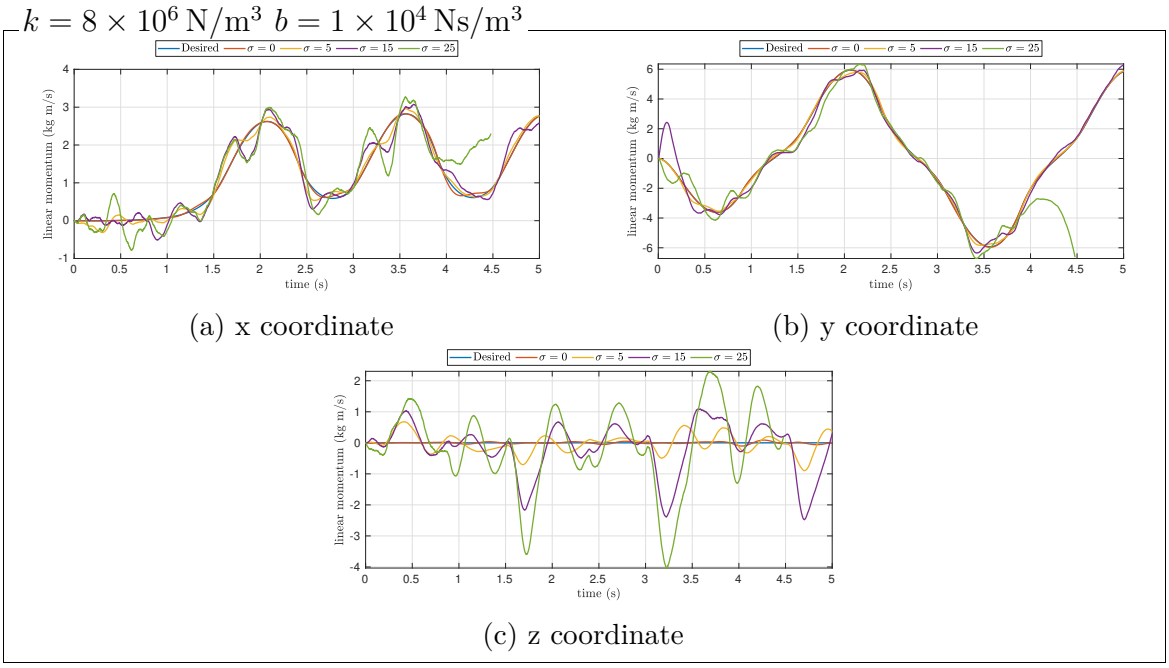


Figure 8.9 Linear momentum tracking for different values of σ. At $t \approx 4\,\mathrm{s}$ and $\sigma = 20$ the robot fall down.

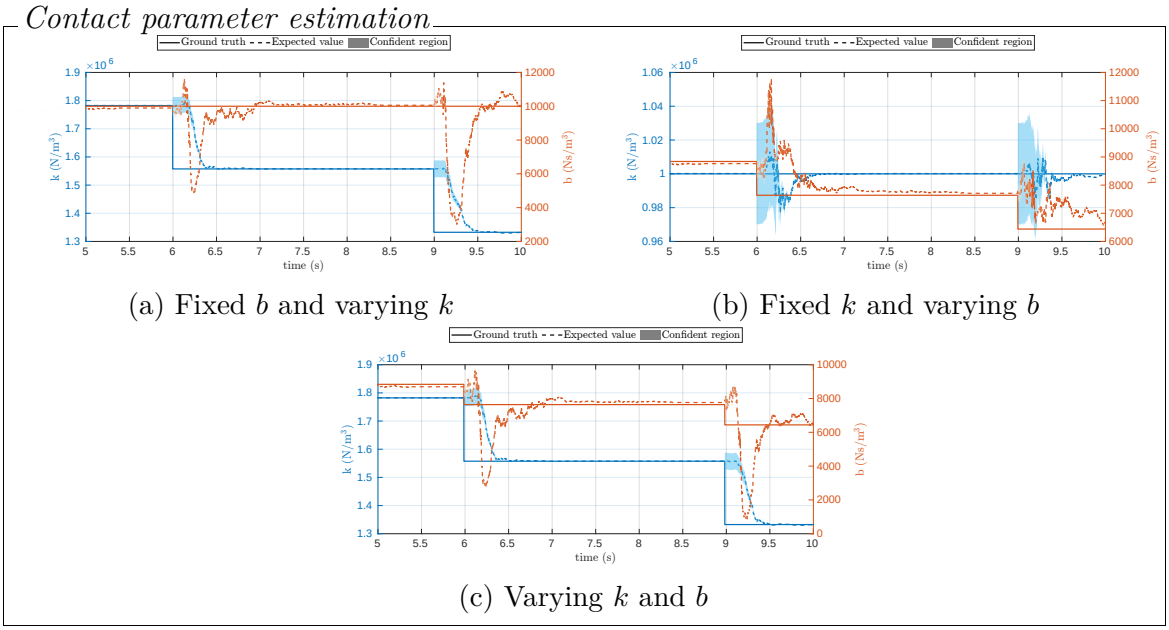


Figure 8.10 Estimation of the contact parameters. The contact wrench is perturbed with zero-mean Gaussian noise with $\sigma = 5$.

the lower the damper parameter b, the higher the acceleration required to change

the contact wrench. This contrasts with the assumption of zero-foot accelerations of the TSID-Rigid controller – see Equation (3.3.7). Consequently, the TSID-Rigid generates fast vibrations of the measured contact wrench and on the foot position – Figures 8.7b and 8.7c, respectively. Clearly, these poor performances induce a bad tracking of the CoM shown in Figure 8.7a at $t \approx 1.5\,\mathrm{s}$, and consequently the robot falls down. A possible solution to mitigate this problem is to increase the CoM gain in the TSID-Rigid controller. This unfortunately gives rise to higher robot oscillations, which in turn degrade the CoM tracking, and still the robot falls, or even worse, the controller may not be able to find a feasible solution.

Experiment 3

The CoM tracking problem presented in Experiment 2 worsens at lower values of the contact parameter k. Figure 8.8a shows the CoM tracking performances of the two controllers. The TSID-Compliant is still capable of ensuring good performance. On the other hand, the TSID-Rigid generates faster variations on the measured contact wrenches and foot positions – Figures 8.6b and 8.6c, respectively. Consequently, the controller, to maintain balance, requires high variations of the joint accelerations and at $t \approx 0.9\,\mathrm{s}$ fails while searching for feasible joint torques. Although tuning the TSID-Rigid controller may mitigate this problem, we were only able to postpone the failure.

8.3.2 Robustness of the TSID-Compliant

In this section, we present the robustness capabilities of the TSID-Compliant controller in case of non-parametric uncertainty in the contact model.

We model the non-parametric uncertainty by using an *additive white Gaussian noise*. So, the contact wrench acting on the robot becomes $f_k^m = f_k + \epsilon$, where ϵ is sampled from a Gaussian distribution with zero mean and standard deviation σ. f is the contact wrench computed with the contact model (B.0.7). Figure 8.9 shows the linear momentum tracking performance obtained with different values of σ. The experiments are performed in the case of $k = 8 \times 10^6\,\mathrm{N/m^3}$ and $b = 1 \times 10^4\,\mathrm{Ns/m^3}$, however, similar considerations hold for the other sets of parameters – see Table 8.1. The higher σ, the higher is the tracking error. The controller is capable of guaranteeing good tracking performance for all $\sigma < 20$. Indeed, at $\sigma = 20$ the controller is no longer able to guarantee acceptable performances and at $t \approx 4\,\mathrm{s}$ the robot falls down.

8.3.3 Anisotropic environment

In this section, we show the performance of the contact parameter estimator in the case of an anisotropic environment. We model the contact parameters as a piecewise function of the forward walking direction assuming that all points of the contact surface have the same contact parameters. Also, to further test the robustness of the estimator, we add a zero-mean Gaussian noise (with standard deviation $\sigma = 5$) to the contact wrench. To deal with contact parameter discontinuities, we reset the state covariance in the recursive least squares algorithm [Colman and Wells, 2006]. Each time a foot impacts the ground, the covariance is reset to the initial value. Figure 8.10a shows the performance of the estimator in the case of $b = 1 \times 10^4\,\mathrm{Ns/m^3}$ and a space-dependent k. The discontinuity on the ground truth ($t \approx 6\,\mathrm{s}$ and $t \approx 9\,\mathrm{s}$) occurs when the swing foot makes contact with the environment. The observed parameters converge to the ground truth in less than a second. The accompanying video shows that during the transient phases (e.g. $t \approx 6.5\,\mathrm{s}$), the controller achieves good overall performance. Similar considerations hold in the case $k = 1 \times 10^6\,\mathrm{N/m^3}$ and varying b – Figure 8.10b and space varying k and b – Figure 8.10c.

8.4 Conclusions

This chapter presents the development of a contact model to represent the interaction between the robot and the environment. Unlike state-of-the-art models, we consider the environment as a continuum of spring-damper systems. This allows us to compute the equivalent contact wrench by integrating the pressure distribution over the contact surface. As a result, rotational springs and dampers are not required to model the interaction between the robot and the environment. We also develop a whole-body controller that stabilizes the robot while walking in a compliant environment. Finally, an estimation algorithm is also introduced to compute the contact parameters in real-time. The proposed controller is then compared with the whole-body controller presented in Chapter 7. We analyze the robustness properties of the architecture with respect to non-parametric uncertainty in the contact model. Finally, we study the performance of the contact parameter estimation in the case of an anisotropic environment. It is worth mentioning that the controller introduced in this chapter assumes that the flexibility is located in the surrounding environment, while the robot is considered rigid. In Chapter 9 we will consider that the robot link deforms during

the locomotion task, while the environment is considered rigid. We characterize the link flexibility by introducing equivalent passive joints where the link deflection is concentrated.

Chapter 9

Whole-Body Control of Humanoid Robots with Link Flexibility

In Chapter 7, we presented a whole-body controller for humanoid robots in contact with a rigid environment. We then extended the controller in the case of visco-elastic walking surface – Chapter 8. However, both architectures assume that the links of the robot do not deform during the locomotion task. This hypothesis is generally valid; however, it may happen that one of the links of the system flexes while walking. In this chapter, we attempt to loosen the rigid body hypothesis that has been considered in Chapters 7 and 8. More specifically, we extend the whole-body controller introduced in Section 7.2 in the case of a humanoid robot affected by undesired link flexibility.

We characterize the link flexibility by introducing equivalent passive joints where the link deflection is concentrated [Nakaoka et al., 2007]. We extend the robot state to consider the underactuated flexible joints in the model. Thanks to this choice, we are able to design a whole-body controller that implicitly considers the deformation of the joints. Since in our case the deflection is not directly measurable, we design an observer aiming at estimating the flexible joint state, namely position, velocity, and torque, only considering the measured contact force and the actuated joint state. We validate the overall approach on a simulated version of the humanoid robot TALOS, since its hip flexibility has a significant impact when performing a locomotion task [Villa et al., 2022]. To address the elasticity of the robot link, the authors of [Villa et al., 2022] locally compensate the effect of deflections by modifying the measured position and velocity of some actuated joints considered in the whole-body controller. By extending the robot state with passive joints, our approach automatically considers the link flexibility in the control stabilization problem.

To further test the proposed strategy, we compare the whole-body control presented in this chapter with the approach discussed in Section 7.2. In this respect, we investigate the flexibility of the link that makes classical approaches fail. Finally, we analyze the performance of the presented control design with respect to different values of the stiffness parameter.

The chapter is organized as follows. Section 9.1 details the model used to characterize the flexibility of the link and extends the humanoid robot model to account for it. Section 9.2 discusses the whole-body controller. Section 9.3 contains the design of an observer to estimate the flexible joint state online. Section 9.4 presents the simulation results on the TALOS humanoid robot. Finally, Section 9.5 concludes the chapter. The content of this chapter has been carried out during my Ph.D. secondment in the *Gepetto* laboratory of the *LAAS-CNRS Laboratory for Analysis and Architecture of Systems* in Toulouse, France. The control architecture presented in this chapter is the subject of a publication to be submitted:

> Romualdi, G., Villa, N., Dafarra, S., Pucci, D., and Stasse, O. (2022b). Control and Estimation of Link Flexibility for Humanoid Robot Motion Control. *IEEE-RAS International Conference on Humanoid Robots (Humanoids) (Submitted)*

9.1 System modeling

TALOS's hip flexibility has a significant impact on its leg control and, as a result, its balance and locomotion [Ramuzat et al., 2021]. In this section, we model the TALOS's hip flexibility by means of underactuated joints. The section also extends the humanoid robot model dynamics presented in Section 3.3 to consider the robot's link visco-elasticity.

9.1.1 Model of the hip flexibility

Following the work of Villa et al. [2022], we model the flexibility by introducing two passive virtual joints between the base link and each leg. The virtual joints simulate the motion caused by the visco-elastic deformation of the waist-leg connection, where the link cross section is reduced. Given the i-th passive joint, we assume that it exerts

a torque τ_i^f that depends on the joint deflection s_i^f and its velocity $\dot{s}_i^f$ [Nakaoka et al., 2007] as

$$\tau_i^f = -k_i s_i^f - d_i \dot{s}_i^f. \tag{9.1.1}$$

where k_i and d_i are, respectively, the stiffness and damping coefficients of the flexible joint i. By assuming to model the link flexibility with n_f joints, we can consider the robot to have n joints where $n = n_a + n_f$ with n_a are the actuated joints.

In the specific case of the TALOS robot, the authors of [Villa et al., 2022] notice that the stiffness is due to the vertical linkage and, consequently, they model the flexibility along the pitch and roll axis, only. In this chapter, the same consideration holds, so we introduce two passive flexible joints for each leg. As a result $n_f = 4$ while $n_a = 32$ – see Section 1.2.

Assuming that it is possible to estimate τ_i^f, we approximate the flexible joint state by discretizing Equation (9.1.1), resulting in

$$s_i^f[k] = \frac{d_i s_i^f[k-1] - \tau_i^f[k]\,\mathrm{d}\,t}{k_i\,\mathrm{d}\,t + d_i}, \tag{9.1.2a}$$

$$\dot{s}_i^f[k] = \frac{s_i^f[k] - s_i^f[k-1]}{\mathrm{d}\,t}. \tag{9.1.2b}$$

Here $\mathrm{d}\,t$ is the sampling time. $s_i^f[k] = s_i^f(t_0 + k\,\mathrm{d}\,t)$ and $\dot{s}_i^f[k] = \dot{s}_i^f(t_0 + k\,\mathrm{d}\,t)$.

9.1.2 Modeling of a floating base system with flexible joints

This section extends the floating base system model presented in Section 3.3 to consider underactuated flexible joints.

Let us consider an inertial frame $\mathcal{I}$ and a floating base system making n_c contact with the environment. We recall:

- $B = (p_B, [B])$ is the frame rigidly attached to the robot base. ${}^{\mathcal{I}}H_B \in \mathrm{SE}(3)$ describes the position and orientation of B with respect to the inertial frame $\mathcal{I}$.
- Hereafter the base velocity expressed in mixed representation ${}^{B[\mathcal{I}]}\mathrm{v}_{\mathcal{I},B}$ such that ${}^{B[\mathcal{I}]}\mathrm{v}_{\mathcal{I},B}^\top = \begin{bmatrix} {}^{\mathcal{I}}\dot{p}_B^\top & {}^{B[\mathcal{I}]}\dot{\omega}_{\mathcal{I},B}^\top \end{bmatrix}$.
- The actuated and flexible joint positions are indicated, respectively, with $s^a \in \mathbb{R}^{n_a}$ and $s^f \in \mathbb{R}^{n_f}$;
- the actuated and flexible joint torques are indicated, respectively, with $\tau^a \in \mathbb{R}^{n_a}$ and $\tau^f \in \mathbb{R}^{n_f}$.

We extend the robot configuration (3.2.8) by introducing the flexible joint value as $q = ({}^{\mathcal{I}}p_B, {}^{\mathcal{I}}R_B, s^a, s^f)$. q is an element of a Lie group $\mathcal{Q} = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_f}$. The associated lie Algebra writes as $\mathfrak{q} = \mathbb{R}^3 \times \mathfrak{so}(3) \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_f}$. Here, we recall that $\mathfrak{q}$ is isomorphic to $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_f}$ – see Appendix A.3.

The *velocity of the multi-body system* belongs to $\mathfrak{q}$ and we denote it with $\nu = \left({}^{\mathcal{I}}\dot{p}_B, {}^{B[\mathcal{I}]}\omega_{\mathcal{I},B}, \dot{s}^a, \dot{s}^f\right)$.

Slightly modifying (3.3.3), we write the dynamics of a floating base system with flexible joints as follows

$$M(q)\dot{\nu} + h(q,\nu) = \begin{bmatrix} 0_{6\times n_a} \\ I_{n_a} \\ 0_{n_f\times n_a} \end{bmatrix} \tau^a + \begin{bmatrix} 0_{6\times n_f} \\ 0_{n_a\times n_f} \\ I_{n_f} \end{bmatrix} \tau^f + J_{\mathcal{C}}(q)^\top \mathrm{f}, \tag{9.1.3}$$

where

$$J_{\mathcal{C}}(q) = \begin{bmatrix} J_{\mathcal{C}_1}(q) \\ \vdots \\ J_{\mathcal{C}_{n_c}}(q) \end{bmatrix}, \quad \mathrm{f} = \begin{bmatrix} {}_{\mathcal{C}_1[\mathcal{I}]}\mathrm{f}_1 \\ \vdots \\ {}_{\mathcal{C}_{n_c}[\mathcal{I}]}\mathrm{f}_{n_c} \end{bmatrix}. \tag{9.1.4}$$

Recalling that $n = n_a + n_f$, $M \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix, $h \in \mathbb{R}^{(n+6)}$ accounts for Coriolis, the centrifugal effects and the gravity term. ${}_{\mathcal{C}_k[\mathcal{I}]}\mathrm{f}_k \in \mathbb{R}^6$ denotes the k-th external wrench applied by the environment on the robot expressed in mixed representation. The Jacobian $J_{\mathcal{C}_k}$ is the mixed velocity Jacobian of the contact $\mathcal{C}_k$.

Following the same approach presented in Section 3.3, the dynamics (9.1.3) is expressed by separating the first 6 rows, which refers to the underactuated floating base, from the last rows $n_a + n_f$, which refers to the actuated and flexible joints as:

$$M_\nu(q)\dot{\nu} + h_\nu(q,\nu) = J_{\mathcal{C}_\nu}^\top(q)\ \mathrm{f}, \tag{9.1.5a}$$

$$M_s(q)\dot{\nu} + h_s(q,\nu) = \begin{bmatrix} I_{n_a} \\ 0_{n_f\times n_a} \end{bmatrix} \tau^a + \begin{bmatrix} 0_{n_a\times n_f} \\ I_{n_f} \end{bmatrix} \tau^f + J_{\mathcal{C}_s}^\top(q)\ \mathrm{f}. \tag{9.1.5b}$$

The subscript ν refers to the first 6 rows of the matrix, while s refers to the last n rows. Equation (9.1.5) is often denoted as the base projection of the floating base dynamics, while Equation (9.1.6) is the joint space projection.

9.2 Whole-body Controller

Similar to what is discussed in Sections 7.2 and 8.2, the goal of the whole-body controller is to guarantee the tracking of desired kinematic quantities while ensuring the feasibility of the contact forces. In this section, we present an extension of the dynamics-based whole-body controller of Section 7.2 that considers the floating base dynamics in the case of under-actuated flexible joints – Equation (9.1.3).

The proposed controller computes the desired robot actuated joint torques τ^a, the generalized acceleration ${}^{B[\mathcal{I}]}\dot{\nu}$, and a set of desired spatial contact forces expressed in mixed representation ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$. Following the same approach as in Section 7.2, we formulate the control problem using the stack of tasks approach. We transcribe the optimal control problem into a constrained optimization problem. Here, we consider the low priority tasks as the terms of the cost function, while the high priority tasks are modeled as constraints.

9.2.1 Low and high priority tasks

This section introduces the list of low and high priority tasks considered in the optimal control problem.

Centroidal momentum task

Given a frame $\bar{G} = (x_{\mathrm{CoM}}, [\mathcal{I}])$, we introduce the centroidal momentum task as in Section 7.2.1:

$$\Psi_h = {}_{\bar{G}}\dot{h}^* - A_c\mathrm{f} - m\bar{g}. \tag{9.2.1}$$

where m is the robot mass, $\bar{g} = \begin{bmatrix}0 & 0 & -g & 0 & 0 & 0\end{bmatrix}^\top$ is the 6D gravity acceleration. A_c is the matrix containing the co-adjoint transformations ${}_{\bar{G}}X^{\mathcal{C}_i[\mathcal{I}]}$, i.e.

$$A_c = \begin{bmatrix}{}_{\bar{G}}X^{\mathcal{C}_1[\mathcal{I}]} & \dots & {}_{\bar{G}}X^{\mathcal{C}_{n_c}[\mathcal{I}]}.\end{bmatrix} \tag{9.2.2}$$

${}_{\bar{G}}\dot{h}^*$ is chosen considering (7.2.4) and (7.2.5) as:

$${}_{\bar{G}}\dot{h}^* = \begin{bmatrix}m\ddot{x}^{\mathrm{ref}}_{\mathrm{CoM}} \\ {}_{\bar{G}}\dot{h}^{\omega^{\mathrm{ref}}}\end{bmatrix} + \begin{bmatrix}mK^d_{\mathrm{CoM}} & 0_{3\times3} \\ 0_{3\times3} & K_{h^\omega}\end{bmatrix}\begin{bmatrix}\dot{x}^{\mathrm{ref}}_{\mathrm{CoM}} - \dot{x}_{\mathrm{CoM}} \\ {}_{\bar{G}}h^{\omega^{\mathrm{ref}}} - {}_{\bar{G}}h^{\omega}\end{bmatrix} + \begin{bmatrix}mK^p_{\mathrm{CoM}} \\ 0_{3\times3}\end{bmatrix}\left(x^{\mathrm{ref}}_{\mathrm{CoM}} - x_{\mathrm{CoM}}\right). \tag{9.2.3}$$

In our scenario, the desired centroidal quantities $x^{\mathrm{ref}}_{\mathrm{CoM}}$ and ${}_{\bar{G}}h^{\omega}$ are provided by a high-level planner.

Cartesian task

Similar to what was discussed in Section 7.2.1, the Cartesian task is implemented as:

$$\Psi_{L_{\mathrm{SE}(3)}} = {}^{L[\mathcal{I}]}\dot{\mathrm{v}}^{*}_{\mathcal{I},L} - J_L\dot{\nu} - \dot{J}_L\nu \tag{9.2.4}$$

where ${}^{L[\mathcal{I}]}\dot{\mathrm{v}}^{*}_{\mathcal{I},L} = \begin{bmatrix}\ddot{p}^{*\top}_{L} & {}^{\mathcal{I}}\dot{\omega}^{*\top}_{\mathcal{I},L}\end{bmatrix}^{\top}$ is chosen as (7.2.11). Similarly, we recall that the positional and rotational tasks are given by Equations (7.2.12) and (7.2.13). As we discuss in more depth in Section 9.2.2, we apply this task to stabilize the orientation of the chest and the root and the pose of the feet.

Floating base dynamics task

If the robot is equipped with under-actuated flexible joints, the whole-body controller should consider the measured (or estimated) joint torques acting on the flexibility. To do so, we modify the floating base dynamics task presented in Section 7.2.1. We project the dynamics (9.1.3) into the base and joint subspaces, and we name the projection as *base dynamics* and *joint dynamics* – see Section 3.3. We define the base dynamics constraint as (7.2.14)

$$\Psi_{\mathrm{dyn}_\nu} = h_\nu + M_\nu\dot{\nu} - J^{\top}_{\mathcal{C}_\nu}\mathrm{f}. \tag{9.2.5}$$

Equation (9.2.5) does not depend on the flexible joint state. The joint dynamics task is given by

$$\Psi_{\mathrm{dyn}_s} = h_s + M_s(q)\dot{\nu} - \begin{bmatrix} I_{n_a} \\ 0_{n_f\times n_a}\end{bmatrix}\tau^a - \begin{bmatrix} 0_{n_a\times n_f} \\ I_{n_f}\end{bmatrix}\tau^f - J^{\top}_{\mathcal{C}_s}\mathrm{f}. \tag{9.2.6}$$

The subscript ν refers to the first six rows of the matrix, while s refers to the last $n_a + n_f$ rows. We notice that the last n_f rows of (9.2.6) represent the underactuated dynamics due to joint flexibility.

Joint position regularization task

To prevent the controller from computing solutions that generate a huge variation in joint acceleration, we introduce a joint regularization task for both the actuated and

flexible joints, as

$$\Psi_{s_a} = \ddot{s}_a^* - \begin{bmatrix} 0_{n_a \times 6} & I_{n_a} & 0_{n_a \times n_f} \end{bmatrix} \dot{\nu} \tag{9.2.7a}$$

$$\Psi_{s_f} = \ddot{s}_f^* - \begin{bmatrix} 0_{n_f \times 6} & 0_{n_f \times n_a} & I_{n_f} \end{bmatrix} \dot{\nu}, \tag{9.2.7b}$$

with $\ddot{s}_a^*$ is equal to

$$\ddot{s}_a^* = \ddot{s}_a^{\text{ref}} + k_{s_a}^d(\dot{s}_a^{\text{ref}} - \dot{s}_a) + k_{s_a}^p(s_a^{\text{ref}} - s_a). \tag{9.2.8}$$

where s_a^{ref} is the desired joint trajectory provided by a high-level planner. $k_{s_a}^d$ and $k_{s_a}^p$ are two positive-defined diagonal matrices. On the other hand, assuming that we estimate the state of the flexible joints, we ask for $\ddot{s}_f^*$ equal to

$$\ddot{s}_f^* = -k_{s_f}^d \dot{s}_f - k_{s_f}^p s_f. \tag{9.2.9}$$

where $k_{s_f}^d$ and $k_{s_f}^p$ are two defined positive diagonal matrices. Thanks to (9.2.9) the controller tries to stabilize the flexible joint position to zero.

Joint torque regularization task

In order to prevent the controller from providing solutions with large actuated joint torques, we introduce the following task:

$$\Psi_\tau = \tau_a^{\text{ref}} - \tau_a, \tag{9.2.10}$$

where, in our case, τ_a^{ref} is provided by a high-level planner.

Feasibile contact force task

The feasibility of the contact wrench ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$ is guaranteed by the set of inequalities introduced in Equation (7.2.26):

$$\Phi_{\mathrm{f}_j} : \quad A_{\mathcal{C}_j[\mathcal{I}]}\, {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j - b \preceq 0. \tag{9.2.11}$$

we recall that $A_{\mathcal{C}_j[\mathcal{I}]}$ depends on the robot generalized state $q \in \mathcal{Q}$.

9.2.2 Quadratic programming problem

Following the same approach as in Section 7.2.2, we achieve the control objective by transcribing the control problem as a constrained quadratic programming problem. Here, we consider the contact forces ${}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j$, the base acceleration ${}^{B[\mathcal{I}]}\dot{\mathrm{v}}_{\mathcal{I},B}$, the actuated joint acceleration $\ddot{s}_a$, and the actuated joint torques τ_a as conditional variables.

The tracking of the left and right feet are considered high-priority SE(3) tasks (9.2.4) and they are denoted respectively as $\Psi_{L_{\mathrm{SE(3)}}}$ and $\Psi_{R_{\mathrm{SE(3)}}}$. We take into account the centroidal momentum tracking as a high priority task (9.2.1). The desired centroidal quantities $x^{\mathrm{ref}}_{\mathrm{CoM}}$ and $h^{\mathrm{ref}}_{\omega}$ in (9.2.3) are provided by a high-level planner that assumes that all robot joints are fully actuated. We also consider the base (9.2.5) and joints dynamics (9.2.6) as high priority tasks. To prevent the controller from asking for a high motion of the upper body while stabilizing the CoM, we introduce two SO(3) tasks, one associated with the chest and the other with the waist orientations, respectively, denoted $\Psi_{T_{\mathrm{SO(3)}}}$ and $\Psi_{R_{\mathrm{SO(3)}}}$. In both cases, we ask to keep the z coordinates of the link frames parallel to the gravity vector g. The postural conditions of the actuated and flexible joints (9.2.7) are considered low priority tasks. We regularize the desired actuated joint torques as a low priority task Ψ_τ (9.2.10). Here τ^{ref}_a is provided by a high-level planner. Finally, to guarantee feasible contact forces for the feet, we add the task (9.2.11), denoted respectively as Φ_{f_L} and Φ_{f_R}.

The above hierarchical control objectives can be cast into an optimization problem described by the following formulation:

$$
\begin{aligned}
\underset{{}^{B[\mathcal{I}]}\mathrm{v}_{\mathcal{I},B},\, \ddot{s}_a,\, \tau_a,\, \mathrm{f}}{\text{minimize}} \quad & \Psi^\top_{T_{\mathrm{SO(3)}}} \Lambda_T \Psi_{T_{\mathrm{SO(3)}}} + \Psi^\top_{R_{\mathrm{SO(3)}}} \Lambda_R \Psi_{R_{\mathrm{SO(3)}}} && \text{(9.2.12a)}\\
& + \Psi_s^\top \Lambda_s \Psi_s + \Psi_\tau^\top \Lambda_\tau \Psi_\tau && \text{(9.2.12b)}\\
\text{subj. to} \quad & \Psi_{L_{\mathrm{SE(3)}}} = 0 && \text{(9.2.12c)}\\
& \Psi_{R_{\mathrm{SE(3)}}} = 0 && \text{(9.2.12d)}\\
& \Psi_h = 0 && \text{(9.2.12e)}\\
& \Psi_{\mathrm{dyn}_\nu} = 0 && \text{(9.2.12f)}\\
& \Psi_{\mathrm{dyn}_s} = 0 && \text{(9.2.12g)}\\
& \Phi_{\mathrm{f}_L} && \text{(9.2.12h)}\\
& \Phi_{\mathrm{f}_R} && \text{(9.2.12i)}
\end{aligned}
$$

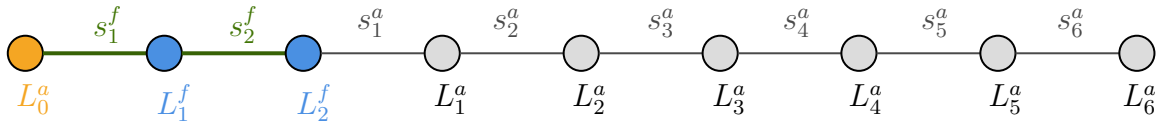


Figure 9.1 Schematic representation of the flexible TALOS leg. The links are represented by the graphs node, while the joints are the arcs. L_0^a is the robot waist, L_1^f and L_2^f the flexible links, L_i^a and s_i^a with $1 \leq i \leq 6$ are respectively the robot link and the actuated joints

Following the same considerations as in Sections 7.2.2 and 8.2.2, we transcribe the optimization problem (9.2.12) into a quadratic programming problem (Section 5.4) and we solve it via an off-the-shelf solver.

9.3 Flexible Joint State Observer

The optimal control problem presented in Section 9.2 assumes the knowledge of the state of the flexible joints, namely position, velocity, and torque. In a simulated environment, these values are perfectly known. However, in the real scenario, an estimation algorithm is required to compute them.

In this section, we discuss an algorithm that estimates the flexible joint torques, position, and velocity considering the measured contact wrenches that act on the robot soles and the actuated joint state. For simplicity, we present the algorithm considering the flexibility of the TALOS link. Given the symmetry of the robot structure, we analyze the approach only for one leg. Even if we approach the problem considering the TALOS use case, we want to underline that the proposed estimation can also be applied to other humanoid robots affected by link flexibility.

For TALOS, we know that each leg is made up of 8 joints, 2 of which are flexible and the other 6 are actuated. Furthermore, we assume that the flexible joints are located near the waist. As discussed in Section 3.1, we model the floating base multi-body system as a kinematic three. Figure 9.1 presents a schematic representation of the TALOS leg structure. Exploring the three from the waist to the sole, we first visit the flexible joints, and then the actuated joints. From now on, we denote by L_0^a the waist of the robot, while L_1^f and L_2^f are the fictitious flexible links required to connect the flexible joints s_1^f and s_2^f. L_i^a and s_i^a with $1 \leq i \leq 6$ are the robot link and the actuated joints, respectively. Figure 9.2 presents the geometric model of the flexible robot leg. Each link is associated with a frame attached to the joint connecting the link to its

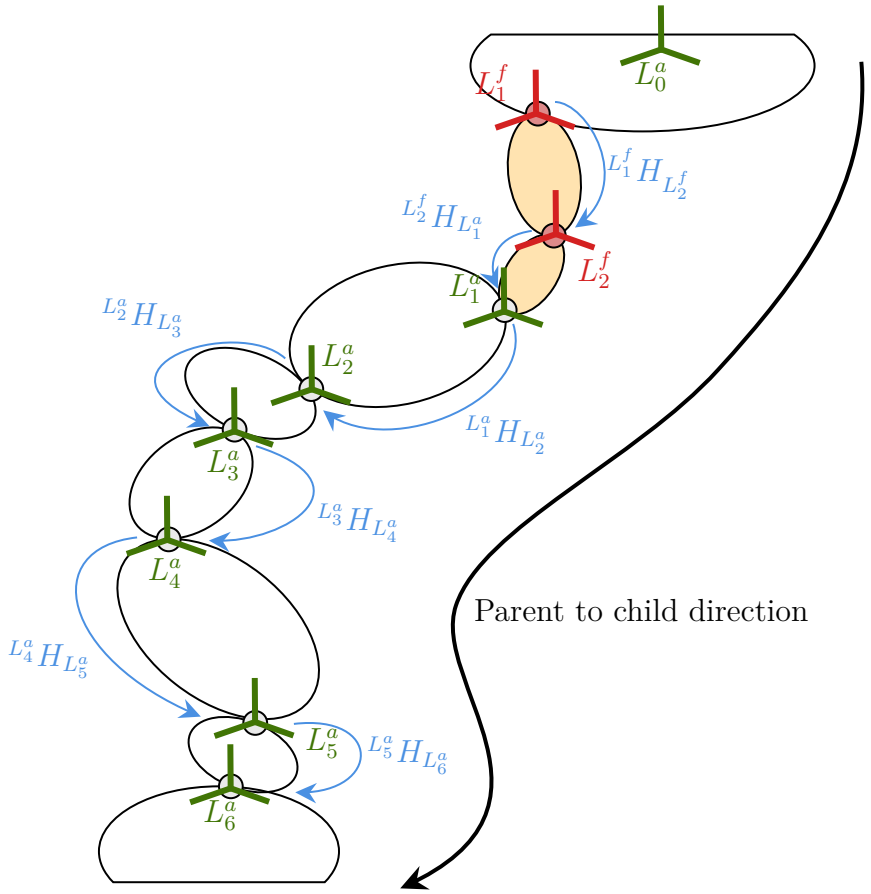


Figure 9.2 Geometric model of the flexible TALOS leg. Each link is associated with a frame attached to the joint connecting the link to its parent. The red frames are associated with the flexible links while the green frames to the robot links.

parent. The name of the frame associated with the link coincides with the name of the link.

Given one link of the leg chain, the associated rigid body dynamics is given by *Eurel-Poincaré Equations* (Equation (2.3.5)):

$$ {}_L\mathbb{M}_L {}^L\dot{\mathrm{v}}_{\mathcal{I},L} + {}^L\mathrm{v}_{\mathcal{I},L} \times^* {}_L\mathbb{M}_L {}^L\mathrm{v}_{\mathcal{I},L} = {}_L\mathbb{M}_L \begin{bmatrix} {}^{\mathcal{I}}R_L^\top g \\ 0_{3\times 1} \end{bmatrix} + \sum_{k=1}^{n_f} {}_L\mathrm{f}_k. \tag{9.3.1} $$

where L is the frame associated with the link. ${}_L\mathbb{M}_L$ is the 6D constant inertial matrix of the link (2.3.1). ${}^L\mathrm{v}_{\mathcal{I},L}$ is the left-trivialized spatial velocity (2.2.12). g is the gravity vector. ${}_L\mathrm{f}_k$ is a 6D spatial force acting on the link, and n_f are the forces acting on the link. We now assume that each link is subject to only two forces. One is exerted by the parent link, and the other is exerted by the child link. Given the chain structure of the

leg (Figure 9.1), each link has one parent and one child, except for the waist L_0^a and the foot sole L_6^a. With an abuse of notation, we consider the environment as the child of the sole. If the foot is in contact with the ground, one of the 6D forces acting on L_6^a is caused by the interaction with the environment. Since TALOS has a force-torque sensor mounted on the ankle, we assume the ground reaction force is measurable.

We now introduce the *proper sensor acceleration* as [Traversaro, 2017, Section 2.4.4]

$$\alpha^g_{\mathcal{I},L} := {}^{L}X_{L[\mathcal{I}]}{}^{L[\mathcal{I}]}\dot{\mathrm{v}}_{\mathcal{I},L} - \begin{bmatrix} {}^{\mathcal{I}}R_L^\top g \\ 0_{3\times 1} \end{bmatrix}. \tag{9.3.2}$$

Where ${}^{L}X_{L[\mathcal{I}]}$ is the adjoint matrix, i.e, ${}^{L}X_{L[\mathcal{I}]} = \mathbf{Ad}_{{}^{L}H_{L[\mathcal{I}]}}$ – see Equation (2.2.22). $\alpha^g_{\mathcal{I},L}$ is the acceleration obtained by an inertial measurement unit (IMU) aligned with L. The linear part is the readout of a linear accelerometer, and the angular part is the derivative of the output of a gyroscope.

Combining the proper sensor acceleration (9.3.2) with (9.3.1), we rewrite the body dynamics as

$${}_{L}\mathbb{M}_L\alpha^g_{\mathcal{I},L} + \begin{bmatrix} 0_{3\times 1} \\ {}_{L}\omega_{\mathcal{I},L} \end{bmatrix} \times^* {}_{L}\mathbb{M}_L \begin{bmatrix} 0_{3\times 1} \\ {}_{L}\omega_{\mathcal{I},L} \end{bmatrix} = \sum_{k=1}^{n_f} {}_{L}\mathrm{f}_k. \tag{9.3.3}$$

For convenience, we define ${}_{L}\phi_L\left(\alpha^g_{\mathcal{I},L}, {}^{L}\omega_{\mathcal{I},L}\right)$ as

$${}_{L}\phi_L\left(\alpha^g_{\mathcal{I},L}, {}^{L}\omega_{\mathcal{I},L}\right) = {}_{L}\mathbb{M}_L\alpha^g_{\mathcal{I},L} + \begin{bmatrix} 0_{3\times 1} \\ {}_{L}\omega_{\mathcal{I},L} \end{bmatrix} \times^* {}_{L}\mathbb{M}_L \begin{bmatrix} 0_{3\times 1} \\ {}_{L}\omega_{\mathcal{I},L} \end{bmatrix}. \tag{9.3.4}$$

We notice that ${}_{L}\phi_L\left(\alpha^g_{\mathcal{I},B}, {}^{B}\omega_{\mathcal{I},L}\right)$ depends on the acceleration, velocity, and inertial parameters of the body. Taking into account (9.3.4) we can finally rewrite the rigid body dynamics of the body L as

$${}_{L}\phi_L = \sum_{k=1}^{n_f} {}_{L}\mathrm{f}_k, \tag{9.3.5}$$

where, for the sake of clarity, we hide the dependencies ${}_{L}\phi_L$.

Recalling the structure of the leg chain, we denote by ${}_{L_i^a}\phi_{L_i^a}$ the terms associated with the robot link L_i^a connected to the parent through the actuated joint s_i^a and by ${}_{L_i^f}\phi_{L_i^f}$ the one associated with the fictitious flexible link L_i^f.

Algorithm 1 Forward kinematics

procedure FORWARDKINEMATICS(s^a, $\dot{s}^a$, $\ddot{s}^a$, ${}^{L_6^a}\omega_{\mathcal{I},L_6^a}$, $\alpha^g_{\mathcal{I},L_6^a}$)
  $n_a \leftarrow 6$ ▷ Actuated joints
  $i \leftarrow n_a$
  while $i \neq 0$ **do**
    if $i = n_a$ **then**
      $\boldsymbol{\omega}[i] \leftarrow {}^{L_6^a}\omega_{\mathcal{I},L_6^a}$
      $\boldsymbol{\alpha}[i] \leftarrow \alpha^g_{\mathcal{I},L_6^a}$
    else
      $\boldsymbol{\omega}[i] \leftarrow$ `ComputeAngularVelocity`$(s^a_{i+1}, \dot{s}^a_{i+1}, \boldsymbol{\omega}[i+1])$ ▷ Eq. (9.3.6)
      $\boldsymbol{\alpha}[i] \leftarrow$ `ComputeAcceleration`$(s^a_{i+1}, \dot{s}^a_{i+1}, \ddot{s}^a_{i+1}, \boldsymbol{\alpha}[i+1])$
    end if
    $i \leftarrow i - 1$
  end while
  return $\boldsymbol{\omega}, \boldsymbol{\alpha}$
end procedure

9.3.1 Forward kinematics

Given the angular velocity and the proper sensor acceleration of the foot sole L_6^a, we can compute ${}_{L_i^a}\phi_{L_i^a}$ recursively. Indeed, given the link L_i^a, we compute its body angular velocity as

$$ {}^{L_i^a}\omega_{\mathcal{I},L_i^a} = {}^{L_i^a}R_{L_{i+1}^a}(s^a_{i+1})\,{}^{L_{i+1}^a}\omega_{\mathcal{I},L_{i+1}^a} + {}^{L_i^a}\omega_{L_{i+1}^a,L_i^a} \tag{9.3.6} $$

where ${}^{L_{i+1}^a}\omega_{\mathcal{I},L_{i+1}^a}$ is the angular velocity of the child link. ${}^{L_i^a}\omega_{L_{i+1}^a,L_i^a}$ is the relative velocity of the link L_i^a with respect to L_{i+1}^a written in L_i^a. Since TALOS is equipped by revolute joints only ${}^{L_i^a}\omega_{L_{i+1}^a,L_i^a}$ depends only on the *left-trivialized joint motion subspace* ${}^{i+1}\mathbf{s}$ (Equation (3.2.12)) and on the joint velocity $\dot{s}^a_{i+1}$. Similarly, we can prove that the sensor proper acceleration $\alpha^g_{\mathcal{I},L_i^a}$ can be computed recursively by considering the child joint position s^a_{i+1} velocity $\dot{s}^a_{i+1}$ and acceleration $\ddot{s}^a_{i+1}$, and the child sensor proper acceleration $\alpha^g_{\mathcal{I},L_{i+1}^a}$ [Traversaro, 2017, Section 4.4.3].

Algorithm 1 summarizes the procedure required to compute the body angular velocities and proper accelerations. We notice that given a link, its velocity and acceleration depend only on the child link state and on the joint that connects the link to its child.

