Predictive Control of Collaborative Robots in Dynamic Contexts

Nicolas Torres Alberto

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR

DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE SCIENCES PHYSIQUES

ET DE L'INGÉNIEUR

AUTOMATIQUE, PRODUCTIQUE, SIGNAL ET IMAGE, INGÉNIERIE COGNITIQUE

Par **Nicolas TORRES ALBERTO**

Predictive Control of Collaborative Robots in Dynamic Contexts

Commande Prédictive de Robots Collaboratifs dans des Contextes Dynamiques

Sous la direction de : **David DANEY**
Co-directeur : **Vincent PADOIS**

Soutenue le 23 Octobre 2023

Membres du jury:

M. Nicolas MANSARD	Directeur de Recherche	LAAS-CNRS (Toulouse)	Rapporteur
M. Richard BEAREE	Professeur	Arts et Métiers, LISPEN (Lille)	Rapporteur
M. Ouiddad LABBANI	Professeure des Universités	Xlim (Limoges)	Examinatrice
M. Sebastien RUBRECHT	Responsable d'Equipe de Recherche	IRT Jules Verne (Nantes)	Examinateur
M. David DANEY	Directeur de Recherche	Inria Université de Bordeaux	Directeur
M. Vincent PADOIS	Directeur de Recherche	Inria Université de Bordeaux	Co-directeur

Contents

List of Figures

List of Tables

Acronyms

CSC Cartesian-space Constraints 107, 115, 117, 129

DoF Degree of Freedom vi, vii, 46, 52–54, 69, 70, 82, 83, 94, 129

FK Forward Kinematics 52–54, 111

HRI Human-Robot Interaction 46, 48, 49

IK Inverse Kinematics 52, 53, 55

JSC Joint-space Constraints 105–108, 129, 133, 139, 144, 147, 149

LTI Linear Time-Invariant 60

MPC Model Predictive Control vi–viii, x, xii, 10–12, 16, 35, 39, 43, 46, 50, 58–75, 77–81, 84, 86, 88–93, 95–97, 99, 100, 104, 105, 107, 112–118, 124–130, 133–135, 137–144, 149–153, 162, 163

PD Proportional Derivative 54, 81, 83, 96, 139, 159, 161, 162

QP Quadratic Program 16, 43, 52, 54, 55, 58, 61–63, 68, 70, 71, 74, 77, 81–83, 85, 86, 88, 95, 99, 105, 107, 109, 110, 114, 115, 130, 155, 158, 160, 161, 163

SS State Space 60

TS Technical Specification 7, 10, 66

Chapter 1

Introduction

Contents

> "When one speaks of increasing power, machinery, and industry there comes up a picture of a cold, metallic sort of world in which great factories will drive away the trees, the flowers, the birds, and the green fields. And that then we shall have a world composed of metal machines and human machines. With all of that I do not agree. I think that unless we know more about machines and their use, unless we better understand the mechanical portion of life, we cannot have the time to enjoy the trees, and the birds, and the flowers, and the green fields."
>
> — Henry Ford in collaboration with Samuel Crowther, My Life and Work

Ever since the introduction of the moving assembly production line in the Ford Motor Company, manufacturers have strived to augment production by increasing automation and removing operators from production lines. However, the race for throughput in a world with limited resources has subtly grown into a chase for efficiency: not only does it affect production costs and competitiveness, but it also directly impacts the long term effects on the environment and, indirectly, the overall increase in the quality of life that the manufactured goods promise to provide.

Indeed, most heavy manufacturing (and specially in the context of highly automated car assembly) that are either unsafe for operators to do or simply impossible for humans to perform consistently have been automated throughout the last centuries. Some clear examples of this are shown in heavy car parts welding and assembly, like with the frames shown in Figure 1.1.

However, many assembly tasks often lack flexibility to evolve in time. Either because it is costly to adapt too often or because some tasks are simply too complex to automate. Yet, in the age of connectednes and efficiency, being able to adapt the manufacturing chain according to real time business data feedback becomes crucial.

This context motivates the resurgence of a new industrial trend towards human-centric manufacturing. Humans provide high levels of dexterity and flexibility; they can also quickly adapt to new tasks. Yet, they still need to overcome their intrinsic mechanical limits in terms of repetitive movements and forces. This is where the advancements in technology are required. Indeed, collaborative robotics arrive as the epitome of automation technologies to provide the missing synergistic boost in the manufacturing efficiency: the merge of humans and robots.

1.1 Industrial Context: Stellantis ↑

French have a long trajectory with car manufacturing. Since the first self-propelled vehicle built (steam-powered), attributed to the french inventor Nicolas-Joseph Cugnot in 1769 [Wikipedia, 2023], a huge automobile manufacturing industry surged, driven by demand and pushing innovation in the fabrication and assembly domains. The current work was carried out in the context of the R&D activities at one of the most emblematic french automakers groups: Group PSA (more recently Stellantis).

Stellantis is the resulting group of a merger between two international automakers: Fiat Chrysler Automobiles and Peugeot S.A. The deal was completed and the new

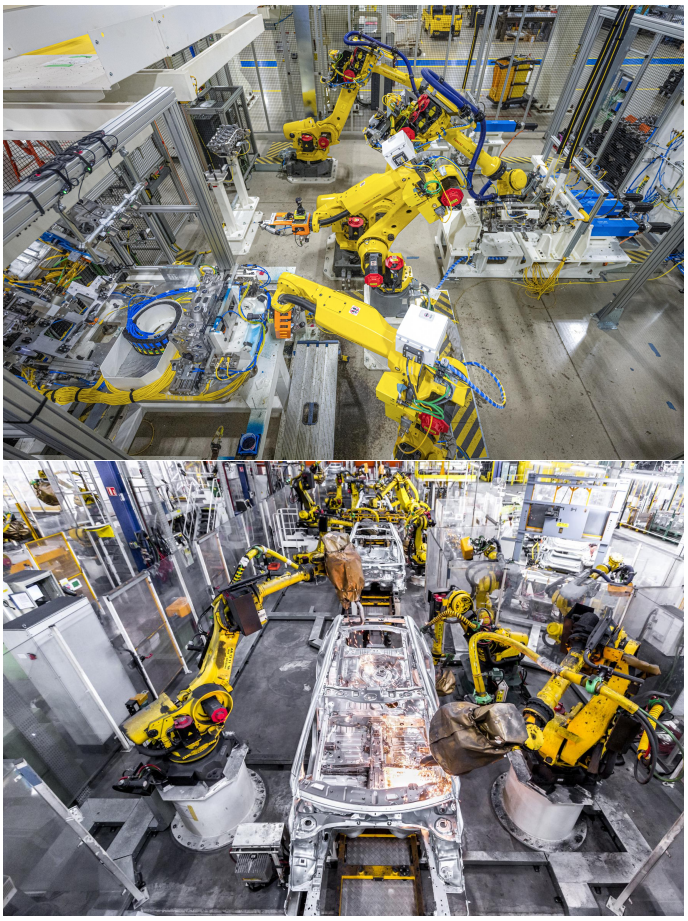


Figure 1.1: Example images of heavyly automated tasks with industrial robots. Obtained from [Stellantis Communication, 2023]; [Julien Cresp Groupe PSA, Direction de la communication, 2017a].

group began trading on the New York Stock Exchange as "STLA" in 2021 to become the 4th largest automaker in the world by volume. Stellantis designs, manufactures, and sells automobiles bearing its 16 brands (see Figure 1.2) and in the first half of 2023 repported a net revenu of €98.4 billion [Stellantis communication, 2023]. In this context, the group has set big ambitions for the coming electric vehicles transition in the years to come and has an interest to continue to invest in research and development to maintain and push its lead on the global market.

A big part of its global strategy is to innovate in manufacturing technologies. This thesis is situated in the context of Stellantis future perspectives for the "Usine du Future" (Factory of the Future), a set of long term objectives for the technological evolution of its factories [Beauville Dit Eynaud, 2020]; [Voilqué, 2020].

1.2 Agile Robotics: Human-centered Agile Manufacturing ↑

Vehicles are complex technological systems that require intricate (and often sequential) manufacturing procedures. This has provoked the car manufacturing industry to always be heavily intertwined with industrial trends, being particularly sensitive to innovations in fabrication.

Figure 1.2: Stellantis brands, after the merger in 2021.

In the current day and age, new environmental standards for sustainable development and the increasing market demand for consumer goods frame a transition towards new trends in manufacturing. To stay competitive, manufacturers have had to adopt strategies related to the flow of data in factories, which exploits the Internet of Things to increase flexibility and efficiency in the so called industrial revolution 4.0 [Ghobakhloo, 2020]. A main feature of these trends entails the capability to obtain real-time information of the factories states and effectively modulate its production capacity according to carefully crafted plannings while also taking advantage of the demand level trends. All this is possible thanks to the introduction of new technologies that have increased the connectednes and stream of sensor data, augmenting information availability and allowing for more effective decision-making strategies to transition into "agile manufacturing" [Gunasekaran, 1999].

However, once this information starts becoming the norm, a new challenge begins to appear in the horizon: manufacturing adaptation. Indeed, there is no use for information that cannot be readily exploited to take optimal decisions. These decisions entail adapting the production capacity, reasigning resources, switching production lines, customizing products according to orders, etc. Although the factory information might be promptly available, implementing all these strategies is far from simple and demands a level of flexibility that is only beginning to be possible in modern times. Factories need to be reconfigurable and this is in line with Stellantis long term objectives [Beauville Dit Eynaud, 2020].

At the same time, the long term effects on the environments of all these production capacities must also be considered. As the environmental effects induced, in part, due to the impulsive and unhindered use of resources in the last centuries start to become more apparent, industries must reevaluate their priorities. What use are the goods and services that increase the quality of life today only to permanently hinder it in the future? Indeed, factories need to start taking special consideration to the conscious use of energy and materials that enable environmentally sustainable production.

It is in this context that new priorities start to emerge to favor human-centered strategies [Demir et al., 2019]; [Lu et al., 2022]. Humans' efficiency and capacity to learn new tasks and quickly transform production lines while limiting the wasted resources becomes crucial again. They are needed to take back the center stage

Figure 1.3: Example of a robot deployed inside a cage. Obtained from [Groupe PSA Communication Metz Tremery, 2018]; [Stellantis Communication, 2022a].

of manufacturing facilities, just like before they were removed in favor of automation technologies due to lower productivity capabilities. This last drawback has not changed. However, this time around, new technologies like cobots [James E. Colgate et al., U.S. Patent 5952796A, Feb. 1996] have arrived to assist humans. These smaller robots are inherently more efficient due to their size and were specially conceived to work along humans. Cobots promise not only to compensate for the human short comings but also to augment their capabilities by working in collaboration while also adapting to the new age of efficiency, with reduced sizes.

This is the context that sets the tone for the current work: a draft proposal for strategies in which robots can start to augment humans, increase efficiency and flexibility while working in true collaboration; a proposal for cobots to start to fulfill their original promises and transition into "Agile Robotics".

1.3 Towards True Human-Robot Collaboration ↑

Even though cobots were conceived with shared workspaces in mind from the start ["Cobot architecture" 2001], their original mission has not completely crystallized yet. Simply using this type of robot does not necessarily make a task collaborative. The transition towards human-robot collaboration requires a series of adaptations in the way robots are conceived, deployed and programmed. A major

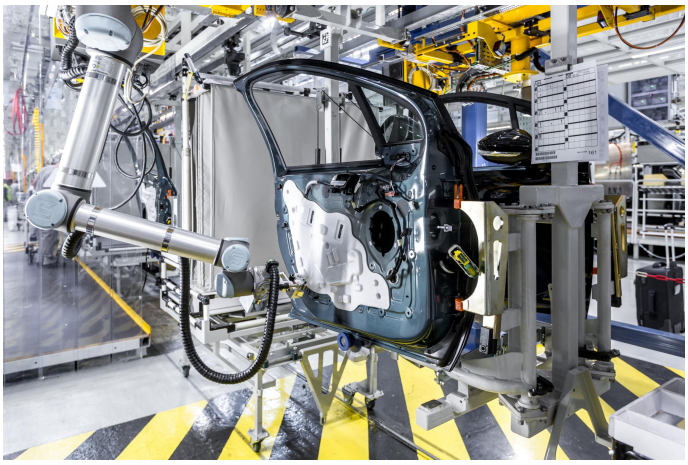


Figure 1.4: Collaborative robots are often used in non-collaborative tasks. [Julien Cresp Groupe PSA, Direction de la communication, 2017b].

concern towards these objectives is safety [Proia et al., 2022]. The very nature of humans and robots sharing a workspace has been inconceivable thus far with big industrial robots. Consequently, to ensure human safety while incorporating robots in accessible areas, the simplest strategy possible has seen the widest industry adoption: putting robots inside restricted zones limited by cages to guarantee physical separation even in accidental scenarios. This kind of situations is shown in Figure 1.3. This results in an inefficient use of the factory space that provokes impractical restrictions to the movement and design assembly lines, instead of a continuous and fluid transition between work stations.

Nevertheless, cobots still remain more practical from an ease of use and flexibility point of view, as they are easier to deploy than heavy industrial robots in general. As such, the sight of collaborative robots being deployed absent of collaborative tasks is common in factories (Figure 1.4). Yet, the fact remains that many high dexterity task still require human manipulation and offer a huge potential for more complex collaborative work cells (Figure 1.5).

While many efforts in the literature have been made to provide methods for systematic assessments of safety and other human-centered factors in collaborative workcells [Haddadin et al., 2009]; [Robla-Gómez et al., 2017], the main strategies widely adopted in the industry must comply with the established ISO standards [Fryman et al., 2012] and/or Technical Specification (TS) [Rosenstrauch et al., 2017]. While the existence of standardizations might seem to indicate a consensus on the mechanisms necessary for safety, these regulations are often considered problematic [Hanna et al., 2022]; [Fryman, 2014]; [Chemweno et al., 2020], due to the limited flexibility in collaborative scenarios.

Another challenge in the way of true human-robot collaboration is the traditional control approaches. Classically, robots are programmed to perform a task on repetition as fast as possible with little to no variability into each iteration. This entails predefined behaviors or trajectories that are not compatible with a symbiotic human-robot relation [L. Wang et al., 2019]. In order for robots to actually assist humans in the best way possible, they must be able to re-actively adapt to the non-deterministic way in which humans work: they must behave coherently with their surroundings and in tandem with what the human actually needs [Tsarouchi et al., 2016].

Figure 1.5: Many tasks benefit from human precision and have the potential to evolve towards collaborative work cells. Obtained from [Stellantis Communication, 2022b]; [julien cresp/Within, 2023].

1.3.1 Shared Workspace Requirements ↑

In order to start providing some answers to the scenario put forth this far, this section proposes a list of criteria or design principles that a robot control architecture needs to implement in order be compliant with the long term objectives of collaborative robotics work cells:

1. **Minimalism** With efficiency in mind, robots must strive to reduce their footprint as much as possible. Not only in the physical sense but rather in all aspects about their operating lifetime. However, a smaller size does in fact help achieve these objectives with multiple consequences branching from the fact that the amount of moving mass is reduced and so smaller robots influence: safety (they pose less danger in case of collisions with humans); energy usage (they consume less); they occupy less physical space and favor transportation for eventual reassignments, etc.

2. **Optimality** Again, stemming from the efficiency objectives, is the fact that robots need to make optimal use of the energy to achieve as much performance a possible. In this case, a high-performance robot is used in a broad sense as one that can achieve the required task with a task-dependent metric (for instance, a pick and place task would require robots with high lift capacity and/or speed).

3. **Reactiveness** In order to actually work in collaborative scenarios, robots are required to act according to what is happening around them. This entails complex behaviors that consider multiple aspects at the same time to continuously adapt in a way that remains safe and useful for the operator/s.

4. **Modularity** To answer the flexibility needs of modern factories, robots are required to be easily reasignable. Meaning they should be easily reconfigured and deployed into new posts without much work. Reparability, reuse and replacement capabilities should be priorities. This entails not only the robot hardware but also the design aspects of the tasks and ease of use, like the programming.

1.3.2 Understanding the Requirements ↑

The minimalism and optimality objectives go hand in hand. In fact, if one were able to correctly dimension both the task requirements (i.e. the weight to be transported, the necessary speed, the exerted forces, etc) and, at the same time, were able to match these to the robot capacities, then choosing the smallest robot that can effectively comply with the required tasks should be trivial. However, the non-linear relation between the robot actuators and the task requirements make this quite challenging [Skuric et al., 2021a]. Beyond that, these requirements are usually non formal specifications (as in not completely determined but rather defined as high level objectives to be performed). Nevertheless, the perfect architecture should strive to make an efficient use of the true robot capacities, while being able to easily program the robot to take optimal decisions.

Also, this objective naturally favors safety improvements: by using the robot in an optimal way (i.e. more efficiently), employing lighter robots becomes possible. From an energetic point of view, the fact that robots operate at lower kinetic energy levels contributes to safety [Joseph et al., 2018a].

Subsequently, the re-activeness requirement sets the challenge a bit higher because it implies that the robot must be able to take these optimal decisions while considering real-time aspects of the environment around it. The robot needs to simultaneously asses multiple criteria: dynamic events occurring around it (like operators and obstacles approaching, the state of an interactive task, etc); the true robot actuation capacities (in order to use them efficiently); its own state to avoid self-collisions; and the actual state of the task that it is currently performing.

This breaks apart from the traditional approach of programming robots to perform tasks at a later stage. It implies that robots are now required to be infused with "logical" behaviors that are capable of taking coherent decisions according to the situation. This entails a control architecture that acts in an online fashion.

And finally, the modularity and reconfigurability aspect of the control architecture are the least abstract aspects of these principles. This translates to how much a robot architecture depends on the actual robot architecture (i.e. its shape, actuators, form, capacities, etc).

In fact, to align with the general ease of use and human-centric (thus operator-centric) point of view, decoupling the tasks aspects from the robot control aspects

of the architecture should be favored. This way, if a designed task needs to be deployed with a different robot, only some parts of the architecture need to be replaced. Conversely, if a robot needs to be reused, only the "task planning" parts need to be switched.

1.3.3 Drafting a Proposal ↑

Even though these principles drive the decision making process in a general way, the scope of this work is to apply them to a control architecture. In order to set a framework that answers to these requirements, a draft solution of a control architecture is proposed.

The previous analysis outlines a solution that goes beyond the possibilities envisioned by the TS for collaborative robotics [ISO/TS-15066, 2016]. While the TS does allow some dynamic aspects like maximum velocities, safety zones and maximum energy levels, it remains a rather inflexible solution from an architectural point of view. For instance, it is not compatible with more intricate approaches to safety awareness through energy modulation [Raiola et al., 2018]; [Meguenani et al., 2015]. Furthermore, its certification is complex and leads to cobots that are used in non-collaborative scenarios or behind cages.

These problems stem from the fact that the robot is inherently controlled through inputs in its actuation space but, in order to reactively consider safety, it needs to account for external events occurring in the workspace. Decoupling these two aspects of the architecture immediately points the approach towards a modeling strategy for the "external events" and for the robot capacities: constraints. Some approaches contribute towards the safety aspects while considering these capacities in joint space [Joseph et al., 2020]. However, a method to consider workspace constraints in a generic way remains to be proposed.

Beyond that, the decoupling also stems from the inherent nature of these constraints: the workspace and the actuation space. It seems only logical that they should be considered separately. Nevertheless, the robot needs to take optimal decisions with respect to both. In order to do so, the constraints need to be formulated in the same space. This is relevant not only from an optimality point of view, but also from a feasibility concern [Rubrecht et al., 2010]; [Prete, 2018]. While some approaches do consider the real capacities and are capable of formulating workspace constraints [Kleff et al., 2021], they often do not decouple the task from the actuation (which does not align with the modularity objective).

The path taken throughout this thesis to start assembling all these aspects into a single architecture is to employ an optimization-based approach. The idea is to translate the decision making process to be performed in real time into an optimization problem that the robot controller can solve as fast as possible. Motion planning problems can be efficiently formulated as optimization problems over gliding time windows. This way the robot does not waste time planning a global motion that will not be executed (until the end) and instead only needs to consider the near future.

Such an approach employs Model Predictive Control (MPC) for motion generation in the workspace similar to what is proposed in [Gold et al., 2023]. This goes in line with the task-centered design point of view.

One final consideration to obtain a generic approach capable of full motion control: the architecture needs to be able to consider the full pose. The special emphasis on these aspects stems from the fact that oftentimes control approaches focus only on the position, as it is the most straightforward aspect of the control and the handling the orientation adds undesirable complexities. However, orientation control must also be taken into account if one strives to provide a generic solution to the problem.

While the robotics community has addressed the full pose motion formulation problem for navigation [Forster et al., 2015], the methodologies are often employed for state estimation in mobile robots [Solà et al., 2018]. However, these principles offer a convenient framework for efficient MPC approaches that properly address full pose control in serial robots, which remains to be proposed.

Finally, the proposal is to employ a decoupled cascade controller:

- A linear MPC for constrained motion generation in position and orientation, capable of taking into account both workspace level constraints along with actuation capacities.
- A constrained optimization-based controller at the actuation level.

The cascade scheme addresses the modularity requirements, while the minimalism depends on correctly formulating the actuation constraints and exploiting them, which is achieved through optimization. Indeed, at the heart of the proposal lies an MPC: it addresses the reactiveness by effectively allowing reference-less control while, at the same time, generating near optimal trajectories. Overall, this framework answers to the drafted requirements in this chapter in a generic way. This manuscript showcases the proposal for serial collaborative robots.

1.4 Structure of this manuscript ↑

The structure of this manuscript is as follows:

Chapter 2: Presents the underlying principles and modeling aspects that enable a generic approach to task-space planning with full pose control. It presents the mathematical formalism to represent the pose of a robot in the $\mathbb{SE}(3)$ manifold and how, its tangent space, can be employed to perform constrained optimizations. The content of this chapter is broadly covered by a published conference paper [Torres Alberto et al., 2022a].

Chapter 3: Presents a first approach to an ideal control architecture for collaborative robotics. It presents the underlying optimization approach based on Model Predictive Control (MPC) that applies the principles from Chapter 2 to perform constrained task-space motion generation at the kinematic level. The pertinence of the proposed control architecture is demonstrated using experiments with the Franka Emika robot in a scenario implying both adaptation of the maximum allowed velocity to comply with human presence and on-the-fly updates of a Cartesian goal pose.

Chapter 4: This chapter extends the proposals from Chapter 3 to provide an acceleration-based formulation of the MPC and robot control algorithms. The underlying principles remain the same, but the solution improves in robustness and smoothness thanks to a more efficient formulation. As an experimental validation,

the architecture is implemented to achieve a high-frequency closed-loop controller, enabling re-planning at the control rate.

Chapter 5: Finally, this chapter further extends the previous control architecture with some formalism for the modeling of robot capacities in the task-space. This is used to explicitly consider the joint-space robot capacities from within the task-space re-planning algorithm (MPC) to obtain trajectories that are coherent from the Cartesian-space constraints as well as the actual actuation capabilities. In the end, a high-frequency actuation-aware control architecture is demonstrated on the real robot by modulating the available joint capacities from tasks-space planning, to obtain a motion that stays coherent throughout the whole control loop.

Chapter 6: Offers some closing remarks and perspectives for future research.

Unpublished or On-going Related Work

- Online task-space trajectory planning using real-time estimations of robot motion capabilities [Skuric et al., 2022a]

  Antun Skuric, Nicolas Torres Alberto, Lucas Joseph, Vincent Padois, David Daney

- Model Predictive Control for robots adapting their task space motion online [Torres Alberto et al., 2023]

  Nicolas Torres Alberto, Antun Skuric, Lucas Joseph, Vincent Padois, David Daney

- Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces [Torres Alberto et al., 2022b]

  Nicolas Torres Alberto, Antun Skuric, Lucas Joseph, Vincent Padois, David Daney

- From a trapezoidal acceleration profile to a learnt time optimal control policy for robot braking [Esquerre-Pourtère et al., 2022]

  Arthur Esquerre-Pourtère, Nicolas Torres Alberto, Vincent Padois

1.5 Navigation and Notation Aspects in this Manuscript ↑

On a practical note, a small clarification about navigation is required.

To facilitate navigation throughout this manuscript, every section title features an arrow on the right that allows quickly going back to the contents table. This enables going to sections far apart by going through to the table of contents.

The general outline of the manuscript progressively builds up the methodology and concepts with each chapter. However, the chapters are intended to be mostly self-contained. As such, some sections summarize concepts from the previous ones, to refresh the concepts that are addressed.

On the same note, when definitions are revisited at later stages and it is deemed

relevant to bring it to the reader's attention, it appears on the manuscript margins like this:

$$\boldsymbol{p} \in \mathbb{R}^3 \tag{1.1}$$ revisited ↓

where the arrow on the right is a reference to the where it is revisited, right below:

$$\boldsymbol{p} = [p_x \quad p_y \quad p_z] \tag{1.2}$$ revisited ↑

This allows the reader to quickly go back and forth between both definitions. This is only used for equations and, less often, for figures. Sometimes the it indicated that the revisited element (equation or figure) is exactly the same (for convenience) and sometimes, it indicates that it is conceptually linked to a previous definition that is extended.

Furthermore, equations are referred with (equation number) (without explicitely saying it corresponds to an equation), to facilitate reading. However, when referring to tables, figures or sections, it is explicitely stated.

Finally, as can be seen in (1.1) and (1.2), matrices and vectors (like $\boldsymbol{p}$) have a bold font, while scalars use a normal font (like p_x, p_y, p_z).

Chapter 2

Linear Pose Estimation in a Time Horizon

This chapter delves into a strategy for pose estimation over small time horizon, expressed through linear algebra. The significance of this approach resides in its capacity to streamline predictive and optimization-based control in robotics, the central theme of this manuscript.

The discussion starts with the representation of rigid body poses—position and orientation—as a singular Lie group element. An alternative parameterization of these elements comes into view next, employing exponential coordinates to highlight their relation to Lie algebras. Extending this concept, the representation of rigid body movements is introduced. This progression allows for the parameterization of pose trajectories as successive group actions on the Lie manifold.

With this foundation laid, an integration strategy emerges, capable of presenting the same trajectory on the tangent space, thereby circumventing the multiplicative propagation scheme. As a final point, the integration leverages a linearization approximation rooted in a truncation of the Magnus expansion. This method approximates the exponential map derivative and showcases promising results in terms of precision and computation times.

Contents

2.1 Introduction ↑

The rise of collaborative robotics has led to a surge in the need for intricate control mechanisms [Ajoudani et al., 2018]. Over time, the focus on safety in human-robot interactions has intensified [Proia et al., 2021]. The essence of these interactions lies in the challenge for robots to operate in constantly changing environments. As a result, robots need to responsively adjust to unforeseen circumstances, all while upholding strict safety standards.

In this context of complex robotic applications, optimization-based controllers offer a convenient way to formulate tasks to be achieved under constraints. These problems can be formulated over one control time step, i.e. reactively [Escande et al., 2014], but can also lead to much more robust behaviours when considering control over receding time horizons. Within this paradigm, the use of Model Predictive Control (MPC) for planning has been widely adopted on humanoid systems to cope with the curse of dimensionality and the inherent complex equilibrium dynamics of these systems [Ibanez et al., 2019]. In contrast, these methods are relatively new in the context of collaborative robotics though its adoption is proving useful in multiple applications [Proia et al., 2022].

One of the main challenges in using a receding horizon formulation is effectively predicting the robot state evolution within a control horizon while ensuring low computational costs for regular updates. For instance, fast ways to estimate predictions can be employed on iterative algorithms [Müller, 2018] for online trajectory generation [Huber et al., 2020]. While some recent studies [Kleff et al., 2021] have managed to compute optimal solutions in a nonlinear fashion, there remains a preference for linear formulations. They are not only computationally efficient but also more straightforward when framing control problems. Simple linear algebraic expressions are more intuitive and can be easily incorporated into a Quadratic Program (QP) — a widely recognized effective framework for robotic problems. However, a comprehensive linearization approach that encompasses both 3D position and orientation[1], and that aligns well with optimization tools like QP, remains to be proposed.

The aim of this chapter is to introduce a method for estimating the pose trajectory of a rigid body within a receding horizon at every time step. This approach can be applied to any body of a serial manipulator even though the end effector is of particular significance. The objective is to reconstruct its pose trajectory based on an initial pose and a predetermined control input time horizon, here considered at the velocity level.

The proposed method relies on the Lie algebra $\mathfrak{se}(3)$ associated to the tangential space of a pose described in $\mathbb{SE}(3)$. This method (based on the Magnus Expansion (ME) [Casas et al., 2006]) leads to a linear expression of the transformation between two poses, using the logarithm map. In its truncated form, it leads to a linear propagation/integration formula which has an explicit algebraic form and takes a vector form adapted to the writing of recursive state propagation general formulas.

Structure

This chapter focuses first on the fundamental concepts of Lie groups and algebras

[1] Typically of the robot end-effector.

used as a mathematical framework for representing and manipulating the poses of rigid bodies in the context of robotics, crucial for the understanding of this work.

The chapter begins with some rigid body motion Lie groups fundamentals in Section 2.2, introducing the position, orientation and unified pose representations used throughout this work. Additionally, it provides parameterization alternatives and offers an overview on how they relate to their respective Lie group manifolds, notably $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$.

Afterwards, Section 2.3 extends these concepts by showing how they can be employed to parameterize a pose trajectory. It begins by introducing a rigid body pose path in space and discretizing it to show how discrete pose trajectories evolve in a manifold. Then it links this parameterization scheme to a Lie group differential equation to show how poses and twists relate to each other and how poses can be propagated as a function of the rigid body twist. Finally, it reformulates this discretization scheme as a power of exponentials, a concept largely studied in the literature and of central relevance to understand pose horizon estimation strategy used in the following sections.

Then, Section 2.4 presents a more generalized view of Lie group fundamentals. Many of these were already introduced for the specific Lie groups of orientations and poses in the previous sections. Nevertheless, this section offers an overview and outlines an integration scheme that exploits the relation between Lie groups and their algebras to propagate the elements in the tangent space. This finally introduces the desired linearization strategy that allows for efficient linear pose estimation in a horizon on the tangent space.

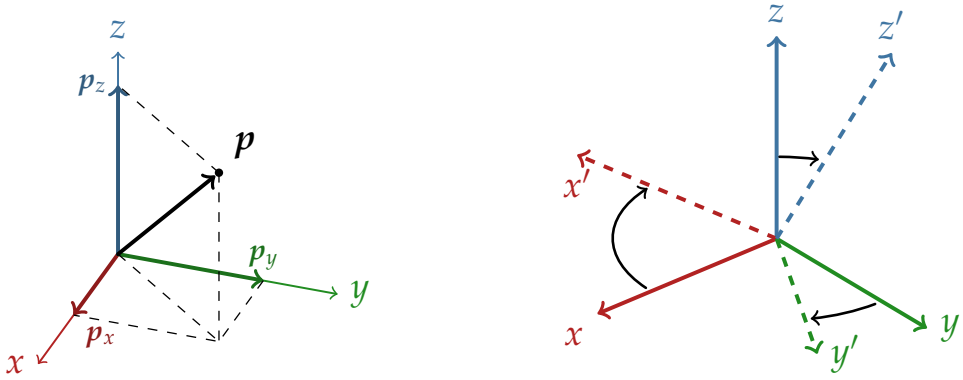
Finally, Section 2.5 formalizes the method of linear pose estimation in a horizon and connects all the concepts previously introduced to provide a formal mathematical formulation. Finally, it presents a simple kinematic simulation to showcase the results that showcases the precision and computational speed of the approach.

2.2 Rigid Body Motion in Space ↑

To solve many of the problems in robotics, one starts from conveniently describing the robot positioning in 3D space. For instance, to effectively control the movements of a robotic arm, it is important to correctly parameterize its position and orientation. This section delves into the representation of a robot rigid body pose, infinitesimal pose increments, conventional frames in robotics and how these relate to the special orthogonal group $\mathbb{SO}(3)$ and the special Euclidean group $\mathbb{SE}(3)$. Furthermore, it presents the relationship between Lie groups and their algebras to help understand their exponential coordinates representations.

Robots can be modeled as an array of bodies that do not deform or change shape (rigid body) and whose relation is fixed. In other words, they are represented as a collection of interconnected rigid bodies whose relative distances and orientations remain constant (unless designed otherwise, like with actuated joints).

The effectiveness of the robot in carrying out tasks depends on both the nature of the task and, more importantly, the level of control over these variables. It is fundamentally important because it directly affects the incidence angle and distance at



(a) Position representation in 3D space.

(b) Orientation representation in 3D space.

Figure 2.1: A simple way to represent 3D positions and orientations in space is through vectors.

which these tools interact with objects. This is why one particularly relevant body of a robotic serial arm is the end-effector, where the tools specific for the task to be performed are mounted.

The robot efficiency in executing tasks is influenced by the nature of the task and, more importantly, the control over the associated variables like the incidence angle and distance to the objects that the robot interacts with. This is why the end-effector of a robotic serial arm is often the body studied in many applications (sometimes implicitly) — it's where the tools are attached.

Many of the formulations in this work implicitly refer to motions of the end effector, but the concepts can be applied to any of the robot bodies.

The following sections introduce a way to parameterize orientations and then how to combine them along with position for a unified pose representation. They offer an overview on the subject to remind the reader of some concepts and clarify some notations. An interested reader may find more comprehensive introductions on these subjects in Chapter 3 of [Lynch et al., 2017], Section 2.2 of [Siciliano et al., 2008], Chapter 2 of [Craig, 1986]; [Murray et al., 1994].

2.2.1 Orientation of a floating body in space ↑

Much of this work is based on a specific representation method for orientations. This section focuses on that specific method and offers an overview of why it is convenient. For a more exhaustive introduction of the different representation alternatives, refer to Section 2.2 of [Siciliano et al., 2008].

The position of a floating body in space can be represented with a 3-vector $\boldsymbol{p} \in \mathbb{R}^3$, where $\boldsymbol{p} = [p_x \; p_y \; p_z]^T$ designates the displacement from the origin of the global frame to arrive at some point $\boldsymbol{p}$, as shown in Figure 2.1(a).

Representing three-dimensional orientations in space requires a bit more work than positions. In general, orientations in space can be visualized as a floating XYZ frame, as shown in Figure 2.1(b). Just like with positions, they can be interpreted as the rotational displacement required for each of its axes to reach a certain angle with respect to some set of "base" axes frame, while sharing the same origin point. As such, one

needs at least 3 parameters to uniquely identify them. There exist multiple ways to parameterize orientations that have different intrinsic advantages and drawbacks that ultimately determines which one is better suited for each application.

REMARK. Commonly used frames in rigid body motions
This work exploits spatial algebra to express rigid body motion quantities in a concise manner. All these quantities require a frame in which they are expressed. This means that the same instantaneous body motion (analogue to a velocity) can be represented, for instance in a local frame and thus be defined with respect to a moving body or in any other arbitrary inertial frame. There are two commonly used frames in the literature when working with this kind of physical quantities: they are normally defined with respect to the body in question or to some arbitrarily chosen universal world frame, as depicted in Figure 3.2:

- **World**: A global, universal or world frame, often referred to as frame S or W.
- **Local**: Also referred to as local or body frame and designated as L or B, represents a frame placed and aligned with the designated body. This corresponds to a frame attached to the moving body.

Both of these frames are used extensively in this manuscript. Their relevance becomes clear once some Lie group fundamentals are introduced, showing that their formulations change depending on the employed frame.

For instance, Euler angles offer easy interpretability and require only 3 parameters but suffer from some singularities like the gimbal lock. To avoid these issues, quaternions offer a good alternative even though they require some deeper understanding to use and their representation is not as compact, as they use 4 parameters. However, they can also be used for composing successive re-orientations, which is a common problem in robotics, where the orientations of bodies are often defined with respect to other bodies or frames in the environment [Brockett, 1984].

Yet another way of representing orientations is rotation matrices, often denoted as $\boldsymbol{R}$. These are 3×3 matrices with some characteristics that make them specially convenient: they can be viewed as a transformation matrix capable of performing a rotation in Euclidean space.

