mαcrϕ@ufmg

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
MACRO Research Group - Mecatronics, Control and Robotics

WHOLE-BODY CONTROL OF HUMANOID ROBOTS AT SECOND ORDER KINEMATICS UNDER UNILATERAL CONSTRAINTS

Juan José Quiroz Omaña

Belo Horizonte, Brazil
2021

Juan José Quiroz Omaña

WHOLE-BODY CONTROL OF HUMANOID ROBOTS AT SECOND ORDER KINEMATICS UNDER UNILATERAL CONSTRAINTS

Ph.D. dissertation submitted to the Programa de Pós-Graduação em Engenharia Elétrica of Escola de Engenharia at the Universidade Federal de Minas Gerais, in partial fulfillment of the requirements for the degree of Doctorate in Electrical Engineering.

Advisor: Bruno Vilhena Adorno

Belo Horizonte, Brazil
2021

To my parents, my sisters, and my wife. Thank you for your unconditional support, encouragement, and love.

Acknowledgements

This Ph.D. dissertation is the result of intense hard work and full-time dedication. However, its completion would not be possible without the help and support of many people. This section is dedicated to all of them who made possible the conclusion of this work. To all of you, thank you for your contributions, advice, friendship, and encouragement.

First, I would like to thank to Prof. Bruno Adorno for accepting me as his student both in the Master and the Doctorate. From the first moment, he provided me with excellent infrastructure conditions to study and work. His great and precise technical observations guided me and helped me to consolidate my academic training in the best way. I am deeply grateful for his notes, suggestions, corrections, and English explanations that transformed my draft text into a decent and understandable one.

A special thanks goes to Dr. Abderrahmane Kheddar, Prof. Dr. Murilo Marques Marinho, Prof. Dr. Luciano Cunha de Araújo Pimenta, and Prof. Dr. Vinícius Mariano Gonçalves. In addition to accepting to be reviewers of my dissertation, they contributed to this work with useful remarks, suggestions, and typo corrections.

A special acknowledgment goes to the faculty members at the engineering school at the UFMG. Thank you for their excellent teachings and explanations about different topics that helped my academic education. Thanks also to the Brazilian agencies CAPES, CNPQ, INCTs, and FAPEMIG for their financial support during these four years.

Thanks to my friends and Laboratory colleagues, especially the ones supervised by Prof. Bruno Adorno. Thank you for the technical discussions that helped me better understand academic concepts, and for the drinking nights that helped me cope with stress and intense pressure.

A special word of thanks to all those people in Colombia who supported me, especially my family. I couldn't have come this far without the support of all of you guys. Thank you for the encouraging video calls that helped me get through those times when I most missed being home.

To those who helped in some way, in my best moments and also in the most difficult ones during this stage of my life, please accept this acknowledgment as an expression of my gratitude.

Last, but most importantly, I would like to thank my parents José Alberto and Magali,

and my sisters Paula and Camila. Thank you for believing in me and for supporting this life project that started in Brazil. All this achievement would not be possible without them. Finally, I would like to thank my wife Stella for all the love, support, and understanding. With you standing by my side, my life is happier.

Agradecimientos

Esta tesis doctoral es el resultado de un intenso y arduo trabajo además de una dedicación de tiempo completo. Sin embargo, su finalización no sería posible sin la ayuda y el apoyo de muchas personas. Esta sección está dedicada a todos los que hicieron posible la conclusión de este trabajo. A todos ustedes, gracias por sus contribuciones, consejos, amistad y aliento.

En primer lugar, quiero agradecer al profesor Bruno Adorno por aceptarme como su alumno tanto en el Maestría como en el Doctorado. Desde el primer momento me brindó excelentes condiciones de infraestructura para estudiar y trabajar. Sus excelentes y precisas observaciones técnicas me guiaron y me ayudaron a consolidar mi formación académica de la mejor manera. Estoy profundamente agradecido por sus notas, sugerencias, correcciones y explicaciones en inglés que transformaron mi texto preliminar en uno decente y comprensible.

Un agradecimiento especial para el Dr. Abderrahmane Kheddar, el Prof. Dr. Murilo Marques Marinho, el Prof. Dr. Luciano Cunha de Araújo Pimenta y el Prof. Dr. Vinícius Mariano Gonçalves. Además de aceptar ser revisores de mi disertación, contribuyeron a este trabajo con comentarios útiles, sugerencias y correcciones de errores tipográficos.

Un reconocimiento especial para los profesores de la escuela de ingeniería de la UFMG. Gracias por sus excelentes enseñanzas y explicaciones sobre diferentes temas que ayudaron a mi formación académica. Gracias también a las agencias brasileñas CAPES, CNPQ, INCTs y FAPEMIG por su apoyo financiero durante estos cuatro años.

Gracias a mis amigos y compañeros de laboratorio, especialmente a los supervisados por el Prof. Bruno Adorno. Gracias por las discusiones técnicas que me ayudaron a comprender mejor los conceptos académicos y por las noches de bebida que me ayudaron a sobrellevar el estrés y la presión intensa.

Un agradecimiento especial a todas aquellas personas en Colombia que me apoyaron, especialmente a mi familia. No podría haber llegado tan lejos sin el apoyo de todos ustedes. Gracias por las alentadoras videollamadas que me ayudaron a superar esos momentos en los que más extrañaba estar en casa.

A quienes ayudaron de alguna manera, en mis mejores momentos y también en los más difíciles durante esta etapa de mi vida, por favor acepten este reconocimiento como una

expresión de mi gratitud.

Por último, pero más importante, quisiera agradecer a mis padres José Alberto y Magali, y a mis hermanas Paula y Camila. Gracias por creer en mí y por apoyar este proyecto de vida que comenzó en Brasil. Todo este logro no sería posible sin ustedes. Finalmente, me gustaría agradecer a mi esposa Stella por todo el amor, el apoyo y compresión. Contigo a mi lado, mi vida es más feliz.

Resumo

Este trabalho usa desigualdades de campos vetoriais (DCV) para prevenir colisões com o ambiente e com o próprio robô. As DCVs são estendidas para cinemática de segunda ordem (DCVSO). Diferentemente de trabalhos anteriores, o método pode ser aplicado a robôs atuados tanto em velocidade quanto em torque. Além disso, é apresentada uma prova formal de prevenção de colisões usando DCVs de segunda ordem. É proposta uma nova função de distância e a sua Jacobiana correspondente para gerar uma DCV que evita atingir um ângulo entre duas linhas de Plücker. Essa nova DCV é usada para evitar atingir os limites das juntas ou orientações indesejadas no efetuador. Além disso, é proposta uma nova Jacobiana relacionada com o polígono de suporte de um robô humanoide. Isso é usado para maximizar a área do polígono de suporte do robô e, potencialmente, aumentar o alcance e a segurança do robô em termos do equilíbrio. As Jacobianas propostas e as DCVs são usadas para realizar controle de corpo completo com múltiplos contatos usando um robô humanoide.

O modelo de Euler-Lagrange, o qual é usado com as DCVSOs em robôs atuados em torque, é derivado por meio do princípio da mínima restrição de Gauss usando álgebra de quatérnios duais. O uso de álgebra de quatérnios duais permite uma representação mais compacta e unificada para os heligiros e as heliforças.

O método proposto é avaliado em uma simulação realista e em um humanoide com 27 graus de liberdade e em três robôs reais: um humanoide com 9 graus de liberdade, um manipulador bimanual com 8 graus de liberdade e um manipulador bimanual móvel não holonômico com 16 graus de liberdade. Os resultados mostram que todas as restrições são respeitadas enquanto o robô realiza tarefas de manipulação.

Palavras-chave: Desigualdades de Campos Vetoriais, Robôs Humanoides, Princípio da Mínima Restrição de Gauss, Quatérnios duais, Programação Quadrática.

Abstract

This work uses vector field inequalities (VFIs) to prevent robot self-collisions and collisions with the workspace. We extend the VFIs to second order kinematics (SOVFIs) and, differently from previous approaches, the method is suitable for both velocity and torque-actuated robots. Furthermore, we present a formal proof of collision avoidance using SOVFIs. We propose a new distance function and its corresponding Jacobian in order to generate a VFIs to limit the angle between two Plücker lines. This new VFI is used to prevent both undesired end-effector orientations and violation of joints limits. In addition, we propose a new Jacobian related with the support polygon of a humanoid robot. This is used to maximize the support polygon area of the robot, and potentially increasing the robot's reachability and the robot safety in terms of its balance. We use the proposed Jacobians and the VFIs framework to enable whole-body control with multi-contacts using a full humanoid robot in simulation.

The Euler-Lagrange model, which is used in conjunction with the SOVFIs for torque-actuated robots, is derived by means of the Gauss's Principle of Least Constraint using dual quaternion algebra. The use of dual quaternion algebra allows a more compact and unified representation for the twists and wrenches.

The proposed method is evaluated in a realistic simulation on a 27-DOF full humanoid robot and on three real platforms: a 9-DOF humanoid robot, a 8-DOF bimanual manipulator, and a 16-DOF nonholonomic bimanual manipulator. Results show that all constraints are respected while the robot performs a manipulation task.

Keywords: Vector Fields Inequalities, Humanoid Robots, Gauss's Principle of Least Constraint, Dual Quaternions, Quadratic Programming.

Contents

List of Figures

List of Tables

Acronyms

CoM	Center of Mass
CPQ	Constrained Quadratic Program
DQ	Dual Quaternion
DOF	Degrees of Freedom
FKM	Forward Kinematic Model
MACRO	Research Group on Mechatronics, Control and Robotics
PINV	Pseudoinverse
PRM	Probabilistic Road Map
QP	Quadratic Programming
ROS	Robot Operating System
UFMG	Universidade Federal de Minas Gerais
3D	Three-Dimensional
2D	Two-Dimensional
VREP	Virtual Robot Experimental Platform
VFIs	Vector Field Inequalities
SOVFIs	Second Order Vector Field Inequalities
SE(3)	Special Euclidean Group of dimension 3
RRT	Rapidly Exploring Random Tree

Notation

$\boldsymbol{a}$	Unconstrained acceleration.
$\boldsymbol{a}_c$	Constrained acceleration.
$\boldsymbol{c}$	Coordinate center of a circular obstacle.
$\mathcal{F}$	Coordinate system or frame.
$\mathbb{H}$	Set of quaternions.
$\mathbb{H}_p$	Set of pure quaternions.
$\mathbb{S}^3$	Set of unit quaternions.
$\mathcal{H}$	Set of dual quaternions.
$\underline{\boldsymbol{\mathcal{S}}}$	Set of unit dual quaternions.
$\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{y}$	Quaternions.
$\underline{\boldsymbol{h}}, \underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}$	Dual quaternions.
$\boldsymbol{h}^*$	Quaternion conjugate.
$\underline{\boldsymbol{h}}^*$	Dual quaternion conjugate.
$\overset{+}{\boldsymbol{H}}_4, \overset{-}{\boldsymbol{H}}_4$	Hamilton operators.
$\overset{+}{\boldsymbol{H}}_8, \overset{-}{\boldsymbol{H}}_8$	Hamilton operators extended for dual quaternions.
$\hat{\imath}, \hat{\jmath}, \hat{k}$	Quaternion units.
$\boldsymbol{I}$	Identity matrix.
$\boldsymbol{J}$	Jacobian matrix.
$\boldsymbol{J}^T$	Transpose of the Jacobian matrix.
$\boldsymbol{J}^+$	Pseudoinverse of the Jacobian matrix.

$\boldsymbol{J}_p$	Position Jacobian matrix.
$\boldsymbol{J}_{\mathcal{P}},\ \boldsymbol{J}_{\mathcal{D}}$	Jacobian matrices related with the primary and dual parts of $\boldsymbol{J}$ that satisfy $\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{\mathcal{P}}^T & \boldsymbol{J}_{\mathcal{D}}^T \end{bmatrix}^T$.
$\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,i}^i}$	Twist Jacobian.
$\boldsymbol{C}_4$	The conjugating matrix that satisfies $\operatorname{vec}_4 \boldsymbol{h}^* = \boldsymbol{C}_4 \operatorname{vec}_4 \boldsymbol{h}$
$\boldsymbol{n}$	Pure unit norm quaternion representing a rotation axis.
$\boldsymbol{p}_{ab}^a$	Pure quaternion representing the translation from frame $\mathcal{F}_a$ to $\mathcal{F}_b$ with respect to $\mathcal{F}_a$.
$\boldsymbol{q}$	Joints vector.
$\dot{\boldsymbol{q}}$	Joint velocities vector.
$\ddot{\boldsymbol{q}}$	Joint accelerations vector.
$\boldsymbol{\tau}_{\mathrm{GP}}$	Joint torques vector using the Gauss's Principle of Least Constraint.
$\boldsymbol{r}_b^a$	Unit quaternion representing the rotation from $\mathcal{F}_a$ to $\mathcal{F}_b$.
$\operatorname{vec}_4, \operatorname{vec}_8$	Operators representing a bijection mapping from $\mathbb{H}$ to $\mathbb{R}^4$ and $\mathcal{H}$ to $\mathbb{R}^8$ respectively.
$\|\underline{\boldsymbol{x}}\|$	Dual quaternion norm.
ε	Dual unit.
η	Scalar gain of the control law.
λ	Damping factor in the optimization problem.
ϕ	Rotation angle.
$\boldsymbol{W}$	Constraint matrix in the optimization problem.
$\boldsymbol{w}$	Constraint vector in the optimization problem.
$\boldsymbol{x}$	Task-space vector.
$\boldsymbol{x}_d$	Desired task-space vector.
$\tilde{\boldsymbol{x}}$	Task error.
$\boldsymbol{u}$	Control inputs.
k_d	Derivative gain of the control law.

k_p	Proportional gain of the control law.
$\boldsymbol{\gamma}_l$	Lower-bound of joint accelerations.
$\boldsymbol{\gamma}_u$	Upper-bound of joint accelerations.
k	Scalar gain used to scale the feasible region in the joint-limit acceleration constraints.
$\boldsymbol{g}$	Positive definite nondecreasing function.
$\mathbf{1}_n$	n-dimensional column vector of ones.
$\boldsymbol{M}$	Inertia matrix.
$\overline{\boldsymbol{n}}$	Nonlinear terms including Coriolis and gravity forces.
$d(t)$	Signed distance function.
$\tilde{d}(t)$	Distance error.
d_{safe}	Safe distance between the robot and an obstacle.
η_a	Scalar gain used to adjust the approach velocity.
η_d	Derivative gain used in the SOVFIs framework.
η_p	Proportional gain used in the SOVFIs framework.
$\dot{\tilde{d}}_0$	Initial error velocity.
$\tilde{d}_0$	Initial error distance.
$\underline{\boldsymbol{l}}_z$	Plücker line attached to one of the links in the robot kinematic chain.
$\underline{\boldsymbol{l}}$	Static Plücker line.
$\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}$	Angle between two Plücker lines.
$\boldsymbol{\Psi}$	Matrix that encapsulates the inertia parameters.

1 Introduction

In the last 20 years, the field of robotics has been experiencing great growth thanks to new technological advances in several branches of knowledge (Siciliano & Khatib, 2018). Now, robots are not restricted to industrial sectors only. New applications for robotics have emerged as military (Carlson & Murphy, 2005), educational (Chin et al., 2014) and medical fields (Taylor, 2006). Furthermore, we are witnessing robots performing tasks and interacting with humans in human environments (Cha et al., 2015). These new paradigms have brought new challenges and robots as humanoid robots, as shown in Fig. 1.1, some of them designed to assist and interact with humans in daily activities (Chen et al., 2013; Siciliano & Khatib, 2018).

Humanoid robots are robotic systems designed to perform human-like manipulation and locomotion tasks in a variety of scenarios, especially in cluttered ones such as human environments. Often, those robots have a large number of degrees of freedom, which increases the dexterity and possibilities of movements.

The motion generation of humanoid robots requires performing contacts with the environment sequentially under several constraints, which ensure, for instance, the robot balance, (self) collision avoidance, preventing of violation of joint limits, etc. The contacts can be cyclic or acyclic according to whether they are periodic or not. Cyclic contacts have been widely used in walking gaits, where in that specific case the contacts are composed of two phases: a single support phase and a double support phase. This strategy is considered a mature topic, with solutions that work in real time (Kajita et al., 2003; Baudouin et al., 2011; Farshidian et al., 2017). On the other hand, acyclic contacts allow a rich set of phases,

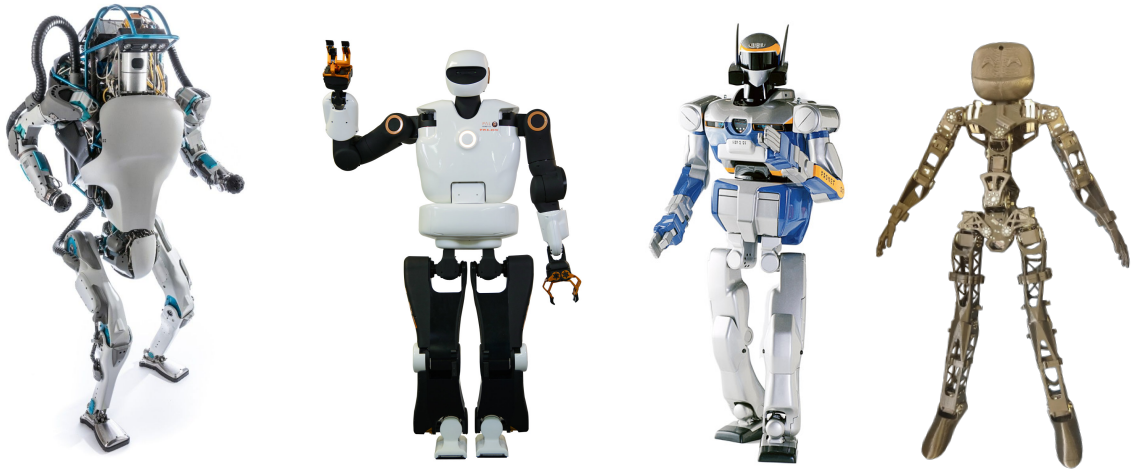

Figure 1.1: Humanoid robots. From *left* to *right*: Atlas from Boston Dynamics, Talos from Pal Robotics, HPR2 from Kawada Industries, and Poppeye robot from Poppy Project.

including multi-contacts with the environment and are a challenging topic of research. These are more suitable to perform tasks in cluttered environments, as there are more options of movement to face the difficulties of the terrain, as shown in Fig. 1.2.

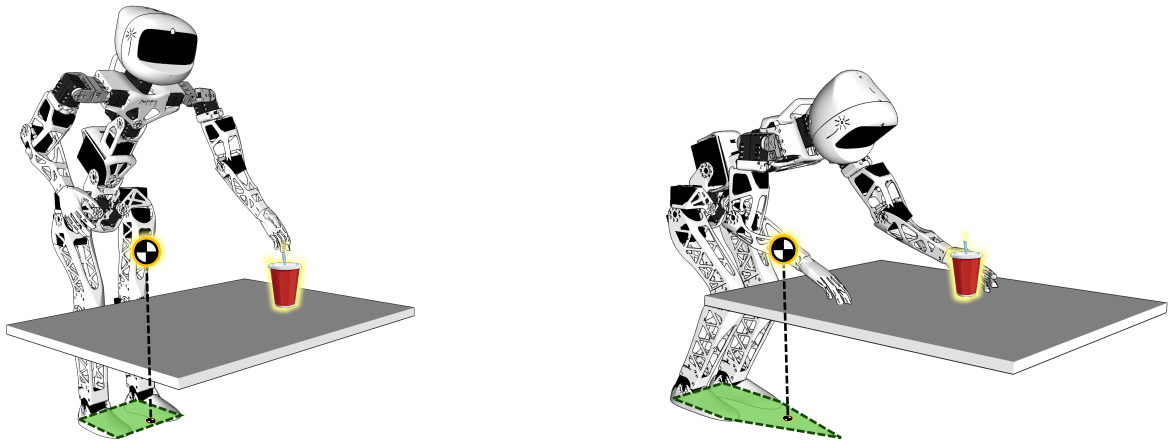

Figure 1.2: Multi-contact example. The goal is to grab a cup that is on the table. On the *left*, the robot reachability is limited by the CoM constraint, which maintains the CoM projection inside the support polygon. On the *right*, the robot reachability is extended by performing a new contact with the table, which changes the balance support polygon.

In order to generate acyclic motion, three problems must be solved simultaneously: computing the discrete contact sequence, the continuous contact locations and the continuous path between two contact combinations (Bouyarmane et al., 2017; Tonneau et al., 2018). However, dealing with those problems at the same time can lead to a combinatorial explosion. Usually, this issue has been addressed separately in two ways: using local optimization and motion planning. The former has been used to trade the computation cost at the expense of local convergence, whereas the latter aims the global convergence at

the expense of prohibitive costs (Deits & Tedrake, 2014; Tonneau et al., 2018).

Although remarkable works have been done using discrete searches (Hauser et al., 2008; Tonneau et al., 2018), local optimization have proved to be a promising road (Mordatch et al., 2012; Dai & Tedrake, 2016), allowing smooth and optimal local solutions, with no need for post-processing computed trajectories (Bouyarmane et al., 2009).

Local optimization strategies can be formulated explicitly by using mathematical programming, which allows dealing with inequality (i.e., unilateral) constraints directly in the optimization formulation, providing an efficient and elegant solution, where all constraints are clearly separated from the main task (Bouyarmane et al., 2019). Analytical solutions, however, usually do not exist and numeric solvers must be used (Kim & Oh, 2013; Escande et al., 2014a; Goncalves et al., 2016).

Marinho et al., 2018 propose active constraints based on Vector Fields Inequalities (VFIs) to deal with collision avoidance in surgical applications. Both robot and obstacles are modeled by using geometric primitives—such as points, planes, and Plücker lines—, and distance functions with their respective Jacobian matrices are computed from these geometric primitives. The advantage of VFIs is that they limit the robot velocities only in the direction towards the collision. This strategy can be extended to second order kinematics (SOVFIs), which enables applications that use the robot dynamics, a more suitable model to multi-contact framework (Sentis, 2007).

Furthermore, inequality constraints can be used to relax tasks. The main idea is to describe the task by target regions instead of specific points. This decreases the number of degrees of freedom (DOF) necessary to perform the task. In this way, by releasing some robot DOF, secondary tasks can be performed.

This work focuses on whole-body control at the task space level using first and second order kinematics under VFIs and SOVFIs respectively. This approach allows to relax specific tasks, to define manipulation tasks and multi-contact applications, and to impose constraints in order to prevent collisions and self collisions, or prevent violation of joints limits. Furthermore, this works develops strategies for robot dynamic modeling using dual quaternion algebra. This allows the use of the SOVFIs framework in robots commanded by joint torques, as shown in Fig. 1.3.

1.1 Objective and Contributions

The main objective of this work is to develop whole-body motion control strategies for humanoid robots with (self) collision avoidance and multi-contact constraints in order to extend the robot manipulation capabilities. The strategies must be efficient enough to be implemented on real robotic systems. The specific objectives are:

1. Define a suitable objective function in order to perform whole-body control at second

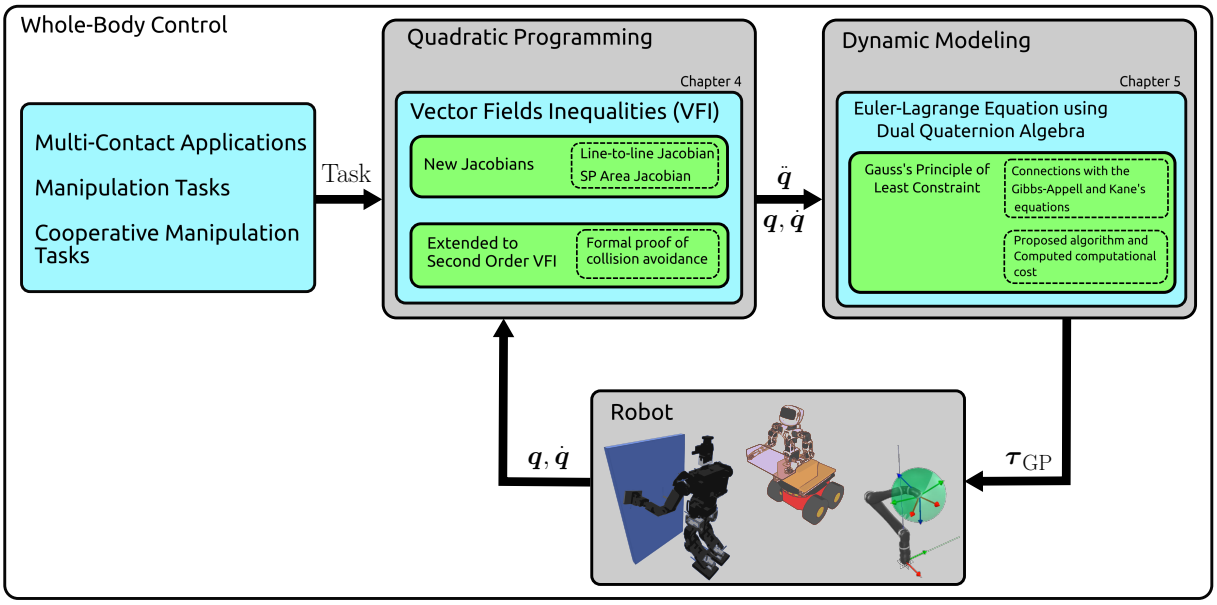

Figure 1.3: Whole-Body control strategy used for manipulation tasks, cooperative manipulation tasks and multi-contact applications. The blue boxes denote the explored topics, whereas the green ones denote the contributions of this thesis.

order kinematics and validate the proposed techniques on applications that use the robot dynamics.

2. Develop suitable constraints in order to ensure collision and self-collision avoidance.

3. Allow task flexibilization to release DOF, which can be used to fulfill secondary tasks.

4. Develop suitable constraints to perform multi-contact applications and general manipulation tasks.

5. Exploit the VFIs framework to impose constraints that ensure balance tasks and contact forces with other robot body parts, namely, the hands.

6. Validate the proposed techniques on simulation and on a real platform available in our research group MACRO.[1]

The contributions of this work can be summarized as follows:

- We extends the VFIs method, which was first proposed using first order kinematics (Marinho et al., 2018), to use second order kinematics, SOVFIs, which is introduced in Chapter 4. This enables applications that use the robot dynamics by means of the relationship between joint torques and joint accelerations in the Euler-Lagrange equations.

[1] http://macro.ppgee.ufmg.br/

- We propose a new distance function related to the angle between two Plücker lines and the corresponding Jacobian matrix to prevent violations of joint limits and avoid undesired end-effector orientations. These are introduced in Chapter 4.

- We present a formal proof ensuring collision avoidance for SOVFIs. The proof is introduced in Chapter 4.

- We propose a new Jacobian related with the support polygon of a humanoid robot. This enables tasks as maximization of the support polygon area, which can potentially increase the robot's reachability and the robot safety in terms of its balance.

- We present the formulation of the Gauss's Principle of Least Constraint using dual quaternion algebra, in Section 5. This strategy allows taking into account additional constraints in the accelerations, which can be exploited, for instance, in nonholonomic robotic systems. In addition, the computational cost in terms of number of multiplications, additions, and trigonometric operations is presented and compared with their classic counterparts. Furthermore, we show the connections between Gauss's principle, Gibbs-Appell equations, and Kane's method.

1.2 Publications

Parts of this dissertation have been published or submitted in the following works:

Journals:

- **Quiroz-Omaña, J. J.**; Adorno, B. V. Whole-Body Control With (Self) Collision Avoidance Using Vector Field Inequalities. *IEEE Robotics and Automation Letters* (RA-L), vol. 4, no. 4, pp. 4048–4053, oct 2019.

- F.F.A.; **Quiroz-Omaña, J. J.**; Adorno, B. V. Dynamics of Mobile Manipulators using Dual Quaternion Algebra. (submitted to Journal of Mechanisms and Robotics (ASME-TMR)).

Workshops:

- **Quiroz-Omaña, J. J.**; Adorno, B. V. Bimanual Mobile Manipulation Using the Cooperative Dual Task-Space Framework and Vector Fields Inequalities. *In Workshop on Applications of Dual Quaternion Algebra to Robotics*, International Conference on Advanced Robotics (ICAR) 2019, in Belo Horizonte, Brazil on December 2019.

1.3 Structure of the Text

This dissertation is organized as follows, Chapter 2 presents some of the most relevant works in acyclic motion with multi-contacts for humanoids robots and (self) collision avoidance. Chapter 3 briefly reviews the mathematical foundation required to understand the presented methods, and establishes the notation used in this work. Chapter 4 introduces one contribution of this dissertation, the Second Order Vector Fields Inequalities (SOVFIs), its formal proof for collision avoidance, and presents a new distance function and its related Jacobian. Furthermore, that chapter presents a new Jacobian related with the support polygon of a humanoid robot. Chapter 5 introduces another contribution of this work, the Gauss's Principle of Least Constraint in dual quaternion algebra, and its connections with the Gibbs-Appell equations and Kane's method. Chapter 6 presents the simulation and experimental results. Chapter 7 presents the conclusions and future works. Finally Appendix A reviews the dual quaternion algebra required to understand the Gauss's Principle of Least Constraint derivation.

2

State of the Art

This chapter reviews some recent works related to the research on humanoid robots and is organized as follows: Section 2.1 presents a review of control strategies for multi-contact control of humanoid robots based on discrete searches. Section 2.2 reviews multi-contact control strategies based on local optimization.

The locomotion principle of a humanoid robot is based on performing contacts sequentially. The first works on humanoid robots focused on cyclic walking gaits, where only two phases compose the contacts: the single support phase and the double support phase. However, because this approach does not consider multiple contacts, it can limit the robot dexterity and its reachability, especially in cluttered scenarios, since multiple contacts with other links are not taken into account. For those cases, where multi-contacts are required, acyclic contacts have been used.

2.1 Multi-Contact Control of Humanoid Robots: Discrete Searches

Early works using acyclic contacts were based on motion graphs (Kovar et al., 2002; Pettré et al., 2003). This strategy requires motion capture data, where a graph encodes how the captured frames could be assembled in different ways. Because the movements are highly data-dependent, this approach does not allow adaptations to new scenarios or new movements directly.

The acyclic contact planning paradigm requires computing the robot base[1] trajectory in SE(3), planning a sequence of configurations that respect a set of constraints along the base trajectory, and interpolating a continuous motion between two configurations (Tonneau et al., 2018; Bouyarmane et al., 2017). This huge amount of choices is a combinatorial problem, and often it has been addressed in two ways: based on local optimization or discrete searches that decouple the problems to reduce the complexity (Deits & Tedrake, 2014; Tonneau et al., 2018).

Roughly speaking, discrete searches strategies are based on motion planning that use probabilistic algorithms. In the context of multi-contact generation, the contacts are performed in the boundary between the free space and the obstacle space, see Fig. 2.1. Here, naive probabilistic methods do not work, since the subspace of the configuration space is zero measure (Escande et al., 2013). That is, the probability of sampling a configuration in the boundary is zero. In those cases, the sampled configurations must be projected in contact with the obstacles, increasing the numerical cost of the operation (Tonneau et al., 2018).

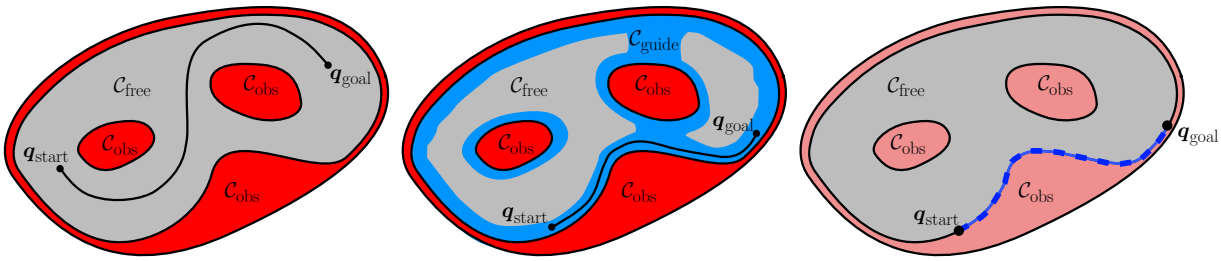

Figure 2.1: Motion planning: On the *left*, collision-free motion planning. On the *middle*, Contacts guide planning. On the *right*, Contact-points planning. (Bouyarmane et al., 2009)

Bretl et al., 2005 proposed an acyclic motion algorithm for free-climbing robots based on probabilistic road maps (PRM) using the contact-before-motion approach. The algorithm works with specific climbing scenarios using a pre-specified steps sequence but is not applicable to general ones. Hauser et al., 2005 extended that strategy to a humanoid robot. A set of contacts is defined as a stance. Stances are sampled and stored in a stance-adjacency graph. The set of stance sequences are searched in the stance-adjacency graph. Two stances are connected if they differ by one contact and contain a feasible configuration, as shown Fig. 2.2. Incremental improvements were performed in the graph search process aiming for smoother trajectories. For instance, using potential functions or guide trajectories to avoid complicated paths and postures (Escande et al., 2006, 2009, 2008; Bouyarmane et al., 2009). Hauser et al. (2008) used a library of motion primitives to obtain smoother and more natural motions at the expense of loss of generality, preventing the discovery of new possible motions.

[1]The base or root of the robot is often the pelvis. It is used to define the localization of the robot with respect to a reference coordinate system.

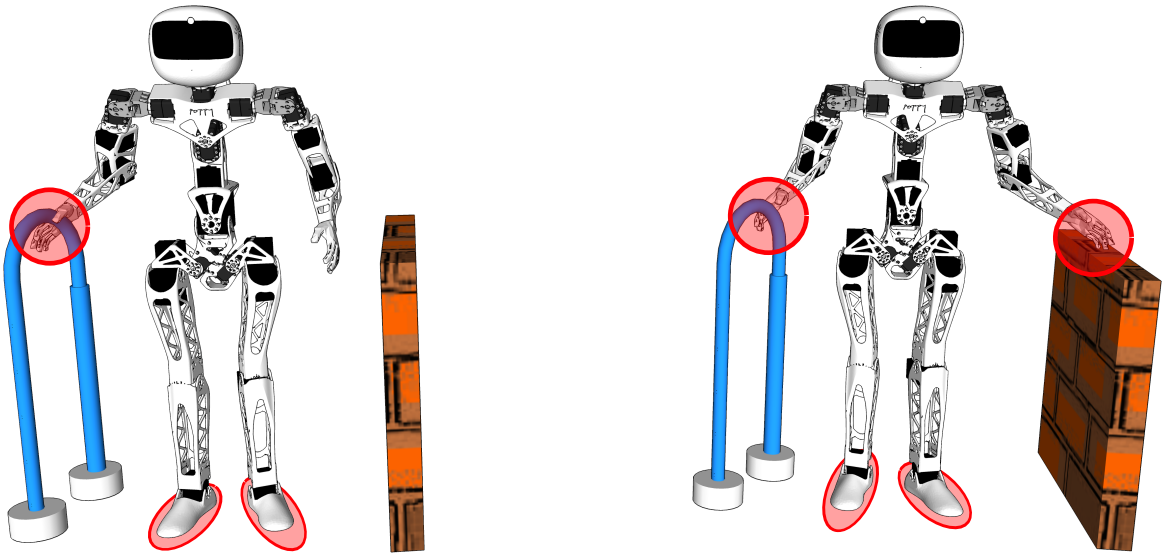

Figure 2.2: Examples of two stances: On the *left*, the robot performs three contacts. On the *right*, the robot performs four contacts. The stances differ by one contact between them.

Tonneau et al. (2018) proposed an acyclic planner for legged robots with remarkable computational efficiency. The strategy is composed of two stages and is based on the the contact reachability. First, a trajectory of the robot base is computed in SE(3) using a PRM. The contact surface must be included in the reachability of the robot to allow a contact creation while the robot base is free of collision. Second, the algorithm computes a sequence of statically balanced and collision-free configurations along of the path of the robot base using an offline database of the robot limbs configurations, which the algorithm chooses based on a heuristic. However, the planning success rate is highly-dependent of the environment.

Table (2.1) summarizes the main approaches for multi-contact control using discrete searches. In this dissertation, the multi-contact control will be addressed using local optimizations, aiming for lower computational times with respect to discrete searches, but at expense of local convergence.

2.2 Multi-Contact Control of Humanoid Robots: Local Optimization

Other strategies for acyclic motion generation have been approached using local optimization. Mordatch et al. (2012) proposed the Contact-Invariant Optimization (CIO) method to perform simultaneous optimization of contacts and behavior. The objective function is composed of four cost functions related to the dynamic model, the desired task, optional hints costs and the contact-invariant cost. This latter affects not only the cost

Table 2.1: Main approaches for multi-contact control using discrete searches.

Work	Strategy	Drawbacks
Kovar et al. (2002)		It requires motion capture data.
Pettré et al. (2003)	Motion graphs	Highly data-dependent.
		No adaptations, no new motions.
Bretl et al., 2005	Probabilistic Road Maps	
Hauser et al., 2005		
Escande et al. (2006)		Computationally expensive.
Escande et al. (2009)	Contact-before-motion	
Bouyarmane et al. (2009)		
Hauser et al. (2008)		
Tonneau et al., 2018	RRT	Requires user-defined parameters
	Contact reachability	Requires highly-constrained scenarios.

function but also the dynamics by enabling and disabling contact forces. The optimization process is composed of three phases. In phase one, only the cost function related to the task is enabled, which allows the rapid discovery of a movement without being physically consistent. In phase two, all cost functions are enabled but down-weighting the one related with the dynamic model. This allows to obtain a rough physical realism while guided by hints. Finally, in phase 3, all cost functions are enabled except the one related with the hints. This is used to refine the final solution. The authors use a reduced model aiming at better performance. However the method is far from real time, requiring between two to ten minutes on each of the three phases in simulation tests. This strategy was implemented on a real a robot by Mordatch et al. (2015) using off-line strategies.

A similar approach was proposed by Al Borno et al. (2013) to handle highly dynamic motions with cyclic and acyclic movements using spacetime constraints (Witkin & Kass, 1988). As in the work of Mordatch et al. (2012), the computational cost is still prohibitive.

Lengagne et al. (2013) addressed the dynamic multi-contact motion generation by using nonlinear optimization and considering the full-body dynamic model. The authors used a B-spline parameterization for the joints and the optimization formulation was written as a semi-infinite program. Unlike Mordatch's work, the method requires the desired contact sequence. Still, the proposed method is computational expensive. For instance, the time required to perform a sitting motion task is about 3 hours.

Saab et al. (2013) used a hierarchical quadratic program (HQP) to handle dynamic motion under inequality constraints. The authors proposed a reduced formulation for rigid planar contacts. Experiments showed a successful and efficient implementation on a real robot in a multi-contact task. However, the solver requires a predefined set of contacts.

Shu-Yun Chung & Khatib (2015) addressed the multi-contact locomotion using a decoupled approach using the elastic strips framework (Brock & Khatib, 2002). The contact regions are extracted from the environment and represented as a 3D point cloud. A global planner searches a sequence of contact-regions based on the reachability. After that,

a prioritized controller simultaneously adjusts the postures and contact regions. However, collisions are not taken into account in the planning phase and therefore, replanning is required in the case of failure. Simulation results have shown that the proposed method finds the contact sequence and the corresponding motions in a few seconds.

Murooka et al. (2020a) addressed the multi-contact generation using quadratic programming to simultaneously compute the control inputs and the points of contacts on the body surface. The authors modeled the robot using a convex polyhedron based on the robot mesh. To prevent discontinuities in the search of points on the edge of the polyhedron, the authors proposed a smoothed normal direction on the body surface. Different from previous approaches (Escande et al., 2014b, 2016), the strategy does not require additional computations to approximate the robot model.

Deits & Tedrake (2014) addressed the contact planning using a mixed-integer problem (MIP). A set of convex obstacle-free configuration space regions are precomputed and integer variables are used to assign footstep to those regions. The method is efficient but only cyclic gaits are handled and the robot dynamic model is not considered. Ponton et al. (2016), extended the work of Deits & Tedrake (2014) allowing multi-contacts. Instead of using a full dynamic model, the authors proposed a convex model, namely the centroidal momentum dynamics (CMD) (Dai et al., 2014) aiming at computational efficiency. Ponton et al. (2018) proposed convex relaxations of the CMD to enable specifications about desired angular momentum in the objective function. That work was then extended (Ponton et al., 2021) to include the optimization of timing by using a sequence of convex approximations of the centroidal dynamics. Although strategies based on MIP have shown the potential for contact planning, it is still computationally expensive. To mitigate that limitation, Tonneau et al. (2020) formulated the MIP contact planning by means of linear programming. This allows taking advantage of the sparse solutions at expense of optimality approximations. Results showed faster contact planning for cases involving both a small number of contacts and contact surfaces.

Table (2.2) summarizes the main approaches for multi-contact control using local optimization. In this dissertation, the multi-contact control will be addressed as proposed by Saab et al. (2013) but using sets of candidate contact regions instead of a set of contacts and exploiting both unilateral and bilateral constraints.

2.3 (Self) Collision Avoidance

Collision avoidance has been addressed either in off-line or on-line approaches. The former is based on motion planning, where probabilistic methods have been widely used (Karaman et al., 2011; Burget et al., 2016). These strategies are usually applied in the configuration space and are computationally expensive, free of local minima and used in known scenarios (Moll et al., 2015). The latter is based on reactive methods and usually

Table 2.2: Main approaches for multi-contact control using local optimization.

Work	Strategy	Drawbacks
Mordatch et al., 2012 Al Borno et al. (2013) Mordatch et al. (2015)	Contact-Invariant Optimization	Reduced Dynamic Model. Computationally expensive.
Saab et al. (2013)	Hierarchical quadratic program	Requires the a predefined set of contacts.
Shu-Yun Chung & Khatib (2015)	Elastic strips	Collisions are not handled in the planning phase.
Murooka et al. (2020b)	Quadratic programming	Robot dynamics not considered.
Deits & Tedrake (2014) Ponton et al. (2016, 2018, 2021)	Mixed-integer problem	Cyclic gaits only. Robot dynamics not considered. Reduced model.
Tonneau et al. (2020)	Linear Programming	Optimality approximations.

require less computation time; therefore, they can be used in real-time feedback control and are suitable for applications within unknown workspaces.

Reactive methods are, in general, based on minimization problems (Laumond et al., 2015), which exploit the robot redundancy by selecting admissible control inputs based on a specified criterion. When the robot is commanded by joint velocity inputs and operates under relatively low velocities and accelerations, its behavior is appropriately described by the kinematic model and, as a consequence, the minimization can be performed in the joint velocities. In that case, since the control law is based entirely on the kinematic model, it is not affected by uncertainties in the inertial parameters (e.g., mass and moment of inertia). On the other hand, if the robot is commanded by torque inputs, the minimization is performed in the joint torques, which usually requires the robot dynamic model. Both methods are widely used to perform whole-body control with reactive behavior.

Whole-body control strategies with collision avoidance usually have been handled by using the task-priority framework, where the overall task is divided into subtasks with different priorities. For instance, distance functions with continuous gradients between convex hulls (or between simple geometrical primitives such as spheres and cylinders) that represent the body parts are used and the lower-priority collision-avoidance tasks are satisfied in the null space of higher-priority ones (Stasse et al., 2008; Schwienbacher et al., 2011). Those secondary tasks are fulfilled as long as they are not in conflict with the higher-priority ones. Therefore, they do not prevent collisions when in conflict with the primary task. One way to circumvent this problem is to place the collision avoidance as the higher priority task (Sentis & Khatib, 2004), at the expense of not guaranteeing the fulfillment of the main task, such as reaching targets with the end-effector. Some authors address this by using a dynamic task prioritization, where the control law is blended continuously between the collision-avoidance task and the end-effector pose control task as a function of the collision distance (Sugiura et al., 2007). Dietrich et al. (2012) propose a torque-based self-collision avoidance also using dynamic prioritization, where the

transitions are designed to be continuous and comply with the robot's physical constraints, such as the limits on joint torque derivatives. However, changing priorities to enforce inequality constraints has an exponential cost in the number of inequalities (Kanoun et al., 2011). Alternatively, non-hierarchical formulations based on weighted least-square solutions are used to solve the problem of constrained closed-loop kinematics in the context of self-collision avoidance (Patel et al., 2005). However, the non-hieararchical approach requires an appropriate weighting matrix that results in a collision-free motion, which may not work in general, specially for high-speed motions (Dariush et al., 2010). That can be solved by combining the weighting matrix with the virtual surface method, which redirects the collision points along a virtual surface surrounding the robot links at the expense of potentially disturbing the trajectory tracking when the distance between collidable parts is smaller than a critical value.

Other strategies addressed the collision-free motion generation problem explicitly by using mathematical programming, which allows dealing with inequality (i.e., unilateral) constraints directly in the optimization formulation (Decre et al., 2009; Quiroz-Omaña & Adorno, 2018; Marinho et al., 2018, 2019; Bouyarmane et al., 2019).

Faverjon & Tournassoud (1987) used inequalities constraints to handle collision avoidance, and Bouyarmane et al. (2017) extended it to second order kinematics. The distance between the robot and the obstacle is computed by using an algorithm that models the robot by means of convex shapes.

Marinho et al. (2018) proposed active constraints based on VFIs to deal with collision avoidance in surgical applications. Both robot and obstacles are modeled by using geometric primitives—such as points, planes, and Plücker lines—, and distance functions with their respective Jacobian matrices are computed from these geometric primitives. The advantage of VFIs is that they limit the robot velocities only in the direction towards the collision.

Koptev et al. (2021) addressed the real-time self-collision avoidance of a humanoid robot by learning feasible regions and using a quadratic program to generate collision-free motions. The idea is to collect an offline dataset by sampling the robot workspace, for collision-free and non collision free joint space configurations, using a precise triangle mesh representation of the robot. The authors used machine learning to obtain smooth boundary functions, which are used as inequality constraints in quadratic programming to prevent robot self-collisions. This strategy is less conservative than the ones that model the robot using convex geometrical approximations. However, collisions with the environment or other objects are neglected.

Marinho et al. (2019) extended the VFIs framework to prevent collisions between moving entities by means of dynamic active constraints. This strategy enables applications with any number of robots sharing the workspace or applications with dynamic objects as long as their velocities are available. However, the proposed VFIs strategy uses first order kinematics only. This limits applications that use the robot dynamics.

Osorio et al. (2020) worked with collision-avoidance unilateral constraints and second order kinematics to perform torque control. However, the geometric primitives used are limited to points only. This can be restrictive since point-based models can increase the number of constraints in some scenarios.

Table (2.3) summarizes the main approaches for collision avoidance based on reactive methods. The strategies based on VFIs and mathematical programming are promising but second order kinematics is not taken into account. In this dissertation, VFIs with second order kinematics are taken into account in conjunction with mathematical programming.

Table 2.3: Main approaches for collision avoidance using inequality constraints.

Work	Strategy	Drawbacks
Stasse et al. (2008) Schwienbacher et al. (2011)	The task priority framework.	Collision-avoidance tasks have lower priorities. There is no guarantee of collision avoidance.
Sentis & Khatib (2004)		Collision-avoidance tasks have higher priorities. No guarantee of the fulfillment of the main task.
Sugiura et al. (2007) Dietrich et al. (2012)	Dynamic prioritization.	Exponential cost in the number of inequalities.
Patel et al. (2005) Dariush et al. (2010)	Weighted least-square solutions.	It may not work in general.
Quiroz-Omaña & Adorno (2018)	Mathematical programming.	
Marinho et al. (2018)	Vector Fields Inequalities (VFIs).	It uses first order kinematics only.
Marinho et al. (2019)	Dynamic VFIs	
Koptev et al. (2021)	Learned feasible regions to compute smooth boundary functions.	Only self-collision avoidance. It uses first order kinematics only.
Osorio et al. (2020)	Unilateral constraints based on point-based primitives.	Limited number of geometric primitives.

2.4 Conclusions

This chapter presented some important works related to multi-contact control of humanoid robots and self-collision and collision avoidance. Section 2.1 reviewed the strategies based on discrete searches for acyclic contact planning. In this paradigm, three problems must be solved simultaneously: computing the sequence of discrete contacts, the continuous contacts locations, and the continuous path between two contacts combinations (Tonneau et al., 2018; Bouyarmane et al., 2017). Since dealing with those problems at the same time can lead to a combinatorial explosion, the problem has been addressed separately in two ways: motion planning and using local optimization. The former is based on probabilistic algorithms and aims for the global convergence at the expense of prohibitive costs. The latter usually has a lower computational cost at the expense of local convergence (Deits &

Tedrake, 2014; Tonneau et al., 2018) and is presented in Section. 2.2. Finally, Section. 2.3 presented some important works about strategies for (self) collision avoidance. This is addressed either using off-line (motion planning) or on-line approaches (reactive methods). Reactive methods are, in general, based on minimization problems (Laumond et al., 2015), which exploit the robot redundancy by selecting admissible control inputs based on a specified criterion. Furthermore, they usually require less computation time than off-line approaches; therefore, they can be used in real-time feedback control and are suitable for applications within unknown workspaces.

3

Mathematical Background

This chapter reviews some concepts, foundations and operations related to task-space control and vector field inequalities.

3.1 Task-Space Control Using Mathematical Programming

A classic strategy used in differential inverse kinematic problems consists in solving an optimization problem that minimizes the joint velocities, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, in the l_2-norm sense. Given a desired task $\boldsymbol{x}_d \in \mathbb{R}^m$, where $\dot{\boldsymbol{x}}_d = \boldsymbol{0}$, $\forall t$, and the task error $\tilde{\boldsymbol{x}} \triangleq \boldsymbol{x} - \boldsymbol{x}_d$, the control input $\boldsymbol{u}$ is obtained as

$$\begin{aligned} \boldsymbol{u} \in \arg\min_{\dot{\boldsymbol{q}}} \quad & \left\| \boldsymbol{J}\dot{\boldsymbol{q}} + \eta\tilde{\boldsymbol{x}} \right\|_2^2 + \lambda^2 \left\| \dot{\boldsymbol{q}} \right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{W}\dot{\boldsymbol{q}} \leq \boldsymbol{w}, \end{aligned} \tag{3.1}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the task Jacobian that satisfies $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$, $\lambda \in [0, \infty)$ is a damping factor, and $\boldsymbol{W} \in \mathbb{R}^{l \times n}$ and $\boldsymbol{w} \in \mathbb{R}^l$ are used to impose linear constraints in the control inputs (Marinho et al., 2019). Furthermore, $\eta \in (0, \infty)$ denotes the convergence rate and is selected in order to obtain a fast and smooth convergence.