We want to stress that if an IMU is mounted on the robot sole, the angular velocity and the proper sensor acceleration of L_6^a can be derived from the sensor readouts. Otherwise, we propose two possible solutions: *i*) In the case of low swing foot velocity and acceleration, we suggest considering ${}^{L_6^a}\omega_{\mathcal{I},L_6^a}$ and $\alpha^g_{\mathcal{I},L_6^a}$ equal to zero. *ii*) Assuming

that an IMU is mounted on the robot waist. We suggest setting the proper acceleration of the foot $\alpha^g_{\mathcal{I},L^a_6}$ and the angular velocity ${}^{L^a_6}\omega_{\mathcal{I},L^a_6}$ at time t equal to the one computed by the forward kinematics that considers the waist angular velocity ${}^{B}\omega_{\mathcal{I},B}$ and the proper acceleration $\alpha^g_{\mathcal{I},B}$ at time t, the actuated joint position velocity and acceleration at time t and the estimated flexible joint position velocity and acceleration at $t - \mathrm{d}\,t$. With $\mathrm{d}\,t$ the control sampling period. To guarantee the convergence of the algorithm, the choice of $\mathrm{d}\,t$ becomes crucial. Here, we suggest setting $\mathrm{d}\,t$ small enough to capture the evolution of the system dynamics.

9.3.2 Inverse dynamics propagation

Assuming that the link proper acceleration $\alpha^g_{\mathcal{I},L^a_i}$ and angular velocity ${}^{L^a_i}\omega_{\mathcal{I},L^a_i}$ have been computed with Algorithm 1. Considering Equation (9.3.5) and assuming that each link is subject to two 6D forces, we write the rigid body dynamics for the link L^a_i as:

$$ {}_{L^a_i}\phi_{L^a_i} = {}_{L^a_i}\mathrm{f}_{\lambda(L^a_i),L^a_i} + {}_{L^a_i}\mathrm{f}_{L^a_{i+1},L^a_i}. \tag{9.3.7} $$

$\lambda(L^a_i)$ gives the parent link of L^a_i – see Section 3.1. ${}_{L^a_i}\mathrm{f}_{\lambda(L^a_i),L^a_i}$ is the spatial force exerted by the parent link $\lambda(L^a_i)$ to L^a_i whose coordinates are expressed in L^a_i. ${}_{L^a_i}\mathrm{f}_{L^a_{i+1},L^a_i}$ is the spatial force exerted by L^a_{i+1} to L^a_i expressed in L^a_i. We now reorganize Equation (9.3.7) to reveal the recursive structure of the algorithm:

$$ \begin{aligned} {}_{L^a_i}\mathrm{f}_{\lambda(L^a_i),L^a_i} &= {}_{L^a_i}\phi_{L^a_i} - {}_{L^a_i}\mathrm{f}_{L^a_{i+1},L^a_i} && \text{(9.3.8a)} \\ &= {}_{L^a_i}\phi_{L^a_i} + {}_{L^a_i}\mathrm{f}_{L^a_i,L^a_{i+1}} && \text{(9.3.8b)} \\ &= {}_{L^a_i}\phi_{L^a_i} + {}_{L^a_i}X^{L^a_{i+1}}\,{}_{L^a_{i+1}}\mathrm{f}_{L^a_i,L^a_{i+1}} && \text{(9.3.8c)} \end{aligned} $$

We notice that by projecting (9.3.8) into the *joint motion subspace* ${}^{i}\mathbf{s}$ we obtain the torque acting on the joint s^a_i, that is,

$$ \tau^a_i = {}^{i}\mathbf{s}^{\top}\,{}_{L^a_i}\mathrm{f}_{\lambda(L^a_i),L^a_i} \tag{9.3.9} $$

We recall that TALOS is equipped with joint torque sensors – see Section 1.2. As a consequence, τ^a_i can be directly measured. Using this information, we attempt to improve the estimation of the 6D force ${}_{L^a_i}\mathrm{f}_{\lambda(L^a_i),L^a_i}$ by considering the measured joint

Algorithm 2 Inverse Dynamics

procedure InverseDynamics($s^a, \beta, \tau^{a^{\text{meas}}}, \boldsymbol{\alpha}, \boldsymbol{\omega}$)
  $n_a \leftarrow 6$
  $i \leftarrow n_a$
  while $i \neq 0$ **do**
    ${}_{L_i^a}\phi_{L_i^a} \leftarrow$ `ComputePhi`$(\omega[i], \alpha[i], {}_{L_i^a}\mathbb{M}_{L_i^a})$
    if $i = n_a$ **then**
      ${}_{L_{i+1}^a}\mathrm{f}_{L_i^a, L_{i+1}^a} \leftarrow$ `GetSoleExternalWrench`()
    else
      ${}_{L_{i+1}^a}\mathrm{f}_{L_i^a, L_{i+1}^a} \leftarrow \boldsymbol{f}[i+1]$
    end if
    $\tau \leftarrow \tau_i^{a^{\text{meas}}}$
    ${}^i\mathbf{s} \leftarrow$ `GetMotionSubspace`(i)
    ${}_{L_i^a}X^{L_{i+1}^a} \leftarrow$ `GetCoAdjointMatrix`(s_i^a)
    $\boldsymbol{f}[i] \leftarrow$ `ComputeWrench`$(\beta, \tau, {}_{L_i^a}\phi_{L_i^a}, {}^i\mathbf{s}, {}_{L_{i+1}^a}\mathrm{f}_{L_i^a, L_{i+1}^a}, {}_{L_i^a}X^{L_{i+1}^a})$ ▷ Eq. (9.3.8)
    $i \leftarrow i - 1$
  end while
  return $\boldsymbol{f}$
end procedure

torque as follows:

$$
\begin{aligned}
{}_{L_i^a}\mathrm{f}_{\lambda(L_i^a), L_i^a} = & \left[(1-\beta)\tau_i^{a^{\text{meas}}} + \beta\, {}^i\mathbf{s}^\top \left({}_{L_i^a}\phi_{L_i^a} + {}_{L_i^a}X^{L_{i+1}^a}\, {}_{L_{i+1}^a}\mathrm{f}_{L_i^a, L_{i+1}^a}\right)\right] {}^i\mathbf{s} && \text{(9.3.10a)} \\
& + \left(I_6 - {}^i\mathbf{s}\, {}^i\mathbf{s}^\top\right) \left({}_{L_i^a}\phi_{L_i^a} + {}_{L_i^a}X^{L_{i+1}^a}\, {}_{L_{i+1}^a}\mathrm{f}_{L_i^a, L_{i+1}^a}\right). && \text{(9.3.10b)}
\end{aligned}
$$

Here, $\beta \in [0, 1]$ is a tunable parameter. To give the reader a better understanding, we notice that whether $\beta = 0$ the torque component parallel to the vector motion subspace is replaced by the readouts of the joint torque sensor. If $\beta = 1$ the joint torque sensor is not considered.

Algorithm 2 summarizes the procedure to compute the wrench acting on an actuated joint. We notice that given a joint s_i^a, the wrench acting on it depends only on the state of the child link.

9.3.3 Flexible joint state estimation

Applying the Algorithms 1 and 2 we can recursively compute the wrench acting on the joint s_1^a. Assuming a neglected mass and inertia for the flexible link L_2^f, i.e.,

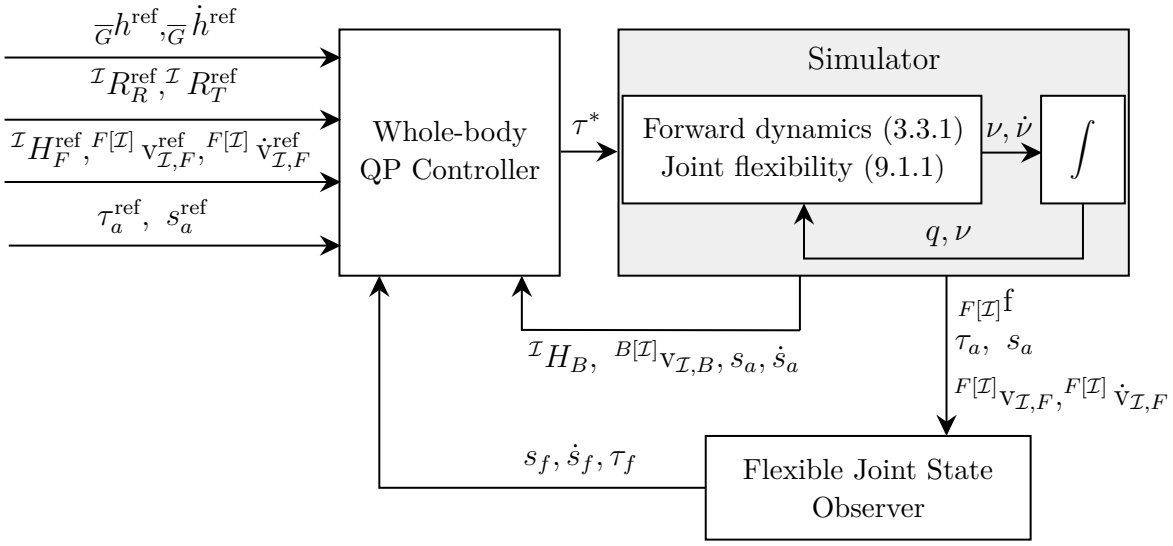


Figure 9.3 Flexible joint controller architecture.

$_{L_2^f}\mathbb{M}_{L_2^f} = 0_{6\times 6}$, we can write the flexible link dynamics as

$$
\begin{aligned}
{}_{L_2^f}\text{f}_{L_1^f,L_2^f} &= -{}_{L_2^f}\text{f}_{L_1^a,L_2^f} && \text{(9.3.11a)}\\
&= {}_{L_2^f}\text{f}_{L_2^f,L_1^a} && \text{(9.3.11b)}\\
&= {}_{L_2^f}X^{L_1^a}\,{}_{L_1^a}\text{f}_{L_2^f,L_1^a} && \text{(9.3.11c)}
\end{aligned}
$$

We notice that $_{L_2^f}X^{L_1^a}$ depends on the position of the joint s_1^a.

Given (9.3.11), we can compute the flexible joint torque τ_2^f by projecting $_{L_2^f}\text{f}_{L_1^f,L_2^f}$ onto the vector motion subspace

$$
\tau_2^f = \left({}_{L_2^f}\text{f}_{L_1^f,L_2^f}\right)^\top\ {}^2\mathbf{s}^f \tag{9.3.12}
$$

Combining (9.3.12) with the discretized flexible joint position (9.1.2a) we estimate the flexible joint position s_2^f. Applying the very same approach, we can estimate the flexible joint torque τ_1^f and the position s_1^f.

At each time step, applying Algorithms 1, 2 and Equations (9.3.11) and (9.3.12), we estimate the flexible joint state. The result is finally considered by the whole-body controller to compute the desired actuated joint torques. Figure 9.3 shows the connection between the whole-body controller, the flexible joint state estimator, and the simulator.

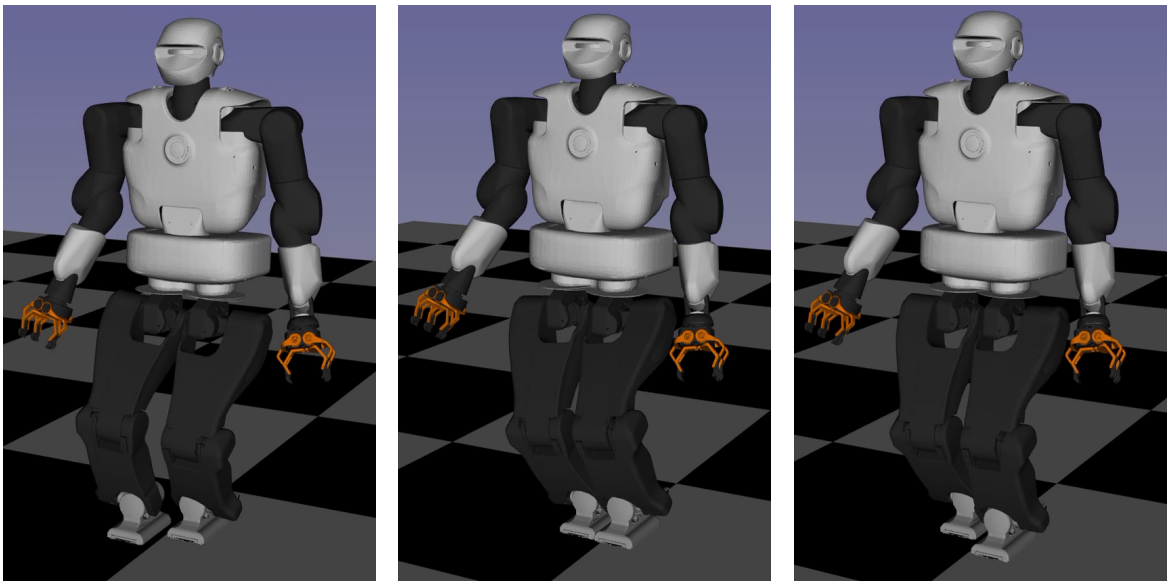


Figure 9.4 A simulation of the TALOS robot walks with the TSID-Flex controller

9.4 Results

In this section, we present the simulation tests of the control strategy presented in Section 9.2 – Figure 9.4. The proposed control strategy is compared with a whole-body controller that assumes all the robot joints actuated – see Section 7.2. From now on, the control approach presented in this chapter is called *TSID-Flex* while the controller introduced in Section 7.2 *TSID-Rigid*. The experiments are carried out on a simulated version of the TALOS humanoid robot – see Section 1.2. The architecture takes (on average) less than 1 ms to evaluate its output. The OSQP [Stellato et al., 2018a] library is used to solve the optimization problems. The code is fully implemented in Python [1]. The simulations are obtained by integrating the forward dynamics (FD) of the robot obtained from (3.3.1). Figure 9.5 shows a zoom in on the flexible joints. To simply visualize the link deformation, we introduced two disks with zero masses and zero inertial at the level of the flexibility. When the two disks on the same leg coincide, the positions of the flexible joints are equal to zero.

To validate the performance of the proposed architecture, we present two main experiments. First, we compare the performance of the TSID-Flex and the TSID-Rigid controllers in the case of different stiffness parameters k. Second, we analyze the

[1]The control objective is implemented exploiting the Python bindings provided by the `bipedal-locomotion-framework` library: `https://github.com/ami-iit/bipedal-locomotion-framework/tree/v0.6.0/bindings`

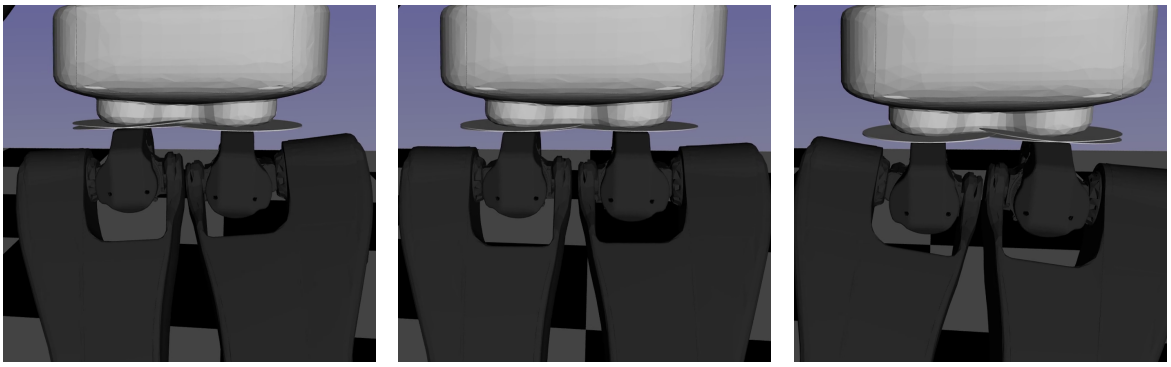


Figure 9.5 Zoom of the flexible joints motion.

performance of the TSID-Flex in the case of different stiffness. In both scenarios, the desired robot CoM, footsteps, and actuated joint trajectories are computed offline [2]. The robot walks 1 meter forward with a step length of 20 cm, starting with the right foot. The first and last steps are 10 cm long. The double support lasts 0.2 s, while the single support lasts 1.2 s.

9.4.1 Comparison between TSID-Flex and TSID-Rigid

In Table 9.1, we summarize the results of the control strategies for different stiffness parameters k. The labels *success* and *failure* mean that the associated controller is either able or not to ensure the robot's balance while walking. In all the experiments presented in this section, the damping parameter is arbitrarily set to $b = 2\sqrt{k}$.

To compare the two controllers, we decided to perform two main experiments. In the former, we choose a set of flexible parameters such that both whole-body controllers guarantee the balance while walking. In the latter, we decrease the value of the stiffness parameter. Namely:

- **Experiment 1** $k = 1 \times 10^4\,\mathrm{N\,rad^{-1}}$;

Table 9.1 Controller implementation outcome in the case of different joint stiffness parameter k. The damping parameter is set to $b = 2\sqrt{k}$.

Whole-Body Control	$10\,\mathrm{kN\,rad^{-1}}$	$5\,\mathrm{kN\,rad^{-1}}$	$3\,\mathrm{kN\,rad^{-1}}$	$2\,\mathrm{kN\,rad^{-1}}$	$1\,\mathrm{kN\,rad^{-1}}$
TSID-Rigid	success	success	failure	failure	failure
TSID-Flex	success	success	success	success	success

[2]The whole-body trajectory are provided by: `https://github.com/loco-3d/multicontact-api/tree/v2.1.0`

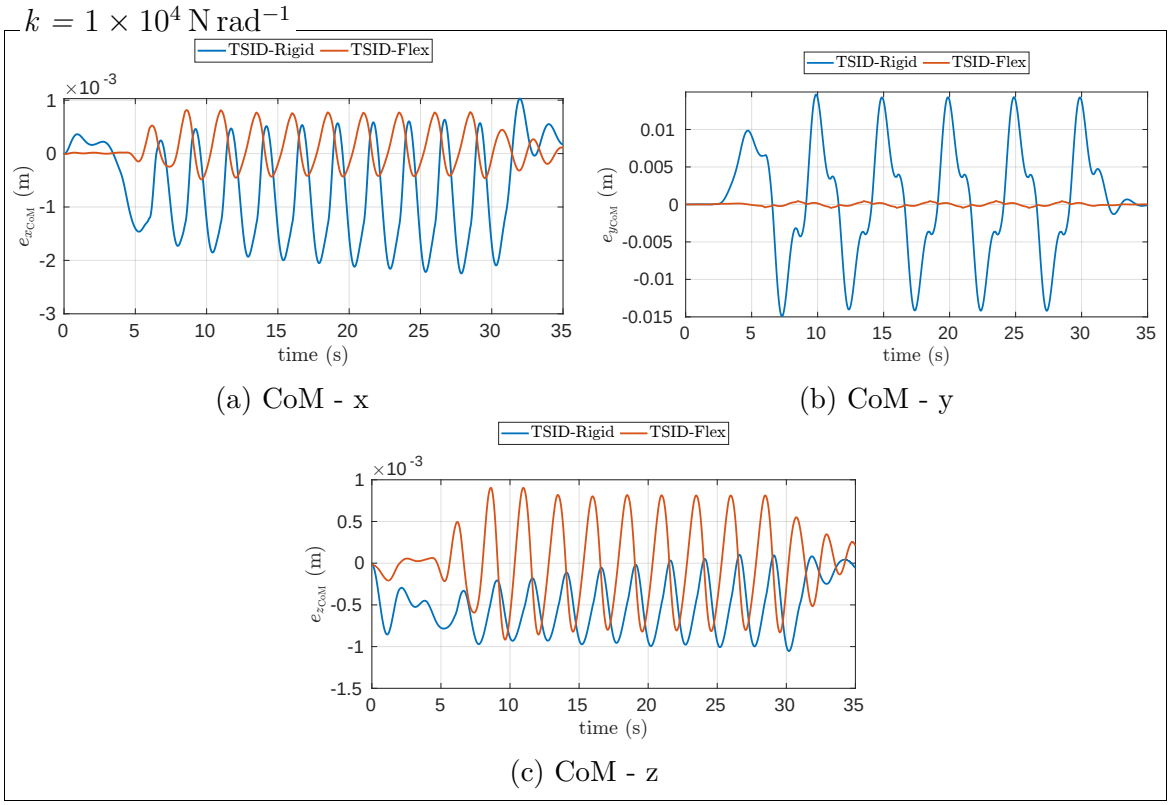


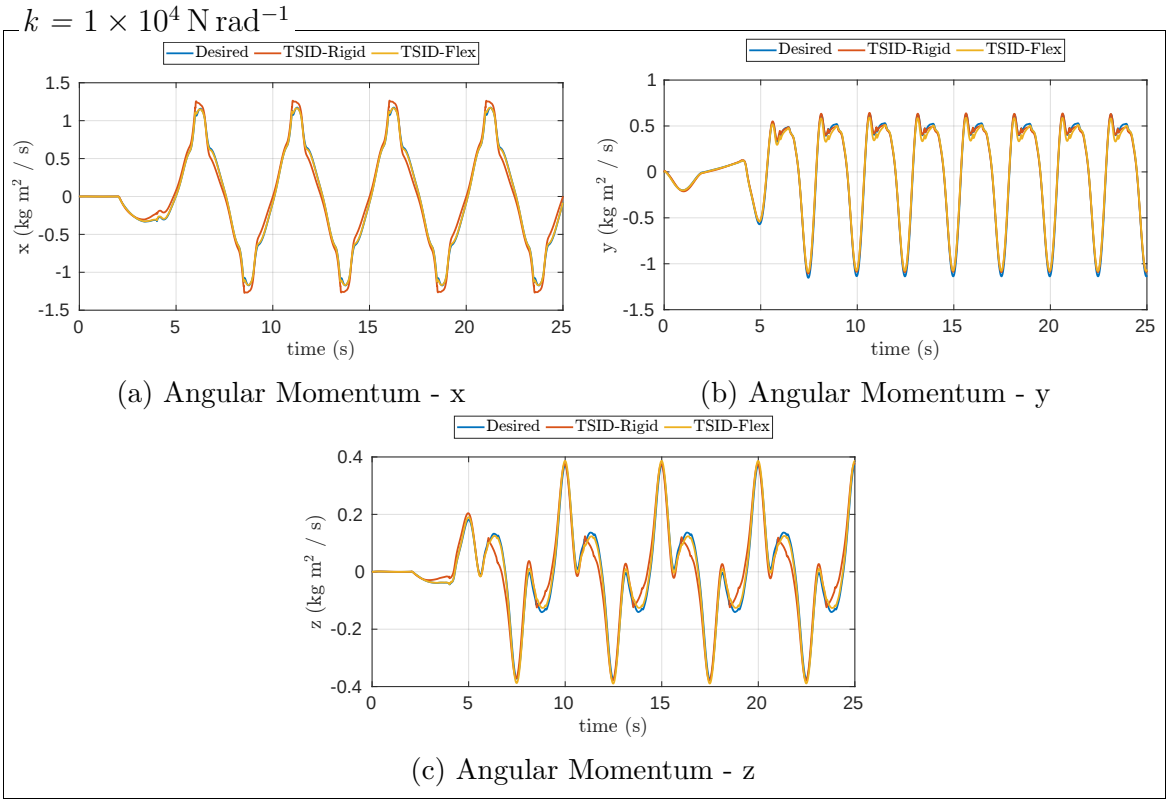
Figure 9.6 CoM tracking: comparison between TSID-Rigid and TSID-Flex.

- **Experiment 2** $k = 3 \times 10^3\,\mathrm{N\,rad^{-1}}$.

Experiment 1

Figure 9.6 shows the CoM tracking performance obtained with TSID-Flex and TSID-Rigid in terms of tracking error. The TSID-Flex controller seems to show good tracking performance and the CoM error is kept below 2 mm. On the other hand, the TSID-Rigid induces a higher error on the CoM tracking. Similar considerations hold also for the tracking of the centroidal angular momentum.

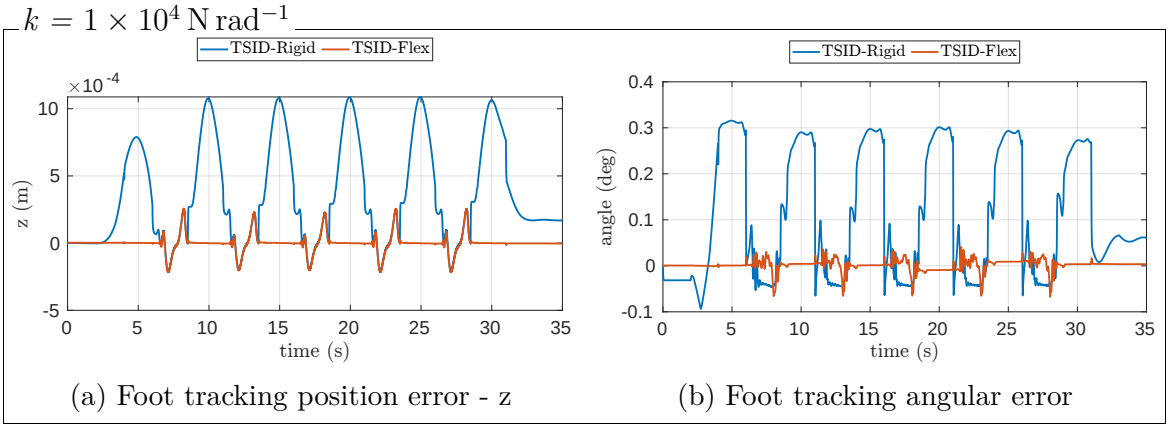
Figure 9.7 presents the tracking of the angular momentum. The TSID-Flex ensures a smaller angular momentum error with respect to the TSID-Rigid. One reason for this behavior is that the TSID-Rigid assumes full control of all the robot joints. This assumption is generally valid in the case of stiff k, but it does not hold if k decreases. Figure 9.8 presents the left foot trajectory error when the whole-body controller is TSID-Flex or TSID-Rigid. The angular error is given by the angle of the axis-angle representation between the foot orientation and the desired orientation [Huynh, 2009].



(a) Angular Momentum - x (b) Angular Momentum - y

(c) Angular Momentum - z

Figure 9.7 Angular momentum tracking: comparison between TSID-Rigid and TSID-Flex.



(a) Foot tracking position error - z (b) Foot tracking angular error

Figure 9.8 Foot tracking: comparison between TSID-Rigid and TSID-Flex.

Since TSID-Rigid does not consider flexible joint deformation, the controller assumes a wrong foot orientation when the robot is on single support – for $2.5\,\mathrm{s} \leq t \leq 5.5\,\mathrm{s}$ in Figure 9.8.

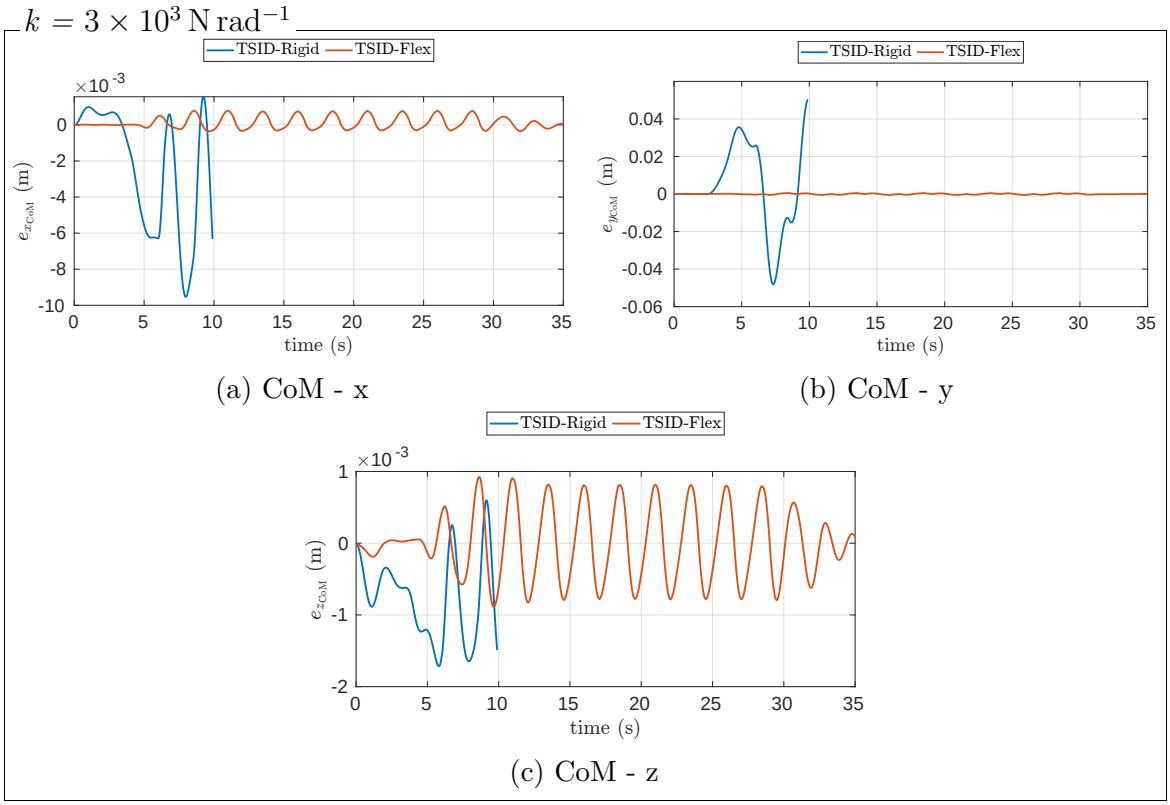


Figure 9.9 CoM tracking: comparison between TSID-Rigid and TSID-Flex.

Experiment 2

The centroidal quantity tracking issue discussed in Experiment 1 worsens at lower values of the stiffness parameter k. Figure 9.9 shows the CoM tracking performances of the two controllers. In Figure 9.10 we present the tracking of the angular momentum. The TSID-Flex is still capable of ensuring good performance. On the other hand, the TSID-Rigid does not consider the flexible joint state. As a consequence, this leads to a non-negligible error on the robot CoM. In order to keep the balance, the TSID-Rigid controller requires high variations of the robot's foot orientation – see Figure 9.11. At $t \approx 10\,\mathrm{s}$ the robot falls.

9.4.2 Performances of the TSID-Flex in the case of different stiffness parameters

In this section, we benchmark the performance of the TSID-Flex controller in the case of different stiffness parameter k, namely $k = 10\,\mathrm{kN\,rad^{-1}}$, $5\,\mathrm{kN\,rad^{-1}}$, $3\,\mathrm{kN\,rad^{-1}}$,

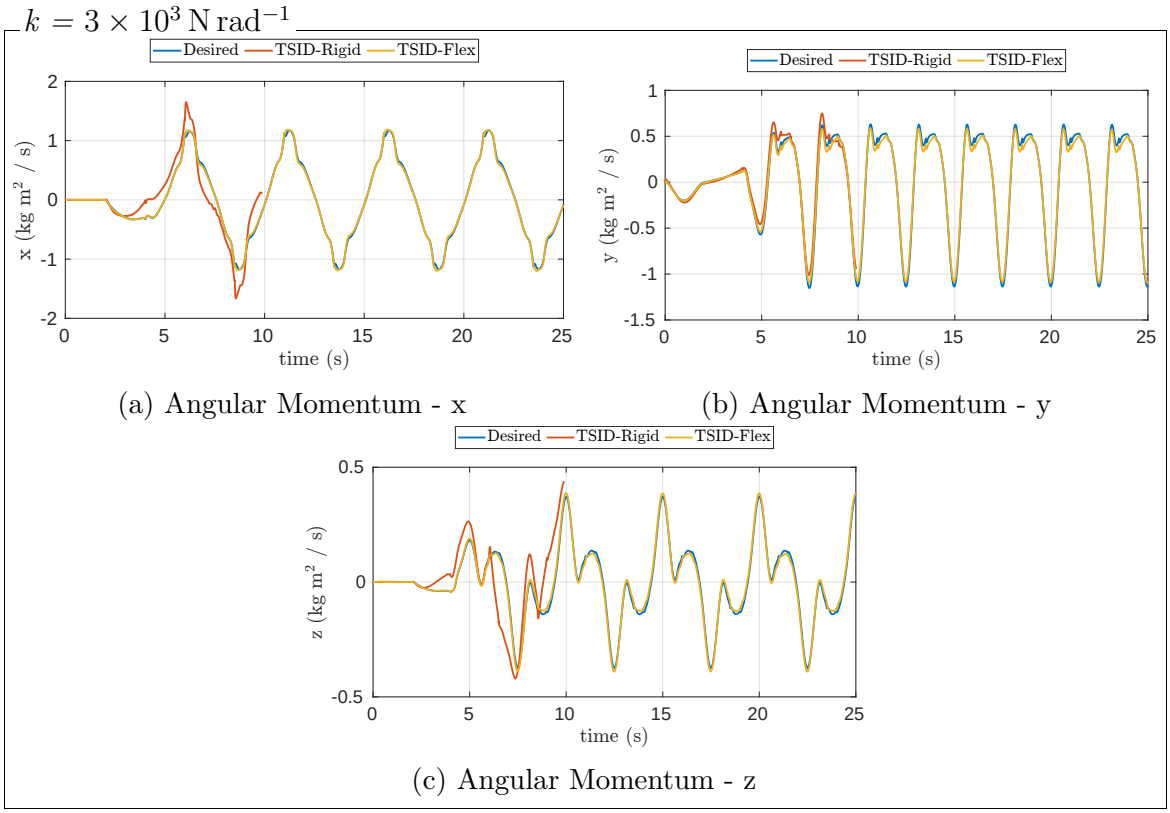


Figure 9.10 Angular momentum tracking: comparison between TSID-Rigid and TSID-Flex.

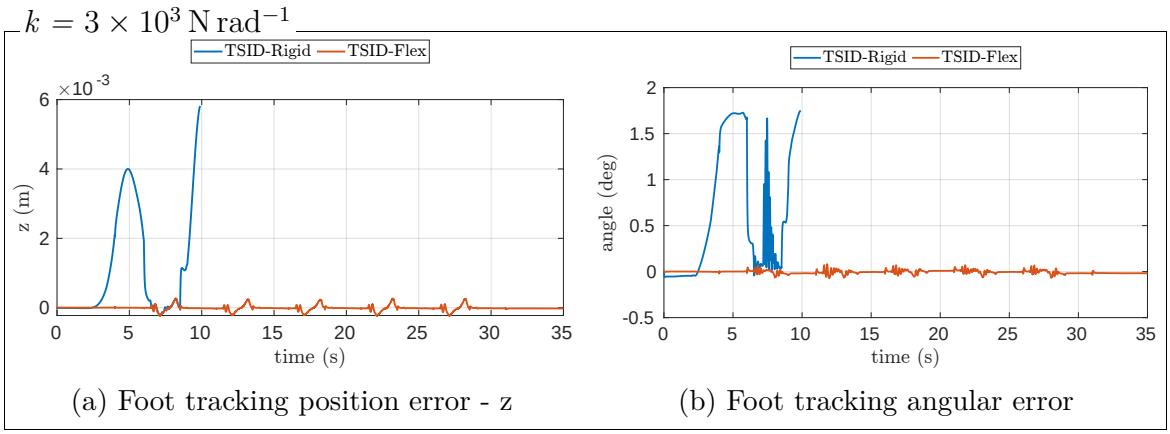
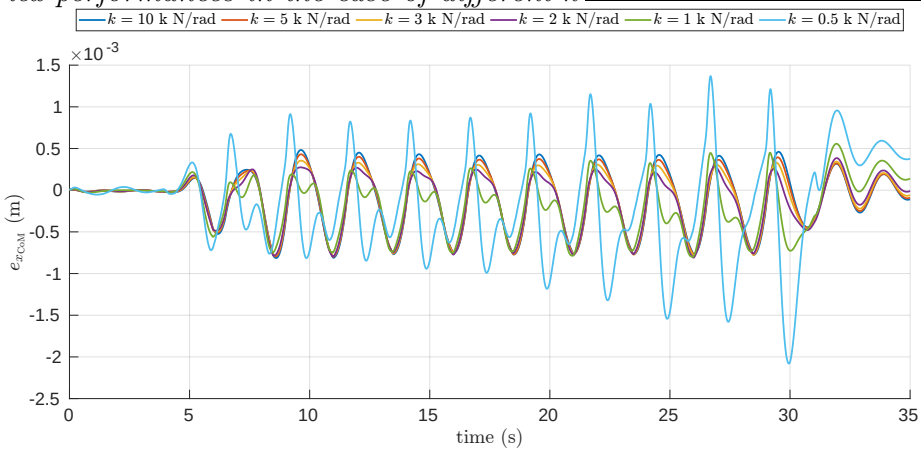
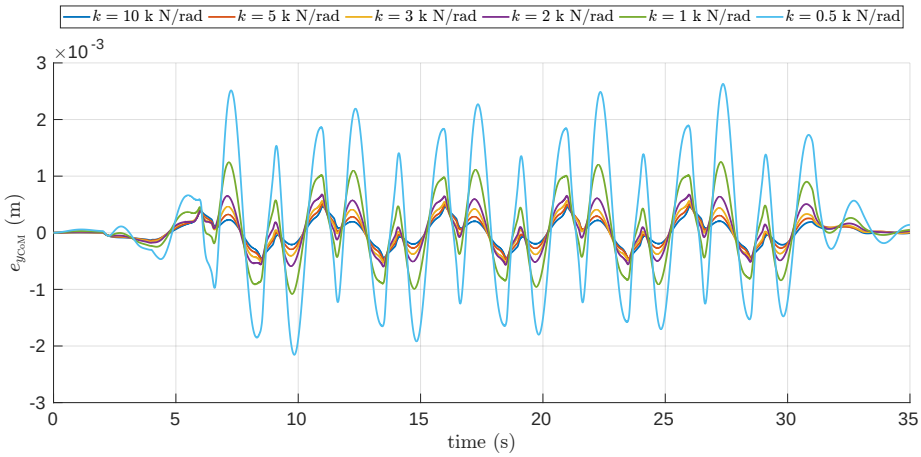


Figure 9.11 Foot tracking: comparison between TSID-Rigid and TSID-Flex.

TSID-Flex performances in the case of different k



(a) CoM x error.



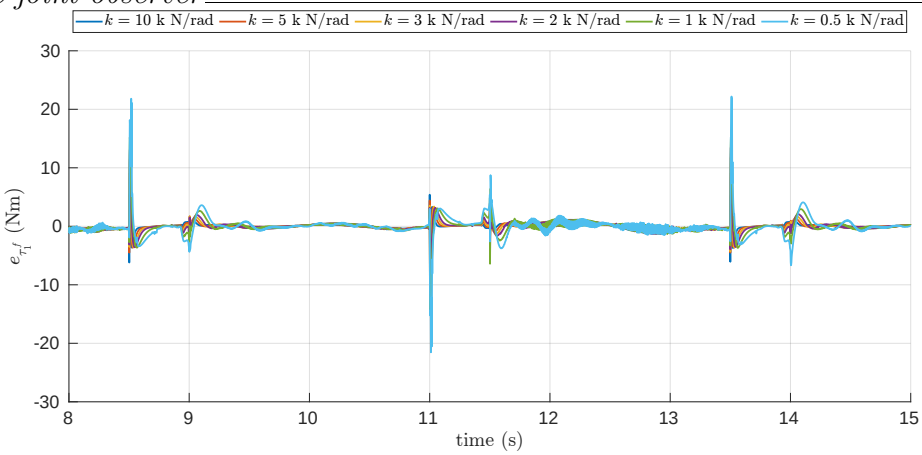
(b) CoM y error.

Figure 9.12 CoM Tracking.

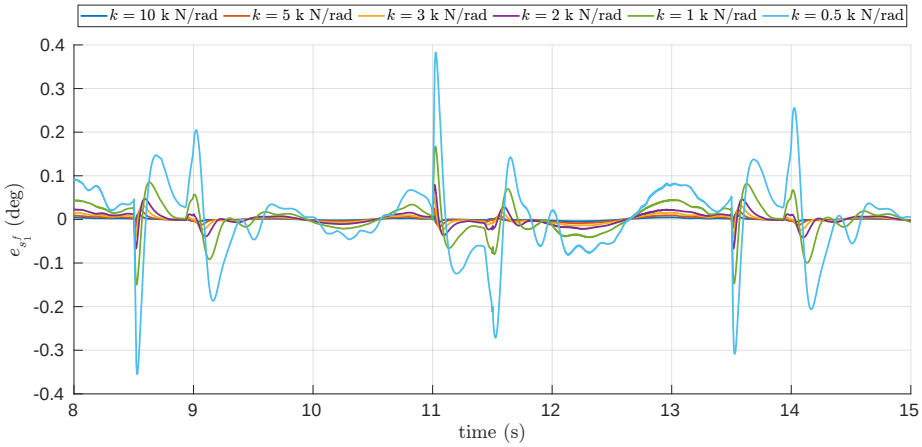
$2\,\mathrm{kN\,rad^{-1}}$, $1\,\mathrm{kN\,rad^{-1}}$, and $0.5\,\mathrm{kN\,rad^{-1}}$. In all experiments presented in this section, the damping parameter is arbitrarily set to $b = 2\sqrt{k}$.

Figure 9.12 shows the CoM tracking error for different values of k. The controller induces an acceptable performance for all values of k, i.e., the error is always below 3 mm. However, the lower the stiffness, the higher the tracking error. This increase in tracking error is caused by the flexible joint state estimation error. Figure 9.13 presents the estimation error of a flexible joint. Similar considerations hold also for the other three joints. When the robot switches from single support to double support, the estimated torque associated with the flexible joint has a spike – at $t \approx 8.5\,\mathrm{s}$, 11 s and 13.5 s in Figure 9.13a. As a consequence, the error propagates in the estimation of the flexible joint position and velocity – see Figures 9.13b and 9.13c. This behavior is

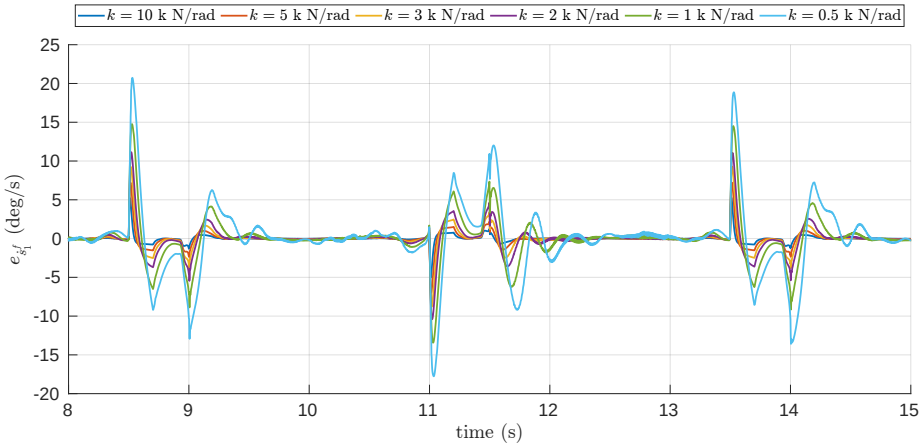
Flexible joint observer



(a) Flexible joint torque error.