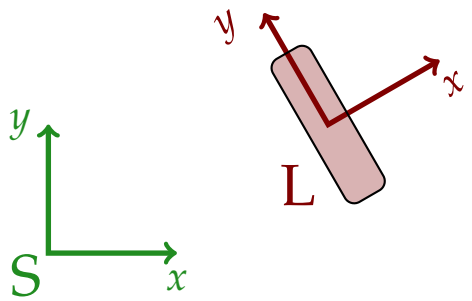


Figure 2.2: Depiction of two conventional frames used to describe rigid body motions in space: S is the world or universal frame and L is a local or body frame and corresponds to a moving frame placed on the concerned body.

To construct a rotation matrix, one can imagine a frame denoted by 3 orthonormal vectors expressed a global frame $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^3$ representing 3 points attached to a body. Re-orienting this body results in 3 new coordinates for these points $\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}' \in \mathbb{R}^3$ that can be used to construct the rotation matrix as $\boldsymbol{R} = [\boldsymbol{x}' \ \boldsymbol{y}' \ \boldsymbol{z}']$. This can be interpreted visually in Figure 2.1(b). In particular, there is a set of matrices belonging to the special orthogonal group $\mathbb{SO}(3)$ such that:

$$\boldsymbol{R}^T = \boldsymbol{R}^{-1} \qquad \det \boldsymbol{R} = 1 \qquad \boldsymbol{R} \in \mathbb{R}^{3\times 3} \tag{2.1}$$
$$\Rightarrow \boldsymbol{R} \in \mathbb{SO}(3)$$

which can perform rotations on the Euclidean space without modifying distances: given some vector $\boldsymbol{e} \in \mathbb{R}^3$, it can be rotated to obtain a new vector with the same length $\boldsymbol{e}' \in \mathbb{R}^3$ as $\boldsymbol{e}' = \boldsymbol{R}\boldsymbol{e}$.

In terms of formulation, rotation matrices are advantageous due to their ability to rotate Cartesian vectors through a matrix multiplication, unlike quaternions, which is a commonly used feature in robotics. Rotation matrices also provide a straightforward method for composing rotations. However, compared to quaternions, they are less efficient in representing orientations as they require 9 parameters instead of just 4.

The $\mathbb{SO}(3)$ group constitutes a Lie group, i.e. a group that is also a differentiable manifold that *locally* resembles the Euclidean space. This kind of groups where first introduced by Sophus Lie in 1888 [Merker, 2010] and have been extensively studied in the literature. They are of special interest for the study of motions in space because they provide a natural framework for their concepts of continuous transformations.

This work uses this kind of matrices to represent orientations and rotation operations. The following sections present other special properties and their relation to the special Euclidean group.

REMARK. Group Definition A group is a non-empty set $\mathcal{G}$ with a binary operation denoted $\cdot$ such that any combination of two elements of $\mathcal{G}$ yields a new one: $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{c}$ where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathcal{G}$. It must also respect the *group axioms* [Lang, 2002] such that:

- Associativity: for $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathcal{G}$, then: $(\boldsymbol{a} \cdot \boldsymbol{b}) \cdot \boldsymbol{c} = \boldsymbol{a} \cdot (\boldsymbol{b} \cdot \boldsymbol{c})$.
- Identity element: there exists an identity element $\boldsymbol{e} \in \mathcal{G}$ such that for $\boldsymbol{a} \in \mathcal{G}$: $\boldsymbol{e} \cdot \boldsymbol{a} = \boldsymbol{a} \cdot \boldsymbol{e} = \boldsymbol{a}$.
- Inverse element: for each element $\boldsymbol{a} \in \mathcal{G}$ there exists $\boldsymbol{b} \in \mathcal{G}$ such that: $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a} = \boldsymbol{e}$.

2.2.1.1 Some properties of rotation matrices ↑

Notice that rotation matrices can be interpreted as the required "re-orientation" transformation to be applied over a base frame to achieve some orientation (see Figure 2.1(b)). For instance, $\boldsymbol{R}_{S1}$ defines the rotation required to go from some universal frame S to some body 1 orientation. Just the same, the rotation necesssary to go from the orientation of body 1 to the frame S is the inverse: $\boldsymbol{R}_{1S} = \boldsymbol{R}_{S1}^{-1}$, which, from the properties of $\mathbb{SO}(3)$ in (2.1), corresponds to its transpose: $\boldsymbol{R}_{S1}^{-1} = \boldsymbol{R}_{S1}^T$.

Additionally, it is also possible to obtain the rotation of a body 1 with respect to some other body 2 (i.e. $\boldsymbol{R}_{12}$), as long as their own orientations with respect to some other frame is known. For example, given $\boldsymbol{R}_{S1}$ and $\boldsymbol{R}_{S2}$:

$$\boldsymbol{R}_{12} = \boldsymbol{R}_{S1}^{-1}\boldsymbol{R}_{S2} = \boldsymbol{R}_{1\not{S}}\boldsymbol{R}_{\not{S}2} \tag{2.2}$$

In robotics, every quantity is frequently defined with respect to some universal frame. When that is the case, the notation can be simplified such that: $\boldsymbol{R}_2 = \boldsymbol{R}_{S2}$. In the same way, its inverse refers to $\boldsymbol{R}_2^{-1} = \boldsymbol{R}_{2S}$. In contrast, if a rotation was defined with respect to some other body, then it needs to be explicitely indicated, for example: $\boldsymbol{R}_{12}$.

A common challenge in robotics is determining the orientation of a body connected to other bodies by utilizing their relative rotations. Take, for instance, determining the orientation of a frame attached to a robot joint. For example, if joint 2 is connected to joint 1 through a link, their rotation matrices can be combined through multiplication, resulting in a sequence of rotations:

$$\boldsymbol{R}_2 = \boldsymbol{R}_{S1}\boldsymbol{R}_{12} \qquad \boldsymbol{R}_{S1}, \boldsymbol{R}_{12}, \boldsymbol{R}_2 \in \mathbb{SO}(3) \tag{2.3}$$

Later sections link these concepts to the rate of change in orientations. Then, these notions are extended for poses to obtain a unified representation of a position and orientation.

REMARK. Commutativity of $\mathbb{SO}(n)$ A special characteristic of the $\mathbb{SO}(n)$ for $n \leq 2$ (which includes orientations in 2D) is that they are abelian groups, meaning they are commutative:

$$\boldsymbol{R}_a\boldsymbol{R}_b = \boldsymbol{R}_b\boldsymbol{R}_a \qquad \boldsymbol{R}_a, \boldsymbol{R}_b \in \mathbb{SO}(n) \quad \text{for } n \leq 2 \tag{2.4}$$

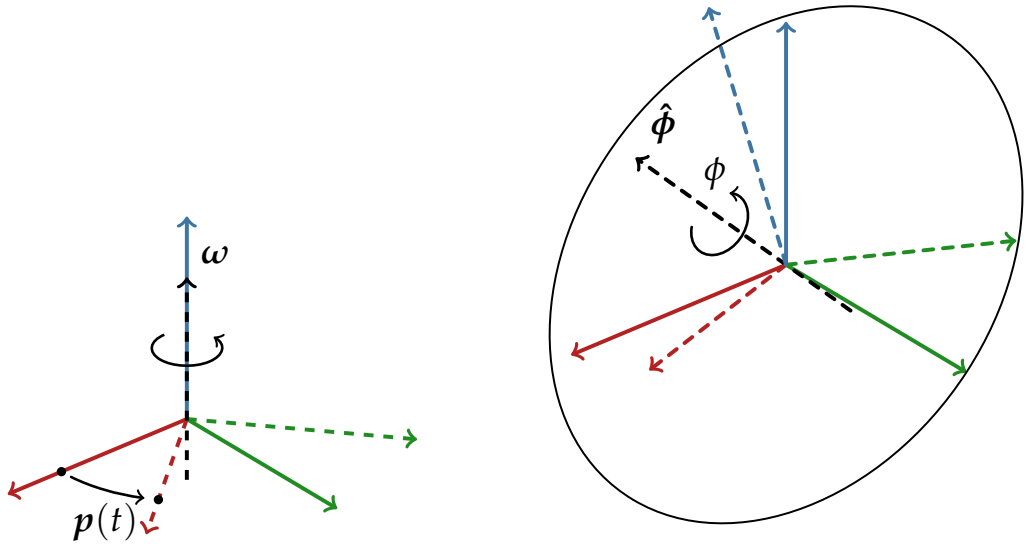
However, this is not the case for the more general special orthogonal groups $\mathbb{SO}(n)$ for $n > 2$ that are non-commutative. So the order in which the rotations are applied has an effect on the result.

2.2.1.2 Axis-angle representation ↑

One implication of Euler's rotation theorem [Euler, 1765] is that any displacement of a rigid body in 3D space such that a point in it remains fixed can be described by a single rotation about an axis passing through that point. In fact, there exists an exponential operation capable of mapping this axis-angle representation to the space of rotation matrices.

This method has been extensively studied in the literature. Some of the first mentions that make the connection with a spatial motion date back to 1840 by Olinde Rodrigues [Rodrigues, 1840].

Figure 2.3(a) shows a rotation example around an axis aligned with the Z axis. Alternatively, more complex rotations can also be represented for an arbitrary rotating axis, as seen in Figure 2.3(b).



(a) An example re-orientation where the rotation axis is coincident with the z axis.

(b) A more complex example of a 3D rotation with an arbitrary rotation axis.

Figure 2.3: Rotations can be represented in an axis-angle form comprised of a rotation angle around a rotation axis (vector). Figure 2.3(a) shows the movement of a point attached to a rigid body moving at an angular velocity *rotvel*. Figure 2.3(b) depicts the exponential coordinates $\boldsymbol{\phi}$ of a rotation.

The axis-angle representation allows describing a rotation by an angle ϕ about a unitary axis denoted $\hat{\boldsymbol{\phi}} \in \mathbb{R}^3$. These parameters can be used to form a single vector $\boldsymbol{\phi} = \phi\hat{\boldsymbol{\phi}}$ (note that $\phi = ||\boldsymbol{\phi}||$) from which an equivalent rotation matrix $\boldsymbol{R}$ can be deducted by employing the Rodrigues' formula:

$$\boldsymbol{R}(\phi, \boldsymbol{\phi}) = \mathbf{I}_3 + \sin\phi\boldsymbol{K} + (1 - \cos\phi)\boldsymbol{K}^2 \qquad \boldsymbol{K} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \tag{2.5}$$
$$\boldsymbol{\phi} = [\phi_1 \quad \phi_2 \quad \phi_3]^T$$

While not crucial to this chapter, this formula presents a closed-form for the exponential to map 3-vectors to rotation matrices [Gogu et al., 1996]. The link with the exponential is further analyzed in the following sections.

2.2.1.3 Angular velocity ↑

The angular velocity is the a rate of change in the orientation. It can be related to the time derivative of a rotation matrix. This result has recently been revisited in [Hamano, 2013]; [Zhao, 2016].

The physical notion can be simply constructed from the motion of a point attached to a body. In particularly, Figure 2.3(a) depicts a point attached to a rigid body frame moving at an angular velocity denoted $\boldsymbol{\omega} \in \mathbb{R}^3$ (a respresentation introduced in the previous section). If $\boldsymbol{p}(t) \in \mathbb{R}^3$ denotes the position of this point in time, then $\dot{\boldsymbol{p}}(t) \in \mathbb{R}^3$ denotes its time derivative, i.e. its velocity, then:

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{\omega}(t) \times \boldsymbol{p}(t) \tag{2.6}$$

where $\cdot \times \cdot$ denotes the vector cross-product and $\boldsymbol{\omega}(t)$ describes a time-varying angular velocity. The vector cross-product can also be expressed as the product of a skew-symmetric matrix and a vector:

$$\dot{\boldsymbol{p}} = \boldsymbol{\omega}^{\wedge} \boldsymbol{p} \tag{2.7}$$

where the time dependency was dropped for a clearer notation. Here, $\boldsymbol{\omega}^{\wedge}$ denotes the Hat operation, described in (2.8).

REMARK. The Hat/Vee map for $\mathfrak{so}(3)$ The Hat map allows re-parameterizing a 3-vector as a 3×3 skew-symmetric matrix. The inverse operation is known as the Vee map. One can go from one to the other through the reciprocal operations:

$$\begin{aligned} \text{Hat:} \quad \cdot^{\wedge} &: \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times 3} \\ \text{Vee:} \quad \cdot^{\vee} &: \mathbb{R}^{3\times 3} \rightarrow \mathbb{R}^3 \end{aligned} \tag{2.8}$$

$$\boldsymbol{\omega}^{\wedge} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad \begin{matrix} \cdot^{\vee} \\ \rightleftharpoons \\ \cdot^{\wedge} \end{matrix} \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.9}$$

The Hat map enables the vector cross-product to be expressed as the product of a skew-symmetric matrix and a vector. For instance, given 2 vectors $\boldsymbol{\omega}, \boldsymbol{p} \in \mathbb{R}^3$:

$$\boldsymbol{\omega}^{\wedge} \boldsymbol{p} = \boldsymbol{\omega} \times \boldsymbol{p} \tag{2.10}$$

This same process can be applied for points attached to each of the body frame axes $\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t) \in \mathbb{R}^3$ moving at the same angular velocity, yielding three velocity vectors: $\dot{\boldsymbol{x}}(t), \dot{\boldsymbol{y}}(t), \dot{\boldsymbol{z}}(t) \in \mathbb{R}^3$ that can be used to construct the angular displacement rate as a rotation matrix time derivative $\dot{\boldsymbol{R}}$:

$$\dot{\boldsymbol{R}}(t) = \begin{bmatrix} \dot{\boldsymbol{x}}(t) & \dot{\boldsymbol{y}}(t) & \dot{\boldsymbol{z}}(t) \end{bmatrix} \tag{2.11}$$

$$= \begin{bmatrix} \boldsymbol{\omega}(t) \times \boldsymbol{x}(t) & \boldsymbol{\omega}(t) \times \boldsymbol{y}(t) & \boldsymbol{\omega}(t) \times \boldsymbol{z}(t) \end{bmatrix} \tag{2.12}$$

which can in turn be factorized as:

$$\dot{\boldsymbol{R}}(t) = \boldsymbol{W}(t) \begin{bmatrix} \boldsymbol{x}(t) & \boldsymbol{y}(t) & \boldsymbol{z}(t) \end{bmatrix} \qquad \boldsymbol{W} = \boldsymbol{\omega}^{\wedge} \tag{2.13}$$

where $\boldsymbol{\omega}^{\wedge}$ denotes the skew-symmetric matrix of the angular velocity $\boldsymbol{\omega}$.

Given both $\boldsymbol{R}, \dot{\boldsymbol{R}}$ belong to $\mathbb{SO}(3)$ (and so they respect the properties presented in (2.1)), then it can be shown that $\boldsymbol{W}$ belongs in its tangent space $\mathfrak{so}(3)$, composed of elements that represent infinitesimal increments of the $\mathbb{SO}(3)$ manifold. In fact, all the elements of $\mathfrak{so}(3)$ share a rather relevant property: they are all skew-symmetric and this enables exploiting the bijective Hat/Vee operators to parameterize the

matricial elements of $\mathfrak{so}(3)$ as vectors in $\mathbb{R}^3$. Such that:

$$\begin{array}{rlcccc} \text{Hat} & \boldsymbol{\omega}^{\wedge} & : & \mathbb{R}^3 & \rightarrow & \mathfrak{so}(3) \\ \text{Vee} & \boldsymbol{W}^{\vee} & : & \mathfrak{so}(3) & \rightarrow & \mathbb{R}^3 \end{array} \tag{2.14}$$

Furthermore, $\mathfrak{so}(3)$ constitutes the Lie algebra of the special orthogonal group $\mathbb{SO}(3)$, a vector space together with an operation known as the Lie bracket $[\cdot, \cdot]$ satisfying the Jacobi identity [Solà et al., 2018].

The following section connects the algebra and the group elements through a map, allowing for a more compressed representations of the rotation matrices.

2.2.1.4 Axis-angle representation as the rotation Matrix exponential parameterization ↑

Section 2.2.1.2 briefly introduces the Rodrigues formula that maps an axis-angle representation into a rotation matrix. More recent works [Grassia, 1998] (1994), [Chirikjian, 2005] (Chapter 2, 2005) show how to exploit some properties of the rotation matrices that allows expressing them as the exponential of skew-symmetric matrices. Nowadays the method has been further generalized to enable parameterizing roto-translational transformations (i.e. pose matrices) from $\mathbb{SE}(3)$. Some detailed overviews can be found in Sections 3.2.3 and 3.3.3 of [Lynch et al., 2017], Section 2.2.5 of [Siciliano et al., 2008] and Sections 2.2 and 3.2 of [Murray et al., 1994].

This section provides an overview of the method without an extensive development. It presents the method for orientations as it is simpler but it is extended by analogy for poses in later sections.

To understand how an axis-angle representation $\boldsymbol{\phi}$ can be used to parameterize $\boldsymbol{R}$ through the exponential, one needs to first understand that (2.13) can also be interpreted as a linear Lie group differential equation:

$$\dot{\boldsymbol{R}}(t) = \boldsymbol{W}(t)\boldsymbol{R}(t) \tag{2.15}$$

In the most general cases, due to the non-commutative nature of the Lie groups and the fact that $\boldsymbol{W}$ is time-varying, this differential equation does not admit a simple solution. For the time being, assuming a constant $\boldsymbol{W}$, it is simplified to:

$$\dot{\boldsymbol{R}}(t) = \boldsymbol{W}\boldsymbol{R}(t) \tag{2.16}$$

Thanks to some well-known results [Hairer et al., 2006] (see Section 3.2.3.1 in [Lynch et al., 2017]), this linear differential equation can be solved through:

$$\boldsymbol{R}(t) = e^{\boldsymbol{W}t}\boldsymbol{R}_0 \tag{2.17}$$

where the exponential refers to the matrix exponential and $\boldsymbol{R}_0 = \boldsymbol{R}(0)$ corresponds to the initial orientation. Furthermore, by assuming that R_0 is in fact the identity element of $\mathbb{SO}(3)$ and that $t = 1$, the equation becomes:

$$\boldsymbol{R} = e^{\boldsymbol{W}} \qquad \boldsymbol{W} = \boldsymbol{\omega}^{\wedge} \tag{2.18}$$

showing that it is possible to find an element $\boldsymbol{W} \in \mathfrak{so}(3)$ such that its exponential represents an element $\boldsymbol{R} \in \mathbb{SO}(3)$.

Remembering that elements of $\mathfrak{so}(3)$ can be mapped to $\mathbb{R}^3$ through the Vee map (see (2.8)), then $\boldsymbol{W} = \boldsymbol{\omega}^\wedge$ can also be interpreted as the axis-angle representation of a rotational movement equivalent to $\boldsymbol{R}$. In fact, $\boldsymbol{R}$ is the result of integrating an angular velocity $||\boldsymbol{\omega}||$ around an axis $\hat{\boldsymbol{\omega}}$ for a unit of time. This vector $\boldsymbol{\omega}$ is referred to as the *exponential coordinates* of $\boldsymbol{R}$.

Finally, (2.18) shows that rotation matrices can be parameterized through the exponentiation of a skew-symmetric matrix constructed from a 3-vector axis-angle representation. One could then wonder if it is also possible to retrieve this vector from a rotation matrix. And indeed it is possible via the reciprocal operation, i.e. the matrix logarithm map:

$$\begin{array}{rcccc} \exp \boldsymbol{\omega} & : & \mathfrak{so}(3) & \rightarrow & \mathbb{SO}(3) \\ \log \boldsymbol{R} & : & \mathbb{SO}(3) & \rightarrow & \mathfrak{so}(3) \end{array} \tag{2.19}$$

Note that $\exp \boldsymbol{C} = e^{\boldsymbol{C}}$ are equivalent can used indifferently. The choice stems from either clarity or convenience. Also, the exponential and logarithm maps denoted here are defined for matrices but, as defined in (2.8), for the Lie algebra elements employed in this work, there is a bijective map between the vector and matrix forms (see (2.8)). By abuse of notation, $\boldsymbol{W} \in \mathfrak{so}(3)$ may be referred to as an "angular velocity" because $\boldsymbol{W} = \boldsymbol{w}^\wedge$ and the following forms are treated as equivalent:

$$\exp \boldsymbol{\omega} \equiv \exp \boldsymbol{W} \qquad \log \boldsymbol{R} = \boldsymbol{w} \equiv \log \boldsymbol{R} = \boldsymbol{W} \tag{2.20}$$

This map is essential for understanding some integration techniques employed throughout this manuscript. These techniques simplify the problem of integrating rigid body dynamics when operating with group elements of a manifold like the orientation. For instance, they can be used to propagate the effects of an angular velocity on the orientation in its tangent space efficiently and later retrieve the result. This is explained in later sections of this chapter.

REMARK. The Rodrigues Parameters Olinde Rodrigues showed a closed-form to solve $e^{\boldsymbol{\omega}^\wedge}$ from its vector form $\boldsymbol{\omega}$ [Rodrigues, 1840], see (2.5).

2.2.2 Pose of a floating body in space ↑

Previous sections introduce a way to represent the position and orientation of a body in space. This section constructs a way to combine them into a unified representation in a single element.

The pose of a rigid body in the robot can be represented as a floating frame in space (see Figure 2.4). This frame is uniquely identified by a position $\boldsymbol{p}$ and an orientation $\boldsymbol{R}$. Both elements can be assembled into a 4×4 homogeneous transformation

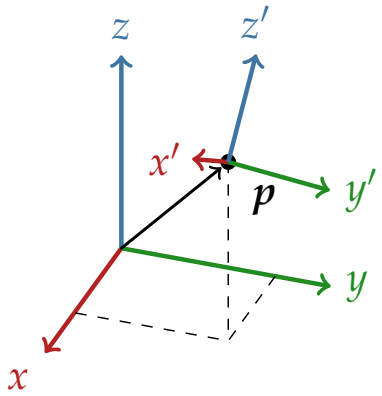


Figure 2.4: Representation of a pose in 3D space.

matrix that goes from the universal frame to the body frame B:

$$\boldsymbol{T}_B = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.21}$$

revisited ↓

$$\text{with:} \qquad \boldsymbol{p} \in \mathbb{R}^3 \qquad \boldsymbol{R} \in \mathbb{SO}(3) \qquad \boldsymbol{T}_B \in \mathbb{R}^{4\times 4}$$

Just as with rotation matrices, the notation is simplified when the transformation is expressed with respect to the universal frame S. This means that for the body pose defined above: $\boldsymbol{T}_B = \boldsymbol{T}_{SB}$.

In fact, the set of all homogeneous transformation matrices is called the special Euclidean group $\mathbb{SE}(3)$ and any element $\mathcal{X} \in \mathbb{SE}(3)$ can be denoted as: $\mathcal{X} = (\boldsymbol{R}, \boldsymbol{p})$.

2.2.2.1 Pose as a frame displacement ↑

Just as with the rotation matrices (see Section 2.2.1.1), the elements of $\mathbb{SE}(3)$ can be interpreted as the roto-translational displacement between two frames. And just as in (2.3), they can be used to construct a body pose from their relative homogeneous transformation matrices:

$$\boldsymbol{T}_2 = \boldsymbol{T}_{S1}\boldsymbol{T}_{12} \qquad \boldsymbol{T}_{S1}, \boldsymbol{T}_{12}, \boldsymbol{T}_2 \in \mathbb{SE}(3) \tag{2.22}$$

To obtain these relative transformations between two bodies 1 and 2 (i.e. $\boldsymbol{T}_{12}$), one can use an expression similar to (2.2) (as long as their own poses are known with respect to some common frame), as this:

$$\boldsymbol{T}_{12} = \boldsymbol{T}_{S1}^{-1}\boldsymbol{T}_{S2} \tag{2.23}$$

REMARK. Commutativity of $\mathbb{SE}(3)$ Just like $\mathbb{SO}(3)$, the $\mathbb{SE}(3)$ group is non-commutative (see (2.4)).

2.2.2.2 Pose derivative ↑

Similarly to how the angular velocity relates to a rate of change in the orientation, linking the orientation manifold $\mathbb{SO}(3)$ to its tangent space $\mathfrak{so}(3)$, there is a way to

describe the rate of change in a pose. This rate lies in the tangent space of the special Euclidean group $\mathbb{SE}(3)$, known as the algebra $\mathfrak{se}(3)$. Elements of this algebra can be referred to as spatial twists $\boldsymbol{\nu} \in \mathfrak{se}(3)$ (see Section 3.3 of [Lynch et al., 2017]) and are composed of an angular component $\boldsymbol{\omega} \in \mathfrak{so}(3)$ (a rotational velocity, that belongs to the Lie algebra of $\mathbb{SO}(3)$) and a linear component $\boldsymbol{v} \in \mathbb{R}^3$ (analog to the linear velocity).

To see how $\boldsymbol{\nu}$ relates to the pose time derivative $\dot{\mathcal{X}}$, it is possible to extend (2.15) for pose elements in $\mathbb{SE}(3)$:

$$\dot{\mathcal{X}}(t) = \mathcal{H}(t)\mathcal{X}(t) \tag{2.24}$$

revisited ↓

such that $\dot{\mathcal{X}}, \mathcal{X} \in \mathbb{SE}(3)$ are elements of the pose manifold and $\mathcal{H} \in \mathfrak{se}(3)$ belongs to its tangent space. This forms a linear time-varying Lie group differential equation that is non trivial to solve.

Note that $\mathcal{H}$ can then be re-parameterized as a vector through the Vee map with (2.25). The relevance of these identities becomes clearer in the following sections. Notably, the following shows how they are used for an alternative parameterization of the pose element matrices.

REMARK. The Hat/Vee map for $\mathbb{SE}(3)$ The Hat/Vee map definitions for the orientations in (2.8) can be extended in a similar way to parameterize the twists either as a matrix or as a 6-vector. This allows going back and forth between $\mathbb{R}^6$ and $\mathbb{R}^{4\times4}$ as:

$$\boldsymbol{\nu}^{\wedge} = \begin{bmatrix} \boldsymbol{\omega}^{\wedge} & \boldsymbol{v} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix} \quad \begin{matrix} \overset{\vee}{\rightarrow} \\ \underset{\wedge}{\leftarrow} \end{matrix} \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.25}$$

$$\text{with:} \qquad \boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \qquad \boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.26}$$

where $\boldsymbol{\nu}^{\wedge} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{\nu} \in \mathbb{R}^6$.

2.2.2.3 Screw motion as the pose matrix exponential parameterization ↑

Thanks to the Mozzy-Chasles theorem [Chasles, 1830]; [Davidson et al., 2004], the concepts from Euler's Rotation theorem can be extended to the more general roto-translational motions. The theorem states that any rigid body motion can be realized by a rotation about an axis along with a translation parallel to that same axis.

This means that the same strategy employed in Section 2.2.1.4 for the parameterization of rotation matrices as axis-angle vectors through the exponential map can be applied for the pose matrices. This time though, the parameterization requires 6 parameters that can be interpreted as two 3-vectors: one associated with the rotation axis and another linked to the linear displacement (see Section 3.3.3 of [Lynch et al., 2017]).

To understand this method, on can start by revisiting (2.24), assuming $\mathcal{H}$ remains constant to simplify the expression

$$\dot{\mathcal{X}}(t) = \mathcal{H}\mathcal{X}(t) \tag{2.27}$$

and utilize some known ordinary linear differential solutions [Hairer et al., 2006] (see Section 3.2.3.1 in [Lynch et al., 2017]):

$$\mathcal{X}(t) = e^{\mathcal{H}t}\mathcal{X}_0 \tag{2.28}$$

where the exponential refers to the matrix operation and $\mathcal{X}_0 = \mathcal{X}(0)$ corresponds to the initial pose. Furthermore, without loss of generality, it can be assumed that the initial pose coincides with the identity element and that for t=1:

$$\mathcal{X} = e^{\mathcal{H}} \qquad \mathcal{H} = \boldsymbol{\xi}^{\wedge} \tag{2.29}$$

which shows that it is possible to map elements from the $\mathbb{SE}(3)$ manifold to elements of its tangent space $\mathfrak{se}(3)$ through the exponential operation. This vector $\boldsymbol{\xi}$ is referred to as the *exponential coordinates* of $\mathcal{X}$. Furthermore, the reciprocal operation (the logarithm map) allows going back and forth between both representations:

$$\begin{array}{rcccc} \exp\mathcal{H} & : & \mathfrak{se}(3) & \rightarrow & \mathbb{SE}(3) \\ \log\mathcal{X} & : & \mathbb{SE}(3) & \rightarrow & \mathfrak{se}(3) \end{array} \tag{2.30}$$

For simplicity, the same notation showed in (2.20) is extended to (2.30). This means that $\mathcal{H} \in \mathfrak{se}(3)$ may be referred to as a "spatial twist" because $\mathcal{H} = \boldsymbol{\xi}^{\wedge}$ and the following expressions are treated as equivalent:

$$\exp\boldsymbol{\xi} \equiv \exp\mathcal{H} \qquad \log\mathcal{X} = \boldsymbol{\xi} \equiv \log\mathcal{X} = \mathcal{H} \tag{2.31}$$

The following sections exploit the representations introduced previously to describe rigid body trajectories in space as time-varying pose elements in time.

2.3 Rigid Body Pose Trajectory in Space ↑

The concepts introduced in the previous sections allow representing static positions and orientations in space and properly parameterizing them in a vector form. This section focuses on how to use these principles to parameterize a roto-translational motion in space, which is the core problem of this work. The objective is to describe the time-varying pose elements that allow the parameterization of pose trajectories.

REMARK. Trajectory vs Path Because there is no associated timing to each step, this is referred as a *path*. Otherwise, it is referred as a *trajectory*. Such case is treated in subsequent sections.

2.3.1 Pose path in space ↑

This section deals with the problem of discretizing a position and orientation (pose) path as a function of some initial pose and the successive increments between each

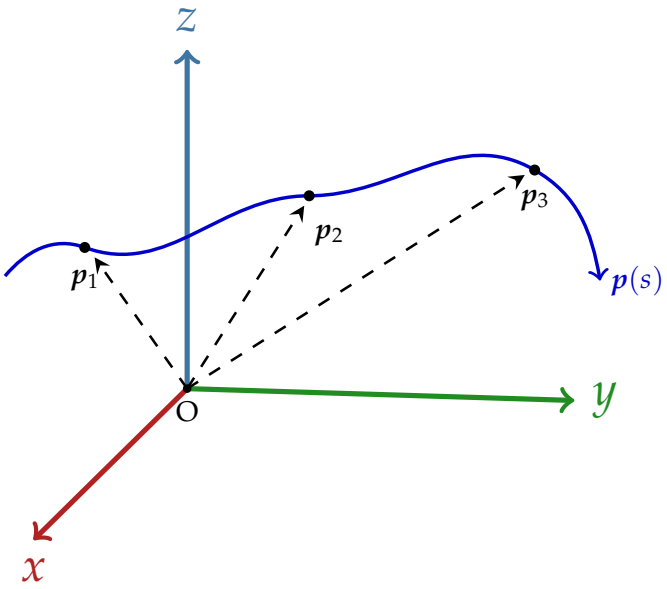


Figure 2.5: Representation of a position path in 3D space.

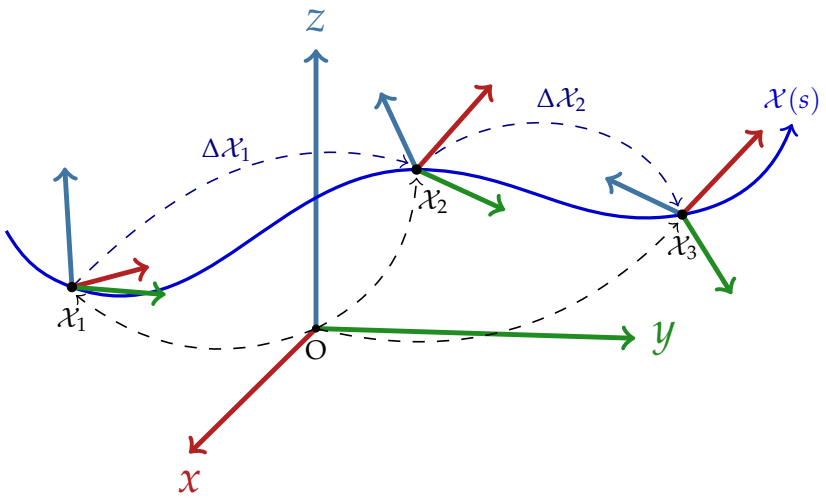


Figure 2.6: Representation of a pose path in 3D space.

step. To do so, it first focuses on a position path that can be introduced more intuitively and then adds the orientations to build a full pose path.

Let us assume that it is possible to define a position path in space with respect to some parameter s as $\boldsymbol{p}(s)$, where s goes from 0 (at the start of the path) to 1 (at the end). Each point in this path (that defines a position) can then be parameterized as a 3-vector such that $\boldsymbol{p}(s) \in \mathbb{R}^3$. A representation of such path is depicted in Figure 2.5.

Extending this definition, it is possible to define a pose path in space with respect to some parameter s in the same way. Each point in this path defines a position and orientation that can then be parameterized as an homogeneous transformation matrix such that $\mathcal{X}(s) \in \mathbb{SE}(3)$. A representation of such path is depicted in Figure 2.6

and can be parameterized as:

$$\mathcal{X}(s) = \begin{bmatrix} \boldsymbol{R}(s) & \boldsymbol{p}(s) \\ \mathbf{0_{1\times 3}} & 1 \end{bmatrix} \tag{2.32}$$

$$\text{with:} \qquad \boldsymbol{p}(s) \in \mathbb{R}^3 \qquad \boldsymbol{R}(s) \in \mathbb{SO}(3) \qquad \mathcal{X}(s) \in \mathbb{SE}(3)$$

where $\boldsymbol{R}(s)$ does for the orientation what $\boldsymbol{p}(s)$ does for the position: it parameterizes an orientation path as a function of some parameter s.

2.3.2 Discretizing a pose path ↑

One can discretize the s parameter to sample some of the poses throughout this path such that $\mathcal{X}_k = \mathcal{X}(s_k)$. Then the pose increment required to go from $\mathcal{X}_k$ to $\mathcal{X}_{k+1}$ is designated $\mathcal{X}_{k,k+1}$; the added comma makes the reference to each discretized step clearer but it is not needed after the following definition:

$$\Delta\mathcal{X}_k = \mathcal{X}_{k,k+1} = \mathcal{X}_k^{-1}\mathcal{X}_{k+1} \tag{2.33}$$

subsequently, the pose at each discretized step can be retrieved through:

$$\mathcal{X}_{k+1} = \mathcal{X}_k\Delta\mathcal{X}_k \tag{2.34}$$

Considering $\mathcal{X}_k$ describes the successive poses of a body moving through space, then (2.33) designates a "local" roto-translational displacement or transformation, because its base frame is $\mathcal{X}_k$, a frame moving with the body. Remembering (2.29) and (2.31), one can see that there exists an equivalent twist that represents this displacement through its exponential coordinates:

$$\Delta\mathcal{X}_k = e^{\mathcal{S}_k} \qquad \Delta\mathcal{X}_k \in \mathbb{SE}(3) \qquad \mathcal{S}_k \in \mathfrak{se}(3) \tag{2.35}$$

yielding:

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\mathcal{S}_k} \tag{2.36}$$

where the equivalent twist $\mathcal{S}_k$ can also be considered a local or body twist. This is due to the chosen right-multiplied propagation scheme in (2.34). As this is a Lie manifold, it may also be formulated to use a left-multiplied convention:

$$\begin{array}{cc} \mathcal{X}_{k+1} = \Delta\mathcal{X}'_k\mathcal{X}_k & \Delta\mathcal{X}'_k = \mathcal{X}_{k+1}\mathcal{X}_k^{-1} \\ \mathcal{X}_k, \mathcal{X}_{k+1}, \Delta\mathcal{X}'_k \in \mathbb{SE}(3) & \Delta\mathcal{X}'_k = e^{\mathcal{H}_k} \qquad \mathcal{H}_k \in \mathfrak{se}(3) \end{array} \tag{2.37}$$

It can be shown that $\mathcal{H}$ is in fact the same twist $\mathcal{S}$ expressed in the global frame (see (2.38)). This, by analogy extends the previous interpretation to imply that $\Delta\mathcal{X}'_k$

refers to a pose variation expressed in the universal frame.