An analogous scheme can be used to perform the minimization at the joint acceleration level. Given the desired error dynamics $\ddot{\tilde{\boldsymbol{x}}}_d = -k_d\dot{\tilde{\boldsymbol{x}}} - k_p\tilde{\boldsymbol{x}}$, with $k_d, k_p \in (0, \infty)$ such that $k_d^2 - 4k_p > 0$ to obtain a non-oscillatory exponential error decay, the minimization problem

in the constrained case is formulated as

$$\begin{aligned} \boldsymbol{u}_a \in \arg\min_{\ddot{\boldsymbol{q}}} \quad & \left\| \underbrace{\boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}}_{\ddot{\tilde{\boldsymbol{x}}}} - \underbrace{(-k_p\tilde{\boldsymbol{x}} - k_d\boldsymbol{J}\dot{\boldsymbol{q}})}_{\ddot{\boldsymbol{x}}_d} \right\|_2^2 + \lambda^2 \left\|\ddot{\boldsymbol{q}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{\Lambda}\ddot{\boldsymbol{q}} \leq \boldsymbol{\rho}, \end{aligned} \tag{3.2}$$

where $\boldsymbol{\Lambda} \triangleq \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} & \boldsymbol{W}^T \end{bmatrix}^T \in \mathbb{R}^{(2n+l)\times n}$ and $\boldsymbol{\rho} \triangleq \begin{bmatrix} \boldsymbol{\gamma}_u^T, -\boldsymbol{\gamma}_l^T, \boldsymbol{w}^T \end{bmatrix}^T \in \mathbb{R}^{2n+l}$.

By regrouping the terms, we can write

$$\begin{aligned} \boldsymbol{u}_a \in \arg\min_{\ddot{\boldsymbol{q}}} \quad & \left\| \boldsymbol{J}\ddot{\boldsymbol{q}} + \underbrace{\dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + k_p\tilde{\boldsymbol{x}} + k_d\boldsymbol{J}\dot{\boldsymbol{q}}}_{\beta} \right\|_2^2 + \lambda^2 \left\|\ddot{\boldsymbol{q}}\right\|_2^2, \\ \text{subject to} \quad & \boldsymbol{\Lambda}\ddot{\boldsymbol{q}} \leq \boldsymbol{\rho}. \end{aligned} \tag{3.3}$$

Analogously to (3.1), $\boldsymbol{W}$ and $\boldsymbol{w}$ are used to impose arbitrary linear constraints in the acceleration inputs, and two additional constraints, that is $\boldsymbol{\gamma}_l \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}_u$, are imposed to minimize the joint velocities by limiting the joint accelerations. The idea is that those bounds depend on the joint velocities, as shown in Fig. 3.1. More specifically, we define the acceleration lower bound $\boldsymbol{\gamma}_l$ and the acceleration upper bound $\boldsymbol{\gamma}_u$, respectively:

$$\boldsymbol{\gamma}_l \triangleq k\left(-\mathbf{1}_n g\left(\dot{\tilde{\boldsymbol{x}}}\right) - \dot{\boldsymbol{q}}\right), \qquad \boldsymbol{\gamma}_u \triangleq k\left(\mathbf{1}_n g\left(\dot{\tilde{\boldsymbol{x}}}\right) - \dot{\boldsymbol{q}}\right), \tag{3.4}$$

where $k \in [0, \infty)$ is used to scale the feasible region, $g : \mathbb{R}^m \to [0, \infty)$ is a positive definite nondecreasing function (e.g., $g\left(\dot{\tilde{\boldsymbol{x}}}\right) \triangleq \left\|\dot{\tilde{\boldsymbol{x}}}\right\|_2$) and $\mathbf{1}_n$ is an n-dimensional column vector composed of ones. As the bounds (3.4) depend on the error velocity $\dot{\tilde{\boldsymbol{x}}}$, then $\boldsymbol{\gamma}_l \to \boldsymbol{\gamma}_u$ when $\dot{\tilde{\boldsymbol{x}}} \to \mathbf{0}$. Therefore, $\boldsymbol{\gamma}_l = \boldsymbol{\gamma}_u = \boldsymbol{\gamma}$ and $\boldsymbol{\gamma} \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}$ becomes $\ddot{\boldsymbol{q}} = \boldsymbol{\gamma} = -k\dot{\boldsymbol{q}}$, whose solution is given by $\dot{\boldsymbol{q}}(t) = \dot{\boldsymbol{q}}(0)\exp(-kt)$; that is, as the task velocity goes to zero, the robot stops accordingly.[1]

[1] Those bounds are necessary because (3.3) minimizes the joint *accelerations*. Without them, the objective function can be minimized even if the joint velocities are not null.

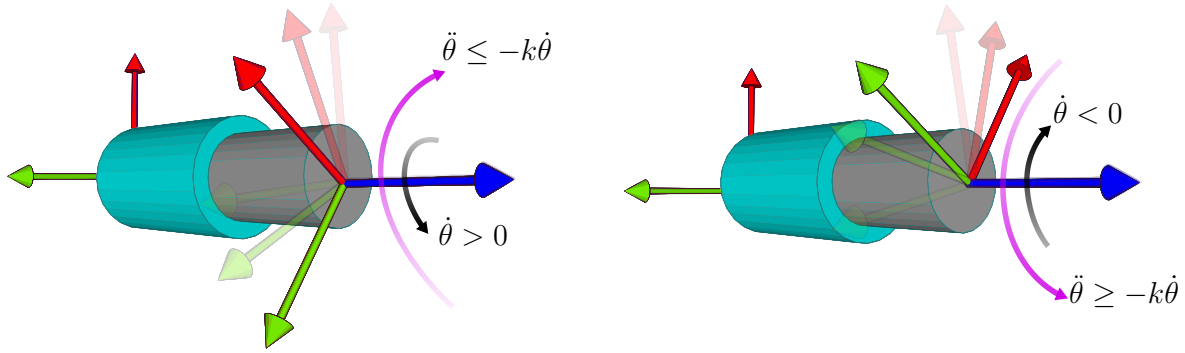

Figure 3.1: Joint acceleration bounds behavior when the task is fulfilled (i.e., $g\left(\dot{\tilde{\boldsymbol{x}}}\right) = 0$). On the *left*, the case when there is a positive angular velocity $\dot{\theta} > 0$, the constraint enforces a negative acceleration $\ddot{\theta} \leq -k\dot{\theta}$ that decrease the velocity $\dot{\theta}$ to zero. On the *right*, the analogous case for $\dot{\theta} < 0$, where the constraint enforces $\ddot{\theta} \geq -k\dot{\theta}$.

In some cases, the constraints $\boldsymbol{W}\ddot{\boldsymbol{q}} \leq \boldsymbol{w}$ and $\boldsymbol{\gamma}_l \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}_u$ may become conflicting and therefore (3.3) could be infeasible. A classic strategy to address this problem is to exploit the robot redundancy using the task priority framework (Kanoun et al., 2011). The idea, if feasible, is to minimize the joints velocities at the lower priority level while ensuring the execution of the higher priority task. The higher priority level control law is formulated as

$$\begin{aligned} \boldsymbol{u}_{a_1} \in \arg\min_{\ddot{\boldsymbol{q}}} \quad & \left\|\boldsymbol{J}\ddot{\boldsymbol{q}} + \boldsymbol{\beta}\right\|_2^2 + \lambda^2 \left\|\ddot{\boldsymbol{q}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{W}\ddot{\boldsymbol{q}} \leq \boldsymbol{w}. \end{aligned} \tag{3.5}$$

The lower priority level control law is formulated as

$$\begin{aligned} \boldsymbol{u}_{a_2} \in \operatorname{argmin} \quad & \left\|\ddot{\boldsymbol{q}} + \alpha\dot{\boldsymbol{q}}\right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{J}\ddot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{u}_{a_1} \\ & \boldsymbol{W}\ddot{\boldsymbol{q}} \leq \boldsymbol{w}, \end{aligned} \tag{3.6}$$

where $\alpha \in (0, \infty)$, and the equality constraint $\boldsymbol{J}\ddot{\boldsymbol{q}} + \boldsymbol{\beta} = \boldsymbol{J}\boldsymbol{u}_{a_1} + \boldsymbol{\beta} \implies \boldsymbol{J}\ddot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{u}_{a_1}$ ensures the error dynamics imposed in 3.5.

If the robot is commanded by means of torque inputs (i.e., $\boldsymbol{\tau} \triangleq \boldsymbol{u}_\tau$), we use (3.3) and the Euler-Langrange equation $\boldsymbol{M}\ddot{\boldsymbol{q}} + \overline{\boldsymbol{n}} = \boldsymbol{\tau}$, where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is the inertia matrix and $\overline{\boldsymbol{n}} \in \mathbb{R}^n$ denotes the nonlinear terms including Coriolis and gravity forces, to compute the control input

$$\boldsymbol{u}_\tau = \overline{\boldsymbol{n}} + \boldsymbol{M}\boldsymbol{a}, \tag{3.7}$$

where $\boldsymbol{a} \in \{\boldsymbol{u}_{a_1}, \boldsymbol{u}_{a_2}\}$ are the joint accelerations computed by (3.3) or using the hierarchical framework given by (3.5) and (3.6).

3.2 Distance Functions and Jacobians

The VFIs are differential inequalities that are used to avoid collisions between pairs of geometrical primitives (Faverjon & Tournassoud, 1987; Marinho et al., 2019). Each VFI requires a distance function between two geometric primitives and both the Jacobian and residual, which relate the robot joint velocities to the time-derivative of the distance function as follows

$$\dot{d} = \boldsymbol{J}\dot{\boldsymbol{q}} + \zeta\left(t\right).$$

The geometric primitives can be represented by dual quaternions (Adorno, 2017) (see appendix A), as shown in Fig. 3.2. For instance, points $\underline{\boldsymbol{\mathcal{S}}} \ni \underline{\boldsymbol{p}} = 1 + \varepsilon\frac{1}{2}\boldsymbol{p}$ are described by their Cartesian coordinates $\boldsymbol{p} \in \mathbb{H}_p$. A plane $\underline{\boldsymbol{\mathcal{S}}} \ni \underline{\boldsymbol{\pi}} = \boldsymbol{n}_\pi + \varepsilon d_\pi$ is described by the unit norm vector normal to the plane $\boldsymbol{n}_\pi \in \mathbb{H}_p \cap \mathbb{S}^3$ and the perpendicular distance $d_\pi = \langle \boldsymbol{p}_\pi, \boldsymbol{n}_\pi \rangle$ from the origin of the reference frame, where $\boldsymbol{p}_\pi \in \mathbb{H}_p$ is an arbitrary point on the plane. Furthermore, a line $\mathcal{H}_p \cap \underline{\boldsymbol{\mathcal{S}}} \ni \underline{\boldsymbol{l}} = \boldsymbol{l} + \varepsilon\boldsymbol{m}$ is defined by the line direction $\mathbb{H}_p \cap \mathbb{S}^3 \ni \boldsymbol{l}$ and the line moment $\boldsymbol{m} = \boldsymbol{p}_l \times \boldsymbol{l}$, in which $\boldsymbol{p}_l \in \mathbb{H}_p$ is an arbitrary point on the line.

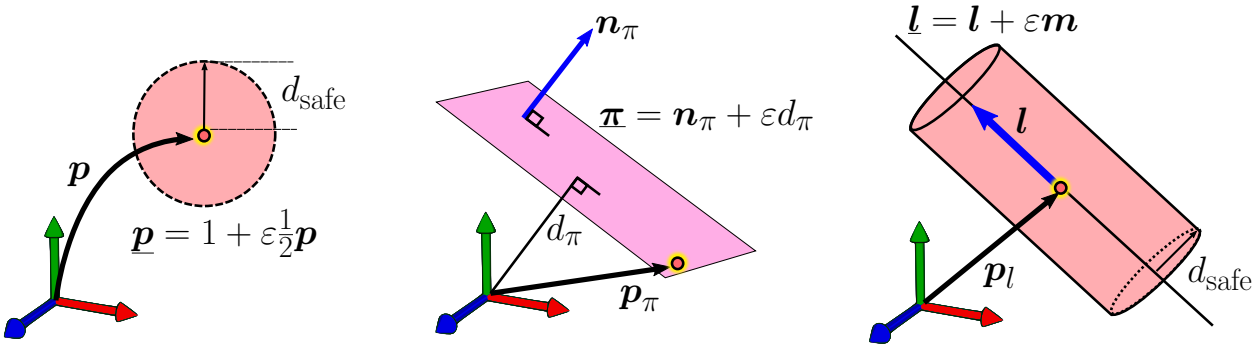

Figure 3.2: Geometric primitives and its representation using dual quaternions. From *left* to *right*: Point, plane and line primitives. Points and lines can also describe spheres and cylinders, respectively, by using safe distances on the VFIs approach.

Marinho et al. (2019) presented some useful distance functions and their corresponding Jacobians based on pair of geometric primitives composed of Plücker lines, planes, and points. To illustrate the computation of the Jacobians and residuals, consider the point-to-plane primitive of Fig. 3.3. Given a robot point $\boldsymbol{p} \triangleq \boldsymbol{p}\left(\boldsymbol{q}\right) \in \mathbb{H}_p$, in which $\boldsymbol{q} \in \mathbb{R}^n$ are the robot joints configurations, and an arbitrary plane $\mathcal{H} \ni \underline{\boldsymbol{\pi}} = \boldsymbol{n}_\pi + \varepsilon d_\pi$, where $d_\pi = \langle \boldsymbol{p}_\pi, \boldsymbol{n}_\pi \rangle$ with $\boldsymbol{p}_\pi$ being an arbitrary point in the plane, the distance between them is given as

$$d_{p,\pi} = \langle \boldsymbol{p}, \boldsymbol{n}_\pi \rangle - d_\pi. \tag{3.8}$$

Using the definition (A.12), the time derivative of (3.8) is given as (Marinho et al.,

2019)

$$\dot{d}_{p,\pi} = \underbrace{\mathrm{vec}_4\left(\boldsymbol{n}_\pi\right)^T \boldsymbol{J}_p}_{\boldsymbol{J}_{p,\pi}}\dot{\boldsymbol{q}} + \underbrace{\langle \boldsymbol{p}, \dot{\boldsymbol{n}}_\pi \rangle - \dot{d}_\pi}_{\zeta_{p,\pi}}, \tag{3.9}$$

where $\boldsymbol{J}_p$ is the translation Jacobian that satisfies the relation $\mathrm{vec}_4\,\dot{\boldsymbol{p}} = \boldsymbol{J}_p\dot{\boldsymbol{q}}$.

The Jacobians and residuals of all primitives are computed analogously to the point-to-point primitive and are summarized in Table 3.1 and illustrated in Fig. 3.3.

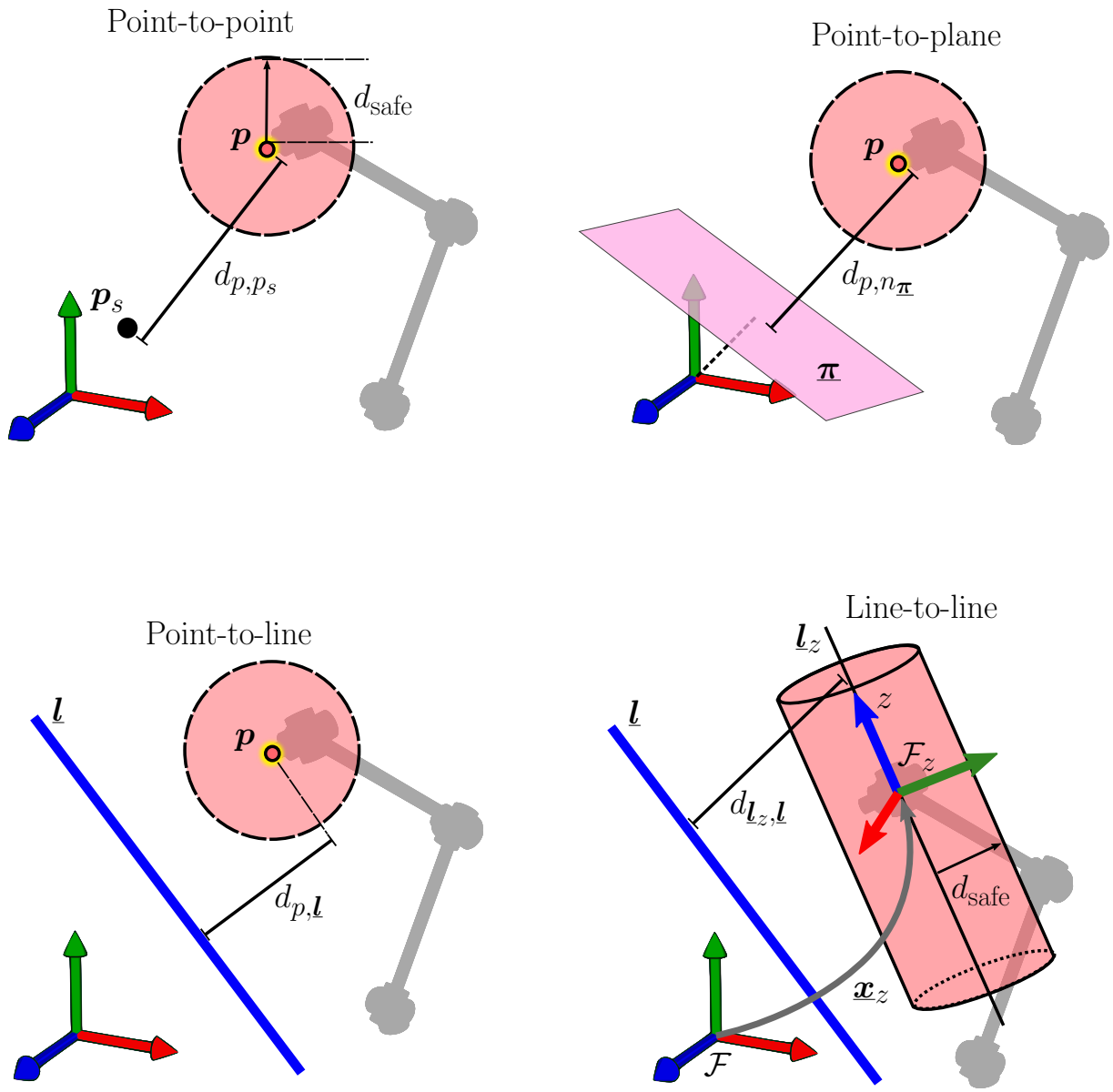

Figure 3.3: Pair of geometric primitives and their distance functions. Point $\boldsymbol{p}$ and frame $\mathcal{F}_z$ are attached to the robot kinematic chain. The pose of frame $\mathcal{F}_z$ is represented by the dual quaternion $\underline{\boldsymbol{x}}_z$.

We can use the geometric primitives to verify collisions between the robot and the obstacles. Consider a robot performing a task near a wall. We can model the wall using a plane and the robot end-effector using a sphere, which corresponds to a robot point with a safe distance, as shown in Fig. 3.3. Then, we can use them as a collision-checker tool by

computing the distance between the primitives as

$$\tilde{d}_{p,\pi} = d_{p,\pi} - d_{\text{safe}},$$

where $d_{p,\pi}$ is the distance between the robot point and the plane, and d_{safe} is a safe distance defined by the user. In this way, if $\tilde{d}_{p,\pi} \geq 0$, the robot is in a free-collision configuration. Otherwise, the robot collided with the obstacle.

Table 3.1: Summary of primitives.(Marinho et al., 2019)

Primitives	Distance function	Jacobian	Residual
Point-to-point	$D_{p,p_s} \triangleq d^2_{p,p_s}$	$\boldsymbol{J}_{p,p_s}$	ζ_{p,p_s}
Point-to-line	$D_{p,\underline{l}} \triangleq d^2_{p,\underline{l}}$	$\boldsymbol{J}_{p,\underline{l}}$	$\zeta_{p,\underline{l}}$
Point-to-plane	$d_{p,\pi}$	$\boldsymbol{J}_{p,\pi}$	$\zeta_{p,\pi}$
Line-to-line	$D_{\underline{l}_z,\underline{l}} \triangleq d^2_{\underline{l}_z,\underline{l}}$	$\boldsymbol{J}_{\underline{l}_z,\underline{l}}$	$\zeta_{\underline{l}_z,\underline{l}}$

3.3 Conclusions

This chapter reviewed some concepts, foundations and operations related to task space control based on quadratic programming and vector field inequalities.

Section. 3.1 presented the control law strategies based on constrained quadratic programming, which are used to minimize the joint velocities or accelerations. These control laws allow the generation of the control inputs under inequality constraints, specifically, the vector field inequalities (VFIs), in the case of robots commanded by joint velocities. In the case of robots commanded by joint accelerations or torques, we use the extension of the VFIs to second order kinematics (SOVFIs), which is presented in Chapter. 4. Section. 3.2 presented a brief introduction about the distance functions between geometric primitives and their respective Jacobians. These concepts are used in Chapter 4 to define new distance functions and Jacobians as the angle between two Plücker lines, which are useful to prevent undesired orientations and violation of the joint limits. Furthermore, the chapter introduces a new Jacobian related to the support polygon (SP) area of a humanoid robot, which is useful to increase the SP area and potentially improve the robot reachability.

4

Second Order Vector Field Inequalities

This chapter shows one of the contributions of this dissertation. First, the VFIs are extended to second order kinematics (SOVFIs). This enables applications that use the robot dynamics by means of the relationship between joint torques and joint accelerations in the Euler-Lagrange equations. Second, a formal proof that shows that SOVFIs ensure collision avoidance is presented. In addition, a new distance function related to the angle between two Plücker lines and the corresponding Jacobian matrix are proposed, which may be useful to prevent the violation of joint limits and/or to avoid undesired end-effector orientations. Next, the chapter shows a new Jacobian related with the support polygon area of a humanoid robot, which may be useful to potentially increase the robot's reachability and the robot safety in terms of its balance. Finally, a strategy for multi-contact applications based on VFIs is presented.

The VFI framework was proposed by Marinho et al. (2018), and it is composed of differential inequalities that are used to prevent collisions between pairs of geometrical primitives within the dual quaternion algebra. It requires a signed distance function $d(t) \in \mathbb{R}$ between two collidable objects and the Jacobian matrix $\boldsymbol{J}_d$ relating the robot joint velocities with the time derivative of the distance; that is, $\dot{d}(t) = \boldsymbol{J}_d \dot{\boldsymbol{q}} + \zeta(t)$, where $\zeta(t)$ is the residual that contains the distance dynamics unrelated to the robot's joints velocities. Furthermore, it is assumed that the residual is known but it cannot be controlled (Marinho et al., 2019).

In order to keep the robot outside a collision zone, the error distance is defined as $\tilde{d}(t) \triangleq d(t) - d_{\text{safe}}$, where $d_{\text{safe}} \triangleq d_{\text{safe}}(t) \in [0, \infty)$ is an arbitrary safe distance, and the

following inequalities must hold for all t (Marinho et al., 2019):

$$\dot{\tilde{d}}(t) \geq -\eta_d \tilde{d}(t) \iff -\boldsymbol{J}_d \dot{\boldsymbol{q}} \leq \eta_d \tilde{d}(t) + \zeta_{\text{safe}}(t), \tag{4.1}$$

where the residual

$$\zeta_{\text{safe}}(t) \triangleq \zeta(t) - \dot{d}_{\text{safe}} \tag{4.2}$$

encodes the effects of a moving obstacle with residual $\zeta(t)$ and of a time-varying safe-zone distance $\dot{d}_{\text{safe}}$. Furthermore $\eta_d \in [0, \infty)$ is used to adjust the approach velocity. The lower is η_d, the lower is the allowed approach velocity.

Alternatively to the work of Marinho et al., 2019, if we consider acceleration inputs, the VFI can be extended by means of a second-order differential inequality

$$\ddot{\tilde{d}}(t) \geq -\eta_d \dot{\tilde{d}}(t) - \eta_p \tilde{d}(t), \tag{4.3}$$

where $\tilde{d}(t) \triangleq d(t) - d_{\text{safe}}$, and $\eta_d, \eta_p \in [0, \infty)$, with $\eta_d \geq 2\left|\dot{\tilde{d}}(0)\right| / \tilde{d}(0)$ are used to adjust the approach acceleration. We enforce the condition $\eta_d^2 - 4\eta_p > 0$ to obtain, in the worst case, a non-oscillatory exponential approach.

Using the fact that $\tilde{d}(t) \triangleq d(t) - d_{\text{safe}}$, we compute its first and second time derivative as $\dot{\tilde{d}}(t) = \dot{d}(t) - \dot{d}_{\text{safe}}$ and $\ddot{\tilde{d}}(t) = \ddot{d}(t) - \ddot{d}_{\text{safe}}$, respectively. Furthermore, we have that $\dot{d}(t) = \boldsymbol{J}_d \dot{\boldsymbol{q}} + \zeta(t) \implies \ddot{d}(t) = \boldsymbol{J}_d \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_d \dot{\boldsymbol{q}} + \dot{\zeta}(t)$. Therefore, using (4.2), we rewrite $\dot{\tilde{d}}(t)$ and $\ddot{\tilde{d}}(t)$, respectively as

$$\dot{\tilde{d}}(t) = \boldsymbol{J}_d \dot{\boldsymbol{q}} + \zeta_{\text{safe}}(t), \tag{4.4}$$

$$\ddot{\tilde{d}}(t) = \boldsymbol{J}_d \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_d \dot{\boldsymbol{q}} + \dot{\zeta}_{\text{safe}}(t). \tag{4.5}$$

Using (4.4) and (4.5), we rewrite the constraint (4.3) as

$$-\boldsymbol{J}_d \ddot{\boldsymbol{q}} \leq \underbrace{\left(\eta_d \boldsymbol{J}_d + \dot{\boldsymbol{J}}_d\right) \dot{\boldsymbol{q}} + \eta_p \tilde{d}(t)}_{\beta_d} + \underbrace{\dot{\zeta}_{\text{safe}}(t) + \eta_d \zeta_{\text{safe}}(t)}_{\beta_{\text{res}}}. \tag{4.6}$$

Analogously, if it is desired to keep the robot inside a safe region (i.e., $\tilde{d}(t) \leq 0$), the constraints (4.1) and (4.3) are rewritten, respectively, as

$$\dot{\tilde{d}}(t) \leq -\eta_d \tilde{d}(t) \iff \boldsymbol{J}_d \dot{\boldsymbol{q}} \leq -\left(\eta_d \tilde{d}(t) + \zeta_{\text{safe}}(t)\right), \tag{4.7}$$

$$\ddot{\tilde{d}}(t) \leq -\eta_d \dot{\tilde{d}}(t) - \eta_p \tilde{d}(t) \iff \boldsymbol{J}_d \ddot{\boldsymbol{q}} \leq -(\beta_d + \beta_{\text{res}}). \tag{4.8}$$

Note that in case of static entities (i.e. fixed obstacles and static safe regions) the term β_{res} equals 0, and the constraints (4.6) and (4.8) are the same as proposed in our previous work (Quiroz-Omana & Adorno, 2019).

In order to show that the second inequality in (4.3) prevents collisions—i.e., $\tilde{d}(t) \geq 0$, $\forall t \in [0, \infty)$ assuming that $\tilde{d}(0) \geq 0$—in robot commanded by means of acceleration inputs,[1] we propose the following lemma.

Lemma 4.1. *Consider the second inequality constraint in (4.3), with* $\eta_d, \eta_p \in (0, \infty)$, $\eta_d^2 - 4\eta_p > 0$, $\tilde{d}(0) \triangleq \tilde{d}_0 > 0$, $\dot{\tilde{d}}(0) \triangleq \dot{\tilde{d}}_0 \in (-\infty, \infty)$ *and* $\eta_d \geq 2\left|\dot{\tilde{d}}_0\right|/\tilde{d}_0$. *Let* $\tilde{d}(t)$ *be a smooth function for all* $t \in [0, \infty)$, *then* $\tilde{d}(t) > 0$, $\forall t \in [0, \infty)$.

Proof. This proof is based on the Comparison Lemma, which is stated for first order scalar differential equations. Therefore, first we reduce the order of the differential inequality. Consider $\ddot{\tilde{d}}(t) = -\eta_d \dot{\tilde{d}}(t) - \eta_p \tilde{d}(t)$, where $\tilde{d}(0) = \tilde{d}_0$, $\dot{\tilde{d}}(0) \in (-\infty, \infty)$, and η_d, η_p respects the aforementioned conditions. Let $\boldsymbol{z} \triangleq \begin{bmatrix} z_1 & z_2 \end{bmatrix}^T$, with $z_1 \triangleq \tilde{d}(t)$ and $z_2 \triangleq \dot{\tilde{d}}(t)$, then, the second ODE is rewritten as

$$\dot{\boldsymbol{z}} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\eta_p & -\eta_d \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{z}. \tag{4.9}$$

By the existence and uniqueness theorem, the solution exists, and is unique (Boyce & DiPrima, 2008, p. 111). To compute the solution, we first pre-multiply both sides of (4.9) by $\exp(-\boldsymbol{A}t)$, which can be rewritten as (Chen, 1998, p. 87)

$$\underbrace{\exp(-\boldsymbol{A}t)\,\dot{\boldsymbol{z}} - \exp(-\boldsymbol{A}t)\,\boldsymbol{A}\boldsymbol{z}}_{\frac{d}{dt}(\exp(-\boldsymbol{A}t)\boldsymbol{z})} = \boldsymbol{0}. \tag{4.10}$$

By integrating (4.10) from 0 to t, we can rewrite

$$\int_0^t \frac{d}{d\tau}\left(\exp(-\boldsymbol{A}\tau)\,\boldsymbol{z}\right) d\tau = \boldsymbol{0} \implies \boldsymbol{z} = \exp(\boldsymbol{A}t)\,\boldsymbol{z}(0), \tag{4.11}$$

where $\exp(\boldsymbol{A}t)$ can be computed as follows (Chen, 1998, p. 87)

$$\exp(\boldsymbol{A}t) = \mathscr{L}^{-1}\left[(s\boldsymbol{I}_2 - \boldsymbol{A})^{-1}\right], \tag{4.12}$$

where $\boldsymbol{I}_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix, and $(s\boldsymbol{I}_2 - \boldsymbol{A})^{-1}$ is given as follows

$$(s\boldsymbol{I}_2 - \boldsymbol{A})^{-1} = \begin{bmatrix} s & -1 \\ n_p & s + n_d \end{bmatrix}^{-1} = \frac{1}{s^2 + \eta_d s + \eta_p} \begin{bmatrix} \eta_d + s & 1 \\ -\eta_p & s \end{bmatrix}. \tag{4.13}$$

[1] Actual robots are usually commanded by means of velocity or torque inputs. In the latter case, acceleration inputs are tranformed into torque inputs by using (3.7).

The roots $r_1, r_2 \in (-\infty, 0)$ of the characteristic equation $s^2 + \eta_p s + \eta_d$ are given as follows

$$r_1 = \frac{-n_d + \sqrt{\eta_d^2 - 4\eta_p}}{2}, \quad r_2 = \frac{-n_d - \sqrt{\eta_d^2 - 4\eta_p}}{2}.$$

The equation (4.13) can be rewritten as,

$$(s\boldsymbol{I}_2 - \boldsymbol{A})^{-1} = \frac{1}{(s - r_1)(s - r_2)} \begin{bmatrix} \eta_d + s & 1 \\ -\eta_p & s \end{bmatrix}. \tag{4.14}$$

Using the Laplace transformations considering zero initial conditions (Ogata, 2010, p. 863),

$$\frac{1}{(s - r_1)(s - r_2)} \Longleftrightarrow \frac{1}{r_1 - r_2}\left(\exp(r_1 t) - \exp(r_2 t)\right)$$
$$\frac{s}{(s - r_1)(s - r_2)} \Longleftrightarrow \frac{1}{r_1 - r_2}\left(r_1 \exp(r_1 t) - r_2 \exp(r_2 t)\right),$$

to compute (4.12), and using (4.11), the solution of equation (4.9) is given by

$$\boldsymbol{z} = \left(\frac{1}{r_1 - r_2}\right) \begin{bmatrix} \exp(r_1 t)(r_1 + \eta_d) + \exp(r_2 t)(-\eta_d - r_2) & \exp(r_1 t) - \exp(r_2 t) \\ -\eta_p\left(\exp(r_1 t) - \exp(r_2 t)\right) & r_1 \exp(r_1 t) - r_2 \exp(r_2 t) \end{bmatrix} \boldsymbol{z}(\boldsymbol{0}). \tag{4.15}$$

Since $\boldsymbol{z}(\boldsymbol{0}) = \begin{bmatrix} \tilde{d}_0 & \dot{\tilde{d}}_0 \end{bmatrix}^T$, and using the facts $r_1 + r_2 = -\eta_d$, and $r_1 \cdot r_2 = \eta_p$, we rewrite (4.15) as follows

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} c_1 \exp(r_1 t) + c_2 \exp(r_2 t) \\ c_1 r_1 \exp(r_1 t) + c_2 r_2 \exp(r_2 t) \end{bmatrix}, \tag{4.16}$$

where $c_1, c_2 \in \mathbb{R}$ are given as

$$c_1 \triangleq \frac{\dot{\tilde{d}}_0 - \tilde{d}_0 r_2}{r_1 - r_2}, \quad c_2 \triangleq \frac{\tilde{d}_0 r_1 - \dot{\tilde{d}}_0}{r_1 - r_2}. \tag{4.17}$$

Let us rewrite analogously the inequality (4.3) as

$$\dot{\boldsymbol{v}} \geq \boldsymbol{A}\boldsymbol{v},$$

where $\boldsymbol{v} \triangleq \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T$, with $v_1 \triangleq \tilde{d}(t)$ and $v_2 \triangleq \dot{\tilde{d}}(t)$. Because the inequality $\dot{v}_1 \geq \dot{z}_1$ is satisfied when $v_1(0) \geq z_1(0)$ for all $t \in [0, \infty)$, then, by the Comparison Lemma[2] (Khalil, 1996, p. 85)

$$v_1 \geq z_1 \quad \forall t \in [0, \infty), \tag{4.18}$$

[2]Let $\dot{u} = f(t, u)$, $\dot{v} \leq f(t, v)$, where f is continuous in t and in the functions $u(t)$ and $v(t)$, and $v_0 \leq u_0$; then $v(t) \leq u(t) \ \forall t \in [0, T)$.

which implies

$$\tilde{d}(t) \geq \underbrace{c_1 \exp(r_1 t)}_{f_1(t)} + \underbrace{c_2 \exp(r_2 t)}_{f_2(t)}, \quad \forall t \in [0, \infty), \tag{4.19}$$

where $r_1 - r_2 > 0$. Both $f_1(t) \triangleq \exp(r_1 t)$ and $f_2 \triangleq \exp(r_2 t)$ are decreasing monotonic functions where $f_2(t)$ decreases at a higher rate than $f_1(t)$ due to the fact that $r_1, r_2 \in (-\infty, 0)$ and $r_2 < r_1$. Therefore, by inspection (see Fig. (4.1)), $\tilde{d}(t) \geq 0, \quad \forall t \in [0, \infty)$ if $c_1 \geq 0$ and $c_1 \geq |c_2|$, which is satisfied when $\dot{\tilde{d}}_0 \geq \tilde{d}_0 r_2$ and $\dot{\tilde{d}}_0 \geq \tilde{d}_0 (r_1 + r_2)/2$. Because $\dot{\tilde{d}}_0 \geq \tilde{d}_0 (r_1 + r_2)/2 \geq \tilde{d}_0 r_2$ and $\eta_d = -(r_1 + r_2)$, then $\eta_d \geq -2\dot{\tilde{d}}_0/\tilde{d}_0$. Since $2\left|\dot{\tilde{d}}_0\right|/\tilde{d}_0 \geq -2\dot{\tilde{d}}_0/\tilde{d}_0$ then we choose $\eta_d \geq 2\left|\dot{\tilde{d}}_0\right|/\tilde{d}_0$, which concludes the proof. □

Since we consider arbitrary values for $\dot{\tilde{d}}_0$, additional bounds on the accelerations (i.e., $\ddot{\boldsymbol{q}}_{\min} \leq \ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{q}}_{\max}$) may result in the unfeasibility of constraints (4.6) and (4.8). Therefore, existing techniques (Del Prete, 2018) may be adapted to determine the maximum bounds on the accelerations such that the VFIs can still be satisfied.

Fig (4.1) shows the behavior of solution (4.19) for different cases where the coefficients c_1 and c_2 have different relations between them. Notice that for both the cases $c_1 \geq 0$, $c_2 \geq 0$ and $c_1 > 0$, $c_2 < 0$, $c_1 = |c_2|$, we also have $\tilde{d}(t) \geq 0, \quad \forall t \in [0, \infty)$. The former case is satisfied when $\tilde{d}_0 r_2 \leq \dot{\tilde{d}}_0 \leq \tilde{d}_0 r_1$ whereas the latter requires $\dot{\tilde{d}}_0 \geq \tilde{d}_0 r_1$ and $\dot{\tilde{d}}_0 > \tilde{d}_0 r_1$. The case $c_1 \geq 0$, $c_1 \geq |c_2|$, which is used in the proof of the Lemma 4.1, is the only one whose requirement is given in terms of the gain η_d. This is convenient to define the criteria design of the constraint (4.3).

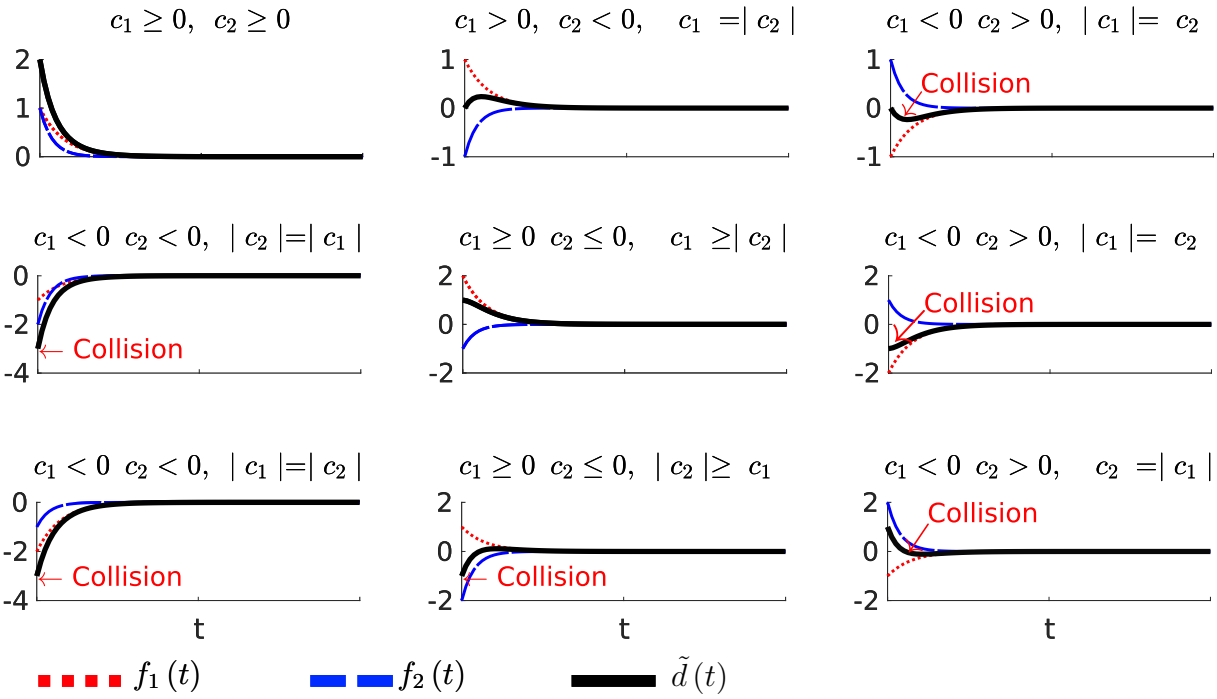

Figure 4.1: Relation between the distance $\tilde{d}(t)$ and the coefficients c_1 and c_2. The distance $\tilde{d}(t)$ is higher than or equal to zero when $c_1 \geq 0$ and $c_2 \geq 0$ or $c_1 \geq 0$ and $c_1 \geq |c_2|$.

4.1 Simulation Test

To evaluate the SOVFIs framework in the presence of a dynamic object, this section presents a simulation using a 6-DOF arm manipulator, commanded by joint torques, performing a task of controlling the end-effector position. The simulation is implemented on Python using the computation library DQ Robotics (Adorno & Marques Marinho, 2020) and CoppeliaSim.[3] Furthermore, the time-varying position of the object is given by $\mathbb{H}_p \ni \boldsymbol{p}_s \triangleq \boldsymbol{p}_s(t) = \boldsymbol{p}_s(0) - (b\sin(wt)\,\hat{\imath} + at\hat{\jmath})$, with $\boldsymbol{p}_s(0) = 0.23\hat{\imath} + 0.24\hat{\jmath} + 0.8\hat{k}$, $a = 0.1$, $b = 0.2$, and $w = 0.6$. The goal is to prevent collisions between the dynamic object and a spherical region in space, whose center is located in the end-effector, which is denoted as $\boldsymbol{p} \triangleq \boldsymbol{p}(\boldsymbol{q}) \in \mathbb{H}_p$, and $\boldsymbol{q} \in \mathbb{R}^6$ represents the robot joint configurations. We use the point-to-point primitive (3.9) to impose the constraint (4.6) with $d_{\text{safe}} = 0.22$, with $\dot{d}_{\text{safe}} = 0, \forall t$, and the control law (3.6). We compute the square distance[4] between the entities as

$$D_{p,p_s} \triangleq d_{p,p_s}^2 = \|\boldsymbol{p} - \boldsymbol{p}_s\|^2 \tag{4.20}$$

Using the definition (A.12), the time derivative of (4.20) is given as

$$\dot{D}_{p,p_s} = \underbrace{2\,\text{vec}_4\left(\boldsymbol{p} - \boldsymbol{p}_s\right)^T \boldsymbol{J}_p}_{\boldsymbol{J}_{p,p_s}}\dot{\boldsymbol{q}} + \underbrace{2\langle \boldsymbol{p} - \boldsymbol{p}_s, -\dot{\boldsymbol{p}}_s\rangle}_{\zeta_{p,p_s}}, \tag{4.21}$$

where $\boldsymbol{J}_p$ is the translation Jacobian that satisfies the relation $\text{vec}_4\,\dot{\boldsymbol{p}} = \boldsymbol{J}_p\dot{\boldsymbol{q}}$, and ζ_{p,p_s} is the residual that contains the information about the velocity of the dynamic object.

The error distance is defined as

$$\tilde{D}_{p,p_s} \triangleq D_{p,p_s} - D_{\text{safe}}, \tag{4.22}$$

where $D_{\text{safe}} \triangleq d_{\text{safe}}^2$ is the square safe distance. Furthermore, the time derivative of the error distance is given as

$$\dot{\tilde{D}}_{p,p_s} = \boldsymbol{J}_{p,p_s}\dot{\boldsymbol{q}} + \zeta_{\text{safe}} \implies \ddot{\tilde{D}}_{p,p_s} = \boldsymbol{J}_{p,p_s}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{p,p_s}\dot{\boldsymbol{q}} + \dot{\zeta}_{\text{safe}}, \tag{4.23}$$

where $\zeta_{\text{safe}} \triangleq \zeta_{p,p_s} - D_{\text{safe}}$. Finally, the SOVFI constraint is given by (4.6) with $\boldsymbol{J}_d \triangleq \boldsymbol{J}_{p,p_s}$, and $\tilde{d}(t) \triangleq \tilde{D}_{p,p_s}$.

Fig. 4.3 shows the snapshots of three simulations that represent three cases. In case I, the SOVFI constraint is disabled. Because of that, the robot performs the task but does not take into account the obstacle, and consequently, the robot and the object collide, as expected. The case II shows a situation where the SOVFI constraint is enabled but the kinematics of the object is not taken into account. In other words, the robot knows

[3] https://www.coppeliarobotics.com/

[4] We use the square distance since $\dot{d}_{p,p_s}$ is singular at $d_{p,p_s} = 0$.

the position of the object all the time, but considers it as a static entity, and therefore, both the velocity and acceleration of the object are neglected. We simulate that situation by setting $\beta_{\text{res}} = 0$ in (4.6). Notice that β_{res} contains the information about the object kinematics. The robot performs the task and tries to avoid the collision but, in this specific example, the evasive maneuvers were not enough, and finally, both the robot and the object collide. This is expected since the robot does not know the required information about the object to prevent the collision. Finally, case III presents the best scenario, in terms of collisions avoidance. The SOVFI constraint is enabled and the kinematics of the object is taken into account. In this case, the robot performs the task and prevents collision with the object. Fig. 4.2 shows the distances, for the three cases, between the dynamic object and the robot end-effector point.

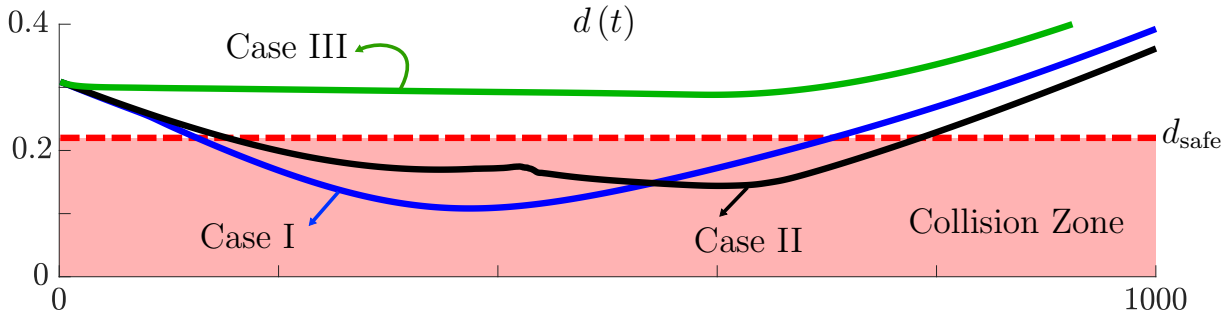

Figure 4.2: Control of the position end-effector using the robot dynamics and the SOVFIs (Distances for the three cases). The distance between the dynamic object and the robot end-effector point is represented by $d\,(t)$. The radius of the sphere, which correspond to the safe distance, is $d_{\text{safe}} = 0.22$.

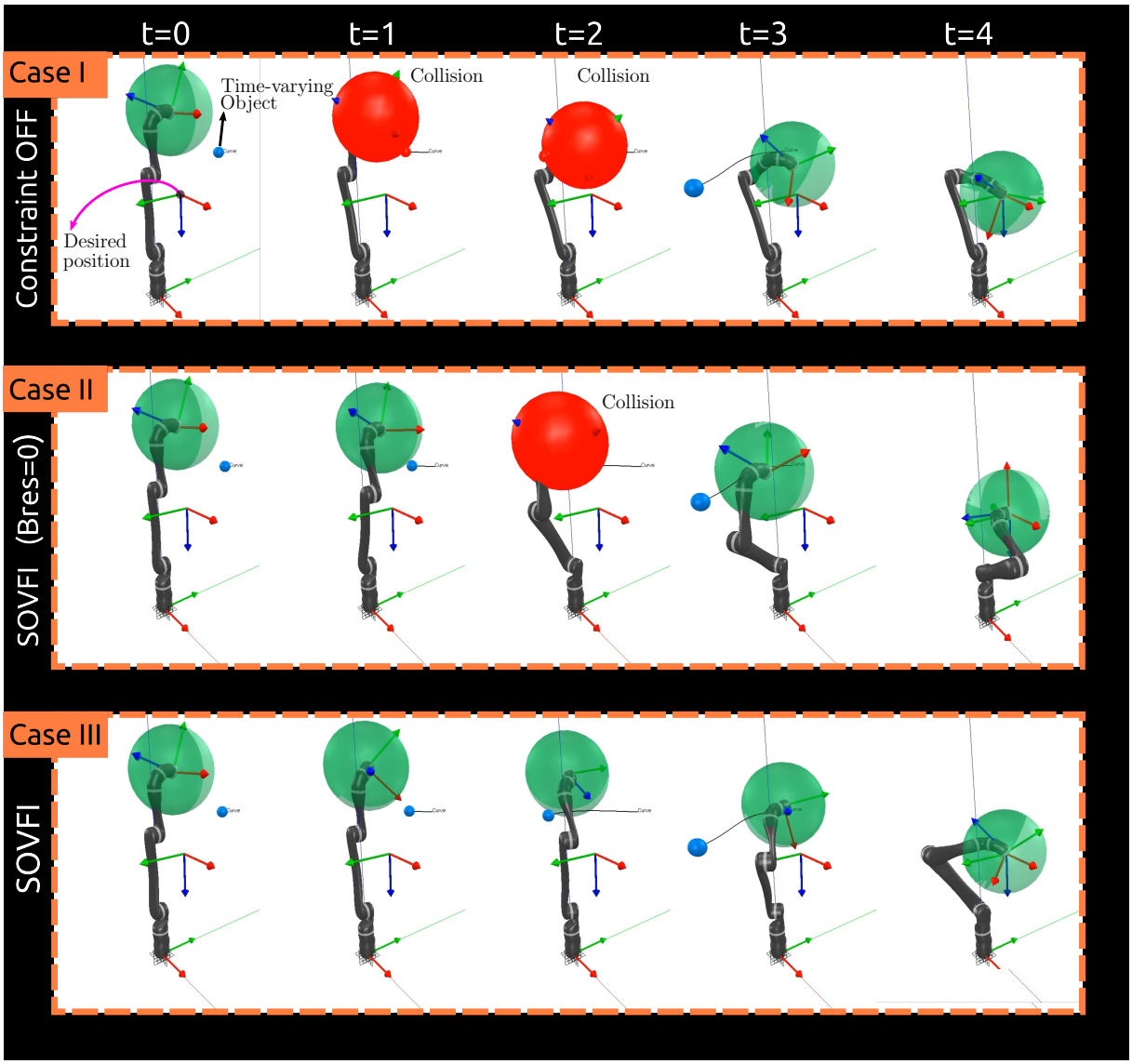

Figure 4.3: Control of the position end-effector using the robot dynamics and the SOVFIs (Snapshots of the simulation). The green ball has radius d_{safe} and represent the safe region. A blue dynamic object collides with the robot safe region when the SOVFIs constraint is disabled or when $\beta_{\text{res}} = 0$. When two objects collide both are presented in red.

4.2 Line-to-line Angle Jacobian

In some applications, it is advantageous to define target regions, instead of one specific position and/or orientation, in order to relax the task and, therefore, release some of the robot degrees of freedom (DOF) for additional tasks. For instance, if the task consists in carrying a cup of water from one place to another, one way to perform it is by defining a time-varying trajectory that determines the robot end-effector pose at all times, which requires six DOF. An alternative is to define a target position to where the cup of water should be moved (which requires three DOF) while ensuring that the cup is not too tilted to prevent spilling, which requires one more DOF (i.e., the angle between a vertical line and the line passing through the center of the cup, as shown in Fig. 4.5). The task can be further relaxed by using only the distance to the final position, which requires only one DOF, plus the angle constraint, which requires, when activated, one more DOF.