(b) Flexible joint position error.



(c) Flexible joint velocity error.

Figure 9.13 Flexible joint state estimation error.

caused by the discontinuity of the contact force. To mitigate this effect, we may try to perform a smother transition from single to double support and vice versa. The lower the stiffness parameter k, the higher the estimation error.

9.5 Conclusions

This chapter presents the design of a whole-body QP control layer for a humanoid robot affected by link flexibility. We model the flexibility by introducing equivalent passive joints that simulate the motion caused by the link deformation. We then considered the passive joints position and velocity as state of the floating base system dynamics. Thanks to this choice, we develop a whole-body controller that implicitly considers the joint flexibility in the stabilization problem. The chapter also details the design of an estimator that aims at computing the flexible joint state in real-time.

The proposed approach is validated in a simulated version of the TALOS humanoid robot, where its hip flexibility has a significant impact while performing locomotion tasks. Moreover, the architecture is then compared with a whole-body controller that considers all links of the robot rigid.

As a future work, we plan to mitigate the discontinuity of the contact forces by performing a smother transition between contiguous support phases. We also plan to make a detailed comparison with other state-of-the-art controllers that consider the flexibility of the robot link [Villa et al., 2022]. In addition, we plan to validate the architecture on the real robot.

Part III

From Simplified to Reduced Models Controllers

Chapter 10

Benchmarking of Simplified-Model Controllers for Locomotion

In Part II, we discussed whole-body controllers that take into account different types of robots, as well as the interaction between the robot and the environment. Independently of the choice of the whole-body controller, we assumed a high-level algorithm that provides the reference for the Cartesian trajectories, e.g., CoM and foot trajectories. This part presents the design of two high-level controllers that compute the commands for the whole-body control layer presented in Part II. This chapter, in particular, defines the three-layer controller architecture illustrated in Figure 6.1 as shown in Figure 10.1. The *whole-body QP control* ensures the tracking of the desired CoM and foot trajectories by considering the complete robot models. A detailed design of different kinds of whole-body controllers is discussed in Part II. The *trajectory optimization* layer is maintained fixed with a unicycle-based planner [Dafarra et al., 2018] that generates the desired DCM and foot trajectories. The *simplified model controller* is responsible for implementing a control law that stabilizes the unstable DCM dynamics. We approach the stabilization problem by designing two different controllers: an *instantaneous* and a *predictive* one. We compare several combinations of the control architecture with the kinematics-based whole-body QP control layer presented in Section 7.1. We carried out the test on the humanoid robot iCub v2.7 – see Section 1.1.1.

The chapter is organized as follows. Section 10.1 introduces a state-of-the-art implementation of the footstep planner and the DCM planner implemented in the *trajectory optimization layer*. Section 10.2 details the components constituting the simplified model blocks of the three-layer controller architecture. Section 10.3 presents the

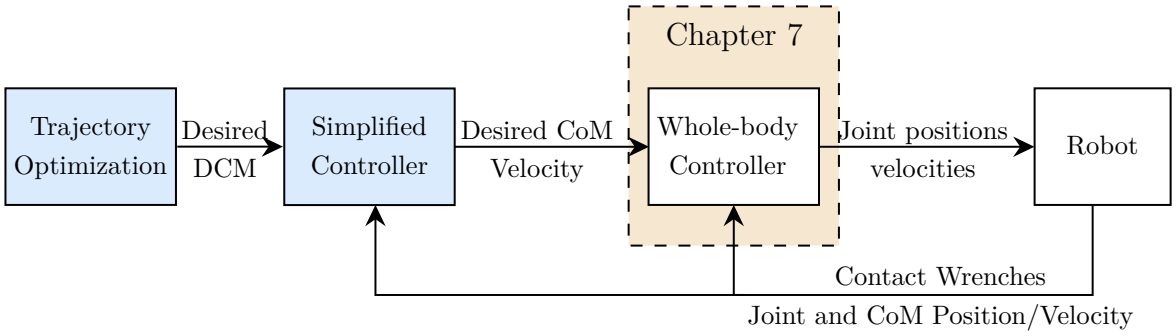


Figure 10.1 The control architecture is composed of three layers: the *trajectory optimization*, the *simplified model control*, and the *whole-body control*. The inner layer is described in Chapter 7.

experimental validation of the proposed approaches and shows an explanatory table comparing the different simplified control strategies. Finally, Section 10.4 concludes the chapter.

The content of this chapter appears partially in:

> Romualdi, G., Dafarra, S., Hu, Y., and Pucci, D. (2018). A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE
>
> Romualdi, G., Dafarra, S., Hu, Y., Ramadoss, P., Chavez, F. J. A., Traversaro, S., and Pucci, D. (2020). A Benchmarking of DCM-Based Architectures for Position, Velocity and Torque-Controlled Humanoid Robots. *International Journal of Humanoid Robotics*, 17(01):1950034
>
> Video `https://www.youtube.com/watch?v=FIqwAO71Fc4`
> GitHub `robotology/walking-controllers`

10.1 Background

In several applications, we can assume flat terrain. In this scenario, a human, while walking, tends to keep the orientation of the body tangential to the path [Flavigne et al., 2010; Mombaur et al., 2010; Truong et al., 2010] to maximize the energy

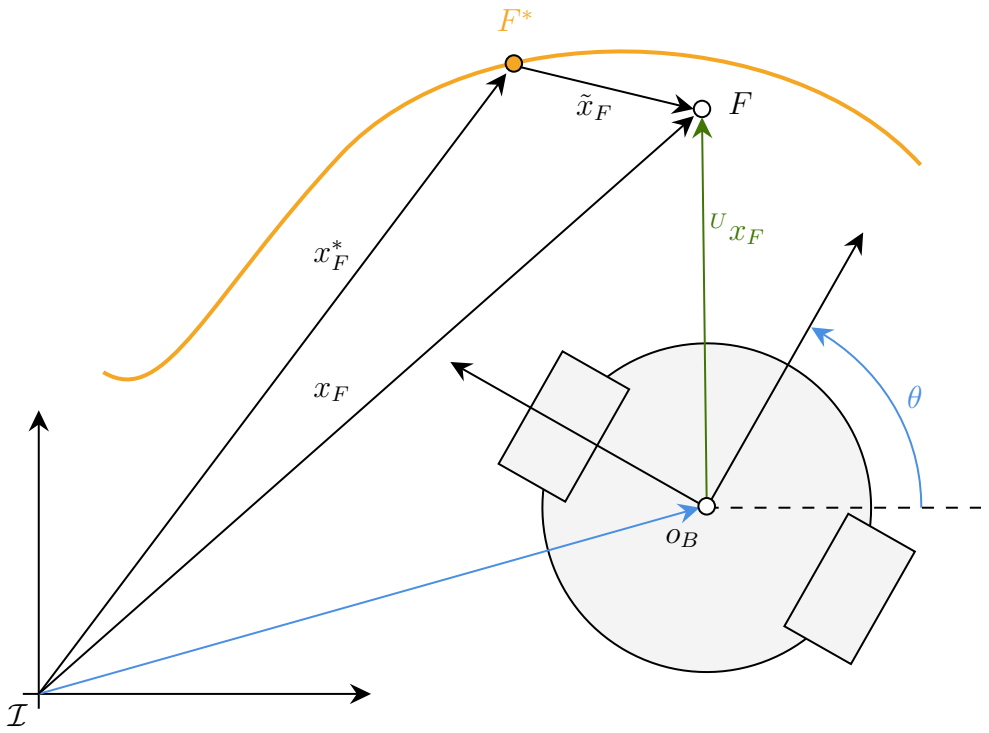


Figure 10.2 Unicycle model.

efficiency [Handford and Srinivasan, 2014]. This consideration suggests designing a simple unicycle model to generate the desired footsteps locations and timings [Cognetti et al., 2016; Dafarra et al., 2018; Faragasso et al., 2013]. Authors of [Faragasso et al., 2013] considers the unicycle model to plan footsteps in a corridor with turns and junctions using cameras. While Cognetti et al. [2016] exploits the unicycle model to perform *evasive* robot maneuvers. However, in both cases, the footstep planners consider a constant velocity and a constant footsteps length. Attempts have been made to consider varying the length and velocity of the steps [Dafarra et al., 2018].

10.1.1 The unicycle model

Let us now consider a 2D unicycle – Figure 10.2, and assume that:

- there exists an inertial frame $\mathcal{I}$;
- there exists a frame U rigidly attached to the center of the unicycle, and we denote by o_B the origin of the frame and by $[B]$ its orientation.

The unicycle model is described by the following dynamical system [Pucci et al., 2013]:

$$\begin{cases} \dot{o}_B = R(\theta) e_1 v \\ \dot{\theta} = \zeta \end{cases} \tag{10.1.1}$$

where $v \in \mathbb{R}$ is the linear velocity of the robot expressed in the frame B. $\zeta \in \mathbb{R}$ is the unicycle angular velocity. o_B is the position of the unicycle relative to the inertial frame $\mathcal{I}$. θ is the orientation of the robot.

The control objective is to asymptotically stabilize the point F, a fixed point with respect to the unicycle, to the desired point F^*. Thus, we define the error $\tilde{x}_F$ as

$$\tilde{x}_F := x_F - x_F^*, \tag{10.1.2}$$

where x_F is given by

$$x_F = o_B + R(\theta)\, {}^B x_F, \tag{10.1.3}$$

with ${}^U x_F$ being a constant vector. The control objective becomes the asymptotic stabilization of $\tilde{x}_F$ to zero. By time-differentiating Equation (10.1.3), we obtain the following dynamical system

$$\dot{x}_F = \dot{o}_B + \zeta R(\theta) H\, {}^B x_F. \tag{10.1.4}$$

where H is

$$H = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

By substituting Equation (10.1.1) into (10.1.4), we can rewrite the output dynamics as

$$\dot{x}_F = B(\theta) u = \begin{bmatrix} R(\theta) e_1 & R(\theta) H\, {}^U x_F \end{bmatrix} u = R(\theta) \begin{bmatrix} 1 & -{}^U x_{F_y} \\ 0 & {}^U x_{F_x} \end{bmatrix} u. \tag{10.1.5}$$

Here, u is the vector containing the controlled input v and ζ, i.e., $u = \begin{bmatrix} v & \zeta \end{bmatrix}^\top$. It can be seen that $\det(B(\theta)) = {}^B x_{F_x}$, consequently, when the control point F is not located on the axis of the wheels, its stabilization to an arbitrary reference trajectory F^* can be achieved by using a simple feedback linearization law of the form

$$u = B(\theta)^{-1}(\dot{x}_F^* - K(x_F - x_F^*)), \tag{10.1.6}$$

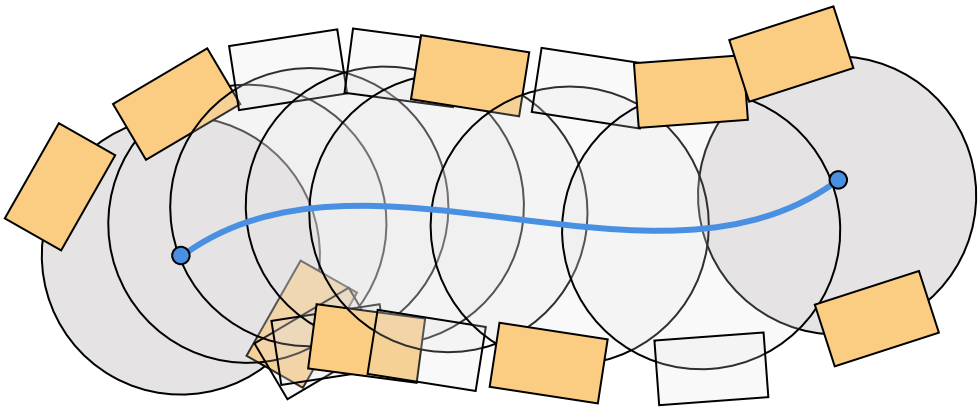


Figure 10.3 Footsteps planning from the unicycle trajectory.

with K a positive definite matrix, that is, $K \succ 0$.

10.1.2 Footsteps trajectory planner

To obtain the footsteps positions, the trajectory of the left foot and the right foot, represented by the wheels of the unicycle, are sampled using a sampling time $\mathrm{d}\,T$ as in [Dafarra et al., 2018]

$$o_{\mathcal{L}_k} = o_{U_k} + R(\theta_k)\,{}^{U}o_{\mathcal{L}}, \qquad o_{\mathcal{R}_k} = o_{U_k} + R(\theta_k)\,{}^{U}o_{\mathcal{R}}, \tag{10.1.7}$$

where $\mathcal{L}$ and $\mathcal{R}$ are, respectively, the frames attached to the left and right foot. The subscript k denotes the time instant, i.e., $t = k\,\mathrm{d}\,T$.

Given the set of unicycle position (which is finite, thanks to discretization), we can select a particular position for the feet. Each position in the set is associated with a time instant, k. This instant is considered the moment in which the corresponding foot is expected to touch the ground and is called the impact time, t_{imp}. Impact time can be used as decision variables to also select the position of feet.

Using the impact time, the step duration can be defined:

$$\Delta_t = |t^{lf}_{\text{imp}} - t^{rf}_{\text{imp}}|. \tag{10.1.8}$$

Another interesting quantity is the distance of the feet during the double support phase

$$\Delta_x = \left\| {}^{\mathcal{I}}o^{ds}_{\mathcal{L}_k} - {}^{\mathcal{I}}o^{ds}_{\mathcal{R}_k} \right\|. \tag{10.1.9}$$

Both Δ_x and Δ_t are used within the cost function in order to reduce the possibility to obtain long and fast steps

$$J = K_t \frac{1}{\Delta_t^2} + K_x \Delta_x^2, \tag{10.1.10}$$

where $K_t > 0$ and $K_x > 0$ are used to allow the robot to perform long/short and fast/slow steps.

However, considering only the cost function, too long/short or too fast/slow steps can be planned. To avoid this undesirable behavior, several constraints are added. First of all, the bound on the step duration is necessary to avoid too fast or too slow steps and a division by zero in the cost function:

$$t_{\min} \leq \Delta_t \leq t_{\max}. \tag{10.1.11}$$

Also, the distance between the feet during the double support phase has to be bound in order to avoid undesired collision between feet and, on the other hand, step length that the robot, according to its mechanical structure cannot achieve:

$$d_{\min} \leq \Delta_x \leq d_{\max}. \tag{10.1.12}$$

Last but not least, an additional constraint has to be taken into account: the relative rotation between the two feet, $|\Delta_\theta|$, has to be lower than $\theta_{\max}$, this is necessary to take into account the mechanical limitations of the humanoid robot.

$$|\Delta_\theta| \leq \theta_{\max}. \tag{10.1.13}$$

Finally, the control objective is achieved by defining the planning problem as a constrained optimization problem whose conditional variable is t_{imp}. In detail, we have:

$$\begin{aligned}
\underset{t_{\text{imp}}}{\text{minimize}} \quad & K_t \frac{1}{\Delta_t^2} + K_x \Delta_x^2 && \text{(10.1.14a)} \\
\text{subj. to} \quad & t_{\min} \leq \Delta_t \leq t_{\max} && \text{(10.1.14b)} \\
& d_{\min} \leq \Delta_x \leq d_{\max} && \text{(10.1.14c)} \\
& |\Delta_\theta| \leq \theta_{\max}. && \text{(10.1.14d)}
\end{aligned}$$

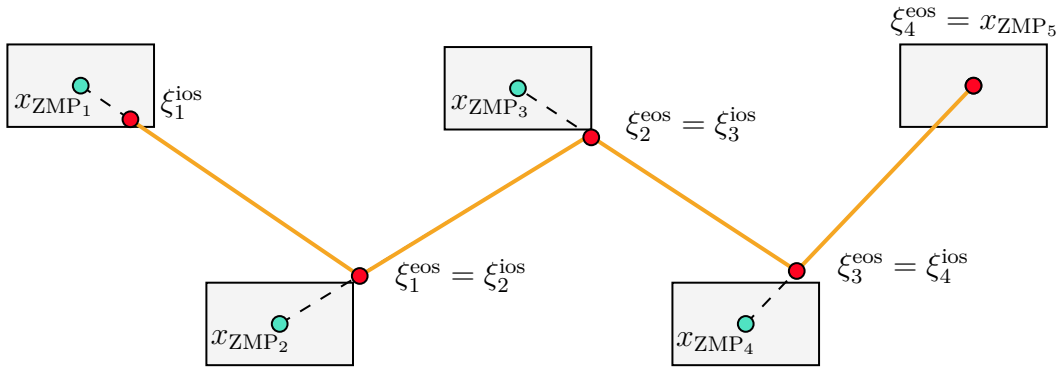


Figure 10.4 Planning of the DCM trajectory on a flat terrain considering only single support phases. The ZMP position is assumed constant during the single support transition. The final position of the DCM coincides with ZMP of the last footstep, $\xi_4^{\text{eos}} = x_{\text{ZMP}_5}$.

10.1.3 DCM trajectory generator

We now introduce the trajectory generation problem. Following the work of Englsberger et al. [2014], we first assume instantaneous transitions between two consecutive single support phases (SS). Later, we relax this assumption by considering the double support phases (DS).

Generation of the single support trajectory

Let us consider a set of footsteps locations and timings generated by a proper footsteps planner, e.g., the unicycle footstep planner in Section 10.1.2. Let us also assume that the LIPM hypothesis is valid – Section 4.1. Then the DCM dynamics (4.4.7) can be projected on the contact plane and rewritten as (4.4.14b):

$$\dot{\xi} = \zeta \left(\xi - x_{\text{ZMP}} \right) \tag{10.1.15}$$

where, for the sake of clarity, we removed the suffix LIP.

For a constant ZMP, the projected DCM dynamics (4.4.14b) admits a close form solution of the form:

$$\xi(t) = x_{\text{ZMP}} + e^{\zeta t}(\xi_0 - x_{\text{ZMP}}), \tag{10.1.16}$$

where $\xi_0 = \xi(0)$ and the time t has to belong to the step domain $t \in [0,\ t_i^{\text{step}}]$, where t_i^{step} is the duration of the i-th step.

Assuming that the final position of the DCM coincides with the ZMP at the last step, i.e., $\xi_{N-1}^{\text{eos}} = x_{\text{ZMP}_N}$, Equation (10.1.16) can be used to find the desired DCM

position at the end of each step as:

$$\xi_{N-1}^{\text{eos}} = x_{\text{ZMP}_N} \tag{10.1.17a}$$

$$\xi_{i-1}^{\text{eos}} = \xi_i^{\text{ios}} = x_{\text{ZMP}_i} + e^{-\zeta t_i^{\text{step}}}(\xi_i^{\text{eos}} - x_{\text{ZMP}_i}), \tag{10.1.17b}$$

where ξ_i^{ios} and ξ_i^{eos} are respectively the initial and final positions of the desired DCM for the i-th step .

By substitution of (10.1.17) into (10.1.16), the reference DCM trajectory is calculated as follows:

$$\xi_i(t) = x_{\text{ZMP}_i} + e^{\zeta\left(t - t_i^{\text{step}}\right)}\left(\xi_i^{\text{eos}} - x_{\text{ZMP}_i}\right). \tag{10.1.18}$$

The DCM velocity trajectory can be easily evaluated by differentiation of (10.1.18)

$$\dot{\xi}_i(t) = \zeta e^{\zeta\left(t - t_i^{\text{step}}\right)}\left(\xi_i^{\text{eos}} - x_{\text{ZMP}_i}\right). \tag{10.1.19}$$

By applying, Equation (10.1.17) and (10.1.18) recursively, we compute the DCM trajectory. Figure 10.4 presents an example of DCM trajectory generation in the case of five footsteps.

Generation of the double support trajectory

The presented planning method is very powerful and allows one to generate the desired DCM trajectory in real-time. Nevertheless, it has the limit of taking into account only single support phases. Indeed, considering only instantaneous transitions between two consecutive single support phases, the ZMP reference is discontinuous [Englsberger et al., 2014]. This makes the external forces discontinuous too, and, in turn, also the desired joint torque may suffer from discontinuity. This can be unfeasible for a physical robot due to its limited actuator dynamics. These considerations motivate the development of a new DCM trajectory generator that handles the non-instantaneous transition between two single support phases [Englsberger et al., 2014].

We notice that by rearranging Equation (10.1.15), the ZMP position depends linearly on the DCM position and velocity

$$x_{\text{ZMP}} = \xi - \zeta\dot{\xi}, \tag{10.1.20}$$

as a consequence, we can conclude that a DCM reference trajectory with continuous derivative, i.e., C^1 continuity, guarantees a continuous ZMP trajectory. This considera-

tion motivates the use of a third-order polynomial interpolation to smooth the edges of the DCM reference trajectory evaluated using the exponential technique

$$\xi^{\mathrm{DS}}(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \tag{10.1.21}$$

where the parameters a_i must be chosen to satisfy the velocity and position boundary conditions, namely:

$$\xi_i^{\mathrm{ios_{DS}}} = x_{\mathrm{ZMP}_{i-1}} + e^{\zeta\left(t_{i-1}^{\mathrm{ios_{DS}}} - t_{i-1}^{\mathrm{step}}\right)}(\xi_{i-1}^{\mathrm{eos}} - x_{\mathrm{ZMP}_{i-1}}) \tag{10.1.22a}$$

$$\dot{\xi}_i^{\mathrm{ios_{DS}}} = \zeta e^{\zeta\left(t_{i-1}^{\mathrm{ios_{DS}}} - t_{i-1}^{\mathrm{step}}\right)}(\xi_{i-1}^{\mathrm{eos}} - x_{\mathrm{ZMP}_{i-1}}) \tag{10.1.22b}$$

$$\xi_i^{\mathrm{eos_{DS}}} = x_{\mathrm{ZMP}_i} + e^{\zeta\left(t_i^{\mathrm{eos_{DS}}} - t_i^{\mathrm{step}}\right)}(\xi_i^{\mathrm{eos}} - x_{\mathrm{ZMP}_i}) \tag{10.1.22c}$$

$$\dot{\xi}_i^{\mathrm{eos_{DS}}} = \zeta e^{\zeta\left(t_i^{\mathrm{eos_{DS}}} - t_i^{\mathrm{step}}\right)}(\xi_i^{\mathrm{eos}} - x_{\mathrm{ZMP}_i}). \tag{10.1.22d}$$

Equations (10.1.22a) and (10.1.22b) denote, respectively, the desired position and velocity of DCM at the beginning of the double support phase. On the other hand, Equations (10.1.22c) and (10.1.22d) represent the desired position and velocity of DCM at the end of the double support phase. $t_i^{\mathrm{ios_{DS}}}$ and $t_i^{\mathrm{eos_{DS}}}$ are, respectively, the initial and final time of the i-th double support phase.

By combining the third-order DCM trajectory (10.1.21) with the boundary conditions (10.1.22), we obtain the following:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ T_i^3 & T_i^2 & T_i & 1 \\ 3T_i^2 & 2T_i & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{3_i}^\top \\ a_{2_i}^\top \\ a_{1_i}^\top \\ a_{0_i}^\top \end{bmatrix} = \begin{bmatrix} (\xi_i^{\mathrm{ios_{DS}}})^\top \\ (\dot{\xi}_i^{\mathrm{ios_{DS}}})^\top \\ (\xi_i^{\mathrm{eos_{DS}}})^\top \\ (\dot{\xi}_i^{\mathrm{eos_{DS}}})^\top \end{bmatrix} \quad \forall i \in \{i \in \mathbb{N} | 1 \leq i \leq N\}. \tag{10.1.23}$$

T_i is the duration of the i-th double support phase, i.e., $T_i = t_i^{\mathrm{eos_{DS}}} - t_i^{\mathrm{ios_{DS}}}$. Equation (10.1.23) is always solvable if and only if the i-th double support duration $T_i \neq 0$. The position and velocity of DCM during the double support phase are finally obtained by combining the solution of (10.1.23) with (10.1.21).

Figure 10.5 presents an example of DCM trajectory generation considering single and double support phases. While the robot is in single support, the DCM is generated by applying the exponential interpolation technique (10.1.18) (orange segment). During the double support phase (light blue curves), the DCM trajectory is evaluated by means of the polynomial interpolation method (10.1.21).

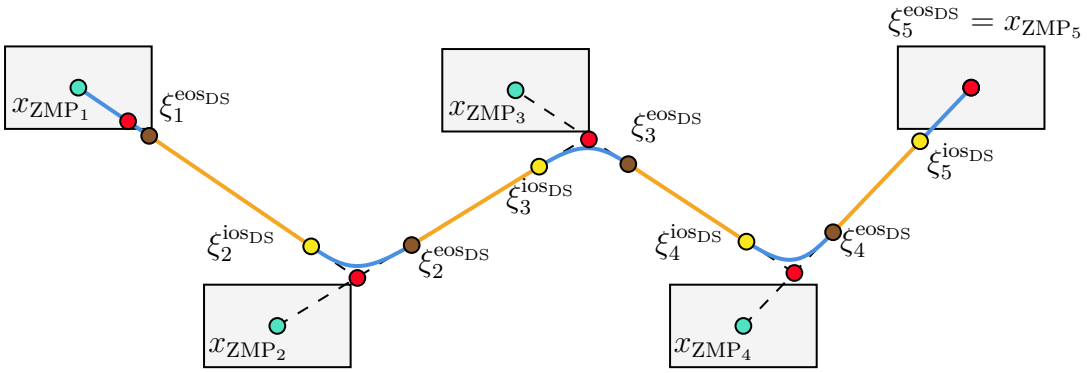


Figure 10.5 Planning of the DCM trajectory on a flat terrain. The SS trajectories, evaluated using the exponential function (10.1.18), are represented by orange segments. The DS trajectories, evaluated using the polynomial function (10.1.21), are represented by light blue curves.

10.2 Simplified model architecture

In this section, we summarize the components constituting the simplified model blocks of the three-layer controller architecture – Figure 6.1. In detail, we present

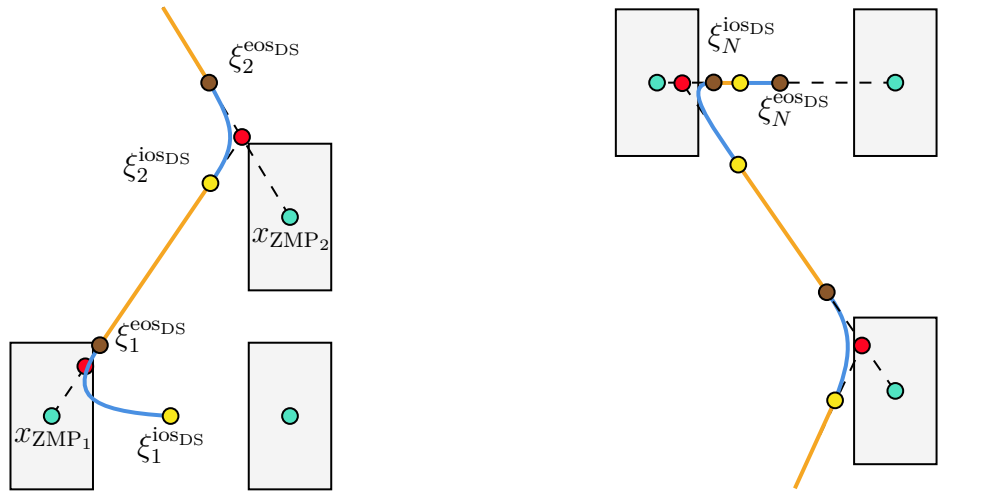
- how we consider the first and last steps in the DCM planner,
- the swing foot trajectory planner,
- the simplified model control layer.

These components share a lot of commonalities with other state-of-the-art approaches. Nevertheless, this section presents how these three components interconnect to define part of a walking architecture.

10.2.1 The DCM trajectory planner

The DCM trajectory planner used in our architecture inherits from the formulation presented in Section 10.1.3. However, differently from Englsberger et al. [2014], we explicitly consider specific boundary conditions for the first and last steps.

For the first double support phase, we ask for an initial DCM position $\xi_1^{\text{ios}_{\text{DS}}}$ that coincides with the projection of the measured CoM on the ground surface. In this particular context, we set the initial boundary conditions of the DCM position (10.1.22a)



(a) DCM trajectory at the first step (b) DCM trajectory at the last step

Figure 10.6 DCM trajectory planner at the first and last step. The SS trajectory is represented by the orange segments, while the DS trajectory is represented by the light blue curves (a) At the first step, the initial DCM position $\xi_1^{\text{ios}_{\text{DS}}}$ coincides with the projection of the CoM on the ground surface. (b) At the last step, the final DCM position $\xi_N^{\text{eos}_{\text{DS}}}$ coincides with the projection of the CoM on the ground surface.

and velocity (10.1.22b) as:

$$\xi_1^{\text{ios}_{\text{DS}}} = x_{\text{LIP}} \tag{10.2.1a}$$

$$\dot{\xi}_1^{\text{ios}_{\text{DS}}} = 0_{2\times 1}. \tag{10.2.1b}$$

Figure 10.6a presents the DCM trajectory generation for the first double support phase.

Similarly, the desired position of the DCM at the end of the double support phase $\xi_N^{\text{eos}_{\text{DS}}}$ must coincide with the projection of the desired CoM on the ground surface, denoted by x_{LIP}^*. This is obtained as the middle point between the two feet. In this particular case, we set the final boundary conditions for the DCM position (10.1.22c) and velocity (10.1.22d) as:

$$\xi_N^{\text{eos}_{\text{DS}}} = x_{\text{LIP}}^* \tag{10.2.2a}$$

$$\dot{\xi}_N^{\text{eos}_{\text{DS}}} = 0_{2\times 1}. \tag{10.2.2b}$$

In Englsberger et al. [2014], the DCM trajectory for the last single support phase is calculated by assuming that the final position of the DCM coincides with the local

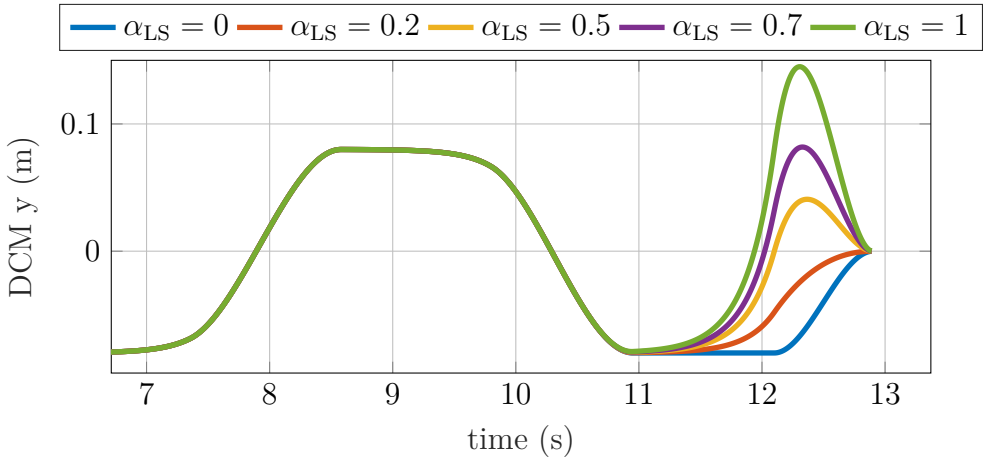


Figure 10.7 Final step DCM trajectory w.r.t. α_{LS}. The greater α_{LS}, the higher the overshoot.

ZMP in the last step – Equation (10.1.17a). In our test, we noticed that this choice may lead to an undesired overshoot of the DCM projection into the *coronal plane* [1], and consequently to an unfeasible DCM trajectory. To mitigate this behavior, we ask that the final condition of the single support DCM trajectory ξ_{N-1}^{eos} belongs to the *affine combination* (see Section 5.1.1) of the two last footprints, i.e.,

$$\xi_{N-1}^{eos} = \alpha_{LS} x_{ZMP_N} + (1 - \alpha_{LS}) x_{ZMP_{N-1}}. \tag{10.2.3}$$

When $\alpha_{LS} = 0$ the DCM position coincides with the ZMP position of the stance foot. On the other hand, if $\alpha_{LS} = 1$ we obtain the same condition proposed by Englsberger et al. [2014]. In Figure 10.7, we present the DCM trajectory in the last step projected on *frontal plane* for different values of α_{LS}. $\alpha_{LS} = 1$ leads to an undesired overshoot and, as a consequence, to an unfeasible DCM trajectory. In our experiments, we always set $\alpha_{LS} = 0.2$, which in our opinion is a good compromise between a responsive trajectory while avoiding undesired overshoots. Figure 10.6b presents the generation of the DCM trajectory for the last double support phase.

[1] A *coronal plane*, also known as the *frontal plane*, is any vertical plane that divides the body into ventral and dorsal sections.

10.2.2 Swing Foot Trajectory

We now discuss the problem of generating the foot trajectory from the footsteps. More formally, given two consecutive footsteps poses, hereafter denoted with ${}^{\mathcal{I}}H_{F_0} \in \mathrm{SE}(3)$ and ${}^{\mathcal{I}}H_{F_1} \in \mathrm{SE}(3)$, we aim to find a trajectory that connects these two poses. A common approach to this problem is to seek a foot trajectory ${}^{\mathcal{I}}H_F(t)$ such that its acceleration or jerk[2] is minimized. Since all the whole-body control layer implementations presented in Part II assume that frame velocity is expressed in *mixed representation* – Section 2.2.8, we split the problem of finding an optimal Cartesian trajectory into two subproblems, namely: evaluating a positional trajectory $p_F(t) \in \mathbb{R}^3$ and a rotation trajectory ${}^{\mathcal{I}}R_F(t) \in \mathrm{SO}(3)$.

Swing foot position

During the walking gait, the swing foot has to move from the initial position $p_F(t_0) = p_{F_0}$ to $p_F(t_N) = p_{F_N}$ to reach a desired maximum height. The instant in which a foot reaches its maximum altitude is called the apex time, t_{apex}.

We want to find a trajectory $p_F(t) \in \mathbb{R}^3$ such that:

- $p_F(t_0) = p_{F_0}$, $p_F(t_N) = p_{F_N}$ and $p_F(t_{\text{apex}}) = p_{F_{\text{apex}}}$
- the integral of a suitable Lagrangian function $\mathcal{L}(t)$ is minimized, i.e.,

$$\text{minimize} \int_{T_0}^{T_N} \mathcal{L}(t)\,\mathrm{d}\,t, \tag{10.2.4}$$

  where $\mathcal{L}(t) \geq 0$.

Depending on the choice of the objective function, we can design a minimum acceleration or a minimum jerk trajectory.

Minimum acceleration trajectory If we set the Lagrangian function $\mathcal{L}(t)$ as the squared-norm of the linear acceleration:

$$\mathcal{L} = \ddot{p}_F^{\top}\ddot{p}_F, \tag{10.2.5}$$

[2]We denote with *jerk* the rate at which an object's acceleration changes with respect to time

it is possible to show that a trajectory composed by the concatenation of the 3rd order polynomial functions

$$s_i : p_F(t) = a_{i,3}t^3 + a_{i,2}t^2 + a_{i,1}t + a_{i,0}, \tag{10.2.6}$$

is a minimum acceleration trajectory. Here the coefficients $a_{i,j}$ are chosen to satisfy the position and velocities boundary conditions for each subtrajectory, The complete foot position minimum acceleration trajectory is finally computed by attaching the 3rd-order polynomial functions (10.2.6) as

$$p_F(t) = \begin{cases} a_{0,3}t^3 + a_{0,2}t^2 + a_{0,1}t + a_{0,0} & \text{if } t_0 \leq t \leq t_{\text{apex}} \\ a_{1,3}t^3 + a_{1,2}t^2 + a_{1,1}t + a_{1,0} & \text{if } t_{\text{apex}} < t \leq t_N \end{cases} \tag{10.2.7}$$

Appendix D.2 presents all the passages required to evaluate a minimum acceleration trajectory in $\mathbb{R}^n$.

Minimum jerk trajectory If we set the Lagrangian function $\mathcal{L}(t)$ as the squared-norm of the linear jerk:

$$\mathcal{L} = \dddot{p}_F^\top \dddot{p}_F \tag{10.2.8}$$

we notice that a trajectory composed by the concatenation of a 5th order polynomial functions:

$$s_i : p_F(t) = a_{i,5}t^5 + a_{i,4}t^4 + a_{i,3}t^3 + a_{i,2}t^2 + a_{i,1}t + a_{i,0}, \tag{10.2.9}$$

minimizes the jerk. The coefficients $a_{i,j}$ are chosen to satisfy the boundary conditions of position, velocity and acceleration. Finally, the complete foot position minimum acceleration trajectory is computed by attaching 5th-order polynomial functions as:

$$p_F(t) = \begin{cases} a_{0,5}t^5 + a_{0,4}t^4 + a_{0,3}t^3 + a_{0,2}t^2 + a_{0,1}t + a_{0,0} & \text{if } t_0 \leq t \leq t_{\text{apex}} \\ a_{1,5}t^5 + a_{1,4}t^4 + a_{1,3}t^3 + a_{1,2}t^2 + a_{1,1}t + a_{1,0} & \text{if } t_{\text{apex}} < t \leq t_N \end{cases} \tag{10.2.10}$$

Appendix D.3 presents all the passages required to evaluate a minimum jerk trajectory in $\mathbb{R}^n$.

Swing foot orientation

During the walking gait, the swing foot rotates from the initial orientation ${}^{\mathcal{I}}R_F(t_0) = {}^{\mathcal{I}}R_{F_0}$ to ${}^{\mathcal{I}}R_F(t_N) = {}^{\mathcal{I}}R_{F_N}$. Similarly to what we discussed for the generation of position trajectory, we now aim to compute a trajectory ${}^{\mathcal{I}}R_F(t)$ such that

- ${}^{\mathcal{I}}R_F(t_0) = {}^{\mathcal{I}}R_{F_0}$ and ${}^{\mathcal{I}}R_F(t_N) = {}^{\mathcal{I}}R_{F_N}$
- a suitable Lagrangian function $\mathcal{L}(t)$ is minimized.

We can now construct a minimum acceleration or a minimum jerk trajectory depending on the objective function, $\mathcal{L}(t)$.

Minimum acceleration trajectory If we set the Lagrangian function $\mathcal{L}(t)$ as the squared-norm of the right trivialized (inertial frame) angular acceleration ${}^{\mathcal{I}}\dot{\omega}_{\mathcal{I},F}$:

$$\mathcal{L} = {}^{\mathcal{I}}\dot{\omega}_{\mathcal{I},F}^{\top}\,{}^{\mathcal{I}}\dot{\omega}_{\mathcal{I},F} \tag{10.2.11}$$

and we ask for a zero initial and final angular velocity, it is possible to show that the

$${}^{\mathcal{I}}R_F(t) = \exp\Big(s(t-t_0)\log\Big({}^{\mathcal{I}}R_F(t_N)\,{}^{\mathcal{I}}R_{F_0}^{\top}\Big)\Big)\,{}^{\mathcal{I}}R_{F_0}, \tag{10.2.12}$$

where $s(\tau)$ is given by

$$s(\tau) = \frac{3}{(t_N - t_0)^2}\tau^2 - \frac{3}{(t_N - t_0)^3}\tau^3, \tag{10.2.13}$$

minimizes the integral of the Lagrangian function (10.2.11) [Lynch and Park, 2017, Section 9.2][Park and Ravani, 1997; Žefran et al., 1998, 1999]

In Appendix E, we present the theory behind the minimum acceleration trajectory in SO(3).

Minimum jerk trajectory If we set the Lagrangian function $\mathcal{L}(t)$ as the squared norm of the right trivialized (inertial frame) angular jerk ${}^{\mathcal{I}}\ddot{\omega}_{\mathcal{I},F}$:

$$\mathcal{L} = {}^{\mathcal{I}}\ddot{\omega}_{\mathcal{I},F}^{\top}\,{}^{\mathcal{I}}\ddot{\omega}_{\mathcal{I},F}. \tag{10.2.14}$$

and we ask for a zero initial and final angular velocity and acceleration, it is possible to show that

$${}^{\mathcal{I}}R_F(t) = \exp\Big(s(t-t_0)\log\Big({}^{\mathcal{I}}R_F(t_N)\,{}^{\mathcal{I}}R_{F_0}^{\top}\Big)\Big)\,{}^{\mathcal{I}}R_{F_0} \tag{10.2.15}$$

where $s(\tau)$ is given by

$$s(\tau) = \frac{6}{(t_N - t_0)^5}\tau^5 - \frac{15}{(t_N - t_0)^4}\tau^4 + \frac{10}{(t_N - t_0)^3}\tau^3, \tag{10.2.16}$$

minimizes the integral of the Lagrangian function (10.2.14) [Liu et al., 2017; Žefran et al., 1999]

Differently from (10.2.12) and (10.2.13), to prove (10.2.15) and (10.2.16) we need to exploit the mathematical tools of *Differential Geometry*. Since this topic is outside the scope of the thesis, we avoid presenting the rigorous proof. The interested reader can refer to the extensive literature, among which it is worth mentioning [Dubrovin et al., 1984; Needham, 2021; Pressley, 2010; Selig, 2007; Žefran et al., 1998, 1999]

10.2.3 Simplified model control layer

As introduced in Section 4.4, the simplified model (4.4.14a) shows that the CoM asymptotically converges to a constant DCM, while the DCM has an unstable first-order dynamics (4.4.14b). In this section, we present control laws that aim at stabilizing the DCM dynamics by assuming the location of the ZMP x_{ZMP} as the control input.

This stabilization problem has been tackled by designing and comparing *instantaneous* and *predictive* (MPC) control laws. Unlike the MPC, the instantaneous control law does not solve any optimization problem, and it uses only the current desired and actual position of the DCM to evaluate its output.

DCM instantaneous control

Differently from Englsberger et al. [2015a], we propose an instantaneous control law that does not perform a cancellation of the unstable system dynamics, which, in practice, is never perfect and can lead to system instabilities. More precisely, we chose

$$x^*_{\text{ZMP}} = \xi^{\text{ref}} - \frac{1}{\zeta}\dot{\xi}^{\text{ref}} + K_p^{\xi}\left(\xi - \xi^{\text{ref}}\right) + K_i^{\xi}\int \xi - \xi^{\text{ref}}\,\mathrm{d}\,t. \tag{10.2.17}$$

where $K_p^{\xi} > I_2$ and $K_i^{\xi} > 0_{2\times 2}$

Applying the control input (10.2.17) to the system (10.1.15) leads to the following closed-loop dynamics:

$$\dot{\xi} - \dot{\xi}^{\text{ref}} = \zeta(I_2 - K_p^{\xi})(\xi - \xi^{\text{ref}}) - \zeta K_i^{\xi}\int \xi - \xi^{\text{ref}}\,\mathrm{d}\,t. \tag{10.2.18}$$

It is easy to show that the DCM error and its integral converge asymptotically to zero. The above law (10.2.17) is very simple to implement and guarantees the tracking of the desired DCM. The main limitation is that it may not ensure the feasibility of the gait, since the position of the ZMP may exit the support polygon and, consequently, the ZMP contact feasibility criterion [Vukobratovic and Juricic, 1969] may not be guaranteed. Furthermore, the above instantaneous control law does not take into account the desired DCM trajectory planned for the future.