REMARK. World vs Body Twist The relation between a body twist ${}^{B}\mathcal{H}$ and a twist expressed in the universal frame ${}^{S}\mathcal{H}$ is (see Section 5.1 of [Traversaro et al., 2019]):

$$ {}^{S}\mathcal{H} = \mathcal{X}_B {}^{B}\mathcal{H} \mathcal{X}_B^{-1} \qquad \mathcal{X}_B \in \mathbb{SE}(3) \qquad \mathcal{H} \in \mathfrak{se}(3) \tag{2.38} $$

This also constitutes the adjoint action map of $\mathbb{SE}(3)$ over $\mathfrak{se}(3)$. One property of this map is that:

$$ e^{\mathcal{X}\mathcal{H}\mathcal{X}^{-1}} = \mathcal{X} e^{\mathcal{H}} \mathcal{X}^{-1} \tag{2.39} $$

Using (2.38) and (2.39) to replace in (2.36) (where ${}^{B}\mathcal{H} = \mathcal{H}$ and ${}^{S}\mathcal{H} = \mathcal{S}$) yields (2.37), showing they are indeed equivalent.

2.3.3 Single step integration as a differential equation ↑

The previous section served to intuitively introduce a way to propagate a pose path through a manifold as a function of a twist. This section connects concepts from the rotation and pose matrices with their derivatives (Sections 2.2.1.4 and 2.2.2.3 with derivatives in Sections 2.2.1.3 and 2.2.2.2). In fact, both derivatives are analogous because they can be interpreted as a Lie group differential equation and so only the case of the pose is revisited here.

The relevance of this comes from the fact that (2.37) is in fact the solution to the differential equation presented by the pose derivative:

$$ \dot{\mathcal{X}}(t) = \mathcal{H}(t)\mathcal{X}(t) \tag{2.40} $$ revisited ↑

where $\mathcal{H}$ is expressed in the global frame as in (2.28); this implies it is a left-trivialized Lie differential equation (refer to world and body twists in (2.38) or, for an external reference, refer to page 112 in Section 3.5 Summary in [Lynch et al., 2017]). In contrast, the right-multiplied version of this differential equation would be:

$$ \dot{\mathcal{X}}(t) = \mathcal{X}(t)\mathcal{S}(t) \tag{2.41} $$

where $\mathcal{S}$ represents a local twist and is integrated for a first step with (2.36).

Note that both solutions constitute a first-order Taylor integration expansion in the manifold [Forster et al., 2015]; [Solà et al., 2018] that is central to the linearization strategy that is presented in the following sections.

2.3.4 Pose trajectory as a product of exponentials ↑

Finally, this section presents how adding a time notion to a pose path allows formulating a trajectory in $\mathbb{SE}(3)$. As shown in (2.34), it is possible to reconstruct each discretized pose in each step as a function of the previous pose and a local pose increment. This notion can be extended for the whole path by discretizing the s parameter in N steps so that s_k is defined for $k = 0, 1, \ldots, N$, then:

$$ \mathcal{X}_{k+1} = \mathcal{X}_0 \Delta\mathcal{X}_1 \Delta\mathcal{X}_2 \ldots \Delta\mathcal{X}_k \tag{2.42} $$

that can be further generalized as:

$$\mathcal{X}_{k+1} = \mathcal{X}_0 \prod_{i=0}^{k} \Delta\mathcal{X}_i \qquad \Delta\mathcal{X}_i = \mathcal{X}_k^{-1}\mathcal{X}_{k+1} \tag{2.43}$$

Remembering (2.35), each pose increment can be replaced for its equivalent exponential coordinates:

$$\mathcal{X}_{k+1} = \mathcal{X}_0 \prod_{i=0}^{k} e^{\mathcal{S}_i} \qquad \mathcal{S}_i = \log(\Delta\mathcal{X}_i) \tag{2.44}$$

where log denotes the matrix logarithm (see (2.30)). However, these exponential coordinates do not have a timing notion. To obtain an equivalent body twist analogous to the instantaneous velocity of the body in space it is necessary to add a time discretization to each s_k such that:

$$s(t) \in [0,1] \qquad s_k = s(t_k) \qquad t_k = \mathrm{k}\,\Delta t \tag{2.45}$$

$$\text{for} \quad \mathrm{k} = 0, 1, \ldots, \mathrm{N} \tag{2.46}$$

And now (2.44) can be reformulated assuming the body is moving at a constant body twist $\mathcal{V}_k$ at each time step:

$$\mathcal{X}_{k+1} = \mathcal{X}_0 \prod_{i=0}^{k} e^{\Delta t \mathcal{V}_i} \qquad \mathcal{V}_i = \frac{\mathcal{S}_i}{\Delta t} \tag{2.47}$$

revisited ↓

This formula shows one more application of the product of exponentials introduced by Brockett in [Brockett, 1984] for open kinematic chains.

For a single step integration, one can obtain an explicit expression as a function of the body twist based on (2.47):

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\Delta t \mathcal{V}_k} \tag{2.48}$$

revisited ↓

Another way to interpret this propagation scheme is under the assumption of a "zero-order hold", i.e. that the body moves at a constant $\mathcal{V}_i$ for a duration of Δt at each step.

REMARK. Rotation Path The equations presented in this section were formulated for $\mathbb{SE}(3)$ but they are analogous to their version in $\mathbb{SO}(3)$ with $\mathcal{X} = \boldsymbol{R}$ and $\Delta\mathcal{X} = \Delta\boldsymbol{R}$.

2.4 Integration on a Lie Manifold ↑

Previous sections introduce the fundamental modeling aspects of floating rigid bodies in space. These were used to parameterize serial robot kinematic chains movements in the workspace as Lie group actions on a manifold. This can be specially appreciated in the product of exponentials, where the successive discrete roto-translational actions parameterize the body pose trajectory (see Section 2.3.4).

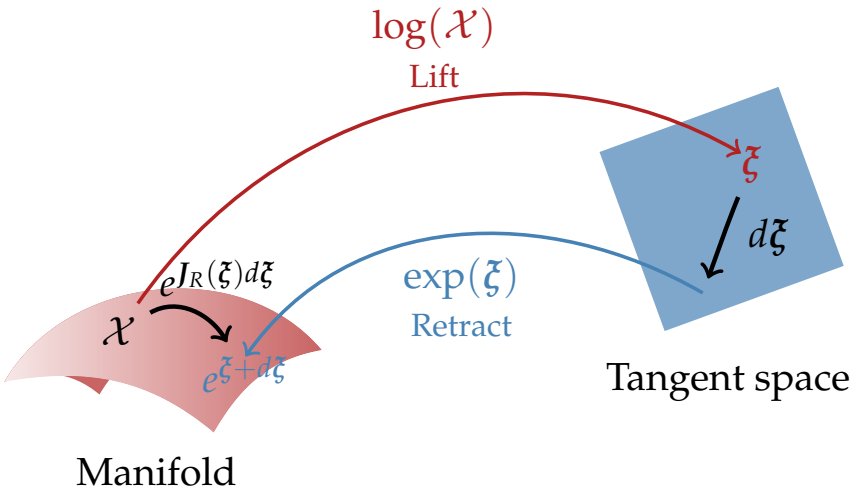


Figure 2.7: Depiction[2] of an integration scheme in the tangent space.

This section exploits the relation between the Lie groups and algebras to offer an equivalent algebraic integration scheme in the tangent space. It approaches the same concepts from a more general point of view to find a more convenient linear expression to the same problem. While the previous presentation relies on rather specific Lie groups (for the orientation and poses), this section borrows more general properties of Lie groups and applies them to present an efficient estimation strategy of the propagated poses in the manifold.

For a more comprehensive overview on the fundamentals employed in here, refer to [Solà et al., 2018].

2.4.1 Relation between a Lie group and its algebra ↑

For a given Lie group $\mathcal{G}$, there exists a tangent space at the identity that defines a Lie algebra $\mathfrak{g}$, the space of infinitesimal increment elements. The elements from the group can be mapped to their tangent space equivalents through the logarithm map (with the reciprocal operation being the exponential) such that for:

$$\begin{array}{rccc} \exp: & \boldsymbol{T}_e\mathcal{G} & \rightarrow & \mathcal{G} \\ \log: & \mathcal{G} & \rightarrow & \boldsymbol{T}_e\mathcal{G} \end{array} \tag{2.49}$$

where $\boldsymbol{T}_e\mathcal{G}$ denotes the tangent space of $\mathcal{G}$ at the identity element $\boldsymbol{e} \in \mathcal{G}$. This same relationship applies to the two previously analyzed cases: for orientations, it applies to $\mathbb{SO}(3)$ and $\mathfrak{so}(3)$ (see Section 2.2.1.4); and for poses, it can be applied to $\mathbb{SE}(3)$ and $\mathfrak{se}(3)$ (see Section 2.2.2.3).

Note that the exponential map surjectively covers the $\mathbb{SO}(3)$ manifold. This means that, for instance, given $\boldsymbol{\phi} \in \mathfrak{so}(3)$, it yields a one to one correspondence to $\boldsymbol{R} \in \mathbb{SO}(3)$ through the logarithm map as long as $||\boldsymbol{\phi}|| < \pi$. This condition in turn extends for the angular component of an element of the tangent space of the pose manifold $\mathfrak{se}(3)$ [Forster et al., 2015]; [Torres Alberto et al., 2022a].

[2] Based on tikz.net/manifold.

The goal of this section is to progressively integrate a path on the manifold through the propagation of the infinitesimal changes. It is possible to exploit this relation with the tangent space to map all elements from the Lie group manifold to elements in the tangent space. This re-parameterization is often referred to as a *push* [Absil et al., 2008] operation (also denoted *lift* [Solà et al., 2018]). Furthermore, an infinitesimal action on the manifold can then be mapped to an equivalent increment in the tangent space, which is referred to as a *pushforward*[3] [Saccon et al., 2013]; [Boumal, 2023]. The inverse action to retrieve the manifold element after the propagation is often referred to as *pull* (or *retract*) operation. A depiction of this process is visualized in Figure 2.7.

The following sections presents this strategy in more detail to integrate the pose actions in the tangent space through a convenient and efficient algebraic expression.

REMARK. The Adjoint Action map for Lie Groups The operator **Ad** represents the *adjoint action* of a Lie group $\mathcal{G}$ that defines a linear transformation on its Lie algebra $\mathfrak{g}$. For instance, given some elements $h \in \mathfrak{g}$, $y \in \mathcal{G}$:

$$\mathbf{Ad} : \mathcal{G} \times \mathfrak{g} \to \mathfrak{g} \tag{2.50}$$

$$\mathbf{Ad}_y\, h = yhy^{-1} \tag{2.51}$$

This constitutes the general definition of the operation defined in (2.38) for $\mathbb{SE}(3)$.

2.4.2 A discrete integration scheme on the tangent space of the manifold ↑

At the heart of this propagation scheme on a group element $\mathcal{X} \in \mathcal{G}$, is the choice of the push-pull function $\boldsymbol{\psi}$. As previously introduced, a convenient choice for this is the logarithm and exponential maps [Lee, 2013] such that:

$$\begin{array}{rcccccc} \boldsymbol{\psi} & : & \mathcal{G} & \to & \mathfrak{g} & \log(\mathcal{X}) & \to & \boldsymbol{\xi} \\ \boldsymbol{\psi}^{-1} & : & \mathfrak{g} & \to & \mathcal{G} & e^{\boldsymbol{\xi}} & \to & \mathcal{X} \end{array} \tag{2.52}$$

revisited ↓

where $\boldsymbol{\xi}$ represents the analogous variable in the tangent space.

As shown in [Forster et al., 2015] for the specific case of the Lie group $\mathbb{SO}(3)$ and in a more general sense in [Boothby, 1986] for a first-order approximation of the derivative in a differentiable manifold, it is possible to exploit the derivative of the exponential map to formulate a pushforward [Absil et al., 2008] integration scheme. Notably, in Equations 68-74 from [Solà et al., 2018], an infinitesimal variation in the manifold $d\boldsymbol{\xi}$ can be approximated as:

$$e^{\boldsymbol{\xi}+d\boldsymbol{\xi}} \approx e^{\boldsymbol{\xi}} e^{\mathbf{J}_R(\boldsymbol{\xi})d\boldsymbol{\xi}} \tag{2.53}$$

[3] In differential geometry, the pushforward refers to a linear approximation of smooth mappings that operates on tangent spaces.

which, for small variations, is shown to be equivalent to these other approximations:

$$e^{\xi}e^{d\xi} \approx e^{\xi + J_R^{-1}(\xi)d\xi} \tag{2.54}$$

$$\log(e^{\xi}e^{d\xi}) \approx \xi + J_R^{-1}(\xi)d\xi \tag{2.55}$$

where J_R maps variations in ξ to the right multiplicative increments in the exponential map. This is in fact the exponential map derivative dexp and J_R^{-1} corresponds to the logarithm map derivative. Furthermore this generalizes to a pushforward because the push function choice coincides with the exponential map.

Alternative notations to these derivatives J_R, J_R^{-1} are respectively dexp, dlog. These are used indifferently throughout this manuscript with the implicit convention that they correspond to their right-trivialized convention.

This enables a first-order Taylor expansion of the integration scheme [Forster et al., 2015] analogous to (2.36) with a twist but in the tangent space:

$$\xi_{k+1} \approx \xi_k + \Delta t J_R^{-1}(\xi_k)\frac{d\xi_k}{dt} \tag{2.56}$$

where $\frac{d\xi_k}{dt} = \mathcal{V}^{\vee}$ and the corresponding element $\mathcal{X}_{k+1}$ can be retrieved through the pull:

$$\mathcal{X}_{k+1} = \psi^{-1}(\xi_{k+1}) \tag{2.57}$$

It can be seen that equation (2.56) has a linear relation with respect to the body twist. This is the foundation that enables linear Model Predictive Control (MPC) in the following chapters.

The following subsections show that there exists a way to approximate J_R^{-1} to speed up computations with acceptable precision. The interest of this expression is revealed in the final section of this chapter, to perform an approximate linear pose propagation in a horizon (where the linearization is performed only at the start of a time window). This assumes that J_R does not change much during the horizon, which is an acceptable compromise for any linearization approach and is also why the approximation gets worse with longer horizons.

2.4.3 Solving a Lie group differential equation via the Magnus expansion ↑

An alternative development path to arrive at (2.56) can follow a development from a Lie group differential equation, which is a recurrent apparition as shown in Sections 2.2.1.4 and 2.2.2.3. This is a more general approach but enables the use of the Magnus expansion solutions for its solution [Magnus, 1954]. This crucial series allows for an efficient approximation of dexp (or, equivalently, the derivative of the exponential map J_R in (2.56)) through a truncation. As it is shown later, this yields some computational advantages.

Consider the following differential equation:

$$\dot{\mathcal{X}}(t) = \mathcal{X}(t)\boldsymbol{U}(t) \qquad \text{with } \mathcal{X}(t_0) = \mathcal{X}_0 \tag{2.58}$$

with $\mathcal{X}(t) \in \mathcal{G}$ and $\boldsymbol{U}(t) \in \mathfrak{g}$ (as proposed in [Zanna, 1999]). It is possible to find a solution in the form:

$$\mathcal{X}(t) = e^{\boldsymbol{\Omega}(t)} \qquad \boldsymbol{\Omega}(t_0) = \log(\mathcal{X}_0) \quad (\text{for } \mathcal{X}(t_0) = \mathcal{X}_0) \tag{2.59}$$

where $\boldsymbol{\Omega}(t)$ incorporates the cumulative effects of the infinitesimal element $\boldsymbol{U}(t)$ in time over the initial element $\mathcal{X}_0$. Note that $\mathcal{X}(t)$ and $\boldsymbol{U}(t)$ do not necessarily commute: this means that $\mathcal{X}_0\boldsymbol{U}_0\boldsymbol{U}_1 \neq \mathcal{X}_0\boldsymbol{U}_1\boldsymbol{U}_0$ (or for any other order for that matter). This is why $\boldsymbol{\Omega}(t)$ cannot be found via a simple integration and why the Magnus expansion is required [Hairer et al., 2006].

A consequence of (2.59), is that $\dot{\mathcal{X}}(t) = e^{\boldsymbol{\Omega}(t)}\text{dexp}_{\boldsymbol{\Omega}(t)}\boldsymbol{\Omega}'(t)$. One can see the resemblance of this expression with the derivative of the exponential map[4] at x:

$$\frac{d}{dt}e^{x(t)} = \text{dexp}_{x(t)}x'(t) \tag{2.60}$$

From which, based on (2.58), one can identify $\boldsymbol{U}(t)$ as

$$\boldsymbol{U}(t) = \text{dexp}_{\boldsymbol{\Omega}(t)}\boldsymbol{\Omega}'(t) \tag{2.61}$$

where $\text{dexp} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$. Then, one can retrieve $\boldsymbol{\Omega}'(t)$ from (2.61) by inverting dexp:

$$\boldsymbol{\Omega}' = \text{dexp}_{\boldsymbol{\Omega}}^{-1}\boldsymbol{U}(t) = \sum_{n=0}^{\infty}\frac{B_n}{n!}\mathbf{ad}_{\boldsymbol{\Omega}}^{\,n}\boldsymbol{U}(t) \tag{2.62}$$

where B_n are the Bernoulli numbers and $\mathbf{ad}$ is the adjoint operator defined a series of Lie brackets (see (2.65) for the case of $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$):

$$\mathbf{ad}_{\boldsymbol{\Omega}}^{\,n} = \underbrace{[\boldsymbol{\Omega}(t), [\ldots, [\boldsymbol{\Omega}(t)}_{n-times}, \boldsymbol{U}(t)]]] \tag{2.63}$$

Actually, the main interest lies in finding $\boldsymbol{\Omega}(t)$ that satisfies (2.58), which can be obtained through the repeated commutation and integration of $\boldsymbol{U}(t)$. The relation between (2.62) and (2.61) has been extensively studied in the literature, for instance in [Iserles et al., 2000]; [Iserles et al., 1999]; [Blanes et al., 2015]; [Schur, 1891]; [Rossmann, 2006]. Note that $\boldsymbol{\Omega}(t) = \int_{t_o}^{t}\boldsymbol{U}(x)dx$ iff $\boldsymbol{U}$ is constant or commutes with its primitive (i.e. $[\boldsymbol{U}(t), \int \boldsymbol{U}(t)dt] = 0$). Otherwise, $\boldsymbol{\Omega}(t)$ is an infinite series known as the *Magnus expansion* [Magnus, 1954]; [Blanes et al., 2008]:

$$\boldsymbol{\Omega}(t) = \int_0^t \boldsymbol{U}(t)dt + \frac{1}{2}\int_0^t\left[\int_0^{t_1}\boldsymbol{U}(t_2)dt_2, \boldsymbol{U}(t_1)\right] + \ldots \tag{2.64}$$

[4]The derivative of the exponential map is often presented as $\frac{d}{dt}e^{x(t)} = e^{x}\frac{(1-e^{\mathbf{ad}_x})}{\mathbf{ad}_x}\frac{dx}{dt}$ or, with a different notation but equivalently: $\frac{d}{dt}e^{x(t)} = \text{dexp}_{x(t)}x'(t)$

REMARK. The Lie bracket and the adjoint operator
Lie algebras such as $\mathfrak{g}$ are equipped with the commutation operator:

$$[X, Y] = XY - YX = \mathbf{ad}_X Y \qquad X, Y \in \mathfrak{g} \tag{2.65}$$

This is also referred to as the *adjoint action* endomorphism, as it is a linear map onto itself such that $\mathbf{ad}_{\mathfrak{g}} \cdot : [\mathfrak{g}, \mathfrak{g}] \to \mathfrak{g}$. For the specific case of the algebras $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$, is defined as [Solà et al., 2018]:
for $\boldsymbol{\phi} \in \mathfrak{so}(3)$:

$$\mathbf{ad}_{\boldsymbol{\phi}_1} \boldsymbol{\phi}_2 = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2] = \boldsymbol{\phi}_1 \times \boldsymbol{\phi}_2 = \boldsymbol{\phi}_1^{\wedge} \boldsymbol{\phi}_2 \tag{2.66}$$

$$\text{with:} \quad \mathbf{ad}_{\boldsymbol{\phi}_1} = \boldsymbol{\phi}_1^{\wedge} \in \mathbb{R}^{3\times 3} \tag{2.67}$$

for $\boldsymbol{\xi} = [\boldsymbol{\rho}^T, \boldsymbol{\phi}^T]^T \in \mathfrak{se}(3)$:
(with a linear component $\boldsymbol{\rho} \in \mathbb{R}^3$ and angular component $\boldsymbol{\phi} \in \mathfrak{so}(3)$)

$$\mathbf{ad}_{\boldsymbol{\xi}_1} \boldsymbol{\xi}_2 = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2] = \begin{bmatrix} \boldsymbol{\phi}_1 \times \boldsymbol{\rho}_2 + \boldsymbol{\rho}_1 \times \boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_1 \times \boldsymbol{\phi}_2 \end{bmatrix} \tag{2.68}$$

$$\text{with:} \quad \mathbf{ad}_{\boldsymbol{\xi}_1} = \begin{bmatrix} \boldsymbol{\phi}_1^{\wedge} & \boldsymbol{\rho}_1^{\wedge} \\ \mathbf{0} & \boldsymbol{\phi}_1^{\wedge} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.69}$$

Note that $\mathbf{ad}_{\boldsymbol{\phi}_1} \boldsymbol{\phi}_2 \in \mathfrak{so}(3)$ and $\mathbf{ad}_{\boldsymbol{\xi}_1} \boldsymbol{\xi}_2 \in \mathfrak{se}(3)$.

2.4.3.1 Truncating the Magnus expansion to approximate the logarithm map derivative ↑

The relevance to this chapter comes from a truncated version of (2.62) up to $n = 2$, referred to as ME2 from now on (remembering that $B_n = 0$ for all odd $n > 1$, which implies that ME3 $\equiv$ ME2). This allows approximating (2.62) in a linear form with respect to $\boldsymbol{U}(t)$:

$$\boldsymbol{\Omega}'(t) \approx \boldsymbol{U}(t) + \frac{[\boldsymbol{\Omega}(t), \boldsymbol{U}(t)]}{2} + \frac{[\boldsymbol{\Omega}(t), [\boldsymbol{\Omega}(t), \boldsymbol{U}(t)]]}{12} \tag{2.70}$$

$$\boldsymbol{\Omega}'(t) \approx \boldsymbol{U}(t) + \frac{\mathbf{ad}_{\boldsymbol{\Omega}(t_k)} \boldsymbol{U}(t)}{2} + \frac{\mathbf{ad}^2_{\boldsymbol{\Omega}(t_k)} \boldsymbol{U}(t)}{12} \tag{2.71}$$

$$\boldsymbol{\Omega}'(t) \approx \boldsymbol{K}\boldsymbol{U}(t) \tag{2.72}$$

$$\text{with:} \quad \boldsymbol{K}(t) = \mathbf{I} + \frac{\mathbf{ad}_{\boldsymbol{\Omega}(t_k)}}{2} + \frac{\mathbf{ad}^2_{\boldsymbol{\Omega}(t_k)}}{12} \tag{2.73}$$

Notice that $\boldsymbol{K}$ allows approximating dexp^{-1} by truncating the series. In fact, $\boldsymbol{K} \approx \boldsymbol{J}_R^{-1}$ from (2.56). This expression is crucial to arrive at a linear algebraic expression of the pose in the logarithm map in the following section and constitutes the main contribution of this chapter.

2.4.3.2 An approximative pose propagation in the logarithm map ↑

Equation (2.58) was formulated for a very general case of Lie groups, but it can be applied to the specific one of poses and orientations if one notices the resemblance with the pose and rotation matrix derivatives[5]:

$$\mathcal{X}^{-1}\dot{\mathcal{X}} = \boldsymbol{\nu}^{\wedge} \Rightarrow \dot{\mathcal{X}} = \mathcal{X}\boldsymbol{\nu}^{\wedge} \qquad \mathcal{X} \in \mathbb{SE}(3) \qquad \boldsymbol{\nu} \in \mathfrak{se}(3) \tag{2.74}$$

$$\boldsymbol{R}^{-1}\dot{\boldsymbol{R}} = \boldsymbol{\omega}^{\wedge} \Rightarrow \dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{\omega}^{\wedge} \qquad \boldsymbol{R} \in \mathbb{SO}(3) \qquad \boldsymbol{\omega} \in \mathfrak{so}(3) \tag{2.75}$$

which, under the assumption that $\boldsymbol{U}(t)$ remains constant throughout an integration step (meaning $\boldsymbol{U}(t) = \boldsymbol{U}$, that is necessary for (2.36)), means that $\boldsymbol{U}$ is in fact the twist or the angular velocity applied, for each respective case.

Furthermore, noting that:

$$\log(\mathcal{X}(t)) = \log(e^{\boldsymbol{\Omega}(t)}) = \boldsymbol{\Omega}(t) \tag{2.76}$$

then it becomes evident that $\boldsymbol{\Omega}(t)$ is in fact the desired logarithm of the pose $\boldsymbol{\xi}(t)$. Using a first-order Taylor expansion of the discretized $\boldsymbol{\Omega}(t)$:

$$\boldsymbol{\Omega}(t_{k+1}) \approx \boldsymbol{\Omega}(t_k) + \Delta t \boldsymbol{\Omega}'(t_k) \tag{2.77}$$

where $\boldsymbol{\Omega}'(t_k)$ can be replaced with (2.73) for $\boldsymbol{U} = \boldsymbol{\nu}_k$ and $\boldsymbol{\Omega}(t_k) = \boldsymbol{\xi}_k$:

$$\boldsymbol{\xi}_{k+1} \approx \boldsymbol{\xi}_k + \Delta t \boldsymbol{K}(t_k)\boldsymbol{\nu}_k \tag{2.78}$$

In this expression, $\boldsymbol{K}$ only depends on $\boldsymbol{\Omega}(t_k) = \boldsymbol{\xi}_k$. Finally, this shows, in a practical way, how to approximate (2.36) in the form of (2.56).

The final section of this chapter performed some numerical simulations to evaluate the performance of the approximation presented here.

2.5 Linear Pose Estimation in a Horizon ↑

The application of this manuscript is within the context of collaborative robotics. This application requires efficient control of the robot end-effector while carefully considering the movement trajectory. This setting implies the robot operates in a continuously changing environment. In this context, the robot needs to constantly take into account these events and adjust its behavior accordingly.

To do so, optimization-based schemes could be proposed that consider multiple aspects of the application as long as there exists a way to estimate future states of the robot and predict the effect of these movements. Previous sections introduced a way to describe the pose of the end-effector in space and devised a strategy to integrate its movement based on the body twist. Additionally, an approximative way to perform this estimation was proposed (see Section 2.4.3.1). The objective of this section is to showcase the precision and computational advantage of this approximation.

[5] Expressions for these can be found in Lynch and Park's "Modern Robotics: Mechanics, Planning, and Control" (2017) [Lynch et al., 2017]. Page 112, Section 3.5 of Chapter 3 offers a table with the rotation and rigid body motion analogies. Section 3.2.3.1 delves into some linear differential equations essentials that are applied in Eq. 3.50 (Page 83), which resembles (2.58).

A numerical experiment experiment is proposed to create the necessary conditions of an acceptable scenario for an estimation algorithm that could potentially be used for optimization methods. This experiment re-creates the hypothetical conditions of a horizon estimation: the only perfectly known information is the measured pose of the robot at the current point in time. To test the precision of estimating the future poses, a discretized and perfectly known reference pose path is necessary, from which the body twist at each descretized step can be computed. Then, the estimated poses propagated with the initial pose and the hypothetical perfectly known twists can be compared to the reference pose path to evaluate the performance of the methodology.

2.5.1 Problem statement: pose propagation in linear form ↑

To begin, it is necessary to describe the end-effector pose as a state-space system to clarify the involved variables and why a linear algebraic form is fundamental.

One can formulate a first order integration scheme of a system as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k \tag{2.79}$$

where $\boldsymbol{x}$ and $\boldsymbol{u}$ designate respectively the state and input (decision) variables of the system. Meanwhile, $\boldsymbol{A}_k, \boldsymbol{B}_k$ are state transition matrices that linearly propagate the state of the system.

In the interest of controlling the robot end-effector, the state $\boldsymbol{x}$ describes its pose in the tangent space (see Sections 2.4.1 and 2.4.2) and the decision variable is its body twist. It can be seen that (2.56) resembles (2.79) and can be further approximated with (2.78) through the ME2 truncation.

Another implication of (2.56) is that it needs a linearization scheme at each time step k for the matrices $\boldsymbol{A}, \boldsymbol{B}$. Nevertheless, the pose propagation in a sliding time window implies that the future states of the robot (those that are estimated throughout the horizon) are unknown. Thus, the linearization is only performed at the beginning of the horizon, on the only known state (the current measured pose). In such case, the assumption is that $\boldsymbol{A}_k, \boldsymbol{B}_k$ can be approximated as $\boldsymbol{A} = \boldsymbol{A}_0, \boldsymbol{B} = \boldsymbol{B}_0$ and then (2.79) is extended for a horizon:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}(\boldsymbol{u}_0 + \boldsymbol{u}_1 + \cdots + \boldsymbol{u}_k) \tag{2.80}$$

This expression constitutes the foundation necessary for a linear estimation of a system state in some optimization methods like linear Model Predictive Control (MPC). The next section shows how it relates to pose estimations in the tangent space.

2.5.2 Discrete pose estimation in a horizon ↑

Assuming the pose $\mathcal{X}_k = e^{\xi_k}$ is measured at the start of any horizon, one can assume that $\boldsymbol{K}$ in (2.78) does not change significantly through a time window. This enables reconstructing the pose in a horizon as a function of the twist, through an approximation of (2.56) in the same form as (2.80), where the state $\boldsymbol{x}$ and the inputs $\boldsymbol{u}$ are

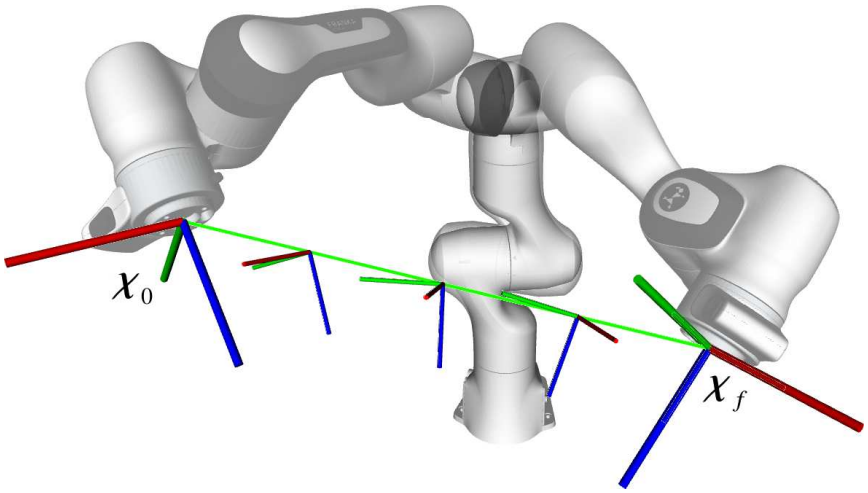


Figure 2.8: Depiction of the path taken taking the end-effector of a Franka Emika robot from an initial to a target pose $\mathcal{X}_i, \mathcal{X}_t \in \mathbb{SE}(3)$.

replaced with their chosen quantities $\boldsymbol{\xi}, \boldsymbol{\nu} \in \mathfrak{se}(3)$, respectively:

$$\mathcal{X}_{k+1} = e^{\boldsymbol{\xi}_{k+1}} \tag{2.81}$$
$$\text{with:} \quad \boldsymbol{\xi}_{k+1} \approx \boldsymbol{\xi}_0 + \Delta t \boldsymbol{K}(t_0)(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_1 + \ldots + \boldsymbol{\nu}_k)$$

where $\boldsymbol{\xi}_0 = \log(\mathcal{X}_0)$ corresponds to the pose at the start of the horizon (at time t_0) and $\boldsymbol{K}_0$ denotes ME2 evaluated at the same time point (see Section 2.4.3.1). Note that this expression effectively serves to approximate (2.47) in the tangent space.

2.5.3 Simulation Experiment ↑

The goal of this experience is to emulate the conditions of estimating the pose of the robot subject to some twist throughout a horizon. More specifically, the objective is to find out how precise are the results obtained from (2.81).

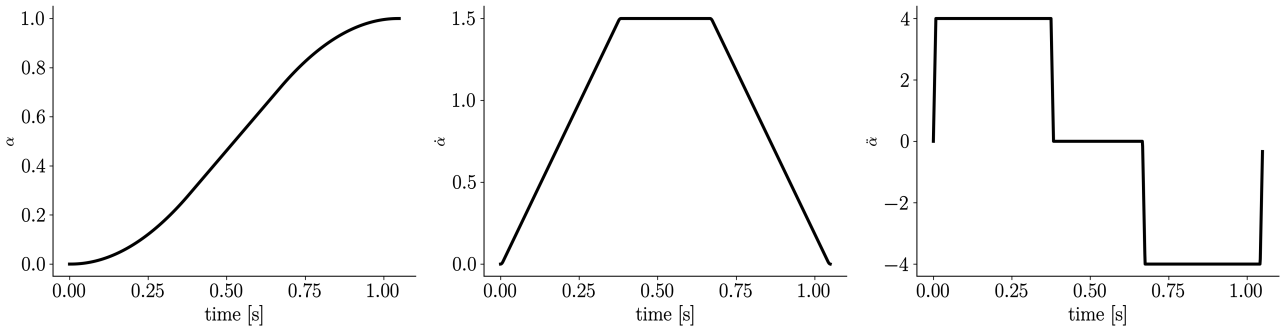
For this experiment, a time-optimal trajectory between two poses is discretized to obtain the ground truth poses and twists at each time step. Then, the same pose trajectory is reconstructed by assuming a known starting pose at the beginning of the horizon and applying the perfectly known twists throughout the time window. The result is then compared to the ground truth pose trajectory.

2.5.4 Simulation results ↑

Let the initial and target poses of a robot end-effector path be $\mathcal{X}_i, \mathcal{X}_t \in \mathbb{SE}(3)$ as depicted in Figure 2.8. For this experiment, in order to represent a typical movement of a Franka Emika robot in its workspace, two paths with different positional and rotational displacement were generated:

- For trajectory 1: $||\boldsymbol{p}_f - \boldsymbol{p}_0|| = 0.71\text{m}$ and $||ang(\delta)|| = 0.94\text{rad}$.
- For trajectory 2: $||\boldsymbol{p}_f - \boldsymbol{p}_0|| = 0.32\text{m}$ and $||ang(\delta)|| = 0.90\text{rad}$.

The rotational displacement (the norm of the angular component $ang(\delta)$) is computed from $\delta = \log(\mathcal{X}_0^{-1}\mathcal{X}_f)$.



(a) $s(t)$ (b) $\dot{s}(t)$ (c) $\ddot{s}(t)$

Figure 2.9: Normalized $s(t)$ for a time-optimal trajectory generated with a trapezoidal profile (in velocity).

Defining $s \in [0, 1]$ to discretize the path between both poses as: $\mathcal{X}_k^* = \mathcal{X}_0 e^{s_k \delta}$ yields a ground truth reference velocity twist $\boldsymbol{\nu}_k^* = \frac{\log(\mathcal{X}_k^{*-1} \mathcal{X}_{k+1}^*)}{\Delta t}$.

This implies that the reference trajectory can be reconstructed with:

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\Delta t \boldsymbol{\nu}_k^*} \tag{2.82}$$

$$\text{where: } \log(\mathcal{X}_k) = \boldsymbol{\xi}_k \tag{2.83}$$

The Ruckig library [Berscheid et al., 2021a] is used[6] to compute a normalized $s(t)$ in time to find the time-optimal trajectory, following a trapezoidal trajectory (in velocity), as shown in Figure 2.9.