The idea of using a conic constraint, which is equivalent to constraining the angle between two intersecting lines, was first proposed by Gienger et al., 2006. However, their description is singular when the angle equals $k\pi$, for all $k \in \mathbb{Z}$. Therefore, we propose a singularity-free conic constraint based on VFIs. Since this constraint requires a new Jacobian, namely the line-to-line angle Jacobian, which is based on the line Jacobian, we briefly review the line Jacobian (Marinho et al., 2018).

Consider the Line-to-line primitive, as showed in Fig. 4.4. Given a frame $\mathcal{F}_z$ attached to the robot kinematic chain, whose pose is given by $\underline{\boldsymbol{x}}_z \triangleq \underline{\boldsymbol{x}}_z(\boldsymbol{q}) = \boldsymbol{r} + \varepsilon\frac{1}{2}\boldsymbol{p}\boldsymbol{r}$, a dynamic Plücker line $\mathcal{H}_p \cap \underline{\boldsymbol{\mathcal{S}}} \ni \underline{\boldsymbol{l}}_z \triangleq \underline{\boldsymbol{l}}_z(\boldsymbol{q})$ collinear to the z-axis of $\mathcal{F}_z$ is described with respect to the inertial frame $\mathcal{F}$ as

$$\underline{\boldsymbol{l}}_z = \boldsymbol{l}_z + \varepsilon\boldsymbol{m}_z, \tag{4.24}$$

where $\mathbb{H}_p \cap \mathbb{S}^3 \ni \boldsymbol{l}_z = \boldsymbol{r}\hat{\boldsymbol{k}}\boldsymbol{r}^*$ is the line direction, and the line moment $\boldsymbol{m}_z = \boldsymbol{p}_z \times \boldsymbol{l}_z$, in which $\boldsymbol{p}_z \in \mathbb{H}_p$ is an arbitrary point on the line. The time derivative of (4.24) is given by (Marinho et al., 2018)

$$\dot{\underline{\boldsymbol{l}}}_z = \dot{\boldsymbol{l}}_z + \varepsilon\dot{\boldsymbol{m}}_z \implies \operatorname{vec}_6 \dot{\underline{\boldsymbol{l}}}_z = \underbrace{\begin{bmatrix} \boldsymbol{J}_{l_z} \\ \boldsymbol{J}_{m_z} \end{bmatrix}}_{\boldsymbol{J}_{\underline{l}_z}} \dot{\boldsymbol{q}}, \tag{4.25}$$

where $\boldsymbol{J}_{\underline{l}_z}$ is the line Jacobian.

The angle between $\underline{\boldsymbol{l}}_z = \boldsymbol{l}_z + \varepsilon\boldsymbol{m}_z$ and an arbitrary Plücker line $\underline{\boldsymbol{l}} = \boldsymbol{l} + \varepsilon\boldsymbol{m}$, is given by

$$\phi_{\underline{l}_z,\underline{l}} = \arccos\left(\langle \boldsymbol{l}_z, \boldsymbol{l} \rangle\right). \tag{4.26}$$

The time derivative of 4.26 is

$$\dot{\phi}_{\underline{l}_z,\underline{l}} = -\frac{1}{\sqrt{1-\left(\langle \boldsymbol{l}_z, \boldsymbol{l}\rangle\right)^2}} \operatorname{vec}_3 \boldsymbol{l}^T \boldsymbol{J}_{l_z} \dot{\boldsymbol{q}} + \langle \boldsymbol{l}_z, \dot{\boldsymbol{l}}\rangle. \tag{4.27}$$

The function (4.26) is an intuitive choice to control the angle between both lines. However, its time derivative, which is given by (4.27), is singular when $\langle \boldsymbol{l}_z, \boldsymbol{l}\rangle = \pm 1$. We propose the function $f : [0,\pi] \to [0,4]$ instead:

$$f\left(\phi_{\underline{l}_z,\underline{l}}\right) \triangleq \|\boldsymbol{l}_z - \boldsymbol{l}\|_2^2 = \langle \boldsymbol{l}_z - \boldsymbol{l}, \boldsymbol{l}_z - \boldsymbol{l}\rangle \tag{4.28}$$

$$= 2 - 2\cos\phi_{\underline{l}_z,\underline{l}}. \tag{4.29}$$

As $f\left(\phi_{\underline{l}_z,\underline{l}}\right)$ is a smooth bijective function (see Lemma (4.2)), controlling the distance function (4.28) is equivalent to controlling the angle $\phi_{\underline{l}_z,\underline{l}} \in [0,\pi]$.

The time derivative of (4.28) is given by

$$\frac{d}{dt} f\left(\phi_{\underline{l}_z,\underline{l}}\right) = \underbrace{2\operatorname{vec}_3\left(\boldsymbol{l}_z - \boldsymbol{l}\right)^T \boldsymbol{J}_{l_z}}_{\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}} \dot{\boldsymbol{q}} + \underbrace{2\langle \boldsymbol{l}_z - \boldsymbol{l}, -\dot{\boldsymbol{l}}\rangle}_{\zeta_{\phi_{\underline{l}_z,\underline{l}}}}. \tag{4.30}$$

Notice that if the angle between the lines $\phi_{\underline{l}_z,\underline{l}}$ equals zero, which happen when both orientation lines are the same (i.e., $\boldsymbol{l}_z - \boldsymbol{l} = \boldsymbol{0}$), we have $\frac{d}{dt} f\left(\phi_{\underline{l}_z,\underline{l}}\right) = 0$.

Proposition 4.1. *Given $f : [0,\pi] \to [0,4]$, with $f\left(\phi_{\underline{l}_z,\underline{l}}\right) = 2 - 2\cos\phi_{\underline{l}_z,\underline{l}}$, for all $a, a' \in [0,\pi]$, $a \neq a'$, implies $f(a) \neq f(a')$. In other words, $f : [0,\pi] \to [0,4]$ is injective (Hammack, 2018, p. 228).*

Proof. We use the contrapositive[5] approach. Suppose that $a, a' \in [0,\pi]$ and $f(a) = f(a')$. Then, we have that $2 - 2\cos a = 2 - 2\cos a' \implies a = a'$. By contraposition, for all $a, a' \in [0,\pi]$, with $a \neq a'$ implies $f(a) \neq f(a')$. Therefore, $f\left(\phi_{\underline{l}_z,\underline{l}}\right)$ is injective. □

Proposition 4.2. *Given $f : [0,\pi] \to [0,4]$, with $f\left(\phi_{\underline{l}_z,\underline{l}}\right) = 2 - 2\cos\phi_{\underline{l}_z,\underline{l}}$, for every $b \in [0,4]$ there is an $a \in [0,\pi]$, with $f(a) = b$. In other words, $f : [0,\pi] \to [0,4]$ is surjective (Hammack, 2018, p. 228).*

Proof. Suppose $b \in [0,4]$, then we seek an $a \in [0,\pi]$ for which $f(a) = b$, that is, for which $2 - 2\cos a = b$. Solving for a, we have $a = \arccos\left((2-b)/2\right)$, which is defined because $b \in [0,4]$. Therefore, $f\left(\phi_{\underline{l}_z,\underline{l}}\right)$ is surjective. □

Lemma 4.2. *The function $f : [0,\pi] \to [0,4]$, which is given as $f\left(\phi_{\underline{l}_z,\underline{l}}\right) = 2 - 2\cos\phi_{\underline{l}_z,\underline{l}}$, is bijective.*

Proof. From propositions (4.1) and (4.2) we have that $f\left(\phi_{\underline{l}_z,\underline{l}}\right)$ is both injective and surjective, respectively. Hence, $f\left(\phi_{\underline{l}_z,\underline{l}}\right)$ is bijective. □

[5] Given the statement $P \implies Q$, its equivalent contrapositive form is $\neg Q \implies \neg P$.

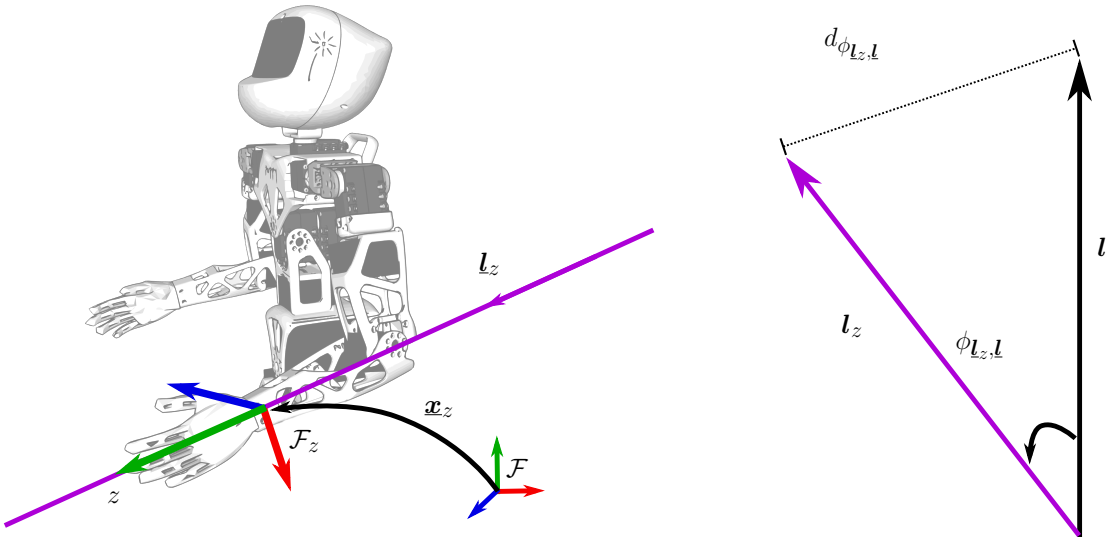

Figure 4.4: The figure on the *left* shows a Plücker line $\underline{\boldsymbol{l}}_z$ attached to one of the links in the robot kinematic chain. The frame $\mathcal{F}_z$ is attached to the robot kinematic chain and its pose is represented by the dual quaternion $\underline{\boldsymbol{x}}_z$. The figure on the *right* shows the angle $\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}$ between two Plücker lines, which is related to the distance $d_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}}$ by means of the law of cosines.

4.3 (Self)-Collision Avoidance Constraints

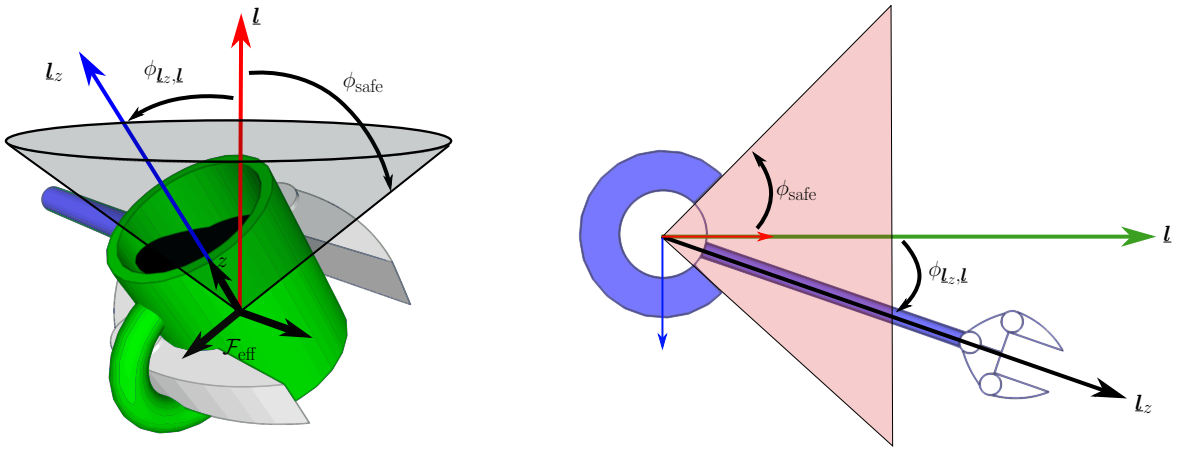

Figure 4.5: Applications of the line-to-line-angle Jacobian. On the *left*, the constraint is used to restrict the orientation of the end-effector with respect to a vertical line passing through the origin of the end-effector frame. On the *right*, the constraint is used to impose joint limits.

As shown in Fig. 4.5, the line-static-line-angle constraint can be used to prevent violation of joint limits (*right*) and also to avoid undesired end-effector orientations (*left*). In order to prevent violation of joint limits, for each joint we place a static line $\underline{\boldsymbol{l}}$, (i.e., $\dot{\underline{\boldsymbol{l}}} = \mathbf{0}$, $\forall t$), perpendicular to the joint rotation axis and in the middle of its angular displacement range. We also place a line, $\underline{\boldsymbol{l}}_z$, along the link attached to the joint; therefore, the angle between them is calculated by using (4.28). Since the angle distance is constrained between

$[0, \pi]$ in any direction (the other one is equivalent to $[-\pi, 0]$), the maximum joint range limit can be defined in the interval $[-\pi, \pi]$, as shown in Fig. 4.6.

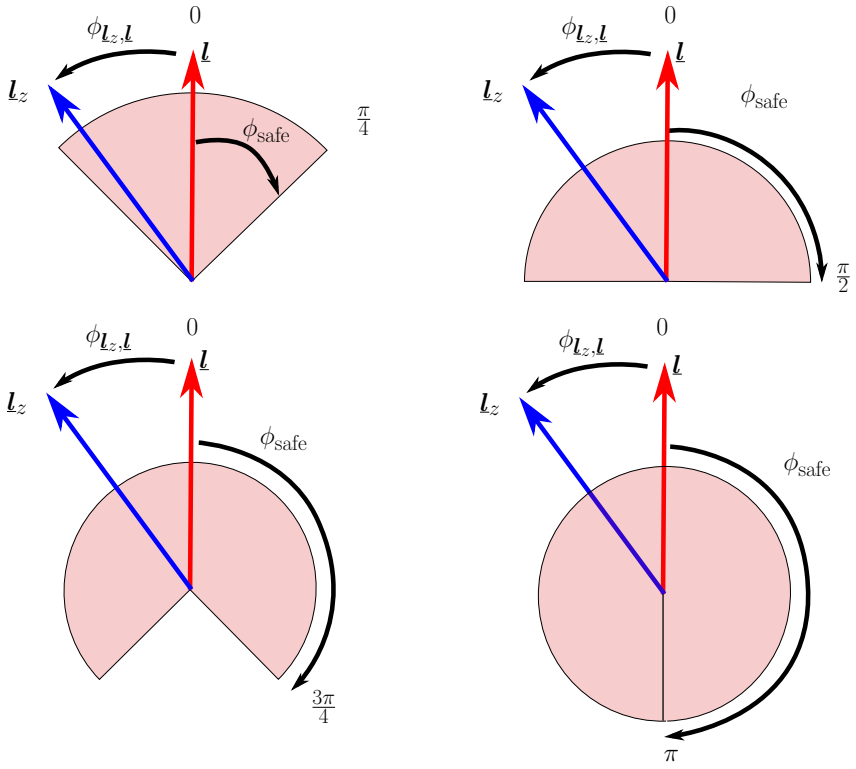

Figure 4.6: Range limits between two lines. The maximum range limit is when $\phi_{\text{safe}} = \pi$. This allows ranges motions between $-\pi \leq \phi_{\underline{l}_z,\underline{l}} \leq \pi$.

Likewise, this strategy allows defining a maximum end-effector angle to avoid undesired orientations. In this case, however, the angle is obtained between a vertical line, $\underline{\boldsymbol{l}}$, passing through the origin of the end-effector frame and a line collinear with the z-axis of the end-effector frame, $\underline{\boldsymbol{l}}_z$. Given a maximum angle ϕ_{safe}, the constraint $\phi_{\underline{l}_z,\underline{l}} \leq \phi_{\text{safe}}$ defines a cone whose centerline is given by $\underline{\boldsymbol{l}}$.

We define the distance error as $\tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right) \triangleq f\left(\phi_{\underline{l}_z,\underline{l}}\right) - f\left(\phi_{\text{safe}}\right)$. Using (4.7) and (4.8), we obtain the corresponding first-order and second-order VFIs used to constrain the angle inside a safe cone, respectively, as

$$\dot{\tilde{f}}\left(\phi_{\underline{l}_z,\underline{l}}\right) \leq -\eta \tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right), \tag{4.31}$$

$$\ddot{\tilde{f}}\left(\phi_{\underline{l}_z,\underline{l}}\right) \leq -\eta_1 \dot{\tilde{f}}\left(\phi_{\underline{l}_z,\underline{l}}\right) - \eta_2 \tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right). \tag{4.32}$$

Using the the facts $\dot{\tilde{f}}\left(\phi_{\underline{l}_z,\underline{l}}\right) = \boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\dot{\boldsymbol{q}} + \zeta_{\phi_{\text{safe}}}$ and $\ddot{\tilde{f}}\left(\phi_{\underline{l}_z,\underline{l}}\right) = \boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\phi_{\underline{l}_z,\underline{l}}}\dot{\boldsymbol{q}} + \dot{\zeta}_{\phi_{\text{safe}}}$, with $\zeta_{\phi_{\text{safe}}} \triangleq \zeta_{\phi_{\underline{l}_z,\underline{l}}} - \dot{f}\left(\phi_{\text{safe}}\right)$, we rewrite (4.31) and (4.32), respectively, as

$$\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\dot{\boldsymbol{q}} \leq -\left(\eta \tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right) + \zeta_{\phi_{\text{safe}}}\right), \tag{4.33}$$

$$\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\ddot{\boldsymbol{q}} \leq -\left(\beta_{\phi_{\underline{l}_z,\underline{l}}} + \beta_{\phi_{\text{res}}}\right), \tag{4.34}$$

where $\beta_{\phi_{\underline{l}_z,\underline{l}}} = \left(\eta_1 \boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}} + \dot{\boldsymbol{J}}_{\phi_{\underline{l}_z,\underline{l}}}\right)\dot{\boldsymbol{q}} + \eta_2 \tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right)$, $\dot{\boldsymbol{J}}_{\phi_{\underline{l}_z,\underline{l}}} = 2\left[\dot{\boldsymbol{q}}^T \boldsymbol{J}_{l_z}^T \boldsymbol{J}_{l_z} + \mathrm{vec}_3\left(\boldsymbol{l}_z - \boldsymbol{l}\right)^T \dot{\boldsymbol{J}}_{l_z}\right]$, and $\beta_{\phi_{\mathrm{res}}} = \dot{\zeta}_{\phi_{\mathrm{safe}}} + \eta_1 \zeta_{\phi_{\mathrm{safe}}}$. Notice that for a static line $\underline{\boldsymbol{l}}$ (i.e., $\dot{\underline{\boldsymbol{l}}} = \boldsymbol{0}$, $\forall t$), and for a static safe region (i.e., $\dot{f}\left(\phi_{\mathrm{safe}}\right) = 0$, $\forall t$), the residual $\zeta_{\phi_{\mathrm{safe}}}$ and the term $\beta_{\phi_{\mathrm{res}}}$ are equal to zero. In those cases, constraints (4.33) and (4.34) are written, respectively, as

$$\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\dot{\boldsymbol{q}} \leq -\eta \tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right), \tag{4.35}$$

$$\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}\ddot{\boldsymbol{q}} \leq -\beta_{\phi_{\underline{l}_z,\underline{l}}}. \tag{4.36}$$

In order to prevent self-collisions, we modeled the robot with spheres and cylinders, as shown in Fig. 4.7.

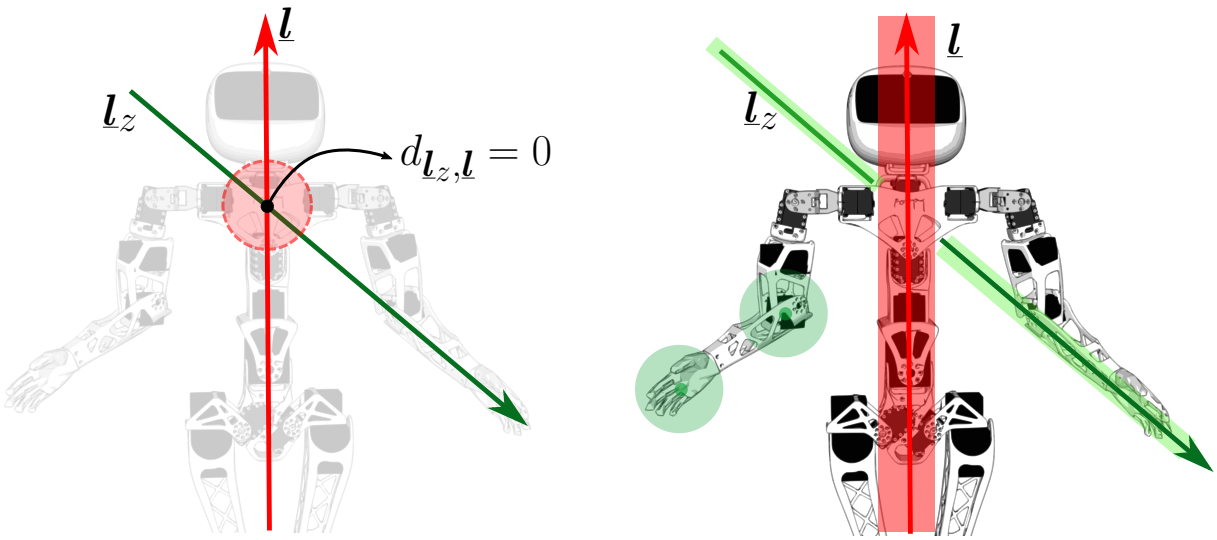

Figure 4.7: On the *left*, the distance $d_{\underline{l}_z,\underline{l}}$ between the lines is zero. However, there is no collisions between the torso and the arm. On the *right*, the robot description using geometric primitives. The torso and the forearms are modeled with infinite cylinders and spheres.

The line-to-line constraint is used to prevent collisions between the arm and the torso, where the line $\underline{\boldsymbol{l}}_z$ is located along the torso and the line $\underline{\boldsymbol{l}}$ is placed along the forearm. In case that the distance $d_{\underline{l}_z,\underline{l}}$ between the lines is zero but there is no collision (see Fig (4.7)), which could happen because lines are infinite, the constraint is disabled and a point-static-line constraint is used instead. In this case, the point located at the hand or the one located at the elbow is used, depending which one is closest to the line. The corresponding first-order and second-order VFIs to describe the torso-arm constraint are written, respectively, as

$$-\boldsymbol{J}_{\mathrm{tarm}}\dot{\boldsymbol{q}} \leq \eta \tilde{d}_{\mathrm{tarm}} + \zeta_{\mathrm{tarm}_{\mathrm{safe}}}, \tag{4.37}$$

$$-\boldsymbol{J}_{\mathrm{tarm}}\ddot{\boldsymbol{q}} \leq \beta_{\mathrm{tarm}} + \beta_{\mathrm{tarm}_{\mathrm{res}}}, \tag{4.38}$$

where $\beta_{\text{tarm}_{\text{res}}} = \dot{\zeta}_{\text{tarm}_{\text{safe}}} + \eta_1 \zeta_{\text{tarm}_{\text{safe}}}$, with

$$\begin{aligned} \boldsymbol{J}_{\text{tarm}} &= \boldsymbol{J}_{\underline{l}_z,\underline{l}}, & \tilde{d}_{\text{tarm}} &= d_{\underline{l}_z,\underline{l}} - d_{\text{safe}}, & \zeta_{\text{tarm}_{\text{safe}}} &= \zeta_{\underline{l}_z,\underline{l}} - \dot{d}_{\text{safe}}, & \text{if } d_{\underline{l}_z,\underline{l}} \neq 0, \\ \text{or } \boldsymbol{J}_{\text{tarm}} &= \boldsymbol{J}_{p,\underline{l}}, & \tilde{d}_{\text{tarm}} &= d_{p,\underline{l}} - d_{\text{safe}}, & \zeta_{\text{tarm}_{\text{safe}}} &= \zeta_{p,\underline{l}} - \dot{d}_{\text{safe}}, & \text{otherwise.} \end{aligned}$$

To prevent collisions with obstacles, which are modeled with spheres, cylinders or planes, we use additional constraints. For instance, in case the robot must prevent collisions with a table, the point-static-plane constraint is suitable. Fig. 4.8 shows other application based on the same constraint, where four lateral planes are used to constrain the robot hand inside a desired region. This way, it is impossible for the robot hand to reach the target region from below because the hand cannot trespass any of the four lateral planes. These four additional constraints are written using first-order and second-order VFIs, respectively as

$$-\boldsymbol{J}_{p,n_{\underline{\pi}_i}} \dot{\boldsymbol{q}} \leq \eta \tilde{d}_{p,n_{\underline{\pi}_i}} + \zeta_{p,n_{\underline{\pi}_i},\text{safe}}, \tag{4.39}$$

$$-\boldsymbol{J}_{p,n_{\underline{\pi}_i}} \ddot{\boldsymbol{q}} \leq \beta_{p,n_{\underline{\pi}_i}} + \beta_{p,n_{\underline{\pi}_i},\text{res}}, \tag{4.40}$$

where $i \in \{1,2,3,4\}$, $\tilde{d}_{p,n_{\underline{\pi}_i}} = d_{p,n_{\underline{\pi}_i}} - d_{\pi,\text{safe}}$, with $d_{\pi,\text{safe}}$ being the safe distance to each plane, $\zeta_{p,n_{\underline{\pi}_i},\text{safe}} = \zeta_{p,n_{\underline{\pi}_i}} - \dot{d}_{\pi,\text{safe}}$, and $\beta_{p,n_{\underline{\pi}_i},\text{res}} = \dot{\zeta}_{p,n_{\underline{\pi}_i},\text{safe}} + \eta_1 \zeta_{p,n_{\underline{\pi}_i},\text{safe}}$.

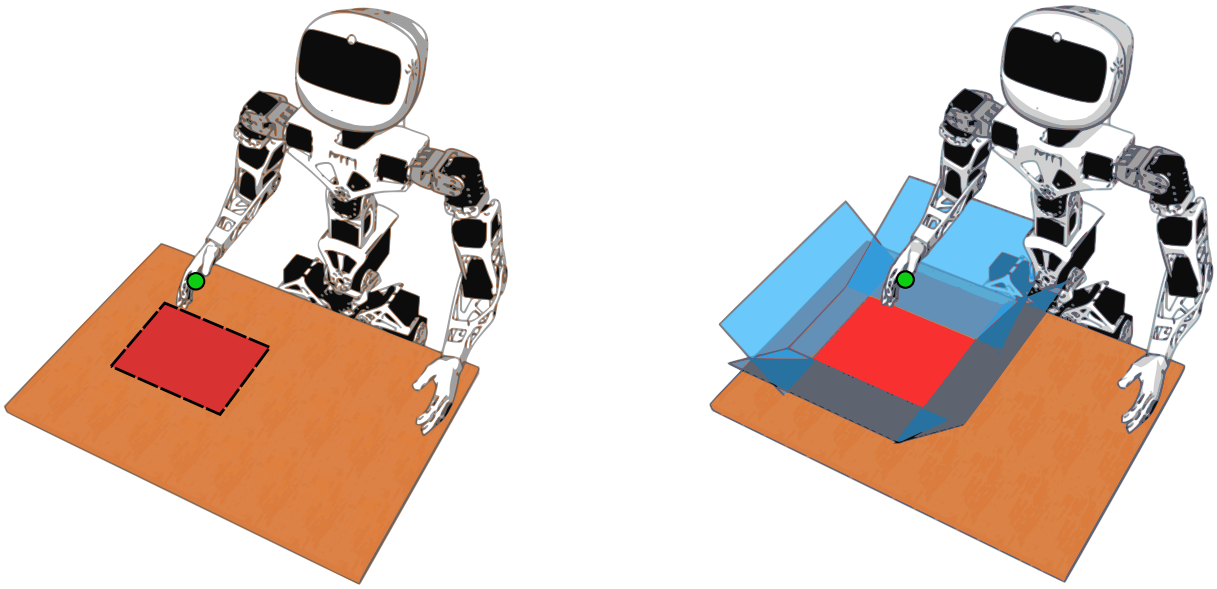

Figure 4.8: Application of the point-static-plane VFI. On the *left*, a target region is defined for the right hand. On the *right*, planes are used to constrain the right hand into an admissible space towards the target region.

4.4 Support Polygon Area Jacobian

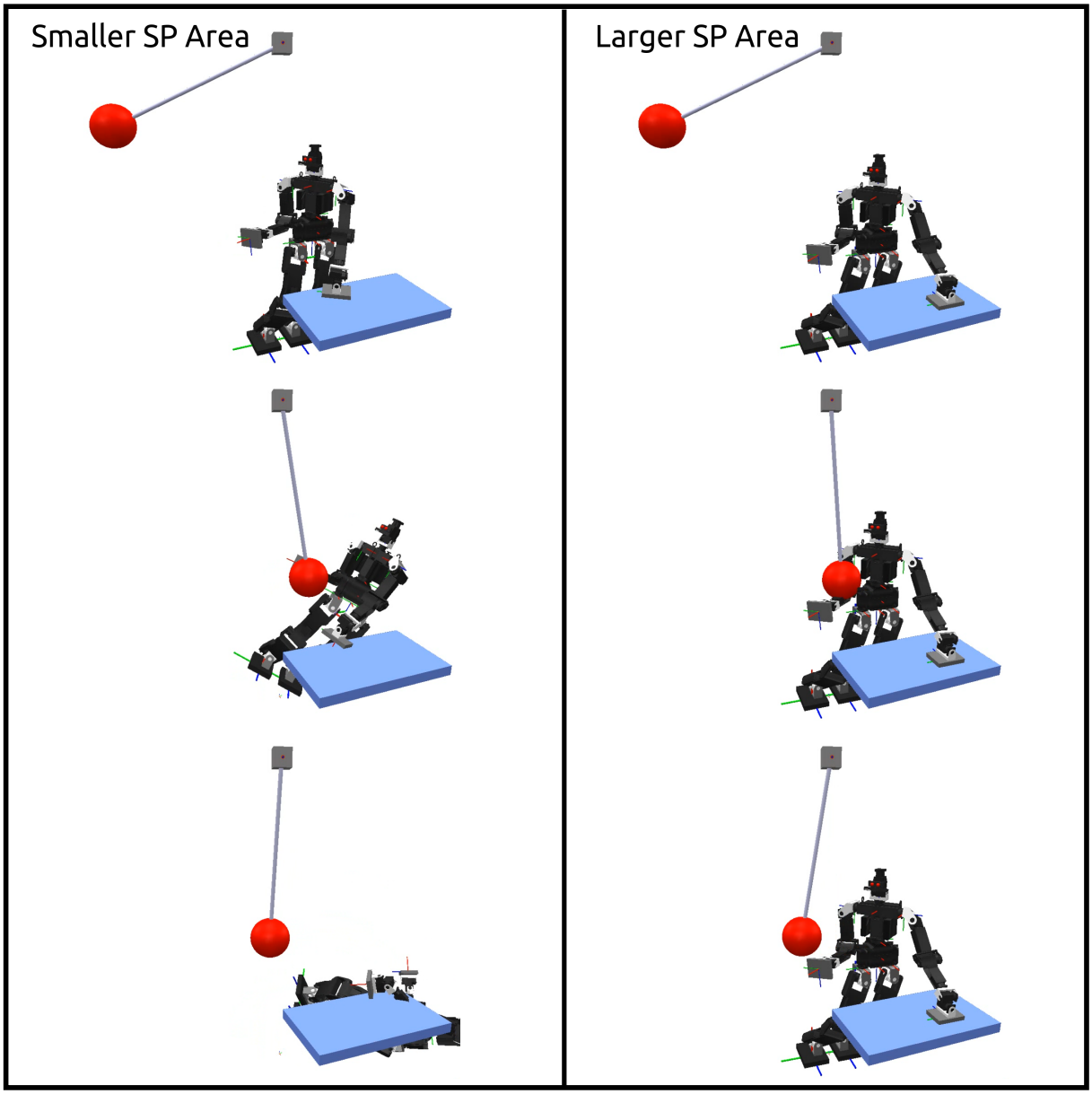

Figure 4.9: Application of the SP Area Jacobian. (Snapshots of the CoppeliaSim simulation). A pendulum is used to generate disturbances. On the *left*, the support polygon area is not maximized after the hand is placed on the table and the robot falls after the collision with the pendulum. On the *right*, the support polygon area is maximized. In this case, the balance is maintained.

In humanoid robot applications, it is fundamental to ensure the robot balance, while performing additional tasks. To accomplish this, some key concepts as the robot center of mass and the support polygon must be taken into account.

When the robot operates under relative low velocities and accelerations to neglect inertia, and assuming rigid contacts and suitable friction forces, the robot must be able to apply contacts on the environment to compensate the gravity forces without causing slip

and therefore, to achieve static equilibrium (Bretl & Lall, 2008). Under these conditions, the robot balance is ensured by keeping the projection of the center of mass inside the support polygon, which corresponds to the convex hull of the contact points (Bretl & Lall, 2008; Collette et al., 2007).

In some applications, it is convenient to increase the area of the support polygon in order to improve the robot reachability or to improve the robot balance in the presence of external disturbances. Consider, for instance, a humanoid robot performing a contact with a table using the left hand, as shown in Fig. 4.9. A pendulum collides with the robot and generates disturbances on its center of mass. Fig. 4.9 shows two cases with different SP. In the case where the SP has a smaller area, the projection of the center of mass does not remain inside the support polygon after the collision with the pendulum, and consequently, the robot falls. However, in the case where the SP has a larger area, the robot balance is maintained, precisely because of the larger area, which is enough to keep the projection of the center of mass inside the SP. This larger support polygon can be obtained, for instance, as the result of the maximization of the support polygon area.

This section shows the derivation of a new Jacobian related to the area of the support polygon. We assume static equilibrium, fixed feet, rigid planar contacts, and enough friction forces to keep the contacts fixed. First, we propose an approximation of the support polygon, which is a function of some contact points. We represent that SP approximation as SPA. Although this approximation is conservative, it simplifies the computation of the Jacobian. Finally, we compute the Jacobian related with the SPA.

Consider Fig. 4.10. When the robot performs contacts just using the feet, the SP region corresponds to the polygon that encapsulates both feet. Once the robot performs a contact with a table using the left hand, for instance, the SP area increases. Assuming a fixed contact, we can compute the new SP by means of an equivalent configuration that use all contacts on the same horizontal plane. This new SP is composed of the convex hull of the contact points, as shown in Fig. 4.10.

We select a contact point of the robot hand to build an approximation of the support polygon (SPA) such that the SPA polygon is inside the SP. In this way, the set of points that compose the SPA is a subset of the points that compose SP by construction. Therefore, the SPA corresponds to the convex hull of the points $\boldsymbol{p}_{i_{xy}} \in \mathbb{R}^2$, with $i \in \{a, b, c, d\}$, as shown in Fig. 4.11.

The SPA area is composed of the areas of two triangles, and it is given as follows

$$A = \frac{1}{2}\left(\underbrace{ed\sin\alpha}_{A_1} + \underbrace{ab\sin\phi_c}_{A_2}\right), \tag{4.41}$$

where $a = \left\|\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{d_{xy}}\right\|$, $b = \left\|\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{b_{xy}}\right\|$, $c = \left\|\boldsymbol{p}_{d_{xy}} - \boldsymbol{p}_{b_{xy}}\right\|$, $d = \left\|\boldsymbol{p}_{b_{xy}} - \boldsymbol{p}_{a_{xy}}\right\|$and $e = \left\|\boldsymbol{p}_{d_{xy}} - \boldsymbol{p}_{a_{xy}}\right\|$. Furthermore, $\boldsymbol{p}_{c_{xy}}$ is defined as the first two lines of $\mathrm{vec}_3\,\boldsymbol{p}_c$, with

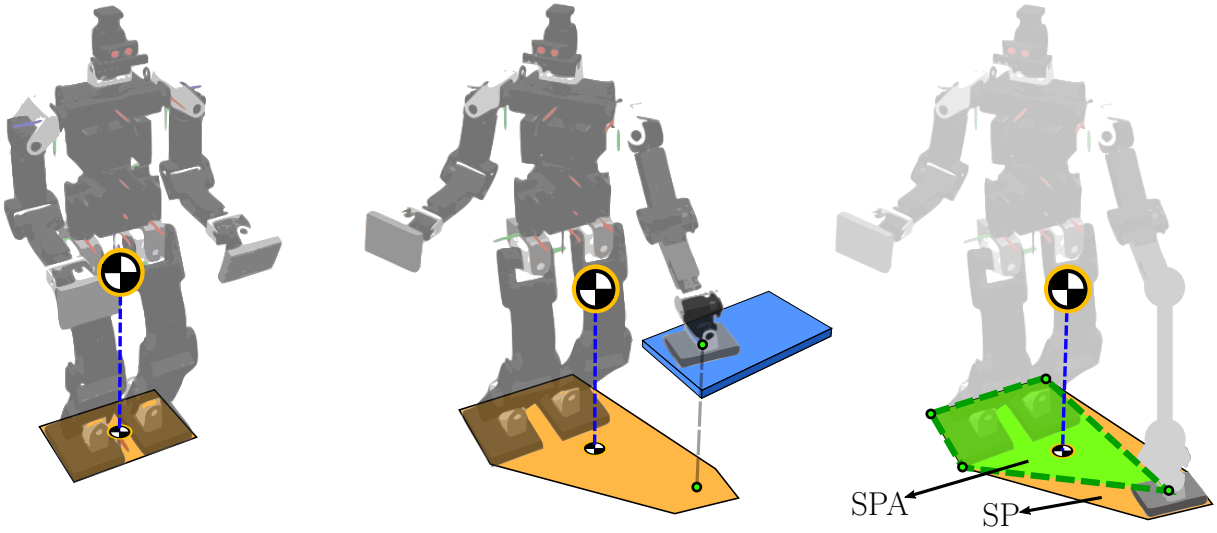

Figure 4.10: The support polygon (SP) and the support polygon approximation (SPA). On the *left*, the robot performs contact using the feet only. On the *middle*, a robot performs a contact on the table. On the *right*, the equivalent flat multi-contact configuration. The yellow region denotes the SP and the green one represents the SPA.

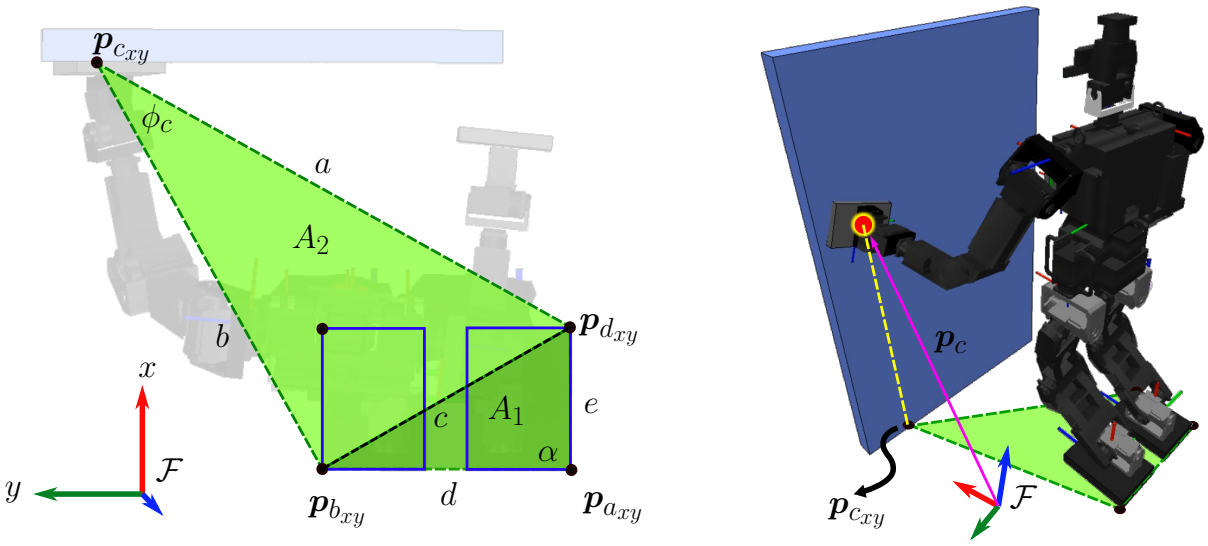

Figure 4.11: The humanoid robot performs a contact with the left hand on the wall. The green region is an approximation of the support polygon, which is composed of the convex hull of the contact points $\boldsymbol{p}_{i_{xy}} \in \mathbb{R}^2$, with $i \in \{a, b, c, d\}$.

$\boldsymbol{p}_c \in \mathbb{H}_p$ being the position of the hand contact.

Considering fixed feet contact, we have that $\dot{c} = \dot{e} = \dot{d} = 0\,\forall t$. Likewise, the goal is to find the relation between the changes of the SPA area and the joint velocities.[6] More specifically, the goal is to find Jacobian matrix $\boldsymbol{J}_A$ that satisfies

$$\dot{A} = \boldsymbol{J}_A \dot{\boldsymbol{q}},$$

The time derivative of (4.41) is

$$\dot{A} = \frac{1}{2}\left(\dot{a}b\sin\phi_c + a\dot{b}\sin\phi_c + ab\dot{\phi}_c\cos\phi_c\right). \tag{4.42}$$

Using the fact that

$$\dot{a} = \frac{1}{a}\left(\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{d_{xy}}\right)^T \dot{\boldsymbol{p}}_{c_{xy}}, \tag{4.43}$$

and

$$\dot{b} = \frac{1}{b}\left(\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{b_{xy}}\right)^T \dot{\boldsymbol{p}}_{c_{xy}}, \tag{4.44}$$

we rewrite (4.42) as follows

$$\dot{\boldsymbol{A}} = \underbrace{\frac{1}{2}\left(\frac{b\sin\phi_c}{a}\left(\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{d_{xy}}\right)^T + \frac{a\sin\phi_c}{b}\left(\boldsymbol{p}_{c_{xy}} - \boldsymbol{p}_{b_{xy}}\right)^T\right)}_{\boldsymbol{J}_{A_1}}\dot{\boldsymbol{p}}_{c_{xy}} + \frac{1}{2}ab\cos\left(\phi_c\right)\dot{\phi}_c. \tag{4.45}$$

Using the law of cosines, $c^2 = a^2 + b^2 - 2ab\cos\phi_c$, the angle ϕ_c is given as follows

$$\phi_c = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right). \tag{4.46}$$

Let $u \triangleq \bar{\alpha}\bar{\beta}^{-1}$, with $\bar{\alpha} \triangleq a^2 + b^2 - c^2$ and $\bar{\beta} \triangleq 2ab$. Then, the time derivative of (4.46) is

$$\dot{\phi}_c = -\frac{\dot{u}}{\sqrt{1-u^2}}, \tag{4.47}$$

where $\dot{u} = \left(\dot{\bar{\alpha}}\bar{\beta} - \dot{\bar{\beta}}\bar{\alpha}\right)\bar{\beta}^{-2}$. Since $\dot{\bar{\alpha}} = 2a\dot{a} + 2b\dot{b}$[7] and $\dot{\bar{\beta}} = 2\dot{a}b + 2a\dot{b}$, we rewrite $\dot{u}$ as

$$\dot{u} = \left(\frac{c^2 + a^2 - b^2}{2a^2b}\right)\dot{a} + \left(\frac{c^2 - a^2 + b^2}{2ab^2}\right)\dot{b}. \tag{4.48}$$

[6] In Fig. 4.11, the feet plane is the X-Y plane.

[7] Notice that $c = \left\|\boldsymbol{p}_{d_{xy}} - \boldsymbol{p}_{b_{xy}}\right\|$, where $\boldsymbol{p}_{d_{xy}}$ and $\boldsymbol{p}_{b_{xy}}$ are fixed feet points. Therefore $\dot{c} = 0\ \forall t$.

Using (4.43) and (4.44) in (4.48), we obtain

$$\dot{u} = \underbrace{\left(\left(\frac{c^2+a^2-b^2}{2a^3b}\right)\left(\boldsymbol{p}_{c_{xy}}-\boldsymbol{p}_{d_{xy}}\right)^T + \left(\frac{c^2-a^2+b^2}{2ab^3}\right)\left(\boldsymbol{p}_{c_{xy}}-\boldsymbol{p}_{b_{xy}}\right)^T\right)}_{\boldsymbol{J}_u}\dot{\boldsymbol{p}}_{c_{xy}}, \tag{4.49}$$

where $a, b > 0$ are ensured by imposing the desired task dynamics that maximize A_2.

Remark 4.1. The equation (4.47) is always non singular. To illustrate this, consider the singular case $u = 1 \implies \bar{\alpha} = \bar{\beta}$. Then, we have that

$$a^2 + b^2 - c^2 = 2ab \implies ((a+c)-b)\,(a-(b+c)) = 0, \tag{4.50}$$

which implies that $a + c = b$ or $b + c = a$. However, for non-degenerate triangles, which is our case because we assume that $\boldsymbol{p}_{b_{xy}}$, $\boldsymbol{p}_{c_{xy}}$, and $\boldsymbol{p}_{d_{xy}}$ are not collinear, the triangle inequality implies $a + c > b$ and $b + c > a$.

Using (4.49) in (4.47), we have

$$\dot{\phi}_c = -\frac{1}{\sqrt{1-u^2}}\boldsymbol{J}_u\dot{\boldsymbol{p}}_{c_{xy}}. \tag{4.51}$$

Furthermore, using (4.51) in (4.45), we obtain the SPA-area Jacobian

$$\dot{A} = \underbrace{(\boldsymbol{J}_{A_1} + \boldsymbol{J}_{A_2})}_{\bar{\boldsymbol{J}}_A}\dot{\boldsymbol{p}}_{c_{xy}}, \tag{4.52}$$

where

$$\boldsymbol{J}_{A_2} \triangleq -\frac{ab\cos\phi_c}{2\sqrt{1-u^2}}\boldsymbol{J}_u.$$

Since $\dot{\boldsymbol{p}}_{c_{xy}}$ is defined as the first two lines of $\mathrm{vec}_3\,\boldsymbol{p}_c$, the Jacobian $\boldsymbol{J}_{c_{xy}}$ correspond to the first two lines of the position Jacobian $\boldsymbol{J}_p$ that respects the relation $\mathrm{vec}_3\,\dot{\boldsymbol{p}}_c = \boldsymbol{J}_p\dot{\boldsymbol{q}}$. Therefore, we have

$$\dot{\boldsymbol{p}}_{c_{xy}} = \boldsymbol{J}_{c_{xy}}\dot{\boldsymbol{q}}. \tag{4.53}$$

Finally, we rewrite (4.52) in terms of the joints velocities using (4.53), as follows

$$\dot{A} = \underbrace{\bar{\boldsymbol{J}}_A\boldsymbol{J}_{c_{xy}}}_{\boldsymbol{J}_A}\dot{\boldsymbol{q}}. \tag{4.54}$$

4.5 Multi-Contact Applications

Multi-contact control strategies have been intensively explored over the last twenty years. One of the reasons is the amount of potential applications for humanoid robots in cluttered scenarios. In some specific cases, this strategy allows the use of multiple contacts with the environment to increase the reachability or increase the support polygon of the robot to improve static balance. For instance, consider the Fig. 4.12. A humanoid is required to touch a ball using its right hand. However, since the projection of the robot center of mass must be maintained inside the support polygon to ensure the robot balance, its workspace reachability is limited, and therefore, the robot is not able to accomplish the task. An alternative is to approach the ball before attempting to touch it. However, this is not always possible since the robot must avoid collisions with possible obstacles, including the table. A better solution in this example is to perform first a contact with the table using the left hand. This improves the balance support polygon, allowing the robot to accomplish the task. However, in some cases, performing the contact is not enough to reach the ball. In those cases, the maximization of the support polygon area could help, since the robot reachability could be potentially increased after that maximization. If it is not possible to reach the ball yet, a last resource can be applied. One can relax the contact constraint in order to allow a sliding contact with the table surface, whereas the robot tries to reach the ball simultaneously. In total, there are four attempts to accomplish the main task. Fig. 4.13 summarizes the proposed multi-contact strategy, which is composed of six states.

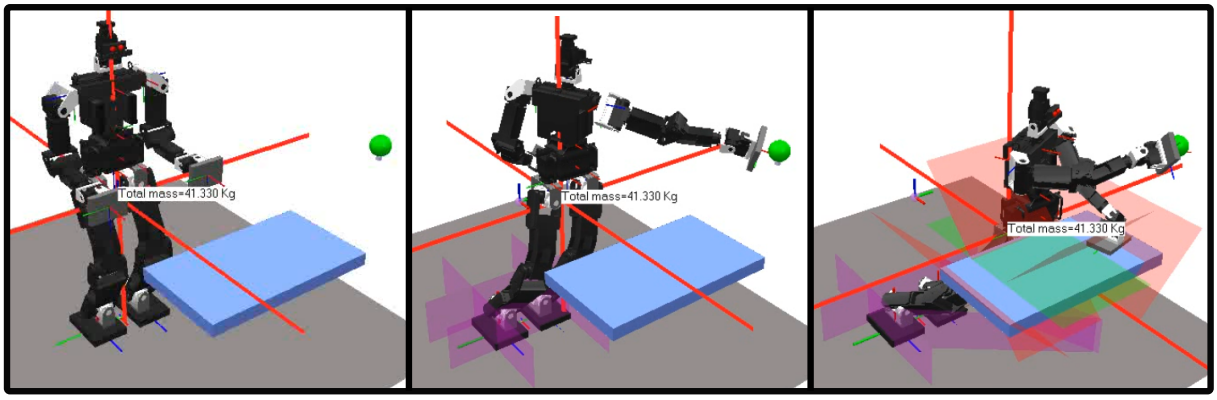

Figure 4.12: Example of a multi-contact task. The goal is to control the right hand of a humanoid robot to touch the green ball. The red lines' intersection represents the robot center of mass. On the *left*, the initial configuration of the robot. On the *middle,* four planes constraints ensure the robot balance by keeping the projection of the center of mass inside the support polygon. However, in this case, the robot does not reach the green ball. On the *right*, the support polygon area is increased by performing a contact on the table with the left hand. Four additional planes define the target left-hand-contact region. This allows reaching the green ball with the right hand.