DCM predictive control

In order to overcome these issues, we design a model predictive controller by following the work of Krause et al. [2012]. The control objective is achieved by framing the controller as a constrained optimization problem composed of three main elements, namely: *the prediction model*, an *objective function* and a set of *inequality constraints*.

Prediction model In the MPC framework, we consider the dynamics of DCM (10.1.15) as a prediction model. Equation (10.1.15) is discretized supposing piecewise constant ZMP trajectories. By applying the *Zero order hold* technique presented in Section 5.5.1, the discrete DCM dynamics writes as follows:

$$\begin{aligned}\xi_{k+1} &= F\xi_k + Gx_{\mathrm{ZMP}_k} \\ &= \begin{bmatrix} e^{\zeta T} & 0 \\ 0 & e^{\zeta T} \end{bmatrix} \xi_k + \begin{bmatrix} 1-e^{\zeta T} & 0 \\ 0 & 1-e^{\zeta T} \end{bmatrix} x_{\mathrm{ZMP}_k}.\end{aligned} \tag{10.2.19}$$

where T is the fixed sampling time.

Objective function The objective function is defined in terms of two main tasks, namely a tracking task and a control input regularization task.

To follow a desired DCM trajectory, we minimize the weighed norm of the error between the robot DCM and the desired trajectory as:

$$\Psi_\xi = \frac{1}{2} \left\| \xi - \xi^{\mathrm{ref}} \right\|^2_{\Lambda_\xi}, \tag{10.2.20}$$

where Λ_ξ is a positive definite diagonal matrix. On the other hand, to reduce the rate of change of the ZMP, we minimize the ZMP derivative by considering the following task:

$$\Psi_{\mathrm{ZMP}} = \frac{1}{2} \left\| \dot{x}_{\mathrm{ZMP}} \right\|^2_{\Lambda_{\mathrm{ZMP}}}. \tag{10.2.21}$$

With Λ_{ZMP} a diagonal positive matrix.

Inequality constraint In order to ensure that the stance foot does not rotate around one of its edges, the desired ZMP must satisfy the contact stability criterion [Vukobratović et al., 2004; Vukobratovic and Juricic, 1969]. This is verified by seeking for a ZMP inside foot print, if the robot is in single support, or inside the convex hull of the two feet, in the double support scenario. The $\mathcal{H}$-rep (Section 5.1.2) is given by the following set of inequalities:

$$A_c x_{\text{ZMP}} \preceq b_c, \tag{10.2.22}$$

where A_c and b_c are time-variant and their dimension depends on the type of support.

MPC formulation Combining the cost functions (10.2.20) and (10.2.21), with the prediction model (10.2.19) and the inequality constraints (10.2.22), we formulate the MPC as an optimization problem. The MPC problem is transcribed using the Direct Single Shooting method with a constant sampling time T – Section 5.5.2. The desired ZMP, x^*_{ZMP} is generated by applying the receding horizon principle [Mayne and Michalska, 1990] – Section 5.6. The MPC formulation is finally obtained by solving the following optimization problem:

$$\underset{x_{\text{ZMP}_k},\, k=[0\ N]}{\text{minimize}} \sum_{k=0}^{N} (\Psi_\xi + \Psi_{\text{ZMP}}) \tag{10.2.23a}$$

$$\text{subject to } \xi_{k+1} = F\xi_k + G x_{\text{ZMP}_k} \tag{10.2.23b}$$

$$A_{c_k} x_{\text{ZMP}_k} \preceq b_{c_k} \tag{10.2.23c}$$

$$\xi_0 = \bar{\xi} \tag{10.2.23d}$$

$$x_{\text{ZMP}_{-1}} = \bar{x}_{\text{ZMP}}. \tag{10.2.23e}$$

where $\bar{x}_{\text{ZMP}}$ is the desired ZMP calculated in the previous control iteration and $\bar{\xi}$ is the estimated DCM position.

Since the cost function is a quadratic positive function and the constraints are linear, the optimal control problem is quadratic and can be transcribed into a strictly convex quadratic programming problem (QP) (Section 5.4) of the form:

$$\begin{aligned} &\underset{w}{\text{minimize}} \ \frac{1}{2} w^\top H_k w + g_k^\top w \\ &\text{subject to } A^i_{c_k} w \preceq b^i_{c_k} \end{aligned}$$

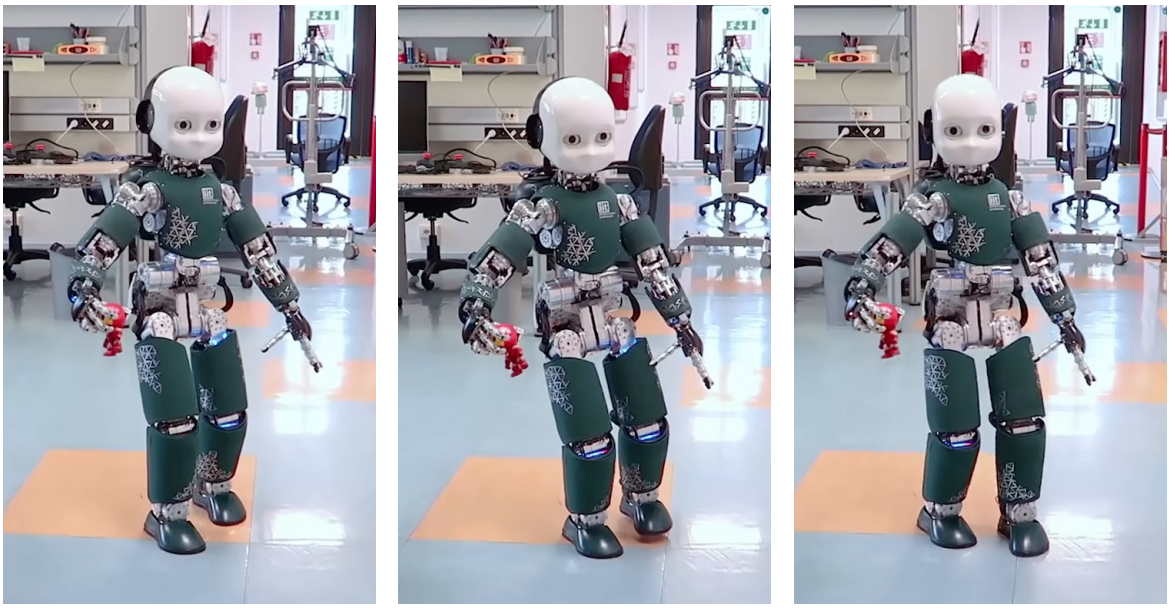


Figure 10.8 The iCub robot walks with the 3 layer controller architecture of Figure 10.1.

Here, the optimization variables are stacked inside the vector

$$w^\top = \begin{bmatrix} x_{\text{ZMP}_0}^\top & \dots & x_{\text{ZMP}_{N-1}}^\top \end{bmatrix}. \tag{10.2.24}$$

The Hessian matrix H_k and the gradient vector g_k derive from (10.2.23a). The inequality constraint matrix $A_{c_k}^i$ and vector $b_{c_k}^i$ embed the ZMP constraint (10.2.23c) and the prediction model (10.2.23b).

ZMP-CoM Controller

Independently of the chosen DCM controller, namely the controller in Section 10.2.3 or in 10.2.3, one obtains the desired position and velocity of ZMP and CoM to stabilize. As a consequence, another control loop is needed after the DCM controller. Here, we choose the proposed control law [Choi et al., 2007], i.e.:

$$\dot{x}_{\text{LIP}}^* = \dot{x}_{\text{LIP}}^{\text{ref}} - K_{\text{ZMP}}(x_{\text{ZMP}}^{\text{ref}} - x_{\text{ZMP}}) + K_{\text{LIP}}(x_{\text{LIP}}^{\text{ref}} - x_{\text{LIP}}), \tag{10.2.25}$$

where $K_{\text{LIP}} > \zeta I_2$ and $0_2 < K_{\text{ZMP}} < \zeta I_2$.

10.3 Results

In this section, we present experiments obtained with several implementations of the simplified model controllers, namely: the *instantaneous* and the *predictive* controllers.

To benchmark the different simplified model controllers, we test the algorithms on the iCub humanoid robot v2.7 – Section 1.1.1. We attach the simplified control layer to the three-layer controller architecture shown in Figure 6.1. In this framework, the whole-body control layer implements the kinematics-based whole-body QP presented in section 7.1. Figure 10.8 shows the humanoid robot iCub walking with the simplified models controller presented in this chapter.

The control architecture runs on the iCub head's computer, namely a 4-th generation Intel ® Core i7 @ 1.7 GHz. In any of its implementations, the architecture takes (on average) less than 3 ms to evaluate its output. The code is open source completely developed in C++: `https://github.com/robotology/walking-controllers`. The MPC problem presented in Section 10.2.3 is solved using the OSQP [Stellato et al., 2018a] library [3].

Table 10.1 summarizes the maximum velocities achieved using the different implementations of the control architecture. In particular, the labels *instantaneous* and *predictive* mean that the associated layer generates its output considering inputs and references either at the single time t or for a time window, respectively. The labels, *velocity* and *position* control, instead, mean that the layer outputs are either desired joint velocities or position, respectively – see Section 7.1.3.

Let us remark that all the implemented control architectures exploit the controller presented in Section 10.2.3 to attempt the stabilization of the desired center of pressure and desired center of mass position and velocity. The performance of this controller is highly dependent on the gains K_{zmp} and K_{com}. In particular, we observed that the

Table 10.1 Maximum forward straight walking velocities achieved using different implementations of the control architecture.

Simplified Model Control	Whole-Body QP Control	Max Straight Velocity (m/s)
Predictive	Velocity	0.1563
Predictive	Position	0.1645
Instantaneous	Velocity	0.1809
Instantaneous	Position	0.3372

[3] Since our code is written in pure C++, the QP problem is written by means of `osqp-eigen` a C++ wrapper for OSQP `https://github.com/robotology/osqp-eigen`

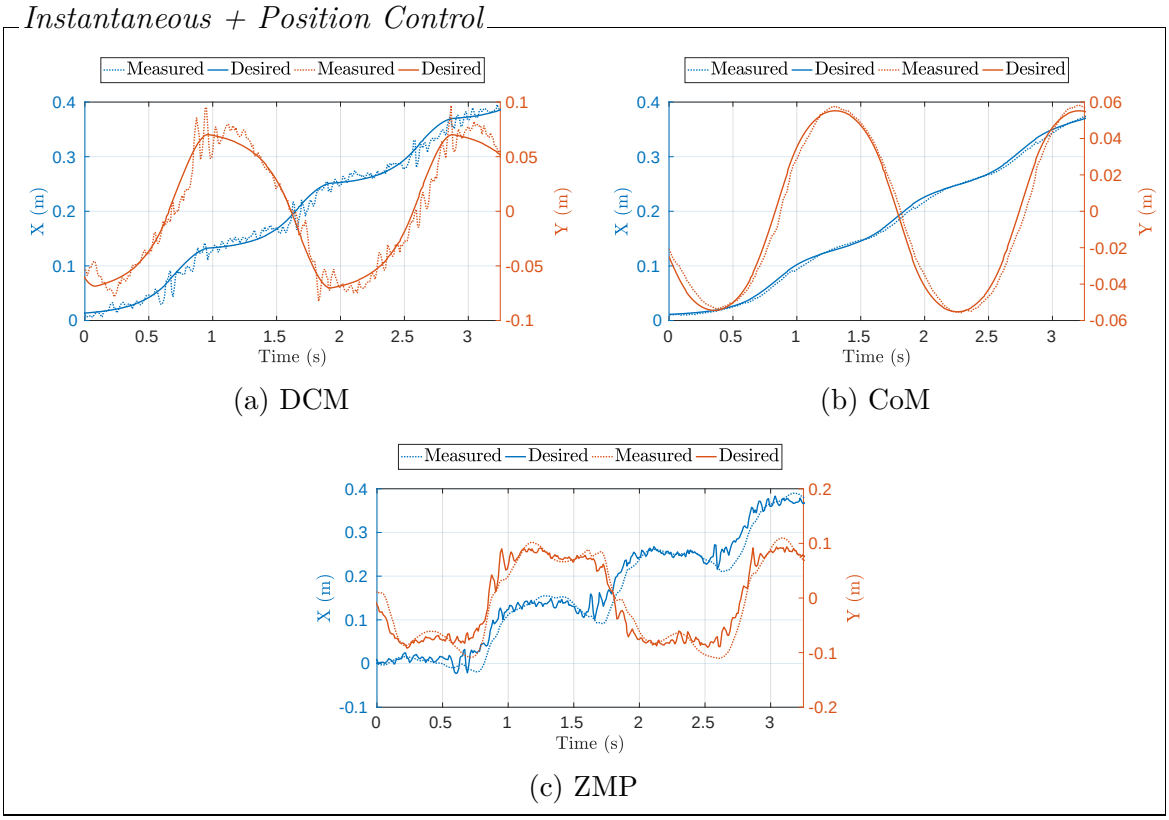


Figure 10.9 Tracking of the DCM (a), CoM (b) and ZMP (c) using the instantaneous controller with the whole-body controller as position control. Walking velocity: $0.19\,\mathrm{m\,s^{-1}}$.

gains in achieving good tracking during standing and walking were not the same. For this reason, we implemented a gain-scheduling technique depending on whether the robot is walking or standing. The transition between the two sets of gains is smoothed with a minimum jerk trajectory [Pattacini et al., 2010].

To compare the simplified models controllers, we decided to perform two main experiments. These two experiments represent the maximum robot velocity that has been achieved with all architectures and the maximum velocity achieved with a specific architecture only – see Table 10.1. That is,

- **Experiment 1**: a forward robot speed of $0.1563\,\mathrm{m\,s^{-1}}$;
- **Experiment 2**: a forward robot speed of $0.3372\,\mathrm{m\,s^{-1}}$.

We compare the control laws (10.2.17) and (10.2.23), which both generate a (desired) center of pressure that attempts to stabilize the desired DCM. To simplify the

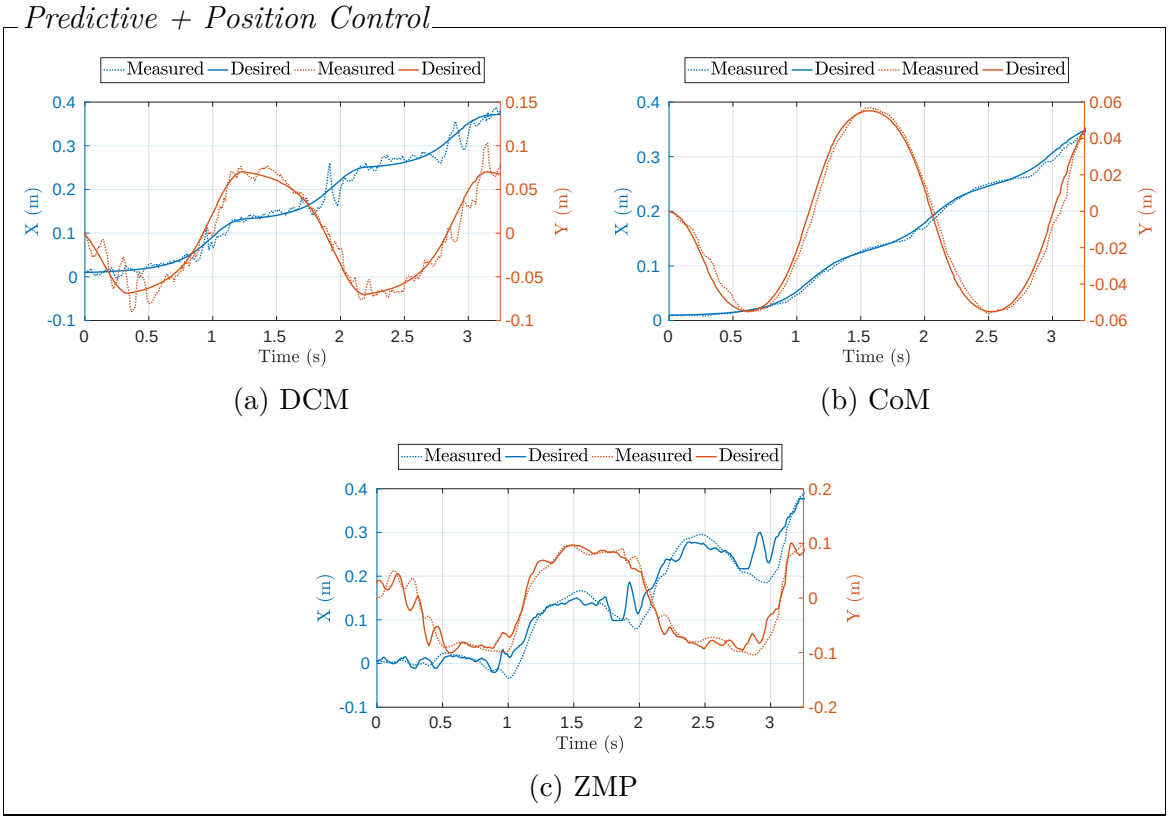


Figure 10.10 Tracking of the DCM (a), CoM (b) and ZMP (c) using the MPC and the whole-body controller as position control. Walking velocity: $0.19\,\mathrm{m\,s^{-1}}$.

comparison, the controller of the *whole-body QP layer* is kept fixed in this section, and we show and discuss only the results when the robot is position controlled. A complete comparison of the kinematics-based whole-body controllers is presented in Section 7.3. In the following experiments, we set the time horizon of the predictive control to 2 s.

10.3.1 Experiment 1: a forward robot speed of 0.1563 m s^{-1}

Figures 10.9a and 10.10a show the DCM tracking performances obtained with the instantaneous and predictive controllers, respectively. Both controllers seem to show good tracking performances, and the DCM error is kept below 5 cm in both cases. Note, however, that the instantaneous controller induces faster variations of the measured DCM. This contributes to the overall higher vibrations of the robot. One of the reasons for this variation is that the instantaneous controller (10.2.17) injects a (desired) center of pressure proportional to the measured DCM, which in turn contains the center of mass velocity. To mitigate this, we may filter the joint velocities appropriately.

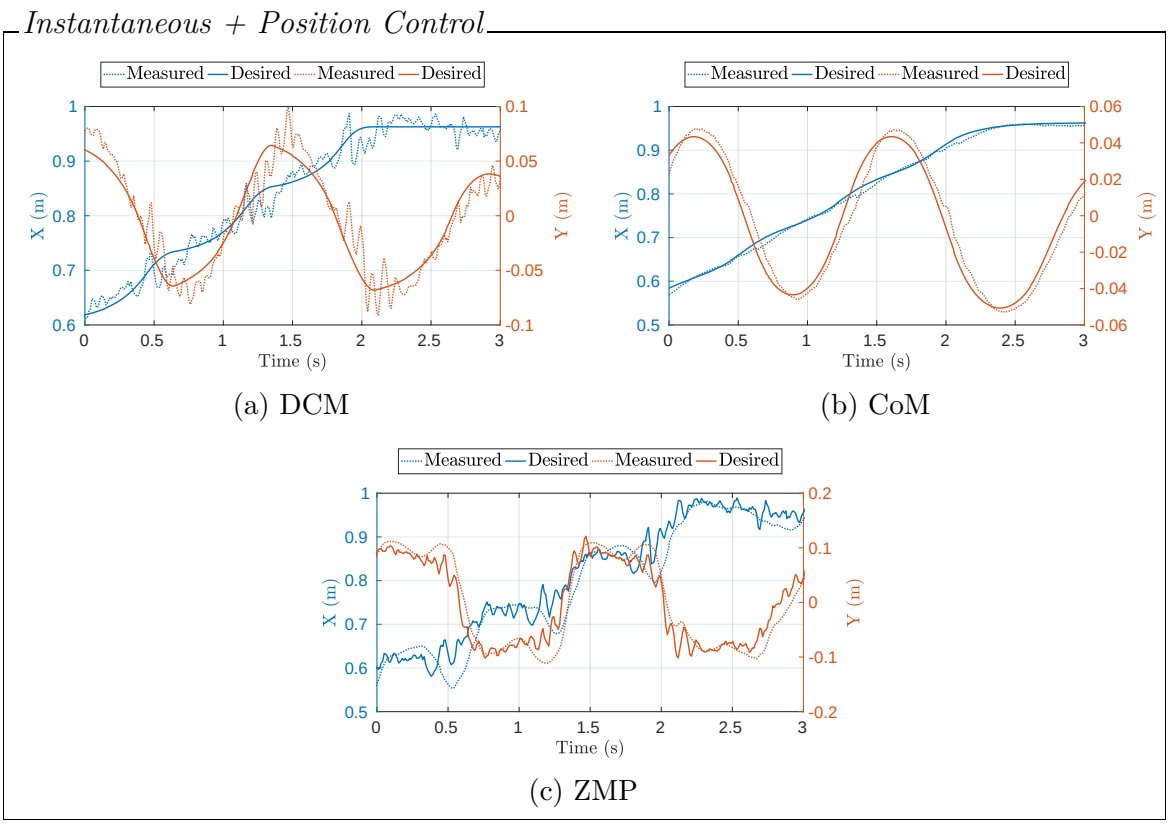


Figure 10.11 Tracking of the DCM (a), CoM (b) and ZMP (c) with the instantaneous and whole-body QP control as position. Walking velocity: $0.41\,\mathrm{m\,s^{-1}}$.

However, in our case, the joint velocities were not filtered to avoid delays in the measured DCM. Our experience showed that adding a filter to joint velocities is not an easy task, and we did not find the right trade-off for obtaining overall performance improvements.

Figures 10.9b and 10.10b present CoM tracking performances, which are mainly dependent on the ZMP-CoM controller (10.2.25). This controller receives the desired DCM values from the *simplified model control* layer, which are obtained with the instantaneous or predictive controllers. In both cases, the CoM error is kept below 2 cm. Figures 10.9c and 10.10c represent the ZMP tracking performance, which is still mainly dependent on the ZMP-CoM controller (10.2.25). It is important to note that the desired ZMP is smoother when the *simplified model control* uses the predictive law (10.2.23) to generate it. Indeed, this is a tunable property that depends on the associated weight in the cost function of the MPC problem. Although this smoother behavior contributes to less robot vibrations, overall robot performance became less reactive and, consequently, less robust to robot falls. Although the extensive hand-made

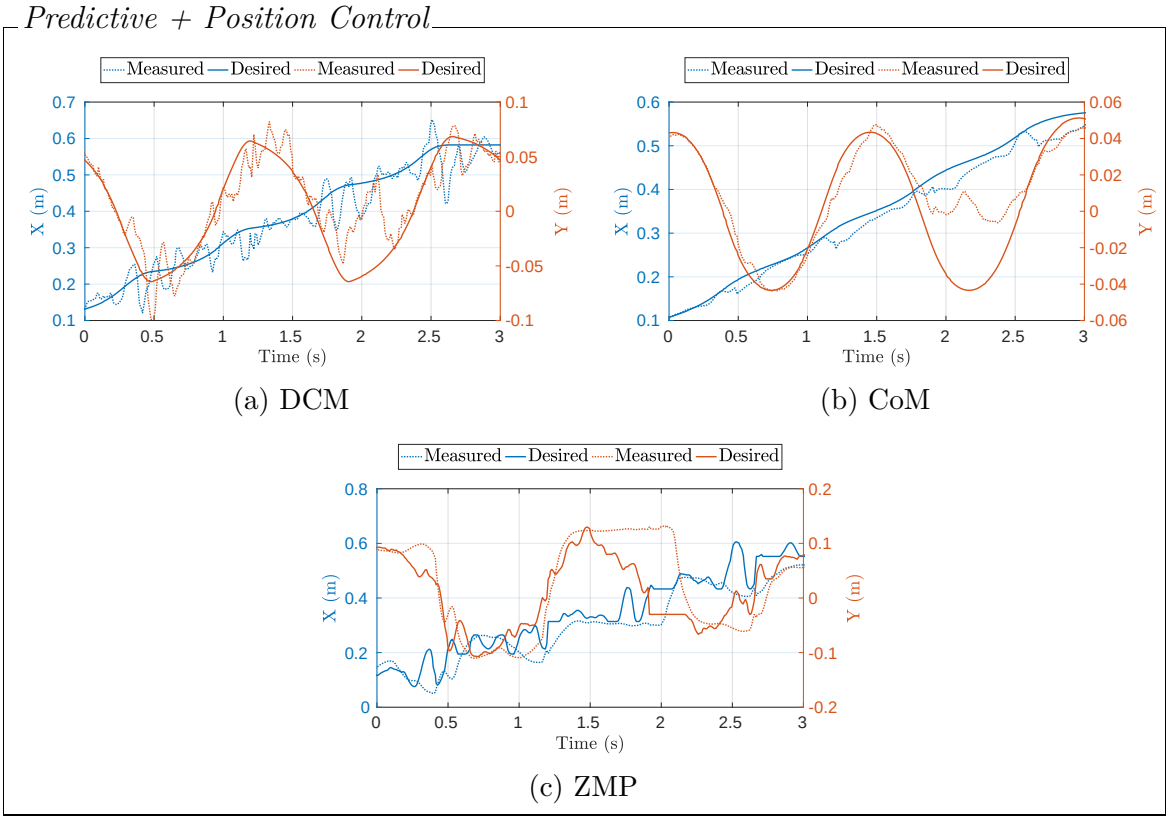


Figure 10.12 Tracking of the DCM (a), CoM (b) and ZMP (c) with the predictive and whole-body QP control as position control. At $t \approx 2\,\mathrm{s}$, the robot falls down. Walking velocity: $0.41\,\mathrm{m\,s^{-1}}$.

tuning, we were not able to increase the robot velocity when the *simplified model control* used the predictive law (10.2.23).

10.3.2 Experiment 2: a forward robot speed of 0.3372 m s^{-1}

At a robot's desired walking speed of $0.3372\,\mathrm{m\,s^{-1}}$, there is initially no significant difference between the DCM tracking obtained with instantaneous and predictive control laws – see Figures 10.12a and 10.11a for $t < 1.5\,\mathrm{s}$. However, fast robot walking velocities require fast variations of the desired CoM and ZMP. This fast variation degrades the performance of the predictive controller around $t = 1.5\,\mathrm{s}$ – see Figure 10.12c. Clearly, these bad performances, in turn, induce poor tracking of the DCM shown in Figure 10.12a at $t \approx 2\,\mathrm{s}$, and consequently the robot falls. At this point, one is tempted to increase the gain K_{ZMP} of the controller (10.2.25), which shall induce a better tracking of the ZMP. Unfortunately, this leads to higher robot oscillations induced by the noise on the estimated ZMP. And, as a consequence, the robot falls.

We can conclude that the *predictive simplified control* is much less robust than the *instantaneous simplified control* with respect to ZMP tracking errors. Adding a low-pass filter to the ZMP measurement may improve the overall performance. However, in our case, adding filters led to slower system response and, consequently, to the robot falling.

10.4 Conclusions

This chapter contributes to the benchmarking of different implementations of state-of-the-art control simplified model controllers for humanoid robot locomotion. In particular, we proposed two simplified controllers which exploit the concept of the Divergent Component of Motion. In particular, we discussed the results obtained with the predictive and instantaneous controller implementations. Furthermore, we compare the performance obtained by controlling the robot in position and velocity.

We show that the proposed instantaneous controllers combined with the robot position control allowed us to achieve a desired walking speed of $0.3372\,\mathrm{m\,s^{-1}}$, which is the highest walking velocity achieved by the iCub robot v2.7.

A possible future work is the implementation of an online footstep adjustment algorithm [Griffin and Leonessa, 2016; Shafiee et al., 2019; Shafiee-Ashtiani et al., 2017b] This will increase the overall robustness in case of a disturbance acting on the robot.

It is worth mentioning that the presented simplified model controller relies on the assumption of a constant height of the center of mass. In the next chapter, we will attempt to loosen this hypothesis by designing a controller that considers the reduced centroidal dynamics of the system. By modeling the system using a reduced model instead of a simplified one, we will achieve highly dynamic robot motions that can be performed online. Furthermore, the footstep adjustment is considered in the centroidal dynamics stabilization problem, so it is not required to design an ad-hoc block for this feature.

Chapter 11

Non-Linear Centroidal Model Predictive Controller

In Chapter 10 we benchmarked several implementations of simplified model controllers. Modeling the robot with a simplified model may restrict the motion of the robot to a well-defined subset of motion primitives. For example, when the robot is described using a LIPM (see Section 4.1), the CoM height must be kept constant and at least one foot must be in contact with the environment. This chapter attempts at lowering this limitation by presenting a Non-Linear Model Predictive Controller for humanoid robot locomotion with online step adjustment capabilities. The proposed controller considers the centroidal dynamics of the system to compute the desired contact forces, torques, and contact locations. Differently from bipedal walking architectures based on simplified models, the presented approach considers the reduced centroidal model, thus allowing the robot to perform highly dynamic movements while keeping the control problem still treatable online. We show that the proposed controller can automatically adjust the contact location in both single- and double-support phases. The overall approach is then tested with a simulation of one-leg and two-leg systems performing jumping and running tasks, respectively. Finally, we validate the proposed controller in a specific version of the three-layer controller architecture of Figure 6.1. However, differently from the original design, the simplified model controller is replaced by the *Reduced model control* layer that implements the controller presented in this chapter – see Figure 11.1. The overall architecture is finally tested on the position-controlled Humanoid Robot iCub v3 – Section 1.1.2. The results show that the proposed strategy prevents the robot from falling while walking and pushed with external forces up to 40 Newton for 1 second applied to the robot arm.

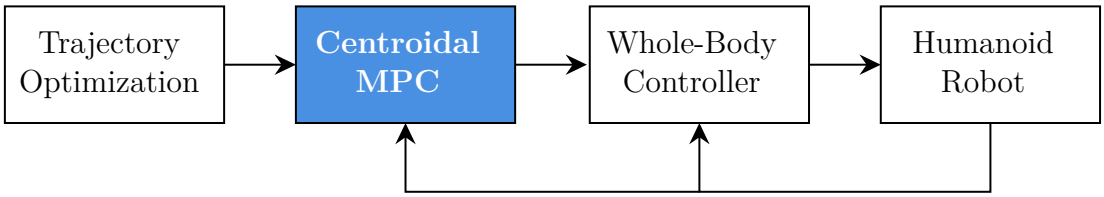


Figure 11.1 Centroidal MPC embedded into a three layer controller architecture.

The chapter is organized as follows. Section 11.1 introduces the control problem. Section 11.2 presents the simulation results for different kinds of floating base systems and on the position-controlled Humanoid Robot iCub v3 – Section 1.1.2. Finally, Section 11.3 concludes the chapter.

The content of this chapter appears in:

> Romualdi, G., Dafarra, S., L'Erario, G., Sorrentino, I., Traversaro, S., and Pucci, D. (2022a). Online Non-linear Centroidal MPC for Humanoid Robot Locomotion with Step Adjustment. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10412–10419. IEEE
>
> Video `https://www.youtube.com/watch?v=u7vCgE2w_vY9`
> GitHub `ami-iit/paper_romualdi_2022_icra_centroidal-mpc-walking`

11.1 Centroidal model predictive controller

Let us assume the presence of a high-level contact planner that generates only the contact location and times, for example, the one presented in Section 10.1.2 [Dafarra et al., 2018]. The control objective of the reduced model is to implement a control law that generates feasible contact wrenches ${}_{\mathcal{C}_i[\mathcal{I}]}\mathrm{f}_i$ and locations p_i while considering the centroidal dynamics of the floating base system and a nominal set of contact positions and timings. Here we denote by $\mathcal{C}_i[\mathcal{I}] = (p_i, \mathcal{I})$, the frame placed on the robot contact location p_i and oriented as the inertial frame $\mathcal{I}$, where $i \in \mathbb{N}$ such that $1 \leq i \leq n_c$, with n_c is the number of admissible contacts. The control problem is formulated using the Model Predictive Control (MPC) framework.

The control objective is achieved by framing the controller as a constrained optimization problem composed of three main elements, namely: the *prediction model*, an *objective function*, and a set of *inequality constraints*.

What follows is a complete description of the task composing the optimal control problem. In mode detail, Section 11.1.1 introduces the model considered in the controller as prediction. Section 11.1.2 presents the terms that describe the control objective. Finally, in Section 11.1.3 we discuss the inequality constraints required to guarantee a feasible walking pattern.

11.1.1 Prediction model

Given a frame $\bar{G} = (x_{\text{CoM}}, [\mathcal{I}])$, the centroidal momentum rate of change ${}_{\bar{G}}\dot{h}$ balances the external spatial force applied to the robot (3.4.6):

$$ {}_{\bar{G}}\dot{h} = \sum_{j=1}^{n_c} \begin{bmatrix} I_3 & 0_{3\times3} \\ (p_j - x_{\text{CoM}})\times & I_3 \end{bmatrix} {}_{\mathcal{C}_j[\mathcal{I}]}\mathrm{f}_j + m\bar{g}, \tag{11.1.1} $$

where $\bar{g} = \begin{bmatrix} 0 & 0 & -g & 0 & 0 & 0 \end{bmatrix}^\top$ is the 6D gravity acceleration vector.

We now recall that, given a rigid body in contact with a rigid environment and assuming that the contact surface is described by a rectangle. Then, the contact wrench acting on the rigid body is uniquely described by four pure forces acting on the corner of the contact surface [Caron et al., 2015]. Indeed, in the case of a rectangular contact surface, twelve variables define the six-dimensional wrench. Thanks to this choice, several contact configurations can be modeled independently, depending on the number of points in contact [Dafarra et al., 2020; Dai et al., 2014]. Given the relation between pure forces and contact wrench, we rewrite the centroidal dynamics (11.1.1) as follows:

$$ {}_{\bar{G}}\dot{h} = \sum_{i=1}^{n_c}\sum_{j=1}^{n_v} \begin{bmatrix} I_3 \\ (p_i + R_{\mathcal{C}_i}\, {}^{\mathcal{C}_i}p_{v_{i,j}} - x_{\text{CoM}})\times \end{bmatrix} f_{i,j} + m\bar{g}. \tag{11.1.2} $$

We denote by n_v the number of vertices associated with the contact surface, commonly $n_v = 4$. ${}^{\mathcal{C}_i}p_{v_{i,j}}$ is the position of the vertex j of the contact i expressed with respect to the frame associated with the contact surface $\mathcal{C}_i$. $f_{i,j}$ is the pure force applied to the vertex j of the contact i expressed in the inertial frame $\mathcal{I}$.

Assume a rigid body that interacts with the environment. The contact force is supposed to be non-null only if the point is in contact with the environment. The

condition is called *complementary condition* and writes as:

$$h(p_i)n(p_i)^\top f_i = 0, \tag{11.1.3}$$

where h computes the distance between the point p_i and the environment and $n(p_i)$ returns the normal direction to the contact surface at the point p_i.

Since the proposed controller assumes the knowledge of the contact sequence, it is possible to define the variable $\Gamma_i \in \{0, 1\}$ for each contact. Γ_i represents the contact state at a given instant. $\Gamma_i(t) = 0$ indicates that the contact i-th is not active at the time t, while, when $\Gamma_i(t) = 1$ the contact is active. Due to this assumption, it is not necessary to introduce the contact force complementary condition (11.1.3). In fact, considering the complementary condition in an optimization algorithm may cause problems for the nonlinear optimization solvers because the constraint Jacobian becomes singular, thus violating the linear independence constraint qualification (LICQ) on which most solvers rely – see Section 5.3.3 [Betts, 2010]. As a consequence of the introduction of Γ_i, (11.1.2) is rewritten as

$$\begin{aligned} {}_{\bar{G}}\dot{h} &= \sum_{i=1}^{n_c}\sum_{j=1}^{n_v} \begin{bmatrix} I_3 \\ (p_i + R_{\mathcal{C}_i}\, {}^{\mathcal{C}_i}p_{v_{i,j}} - x_{\mathrm{CoM}})\times \end{bmatrix} \Gamma_i f_{i,j} + m\bar{g} && \text{(11.1.4a)} \\ &= \mathcal{F}\left(p_i, x_{\mathrm{CoM}}, f_{i,j}\right). && \text{(11.1.4b)} \end{aligned}$$

In $\mathcal{F}$, we explicitly express the dependency on the unknown variables, namely: the contact location p_i, the CoM position x_{CoM} and the contact forces $f_{i,j}$. We finally notice that the centroidal angular momentum dynamics (11.1.4) is not convex in the variables $f_{i,j}$, p_i x_{CoM}, this fact may induce to complexity in treating the centroidal dynamics as a prediction model.

Since the MPC aims to compute the control outputs online, the optimal control problem formulation should be as general as possible in the number of active contact phases in the prediction windows. For this reason, we consider each contact location as a continuous variable subject to the following dynamics:

$$\dot{p}_i = (1 - \Gamma_i)v_i, \tag{11.1.5}$$

where we define $v_{\mathcal{C}_i}$ as the *contact velocity*[1]. To give the reader a better understanding of Equation (11.1.5), we can imagine that when the contact is active, that is, $\Gamma_i = 1$,

[1]We want to warn the reader that the term *contact velocity* could be misleading. Indeed, when the contact is active, its velocity is always assumed to be zero. v_i is just an auxiliary variable that

(11.1.5) becomes $\dot{p}_i = 0$. In other words, the contact location is kept constant if the contact is active.

11.1.2 Objective function

The objective function is defined in terms of several tasks. In the following sections, we discuss the contribution of each task.

Contact force regularization

Each contact link is subject to the effect of different contact forces. Since the net effect is given by the sum of all these forces, we want them to be as similar as possible. As a consequence, we add a task that weighs the difference of each contact force from the average for a given contact link:

$$\Psi_{f_{i,j}} = \frac{1}{2} \left\| \frac{1}{n_v} \sum_{w=1}^{n_v} f_{i,w} - f_{i,j} \right\|^2_{\Lambda_{f_{i,j}}}, \tag{11.1.6}$$

where $\Lambda_{f_{i,j}}$ is a positive definite diagonal matrix, i.e., $\Lambda_{f_{i,j}} \succeq 0$. $f_{i,w}$ is given by the sum of all the forces acting by the environment on the link i in contact as

$$f_{i,w} = \sum_{w=1}^{n_v} f_{i,j}. \tag{11.1.7}$$

To prevent the controller from providing solutions with a huge rate of change of the contact force, we decided to minimize the contact force derivative by considering the following task:

$$\Psi_{\dot{f}_{i,j}} = \frac{1}{2} \left\| \dot{f}_{i,j} \right\|^2_{\Lambda_{\dot{f}_{i,j}}}, \tag{11.1.8}$$

where $\Lambda_{\dot{f}_{i,j}}$ is a positive defined diagonal matrix. In our control problem, the time derivative of the contact force $\dot{f}_{i,j}$ is not considered as an optimization variable. To overcome this limitation, we decided to replace $\dot{f}_{i,j}$ with its first-order numerical approximation:

$$\dot{f}_{i,j} = \frac{f_{i,j}[k] - f_{i,j}[k-1]}{\mathrm{d}\,t}. \tag{11.1.9}$$

Here, $\mathrm{d}\,t$ is the controller sampling rate. $f_{i,j}[k]$ $f_{i,j}[k-1]$ represent, respectively, the contact force at the time instant $t_0 + k\,\mathrm{d}\,t$ and $t_0 + (k-1)\,\mathrm{d}\,t$. Substituting (11.1.9) into

allows us to treat the contact location as a continuous variable; v_i will be different from zero only when the contact is not active.

the task (11.1.8), we obtain the final formulation of the contact force rate-of-change regularization task.

$$\Psi_{\dot{f}_{i,j}} = \frac{1}{2} \left\| \frac{f_{i,j}[k] - f_{i,j}[k-1]}{\mathrm{d}\,t} \right\|^2_{\Lambda_{\dot{f}_{i,j}}} . \tag{11.1.10}$$

Centroidal momentum tracking task

To follow a desired centroidal momentum trajectory, we minimize the weighted norm of the error between the robot's centroidal quantities and the desired nominal trajectory:

$$\Psi_h = \frac{1}{2} \left\| {}_{\bar{G}}h^{\omega^n} - {}_{\bar{G}}h^{\omega} \right\|^2_{\Lambda_h} + \frac{1}{2} \left\| x^n_{\mathrm{CoM}} - x_{\mathrm{CoM}} \right\|^2_{\Lambda_{\mathrm{CoM}}} , \tag{11.1.11}$$

where Λ_h and Λ_{CoM} are two positive definite diagonal matrices. The desired angular momentum ${}_{\bar{G}}h^{\omega^n}$ and CoM position x^n_{CoM}, should be considered as regularization terms. As a consequence, their purpose is to *drive* the optimal control problem to the desired feasible solution. In other words, ${}_{\bar{G}}h^{\omega^n}$ and x^n_{CoM} could not be a dynamically consistent trajectory. In our specific scenario we consider ${}_{\bar{G}}h^{\omega^n}$ always equal to zero, while x^n_{CoM} is a fifth-order spline passing through the nominal contact locations, whose initial and final velocity and acceleration are zero.

Contact location regularization

To reduce the difference between the nominal contact location and the one computed by the controller, we consider the following regularization task:

$$\Psi_{p_i} = \frac{1}{2} \left\| p^n_i - p_i \right\|^2_{\Lambda_{p_i}} . \tag{11.1.12}$$

Here $p_{\mathcal{C}^n_i}$ is the nominal contact position provided by a high-level planner, and Λ_{p_i} is a positive definite diagonal matrix.

11.1.3 Inequality constraints

This section contains the two sets of inequality constraints considered in the optimal control problem.

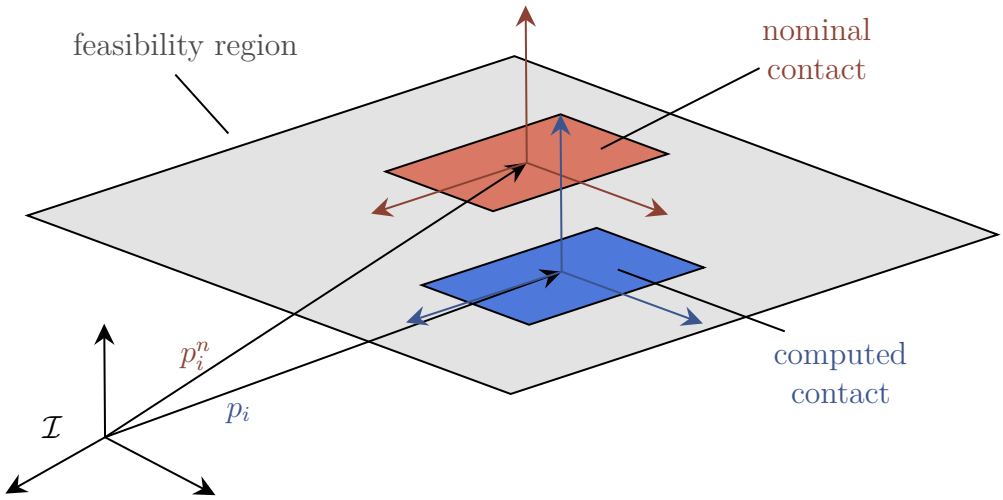


Figure 11.2 The contact feasibility region.