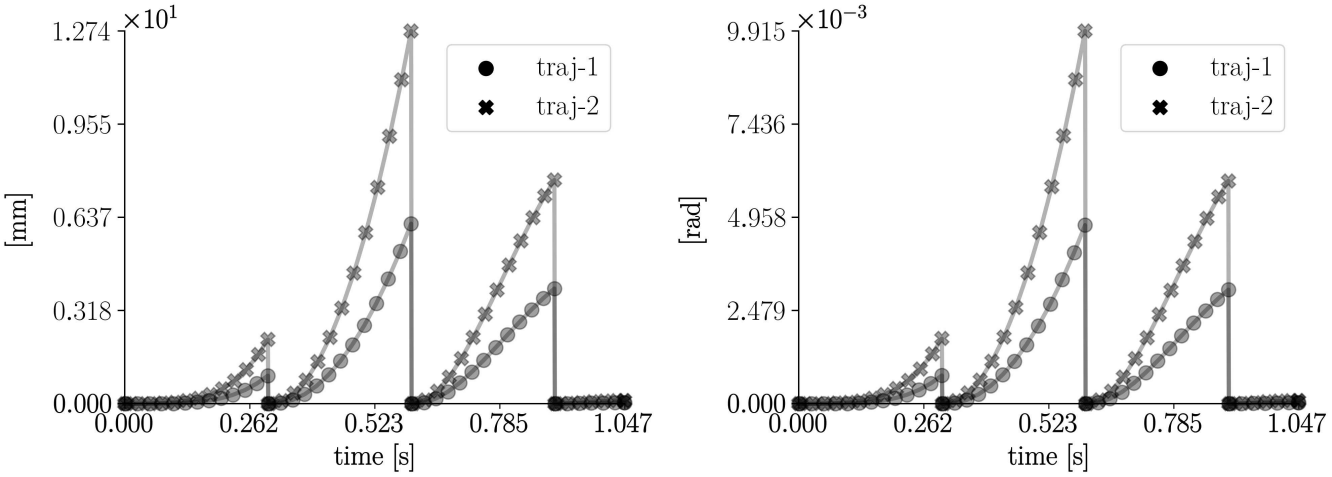
This showcases the precision of the linear integration of a robot end-effector pose by using (2.81) throughout an H-step horizon and comparing it against (2.82), the ground truth. Figure 2.10 depicts the obtained errors with respect to the perfect propagation.

Assuming a receding horizon approach where only horizon estimation matters, the pose is reset back to ground truth at the end of each horizon (every 300ms) to showcase the maximum error obtained with the method and to exemplify how the precision worsens towards the end of the horizon.

Table 2.1 shows the computation time and error averages obtained for the ME2 approximation. The method achieves a linear and angular error of 1.09 ± 6.1mm and $0.048 \pm 0.27°$ (for the longest trajectory), respectively. This is done in about two thirds of the time it takes to do it with the exponential (exact but non linear).

To further illustrate the effects of enlarging the horizon duration, we show in Figure 2.11 the same experiment run multiple times for different values of H (number of time steps). It can be seen that the longer the horizon, the greater the error is. Thus, a compromise must be made in order to define a horizon that allows to predict further enough in time without compromising precision.

[6]The code used for experiments in this chapter is publicly available at: https://gitlab.inria.fr/auctus-team/publications/shared-paper-code/ark2022-lin



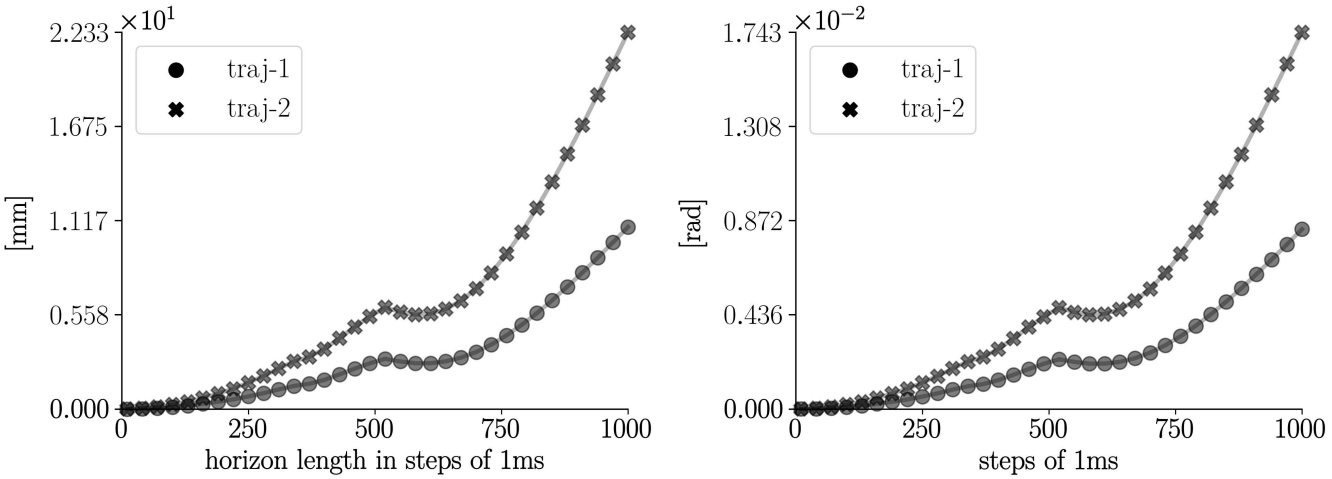
(a) Norm of linear error

(b) Norm of angular error

Figure 2.10: Variable speed integration using ME2 for a 300ms horizon.

Trajectory	Compute Time			Linear Error			Angular Error		
	Avg	Rel	Max	Avg	Max	Std	Avg	Max	Std
ref-1	0.23μs	100.0%	1.57μs						
traj-1	0.15μs	62.4%	3.74μs	1.09mm	6.15mm	$\pm$1.47mm	0.05°	0.27°	$\pm$0.06°
ref-2	0.18μs	100.0%	0.82μs						
traj-2	0.13μs	72.3%	0.23μs	2.26mm	12.74mm	$\pm$2.99mm	0.10°	0.57°	$\pm$0.13°

Table 2.1: **Shows the average compute time and error for the horizons shown in Figure 2.10.**
To showcase the trade off between precision and computation time, this comparison shows the error between the ground truth reference (computed with (2.82)) and the proposed ME-based method (using (2.81)) for each discretized step (as well as the maximum and standard deviations). The results are showed for 2 generated trajectories with their respective references. Compute time refers to the average for each Δt step on a C++ implementation. Relative computation time is calculated with respect to each reference average computation time. The tests were executed on a laptop with a CPU Intel(R) Core(TM) i7-10610U.



(a) Norm of linear error

(b) Norm of angular error

Figure 2.11: Average error for different horizon lengths.

2.6 Conclusions ↑

ME2 offers not only a precision level practical for many purposes but it could also offer a speed improvement when both methods of integration are possible (the exponential integration method cannot be embedded in a Quadratic Program (QP), which is the main goal of this work as it is required for linear Model Predictive Control (MPC)). This computational time improvement could also be exploited on robots with limited resources.

Lie algebras offer a convenient framework for pose description and system linearization on manifolds, allowing for vector expressions that can be embedded in a linear MPC problem, unlike their matricial forms.

A prospective interest in the future is the estimation of a bound on the maximum error induced by the proposed linear approximation.

Future chapters revisit the techniques presented here for the dynamic re-planning of end-effector *pose* trajectories subject to kino-dynamic constraints via a linear MPC implementation.

Chapter 3

Linear Model Predictive Control on SE(3): Velocity-based

Efficient workspace sharing of collaborative robots and human operators entails the need for robots to smoothly adapt tasks transitions while respecting changing workspace constraints. This remains an unsolved problem in the industry and goes beyond the use of a priori or a posteriori safety measures: it must be tackled at the control level.

To address the need of adaptation to human presence as well as to endow the robot with the ability to adapt interactively to Cartesian targets moving in 3D space, a linear Model Predictive Controller is proposed in this Chapter. This controller computes acceleration-bounded optimal trajectories in SE(3) over a receding horizon, enabling position and orientation online re-planning.

The pertinence of the proposed control architecture is demonstrated using experiments with the Franka Emika robot in a scenario implying both adaptation of the maximum allowed velocity to comply with human presence and on-the-fly updates of a Cartesian goal pose.

Contents

3.1 Introduction ↑

The arrival of the so-called "collaborative robotics" has brought forth an on-going paradigm shift from traditional, static robot work-spaces to dynamic environments of shared collaboration with humans. This vision projects efficient factories that seamlessly transition between robotic cells to fully interactive human-robot stations that exploit each others qualities in synergy.

Throughout this chapter, a control architecture for interactive collaborative robotics is introduced. It is based on a cascade controller composed of a task-space planner and a model-based solver. The rationale behind this architecture is presented in Section 3.1.1.

Overall the first sections present some groundwork concepts and notation conventions employed in this architecture proposal. To begin, Section 3.2 focuses on the inner stage, a constrained model and optimization-based controller. To do so, it first offers an overview of some robotics basics and then follows with some state-of-the-art applications of these modeling fundamentals applied to optimization-based control.

Afterwards, Section 3.3 delves into the task-space planner for online motion generation based on Model Predictive Control (MPC). First, some foundations for linear MPC are presented. Subsequently, it revisits the concepts introduced in Chapter 2 to formulate an optimization scheme on the pose manifold, allowing the generation of constrained motions in position and orientation. This constitutes the main contribution of this chapter.

Afterwards, Section 3.4 provides an experimental validation of the proposed architecture for a safety-aware collaborative scenario. It begins with the description of the experimental setup and objectives. Then, it goes on to offer some preliminary validation results of online re-planning on a simulated environment and finally, an experiment is carried out on a 7 Degree of Freedom (DoF) manipulator to validate the pertinence of the approach.

Finally, Section 3.5 concludes the chapter by summarizing the contributions.

An accompanying video (see Figure 3.10) offers a compact summary of the results.

3.1.1 Rationale: Constrained Online Motion Generation ↑

Collaborative robotics has garnered significant attention in the literature, with numerous studies exploring different aspects of the nature of Human-Robot Interaction (HRI) in work-cells. Some of the main concerns relate to safety and ergonomics [Haddadin et al., 2016], which are often analyzed from the application point of view to reduce complexity [Gualtieri et al., 2021]. Yet, a systematic approach that addresses all or most requirements of HRI from a control perspective in a generic way remains to be proposed.

To achieve this, a paradigm shift in control strategies is necessary. Controllers must address the needs of factories in a way that seamlessly integrates collaborative robotic cells without fences. In such scenario, robots need to evolve in environments that are not only continuously changing but can also do so in unpredictable ways.

To successfully achieve a responsive robotic behavior that adapts to these changes, ensures safety for nearby humans, and performs tasks effectively, it is crucial for the robot to smoothly transition between different movements.

These transitions require the motion to be generated in an online fashion to be able to consider the current conditions in the environment and the state of collaborative tasks. This entails a two-fold problem:

1. Firstly, the robot requires the ability to fluidly switch between task objectives.

2. Secondly, the robot needs to perform movements that will remain coherently feasible during operation.

The first problem has been addressed in [M. Liu et al., 2015]; [Salini et al., 2011] in a reactive manner, i.e. taking a decision that considers only the next state of the robot. However, ignoring the future states of the planned motion can lead to risky situations. This is why these problems normally require a motion generation that considers the task from start to finish [Kurniawati et al., 2011]. Nevertheless, such strategy is inefficient due to two reasons: for one, it might get computationally non-viable for the real-time requirements of the collaborative scenarios; and secondly, the dynamic nature of the scenario implies that the task will need to be re-planned in order to adapt to new changes before it is ever completed.

Meanwhile, the second problem may seem simpler at first glance but due to the limited intrinsic actuation capabilities of the robot, some instantaneously viable states can lead to inevitable feasibility problems. In other words, there might exist a problem of "coherency" between actions / decisions taken in the present if they do not consider how their outcome might lead the robot into invalid states in the future. Because these actuation capacities are governed by non-linear dynamics, some simplifying strategies have been proposed to preemptively avoid entering these states [Rubrecht et al., 2010]; [Prete, 2018] and even consider their effects in the workspace motions [Rubrecht et al., 2012]. Some of these strategies resort to linking workspace and joint constraints with "interfacing" quantities like the minimal motion stopping time [Joseph et al., 2020]. Nevertheless, the consideration of the workspace constraints on their own and in combination with the above problems remains an open question. While this chapter main proposal focuses mainly on a task-planner that considers the compatibility of the work-space related constraints, it still accounts for the robot capabilities through the model-based embedded controller.

In order to start providing solutions to these issues, this chapter proposes a predictive control scheme that allows considering the future states of the robot in an effort to smoothly transition between tasks while considering the long term effects of these actions (long term as in robot task execution times).

An ideal solution would be a global planning strategy able to output complete trajectories (from start to finish). However, these approaches are usually computationally expensive and quickly become not viable when the scenario requires updating the whole trajectory for every change in the environment (without really taking advantage of having computed the complete trajectory).

The proposal is to avoid the pitfalls of global planning in dynamic scenarios by

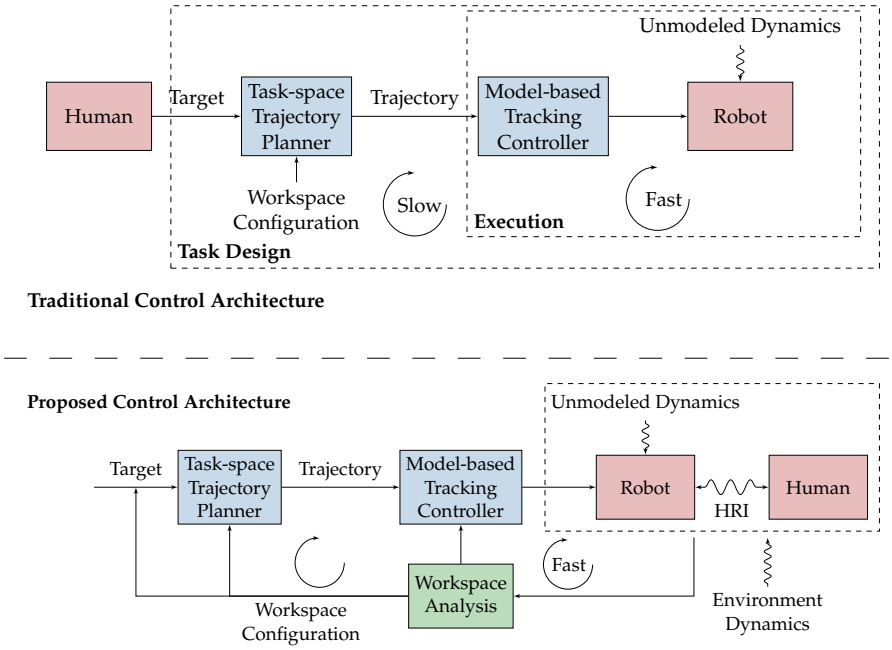


Figure 3.1: Both control architectures employ a task-planning block and a controller block. The first one is in charge of generating motions that will be executed by the latter. The planning needs to account for the "Workspace Configuration" (the state and constraints of the shared workspace). In traditional industrial robotics, the operator intervens during task design through a slow iterative process. Once the task is defined, it only needs to be executed on repetition. In contrast, collaborative robotics requires to have the human in the loop and be considered in the task planning. This implies that online re-planning is necessary, as is proposed in the architecture at the bottom.

revisited ↓

adopting a strategy that performs optimizations on a receding horizon, effectively providing computationally efficient solutions that often yield approximately global optimal strategies.

3.1.2 Control Architecture Proposal ↑

The traditional control architecture[1] shown in Figure 3.1 roughly depicts a classical approach to industrial robot cells. In this approach, the operator only intervenes during the task design process, by trying to find an approriate trajectory for some specified precision and cycle times requirements through a slow iterative process. For each iteration, the target and/or workspace constraints will be changed and executed on the robot to asses the result (or possibly simulated). Once a task trajectory is accepted, it only needs to be deployed to be executed on repetition.

In contrast, the collaborative control architecture presented in Figure 3.1 proposes a new perspective to the human in the control loop: the task now involves the HRI and so the human becomes a part of the dynamics that need to be accounted for during planning. It is one potential solution that accounts for the requirements stated in Chapter 1 and is composed of three main blocks:

1. Task-space Motion Generation

2. Model-based Tracking Controller

[1] In this case, control architecture is used in a vague way to refer to both the task design process and control during execution.

3. Workspace Analyzer

The first block consists of a task planner. From a human and task-centered point of view, task planning at the Cartesian level offers some advantages:

- It simplifies the robot programmer's work, as it corresponds to the natural space where the task will be performed.
- All the task-related information regarding objectives, limits, etc is expressed in the same space (as opposed to having to mix joint-space and Cartesian constraints).
- And finally, it decouples task description from the robot architecture, embedding some desirable modularity properties potentially either reusing designs or replacing parts.

The Cartesian trajectory planner block consists of an algorithm to generate a motion primitive that responds to the tasks needs while respecting the workspace configuration (constraints, sensor information, etc) and task objectives. For example, a planner could define a task that exploits the ergonomics of the HRI [Spaa et al., 2020] to reduce operator musculoskeletal stress. Furthermore, this block is in charge not only of dealing with task-related information (like its objectives) but it is also in charge of seamlessly articulating the task objectives with the workspace configuration (the state and constraints of the shared workspace). Another common example scenario, for tasks without physical interaction, is the essential requisite for safety: being able to reactively create paths that avoid collissions [Du et al., 2018]; [Mohammed et al., 2017]; [Safeea et al., 2019] or minimize the hazard. Due to the feedback loop between the task state and the environment information, this effectively implements trajectory planning in an online fashion.

At the second level, the model-based tracking controller block is in charge of executing the planned trajectory, i.e. translating the Cartesian movement into a joint-space motion to comply with the task needs. As this involves translating the motion into a control input, the constraints involved in this block mainly concern the actuation model of the robot in order to produce feasible servoing. While the mathematical formulation of this block might be generic and applicable to a multitude of robots, their implementation is actually robot-dependent, as it embeds the physical properties.

Finally, at the heart of this architecture, lies the workspace analyzer block. Though accurately sensing the state of the environment around the robot is challenging in itself [Halme et al., 2018], this approach to a reactive control architecture also requires sensing the workspace and actually translating this information in a way that can be exploited by the model-based controller and trajectory re-planner blocks in a coherent way (referred as workspace configuration in this manuscript). This is implementation-dependent and aims at extracting and synthesizing the environment information. For instance, in the case of some motion generating algorithms, this can mean using specific mathematical formulations for describing obstacles [Krämer et al., 2020]; [Spahn et al., 2023] or predicting human motions in the workspace [Bajcsy et al., 2021]; [Weitschat et al., 2018].

Added to this problem is the fact that environmental conditions are susceptible to

sensor noise and sudden changes, though to circumvent this problem, some special considerations have been proposed that avoid discontinuous changes that could result in instantly incompatible problems [Tan et al., 2015].

Note that the architecture is presented in a generic way to showcase its modularity as a framework for collaborative scenarios. Indeed, its blocks can be replaced depending on the application but the genericity of task-related planning with constraints can be applied to other safety-aware control architecture contributions [Benzi et al., 2021]; [Lihui Wang et al., 2013]; [Scalera et al., 2022].

For the collaborative scenarios considered in this manuscript, the proposal is to use a linear MPC for task-space planning that can be formulated as a quadratic cost function subject to linear constraints. While similar architecture approaches have been employed in humanoid control [Wieber, 2006]; [Lober et al., 2020], the application to collaborative robotics remains relatively new. In this context, the added value comes from offering a mechanism to formulate the workspace configuration through a convex polyhedral (linear constraints) and effectively allows the optimization problem to yield constrained trajectories over a sliding time window.

The proposed architecture aims at addressing the needs of agile manufacturing for robotics presented in Chapter 1, which is the reason behind some of the choices. For instance, by decoupling the Cartesian-space planning from the joint-space controller, the architecture effectively decouples task-related aspects (cost function, constraints, etc) from the actuation-related concerns (actuator limits, type of control input, robot model). The rationale behind this choice is to offer some flexibility if ever the work-cell needs to be reassigned (thus changing the task) or adapted to different robots (swapping the robot).

The following sections offer delve into the implementation of this architecture. To begin, an optimization-based approach for task-space tracking is proposed to implement the planning block. Afterwards, the motion generation (the planning block) is presented as the main contribution of this manuscript: a linear MPC implementation capable of constrained full pose planning (i.e. position and orientation). Finally, to offer some insights into the work-space analyzer, the final section introduces a practical experiment to showcase a safety-aware implementation.

3.2 Task-space Tracking ↑

Trajectory tracking refers to the ability of a robot to follow a predefined trajectory accurately. This trajectory is typically specified as a series of desired positions or configurations that the robot needs to reach over time. The robot objective is to adjust its control inputs (i.e. motor commands) in real-time to move along the trajectory as closely as possible.

Robots are controlled through actuators that define its control input space. Oftentimes, there is one actuator per joint and so this space directly relates to the robot configuration space, which defines the state of each joint (commonly the joint angles and/or velocities). Consequently, a task defined as a configuration-space trajectory is often simpler to execute because the translation to the robot control input space is often also rather direct.

In a more general sense though, for a robot controlled through some decision variable[2] $\boldsymbol{x} \in \mathbb{R}^n$, a tracking task can be described as an optimization problem with a cost function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ such that [Joseph, 2018]:

$$\begin{aligned} \boldsymbol{x}^{\text{opt}} &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0} \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}^{\text{opt}}$ designates the optimal control value to perform said task $f(\boldsymbol{x})$ while subject to some constraints $\boldsymbol{h}(\boldsymbol{x})$.

In this way, $f(\boldsymbol{x})$ relates the desired trajectory with the robot model. Here, *trajectory* is used in a generic way to designate a pre-defined series of interest points with an associated time. In this way, $\boldsymbol{x}^{\text{opt}}$ actually denotes the optimal control input in order to achieve the next desired interest point $f(\boldsymbol{x})$.

As mentioned earlier, if the trajectory is defined in the actuation space as $\boldsymbol{x}(t) \in \mathbb{R}^n$, then the unconstrained optimal control input $\boldsymbol{x}^{\text{opt}}$ can be directly computed. However, formulating the task as an optimization like in (3.1) can still be convenient due to the ability to explicitly and generically relate the control input to constraints in $\boldsymbol{h}(\boldsymbol{x})$ (for example, for the actuation limits). This results in a more generic way to consider the feasibility of the task, effectively allowing a "best effort" strategy, i.e. try to perform the task as best as possible.

Furthermore, one can define a position trajectory $\boldsymbol{p}(t) \in \mathbb{R}^3$ that takes the form of a 3-vector in time; or alternatively, if a trajectory is defined for position and orientation simultaneously, then it can take the form of a homogeneous transformation matrix in time $\mathcal{X}(t) \in \mathbb{SE}(3)$ (as in Section 2.3). The focus of this manuscript is on tracking *pose trajectories*, i.e. the tracking of a given position and orientation in time.

Actually, determining the appropriate input for a given task-space trajectory can be addressed differently depending on the type of control variable utilized by the robot. This defines two broad sets of problems: inverse kinematics problems, when the input variable of the robot is either the position or the joint velocities (addressed in this section); and inverse dynamics problems, when the control variable is of higher order and so requires exploiting the robot dynamics model (see Chapter 4), for example, joint accelerations, torques, etc.

Roughly speaking, this section focuses on formulating problem (3.1) as a constrained least squares problem similar to the control strategy proposed in [Salini et al., 2011] for humanoids:

$$\boldsymbol{x}^{\text{opt}} = \arg\min_{\boldsymbol{x}} \quad ||\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b}||^2 \tag{3.2}$$

$$\text{subject to} \quad \boldsymbol{A}_{\boldsymbol{u}} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{A}_{\boldsymbol{u}} \tag{3.3}$$

where (3.2) denotes the task space trajectory tracking problem, that can be efficiently

[2] Here $\boldsymbol{x}$ offers a generic way to define the control variable. Industrial robots often use joint angles, velocities and, in some rare cases, torques as an input.

solved as a Quadratic Program (QP)[3] (see the details on this reformulation on Appendix A). Additionally, (3.3) serves as a convenient way to formulate linear constraints with respect to the decision variable. The focus of the following sections is, using this form and some robot modeling foundations, to introduce a quadratic optimization-based controller that considers the actuation limits, for robots servoed at the kinematic level.

To skip the rationale and modeling aspects, a summary of the resulting controller is presented in Appendix C.

3.2.1 Kinematic Robot Model ↑

The tasks to be performed by a robot are naturally occurring in the 3D space, i.e. position and orientations. This is why trajectories are often defined in the same Cartesian or operational space as the task they parameterize. For example, a pick and place task requires the robot end-effector to move from some pose A towards some other pose B. Depending on the robot mechanical design, this same path from A to B may correspond to multiple configuration-space paths. Nevertheless, the robot still needs a correct joint-space control input to perform the task. This implies that some mathematical resolution is needed to articulate the tranformation from one space to the other.

While determining the pose of a robot body given some configuration can be done in a rather direct fashion in the case of serial robots, the inverse problem requires more thought. The first mathematical problem is concerned with calculating the position and/or orientation of a robot body in the task-space for a given robot configuration (its joint angles) and is known as Forward Kinematics (FK)[4]. The inverse problem of finding the robot configuration that achieves a certain position and/or orientation is known as Inverse Kinematics (IK).

For a given *configuration* of an n-Degree of Freedom (DoF) manipulator represented as a vector $\boldsymbol{q} \in \mathbb{R}^n$ containing the joint angles q_i for $i = 1...n$, the *pose* of a robot body (commonly the end-effector) in Cartesian space can be represented as a roto-translational displacement from the world frame. This pose is described by a homogeneous transformation matrix $\mathcal{X}_{WB}$, a function of the robot configuration that combines both the translation and rotation information necessary to go from the world to the body frame:

$$\mathcal{X}_{WB}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}_{WB}(\boldsymbol{q}) & \boldsymbol{p}(\boldsymbol{q}) \\ \mathbf{0_{1\times 3}} & 1 \end{bmatrix} \qquad \mathcal{X} \in \mathbb{SE}(3) \tag{3.4}$$

revisited ↑

This allows a more formal definition of FK and IK as:

$$\begin{aligned} \mathrm{FK}(\boldsymbol{q}) : \mathbb{R}^n \to \mathbb{SE}(3) \qquad & \mathrm{IK}(\mathcal{X}) : \mathbb{SE}(3) \to \mathbb{R}^n \\ \mathrm{FK}(\boldsymbol{q}) = \mathcal{X}_{WB} \qquad & \mathrm{IK}(\mathcal{X}) = \{\boldsymbol{q} \in \mathbb{R}^n, \mathcal{X} \in \mathbb{SE}(3) \mid \mathrm{FK}(\boldsymbol{q}) = \mathcal{X}\} \end{aligned} \tag{3.5}$$

[3] Quadratic programming (see Appendix A) is a mathematical optimization technique used to solve a wide range of problems in various fields. It deals with the optimization of a quadratic objective function subject to linear constraints.

[4] Forward Kinematics refers to the geometric analysis of a robot body in space, conferring it a rather static view. Nevertheless, kinematics is also used to refer to the study of motions without considering the forces that cause them. Interestingly, in french, *cinématique* is mostly associated to the latter sense while *géométrique* is preferred when the differential aspect is not considered.

While, for serial robots, the solution of FK is unique, the same cannot be said of IK. For redundant robots where the DoF of the robot is higher than the DoF of interest, IK has an infinite set of solutions. For example, such is the case of 7 joint serial manipulators used for pose tracking (6 DoF, to account for the position and orientation). Yet, in a more realistic scenario, the joint angles $\boldsymbol{q}$ are constrained to the set of feasible configurations limited by the joint bounds.

Moreover, considering real robots in operation implies that only the configurations near the current one are actually realizable, which also has some interesting implications that can be analyzed by considering a differential analysis of the FK.

The Differential Kinematics or Forward Instantaneous Kinematics [Siciliano et al., 2008] problem deals with the problem of computing the infinitesimal rigid body movement associated with an infinitesimal increment in the robot configuration.

For an n-DoF robot at any point in time, the *Jacobian* of a robot is defined as a matrix $\boldsymbol{J}$ that describes the relation between the Cartesian space end-effector pose rate of change and the rate of change of the joints $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ as [Lynch et al., 2017]:

$$ {}^{W}\boldsymbol{\nu} = {}^{W}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad \boldsymbol{J} \in \mathbb{R}^{6\times n} \quad \boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^n \quad \boldsymbol{\nu} \in \mathfrak{se}(3) \tag{3.6} $$ revisited ↓

where $\boldsymbol{\nu}$ corresponds to the rate of movement of the body, a spatial twist in the world frame. The frame in which the Jacobian is expressed will ultimately determine the resulting frame for the twist $\boldsymbol{\nu}$. If the jacobian is in a body frame, then the twist is referred to as a *body twist*.

A remarkable implication of (3.6) is that even if IK in (3.5) admits multiple solutions, given a robot *in operation*, its motion (i.e. the realized twists) might still be unique. For example, for a 6-DoF robot executing a task, given its current initial pose, the joint-space motion required to go to some other pose is still unique.

Nonetheless, in the more general case of robots with more than 6 DoF, not only does the IK admit infinite joint configurations, but it is also kinematically redundant in the motion sense, admitting infinitely many joint velocities that can achieve the same spatial twist.

Some final considerations for the robot model used in this chapter are the robot constraints. For the kinematic model of this section, the robot is considered up to the joint velocities level, implying that FK, IK and all the spatial twists $\boldsymbol{\nu}$ are defined for $\boldsymbol{q} \in \mathcal{Q}$ and $\dot{\boldsymbol{q}} \in \dot{\mathcal{Q}}$ such that:

$$ \begin{aligned} \mathcal{Q} &= \{\boldsymbol{q} \in \mathbb{R}^n \mid \boldsymbol{q}_m \leq \boldsymbol{q} \leq \boldsymbol{q}_M\} \\ \dot{\mathcal{Q}} &= \{\dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \dot{\boldsymbol{q}}_m \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_M\} \end{aligned} \tag{3.7} $$ revisited ↓

which correspond to sets defined by the lower and upper limits for the joint configurations and velocities.

The following sections articulate the foundations presented here to formulate the control problem introduced in (3.1) while taking into consideration (3.7).

3.2.2 Pose Tracking as a Quadratic Program ↑

Combining the robot modeling foundations introduced in Section 3.2.1 allows formulating many of the challenges in robotics as mathematical problems. These problems can then be efficiently solved through optimization techniques, enabling for a systematic approach to robot control.

One of these problems constitutes a major focus of this work: pose trajectory tracking. In practical terms, this section shows how to formulate the cost function of the problem introduced in (3.1) in the constrained normed least squares form shown in (3.2) so that it can be solved through QP (to see how, refer to Appendix A).

The case in consideration assumes the robot is controlled at the kinematic level, i.e. through its joint velocities $\dot{\boldsymbol{q}}$ and that the current robot configuration $\boldsymbol{q}_k$ can be measured.

A given a reference pose trajectory $\mathcal{X}^r(t) \in \mathbb{SE}(3)$, once discretized, yields a reference pose and twist (see Section 2.3) to be achieved at the current time step k: $\mathcal{X}^r_k = \mathcal{X}^r(t_k)$ and $\boldsymbol{\nu}^r_k = \boldsymbol{\nu}^r(t_k) \in \mathfrak{se}(3)$ (with $t_k = k\Delta t$ and $k \in \mathbb{N}$). The k sub-indexes are dropped for clarity in the following expressions, as they always refer to the current step. In an ideal world, this should be enough to plug right into (3.2) to relate it to the joint velocities:

$$\dot{\boldsymbol{q}}^{\text{opt}} = \arg\min_{\dot{\boldsymbol{q}}} \quad ||\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} - \boldsymbol{\nu}^r||^2 \tag{3.8}$$

Nevertheless, due to unmodeled dynamics, a Proportional Derivative (PD) term might be necessary for task servoing.

In that case, one can obtain the current pose of the robot $\mathcal{X}(\boldsymbol{q})$ using FK and then compute the ideal reference error twist in the body frame ${}^B\boldsymbol{e} \in \mathfrak{se}(3)$ required to arrive at $\mathcal{X}^r$ as:

$${}^B\boldsymbol{e} = \log(\mathcal{X}^{-1}\mathcal{X}^r) \tag{3.9}$$

Then, (3.8) can be reformulated with:

$$\dot{\boldsymbol{q}}^{\text{opt}} = \arg\min_{\dot{\boldsymbol{q}}} \quad ||{}^B\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} - {}^B\boldsymbol{\nu}^*||^2 \tag{3.10}$$

$${}^B\boldsymbol{\nu}^* = \boldsymbol{K}_p{}^B\boldsymbol{e} + {}^B\boldsymbol{\nu}^r \tag{3.11}$$

where $\boldsymbol{\nu}^*$ denotes the desired twist, that leverages the reference twist from the trajectory and a proportional term with $\boldsymbol{K}_p$. Note that in this case, the equations are expressed in the body frame for convenience but it can be reformulated as needed (see Section 3.2.4).

In order to generalize this formulation, one final problem remains: for kinematically redundant robots, $\boldsymbol{q}^{\text{opt}}$ will not be unique and this might result in non desirable vibrations in the redundant DoFs of the robot. To overcome this issue, one can add a "regularisation task" to the cost function with a very small weight value, so that it has a negligible effect on the main tracking task.

There is a wide variety of regularisation tasks [Joseph, 2018] that can be chosen depending on the application, ranging from projected mass minimization for safety purposes [Joseph et al., 2019] to posture tasks that try to maintain a configuration. For simplicity, in this manuscript, the later is adopted, yielding the final formulation of the cost function:

$$\dot{\boldsymbol{q}}^{\text{opt}} = \arg\min_{\dot{\boldsymbol{q}}} \quad ||{}^{B}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} - {}^{B}\boldsymbol{v}^{*}||^{2} + w_{\text{reg}}||\dot{\boldsymbol{q}}_{\text{reg}} - \dot{\boldsymbol{q}}||_{2}^{2} \tag{3.12}$$

$${}^{B}\boldsymbol{v}^{*} = \boldsymbol{K}_{p}{}^{B}\boldsymbol{e} + {}^{B}\boldsymbol{v}^{r} \tag{3.13}$$

$$\dot{\boldsymbol{q}}_{\text{reg}} = \boldsymbol{K}_{q}(\boldsymbol{q}_{\text{reg}} - \boldsymbol{q}) \tag{3.14}$$

$$\boldsymbol{q}_{\text{reg}} = \frac{(\boldsymbol{q}_{M} - \boldsymbol{q}_{m})}{2} \tag{3.15}$$

where $\boldsymbol{K}_{q}$ is a positive proportional term; $w_{\text{reg}} \ll 1$ and $\dot{\boldsymbol{q}}_{\text{reg}}$ are defined as in [Joseph et al., 2020], to maintain a configuration $\boldsymbol{q}_{\text{reg}}$ distant from the joint bounds.

This definition allows for a pose tracking cost function that effectively solves the IK problem through a QP. That is why this controller is also referred to as a constrained differential IK solver.

3.2.3 Constraints Formulation ↑

This section explains how the linear constraints in (3.3) can be used to formulate the actuation limits in (3.7). For a robot controlled through its joint velocity, $\dot{\mathcal{Q}}$ can be directly included into the linear constraints as:

$$\dot{\boldsymbol{q}}_{m} \leq \mathbf{I}_{n}\dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{M} \tag{3.16}$$

where $\mathbf{I}_{n}$ denotes an n-sized identity matrix (here, it is made explicit in order to show a linear inequalities form).