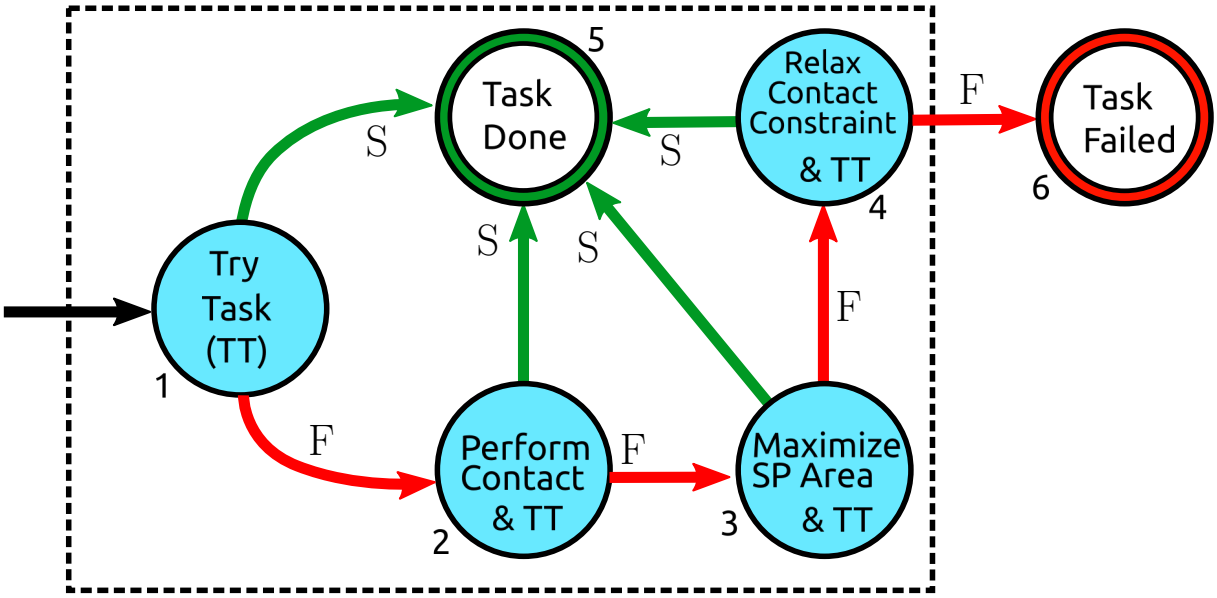

Figure 4.13: State transition diagram for the multi-contact manipulation task algorithm. The green arrows represent success (S) events. The red arrows are fails (F) events. 1) The robot tries to accomplish the main task (TT). 2) The robot performs a contact with the environment and after, tries (TT). 3) The robot maximizes the support polygon and after, tries (TT). 4) The hand-contact constraint is relaxed allowing a sliding contact on the surface and simultaneously tries (TT). 5) The task is fulfilled. 6) The robot fails to accomplish the main task.

4.5.1 Contact Task

A contact task requires both to align and to approach the end-effector to the contact surface. This can be defined using a time-varying trajectory specifying the end-effector pose at all times. However, the idea is to use relaxed tasks to release some robot DOF for additional tasks, as discussed in section 4.2. Instead of defining a specific desired end-effector pose, we define a target region using the constraints 4.39. In addition, we attach a plane $\underline{\boldsymbol{\pi}} \triangleq \underline{\boldsymbol{\pi}}(\boldsymbol{q}) = \boldsymbol{n}_{\pi} + \varepsilon d_{\pi}$ to the end-effector and a plane $\underline{\boldsymbol{\pi}}_d = \boldsymbol{n}_{\pi_d} + \varepsilon d_{\pi_d}$ to the contact surface, as shown in Fig. 4.14. The task error is defined as

$$\tilde{\underline{\boldsymbol{\pi}}} \triangleq \underline{\boldsymbol{\pi}} - \underline{\boldsymbol{\pi}}_d. \tag{4.55}$$

Furthermore, the plane velocities are related with the joints velocities by means of the plane Jacobian (Marinho et al., 2019) as follows

$$\mathrm{vec}_8\, \dot{\underline{\boldsymbol{\pi}}} = \boldsymbol{J}_{\pi} \dot{\boldsymbol{q}}. \tag{4.56}$$

In this way, when $\tilde{\underline{\boldsymbol{\pi}}} \to \boldsymbol{0}$ then both the error of the orientation between the end-effector and the table and the distance between them go to zero, and consequently, the robot performs the contact successfully.

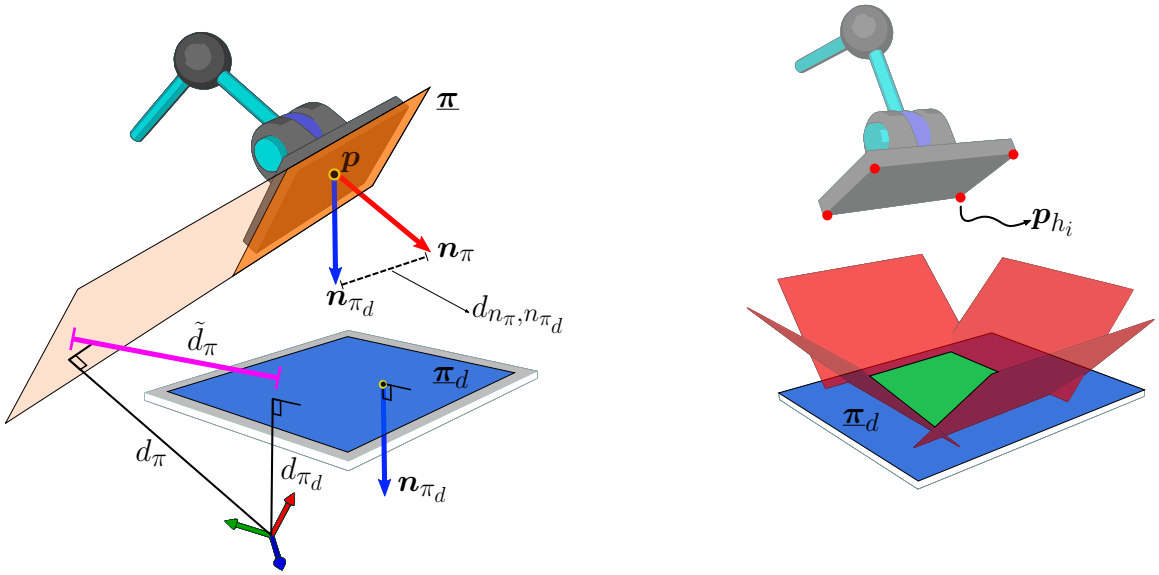

Figure 4.14: Task definition using geometric primitives. The goal is to control the robot hand to perform a contact on the table. On the *left*, a plane $\underline{\boldsymbol{\pi}} = \boldsymbol{n}_\pi + \varepsilon d_\pi$ is attached to the robot hand. Furthermore, a target plane (displayed in blue) $\underline{\boldsymbol{\pi}}_d = \boldsymbol{n}_{\pi_d} + \varepsilon d_{\pi_d}$ is attached to the table. On the *right,* four planes (displayed in red) define the target region (green zone). In addition, four point-to-plane constraints (red points $\boldsymbol{p}_{h_i}, i \in \{1,2,3,4\}$ and the target plane $\underline{\boldsymbol{\pi}}_d$) ensure avoidance collision between the hand and the table.

To prevent collisions between the end-effector and the surface contact, we use four point-to-plane constraints. The points are attached to the hand-surface vertex, as shown in Fig. 4.14. In this way, the only way to perform the contact is with the end-effector aligned to the target region. These four additional constraints are written as

$$-\boldsymbol{J}_{p_{h_i},n_{\pi_d}}\dot{\boldsymbol{q}} \leq \eta \tilde{d}_{p_{h_i},n_{\pi_d}}, \tag{4.57}$$

where $i \in \{1,2,3,4\}$ and $\tilde{d}_{p_{h_i},n_{\pi_d}} = d_{p_{h_i},n_{\pi_d}} - d_{p_h,\text{safe}}$, with $d_{p_{h_i},\text{safe}}$ being the safe distance to each point.

In addition, to keep stationary the robot's feet on the ground, we use the cooperative dual task space framework (Adorno, 2011), as shown in Fig. 4.15. In this framework, both serial kinematic chains (in this case the legs) are encapsulated by means of the cooperative variables: the absolute pose $\underline{\boldsymbol{x}}_{\text{abs}}$ and the relative pose $\underline{\boldsymbol{x}}_{\text{rel}}$. The former represents the pose of a frame located in the middle of both legs. The latter represents the pose of the left leg with respect to the right one. To keep both feet ideally stationary on the ground, we must ensure (Fonseca & Adorno, 2016) $\dot{\underline{\boldsymbol{x}}}_{\text{rel}} = 0$ to prevent changes in relative feet pose, and $\mathcal{P}\left(\dot{\underline{\boldsymbol{x}}}_{\text{abs}}\right) = 0$ to prevent changes in the orientation of the absolute frame $\mathcal{F}_{\text{abs}}$. These constraints are written as

$$\boldsymbol{J}_{\text{rel}}\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{4.58}$$

$$\boldsymbol{J}_{\mathcal{P}_{\text{abs}}}\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{4.59}$$

where $\boldsymbol{J}_{\text{rel}}$ and $\boldsymbol{J}_{\text{abs}}$ are the cooperative Jacobians that respect the relations $\text{vec}_8\,\dot{\underline{\boldsymbol{x}}}_{\text{rel}} = \boldsymbol{J}_{\text{rel}}\dot{\boldsymbol{q}}$ and $\text{vec}_8\,\dot{\underline{\boldsymbol{x}}}_{\text{abs}} = \boldsymbol{J}_{\text{abs}}\dot{\boldsymbol{q}}$. Furthermore, $\boldsymbol{J}_{\text{abs}} = \begin{bmatrix} \boldsymbol{J}_{\mathcal{P}_{\text{abs}}}^T & \boldsymbol{J}_{\mathcal{D}_{\text{abs}}}^T \end{bmatrix}^T$.

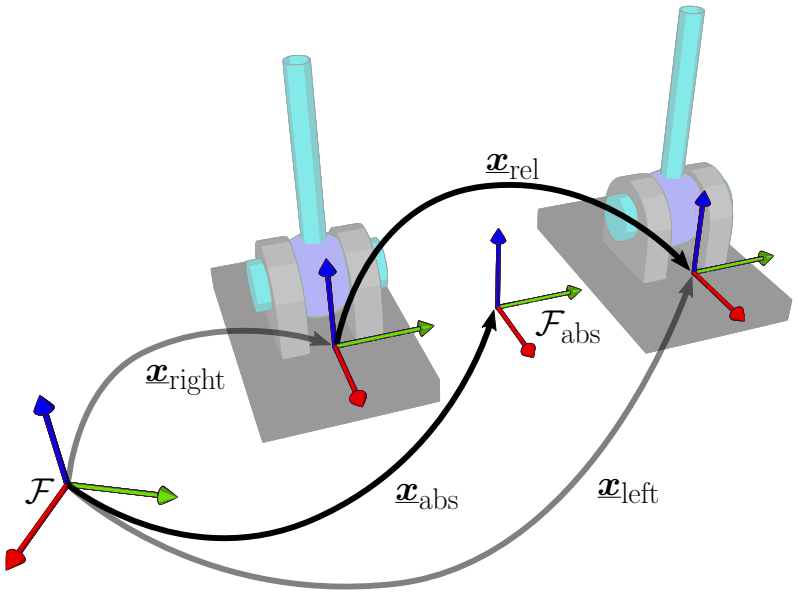

Figure 4.15: Cooperative variables: The relative pose $\underline{\boldsymbol{x}}_r$ and the absolute pose $\underline{\boldsymbol{x}}_a$ related to the humanoid's feet. The pose of the left and right leg are $\underline{\boldsymbol{x}}_{\text{left}}$ and $\underline{\boldsymbol{x}}_{\text{right}}$, respectively.

In some cases, it is required to keep the hand contact fixed whereas the robot perform a given task, as shown in Fig. 4.16. For instance, state 2 in Fig. 4.13 requires to perform the main task (TT) keeping the hand contact fixed, whose pose is $\underline{\boldsymbol{x}}_c$. To ensure this, we use the constraint

$$\boldsymbol{J}_c\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{4.60}$$

where the Jacobian matrix $\boldsymbol{J}_c$ respects the relation $\text{vec}_8\,\dot{\underline{\boldsymbol{x}}}_c = \boldsymbol{J}_c\dot{\boldsymbol{q}}$. The constraint (4.60) keeps both the position and orientation of the end-effector fixed, and consequently, keeps the hand contact fixed.

To relax the constraint (4.60), as required by state 4 in Fig. 4.13, we use a plane-to-plane constraint. The idea is to limit the robot hand movement to the surface contact. This allows sliding the robot hand on the surface but always maintaining the contact, as shown in Fig. 4.16. This constraint is written as

$$\boldsymbol{J}_\pi\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{4.61}$$

where the plane Jacobian $\boldsymbol{J}_\pi$ respects the relation $\text{vec}_8\,\dot{\underline{\boldsymbol{\pi}}}_c = \boldsymbol{J}_\pi\dot{\boldsymbol{q}}$, with $\underline{\boldsymbol{\pi}}_c$ being a plane attached to the robot hand once the robot perform the contact.

Notice that the discretization effects can introduce an eventual drift. However, assuming physical constraints, as for instance, enough friction forces, those effects can be minimized.

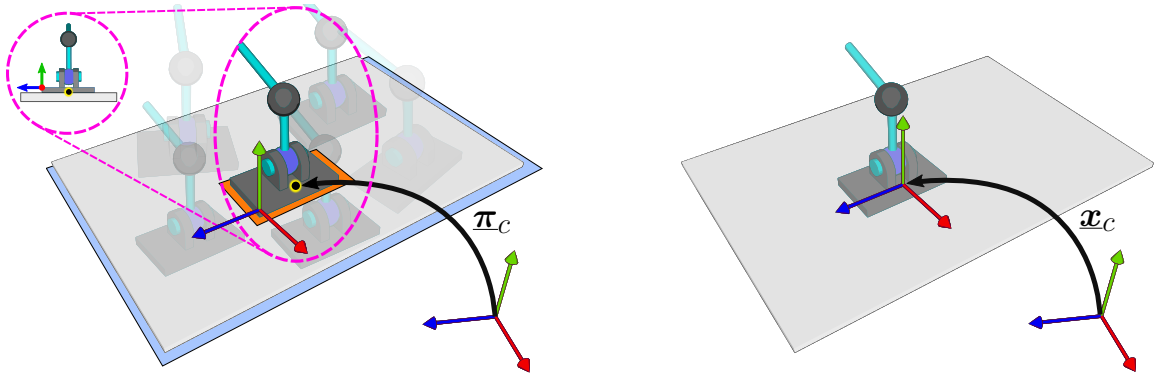

Figure 4.16: Fixed and sliding contacts. On the *left*, the plane $\underline{\boldsymbol{\pi}}_c$ is attached to the robot hand once the contact is performed. The constraint $\dot{\underline{\boldsymbol{\pi}}}_c = 0$ allows sliding the robot hand on the surface. On the *right*, $\underline{\boldsymbol{x}}_c$ represents the pose of the robot hand once the contact is performed. The constraint $\dot{\underline{\boldsymbol{x}}}_c = 0$ keeps the end-effector pose fixed, and consequently, the contact is maintained fixed.

4.5.2 Center of Mass Constraints

As shown in Fig. 4.17, the point-to-plane constraint can be used to maintain the robot the projection of the center of mass $p_{\text{com}_{x,y}}$ inside the support polygon. These constraints are written as

$$\boldsymbol{J}_{p_{\text{com}_{x,y}}, n_{\underline{\boldsymbol{\pi}}_{c_i}}} \dot{\boldsymbol{q}} \leq -\eta \tilde{d}_{p_{\text{com}_{x,y}}, n_{\underline{\boldsymbol{\pi}}_{c_i}}}, \tag{4.62}$$

where $i = \{1, 2, 3, 4\}$ and $\tilde{d}_{p_{\text{com}_{x,y}}, n_{\underline{\boldsymbol{\pi}}_{c_i}}} = d_{p_{\text{com}_{x,y}}, n_{\underline{\boldsymbol{\pi}}_{c_i}}} - d_{\pi_c,\text{safe}}$, with $d_{\pi_c,\text{safe}}$ being the safe distance to each plane.

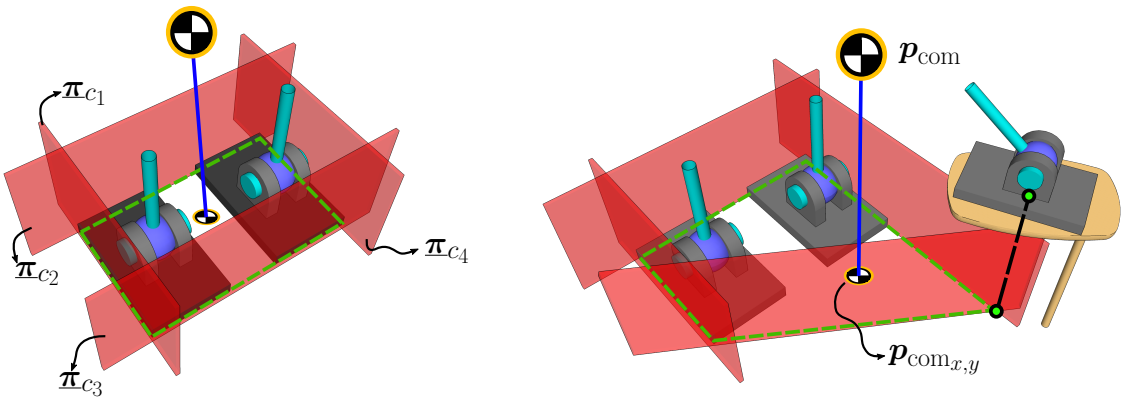

Figure 4.17: Center of mass plane constraints. The goal is to keep the projection of the robot center of mass inside the support polygon, which is denoted by the green dashed line. Four planes $\underline{\boldsymbol{\pi}}_{c_i}, i \in \{1, 2, 3, 4\}$ On the *left*, the robot performs feet contacts only. On the *right*, a third contact is performed with the hand. The point-to-plane constraints are updated according the multi-contact configuration.

4.6 Conclusions

This chapter presented one of the contributions of this thesis. First, the chapter extended the VFIs framework to second order kinematics (SOVFIs) and presented a formal proof that the method ensures collision avoidance. This ensures collision avoidance between pairs of geometrical primitives by means of second order differential inequalities taking into account their positions, velocities and accelerations involved. The main motivation of the SOVFIs framework is to enable applications that use the robot dynamics by means of the relationship between joint torques and joint accelerations in the Euler-Lagrange equations, which are derived in Chapter 4 using dual quaternion algebra.

Second, the chapter showed a novel singularity-free conic constraint to limit the angle between two Plücker lines. This new constraint is used to prevent the violation of joint limits or to avoid undesired end-effector orientations.

Next, the chapter showed a novel Jacobian related with the support polygon area of a humanoid robot. This enables tasks as maximization of the SP area, which can potentially increase the robot's reachability and the robot safety in terms of its balance. The SP Jacobian is computed by means of an approximation of the support polygon (SPA) as a function of some contact points. Although this approximation is conservative, it simplifies the computation of the Jacobian. This strategy, however, has some limitations. For instance, the frictions forces are neglected by assuming enough friction in each contact, and static and dynamic friction coefficients are not taken into account in fixed or sliding contacts. Furthermore, only flat surfaces of the robot (planar feet and hands) and flat surfaces of the environment (e.g., tables, walls, etc.) are taken into account.

Finally, the chapter presented a brief motivation of multi-contact applications and proposed a strategy to enable it based on VFIs framework. Likewise, the chapter showed the tasks definitions to perform contacts on flat surfaces and the constraints to maintain the robot balance. However, this strategy is initially defined to first-order kinematics applications only. In addition, the strategy assumes enough robot DoF to execute all desired tasks.

5

Dynamic Modeling in Dual Quaternion Algebra

This chapter shows another contributions of this thesis. First, we rewrite the Gauss's Principle of Least Constraint for articulated bodies, similar to the formulation proposed by Wieber (2005), but using dual quaternion algebra. This leads to the Euler-Lagrange dynamic equation of a robotic system. Dual quaternion algebra allows a compact representation of the linear and angular accelerations. The connections between the Gauss's Principle of Least Constraint and the Gibbs-Appell and Kane's equations are shown. Furthermore, additional constraints are explored and the dynamic model of a humanoid robot is presented. Finally, the section is closed with a cost comparison between a proposed algorithm for obtaining the Euler-Lagrange dynamic equation for a serial manipulator and their classic counterparts.

5.1 Motivation

The SOVFIs framework, presented in the previous chapter, operates in the second order kinematics level and therefore, allows using it in robot dynamics applications. Consider, for instance, a robot commanded by joint torques, whose goal is to perform some set of desired tasks subject to SOVFI constraints. First, we compute the robot kinematics and the VFIs using geometric primitives. Second, we obtain the joint accelerations that minimize the task error and respect all constraints using mathematical programming, as

shown in Section. 3.1. Then, we compute the joint torques using the Euler Lagrange dynamic equation.

The robot kinematic model is obtained using dual quaternion algebra. This formalism enables a high-level mathematical abstraction and provides a straightforward and compact representation of both rigid motions and geometric primitives (e.g., points, lines and planes). Furthermore, in the context of robot kinematics, dual quaternion algebra allows modeling a variety of different robots using the same systematic procedure, thanks to their strong algebraic properties (Adorno, 2017)(Perez & McCarthy, 2004; Adorno, 2011; Gouasmi, 2012; Cohen & Shoham, 2016; Özgür & Mezouar, 2016; Kong, 2017; Dantam, 2020). Therefore, a question that rises at the moment of describing the robot dynamics is how to compute it using dual quaternions?

That question motivates the derivation of the of the Gauss's Principle of Least Constraint using the tools from dual quaternion algebra.

This principle has been used in robotics to describe the dynamics of robot manipulators (Bruyninckx & Khatib, 2000) and rigid body simulations (Redon et al., 2002). Wieber (2006) uses the GPLC to derive the analytic expression of the Lagrangian dynamics of a humanoid robot. Bouyarmane & Kheddar (2012) extend Wieber's work by handling arbitrary parameterization of free-floating-base mechanisms. This allows using rotation matrices or unit quaternions to represent the free-floating-base orientations. In this section, we rewrite the GPLC for articulated bodies, similar to Wieber's formulation (Wieber, 2006), but using dual quaternion algebra. This allows a more compact and unified representation than the one by Bouyarmane & Kheddar (2012).

5.2 Gauss's Principle of Least Constraint

The GPLC (Kalaba & Udwadia, 1993) is a differential variational principle, equivalent to the D'Alembert one, that is based on the variation of the acceleration. For a system composed of n bodies, it can be stated as the least-squares minimization problem

$$\min \quad \sum_{i=1}^{n} \frac{1}{2} \left(\boldsymbol{a}_{c_i} - \bar{\boldsymbol{a}}_{c_i}\right)^T \boldsymbol{\Psi}_{c_i} \left(\boldsymbol{a}_{c_i} - \bar{\boldsymbol{a}}_{c_i}\right) \quad , \tag{5.1}$$

where $\boldsymbol{a}_{c_i}$ and $\bar{\boldsymbol{a}}_{c_i}$ are the accelerations of the center of mass of the ith rigid body under constraints and without constraints, respectively. Furthermore, $\boldsymbol{\Psi}_{c_i} \triangleq \boldsymbol{\Psi}_{c_i}\left(\mathbb{I}_{c_i}, m_i\right)$ encapsulates the inertial parameters of the ith rigid body, such as the inertia tensor $\mathbb{I}_i \in \mathbb{R}^{3\times3}$ and the mass m_i.

In the next subsection, we write the constrained accelerations as a linear function of the vector of joint velocities and joint accelerations. This allows solving (5.1) for the joint accelerations and, therefore, additional constraints can be directly imposed in the

optimization formulation.

5.2.1 Constrained acceleration

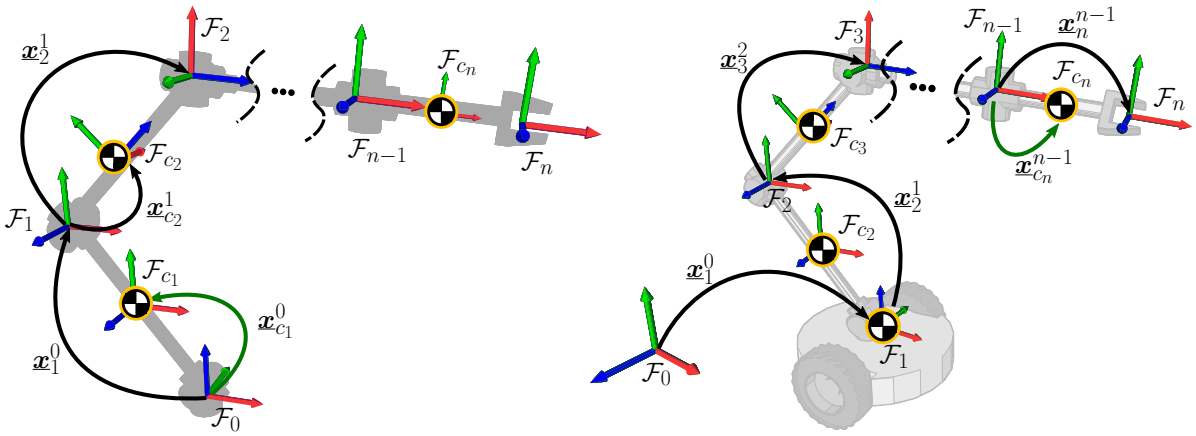

Figure 5.1: On the *left*, a n-DOF robot manipulator. On the *right*, a n-DOF nonholonomic mobile manipulator.

Consider the robotic system in Fig. 5.1. The robot is composed of rigid bodies that are constrained[1] to one another by joints. To express the twist $\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}$ of the ith center of mass explicitly as a linear combination between its Jacobian $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ and the vector of joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, we use the operators $\operatorname{vec}_8 : \mathcal{H} \to \mathbb{R}^8$, which maps the coefficients of a dual quaternion into an eight-dimensional vector,[2] and $\overset{+}{\boldsymbol{H}}_8 : \mathcal{H} \to \mathbb{R}^{8\times 8}$, such that $\operatorname{vec}_8(\underline{\boldsymbol{h}}_1\underline{\boldsymbol{h}}_2) = \overset{+}{\boldsymbol{H}}_8(\underline{\boldsymbol{h}}_1)\operatorname{vec}_8\underline{\boldsymbol{h}}_2$ Adorno (2011). Therefore, from (A.19) we obtain $\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = 2\left(\underline{\boldsymbol{x}}_{c_i}^0\right)^* \dot{\underline{\boldsymbol{x}}}_{c_i}^0$, which implies $\operatorname{vec}_8 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = 2\overset{+}{\boldsymbol{H}}_8(\underline{\boldsymbol{x}}_0^{c_i})\operatorname{vec}_8 \dot{\underline{\boldsymbol{x}}}_{c_i}^0$.

Because $\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} \in \mathcal{H}_p$, the first and fifth elements of $\operatorname{vec}_8 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}$ equal zero, thus we also use the operator $\operatorname{vec}_6 : \mathcal{H}_p \to \mathbb{R}^6$ such that $\operatorname{vec}_6 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} \triangleq \bar{\boldsymbol{I}} \operatorname{vec}_8 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}$, where

$$\bar{\boldsymbol{I}} \triangleq \begin{bmatrix} \boldsymbol{0}_{3\times 1} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times 1} & \boldsymbol{I}_3 \end{bmatrix},$$

with $\boldsymbol{I}_3 \in \mathbb{R}^{3\times 3}$ being the identity matrix and $\boldsymbol{0}_{m\times n} \in \mathbb{R}^{m\times n}$ being a matrix of zeros. Moreover, $\operatorname{vec}_8 \dot{\underline{\boldsymbol{x}}}_{c_i}^0 = \boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0}\dot{\boldsymbol{q}}_i$, with $\dot{\boldsymbol{q}}_i = \begin{bmatrix} \dot{q}_1 & \cdots & \dot{q}_i \end{bmatrix}^T$, and $\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0} \in \mathbb{R}^{8\times i}$ is the Jacobian matrix that is obtained algebraically (Adorno, 2011). Hence,

$$\boldsymbol{\nu}_{c_i} \triangleq \operatorname{vec}_6 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = \underbrace{\begin{bmatrix} \bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} & \boldsymbol{0}_{6\times(n-i)} \end{bmatrix}}_{\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}} \dot{\boldsymbol{q}}, \tag{5.2}$$

[1] In this case, the constraints are holonomic. However, nonholonomic constraints can be taken into account as well.

[2] Given $\underline{\boldsymbol{h}} = h_1 + \hat{\imath}h_2 + \hat{\jmath}h_3 + \hat{k}h_4 + \varepsilon\left(h_5 + \hat{\imath}h_6 + \hat{\jmath}h_7 + \hat{k}h_8\right)$, $\operatorname{vec}_8 \underline{\boldsymbol{h}} = \begin{bmatrix} h_1 & \cdots & h_8 \end{bmatrix}^T$.

where $\bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} = 2\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right)\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0} \in \mathbb{R}^{6\times i}$.

Finally, the constrained acceleration of the ith center of mass is given by

$$\boldsymbol{a}_{c_i} \triangleq \operatorname{vec}_6 \dot{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i} = \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\boldsymbol{q}}. \tag{5.3}$$

5.2.2 Unconstrained acceleration

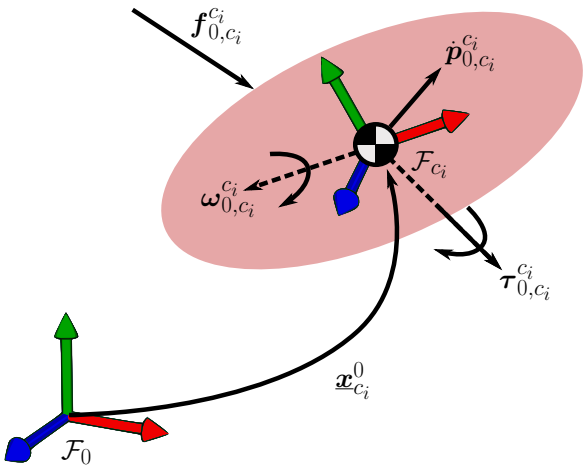

Figure 5.2: Linear and angular momentum acting on a rigid ith body. The unit dual quaternion $\underline{\boldsymbol{x}}_{c_i}^0$ represents its pose with respect to the body frame $\mathcal{F}_{c_i}$.

Consider the Fig. 5.2, where $\underline{\boldsymbol{x}}_{c_i}^0 = \boldsymbol{r}_{c_i}^0 + (1/2)\varepsilon \boldsymbol{p}_{0,c_i}^0 \boldsymbol{r}_{c_i}^0$ represents the rigid motion from $\mathcal{F}_0$ to $\mathcal{F}_{c_i}$. The twist $\bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i}$,[3] at $\mathcal{F}_{c_i}$ of the ith body under no constraints, is[4]

$$\bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i} = \boldsymbol{\omega}_{0,c_i}^{c_i} + \varepsilon \dot{\boldsymbol{p}}_{0,c_i}^{c_i} \implies \operatorname{vec}_6 \bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i} = \begin{bmatrix} \operatorname{vec}_3 \boldsymbol{\omega}_{0,c_i}^{c_i} \\ \operatorname{vec}_3 \dot{\boldsymbol{p}}_{0,c_i}^{c_i} \end{bmatrix}, \tag{5.4}$$

where $\operatorname{vec}_3 : \mathbb{H}_p \to \mathbb{R}^3$ such that $\operatorname{vec}_3(a\hat{\imath} + b\hat{\jmath} + c\hat{k}) = \begin{bmatrix} a & b & c \end{bmatrix}^T$.

The unconstrained acceleration $\dot{\bar{\underline{\boldsymbol{\xi}}}}_{0,c_i}^{c_i}$ is computed by taking the time derivative of the twist $\bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i} = \underline{\boldsymbol{x}}_0^{c_i} \bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{0} \underline{\boldsymbol{x}}_{c_i}^0$, and is given explicitly as[5]

$$\bar{\boldsymbol{a}}_{c_i} \triangleq \operatorname{vec}_6 \dot{\bar{\underline{\boldsymbol{\xi}}}}_{0,c_i}^{c_i} = \begin{bmatrix} \operatorname{vec}_3 \dot{\boldsymbol{\omega}}_{0,c_i}^{c_i} \\ \operatorname{vec}_3 \left(\ddot{\boldsymbol{p}}_{0,c_i}^{c_i} + \dot{\boldsymbol{p}}_{0,c_i}^{c_i} \times \boldsymbol{\omega}_{0,c_i}^{c_i} \right) \end{bmatrix}. \tag{5.5}$$

Although the final form of (5.3) and (5.5) are essentially the same, the latter does not depend on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$, precisely because it is unconstrained.

[3] We recall that $\bar{\underline{\boldsymbol{\xi}}}_{0,c_i}^{c_i}$ is the twist of body frame $\mathcal{F}_{c_i}$ with respect to frame $\mathcal{F}_0$, expressed in the body frame $\mathcal{F}_{c_i}$.

[4] See (A.1).

[5] See (A.2).

5.2.3 Euler-Lagrange equations

Let $\mathcal{G}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \sum_{i=1}^{n} \frac{1}{2} (\boldsymbol{a}_{c_i} - \bar{\boldsymbol{a}}_{c_i})^T \boldsymbol{\Psi}_{c_i} (\boldsymbol{a}_{c_i} - \bar{\boldsymbol{a}}_{c_i})$, in which $\boldsymbol{a}_{c_i}$ and $\bar{\boldsymbol{a}}_{c_i}$ are given by (5.3) and (5.5), and the generalized inertia tensor $\boldsymbol{\Psi}_{c_i}$ is defined as

$$\boldsymbol{\Psi}_{c_i} \triangleq \text{blkdiag}\left(\mathbb{I}_i^{c_i}, m_i \boldsymbol{I}_3\right). \tag{5.6}$$

Expanding $\mathcal{G}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, we obtain

$$\mathcal{G}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \sum_{i=1}^{n} \left(\mathcal{G}_{a_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + \mathcal{G}_{b_i}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right), \tag{5.7}$$

where[6]

$$\mathcal{G}_{a_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \triangleq \frac{1}{2} \ddot{\boldsymbol{q}}^T \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}^T \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{a}}_{c_i}$$

and

$$\mathcal{G}_{b_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \triangleq \frac{1}{2} \bar{\boldsymbol{a}}_{c_i}^T \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{a}}_{c_i} + \frac{1}{2} \dot{\boldsymbol{q}}^T \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^T \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{a}}_{c_i}.$$

From the optimality condition, the solution of (5.1) is computed as (Wieber, 2005)

$$\frac{\partial \mathcal{G}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})}{\partial \ddot{\boldsymbol{q}}} = \frac{\partial}{\partial \ddot{\boldsymbol{q}}} \left(\sum_{i=1}^{n} \mathcal{G}_{a_i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \right) = \boldsymbol{0}_{1 \times n}. \tag{5.8}$$

Using (5.7) in (5.8), we have

$$\boldsymbol{0}_{n \times 1} = \sum_{i=1}^{n} \left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}} + \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\boldsymbol{q}} + \Phi \right), \tag{5.9}$$

where $\Phi = -\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{a}}_{c_i}$.

Since $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} = \begin{bmatrix} \boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T & \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \end{bmatrix}^T$, using (5.5) and the elements $\mathbb{I}_i^{c_i}$ and m_i of $\boldsymbol{\Psi}_{c_i}$, the term Φ from (5.9) can be rewritten as

$$\Phi = -\boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \mathbb{I}_i^{c_i} \operatorname{vec}_3 \dot{\boldsymbol{\omega}}_{0,c_i}^{c_i} - \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \boldsymbol{f}_{0,c_i}^{c_i} - m_i \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \left(\dot{\boldsymbol{p}}_{0,c_i}^{c_i} \times \boldsymbol{\omega}_{0,c_i}^{c_i} \right) \tag{5.10}$$

where $\operatorname{vec}_3 \left(\dot{\boldsymbol{p}}_{0,c_i}^{c_i} \times \boldsymbol{\omega}_{0,c_i}^{c_i} \right) = -\boldsymbol{S}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}\right) \operatorname{vec}_3 \dot{\boldsymbol{p}}_{0,c_i}^{c_i}$, with $\operatorname{vec}_3 \dot{\boldsymbol{p}}_{0,c_i}^{c_i} = \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)} \dot{\boldsymbol{q}}$, $\boldsymbol{f}_{0,c_i}^{c_i} = m_i \ddot{\boldsymbol{p}}_{0,c_i}^{c_i}$, and $\boldsymbol{S}(\cdot) \in \text{so}(3)$ is the skew-symmetric matrix used as an operator that performs the cross-product (Spong et al., 2006).

Furthermore, as $\operatorname{vec}_3 \boldsymbol{\omega}_{0,c_i}^{c_i} = \boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)} \dot{\boldsymbol{q}}$ and $\boldsymbol{\tau}_{0,c_i}^{c_i}$ is the torque about the ith link's

[6] Notice that $\ddot{\boldsymbol{q}}^T \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^T \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}^T \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{a}}_{c_i} = \bar{\boldsymbol{a}}_{c_i}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \ddot{\boldsymbol{q}}$.

center of mass, given by the Euler's rotation equation

$$\mathbb{I}_i^{c_i} \operatorname{vec}_3 \dot{\boldsymbol{\omega}}_{0,c_i}^{c_i} = \operatorname{vec}_3 \boldsymbol{\tau}_{0,c_i}^{c_i} + \boldsymbol{S}\left(\boldsymbol{s}_{c_i}\right) \boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)} \dot{\boldsymbol{q}}, \tag{5.11}$$

where $\boldsymbol{s}_{c_i} \triangleq \mathbb{I}_i^{c_i} \operatorname{vec}_3 \boldsymbol{\omega}_{0,c_i}^{c_i}$, and use it in (5.10) to obtain

$$\Phi \;=\; -\boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \boldsymbol{\tau}_{0,c_i}^{c_i} \;-\; \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \boldsymbol{f}_{0,c_i}^{c_i} \;+\; \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\boldsymbol{q}} \tag{5.12}$$

with

$$\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \triangleq \operatorname{blkdiag}\Big(-\boldsymbol{S}\left(\boldsymbol{s}_{c_i}\right), m_i \boldsymbol{S}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}\right)\Big). \tag{5.13}$$

Finally, using (5.12) in (5.9) yields

$$\boldsymbol{M}_{\mathrm{GP}} \ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}} \dot{\boldsymbol{q}} = \bar{\boldsymbol{\tau}}_{\mathrm{GP}}, \tag{5.14}$$

where $\boldsymbol{M}_{\mathrm{GP}} \triangleq \boldsymbol{M}_{\mathrm{GP}}\left(\boldsymbol{q}\right) \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\boldsymbol{C}_{\mathrm{GP}} \triangleq \boldsymbol{C}_{\mathrm{GP}}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) \in \mathbb{R}^{n \times n}$ denotes the nonlinear dynamic effects (including the Coriolis terms), and $\bar{\boldsymbol{\tau}}_{\mathrm{GP}} \triangleq \bar{\boldsymbol{\tau}}_{\mathrm{GP}}\left(\boldsymbol{q}\right) \in \mathbb{R}^n$ represents the generalized forces acting on the system; also,

$$\boldsymbol{M}_{\mathrm{GP}} \triangleq \sum_{i=1}^{n} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}, \tag{5.15}$$

$$\boldsymbol{C}_{\mathrm{GP}} \triangleq \sum_{i=1}^{n} \left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} + \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \right), \tag{5.16}$$

$$\bar{\boldsymbol{\tau}}_{\mathrm{GP}} \triangleq \sum_{i=1}^{n} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \bar{\boldsymbol{\varsigma}}_{c_i}, \tag{5.17}$$

where $\bar{\boldsymbol{\varsigma}}_{c_i}$ is the wrench at the ith center of mass, defined as

$$\bar{\boldsymbol{\varsigma}}_{c_i} \triangleq \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix} \operatorname{vec}_6 \underline{\boldsymbol{\zeta}}_{0,c_i}^{c_i}, \tag{5.18}$$

with $\underline{\boldsymbol{\zeta}}_{0,c_i}^{c_i} = \boldsymbol{f}_{0,c_i}^{c_i} + \varepsilon \boldsymbol{\tau}_{0,c_i}^{c_i}$.

Specifically, the generalized forces $\bar{\boldsymbol{\tau}}_{\mathrm{GP}}$ vector is given as follows

$$\bar{\boldsymbol{\tau}}_{\mathrm{GP}} = \sum_{i=1}^{n} \left(\boldsymbol{J}_{\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \boldsymbol{\tau}_{0,c_i}^{c_i} + \boldsymbol{J}_{\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}\right)}^T \operatorname{vec}_3 \boldsymbol{f}_{0,c_i}^{c_i} \right).$$

Furthermore, since the gravity does not generate any resultant moment at the center of mass of a link, the vector of gravitational forces $\boldsymbol{\tau}_g \triangleq \boldsymbol{\tau}_g\left(\boldsymbol{q}\right)$ is obtained from $\bar{\boldsymbol{\tau}}_{\mathrm{GP}}$ by letting $\boldsymbol{\tau}_{0,c_i}^{c_i} = 0$ and $\boldsymbol{f}_{0,c_i}^{c_i} = \operatorname{Ad}\left(\boldsymbol{r}_0^{c_i}\right) \boldsymbol{f}_{g_i}$, where $\boldsymbol{f}_{g_i} = m_i \boldsymbol{g}$ and $\boldsymbol{g} \in \mathbb{H}_p$ are the gravitational force

and gravitational acceleration, respectively, both expressed in the inertial frame. Hence,

$$\boldsymbol{\tau}_g = \sum_{i=1}^{n} \boldsymbol{J}^T_{\mathcal{D}(\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i})} \operatorname{vec}_3 \left(\operatorname{Ad}\left(\boldsymbol{r}_0^{c_i}\right) \boldsymbol{f}_{g_i} \right). \tag{5.19}$$

By considering the generalized forces $\boldsymbol{\tau}_{\mathrm{GP}}$ applied in the joints and the gravitational forces $\boldsymbol{\tau}_g$, the resultant forces acting on the system are $\bar{\boldsymbol{\tau}}_{\mathrm{GP}} = \boldsymbol{\tau}_{\mathrm{GP}} + \boldsymbol{\tau}_g$. Let $\boldsymbol{g}_{\mathrm{GP}} \triangleq -\boldsymbol{\tau}_g$, then (5.14) is rewritten in the canonical form as

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}} + \boldsymbol{g}_{\mathrm{GP}} = \boldsymbol{\tau}_{\mathrm{GP}}. \tag{5.20}$$

In this way, solving (5.1) leads to the Euler-Lagrange dynamic description of a mechanical system by means of dual quaternion algebra. Algorithm 5.1 summarizes the procedure for obtaining the Euler-Lagrange dynamic equation for a serial manipulator. Once again, we assume that the robot forward kinematics and differential kinematics are available in dual quaternion space (Adorno, 2011).

Algorithm 5.1 Euler Lagrange model using the Dual Quaternion Gauss's Principle of Least Constraint Formalism for a robot manipulator.

Require: The forward kinematics, the differential kinematics, and the generalized inertia tensor $\boldsymbol{\Psi}_{c_i}$ for all links' centers of mass

function EULER_LAGRANGE($\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$)
  $\boldsymbol{M}_{\mathrm{GP}} \leftarrow \boldsymbol{0}$, $\boldsymbol{C}_{\mathrm{GP}} \leftarrow \boldsymbol{0}$, $\boldsymbol{g}_{\mathrm{GP}} \leftarrow \boldsymbol{0}$
  for $i \leftarrow 1, n$ **do**
    ▷ Calculation of necessary kinematic information for each center of mass
    $\underline{\boldsymbol{x}}_0^{c_i} \leftarrow$FORWARD_KINEMATICS($\boldsymbol{q}$)
    $\boldsymbol{J}_{\underline{\boldsymbol{x}}^0_{c_i}} \leftarrow$DIFFERENTIAL_KINEMATICS($\boldsymbol{q}$)
    $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \leftarrow 2\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right)\boldsymbol{J}_{\underline{\boldsymbol{x}}^0_{c_i}}$
    $\operatorname{vec}_6 \underline{\boldsymbol{\xi}}^{c_i}_{0,c_i} \leftarrow \boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}\dot{\boldsymbol{q}}$
    $\boldsymbol{\omega}^{c_i}_{0,c_i} \leftarrow \mathcal{P}\left(\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}\right)$
    $\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \leftarrow$ Twist Jacobian derivative
    ▷ Calculation of the dynamic model
    $\boldsymbol{M}_{\mathrm{GP}} \leftarrow \boldsymbol{M}_{\mathrm{GP}} + \boldsymbol{J}^T_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}$
    $\boldsymbol{N} \leftarrow \overline{\boldsymbol{S}}\left(\boldsymbol{\omega}^{c_i}_{0,c_i}, \boldsymbol{\Psi}_{c_i}\right)\boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} + \boldsymbol{\Psi}_{c_i}\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}$
    $\boldsymbol{C}_{\mathrm{GP}} \leftarrow \boldsymbol{C}_{\mathrm{GP}} + \boldsymbol{J}^T_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \boldsymbol{N}$
    ▷ Recall that $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} = \begin{bmatrix} \boldsymbol{J}_{\mathcal{P}}^T & \boldsymbol{J}_{\mathcal{D}}^T \end{bmatrix}^T$
    $\boldsymbol{g}_{\mathrm{GP}} \leftarrow \boldsymbol{g}_{\mathrm{GP}} + \boldsymbol{J}^T_{\mathcal{D}} \operatorname{vec}_3\left(-\boldsymbol{f}^{c_i}_{0,c_i}\right)$
  end for
  $\boldsymbol{\tau}_{\mathrm{GP}} \leftarrow \boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}} + \boldsymbol{g}_{\mathrm{GP}}$
  return $\boldsymbol{\tau}_{\mathrm{GP}}$
end function

Remark 5.1. Let $\boldsymbol{A} \triangleq (1/2)\,\dot{\boldsymbol{M}}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}}$, then

$$\boldsymbol{A} = -\sum_{i=1}^{n} \boldsymbol{J}^{T}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \overline{\boldsymbol{S}}\left(\boldsymbol{\omega}^{c_i}_{0,c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}.$$

Since $\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}^{c_i}_{0,c_i}, \boldsymbol{\Psi}_{c_i}\right)$ is skew-symmetric by construction, by direct calculation, $\boldsymbol{A}^T = -\boldsymbol{A}$, which implies

$$\boldsymbol{u}^T \left(\frac{1}{2}\dot{\boldsymbol{M}}_{\mathrm{GP}}(\boldsymbol{q}) - \boldsymbol{C}_{\mathrm{GP}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right) \boldsymbol{u} = 0 \tag{5.21}$$

for all $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u} \in \mathbb{R}^n$. Property (5.21) is useful to show formal closed-loop stability in robot dynamic control using strategies based on Lyapunov functions (Kelly et al., 2005).

5.3 Connections with the Gibbs-Appell and Kane's equations

The Gibbs-Appell and Kane's equations have proven to be a powerful mathematical tool to describe both unconstrained and constrained mechanical systems without the use of Lagrange multipliers (Storch & Gates, 1989; Honein & O'Reilly, 2021). Both are different ways to get the equations of motion, but equivalent in the sense that a set of equations implies the other (Townsend, 1992; Desloge, 1987; Levinson, 1987).

The Gibbs-Appell method is closely related with the Gauss's Principle of Least Constraint since both approaches use scalar quadratic functions in terms of accelerations. The former could be seen as a generalization of the latter (Ray, 1972, 1992). However, they are equivalent and both can be derived from the other (Ray, 1972; Lewis, 1996; Udwadia & Kalaba, 1998). Nonetheless, different from the Gibbs-Appell and Kane's equations, the Gauss's principle strategy does not require setting up quasi-velocities and allows taking into account additional constraints directly in the optimization formulation.

In this section, we rewrite the Gibbs-Appell and Kane's equations using the equations derived in Sections 5.2.2–5.2.3. Furthermore, we show that the Euler-Lagrange dynamic description of a mechanical system can be shown to be a particular case of the Gibbs-Appell and Kane's equations. This is done by selecting the quasi-velocities to be the same as the generalized velocities.

5.3.1 Gibbs-Appell equation

For n rigid bodies, the Gibbs-Appell equation of motion is given by (Desloge, 1988)

$$\frac{\partial S}{\partial \dot{\boldsymbol{u}}} = \underbrace{\sum_{i=1}^{n} \left(\frac{\partial}{\partial \boldsymbol{u}} \boldsymbol{\nu}_{c_i} \right)^T \bar{\boldsymbol{\varsigma}}_{c_i}}_{\bar{\boldsymbol{\tau}}_{\mathrm{GA}}}, \tag{5.22}$$

where S is the Gibbs-Appell function,[7] which is a scalar function in terms of accelerations. The vector $\bar{\boldsymbol{\tau}}_{\mathrm{GA}}$ contains generalized forces associated with the quasi-velocities $\boldsymbol{u}$. Furthermore, $\boldsymbol{\nu}_{c_i}$ is the twist of the ith body, and

$$\bar{\boldsymbol{\varsigma}}_{c_i} = \begin{bmatrix} \mathrm{vec}_3 \left(\boldsymbol{\tau}_{0,c_i}^{c_i} \right)^T & \mathrm{vec}_3 \left(\boldsymbol{f}_{0,c_i}^{c_i} \right)^T \end{bmatrix}^T \tag{5.23}$$

are generalized forces defined by (5.18).

Let $\boldsymbol{u} \triangleq \dot{\boldsymbol{q}}$, then, the Gibbs function in dual quaternion algebra can be written as a function of the configuration $\boldsymbol{q}$, configuration velocity $\dot{\boldsymbol{q}}$, and configuration acceleration $\ddot{\boldsymbol{q}}$ as follows

$$S\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) \triangleq \sum_{i=1}^{n} \left(\frac{1}{2} \boldsymbol{a}_{c_i}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{a}_{c_i} + \boldsymbol{a}_{c_i}^T \overline{\boldsymbol{S}} \left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i} \right) \boldsymbol{\nu}_{c_i} \right), \tag{5.24}$$

where $\boldsymbol{v}_{c_i}$ and $\boldsymbol{a}_{c_i}$ are given by (5.2) and (5.3), respectively. Furthermore, $\overline{\boldsymbol{S}} \left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i} \right)$ is given by (5.13) with $\boldsymbol{\Psi}_{c_i}$ given by (5.6).

Using (5.24), we rewrite (5.22) as

$$\frac{\partial S\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right)}{\partial \ddot{\boldsymbol{q}}} = \underbrace{\sum_{i=1}^{n} \left(\frac{\partial}{\partial \dot{\boldsymbol{q}}} \mathrm{vec}_6 \, \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} \right)^T , \bar{\boldsymbol{\varsigma}}_{c_i}}_{\bar{\boldsymbol{\tau}}_{\mathrm{GA}}}. \tag{5.25}$$

The left side of equation (5.25) is computed as follows

$$\frac{\partial S\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right)}{\partial \ddot{\boldsymbol{q}}} = \underbrace{\sum_{i=1}^{n} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}}_{\boldsymbol{M}_{\mathrm{GP}}} \ddot{\boldsymbol{q}} + \underbrace{\sum_{i=1}^{n} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T \left(\overline{\boldsymbol{S}} \left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i} \right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} + \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \right)}_{\boldsymbol{C}_{\mathrm{GP}}} \dot{\boldsymbol{q}},$$

$$\frac{\partial S\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right)}{\partial \ddot{\boldsymbol{q}}} = \boldsymbol{M}_{\mathrm{GP}} \ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}} \dot{\boldsymbol{q}}, \tag{5.26}$$

where $\boldsymbol{M}_{\mathrm{GP}} \in \mathbb{R}^{n \times n}$ is the inertia matrix and $\boldsymbol{C}_{\mathrm{GP}} \in \mathbb{R}^{n \times n}$ denotes the nonlinear dynamic effects (including the Coriolis terms). Note that both $\boldsymbol{M}_{\mathrm{GP}}$ and $\boldsymbol{C}_{\mathrm{GP}}$ are identical to the ones obtained by the Gauss's principle, which are given by (5.15) and 5.16, respectively.