Contact force feasibility

Similar to what we discuss for the whole-body controllers 7.2.1 and 8.2.1, to guarantee a *weakly stable contact* [Caron et al., 2015], the contact force should belong to a second-order cone (4.1.1). However, to simplify the friction constraint, the friction cone is often approximated by the conic combination of n vectors – Section 5.1.2. The half-space representation of the friction cone approximation is given by a set of linear inequalities of the form

$$A\, {}^{\mathcal{C}_i}R_{\mathcal{C}_i[\mathcal{I}]}\, f_{i,j} \preceq b. \tag{11.1.13}$$

Here A and b are constants and depend only on the static friction coefficient.

Contact location constraint

The proposed controller aims to compute the contact location. In particular, the new contact position should belong to the feasibility region described by a rectangle containing the nominal contact location – Figure 11.2.

We introduce the contact location constraint as

$$l_b \preceq {}^{\mathcal{C}_i}R_{\mathcal{I}}(p_i^n - p_i) \preceq u_b, \tag{11.1.14}$$

where l_b and u_b are the lower and upper bounds of the rectangle represented in the frame attached to the contact $\mathcal{C}_i$.

11.1.4 MPC formulation

Combining the set of tasks presented in Section 11.1.2, with the prediction models of Section 11.1.1 and the inequality constraints described in Section 11.1.3), we formulate the MPC as an optimization problem.

The MPC problem is solved using a Direct Multiple Shooting method [Betts, 2010] – Section 5.5.2. We discretize the centroidal dynamics (11.1.4) and the contact location dynamics (11.1.5) by applying the Forward Euler technique with a constant sampling time $\mathrm{d}\,t$ – Equation (5.5.8). The controller outputs are generated using the Receding Horizon Principle [Mayne and Michalska, 1990], adopting a fixed prediction window with a length equal to N samples – Section 5.6.

The MPC formulation is finally obtained by solving the following optimization problem:

$$\underset{\substack{\mathcal{X}_k,\mathcal{U}_k,\\ k=[0,N]}}{\text{minimize}} \quad \sum_{k=0}^{N} \left(\sum_{i,j} \Psi_{f_{i,j}} + \sum_{i,j} \Psi_{\dot{f}_{i,j}} + \Psi_h + \sum_{i} \Psi_{p_i} \right) \tag{11.1.15a}$$

$$\text{subject to } {}_{\bar{G}}h[k+1] = \mathcal{F}\left(p_i, x_{\mathrm{CoM}}, f_{i,j}\right) \mathrm{d}\,t + {}_{\bar{G}}h[k] \tag{11.1.15b}$$

$$x_{\mathrm{CoM}}[k+1] = \frac{{}_{\bar{G}}h^{p}[k]}{m}\,\mathrm{d}\,t + x_{\mathrm{CoM}}[k] \tag{11.1.15c}$$

$$p_i[k+1] = p_i[k] + (1 - \Gamma_i[k]) v_i[k]\,\mathrm{d}\,t \tag{11.1.15d}$$

$$A\,{}^{\mathcal{C}_i}R_{\mathcal{C}_i[\mathcal{I}]}\,f_{i,j} \preceq b. \tag{11.1.15e}$$

$$l_b \preceq {}^{\mathcal{C}_i}R_{\mathcal{I}}(p_i^n - p_i) \preceq u_b. \tag{11.1.15f}$$

Where the contact dynamics constraint (11.1.15d), the contact force feasibility (11.1.15e) and the contact position constraint (11.1.15f) are repeated for each admissible contact. $\mathcal{X}_k$ and $\mathcal{U}_k$ contain, respectively, the controller state and output at a time instant k:

$$\mathcal{X}_k^{\top} = \begin{bmatrix} x_{\mathrm{CoM}}[k]^{\top} & {}_{\bar{G}}h[k]^{\top} & p_i[k]^{\top} \end{bmatrix}, \tag{11.1.16a}$$

$$\mathcal{U}_k^{\top} = \begin{bmatrix} f_{i,j}[k]^{\top} & v_i[k]^{\top} \end{bmatrix}. \tag{11.1.16b}$$

At every time step $\mathrm{d}\,t$, we solve the optimization problem (11.1.15), then we apply the control output $\mathcal{U}_0$ only, discarding all the other control inputs. The time horizon is shifted to an amount equal to $\mathrm{d}\,t$ and the optimal control problem (11.1.15) is solved again with different initial conditions.

Since the centroidal dynamics (11.1.2) is a nonlinear non-convex function, the optimizer may find a local minimum. This may result in a suboptimal solution for the

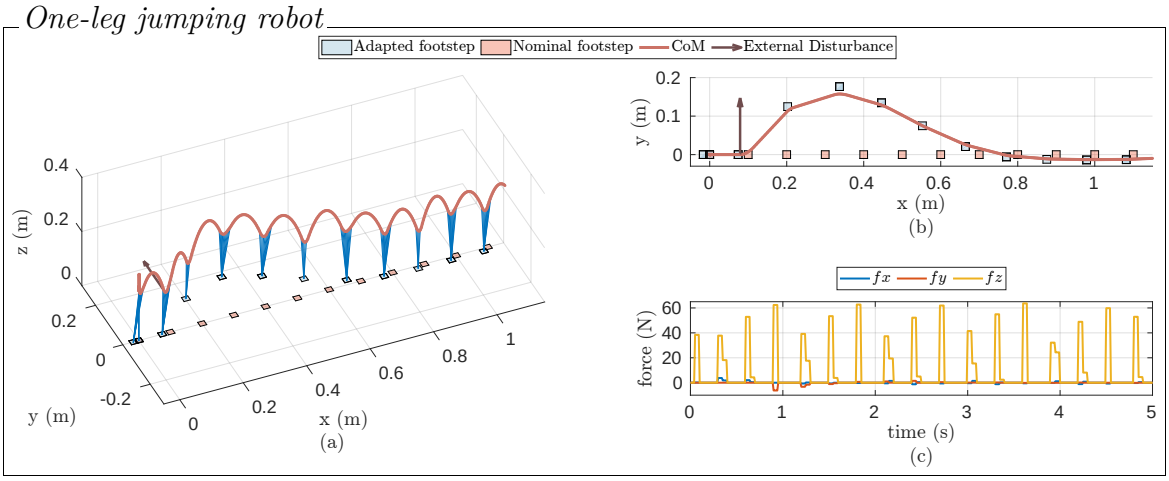


Figure 11.3 (a)-(b) Trajectories generated by the MPC on a one-leg robot performing a jumping task. (c) Desired contact forces.

given task. As a consequence, the initialization of the solver may play a crucial role to drive the optimizer to the desired solution. In our case, the CoM is initialized with the nominal CoM trajectory x^n_{CoM}, while all other variables are set to zero.

11.2 Results

In this section, we present the validation results of the control strategy presented in Section 11.1. CasADi [Andersson et al., 2018] and IPOPT 3.13.4 [Wächter and Biegler, 2006] with HSL_MA97 [Hogg and Scott, 2011] libraries are used to solve the non-linear optimization problem. The code is written in pure C++ and is available at `https://github.com/ami-iit/paper_romualdi_2022_icra_centroidal-mpc-walking`.

To validate the performance of the proposed control, we present two main experiments. First, we test the centroidal MPC considering only the centroidal Dynamics (11.1.2) for a one-leg and two-legs floating base systems. Secondly, we present the results obtained with the implementation of the control architecture shown in Figure 11.1 on the iCub Humanoid Robot V3 – Section 1.1.2. In both scenarios, we analyze the performance of the controller while running on a 10^{th} generation Intel® Core i7-10750H laptop equipped with Ubuntu Linux 20.04.

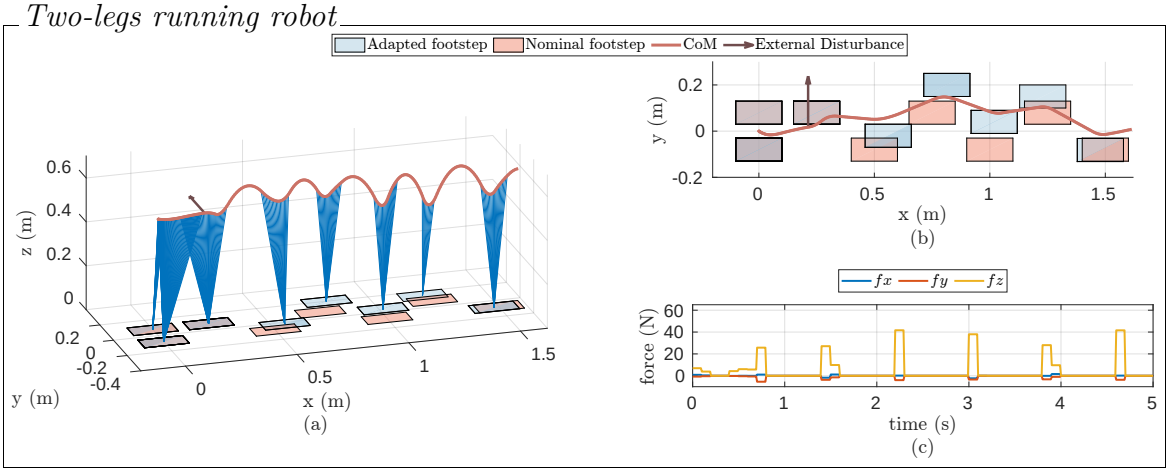


Figure 11.4 (a)-(b) Trajectories generated by the MPC on a two-legs robot performing a running task. (c) Desired contact forces.

11.2.1 Reduced models simulation

Figure 11.3 shows the trajectory generated by the centroidal MPC in the case of a floating base system equipped with one leg. The system has a mass of 1 kg and the foot is approximated by a point, i.e., $n_c = 1$ and $n_v = 1$ in Equation (11.1.2). The MPC takes (on average) less than 20 ms for evaluating its output. At $t \approx 1$ s an external force of magnitude 5 N acts for 0.5 s at the system CoM. The MPC automatically compensates the disturbance effect by adapting the location of the footstep with an average of 10 cm - Figures 11.3a and 11.3b. Figure 11.3c shows the contact force computed by the controller.

Figure 11.4 presents the trajectory generated by the centroidal MPC in the case of a floating base system equipped with two legs. The system weighs 1 kg and has a foot length and width of 20 cm and 10 cm, respectively. In this case, $n_c = 2$ and $n_v = 4$ in Equation (11.1.2). The MPC takes (on average) less than 80 ms for evaluating its output. In this experiment, we analyze the capabilities of the MPC in the transition from locomotion to running. At $t \approx 1$ s the planned contact sequence switches from a bipedal locomotion pattern, where a single support phase is always preceded by a double support phase, to a running pattern, where the single support phases are followed by aerial phases. Furthermore, at $t \approx 1.5$ s an external force of magnitude 5 N acts for 0.5 s at the robot CoM. The MPC handles the transition from locomotion to running, while dealing with the external disturbance effect. Figs. 11.4a and 11.4b show, in blue, the optimal contact location computed by the controller. The distance between

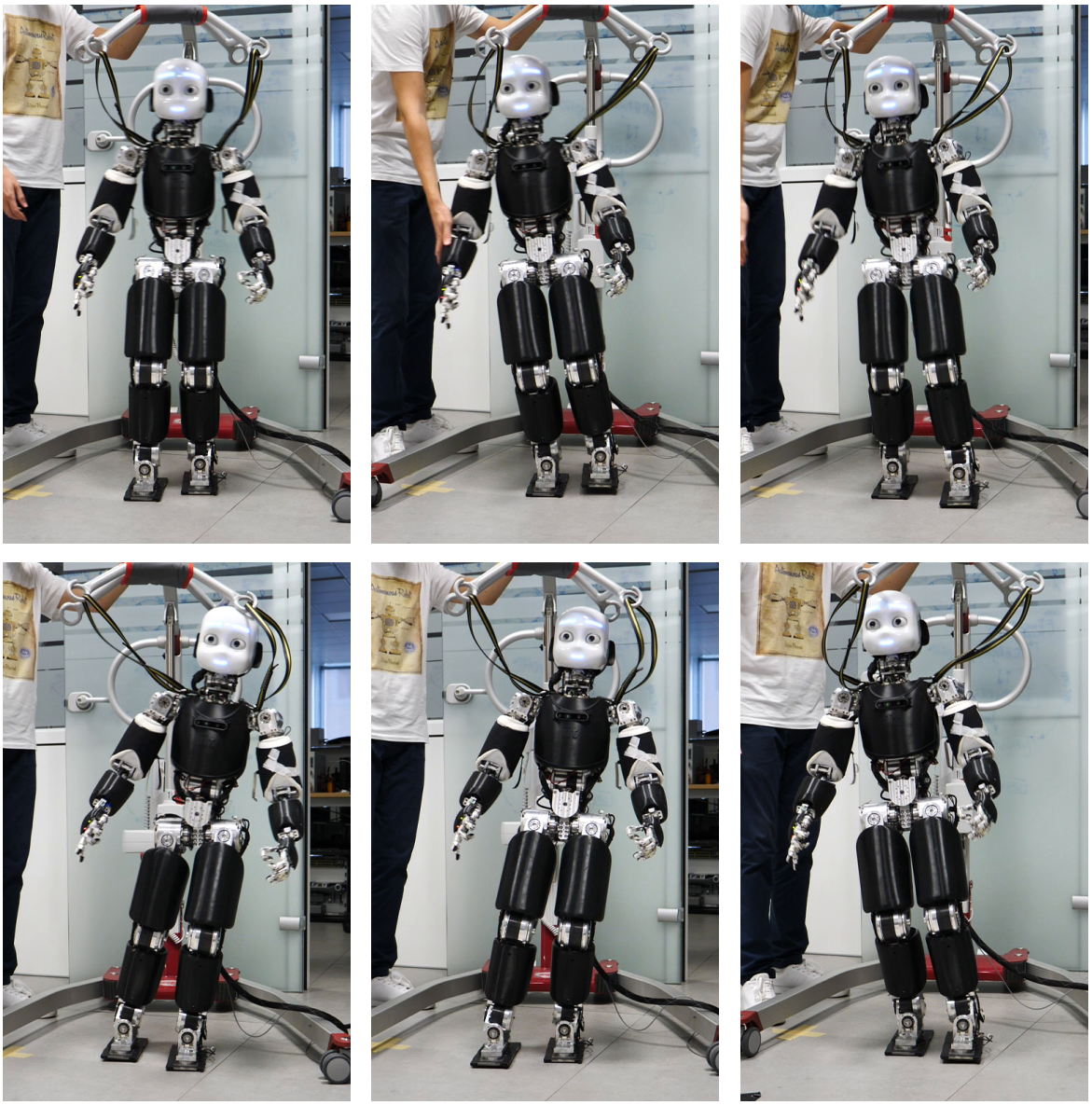


Figure 11.5 The iCub humanoids robot react to an external disturbance.

the nominal contact and the computed one is (on average) 6 cm. Finally, Figure 11.4c shows the contact force computed by the controller.

11.2.2 Test on the iCub Humanoid Robot

To validate the performance of the centroidal MPC on humanoid robots, we attached the controller to the three-layer architecture presented in Chapter 8 and 7 as show in Figure 11.1. In this scenario, *trajectory optimization layer* is responsible for generating

the nominal contact locations and timings. The nominal contact pose is considered as a regularization term for the contact position (11.1.12) and to compute the regularized CoM trajectory (11.1.11). The *centroidal MPC* generates the feasible contact wrenches for the current active contacts and the new locations of the future active contacts. The future contact location is then set in the swing foot trajectory planner to generate a smooth trajectory for the foot. Finally, the inner *whole-body control* loop evaluates the generalized velocity of the robot, ν, by implementing the kinematics-based control law presented in Section 7.1. Here, the stack of tasks considers the references computed by the centroidal MPC. As in Section 7.1, the tracking of the feet and the CoM trajectories are considered as high priority tasks, while the torso orientation is treated as a low priority task. Furthermore, to attempt the stabilization of the zero-dynamics of the system, a postural term is added as a low-priority task. The joint velocities $\dot{s}$ included in the solution of the above problem are then integrated to obtain the joint position references for the low-level position controller. In our implementation, the whole-body control layer takes (on average) less than 1 ms to evaluate its outputs.

To analyze the step recovery capabilities of the whole architecture, the robot is perturbed by an external force acting on the right arm while walking. Since the robot is position controlled, it behaves rigidly when an external force is applied. Consequently, the position of the CoM is not perturbed. To mitigate this effect, we consider the estimated external force as a measured disturbance in the MPC. To do so, we modified the centroidal dynamics considered as the prediction model (11.1.4) as:

$$
\begin{aligned}
{}_{\bar{G}}\dot{h} &= \sum_{i=1}^{n_c}\sum_{j=1}^{n_v}\begin{bmatrix} I_3 \\ (p_i + R_{\mathcal{C}_i}\,{}^{c_i}p_{v_{i,j}} - x_{\mathrm{CoM}})\times \end{bmatrix}\Gamma_i f_{i,j} + m\bar{g} + {}_{\bar{G}}\mathrm{f}_{\mathrm{ext}} && \text{(11.2.1a)}\\
&= \mathcal{F}\left(p_i, x_{\mathrm{CoM}}, f_{i,j}, {}_{\bar{G}}\mathrm{f}_{\mathrm{ext}}\right). && \text{(11.2.1b)}
\end{aligned}
$$

where ${}_{\bar{G}}\mathrm{f}_{\mathrm{ext}}$ is the measured external wrench expressed in the $\bar{G}$ frame. The measured contact wrench is considered different from zero only during the first index in the prediction horizon. in other words, in (11.1.15b), ${}_{\bar{G}}\mathrm{f}_{\mathrm{ext}}[k] \neq 0$ only if $k = 0$.

As shown in Figure 11.6c, the centroidal MPC takes less than 60 ms to evaluate its output. At $t \approx 8$ s and $t \approx 11$ s an external force of magnitude 40 N acts for 1 s on the robot right arm – second picture in Figure 11.5.

The external force is estimated considering the Force Torque sensors mounted on the robot arms and the joint state applying the algorithm discussed in Traversaro [2017]. The MPC considers the external disturbance to propagate the centroidal

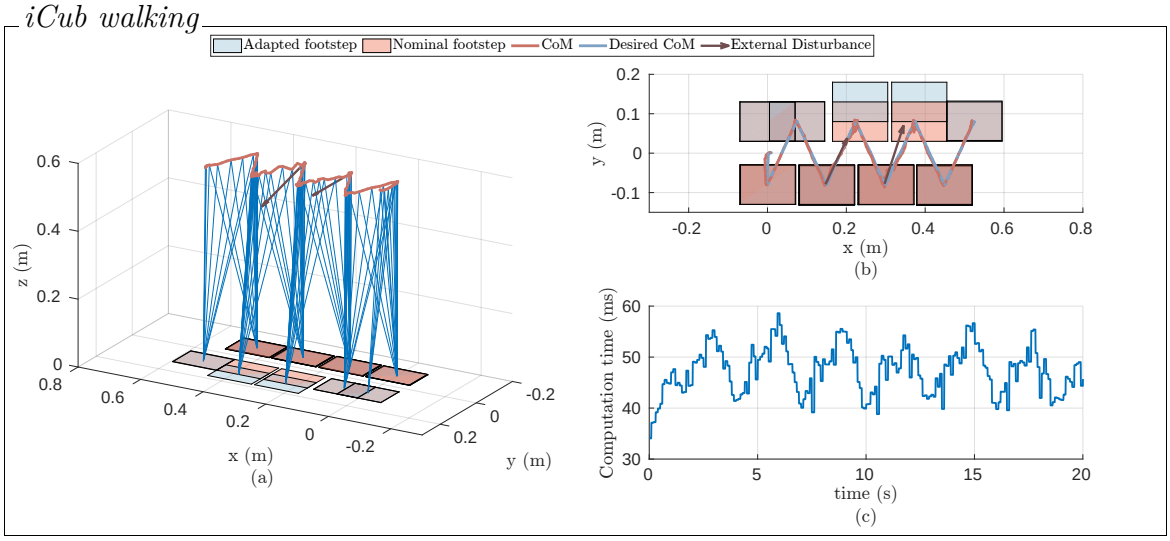


Figure 11.6 (a)-(b) Trajectories generated by the three-layer controller architecture on the iCub robot. (c) Computation time.

dynamics (11.2.1) and automatically compensates for the disturbance effect by adapting the location of the footstep with an average of 5 cm – Figures 11.6a and 11.6b.

11.3 Conclusions

This chapter discusses the development of an online centroidal momentum non-linear MPC for humanoid robots. The controller aims to generate feasible contact locations and wrenches for locomotion. Different from state-of-the-art architectures based on simplified models (e.g. LIPM), the proposed controller can be used to perform highly dynamic movements, such as jumping and running. Furthermore, the contact location adjustment is considered in the centroidal dynamics stabilization problem, so it is not required to design an ad-hoc block for this feature. We validate the controller with a simulation of one-leg and two-leg systems performing jumping and running tasks, respectively. The centroidal MPC is also embedded in the three-layer position-based control architecture and tested on the humanoid robot iCub v3 – see Section 1.1.2. The proposed strategy prevents the robot from falling while walking and pushed with external forces up to 40 N for 1 second applied to the robot arm.

In future work, we may extend the MPC to consider the contact timing adjustment, thus increasing the robustness properties against unpredictable external disturbances. Another interesting research direction is to substitute the Euler integrator required

to transcribe the optimal control problem into an MPC 11.1.4 with a multi-rate sampling technique [Elobaid et al., 2019, 2020b]. Considering a multi-rate sampling technique, it would be possible to reduce the prediction model's discretization error. To improve the time performance, we may consider applying the convex relaxation of the angular momentum dynamics [Ponton et al., 2018, 2016] in the controller prediction model. As a consequence, it would be possible to solve the non-linear optimization problem (11.1.15) by using a convex programming solver and thus increasing the MPC frequency. Finally, to improve the overall time performance, we plan to warm start the non-linear optimization problem with the result of a human-like trajectory planner [Viceconte et al., 2022].

Epilogue

This thesis presented an application of different model-based controllers for time-critical humanoid locomotion. Exploiting a three-layer model-based architecture, we investigated the specific implementation of each block when the considered models change. Part II presents the design of three whole-body controllers for humanoid robot locomotion. The proposed controllers employ similar control algorithms and the main distinguishing factors are the robot and the description of the environment. In Chapter 7 we compared whole-body controllers for locomotion on rigid surfaces. In this chapter, we proposed a kinematics-based and a dynamics-based whole-body controllers. The experiments were carried out on the Humanoid Robot iCub v2.7. We showed that when the robot is position controlled, the architecture was able to achieve the highest walking velocity. On the other hand, we noticed that unstructured uncertainty and the low reliability of the low-level torque-control loop, possibly due to the lack of collocated joint torque sensing, prevented the torque-based architecture from reaching the same performance obtained in simulation.

Motivated by curiosity, we began investigating the consequences of loosening the assumption of the rigid contact model hypothesized in Chapter 7. Chapter 8 proposes a contact model that describes the mechanical characteristics of a visco-elastic carpet. To keep the problem still treatable online, we model the system as a continuum spring-damper system rather than exploiting the finite element method (FEM). In fact, while providing enhanced modeling capabilities, FEM methods demand heavy computational time, which usually prohibits their use in time-critical feedback control applications. The whole-body controller then considers the model to compute viable joint torques, allowing the robot to perform a locomotion task. Results are shown only in simulation. The poor performance of the low-level controller along with too noisy contact force information prevents us from achieving acceptable results on iCub.

Both the architectures presented in Chapters 7 and 8 assume that the robot links do not deform during this locomotion task. In Chapter 9 we attempted to loosen

this assumption by modeling the link flexibility with passive visco-elastic joints. We proposed a whole-body controller that considers the joint elasticity while computing the desired actuated joint torques. Furthermore, since in our case the deflection was not directly measurable, we proposed an observer aiming at estimating the flexible joint state, considering the measured contact force and the actuated joint state. Results are shown only in simulation.

Part III analyzed the outer loops of the three-layer controller architecture. Chapter 10 presented and compared several DCM based kinematic-based architectures. Keeping a fixed trajectory optimization layer, we designed two simplified model controllers: an instantaneous controller and an MPC controller. We benchmarked the two strategies on the Humanoid Robot iCub v2.7. We noticed that the computed ZMP is smoother when the simplified model control uses the predictive law to generate it. However, although this smoother behavior did contribute to fewer robot vibrations, the overall robot performance became less reactive, and as a consequence, the robot may fall. On the other hand, when the robot is position-controlled and the simplified layer implements the proposed instantaneous controller, iCub was able to reach the desired walking speed of $0.3372\,\mathrm{m\,s^{-1}}$. That is, to the best of our knowledge, the highest walking velocity achieved by the iCub robot v2.7.

Motivated by the results obtained from the simplified controller, we decided to investigate whether, by increasing the complexity of the model, we were able to improve the robustness of the control architecture in the case of unexpected interactions with the environment. Chapter 11 attempted to answer this question. We presented a Non-Linear Model Predictive Controller for humanoid robot locomotion with online step adjustment capabilities. Differently from bipedal walking architectures based on simplified models, the presented approach considers the reduced centroidal model, thus allowing us to consider the contact location adjustment directly in the dynamics stabilization problem, while keeping the control problem still treatable online. The approach was validated in the position-controlled Humanoid Robot iCub v3.

In summary, satisfactory experiments on the real robot could be achieved only when the robot was in position control, keeping the torque control walking still an open question in the case of a robot with non-collocated torque sensing. In all the architectures presented in Part II and Chapter 10, the contacts are planned without taking into consideration the state of the robot. Chapter 11, addressed this issue by letting the optimizer determine where to place the contact. This choice had the drawback of complexifying the overall architecture. In the case of flat terrain, simplified

models are still one of the best choices in terms of complexity and robustness capabilities. On the other hand, we believe that reduced model controllers are one of the best trade-offs between an offline predictive planner that considers the complete robot description and a simplified model controller that relays on a tailored model depending on the desired task.

In the Prologue, we mentioned that the future of humanoid robotics is the safe interaction between robots and humans in a human-like environment. The road to achieving this goal is still long and full of pitfalls. We hope that the controllers presented in this thesis may help humankind to take a step forward in this direction.

References

Aceituno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., Cappelletto, J., Grieco, J. C., Fernandez-Lopez, G., and Semini, C. (2017). Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization. *IEEE Robotics and Automation Letters*, 3(3):1–1.

Allgöwer, F., Badgwell, T. A., Qin, J. S., Rawlings, J. B., and Wright, S. J. (1999). Nonlinear Predictive Control and Moving Horizon Estimation — An Introductory Overview. *Advances in Control*, pages 391–449.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2018). CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation 2018 11:1*, 11(1):1–36.

Arakawa, T. and Fukuda, T. (1997). Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers. In *Proceedings of International Conference on Robotics and Automation*, pages 211–216. IEEE.

Ascher, U. M., Ruuth, S. J., and Spiteri, R. J. (1997). Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2-3):151–167.

Axehill, D. and Hansson, A. (2006). A Mixed Integer Dual Quadratic Programming Algorithm Tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698. IEEE.

Azad, M. and Featherstone, R. (2014). A new nonlinear model of contact normal force. *IEEE Transactions on Robotics*.

Azad, M., Ortenzi, V., Lin, H. C., Rueckert, E., and Mistry, M. (2016). Model estimation and control of compliant contact normal force. In *IEEE-RAS International Conference on Humanoid Robots*.

Bellman, R. (1952). On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719.

Bemporad, A., Fukuda, K., and Torrisi, F. D. (2001). Convexity recognition of the union of polyhedra. *Computational Geometry: Theory and Applications*, 18(3):141–154.

Bemporad, A., Heemels, W., and De Schutter, B. (2002). On hybrid systems and closed-loop MPC systems. *IEEE Transactions on Automatic Control*, 47(5):863–869.

Bertolazzi, E., Biral, F., and Da Lio, M. (2005). Symbolic–Numeric Indirect Method for Solving Optimal Control Problems for Large Multibody Systems. *Multibody System Dynamics*, 13(2):233–252.

Betts, J. T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics.

Biral, F., Bertolazzi, E., and Bosetti, P. (2016). Notes on Numerical Methods for Solving Optimal Control Problems. *IEEJ Journal of Industry Applications*, 5(2):154–166.

Bock, H. and Plitt, K. (1984). A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems *. *IFAC Proceedings Volumes*, 17(2):1603–1608.

Bombile, M. and Billard, A. (2017). Capture-Point Based Balance and Reactive Omnidirectional Walking Controller. In *IEEE RAS International Conference on Humanoid Robots*, number EPFL-CONF-231920.

Borrelli, F., Bemporad, A., and Morari, M. (2017). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press.

Borst, C., Wimböck, T., Schmidt, F., Fuchs, M., Brunner, B., Zacharias, F., Giordano, P. R., Konietschke, R., Sepp, W., Fuchs, S., Rink, C., Albu-Schäffer, A., and Hirzinger, G. (2009). Rollin' Justin - Mobile platform with variable base. *Proceedings - IEEE International Conference on Robotics and Automation*.

Bouyarmane, K. and Kheddar, A. (2011). FEM-based static posture planning for a humanoid robot on deformable contact support. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 487–492. IEEE.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Budhiraja, R., Carpentier, J., Mastalli, C., and Mansard, N. (2018). Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE.

Buss, S. R. and Kim, J.-S. (2005). Selectively Damped Least Squares for Inverse Kinematics. *Journal of Graphics Tools*, 10(3):37–49.

Caron, S. (2020). Biped Stabilization by Linear Feedback of the Variable-Height Inverted Pendulum Model. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9782–9788. IEEE.

Caron, S., Kheddar, A., and Kheddar Multi, A. (2016). Multi-contact Walking Pattern Generation based on Model Preview Control of 3D COM Accelerations. pages 550–557.

Caron, S. and Pham, Q. C. (2017). When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC. *IEEE-RAS International Conference on Humanoid Robots*, pages 522–528.

Caron, S., Pham, Q. C., and Nakamura, Y. (2015). Stability of surface contacts for humanoid robots: Closed-form formulae of the Contact Wrench Cone for rectangular support areas. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015-June(June):5107–5112.

Caron, S., Pham, Q.-C., and Nakamura, Y. (2017). ZMP Support Areas for Multicontact Mobility Under Frictional Constraints. *IEEE Transactions on Robotics*, 33(1):67–80.

Carpentier, J., Tonneau, S., Naveau, M., Stasse, O., and Mansard, N. (2016). A versatile and efficient pattern generator for generalized legged locomotion. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3555–3561. IEEE.

Catalano, M. G., Frizza, I., Morandi, C., Grioli, G., Ayusawa, K., Ito, T., and Venture, G. (2020). HRP-4 walks on Soft Feet. *IEEE Robotics and Automation Letters*, pages 1–1.

Chachuat, B. (2007). *Nonlinear and Dynamic Optimization: From Theory to Practice*.

Chiaverini, S., Siciliano, B., and Villani, L. (1994). Force/position regulation of compliant robot manipulators. *IEEE Transactions on Automatic Control*, 39(3):647–652.

Choi, Y., Kim, D., Oh, Y., and You, B.-j. J. (2007). On the Walking Control for Humanoid Robot Based on Kinematic Resolution of CoM Jacobian With Embedd ed Motion. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 23(6):1285–1293.

Cognetti, M., De Simone, D., Lanari, L., and Oriolo, G. (2016). Real-time planning and execution of evasive motions for a humanoid robot. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4200–4206. IEEE.

Colman, G. and Wells, J. (2006). On the Use of RLS with Covariance Reset in Tracking Scenarios with Discontinuities. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 693–696. IEEE.

Dafarra, S., Nava, G., Charbonneau, M., Guedelha, N., Andradel, F., Traversaro, S., Fiorio, L., Romano, F., Nori, F., Metta, G., and Pucci, D. (2018). A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE.

Dafarra, S., Romano, F., and Nori, F. (2016). Torque-controlled stepping-strategy push recovery: Design and implementation on the iCub humanoid robot. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 152–157. IEEE.

Dafarra, S., Romualdi, G., Metta, G., and Pucci, D. (2020). Whole-Body Walking Generation using Contact Parametrization: A Non-Linear Trajectory Optimization Approach. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1511–1517.

Dafarra, S., Romualdi, G., and Pucci, D. (2022). Dynamic Complementary Conditions and Whole-Body Trajectory Optimization for Humanoid Robot Locomotion. *IEEE Transactions on Robotics*.

Dai, H., Valenzuela, A., and Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, volume 2015-Febru, pages 295–302. IEEE, IEEE Computer Society.

Dantec, E., Budhiraja, R., Roig, A., Lembono, T., Saurel, G., Stasse, O., Fernbach, P., Tonneau, S., Vijayakumar, S., Calinon, S., Taix, M., and Mansard, N. (2021). Whole Body Model Predictive Control with a Memory of Motion: Experiments on a Torque-Controlled Talos. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8202–8208. IEEE.

Dantec, E., Taix, M., and Mansard, N. (2022). First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback. *IEEE Robotics and Automation Letters*, 7(2):4448–4455.

Darvish, K., Tirupachuri, Y., Romualdi, G., Rapetti, L., Ferigo, D., Chavez, F. J. A., and Pucci, D. (2019). Whole-Body Geometric Retargeting for Humanoid Robots. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 679–686. IEEE.

Dean-Leon, E., Guadarrama-Olvera, J. R., Bergner, F., and Cheng, G. (2019). Whole-Body Active Compliance Control for Humanoid Robots with Robot Skin. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5404–5410. IEEE.

Deits, R. and Tedrake, R. (2015). Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, volume 2015-Febru, pages 279–286.

Del Prete, A., Mansard, N., Ramos, O. E., Stasse, O., and Nori, F. (2016). Implementing Torque Control with High-Ratio Gear Boxes and Without Joint-Torque Sensors. *International Journal of Humanoid Robotics*, 13(01):1550044.

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., and Kim, S. (2018). Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE.

Diedam, H., Dimitrov, D., Wieber, P.-B., Mombaur, K., and Diehl, M. (2008). Online walking gait generation with adaptive foot positioning through Linear Model Predictive control. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1121–1126. IEEE, IEEE.

Diehl, M., Bock, H. G., Diedam, H., and Wieber, P.-B. (2006). Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer Berlin Heidelberg.

Diehl, M., Ferreau, H. J., and Haverbeke, N. (2009). Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation. In *Nonlinear model predictive control*, pages 391–417. Springer.

Dreyfus, S. (2002). Richard Bellman on the Birth of Dynamic Programming. *Operations Research*, 50(1):48–51.

Dubrovin, B. A., Fomenko, A. T., and Novikov, S. P. (1984). *Modern Geometry — Methods and Applications*, volume 93. Springer New York, New York, NY.

Elobaid, M., Hu, Y., Romualdi, G., Dafarra, S., Babic, J., and Pucci, D. (2020a). Telexistence and Teleoperation for Walking Humanoid Robots. pages 1106–1121.

Elobaid, M., Mattioni, M., Monaco, S., and Normand-Cyrot, D. (2019). On unconstrained MPC through multirate sampling. *IFAC-PapersOnLine*, 52(16):388–393.

Elobaid, M., Mattioni, M., Monaco, S., and Normand-Cyrot, D. (2020b). Sampled-data tracking under model predictive control and multi-rate planning. *IFAC-PapersOnLine*, 53(2):3620–3625.

Englsberger, J., Koolen, T., Bertrand, S., Pratt, J., Ott, C., and Albu-Schaffer, A. (2014). Trajectory generation for continuous leg forces during double support and heel-to-toe shift based on divergent component of motion. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4022–4029. IEEE.

Englsberger, J., Mesesan, G., Ott, C., and Albu-Schaffer, A. (2018a). DCM-Based Gait Generation for Walking on Moving Support Surfaces. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE.

Englsberger, J., Mesesan, G., Werner, A., and Ott, C. (2018b). Torque-based dynamic walking - A long way from simulation to experiment. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 440–447.

Englsberger, J., Ott, C., and Albu-Schaffer, A. (2013). Three-dimensional bipedal walking control using Divergent Component of Motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2600–2607. IEEE.

Englsberger, J., Ott, C., and Albu-Schaffer, A. (2015a). Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion. *IEEE Transactions on Robotics*, 31(2):355–368.

Englsberger, J., Ott, C., Roa, M. A., Albu-Schäffer, A., and Hirzinger, G. (2011). Bipedal walking control based on capture point dynamics. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4420–4427.

Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K., and Albu-Schäffer, A. (2015b). Overview of the torque-controlled humanoid robot TORO. *IEEE-RAS International Conference on Humanoid Robots*, 2015-February:916–923.

Fahmi, S., Focchi, M., Radulescu, A., Fink, G., Barasuol, V., and Semini, C. (2020). STANCE: Locomotion Adaptation Over Soft Terrain. *IEEE Transactions on Robotics*, 36(2):443–457.

Fahmi, S., Mastalli, C., Focchi, M., and Semini, C. (2019). Passive Whole-Body Control for Quadruped Robots: Experimental Validation Over Challenging Terrain. *IEEE Robotics and Automation Letters*, 4(3):2553–2560.

Falcon, E., Laroche, C., Fauve, S., and Coste, C. (1998). Behavior of one inelastic ball bouncing repeatedly off the ground. *The European Physical Journal B*, 3(1):45–57.

Faragasso, A., Oriolo, G., Paolillo, A., and Vendittelli, M. (2013). Vision-based corridor navigation for humanoid robots. In *Proceedings - IEEE International Conference on Robotics and Automation*.

Featherstone, R. (2014). *Rigid Body Dynamics Algorithms*. Springer, Boston, MA.

Feng, S., Whitman, E., Xinjilefu, X., and Atkeson, C. G. (2015a). Optimization-based Full Body Control for the DARPA Robotics Challenge. *Journal of Field Robotics*, 32(2):293–312.

Feng, S., Xinjilefu, X., Atkeson, C. G., and Kim, J. (2015b). Optimization based controller design and implementation for the Atlas robot in the DARPA Robotics Challenge Finals. In *IEEE-RAS International Conference on Humanoid Robots*.

Feng, S., Xinjilefu, X., Atkeson, C. G., and Kim, J. (2016). Robust dynamic walking using online foot step optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2016-November, pages 5373–5378. IEEE.

Fernbach, P., Tonneau, S., and Taix, M. (2018). CROC: Convex Resolution of Centroidal Dynamics Trajectories to Provide a Feasibility Criterion for the Multi Contact Planning Problem. *IEEE International Conference on Intelligent Robots and Systems*, pages 8367–8373.

Flavigne, D., Pettrée, J., Mombaur, K., Laumond, J.-P. P., others, Truong, T. V. A., Flavigne, D., Pettré, J., Mombaur, K., and Laumond, J.-P. P. (2010). Reactive synthesizing of human locomotion combining nonholonomic and holonomic behaviors. In *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pages 632–637. IEEE.

Flayols, T., Prete, A., Khadiv, M., Mansard, N., Flayols, T., Prete, A., Khadiv, M., Mansard, N., Balancing, L. R., Flayols, T., Prete, A. D., Khadiv, M., Mansard, N., and Righetti, L. (2020). Balancing Legged Robots on Visco-Elastic Contacts. Technical report.

Fukuda, K. and Prodon, A. (1995). Double description method revisited. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1120:91–111.

Fumagalli, M., Ivaldi, S., Randazzo, M., Natale, L., Metta, G., Sandini, G., and Nori, F. (2012). Force feedback exploiting tactile and proximal force/torque sensing. *Autonomous Robots*, 33(4):381–398.

García, C. E., Prett, D. M., and Morari, M. (1989). Model predictive control: Theory and practice—A survey. *Automatica*, 25(3):335–348.

Gilardi, G. and Sharf, I. (2002). Literature survey of contact dynamics modelling. *Mechanism and Machine Theory*.

Goldenberg, A., Benhabib, B., and Fenton, R. (1985). A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation*, 1(1):14–20.

Griffin, R. J. and Leonessa, A. (2016). Model predictive control for dynamic footstep adjustment using the divergent component of motion. In *Proceedings - IEEE International Conference on Robotics and Automation*.

Griffin, R. J., Leonessa, A., and Asbeck, A. (2016). Disturbance compensation and step optimization for push recovery. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5385–5390. IEEE.

Griffin, R. J., Wiedebach, G., Bertrand, S., Leonessa, A., and Pratt, J. (2017). Walking stabilization using step timing and location adjustment on the humanoid robot, Atlas. *IEEE International Conference on Intelligent Robots and Systems*, 2017-September:667–673.

Gross, E. (2016). On the Bellman's principle of optimality. *Physica A: Statistical Mechanics and its Applications*, 462:217–221.

Guedelha, N., Kuppuswamy, N., Traversaro, S., and Nori, F. (2016). Self-calibration of joint offsets for humanoid robots using accelerometer measurements. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1233–1238. IEEE.

Guo, Y., Zhang, M., Dong, H., and Zhao, M. (2021). Fast Online Planning for Bipedal Locomotion via Centroidal Model Predictive Gait Synthesis. *IEEE Robotics and Automation Letters*, 6(4):6450–6457.

Gurobi Optimization, L. (2022). Gurobi Optimizer Reference Manual.

Hajikarimi, P. and Moghadas Nejad, F. (2021). Mechanical models of viscoelasticity. In *Applications of Viscoelasticity*, pages 27–61. Elsevier.

Hall, B. C. (2015). *Lie Groups, Lie Algebras, and Representations*.

Handford, M. L. and Srinivasan, M. (2014). Sideways walking: preferred is slow, slow is optimal, and optimal is expensive. *Biology letters*, 10(1):20131006.

Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc, 605 Third Ave. New York, NYUnited States, 1st edition.

Henze, B., Roa, M. A., and Ott, C. (2016). Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios. *International Journal of Robotics Research*.

Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S., and Righetti, L. (2016). Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*.

Herzog, A., Rotella, N., Schaal, S., and Righetti, L. (2015). Trajectory generation for multi-contact momentum control. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, volume 2015-Decem, pages 874–880. IEEE.

Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. (1998). The development of Honda humanoid robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, pages 1321–1326. IEEE.

Hof, A. L. (2008). The 'extrapolated center of mass' concept suggests a simple control of balance in walking. *Human Movement Science*.

Hogg, J. and Scott, J. A. (2011). HSL_MA97 : a bit-compatible multifrontal code for sparse symmetric systems. *Rutherford Appleton Laboratory Technical Reports*.