However, other type of constraints need to be reformulated as a function of the decision variable $\dot{\boldsymbol{q}}$. For instance, the set $\mathcal{Q}$ (from (3.7)) and $\ddot{\mathcal{Q}}$ (for the joint accelerations):

$$\ddot{\mathcal{Q}} = \{\ddot{\boldsymbol{q}} \in \mathbb{R}^{n} \mid \ddot{\boldsymbol{q}}_{m} \leq \ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{q}}_{M}\} \tag{3.17}$$

can be reformulated through a first order taylor expansion as:

$$\mathcal{C}_{\dot{\boldsymbol{q}}|\boldsymbol{q}} = \left\{\dot{\boldsymbol{q}} \in \mathbb{R}^{n} \mid \dot{\boldsymbol{q}} \in \left[\frac{\boldsymbol{q}_{m} - \boldsymbol{q}_{k}}{\Delta t}, \frac{\boldsymbol{q}_{M} - \boldsymbol{q}_{k}}{\Delta t}\right]\right\} \tag{3.18}$$

$$\mathcal{C}_{\dot{\boldsymbol{q}}|\ddot{\boldsymbol{q}}} = \{\dot{\boldsymbol{q}} \in \mathbb{R}^{n} \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{k} + \ddot{\boldsymbol{q}}_{m}\Delta t, \dot{\boldsymbol{q}}_{k} + \ddot{\boldsymbol{q}}_{M}\Delta t]\} \tag{3.19}$$

Note that the intersection of $\mathcal{C}_{\dot{\boldsymbol{q}}|\boldsymbol{q}}, \mathcal{C}_{\dot{\boldsymbol{q}}|\ddot{\boldsymbol{q}}}$ may create a feasibility problem [Rubrecht et al., 2010]. Then expressed as a linear function of $\dot{\boldsymbol{q}}$:

$$\begin{bmatrix} \frac{\boldsymbol{q}_{m} - \boldsymbol{q}_{k}}{\Delta t} \\ \dot{\boldsymbol{q}}_{k} + \ddot{\boldsymbol{q}}_{m}\Delta t \end{bmatrix} \leq \begin{bmatrix} \mathbf{I}_{n} \\ \mathbf{I}_{n} \end{bmatrix} \dot{\boldsymbol{q}} \leq \begin{bmatrix} \frac{\boldsymbol{q}_{M} - \boldsymbol{q}_{k}}{\Delta t} \\ \dot{\boldsymbol{q}}_{k} + \ddot{\boldsymbol{q}}_{M}\Delta t \end{bmatrix} \tag{3.20}$$

This section complements the QP proposition for a constrained optimization-based controller approach to pose tracking at the kinematic level. An overview integrating

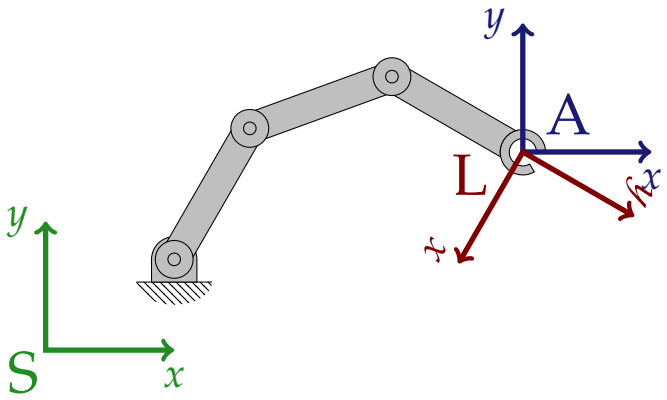


Figure 3.2: Depiction of three conventional frames used to describe quantities in robotics: S is the world or universal frame; L is a local or body frame; A corresponds to a combined frame, placed in the body but aligned with the world frame.

all the equations is shown in Appendix C. The following section presents some concepts regarding reformulating this controller in a different reference frame but does not further expand the current definition of the controller.

3.2.4 On the Solver Reference Frame ↑

This work exploits spatial algebra to express rigid body motion quantities in a concise manner. All these quantities require a frame in which they are expressed. This means that the same instant body motion can be represented, for instance in a local frame and thus be referred to as a body twist or in any other arbitrary reference frame.

Three widely used frames in the robotics literature are defined either with respect to the body in question or some arbitrary universal world frame. This work adopts the notation used in [Carpentier et al., 2019]. They are depicted in Figure 3.2:

- **World**: A universal or world frame, often referred to as frame S or W.
- **Local**: Also referred to as body frame and designated as L or B, represents a frame placed and aligned with the designated body. Notably, this frame presents a performance advantage in some algorithms (see page 95, Section 5.3 in [Featherstone, 2008]).
- **Local World-Aligned**: This frame is a combination of the previous two. It expresses a quantity with the same translation as a local frame but aligned with the world frame, it will be referred to as A. Some authors refer to it as the *moving-base frame* (see page 16 in [Traversaro et al., 2019]).

REMARK. The Adjoint action map for $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ for changing Twist Frames The adjoint action operator **Ad** presented on (2.50) for general Lie groups and briefly introduced in (2.38) for $\mathbb{SE}(3)$, is defined for the pose and orientation groups as:

$$\begin{aligned} \boldsymbol{\nu} \in \mathfrak{se}(3),\ \mathcal{X} = (\boldsymbol{R}, \boldsymbol{p}) \in \mathbb{SE}(3) \qquad & \boldsymbol{w} \in \mathfrak{so}(3),\ \boldsymbol{R} \in \mathbb{SO}(3) \\ \mathbf{Ad} : \mathbb{SE}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3) \qquad & \mathbf{Ad} : \mathbb{SO}(3) \times \mathfrak{so}(3) \to \mathfrak{so}(3) \\ \mathbf{Ad}_{\mathcal{X}}\, \boldsymbol{\nu} = \mathcal{X} \boldsymbol{\nu} \mathcal{X}^{-1} \qquad & \mathbf{Ad}_{\boldsymbol{R}}\, \boldsymbol{w} = \boldsymbol{R} \boldsymbol{w} \boldsymbol{R}^{-1} = \boldsymbol{R} \boldsymbol{w} \\ \mathbf{Ad}_{\mathcal{X}} = \begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \boldsymbol{p}^{\wedge} \boldsymbol{R} & \boldsymbol{R} \end{bmatrix} \qquad & \mathbf{Ad}_{\boldsymbol{R}} = \boldsymbol{R} \end{aligned} \tag{3.21}$$

Changing a Twist From the Local Frame to the World Frame Given some frame C described by a $\mathcal{X}_{WC} = (\boldsymbol{R}_{WC}, \boldsymbol{p})$ and some twist expressed in this frame ${}^{C}\boldsymbol{\nu}$, it is possible to change the twist coordinates frame with:

$$ {}^{W}\boldsymbol{\nu} = \mathbf{Ad}_{\mathcal{X}_{WC}}\, {}^{C}\boldsymbol{\nu} \tag{3.22}$$

$$\mathbf{Ad}_{\mathcal{X}_{WC}} = \begin{bmatrix} \boldsymbol{R}_{WC} & \mathbf{0} \\ \boldsymbol{p}^{\wedge} \boldsymbol{R}_{WC} & \boldsymbol{R}_{WC} \end{bmatrix} \qquad \boldsymbol{p}^{\wedge} = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}$$

From Local to Local World-Aligned For a given body B described by a pose $\mathcal{X}_{WB} = (\boldsymbol{R}_{WB}, \boldsymbol{p}_{WB})$, the transformation matrix goin from the world frame to the local world-aligned frame $\mathcal{X}_{WA}$ can be obtained as:

$$\mathcal{X}_{WA} = \mathcal{X}_{WB} \begin{bmatrix} \boldsymbol{R}_{WB}^{T} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \boldsymbol{p}_{WB} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.23}$$

The controller described in this section was formulated in the local or body frame. Special care was taken to specify the quantities to which this applies. Each frame offers some advantages either in intepretability of the equations and/or simplicity of the formulation. Some quantities are more naturally expressed in certain frames, like the error in a pose $\boldsymbol{e} = \log(\mathcal{X}_k^{-1} \mathcal{X}^r)$ is intrinsically expressed in the local frame; in the same way, the velocities in reference trajectories (a twist for a rigid body motion including the orientation) are often formulated either in the world or the local world-aligned frames, which also make it easier to interpret the motion in the workspace.

Following the method described in (3.22), the twists can be transported to other frames. For instance, for a body twist ${}^{B}\boldsymbol{\nu}$ on a body whose pose is described by a pose $\mathcal{X}$ (its roto-translation from the world frame), the same twist may be expressed in the world frame as:

$$ {}^{S}\boldsymbol{\nu} = \mathbf{Ad}_{\mathcal{X}}\, {}^{B}\boldsymbol{\nu} \tag{3.24}$$

Then (3.13) in the world frame becomes:

$$\begin{aligned} {}^{S}\boldsymbol{\nu}^{*} &= {}^{S}\boldsymbol{e} + {}^{S}\boldsymbol{\nu}^{r} \qquad && \boldsymbol{e}, \boldsymbol{\nu} \in \mathfrak{se}(3) \\ {}^{S}\boldsymbol{e} &= \boldsymbol{K}_{p}\mathbf{Ad}_{\mathcal{X}}\,{}^{B}\boldsymbol{e} && \mathcal{X} \in \mathbb{SE}(3) \\ {}^{B}\boldsymbol{e} &= \log(\mathcal{X}_{k}^{-1}\mathcal{X}^{r}) \end{aligned} \tag{3.25}$$

Note that a bigger proportional gain $\boldsymbol{K}_p$ produces a scaling on the resulting error ${}^{S}\boldsymbol{e}$ that forces the element out of the manifold. This is why a formulation in the local frame should be favored.

A similar method with (3.26) can be used to change the Jacobian reference frame and reformulate the cost function in (4.7).

REMARK. Changing the Jacobian frame As shown in (3.22), it is possible to change a twist frame with the adjoint operator. The same method can be applied to the Jacobian to obtain the resulting twist in the world frame:

$$\begin{aligned} {}^{S}\boldsymbol{\nu} &= {}^{S}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad && \boldsymbol{J} \in \mathbb{R}^{6\times n} \quad \boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^{n} \quad \boldsymbol{\nu} \in \mathfrak{se}(3) \\ \text{with } {}^{S}\boldsymbol{J}(\boldsymbol{q}) &= \mathbf{Ad}_{\mathcal{X}}\,{}^{B}\boldsymbol{J} \end{aligned} \tag{3.26}$$

where $\mathcal{X} \in \mathbb{SE}(3)$ expresses transformation from the world to the Jacobian frame.

3.3 Constrained Motion Generation with Model Predictive Control ↑

The focus of this section is to propose an Model Predictive Control (MPC) implementation on the $\mathbb{SE}(3)$ manifold, allowing pose control. It does so by revisiting some concepts from Chapter 2.

The section begins with an overview of some essential aspects of MPC. Starting from the description of a system in a state space representation, the discussion delves into formulating a linear system propagation for a receding horizon. This approach hinges on the assumption of constant linearized dynamics over a short time, which is essential for optimization applications over estimated future states. Then, the time discrete dynamic model is presented in a recursive matrix form over the entire horizon. This provides the foundation that allows formulating motion generation as a linear optimal control problem in a receding horizon that can be solved through a Quadratic Program (QP).

Finally, the section finishes by providing an overview on manifold optimization and on how to exploit these techniques for a linear MPC strategy formulated over the pose manifold for constrained motion generation.

For a more exhaustive explanation on optimal control and MPC fundamentals, the reader may consult [Rawlings et al., 2017]; [Borrelli et al., 2017] and [Rossiter, 2003], for a more practical approach.

3.3.1 Problem Statement: Motion Generation via MPC ↑

Constrained motion generation consists in the problem of generating viable motions that account for changing conditions in the dynamic environment of a collaborative scenario. Task-level Cartesian re-planning performing this action needs to overcome two main challenges:

- Optimal task and constraint transitions
- Computational complexity

The first challenge implies that the re-planner needs to be able to account for the effects of its current actions. Moreover, it needs to do so while transitioning between task objectives in an optimal manner and while maintaining a strict consideration of the constraints throughout time. While this is in itself an already complex problem, it builds up to the second challenge: it becomes intractable given the real-time urgency requirements of dynamic environment events.

In this context, linear MPC strikes a balance between continuously evaluating the environment conditions to achieve a task while achieving an acceptable optimality (compared to, for example, results of offline global planners). Indeed, receding horizon optimization is able to continuously re-consider the future actions of a task subject to work-space configurations that are evolving in time, producing some level of guarantee on on the viability of the generated motions [Rubrecht et al., 2010] and also offering smooth transition between the task objectives.

Another main advantage of MPC as motion generator is obtaining demonstrably trustworthy (as in constraint-respecting) motions [Kress-Gazit et al., 2021] which cannot formally be obtained from learning approaches [Matschek et al., 2023]. Indeed, MPC being a constrained optimization-based approach either provides a coherent solution or fails if it is non-existent. Conversely, there isn't formal approaches capable of guaranteeing that all solutions provided by learning algorithms respect the imposed constraints.

Recent contributions highlight the MPC features mentioned above for different applications [Bednarczyk et al., 2020]. Using MPC for motion generation in a similar way is proposed in [Gold et al., 2023] but ultimately they approach the technique from a servoing point of view, employing joint-level models [Oleinikov et al., 2021] and, quite often, use non-linear solvers, both of which increase the complexity of the formulations and, depending on the type of method employed, are not able to explicitly guarantee constraints [Kleff et al., 2021].

In contrast, this chapter proposal looks to streamline the control architecture from a task-centric point of view, which is why it focuses on point to point motions in the Cartesian space. The originality of the proposal goes beyond a philosophical point of view, as it complements this vision by proposing full pose motion generation: it exploits a mathematical formalism to also account for the orientation in the same MPC.

While the same Lie group formalism that enables optimization on the pose manifold has been employed for rigid body motion modeling [Terze et al., 2015] in the recent literature, they often resort to non-linear solvers [Knyazev et al., 2015] or were applied in unrelated applications [Chang et al., 2020]. In contrast, the current proposal

is to model this propagation in a linear way, allowing a quadratic cost function, necessary for linear MPC but most importantly, leading to a reasonable computation time.

Overall, the contribution is an efficient way to formulate a full body pose point to point constrained motion, obtaining horizon-based trajectories that can be updated continuously to account for the dynamic changes in the environment.

3.3.2 Linear Model Predictive Control ↑

This section presents a some fundamentals on modeling systems in a State Space (SS) representation, which enables state estimation and optimization over a horizon. This kind of formulation is crucial for the later parameterization of the Cartesian pose dynamics for constrained online motion generation.

A given non-linear system with dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ is parameterized with state $\boldsymbol{x} \in \mathbb{R}^d$ and input $\boldsymbol{u} \in \mathbb{R}^m$ vectors (where d is the size of the state and m the size of the input variables). The state vector is the smallest subset of system variables that contains all the information required to describe the system at some time instant. The input variable corresponds to the decision variable used to drive it to some desired or target state. The system can be discretized under Linear Time-Invariant (LTI) simplifying assumptions as (see Appendix B for more details):

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{B}\boldsymbol{u}_n \qquad \boldsymbol{A} \in \mathbb{R}^{d\times d},\ \boldsymbol{B} \in \mathbb{R}^{d\times m} \tag{3.27}$$ revisited ↓

$$\boldsymbol{y}_n = \boldsymbol{C}\boldsymbol{x}_n + \boldsymbol{D}\boldsymbol{u}_n \qquad \boldsymbol{C} \in \mathbb{R}^{r\times d},\ \boldsymbol{D} \in \mathbb{R}^{r\times m} \tag{3.28}$$

where $\boldsymbol{y}$ designates an r-sized output or observed variable and the matrices $\boldsymbol{C}, \boldsymbol{D}$ express its dependency on the current states and inputs; meanwhile, $\boldsymbol{A}, \boldsymbol{B}$ are the state and input matrices that propagate the future states and inputs through a linearization.

This kind of parameterization allows generically representing a system over a receding horizon. Note that for the general case of a non-linear system, all these matrices also depend on the state. This means that, for the interest of horizon estimation, the system needs to be linearized at some time instant t_n and then $\boldsymbol{A}_n$ can be assumed constant for a limited time window (for the unknown states that are to be estimated). The same assumption can be applied to $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$.

This last assumption is often exploited in the control field for predictive approaches. Assuming the constant linearized dynamics for a short time allows performing an optimization over the estimated future states. While the later estimations become less precise when the horizon grows (the constant linearized dynamics assumption is less true), this is not a problem if the linearization and the horizon recalculation are performed often (thus updating the linearization approximation to the new state).

In order to exploit this representation for state estimation in a horizon, one must first discretize it. Discretizing a time window in h time steps of duration T starting at time t_n (total duration hT) yields the discretized time steps inside the horizon, defined as $t_k = t_n + k\mathrm{T}$ for $k = 0, 1, \ldots, \mathrm{h}$ and one can define the discretized state and input at each time-step t_k within the horizon as $\boldsymbol{x}(t_k) = \boldsymbol{x}_{k|n}$, $\boldsymbol{u}(t_k) = \boldsymbol{u}_{k|n}$. The sub-index k indicates the horizon discretization (an estimated value in the horizon)

and the n refers to the current time of the system so that $\boldsymbol{x}(t_0) = \boldsymbol{x}_{0|n}$ refers to the initial state of the system at the start of the horizon.

One can formulate a linear receding horizon expression by extending (3.27) over all the states and inputs in this time period as:

$$\boldsymbol{x}_{k+1|n} = \boldsymbol{A}\boldsymbol{x}_{k|n} + \boldsymbol{B}\boldsymbol{u}_{k|n} \qquad \text{for } k = 0, 1, \ldots, \mathrm{h}-1 \tag{3.29}$$ revisited ↓

This time discrete linearized model expression can straightforwardly be written in a recursive matricial form over the whole horizon. Subsequent sections will drop the n subscript to simplify the notation, making $\boldsymbol{x}_{k|n}$ equivalent to $\boldsymbol{x}_k$.

Leveraging this linear propagation of the state in a horizon is central to the formulation of MPC as a constrained QP and can be packed into a single equation by first constructing a vector for the propagated states and inputs in a horizon:

$$\boldsymbol{X}_{0\ldots\mathrm{h}} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_{k+1} \\ \vdots \\ \boldsymbol{x}_{k+\mathrm{h}} \end{bmatrix} \qquad \boldsymbol{U}_{0\ldots\mathrm{h}-1} = \begin{bmatrix} \boldsymbol{u}_k \\ \boldsymbol{u}_{k+1} \\ \vdots \\ \boldsymbol{u}_{k+\mathrm{h}-1} \end{bmatrix} \tag{3.30}$$ revisited ↓

$$\overline{\boldsymbol{X}} = \boldsymbol{X}_{1\ldots\mathrm{h}} \qquad \underline{\boldsymbol{X}} = \boldsymbol{X}_{0\ldots\mathrm{h}-1} \qquad \underline{\boldsymbol{U}} = \boldsymbol{U}_{0\ldots\mathrm{h}-1}$$

This allows compressing (3.29) to a single algebraic formula:

$$\overline{\boldsymbol{X}} = \boldsymbol{A}'\underline{\boldsymbol{X}} + \boldsymbol{B}'\underline{\boldsymbol{U}} \tag{3.31}$$ revisited ↓

where $\overline{\boldsymbol{X}}$ and $\underline{\boldsymbol{U}}$ respectively correspond to the future states and control inputs of the system. Furthermore, $\boldsymbol{A}', \boldsymbol{B}'$ are matrices constructed from $\boldsymbol{A}, \boldsymbol{B}$ as:

$$\boldsymbol{A}' = \underbrace{\begin{bmatrix} \boldsymbol{A} & & & & \\ & \boldsymbol{A} & & & \\ & & \boldsymbol{A} & & \\ & & & \ddots & \\ & & & & \boldsymbol{A} \end{bmatrix}}_{\mathrm{h\ times}} \qquad \boldsymbol{B}' = \underbrace{\begin{bmatrix} \boldsymbol{B} & & & & \\ & \boldsymbol{B} & & & \\ & & \boldsymbol{B} & & \\ & & & \ddots & \\ & & & & \boldsymbol{B} \end{bmatrix}}_{\mathrm{h\ times}} \tag{3.32}$$

Finally, one can formulate the linear optimal control problem in a receding horizon as a QP with the following general quadratic cost function:

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}_\mathrm{h}} \frac{1}{2}\underline{\boldsymbol{U}}^T\boldsymbol{R}\underline{\boldsymbol{U}} + \frac{1}{2}\underline{\boldsymbol{X}}^T\boldsymbol{P}\underline{\boldsymbol{X}} + \frac{1}{2}{\boldsymbol{x}_\mathrm{h}}^T\boldsymbol{P}_T\boldsymbol{x}_\mathrm{h} \tag{3.33}$$

subject to:

$$\overline{\boldsymbol{X}} = \boldsymbol{A}'\underline{\boldsymbol{X}} + \boldsymbol{B}'\underline{\boldsymbol{U}} \tag{3.34}$$ revisited ↓

$$\boldsymbol{C}_{\boldsymbol{x}_m} \leq \boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{x}_k \leq \boldsymbol{C}_{\boldsymbol{x}_M} \tag{3.35}$$ revisited ↓

$$\boldsymbol{C}_{\boldsymbol{u}_m} \leq \boldsymbol{C}_{\boldsymbol{u}}\boldsymbol{u}_k \leq \boldsymbol{C}_{\boldsymbol{u}_M} \tag{3.36}$$ revisited ↓

where $\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{P}_T$ represent, respectively, weighting matrices for the input, state and the

terminal state. Meanwhile, $\boldsymbol{C_x}, \boldsymbol{C_u}$ allow formulating linear constraints[5] with respect to the state and input; $\boldsymbol{x}_m, \boldsymbol{x}_M$ and $\boldsymbol{u}_m, \boldsymbol{u}_M$ respectively designate the state and input bounds. In a general sense this cost function minimizes the distance towards the desired state (giving an explicit treatment to the terminal state in the horizon $\boldsymbol{x}_\text{h}$) while penalizing the effort (as in the control input $\boldsymbol{u}$).

The choice of values for the weighting matrices $\boldsymbol{R}, \boldsymbol{P}$ and specially the terminal state weight $\boldsymbol{P}_T$ have significant effects over the resulting form of the optimal solution. Some insights are provided in [Tan, 2016] (Section 5.2.3) on how varying weights can lead to under/over shooting. The same source also highlights the effects of varying the horizon length. In a broad sense, these parameters affect the convergence towards the solution of the optimization and thus influence the stability of the problem (specially the terminal state) [Mayne et al., 2000].

Note that (3.34) expresses the link between the propagated states, leveraging the equality constraints of a QP solver. This establishes the optimization scheme as a direct single shooting method [Bard, 1974].

Equation (3.33) under constraints (3.34) to (3.36) constitutes the general linear MPC formulation as a constrained quadratic optimization [Alexis et al., 2012]; [Bemporad et al., 2002].

3.3.3 Optimization on the Pose Manifold ↑

Using a receding horizon optimization scheme to generate a pose trajectory requires formulating the control problem of predicting the end-effector pose given a horizon of control inputs, here considered at the velocity level. This section quickly revisits concepts from Section 2.4 to re-introduce the integration scheme of the pose dynamics in the tangent space for MPC.

Given some system with an initial pose $\mathcal{X}_i \in \mathbb{SE}(3)$ subject to a body twist $\boldsymbol{\nu}(t) \in \mathfrak{se}(3)$, its discretized pose trajectory $\mathcal{X}_k = \mathcal{X}(t)\big|_{t=k\Delta t} \in \mathbb{SE}(3)$ can be described as:

$$\begin{aligned} \mathcal{X}_k &= \mathcal{X}_i e^{\boldsymbol{\nu}_0 \Delta t} e^{\boldsymbol{\nu}_1 \Delta t} \dots e^{\boldsymbol{\nu}_{k-1} \Delta t} \\ &= \mathcal{X}_i \textstyle\prod_{n=0}^{k-1} e^{\boldsymbol{\nu}_n \Delta t} \end{aligned} \tag{3.37}$$

revisited ↑

This relation can be used incrementally to compute the desired twists at each time instant given some tracking error [Lynch et al., 2017]. Nevertheless, the pose trajectory in a horizon remains a non-linear function with respect to the twist. In other words, the system evolves as the successive Lie group actions, acting on its state (pose) to glide through the $\mathbb{SE}(3)$ manifold at each time point, as shown in (3.37). Hence, to be solved at a decent control rate for reactivity, the MPC should be expressed in a linear form like in (3.31) (Section 3.3.2).

This section presents how to link both forms while providing a rough explanation on the conceptual steps for optimizations on manifolds employed in this work (for more details see [Forster et al., 2015]; [Saccon et al., 2013]).

Consider the optimization problem of a variable belonging to a smooth manifold $\mathcal{X} \in \mathcal{M}$; hence there exists a map $\boldsymbol{f} : \mathcal{M} \to \mathbb{R}^n$ (with its inverse denoted $\boldsymbol{f}^{-1}$), transporting elements from the solution space into an optimization space $\boldsymbol{f} : \mathcal{X} \to \boldsymbol{x}$,

[5] Linear constraints can be used to formulate convex polytopes as half planes.

allowing the definition of the optimization problem with cost function C as:

$$\boldsymbol{x}^{\text{opt}} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}\, C(\boldsymbol{x}) \tag{3.38}$$

$$\mathcal{X}^{\text{opt}} = \boldsymbol{f}^{-1}(\boldsymbol{x}^{\text{opt}}) \tag{3.39}$$

With a conveniently chosen lifted space (see Section 2.4.1), optimizing can take the form of a quadratic cost function like for linear MPC as in (3.33) and be solved as a QP.

This re-parameterization of the original space into an optimization space that offers some desirable properties is often referred to as *lifting* or *pushing* while the inverse operation is a *retract* or *pull*. Refer to Figure 2.7 for a visual representation. Then the solution can be recovered through a retraction (as in (3.38)), resulting in a lift, optimize and retract scheme. The lift and retract functions are respectively denoted with $\boldsymbol{\psi}, \boldsymbol{\psi}^{-1}$.

As explained in Chapter 2, the $\mathbb{SE}(3)$ manifold can be mapped to $\mathfrak{se}(3)$, through a differentiable map:

$$\begin{array}{rccc|rcc} \log: & \mathbb{SE}(3) & \to & \mathfrak{se}(3) & \mathcal{X} & \to & \boldsymbol{\xi} \\ \exp: & \mathfrak{se}(3) & \to & \mathbb{SE}(3) & \boldsymbol{\xi} & \to & \mathcal{X} \end{array} \tag{3.40}$$

revisited ↑ ↓

The log function[6] relates the Lie group $\mathbb{SE}(3)$ to its Lie algebra $\mathfrak{se}(3)$, its tangent space, at the origin. As explained in Section 2.4.1, the bijectivity properties are maintained[7] for $|\boldsymbol{\phi}| < \pi$ (where $\boldsymbol{\phi}$ is the angular component of $\boldsymbol{\xi}$).

The tangent space behaves like an euclidean space for optimization purposes: it allows representing a pose as its exponential coordinates (see Section 2.2.2.3), a 6-vector instead of an homogeneous transformation matrix.

Analyzing (3.37) for a single step yields (2.48), the relation between an infinitesimal increment in pose (body twist) and the resulting pose:

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\Delta t \boldsymbol{\nu}_k} \tag{3.41}$$

revisited ↑ ↓

This is a first-order approximation, analogous to the one presented in [Forster et al., 2015] for $\mathbb{SO}(3)$ and [Solà et al., 2018] (for more general manifolds).

As introduced in Section 2.4.2, it can be reformulated the tangent space as:

$$\boldsymbol{\xi}_{k+1} = \log(e^{\boldsymbol{\xi}_k} e^{\Delta t \boldsymbol{\nu}_k}) \approx \boldsymbol{\xi}_k + \text{dlog}_{\boldsymbol{\xi}_k} \Delta t \boldsymbol{\nu}_k \tag{3.42}$$

revisited ↑ ↓

where $\text{dlog}_{\boldsymbol{\xi}_k}$ is the right-trivialized log map derivative[8] at $\boldsymbol{\xi}_k$, that relates additive increments in the tangent space at the origin to the right-multiplied increments in

[6] Throughout this work, the log and exp operations adopts the vectors forms. This could be considered an abuse of notation but one can go back and forth between the matrix and vector forms through the Vee map and its inverse (see Sections 2.2.1.3 and 2.2.2.2) or [Solà et al., 2018]; [Torres Alberto et al., 2022a].

[7] The one to one correspondence is maintained under the conditions detailed in [Forster et al., 2015] and [Torres Alberto et al., 2022a] (Section 2.1). Otherwise, the exp map is surjective.

[8] Given the adopted vector form of the tangent space in this work, the $\text{dlog}_{\boldsymbol{\xi}_k}$ takes the form of a 6×6 matrix.

the pose space. In layman's terms[9], this relates the body twist with an increment in the logarithm of the pose. This constitutes the *push-forward* operation that enables optimization in a manifold [Boumal, 2023], extending the concepts introduced in this section.

Finally, the missing link that allows the present work to formulate a pose propagation in a linear form (as in (3.29)) for a receding horizon is shown in (3.42), as required for the linear MPC in (3.33).

3.3.4 Geodesic Path in the Tangent Space ↑

As introduced in Section 3.3.3, this work aims at finding the pose and input trajectory that will drive the system to some target pose $\mathcal{X}_t \in \mathbb{SE}(3)$. In order to embed this in the MPC cost function, a distance metric formulated in the tangent space is necessary. In fact, $\mathfrak{se}(3)$ is a Riemannian manifold [Lee, 2019] equipped with the geodesic distance metric known as *Log-Euclidean* ([Jayasumana et al., 2014], Table 1):

$$e_k = || \underbrace{\boldsymbol{\xi}_k}_{\log(\mathcal{X}_k)} - \underbrace{\boldsymbol{\xi}_t}_{\log(\mathcal{X}_t)} ||_2^2 \tag{3.43}$$

The last challenge to overcome is that there exist multiple paths that connect two extreme poses $\mathcal{X}_i, \mathcal{X}_t \in \mathbb{SE}(3)$. Given some $\alpha(t) \in [0, 1]$ the shortest path (geodesic) [Novelia et al., 2015]; [Marthinsen, 1999] can be interpolated by using the path that passes through the origin and computing its "distance" in tangent space $\boldsymbol{\delta}$:

$$\mathcal{X}_k = \mathcal{X}_i e^{\alpha_k \boldsymbol{\delta}} \qquad \boldsymbol{\delta} = \log(\mathcal{X}_i^{-1} \mathcal{X}_t) \tag{3.44}$$

which is depicted in Figure 3.3. One way to ensure that the trajectory optimization in the lifted space corresponds to (3.44) is through the choice of the lift function (and consequently also the retract function):

$$\boldsymbol{\psi}(\boldsymbol{x}) = \log(\mathcal{X}_l^{-1} \boldsymbol{x}) \qquad \boldsymbol{\psi}^{-1}(x) = \mathcal{X}_l \exp(x) \tag{3.45}$$
$$\mathcal{X}_l = \mathcal{X}_t$$

revisited ↓

where $\mathcal{X}_l$ denotes the lift pose. This way, the optimization can be "directed" from $\boldsymbol{\psi}(\mathcal{X}_i) \to \boldsymbol{\psi}(\mathcal{X}_t)$, which yields the geodesic. The choice of $\mathcal{X}_l = \mathcal{X}_t$ stems from the application: it allows the optimization to always converge to the origin in the tangent space $\boldsymbol{\xi}_I$ because: $\boldsymbol{\psi}(\mathcal{X}_t) = \boldsymbol{\xi}_I$. In fact, the lift function in (3.45) implies that $\boldsymbol{\psi}$ is in the tangent space at $\mathcal{X}_l^{-1}$.

3.3.5 Task-Space Linear MPC in the Tangent Space ↑

This section reformulates the problem introduced in Section 3.3.3 using the same notation as in (3.30).

The first step consists in defining the state and input variables:

$$\boldsymbol{x}_k = \boldsymbol{\psi}(\mathcal{X}_k) \qquad \boldsymbol{x}_t = \boldsymbol{\psi}(\mathcal{X}_t) \qquad \boldsymbol{u}_k = \boldsymbol{\nu}_k \tag{3.46}$$

revisited ↓

[9] Plain English (layman's terms) is a mode of writing or speaking the English language intended to be easy to understand regardless of one's familiarity with a given topic. https://en.wikipedia.org/wiki/Plain_English

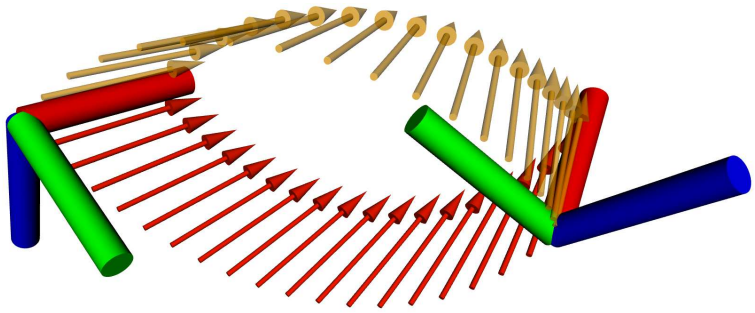


Figure 3.3: There exist multiple paths that connect some initial and target poses $\mathcal{X}_i, \mathcal{X}_t \in \mathbb{SE}(3)$. The image shows the successive directions of the red axis of a frame (3-axis) moving through the path. It illustrates the geodesic (in red) and an alternative path (in yellow). The yellow path can be interpolated in the tangent space and then transformed to poses as $\mathcal{X}_k = e^{(1-\alpha_k)\boldsymbol{\xi}_0 + \alpha_k \boldsymbol{\xi}_t}$ for some $\alpha(t) \in [0, 1]$.

where $\boldsymbol{x}_t$ is lifted the target state; $\boldsymbol{x}_k = \boldsymbol{\xi}_k$ is the lifted state and $\boldsymbol{\nu}_k$ is the body twist.

Then it is possible to extend the distance metric in the tangent space (3.43) over a horizon to use it as a cost function:

$$\overline{\boldsymbol{X}}^*, \underline{\boldsymbol{U}}^* = \arg\min_{\boldsymbol{x},\boldsymbol{u}} ||\overline{\boldsymbol{X}} - \overline{\boldsymbol{X}}_t||_2^2 + \gamma ||\underline{\boldsymbol{U}}||_2^2 \tag{3.47}$$

revisited ↓

$$\overline{\boldsymbol{X}}_t^T = [\underbrace{\boldsymbol{\psi}(\mathcal{X}_t)^T \dots \boldsymbol{\psi}(\mathcal{X}_t)^T}_{\text{h}-1 \text{ times}}]^T \tag{3.48}$$

revisited ↓

$$\begin{aligned} \text{s.t:} \quad \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \quad \text{for } k = 0, \dots, \text{h}-1 \\ \boldsymbol{u}_m &\leq \boldsymbol{u}_k \leq \boldsymbol{u}_M \quad \text{for } k = 0, \dots, \text{h}-1 \\ \dot{\boldsymbol{u}}_m &\leq \tfrac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{\Delta t} \leq \dot{\boldsymbol{u}}_M \quad \text{for } k = 0, \dots, \text{h}-2 \end{aligned} \tag{3.49}$$

where γ designates a weighting term for the input throughout the horizon and $\boldsymbol{\psi}(\mathcal{X}_t) = \boldsymbol{\xi}_t$ corresponds to the lifted target pose while $\overline{\boldsymbol{X}}_t$ is its vector version throughout the horizon. The input constraints (for simplicity in the formulation, often defined as $\boldsymbol{u}_M = -\boldsymbol{u}_m$), establish the limits for the body twist and spatial body acceleration (through a finite difference).

Note that the definition in (3.48) is defined such that the MPC is in charge of generating the motion that minimizes the distance towards the target state while respecting the constraints. This is part of the reference-less approach adopted.

Furthermore, (3.47) shows the standard linear MPC formulation in (3.33). The state can be propagated throughout the horizon as in (3.29) by using the relationship shown in (3.42), which yields:

$$\boldsymbol{A} = \mathbf{I}_6 \qquad \boldsymbol{B} = \text{dlog}_{\boldsymbol{\xi}_i} \Delta_\text{h} t \tag{3.50}$$

revisited ↓

where $\boldsymbol{A}$ is an identity matrix and $\boldsymbol{B}$ reflects the tangent space linearization at the initial robot pose, as $\boldsymbol{\xi}_i = \boldsymbol{\psi}(\mathcal{X}_i)$. This constitutes the central relation that enables tangent space linear MPC.

Figure 3.4: Safe workspaces in a collaborative environment cannot be defined with static zones as it depends on the robot-operator states and tasks.

Finally, in order to recover the desired poses $\mathcal{X}_k^{\text{des}}$ from $\overline{\boldsymbol{X}}^*$:

$$\mathcal{X}_k^{\text{des}} = \boldsymbol{\psi}^{-1}(\boldsymbol{x}_k^*) \tag{3.51}$$

This constitutes the linear MPC formulation that allows full pose and constrained trajectory generation in an online manner.