Furthermore, as $\boldsymbol{\nu}_{c_i} = \mathrm{vec}_6 \, \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}$, we have

[7] The Gibbs-Appell function is also called the *energy of acceleration,* or simply *the Gibbs function.*

$$\frac{\partial \boldsymbol{\nu}_{c_i}}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \dot{\boldsymbol{q}}} \left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \dot{\boldsymbol{q}} \right) = \boldsymbol{J}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}. \tag{5.27}$$

Using (5.23) and (5.27), the vector $\bar{\boldsymbol{\tau}}_{\mathrm{GA}}$ can be rewritten as

$$\bar{\boldsymbol{\tau}}_{\mathrm{GA}} = \boldsymbol{J}^T_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \bar{\boldsymbol{\varsigma}}_{c_i} = \bar{\boldsymbol{\tau}}_{\mathrm{GP}}. \tag{5.28}$$

Notice that the generalized forces $\bar{\boldsymbol{\tau}}_{\mathrm{GA}}$ are identical to the ones obtained by the Gauss's principle, which are denoted by $\bar{\boldsymbol{\tau}}_{\mathrm{GP}}$ (5.17).

Finally, using (5.26) and (5.28) in (5.25) yields

$$\boldsymbol{M}_{\mathrm{GP}} \ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}} \dot{\boldsymbol{q}} = \bar{\boldsymbol{\tau}}_{\mathrm{GP}}. \tag{5.29}$$

In this way, solving (5.22) leads to the Euler-Lagrange equation, which is identical to the one obtained by using the Gauss's principle (5.14), as expected.

5.3.2 Kane's equation

The Kane's equation of motion is given by (Kane, 1983)

$$\boldsymbol{\varphi} - \bar{\boldsymbol{\varphi}} = \boldsymbol{0}, \tag{5.30}$$

where $\boldsymbol{\varphi}$ contains the generalized active forces and $\bar{\boldsymbol{\varphi}}$ contains the generalized inertia forces.

To obtain the equation of motion using Kane's method, we use the Newton's and Euler's equations for n rigid bodies, which are given respectively as

$$\sum_{i=1}^{n} m_i \operatorname{vec}_3 \ddot{\boldsymbol{p}}^{c_i}_{0,c_i} = \sum_{i=1}^{n} \operatorname{vec}_3 \boldsymbol{f}^{c_i}_{0,c_i}, \tag{5.31}$$

and

$$\sum_{i=1}^{n} \left(\mathbb{I}^{c_i}_i \operatorname{vec}_3 \dot{\boldsymbol{\omega}}^{c_i}_{0,c_i} - \boldsymbol{S}\left(\boldsymbol{s}_{c_i}\right) \operatorname{vec}_3 \boldsymbol{\omega}^{c_i}_{0,c_i} \right) = \sum_{i=1}^{n} \operatorname{vec}_3 \boldsymbol{\tau}^{c_i}_{0,c_i}, \tag{5.32}$$

where the vector $\boldsymbol{s}_{c_i}$ is given by (5.11).

Equations (5.32) and (5.31) can be grouped as follows:

$$\sum_{i=1}^{n} \begin{bmatrix} \left(\mathbb{I}^{c_i}_i \operatorname{vec}_3 \dot{\boldsymbol{\omega}}^{c_i}_{0,c_i} - \boldsymbol{S}\left(\boldsymbol{s}_{c_i}\right) \operatorname{vec}_3 \boldsymbol{\omega}^{c_i}_{0,c_i} \right) \\ m_i \operatorname{vec}_3 \ddot{\boldsymbol{p}}^{c_i}_{0,c_i} \end{bmatrix} = \underbrace{\sum_{i=1}^{n} \begin{bmatrix} \operatorname{vec}_3 \boldsymbol{\tau}^{c_i}_{0,c_i} \\ \operatorname{vec}_3 \boldsymbol{f}^{c_i}_{0,c_i} \end{bmatrix}}_{\bar{\boldsymbol{\varsigma}}_{c_i}}. \tag{5.33}$$

Using (5.13), and the fact that $\boldsymbol{\nu}_{c_i} = \operatorname{vec}_6 \underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}$, with $\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i} \triangleq \underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}(\boldsymbol{q})$ given as in (5.4), $\boldsymbol{a}_{c_i} = \operatorname{vec}_6 \dot{\underline{\boldsymbol{\xi}}}^{c_i}_{0,c_i}$, with $\dot{\underline{\boldsymbol{\xi}}}^{c_i}_{0,c_i} \triangleq \dot{\underline{\boldsymbol{\xi}}}^{c_i}_{0,c_i}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ given as in (5.5),[8] and $\operatorname{vec}_3 \left(\dot{\boldsymbol{p}}^{c_i}_{0,c_i} \times \boldsymbol{\omega}^{c_i}_{0,c_i} \right) =$

[8]Notice that, although $\boldsymbol{a}_{c_i}$ is analogous to (5.5), it refers to the actual accelerations and, therefore, the

$-\boldsymbol{S}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}\right)\mathrm{vec}_3\,\dot{\boldsymbol{p}}_{0,c_i}^{c_i}$, we rewrite (5.33) as

$$\sum_{i=1}^{n}\underbrace{\left(\boldsymbol{\Psi}_{c_i}\boldsymbol{a}_{c_i}+\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i},\boldsymbol{\Psi}_{c_i}\right)\boldsymbol{\nu}_{c_i}\right)}_{\kappa_i}=\sum_{i=1}^{n}\bar{\boldsymbol{\varsigma}}_{c_i}. \tag{5.34}$$

Since $\kappa_i=\bar{\varsigma}_{c_i}$ for $i\in\{1,\ldots,n\}$, we multiply both sides of (5.34) by $\left(\frac{\partial}{\partial\boldsymbol{u}}\boldsymbol{\nu}_{c_i}\right)^T=\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T$, and use (5.3) to obtain the Kane's equations (Kane, 1983), which yields

$$\underbrace{\sum_{i=1}^{n}\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T\left(\boldsymbol{\Psi}_{c_i}\boldsymbol{a}_{c_i}+\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i},\boldsymbol{\Psi}_{c_i}\right)\boldsymbol{\nu}_{c_i}\right)}_{\boldsymbol{\varphi}}=\underbrace{\sum_{i=1}^{n}\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T\bar{\boldsymbol{\varsigma}}_{c_i}}_{\bar{\boldsymbol{\varphi}}}, \tag{5.35}$$

$$\underbrace{\sum_{i=1}^{n}\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T\boldsymbol{\Psi}_{c_i}\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}}_{\boldsymbol{M}_{\mathrm{GP}}}\ddot{\boldsymbol{q}}+\underbrace{\sum_{i=1}^{n}\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^T\left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i},\boldsymbol{\Psi}_{c_i}\right)\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}+\boldsymbol{\Psi}_{c_i}\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right)}_{\boldsymbol{C}_{\mathrm{GP}}}\dot{\boldsymbol{q}}=\underbrace{\bar{\boldsymbol{\tau}}_{\mathrm{GP}}}_{\bar{\boldsymbol{\varphi}}} \tag{5.36}$$

$$\underbrace{\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}}+\boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}}}_{\boldsymbol{\varphi}}=\underbrace{\bar{\boldsymbol{\tau}}_{\mathrm{GP}}}_{\bar{\boldsymbol{\varphi}}}, \tag{5.37}$$

which is the same dynamic equation as the the Gibbs-Appell equation (5.29), and the one obtained by the Gauss's principle (5.14), as expected.

Therefore, when considering the quasi-velocities to be the same as the generalized velocities (i.e., $\boldsymbol{u}\triangleq\dot{\boldsymbol{q}}$) the relations between Gauss's principle, Gibbs-Appell equations and Kane's method are given by

$$\frac{\partial\mathcal{G}\left(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}\right)}{\partial\ddot{\boldsymbol{q}}}=\frac{\partial S\left(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}\right)}{\partial\ddot{\boldsymbol{q}}}-\bar{\boldsymbol{\tau}}_{\mathrm{GA}}=\boldsymbol{\varphi}-\bar{\boldsymbol{\varphi}}=\boldsymbol{0}_{n\times1}. \tag{5.38}$$

Notice that the relations given by 5.38 are computed using the tools from dual quaternion algebra. However, they are valid using any representation.

5.4 Constrained Robotic Systems

Additional constraints can be imposed in the GPLC formulation. This can be done by means of Lagrange multipliers (Bouyarmane & Kheddar, 2012) or using the Udwadia-Kalaba formulation (Kalaba & Udwadia, 1992). The former requires the computation of the Lagrange multipliers, whereas the latter employs a simpler method, albeit equivalent (See Appendix (B)), which is based on generalized inverses as the solution to a constrained quadratic program and can take into account inequality constraints directly in the optimization formulation. However, unilateral constraints in the GPLC formulation are not explored in this dissertation.

constrained ones. Consequently, $\boldsymbol{a}_{c_i}$ depends on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, whereas (5.5) does not.

Kalaba & Udwadia, 1992 proposed an elegant solution to the constrained Gauss's Principle, which is stated as

$$\begin{aligned} \min_{\ddot{\boldsymbol{x}}} \quad & (\ddot{\boldsymbol{x}} - \boldsymbol{a})^T \boldsymbol{M} (\ddot{\boldsymbol{x}} - \boldsymbol{a}) \\ \text{subject to} \quad & \boldsymbol{A}\ddot{\boldsymbol{x}} = \boldsymbol{b}, \end{aligned} \tag{5.39}$$

where $\ddot{\boldsymbol{x}}$ and $\boldsymbol{a}$ are the accelerations under constraints and without constraints, respectively, of a system of n particles and $\boldsymbol{M}$ is a positive definite matrix. Furthermore, $\boldsymbol{A}$ and $\boldsymbol{b}$ are the matrix and vector constraints, respectively. The solution to the problem (5.39) is called the fundamental equation and is given by

$$\ddot{\boldsymbol{x}} = \boldsymbol{a} + \boldsymbol{M}^{-1/2} \left(\boldsymbol{A}\boldsymbol{M}^{-1/2} \right)^{+} (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{a}), \tag{5.40}$$

where the superscript " + " denotes the Moore-Penrose generalized inverse.

We rewrite the problem (5.39) in the joint space as follows. Let $\ddot{\boldsymbol{q}}_a$ be the joint accelerations without considering constraints, and $\ddot{\boldsymbol{q}}$ be the joint accelerations under constraints. Then, we have

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}} \quad & (\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_a)^T \boldsymbol{M}_{\mathrm{GP}} (\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_a) \\ \text{subject to} \quad & \boldsymbol{A}\ddot{\boldsymbol{q}} = \boldsymbol{b}, \end{aligned} \tag{5.41}$$

where

$$\ddot{\boldsymbol{q}}_a = \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q}, \tag{5.42}$$

with

$$\boldsymbol{Q} \triangleq \bar{\boldsymbol{\tau}}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}} \dot{\boldsymbol{q}}. \tag{5.43}$$

Using (5.40), the solution to the problem (5.41) is given as

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q} + \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \left(\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1/2} \right)^{+} \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q} \right). \tag{5.44}$$

Multiplying by $\boldsymbol{M}_{\mathrm{GP}}$ and using (5.42) and (5.43)

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}} = \bar{\boldsymbol{\tau}}_{\mathrm{GP}} + \boldsymbol{M}_{\mathrm{GP}}^{1/2} \left(\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1/2} \right)^{+} \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1} (\bar{\boldsymbol{\tau}}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}}) \right). \tag{5.45}$$

Letting $\boldsymbol{D} \triangleq \left(\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1/2} \right)$, the equation (5.45) is rewritten as

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} + \left(\boldsymbol{I} - \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^{+} \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1} \right) \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}} = \left(\boldsymbol{I} - \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^{+} \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1} \right) \bar{\boldsymbol{\tau}}_{\mathrm{GP}} + \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^{+} \boldsymbol{b}. \tag{5.46}$$

Finally, letting $\boldsymbol{\Omega} \triangleq \left(\boldsymbol{I} - \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^{+} \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1} \right)$, the Euler-Lagrange description of a

constrained robotic system using Gauss's Principle of Least Constraint is

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} + \boldsymbol{\Omega}\boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}} - \boldsymbol{M}_{\mathrm{GP}}^{1/2}\boldsymbol{D}^{+}\boldsymbol{b} = \boldsymbol{\Omega}\bar{\boldsymbol{\tau}}_{\mathrm{GP}}. \tag{5.47}$$

For example, consider the well-known differential-drive mobile robot, in which the nonholonomic constraint ensures the conditions of pure rolling and non-slipping movements (Fierro & Lewis, 1997). The robot configuration is specified by the vector $\boldsymbol{q} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$, where x, y is the position coordinates and ϕ is the orientation of the robot on the plane. The nonholonomic constraint is given by

$$\underbrace{\begin{bmatrix} -\sin\phi & \cos\phi & 0 \end{bmatrix}}_{\boldsymbol{A}}\dot{\boldsymbol{q}} = 0, \tag{5.48}$$

which can be enforced in (5.47) by taking the time derivative of (5.48) such that $\dot{\boldsymbol{A}}\dot{\boldsymbol{q}} + \boldsymbol{A}\ddot{\boldsymbol{q}} = 0$. Therefore, $\boldsymbol{b} = -\dot{\boldsymbol{A}}\dot{\boldsymbol{q}}$.

5.4.1 Example: Dynamic Modeling of Humanoid Robot

In this section, we use the Gauss's Principle of Least Constraint presented in Section 5.2 to write the Euler-Lagrange equation of a humanoid robot. First, we define the generalized coordinates to fully describe the humanoid robot. Second, we compute the twist Jacobians, and finally, we impose the unit norm constraint in the optimization problem using (5.47), similar to the formulation proposed by Bouyarmane & Kheddar (2012).

The movement of a humanoid robot is determined by its joint configurations and the pose of its body with respect to a reference frame. Furthermore, the contacts between the robot and the environment generate reaction forces, which are used for the balance and locomotion tasks.

Let $\mathbb{R}^{n+8} \ni \bar{\boldsymbol{q}} \triangleq \begin{bmatrix} \boldsymbol{q}^T & \left(\mathrm{vec}_8\, \underline{\boldsymbol{x}}_p^0\right)^T \end{bmatrix}^T$ be the optimization vector in the problem (5.1), where $\boldsymbol{q} \in \mathbb{R}^n$ is the robot's joint configuration vector and $\underline{\boldsymbol{x}}_p^0 \in \underline{\boldsymbol{\mathcal{S}}}$ is the pose of its body frame $\mathcal{F}_p$ with respect to an inertial frame $\mathcal{F}_0$, as shown in Fig. 5.3.

5.4.2 Twist Jacobians

Let $\underline{\boldsymbol{x}}_{c_i}^p = \underline{\boldsymbol{f}}(\boldsymbol{q})$ be a function of the robot's joint configurations $\boldsymbol{q}$, with $\underline{\boldsymbol{f}} : \mathbb{R}^n \to \underline{\boldsymbol{\mathcal{S}}}$, where $\underline{\boldsymbol{x}}_{c_i}^p$ denotes the pose of the frame $\mathcal{F}_{c_i}$ with respect to the pelvis frame $\mathcal{F}_p$. Moreover, its time derivative and the associated twist (see. (A.19)) is given as

$$\dot{\underline{\boldsymbol{x}}}_{c_i}^p = \frac{1}{2}\underline{\boldsymbol{x}}_{c_i}^p \underline{\boldsymbol{\xi}}_{p,c_i}^{c_i} \implies \underline{\boldsymbol{\xi}}_{p,c_i}^{c_i} = 2\underline{\boldsymbol{x}}_{c_i}^{p*}\dot{\underline{\boldsymbol{x}}}_{c_i}^p. \tag{5.49}$$

Applying the $\mathrm{vec}_6(\cdot)$ operator, and using the fact $\mathrm{vec}_8\, \dot{\underline{\boldsymbol{x}}}_{c_i}^p = \boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^p}\dot{\boldsymbol{q}}$, we rewrite (5.49)

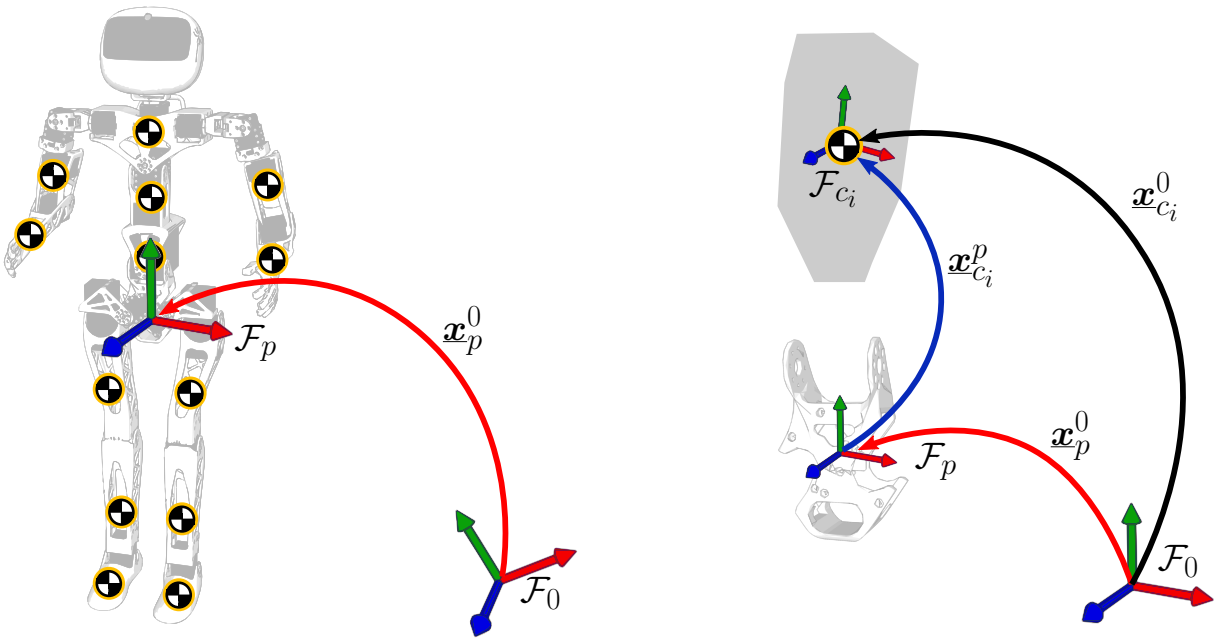

Figure 5.3: Humanoid robot modeling. On the left, $\underline{\boldsymbol{x}}_p^0$ represents the pose of the pelvis frame $\mathcal{F}_p$ with respect to the frame $\mathcal{F}_0$. On the right, $\underline{\boldsymbol{x}}_{c_i}^0$ represents the pose of a frame $\mathcal{F}_{c_i}$ with respect to the frame $\mathcal{F}_0$, and whose origin is the CoM of the ith body.

as

$$\operatorname{vec}_6 \underline{\boldsymbol{\xi}}_{p,c_i}^{c_i} = \underbrace{2\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8 \left(\underline{\boldsymbol{x}}_{c_i}^{p*}\right) \boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^p}}_{\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}} \dot{\boldsymbol{q}},$$

where $\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{q}_1 & \cdots & \dot{q}_n \end{bmatrix}^T$, with $\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^p} \in \mathbb{R}^{8\times n}$ and $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} \in \mathbb{R}^{6\times n}$ being the pose Jacobian and the twist Jacobian matrices, respectively, both obtained algebraically (Adorno, 2011). The goal is to find the twist Jacobian $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \in \mathbb{R}^{6\times(n+8)}$ that satisfies $\operatorname{vec}_6 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\tilde{\boldsymbol{q}}}$, which is required to compute $\boldsymbol{M}_{\text{GP}}$ (5.15), $\boldsymbol{C}_{\text{GP}}$ (5.16) and $\bar{\boldsymbol{\tau}}_{\text{GP}}$ (5.17).

Consider the frame place at the ith center of mass $\underline{\boldsymbol{x}}_{c_i}^0$, which is computed as

$$\underline{\boldsymbol{x}}_{c_i}^0 = \underline{\boldsymbol{x}}_p^0 \underline{\boldsymbol{x}}_{c_i}^p. \tag{5.50}$$

The time derivative of (5.50) is given as

$$\dot{\underline{\boldsymbol{x}}}_{c_i}^0 = \dot{\underline{\boldsymbol{x}}}_p^0 \underline{\boldsymbol{x}}_{c_i}^p + \underline{\boldsymbol{x}}_p^0 \dot{\underline{\boldsymbol{x}}}_{c_i}^p. \tag{5.51}$$

Since $\dot{\underline{\boldsymbol{x}}}_{c_i}^0 = \frac{1}{2}\underline{\boldsymbol{x}}_{c_i}^0 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}$, we rewrite (5.51) as

$$\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = 2\underline{\boldsymbol{x}}_{c_i}^{0*} \dot{\underline{\boldsymbol{x}}}_p^0 \underline{\boldsymbol{x}}_{c_i}^p + 2\underline{\boldsymbol{x}}_{c_i}^{p*} \dot{\underline{\boldsymbol{x}}}_{c_i}^p. \tag{5.52}$$

Applying the $\mathrm{vec}_6(\cdot)$ operator to (5.52), we have

$$\mathrm{vec}_6 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = \left[\underbrace{2\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_{c_i}^{p*}\right)\boldsymbol{J}_{\underline{\dot{\boldsymbol{x}}}_{c_i}^{p}}}_{\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}}\dot{\boldsymbol{q}} \quad \underbrace{2\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_{c_i}^{0*}\right)\bar{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_{c_i}^{p}\right)}_{\boldsymbol{\Xi}_{c_i}} \mathrm{vec}_8\, \dot{\underline{\boldsymbol{x}}}_p^0 \right]. \tag{5.53}$$

Since $\dot{\bar{\boldsymbol{q}}} = \left[\begin{array}{cc} \dot{\boldsymbol{q}}^T & \left(\mathrm{vec}_8\, \dot{\underline{\boldsymbol{x}}}_p^0\right)^T \end{array} \right]^T$, we rewrite (5.53) as

$$\mathrm{vec}_6 \underline{\boldsymbol{\xi}}_{0,c_i}^{c_i} = \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \dot{\bar{\boldsymbol{q}}},$$

where the twist Jacobian $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ is defined as

$$\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \triangleq \left[\begin{array}{cc} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} & \boldsymbol{\Xi}_{c_i} \end{array} \right]. \tag{5.54}$$

To compute the inertia matrix $\boldsymbol{M}_{\mathrm{GP}} \in \mathbb{R}^{(n+8)\times(n+8)}$, for instance, we use (5.54) in (5.15), which yields

$$\boldsymbol{M}_{\mathrm{GP}} = \sum_{i=1}^{n} \left[\begin{array}{cc} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} & \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{\Xi}_{c_i} \\ \boldsymbol{\Xi}_{c_i}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} & \boldsymbol{\Xi}_{c_i}^T \boldsymbol{\Psi}_{c_i} \boldsymbol{\Xi}_{c_i} \end{array} \right].$$

Analogously, we compute $\boldsymbol{C}_{\mathrm{GP}}$ and $\bar{\boldsymbol{\tau}}_{\mathrm{GP}}$ using (5.16) and (5.17), respectively, which results in

$$\boldsymbol{C}_{\mathrm{GP}} = \sum_{i=1}^{n} \left[\begin{array}{cc} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}^T \left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} + \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} \right) & \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}^T \left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{\Xi}_{c_i} + \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{\Xi}}_{c_i} \right) \\ \boldsymbol{\Xi}_{c_i}^T \left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} + \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}} \right) & \boldsymbol{\Xi}_{c_i}^T \left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right) \boldsymbol{\Xi}_{c_i} + \boldsymbol{\Psi}_{c_i} \dot{\boldsymbol{\Xi}}_{c_i} \right) \end{array} \right],$$

and

$$\bar{\boldsymbol{\tau}}_{\mathrm{GP}} = \sum_{i=1}^{n} \left[\begin{array}{c} \boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{p,c_i}^{c_i}}^T \\ \boldsymbol{\Xi}_{c_i}^T \end{array} \right] \bar{\boldsymbol{\varsigma}}_{c_i}.$$

5.4.3 Unit Norm Constraint

Since $\underline{\boldsymbol{x}}_p^0$ has unit norm, then

$$\underline{\boldsymbol{x}}_p^{0*} \underline{\boldsymbol{x}}_p^0 = 1. \tag{5.55}$$

The time derivative of (5.55) is

$$\dot{\underline{\boldsymbol{x}}}_p^{0*} \underline{\boldsymbol{x}}_p^0 + \underline{\boldsymbol{x}}_p^{0*} \dot{\underline{\boldsymbol{x}}}_p^0 = 0. \tag{5.56}$$

Applying the $\mathrm{vec}_8(\cdot)$ and the Hamilton operators to (5.56), we have

$$\boldsymbol{P}\left(\underline{\boldsymbol{x}}_p^0\right) \mathrm{vec}_8\left(\dot{\underline{\boldsymbol{x}}}_p^0\right) = \boldsymbol{0}, \tag{5.57}$$

where $\boldsymbol{P}\left(\underline{\boldsymbol{x}}_p^0\right) \triangleq \left(\bar{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_p^0\right)\boldsymbol{C}_8 + \overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_p^{0*}\right)\right)$.

Using the time derivative of (5.57), we obtain the constraint at the acceleration level as follows

$$\boldsymbol{P}\left(\underline{\boldsymbol{x}}_p^0\right)\mathrm{vec}_8\left(\ddot{\underline{\boldsymbol{x}}}_p^0\right) = -\dot{\boldsymbol{P}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\mathrm{vec}_8\left(\dot{\underline{\boldsymbol{x}}}_p^0\right), \tag{5.58}$$

where $\dot{\boldsymbol{P}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right) = \left(\dot{\bar{\boldsymbol{H}}}_8\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\boldsymbol{C}_8 + \dot{\overset{+}{\boldsymbol{H}}}_8\left(\dot{\underline{\boldsymbol{x}}}_p^{0*}\right)\right)$.

The constraint (5.58) is rewritten in function of $\ddot{\boldsymbol{q}}$ as follows

$$\underbrace{\begin{bmatrix}\boldsymbol{0}_{8\times n} & \boldsymbol{P}\left(\underline{\boldsymbol{x}}_p^0\right)\end{bmatrix}}_{\bar{\boldsymbol{P}}\left(\underline{\boldsymbol{x}}_p^0\right)}\underbrace{\begin{bmatrix}\ddot{\boldsymbol{q}} \\ \mathrm{vec}_8\left(\ddot{\underline{\boldsymbol{x}}}_p^0\right)\end{bmatrix}}_{\ddot{\bar{\boldsymbol{q}}}} = -\underbrace{\begin{bmatrix}\boldsymbol{0}_{8\times n} & \dot{\boldsymbol{P}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\end{bmatrix}}_{\dot{\bar{\boldsymbol{P}}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)}\underbrace{\begin{bmatrix}\dot{\boldsymbol{q}} \\ \mathrm{vec}_8\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\end{bmatrix}}_{\dot{\bar{\boldsymbol{q}}}},$$

$$\bar{\boldsymbol{P}}\left(\underline{\boldsymbol{x}}_p^0\right)\ddot{\bar{\boldsymbol{q}}} = -\dot{\bar{\boldsymbol{P}}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\dot{\bar{\boldsymbol{q}}}. \tag{5.59}$$

The final dynamic equation of the humanoid robot is given by (5.47), where the matrix and vector constraints are $\boldsymbol{A} \triangleq \bar{\boldsymbol{P}}\left(\underline{\boldsymbol{x}}_p^0\right)$ and $\boldsymbol{b} \triangleq -\dot{\bar{\boldsymbol{P}}}\left(\dot{\underline{\boldsymbol{x}}}_p^0\right)\dot{\bar{\boldsymbol{q}}}$, respectively.

5.5 Computational Cost

The comparison between the proposed algorithm (5.1) and their classic counterparts is made in terms of number of multiplications, additions, and trigonometric operations involved in each method. To that aim, first we define a cost operator $\mathfrak{C}(\mathrm{op})$ that is used to calculate the cost of the operation op as a function of the cost of simpler operations (Adorno, 2011). For example, given $\underline{\boldsymbol{a}}, \underline{\boldsymbol{b}} \in \mathcal{H}$, the cost of their multiplication is given by $\mathfrak{C}(\underline{\boldsymbol{a}}\underline{\boldsymbol{b}})$ and, since $\mathrm{Ad}(\underline{\boldsymbol{a}})\,\underline{\boldsymbol{b}} = \underline{\boldsymbol{a}}\underline{\boldsymbol{b}}\underline{\boldsymbol{a}}^*$ then $\mathfrak{C}(\mathrm{Ad}(\underline{\boldsymbol{a}})\,\underline{\boldsymbol{b}}) = 2\mathfrak{C}(\underline{\boldsymbol{a}}\underline{\boldsymbol{b}}) + \mathfrak{C}(\underline{\boldsymbol{a}}^*)$. In other words, the cost of one adjoint operation is equivalent to the cost of two dual quaternion multiplications plus one dual quaternion conjugation. Table 5.1 summarizes the cost of elementary operations used throughout this section.

5.5.1 Dual Quaternion Euler Lagrange algorithm using Gauss's Principle of Least Constraint

From (5.2), the cost of calculating $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ is

$$\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = \mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0}\right) + \mathfrak{C}(\lambda\underline{\boldsymbol{a}}) + \mathfrak{C}\left(\underset{6\times 8}{\boldsymbol{A}}\,\underset{8\times i}{\boldsymbol{B}}\right), \tag{5.60}$$

where we used the facts that the product $\bar{\boldsymbol{I}}\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right)$ is equivalent to removing the first and fifth rows from $\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right)$ and $2\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right) = \overset{+}{\boldsymbol{H}}_8\left(2\underline{\boldsymbol{x}}_0^{c_i}\right)$.

Table 5.1: Cost of operations with quaternions, dual quaternions, matrices and vectors in terms of multiplication and addition of real numbers.

	Mult.	Add.
Quaternions ($\mathbb{I} \in \mathbb{R}^{3\times3}$, $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{H}$, and $\lambda \in \mathbb{R}$)		
$\mathfrak{C}(\lambda \boldsymbol{a})$	4	0
$\mathfrak{C}(\mathrm{Ad}(\boldsymbol{a})\,\boldsymbol{b}) = 2\mathfrak{C}(\boldsymbol{a}\boldsymbol{b}) + \mathfrak{C}(\boldsymbol{a}^*)$	35	24
Dual quaternions ($\underline{\boldsymbol{a}}, \underline{\boldsymbol{b}} \in \mathcal{H}$)		
$\mathfrak{C}(\lambda \underline{\boldsymbol{a}})$	8	0
$\mathfrak{C}(\mathrm{Ad}(\underline{\boldsymbol{a}})\,\underline{\boldsymbol{b}}) = 2\mathfrak{C}(\underline{\boldsymbol{a}}\underline{\boldsymbol{b}}) + \mathfrak{C}(\underline{\boldsymbol{a}}^*)$	102	80
Matrices and vectors ($\boldsymbol{c} \in \mathbb{R}^3$)		
$\mathfrak{C}\left(\lambda \underset{m\times p}{\boldsymbol{A}}\right)$	mp	0
$\mathfrak{C}\left(\underset{m\times p}{\boldsymbol{A}} + \underset{m\times p}{\boldsymbol{B}}\right)$	0	mp
$\mathfrak{C}\left(\underset{m\times p}{\boldsymbol{A}}\ \underset{p\times r}{\boldsymbol{B}}\right)$	mpr	$mr(p-1)$
$\mathfrak{C}(\boldsymbol{S}(\boldsymbol{c})) = \mathfrak{C}(\lambda \boldsymbol{c})$	3	0
$\mathfrak{C}\left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}_{0,c_i}^{c_i}, \boldsymbol{\Psi}_{c_i}\right)\right)$ (See Eq. 5.13)	18	6

The time derivative of the Jacobian $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ is given by

$$\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} = \left[\begin{array}{cc} \dot{\bar{\boldsymbol{J}}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} & \boldsymbol{0}_{6\times(n-i)} \end{array}\right] \tag{5.61}$$

with

$$\dot{\bar{\boldsymbol{J}}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} = 2\bar{\boldsymbol{I}}\left(\overset{+}{\boldsymbol{H}}_8\left(\dot{\underline{\boldsymbol{x}}}_0^{c_i}\right)\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0} + \overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}_0^{c_i}\right)\dot{\boldsymbol{J}}_{\underline{\boldsymbol{x}}_{c_i}^0}\right).$$

Therefore, the cost of calculating $\dot{\bar{\boldsymbol{J}}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ is given by

$$\mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = \mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{x}}_{c_i}^0}\right) + 2\mathfrak{C}\left(\underset{6\times8}{\boldsymbol{A}}\ \underset{8\times i}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{6\times i}{\boldsymbol{A}} + \underset{6\times i}{\boldsymbol{B}}\right) + \mathfrak{C}(\lambda\underline{\boldsymbol{a}}) \tag{5.62}$$

since $\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0}$ and $\overset{+}{\boldsymbol{H}}_8\left(2\underline{\boldsymbol{x}}_0^{c_i}\right)$ were already calculated for $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$.

From (5.60), the cost of calculating the n Jacobians $\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}$ is

$$\sum_{i=1}^{n}\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = \sum_{i=1}^{n}\left(\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0}\right) + \mathfrak{C}\left(\underset{6\times8}{\boldsymbol{A}}\ \underset{8\times i}{\boldsymbol{B}}\right)\right) + n\mathfrak{C}(\lambda\underline{\boldsymbol{a}}). \tag{5.63}$$

Since $\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^0}\right) + \mathfrak{C}\left(\underset{6\times8}{\boldsymbol{A}}\ \underset{8\times i}{\boldsymbol{B}}\right) = c_\alpha i + c_\beta$, where c_α and c_β are found by inspection, then we

use the relation $\sum_{i=1}^{n} i = n\left(n+1\right)/2$ to find

$$\begin{aligned}\sum_{i=1}^{n} \mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) &= \sum_{i=1}^{n}\left(c_\alpha i + c_\beta\right) + n\mathfrak{C}\left(\lambda\underline{\boldsymbol{a}}\right) \\ &= \frac{c_\alpha n^2}{2} + \left(\frac{c_\alpha}{2} + c_\beta + \mathfrak{C}\left(\lambda\underline{\boldsymbol{a}}\right)\right) n. \end{aligned} \tag{5.64}$$

When considering the cost in terms of multiplications, $c_\alpha = 237$ and $c_\beta = -48$. Therefore,

$$\underset{\text{mult}}{\mathfrak{C}}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}}\right) \triangleq \sum_{i=1}^{n} \underset{\text{mult}}{\mathfrak{C}}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = 118.5n^2 + 78.5n.$$

Analogously, when considering the cost in terms of additions, $c_\alpha = 184$ and $c_\beta = -40$. Hence,

$$\underset{\text{add}}{\mathfrak{C}}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}}\right) \triangleq \sum_{i=1}^{n} \underset{\text{add}}{\mathfrak{C}}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = 92n^2 + 52n.$$

Analogously, from (5.62) we let $\mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{x}}_{c_i}^{0}}\right) + 2\mathfrak{C}\left(\underset{6\times 8}{\boldsymbol{A}}\ \underset{8\times i}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{6\times i}{\boldsymbol{A}} + \underset{6\times i}{\boldsymbol{B}}\right) = c_\gamma i + c_\lambda$ to find an expression identical to (5.64), in which c_α and c_β are replaced by c_γ and c_λ, respectively. When considering the cost in terms of multiplications, $c_\gamma = 408$ and $c_\lambda = 0$, whereas in terms of additions, $c_\gamma = 358$ and $c_\lambda = -8$. Hence,

$$\underset{\text{mult}}{\mathfrak{C}}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}}\right) \triangleq \sum_{i=1}^{n} \underset{\text{mult}}{\mathfrak{C}}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = 204n^2 + 212n,$$

$$\underset{\text{add}}{\mathfrak{C}}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}}\right) \triangleq \sum_{i=1}^{n} \underset{\text{add}}{\mathfrak{C}}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right) = 179n^2 + 171n.$$

The costs for calculating all Jacobians matrices are summarized in Table 5.2.

Table 5.2: Number of operations in different parts of the Jacobians and its derivatives

	Mult.	Add.
$\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{x}}_{c_i}^{0}}\right)$ (Adorno, 2011)	$189i - 48$	$142i - 40$
$\mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{x}}_{c_i}^{0}}\right)$(Adorno, 2011)	$312i$	$268i - 8$
$\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right)$	$237i - 40$	$184i - 40$
$\mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}\right)$	$408i + 8$	$358i - 8$
$\mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}}\right)$	$118.5n^2 + 78.5n$	$92n^2 + 52n$
$\mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}}\right)$	$204n^2 + 212n$	$179n^2 + 171n$

To obtain the computational cost of (5.15), we define $\bar{\boldsymbol{M}}_i \triangleq \bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}}^{T} \boldsymbol{\Psi}_{c_i} \bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}_{0,c_i}^{c_i}} \in \mathbb{R}^{i\times i}$ such that

$$\mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right) = \mathfrak{C}\left(\underset{i\times 6}{\boldsymbol{A}}\ \underset{6\times 6}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{i\times 6}{\boldsymbol{A}}\ \underset{6\times i}{\boldsymbol{B}}\right). \tag{5.65}$$

Since

$$\boldsymbol{M}_i = \begin{bmatrix} \bar{\boldsymbol{M}}_i & \boldsymbol{0}_{i\times(n-i)} \\ \boldsymbol{0}_{(n-i)\times i} & \boldsymbol{0}_{(n-i)\times(n-i)} \end{bmatrix} \in \mathbb{R}^{n\times n},$$

then $\boldsymbol{M}_{\mathrm{GP}} = \sum_{i=1}^{n} \boldsymbol{M}_i$, whose cost is given by

$$\mathfrak{C}\left(\boldsymbol{M}_{\mathrm{GP}}\right) = \sum_{i=1}^{n} \mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right) + (n-1)\,\mathfrak{C}\left(\underset{n\times n}{\boldsymbol{A}} + \underset{n\times n}{\boldsymbol{B}}\right). \tag{5.66}$$

To calculate the computational cost of (5.16), we define the matrix

$$\bar{\boldsymbol{C}}_i \triangleq \bar{\boldsymbol{J}}^{T}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} \left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}^{c_i}_{0,c_i}, \boldsymbol{\Psi}_{c_i}\right) \bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} + \boldsymbol{\Psi}_{c_i} \dot{\bar{\boldsymbol{J}}}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}}\right) \in \mathbb{R}^{i\times i},$$

whose calculation cost is given by

$$\begin{aligned}\mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right) = \mathfrak{C}\left(\overline{\boldsymbol{S}}\left(\boldsymbol{\omega}^{c_i}_{0,c_i}, \boldsymbol{\Psi}_{c_i}\right)\right) + 2\mathfrak{C}\left(\underset{6\times 6}{\boldsymbol{A}}\,\underset{6\times i}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{6\times i}{\boldsymbol{A}} + \underset{6\times i}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{i\times 6}{\boldsymbol{A}}\,\underset{6\times i}{\boldsymbol{B}}\right) \\ + \mathfrak{C}\left(\underset{6\times i}{\boldsymbol{A}} + \underset{6\times i}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\underset{i\times 6}{\boldsymbol{A}}\,\underset{6\times i}{\boldsymbol{B}}\right).\end{aligned} \tag{5.67}$$

Considering the matrix

$$\boldsymbol{C}_i = \begin{bmatrix} \bar{\boldsymbol{C}}_i & \boldsymbol{0}_{i\times(n-i)} \\ \boldsymbol{0}_{(n-i)\times i} & \boldsymbol{0}_{(n-i)\times(n-i)} \end{bmatrix} \in \mathbb{R}^{n\times n},$$

we have $\boldsymbol{C}_{\mathrm{GP}} = \sum_{i=1}^{n} \boldsymbol{C}_i$ with corresponding cost given by

$$\mathfrak{C}\left(\boldsymbol{C}_{\mathrm{GP}}\right) = \sum_{i=1}^{n} \mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right) + (n-1)\,\mathfrak{C}\left(\underset{n\times n}{\boldsymbol{A}} + \underset{n\times n}{\boldsymbol{B}}\right). \tag{5.68}$$

To calculate $\sum_{i=1}^{n} \mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right)$ and $\sum_{i=1}^{n} \mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right)$ in (5.66) and (5.68), respectively, we must realize that the summands $\mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right)$ and $\mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right)$ have similar structure, such as $\mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right) = c_{\alpha_C} i^2 + c_{\beta_C} i + c_{\gamma_C}$.[9] Therefore,

$$\begin{aligned}\sum_{i=0}^{n} \mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right) &= c_{\alpha_C} \sum_{i=0}^{n} i^2 + c_{\beta_C} \sum_{i=0}^{n} i + c_{\gamma_C} \sum_{i=0}^{n} 1 \\ &= c_{\alpha_C} \frac{(n^2+n)(2n+1)}{6} + c_{\beta_C} \frac{n\,(n+1)}{2} + c_{\gamma_C} n,\end{aligned} \tag{5.69}$$

where used the fact that $\sum_{i=0}^{n} i^2 = (n^2+n)(2n+1)\,/6$ (Beardon, 1996). Using (5.68) and

[9] Expanding (5.67), we obtain $c_{\alpha_C} = 6$, $c_{\beta_C} = 72$, and $c_{\gamma_C} = 18$ for the number of multiplications in $\mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right)$ and $c_{\alpha_C} = 5$, $c_{\beta_C} = 66$, and $c_{\gamma_C} = 6$ for the number of additions. An analogous reasoning can be done for $\mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right)$.

(5.69), we obtain the final cost for computing the Coriolis matrix in (5.16). An analogous reasoning is done to obtain the final cost (5.66) for computing the inertia matrix in (5.15).

Last, to obtain the cost of calculating (5.19), we define the vector

$$\bar{\boldsymbol{g}}_i \triangleq \bar{\boldsymbol{J}}^T_{\mathcal{D}(\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i})} \operatorname{vec}_3 \left(-\operatorname{Ad}\left(\boldsymbol{r}_0^{c_i}\right) \boldsymbol{f}_{g_i}\right) \in \mathbb{R}^{i\times 1},$$

where $\bar{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i}} = \begin{bmatrix} \bar{\boldsymbol{J}}^T_{\mathcal{P}(\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i})} & \bar{\boldsymbol{J}}^T_{\mathcal{D}(\underline{\boldsymbol{\xi}}^{c_i}_{0,c_i})} \end{bmatrix}^T$, and whose calculation cost is given by

$$\begin{aligned} \mathfrak{C}\left(\bar{\boldsymbol{g}}_i\right) &= \mathfrak{C}\left(\underset{i\times 3}{\boldsymbol{A}}\ \underset{3\times 1}{\boldsymbol{B}}\right) + \mathfrak{C}\left(\operatorname{Ad}\left(\boldsymbol{a}\right)\boldsymbol{b}\right) \\ &= c_{\alpha_g} i + c_{\beta_g}, \end{aligned} \tag{5.70}$$

in which c_{α_g} and c_{β_g} are constants that are found by inspection.[10] Considering $\boldsymbol{g}_i = \begin{bmatrix} \bar{\boldsymbol{g}}_i^T & \boldsymbol{0}^T_{(n-i)} \end{bmatrix}^T \in \mathbb{R}^{n\times 1}$, then $\boldsymbol{g}_{\mathrm{GP}} = \sum_{i=1}^{n} \boldsymbol{g}_i$, with cost given by

$$\begin{aligned} \mathfrak{C}\left(\boldsymbol{g}_{\mathrm{GP}}\right) &= \sum_{i=1}^{n} \mathfrak{C}\left(\bar{\boldsymbol{g}}_i\right) + (n-1)\,\mathfrak{C}\left(\underset{n\times 1}{\boldsymbol{A}} + \underset{n\times 1}{\boldsymbol{B}}\right) \\ &= c_{\alpha_g} \frac{n\,(n+1)}{2} + c_{\beta_g} n + (n-1)\,\mathfrak{C}\left(\underset{n\times 1}{\boldsymbol{A}} + \underset{n\times 1}{\boldsymbol{B}}\right). \end{aligned} \tag{5.71}$$

The costs (5.65)–(5.71) are summarized in Table 5.3, with their explicit values in terms of additions and multiplications of real numbers.

Finally, the total cost of obtaining $\underline{\boldsymbol{\tau}}_{\mathrm{GP}}$ in (5.20) is given by

$$\begin{aligned} \mathfrak{C}\left(\mathrm{GP_{DQ}}\right) = \mathfrak{C}\left(\boldsymbol{M}_{\mathrm{GP}}\right) + \mathfrak{C}\left(\boldsymbol{C}_{\mathrm{GP}}\right) + \mathfrak{C}\left(\boldsymbol{g}_{\mathrm{GP}}\right) + \mathfrak{C}\left(\boldsymbol{J}_{\underline{\boldsymbol{\xi}}}\right) + \mathfrak{C}\left(\dot{\boldsymbol{J}}_{\underline{\boldsymbol{\xi}}}\right) + 2\mathfrak{C}\left(\underset{n\times n}{\boldsymbol{A}}\ \underset{n\times 1}{\boldsymbol{B}}\right) \\ + 2\mathfrak{C}\left(\underset{n\times 1}{\boldsymbol{A}} + \underset{n\times 1}{\boldsymbol{B}}\right), \end{aligned}$$

for which the explicit values, in terms of number of multiplications and additions involved, is presented in Table 5.4. Furthermore, Table 5.4 presents the cost comparison between the proposed algorithm 5.1 and their classic counterparts, in terms of number of multiplications and additions of real numbers.

[10] In case of multiplications, $c_{\alpha_g} = 3$ and $c_{\beta_g} = 35$; in case of additions, $c_{\alpha_g} = 2$ and $c_{\beta_g} = 24$.

Table 5.3: Number of operations in the Euler-Lagrange Description

	Mult.	Add.
$\mathfrak{C}\left(\bar{\boldsymbol{M}}_i\right)$	$6i^2 + 36i$	$5i^2 + 30i$
$\mathfrak{C}\left(\bar{\boldsymbol{C}}_i\right)$	$6i^2 + 72i + 18$	$5i^2 + 66i + 6$
$\mathfrak{C}\left(\bar{\boldsymbol{g}}_i\right)$	$3i + 35$	$2i + 24$
$\mathfrak{C}\left(\boldsymbol{M}_{\mathrm{GP}}\right)$	$2n^3 + 21n^2 + 19n$	$\frac{8}{3}n^3 + \frac{33}{2}n^2 + \frac{95}{6}n$
$\mathfrak{C}\left(\boldsymbol{C}_{\mathrm{GP}}\right)$	$2n^3 + 39n^2 + 55n$	$\frac{8}{3}n^3 + \frac{69}{2}n^2 + \frac{239}{6}n$
$\mathfrak{C}\left(\boldsymbol{g}_{\mathrm{GP}}\right)$	$1.5n^2 + 36.5n$	$2n^2 + 24n$

Table 5.4: Cost comparison between the proposed method and their classic counterparts for obtaining the dynamical model for an n-DOF serial robot.

Method	Mult.	Add.
Dual Quaternion Euler-Lagrange algorithm using Gauss's Principle of Least Constraint	$4n^3 + 386n^2 + 401n$	$\frac{16}{3}n^3 + 326n^2 + \frac{908}{3}n$
Classic Euler-Lagrange algorithm (Hollerbach, 1980)	$412n - 277$	$320n - 201$
Dual Quaternion Newton-Euler algorithm (cost for arbitrary joints) (Silva et al., 2020)	$882n - 48$	$724n - 40$
Classic Newton-Euler algorithm (Balafoutis, 1994)	$150n - 48$	$131n - 48$

5.6 Conclusions

The first part of this chapter presented the Gauss's Principle of Least Constraint (GPLC) using dual quaternion algebra, which is a contribution of this thesis. This strategy is stated as a least-squares minimization problem and leads to the Euler-Lagrange dynamic description of a mechanical system.

Different from the work of Wieber (2006) and Bouyarmane & Kheddar (2012), the formulation presented in this chapter relies on dual quaternions. This enables a high-level mathematical abstraction and provides a compact and unified representation.

Although the matrix representation of dual quaternions is used, all aforementioned advantages are still valid because operations in the matrix form respect the group operations of $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$, and therefore, both representations are equivalent. In addition, the matrix form allows a minimization over the field of real vectors. This enables the use of well-established analytical and numerical strategies to solve the least-squares minimization problem, which are not available for dual quaternion elements.

The second part of the chapter presented the connections between Gauss's principle, Gibbs-Appell equations, and Kane's method using the tools from dual quaternion algebra. By selecting the quasi-velocities to be the same as the generalized velocities, the chapter showed that the Euler-Lagrange dynamic description of a mechanical system is a particular case of the Gibbs-Appell and Kane's equations. Although the three methods deal with nonholonomic constraints without the necessity of Lagrange multipliers, only Gauss's principle does not require setting up quasi-velocities and allows taking into account additional constraints directly in the optimization formulation.

Next, the chapter showed how to impose bilateral constraints in the Gauss's principle by means of the Udwadia-Kalaba formulation (Kalaba & Udwadia, 1992). In addition, an example about how to model a humanoid robot including additional constraints and using the Euler-Lagrange equation based on the Gauss's Principle is presented.

Last, the chapter presented the computational cost of the proposed algorithm for a robot manipulator and presented a comparison with their classic counterparts in terms of number of multiplications, additions, and trigonometric operations involved in each method. The Euler-Lagrange method based on the Gauss's Principle of Least Constraint is, as expected, more expensive than the ones based on the Newton-Euler and classic Euler-Lagrange formalism since it is not based on recursive strategies. However, this strategy allows taking into account additional constraints in the accelerations, which can be exploited, for instance, in nonholonomic robotic systems. Furthermore, we made no hard attempt, if any, to optimize our implementation since the main interest, at this moment, are theoretical aspects of the dynamic modeling using dual quaternion algebra rather than in ensuring computational efficiency.

6

Simulations and Experimental Results

This chapter presents some simulations and experimental results performed to validate the proposed strategies related to SOVFIs (discussed in Chapter 4) and the Gauss's Principle of Least Constraint based on dual quaternion algebra (discussed in Chapter 5).