Holm, D. D. (2008). *Geometric Mechanics Part II: Rotating, Translating and Rolling*. IMPERIAL COLLEGE PRESS.

Hopkins, M. A., Hong, D. W., and Leonessa, A. (2014). Humanoid locomotion on uneven terrain using the time-varying divergent component of motion. In *2014 IEEE-RAS International Conference on Humanoid Robots*, volume 2015-Febru, pages 266–272. IEEE.

Hopkins, M. A., Hong, D. W., and Leonessa, A. (2015). Compliant locomotion using whole-body control and Divergent Component of Motion tracking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5726–5733. IEEE.

Howe, R. (1983). Very Basic Lie Theory. *The American Mathematical Monthly*, 90(9):600–623.

Hunt, K. H. and Crossley, F. R. E. (1975). Coefficient of Restitution Interpreted as Damping in Vibroimpact. *Journal of Applied Mechanics*, 42(2):440–445.

Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*.

Ibanez, A., Bidaud, P., and Padois, V. (2014). Emergence of humanoid walking behaviors from mixed-integer model predictive control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4014–4021. IEEE.

IEC 61158-1 (2019). Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series.

Jeong, H., Lee, I., Oh, J., Lee, K. K., and Oh, J. H. (2019). A Robust Walking Controller Based on Online Optimization of Ankle, Hip, and Stepping Strategies. *IEEE Transactions on Robotics*, 35(6):1367–1386.

Joe, H.-M. and Oh, J.-H. (2018). Balance recovery through model predictive control based on capture point dynamics for biped walking robot. *Robotics and Autonomous Systems*, 105:1–10.

Johnson, K. L. (1985). *Contact Mechanics*. Cambridge University Press.

Kajita, S., Hirukawa, H., Harada, K., and Yokoi, K. (2014). *Introduction to Humanoid Robotics*, volume 101. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 1620–1626. IEEE.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, pages 239–246. IEEE.

Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., and Yokoi, K. (2010). Biped walking stabilization based on linear inverted pendulum tracking. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4489–4496. IEEE.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.

Kamioka, T., Kaneko, H., Takenaka, T., and Yoshiike, T. (2018). Simultaneous Optimization of ZMP and Footsteps Based on the Analytical Solution of Divergent Component of Motion. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1763–1770. IEEE.

Kaneko, K., Kaminaga, H., Sakaguchi, T., Kajita, S., Morisawa, M., Kumagai, I., and Kanehiro, F. (2019). Humanoid Robot HRP-5P: An Electrically Actuated Humanoid Robot With High-Power and Wide-Range Joints. *IEEE Robotics and Automation Letters*, 4(2):1431–1438.

Kaneko, K., Kanehiro, F., Morisawa, M., Akachi, K., Miyamori, G., Hayashi, A., and Kanehira, N. (2011). Humanoid robot HRP-4 - Humanoid robotics platform with lightweight and slim body. In *IEEE International Conference on Intelligent Robots and Systems*.

Kanoun, O., Lamiraux, F., and Wieber, P.-B. (2011). Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, 27(4):785–792.

Khadiv, M., Herzog, A., Moosavian, S. A. A., and Righetti, L. (2016). Step timing adjustment: A step toward generating robust gaits. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 35–42. IEEE.

Khudher, D. and Powell, R. (2016). Quadratic programming for inverse kinematics control of a hexapod robot with inequality constraints. In *2016 International Conference on Robotics: Current Trends and Future Challenges (RCTFC)*, pages 1–5. IEEE.

Kirillov, A. (2008). *An Introduction to Lie Groups and Lie Algebras*. Cambridge University Press, Cambridge.

Kirk, D. E. (1970). *Optimal control theory: an introduction*. Courier Corporation.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154. IEEE.

Koolen, T., de Boer, T., Rebula, J., Goswami, A., and Pratt, J. (2012). Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113.

Koolen, T., Posa, M., and Tedrake, R. (2016). Balance control using center of mass height variation: Limitations imposed by unilateral contact. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 8–15. IEEE.

Krause, M., Englsberger, J., Wieber, P. B., and Ott, C. (2012). Stabilization of the Capture Point dynamics for bipedal walking based on model predictive control. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, pages 165–171.

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

Lankarani, H. M. and Nikravesh, P. E. (1990). A contact force model with hysteresis damping for impact analysis of multibody systems. *Journal of Mechanical Design, Transactions of the ASME*, 112(3):369–376.

Lee, T., Leok, M., and McClamroch, N. H. (2018). *Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*. Springer International Publishing, Cham.

Lee, Y., Hwang, S., and Park, J. (2016). Balancing of humanoid robot using contact force/moment control by task-oriented whole body control framework. *Autonomous Robots*, 40(3):457–472.

Leng, X., Piao, S., Chang, L., He, Z., and Zhu, Z. (2020). Universal Walking Control Framework of Biped Robot Based on Dynamic Model and Quadratic Programming. *Complexity*, 2020:1–13.

Li, C., Ding, Y., and Park, H. W. (2020). Centroidal-momentum-based trajectory generation for legged locomotion. *Mechatronics*, 68:102364.

Li, Q., Takanishi, A., and Kato, I. (1998). Learning control for a biped walking robot with a trunk. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, pages 1771–1777. IEEE.

Li, Q., Yu, Z., Chen, X., Zhou, Q., Zhang, W., Meng, L., and Huang, Q. (2019). Contact Force/Torque Control Based on Viscoelastic Model for Stable Bipedal Walking on Indefinite Uneven Terrain. *IEEE Transactions on Automation Science and Engineering*, 16(4):1627–1639.

Liberzon, D. (2012). *Calculus of Variations and Optimal Control Theory*. Princeton University Press.

Liu, J., Cai, S., Chen, W., and Chen, I.-M. (2017). Minimum-jerk trajectory generation and global optimal control for permanent magnet spherical actuator. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2249–2254. IEEE.

Ljung, L. (1999). *System Identification: Theory for the User*.

Lynch, K. M. and Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 40 W. 20 St. New York, NYUnited States, 1st edition.

Marhefka, D. and Orin, D. (1999). A compliant contact model with nonlinear damping for simulation of robotic systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(6):566–572.

Marsden, J. E. and Ratiu, T. S. (2010). *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer Publishing Company, Incorporated.

Mason, M. T. and Wang, Y. (1988). On the inconsistency of rigid-body frictional planar mechanics. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 524–528. IEEE Comput. Soc. Press.

Mason, S., Rotella, N., Schaal, S., and Righetti, L. (2018). An MPC Walking Framework with External Contact Forces. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1785–1790. IEEE.

Mayne, D. Q. and Michalska, H. (1990). Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824.

Mesesan, G., Englsberger, J., Garofalo, G., Ott, C., and Albu-Schaffer, A. (2019). Dynamic Walking on Compliant and Uneven Terrain using DCM and Passivity-based Whole-body Control. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 25–32. IEEE.

Mesesan, G., Englsberger, J., and Ott, C. (2021). Online DCM Trajectory Adaptation for Push and Stumble Recovery during Humanoid Locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12780–12786. IEEE.

Metta, G., Fitzpatrick, P., and Natale, L. (2006). YARP: Yet Another Robot Platform. *International Journal of Advanced Robotic Systems*, 3(1):8.

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., and Montesano, L. (2010). The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*.

Michaelis, W. (1980). Lie coalgebras. *Advances in Mathematics*, 38(1):1–54.

Mingo Hoffman, E., Traversaro, S., Rocchi, A., Ferrati, M., Settimi, A., Romano, F., Natale, L., Bicchi, A., Nori, F., and Tsagarakis, N. G. (2014). Yarp Based Plugins for Gazebo Simulator. pages 333–346. Springer, Cham.

Mombaur, K., Truong, A., and Laumond, J. P. (2010). From human to humanoid locomotion-an inverse optimal control approach. *Autonomous Robots*.

Morin, P. and Samson, C. (2008). Handbook of Robotics. chapter Motion con, pages 799–826. Springer.

Motzkin, T. S., Raiffa, H., Thompson, G. L., and Thrall, R. M. (1953). The double description method. *Contributions to the Theory of Games*, 2(28):51–73.

Murray, R. M., Sastry, S. S., and Zexiang, L. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

Nakaoka, S., Hattori, S., Kanehiro, F., Kajita, S., and Hirukawa, H. (2007). Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3641–3647. IEEE.

Natale, L., Bartolozzi, C., Pucci, D., Wykowska, A., and Metta, G. (2017). iCub: The not-yet-finished story of building a robot child. *Science Robotics*, 2(13):eaaq1026.

Nava, G., Romano, F., Nori, F., and Pucci, D. (2016). Stability Analysis and Design of Momentum-based Controllers for Humanoid Robots. *Intelligent Robots and Systems (IROS) 2016. IEEE International Conference on*.

Naveau, M., Kudruss, M., Stasse, O., Kirches, C., Mombaur, K., and Soueres, P. (2017). A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control. *IEEE Robotics and Automation Letters*, 2(1):10–17.

Needham, T. (2021). *Visual Differential Geometry and Forms*. Princeton University Press.

Nguyen, Q., Da, X., Grizzle, J. W., and Sreenath, K. (2020). Dynamic Walking on Stepping Stones with Gait Library and Control Barrier Functions. pages 384–399.

Nori, F., Traversaro, S., Eljaik, J., Romano, F., Del Prete, A., and Pucci, D. (2015). iCub Whole-Body Control through Force Regulation on Rigid Non-Coplanar Contacts. *Frontiers in Robotics and AI*, 2.

Olfati-Saber, R. (2001). *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. PhD thesis, Cambridge, MA, USA.

Orin, D. and Goswami, A. (2008). Centroidal Momentum Matrix of a humanoid robot: Structure and properties. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 653–659. IEEE.

Orin, D., Goswami, A., and Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*.

Park, F. C. and Ravani, B. (1997). Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, 16(3):277–295.

Parmiggiani, A., Maggiali, M., Natale, L., Nori, F., Schmitz, A., Tsagarakis, N., Santos-Victor, J., Becchi, F., Sandini, G., and Metta, G. (2012). The Design of the iCub Humanoid Robot. *International Journal of Humanoid Robotics*, 9.

Pattacini, U., Nori, F., Natale, L., Metta, G., and Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*.

Ponton, B., Herzog, A., Del Prete, A., Schaal, S., and Righetti, L. (2018). On Time Optimization of Centroidal Momentum Dynamics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5776–5782. IEEE.

Ponton, B., Herzog, A., Schaal, S., and Righetti, L. (2016). A convex model of humanoid momentum dynamics for multi-contact motion generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 842–849. IEEE.

Pontriagin, L. S. L. S. (1962). The mathematical theory of optimal processes. page 360.

Popovic, M. B., Goswami, A., and Herr, H. (2005). Ground reference points in legged locomotion: Definitions, biological trajectories and control implications. *International Journal of Robotics Research*, 24(12):1013–1032.

Poulakakis, I. and Grizzle, J. (2009). The Spring Loaded Inverted Pendulum as the Hybrid Zero Dynamics of an Asymmetric Hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793.

Poulakakis, I., Smith, J. A., and Buehler, M. (2005). Modeling and experiments of untethered quadrupedal running with a bounding gait: The scout II robot. *International Journal of Robotics Research*.

Pratt, J., Carff, J., Drakunov, S., and Goswami, A. (2006). Capture Point: A Step toward Humanoid Push Recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 200–207. IEEE.

Pratt, J., Koolen, T., de Boer, T., Rebula, J., Cotton, S., Carff, J., Johnson, M., and Neuhaus, P. (2012). Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133.

Pressley, A. (2010). *Elementary Differential Geometry*. Springer London, London.

Pucci, D., Marchetti, L., and Morin, P. (2013). Nonlinear control of unicycle-like robots for person following. In *IEEE International Conference on Intelligent Robots and Systems*.

Qin, S. J. and Badgwell, T. A. (2000). An Overview of Nonlinear Model Predictive Control Applications. *Nonlinear Model Predictive Control*, pages 369–392.

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

Ramadoss, P., Romualdi, G., Dafarra, S., Andrade Chavez, F. J., Traversaro, S., and Pucci, D. (2021). DILIGENT-KIO: A Proprioceptive Base Estimator for Humanoid Robots using Extended Kalman Filtering on Matrix Lie Groups. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2904–2910. IEEE.

Ramuzat, N., Boria, S., and Stasse, O. (2022). Passive Inverse Dynamics Control Using a Global Energy Tank for Torque-Controlled Humanoid Robots in Multi-Contact. *IEEE Robotics and Automation Letters*, 7(2):2787–2794.

Ramuzat, N., Buondonno, G., Boria, S., and Stasse, O. (2021). Comparison of Position and Torque Whole-Body Control Schemes on the Humanoid Robot TALOS. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 785–792. IEEE.

Rapetti, L., Tirupachuri, Y., Darvish, K., Dafarra, S., Nava, G., Latella, C., and Pucci, D. (2020). Model-Based Real-Time Motion Tracking Using Dynamical Inverse Kinematics. *Algorithms*, 13(10):266.

Romano, F., Nava, G., Azad, M., Camernik, J., Dafarra, S., Dermy, O., Latella, C., Lazzaroni, M., Lober, R., Lorenzini, M., Pucci, D., Sigaud, O., Traversaro, S., Babic, J., Ivaldi, S., Mistry, M., Padois, V., and Nori, F. (2018). The CoDyCo Project Achievements and Beyond: Toward Human Aware Whole-Body Controllers for Physical Human Robot Interaction. *IEEE Robotics and Automation Letters*, 3(1):516–523.

Romualdi, G., Dafarra, S., Hu, Y., and Pucci, D. (2018). A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE.

Romualdi, G., Dafarra, S., Hu, Y., Ramadoss, P., Chavez, F. J. A., Traversaro, S., and Pucci, D. (2020). A Benchmarking of DCM-Based Architectures for Position, Velocity and Torque-Controlled Humanoid Robots. *International Journal of Humanoid Robotics*, 17(01):1950034.

Romualdi, G., Dafarra, S., L'Erario, G., Sorrentino, I., Traversaro, S., and Pucci, D. (2022a). Online Non-linear Centroidal MPC for Humanoid Robot Locomotion with Step Adjustment. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10412–10419. IEEE.

Romualdi, G., Dafarra, S., and Pucci, D. (2021). Modeling of Visco-Elastic Environments for Humanoid Robot Motion Control. *IEEE Robotics and Automation Letters*, 6(3):4289–4296.

Romualdi, G., Villa, N., Dafarra, S., Pucci, D., and Stasse, O. (2022b). Control and Estimation of Link Flexibility for Humanoid Robot Motion Control. *IEEE-RAS International Conference on Humanoid Robots (Humanoids) (Submitted)*.

Scianca, N., Cognetti, M., De Simone, D., Lanari, L., and Oriolo, G. (2016). Intrinsically stable MPC for humanoid gait generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 601–606. IEEE.

Scianca, N., De Simone, D., Lanari, L., and Oriolo, G. (2020). MPC for Humanoid Gait Generation: Stability and Feasibility. *IEEE Transactions on Robotics*, 36(4):1171–1188.

Sciavicco, L. and Siciliano, B. (1988). A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators. *IEEE Journal on Robotics and Automation*.

Selig, J. M. (2007). Curves of stationary acceleration in SE(3). *IMA Journal of Mathematical Control and Information*, 24(1):95–113.

Seyde, T., Shrivastava, A., Englsberger, J., Bertrand, S., Pratt, J., and Griffin, R. J. (2018). Inclusion of angular momentum during planning for capture point based walking. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1791–1798.

Shafiee, M., Romualdi, G., Dafarra, S., Chavez, F. J. A., and Pucci, D. (2019). Online dcm trajectory generation for push recovery of torque-controlled humanoid robots. *IEEE-RAS International Conference on Humanoid Robots*, 2019-October:671–678.

Shafiee-Ashtiani, M., Yousefi-Koma, A., Mirjalili, R., Maleki, H., and Karimi, M. (2017a). Push Recovery of a Position-Controlled Humanoid Robot Based on Capture Point Feedback Control. In *2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 126–131. IEEE.

Shafiee-Ashtiani, M., Yousefi-Koma, A., and Shariat-Panahi, M. (2017b). Robust bipedal locomotion control based on model predictive control and divergent component of motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3505–3510. IEEE.

Shahriar, M. S., Ahmed, M. A., Rahman, M. I., and Khan, A. I. (2013). Comparison of MPC and conventional control methods for the stability enhancement of UPFC connected SMIB system. In *2013 2nd International Conference on Advances in Electrical Engineering (ICAEE)*, pages 223–228. IEEE.

Sheridan, T. B. (2016). Human–Robot Interaction. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 58(4):525–532.

Shih, C.-L. (1996). The Dynamics and Control of a Biped Walking Robot With Seven Degrees of Freedom. *Journal of Dynamic Systems, Measurement, and Control*, 118(4):683–690.

Solà, J., Deray, J., and Atchuthan, D. (2018). A micro Lie theory for state estimation in robotics.

Spenko, M., Buerger, S., and Iagnemma, K. (2018). *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, volume 121 of *Springer Tracts in Advanced Robotics*. Springer International Publishing, Cham.

Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., Del Prete, A., Soueres, P., Mansard, N., Lamiraux, F., Laumond, J. P., Marchionni, L., Tome, H., and Ferro, F. (2017). TALOS: A new humanoid research platform targeted for industrial applications. *IEEE-RAS International Conference on Humanoid Robots*, pages 689–695.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2018a). OSQP: An Operator Splitting Solver for Quadratic Programs. *2018 UKACC 12th International Conference on Control, CONTROL 2018*, page 339.

Stellato, B., Naik, V. V., Bemporad, A., Goulart, P., and Boyd, S. (2018b). Embedded Mixed-Integer Quadratic optimization Using the OSQP Solver. In *2018 European Control Conference (ECC)*, pages 1536–1541. IEEE.

Stephens, B. J. and Atkeson, C. G. (2010a). Dynamic Balance Force Control for compliant humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1248–1255.

Stephens, B. J. and Atkeson, C. G. (2010b). Push Recovery by stepping for humanoid robots with force controlled joints. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 52–59. IEEE.

Stronge, W. J. (1991). Unraveling paradoxical theories for rigid body collisions. *Journal of Applied Mechanics, Transactions ASME*.

Sygulla, F. and Rixen, D. (2020). A force-control scheme for biped robots to walk over uneven terrain including partial footholds:. *https://doi.org/10.1177/1729881419897472*, 17(1).

Takanishi, A. (2019). Historical Perspective of Humanoid Robot Research in Asia. In *Humanoid Robotics: A Reference*, pages 35–52. Springer Netherlands, Dordrecht.

Takenaka, T., Matsumoto, T., and Yoshiike, T. (2009). Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091. IEEE.

Tawiah, I. and Song, Y. (2021). Optimal control of nonlinear systems with dynamic programming. *International Journal of Nonlinear Sciences and Numerical Simulation*, 22(2):145–168.

Tirupachuri, Y., Nava, G., Latella, C., Ferigo, D., Rapetti, L., Tagliapietra, L., Nori, F., and Pucci, D. (2020). Towards Partner-Aware Humanoid Robot Control Under Physical Interactions. pages 1073–1092.

Torricelli, D., Gonzalez-Vargas, J., Veneman, J. F., Mombaur, K., Tsagarakis, N., del Ama, A. J., Gil-Agudo, A., Moreno, J. C., and Pons, J. L. (2015). Benchmarking Bipedal Locomotion: A Unified Scheme for Humanoids, Wearable Robots, and Humans. *IEEE Robotics & Automation Magazine*, 22(3):103–115.

Traversaro, S. (2017). *Modelling, Estimation and Identification of Humanoid Robots Dynamics*. PhD thesis.

Traversaro, S., Pucci, D., and Nori, F. (2017). A Unified View of the Equations of Motion used for Control Design of Humanoid Robots. *On line*.

Truong, T. V. A., Flavigne, D., Pettré, J., Mombaur, K., and Laumond, J. P. (2010). Reactive synthesizing of human locomotion combining nonholonomic and holonomic behaviors. In *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2010*.

Tsagarakis, N. G., Caldwell, D. G., Negrello, F., Choi, W., Baccelliere, L., Loc, V. G., Noorden, J., Muratore, L., Margan, A., Cardellino, A., Natale, L., Mingo Hoffman, E., Dallali, H., Kashiri, N., Malzahn, J., Lee, J., Kryczka, P., Kanoulas, D., Garabini, M., Catalano, M., Ferrati, M., Varricchio, V., Pallottino, L., Pavan, C., Bicchi, A., Settimi, A., Rocchi, A., and Ajoudani, A. (2017). WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments. *Journal of Field Robotics*.

Tsagarakis, N. G., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., Righetti, L., Santos-Victor, J., Ijspeert, A. J., Carrozza, M. C., and Caldwell, D. G. (2007). ICub: The design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*.

Tu, L. W. (2011). *An Introduction to Manifolds*. Springer New York, New York, NY.

Viceconte, P. M., Camoriano, R., Romualdi, G., Ferigo, D., Dafarra, S., Traversaro, S., Oriolo, G., Rosasco, L., and Pucci, D. (2022). ADHERENT: Learning Human-like Trajectory Generators for Whole-body Control of Humanoid Robots. *IEEE Robotics and Automation Letters*, 7(2):2779–2786.

Villa, N. A., Fernbach, P., Naveau, M., Saurel, G., Dantec, E., Mansard, N., and Stasse, O. (2022). Torque Controlled Locomotion of a Biped Robot with Link Flexibility. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids) [Submitted]*. IEEE.

Vukobratović, M., Borovac, B., Vukobratov, M., and Borovac, B. (2004). Zero - Moment Point — Thirty Five Years of Its Life. *International Journal of Humanoid Robotics*, 1(1):157–173.

Vukobratovic, M. and Juricic, D. (1969). Contribution to the Synthesis of Biped Gait. *IEEE Transactions on Biomedical Engineering*, BME-16(1):1–6.

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

Warner, F. W. (1971). *Foundations of Differentiable Manifolds and Lie Groups*. Graduate Texts in Mathematics. Springer.

Whittaker, E. T. and McCrae, S. W. (1988). *A Treatise on the Analytical Dynamics of Particles and Rigid Bodies*. Cambridge University Press.

Wieber, P.-b. (2006). Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 137–142. IEEE, IEEE.

Winkler, A. W., Bellicoso, C. D., Hutter, M., and Buchli, J. (2018). Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567.

Žefran, M., Kumar, V., and Croke, C. (1998). On the generation of smooth three-dimensional rigid body motions. *IEEE Transactions on Robotics and Automation*, 14(4):576–589.

Žefran, M., Kumar, V., and Croke, C. (1999). Metrics and Connections for Rigid-Body Kinematics. *The International Journal of Robotics Research*, 18(2):242–1.

Appendix A

Lie Group: a Survival Kit

The appendix aims to present some basic concepts of the *Lie groups* theory. The Lie group is a mathematical abstract entity that dates back to the nineteenth century when mathematician Sophus Lie established the idea of continuous transformation groups. Many years later, its effect has expanded over a wide range of scientific and technological fields. Lie groups, on the other hand, are very abstract creations for the great majority of roboticists, making them difficult to grasp and employ. This chapter will take an *utilitaristic* approach to describe such frameworks, emphasizing intuitive comprehension of their meaning and functions above the breadth and depth of arguments and proofs. Once the Lie groups' formalism has been introduced, we present two examples of Lie groups often exploited in robotics, namely, the group of the rotation matrices and the group of the roto-translation matrices.

The reader who wants a more thorough and rigorous understanding of the Lie groups theory should consult the extensive literature, where it is worth mentioning [Solà et al., 2018] from which part of this appendix took inspiration. Other complete and more rigorous dissertations are [Hall, 2015; Howe, 1983; Kirillov, 2008; Warner, 1971].

A.1 Matrix Lie Group

A *group* $(\mathcal{G}, \circ)$ is a set $\mathcal{G}$, with a composition operation $\circ$, of which the following axioms are satisfied:

- **Closure under** $\circ$: For every elements X, Y belonging to $\mathcal{G}$, the composition of X and Y belongs to the group, i.e., $\forall X, Y \in \mathcal{G}$, $X \circ Y \in \mathcal{G}$;

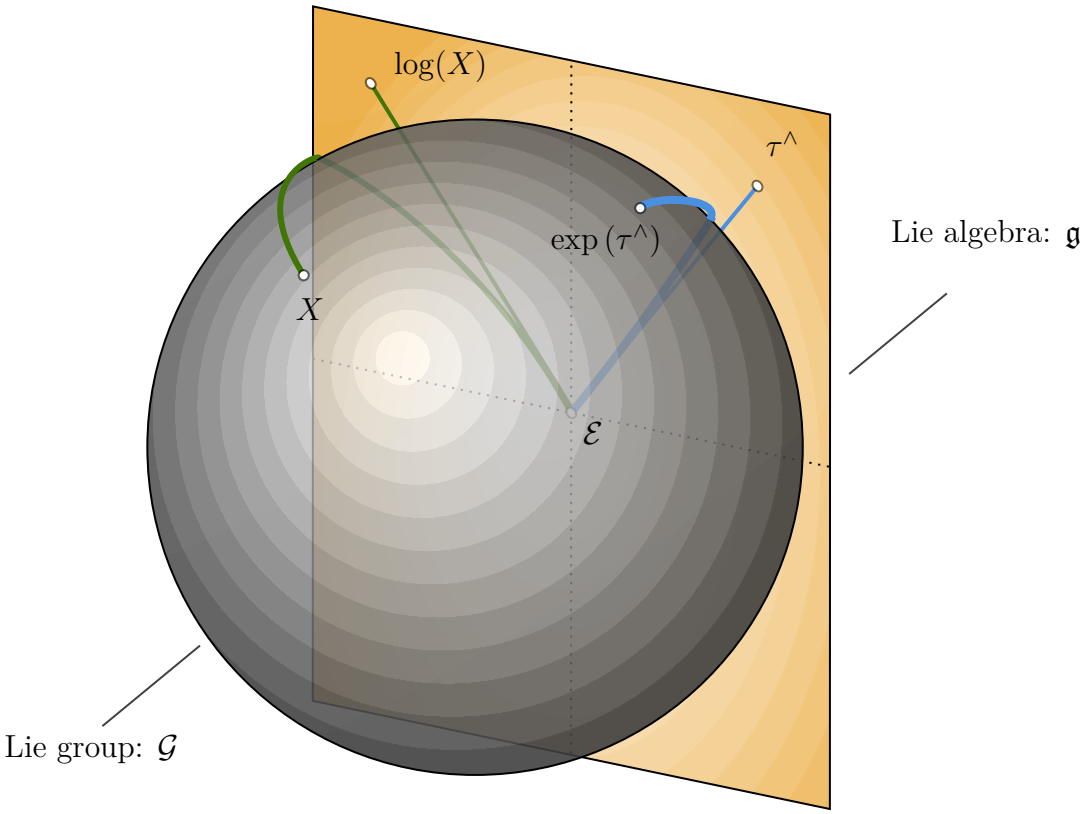


Figure A.1 Lie group and the corresponding Lie algebra as tangent space at identity.

- **Existence of the Identity element**: There exists an element $\mathcal{E} \in \mathcal{G}$ such that, for every $X \in \mathcal{G}$, $\mathcal{E} \circ X = X \circ \mathcal{E} = X$. This element is unique and is called the *identity element* of the group;
- **Existence of the inverse element**: For each $X \in \mathcal{G}$, there exists an element $X^{-1} \in \mathcal{G}$, such that $X^{-1} \circ X = X \circ X^{-1} = \mathcal{E}$, where e is the identity element. For each X, the element X^{-1} is unique and is called the *inverse of* X;
- **Associativity**: For $X, Y, Z \in \mathcal{G}$, $X \circ (Y \circ Z) = (X \circ Y) \circ Z$.

We define a *Smooth Manifold* as a topological space that locally resembles a linear space. We now define a *Lie group* as a group that is also a finite-dimensional real smooth manifold, in which the group operations of composition $\circ$ and inversion $\cdot^{-1}$ are smooth maps. Let $\mathrm{GL}(n, \mathbb{R})$ denote the group of $n \times n$ invertible matrices with entries in $\mathbb{R}$. Any *topologically closed subgroup* [1] of $\mathrm{GL}(n, \mathbb{R})$ is a Lie group. Lie groups of this sort are called *matrix Lie group*.

[1] A subgroup $\mathcal{G} \subset \mathrm{GL}(n, \mathbb{R})$ is said topologically closed if given a sequence of $X_1, X_2, \cdots \in \mathcal{G}$ such that X_k converges in $\mathrm{GL}(n, \mathbb{R})$, then $\lim_{k\to\infty} X_k \in \mathcal{G}$.

In this manuscript we consider only Lie groups that are also matrix Lie groups, hereafter, to simplify the notation, we will remove the *matrix* prefix, and we indicate a matrix Lie group as a Lie group.

A.2 Action of a Lie Group

Given a Lie group $\mathcal{G}$ and a set V, we introduce the *action of* $X \in \mathcal{G}$ *on* $v \in V$ as

$$\cdot : \mathcal{G} \times V \longrightarrow V, \tag{A.2.1}$$

The action $\cdot$ must satisfy the following properties:

- **Compatibility**: For $X, Y \in \mathcal{G}$ and $v \in V$, $(X \circ Y) \cdot v = X \cdot (Y \cdot v)$;
- **Identity**: For each $v \in V$, $\mathcal{E} \cdot v = v$, where $\mathcal{E}$ is the identity element of the Lie group $\mathcal{G}$.

To provide the reader with a better understanding, *group action* is the ability of a Lie group to transform an element of other sets. For example, in the case of SO(3), the action of a rotation matrix on a 3D vector results in a coordinate transformation – see Section 2.1. On the other hand, in SE(3), the group action converts a vector expressed in a frame into another frame, where the two frames have a different origin and orientation – see Equation (2.2.3).

A.3 Tangent space and Lie algebra

The *tangent space at* $X \in \mathcal{G}$, is the space $T_X\mathcal{G}$ of all the tangent vectors of all the curves passing through X. Since the Lie group is a smooth manifold, the tangent space is defined for every element of $\mathcal{G}$ and its structure is an invariant of the group. As a consequence, it is possible to associate with each Lie group a particular tangent space called tangent space at the identity $T_\mathcal{E}\mathcal{G}$, or simply *Lie algebra of* $\mathcal{G}$ and usually denoted with $\mathfrak{g}$. The Lie algebra $\mathfrak{g}$ is a vector space together with a bilinear operation called *Lie bracket*:

$$[.,.] : \mathfrak{g} \times \mathfrak{g} \longrightarrow \mathfrak{g}, \tag{A.3.1}$$

obeying the following axioms:

- **Bilinearity** For all scalars $a, b \in \mathbb{R}$ and for all elements $x, y, z \in \mathfrak{g}$, the following identity is verified $[ax + by, z] = a\,[x, z] + b\,[y, z]$
- **Alternativity** For each $x \in \mathfrak{g}$, $[x, x] = 0$
- **Jacobi identity** Given $x, y, z \in \mathfrak{g}$ the following identity is verified $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$.

It is worth recalling that, using the bilinearity and the alternativity axioms, it is possible to prove that the Lie bracket is an *anticommutative* operator, i.e., $[x, y] = -\,[y, x]$. For a matrix Lie group, the elements in $\mathfrak{g}$ are matrices, consequently given $x, y \in \mathfrak{g}$, it is possible to prove that the Lie bracket is always the *commutator* of matrices [Hall, 2015, Definition 2.19], i.e.,

$$[x, y] = xy - yx. \tag{A.3.2}$$

The elements of a Lie algebra $\mathfrak{g}$ can be uniquely *identified* with vectors in $\mathbb{R}^m$ where m is the number of degrees of freedom of $\mathcal{G}$. In fact, since $\mathfrak{g}$ is a vector space, every element $\tau^\wedge \in \mathfrak{g}$ can be expressed as a linear combination of some base elements E_i, where E_i are called the *generators* of $\mathfrak{g}$. It is now possible to define two *isomorphisms*, commonly denoted *hat* and *vee* such that:

$$\text{hat} : \mathbb{R}^m \longrightarrow \mathfrak{g} \quad \tau^\wedge = \sum_{i=1}^{m} \alpha_i E_i \tag{A.3.3a}$$

$$\text{vee} : \mathfrak{g} \longrightarrow \mathbb{R}^m \quad (\tau^\wedge)^\vee = \sum_{i=1}^{m} \alpha_i e_i \tag{A.3.3b}$$

where e_i represents a vector of the canonical base $\mathbb{R}^m$, that is, $e_i = E_i^\vee$ and $E_i = e_i^\wedge$.

As already mentioned in Section 2.4, the angular velocity belongs to the Lie algebra of SO(3), denoted with $\mathfrak{so}(3)$. While the spatial velocity is an element of $\mathfrak{se}(3)$, that is, the Lie algebra of SE(3).

A.4 Co-tangent space and Lie co-algebra

Given a Lie group $\mathcal{G}$, let $X \in \mathcal{G}$ and $T_X\mathcal{G}$ the tangent space at X. Then we define the *co-tangent space* at X as the dual space[2] of $T_X\mathcal{G}$, and denoted with $T_X^*\mathcal{G}$. Given a Lie group $\mathcal{G}$ and its associated Lie algebra $\mathfrak{g}$, we define the *Lie co-algebra* $\mathfrak{g}^*$ as the dual

[2]Given a vector space V having a basis $e_1, \ldots, e_n$. The dual space of V, denoted V^*, has the same dimension of V and is a vector space. The basis of V^* is the set of linear functions $\mu_1, \ldots, \mu_n$ that satisfy $\mu_i(e_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta.

space of the tangent space at the identity, i.e., $\mathfrak{g}^* = T^*_{\mathcal{E}}X$ [Michaelis, 1980]. The Lie co-algebra $\mathfrak{g}^*$ is a vector space and its element can be identified with the vectors in $\mathbb{R}^m$, with m equal to the number of degrees of freedom of $\mathcal{G}$. Indeed, given a set of generators of $\mathfrak{g}^*$, denoted as E^*_i, it is possible to define two *isomorphisms*, called *hat* and *vee* such that:

$$\text{hat} : \mathbb{R}^m \longrightarrow \mathfrak{g}^* \quad w^\wedge = \sum_{i=1}^{m} \alpha_i E^*_i \tag{A.4.1a}$$

$$\text{vee} : \mathfrak{g}^* \longrightarrow \mathbb{R}^m \quad (w^\wedge)^\vee = \sum_{i=1}^{m} \alpha_i e_i \tag{A.4.1b}$$

where $w^\wedge \in \mathfrak{g}^*$ and e_i represent a vector of the canonical base $\mathbb{R}^m$, i.e $e_i = E_i^{*^\vee}$ and $E^*_i = e_i^\wedge$.

Given $\tau^\wedge \in \mathfrak{g}$ and $w^\wedge \in \mathfrak{g}^*$, the *dual pairing* is the map

$$\langle ., . \rangle : \mathfrak{g} \times \mathfrak{g}^* \longrightarrow \mathbb{R} \quad \langle \tau^\wedge, w^\wedge \rangle = w^\wedge (\tau^\wedge) . \tag{A.4.2}$$

We usually say that the dual vector $w^\wedge$ acts on the tangent vector $\tau^\wedge$. For matrix Lie groups, both elements in $\mathfrak{g}$ and $\mathfrak{g}^*$ are matrices, and the dual pairing between the Lie algebra and its dual is equivalent to

$$\langle \tau^\wedge, w^\wedge \rangle = \text{tr} (\tau^\wedge w^\wedge) . \tag{A.4.3}$$

Since both $\mathfrak{g}$ and $\mathfrak{g}^*$ are isomorphic to $\mathbb{R}^n$, given $\tau^\wedge \in \mathfrak{g}$ and $w^\wedge \in \mathfrak{g}^*$, the dual pairing map can be applied directly to the elements $w \in \mathbb{R}^n$ and $\tau \in \mathbb{R}^n$. In this context, the dual pairing is equivalent to the scalar product as:

$$\langle \tau, w \rangle = \tau^\top w. \tag{A.4.4}$$

Applying the Lie group formalism, we notice that the 6D spatial force is an element of Lie co-algebra $\mathfrak{se}(3)^*$ [Holm, 2008, Chapter 6] – see Section 2.2.2.

A.5 Left and right trivialization

Given an element of a Lie group $X \in \mathcal{G}$, we denote the *left* and the *right* translation on $\mathcal{G}$ as [Tu, 2011, Chapter 4]

$$\begin{aligned} L_X &: \mathcal{G} \longrightarrow \mathcal{G} \\ R_X &: \mathcal{G} \longrightarrow \mathcal{G} \end{aligned} \tag{A.5.1}$$

If $\mathcal{G}$ is a matrix Lie group, L_X and R_X are just standard multiplication such that, given $Y \in \mathcal{G}$, $L_X Y = XY$ and $R_X Y = YX$. It is worth noting that L_X and R_X are smooth maps whose inverses $L_X^{-1} = L_{X^{-1}}$ and $R_X^{-1} = R_{X^{-1}}$ are also smooth, i.e., L_X and R_X are *diffeomorphisms*.

Given an element of the Lie Group $X \in \mathcal{G}$, the left and right translations (A.5.1) induce an isomorphism of tangent spaces, denoted as $L_{X^*} : \mathfrak{g} \longrightarrow T_X\mathcal{G}$ and $R_{X^*} : \mathfrak{g} \longrightarrow T_X\mathcal{G}$ named *left* and *right trivialization*. Given $X \in \mathcal{G}$ and a curve $Y(t) \in \mathcal{G}$ such that $Y(0) = \mathcal{E}$ and $\frac{\mathrm{d}}{\mathrm{d}t} Y\Big|_0 = \tau^\wedge \in \mathfrak{g}$, L_{X^*} and R_{X^*} are defined as [Tu, 2011, Chapter 4]:

$$L_{X^*}\tau^\wedge = \frac{\mathrm{d}}{\mathrm{d}\,t} L_X Y(t)\bigg|_{t=0} = L_X \tau^\wedge, \tag{A.5.2a}$$

$$R_{X^*}\tau^\wedge = \frac{\mathrm{d}}{\mathrm{d}\,t} R_X Y(t)\bigg|_{t=0} = R_X \tau^\wedge. \tag{A.5.2b}$$

For matrix Lie groups, both elements in $\mathcal{G}$ and in $\mathfrak{g}$ are matrices such that we may multiply them together using matrix multiplication. Consequently, $L_{X^*}\tau^\wedge$ and $R_{X^*}\tau^\wedge$ are simply given by:

$$L_{X^*}\tau^\wedge = X\tau^\wedge, \tag{A.5.3a}$$

$$R_{X^*}\tau^\wedge = \tau^\wedge X. \tag{A.5.3b}$$

In conclusion, given any element $\tau^\wedge \in \mathfrak{g}$, the left and right induced tangent maps L_{X^*} and R_{X^*} describe the tangent space $T_X\mathcal{G}$ at a point $X \in \mathcal{G}$.

In light of the above, it is now clear why we defined the left and right trivialized velocity in Section 2.1.1 and 2.2.1. Indeed, given ${}^A H_B \in \mathrm{SE}(3)$ and left trivialized spatial velocity ${}^B\mathrm{v}^\wedge_{A,B} \in \mathfrak{se}(3)$. The right trivialization map results in an element of the tangent space at ${}^A H_B$, i.e., ${}^A\dot{H}_B = L_{{}^A H_B}\,{}^B\mathrm{v}^\wedge_{A,B} = {}^A H_B\,{}^B\mathrm{v}^\wedge_{A,B} \in T_{{}^A H_B}\mathrm{SE}(3)$. Similar considerations hold also for the right trivialized velocity.

A.6 Exponential and logarithmic map

Given a tangent increment $\tau := vt \in \mathbb{R}^m$ as velocity $v \in \mathbb{R}^m$ per time $t \in \mathbb{R}$, the *exponential map*, is defined as

$$\exp : \mathfrak{g} \longrightarrow \mathcal{G} \quad X = \exp(\tau^\wedge); \tag{A.6.1}$$

where X belongs to $\mathcal{G}$. The inverse of the *exponential map* is the *logarithmic map* and it is defined as:

$$\log : \mathcal{G} \longrightarrow \mathfrak{g} \quad \tau^\wedge = \log(X). \tag{A.6.2}$$

. Given an element on the Lie group $X \in \mathcal{G}$, an element on the Lie algebra $\tau \in \mathfrak{g}$ and two real scalars $s, t \in \mathbb{R}$, the following properties are satisfied:

- $\exp((t+s)\tau^\wedge) = \exp(t\tau^\wedge)\exp(s\tau^\wedge)$;
- $\exp(t\tau^\wedge) = \exp(\tau^\wedge)^t$;
- $\exp(X\tau^\wedge X^{-1}) = X\exp(\tau^\wedge)X^{-1}$.

To further simplify the notation, we introduce the *capitalized Exp and Log* as

$$x = \mathrm{Exp}(\tau) := \exp(\tau^\wedge) \quad \tau = \mathrm{Log}(x) := \log(x)^\vee. \tag{A.6.3}$$

A.7 The adjoint and the co-adjoint representation of a Lie group

We now introduce the *adjoint representation* of a Lie group $\mathcal{G}$ at $X \in \mathcal{G}$ on an element of the Lie algebra $\tau^\wedge \in \mathfrak{g}$ as

$$\mathrm{Ad}_X : \mathcal{G} \times \mathfrak{g} \longrightarrow \mathfrak{g} \quad \mathrm{Ad}_X(\tau^\wedge) := X\tau^\wedge X^{-1}. \tag{A.7.1}$$

In other words, given $X \in \mathcal{G}$ and a vector ${}^X\tau^\wedge \in \mathfrak{g}$ the adjont representation Ad_X applied to ${}^X\tau^\wedge$ returns a vector in the Lie algebra ${}^\mathcal{E}\tau^\wedge \in \mathfrak{g}$ such that the left and right trivialization of ${}^X\tau^\wedge$ and ${}^\mathcal{E}\tau^\wedge$ returns the same element in $T_X\mathcal{G}$, i.e., $X{}^X\tau^\wedge = {}^\mathcal{E}\tau^\wedge X \in T_X\mathcal{G}$. Figure A.2 shows the effect of the adjoint representation on an element of a Lie algebra.