3.4 Experimental Validation: The Case of Velocity Modulation ↑

This section presents an implementation of the proposed Model Predictive Control (MPC) for a collaborative robotics scenario that emphasizes the safety features of the approach.

Safety in robotics has always been a concern, but modern industrial trends have specially exacerbated this in search for better integration of robots and humans in working environments. For collaborative robots, this is addressed through norms or technical specifications.

One of the most recent standards materialized in a Technical Specification (TS) [ISO/TS-15066, 2016]. Although innovative, it still tends to be a somewhat coarse method that frequently results in impractical configurations and installations that are challenging to certify. This leads to collaborative robots in the industry that are mostly used as standard robots, i.e. behind cages or inefficient and empty safety zones.

Addressing safety in an efficient way when considering shared workspaces is a complex problem that requires an online evaluation of multiple moving bodies. Nevertheless, just like with the TS, this problem is often simplified as a "workspace safety state machine" through discrete zones [Amaya-Mejía et al., 2022]; [Long et al., 2018].

Some recent promising methods focus on considering safety from an energetic level [Joseph et al., 2018a]; [Raiola et al., 2018]; [Meguenani et al., 2015], which, although more complex in implementation, allow for more flexible solutions.

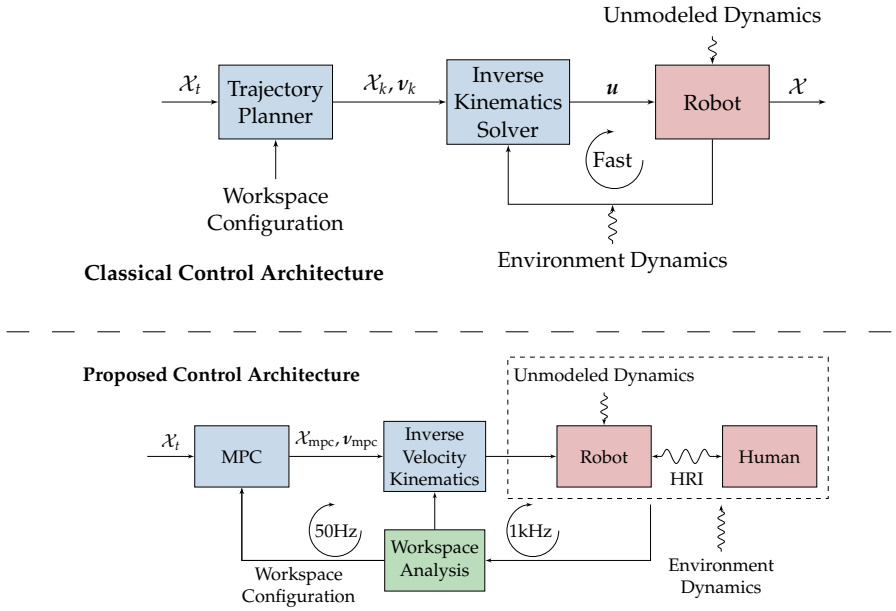


Figure 3.5: Classical vs proposed control architecture. (Top) Constrained offline planning employed in classical control approaches cannot deal with dynamic environments that evolve over time. (Bottom) Proposes a cascade of closed-loop composed of task-space online trajectory re-planning over a receding horizon (MPC block) and a high-frequency lower-level task-to-joint-space controller (inverse velocity kinematics block). A third block (Workspace Analysis) feeds the changing environment information to the MPC to achieve safety-awareness. revisited ↑ ↓

Figure 3.4 aims to depict this situation: a dynamically safe area for the robot to evolve can be computed based on the state of both the operator and the robot as well as on their near future motions. However, the task of accurately calculating the potential area and making real-time control decisions based on this calculation, which is essential for facilitating close encounters as the one in Figure 3.4, remains an open research challenge [Long et al., 2017]; [Mainprice et al., 2016]; [Eckhoff et al., 2022].

In its most general form, it boils down to solving a global optimal control problem online that determines at each time a control trajectory that satisfactorily combines safety and efficiency.

Despite some research effort in this direction, this very general problem is far from being solved in a generic way and goes far beyond the scope of this work. Yet, a clear required feature emerges from the search for optimality: the need to predict the effect of control actions over a time horizon to account for the continuous changes to workspace constraints.

This last requirements is directly addressed by the control architecture proposed in Section 3.1.1 and this experiment showcases one implementation of this feature.

3.4.1 Implementing Online Re-planning ↑

Beyond safety, a close dynamic interaction between a human operator and a robot may also imply on-the-fly re-definitions of the task to be achieved by the robot. This cannot be solved by "classical control architectures" (such as the one depicted in the top of Figure 3.5) which generally consider offline planning solutions, incapable of

reacting to dynamic events implied in the considered collaborative scenarios. Instead one needs to consider online re-planning at the task level.

MPC is capable of combining both online re-planning and prediction of the control action over a horizon. It offers a convenient framework to optimally and interactively exploit allowed motion constraints while respecting the robot capacities and safety constraints.

As a consequence, this chapters implements a hierarchical cascade of closed-loops controllers in a similar way to [Lober et al., 2020], combining both online trajectory re-planning over a receding horizon and task space-control in two stages: a high-level safety-aware operational space MPC and a high-frequency lower-level task-to-joint-space controller.

An architecture schematic is shown at the bottom of Figure 3.5. A third block is in charge of processing the environment information and translating this into safety-aware constraints and tasks that can be used during re-planning. The task-space re-planning is performed by an MPC described in Section 3.3.5 and the high-frequency joint-space controller (an inverse velocity kinematics solver) is described in Section 3.2.

More specifically, the focus of this section lies in the constrained online re-planning: continuously finding the optimal pose and twist trajectory that will drive a system from an initial pose to some target pose, subject to velocity and acceleration limits that account for safety and the real robot capacities. While this is not a fully novel problem, predictive algorithms in the literature:

- often use non-linear solvers [Kleff et al., 2021]; [Sathya et al., 2020] which may not be appropriate for frequent re-planning;
- are formulated at the joint-space level [Nicolis et al., 2020]; [Incremona et al., 2017] which is not ideal when considering task-space specification as a central feature;
- mostly disregard orientation.

By leveraging the relation between the pose space (the Lie Group $\mathbb{SE}(3)$) and its tangents spaces (in the Lie algebras $\mathfrak{se}(3)$) [Saccon et al., 2013]; [Torres Alberto et al., 2022a]; [Forster et al., 2015], this work proposes a Linear MPC capable of estimating the evolution of the position and orientation with fast Quadratic Program (QP) solvers.

Figure 3.6 shows an example horizon computed by the MPC towards one of the target poses. It exemplifies that the MPC only plans over a limited time horizon (starting from the left). As the robot moves towards the target pose, the MPC computes a new trajectory for each horizon according to the new robot state (middle image), until it reaches the target pose (on the right).

3.4.2 Numerical Simulation ↑

To showcase the result of a task-space MPC employed for online re-planning, a numerical simulation was constructed for a simple Cartesian motion along one of the

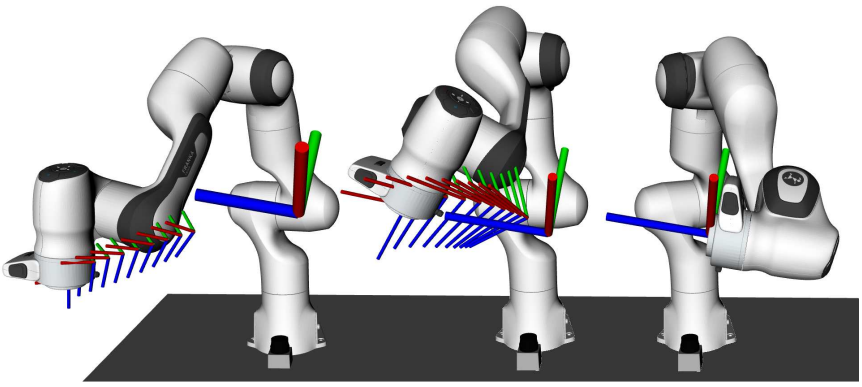


Figure 3.6: The planned horizon in time. The online MPC progressively "discovers" a path from the initial pose (on the left) towards the target pose (on the right). Given the MPC optimizes over a receding time window, the optimizing horizon does not cover the complete path until it is close enough to the target pose.

axis of the reference frame attached to the robot base (x in this case). While the experience might seem trivial, it emulates the operation of the MPC by re-planning at a constant rate. It was executed using the full implementation of the algorithm even though only the axis in question actually performs a motion (as intended by the design of the experiment).

The example output of the MPC control loop is shown in Figure 3.7. The prediction horizon is 75ms composed of 15 time steps of 5ms each. The update rate of the MPC is 50Hz thus implying an interpolation of the first computed time-step as the Cartesian space state is updated at 1kHz. For an intuitive comparison, the resulting trajectory is compared to a time optimal trapezoidal acceleration profile computed once using Ruckig [Berscheid et al., 2021b].

Unlike the globally planned trajectory which includes jerk bounds, the current formulation of this MPC does not embed jerk limitations, yielding some large acceleration variations. Yet, the resulting profile is very similar and demonstrates the ability of MPC to obtain quasi-time-optimal trajectories while conferring online adaptation capabilities to the overall architecture. Moreover, while Ruckig is in practice a decoupled multi-dofs planner, the proposed approach truly accounts for 3D motions in both position and orientation.

3.4.3 Experimental Scenario & Safety Constraints ↑

The setup was constructed with a collaborative scenario in mind (shown in Figure 3.8): the robot is placed on a table and requested to cycle over a series of poses on repetition. Meanwhile, to depict a safety concern, the operator approaches the robot and the robot operating velocity is modulated according to the distance. The objective of the following experiment is to show the controller ability to

1. find a trajectory towards arbitrary target poses while respecting constraints;

2. adapt to changing constraints online.

To do so, a 7-DoF Panda robot from Franka Emika is requested to move to four different targets. These targets are not known in advance by the MPC but rather given on-the-fly.

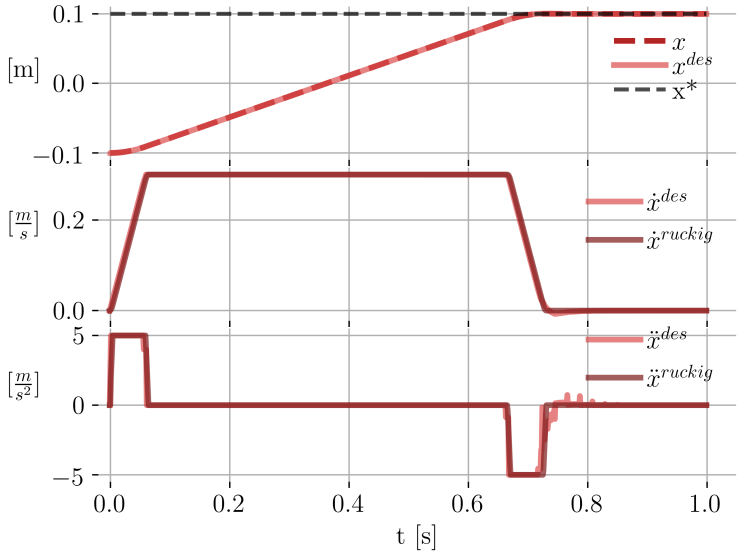


Figure 3.7: Numerical simulation of the replanning controller for 1-Degree of Freedom (DoF). This example shows the result of employing the MPC to replanify at 50Hz and apply the resulting twists for a Cartesian motion along one of the axis (x from the frame attached to the robot base). To compare optimality, the output is compared to the jerk-bounded algorithm in [Berscheid et al., 2021b].

For this experiment, a cascade controller as shown in Figure 3.5 is implemented. This schema also shows the working frequency of each controller and the feedback flow. The MPC implemented uses OSQP [Stellato et al., 2020] as a QP solver. The joint velocity controller used for this experiments is available online[10] and uses qpOASES [Ferreau et al., 2014]. The robot model is computed using the Pinocchio library [Carpentier et al., 2019]. The robot control architecture is implemented using the Robot Operating System (ROS) framework and ran in real-time at a frequency of 1kHz using the franka_ros library.

[10] The code used for the controller is hosted at gitlab.inria.fr/auctus-team/components/robots/panda/panda_qp_control. That said, future versions of the controller will be integrated in the qontrol library at gitlab.inria.fr/auctus-team/components/control/qontrol.

video/link

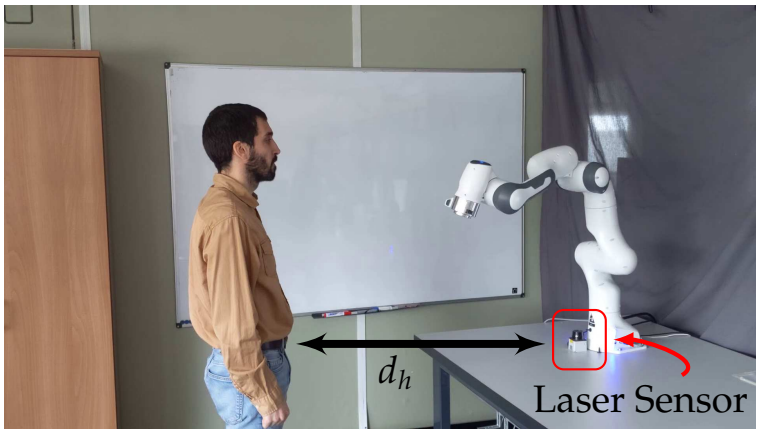


Figure 3.8: Velocity modulation experimental setup. The photo shows the setup. The robot is placed on a table to perform a task provided on-the-fly. The distance to the operator d_h used for safety-aware velocity modulation is measured via the laser sensor.

When solving the MPC problem for a new target pose, computation times vary a lot depending on the amount of steps and constraints in the horizon. Moreover, the first solving step (cold start) always takes more time. For a typical horizon of $h = 10, \Delta t_h = 50ms$ OSQP solves the problem in under 10ms. The solution of the previous QP is used as a first guess for the next one (hot start). By assuming that the optimal solution between two resolutions closely resembles the previous one, the resolution process can be accelerated. This leads to computation times under 1ms with a warm start.

Safety constraints

To showcase the application of the MPC with safety in mind, it was employed for velocity modulation. The velocity constraints are adapted according to the distance between the operator and robot and provided to the MPC online. Notice that this implies that the constraints in (3.49) are changing over time.

The constraint modulation in this scenario is chosen for its simplicity but it showcases how to potentially integrate arbitrarily complex safety behaviours as velocity modulation functions, such as, for example, dynamic risk assesment strategies based on probabilistic scenarios [Z. Liu et al., 2020]; [Mayyas et al., 2020], the robot stopping distance [Joseph et al., 2020] or damage-based maximum velocities [Mansfeld et al., 2017].

The distance between the end-effector and the human is calculated as $d_h = ||\boldsymbol{p}_{ee} - \boldsymbol{p}_h||^2$ (where $\boldsymbol{p}_{ee}, \boldsymbol{p}_h$ respectively refer to the end-effector and the human position). A simple piece-wise function is employed for each component of $\boldsymbol{u}_M$ designated $u_{M,i}$ for $i = 1, \ldots, 6$ (to modulate both the linear and angular velocities):

$$u_{M,i}(d_h) = \begin{cases} V_m & d_h \leq d_m \\ \frac{d_h - d_m}{d_M - d_m} V_M & d_m < d_h < d_M \\ V_M & d_h \geq d_M \end{cases} \tag{3.52}$$

that has user defined extreme values for the minimal and maximal distances d_m, d_M and velocities V_m, V_M; all greater than zero.

This function ensures a linear relation between maximum velocity and the distance inside the distance interval; outside of this range $\boldsymbol{u}_M$ is assigned either the minimal or the maximum velocity. The minimum velocity $\boldsymbol{V}_m$ is chosen to be close to zero to completely stop the robot motion when the distance is below the threshold d_m.

The robot Cartesian velocity is constrained using (3.52). The distance between the operator and the robot is determined by a Hokuyo Laser range finder placed at the robot base, as shown in Figures 3.4 and 3.8 (in the real setup); and Figure 3.6 (in the simulation). The following values are used to modulate the robot velocity according to (3.52):

$$\begin{array}{lll} V_{m,\text{lin}} = 0.01 m.s^{-1} & V_{m,\text{ang}} = 0.01 s^{-1} & d_m = 0.2m \\ V_{M,\text{lin}} = 1 m.s^{-1} & V_{M,\text{ang}} = 1.5 s^{-1} & d_M = 1m \end{array} \tag{3.53}$$

In an industrial scenario, these values would be determined by the type of human-robot collaboration and technical specification used, such as the ISO TS 15066.

3.4.4 Results ↑

This section shows the results from periodically requesting the robot to move to a different pose. Figure 3.9 presents the experiments performed on the Panda robot, including

1. the error between the current robot pose and the target pose;
2. the body twist used during the trajectory, as well as the maximum available velocities.

Result 1) shows the error between the current position $\boldsymbol{p}_k \in \mathbb{R}^3$ and orientation of the robot $\boldsymbol{R}_k \in \mathbb{SO}(3)$ and the target ones. There is a spike every time a new pose is requested. It is important to highlight that this is not the resulting tracking error of following the trajectory planned with the MPC but rather the "distance" between the current pose and the objective.

The following formula was used to compute this error:

$$\boldsymbol{e} = \boldsymbol{p}_k - \boldsymbol{p}_t \qquad \boldsymbol{\phi} = \log(\boldsymbol{R}_k^{-1}\boldsymbol{R}_t) \tag{3.54}$$

where $\boldsymbol{p}_t, \boldsymbol{R}_t$ denote the target position and orientation, respectively; $\boldsymbol{e} \in \mathbb{R}^3$ reflects the position distance and $\boldsymbol{\phi} \in \mathfrak{so}(3)$ is the orientation error.

For result 2), the velocity curves show the body twist $\boldsymbol{\nu} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T]^T$ as well as the available velocity (in a red shade) that is being scaled by the distance between the robot end-effector and the human.

The main highlight of these results is that the MPC is able to modulate velocities according to constraints changing in real-time achieving the two main objectives of this work:

1. when the human is far, the algorithm is able to saturate the velocity, maximizing the robot movement performance;
2. while the human approaches, the robot movement is handicapped until it no longer moves.

A subsidiary result of this MPC formulation is that it allows changing the robot target position on the fly, overriding the current target. The resulting robot motion keeps being smooth since it respects an acceleration-bounded profile as shown in Figure 3.9. More details can be observed in the attached video (see Figure 3.10).

3.5 Conclusion ↑

This Chapter proposes an efficient linear Model Predictive Control (MPC) approach that can deal with the online planning of SE(3) motion in task space. The main features of the proposed control approach lie in its ability to generate optimal Cartesian motion which dynamically accounts both for targets updated on-the-fly and evolving motion constraints. This receding horizon approach endows the robot with the ability to interactively adapt to its environment. More particularly the safety of a human operator sharing its workspace with the robot can be accounted for through the sensor-based adaptation of maximum velocity constraints.

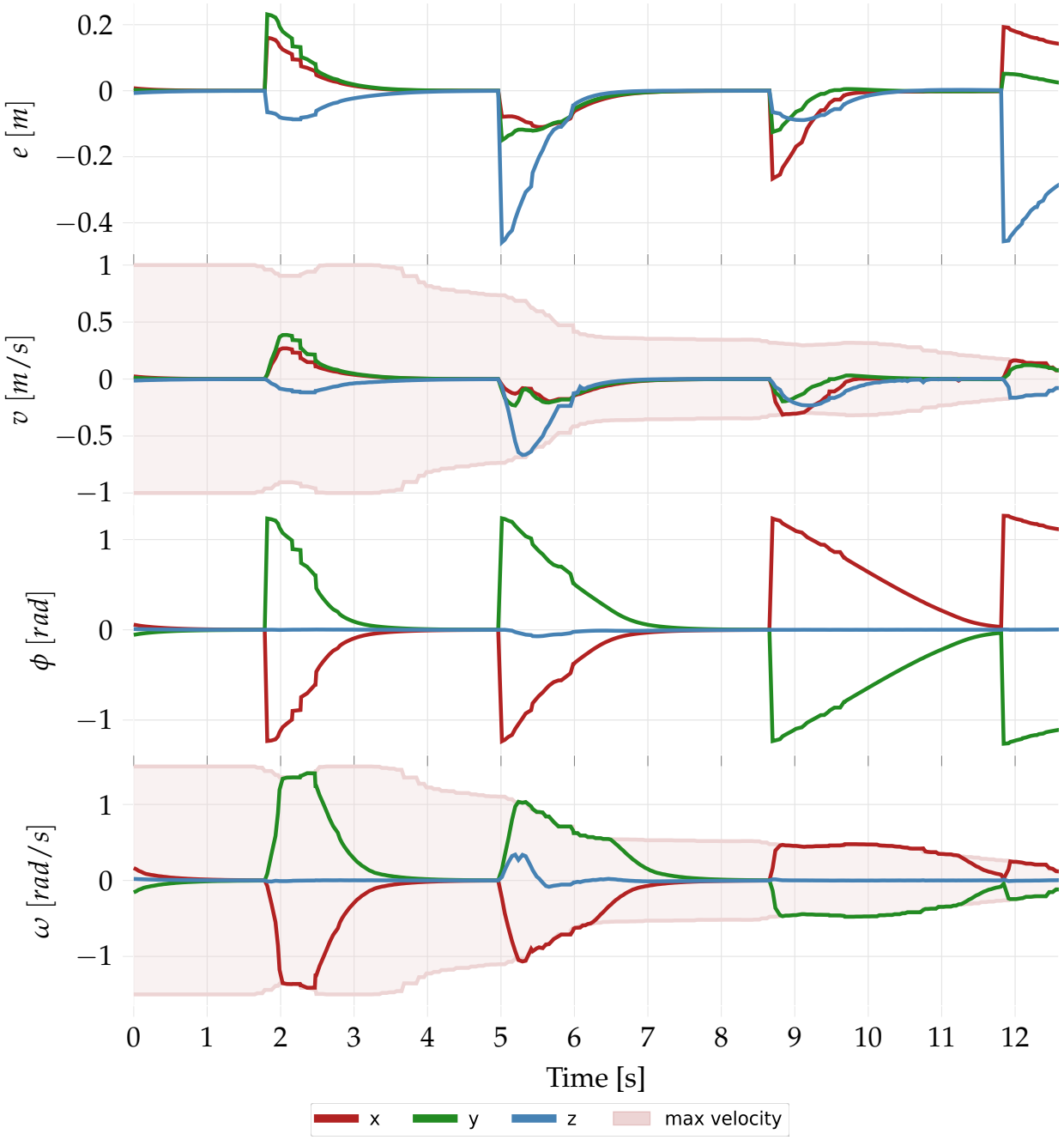


Figure 3.9: Shows the result of requesting the robot to move between 4 preset poses. While the robot moves, a laser sensor is used to capture the minimal distance to a human and modulate the maximum velocity (shown in red shade). The error curves show the distance between the current position and orientation $(\boldsymbol{p}_k, \boldsymbol{R}_k)$ and the preset target poses $(\boldsymbol{p}_t, \boldsymbol{R}_t)$. The linear error is $\boldsymbol{e} = \boldsymbol{p}_t - \boldsymbol{p}_k$, while the angular error is computed as $\boldsymbol{\phi} = \log(\boldsymbol{R}_k^{-1}\boldsymbol{R}_t)$. The MPC (T=300ms, h = 10, $\Delta_h = 30ms$, $f_{MPC} = 50Hz$) modulates the body twist $\boldsymbol{\nu} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T]^T$ respecting the safe limits imposed even when they become so low that the robot stops moving.

The chapter proposes a velocity-based MPC formulation for acceleration-bounded trajectory planning. This allows formulating constraints up to the acceleration level but presents two big limitations: the resulting trajectories is not smooth on the acceleration level; the MPC cost function is very sensitive to the regularisation task.

The next chapters of this work will focus on a acceleration-based formulation that allows incorporating jerk bounds to obtain smoother trajectories. At the same time, they focus on formulating the linear constraints in the MPC for the Cartesian pose state, to limit the effective position and orientation space.

Furthermore, the current experiment was implemented with explicit equality constraints in the Quadratic Program (QP) solver. This adds a computation complexity to the solver that results in slower convergence. The following chapters show an improved version that employs a compressed matricial, speeding up resolution times.

Figure 3.10: QR code for the video of the velocity modulation experiment. On PDF file readers, it is also a clickable hyperlink.

Chapter 4

Linear Model Predictive Control on SE(3): Acceleration-based

Robots require the ability to autonomously, continuously and smoothly react to unexpected online changes in the task definition and in the environment, especially those cohabited with humans. To react to these changes, the task, from the current state up to the finish, must instantly be re-considered. This implies a prohibitive re-computation cost.

This Chapter proposes a modular control architecture based on Model Predictive Control, that offers a good compromise between optimally achieving the task and the required computation time, by only re-considering the near future. This framework offers a generic way to formulate task-related objectives and constraints that dissociates the planning from the execution, which depends mainly on the robot dynamics. The proposal exploits a linear formalization of the Model Predictive Control (MPC) in $\mathbb{SE}(3)$ to implement this architecture in a high-frequency closed-loop controller, achieving task re-planning at the control rate.

The pertinence of the proposed control architecture is demonstrated using experiments with the Franka Emika robots in scenarios where the task to be achieved is modified on the fly.

Contents

4.1 Introduction ↑

The following sections present the rationale behind the task adaptation in interactive and collaborative scenarios from a dynamics modeling point of view. The proposal is to make a slight adjustment to the control architecture presented in Chapter 3, exploiting a more efficient mathematical formulation of the constrained motion re-planning problem.

Section 4.2 focuses on the modeling and control assumptions of the robot. In order to start considering motion dynamics for collaborative tasks, these aspects must also be embedded in the robot model. With this objective in mind, the kinematics-based approach presented before is extended to control a robot through the joint torques. This model is then used for task-space tracking via a Quadratic Program (QP).

Furthermore, Section 4.3 focuses on the mathematical formalisms that allow continuous point to point pose trajectory re-planning. So far, Chapter 2 presented the algebraic formulation that allows for pose integration in the manifold; afterwards, Chapter 3 employed these principles to obtain a linear form that can be employed in QP optimization in a Model Predictive Control (MPC) control scheme. These allows constrained point to point motion generation over a gliding time horizon. This chapter extends this to higher order motions to further improve the motion smoothness, achieve higher frequency re-planning and extend the strategy to be able to include the dynamics of collaborative tasks.

Finally, Section 4.4 presents an experimental validation of the control scheme. It introduces a real-world implementation of the architecture along with the experimental setup. The experiment focuses on achieving smooth task transitions to showcase the online adaptation in unplanned interactive tasks.

Collaborative robotics has increasingly gained interest over the last decades. From an application point of view, it crystallises early expectations [Makinson, 1971]; [J Edward Colgate et al., 2003] around the improvement of human working conditions [Maurice et al., 2017]; [Schoose et al., 2022]. From a research stand point, it raises several essential questions, for example regarding the prediction of human motion [Mainprice et al., 2015] or the underlying cognitive principles behind the concept of collaboration [Chen et al., 2021].

Among these research questions, the more general one of controlling robots in dynamic environments[1] is a recurring yet unsolved problem. A large amount of literature has been dedicated to this problem but the most recent works in the domain of collaborative robotics have focused on human avoidance [Eckhoff et al., 2022]; [Merckaert et al., 2022]. Another essential feature when considering dynamic environments is the general ability to autonomously, continuously and smoothly adapt to unplanned task updates. While Chapter 3 focuses on a control approach for continuous task adaptation, this Chapter proposes an extension that more comprehensively tackles the problem to achieve smoother reactive behaviors from the robot.

[1] as opposed to static ones where robots are physically separated from anything that could provoke a change in either the task or the surrounding objects during execution

4.1.1 Robots in Changing Environments for Dynamic Tasks ↑

Given the existence of mechanisms for updating the robot current state in relation to its environment and task objective (i.e. perception) in real-time, the most straightforward approach to grant robots with the ability to adapt is through real-time adjustments in their movement planning to reach the desired goal [Palleschi et al., 2021].

Actually, while it is essential to find a viable path towards the target and continuously update it in order to have any possibility of accomplishing the task, the reconsideration of the temporal aspects of the motion from the beginning to the end is debatable.

Indeed, it is unlikely for the entire pre-computed trajectory to be successfully executed in the presence of unforeseen changes to the task. Additionally, despite advancements in motion planning [Coleman et al., 2014] and optimal trajectory computation [Pham et al., 2018], the online adjustment of both the path and the time profile throughout the entire motion remains an extremely challenging problem [Palleschi et al., 2021]. This difficulty becomes even more pronounced when the planning approach involves computing a trajectory for control inputs.

Re-planning over a receding horizon presents a viable alternative [Ghazaei Ardakani et al., 2019] to ensure online computational feasibility of the control problem while striving for optimal task performance. This approach represents a compromise between methods that compute an optimal trajectory once offline and those that adapt to environmental changes without fully revising the whole trajectory [Joseph et al., 2020].

Another consequence of robots operating in dynamic environments is the inability to verify in advance whether all constraints will be satisfied, including those associated with the task and inherent robot capabilities. In contrast, on-line re-planning provides some level of guarantee that the generated motion will not lead constraint violations although it is unlikely to help in the event of unforeseen events occurring during motion execution.

As a consequence, situations may arise where emergency stops become the sole recourse to prevent accidents. To circumvent such control exceptions, controllers need to be designed as optimization problems that explicitly incorporate constraints in the resolution [M. Liu et al., 2015].

MPC appears as a good candidate due to its capability to perform optimal motion re-planning based on the current system state (i.e. in closed-loop fashion) while taking into account some constraints. In fact reasoning over a receding horizon not only achieves local optimality but also implicitly ensures compatibility among the set of constraints. This approach is more likely to succeed compared to methods that attempt to address constraint compatibility offline [Rubrecht et al., 2010]; [Meguenani, 2017]; [Prete, 2018].

Assuming that MPC can keep up with the robot control rate [D. Kouzoupis et al., 2015]; [Dimitris Kouzoupis et al., 2018], one can rely on a control architecture where the joint velocity or torque control input horizon is re-computed at each control time step and fed to the robot, based on a given operational space target.

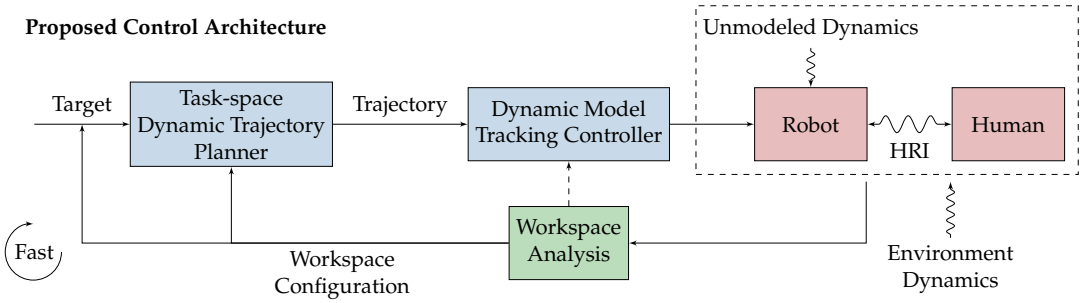


Figure 4.1: The proposed modular control architecture is based on a cascade loop composed of: 1) a task-space Model Predictive Controller (MPC) for online re-planning; 2) a constrained Task Space Inverse Dynamics solver to follow the planned trajectory. The whole control loop runs synchronously at the control rate. revisited ↑ ↓

While Chapter 3 proposed a kinematic approach to the online re-planning problematic, the interactive nature of collaborative tasks enforce the need for forces to be considered in the architecture. Yet, to do so, this approach must be extended to account for the dynamics.

This is proposed in [Kleff et al., 2021] where MPC is performed through Differential Dynamic Programming which can be solved efficiently [Mastalli et al., 2020] at each control step over a receding horizon.

This approach couples two underlying sub-problems and solves them together: 1) the computation of an operational space trajectory towards the target and 2) the resolution of the operational to joint space mapping. Even though constraints are implemented in the cost function and cannot be formally guaranteed, this approach presents the advantage of considering joint and actuation constraints directly at the MPC problem level. As a consequence, except for the joint torque servoing performed at the actuation level, no inner control loop is required.

Yet, alternative approaches can be considered where an MPC is solved at the operational space level to incrementally generate a feasible task-space trajectory given some task-related constraints [Faroni et al., 2019]; [Eckhoff et al., 2022]. The output of the MPC is then fed to a joint level controller which can, for example, solve for the joint torques at each control instant using a constrained optimization approach [M. Liu et al., 2015]. This approach does not allow to account for the joint level limits of the robot at the MPC level. However, it provides a modular architecture closing the loop on the current operational state of the robot, independently from the nature of the controller provided with a given robot. The later approach is retained in this work.

4.1.2 Control Architecture ↑

The scenario presented above builds upon the one introduced in Chapter 3. Namely, it considers the same collaborative work cells wherein robots and humans share the workspace and the events to which the robot must react. It also shares the same task-centric design philosophy that allows decoupling task planning from task execution, achieving higher modularity from the controller point of view as well as from the components choice: a workspace planner can be utilized with multiple robots. Moreover, re-purposing a robot does not require re-designing a task planner. In contrast, joint-space planning is coupled to the robot architecture and, in the same

way, task planning in the joint space can hardly be straightforwardly reused when changing robot architecture.

Beyond that, the architecture must also consider the interactive nature of collaborative tasks. In fact, these kind of tasks often involve physical interactions with the environment and human operators. In such case, the robot must not only re-actively adapt to its surroundings but it must also properly act upon it in a measured and coherent way.

In fact, the inter-activeness of these scenarios imply that the robot is capable of performing tasks that consider the acting forces and torques to achieve an objective. Two main challenges arise from this scenario that cannot be properly addressed by the architecture proposed in Chapter 3:

- For one, the robot must consider the dynamics of the interactions
- Secondly, the closed feedback loop needs to be highly reactive

Indeed, the first challenge accounts for the fact that in order to properly consider the acting forces between a robot and its environment (and / or humans), the control models must be extended for higher order motions, i.e. dynamics. This offers a way to better account for inertial effects while also relating the motion to the actual acting quantities.

Furthermore, the asynchronous design choice of the proposed architecture in Chapter 3 is a workaround to the heavy computing requirements of motion planning. Nevertheless, when controlling a motion at the dynamics level, a faster and more reactive controller is desired.

This directly connects to the second challenge: the computational demands of considering more complex models that account for dynamics. In fact, the faster the control architecture the better: the closed feedback loop is able to better adapt to what is happening at a higher frequency.

The control architecture drafted here and summarily described on Figure 4.1 consists of two main components:

- The first one is a linear MPC, formulated at the operational space level. It generates an optimal trajectory in $\mathbb{SE}(3)$ over a receding horizon, given a reference target and constraints on the admissible motion, both provided on the fly. These constraints can be both related to the real joint space motion capabilities of the robot and to restrictions relative to the performed task.
- The second one computes the robot control input leading to the execution of the planned trajectory while respecting the joint-level constraints using an inverse dynamics solver.

These two components are described hereafter. In all, the architecture enables the robot to re-plan a trajectory at the control frequency of the robot while accounting for pose constraints on the end-effector up to the jerk order.

4.2 Acceleration-based Task-space Tracking ↑

The previous chapter presented an inverse kinematics solver as a Quadratic Program (QP) to provide task-space tracking at the kinematic level. This section extends the previous strategy to formulate task-space tracking with an inverse dynamics solver.

In order to control the robot at the acceleration level, one must first extend the kinematic model introduced in Section 3.2.1 to consider the joint accelerations and torques. This is done in Section 4.2.1. Then, a proper torque-based QP cost function for tracking is presented in Section 4.2.2. Finally, the constraints are formulated in Section 4.2.3.

Given the interpolated first step of the trajectory computed by the Model Predictive Control (MPC), task space inverse dynamics has to be performed to compute the input joint torque for the robot.