First, some simulations were performed to assess the VFIs and SOVFIs framework in a humanoid robot composed of the upper body only. The constraints are used to prevent collisions and self-collisions. Furthermore, the conic constraint for VFIs (4.35) and SOVFIs (4.36) are used to prevent undesired end-effector orientations and violation of joint limits. In addition, some experiments using Python and ROS were performed using real robots commanded by joint velocities. Specifically, we used a 9-DOF humanoid robot (the 5-DOF torso serialized with a 4-DOF arm), a 13-DOF bimanual manipulator, and finally a 15-DOF nonholonomic mobile bimanual manipulator.

Also, simulations were performed using a 27-DOF full humanoid robot to evaluate the multi-contact strategy and the maximization of the SP area, which were discussed in Section 4.5 and Section 4.4, respectively.

Finally, simulations were performed in Matlab and V-REP to validate the Euler-Lagrange model obtained using the dual quaternion Gauss's Principle of Least Constraint (DQGP) in both holonomic and nonholonomic robots.

6.1 (Self)-Collision Avoidance and Conic Constraint Tests

We performed simulations on V-REP[1] and Matlab,[2] and real experiments using ROS[3] and Python. Furthermore, we used DQ Robotics (Adorno & Marques Marinho, 2020) for both robot modeling and the description of the geometric primitives.

To illustrate the effect of the constraints when are enabled and disabled, we implemented two controllers based on quadratic programming, namely QP and CQP, where QP does not take into account constraints and CQP denotes the constrained case. The control laws used are (3.1) for robots using velocity inputs, and (3.3) for robots commanded by torque inputs. Furthermore, we validated both controllers QP and CQP in simulation and in a real-time implementation on a real robot. In all cases, we used constant safe distances and static primitives such that the residual $\zeta_{\text{safe}}(t)$ in both 4.1 and 4.6 are equal to zero.

The goal is to control the left-hand of a humanoid robot to put a cup on a table while keeping the cup tilting below threshold (constraint (4.33)), avoiding reaching joint limits (one constraint (4.33) for each joint) and preventing self-collisions (constraint (4.37)) and collisions with the workspace (constraint (4.39)), as shown in Fig. 6.1. To accomplish this goal, the control law minimizes the distance between the left-hand and the table, which is modeled as a plane, and the control inputs take into account the kinematic chain composed of the 5-DOF torso and the 4-DOF left arm, which has a total of nine DOF. The safe angle, $\phi_{\text{safe}} = 0.1\text{rad}$, denotes the cup maximum tilting angle, as shown in Fig. 4.5. We defined four additional planes, as shown in Fig. 4.8, to prevent reaching the table from below, where $d_{\pi,\text{safe}} = 0.01$ for all four planes. Furthermore, the safe distance between the arm and the torso is $d_{\text{safe}} \triangleq d_{\text{tarm}} = 0.07$. The same setup was used for the simulation using velocity inputs and the one using torque inputs.

6.1.1 Task space control: Joint velocity inputs

For both QP and CQP, the objective function is defined as in (3.1), where $\tilde{\boldsymbol{x}} \triangleq d_{p,n_{\underline{\boldsymbol{\pi}}}}$ is the distance between the end-effector and the table, and $\boldsymbol{J} \triangleq \boldsymbol{J}_{p,n_{\underline{\boldsymbol{\pi}}}}$ is the corresponding point-static-plane Jacobian matrix (see Table 3.1). Furthermore, the convergence parameter ($\eta = 0.36$) and the damping factor ($\lambda = 0.01$) were defined empirically in order to provide a smooth and fast enough convergence rate.

In the constrained case (CQP), the control inputs $\boldsymbol{u} \triangleq \dot{\boldsymbol{q}}$ are computed using (3.1) with the aforementiond task-Jacobian and error function, where

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{J}^T_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}} & -\boldsymbol{J}^T_{p,n_{\underline{\boldsymbol{\pi}}_{\mathcal{I}}}} & \boldsymbol{J}^T_{\phi_{\underline{\boldsymbol{l}}_{z\mathcal{I}},\underline{\boldsymbol{l}}_{\mathcal{I}}}} & -\boldsymbol{J}^T_{\text{tarm}} \end{bmatrix}^T$$

[1] http://www.coppeliarobotics.com/
[2] https://www.mathworks.com/
[3] https://www.ros.org/

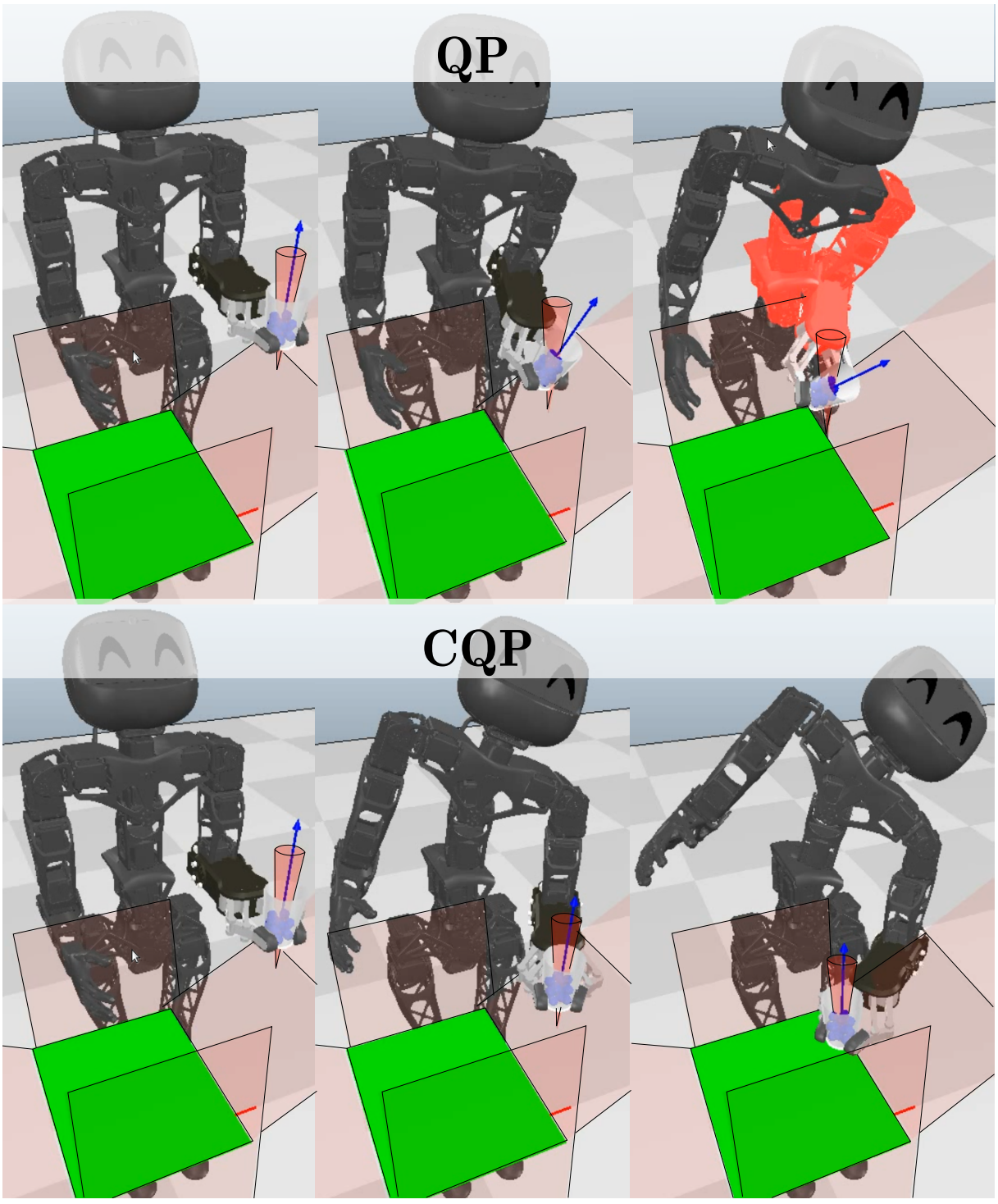

Figure 6.1: Control of the left-hand using the robot kinematics and first-order VFIs (snapshots of the simulation). On *top,* only the plane constraints were used, whereas on *bottom* all constraints were used. The *blue axis* denotes the cup orientation and the *red cone* represents the maximum allowable tilting for the *blue axis*. Red-shaded body parts indicate a collision.

with

$$\boldsymbol{J}_{p,n_{\underline{\pi}_{\mathcal{I}}}} = \begin{bmatrix} \boldsymbol{J}^T_{p,n_{\underline{\pi}_1}} & \cdots & \boldsymbol{J}^T_{p,n_{\underline{\pi}_4}} \end{bmatrix}^T, \tag{6.1}$$

$$\boldsymbol{J}_{\phi_{\underline{l}_{z\mathcal{I}},\underline{l}_{\mathcal{I}}}} = \begin{bmatrix} \boldsymbol{J}^T_{\phi_{\underline{l}_{z_1},\underline{l}_1}} & \cdots & \boldsymbol{J}^T_{\phi_{\underline{l}_{z_9},\underline{l}_9}} \end{bmatrix}^T, \tag{6.2}$$

and $\boldsymbol{w} = \eta \begin{bmatrix} -\tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right) & \tilde{\boldsymbol{d}}^T_{p,n_{\underline{\pi}_{\mathcal{I}}}} & -\tilde{\boldsymbol{F}}^T & \tilde{d}_{\text{tarm}} \end{bmatrix}^T$ with

$$\tilde{\boldsymbol{d}}_{p,n_{\underline{\pi}_{\mathcal{I}}}} = \begin{bmatrix} \tilde{d}_{p,n_{\underline{\pi}_1}} & \cdots & \tilde{d}_{p,n_{\underline{\pi}_4}} \end{bmatrix}^T, \tag{6.3}$$

$$\tilde{\boldsymbol{F}} = \begin{bmatrix} \tilde{f}\left(\phi_{\underline{l}_{z_1},\underline{l}_1}\right) & \cdots & \tilde{f}\left(\phi_{\underline{l}_{z_9},\underline{l}_9}\right) \end{bmatrix}^T. \tag{6.4}$$

Eq. (6.1) and (6.3) are used to enforce the four plane constraints (recall (4.39)), whereas (6.2) and (6.4) are used to avoid joint limits in all nine joints.

Figure 6.1 shows the simulation snapshots. Although the task is fulfilled for both controllers, only CQP prevents undesired cup orientations and respects all other constraints as shown in Fig. 6.2.

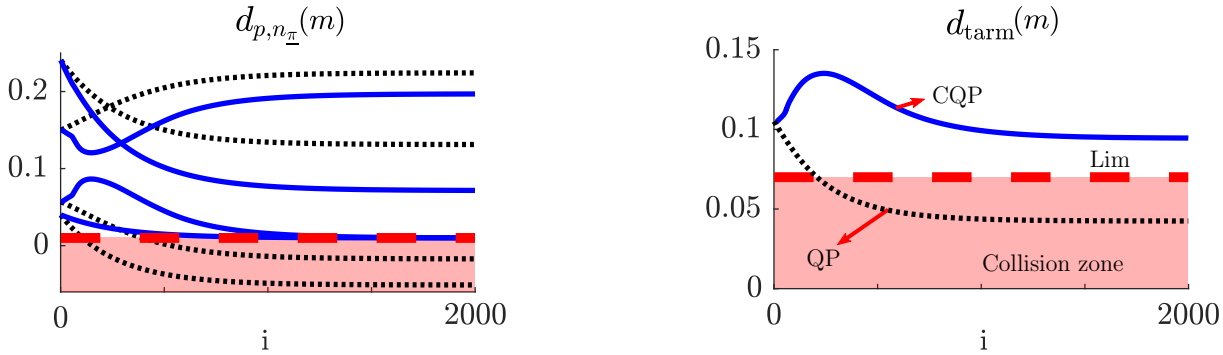

Figure 6.2: Control of the left-hand in simulation using the robot kinematics and first-order VFIs. The *dashed-black* line corresponds to QP and the *solid-blue line* denotes the CQP. On the *left*, the distance $d_{p,n_{\underline{\pi}}}$ between the robot hand and the four lateral planes. On the *right*, the distance d_{tarm} between the torso and the arm. The horizontal axis are iterations. The vertical axis are distances in meters.

6.1.2 Task Space Control: Joint torque inputs

We performed a second simulation using the robot dynamic model with second-order VFIs. V-REP was used only as a visualization and collision-checking tool. We assume perfect knowledge of the inertial parameters, which are obtained directly from V-REP. The minimization is performed in the joint accelerations using formulation (3.3) with the constraints 3.4, where $\boldsymbol{J}$ and $\tilde{\boldsymbol{x}}$, which are needed to obtain $\boldsymbol{\beta}$, are the same as in Section 6.1.1. In addition, all constraints are rewritten as in (4.6) and (4.8) to obtain $\boldsymbol{W}$ and $\boldsymbol{w}$, where $\boldsymbol{W}$ is exactly the same as in the first-order case and $\boldsymbol{w} = \begin{bmatrix} -\beta_{\phi_{\underline{l}_z,\underline{l}}} & \boldsymbol{\beta}^T_{p,n_{\underline{\pi}_{\mathcal{I}}}} & -\boldsymbol{\beta}^T_{\text{joints}} & \beta_{\text{tarm}} \end{bmatrix}^T$,

with $\boldsymbol{\beta}_{p,n_{\underline{\pi}_\mathcal{I}}} = \begin{bmatrix} \beta_{p,n_{\underline{\pi}_1}} & \cdots & \beta_{p,n_{\underline{\pi}_4}} \end{bmatrix}^T$ and $\boldsymbol{\beta}_{\text{joints}} = \begin{bmatrix} \beta_{\phi_{\underline{l}_{z_1},\underline{l}_1}} & \cdots & \beta_{\phi_{\underline{l}_{z_9},\underline{l}_9}} \end{bmatrix}^T$. Furthermore, $k_d = \eta_1 = 1.5$, $k_p = \eta_2 = 0.3$, and $g\left(\dot{\tilde{\boldsymbol{x}}}\right) \triangleq \left\|\dot{\tilde{\boldsymbol{x}}}\right\|_2$. Finally, the control inputs $\boldsymbol{u} \triangleq \boldsymbol{\tau}$ are computed using (3.7) where the dynamic description is obtained using the Gauss's Principle of Least Constraint, see chapter 5.

Fig. 6.3 shows that both QP and CQP minimize the task error and the joint accelerations, as expected. The joint accelerations in CQP, however, are more abrupt because the collision-avoidance constraints enforce abrupt changes in the robot velocities to prevent collisions. Because of that, CQP required a higher control effort, as shown in Table 6.1.

Table 6.1: Control of the left-hand: control effort.

Control law	Metric	*QP*	*CQP*
Problem (3.1)	$\sqrt{\int_0^\infty \|\dot{\mathbf{q}}(\mathbf{t})\|_2^2 \, dt}$	0.1637	0.5132
Problem (3.7)	$\sqrt{\int_0^\infty \|\boldsymbol{\tau}(\mathbf{t})\|_2^2 \, dt}$	61.508	115.07

Since CQP enforces the constraints $\boldsymbol{\gamma}_l \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}_u$, where $\boldsymbol{\gamma}_l$ and $\boldsymbol{\gamma}_u$ are calculated according to (3.4), the joint velocities go to zero as the end-effector stops. As QP does not enforce those constraints, the robot joints continue to move after the end-effector stops, as shown in the third graph of Fig. 6.3.

Finally, the cup maximum tilting angle is respected, as well as the other constraints for collision avoidance and joint limit avoidance, as shown in Fig. 6.4, only when CQP is used, as expected.

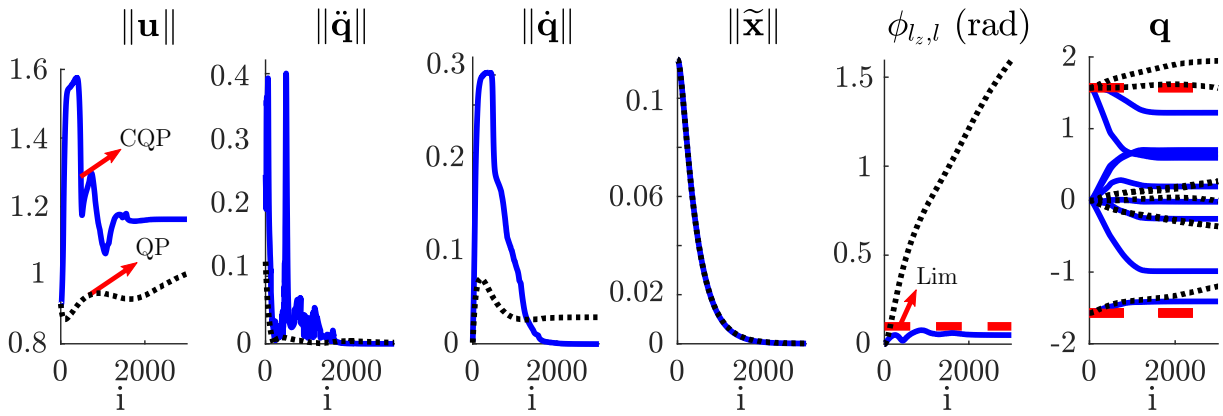

Figure 6.3: Control of the left-hand using the robot dynamics and second-order VFIs on simulations. The *dashed-black* line corresponds to QP and the *solid-blue line* denotes the CQP. From left to right: norm of the control inputs (torques); norm of the joint accelerations; norm of the joint velocities; norm of the task error; angle $\phi_{\underline{l}_z,\underline{l}}$ and the joints positions of the torso and the left arm. The *red-dashed line* in the last two graphs corresponds to the maximum allowable tilting angle ϕ_{safe} and the joint limits. The horizontal axis are iterations.

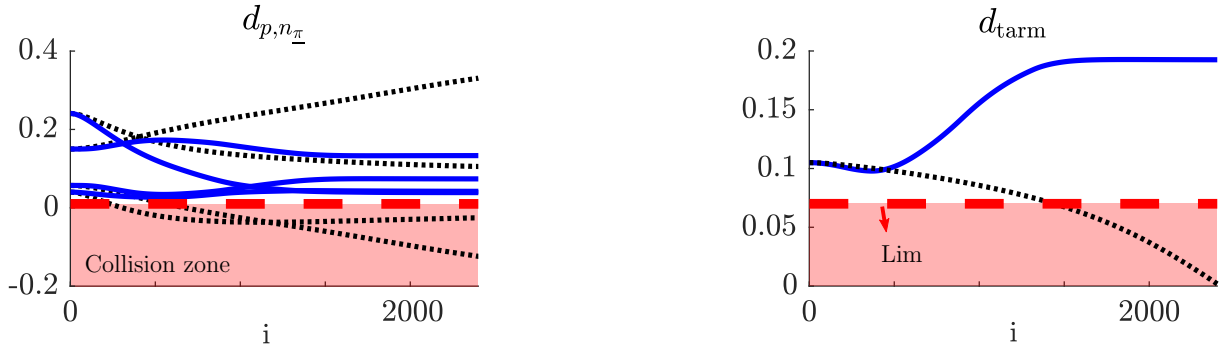

Figure 6.4: Control of the left-hand using the robot dynamics and second-order VFIs on simulations. The *dashed-black* line corresponds to QP and the *solid-blue line* denotes the CQP. On the *left*, the distance $d_{p,n_{\underline{\pi}}}$ between the robot hand and the four lateral planes. On the *right*, the distance d_{tarm} between the torso and the arm. The horizontal axis are iterations. The vertical axis are distances in meters.

6.1.3 Experimental Results

We performed a real experiment using the platform composed of the upper body of the Poppy humanoid robot.[4] We used Python and ROS Melodic in a computer running Ubuntu 18.04 64 bits, equipped with a Intel i7 4712HQ with 16GB RAM, in addition to quadprog[5] to solve (3.1). The solver required about $191\mu\text{s} \pm 47\mu\text{s}$ to generate the control inputs in the constrained case. Furthermore, it was required about $5.2\text{ms} \pm 1\text{ms}$ to compute 15 constraint Jacobians (one angle constraint, four plane constraints, one torso-left-arm constraint, and nine joint limit constraints). However, we set the loop rate control to 20ms due to low-level drivers limitations.

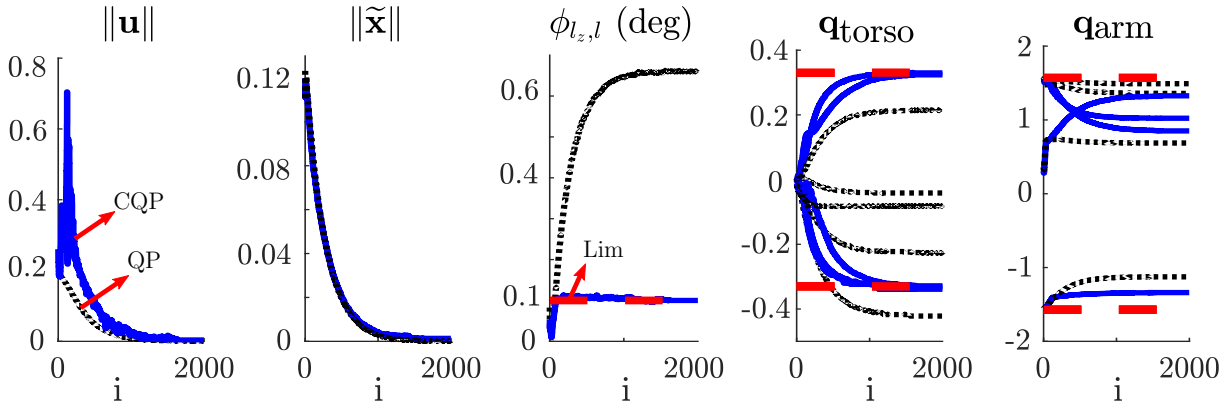

Figure 6.5: Control of the left-hand using the robot kinematics and first-order VFIs using the real robot. From left to right: norm of the control inputs (joint velocities); norm of the task error; the angle $\phi_{\underline{l}_z,\underline{l}}$ between the cup centerline and a vertical line, and the joints positions of the torso and the left arm. The *dashed red line* denotes the maximum allowable tilting angle and the joint limits. The horizontal axis are iterations.

We used the same task specification and parameters defined in Sections 6.1 and 6.1.1.

[4] https://www.poppy-project.org/en/robots/poppy-humanoid
[5] https://pypi.org/project/quadprog/

All collision-avoidance constraints were activated, for both QP and CQP, to ensure the robot safety. This way, the only difference between them is that QP does not have the cup tilting angle constraint whereas CQP has. Fig. 6.5 shows the task error decay and the angle $\phi_{\underline{l}_z,\underline{l}}$ for both controllers. In both cases, the task is fulfilled with the same convergence rate, which is determined by η. However, only CQP respects the angle constraint and the limit joints constraints, as expected.

Figure 6.6 and 6.7 shows that, in both cases, the end-effector reached the target zone, but only CQP does not violate the cup tilting angle constraint, as expected.

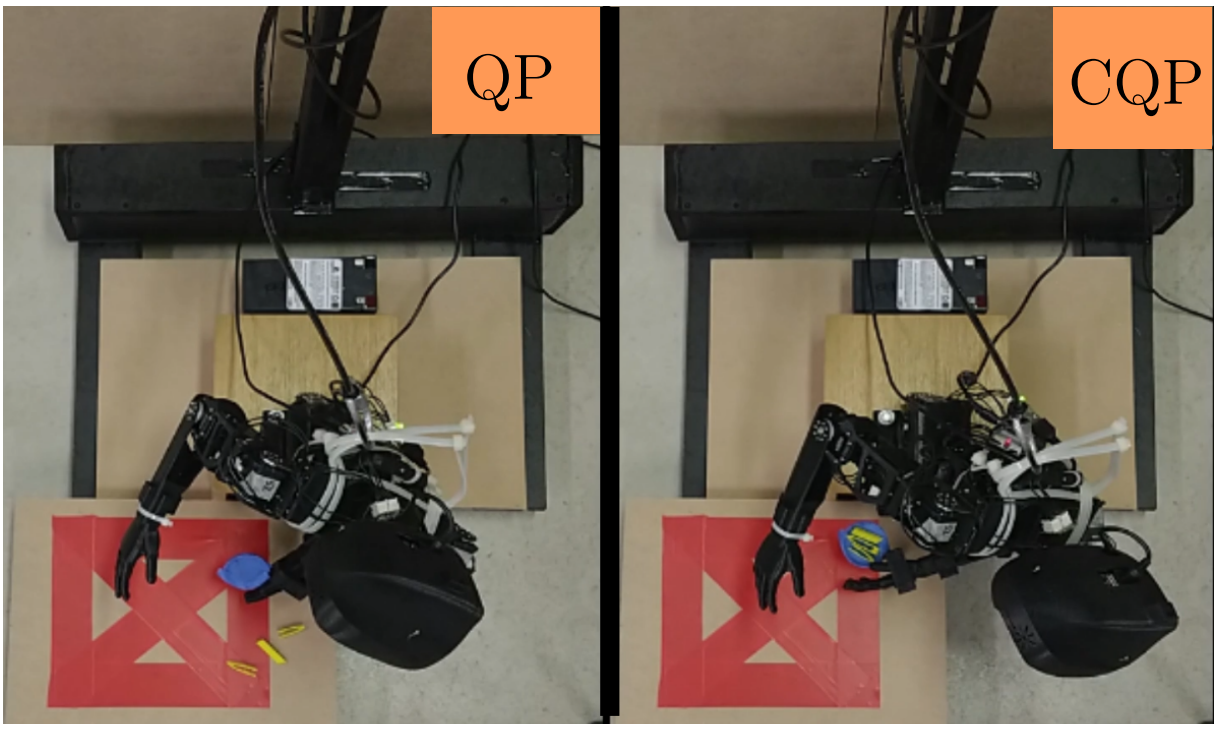

Figure 6.6: Control of the left-hand using the real robot (top view of the real-time experiment in the final configuration). On the *left*, the cup tilting angle constraint is disabled, whereas on the *right* this constraint is enabled. The red square denotes the target region

We validated the proposed conic constraint discussed in Section. 2.3 using the Cooperative Dual Task Space (CDTS) framework (Adorno, 2011) in a bimanual manipulation using the same parameters and task specification,[6] as shown in Fig. 6.8. The difference here is that the cup tilting angle constraint was imposed in the orientation of the absolute cooperative variable $\underline{\boldsymbol{x}}_{\text{abs}}$ instead of the left hand orientation. We imposed, in addition, the constraint $\underline{\dot{\boldsymbol{x}}}_{\text{rel}} = 0$, which is required to keep constant the relative pose between the hands.

Figure. 6.9 shows the snapshots of the real-time experiment and Fig. 6.10 shows that in both cases the task is fulfilled but only the CQP respects the angle constraint and the limit joints constraints, as expected.

[6] In this case we use $\phi_{\text{safe}} = 0.3\text{rad}$.

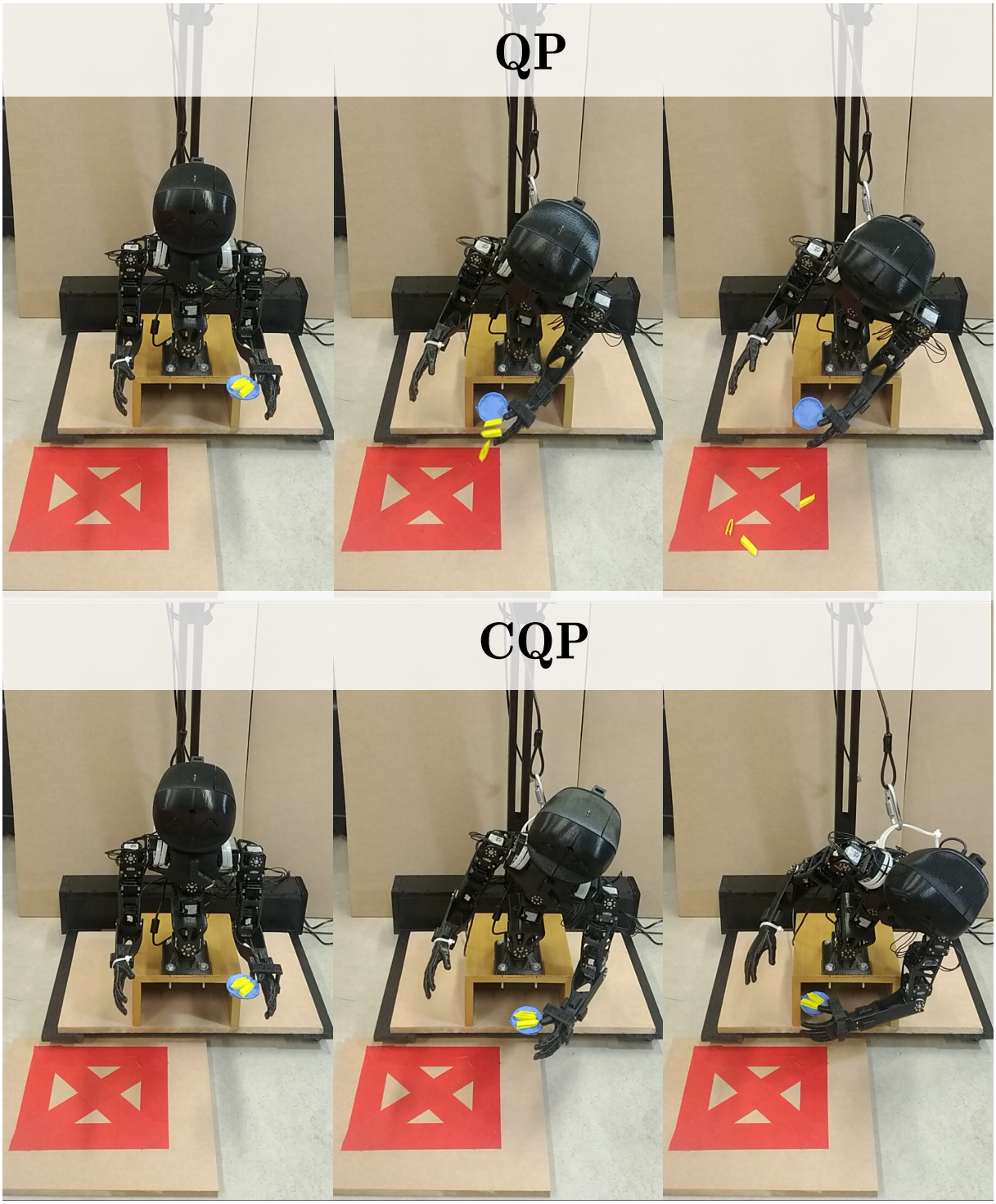

Figure 6.7: Control of the left-hand using the real robot (snapshots of the real-time implementation experiment). On top three snapshots, the cup tilting angle constraint is disabled, whereas on *bottom* three snapshots this constraint is enabled. All other collision-avoidance constraints are activated to ensure the robot safety. The red square denotes the target region.

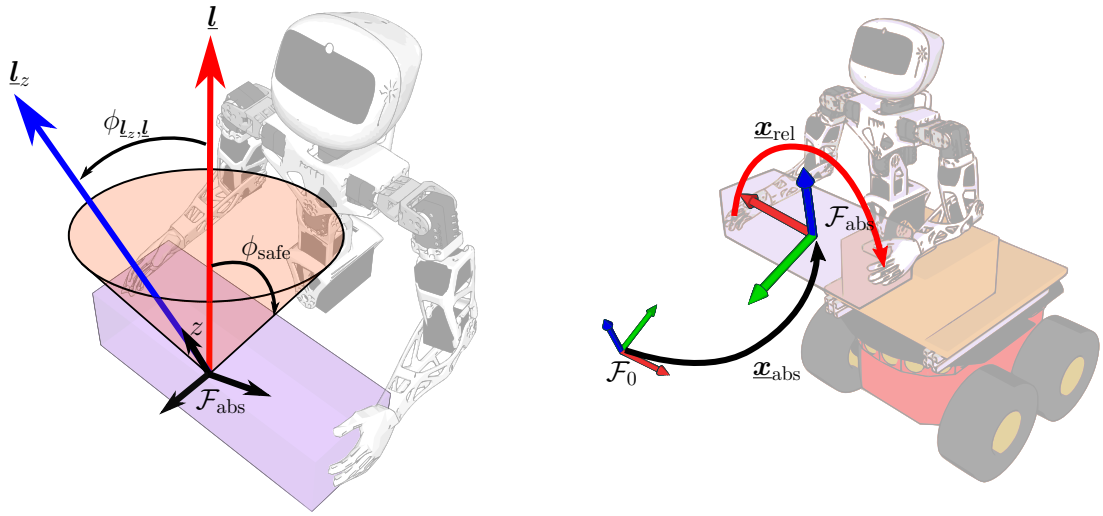

Figure 6.8: Experimental setup using the Cooperative Dual Task Space (CDTS) framework. On the *left*, the the cup tilting angle constraint is imposed in the orientation of the frame $\mathcal{F}_{\text{abs}}$. On the *right*, a nonholonomic mobile bimanual manipulator. The relative pose between the hands is denoted by $\underline{\boldsymbol{x}}_{\text{rel}}$ and the absolute pose of the frame $\mathcal{F}_{\text{abs}}$ is given by $\underline{\boldsymbol{x}}_{\text{abs}}$.

Finally, we used a nonholonomic mobile bimanual manipulator, which is composed of a differential platform and the upper body of the Poppy humanoid robot, as shown in Fig. 6.11. In this case, the goal was to minimize the distance between the absolute cooperative variable and the target zone. We used all collision-avoidance constraints, the limit joints constraints and the relative pose constraint $\dot{\underline{\boldsymbol{x}}}_{\text{rel}} = 0$. In the initial configuration, the relative pose $\underline{\boldsymbol{x}}_{\text{rel}}$ is set to grasp the ball using both hands. This relative pose is maintained fixed all the time by the relative pose constraint. When the robot reaches the target zone, the robot stops and disables all constraints, and finally, the robot releases the ball increasing the distance between the hands. That is the reason why in the third snapshot of Fig. 6.11 the relative pose between the hands is different from the first two snapshots.

Figure .6.12 shows that, in both cases, the frame $\mathcal{F}_{\text{abs}}$ reached the target zone, but only CQP does not violate the joints limits, as expected. The small violations of the joint limits in the CQP (see the third graph of Fig. 6.12) are due to discretization effects and because the movement of the nonholonomic drive robot excites the inertia of the robot, which is not taken into account in the kinematic model. These effects can be minimized using a lower discretization time and working with lower velocities and accelerations.

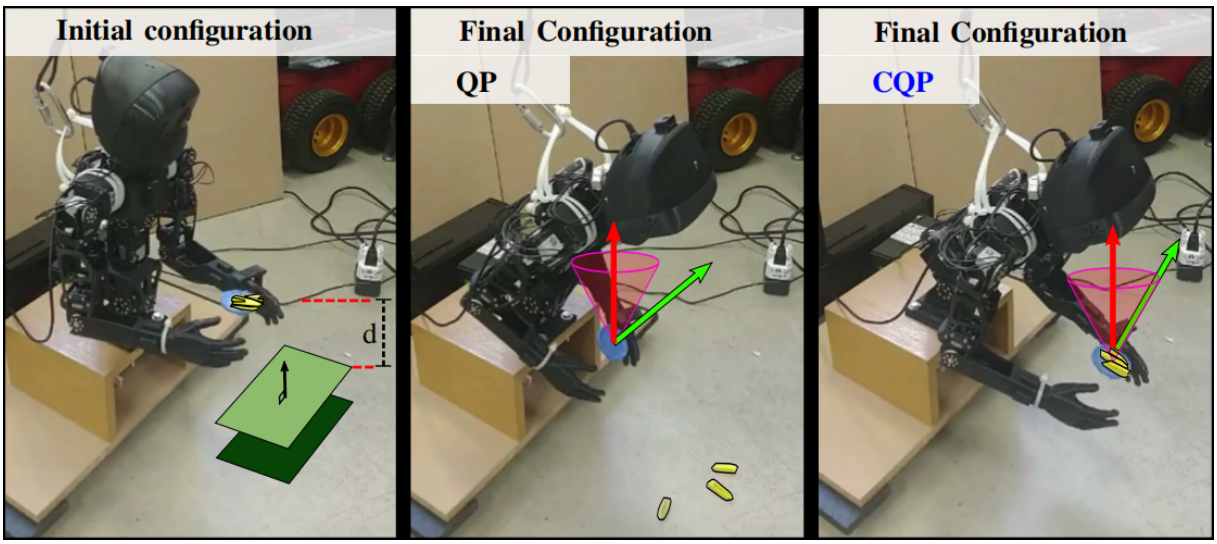

Figure 6.9: Control of both hands using the CDTS framework (snapshots of the real-time experiment). On the *left,* the robot is in the initial configuration. On the *middle,* the cup tilting angle constraint is disabled, whereas on the *right* this constraint is enabled.

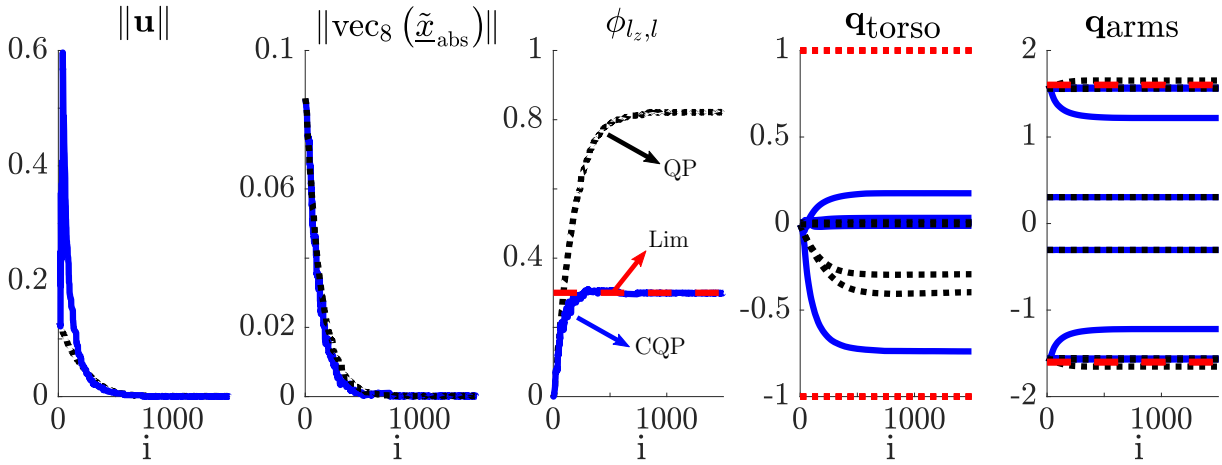

Figure 6.10: Control of both hands using the CDTS framework. From left to right: norm of the control inputs (joint velocities); norm of the task error; the angle $\phi_{\underline{l}_z,\underline{l}}$ between the cup centerline and a vertical line and the joints positions of the torso and the left arm. The *dashed red line* on the third graph denotes the maximum allowable tilting angle whereas in the fourth and fifth graph it correspond to the joint limits. The horizontal axis are iterations.

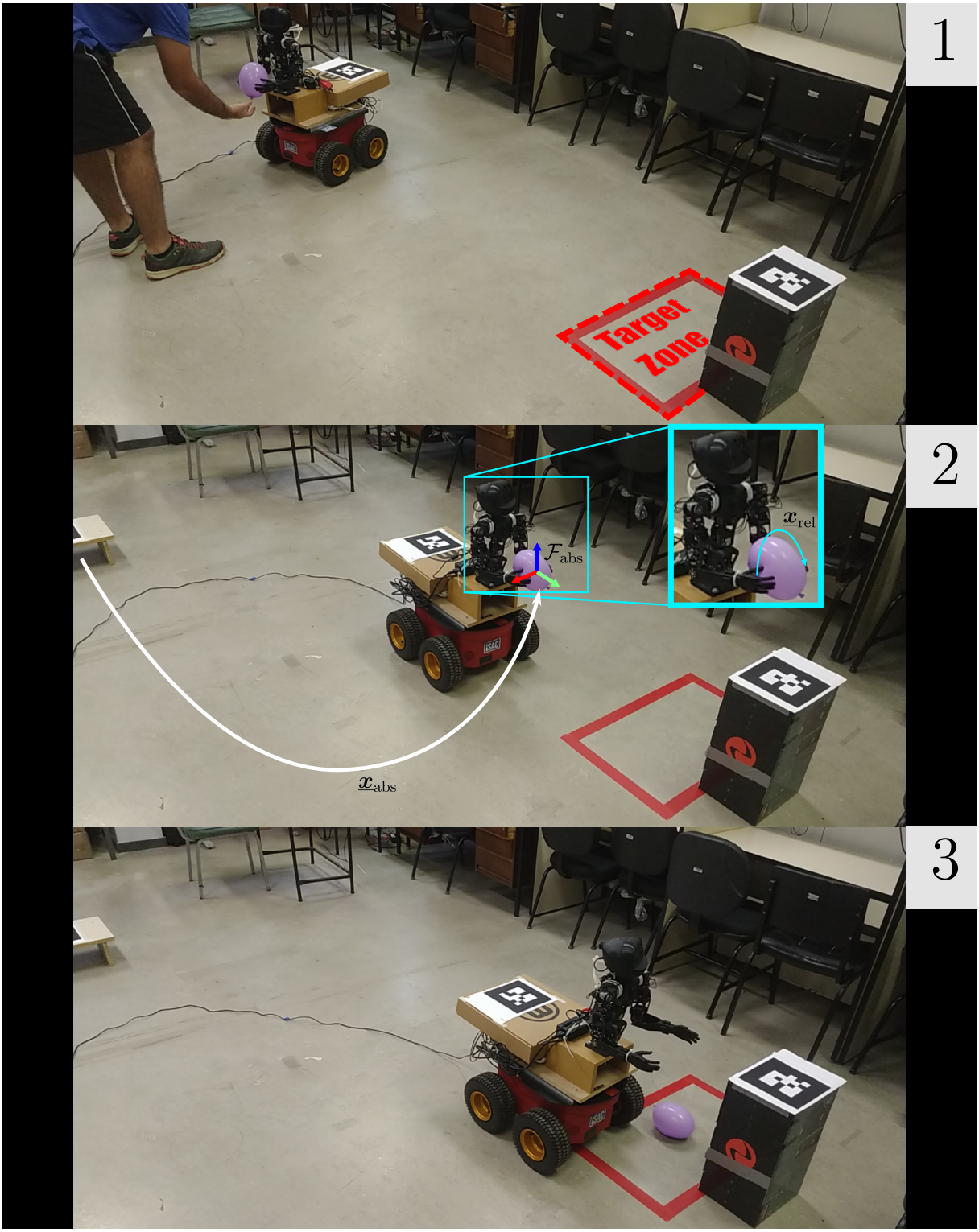

Figure 6.11: Control of both hands using the CDTS framework using a nonholonomic mobile bimanual manipulator (snapshots of the real-time experiment). The goal is to minimize the distance between the frame $\mathcal{F}_{\text{abs}}$ to the target zone. In the third snapshot, the robot releases the ball. Because of that, the relative pose changes.

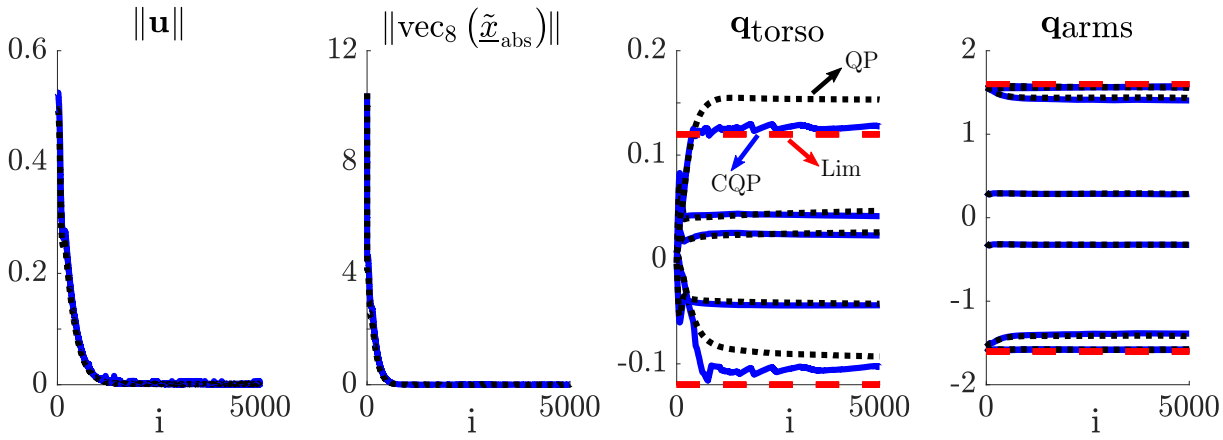

Figure 6.12: Control of both hands using the CDTS framework using a nonholonomic mobile bimanual manipulator. From left to right: norm of the control inputs (joint velocities); norm of the task error and the joints configuration of the torso and the left arm. The *dashed red line* on the third and fourth graph denotes the joint limits. The horizontal axis are iterations.

6.2 Multi-Contact Control: Simulation Setup

In order to validate the proposed multi-contact strategy, we performed simulations on V-REP[7] and Python. Furthermore, we used the 27-DOF Thormang[8] humanoid robot, which is composed of two 7-DOF arms, two 6-DOF legs, and a 1-DOF torso.

The goal is to control the right-hand of a humanoid robot to touch a target ball preventing self-collisions (constraint (4.37)) and ensuring the robot balance (constraint (4.62)), as shown in Fig. 6.13. In this specific case, we set three steps to accomplish the task, labeled as step I, II, and III. In the first step, the robot performs a contact with the environment (the blue table) using the left hand. In the second one, the SP area is maximized. Finally, in the third step, the robot touches the ball. All three steps are required in this specific example, as shown in Fig. 6.14. Otherwise, the robot fails to accomplish the task.

6.2.1 Step I. Perform Contact

To accomplish this task, we use two planes to define the task error, as described in section 4.5.1. One plane $\underline{\boldsymbol{\pi}}$ is attached to the left-hand of the robot, and the other plane, denoted by $\underline{\boldsymbol{\pi}}_d$, is used to model the table. The control law (3.1) minimizes the task error $\tilde{\underline{\boldsymbol{\pi}}} \triangleq \underline{\boldsymbol{\pi}} - \underline{\boldsymbol{\pi}}_d = \boldsymbol{n}_\pi - \boldsymbol{n}_{\pi_d} + \varepsilon\,(d_\pi - d_{\pi_d})$. Furthermore, the kinematic chain is composed of the 12-DOF legs, the 1-DOF torso and the 7-DOF left arm. To show the task error behavior, we define the distance $d_{\tilde{\underline{\boldsymbol{\pi}}}} \triangleq d_{n_\pi, n_{\pi_d}} + \tilde{d}_\pi$, which is composed of two distances, $d_{n_\pi, n_{\pi_d}} \triangleq \|\boldsymbol{n}_\pi - \boldsymbol{n}_{\pi_d}\|$ and $\tilde{d}_\pi \triangleq |d_\pi - d_{\pi_d}|$. The former quantifies the error of

[7] http://www.coppeliarobotics.com/
[8] https://www.robotis.us/thormang/

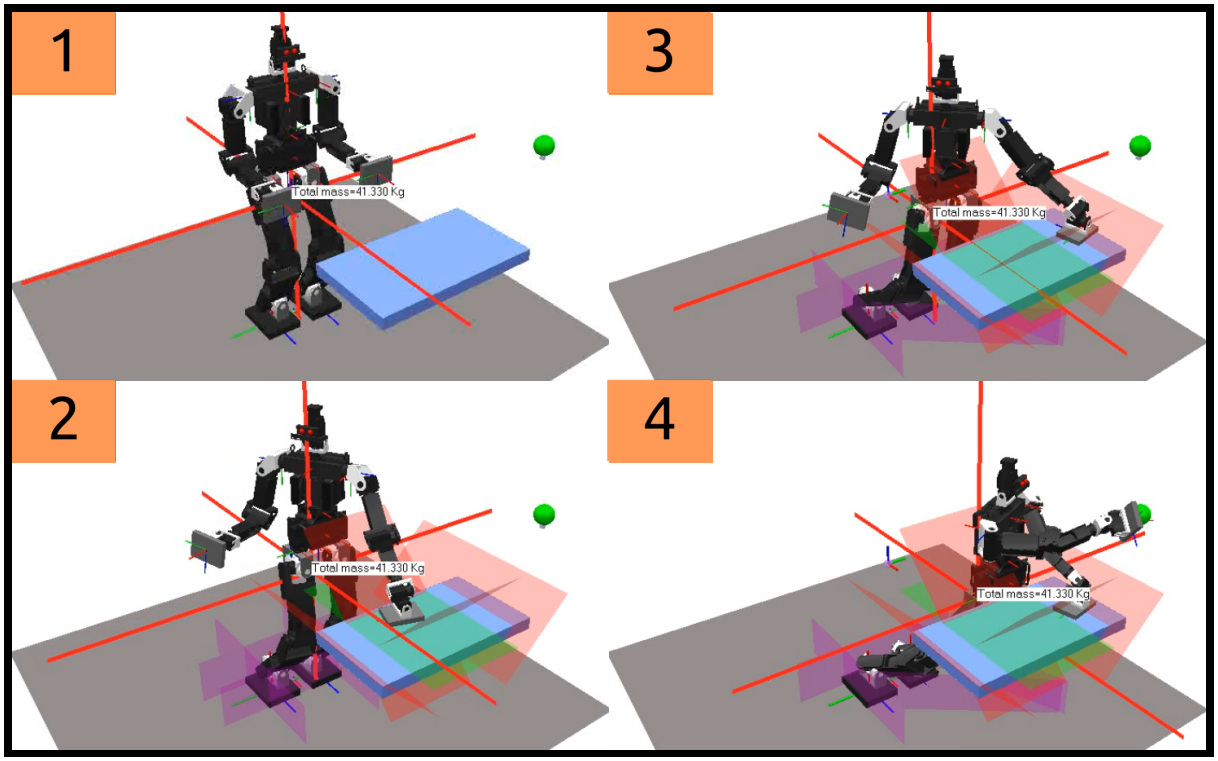

Figure 6.13: Example 1 of a multi-contact task using the robot kinematics and first-order VFIs (snapshots of the simulation). 1) Initial configuration. The goal is to control the right hand of the robot to touch the green ball in three steps. 2) Step I: the robot performs a contact with the left hand on a target region. 3) Step II: the SP area is maximized. 4) Step III: the robot touches the ball keeping the contact with the table fixed.