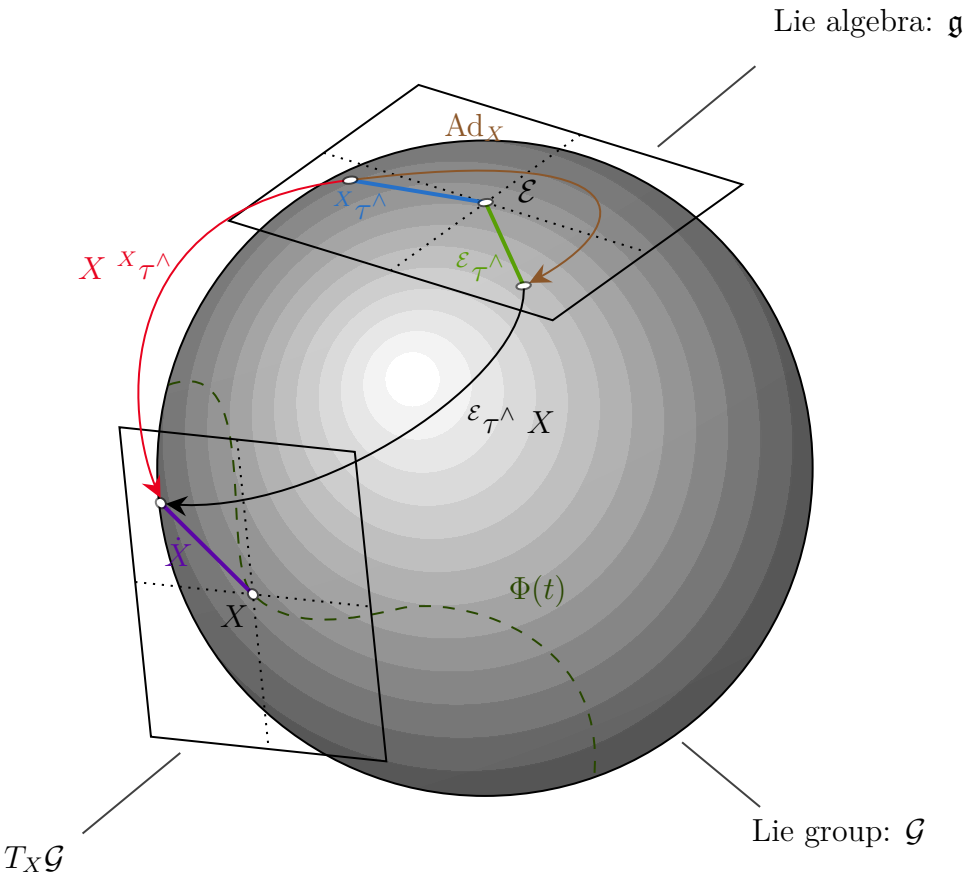


Figure A.2 The Adjoint representation.

The adjoint representation is a linear operator. Indeed, given an element $X \in \mathcal{G}$, two elements of the Lie algebra $\tau^{\wedge}, \gamma^{\wedge} \in \mathfrak{g}$ and two real numbers $\alpha, \theta \in \mathbb{R}$ we have:

$$\begin{aligned} \mathrm{Ad}_X(\alpha\tau^{\wedge} + \theta\gamma^{\wedge}) &= (\alpha\tau^{\wedge} + \theta\gamma^{\wedge})X^{-1} && \text{(A.7.2a)} \\ &= \alpha X\tau^{\wedge}X^{-1} + \theta X\gamma^{\wedge}X^{-1} && \text{(A.7.2b)} \\ &= \alpha\,\mathrm{Ad}_X(\tau^{\wedge}) + \theta\,\mathrm{Ad}_X(\gamma^{\wedge}). && \text{(A.7.2c)} \end{aligned}$$

Since the adjoint representation is a linear transformation and the elements of the tangent space are isomorphic to the vectors in $\mathbb{R}^n$, i.e., $\mathfrak{g} \cong \mathbb{R}^n$, we can define a matrix called *adjoint matrix of a Lie group* $\mathcal{G}$, denoted with $\mathbf{Ad}_X$, that maps the Cartesian tangent vectors ${}^{\mathcal{E}}\tau$ and ${}^{X}\tau$ and writes as

$$\mathbf{Ad}_X : \mathbb{R}^m \longrightarrow \mathbb{R}^m; \quad {}^{\mathcal{E}}\tau = \mathbf{Ad}_X\,{}^{X}\tau. \tag{A.7.3}$$

Considering the SE(3) group, the adjoint representation aims to map a left trivialized spatial velocity into a right trivialized one.

We now introduce the *co-adjoint representation* of a Lie group $\mathcal{G}$ at $X \in \mathcal{G}$ on an element of the Lie co-algebra $w^\wedge \in \mathfrak{g}^*$ by means of the dual pairing map $\langle ., . \rangle$ as [Holm, 2008, Chapter 4]:

$$\mathrm{Ad}^*_X : \mathcal{G} \times \mathfrak{g}^* \longrightarrow \mathfrak{g}^* \quad \langle \mathrm{Ad}^*_X(w^\wedge), \tau^\wedge \rangle := \langle w^\wedge, \mathrm{Ad}_{X^{-1}}(\tau^\wedge) \rangle, \tag{A.7.4}$$

where $\tau^\wedge \in \mathfrak{g}$. It is worth noting that for a matrix Lie group the co-adjoint representation has a closed form and it is equal to

$$\mathrm{Ad}^*_X(w^\wedge) = X^{-1} w^\wedge X. \tag{A.7.5}$$

Indeed using the trace as dual pairing between $\mathfrak{g}$ and $\mathfrak{g}^*$

$$\begin{aligned}
\langle w^\wedge, \mathrm{Ad}_{X^{-1}}(\tau^\wedge) \rangle &= \mathrm{tr}\left(w^\wedge X^{-1} \tau^\wedge X \right) && \text{(A.7.6a)} \\
&= \mathrm{tr}\left(w^\wedge X \tau^\wedge X^{-1} \right) && \text{(A.7.6b)} \\
&= \mathrm{tr}\left(X^{-1} w^\wedge X \tau^\wedge \right) && \text{(A.7.6c)} \\
&= \langle X^{-1} w^\wedge X, \tau^\wedge \rangle && \text{(A.7.6d)} \\
&= \langle \mathrm{Ad}^*_X(w^\wedge), \tau^\wedge \rangle. && \text{(A.7.6e)}
\end{aligned}$$

Similarly to the adjoint representation, the co-adjoint representation is also a linear transformation, so we can introduce a matrix called *co-adjoint matrix of a Lie group* $\mathcal{G}$, denoted with $\mathbf{Ad}^*_X$, as

$$\mathbf{Ad}^*_X : \mathbb{R}^m \longrightarrow \mathbb{R}^m; \quad \langle \mathbf{Ad}^*_X \, w, \tau \rangle := \langle w, \mathbf{Ad}_{X^{-1}} \, \tau \rangle. \tag{A.7.7}$$

There exists a relation between the co-adjoint matrix $\mathbf{Ad}^*_X$ and adjoint matrix $\mathbf{Ad}_X$, indeed, by applying the scalar product as the dual pairing between $\mathbb{R}^n \cong \mathfrak{g}$ and $\mathbb{R}^n \cong \mathfrak{g}^*$ we obtain

$$\begin{aligned}
\langle \mathbf{Ad}^*_X \, w, \tau \rangle &= w^\top \, \mathbf{Ad}^{*\top}_X \, \tau && \text{(A.7.8a)} \\
&= \langle w, \mathbf{Ad}^{*\top}_X \, \tau \rangle && \text{(A.7.8b)} \\
&= \langle w, \mathbf{Ad}_{X^{-1}} \, \tau \rangle, && \text{(A.7.8c)}
\end{aligned}$$

where $\mathbf{Ad}_X^*$ is given by

$$\mathbf{Ad}_X^* = \mathbf{Ad}_{X^{-1}}^\top . \tag{A.7.9}$$

Given an element of the SE(3) group, e.g., ${}^A H_B \in \mathrm{SE}(3)$, the co-adjoint representation expresses a left trivialized spatial force as a function of a right trivialized one, and it writes as:

$${}_A\mathrm{f}^\wedge = \mathrm{Ad}^*_{{}_A H_B}({}_B\mathrm{f}^\wedge) = {}^A H_B^{-1}\, {}_B\mathrm{f}^\wedge\, {}^A H_B. \tag{A.7.10}$$

A.8 The adjoint and the co-adjoint representation of the Lie algebra

We now introduce the *adjoint representation of the Lie algebra* $\mathfrak{g}$ at $x^\wedge \in \mathfrak{g}$ on an element of the Lie algebra $y^\wedge \in \mathfrak{g}$ as

$$\mathrm{ad}_{x^\wedge} : \mathfrak{g} \times \mathfrak{g} \longrightarrow \mathfrak{g} \quad \mathrm{ad}_{x^\wedge}(y^\wedge) := \frac{\mathrm{d}}{\mathrm{d}\, t}\, \mathrm{Ad}_{\exp(t x^\wedge)}\, y^\wedge \Big|_{t=0} . \tag{A.8.1}$$

To give the reader a better understanding we may imagine having an adjoint representation of the Lie group $\mathrm{Ad}_{X(t)}$ such that ${}^{\mathcal{E}}\tau^\wedge(t) = \mathrm{Ad}_{X(t)}\left({}^X\tau^\wedge\right)$ where $X(t)$ is represented by the element of the Lie algebra, i.e., $X(t) = \exp(t x^\wedge)$, and we aim to compute the time derivative of ${}^{\mathcal{E}}\tau^\wedge(t)$ at $t = 0$, i.e., $\frac{\mathrm{d}}{\mathrm{d}\, t}\, {}^{\mathcal{E}}\tau^\wedge(t)\Big|_0 = \mathrm{ad}_{x^\wedge}\left({}^X\tau^\wedge\right)$.

It is worth noting that *adjoint representation of the Lie algebra on itself*, $\mathrm{ad}_{x^\wedge}(y^\wedge)$ is given the Lie bracket – Equation (A.3.2). In fact, expanding (A.8.1) we obtain

$$\begin{aligned}
\mathrm{ad}_{x^\wedge}(y^\wedge) &= \frac{\mathrm{d}}{\mathrm{d}\, t}\, \mathrm{Ad}_{\exp(t x^\wedge)}\, y^\wedge \Big|_{t=0} && \text{(A.8.2a)}\\
&= \frac{\mathrm{d}}{\mathrm{d}\, t} \exp(t x^\wedge) y^\wedge \exp(-t x^\wedge) \Big|_{t=0} && \text{(A.8.2b)}\\
&= \left[x^\wedge \exp(t x^\wedge) y^\wedge \exp(-t x^\wedge) - \exp(t x^\wedge) y^\wedge x^\wedge \exp(-t x^\wedge)\right]\big|_{t=0} && \text{(A.8.2c)}\\
&= x^\wedge y^\wedge - y^\wedge x^\wedge && \text{(A.8.2d)}\\
&= [x^\wedge, y^\wedge] . && \text{(A.8.2e)}
\end{aligned}$$

Since the *adjoint representation of the Lie algebra on itself* is a linear transformation, i.e., the *Lie bracket* is a linear map, we can define a matrix called *adjoint matrix of the Lie algebra*, denoted with $\mathbf{ad}_{x^\wedge}$ that maps the time derivative of the Cartesian tangent

vectors ${}^{\mathcal{E}}\tau^{\wedge}$ and ${}^{X}\tau^{\wedge}$ and writes as:

$$\mathbf{ad}_{x^{\wedge}} : \mathbb{R}^m \longrightarrow \mathbb{R}^m; \quad {}^{\mathcal{E}}\dot{\tau} = \mathbf{ad}_{x^{\wedge}} {}^{X}\tau. \tag{A.8.3}$$

In this context, it is also important to remark on an important result of Lie algebra theory, known as *Adjoint motion equation* [Holm, 2008, Proposition 4.2.2]. Let a smooth path $X(t) \in \mathcal{G}$ and $\tau^{\wedge}(t) \in \mathfrak{g}$ be a path in the Lie algebra, then the following relation holds [Holm, 2008, Proposition 4.2.2]:

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left\{\mathrm{Ad}_{X(t)}\,\tau^{\wedge}(t)\right\} = \mathrm{Ad}_{X(t)}\left[\frac{\mathrm{d}}{\mathrm{d}\,t}\tau^{\wedge}(t) + \mathrm{ad}_{\xi(t)}\,\tau^{\wedge}(t)\right]. \tag{A.8.4}$$

where $\xi(t) = X(t)^{-1}\dot{X}(t) \in \mathfrak{g}$ – i.e., the left trivialization. Equation (A.8.4) can be expressed in matrix form as:

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left\{\mathbf{Ad}_{X(t)}\,\tau(t)\right\} = \mathbf{Ad}_{X(t)}\left[\frac{\mathrm{d}}{\mathrm{d}\,t}\tau(t) + \mathbf{ad}_{\xi(t)}\,\tau(t)\right]. \tag{A.8.5}$$

In the case of the roto-translation group (A.8.5) results in the Equation (2.2.29), where $\mathbf{ad}_{\xi(t)}$ is given by $\xi(t)\times$.

We now introduce the *adjoint representation* of a Lie algebra $\mathfrak{g}$ at $x^{\wedge} \in \mathfrak{g}$ on an element of the Lie co-algebra $w^{\wedge} \in \mathfrak{g}^*$ by means of the dual pairing map $\langle ., . \rangle$:

$$\mathrm{ad}^*_{x^{\wedge}} : \mathfrak{g} \times \mathfrak{g}^* \longrightarrow \mathfrak{g}^* \quad \langle \mathrm{ad}^*_{x^{\wedge}}(w^{\wedge}), \tau^{\wedge} \rangle := \langle w^{\wedge}, -\mathrm{ad}_{x^{\wedge}}(\tau^{\wedge}) \rangle, \tag{A.8.6}$$

where $\tau^{\wedge} \in \mathfrak{g}$.

It is worth noting that for a matrix, the Lie group $\mathrm{ad}^*_{x^{\wedge}}(w^{\wedge})$ has a closed form and it is equal to

$$\mathrm{ad}^*_{x^{\wedge}}(w^{\wedge}) = x^{\wedge}w^{\wedge} - w^{\wedge}x^{\wedge}. \tag{A.8.7}$$

Indeed using the trace as dual pairing between $\mathfrak{g}$ and $\mathfrak{g}^*$

$$\begin{aligned}
\langle w^\wedge, -\operatorname{ad}_{x^\wedge}(\tau^\wedge)\rangle &= \operatorname{tr}\left(w^\wedge\left(\tau^\wedge x^\wedge - x^\wedge\tau^\wedge\right)\right) && \text{(A.8.8a)}\\
&= \operatorname{tr}\left(w^\wedge\tau^\wedge x^\wedge - w^\wedge x^\wedge\tau^\wedge\right) && \text{(A.8.8b)}\\
&= \operatorname{tr}\left(w^\wedge\tau^\wedge x^\wedge\right) - \operatorname{tr}\left(w^\wedge x^\wedge\tau^\wedge\right) && \text{(A.8.8c)}\\
&= \operatorname{tr}\left(x^\wedge w^\wedge\tau^\wedge\right) - \operatorname{tr}\left(w^\wedge x^\wedge\tau^\wedge\right) && \text{(A.8.8d)}\\
&= \operatorname{tr}\left(\left(x^\wedge w^\wedge - w^\wedge x^\wedge\right)\tau^\wedge\right) && \text{(A.8.8e)}\\
&= \langle x^\wedge w^\wedge - w^\wedge x^\wedge, \tau^\wedge\rangle && \text{(A.8.8f)}\\
&= \langle \operatorname{ad}^*_{x^\wedge}(w^\wedge), \tau^\wedge\rangle. && \text{(A.8.8g)}
\end{aligned}$$

Similarly to $\operatorname{ad}_{x\wedge}$, $\operatorname{ad}^*_{x\wedge}$ is a linear transformation, so we can introduce a matrix called *co-adjoint matrix of the lie algebra* $\mathbf{ad}^*_{x^\wedge}$, defined as

$$\mathbf{ad}^*_{x^\wedge} : \mathbb{R}^m \longrightarrow \mathbb{R}^m; \quad \langle \mathbf{ad}^*_{x^\wedge}\, w, \tau\rangle := \langle w, -\,\mathbf{ad}_{x^\wedge}\,\tau\rangle. \tag{A.8.9}$$

In the case of matrix Lie group, there exists a relation between $\mathbf{ad}^*_{x^\wedge}$ and $\mathbf{ad}_{x^\wedge}$. Indeed, applying the scalar product as the dual pairing between $\mathbb{R}^n \cong \mathfrak{g}$ and $\mathbb{R}^n \cong \mathfrak{g}^*$, we obtain the following:

$$\begin{aligned}
\langle \mathbf{ad}^*_{x^{\wedge *}}\, w, \tau\rangle &= w^\top\, \mathbf{ad}^{*\top}_{x^\wedge}\,\tau && \text{(A.8.10a)}\\
&= \langle w, \mathbf{ad}^{*\top}_{x^\wedge}\,\tau\rangle && \text{(A.8.10b)}\\
&= \langle w, -\,\mathbf{ad}_{x^\wedge}\,\tau\rangle, && \text{(A.8.10c)}
\end{aligned}$$

where $\mathbf{ad}^*_{x^\wedge}$ is given by

$$\mathbf{ad}^*_{x^\wedge} = -\,\mathbf{ad}^\top_{x^\wedge}\,. \tag{A.8.11}$$

In this context, it is also important to recall an important result of the Lie algebra theory, known as *co-adjoint motion equation* [Holm, 2008, Proposition 4.2.5]. Let a smooth path $X(t) \in \mathcal{G}$ and $\mu^\wedge(t) \in \mathfrak{g}^*$ be a path in the Lie co-algebra, then the following relation holds [Holm, 2008, Proposition 4.2.5]:

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left\{\operatorname{Ad}^*_{X(t)}\,\mu^\wedge(t)\right\} = \operatorname{Ad}_{X(t)}\left[\frac{\mathrm{d}}{\mathrm{d}\,t}\mu^\wedge(t) + \operatorname{ad}^*_{\xi(t)}\,\mu^\wedge(t)\right]. \tag{A.8.12}$$

where $\xi(t) = X(t)^{-1}\dot{X}(t) \in \mathfrak{g}$ – i.e., the left trivialization. Equation (A.8.12) can be expressed in matrix form as

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left\{\mathbf{Ad}^*_{X(t)}\,\mu(t)\right\} = \mathbf{Ad}^*_{X(t)}\left[\frac{\mathrm{d}}{\mathrm{d}\,t}\mu(t) + \mathbf{ad}^*_{\xi(t)}\,\mu(t)\right]. \tag{A.8.13}$$

In SE(3) Equation (A.8.13) results in Equation (2.2.31), where $\mathbf{ad}^*_{\xi(t)}$ is given by $\xi(t)\times^*$.

A.9 Eurel-Poincaré equations

The Euler-Poincaré Equations are the generalization of the Euler-Lagrange equations to a system whose configuration space is a Lie group.

Given a system Σ whose state belongs to a matrix Lie group $\mathcal{G}$ and let a *left-trivialized Lagrangian function* $\mathcal{L} : \mathcal{G} \times \mathfrak{g} \longrightarrow \mathbb{R}$. Let $\mathcal{P}$ be the set of smooth paths $X : [t_0, t_f] \longrightarrow \mathcal{G}$ such that $X(t_0) = X_0$ and $X(t_f) = X_f$. We aim to compute the trajectory $X(t)$ so that it is a *stationary point* of the *action functional*:

$$\mathfrak{G} = \int_{t_0}^{t_f} \mathcal{L}(X, \xi^\wedge)\,\mathrm{d}\,t. \tag{A.9.1}$$

Applying *Hamilton's Variational Principle* [Lee et al., 2018] on Equation (A.9.1) we can conclude that a path $X \in \mathcal{P}$ is a stationary point of J if and only if

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\frac{\partial \mathcal{L}}{\partial \xi} + \mathbf{ad}^*_{\xi}\frac{\partial \mathcal{L}}{\partial \xi} = X^{-1}\frac{\partial \mathcal{L}}{\partial X} \tag{A.9.2}$$

where $\xi^\wedge = X^{-1}\dot{X}$ is the left trivialization of the Lie algebra.

Equation (A.9.2) plays a crucial role in the definition of the dynamics of a rigid body system. Indeed, considering a SE(3) group and the Langrangian function (2.3.2) we notice that (A.9.2) is equivalent to (2.3.4) with

$$\mathbf{ad}^*_{{}^B\mathrm{v}^\wedge_{\mathcal{I},B}} = {}^B\mathrm{v}_{\mathcal{I},B}\times^*, \quad \frac{\partial \mathcal{L}}{\partial {}^B\mathrm{v}_{\mathcal{I},B}} = {}_B\mathbb{M}_B\,{}^B\mathrm{v}_{\mathcal{I},B}, \quad {}^{\mathcal{I}}H_B^{-1}\frac{\partial \mathcal{L}}{\partial {}^{\mathcal{I}}H_B} = -{}_B\mathbb{M}_B\begin{bmatrix}{}^{\mathcal{I}}R_B^\top g \\ 0_{3\times 1}\end{bmatrix}. \tag{A.9.3}$$

Appendix B

Proof of Lemma 1

Let us now consider a rigid body that makes a contact with a visco-elastic surface, and we assume that:

1. there exists an inertial frame $\mathcal{I}$;
2. there exists a frame B rigidly attached to the body, and we denote o_B the origin of the frame and $[B]$ its orientation;
3. there exists a contact domain $\Omega \in \mathbb{R}^3$, we denote with ${}^{B}x$ a point in the contact surface expressed in the the frame B;
4. the characteristics of the environment are isotropic;
5. while in contact, the rigid body moves with a 6D velocity, denoted as ${}^{B[\mathcal{I}]}\mathrm{v}$ such that
$$ {}^{B[\mathcal{I}]}\mathrm{v} = \begin{bmatrix} {}^{\mathcal{I}}\dot{o}_B \\ {}^{\mathcal{I}}\omega_{\mathcal{I},B} \end{bmatrix}; \tag{B.0.1} $$
6. $\forall x \in \Omega$, there exists a continuous pure force distribution that depends on the point ${}^{\mathcal{I}}x$ and its velocity ${}^{\mathcal{I}}\dot{x}$ expressed in the inertial frame, as in (8.1.6)
$$ \rho\left({}^{\mathcal{I}}x, {}^{\mathcal{I}}\dot{x}\right) = k\left({}^{\mathcal{I}}\bar{x} - {}^{\mathcal{I}}x\right) - b\,{}^{\mathcal{I}}\dot{x}, \tag{B.0.2} $$
where $\bar{x} \in \bar{\mathcal{X}}$.

We now introduce u and v as the coordinates, in the body frame B, of the point belongs to the contact surface ${}^{B}x$, such as
$$ {}^{B}x = \begin{bmatrix} u & v & 0 \end{bmatrix}^{\top}. \tag{B.0.3} $$

The position of the contact point in the inertial frame, denoted as ${}^{\mathcal{I}}x$, is given by applying the homogeneous transformation ${}^{\mathcal{I}}H_B$ to ${}^{B}x$ as

$$ {}^{\mathcal{I}}x = {}^{\mathcal{I}}H_B {}^{B}x = o_B + {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top, \tag{B.0.4} $$

The contact point velocity ${}^{\mathcal{I}}\dot{x}$ is given by time differentiating Equation (B.0.4):

$$ {}^{\mathcal{I}}\dot{x} = \dot{o}_B + \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right) {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top, \tag{B.0.5} $$

Using the hypothesis of rigid-body, $\bar{x}$ can be computed as:

$$ {}^{\mathcal{I}}\bar{x} = \bar{o}_B + {}^{\mathcal{I}}\bar{R}_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top. \tag{B.0.6} $$

Here, $\bar{o}_B$ and ${}^{\mathcal{I}}\bar{R}_B$ are the position and rotation of the body frame associated with a null force in the case of zero velocity.

Combining (B.0.2) with (B.0.4), (B.0.5) and (B.0.6), the force acting on a point lying on the contact surface becomes:

$$ \rho = k \left\{ \bar{o}_B - o_B + \left({}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B\right) \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \right\} \tag{B.0.7a} $$

$$ - b \left\{ \dot{o}_B + \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right) {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \right\}. \tag{B.0.7b} $$

where (B.0.7a) is the force generated by the spring and (B.0.7b) is the one produced by the damper. To facilitate the process of finding the solutions to the integrals (8.1.4), let us recall that given a double integral of a function $g(x, y)$, a variable change of the form (B.0.4), (B.0.5) and (B.0.6) yields

$$ \iint g(x,y)\,\mathrm{d}\,x\,\mathrm{d}\,y = \iint g\left(x(u,v), y(u,v)\right) |\det(J)|\,\mathrm{d}\,u\,\mathrm{d}\,v, \tag{B.0.8} $$

where J is the Jacobian of the variable transformation, i.e.

$$ J = \begin{bmatrix} \dfrac{\partial x_1}{\partial u} & \dfrac{\partial x_1}{\partial v} \\ \dfrac{\partial x_2}{\partial u} & \dfrac{\partial x_2}{\partial v} \end{bmatrix}. \tag{B.0.9} $$

Here, ${}_k$ operator extracts the k element of a vector, that is, $x_k = e_k^\top x$.

Given the variable change defined in (B.0.4) and writing ${}^{\mathcal{I}}R_B$ as the horizontal concatenation of three vectors i, j, n. It is straightforward to verify that $|\det(J)|$ is equal to

$$|\det(J)| = |i_1 j_2 - i_2 j_1| = |n^\top e_3| = |e_3^\top {}^{\mathcal{I}}R_B e_3|. \tag{B.0.10}$$

B.1 Compliant contact force computation

Equations (B.0.8), (B.0.7) and (B.0.10) can be used to evaluate the total force applied from the environment to a generic contact surface as

$$\begin{aligned} {}_{\mathcal{I}}f = k|e_3^\top {}^{\mathcal{I}}R_B e_3| \iint \left\{ \bar{o}_B - o_B + \left({}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B \right) \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \right\} \mathrm{d}\,u\,\mathrm{d}\,v & \quad \text{(B.1.1a)} \\ - b|e_3^\top {}^{\mathcal{I}}R_B e_3| \iint \left\{ \dot{o}_B + \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times \right) {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \right\} \mathrm{d}\,u\,\mathrm{d}\,v. & \quad \text{(B.1.1b)} \end{aligned}$$

If Ω is represented by a rectangle with a length l and a width w, the contact force ${}_{\mathcal{I}}f$ in (B.1.1) writes as

$$\begin{aligned} {}_{\mathcal{I}}f = k|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2} \int_{-w/2}^{w/2} \bar{o}_B - o_B \,\mathrm{d}\,u\,\mathrm{d}\,v & \quad \text{(B.1.2a)} \\ + k|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2} \int_{-w/2}^{w/2} \left({}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B \right) \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \mathrm{d}\,u\,\mathrm{d}\,v & \quad \text{(B.1.2b)} \\ - b|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2} \int_{-w/2}^{w/2} \dot{o}_B \,\mathrm{d}\,u\,\mathrm{d}\,v & \quad \text{(B.1.2c)} \\ - b|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2} \int_{-w/2}^{w/2} \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times \right) {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \mathrm{d}\,u\,\mathrm{d}\,v. & \quad \text{(B.1.2d)} \end{aligned}$$

In Equation (B.1.2) we can recognize two common structures. The integrand functions in (B.1.2) and (B.1.3) are constants. On the other hand, the integrands in (B.1.3) and (B.1.3) are odd functions:

$$\begin{aligned} \Gamma \sim \bar{o}_B - o_B \sim \dot{o}_B, & \quad \text{(B.1.3a)} \\ \Xi(u,v) \sim \left({}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B \right) \begin{bmatrix} u & v & 0 \end{bmatrix}^\top \sim \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times \right) {}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top. & \quad \text{(B.1.3b)} \end{aligned}$$

The integral of the constant term Γ in the rectangle contact domain is given by

$$\int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \Gamma \,\mathrm{d}\,u\,\mathrm{d}\,v = \Gamma \int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \mathrm{d}\,u\,\mathrm{d}\,v = lw\Gamma, \tag{B.1.4}$$

on the other hand, since the integration domain is symmetric, the integral of the odd term $\Xi(u,v)$ is equal to the zero vector. To conclude, the equivalent contact force ${}_{\mathcal{I}}f$ writes as (8.1.7a)

$${}_{\mathcal{I}}f = k(\bar{o}_B - o_B) - b\dot{o}_B. \tag{B.1.5}$$

B.2 Compliant contact torque computation

The torque about the origin of B, o_B produced by the force distribution ρ is:

$$\sigma_{o_B}(u,v) = \left({}^{\mathcal{I}}R_B \begin{bmatrix} u & v & 0 \end{bmatrix}^\top\right) \times \rho(u,v), \tag{B.2.1}$$

Applying (B.0.8), the integral of (B.2.1) on a rectangular contact surface leads to

$$\begin{aligned}
{}_{B[\mathcal{I}]}\mu = {} & k|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right) \times (\bar{o}_B - o_B)\,\mathrm{d}\,u\,\mathrm{d}\,v && \text{(B.2.2a)}\\
& + k|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right) \times \left({}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B\right) \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \mathrm{d}\,u\,\mathrm{d}\,v && \text{(B.2.2b)}\\
& - b|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right) \times \dot{o}_B\,\mathrm{d}\,u\,\mathrm{d}\,v && \text{(B.2.2c)}\\
& - b|e_3^\top {}^{\mathcal{I}}R_B e_3| \int_{-l/2}^{l/2}\int_{-w/2}^{w/2} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right) \times \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right) {}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \mathrm{d}\,u\,\mathrm{d}\,v. && \text{(B.2.2d)}
\end{aligned}$$

In Equation (B.2.2) we can recognize two common structures. The integral terms in (B.2.2a) and (B.2.2c) are linear odd functions on the integral variables u and v. On

the other hand the integrands in (B.2.2b) and (B.2.2d) can be rewritten as

$$\left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \right) \times \mathcal{A} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}, \tag{B.2.3}$$

where $\mathcal{A}$ is equal to $\mathcal{A} = {}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B$ in Equation (B.2.2b), and $\mathcal{A} = \left({}^{\mathcal{I}}\omega_{\mathcal{I},B} \times \right) {}^{\mathcal{I}}R_B$ in Equation (B.2.2d).

We notice that since the integration domain is symmetric, the integral of the odd terms (B.2.2a) and (B.2.2c) is equal to zero.

We now aim to compute the following integral

$$\int_{-l/2}^{l/2} \int_{-w/2}^{w/2} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \right) \times \mathcal{A} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \mathrm{d}\, u \, \mathrm{d}\, v. \tag{B.2.4}$$

Let us rewrite the rotation matrix ${}^{\mathcal{I}}R_B$ and the matrix $\mathcal{A}$ as the column concatenation of the following vectors

$${}^{\mathcal{I}}R_B = \begin{bmatrix} i & j & k \end{bmatrix} \qquad \mathcal{A} = \begin{bmatrix} a & b & c \end{bmatrix}. \tag{B.2.5}$$

Then the integrand function (B.2.3) can be rewritten as

$$\begin{aligned} \left({}^{\mathcal{I}}R_B \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \right) \times \mathcal{A} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} &= (ui + vj) \times (ua + vb) && \text{(B.2.6a)} \\ &= u^2(i \times a) + v^2(j \times b) && \text{(B.2.6b)} \\ &\quad + uv[(i \times b) + (j \times a)]. && \text{(B.2.6c)} \end{aligned}$$

The term (B.2.6c) is an odd function, and as a consequence the integral in the symmetric domain is equal to zero. On the other hand, the integral (B.2.6b) is given by

$$\int_{-l/2}^{l/2}\int_{-w/2}^{w/2}\left({}^{\mathcal{I}}R_B\begin{bmatrix}u\\v\\0\end{bmatrix}\right)\times\mathcal{A}\begin{bmatrix}u\\v\\0\end{bmatrix}\mathrm{d}\,u\,\mathrm{d}\,v \quad = \quad \text{(B.2.7a)}$$

$$\int_{-l/2}^{l/2}\int_{-w/2}^{w/2}u^2(i\times a)+v^2(j\times b)\,\mathrm{d}\,u\,\mathrm{d}\,v \quad = \quad \text{(B.2.7b)}$$

$$\frac{lw}{12}\{l^2(i\times a)+w^2(j\times b)\} \quad = \quad \text{(B.2.7c)}$$

$$\frac{lw}{12}\left\{l^2\left({}^{\mathcal{I}}R_Be_1\right)\times(\mathcal{A}e_1)+w^2\left({}^{\mathcal{I}}R_Be_2\right)\times(\mathcal{A}e_2)\right\} \quad . \quad \text{(B.2.7d)}$$

Recalling that $\mathcal{A}$ is equal to $\mathcal{A} = {}^{\mathcal{I}}\bar{R}_B - {}^{\mathcal{I}}R_B$ in Equation (B.2.2b), and $\mathcal{A} = \left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right){}^{\mathcal{I}}R_B$ in Equation (B.2.2d), the solution of the integral (B.2.2) becomes

$$\begin{aligned}
{}_{B[\mathcal{I}]}\mu = \frac{klw}{12}|e_3^\top{}^{\mathcal{I}}R_Be_3|\Big\{&l^2\left({}^{\mathcal{I}}R_Be_1\right)\times\left[\left({}^{\mathcal{I}}\bar{R}_B-{}^{\mathcal{I}}R_B\right)e_1\right]+ && \text{(B.2.8a)}\\
&w^2\left({}^{\mathcal{I}}R_Be_2\right)\times\left[\left({}^{\mathcal{I}}\bar{R}_B-{}^{\mathcal{I}}R_B\right)e_2\right]\Big\}+ && \text{(B.2.8b)}\\
-\frac{blw}{12}|e_3^\top{}^{\mathcal{I}}R_Be_3|\Big\{&l^2\left({}^{\mathcal{I}}R_Be_1\right)\times\left[\left(\left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right){}^{\mathcal{I}}R_B\right)e_1\right]+ && \text{(B.2.8c)}\\
&w^2\left({}^{\mathcal{I}}R_Be_2\right)\times\left[\left(\left({}^{\mathcal{I}}\omega_{\mathcal{I},B}\times\right){}^{\mathcal{I}}R_B\right)e_2\right]\Big\}. && \text{(B.2.8d)}
\end{aligned}$$

By applying some basic rules of the cross product, Equation (B.2.8) leads to (8.1.7b).

Appendix C

Proof of Corollary 1

Let ${}_{\mathcal{I}}f$ and ${}_{B[\mathcal{I}]}\mu$ the contact force and torque given by (8.1.7a) and (8.1.7b), respectively. Assume that ${}^{\mathcal{I}}\bar{R}_B = I_3$ and ${}^{\mathcal{I}}R_B$ is approximated with its first order of the Taylor expansion, i.e., ${}^{\mathcal{I}}R_B = I_3 + \Theta\times$, with $\Theta \in \mathbb{R}^3$. Assume that Θ represents the classical roll-pitch-yaw sequence, namely ${}^{\mathcal{I}}R_B(\Theta) = R_z(\Theta_3)R_y(\Theta_2)R_x(\Theta_1)$. Then, we want to prove the state of Corollary 1.

By substituting the first-order Taylor expansion ${}^{\mathcal{I}}R_B = I_3 + \Theta\times$ into Equation (8.1.7a), we obtain the following:

$$
\begin{aligned}
{}_{\mathcal{I}}f &\approx lw|e_3^\top \left(I_3 + \Theta\times\right) e_3| \left[k(\bar{o}_B - o_B) - b\dot{o}_B\right] && \text{(C.0.1a)}\\
&= lw \left[k(\bar{o}_B - o_B) - b\dot{o}_B\right], && \text{(C.0.1b)}
\end{aligned}
$$

where the last term is equivalent to Equation (8.1.9a) with $\mathcal{K}_l = lwkI_3$ and $\mathcal{B}_l = lwbI_3$.

To prove Equation (8.1.9b) we first set ${}^{\mathcal{I}}\bar{R}_B = I_3$

$$
\begin{aligned}
{}_{B[\mathcal{I}]}\mu = \frac{lw}{12}|e_3^\top {}^{\mathcal{I}}R_B e_3| && \text{(C.0.2a)}\\
\Big\{ l^2 k({}^{\mathcal{I}}R_B e_1) \times e_1 && \text{(C.0.2b)}\\
+ l^2 b({}^{\mathcal{I}}R_B e_1) \times ({}^{\mathcal{I}}R_B e_1) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} && \text{(C.0.2c)}\\
+ w^2 k({}^{\mathcal{I}}R_B e_2) \times e_2 && \text{(C.0.2d)}\\
+ w^2 b({}^{\mathcal{I}}R_B e_2) \times ({}^{\mathcal{I}}R_B e_2) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} \Big\}. && \text{(C.0.2e)}
\end{aligned}
$$

We now analyze the contribution of each term in Equation (C.0.2). By substituting the first order Taylor expansion ${}^{\mathcal{I}}R_B = I_3 + \Theta\times$ in Equation (C.0.2b) and applying

the vector triple product [1], we obtain

$$l^2k({}^{\mathcal{I}}R_B e_1) \times e_1 = l^2k[(I_3 + \Theta\times)e_1] \times e_1 \tag{C.0.3a}$$
$$= l^2k[e_1 \times e_1 + (\Theta \times e_1) \times e_1] \tag{C.0.3b}$$
$$= l^2k[(e_1^\top \Theta)e_1 - \Theta] \tag{C.0.3c}$$
$$= l^2k \begin{bmatrix} 0 & -\Theta_2 & -\Theta_3 \end{bmatrix}^\top. \tag{C.0.3d}$$

The very same approach is also valid for (C.0.2d):

$$w^2k({}^{\mathcal{I}}R_B e_2) \times e_2 = w^2k[(I_3 + \Theta\times)e_2] \times e_2 \tag{C.0.4a}$$
$$= w^2k[e_2 \times e_2 + (\Theta \times e_2) \times e_2] \tag{C.0.4b}$$
$$= w^2k[(e_2^\top \Theta)e_2 - \Theta] \tag{C.0.4c}$$
$$= w^2k \begin{bmatrix} -\Theta_1 & 0 & -\Theta_3 \end{bmatrix}^\top. \tag{C.0.4d}$$

Let us now compute the small-angle approximation for the angular velocity ${}^{\mathcal{I}}\omega_{\mathcal{I},B}$. We recall that it is always possible to compute the angular velocity from the rate of change of the Euler parametrization. In the case of roll-pitch-yaw parametrization we have:

$${}^{\mathcal{I}}\omega_{\mathcal{I},B} = \begin{bmatrix} R_z(\Theta_3)R_y(\Theta_2)e_1 & R_z(\Theta_3)e_2 & e_3 \end{bmatrix} \dot{\Theta}. \tag{C.0.5}$$

We now evaluate the first-order approximation of the angular velocity as a function of the Euler angle rate of change:

[1] Given three vectors $a, b, c \in \mathbb{R}^3$, the following relationship holds:

$$a \times (b \times c) = \left(a^\top c\right) b - \left(a^\top b\right) c.$$

$$
\begin{aligned}
{}^{\mathcal{I}}\omega_{\mathcal{I},B} &= \begin{bmatrix} R_z(\Theta_3)R_y(\Theta_2)e_1 & R_z(\Theta_3)e_2 & e_3 \end{bmatrix} \dot{\Theta} && \text{(C.0.6a)}\\
&= \begin{bmatrix} \cos(\Theta_2)\cos(\Theta_3) & -\sin(\Theta_3) & 0 \\ \cos(\Theta_2)\sin(\Theta_3) & \cos(\Theta_3) & 0 \\ -\sin(\Theta_2) & 0 & 1 \end{bmatrix} \dot{\Theta} && \text{(C.0.6b)}\\
&= \begin{bmatrix} -\cos(\Theta_2)\cos(\Theta_3) - 1 & -\sin(\Theta_3) & 0 \\ \cos(\Theta_2)\sin(\Theta_3) & \cos(\Theta_3) - 1 & 0 \\ -\sin(\Theta_2) & 0 & 0 \end{bmatrix} \dot{\Theta} + \dot{\Theta} && \text{(C.0.6c)}\\
&\approx \begin{bmatrix} -\frac{\Theta_2^2\Theta_3^2}{2} & -\Theta_3 & 0 \\ -\Theta_3 & -\frac{\Theta_3}{2} & 0 \\ -\Theta_2 & 0 & 0 \end{bmatrix} \dot{\Theta} + \dot{\Theta} && \text{(C.0.6d)}\\
&\approx \dot{\Theta}. && \text{(C.0.6e)}
\end{aligned}
$$

We can conclude that for small angles, the angular velocity can be approximated to the rate of change of the Euler angle, i.e. ${}^{\mathcal{I}}\omega_{\mathcal{I},B} \approx \dot{\Theta}$.

Considering the angular velocity approximation and substituting the first order Taylor expansion ${}^{\mathcal{I}}R_B = I_3 + \Theta\times$ in the term (C.0.2c), we obtain

$$
\begin{aligned}
({}^{\mathcal{I}}R_B e_1) \times ({}^{\mathcal{I}}R_B e_1) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} &= \left[({}^{\mathcal{I}}R_B e_1)\times\right]^2 {}^{\mathcal{I}}\omega_{\mathcal{I},B} && \text{(C.0.7a)}\\
&\approx \left\{\left[\left(I_3 + \Theta\times\right) e_1\right] \times\right\}^2 \dot{\Theta} && \text{(C.0.7b)}\\
&= \begin{bmatrix} 0 & \Theta_2 & \Theta_3 \\ -\Theta_2 & 0 & -1 \\ -\Theta_3 & 1 & 0 \end{bmatrix}^2 \dot{\Theta} && \text{(C.0.7c)}\\
&= \begin{bmatrix} -\Theta_2^2 - \Theta_3^2 & \Theta_3 & -\Theta_2 \\ \Theta_3 & -\Theta_2^2 - 1 & -\Theta_3\Theta_2 \\ -\Theta_2 & -\Theta_3\Theta_2 & -\Theta_3^2 - 1 \end{bmatrix} \dot{\Theta} && \text{(C.0.7d)}\\
&\approx \begin{bmatrix} 0 & \Theta_3 & -\Theta_2 \\ \Theta_3 & -1 & 0 \\ -\Theta_2 & 0 & -1 \end{bmatrix} \dot{\Theta} && \text{(C.0.7e)}\\
&\approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \dot{\Theta}. && \text{(C.0.7f)}
\end{aligned}
$$

Here we neglect all the second-order terms.