Feeding the inverse dynamics solver with the desired acceleration computed by the MPC may sound tempting. Nevertheless, the feedback provided by the closed-loop MPC is, partly due to the interpolated output, not efficient enough to properly reject tracking errors related to the inaccuracies in the model of the robot.

These inaccuracies are mostly related to dry friction at the joint level, imperfectly rejected by the lower-level torque control loop in most robots. As a consequence, a Proportional Derivative (PD) controller including a feed-forward term in acceleration is used to compute a corrected desired acceleration.

4.2.1 Dynamic Robot Model ↑

The dynamical model of a manipulator is one that describes the interaction between the joint torques and external contact forces to produce a joint acceleration $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ (see the Forward Dynamics Section 8.5 of [Lynch et al., 2017] for the joint-space formulation and Section 3.3 of [Siciliano et al., 2008], for the task-space one):

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{\tau} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f} - \boldsymbol{B}(\boldsymbol{q}, \dot{\boldsymbol{q}})) \qquad \boldsymbol{B}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) \tag{4.1}$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ denotes the torques generated by each joint; $\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}$ corresponds to the joint torques resulting of applying an external wrench $\boldsymbol{f} \in \mathbb{R}^6$; $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ corresponds to the joint space inertia matrix; and the $\boldsymbol{B}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ term is referred to as the *bias forces*, comprised of the effects of the Coriolis and centrifugal forces in $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the gravity in $\boldsymbol{G}(\boldsymbol{q})$.

A dynamics-based task-space control formulation requires linking the task-space acceleration (see Section 8.5.1 of [Siciliano et al., 2008]) with the joint accelerations from (4.1). The link between both is visible once the time derivative of the twist $\dot{\boldsymbol{\nu}} \in \mathfrak{se}(3)$ in (3.6) is considered:

$${}^W\boldsymbol{\nu} = {}^W\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.2}$$ revisited ↑ ↓

$${}^W\dot{\boldsymbol{\nu}} = {}^W\boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + {}^W\dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{4.3}$$

and now $\ddot{\boldsymbol{q}}$ can be replaced with (4.1). The kinematic model assumes actuation limits on the joint configuration bounds and joint velocities:

$$\begin{aligned} \mathcal{Q} &= \{\boldsymbol{q} \in \mathbb{R}^n \mid \boldsymbol{q}_m \leq \boldsymbol{q} \leq \boldsymbol{q}_M\} \\ \dot{\mathcal{Q}} &= \{\dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \dot{\boldsymbol{q}}_m \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_M\} \end{aligned} \tag{4.4}$$

revisited ↑ ↓

These need to be extended to account for the joint torques:

$$\mathcal{T} = \{\boldsymbol{\tau} \in \mathbb{R}^n \mid \boldsymbol{\tau}_m \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_M\} \tag{4.5}$$

which can be even further extended in case one needs to consider the torque variation limits:

$$\dot{\mathcal{T}} = \{\dot{\boldsymbol{\tau}} \in \mathbb{R}^n \mid \dot{\boldsymbol{\tau}}_m \leq \dot{\boldsymbol{\tau}} \leq \dot{\boldsymbol{\tau}}_M\} \tag{4.6}$$

This would effectively impose varying limits on the joint jerks; as can be seen by taking the time derivative of (4.1).

4.2.2 Inverse Dynamics Controller as a Quadratic Program ↑

Task-space tracking focuses on the problem of computing the optimal control input to achieve a certain end-effector motion. The kinematics-based strategy solves for the joint velocities whereas a dynamics-based solver calculates the joint accelerations. The first approach focuses on the geometric view of the problem, ignoring the required forces or torques. In contrast, the inverse dynamics strategy can formulate the accelerations as a function of the joint torques and external forces acting on the end-effector. This accounts for the system dynamic properties such as the mass, inertia, external forces, etc.

The desired motion for an end-effector is parameterized with reference pose trajectory $\mathcal{X}^r(t) \in \mathbb{SE}(3)$. Once discretized, this yields a reference pose, twist and twist derivative (also referred to as spatial acceleration) to be achieved at the current time step k: $\mathcal{X}_k^r = \mathcal{X}^r(t_k), \boldsymbol{\nu}_k^r = \boldsymbol{\nu}^r(t_k), \dot{\boldsymbol{\nu}}_k^r = \dot{\boldsymbol{\nu}}^r(t_k) \in \mathfrak{se}(3)$ (with $t_k = k\Delta t$ and $k \in \mathbb{N}$).

The controller presented here is similar to the one in [Joseph et al., 2018b]; [Prete, 2018]. The objective is to devise a strategy to control manipulators based on joint torque. For this, one needs to link the desired Cartesian spatial acceleration from the reference trajectory with the joint torques. First, the spatial acceleration can be expressed as a function of the joint accelerations with (4.3) using the Jacobian and its derivative. Then, by exploiting the robot dynamics model (see Section 4.2.1), one can formulate the joint accelerations as a function of the joint torques. This enables a QP formulation (as in (3.2)) of the task-space tracking problem, just like the kinematic formulation in (3.8):

$$\boldsymbol{\tau}^{\text{opt}} = \arg\min_{\boldsymbol{\tau}} \quad ||{}^B\dot{\boldsymbol{\nu}}(\boldsymbol{\tau}) - {}^B\dot{\boldsymbol{\nu}}_k^*||^2 + f_{\text{reg}}(\boldsymbol{\tau}) \tag{4.7}$$

$${}^B\dot{\boldsymbol{\nu}}(\boldsymbol{\tau}) = {}^B\boldsymbol{J}(\boldsymbol{q}_k)\ddot{\boldsymbol{q}}(\boldsymbol{\tau}) + {}^B\dot{\boldsymbol{J}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \tag{4.8}$$

where the sub-index k indicates measured states of the robot (for instance, the Jacobian $\boldsymbol{J}$ is computed at the measured robot configuration $\boldsymbol{q}_k$); and f_{reg} defines a regularisation task used to optimize over the redundant Degree of Freedom (DoF)s

of the robot and ${}^B\dot{\boldsymbol{\nu}}^*$ denotes the desired twist derivative expressed in the body frame (spatial acceleration).

Additionally, the term $\boldsymbol{J}(\boldsymbol{q}_k)\dot{\boldsymbol{q}}_k$ can be computed from the current measured state and be treated as a constant bias in the optimization. Thus, the spatial acceleration $\dot{\boldsymbol{\nu}}(\boldsymbol{\tau})$ is a linear function of the optimization variable (the joint torque $\boldsymbol{\tau}$).

If the model of the robot was perfectly known (i.e. $\dot{\boldsymbol{\nu}}(\boldsymbol{\tau})$ is calculated without error), then the spatial acceleration would be the one corresponding to the reference trajectory $\dot{\boldsymbol{\nu}}^* = \dot{\boldsymbol{\nu}}^r$. However, there are always unmodeled dynamics and a proper strategy to deal with those is to use a PD servoing by defining a desired spatial acceleration:

$$ {}^B\dot{\boldsymbol{\nu}}^* = \boldsymbol{K}_p\,{}^B\boldsymbol{e} + \boldsymbol{K}_d(\boldsymbol{\nu}_k^r - \boldsymbol{\nu}_k) + {}^B\dot{\boldsymbol{\nu}}^r \tag{4.9} $$

$$ {}^B\boldsymbol{e} = \log(\mathcal{X}_k^{-1}\mathcal{X}_k^r) \tag{4.10} $$

where the reference spatial acceleration ${}^B\dot{\boldsymbol{\nu}}^r$ acts as a feed forward term.

Finally, the regularisation task f_{reg} is defined in such a way that it uniquely defines an optimal solution even on redundant robots:

$$ f_{\text{reg}}(\boldsymbol{\tau}) = w_{\text{reg}}||\boldsymbol{M}^{-1}(\boldsymbol{\tau}_{\text{reg}} - \boldsymbol{\tau})||_2^2 \tag{4.11} $$

with $w_{\text{reg}} \ll 1$, ensuring it will not affect the main tracking task; and $\boldsymbol{\tau}_{\text{reg}}$ is defined depending on the desired use of the redundant DoFs. The inclusion of the inertia matrix $\boldsymbol{M}$ helps modulate the importance of each joint according to the configuration.

For instance, if it was used to keep the robot configuration as far as possible (i.e. without affecting the tracking performance) from the joint limits while damping the joint velocity. It is defined as:

$$ \boldsymbol{\tau}_{\text{reg}} = \boldsymbol{K}_{\boldsymbol{\tau}}(\boldsymbol{q}_{\text{reg}} - \boldsymbol{q}) - \boldsymbol{K}_{\dot{\boldsymbol{q}}}\dot{\boldsymbol{q}} \tag{4.12} $$

$$ \boldsymbol{q}_{\text{reg}} = \frac{(\boldsymbol{q}_M - \boldsymbol{q}_m)}{2} \tag{4.13} $$

where $\boldsymbol{K}_{\boldsymbol{\tau}}, \boldsymbol{K}_{\dot{\boldsymbol{q}}}$ are positive proportional terms and $\boldsymbol{q}_M, \boldsymbol{q}_m$ respectively denote the upper and lower joint bounds.

4.2.3 Inverse Dynamics Constraints ↑

This section presents a way to integrate the dynamics-level actuation limits into the linear constraints of a QP, for a robot controlled through its joint torques.

First of all, the intrinsic joint torque constraints from (4.3) can simply be embedded as:

$$ \boldsymbol{\tau}_m \leq \mathbf{I}_n\boldsymbol{\tau} \leq \boldsymbol{\tau}_M \tag{4.14} $$

where $\mathbf{I}_n$ denotes an n-sized identity matrix.

The kinematic constraints (3.7) need to be reformulated as a function of the control

input. To achieve this, one can perform a taylor expansion of future joint configuration and velocities (assuming the current ones can be measured, i.e. $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k \in \mathbb{R}^n$):

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \Delta t \dot{\boldsymbol{q}}_k + \frac{\Delta t^2}{2} \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \tag{4.15}$$

$$\dot{\boldsymbol{q}}_{k+1} = \dot{\boldsymbol{q}}_k + \Delta t \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \tag{4.16}$$

where $\ddot{\boldsymbol{q}}$ is explicitely dependent on the decision variable, the joint torques $\boldsymbol{\tau}$, as seen in (4.1). Then, knowing that $\boldsymbol{q}_{k+1} \in \mathcal{Q}$ and $\dot{\boldsymbol{q}}_{k+1} \in \dot{\mathcal{Q}}$, one can formulate the kinematic constraints as:

$$\begin{aligned} \boldsymbol{q}_M &\geq \boldsymbol{q}_k + \Delta t \dot{\boldsymbol{q}}_k + \tfrac{\Delta t^2}{2} \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) & \dot{\boldsymbol{q}}_M &\geq \dot{\boldsymbol{q}}_k + \Delta t \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \\ \boldsymbol{q}_m &\leq \boldsymbol{q}_k + \Delta t \dot{\boldsymbol{q}}_k + \tfrac{\Delta t^2}{2} \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) & \dot{\boldsymbol{q}}_m &\leq \dot{\boldsymbol{q}}_k + \Delta t \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \end{aligned} \tag{4.17}$$

where Δt is chosen large enough to cover a few control periods, as a safe margin to avoid entering the limits [Rubrecht et al., 2010]. These equations allow formulating the constraints as a function of the joint accelerations:

$$\mathcal{C}_{\tau|q} = \left\{ \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \mathbb{R}^n \mid \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \left[2\frac{(\boldsymbol{q}_m - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k \Delta t)}{\Delta t^2}, 2\frac{(\boldsymbol{q}_M - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k \Delta t)}{\Delta t^2} \right] \right\} \tag{4.18}$$

$$\mathcal{C}_{\tau|\dot{q}} = \left\{ \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \mathbb{R}^n \mid \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \left[\frac{(\dot{\boldsymbol{q}}_m - \dot{\boldsymbol{q}}_k)}{\Delta t}, \frac{(\dot{\boldsymbol{q}}_M - \dot{\boldsymbol{q}}_k)}{\Delta t} \right] \right\} \tag{4.19}$$

which is possible thanks to the definition of $\ddot{\boldsymbol{q}}$ in (4.1), under the assumption that there are no external forces applied ($\boldsymbol{f} = 0$).

Finally, the torque rate constraints may be integrated as (4.6):

$$\mathcal{C}_{\tau|\dot{\tau}} = \{ \boldsymbol{\tau} \in \mathbb{R}^n \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_k + \Delta t \dot{\boldsymbol{\tau}}_m, \boldsymbol{\tau}_k + \Delta t \dot{\boldsymbol{\tau}}_M] \} \tag{4.20}$$

4.3 MPC-based motion re-planning over a receding horizon ↑

The focus of this section is to propose an Model Predictive Control (MPC) implementation on the $\mathbb{SE}(3)$ manifold, allowing pose control. In contrast to the kinematics-based approach of Chapter 3, this one looks to extend the definitions to a dynamics point of view. It does so by revisiting some concepts from Chapter 2 and briefly revisiting formulations from Section 3.3.2 to extend them to the case of jerk-bounded (acceleration-based) point to point pose motion generation.

Given a d-sized state and m-sized input system, it can be described with a discrete time linear system (indexed with n) as

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{B}\boldsymbol{u}_n \qquad \boldsymbol{A} \in \mathbb{R}^{d \times d},\ \boldsymbol{B} \in \mathbb{R}^{d \times m} \tag{4.21}$$

revisited ↑ ↓

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent its state and input matrices whereas $\boldsymbol{x}$ and $\boldsymbol{u}$ are the state and input vectors. Smooth point to point pose motion generation can be achieved by generating trajectories at the acceleration level. Choosing the control input at the task level to be the acceleration, allows predicting the evolution of the end effector (system) state — pose and twist — over a short period of time to generate a jerk-bounded motion.

Discretizing a receding time window in h time steps of duration T starting at time t_n (total duration hT) yields the discretized time steps inside the horizon, defined as $t_k = t_n + k\mathrm{T}$ for $k = 0, 1, \ldots, \mathrm{h}$ and one can obtain $\boldsymbol{x}(t_k) = \boldsymbol{x}_{k|n}$, $\boldsymbol{u}(t_k) = \boldsymbol{u}_{k|n}$.

One can formulate a linear receding horizon expression as:

$$\boldsymbol{x}_{k+1|n} = \boldsymbol{A}\boldsymbol{x}_{k|n} + \boldsymbol{B}\boldsymbol{u}_{k|n} \qquad \text{for } k = 0, 1, \ldots, \mathrm{h}-1 \tag{4.22}$$ revisited ↑ ↓

Subsequent sections drop the n subscript to simplify the notation, making $\boldsymbol{x}_{k|n}$ equivalent to $\boldsymbol{x}_k$.

This linear time discrete dynamic model expression can straightforwardly be written in a recursive matricial form over the whole horizon:

$$\overline{\boldsymbol{X}} = \boldsymbol{A}'\underline{\boldsymbol{X}} + \boldsymbol{B}'\underline{\boldsymbol{U}} \tag{4.23}$$ revisited ↑

where $\boldsymbol{A}', \boldsymbol{B}'$ are matrices constructed from $\boldsymbol{A}, \boldsymbol{B}$ via the recursive propagation of (3.29) over the time horizon. Moreover, $\overline{\boldsymbol{X}}$ and $\underline{\boldsymbol{U}}$ respectively correspond to the future states and control inputs of the system, as described in:

$$\boldsymbol{X}_{0\ldots\mathrm{h}} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_{k+1} \\ \vdots \\ \boldsymbol{x}_{k+\mathrm{h}} \end{bmatrix} \qquad \boldsymbol{U}_{0\ldots\mathrm{h}-1} = \begin{bmatrix} \boldsymbol{u}_k \\ \boldsymbol{u}_{k+1} \\ \vdots \\ \boldsymbol{u}_{k+\mathrm{h}-1} \end{bmatrix} \tag{4.24}$$ revisited ↑

$$\overline{\boldsymbol{X}} = \boldsymbol{X}_{1\ldots\mathrm{h}} \qquad \underline{\boldsymbol{X}} = \boldsymbol{X}_{0\ldots\mathrm{h}-1} \qquad \underline{\boldsymbol{U}} = \boldsymbol{U}_{0\ldots\mathrm{h}-1}$$

From these propagated linear dynamics, one can formulate the problem of finding a trajectory over a gliding time horizon. In a generic way, this concerns the problem of generating a constrained and time discrete trajectory over the system state. Point to point trajectories in a system state minimize the distance between the starting and desired states of a system. This problem can be mathematically expressed as a the linear optimal control problem over a receding horizon as a Quadratic Program (QP) with the following cost function:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \left\|\overline{\boldsymbol{X}}_t - \overline{\boldsymbol{X}}\right\|_{\boldsymbol{P}}^2 + \left\|\underline{\boldsymbol{U}}\right\|_{\boldsymbol{R}}^2 \tag{4.25}$$ revisited ↑ ↓

subject to:

$$\overline{\boldsymbol{X}} = \boldsymbol{A}'\underline{\boldsymbol{X}} + \boldsymbol{B}'\underline{\boldsymbol{U}} \tag{4.26}$$ revisited ↑

$$\boldsymbol{C}_{\boldsymbol{x}_m} \leq \boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{x}_k \leq \boldsymbol{C}_{\boldsymbol{x}_M} \tag{4.27}$$ revisited ↑ ↓

$$\boldsymbol{C}_{\boldsymbol{u}_m} \leq \boldsymbol{C}_{\boldsymbol{u}}\boldsymbol{u}_k \leq \boldsymbol{C}_{\boldsymbol{u}_M} \tag{4.28}$$ revisited ↑ ↓

where $\overline{\boldsymbol{X}}_t$ is the target state $\boldsymbol{x}_t$ vectorized over the horizon. This is an important detail and one of the original features of the proposed approach: no pre-interpolation or a priori trajectory generation is required. The resulting motion is solely shaped by the considered constraints on the admissible task space motions. This implicitly yields a time optimal trajectory over the horizon.

Regarding $\boldsymbol{P}, \boldsymbol{R}$, they respectively represent symmetric positive definite weighting

matrices for the states and inputs throughout the horizon. These are the same weighing matrices shown in (3.33), where $\boldsymbol{P}_T$ explicitly expresses the importance of the terminal state, whereas here it is embedded within $\boldsymbol{P}$. The same logic applies to the control input weights, which are embedded in $\boldsymbol{R}$:

$$\boldsymbol{P} = \underbrace{\begin{bmatrix} \boldsymbol{P}_k & & & & \\ & \boldsymbol{P}_{k+1} & & & \\ & & \ddots & & \\ & & & \boldsymbol{P}_{k+h-1} & \\ & & & & \boldsymbol{P}_T \end{bmatrix}}_{\text{h times}} \quad \boldsymbol{R} = \underbrace{\begin{bmatrix} \boldsymbol{R}_k & & & & \\ & \boldsymbol{R}_{k+1} & & & \\ & & \ddots & & \\ & & & \boldsymbol{R}_{k+h-1} & \\ & & & & \boldsymbol{R}_{k+h} \end{bmatrix}}_{\text{h times}} \tag{4.29}$$

For a typical usage, the weights employed remain constant throughout the horizon and these matrices are defined as:

$$\boldsymbol{P}_{k+i} = \mathbf{I}_{d\times d} \qquad \text{for } i = 0, \ldots, \mathrm{h}-1 \tag{4.30}$$

$$\boldsymbol{P}_T = \mathbf{I}_{d\times d}\, w_T \tag{4.31}$$

$$\boldsymbol{R}_{k+i} = \mathbf{I}_{d\times d}\, \gamma \qquad \text{for } i = 0, \ldots, \mathrm{h} \tag{4.32}$$

where w_T, γ denote the weight for the terminal state and the control inputs in the horizon, while the weight of the remaining states is set to 1. The control input weight γ shown in (3.47) is the vector version that contains the weight values for each step (assuming they are not the same).

Finally, $\boldsymbol{C_x}, \boldsymbol{C_u}$ allow formulating linear constraints with respect to the state and input, while $\boldsymbol{x}_m, \boldsymbol{x}_M$ and $\boldsymbol{u}_m, \boldsymbol{u}_M$ respectively designate the state and input bounds.

This problem needs to be continuously evaluated to find a new horizon trajectory respecting the constraints. The starting and target states $\boldsymbol{X}_0, \boldsymbol{X}_t$ ultimately close the re-planning feedback loop and add the reactivity needed in an interactive scenario.

For the problem at hand, to generate a constrained position and orientation trajectory, the system state needs to contain the pose and twist of the end effector. Meanwhile, to obtain a smooth motion, the control input is considered at the acceleration level. The goal at each discrete step of the horizon is to reach the final target with a null velocity. A greater importance (weight) is given to the last step of the horizon to favour convergence towards the target with $\boldsymbol{P}_T$.

Equation (3.47) under constraints (3.31) and (3.34) constitutes the general linear MPC formulation as a QP. It progressively provides a trajectory to track for the end-effector as the optimization window evolves.

The linearity of this MPC formulation is a mandatory feature for the corresponding optimization problem to be solved at each control time step. This problem needs to overcome two main challenges: it needs to be computationally tractable to be solved in coherence with the real time requirements of collaborative scenarios; furthermore, writing this problem in a linear fashion, given that the Cartesian pose of the end-effector $\mathcal{X}$ belongs to $\mathbb{SE}(3)$, is not trivial. The following sections propose a way to tackle both problems.

4.3.1 Reduced Formulation ↑

The cost function (3.47) minimizes the distance between the current and the desired states over the state and control input variables. However, due to the equality condition over a horizon shown in (3.31), the states of the system throughout the time horizon are completely determined by the control input in that same horizon (for a fixed starting state). This section shows how the same problem may be reduced to an optimization over the control input, effectively reducing the size of the optimizing variables, to obtain a less computationally intensive problem that can be solved at the control rate of the robot.

Assuming constant linearized dynamics for a short period of time allows performing an optimization over the estimated future states. The assumption is explicitely considered in (3.29). While subsequent estimations become progressively less precise (due to the decreasing validity of the constant linearized dynamics assumption), this does not pose an issue if the linearization and the horizon recalculation are performed frequently.

However, the key consideration is that this assumption allows formulating the future state in a horizon as a function of the initial state $\boldsymbol{x}_0$ and the control horizon $\underline{\boldsymbol{U}}$: more explicitly, a function $\overline{\boldsymbol{X}}(\boldsymbol{x}_0, \underline{\boldsymbol{U}})$.

To do so, one may develop (3.29) for every state $\boldsymbol{x}_k$ (for $k = 1 \dots \mathrm{h}$) as a function of the initial state $\boldsymbol{x}_0$ and the successive control inputs $\boldsymbol{u}_0, \boldsymbol{u}_1 \dots \boldsymbol{u}_{\mathrm{h}}$:

$$\boldsymbol{x}_1 = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\boldsymbol{u}_0 \tag{4.33}$$

$$\begin{aligned} \boldsymbol{x}_2 &= \boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{u}_1 \\ &= \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\boldsymbol{u}_0) + \boldsymbol{B}\boldsymbol{u}_1 \\ &= \boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{B}\boldsymbol{u}_1 \end{aligned} \tag{4.34}$$

$$\begin{aligned} \boldsymbol{x}_3 &= \boldsymbol{A}\boldsymbol{x}_2 + \boldsymbol{B}\boldsymbol{u}_2 \\ &= \boldsymbol{A}(\boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{B}\boldsymbol{u}_1) + \boldsymbol{B}\boldsymbol{u}_2 \\ &= \boldsymbol{A}^3\boldsymbol{x}_0 + \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{B}\boldsymbol{u}_2 \end{aligned} \tag{4.35}$$

$$\vdots$$

$$\boldsymbol{x}_H = \boldsymbol{A}^H\boldsymbol{x}_0 + \boldsymbol{A}^{H-1}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{A}^{H-2}\boldsymbol{B}\boldsymbol{u}_1 \dots + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_{\mathrm{h}-2} + \boldsymbol{B}\boldsymbol{u}_{\mathrm{h}-1} \tag{4.36}$$

arriving at a matricial form:

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_{\mathrm{h}} \end{bmatrix}}_{\overline{\boldsymbol{X}}} = \underbrace{\begin{bmatrix} \boldsymbol{B} & & & & \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & & & \\ \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & & \\ \vdots & & & & \\ \boldsymbol{A}^{\mathrm{h}-1}\boldsymbol{B} & \boldsymbol{A}^{\mathrm{h}-2}\boldsymbol{B} & \dots & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} \end{bmatrix}}_{\hat{\boldsymbol{A}}} \underbrace{\begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{\mathrm{h}-1} \end{bmatrix}}_{\underline{\boldsymbol{U}}} + \underbrace{\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \boldsymbol{A}^3 \\ \vdots \\ \boldsymbol{A}^{\mathrm{h}} \end{bmatrix}}_{\hat{\boldsymbol{B}}} \boldsymbol{x}_0 \tag{4.37}$$

revisited ↓

that can be synthesized into a simple algebraic expression:

$$\overline{\boldsymbol{X}}(\boldsymbol{x}_0, \underline{\boldsymbol{U}}) = \hat{\boldsymbol{A}}\underline{\boldsymbol{U}} + \hat{\boldsymbol{B}}\boldsymbol{x}_0 \tag{4.38}$$

that can be directly embedded into (3.47):

$$\min_{\boldsymbol{u}} \left\| \overline{\boldsymbol{X}}_t - \overline{\boldsymbol{X}}(\boldsymbol{x}_0, \underline{\boldsymbol{U}}) \right\|_{\boldsymbol{P}}^2 + \left\| \underline{\boldsymbol{U}} \right\|_{\boldsymbol{R}}^2 \tag{4.39}$$

but optimizing only over the control inputs $\boldsymbol{u}$. Nevertheless, it can still be subject to linear constraints:

$$\boldsymbol{C}_{\boldsymbol{x}_m} \leq \boldsymbol{C}_{\boldsymbol{x}} \boldsymbol{x}_k \leq \boldsymbol{C}_{\boldsymbol{x}_M} \tag{4.40}$$ revisited ↑ ↓

$$\boldsymbol{C}_{\boldsymbol{u}_m} \leq \boldsymbol{C}_{\boldsymbol{u}} \boldsymbol{u}_k \leq \boldsymbol{C}_{\boldsymbol{u}_M} \tag{4.41}$$ revisited ↑ ↓

$$\begin{aligned} &\boldsymbol{C}_{\dot{\boldsymbol{u}}_m} \leq \boldsymbol{C}_{\dot{\boldsymbol{u}}} \dot{\boldsymbol{u}}_k \leq \boldsymbol{C}_{\dot{\boldsymbol{u}}_M} \\ &\text{with:} \quad \dot{\boldsymbol{u}}_k = \tfrac{(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)}{\mathrm{T}} \end{aligned} \tag{4.42}$$

where the control input constraints in (4.41) are kept the same as in (3.35) and (3.36). Meanwhile, the state constraints need to be expressed with respect to the optimization variable with (4.38). In the case of simple state bounds (where $\boldsymbol{C}_{\boldsymbol{x}}$ is an identity matrix), it can be done straightforwardly with:

$$\overline{\boldsymbol{X}}_m - \hat{\boldsymbol{B}}\boldsymbol{x}_0 \leq \hat{\boldsymbol{A}}\underline{\boldsymbol{U}} \leq \overline{\boldsymbol{X}}_M - \hat{\boldsymbol{B}}\boldsymbol{x}_0 \tag{4.43}$$

where the term $\hat{\boldsymbol{B}}\boldsymbol{x}_0$ can be treated as constant, because $\boldsymbol{x}_0$ is the measured initial state of the system (at the start of the horizon) and:

$$\overline{\boldsymbol{X}}_m = \begin{bmatrix} \boldsymbol{x}_{m|k+1} \\ \boldsymbol{x}_{m|k+2} \\ \vdots \\ \boldsymbol{x}_{m|k+\mathrm{h}} \end{bmatrix} \qquad \overline{\boldsymbol{X}}_M = \begin{bmatrix} \boldsymbol{x}_{M|k+1} \\ \boldsymbol{x}_{M|k+2} \\ \vdots \\ \boldsymbol{x}_{M|k+\mathrm{h}} \end{bmatrix} \tag{4.44}$$

express the limits for the state at each discretized step in the horizon. Here, they are chosen constant to coincide with those in (3.35), but the expression can accommodate potentially evolving limits.

The relevance of this formulation comes from optimizing the QP expression to reduce computation times. In fact, the solving times suffer from the "curse of dimensionality" and scale exponentially with the size of the optimizing variable. This is why reducing its size is a simple strategy with a direct impact on performance [Ferreau, 2011] without recurring to customized solvers that need to better exploit the sparsity of the problem matrices to achieve the same gains.

4.3.2 Dynamics-based MPC in the Tangent Space ↑

This section revisits some concepts from Sections 3.3.3 and 3.3.4 to focus on the formulations introduced in Section 3.3.5, that were employed to perform optimizations in the tangent space of pose manifold, for a first approach to motion generation with an MPC. This time, even though some of the equations are the same, they are specialized to obtain an acceleration-based formulation. Additionally, the lift function is different and offers some computational advantages that aid in achieving

re-planning at the control-rate.

The first step is understanding how (3.27) can express the pose dynamics in a linear form. Consider the first-order integration of a pose in $\mathbb{SE}(3)$ subject to an infinitesimal body twist increment for a single time step T:

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k} \tag{4.45}$$ revisited ↑ ↓

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + \mathrm{T}\dot{\boldsymbol{\nu}}_k \tag{4.46}$$ revisited ↓

where (4.46) is added to relate the twist to a spatial acceleration $\dot{\boldsymbol{\nu}}$ integration step for the same time period T. The objective is to include the pose and the twist into the state $\boldsymbol{x}$ of the system, and use the spatial acceleration as a decision variable or control input $\boldsymbol{u}$. This way, (4.45) and (4.46) can be expressed in the form of (3.27) to be used for linear MPC with (4.39).

The term $e^{\mathrm{T}\boldsymbol{\nu}_k}$ in (4.45) is the equivalent homogeneous matrix to a floating body moving at $\boldsymbol{\nu}_k$ for a duration of T. It is computed using the matrix exponential, that surjectively maps the Lie algebra $\mathfrak{se}(3)$ onto the Lie group $\mathbb{SE}(3)$. Its inverse operation is known as the logarithmic map.

This relation can be exploited to represent $\mathcal{X}$ in a vector form. In fact, given some initial pose $\mathcal{X}_0$, one can devise a function $\boldsymbol{\psi}$ (referred to as the *lift function*) that maps $\mathcal{X}_k$ to a vector $\boldsymbol{\xi}_k$. The general principles behind this approach are further explained in Section 2.4 or, more briefly, in [Forster et al., 2015]; [Solà et al., 2018]; [Torres Alberto et al., 2022a]. Such function is defined as:

$$\boldsymbol{\psi}(\mathcal{X}_k) = \log(\mathcal{X}_0^{-1}\mathcal{X}_k) \qquad \boldsymbol{\psi}^{-1}(\boldsymbol{\xi}_k) = \mathcal{X}_0\, e^{\boldsymbol{\xi}_k} \tag{4.47}$$ revisited ↑

$$\begin{array}{rcccc} \boldsymbol{\psi} & : & \mathbb{SE}(3) & \to & \mathfrak{se}(3) \\ \boldsymbol{\psi}^{-1} & : & \mathfrak{se}(3) & \to & \mathbb{SE}(3) \end{array} \qquad \begin{array}{rcl} \log(\mathcal{X}_0^{-1}\mathcal{X}_k) & \to & \boldsymbol{\xi}_k \\ \mathcal{X}_0\, e^{\boldsymbol{\xi}_k} & \to & \mathcal{X}_k \end{array} \tag{4.48}$$ revisited ↑

The lift function maps elements from the $\mathbb{SE}(3)$ manifold $\mathcal{M}$ to its tangent space at $\mathcal{X}_0$: $\boldsymbol{\psi} : \mathcal{M} \to \mathcal{T}_{\mathcal{X}_0}\mathcal{M}$. Notice that the roto-translational displacement $\Delta\mathcal{X}_k \in \mathbb{SE}(3)$ (or equivalent homogeneous matrix) needed to move from $\mathcal{X}_0$ to $\mathcal{X}_k$ can be computed with $\Delta\mathcal{X}_k = e^{\boldsymbol{\psi}(\mathcal{X}_k)}$. The function $\boldsymbol{\psi}^{-1}$ is referred to as the *retract function*.

With the MPC being formulated in $\mathfrak{se}(3)$, the lift operator $\boldsymbol{\psi}$ is used to map the current state and the desired pose into $\mathfrak{se}(3)$ to perform the optimization. Meanwhile the retract operator $\boldsymbol{\psi}^{-1}$ is used to map the computed trajectory back to $\mathbb{SE}(3)$. This enables directly embedding the *lifted* pose $\boldsymbol{\xi} = \boldsymbol{\psi}(\mathcal{X})$ into the discretized state of the system $\boldsymbol{x}$, along with the body twist $\boldsymbol{\nu}$. Then, choosing the spatial acceleration $\dot{\boldsymbol{\nu}}$ as the input:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\nu}_k \end{bmatrix} \qquad \boldsymbol{u}_k = \dot{\boldsymbol{\nu}}_k \tag{4.49}$$ revisited ↑ ↓

where the reader is reminded that the k sub-index symbolizes the discretized step within the horizon. In other words, with this notation, $\boldsymbol{x}_k$ corresponds to the state at the step k in the horizon starting at time t_n, also denoted as $\boldsymbol{x}_{k|n}$. The same notation applies to the other variables. Then, once the MPC optimization is solved, it is

straightforward to obtain the desired twists and spatial accelerations from $\boldsymbol{x}$ and $\boldsymbol{u}$. Yet, for the desired pose $\mathcal{X}$, the retract function is required:

$$\mathcal{X}_k = \boldsymbol{\psi}^{-1}(\boldsymbol{\xi}_k) \tag{4.50}$$

Furthermore, in order to link (4.45) and (4.46) with (3.27) (with linear MPC in mind), it needs to be linearized. Computing $\boldsymbol{\psi}(\mathcal{X}_{k+1})$ yields:

$$\boldsymbol{\psi}(\mathcal{X}_{k+1}) = \log(\Delta\mathcal{X}_k e^{\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k}) \tag{4.51}$$

This expression has a first-order approximation in its Lie algebra (see Equations 68-74 in [Solà et al., 2018]):

$$\boldsymbol{\xi}_{k+1} = \log(e^{\boldsymbol{\xi}_k} e^{\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k}) \approx \boldsymbol{\xi}_k + \mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k)(\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k) \tag{4.52}$$ revisited ↑ ↓

where $\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k)$ is the right-trivialized Jacobian of the logarithmic map, evaluated at $\boldsymbol{\xi}_k$. A closed-form expression can be found in [Barfoot et al., 2014]; [Solà et al., 2018] and is implemented in the Pinocchio library [Carpentier et al., 2019]. It relates the additive increments in $\mathcal{T}_0\mathcal{M}$ (the tangent map at the origin) to the right-multiplied increments in $\mathbb{SE}(3)$.

Equation (4.52) offers the missing piece to have a concrete definition for (3.27) when considering the acceleration $\dot{\boldsymbol{\nu}}_k$ as the control input:

$$\boldsymbol{A}_k = \begin{bmatrix} \mathbf{I}_6 & \mathrm{T}\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k) \\ \mathbf{0}_6 & \mathbf{I}_6 \end{bmatrix} \qquad \boldsymbol{B}_k = \begin{bmatrix} \frac{\mathrm{T}^2}{2}\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k) \\ \mathrm{T}\mathbf{I}_6 \end{bmatrix} \tag{4.53}$$ revisited ↑ ↓

finally offering a concrete definition for the propagation of the state of the system as in (4.38).

The final step is to exploit these identities in the cost function (4.39), but a proper way to calculate the distance in the tangent space has not yet been presented. In fact, a remarkable property of the lift and retract functions in (4.48) is that the geodesic path between $\mathcal{X}_0$ and $\mathcal{X}_t$ can be interpolated for any pose in between with a parameter $\alpha \in [0, 1]$:

$$\mathcal{X}(\alpha) = \boldsymbol{\psi}^{-1}(\alpha\boldsymbol{\psi}(\mathcal{X}_t)) = \mathcal{X}_0 e^{\alpha\boldsymbol{\psi}(\mathcal{X}_t)} \tag{4.54}$$

It is straightforward to notice that $\mathcal{X}(0) = \mathcal{X}_0$ and $\mathcal{X}(1) = \mathcal{X}_t$.