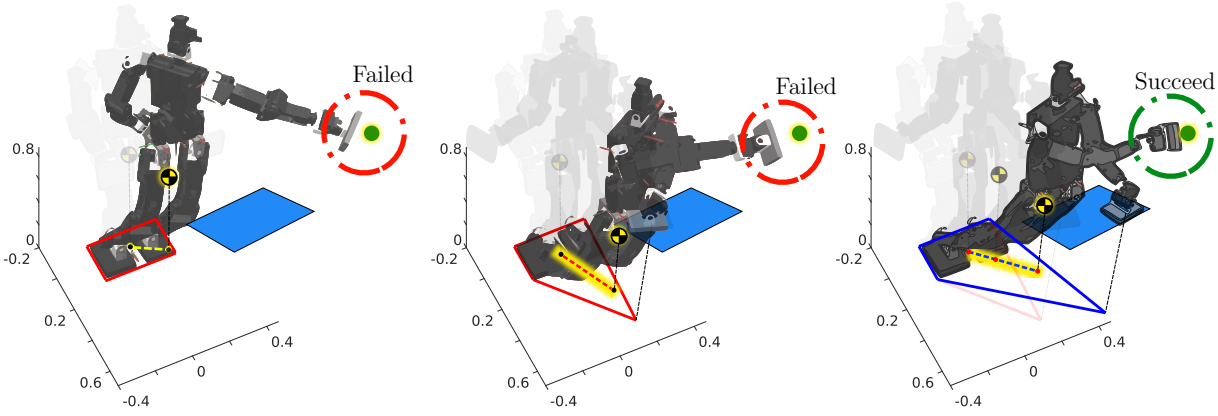

Figure 6.14: Example 1 of a multi-contact task using the robot kinematics and first-order VFIs: final configurations (snapshots of the simulation). On the *left,* the robot fails to accomplish the task (touching the ball). On the *middle,* the robot tries to touch the ball, by performing first a contact with the table, however, the robot fails again. On the *right,* the robot maximizes the area after performing a contact with the table. This allows fulfilling the task.

the orientation between the left hand and the table. The latter quantifies the distance between them, as shown in Fig. 4.14. In this way, when $\underline{\tilde{\boldsymbol{\pi}}} \to \mathbf{0}$ then both the distance between the plane orientations $d_{n_\pi, n_{\pi_d}}$ and the distance between the planes $\tilde{d}_\pi$ go to zero. We use (3.1) with $\boldsymbol{J} \triangleq \boldsymbol{J}_\pi$, $\tilde{\boldsymbol{x}} \triangleq D_{rh,b}$, $\eta = 1$, and $\lambda = 0.08$. The target region is defined using the plane constraints (4.39). We impose the constraints (4.57) and (4.37) to maintain both feet stationary on the ground, and the CoM plane constraints (4.62) to maintain the robot COM projection $p_{\mathrm{com}_{x,y}}$ inside the support polygon. These constraints, in addition to the self-collision avoidance constraint (4.37), are used in all three steps. Finally, constraint (4.39) prevents collisions between the hand and the table.$d_{\underline{\tilde{\pi}}} = d_{n_\pi, n_{\pi_d}} + \tilde{d}_\pi$

6.2.2 Step II. Maximize SP Area

Once the robot has performed the contact with the table, the goal is to maximize the support polygon area (Step II) using a control law as in (3.1) where $\boldsymbol{J}$ correspond to the SP area Jacobian, which is given by (4.54). Since the task error dynamics is governed by the equation $\dot{A}(t) = \eta A(t)$, whose solution is given by $A(t) = A(0)\exp(-\eta t)$, a negative value for the convergence factor (i.e., $\eta < 0$) is used to impose an exponential increase of the SP area. Nonetheless, its growth is limited by the plane constraints (4.39) and the CoM plane constraints (4.62), which define the target region and keep the robot CoM inside the support polygon, respectively. Furthermore, the SP area is maximized performing movements of the left hand only. To keep the left hand constrained to movements over the contact surface while the area is maximized, we impose the plane-to-plane constraint (4.61).

6.2.3 Step III. Try Task

In the last step, the control law minimizes the square distance between the right-hand and the ball, which is represented by $D_{rh,b} \triangleq d_{rh,b}^2$. Both the right hand and the ball are modeled using the point-to-point primitive with an associated radius $d_{rh,b_{\mathrm{safe}}} = 0.04$ (see Table. 3.1). The same constraints of Step II are used, except the four planes constraint to define the target region ((4.39)), which is not required anymore.[9] In addition, the constraint (4.61) is replaced by (4.60) to keep fixed the contact between the left hand and the table. Table. 6.2 summarizes the steps with their respective objective functions and constraints.

Fig 6.15 shows that the distance $d_{\underline{\tilde{\pi}}}$ and the square distance $D_{rh,b}$ are minimized, as expected. Furthermore, the exponential growth of the SP area is limited by the additional constraints, which maintain the left hand on the table and the robot CoM inside the support polygon.

[9]The target region is used to limit the region of the left hand. In this step, the contact of the left hand with the table must be maintained fixed.

Table 6.2: Objective function definition and constraints used for steps I, II, and III.

Control Law (3.1)		**I. Perform Contact**	**II. Maximize SP Area**	**III. Try Task**
Jacobian Task	$\boldsymbol{J}$	$\boldsymbol{J}_{\pi}$ (4.56)	$\boldsymbol{J}_{A}$ (4.54)	$\boldsymbol{J}_{rh,b}$ (3.9)
Task error	$\tilde{\boldsymbol{x}}$	$\text{vec}_8\,\underline{\tilde{\boldsymbol{\pi}}}$ (4.55)	A (4.41)	$D_{rh,b}$ (3.8)
Convergence parameter	η	2	-0.5	1
Damping factor	λ	0.05	0.1	0.08
Constraints				
Target region		(4.39)	(4.39)	
Collision avoidance		(4.57), (4.37)	(4.37)	(4.37)
Stationary feet		(4.58), (4.59)	(4.58), (4.59)	(4.58), (4.59)
Center of mass		(4.62)	(4.62)	(4.62)
Sliding contact			(4.61)	
Fixed contact				(4.60)

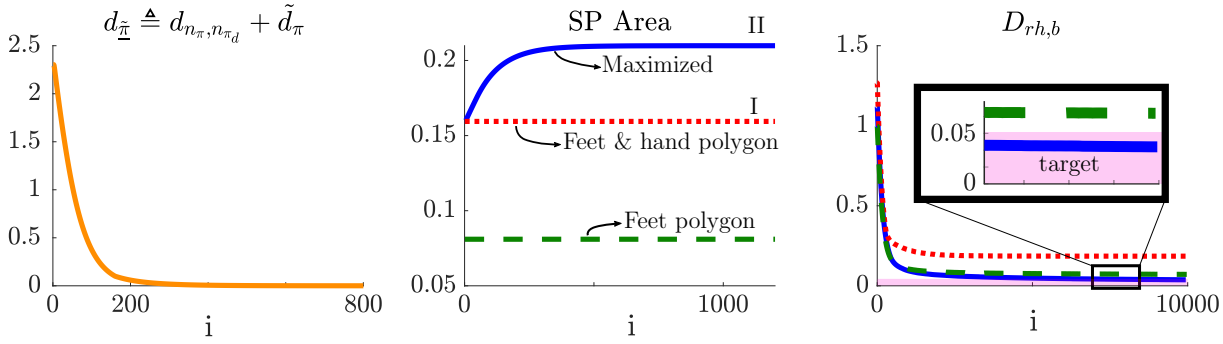

Figure 6.15: Example 1 of a multi-contact task using the robot kinematics and first-order VFIs. On the *left,* the distance $d_{\underline{\tilde{\pi}}}$ between the robot's left hand and the table. On the *middle,* the support polygon area. The green dashed line denotes the SP area in the initial configuration. The red dashed line correspond to the SP area after the hand contact, in Step I. Finally, the blue line is the SP area in Step II. On the *right,* the distance between the right hand and the green ball. The robot touches the ball only when it performs a contact with the table and maximizes the area (blue line). When the robot tries to touch the ball without performing the hand contact (green dashed line) or performing the hand contact but with no SP area maximization, the robot fails to accomplish the task. The horizontal axis are iterations.

Fig. 6.16 shows that in all three steps, the CQP controller respects the robot CoM constraints and minimizes the norm of the control inputs, as expected. Furthermore, the support polygon, provided by the maximization of the SP area, allowed the robot to put the $p_{\text{com}_{x,y}}$ in a suitable position such that the ball can be reached by the right hand, as shown in Fig. 6.14 and Fig 6.15. Notice that when the robot tries to touch the ball without performing the hand contact (green dashed line), the distance $D_{rh,b}$ is lower than the one when the robot performs the hand contact (red dashed line). In other words, in this specific example, the robot configuration is worst, in terms of the robot reachability, when performing a hand contact when compared to the robot configuration without the hand contact. This is not surprising, since the proposed strategy to maximize the SP area does not ensure better robot reachability when the SP is larger.

Fig. 6.13 shows the simulation snapshots. All intermediary tasks are performed successfully, and the final task is fulfilled.

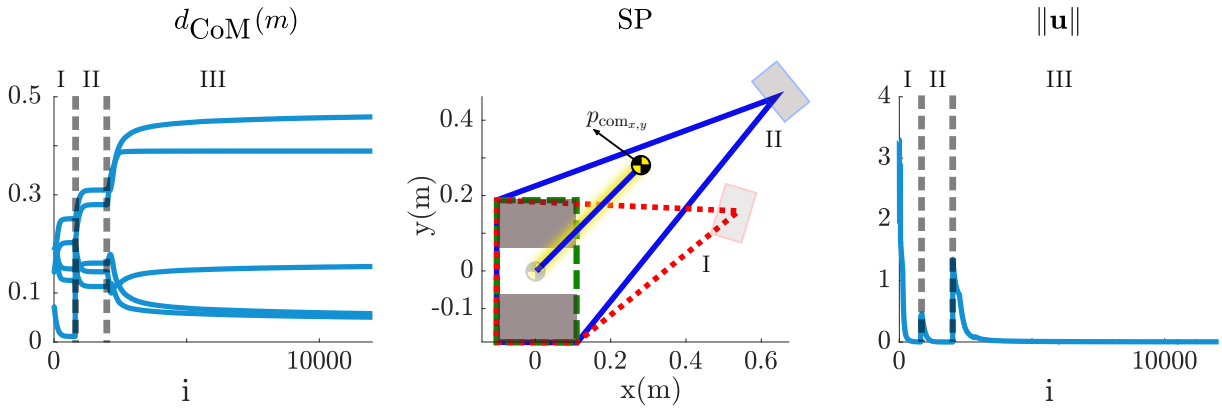

Figure 6.16: On the *left,* the distance between the center of mass (CoM) projection and the CoM plane constraints for steps I, II and III. On the *middle,* the support polygon region. On the *right,* norm of the control inputs (joints velocities) for for steps I, II and III. The horizontal axis are iterations.

We performed a second simulation using the same task specification aforementioned for the multi-contact strategy. In this case, the robot performs the hand contact with a wall.

Fig. 6.17 shows the simulation snapshots. In this case, the robot accomplishes the task by performing the wall contact without the necessity of maximizing the SP area. Even, the maximization of the SP area led to a worse configuration in terms of the robot reachability, as shown in Fig. 6.18. However, this is addressed by relaxing the wall contact, which allows to reach the ball and potentially get a higher SP area, when compared to the SP area obtained by performing the contact without the area maximization.

Often, the exact sequence of states needed to accomplish a given task is unknown. In those cases, we can use the proposed strategy described in Section. 4.5. Table 6.3 summarizes the states and Fig. 6.19 shows the state transitions for a wall contact example.

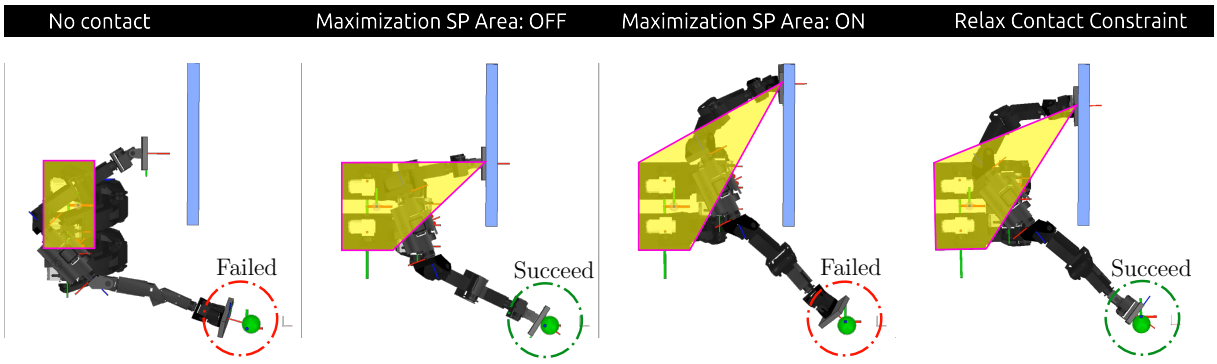

Figure 6.17: Final configuration of an example of a multi-contact task using the robot kinematics and first-order VFIs. From left to right: The robot tries to touch the ball and fails; The robot performs a contact with the wall and succeeds; The robot maximizes the SP area but fails; The robot relaxes the wall contact and accomplishes the task.

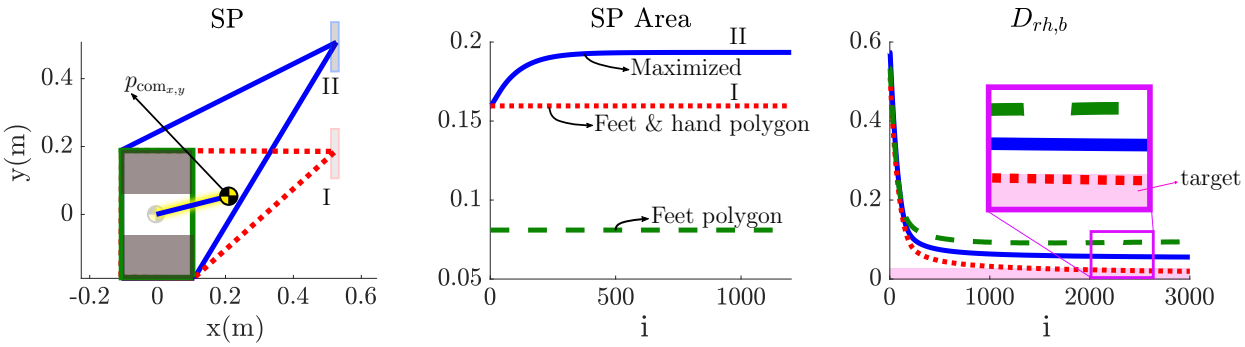

Figure 6.18: Example 2 of a multi-contact task using the robot kinematics and first-order VFIs. On the *left,* the support polygon. On the *middle,* the support polygon area. On the *right,* distance between the right hand and the green ball. In this case, the robot touches the ball only when performs a contact with the wall without the SP area maximization (red dashed line). When the robot tries to touch the ball without performing the hand contact (green dashed line) or performing the hand contact with SP area maximization (blue line), the robot fails.

Table 6.3: State description for the proposed multi-contact strategy to touch a ball.

	State	Description
1	**Try Task (TT).**	The robot tries to touch the target ball.
2	**Perform Contact & TT.**	The robot performs a contact with the environment and after, tries (TT).
3	**Maximize SP Area & TT.**	The robot maximizes the support polygon and after, tries (TT).
4	**Relax Contact Constraint & TT.**	The hand-contact constraint is relaxed allowing a sliding contact on the surface and simultaneously tries (TT).
5	**Task Done.**	The task is fulfilled.
6	**Task Failed.**	The robot fails to accomplish the task.

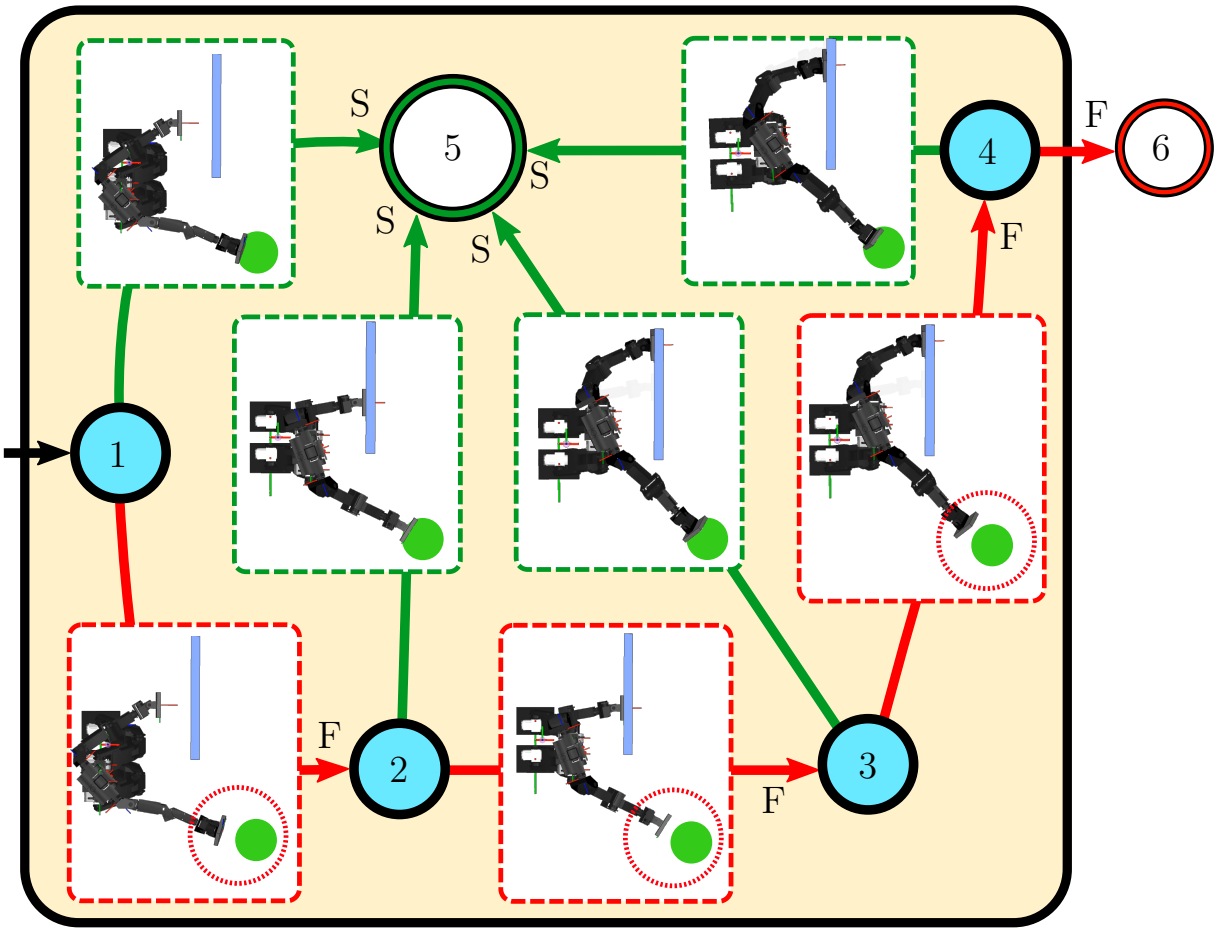

Figure 6.19: Proposed strategy for multi-contact tasks to touch a ball. The strategy is based on the state transition described in Fig. (4.13). The green arrows represent success (S) events. The red arrows are fails (F) events. The robot executes a maximum of four attempts to accomplish the main task.

6.3 Numerical Tests Using the GLPC

To evaluate the Euler-Lagrange model obtained using the dual quaternion Gauss's Principle of Least Constraint (DQGP), this section presents simulations using robot manipulators and nonholonomic mobile manipulators.

6.3.1 7-DOF Robot Manipulator

First, we perform simulations of a 7-DOF robot manipulator, which are implemented on Matlab 2018a using the computational library DQ Robotics (Adorno & Marques Marinho, 2020). The implementation is performed using a computer equipped with an Intel i7 4712HQ with 16GB RAM running Ubuntu 18.04 64 bits. In addition, we computed the average computational time required to obtain the joint torques of a 7-DOF robot manipulator in Python.

Since the Robotics Toolbox (Corke, 2017), which implements a classic Newton-Euler algorithm (RTNE), is widely used and its accuracy has been verified throughout the years, the goal is to compare the torques it generates to the torques generated by Algorithm 5.1.

First, 10000 random joint configurations, velocities, and accelerations are generated and then obtained the corresponding torque vectors acting in each joint of the manipulator.

The percentage relative error of each joint is

$$\tau_{i_{\text{error}}} = 100 \cdot \frac{|\tau_{i_{\text{method}}} - \tau_{i_{\text{baseline}}}|}{\tau_{i_{\text{baseline}}}}, \tag{6.5}$$

where $\tau_{i_{\text{baseline}}}$ is the torque of the ith joint generated by RTNE. Moreover, $\tau_{i_{\text{method}}}$ is the torque of the ith joint generated by Algorithm 5.1 that is being compared to the baseline.

Finally, the mean percentage error of each joint is computed as follows

$$\bar{\tau}_{i_{\text{error}}} = \frac{1}{n} \sum_{j=1}^{n} \tau_{j,i_{\text{error}}},$$

where $n = 10000$, and $\tau_{j,i_{\text{error}}}$ is the percentage relative error of the joint ith at trial j. A trial correspond to an execution of the Algorithm 5.1. Furthermore, it is computed the corresponding standard deviation $\sigma_{i_{\text{error}}}$ for both simulations.

Fig. 6.20 shows that, when compared to RTNE, Algorithm 5.1, which is based on the dual quaternion Gauss's Principle of Least Constraint, is as accurate as Robotics Toolbox. The mean percentage error and the standard deviation are very small. This demonstrates the similarity between both methods, in terms of accuracy.

Fig. 6.21 shows the average computational time of 10000 trials and its respective standard deviation (s.d.). Both strategies, RTNE[10] and DQGP, are implemented on

[10]https://github.com/petercorke/robotics-toolbox-python

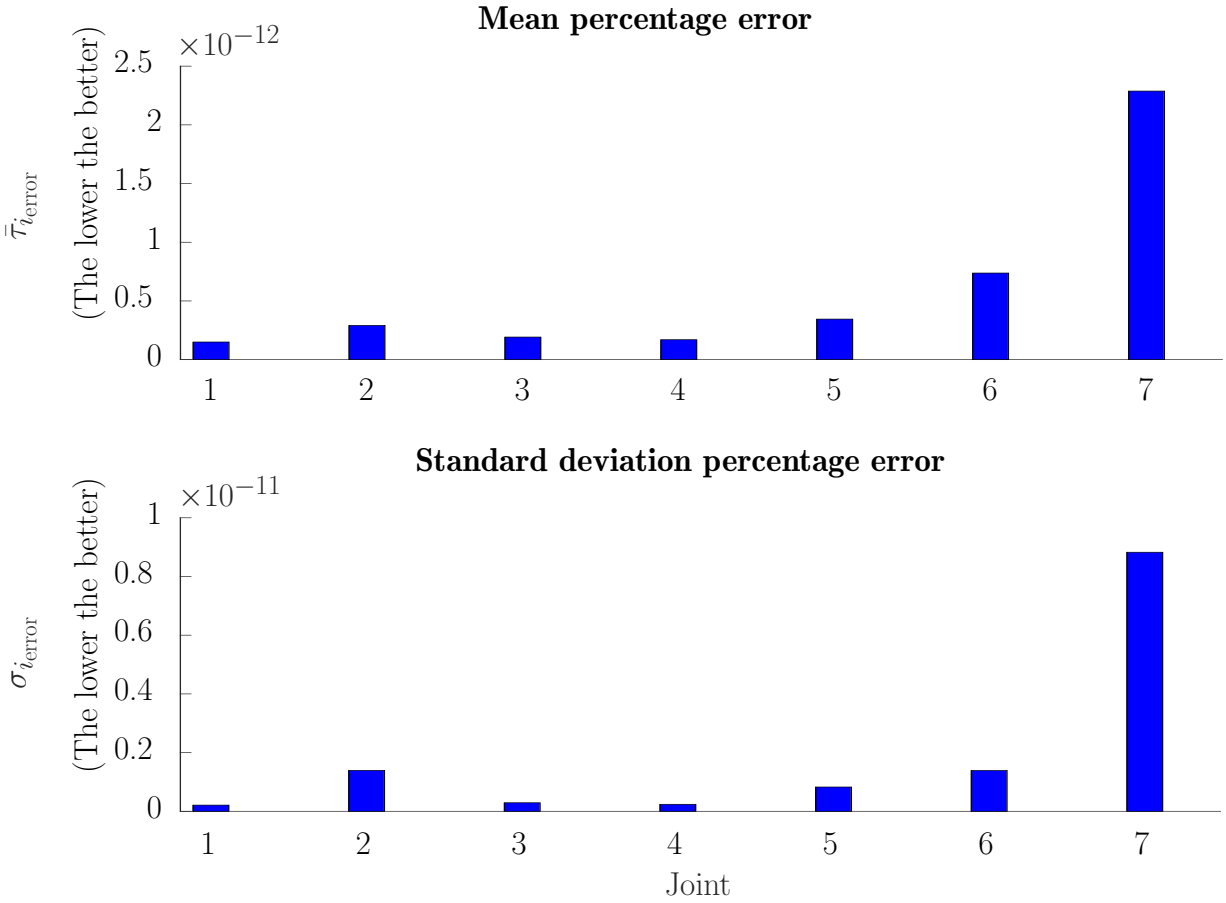

Figure 6.20: Mean percentage error $\bar{\tau}_{i_{\text{error}}}$, and corresponding standard deviation $\sigma_{i_{\text{error}}}$ of the comparison between RTNE, and the algorithm 5.1.

Python.

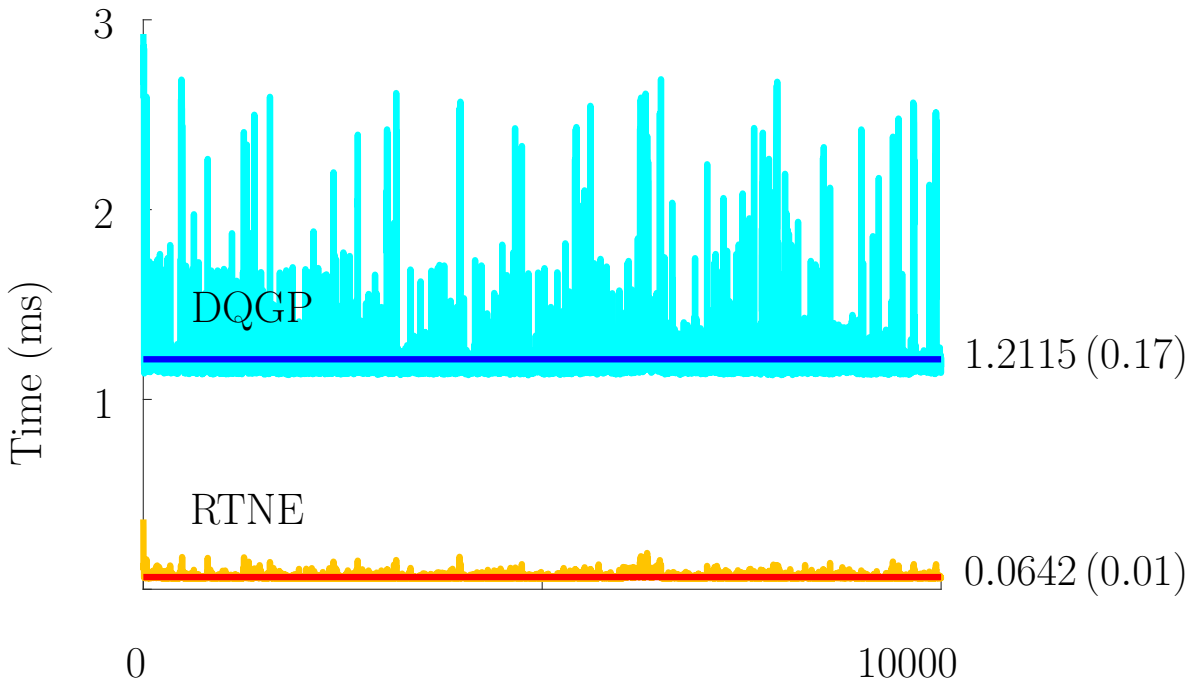

Figure 6.21: Mean (s. d.) computational time in miliseconds.

6.3.2 V-REP Simulations

The goal is to evaluate the Euler-Lagrange model based on DQGP using mobile manipulators, namely a 9-DOF holonomic mobile manipulator, and an 8-DOF nonholonomic mobile manipulator. The idea is to illustrate that the proposed strategy works with several types of robotics systems. Since the Robotics Toolbox only supports dynamic modeling of fixed-base robots, this section presents simulations[11] implemented on the robot simulator V-REP PRO EDU V3.6.2[12] using an interface with Matlab 2020a and the computational library DQ Robotics (Adorno & Marques Marinho, 2020). In addition, we evaluate the Euler-Lagrange model based on DQGP using a fixed-base 50-DOF serial manipulator, to validate the Algorithm (5.1) in a highly redundant robot.

In Section 6.3.1, we generated random joint configurations, velocities, and accelerations and, we use them to compare the joint torques between two models: RTNE and DQGP. In order to apply the same strategy using V-REP, we require to set joint configurations $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and accelerations $\ddot{\boldsymbol{q}}$ and the read the torques $\boldsymbol{\tau}_{\mathrm{s}}$. Then we can compare the read torques from V-REP $\boldsymbol{\tau}_{\mathrm{s}}$ with the torques computed using DQGP (5.20). However,

[11]The work described in this subsection was developed in collaboration with Frederico Fernandes Afonso Silva, a PhD Candidate at UFMG (Silva et al., 2020).

[12]Available at: https://www.coppeliarobotics.com/

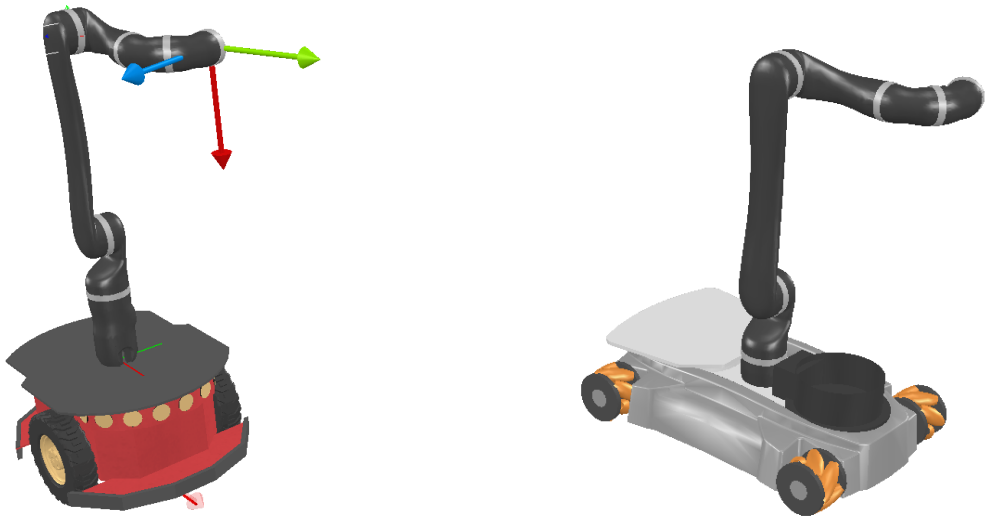

Figure 6.22: Snapshots of the V-REP simulation. On the *left*, a nonholonomic mobile manipulator, which is composed of a differential-drive robot, and a 6-DOF arm manipulator. On the *right*, the mobile manipulator is composed of a holonomic-drive robot and the same 6-DOF arm manipulator.

V-REP does not support commands to set joint accelerations. Because of that, we do not compare the torques.

Instead, we compare the generalized accelerations $\ddot{\boldsymbol{q}}_m$ obtained through the DQGP and the values $\ddot{\boldsymbol{q}}$ from V-REP. To perform the comparison, we use the *coefficient of multiple correlation (CMC)* (Ferrari et al., 2010) between the waveforms. The CMC provides a coefficient ranging between zero and one that indicates how similar two given waveforms are. Identical waveforms have CMC equal to one, whereas completely different waveforms have CMC equal to zero.

First, we generate a trajectory of torques $\boldsymbol{\tau}_{\mathrm{GP}}$ to be applied to both the V-REP simulator and the DQGP model, as shown in Fig. 6.23. To take into account the same initial conditions in the DQGP model, we read from V-REP, before applying the torques $\boldsymbol{\tau}_{\mathrm{GP}}$, the generalized configurations and generalized velocities, which are denoted as $\boldsymbol{q}_0$ and $\dot{\boldsymbol{q}}_0$, respectively.

In the cases of the fixed-base 50-DOF serial manipulator, and the 9-DOF holonomic mobile manipulator, we compute the generalized accelerations $\ddot{\boldsymbol{q}}$ using (5.20), which yields

$$\ddot{\boldsymbol{q}}_m = \left(\boldsymbol{M}_{\mathrm{GP}}\left(\boldsymbol{q}_0\right)\right)^{-1}\left(\boldsymbol{\tau}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}}\left(\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0\right)\dot{\boldsymbol{q}}_0 - \boldsymbol{g}_{\mathrm{GP}}\left(\boldsymbol{q}_0\right)\right). \tag{6.6}$$

Likewise, for the 8-DOF nonholonomic mobile manipulator, we use (5.47) to compute the generalized accelerations, which yields

$$\ddot{\boldsymbol{q}}_m = \left(\boldsymbol{M}_{\mathrm{GP}}\left(\boldsymbol{q}_0\right)\right)^{-1}\left(\boldsymbol{\Omega}\left(\boldsymbol{q}_0\right)\left(\bar{\boldsymbol{\tau}}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}}\left(\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0\right)\dot{\boldsymbol{q}}_0\right) + \boldsymbol{M}_{\mathrm{GP}}^{1/2}\left(\boldsymbol{q}_0\right)\boldsymbol{D}^{+}\left(\boldsymbol{q}_0\right)\boldsymbol{b}\right), \tag{6.7}$$

where $\bar{\boldsymbol{\tau}}_{\mathrm{GP}}$ are the the generalized forces. As discussed in Section 5.2.3, we consider the generalized forces $\boldsymbol{\tau}_{\mathrm{GP}}$ applied in the joints and the gravitational forces $\boldsymbol{\tau}_g$, such that the resultant forces acting on the system are $\bar{\boldsymbol{\tau}}_{\mathrm{GP}} = \boldsymbol{\tau}_{\mathrm{GP}} + \boldsymbol{\tau}_g$. By letting $\boldsymbol{g}_{\mathrm{GP}} \triangleq -\boldsymbol{\tau}_g$, we have that $\bar{\boldsymbol{\tau}}_{\mathrm{GP}} = \boldsymbol{\tau}_{\mathrm{GP}} - \boldsymbol{g}_{\mathrm{GP}}$. Furthermore, the nonholonomic constraint is written analogously to (5.48), with $\boldsymbol{A} \triangleq \begin{bmatrix} -\sin\phi & \cos\phi & 0 & \mathbf{0}_{1\times 6} \end{bmatrix}$ and $\boldsymbol{b} = -\dot{\boldsymbol{A}}\dot{\boldsymbol{q}}$.

The V-REP simulator does not allow the direct reading of accelerations. Therefore, to obtain the configuration acceleration vector $\ddot{\boldsymbol{q}}$, we first read the velocity vector $\dot{\boldsymbol{q}} \in \mathbb{R}^9$, then filtered all elements $\dot{q}_1, \ldots, \dot{q}_9$ with a discrete classic low-pass Butterworth filter, since it provides good performance and smooth responses. We used the filtered values to obtain the accelerations by means of numerical differentiation based on a second forward finite difference approximation, as shown in Fig. 6.23. We then calculated the CMC between each element $\ddot{q}_i$, with $i \in \{1, \ldots, 9\}$, of the generalized acceleration waveform and its counterpart from V-REP. Afterward, we used those CMCs to obtain the mean, minimum, and maximum CMCs for the model, alongside their standard deviation.

Furthermore, for the simulation of the fixed-base 50-DOF serial manipulator, we also used the classic Newton-Euler algorithm (RTNE) implemented on the Robotics Toolbox (Corke, 2017), and calculated the CMC between the joint acceleration waveforms yielded by it and the ones from V-REP.[13]

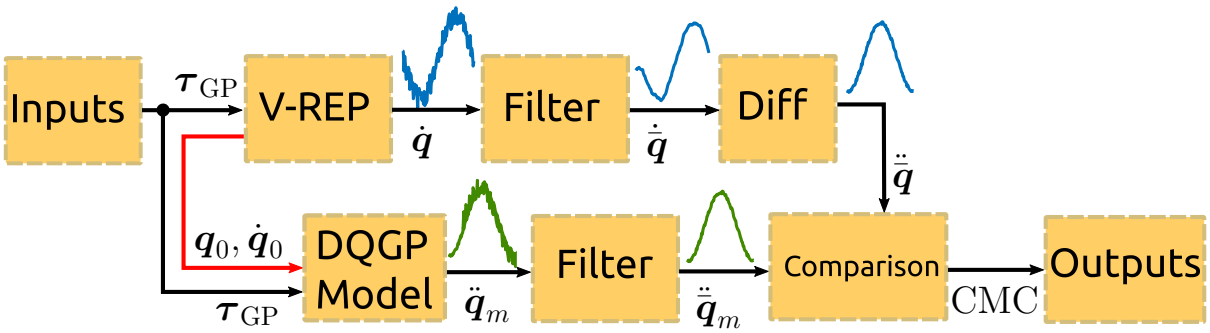

Figure 6.23: Strategy used to validate the models based on DQGP on V-REP. The inputs are the joints torques $\boldsymbol{\tau}$. The generalized accelerations $\ddot{\boldsymbol{q}}_m$ are computed using $\boldsymbol{\tau}$, $\boldsymbol{q}_0$, and $\dot{\boldsymbol{q}}_0$, where $\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0$ are the generalized configurations and generalized velocities, respectively, both read from V-REP before applying the torques $\boldsymbol{\tau}_{\mathrm{GP}}$. Next, $\ddot{\boldsymbol{q}}_m$ is filtered and denoted as $\ddot{\bar{\boldsymbol{q}}}_m$. On the other hand, the generalized velocities $\dot{\boldsymbol{q}}$ are read from V-REP after applying the torques $\boldsymbol{\tau}_{\mathrm{GP}}$. Next, $\dot{\boldsymbol{q}}$ is filtered using the same filter previously used, and denoted as $\dot{\bar{\boldsymbol{q}}}$. Likewise, $\ddot{\bar{\boldsymbol{q}}}$ is computed by means of numerical differentiation. Finally, both $\ddot{\bar{\boldsymbol{q}}}_m$ and $\ddot{\bar{\boldsymbol{q}}}$ are compared using the CMC.

Table 6.4 presents the CMC between the generalized acceleration waveforms obtained through two different dynamic model strategies (DQGP and RTNE) and the values obtained from V-REP.

The current version of the Robotics Toolbox only supports dynamic modeling of fixed-

[13]In this case, we used the Vortex Studio engine (www.cm-labs.com) because it presented better numerical stability for the 50-DOF manipulator.

base robots and was only applied to the 50-DOF serial manipulator. The cases where the model could not be obtained using the listed strategy are indicated in Table 6.4 by N/A. (i.e., not available). For all other cases, all models presented mean and minimum CMC close to one, with small standard deviation (s.d.) and high maximum (max) CMC; thus indicating high similarity between the generalized acceleration waveform obtained from them and the values from V-REP.

Table 6.4: CMC between the joint acceleration waveforms obtained through different dynamic model strategies and the values obtained from V-REP. The closer to one, the more similar the waveforms are.

	50-DOF serial manipulator				9-DOF holonomic mobile manipulator				8-DOF nonholonomic mobile manipulator			
Method	min	mean	std	max	min	mean	std	max	min	mean	std	max
DQGP	0.9044	0.9893	0.0182	0.9993	0.9934	0.9973	0.0026	0.9999	0.8860	0.9839	0.0368	0.9999
RTNE	0.9044	0.9893	0.0182	0.9993	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

For qualitative analysis, Fig. 6.24 presents the generalized accelerations obtained using DQGP, alongside the V-REP values, for the minimum, maximum, and intermediate CMCs found during simulations. Even for the smallest value of CMC (i.e., 0.8860), the accelerations obtained using our formulation match closely the V-REP values. The small discrepancies arise from both discretization effects and because the accelerations in V-REP are estimated from noisy velocity values.

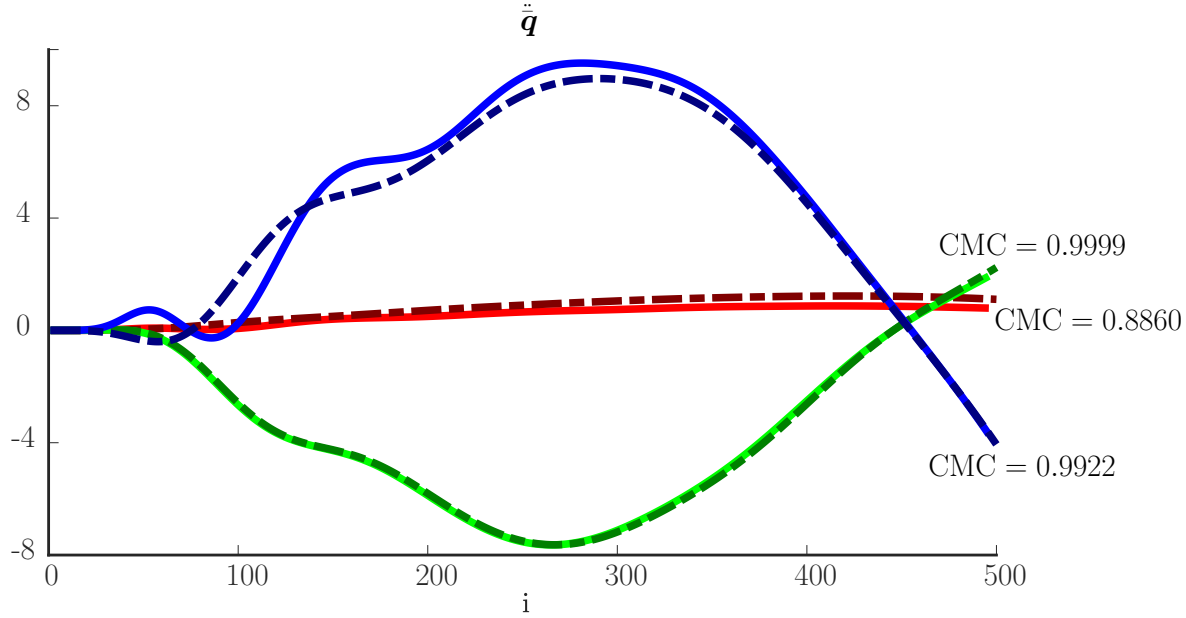

Figure 6.24: Generalized acceleration waveforms of the 8-DOF nonholonomic mobile manipulator. *Solid* curves correspond to the V-REP values $\ddot{\tilde{\boldsymbol{q}}}$, whereas *dot-dashed* curves correspond to the values $\ddot{\tilde{\boldsymbol{q}}}_m$ obtained using the DQGP for the generalized acceleration waveforms of the first (CMC = 0.8860), ninth (CMC = 0.9999), and fifth (CMC = 0.9922) coefficients of the configuration vector, respectively.

6.4 Conclusions

The chapter presented the results of the simulation and experiments performed. Section. 6.1 detailed the specification of the experimental environment based on the upper body of the Poppy humanoid robot and two control laws: QP and CQP. In simulation environments, QP does not take into account constraints, and CQP denotes the constrained case. Section. 6.1.1 showed the simulation results using first-order kinematics and the line-to-line angle Jacobian, which was presented in Section 4.2. In this case, the goal is to put a cup on a table, using the left-hand of a humanoid robot, while keeping the cup tilting below a threshold, avoiding reaching joint limits, and preventing self-collisions and collisions with the workspace. The results showed that the task error converges to zero for both controllers, however, only CQP prevents undesired cup orientations and respects all other constraints. Section. 6.1.2 showed a second simulation using the robot dynamic model with second order vector fields inequalities (SOVFIs). Although CQP respects all constraints and minimizes both the joint accelerations and the task error, it generates more abrupt control inputs because the collision-avoidance constraints enforce abrupt changes in the robot velocities to prevent collisions. Because of that, CQP required a higher control effort. Section 6.2 presented an example of the multi-contact strategy proposed in Section 4.5. The goal is to control the right-hand of a humanoid robot to touch a target ball, preventing self-collisions and ensuring the robot balance. To accomplish the task, the robot performs a contact with the environment and maximizes the SP area. The results showed that CQP controller respects the robot CoM constraints and minimizes the norm of the control inputs and the task error, as expected. Although in the first simulation performed in section (6.2), the maximization of the support polygon increased the robot reachability, this behavior is not ensured. In other words, the proposed SP maximization does not guarantee the improvement of the robot reachability and therefore, the robot could fail to accomplish a specific task. This case is showed with a second simulation. To address these cases, the chapter presented a strategy based on a finite-state machine. When the robot fails to accomplish a task performing first an SP maximization, the hand contact constraint can be relaxed to allow a sliding contact. Section. 6.1.3 presented the experimental results and showed the applications of the line-to-line angle Jacobian on three different platforms, including serial manipulators (torso and left hand), bimanual manipulators (both arms using the cooperative manipulation) and nonholonomic bimanual manipulators (differential base and both arms using the cooperative manipulation).

7

Conclusion and Future Works

This work extended the VFIs framework based on dual quaternion algebra, which was initially proposed using first order kinematics by Marinho et al. (2018), to second order kinematics. This allows its use in applications that require the robot dynamics by means of the relationship between the joint torques and joint accelerations in the Euler-Lagrange equations. Furthermore, we proposed a novel singularity-free conic constraint to limit the angle between two Plücker lines. This new constraint is used to prevent the violation of joint limits and/or to avoid undesired end-effector orientations. In addition, we proposed a new Jacobian related with the support polygon area of a humanoid robot. This Jacobian is used to increase the area of the support polygon and, potentially to improve the workspace reachability or the robot balance in the presence of external disturbances. This is done by doing an approximation of the support polygon (SPA) function of some contact points. Although this approximation is conservative, it simplifies the computation of the Jacobian. The VFIs framework requires an environment modeled with sufficient geometric primitives and is not free of local minima, but it does provide reactive behaviors.

We evaluated the proposed method using kinematic and dynamic formulations, both in simulation and on a real humanoid robot. The results showed that the robot always avoids collisions with static obstacles, self-collisions, and undesired orientations while performing manipulation tasks. However, the generation of the control inputs under SOVFIs, which is based on quadratic programming, assumes feasibility. Future works will be focused on strategies to ensure feasibility in the optimization formulation.

The strategy for multi-contacts applications, however, was explored only for first order

kinematics. This enables the use in robots commanded by joints velocities at expense of neglecting the forces and moments acting on the robot. Future works will be focused on multi-contacts applications based on SOVFIs. This allows the use of the robot dynamics to take into account the contact forces and friction forces to compute the support polygon area (Samadi et al., 2020), to account for center of mass accelerations (Audren & Kheddar, 2018), and handling non-coplanar contacts (Samadi et al., 2021). In addition, the second order kinematics allows the use the of zero moment point (ZMP) concept (VUKOBRATOVIĆ & BOROVAC, 2004), which is useful to achieve dynamic stability (Caron et al., 2017).

Last, this thesis presented the Gauss's Principle of Least Constraint (GPLC) for articulated bodies using dual quaternion algebra. This formulation leads to the Euler-Lagrange dynamic equation of a robotic system. The use of dual quaternion algebra allows a compact representation of the wrenches and accelerations. Furthermore, we presented the connections with other classic treatises of mechanics as Gibbs-Appell equations, and Kane's method. All three approaches deal with nonholonomic constraints without the necessity of Lagrange multipliers. However, the Gibbs-Appell equations and Kane's method require setting up quasi-velocities, whereas the Gauss's Principle does not, and allows taking into account additional constraints directly in the optimization formulation. We proposed an algorithm to compute the Euler-Lagrange model using the dual quaternion GPLC Formalism for a robot manipulator. Furthermore, we presented its computation cost. The proposed algorithm is, as expected, more expensive than the ones based on the Newton-Euler and classic Euler-Lagrange formalism since it is not based on recursive strategies. However, this strategy allows taking into account additional constraints in the accelerations, which can be exploited, for instance, in nonholonomic robotic systems or humanoid robots. Future works will be focused on exploiting inequality constraints in the optimization formulation and.

Although the representation for robot modeling and task description is based on dual quaternion algebra, both the control law and the modeling based on the Gauss's Principle of Least Constraint require the matrix form to perform the optimization over the fields of real vectors. This opens some theoretical questions about how to perform an optimization over the ring of dual quaternions. Future works will be focused on the robot dynamic modeling for a variety of different robots using the GPLC and its formulation in dual quaternion algebra using the hypercomplex form. The use of optimization formulations over non-Euclidean manifolds usually allows an elegant mathematical description of the tasks and avoid the necessity of additional variables and constraints (Audren & Kheddar, 2018).

Bibliography

Adorno, B. V. (2011). *Two-arm Manipulation: From Manipulators to Enhanced Human-Robot Collaboration [Contribution à la manipulation à deux bras : des manipulateurs à la collaboration homme-robot].* Phd thesis, Université Montpellier 2.

Adorno, B. V. (2017). *Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra — Part I: Fundamentals.*

Adorno, B. V. & Fraisse, P. (2017). The cross-motion invariant group and its application to kinematics. *IMA Journal of Mathematical Control and Information*, (pp. dnw032).

Adorno, B. V., Fraisse, P., & Druon, S. (2010). Dual position control strategies using the cooperative dual task-space framework. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3955–3960). Taipei: IEEE.

Adorno, B. V. & Marques Marinho, M. (2020). DQ Robotics: A Library for Robot Modeling and Control. *IEEE Robotics & Automation Magazine*, (pp. 0–0).

Al Borno, M., de Lasa, M., & Hertzmann, A. (2013). Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE Transactions on Visualization and Computer Graphics*, 19(8), 1405–1414.

Audren, H. & Kheddar, A. (2018). 3-D Robust Stability Polyhedron in Multicontact. *IEEE Transactions on Robotics*, 34(2), 388–403.

Balafoutis, C. A. (1994). A survey of efficient computational methods for manipulator inverse dynamics. *Journal of Intelligent & Robotic Systems*, 9(1-2), 45–71.

Baudouin, L., Perrin, N., Moulard, T., Lamiraux, F., Stasse, O., & Yoshida, E. (2011). Real-time replanning using 3D environment for humanoid robot. In *2011 11th IEEE-RAS International Conference on Humanoid Robots* (pp. 584–589).: IEEE.

Beardon, A. F. (1996). Sums of Powers of Integers. *The American Mathematical Monthly*, 103(3), 201–213.