The very same approach is also valid for (C.0.2e):

$$
\begin{aligned}
({}^{\mathcal{I}}R_B e_2) \times ({}^{\mathcal{I}}R_B e_2) \times {}^{\mathcal{I}}\omega_{\mathcal{I},B} &= \left[({}^{\mathcal{I}}R_B e_2)\times\right]^2 {}^{\mathcal{I}}\omega_{\mathcal{I},B} && \text{(C.0.8a)}\\
&\approx \left\{\left[\left(I_3 + \Theta\times\right) e_2\right] \times\right\}^2 \dot{\Theta} && \text{(C.0.8b)}\\
&= \begin{bmatrix} -\Theta_1^2 - 1 & -\Theta_3 & -\Theta_1\Theta_3 \\ -\Theta_3 & -\Theta_1^2 - \Theta_3^2 & \Theta_1 \\ -\Theta_1\Theta_3 & \Theta_1 & -\Theta_3^2 - 1 \end{bmatrix} \dot{\Theta} && \text{(C.0.8c)}\\
&\approx \begin{bmatrix} -1 & -\Theta_3 & 0 \\ -\Theta_3 & 0 & \Theta_1 \\ 0 & \Theta_1 & -1 \end{bmatrix} \dot{\Theta} && \text{(C.0.8d)}\\
&\approx \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \dot{\Theta}. && \text{(C.0.8e)}
\end{aligned}
$$

By substituting (C.0.3) (C.0.7) (C.0.4) and (C.0.2) and remembering that $|e_3^\top {}^{\mathcal{I}}R_B e_3| \approx 1$ for a small rotation we obtain

$$
{}_{B[\mathcal{I}]}\mu \approx \frac{lw}{12}\left(l^2 k \begin{bmatrix} 0 \\ -\Theta_2 \\ -\Theta_3 \end{bmatrix} + w^2 k \begin{bmatrix} -\Theta_1 \\ 0 \\ -\Theta_3 \end{bmatrix} + l^2 b \begin{bmatrix} 0 \\ -\dot{\Theta}_2 \\ -\dot{\Theta}_3 \end{bmatrix} + w^2 b \begin{bmatrix} -\dot{\Theta}_1 \\ 0 \\ -\dot{\Theta}_3 \end{bmatrix} \right), \qquad \text{(C.0.9)}
$$

that is equivalent to Equation (8.1.9b) with

$$
\mathcal{K}_a = k\frac{lw}{12}\begin{bmatrix} w^2 & 0 & 0 \\ 0 & l^2 & 0 \\ 0 & 0 & l^2{+}w^2 \end{bmatrix} \qquad \mathcal{B}_a = b\frac{lw}{12}\begin{bmatrix} w^2 & 0 & 0 \\ 0 & l^2 & 0 \\ 0 & 0 & l^2{+}w^2 \end{bmatrix}. \qquad \text{(C.0.10)}
$$

Appendix D

Optimal Trajectory Planning in $\mathbb{R}^n$

This appendix addresses the problem of the optimal trajectory planning in $\mathbb{R}^n$. We first recall the *Hamilton's Variational Principle* method as a solution of two-point trajectory planning, then we apply the technique to compute a minimum acceleration and minimum jerk trajectory in $\mathbb{R}^n$.

D.1 Notes on Hamilton's Variational Principle

Given a fixed initial point $x(t_0) = x_0 \in \mathbb{R}^n$ and a final point $x(t_f) = x_f \in \mathbb{R}^n$ we aim to compute the trajectory $x(t)$ such that is a *stationary point*[1] of the *action functional*:

$$\mathfrak{G} = \int_{t_0}^{t_f} \mathcal{L}\left(x, \dot{x}, \ldots, x^{(m)}\right) \mathrm{d}\, t, \tag{D.1.1}$$

where $\mathcal{L}\left(x, \dot{x}, \ldots, x^{(m)}\right)$ is the Lagrangian function and depends on the trajectory x and its derivative, here the superscripts (m) denotes the m-order time derivative of $x(t)$. The optimization problem can be solved by applying *Hamilton's Variational Principle* [Liberzon, 2012]. To do so, we first introduce the concept of variations on $\mathbb{R}^n$ and then we present the *Hamilton's Variational Principle*.

Suppose that a curve $x : [t_0, t_f] \to \mathbb{R}^n$ describes a trajectory in $\mathbb{R}^n$. We now introduce the variation of the curve $x(t)$, which is the ϵ-parameterized family of curves $x_\epsilon(t)$ taking values in $\mathbb{R}^n$, where $\epsilon \in (-c, c)$ with $c > 0$, $x_0(t) = x(t)$ and the endpoints are fixed, i.e., $x_\epsilon(T_i) = x(T_i)$ and $x_\epsilon(T_{i+1}) = x(T_{i+1})$. The Taylor expansion of $x_\epsilon(t)$ is

[1] A *stationary point* of a differentiable function of one variable is a point on the graph of the function where the function's derivative is zero.

given by

$$x_\epsilon(t) = x(t) + \epsilon\delta x(t) + \mathcal{O}\left(\epsilon^2\right). \tag{D.1.2}$$

The i-th order time derivative of (D.1.2) writes as

$$x_\epsilon^{(i)} = x^{(i)} + \epsilon\delta x^{(i)} + \mathcal{O}\left(\epsilon^2\right), \tag{D.1.3}$$

where for the sake of clarity the explicit dependency on the time is hidden. The infinitesimal variation of motion $\delta x^{(i)}$ is given by

$$\delta x^{(i)} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} x_\epsilon^{(i)}\bigg|_{\epsilon=0}, \tag{D.1.4}$$

where the infinitesimal variation satisfies the fixed-endpoint conditions $\delta x^{(i)}(t_0) = 0$ and $\delta x^{(i)}(t_f) = 0$ for $i \in [0, m-1]$.

We now introduce the *action functional* along a variation of a motion $\mathfrak{G}_\epsilon$ as

$$\mathfrak{G}_\epsilon = \int_{t_0}^{t_f} \mathcal{L}\left(x_\epsilon, \dot{x}_\epsilon, \ldots, x_\epsilon^{(m)}\right) \mathrm{d}\,t. \tag{D.1.5}$$

Similarly to what is discussed in Equation (D.1.2), the first-order Taylor expansion of the $\mathfrak{G}_\epsilon$ is given by

$$\mathfrak{G}_\epsilon = \mathfrak{G} + \epsilon\delta\mathfrak{G} + \mathcal{O}\left(\epsilon^2\right), \tag{D.1.6}$$

where the infinitesimal variation of the action functional is given by

$$\delta\mathfrak{G} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\mathfrak{G}_\epsilon\bigg|_{\epsilon=0}. \tag{D.1.7}$$

Hamilton's principle states that the evolution of a system that minimizes the action functional (D.1.1) is a stationary point of $\mathfrak{G}$. More formally, for all possible variations $x_\epsilon(t)$ with fixed endpoints, we have

$$\delta\mathfrak{G} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\mathfrak{G}_\epsilon\bigg|_{\epsilon=0} = 0. \tag{D.1.8}$$

Since the co-domain of $x(t)$, $\mathbb{R}^n$, is a vector space, we determine the action integral by differentiating Equation (D.1.5) as

$$\delta\mathfrak{G} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\mathfrak{G}_\epsilon\bigg|_{\epsilon=0} = \int_{t_0}^{t_f}\sum_{k=0}^{m}\left\langle\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x^{(k)}\right\rangle\mathrm{d}\,t \tag{D.1.9a}$$

$$= \sum_{k=0}^{m}\int_{t_0}^{t_f}\left\langle\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x^{(k)}\right\rangle\mathrm{d}\,t, \tag{D.1.9b}$$

where in (D.1.9b) we exploit the fact that the summation set is finite and the integral converges. Here $\langle .,. \rangle$ is the scalar product operator. By integrating (D.1.9b) by parts, we obtain

$$\delta\mathfrak{G} = \sum_{k=0}^{m}\int_{t_0}^{t_f}\left\langle\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x^{(k)}\right\rangle\mathrm{d}\,t \tag{D.1.10a}$$

$$= \sum_{k=0}^{m}\sum_{i=0}^{k-1}(-1)^i\left\langle\frac{\mathrm{d}^i}{\mathrm{d}\,t^i}\frac{\partial\mathcal{L}}{\partial x^{(i)}},\delta x^{(k-1-i)}\right\rangle\Bigg|_{t_0}^{t_f} \tag{D.1.10b}$$

$$+\sum_{k=0}^{m}(-1)^k\int_{t_0}^{t_f}\left\langle\frac{\mathrm{d}^k}{\mathrm{d}\,t^k}\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x\right\rangle\mathrm{d}\,t. \tag{D.1.10c}$$

Using the fact that the infinitesimal variations $\delta x^{(i)}$ vanish at t_0 and t_f for $i \in [0, m-1]$, we simplify Equation (D.1.10) as

$$\delta\mathfrak{G} = \sum_{k=0}^{m}(-1)^k\int_{t_0}^{t_f}\left\langle\frac{\mathrm{d}^k}{\mathrm{d}\,t^k}\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x\right\rangle\mathrm{d}\,t \tag{D.1.11a}$$

$$= \int_{t_0}^{t_f}\left\langle\sum_{k=0}^{m}(-1)^k\frac{\mathrm{d}^k}{\mathrm{d}\,t^k}\frac{\partial\mathcal{L}}{\partial x^{(k)}},\delta x\right\rangle\mathrm{d}\,t, \tag{D.1.11b}$$

where in (D.1.11b) we exploit the fact that the summation set is finite and the integral converges.

We now recall that Hamilton's principle has to valid for all possible variations, consequently $\delta\mathfrak{G} = 0$ implies that

$$\sum_{k=0}^{m}(-1)^k\frac{\mathrm{d}^k}{\mathrm{d}\,t^k}\frac{\partial\mathcal{L}}{\partial x^{(k)}} = 0. \tag{D.1.12}$$

To conclude, a trajectory $x(t)$ is a *stationary point* of the *action functional* (D.1.1) if and only if it is a solution of the partial differential equation (D.1.12).

D.2 Minimum acceleration trajectory in $\mathbb{R}^n$

Given a set of points, also denoted as knots, associated to a scalar parameter t representing the time, i.e., (t_0, x_0), (t_1, x_1), ..., (t_N, x_N) such that $t_i \geq 0$ and $x_i \in \mathbb{R}^n$ and given a desired initial and final velocity, $(t_0, \dot{x}_0)$ and $(t_N, \dot{x}_N)$. We aim to compute a trajectory $x : \mathbb{R}_+ \to \mathbb{R}^n$ passing to knots having the desired initial and final velocity such that the second order derivative of $x(t)$ with respect to the time, i.e., the acceleration, is minimized.

More formally, we seek for a trajectory $x^*(t)$ such that is a stationary point of the action (D.1.1) with a Lagrangian function defined as

$$\mathcal{L}(\ddot{x}) = \ddot{x}^\top \ddot{x}. \tag{D.2.1}$$

Considering the set of points $\{(t_i, x_i)\}$ and the Lagrangian function (D.2.1), the action functional $\mathfrak{G}$ (D.1.1) writes as

$$\mathfrak{G}_{\ddot{x}} = \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \ddot{x}^\top \ddot{x} \, \mathrm{d}\, t. \tag{D.2.2}$$

Since all the terms within the sum of (D.2.2) are positive, the minimization of $\mathfrak{G}_{\ddot{x}}$ is equivalent to the minimization of each integral term in (D.2.2).

Substituting (D.2.1) into (D.1.12), we obtain

$$x^{(4)}(t) = 0, \tag{D.2.3}$$

we can conclude that a trajectory $x(t)$ that satisfies (D.2.3) minimizes

$$\int_{t_i}^{t_{i+1}} \ddot{x}^\top \ddot{x} \, \mathrm{d}\, t, \tag{D.2.4}$$

and consequently it is a minimum acceleration trajectory in the interval t_i, t_{i+1}. One of the infinite solutions of (D.2.3) is given by the family of the 3$^{\text{rd}}$-order polynomial functions

$$x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0. \tag{D.2.5}$$

The optimal trajectory that minimizes (D.2.2) is given by the concatenation of the 3$^{\text{rd}}$-order polynomial functions (D.2.5), where the coefficients a_i are chosen to satisfy

the boundary conditions of position and velocity

$$x(t_i) = x_i \qquad \dot{x}(t_i) = \dot{x}_i \tag{D.2.6a}$$

$$x(t_{i+1}) = x_{i+1} \quad \dot{x}(t_{i+1}) = \dot{x}_{i+1}. \tag{D.2.6b}$$

Given a subtrajectory

$$s_i : x(t) = a_{i,3}(t - t_i)^3 + a_{i,2}(t - t_i)^2 + a_{i,1}(t - t_i) + a_{i,0}, \tag{D.2.7}$$

defined in the closed domain $[t_i, t_{i+1}]$, the coefficients $a_{i,j}$ are equal to:

$$a_{i,0} = x_i \tag{D.2.8a}$$

$$a_{i,1} = \dot{x}_i \tag{D.2.8b}$$

$$a_{i,2} = -\frac{3x_i - 3x_{i+1} + 2\delta_i\dot{x}_i + \delta_i\dot{x}_{i+1}}{\delta_i^2} \tag{D.2.8c}$$

$$a_{i,3} = \frac{2x_i - 2x_{i+1} + \delta_i\dot{x}_i + \delta_i\dot{x}_{i+1}}{\delta_i^3}, \tag{D.2.8d}$$

where $\delta_i = t_{i+1} - t_i$. Given a knot (t_i, x_i) such that $i \neq 0$ and $i \neq N$, we compute the velocity $\dot{x}_i$ by asking for continuous acceleration at the knots, i.e.,

$$\frac{\mathrm{d}^2}{\mathrm{d}\,t^2} s_{i-1}\bigg|_{t=t_i} = \frac{\mathrm{d}^2}{\mathrm{d}\,t^2} s_i\bigg|_{t=t_i} \tag{D.2.9}$$

By substituting (D.2.8) into (D.2.7) applying the constraint (D.2.9) and imposing the continuity at t_i, we obtain the following linear equation

$$\frac{3x_{i-1} - 3x_i + 2\delta_{i-1}\dot{x}_i + \delta_{i-1}\dot{x}_{i-1}}{\delta_{i-1}^2} + \frac{3x_i - 3x_{i+1} + 2\delta_i\dot{x}_i + \delta_i\dot{x}_{i+1}}{\delta_i^2} = 0. \tag{D.2.10}$$

We note that for $i = 1$ and $i = N - 1$ the terms $\dot{x}_{i-1}$ and $\dot{x}_{i+1}$ are, respectively, known. In fact, when $i = 1$, $\dot{x}_{i-1} = \dot{x}_0$, whether $i = N - 1$, $\dot{x}_{i+1} = \dot{x}_N$. By stacking Equation (D.2.10) for all the points (t_i, x_i) such that $i \neq 0$ and $i \neq N$, we obtain a sparse linear system which can be easily solved online.

Once the velocity of all the knots has been determined, the coefficients (D.2.8) can be evaluated and, consequently, the minimum acceleration trajectory is completely defined.

D.3 Minimum jerk trajectory in $\mathbb{R}^n$

Given a set of knots associated to the a time instant t, i.e., (t_0, x_0), (t_1, x_1), ..., (t_N, x_N) such that $t_i \geq 0$ and $x_i \in \mathbb{R}^n$ and given a desired initial and final velocity, $(t_0, \dot{x}_0)$, $(t_N, \dot{x}_N)$, and acceleration $(t_0, \ddot{x}_0)$, $(t_N, \ddot{x}_N)$. We aim to compute a trajectory $x : \mathbb{R}_+ \to \mathbb{R}^n$ passing to knots satisfying the boundary conditions such that the third-order derivative of $x(t)$ with respect to the time, i.e., the jerk, is minimized.

Following the same approach presented in Appendix D.2 and defining the Lagrangian function as

$$\mathcal{L}(\ddot{x}) = \dddot{x}^\top \dddot{x}. \tag{D.3.1}$$

we can conclude that a trajectory $x(t)$ that satisfies

$$x^{(6)}(t) = 0, \tag{D.3.2}$$

is a minimum jerk trajectory in the interval $[t_i, t_{i+1}]$. One of the infinite solutions of (D.3.2) is given by the family of the 5^{th}-order polynomial function

$$x(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0. \tag{D.3.3}$$

The optimal trajectory that minimizes the jerk in the interval $[t_0, t_N]$ is given by the concatenation of the 5^{th}-order polynomial functions (D.3.3), where the coefficients a_i are chosen to satisfy the position, velocity, and acceleration boundary conditions

$$x(t_i) = x_i \qquad \dot{x}(t_i) = \dot{x}_i \qquad \ddot{x}(t_i) = \ddot{x}_i \tag{D.3.4a}$$

$$x(t_{i+1}) = x_{i+1} \quad \dot{x}(t_{i+1}) = \dot{x}_{i+1} \quad \ddot{x}(t_{i+1}) = \ddot{x}_{i+1}. \tag{D.3.4b}$$

Given a subtrajectory

$$s_i : x(t) = a_{i,5}(t-t_i)^5 + a_{i,4}(t-t_i)^4 + a_{i,3}(t-t_i)^3 + a_{i,2}(t-t_i)^2 + a_{i,1}(t-t_i) + a_{i,0}, \tag{D.3.5}$$

defined in the closed domain $[t_i, t_{i+1}]$, the coefficients $a_{i,j}$ are equal to:

$$a_{i,0} = x_i \tag{D.3.6a}$$

$$a_{i,1} = \dot{x}_i \tag{D.3.6b}$$

$$a_{i,2} = \frac{\ddot{x}_i}{2} \tag{D.3.6c}$$

$$a_{i,3} = -\frac{20x_i - 20x_{i+1} + 12\delta_i\dot{x}_i + 8\delta_i\dot{x}_{i+1} + 3\delta_i^2\ddot{x}_i - \delta_i^2\ddot{x}_{i+1}}{2\delta_i^3} \tag{D.3.6d}$$

$$a_{i,4} = \frac{30x_i - 30x_{i+1} + 16\delta_i\dot{x}_i + 14\delta_i\dot{x}_{i+1} + 3\delta_i^2\ddot{x}_i - 2\delta_i^2\ddot{x}_{i+1}}{2\delta_i^4} \tag{D.3.6e}$$

$$a_{i,5} = -\frac{12x_i - 12x_{i+1} + 6\delta_i\dot{x}_i + 6\delta_i\dot{x}_{i+1} + \delta_i^2\ddot{x}_i - \delta_i^2\ddot{x}_{i+1}}{2\delta_i^5}, \tag{D.3.6f}$$

where $\delta_i = t_{i+1} - t_i$. Given a knot (t_i, x_i) such that $i \neq 0$ and $i \neq N$, we compute the velocity $\dot{x}_i$ and the acceleration $\ddot{x}_i$ by asking for continuous jerk and snap at the knots, i.e.,

$$\frac{\mathrm{d}^3}{\mathrm{d}\,t^3} s_{i-1}\bigg|_{t=t_i} = \frac{\mathrm{d}^3}{\mathrm{d}\,t^3} s_i\bigg|_{t=t_i}, \qquad \frac{\mathrm{d}^4}{\mathrm{d}\,t^4} s_{i-1}\bigg|_{t=t_i} = \frac{\mathrm{d}^4}{\mathrm{d}\,t^4} s_i\bigg|_{t=t_i}. \tag{D.3.7}$$

By substituting (D.3.6) into (D.3.5) applying the constraint (D.3.7) and imposing the continuity at t_i, we obtain the following linear equations

$$\left[\begin{array}{cc|cc|cc} \frac{8}{\delta t_{i-1}^2} & \frac{1}{\delta t_{i-1}} & 12\left(\frac{1}{\delta t_{i-1}^2} - \frac{1}{\delta t_i^2}\right) & -3\left(\frac{1}{\delta t_{i-1}} + \frac{1}{\delta t_i}\right) & -\frac{8}{\delta t_i^2} & \frac{1}{\delta t_i} \\ \frac{14}{\delta t_{i-1}^3} & \frac{2}{\delta t_{i-1}^2} & 16\left(\frac{1}{\delta t_{i-1}^3} + \frac{1}{\delta t_i^3}\right) & 3\left(-\frac{1}{\delta t_{i-1}^2} + \frac{1}{\delta t_i^2}\right) & \frac{14}{\delta t_i^3} & -\frac{2}{\delta t_i^2} \end{array}\right] \left[\begin{array}{c} \dot{x}_{i-1} \\ \ddot{x}_{i-1} \\ \hline \dot{x}_i \\ \ddot{x}_i \\ \hline \dot{x}_{i+1} \\ \ddot{x}_{i+1} \end{array}\right] = \left[\begin{array}{c} \frac{20(x_i - x_{i-1})}{\delta t_{i-1}^3} + \frac{20(x_i - x_{i+1})}{\delta t_i^3} \\ \frac{30(x_i - x_{i-1})}{\delta t_{i-1}^4} - \frac{30(x_i - x_{i+1})}{\delta t_i^4} \end{array}\right]. \tag{D.3.8}$$

By stacking (D.3.8) for all points (t_i, x_i) such that $i \neq 0$ and $i \neq N$, we obtain a sparse linear system which can be easily solved online to compute the velocity and the acceleration at $t = t_i$. Once the boundary conditions for each subtrajectory have been computed, we determine the coefficients (D.3.6) for each subtrajectory. The minimum jerk trajectory is finally given by the concatenation of all the subtrajectories s_i.

Appendix E

Optimal Trajectory Planning in SO(3)

This appendix faces the problem of planning optimal trajectory in SO(3). We first extend *Hamilton's Variational Principle* presented in Appendix D.1 as a solution of the two-point trajectory planning in SO(3), then we apply the technique to compute a minimum acceleration trajectory in SO(3).

E.1 Hamilton's Variational Principle in SO(3)

Given a fixed initial rotation $R(t_0) = R_0 \in \mathrm{SO}(3)$ and a final rotation $R(t_f) = R_f \in \mathrm{SO}(3)$ we aim to compute the trajectory $R(t)$ such that it is a *stationary point* of the *action functional*:

$$\mathfrak{G} = \int_{t_0}^{t_f} \mathcal{L}\left(R, \omega, \dot{\omega}\right) \mathrm{d}\, t, \tag{E.1.1}$$

where $\mathcal{L}\left(R, \omega, \dot{\omega}\right)$ is the Lagrangian function and depends on the trajectory R, on the angular velocity ω, and on the angular acceleration $\dot{\omega}$. In this appendix, we assume that the angular velocity is left trivialized, i.e. $\dot{R} = R\omega\times$. The angular acceleration is the time differentiation of the angular velocity [Traversaro, 2017, Section 2.4.2] The optimization problem can be solved by applying *Hamilton's Variational Principle* [Lee et al., 2018, Section 6.3.1]. Similarly to Appendix D.1 we introduce the concept of variations on SO(3).

Suppose that a curve $R : [t_0, t_f] \to \mathrm{SO}(3)$ describes a trajectory in SO(3). We now introduce the variation of the curve $R(t)$, which is the ϵ-parameterized family of curves $R_\epsilon(t)$ taking values in SO(3), where $\epsilon \in (-c, c)$ with $c > 0$, $R_0(t) = R(t)$ and

the endpoints are fixed, that is, $R_\epsilon(t_0) = R(t_0)$ and $R_\epsilon(t_f) = R(t_f)$. We describe the variation of the rotational motion by means of the exponential map

$$R_\epsilon(t) = R(t) \exp\left(\epsilon \eta(t)^\wedge\right). \tag{E.1.2}$$

The variation of the angular velocity ω_ϵ is given by

$$\begin{aligned} \omega_\epsilon &= R_\epsilon^\top \dot{R}_\epsilon && \text{(E.1.3a)} \\ &= \exp\left(-\epsilon\eta^\wedge\right) R^\top \frac{\mathrm{d}}{\mathrm{d}\,t}\left[R \exp\left(\epsilon\eta^\wedge\right)\right], && \text{(E.1.3b)} \end{aligned}$$

where, for the sake of clarity, we suppress the time dependency. The variation in angular acceleration derives from the time differentiation of (E.1.3) as:

$$\begin{aligned} \dot{\omega}_\epsilon &= \frac{\mathrm{d}}{\mathrm{d}\,t}\left(R_\epsilon^\top \dot{R}_\epsilon\right) && \text{(E.1.4a)} \\ &= \frac{\mathrm{d}}{\mathrm{d}\,t}\left\{\exp\left(-\epsilon\eta^\wedge\right) R^\top \frac{\mathrm{d}}{\mathrm{d}\,t}\left[R \exp\left(\epsilon\eta^\wedge\right)\right]\right\}. && \text{(E.1.4b)} \end{aligned}$$

We now determine the infinitesimal variation of (E.1.2) (E.1.3) (E.1.4). The infinitesimal variation of R, denoted with δR, is given by

$$\begin{aligned} \delta R &= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} R_\epsilon \Big|_{\epsilon=0} && \text{(E.1.5a)} \\ &= R \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} \exp\left(\epsilon\eta^\wedge\right)\Big|_{\epsilon=0} && \text{(E.1.5b)} \\ &= R\eta^\wedge \exp\left(\epsilon\eta^\wedge\right)|_{\epsilon=0} && \text{(E.1.5c)} \\ &= R\eta^\wedge. && \text{(E.1.5d)} \end{aligned}$$

The infinitesimal variation of ω, denoted with $\delta\omega$ derives from Equation (E.1.3) as

$$\begin{aligned} \delta\omega &= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} \omega_\epsilon \Big|_{\epsilon=0} && \text{(E.1.6a)} \\ &= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\left(R_\epsilon^\top \dot{R}_\epsilon\right)\Big|_{\epsilon=0} && \text{(E.1.6b)} \\ &= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} R_\epsilon^\top \dot{R}_\epsilon \Big|_{\epsilon=0} + R_\epsilon^\top \frac{\mathrm{d}}{\mathrm{d}\,\epsilon} \dot{R}_\epsilon \Big|_{\epsilon=0}. && \text{(E.1.6c)} \end{aligned}$$

We now notice that $\dot{R}_\epsilon$ is given by

$$\dot{R}_\epsilon = \frac{\mathrm{d}}{\mathrm{d}\,t}\left(R\exp\left(\epsilon\eta^\wedge\right)\right) \tag{E.1.7a}$$
$$= R\omega^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\frac{\mathrm{d}}{\mathrm{d}\,t}\exp\left(\epsilon\eta^\wedge\right) \tag{E.1.7b}$$
$$= R\omega^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\frac{\mathrm{d}}{\mathrm{d}\,t}\left(I_3 + \epsilon\eta^\wedge + \mathcal{O}(\epsilon^2)\right) \tag{E.1.7c}$$
$$= R\omega^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right). \tag{E.1.7d}$$

By substituting (E.1.7d) into (E.1.6) and recalling that $\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}R_\epsilon = R\eta^\wedge\exp\left(\epsilon\eta^\wedge\right)$ (E.1.5c), we have the following:

$$\delta\omega^\wedge = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}R_\epsilon^\top\dot{R}_\epsilon\Big|_{\epsilon=0} + R_\epsilon^\top\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon\Big|_{\epsilon=0} \tag{E.1.8a}$$
$$= \left[-\exp\left(-\epsilon\eta^\wedge\right)\eta^\wedge R^\top\right]\left[R\omega^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right)\right]\Big|_{\epsilon=0} \tag{E.1.8b}$$
$$+ \left[\exp\left(-\epsilon\eta^\wedge\right)R^\top\right]\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\left[R\omega^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right)\right]\Big|_{\epsilon=0} \tag{E.1.8c}$$
$$= -\eta^\wedge R^\top R\omega^\wedge \tag{E.1.8d}$$
$$+ \left[\exp\left(-\epsilon\eta^\wedge\right)R^\top\right]\left[R\omega^\wedge\eta^\wedge\exp\left(\epsilon\eta^\wedge\right) + R\left(\dot{\eta}^\wedge + \mathcal{O}(\epsilon)\right)\right]\Big|_{\epsilon=0} \tag{E.1.8e}$$
$$= -\eta^\wedge R^\top R\omega^\wedge + R^\top\left[R\omega^\wedge\eta^\wedge + R\dot{\eta}^\wedge\right] \tag{E.1.8f}$$
$$= \dot{\eta}^\wedge + \omega^\wedge\eta^\wedge - \eta^\wedge\omega^\wedge. \tag{E.1.8g}$$

Here, for the sake of clarity, we use the colors to simplify the reader in following the passages. Applying the *vee* operator (Equation (A.3.3b)) to $\delta\omega^\wedge$ (E.1.8), we obtain the infinitesimal variation of the angular velocity

$$\delta\omega = \dot{\eta} + \omega\times\eta. \tag{E.1.9}$$

The infinitesimal variation of the angular acceleration $\dot{\omega}$, denoted by $\delta\dot{\omega}$, is

$$\delta\dot{\omega}^\wedge = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{\omega}_\epsilon\Big|_{\epsilon=0} \tag{E.1.10a}$$
$$= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\frac{\mathrm{d}}{\mathrm{d}\,t}\left(R_\epsilon^\top\dot{R}_\epsilon\right)\Big|_{\epsilon=0} \tag{E.1.10b}$$
$$= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\left[\dot{R}_\epsilon^\top\dot{R}_\epsilon + R_\epsilon^\top\frac{\mathrm{d}}{\mathrm{d}\,t}\dot{R}_\epsilon\right]\Big|_{\epsilon=0} \tag{E.1.10c}$$
$$= \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon^\top\dot{R}_\epsilon\Big|_{\epsilon=0} + \dot{R}_\epsilon^\top\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon\Big|_{\epsilon=0} + \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}R_\epsilon^\top\ddot{R}_\epsilon\Big|_{\epsilon=0} + R_\epsilon^\top\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\ddot{R}_\epsilon\Big|_{\epsilon=0}. \tag{E.1.10d}$$

We now notice that $\ddot{R}_\epsilon$ can be written as

$$\begin{aligned}
\ddot{R}_\epsilon =& \frac{\mathrm{d}}{\mathrm{d}\,t}\dot{R}_\epsilon && \text{(E.1.11a)}\\
=& \frac{\mathrm{d}}{\mathrm{d}\,t}\left(R\omega^\wedge \exp\left(\epsilon\eta^\wedge\right) + R\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right)\right) && \text{(E.1.11b)}\\
=& \frac{\mathrm{d}}{\mathrm{d}\,t}\left(R\omega^\wedge \exp\left(\epsilon\eta^\wedge\right)\right) + \frac{\mathrm{d}}{\mathrm{d}\,t}\left(R\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right)\right) && \text{(E.1.11c)}\\
=& R(\omega^\wedge)^2 \exp(\epsilon\eta^\wedge) + R(\dot{\omega}^\wedge)\exp(\epsilon\eta^\wedge) + R\omega^\wedge\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right) && \text{(E.1.11d)}\\
&+ R\omega^\wedge\left(\epsilon\dot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right) + R\left(\epsilon\ddot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right) && \text{(E.1.11e)}\\
=& R\left[\left(\dot{\omega}^\wedge + (\omega^\wedge)^2\right)\exp(\epsilon\eta^\wedge) + 2\epsilon\omega^\wedge\dot{\eta}^\wedge + \epsilon\ddot{\eta}^\wedge + \mathcal{O}(\epsilon^2)\right]. && \text{(E.1.11f)}
\end{aligned}$$

By substituting (E.1.11f) into (E.1.10), recalling that $\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}R_\epsilon = R\eta^\wedge \exp\left(\epsilon\eta^\wedge\right)$ (E.1.5c), and $\frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon = R\left[\omega^\wedge\eta^\wedge \exp\left(\epsilon\eta^\wedge\right) + \dot{\eta}^\wedge + \mathcal{O}(\epsilon)\right]$ (E.1.8e) we have:

$$\begin{aligned}
\delta\dot{\omega}^\wedge =& \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon^\top \dot{R}_\epsilon\Big|_{\epsilon=0} + \dot{R}_\epsilon^\top \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\dot{R}_\epsilon\Big|_{\epsilon=0} + \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}R_\epsilon^\top \ddot{R}_\epsilon\Big|_{\epsilon=0} + R_\epsilon^\top \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\ddot{R}_\epsilon\Big|_{\epsilon=0} && \text{(E.1.12a)}\\
=& \left(-\dot{\eta}^\wedge + \eta^\wedge\omega^\wedge\right)\omega^\wedge - \omega^\wedge\left(\dot{\eta}^\wedge + \eta^\wedge\omega^\wedge\right) && \text{(E.1.12b)}\\
&- \eta^\wedge\left(\dot{\omega}^\wedge + (\omega^\wedge)^2\right) + \dot{\omega}^\wedge\eta^\wedge + (\omega^\wedge)^2\eta^\wedge + 2\omega^\wedge\dot{\eta}^\wedge + \ddot{\eta}^\wedge && \text{(E.1.12c)}\\
=& \ddot{\eta}^\wedge + (\dot{\omega}^\wedge\eta^\wedge - \eta^\wedge\dot{\omega}^\wedge) + (\omega^\wedge\dot{\eta}^\wedge - \dot{\eta}^\wedge\omega^\wedge). && \text{(E.1.12d)}
\end{aligned}$$

Finally, by applying the *vee* operator (Equation (A.3.3b)) to $\delta\dot{\omega}^\wedge$ (E.1.12), we obtain the infinitesimal variation of the angular acceleration

$$\delta\dot{\omega} = \ddot{\eta} + \dot{\omega}\times\eta + \omega\times\dot{\eta}. \tag{E.1.13}$$

We now introduce the *action functional* along a variation of a motion $\mathfrak{G}_\epsilon$ as

$$\mathfrak{G}_\epsilon = \int_{t_0}^{t_f} \mathcal{L}\left(R_\epsilon, \omega_\epsilon, \dot{\omega}_\epsilon\right)\mathrm{d}\,t, \tag{E.1.14}$$

then, the first-order Taylor expansion of the $\mathfrak{G}_\epsilon$ is given by

$$\mathfrak{G}_\epsilon = \mathfrak{G} + \epsilon\delta\mathfrak{G} + \mathcal{O}\left(\epsilon^2\right), \tag{E.1.15}$$

where the infinitesimal variation of the action functional is given by

$$\delta\mathfrak{G} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\mathfrak{G}_\epsilon\Big|_{\epsilon=0}. \tag{E.1.16}$$

Hamilton's principle states that the infinitesimal variation of the action integral along any rotational motion is zero

$$\delta\mathfrak{G} = \frac{\mathrm{d}}{\mathrm{d}\,\epsilon}\mathfrak{G}_\epsilon\bigg|_{\epsilon=0} = 0, \tag{E.1.17}$$

for all possible infinitesimal variations η with fixed endpoints, i.e., $\eta(t_0) = 0$ and $\eta(t_f) = 0$. To conclude, combining Hamilton's principle (E.1.17) with the functional (E.1.1) we have the following

$$\delta\mathfrak{G} = \int_{t_0}^{t_f} \left\{ \left\langle \frac{\partial\mathcal{L}}{\partial R}, \delta R \right\rangle + \left\langle \frac{\partial\mathcal{L}}{\partial\omega}, \delta\omega \right\rangle + \left\langle \frac{\partial\mathcal{L}}{\partial\dot{\omega}}, \delta\dot{\omega} \right\rangle \right\} \mathrm{d}\,t = 0, \tag{E.1.18}$$

where δR, $\delta\omega$, and $\delta\dot{\omega}$ are defined in (E.1.5), (E.1.9) and (E.1.13), respectively. Here $\langle ., . \rangle$ is the scalar product operator.

In the next section, we apply Hamilton's principle to design a minimum acceleration trajectory in SO(3).

E.2 Minimum acceleration trajectory in SO(3)

Given a fixed initial rotation (t_0, R_0) and a final rotation (t_f, R_f) and the associated right trivialized angular velocities, (t_0, ω_0) and (t_f, ω_f), we want to compute a trajectory $R : \mathbb{R}_+ \to \mathrm{SO}(3)$ such that $R(t_0) = R_0$, $R(t_f) = R_f$, $\omega(t_0) = \omega_0$, $\omega(t_f) = \omega_f$ such that right trivialized angular acceleration is minimized. More formally, we seek a trajectory $R(t)$ such that it is a stationary point of the action (E.1.1) with a Lagrangian function defined as

$$\mathcal{L}(\dot{\omega}) = \dot{\omega}^\top \dot{\omega}. \tag{E.2.1}$$

By substituting (E.2.1) into (E.1.18) we have

$$\begin{aligned}
\delta\mathfrak{G} &= \int_{t_0}^{t_f} \left\langle \frac{\partial\mathcal{L}(\dot{\omega})}{\partial\dot{\omega}}, \delta\dot{\omega} \right\rangle \mathrm{d}\,t = \int_{t_0}^{t_f} \left\langle \frac{\partial\mathcal{L}(\dot{\omega})}{\partial\dot{\omega}}, \ddot{\eta} + \dot{\omega}\times\eta + \omega\times\dot{\eta} \right\rangle \mathrm{d}\,t && \text{(E.2.2a)}\\
&= \int_{t_0}^{t_f} \left\langle \frac{\partial\mathcal{L}(\dot{\omega})}{\partial\dot{\omega}}, \ddot{\eta} \right\rangle \mathrm{d}\,t && \text{(E.2.2b)}\\
&\quad + \int_{t_0}^{t_f} \left\langle \frac{\partial\mathcal{L}(\dot{\omega})}{\partial\dot{\omega}}, \dot{\omega}\times\eta \right\rangle \mathrm{d}\,t && \text{(E.2.2c)}\\
&\quad + \int_{t_0}^{t_f} \left\langle \frac{\partial\mathcal{L}(\dot{\omega})}{\partial\dot{\omega}}, \omega\times\dot{\eta} \right\rangle \mathrm{d}\,t. && \text{(E.2.2d)}
\end{aligned}$$

We now analyze each term of (E.2.2) explicitly. By integrating (E.2.2b) by parts, we obtain the following:

$$\int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \ddot{\eta} \right\rangle \mathrm{d}\, t = \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \dot{\eta} \right\rangle \Big|_{t_0}^{t_f} \tag{E.2.3a}$$

$$- \left\langle \frac{\mathrm{d}}{\mathrm{d}\, t} \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \eta \right\rangle \Big|_{t_0}^{t_f} \tag{E.2.3b}$$

$$+ \int_{t_0}^{t_f} \left\langle \frac{\mathrm{d}^2}{\mathrm{d}\, t^2} \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \eta \right\rangle \mathrm{d}\, t. \tag{E.2.3c}$$

Recalling the fact that the infinitesimal variations η and $\dot{\eta}$ vanish at t_0 and t_f, Equation (E.2.3) can be simplified as follows:

$$\int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \ddot{\eta} \right\rangle \mathrm{d}\, t = \int_{t_0}^{t_f} \left\langle \frac{\mathrm{d}^2}{\mathrm{d}\, t^2} \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \eta \right\rangle \mathrm{d}\, t. \tag{E.2.4}$$

Applying the properties *scalar triple product*, (E.2.2c) writes as

$$\int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \dot{\omega} \times \eta \right\rangle \mathrm{d}\, t = \int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \dot{\omega}, \eta \right\rangle \mathrm{d}\, t. \tag{E.2.5}$$

Similarly, we rewrite (E.2.2d) applying the properties of the scalar triple product and integrating it by parts:

$$\int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}}, \omega \times \dot{\eta} \right\rangle \mathrm{d}\, t = \int_{t_0}^{t_f} \left\langle \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \omega, \dot{\eta} \right\rangle \mathrm{d}\, t \tag{E.2.6a}$$

$$= - \int_{t_0}^{t_f} \left\langle \frac{\mathrm{d}}{\mathrm{d}\, t} \left(\frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \omega \right), \eta \right\rangle \mathrm{d}\, t, \tag{E.2.6b}$$

where in (E.2.6b), we use the fact that $\eta(t_0) = \eta(t_f) = 0$.

By substituting (E.2.4), (E.2.5) and (E.2.6b) into (E.2.2), we obtain the final formulation of the infinitesimal variation of the action functional:

$$\delta \mathfrak{G} = \int_{t_0}^{t_f} \left\langle \frac{\mathrm{d}^2}{\mathrm{d}\, t^2} \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} - \frac{\mathrm{d}}{\mathrm{d}\, t} \left(\frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \omega \right) + \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \dot{\omega}, \eta \right\rangle \mathrm{d}\, t. \tag{E.2.7}$$

We now recall that Hamilton's principle must be valid for all possible variations η, and consequently $\delta \mathfrak{G} = 0$ implies that

$$\frac{\mathrm{d}^2}{\mathrm{d}\, t^2} \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} - \frac{\mathrm{d}}{\mathrm{d}\, t} \left(\frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \omega \right) + \frac{\partial \mathcal{L}(\dot{\omega})}{\partial \dot{\omega}} \times \dot{\omega} = 0. \tag{E.2.8}$$

Combining the Lagrangian definition (E.2.1) with the partial differential equation (E.2.8) we obtain

$$\dddot{\omega} + \omega \times \ddot{\omega} = 0. \tag{E.2.9}$$

Finally, we can conclude that a trajectory $R(t)$ that satisfies (E.2.9) is a minimum acceleration trajectory.

It is worth noting that (E.2.9) does not admit an analytic solution for arbitrary boundary conditions. However, in the case of zero initial and final velocity ω, it is possible to show that

$$R(t) = \exp\left(s(t-t_0)\log\left(R_f R_0^\top\right)\right) R_0 \tag{E.2.10a}$$

$$s(\tau) = \frac{3}{(t_f - t_0)^2}\tau^2 - \frac{3}{(t_f - t_0)^3}\tau^3, \tag{E.2.10b}$$

satisfies condition (E.2.9). To prove the latest statement, we first compute the right trivialized angular velocity as

$$\begin{aligned}
\omega^\wedge &= R^\top \dot{R} && \text{(E.2.11a)}\\
&= \dot{s} R_0^\top \exp\left(-s\log\left(R_f R_0^\top\right)\right)\log\left(R_f R_0^\top\right)\exp\left(s\log\left(R_f R_0^\top\right)\right)R_0 && \text{(E.2.11b)}\\
&= \dot{s} R_0^\top \exp\left(-s\log\left(R_f R_0^\top\right)\right)R_0\log\left(R_0^\top R_f\right)R_0^\top\exp\left(s\log\left(R_f R_0^\top\right)\right)R_0 && \text{(E.2.11c)}\\
&= \dot{s}\exp\left(-s\log\left(R_0^\top R_f\right)\right)\log\left(R_0^\top R_f\right)\exp\left(s\log\left(R_0^\top R_f\right)\right) && \text{(E.2.11d)}\\
&= \dot{s}\exp\left(-s\log\left(R_0^\top R_f\right)\right)\exp\left(s\log\left(R_0^\top R_f\right)\right)\log\left(R_0^\top R_f\right) && \text{(E.2.11e)}\\
&= \dot{s}\log\left(R_0^\top R_f\right). && \text{(E.2.11f)}
\end{aligned}$$

Here, we exploit the exponential and the adjoint maps properties – Equation (A.7.1) and Section A.6. We notice that ω satisfies the constraint (E.2.9):

$$\begin{aligned}
\dddot{\omega} + \omega \times \ddot{\omega} =& \frac{\mathrm{d}^4}{\mathrm{d}\,t^4} s\,\mathrm{Log}\left(R_0^\top R_f\right) && \text{(E.2.12a)}\\
&+ \left(\frac{\mathrm{d}}{\mathrm{d}\,t} s\,\mathrm{Log}\left(R_0^\top R_f\right)\right) \times \left(\frac{\mathrm{d}^3}{\mathrm{d}\,t^3} s\,\mathrm{Log}\left(R_0^\top R_f\right)\right) && \text{(E.2.12b)}\\
=& \frac{\mathrm{d}^4}{\mathrm{d}\,t^4} s\,\mathrm{Log}\left(R_0^\top R_f\right) && \text{(E.2.12c)}\\
=& 0. && \text{(E.2.12d)}
\end{aligned}$$

Furthermore, the trajectory (E.2.10) satisfies the boundary conditions: namely $R(t_0) = R_0$, $R(t_f) = R_f$, $\omega(t_0) = \omega(t_f) = 0$. To conclude, the trajectory (E.2.10) is a minimum acceleration trajectory in the interval $[t_0, t_f]$.