Bouyarmane, K., Caron, S., Escande, A., & Kheddar, A. (2017). Multi-contact Motion Planning and Control. In *Humanoid Robotics: A Reference* (pp. 1–42). Dordrecht: Springer Netherlands.

Bouyarmane, K., Chappellet, K., Vaillant, J., & Kheddar, A. (2019). Quadratic Programming for Multirobot and Task-Space Force Control. *IEEE Transactions on Robotics*, 35(1), 64–77.

Bouyarmane, K., Escande, A., Lamiraux, F., & Kheddar, A. (2009). Potential field guide for humanoid multicontacts acyclic motion planning. In *2009 IEEE International Conference on Robotics and Automation* (pp. 1165–1170).: IEEE.

Bouyarmane, K. & Kheddar, A. (2012). On the dynamics modeling of free-floating-base articulated mechanisms and applications to humanoid whole-body dynamics and control. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)* (pp. 36–42).: IEEE.

Boyce, W. E. & DiPrima, R. C. (2008). *Elementary Differential Equations and boundary value problems*. 9th edition edition.

Bretl, T. & Lall, S. (2008). Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4), 794–807.

Bretl, T., Latombe, J.-C., & Rock, S. (2005). Toward Autonomous Free-Climbing Robots. In *Robotics Research. The Eleventh International Symposium. Springer Tracts in Advanced Robotics, vol 15. Springer, Berlin, Heidelberg* (pp. 6–15).

Brock, O. & Khatib, O. (2002). Elastic Strips: A Framework for Motion Generation in Human Environments. *The International Journal of Robotics Research*, 21(12), 1031–1052.

Bruyninckx, H. & Khatib, O. (2000). Gauss' principle and the dynamics of redundant and constrained manipulators. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3 (pp. 2563–2568).: IEEE.

Burget, F., Bennewitz, M., & Burgard, W. (2016). BI 2 RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3714–3721).: IEEE.

Carlson, J. & Murphy, R. (2005). How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3), 423–437.

Caron, S., Pham, Q.-C., & Nakamura, Y. (2017). ZMP Support Areas for Multicontact Mobility Under Frictional Constraints. *IEEE Transactions on Robotics*, 33(1), 67–80.

Cha, E., Forlizzi, J., & Srinivasa, S. S. (2015). Robots in the Home. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction - HRI '15* (pp. 319–326). New York, New York, USA: ACM Press.

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

Chen, T. L., Ciocarlie, M., Cousins, S., Grice, P. M., Hawkins, K., Kaijen Hsiao, Kemp, C. C., Chih-Hung King, Lazewatsky, D. a., Leeper, A. E., Hai Nguyen, Paepcke, A., Pantofaru, C., Smart, W. D., & Takayama, L. (2013). Robots for humanity: using assistive robotics to empower people with disabilities. *IEEE Robotics & Automation Magazine*, 20(1), 30–39.

Chin, K.-Y., Hong, Z.-W., & Chen, Y.-L. (2014). Impact of Using an Educational Robot-Based Learning System on Students' Motivation in Elementary Education. *IEEE Transactions on Learning Technologies*, 7(4), 333–345.

Cohen, A. & Shoham, M. (2016). Application of Hyper-Dual Numbers to Multibody Kinematics. *Journal of Mechanisms and Robotics*, 8(1), 2–5.

Collette, C., Micaelli, A., Andriot, C., & Lemerle, P. (2007). Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts. In *2007 7th IEEE-RAS International Conference on Humanoid Robots* (pp. 81–88).: IEEE.

Corke, P. I. (2017). *Robotics, Vision & Control*. Springer.

Dai, H. & Tedrake, R. (2016). Planning robust walking motion on uneven terrain via convex optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (pp. 579–586).: IEEE.

Dai, H., Valenzuela, A., & Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots* (pp. 295–302).: IEEE.

Dantam, N. T. (2020). Robust and efficient forward, differential, and inverse kinematics using dual quaternions. *The International Journal of Robotics Research*, (pp. 027836492093194).

Dariush, B., Youding Zhu, Arumbakkam, A., & Fujimura, K. (2010). Constrained closed loop inverse kinematics. In *2010 IEEE International Conference on Robotics and Automation* (pp. 2499–2506).: IEEE.

Decre, W., Smits, R., Bruyninckx, H., & De Schutter, J. (2009). Extending iTaSC to support inequality constraints and non-instantaneous task specification. In *2009 IEEE International Conference on Robotics and Automation* (pp. 964–971).: IEEE.

Deits, R. & Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, volume 2015-Febru (pp. 279–286).: IEEE.

Del Prete, A. (2018). Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators. *IEEE Robotics and Automation Letters*, 3(1), 281–288.

Desloge, E. A. (1987). Relationship between Kane's equations and the Gibbs-Appell equations. *Journal of Guidance, Control, and Dynamics*, 10(1), 120–122.

Desloge, E. A. (1988). The Gibbs-Appell equations of motion. *American Journal of Physics*, 56(9), 841–846.

Dietrich, A., Wimbock, T., Albu-Schaffer, A., & Hirzinger, G. (2012). Integration of Reactive, Torque-Based Self-Collision Avoidance Into a Task Hierarchy. *IEEE Transactions on Robotics*, 28(6), 1278–1293.

Escande, A., Brossette, S., & Kheddar, A. (2016). Parametrization of Catmull-Clark subdivision surfaces for posture generation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2016-June (pp. 1608–1614).: IEEE.

Escande, A., Kheddar, A., & Miossec, S. (2006). Planning support contact-points for humanoid robots and experiments on HRP-2. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2974–2979).: IEEE.

Escande, A., Kheddar, A., & Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5), 428–442.

Escande, A., Kheddar, A., Miossec, S., & Garsault, S. (2008). Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2. In *ISER'08: 11th International Symposium on Experimental Robotics* (pp. 293–302). Athens, Greece.

Escande, A., Kheddar, A., Miossec, S., & Garsault, S. (2009). *Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2*.

Escande, A., Mansard, N., & Wieber, P.-B. (2014a). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7), 1006–1028.

Escande, A., Miossec, S., Benallegue, M., & Kheddar, A. (2014b). A Strictly Convex Hull for Computing Proximity Distances With Continuous Gradients. *IEEE Transactions on Robotics*, 30(3), 666–678.

Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., & Buchli, J. (2017). Real-Time motion planning of legged robots: A model predictive control approach. *IEEE-RAS International Conference on Humanoid Robots*, (pp. 577–584).

Faverjon, B. & Tournassoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4 (pp. 1152–1159).: Institute of Electrical and Electronics Engineers.

Ferrari, A., Cutti, A. G., & Cappello, A. (2010). A new formulation of the coefficient of multiple correlation to assess the similarity of waveforms measured synchronously by different motion analysis protocols. *Gait & Posture*, 31(4), 540–542.

Fierro, R. & Lewis, F. L. (1997). Control of a nonholomic mobile robot: Backstepping kinematics into dynamics. *Journal of Robotic Systems*, 14(3), 149–163.

Figueredo, L. F. C., Adorno, B. V., Ishihara, J. Y., & Borges, G. A. (2013). Robust kinematic control of manipulator robots using dual quaternion representation. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 1949–1955). Karlsruhe: IEEE.

Fonseca, M. D. P. A. & Adorno, B. V. (2016). Whole-Body Modeling and Hierarchical Control of a Humanoid Robot Based on Dual Quaternion Algebra. In *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)* (pp. 103–108).: IEEE.

Gienger, M., Janben, H., & Goerick, C. (2006). Exploiting Task Intervals for Whole Body Robot Control. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2484–2490).: IEEE.

Goncalves, V. M., Fraisse, P., Crosnier, A., & Adorno, B. V. (2016). Parsimonious Kinematic Control of Highly Redundant Robots. *IEEE Robotics and Automation Letters*, 1(1), 65–72.

Gouasmi, M. (2012). Robot Kinematics using Dual Quaternions. *IAES International Journal of Robotics and Automation (IJRA)*, 1(1).

Hamilton, W. R. (1844). II. On quaternions; or on a new system of imaginaries in algebra. *Philosophical Magazine Series 3*, 25(163), 10–13.

Hammack, R. (2018). *Book of Proof.* 3th edition edition.

Hauser, K., Bretl, T., & Latomb, J.-C. (2005). Non-gaited humanoid locomotion planning. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* (pp. 7–12).: IEEE.

Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., & Wilcox, B. (2008). Motion Planning for Legged Robots on Varied Terrain. *The International Journal of Robotics Research*, 27(11-12), 1325–1349.

Hollerbach, J. M. (1980). A Recursive Lagrangian Formulation of Maniputator Dynamics and a Comparative Study of Dynamics Formulation Complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11), 730–736.

Honein, T. E. & O'Reilly, O. M. (2021). On the Gibbs-Appell Equations for the Dynamics of Rigid Bodies. *Journal of Applied Mechanics*, 88(7), 1–8.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & 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)* (pp. 1620–1626).: IEEE.

Kalaba, R. E. & Udwadia, F. E. (1992). A new perspective on constrained motion. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1906), 407–410.

Kalaba, R. E. & Udwadia, F. E. (1993). Equations of Motion for Nonholonomic, Constrained Dynamical Systems via Gauss's Principle. *Journal of Applied Mechanics*, 60(3), 662–668.

Kane, T. R. (1983). Formulation of dynamical equations of motion. *American Journal of Physics*, 51(11), 974–977.

Kanoun, O., Lamiraux, F., & 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.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011). Anytime Motion Planning using the RRT*. In *2011 IEEE International Conference on Robotics and Automation* (pp. 1478–1483).: IEEE.

Kelly, R., Santibanez, V., & Loria, A. (2005). *Control of Robot Manipulators in Joint Space.* Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag.

Khalil, H. K. (1996). *Nonlinear Systems.* Prentice Hall, 2nd edition.

Kim, I. & Oh, J.-H. (2013). Inverse Kinematic Control of Humanoids under Joint Constraints. *International Journal of Advanced Robotic Systems*, 10(1), 74.

Kong, X. (2017). Reconfiguration Analysis of Multimode Single-Loop Spatial Mechanisms Using Dual Quaternions. *Journal of Mechanisms and Robotics*, 9(5), 1–8.

Koptev, M., Figueroa, N., & Billard, A. (2021). Real-Time Self-Collision Avoidance in Joint Space for Humanoid Robots. *IEEE Robotics and Automation Letters*, 6(2), 1240–1247.

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, 21(3), 2012–2014.

Kussaba, H. T., Figueredo, L. F., Ishihara, J. Y., & Adorno, B. V. (2017). Hybrid Kinematic Control for Rigid Body Pose Stabilization using Dual Quaternions. *Journal of the Franklin Institute*.

Laumond, J.-P., Mansard, N., & Lasserre, J. B. (2015). Optimization as motion selection principle in robot action. *Communications of the ACM*, 58(5), 64–74.

Lengagne, S., Vaillant, J., Yoshida, E., & Kheddar, A. (2013). Generation of whole-body optimal dynamic multi-contact motions. *The International Journal of Robotics Research*, 32(9-10), 1104–1119.

Levinson, D. A. (1987). Comment on 'Relationship between Kane's equations and the Gibbs-Appell equations'. *Journal of Guidance, Control, and Dynamics*, 10(6), 593–593.

Lewis, A. D. (1996). The geometry of the Gibbs-Appell equations and Gauss' principle of least constraint. *Reports on Mathematical Physics*, 38(1), 11–28.

Marinho, M. M., Adorno, B. V., Harada, K., & Mitsuishi, M. (2018). Active Constraints Using Vector Field Inequalities for Surgical Robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5364–5371).: IEEE.

Marinho, M. M., Adorno, B. V., Harada, K., & Mitsuishi, M. (2019). Dynamic Active Constraints for Surgical Robots Using Vector-Field Inequalities. *IEEE Transactions on Robotics*, 35(5), 1166–1185.

Marinho, M. M., Figueredo, L. F. C., & Adorno, B. V. (2015). A dual quaternion linear-quadratic optimal controller for trajectory tracking. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2015-Decem (pp. 4047–4052).: IEEE.

Moll, M., Sucan, I. A., & Kavraki, L. E. (2015). Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. *IEEE Robotics & Automation Magazine*, 22(3), 96–102.

Mordatch, I., Lowrey, K., & Todorov, E. (2015). Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5307–5314).: IEEE.

Mordatch, I., Todorov, E., & Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4), 1–8.

Murooka, M., Okada, K., & Inaba, M. (2020a). Optimization-Based Posture Generation for Whole-Body Contact Motion by Contact Point Search on the Body Surface. *IEEE Robotics and Automation Letters*, 5(2), 2905–2912.

Murooka, M., Okada, K., & Inaba, M. (2020b). Optimization-Based Posture Generation for Whole-Body Contact Motion by Contact Point Search on the Body Surface. *IEEE Robotics and Automation Letters*, 5(2), 2905–2912.

Ogata, K. (2010). *Modern Control Engineering*. Prentice Hall, 5th edition edition.

Oliveira, A. C. & Adorno, B. V. (2015). Balance Control of a Humanoid Robot Based on the Cooperative Dual Task-Space Framework. In *XII Simpósio Brasileiro de Automação Inteligente (SBAI)* (pp. 485 – 490).

Osorio, J. D. M., Abdelazim, A., Allmendinger, F., & Zimmermann, U. E. (2020). Unilateral Constraints for Torque-based Whole-Body Control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 7623–7628).: IEEE.

Özgür, E. & Mezouar, Y. (2016). Kinematic modeling and control of a robot arm using unit dual quaternions. *Robotics and Autonomous Systems*, 77, 66–73.

Patel, R. V., Shadpey, F., Ranjbaran, F., & Angeles, J. (2005). A collision-avoidance scheme for redundant manipulators: Theory and experiments. *Journal of Robotic Systems*, 22(12), 737–757.

Perez, A. & McCarthy, J. M. (2004). Dual Quaternion Synthesis of Constrained Robotic Systems. *Journal of Mechanical Design*, 126(3), 425.

Pettré, J., Laumond, J., & Siméon, T. (2003). A 2-Stages Locomotion Planner for Digital Actors. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 3, 258–264.

Pham, H.-L., Perdereau, V., Adorno, B. V., & Fraisse, P. (2010). Position and orientation control of robot manipulators using dual quaternion feedback. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 658–663). Taipei: IEEE.

Ponton, B., Herzog, A., Del Prete, A., Schaal, S., & Righetti, L. (2018). On Time Optimization of Centroidal Momentum Dynamics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5776–5782).: IEEE.

Ponton, B., Herzog, A., Schaal, S., & 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)* (pp. 842–849).: IEEE.

Ponton, B., Khadiv, M., Meduri, A., & Righetti, L. (2021). Efficient Multicontact Pattern Generation With Sequential Convex Approximations of the Centroidal Dynamics. *IEEE Transactions on Robotics*, (pp. 1–19).

Quiroz-Omaña, J. J. & Adorno, B. V. (2018). Whole-Body Kinematic Control of Nonholonomic Mobile Manipulators Using Linear Programming. *Journal of Intelligent & Robotic Systems*, 91(2), 263–278.

Quiroz-Omana, J. J. & Adorno, B. V. (2019). Whole-Body Control With (Self) Collision Avoidance Using Vector Field Inequalities. *IEEE Robotics and Automation Letters*, 4(4), 4048–4053.

Ray, J. R. (1972). Nonholonomic Constraints and Gauss's Principle of Least Constraint. *American Journal of Physics*, 40(1), 179–183.

Ray, J. R. (1992). Geometry of constraints and the Gauss-Appell principle of least constraint. *Kuwait Journal of Science*, 19(1), 11–15.

Redon, S., Kheddar, A., & Coquillart, S. (2002). Gauss' least constraints principle and rigid body simulations. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1 (pp. 517–522).: IEEE.

Saab, L., Ramos, O. E., Keith, F., Mansard, N., Soueres, P., & Fourquet, J.-Y. (2013). Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints. *IEEE Transactions on Robotics*, 29(2), 346–362.

Samadi, S., Caron, S., Tanguy, A., & Kheddar, A. (2020). Balance of Humanoid Robots in a Mix of Fixed and Sliding Multi-Contact Scenarios. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, number 1 (pp. 6590–6596).: IEEE.

Samadi, S., Roux, J., Tanguy, A., Caron, S., & Kheddar, A. (2021). Humanoid Control Under Interchangeable Fixed and Sliding Unilateral Contacts. *IEEE Robotics and Automation Letters*, 6(2), 4032–4039.

Schwienbacher, M., Buschmann, T., Lohmeier, S., Favot, V., & Ulbrich, H. (2011). Self-collision avoidance and angular momentum compensation for a biped humanoid robot. In *2011 IEEE International Conference on Robotics and Automation*, number January 2016 (pp. 581–586).: IEEE.

Selig, J. (2005). *Geometric Fundamentals of Robotics*. Monographs in Computer Science. New York, NY: Springer New York.

Sentis, L. (2007). *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford University.

Sentis, L. & Khatib, O. (2004). Task-oriented control of humanoid robots through prioritization. In *IEEE International Conference on Humanoid Robots.* (pp. 1–16).: IEEE.

Shu-Yun Chung & Khatib, O. (2015). Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6289–6294).: IEEE.

Siciliano, B. & Khatib, O. (2018). Humanoid Robots: Historical Perspective, Overview and Scope. In *Humanoid Robotics: A Reference* (pp. 1–6). Dordrecht: Springer Netherlands.

Silva, F. F. A., Quiroz-Omaña, J. J., & Adorno, B. V. (2020). Dynamics of serial manipulators using dual quaternion algebra.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot Modeling and Control*. New York: Wiley.

Stasse, O., Escande, A., Mansard, N., Miossec, S., Evrard, P., & Kheddar, A. (2008). Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In *2008 IEEE International Conference on Robotics and Automation*, volume 25 (pp. 3200–3205).: IEEE.

Storch, J. & Gates, S. (1989). Motivating Kane's method for obtaining equations of motion for dynamic systems. *Journal of Guidance, Control, and Dynamics*, 12(4), 593–595.

Sugiura, H., Gienger, M., Janssen, H., & Goerick, C. (2007). Real-time collision avoidance with whole body motion control for humanoid robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2053–2058).: IEEE.

Taylor, R. (2006). A Perspective on Medical Robotics. *Proceedings of the IEEE*, 94(9), 1652–1664.

Tonneau, S., Del Prete, A., Pettre, J., Park, C., Manocha, D., & Mansard, N. (2018). An Efficient Acyclic Contact Planner for Multiped Robots. *IEEE Transactions on Robotics*, 34(3), 586–601.

Tonneau, S., Song, D., Fernbach, P., Mansard, N., Taix, M., & Del Prete, A. (2020). SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, number Section IV (pp. 6604–6610).: IEEE.

Townsend, M. A. (1992). Equivalence of Kane's, Gibbs-Appell's, and Lagrange's equations. *Journal of Guidance, Control, and Dynamics*, 15(5), 1289–1292.

Udwadia, F. E. & Kalaba, R. E. (1998). The explicit Gibbs-Appell equation and generalized inverse forms. *Quarterly of Applied Mathematics*, 56(2), 277–288.

VUKOBRATOVIĆ, M. & BOROVAC, B. (2004). ZERO-MOMENT POINT - THIRTY FIVE YEARS OF ITS LIFE. *International Journal of Humanoid Robotics*, 01(01), 157–173.

Wang, X. & Yu, C. (2013). Unit dual quaternion-based feedback linearization tracking problem for attitude and position dynamics. *Systems & Control Letters*, 62(3), 225–233.

Wieber, P.-B. (2005). Some comments on the structure of the dynamics of articulated motion. *the Ruperto Carola Symp. on Fast Motions in Biomechanics and Robotics*.

Wieber, P.-B. (2006). Holonomy and Nonholonomy in the Dynamics of Articulated Motion. In *Fast Motions in Biomechanics and Robotics*, volume 340 (pp. 411–425). Berlin, Heidelberg: Springer Berlin Heidelberg.

Witkin, A. & Kass, M. (1988). Spacetime constraints. *ACM SIGGRAPH Computer Graphics*, 22(4), 159–168.

Xiangke Wang, Changbin Yu, & Zhiyun Lin (2012). A Dual Quaternion Solution to Attitude and Position Control for Rigid-Body Coordination. *IEEE Transactions on Robotics*, 28(5), 1162–1170.

A

Dual Quaternion Algebra

This chapter reviews some concepts, foundations and operations related to quaternions and dual quaternions.

A.1 Fundamentals of Dual Quaternion Algebra

Unit dual quaternions have proven to be a powerful mathematical tool in robotics, not only in the representation of rigid motions, but also in robot modeling (Adorno 2011; Selig 2005), robot design (Perez & McCarthy, 2004), and control (Pham et al., 2010; Xiangke Wang et al., 2012; Figueredo et al., 2013; Wang & Yu, 2013; Marinho et al., 2015; Kussaba et al., 2017). They are more compact and computationally efficient than homogeneous transformation matrices and also do not present representational singularities (Adorno, 2011; Adorno & Fraisse, 2017). Thanks to their strong algebraic properties, different robots can be modeled using the same systematic procedure (e.g., single or cooperative manipulators (Adorno et al., 2010), mobile manipulators (Adorno, 2011) and humanoids (Oliveira & Adorno, 2015; Fonseca & Adorno, 2016), and the resultant models can be directly used with standard kinematic controllers without the need of any intermediate parameterization (Pham et al., 2010; Figueredo et al., 2013). Unit dual quaternions represents rigid motions in a very compact way, by combining a unit quaternion[1] representing rotation and a pure quaternion[2] representing translation.

[1] $\boldsymbol{h} \in \mathbb{H}$ is a unit quaternion if $\|\boldsymbol{h}\| = 1$.
[2] $\boldsymbol{h} \in \mathbb{H}$ is a pure quaternion if $\mathrm{Re}\,(\boldsymbol{h}) = 0$.

A.1.1 Quaternions (Adorno, 2017)

Quaternions are algebraic structures that can be regarded as an extension of complex numbers introduced first by Sir William Rowan Hamilton in 1843 (1844, apud Adorno 2011). They are composed of a real part and three imaginary components $\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$, also called imaginary or quaternionic units. The imaginary units have the following properties

$$\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = \hat{\imath}\hat{\jmath}\hat{k} = -1. \tag{A.1}$$

The set $\mathbb{H}$ of quaternions is defined as

$$\mathbb{H} \triangleq \left\{ h_1 + \hat{\imath}h_2 + \hat{\jmath}h_3 + \hat{k}h_4 \ : \ h_1, h_2, h_3, h_4 \in \mathbb{R} \right\}.$$

Definition A.1. Given $\boldsymbol{h} \in \mathbb{H}$, its real part is denoted by $\mathrm{Re}\,(\boldsymbol{h}) \triangleq h_1$, and its imaginary part is denoted by $\mathrm{Im}\,(\boldsymbol{h}) \triangleq \hat{\imath}h_2 + \hat{\jmath}h_3 + \hat{k}h_4$, such that $\boldsymbol{h} = \mathrm{Re}\,(\boldsymbol{h}) + \mathrm{Im}\,(\boldsymbol{h})$.

Definition A.2. Given $\boldsymbol{h} \in \mathbb{H}$, its conjugate is defined as

$$\boldsymbol{h}^* \triangleq \mathrm{Re}\,(\boldsymbol{h}) - \mathrm{Im}\,(\boldsymbol{h}). \tag{A.2}$$

Definition A.3. Given $\boldsymbol{h} \in \mathbb{H}$, its norm is defined as

$$\|\boldsymbol{h}\| \triangleq \sqrt{\boldsymbol{h}^*\boldsymbol{h}} = \sqrt{\boldsymbol{h}\boldsymbol{h}^*}. \tag{A.3}$$

Definition A.4. The $\mathrm{vec}_4 : \mathbb{H} \to \mathbb{R}^4$ operator performs a one-to-one mapping. Given $\boldsymbol{h} \in \mathbb{H}$, this operator is defined as (Adorno, 2017)

$$\mathrm{vec}_4\,(\boldsymbol{h}) \triangleq \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix}^T. \tag{A.4}$$

Definition A.5. Given $\boldsymbol{h} \in \mathbb{H}$, the Hamilton operators are defined as (Adorno, 2017)

$$\overset{+}{\boldsymbol{H}}_4(\boldsymbol{h}) \triangleq \begin{bmatrix} h_1 & -h_2 & -h_3 & -h_4 \\ h_2 & h_1 & -h_4 & h_3 \\ h_3 & h_4 & h_1 & -h_2 \\ h_4 & -h_3 & h_2 & h_1 \end{bmatrix}, \quad \overset{-}{\boldsymbol{H}}_4(\boldsymbol{h}) \triangleq \begin{bmatrix} h_1 & -h_2 & -h_3 & -h_4 \\ h_2 & h_1 & h_4 & -h_3 \\ h_3 & -h_4 & h_1 & h_2 \\ h_4 & h_3 & -h_2 & h_1 \end{bmatrix} \tag{A.5}$$

Definition A.6. Given $\boldsymbol{a}$,$\boldsymbol{b} \in \mathbb{H}$, the Hamilton operators satisfy the following equalities (Adorno, 2017)

$$\mathrm{vec}_4\,(\boldsymbol{a}\boldsymbol{b}) = \overset{+}{\boldsymbol{H}}_4(\boldsymbol{a})\,\mathrm{vec}_4\,(\boldsymbol{b}) = \overset{-}{\boldsymbol{H}}_4(\boldsymbol{b})\,\mathrm{vec}_4\,(\boldsymbol{a}). \tag{A.6}$$

Definition A.7. The conjugating matrix $\boldsymbol{C}_4$ is defined as (Adorno, 2011)

$$\boldsymbol{C}_4 \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{A.7}$$

Given $\boldsymbol{h} \in \mathbb{H}$, this matrix satisfies the following condition

$$\operatorname{vec}_4(\boldsymbol{h}^*) = \boldsymbol{C}_4 \operatorname{vec}_4(\boldsymbol{h}).$$

The set of pure quaternions is defined as (Adorno, 2017)

$$\mathbb{H}_p \triangleq \{\boldsymbol{p} \in \mathbb{H} \ : \ \operatorname{Re}(\boldsymbol{p}) = 0\}$$

Definition A.8. Given $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{H}_p$, the cross product is defined as (Adorno, 2017)

$$\boldsymbol{a} \times \boldsymbol{b} \triangleq \frac{\boldsymbol{ab} - \boldsymbol{ba}}{2}. \tag{A.8}$$

Definition A.9. Given $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{H}_p$, the dot product is defined as (Adorno, 2017)

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = -\frac{\boldsymbol{ab} + \boldsymbol{ba}}{2} = \operatorname{vec}_4 \boldsymbol{a}^T \operatorname{vec}_4 \boldsymbol{b} = \operatorname{vec}_4 \boldsymbol{b}^T \operatorname{vec}_4 \boldsymbol{a}. \tag{A.9}$$

Definition A.10. Given $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{H}_p$, the $\overset{+}{\boldsymbol{S}}()$ operator is defined as (Adorno, 2011)

$$\boldsymbol{S}(\boldsymbol{a}) = \frac{1}{2}\left[\overset{+}{\boldsymbol{H}}_4(\boldsymbol{a}) - \overset{-}{\boldsymbol{H}}_4(\boldsymbol{a})\right], \tag{A.10}$$

and satisfies the following equalities

$$\operatorname{vec}_4(\boldsymbol{a} \times \boldsymbol{b}) = \boldsymbol{S}(\boldsymbol{a}) \operatorname{vec}_4(\boldsymbol{b}) = -\boldsymbol{S}(\boldsymbol{b}) \operatorname{vec}_4(\boldsymbol{a}) = \boldsymbol{S}(\boldsymbol{b})^T \operatorname{vec}_4(\boldsymbol{a}).$$

Definition A.11. Given a unit quaternion $\boldsymbol{r}_b^a$ and a pure quaternion $\boldsymbol{p}_{a,b}^a$, the adjoint transformation is defined as

$$\operatorname{Ad}(\boldsymbol{r}_b^a)\, \boldsymbol{p}_{a,b}^a \triangleq \boldsymbol{r}_b^a \boldsymbol{p}_{a,b}^a \boldsymbol{r}_b^{a*}. \tag{A.11}$$

Definition A.12. The set of quaternions with unit norm is defined as $\mathbb{S}^3 \triangleq \{\boldsymbol{h} \in \mathbb{H} \ : \ \|\boldsymbol{h}\| = 1\}$. Unit quaternions are useful to represent rigid rotations, and $\boldsymbol{r} \in \mathbb{S}^3$ always can be written as $\boldsymbol{r} = \cos(\phi/2) + \boldsymbol{n}\sin(\phi/2)$, where $\phi \in \mathbb{R}$ is the rotation angle around the rotation axis $\boldsymbol{n} \in \mathbb{S}^3 \cap \mathbb{H}_p$ (Adorno, 2017).

Definition A.13. Given $\boldsymbol{a} \in \mathbb{H}_p$, the time-derivative of the squared norm of $\boldsymbol{a}$, when it

exists, is given by

$$\frac{d}{dt}\left(\|\boldsymbol{a}\|^2\right) = \dot{\boldsymbol{a}}\boldsymbol{a}^* + \boldsymbol{a}\dot{\boldsymbol{a}}^* = 2\langle\dot{\boldsymbol{a}}, \boldsymbol{a}\rangle. \tag{A.12}$$

Given $\boldsymbol{r}_b^a \in \mathbb{S}^3$, its time derivative $\dot{\boldsymbol{r}}$ is related with the angular velocity of a frame as (Adorno, 2017)

$$\dot{\boldsymbol{r}}_b^a = \frac{1}{2}\boldsymbol{\omega}_{ab}^a \boldsymbol{r}_b^a = \frac{1}{2}\boldsymbol{r}_b^a \boldsymbol{\omega}_{ab}^b, \tag{A.13}$$

Definition A.14. where $\boldsymbol{r}_b^a$ is the rotation of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$, $\boldsymbol{\omega}_{ab}^a$ and $\boldsymbol{\omega}_{ab}^b$ are the angular velocity of $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$, expressed in frame $\mathcal{F}_a$ and $\mathcal{F}_b$, respectively.

A.1.2 Dual Quaternions (Adorno, 2017)

Dual quaternions are dual numbers where the primary and dual part are quaternions. The dual numbers were introduced by Clifford (Adorno, 2011) and can be regarded as an extension of quaternions.

Definition A.15. Given two numbers $d_{\mathcal{P}}$ and $d_{\mathcal{D}}$, the dual number $\underline{\boldsymbol{d}}$ is defined as (Adorno, 2011)

$$\underline{\boldsymbol{d}} = d_{\mathcal{P}} + \varepsilon d_{\mathcal{D}}, \tag{A.14}$$

where ε is the dual unit proposed by Clifford (1873), which is nilpotent and follows the following property

$$\varepsilon^2 = 0 \text{ with } \varepsilon \neq 0. \tag{A.15}$$

The primary part and the dual part can be extracted using the operators $\mathcal{P}(\underline{\boldsymbol{d}})$ and $\mathcal{D}(\underline{\boldsymbol{d}})$, respectively. For instance, in (A.14) $\mathcal{P}(\underline{\boldsymbol{d}}) = d_{\mathcal{P}}$ and $\mathcal{D}(\underline{\boldsymbol{d}}) = d_{\mathcal{D}}$.

The set of dual quaternions is defined as

$$\mathcal{H} \triangleq \left\{\boldsymbol{h}_1 + \varepsilon\boldsymbol{h}_2 \ : \ \boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{H},\ \varepsilon \neq 0,\ \varepsilon^2 = 0\right\}.$$

Definition A.16. Given the dual quaternion $\underline{\boldsymbol{h}} = \boldsymbol{h}_{\mathcal{P}} + \varepsilon\boldsymbol{h}_{\mathcal{D}}$, its conjugate is defined as

$$\underline{\boldsymbol{h}}^* \triangleq \boldsymbol{h}_{\mathcal{P}}^* + \varepsilon\boldsymbol{h}_{\mathcal{D}}^*. \tag{A.16}$$

Definition A.17. Given $\underline{\boldsymbol{h}} \in \mathcal{H}$, its norm is defined as

$$\|\boldsymbol{h}\| \triangleq \sqrt{\underline{\boldsymbol{h}}^*\underline{\boldsymbol{h}}} = \sqrt{\underline{\boldsymbol{h}}\underline{\boldsymbol{h}}^*}.$$

Definition A.18. The $\mathrm{vec}_8 : \mathbb{H} \to \mathbb{R}^8$ operator performs a one-to-one mapping. Given the quaternion $\underline{\boldsymbol{h}} = h_1 + \hat{\imath}h_2 + \hat{\jmath}h_3 + \hat{k}h_4 + \varepsilon\left(h_5 + \hat{\imath}h_6 + \hat{\jmath}h_7 + \hat{k}h_8\right)$, this operator is defined as

$$\mathrm{vec}_8(\underline{\boldsymbol{h}}) \triangleq \begin{bmatrix} h_1 & \dots & h_8 \end{bmatrix}^T.$$

The multiplication and addition operations between dual quaternions follow the same rules of their counterparts between real numbers, but respecting the additional rules determined by (A.1) and (A.15). It can be verified that, in general, multiplication of dual quaternions is not commutative (i.e., given $\underline{\boldsymbol{x}},\underline{\boldsymbol{y}} \in \mathcal{H}$, in general $\underline{\boldsymbol{x}}\underline{\boldsymbol{y}} \neq \underline{\boldsymbol{y}}\underline{\boldsymbol{x}}$). However, one can use Hamilton operators (Adorno, 2011) for manipulating algebraic expressions containing dual quaternions such that

$$\operatorname{vec}_8\left(\underline{\boldsymbol{x}}\underline{\boldsymbol{y}}\right) = \overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{x}}\right)\operatorname{vec}_8\underline{\boldsymbol{y}} = \overset{-}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{y}}\right)\operatorname{vec}_8\left(\underline{\boldsymbol{x}}\right),$$

where

$$\overset{+}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{h}}\right) \triangleq \begin{bmatrix} \overset{+}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{P}}\right) & \boldsymbol{0}_{4\times 4} \\ \overset{+}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{D}}\right) & \overset{+}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{P}}\right) \end{bmatrix}, \quad \overset{-}{\boldsymbol{H}}_8\left(\underline{\boldsymbol{h}}\right) \triangleq \begin{bmatrix} \overset{-}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{P}}\right) & \boldsymbol{0}_{4\times 4} \\ \overset{-}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{D}}\right) & \overset{-}{\boldsymbol{H}}_4\left(\boldsymbol{h}_{\mathcal{P}}\right) \end{bmatrix}. \tag{A.17}$$

Definition A.19. The conjugating matrix $\boldsymbol{C}_8$ is defined as (Adorno, 2011)

$$\boldsymbol{C}_8 \triangleq \begin{bmatrix} \boldsymbol{C}_4 & 0 \\ 0 & \boldsymbol{C}_4 \end{bmatrix}. \tag{A.18}$$

Given $\underline{\boldsymbol{h}} \in \mathcal{H}$, this matrix satisfies the following condition

$$\operatorname{vec}_8\left(\boldsymbol{h}^*\right) = \boldsymbol{C}_8\operatorname{vec}_8\left(\boldsymbol{h}\right).$$

The set of unit dual quaternions defined as $\underline{\boldsymbol{\mathcal{S}}} \triangleq \{\underline{\boldsymbol{h}} \in \mathcal{H} \ : \ \|\underline{\boldsymbol{h}}\| = 1\}$ can represent rigid motions, and $\underline{\boldsymbol{x}} \in \underline{\boldsymbol{\mathcal{S}}}$ always can be written as $\underline{\boldsymbol{x}} = \boldsymbol{r} + \varepsilon\frac{1}{2}\boldsymbol{p}\boldsymbol{r}$, where $\boldsymbol{r} \in \mathbb{S}^3$ and $\boldsymbol{p} \in \mathbb{H}_p$ represent the orientation and position, respectively.

Furthermore, elements of the set $\mathcal{H}_p \triangleq \{\underline{\boldsymbol{h}} \in \mathcal{H} \ : \ \operatorname{Re}\left(\underline{\boldsymbol{h}}\right) = 0\}$ are called pure dual quaternions.

Definition A.20. Given the frames $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$, the unit dual quaternions $\underline{\boldsymbol{x}}_1^0$ and $\underline{\boldsymbol{x}}_2^1$ represent the rigid motions from $\mathcal{F}_0$ to $\mathcal{F}_1$ and $\mathcal{F}_1$ to $\mathcal{F}_2$, respectively. The transformation from $\mathcal{F}_0$ to $\mathcal{F}_2$ is given by $\underline{\boldsymbol{x}}_2^0 = \underline{\boldsymbol{x}}_1^0\underline{\boldsymbol{x}}_2^1$ (Adorno, 2011), as shown in Fig. A.1.

Definition A.21. Let $\underline{\boldsymbol{h}} = \boldsymbol{r} + \varepsilon\frac{1}{2}\boldsymbol{p}\boldsymbol{r}$, with $\boldsymbol{r} = \cos\left(\phi/2\right) + \boldsymbol{n}\sin\left(\phi/2\right)$. The logarithm of $\underline{\boldsymbol{h}}$ is

$$\log\underline{\boldsymbol{h}} = \frac{\phi\boldsymbol{n}}{2} + \varepsilon\frac{\boldsymbol{p}}{2}.$$

Definition A.22. Let $\underline{\boldsymbol{g}} \in \mathcal{H}_p$, the exponential of $\underline{\boldsymbol{g}}$ is

$$\exp\underline{\boldsymbol{g}} = \mathcal{P}\left(\exp\underline{\boldsymbol{g}}\right) + \varepsilon\,\mathcal{D}\left(\underline{\boldsymbol{g}}\right)\mathcal{P}\left(\exp\underline{\boldsymbol{g}}\right),$$

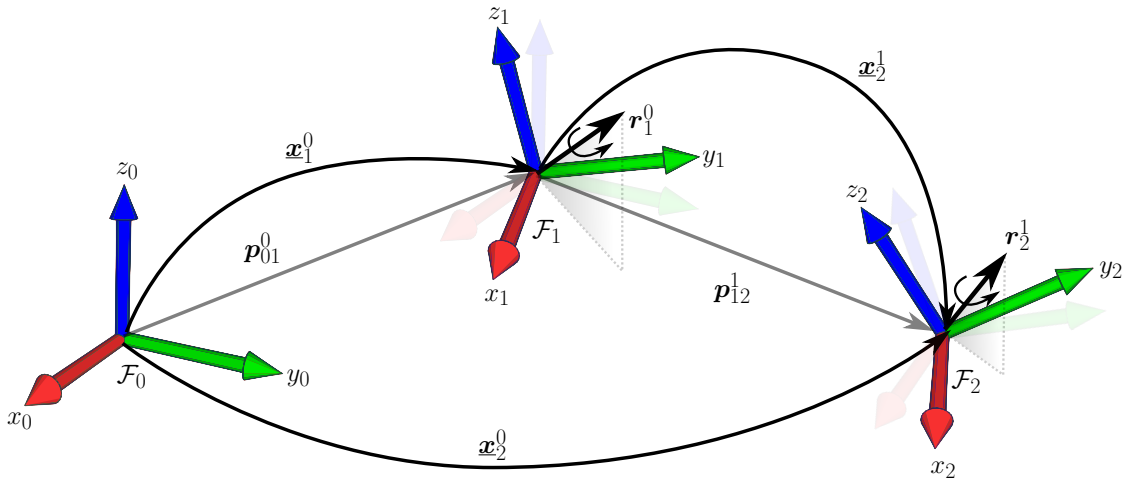

Figure A.1: Sequence of rigid transformations using dual quaternions.

where

$$\mathcal{P}\left(\exp \underline{\boldsymbol{g}}\right) = \begin{cases} \cos\left\|\mathcal{P}\left(\underline{\boldsymbol{g}}\right)\right\| + \frac{\sin\left\|\mathcal{P}\left(\underline{\boldsymbol{g}}\right)\right\|}{\left\|\mathcal{P}\left(\underline{\boldsymbol{g}}\right)\right\|} \mathcal{P}\left(\underline{\boldsymbol{g}}\right) & \text{if } \left\|\mathcal{P}\left(\underline{\boldsymbol{g}}\right)\right\| \neq 0 \\ 1 & \text{otherwise.} \end{cases}$$

Definition A.23. The geometrical exponentiation of the unit dual quaternion $\underline{\boldsymbol{h}} \in \underline{\boldsymbol{\mathcal{S}}}$ (i.e., $\underline{\boldsymbol{h}}$ raised to the λ-th power) is defined as

$$\underline{\boldsymbol{h}}^{\{\lambda\}} \triangleq \exp\left(\lambda \log \underline{\boldsymbol{h}}\right).$$

Given $\underline{\boldsymbol{x}}_b^a \in \underline{\boldsymbol{\mathcal{S}}}$, its time derivative can be expressed as (Adorno, 2017)

$$\dot{\underline{\boldsymbol{x}}}_b^a = \frac{1}{2}\underline{\boldsymbol{\xi}}_{a,b}^a \underline{\boldsymbol{x}}_b^a = \frac{1}{2}\underline{\boldsymbol{x}}_b^a \underline{\boldsymbol{\xi}}_{a,b}^b, \tag{A.19}$$

where $\underline{\boldsymbol{x}}_b^a$ denotes the rigid transformation of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$. The twist

$$\underline{\boldsymbol{\xi}}_{a,b}^a = \boldsymbol{\omega}_{a,b}^a + \varepsilon\left(\dot{\boldsymbol{p}}_{a,b}^a + \boldsymbol{p}_{a,b}^a \times \boldsymbol{\omega}_{a,b}^a\right) \tag{A.20}$$

represents the twist of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$, expressed in frame $\mathcal{F}_a$. Furthermore, the twist $\underline{\boldsymbol{\xi}}_{a,b}^b$ is the twist of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$, expressed in frame $\mathcal{F}_b$.[3]

Remark A.1. Consider the twists $\underline{\boldsymbol{\xi}}_{ab}^a$ given by (A.20). Then, the twist $\underline{\boldsymbol{\xi}}_{ab}^b$ is given as

$$\underline{\boldsymbol{\xi}}_{a,b}^b = \underline{\boldsymbol{x}}_b^{a*} \underline{\boldsymbol{\xi}}_{a,b}^a \underline{\boldsymbol{x}}_b^a. \tag{A.21}$$

[3]So, for example, $\underline{\boldsymbol{\xi}}_{a,b}^c$ is the twist of frame $\mathcal{F}_b$ with respect to frame $\mathcal{F}_a$, expressed in frame $\mathcal{F}_c$.

Expanding (A.21), we have

$$\underline{\boldsymbol{\xi}}_{a,b}^{b} = \mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{D}\left(\underline{\boldsymbol{\xi}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{D}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\underline{\boldsymbol{\xi}}_{a,b}^{a}\right)\mathcal{D}\left(\underline{\boldsymbol{x}}_{b}^{a}\right). \quad \text{(A.22)}$$

Using (A.21),(A.22), (A.8), and using the facts $\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) = \boldsymbol{r}_{b}^{a}$, $\mathcal{D}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) = (1/2)\,\boldsymbol{p}_{a,b}^{a}\boldsymbol{r}_{b}^{a}$ and $\boldsymbol{p}_{a,b}^{a*} = -\boldsymbol{p}_{a,b}^{a}$ we rewrite (A.22) as follows

$$\underline{\boldsymbol{\xi}}_{a,b}^{b} = \mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\boldsymbol{\omega}_{a,b}^{a} + \varepsilon\mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\dot{\boldsymbol{p}}_{a,b}^{a} + \varepsilon\mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\left(\boldsymbol{p}_{a,b}^{a} \times \boldsymbol{\omega}_{a,b}^{a}\right) + \varepsilon\mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\left(\boldsymbol{\omega}_{a,b}^{a} \times \boldsymbol{p}_{a,b}^{a}\right). \quad \text{(A.23)}$$

Finally, since $\boldsymbol{\omega}_{a,b}^{a} \times \boldsymbol{p}_{a,b}^{a} = -\boldsymbol{p}_{a,b}^{a} \times \boldsymbol{\omega}_{a,b}^{a}$ from (A.23), we obtain

$$\underline{\boldsymbol{\xi}}_{a,b}^{b} = \boldsymbol{\omega}_{a,b}^{b} + \varepsilon\dot{\boldsymbol{p}}_{a,b}^{b}. \quad \text{(A.24)}$$

Remark A.2. The time derivative of (A.21) is computed as

$$\dot{\underline{\boldsymbol{\xi}}}_{ab}^{b} = \dot{\underline{\boldsymbol{x}}}_{b}^{a*}\underline{\boldsymbol{\xi}}_{ab}^{a}\underline{\boldsymbol{x}}_{b}^{a} + \underline{\boldsymbol{x}}_{b}^{a*}\dot{\underline{\boldsymbol{\xi}}}_{ab}^{a}\underline{\boldsymbol{x}}_{b}^{a} + \underline{\boldsymbol{x}}_{b}^{a*}\underline{\boldsymbol{\xi}}_{ab}^{a}\dot{\underline{\boldsymbol{x}}}_{b}^{a} \implies \dot{\underline{\boldsymbol{\xi}}}_{ab}^{b} = \mathrm{Ad}\left(\dot{\underline{\boldsymbol{x}}}_{b}^{a*}\right)\dot{\underline{\boldsymbol{\xi}}}_{ab}^{a}. \quad \text{(A.25)}$$

Expanding (A.25), we have

$$\dot{\underline{\boldsymbol{\xi}}}_{ab}^{b} = \mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\dot{\underline{\boldsymbol{\xi}}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{D}\left(\dot{\underline{\boldsymbol{\xi}}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{D}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\dot{\underline{\boldsymbol{\xi}}}_{a,b}^{a}\right)\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a}\right) + \varepsilon\,\mathcal{P}\left(\underline{\boldsymbol{x}}_{b}^{a*}\right)\mathcal{P}\left(\dot{\underline{\boldsymbol{\xi}}}_{a,b}^{a}\right)\mathcal{D}\left(\underline{\boldsymbol{x}}_{b}^{a}\right). \quad \text{(A.26)}$$

Using

$$\dot{\underline{\boldsymbol{\xi}}}_{ab}^{b} = \mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\dot{\boldsymbol{\omega}}_{a,b}^{a} + \varepsilon\mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\left(\ddot{\boldsymbol{p}}_{a,b}^{a} + \dot{\boldsymbol{p}}_{a,b}^{a} \times \boldsymbol{\omega}_{a,b}^{a} + \boldsymbol{p}_{a,b}^{a} \times \dot{\boldsymbol{\omega}}_{a,b}^{a}\right) + \varepsilon\mathrm{Ad}\left(\boldsymbol{r}_{b}^{a*}\right)\left(\dot{\boldsymbol{\omega}}_{a,b}^{a} \times \boldsymbol{p}_{a,b}^{a}\right)$$

$$\dot{\underline{\boldsymbol{\xi}}}_{ab}^{b} = \dot{\boldsymbol{\omega}}_{a,b}^{b} + \varepsilon\left(\ddot{\boldsymbol{p}}_{a,b}^{b} + \dot{\boldsymbol{p}}_{a,b}^{a} \times \boldsymbol{\omega}_{a,b}^{a}\right). \quad \text{(A.27)}$$

B

Constrained Euler-Lagrange Model using Lagrange Multipliers

The Euler-Lagrange equations under equality constraints in the form $\boldsymbol{A}\ddot{\boldsymbol{q}} = \boldsymbol{b}$ can be written by means of the Lagrange multipliers as

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} = \boldsymbol{Q} + \boldsymbol{A}^T\boldsymbol{\lambda}, \tag{B.1}$$

where $\boldsymbol{Q} \triangleq \boldsymbol{Q}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \bar{\boldsymbol{\tau}}_{\mathrm{GP}} - \boldsymbol{C}_{\mathrm{GP}}\dot{\boldsymbol{q}}$, and $\boldsymbol{\lambda}$ denotes the Lagrange multipliers. Solving for the joint accelerations from (B.1), we have

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}_{\mathrm{GP}}^{-1}\left(\boldsymbol{Q} + \boldsymbol{A}^T\boldsymbol{\lambda}\right). \tag{B.2}$$

Using (B.2) in the equality constraints $\boldsymbol{A}\ddot{\boldsymbol{q}} = \boldsymbol{b}$, we have

$$\begin{aligned}
\underbrace{\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\left(\boldsymbol{Q} + \boldsymbol{A}^T\boldsymbol{\lambda}\right)}_{\ddot{\boldsymbol{q}}} &= \boldsymbol{b}, \\
\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\boldsymbol{Q} + \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\boldsymbol{A}^T\boldsymbol{\lambda} &= \boldsymbol{b}, \\
\boldsymbol{\lambda} &= \left(\boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\boldsymbol{A}^T\right)^{-1}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\boldsymbol{Q}\right).
\end{aligned} \tag{B.3}$$

Finally, using (B.3) we rewrite (B.1) as

$$\boldsymbol{M}_{\mathrm{GP}}\ddot{\boldsymbol{q}} = \boldsymbol{Q} + \boldsymbol{\mathcal{R}}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{M}_{\mathrm{GP}}^{-1}\boldsymbol{Q}\right), \tag{B.4}$$

where

$$\boldsymbol{\mathcal{R}} \triangleq \boldsymbol{A}^T \left(\boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{A}^T \right)^{-1} = \boldsymbol{A}^T \left(\boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \boldsymbol{A}^T \right)^{-1}. \tag{B.5}$$

Notice that (B.4) is equivalent to the solution of the constrained GPLC formulation,[1] which is based on the Udwadia-Kalaba formulation (Kalaba & Udwadia, 1992), and is given by (5.44). To illustrate this, let $\boldsymbol{D} \triangleq \boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1/2}$. Then, we have that

$$\boldsymbol{D}^T = \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \boldsymbol{A}^T \implies \boldsymbol{A}^T = \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^T. \tag{B.6}$$

We rewrite (B.5) using (B.6) as follow

$$\boldsymbol{\mathcal{R}} = \boldsymbol{M}_{\mathrm{GP}}^{1/2} \underbrace{\boldsymbol{D}^T \left(\boldsymbol{D} \boldsymbol{D}^T \right)^{-1}}_{\boldsymbol{D}^+}. \tag{B.7}$$

Using (B.4) and (B.7), we obtain

$$\boldsymbol{M}_{\mathrm{GP}} \ddot{\boldsymbol{q}} = \boldsymbol{Q} + \boldsymbol{M}_{\mathrm{GP}}^{1/2} \boldsymbol{D}^+ \left(\boldsymbol{b} - \boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q} \right), \tag{B.8}$$

which implies

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q} + \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \left(\boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1/2} \right)^+ \left(\boldsymbol{b} - \boldsymbol{A} \boldsymbol{M}_{\mathrm{GP}}^{-1} \boldsymbol{Q} \right). \tag{B.9}$$

The final solution (B.9) is identical to the one obtained by means of the Udwadia-Kalaba formulation, which is given by (5.44), as expected.

[1] See section (5.4).