This geodesic path is obtained by minimizing the *Log-Euclidean* ([Jayasumana et al., 2014], Table 1) distance metric between the lifted pose $\boldsymbol{\xi}_k$ and the lifted target pose $\boldsymbol{\xi}_t = \boldsymbol{\psi}(\mathcal{X}_t)$:

$$\|\boldsymbol{\xi}_t - \boldsymbol{\xi}_k\|_2^2 \tag{4.55}$$

Then one can define a metric for the state space that calculates the target-related cost in (4.39). Assuming the target state $\boldsymbol{x}_t$ contains the lifted target pose and a null (zero)

body twist:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{\xi}_t \\ \mathbf{0} \end{bmatrix} \tag{4.56}$$

revisited ↑ ↓

for each step in the discretized horizon, one can use the metric:

$$\left\| \boldsymbol{x}_t - \boldsymbol{x}_k(\boldsymbol{x}_0, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1}) \right\|_2^2 \tag{4.57}$$

where $\boldsymbol{x}_k(\boldsymbol{x}_0, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1})$ is just (4.38) defined for a single step k in the horizon. In this way, minimizing (4.57) for all discretized steps, it leads to minimizing the geodesic distance towards the target pose over the horizon. This effectively induces a generated motion that follows the geodesic path in the direction of the target while respecting the constraints imposed in the optimization.

4.3.3 Interpolating the Horizon Trajectory ↑

It is also important to highlight that the discretization step T within the horizon typically needs to be longer than the robot control period. This is necessary in order to decrease resolution times (by reducing the size of the decision variables in the optimization problem while still being able to optimize long enough into the future for the solution to be significant). This remains true even if the MPC optimization runs at every control loop. By doing this, the MPC can contemplate longer horizons, leading to enhanced performance and ensuring that constraints are met. Nonetheless, it implies that the output horizon trajectory from the MPC must be interpolated to determine the subsequent desired state / input.

In the case of the twist and input, their value at some desired time can be obtained by a simple linear interpolation. Consider a desired time within two steps in the horizon $t_d \in [t_k, t_{k+1}]$ and some variable discretized within a horizon $\boldsymbol{y}$, then the linear interpolation function $\boldsymbol{f}_{\text{lerp}}(\boldsymbol{y}, t_d)$ is defined as:

$$\boldsymbol{y}(t_d) \approx \boldsymbol{f}_{\text{lerp}}(\boldsymbol{y}, t_d) = \boldsymbol{y}_k + \frac{(t_d - t_k)}{\mathrm{T}}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k) \tag{4.58}$$

such that $\boldsymbol{y}(t_k) = \boldsymbol{y}_k$ and $\boldsymbol{y}(t_{k+1}) = \boldsymbol{y}_{k+1}$. Then the desired trajectory can be interpolated from the body twist and spatial acceleration with $\boldsymbol{f}_{\text{lerp}}(\boldsymbol{\nu}, t_d)$ and $\boldsymbol{f}_{\text{lerp}}(\dot{\boldsymbol{\nu}}, t_d)$, respectively.

Interpolating the desired pose in the obtained horizon trajectory requires a bit more work but can be done by using the exponential map in a similar way to the lift function. Under the same assumptions for t_d as in (4.58), one can define an interpolation scheme in the pose manifold as:

$$\mathcal{X}(t_d) \approx \mathcal{X}_k e^{\alpha\delta} \qquad \delta = \log(\mathcal{X}_k^{-1}\mathcal{X}_{k+1}) \qquad \alpha = \frac{(t_d - t_k)}{\mathrm{T}} \tag{4.59}$$

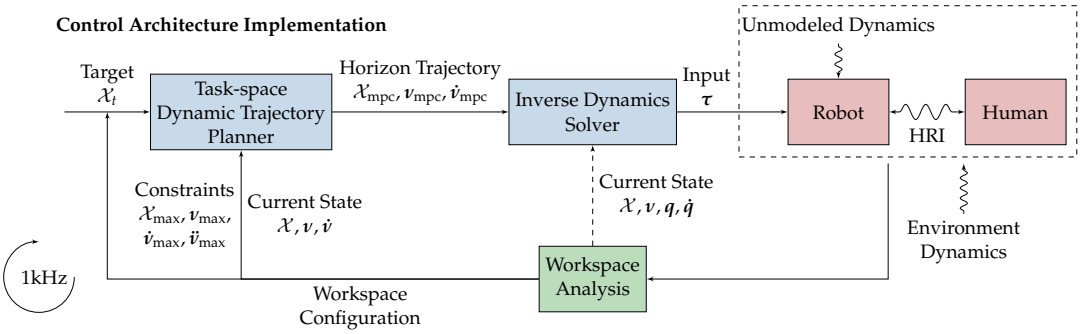


Figure 4.2: The implemented control architecture is based on the one proposed in Section 4.1.2. It is composed of a cascade loop: 1) a task-space Model Predictive Controller for online re-planning (described in Section 4.3); 2) a constrained Task Space Inverse Dynamics solver to follow the planned trajectory (described in Section 4.3). The feedback is closed with the current state of the robot. revisited ↑ ↓

4.4 Experimental Validation: Online Re-planning at the Control Rate ↑

In the experiment, the robot is directed to align itself in order to grasp a plastic piece in the workspace, despite not having prior knowledge of the piece location or orientation (the pose of the object is sensed in real time by a camera array). The purpose is to test the robot capability to adapt on-the-fly to unexpected changes in the target pose. The outcomes illustrate a near-time optimal movement, combined with frequent online adjustments, all while adhering to the imposed workspace constraints.

Some of these results are exemplified in Figure 4.3 with an online trajectory adaptation in two situations: during the alteration of the maximum Cartesian velocity (constraints); and in the case of a unplanned changes to the target pose. These results are qualitative but they confirm the overall ability of the proposed control approach to adapt the robot motion to changing tasks conditions, highlighting several noteworthy features of the approach including:

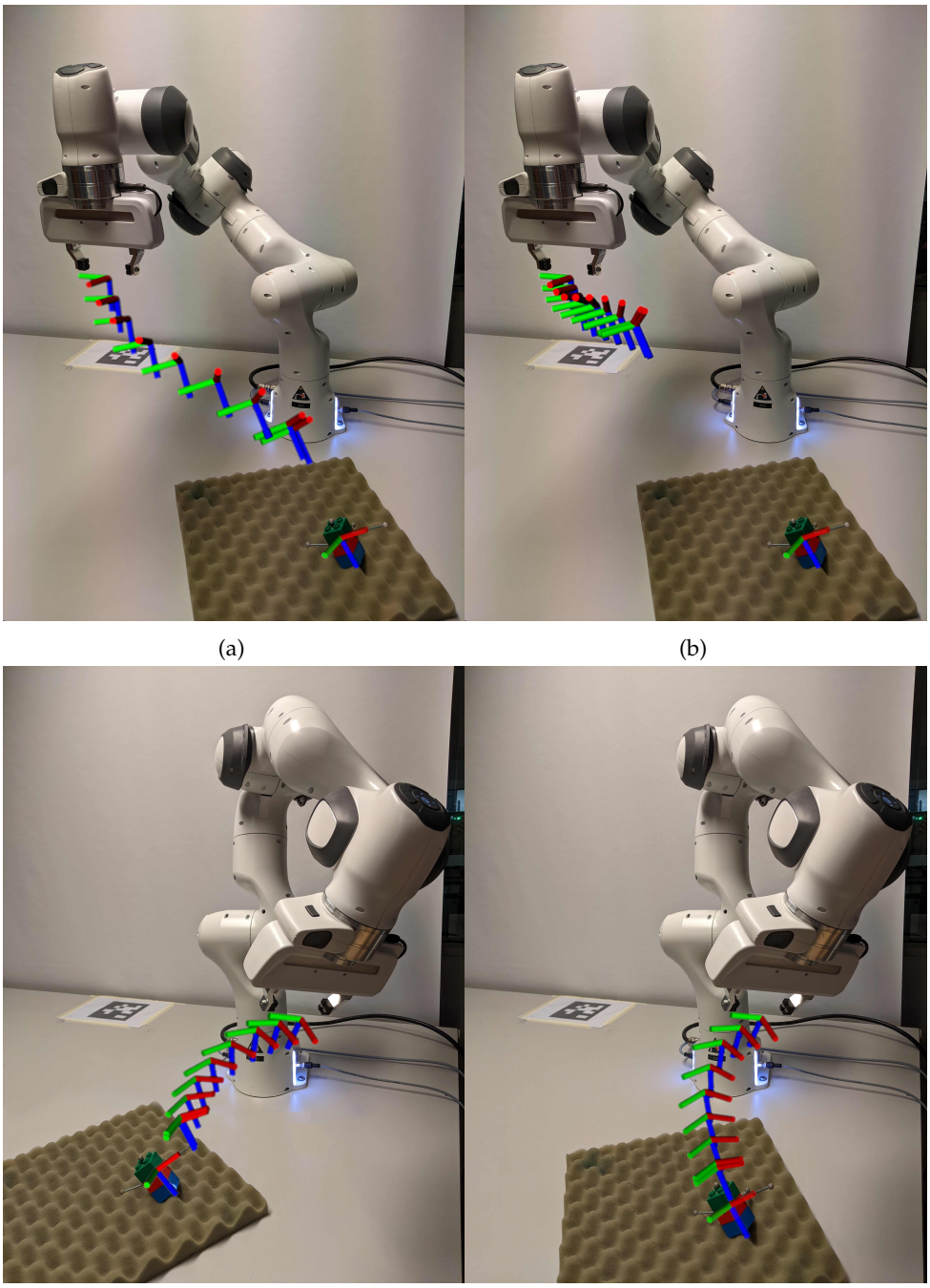
1. constraints compliance.
2. the ability to manage instantaneously switching targets.
3. the ability to continuously adapt the trajectory towards a continuously moving target.
4. the effect of the modification of velocity constraints in the MPC on the generated motion horizon at each control step.

A detailed video of the experiment may be found by following the link in Figure 4.7.

4.4.1 Experimental Procedure & Setup ↑

A three phase experiment is proposed, each progressively increasing the complexity of the task. The setup and phases are illustrated in Figure 4.4. After completing each phase, the robot returns to its "home pose" before embarking on the subsequent phase. The specifics of the phases are as follows:

video/link



(a) (b)

(c) (d)

Figure 4.3: A linear Model Predictive Control (MPC) is used to compute a position and orientation trajectory towards a target (plastic brick) in a receding horizon. (a) and (b) highlight the effects of changing the task space constraints on the fly (in this case max velocities) on the covered distance for one horizon. (c) and (d) illustrate the continuous adaptation of the trajectory when the target location is modified online.

video/link

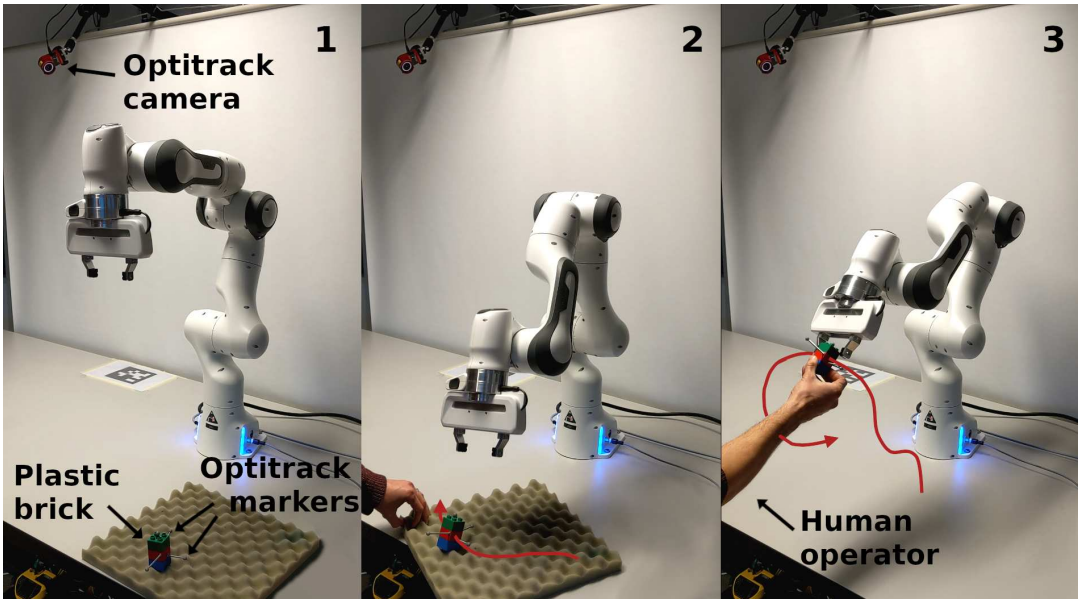


Figure 4.4: The experimental setup is composed of a torque-controlled 7-DoF Franka Emika Panda serial robot and an Optitrack camera array. Reflective markers are placed on a plastic brick. The camera array is used to detect the position and orientation of the plastic brick in the workspace. The robot is instructed to grab the brick without prior knowledge of its pose. The experiment consists of three stages of increasing complexity: 1- non-moving brick; 2- slowly moving brick; 3- the brick is held by the operator and tracked in real time.

1. **Non moving target:** The simplest task is presented in the first phase. The robot is requested to pick-up the plastic brick placed in the workspace on top of an irregular surface (to induce non-trivial orientations). The pose of the brick is not known a priori and is detected via a vision system. The plastic piece is not moved during this phase. Given this target pose information $\mathcal{X}_t$, the robot starts its motion towards the brick using the control scheme shown in Figure 4.1. Once the object is reached, the robot grasps it and releases it after a few seconds. The tracking results of this phase are shown in Figure 4.5.

2. **Slowly moving target:** For the second phase, the robot goes back to its neutral position and the brick is placed at a new starting pose. After a few seconds, the robot starts its motion towards this new goal. While in motion, the brick is slowly displaced by a human and the robots adapts its trajectory online to reach it.

3. **Operator moving target:** In the third phase, the human holds the plastic brick and moves it around randomly at moderate speed while the robot tracks it in real time. To increase the complexity the operator also holds the robot for a few seconds to interrupt its movement, showcasing the re-planning capabilities of the approach. This phase is illustrated in snapshots of the video in Figure 4.6.

The setup is shown in Figure 4.4. Perception of the plastic brick is provided by an Optitrack camera array with reflective markers attached. The experiment is carried out on a torque-controlled 7-Degree of Freedom (DoF) Franka Emika Panda serial robot.

The control architecture in use is grounded on the design detailed in Section 4.1.2. It

is shown in Figure 4.2 and runs at 1kHz. This system is structured as a cascade loop, consisting of two primary components:

1. A task-space Model Predictive Controller for real-time re-planning, as outlined in Section 4.3.

2. A constrained Task Space Inverse Dynamics solver, dedicated to adhering to the mapped trajectory, also elucidated in Section 4.3.

The feedback loop is completed using the robot current state. In order to keep up with the control rate, the MPC uses a 750ms control horizon ($h = 5, T = 150$ms), which yields an average computation time[2] of 0.1ms (max 0.2ms) using qpOASES [Ferreau et al., 2014].

The same Quadratic Program (QP) solver is used to solve the constrained task space inverse dynamics. The values of gains and weights for this inner QP controller are summarized in Table 4.1. It is just recalled here that a larger cost is assigned to the terminal step of the horizon to favour convergence towards the target. Implementation details may be found in the source code[3].

Gains / Weights	K_p	K_d	K_q	$K_{\dot{q}}$	$w_{\mathbf{reg}}$
Value	150	24.5	100	10	10^{-5}
Units	s^{-2}	s^{-1}	$N.m.s^{-2}$	$N.m.s^{-1}$	$N^{-2}.m^2$

Table 4.1: Values of the gains and weights retained for the inverse dynamic solver used for task-space tracking shown in Figure 4.2 and described in Section 4.2.

Details on the resulting motion from one run of this experiment are provided in Figure 4.5 for the first of the described phases. While the brick stays in a fixed pose during this phase, it serves to showcase two behaviors: first, it shows the online re-planning algorithm uses all the acceleration available to attain pseudo-optimal trajectories; secondly, it does so even when the provided target pose is discontinuous (at the start of the phase).

The nuances of the other two experiments are best understood through the video accompanying this study, which can be referenced in Figure 4.7. The dynamic nature of the moving target produces an adaptive behavior that is most effectively interpreted within the video context. Nonetheless, snapshots from the video can be seen in Figure 4.6. These captures illustrate the experiment third phase where an operator holds the brick. The objective here is to demonstrate the robot real-time adaptability, even when faced with external disturbances, such as being manipulated by an operator, as highlighted in Figures 4.6(d) and 4.6(e).

4.4.2 Analysis ↑

Figure 4.5(a) shows the error between the current and the target poses. The discontinuity seen at the beginning corresponds to the start of the experiment sequence, where the target pose switches from the "home pose" to the plastic brick pose. This

[2]The experiments were run on desktop computer equipped with a CPU i9-10900K and 32GB of ram.
[3]https://gitlab.inria.fr/auctus-team/people/nicolas-torres/lmpcpoly-ws/

	ν_{max}	$\dot{\nu}_{max}$	$\ddot{\nu}_{max}$
linear	0.2 (m/s)	2 (m/s^2)	1000 (m/s^3)
angular	0.8 (rad/s)	5 (rad/s^2)	3000 (rad/s^3)

$$-\nu_{max} \leq \nu \leq \nu_{max}, \quad -\dot{\nu}_{max} \leq \dot{\nu} \leq \dot{\nu}_{max}, \quad -\ddot{\nu}_{max} \leq \ddot{\nu} \leq \ddot{\nu}_{max}$$

Table 4.2: Cartesian constraints for $\nu, \dot{\nu}, \ddot{\nu}$ for the MPC during the experiment. The MPC used for motion generation is shown in Figure 4.2 and described in Section 4.2.

highlights the convergence towards the requested pose without any a priori knowledge on the trajectory to follow. Moreover, it shows that even in the presence of a discontinuous target (a task change), the approach succeeds in providing a smooth transition that, while respecting constraints, progressively adapts the motion.

Notice that this is not the resulting tracking error of following the trajectory planned by the MPC with the constrained task-space inverse dynamics solver, i.e. the tracking performance of following a desired trajectory computed with the MPC. Such curve is shown in Figure 4.5(b). It is bounded to 19.7mm with a mean value of 4mm $\pm$ 1 for the linear component of the motion and 5.18×10^{-2}rad with a mean value of 2.5×10^{-3}rad $\pm 5.5 \times 10^{-3}$ for the angular component.

The final error is not exactly zero (2.1mm and 10.4×10^{-2}rad) due to the relatively low gains of the Proportional Derivative (PD) controller and the absence of an integral term to reject disturbances due to dry friction at the joint level. Yet, the robot is able to grasp the targeted object at the end of the motion as can be seen in the accompanying video (see Figure 4.7).

Figure 4.5(c) shows both the planned (by the MPC) and the actual velocities; as opposed to the acceleration and jerk curves (respectively in Figures 4.5(d) and 4.5(e)) that only show the planned trajectories.

The first thing to notice is that the constraints on the planned trajectory are met by the MPC, as can be seen on the velocity and jerk curves.

It is also interesting to see that the obtained linear velocity profile is similar, to some extent, to a trapezoidal acceleration profiles, maximizing the use of some of the prescribed Cartesian limits shown in Table 4.2. This confirms the near-optimal character of the generated trajectory.

Finally, one can notice that the robot end-effector velocity surpasses the established limits. This is discussed in Section 4.5.

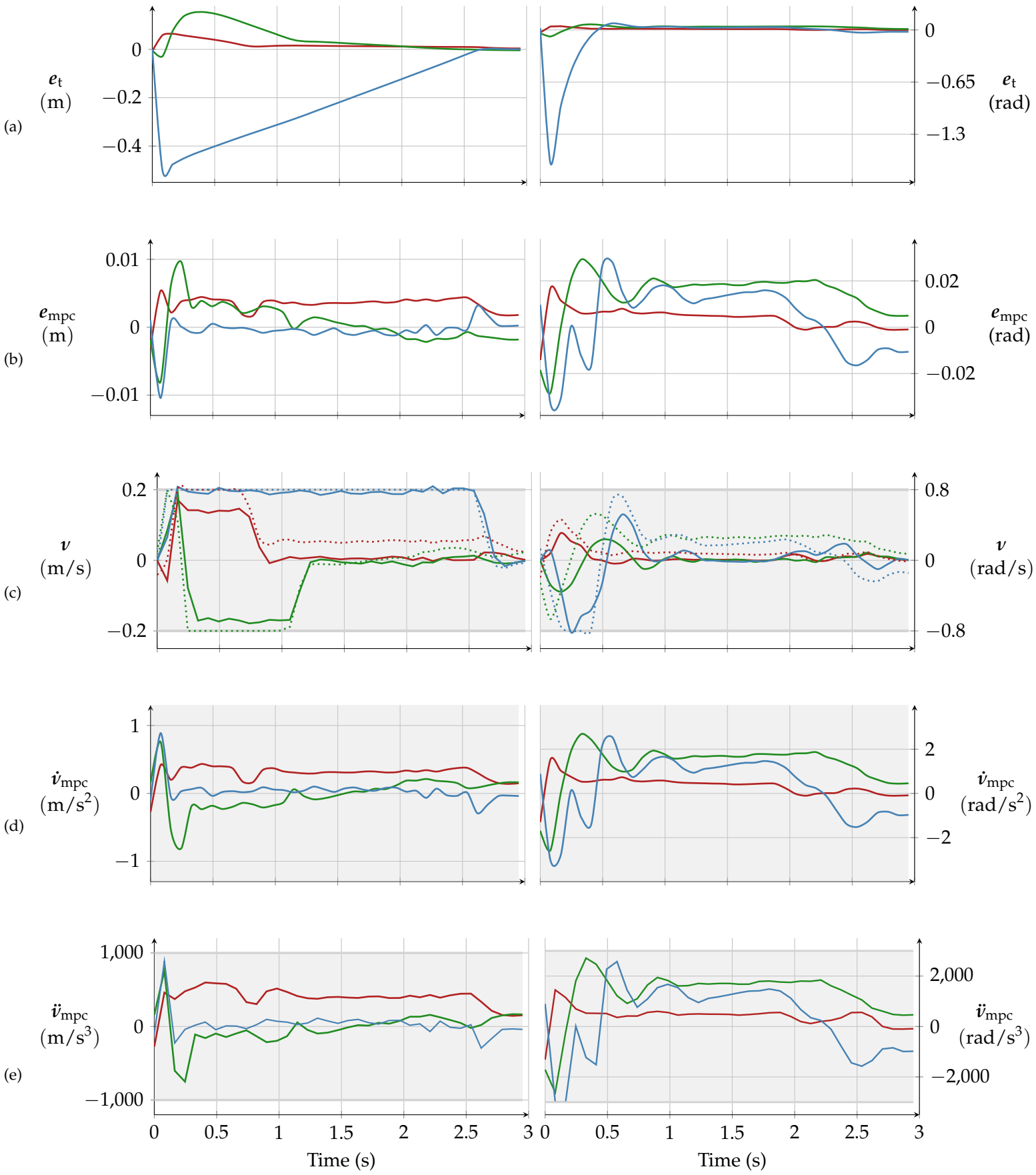


Figure 4.5: In the first phase, the robot is tasked to pick up the plastic brick positioned on an uneven surface to induce a non-trivial orientation. The brick pose, unknown beforehand, is identified through a vision system and remains stationary. Using this information, the robot moves to the brick as per the control scheme shown in Figure 4.1. After grasping, it holds the brick for a few moments and then releases it. The tracking results indicate target poses set in real-time. In each graph, the red, green, and blue curves denote x, y, and z axis components. Actual robot velocities are shown with solid lines, while the desired velocities (calculated by the MPC and denoted ν_{mpc}) are dotted. Shaded regions mark the bounds set for the MPC.

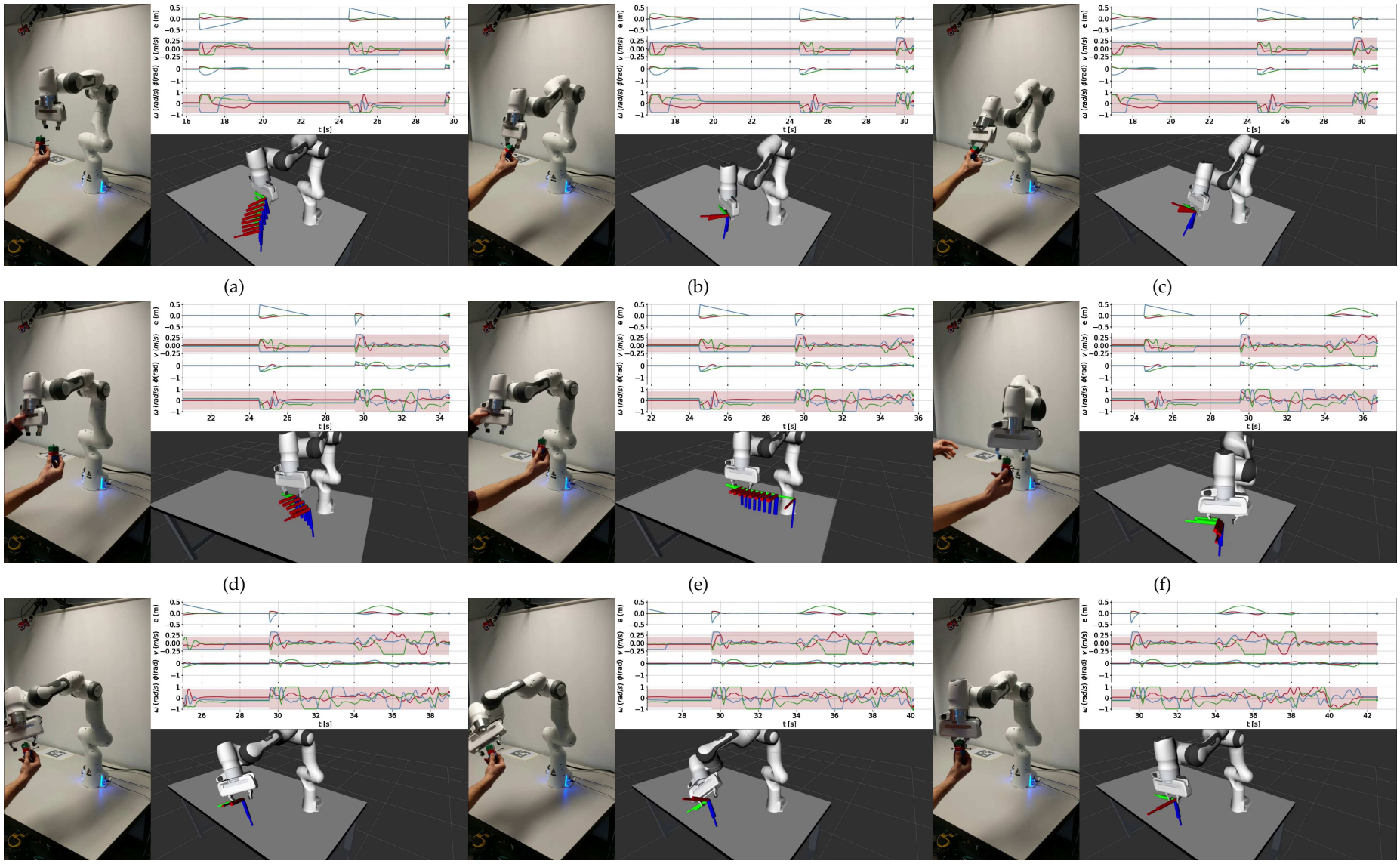


Figure 4.6: This showcases the third phase of the experiment. Certain key moments of the accompanying video have been captured in still images to provide a more immediate understanding. During the third phase of the experiment, the focus shifts to the robot real-time adaptability and its capability to re-plan its actions in response to dynamic changes. During this phase, a human operator holds the plastic brick, moving it in an unpredictable manner at a moderate pace. As the brick moves, the robot diligently tracks its position and orientation in real-time. Adding to the complexity, the operator intervenes by restraining the robot movement for a short period (shown in (d) and (e)), serving as an external disturbance. This interruption aims to highlight the robot proficiency in rapidly recalculating and adjusting its strategy. The images depict three elements: on the left, the real robot as seen during the test is shown; on the bottom right, a virtual counterpart displays both the target pose (the brick pose) and the horizon trajectory calculated in real-time; meanwhile, the top right presents graphs illustrating the tracking error and planned velocities.

4.5 Discussion ↑

The modular nature of the proposed architecture offers the potential to generically consider different types of task inputs. It can for example be easily adapted to use a discretized pose path, which could come from an obstacle avoidance solver. Hence, the same architecture can accommodate a complex pathing solution while retaining its core structure.

This architecture can also accommodate different types of task space to joint space solvers. This is an important feature as not all robots allow for joint torque control.

Moreover, the cost function of the Model Predictive Control (MPC) and the Cartesian constraints offer a generic way to design human-aware solutions considering the non-trivial cognitive aspects of human-robot collaboration. For example, representing the human motion capabilities as Cartesian constraints [Skuric et al., 2022b].

Furthermore, the controller's interactive capabilities demonstrate its ability to adjust to the physical attributes of the operator. For instance, in a pick-and-place scenario where the robot retrieves an item from a human, it will approach the human. This allows the operator to choose a comfortable height or location for interaction. This example is comparable to the scenario shown in the experimental interactive tracking task. However, more complex tasks where the robot needs to assist the human have the potential to showcase a stronger effect.

This is further enabled by decoupling the task-related definition and constraints from the joint-level controller. Indeed, some constraints are intrinsically task related and somewhat independent of the robot performing the task while others are strongly related to the whole robot motion. It is for example the case for safety-aware constraints [Joseph et al., 2018a] or the exploitation of the robot redundancy to perform secondary tasks without compromising performance. Examples of the later case are shown for safety purposes in [Joseph et al., 2019]; [Mansfeld et al., 2017] and for situation awareness in [Camblor et al., 2022].

Figure 4.5(c) shows that the real end-effector velocity loosely follows the planned velocity, going beyond the established limits in some cases. The measured velocity is being used as the initial state of the MPC at each time step, causing the associated Quadratic Program (QP) to become infeasible. This characteristic is intrinsic to any QP-based architecture and it can be considered both a problem and an advantage: for one, it allows easily detecting ill-conditioned initial states that could result in undefined behaviors (for example, to trigger an emergency stop mode); on the other hand, it demands that a proper strategy is put in-place to deal with this situation.

For this experiment, the inner loop controller continues to follow the last horizon that was successfully computed until the MPC can compute a new one. This allows the execution to continue smoothly.

Nevertheless, this opens one fundamental question that naturally arises from this type of cascade control architectures: the compatibility between outputs of each stage. For instance, the MPC could plan a trajectory that respects the Cartesian constraints even if it is instantaneously unfeasible in a control step or may lead to an inevitable joint-level constraint violation in the near future [Rubrecht et al., 2010].

To solve this problem, joint-level constraints [Prete, 2018] should be implicitly considered at the MPC level.

There exist efficient methods to effectively formulate the actual robot capacities as Cartesian constraints [Skuric et al., 2021a]. This type of constraints goes beyond classical box-like bounds and is a step towards more complex adaptation of the robot behaviour based on its motion capabilities. Integrating these into the MPC is addressed in the next chapter.

4.6 Conclusion ↑

This Chapter proposes an efficient linear model predictive control approach that can deal with the online planning of a motion in $\mathbb{SE}(3)$ in the task space.

The main features of the proposed control approach lie in its ability to generate optimal Cartesian motions which dynamically account both for targets updated on-the-fly and evolving constraints. By utilizing a receding horizon approach, the robot gains the capability to interactively adjust and adapt to task changes.

In the context of collaborative robotics, utilizing this controller ensures the safety of a human operator sharing the workspace with the robot by adjusting maximum velocity constraints based on sensor feedback.

The next chapter focuses on extending linear constraints in the Model Predictive Control (MPC) for the Cartesian pose state, to limit the effective position and orientation space. This work also opens doors to potentially exploiting a more efficient expression of Cartesian constraints such as convex polytopes that consider the actual robot joint space capabilities in the receding horizon optimization stage.

The reduction of the computation time required to solve the MPC problem is also an important matter. It will be pursued to refine the computation of the trajectory over the horizon and potentially improve the performance of the closed-loop.

Figure 4.7: QR code for the video of the high-frequency acceleration-based Model Predictive Controller experiment for full-pose online constrained motion generation. On PDF file readers, it is also a clickable hyperlink.

Chapter 5

Actuation-aware Task-space Model Predictive Control

In collaborative robotics, the ability to adapt swiftly and seamlessly to to changes in the task and the surrounding environment, particularly in shared spaces with humans, is of paramount importance. The proposed modular control architecture grounded in Model Predictive Control strikes a balance between optimal task achievement and efficient computation. However, in order to obtain optimal behaviors that efficiently exploit the robot capabilities, they must be considered during the trajectory planning stages.

To further enhance the control strategy, this chapter bridges the gap between task-space planning and actuation. It introduces joint space constraints to better represent the robot actual actuation capabilities. It outlines a systematic method for the explicit consideration of these constraints from the task space, resulting in a dynamic and responsive control strategy. This innovative approach allows for the fine-tuning of task-space movements while maintaining some level of "actuation awareness" to ensure the robot stays within its operational limits.

Through experiments with Franka Emika robots, the relevance and effectiveness of this control architecture are demonstrated in scenarios where tasks are dynamically modified in real-time, showcasing its potential in real-world applications of collaborative robotics.

Contents

5.1 Introduction ↑

Previous chapters introduced a control strategy tailored for the requirements of collaborative robotics scenarios. For instance, these scenarios imply a dynamic shared working environment (between the robot and the human). This entails the need for the robot to adapt on-the-fly to a set of changing conditions that describe its environment. To add to this, the task can also evolve throughtout the collaborative work cell operation depending on multiple factors like the task sequence, the physical characteristics of the operator or certain unpredictable events that may occur. All these complex conditions must be considered by the control architecture, which is in charge of continuously providing a coherent answer that correctly adapts the robot behaviour in ways that stay safe and properly exploit the robot capacities to perform the productive activity.

This last characteristic can be further decomposed in multiple points. For instance, "continously adapting" speaks to the real-time immediacy requirement of the adaptation. The robot needs to evaluate the new conditions and provide an answer in a timely manner not only because the performance requirements of the collaborative task demand it but also because human safety depends on these decisions. Secondly, this also implies that the robot must continuously switch between generated trajectories that properly consider the current task priorities and constraints. This transitions need to be smooth enough to avoid undesireble jerky behaviors and to stay within operating limits.

5.1.1 Rationale: Feasible Online Motion Generation ↑

This last point is the focus of this chapter. Integrating the operating limits of a task is already a complex challenge because it requires the generated trajectories to remain valid with respect to arbitrary constraints. As proposed in Chapter 4 this is achievable thanks to optimization-based re-planning. Yet, the challenge is exacerbated by the layered control loop that must constantly take decisions that are coherent at all stages. In other words, the workspace constraints and/or task may impose some safety requirements (like limiting the operating velocity) but these need to be somehow linked to the robot actuation capabilities which highlights a crucial underlying question: What use is a trajectory that cannot be executed? Indeed, trajectory planning is implicitely intertwined with the control inputs and it should always (ideally) yield feasible trajectories.

Considering the robot actuation capabilities during task planning is complex enough due to having to "articulate" task-space motion requirements to joint-space actuators, which already entails choosing the optimal robot configuration to realize the desired motion. Beyond that, doing so in a reactive manner (i.e. in real time) quickly becomes an untractable problem. Evidently, the robustness of the planning algorithms, meaning the capabilitiy to consistently generate realizable trajectories is the highest priority. This is why recurring to simplifications allow achieving the desired robustness by accepting a performance loss trade off.

Indeed, often manufacturers offer constant Cartesian limits that can be used to solve planning problems. This constitutes a simplifying strategy that allows modeling actuation capabilities with an acceptable approximation as a constant task space limit. The tradeoff is two-sided: for one, sometimes this corresponds to undervaluing the capabilities, provoking a performance loss but remaining robust from a feasibility

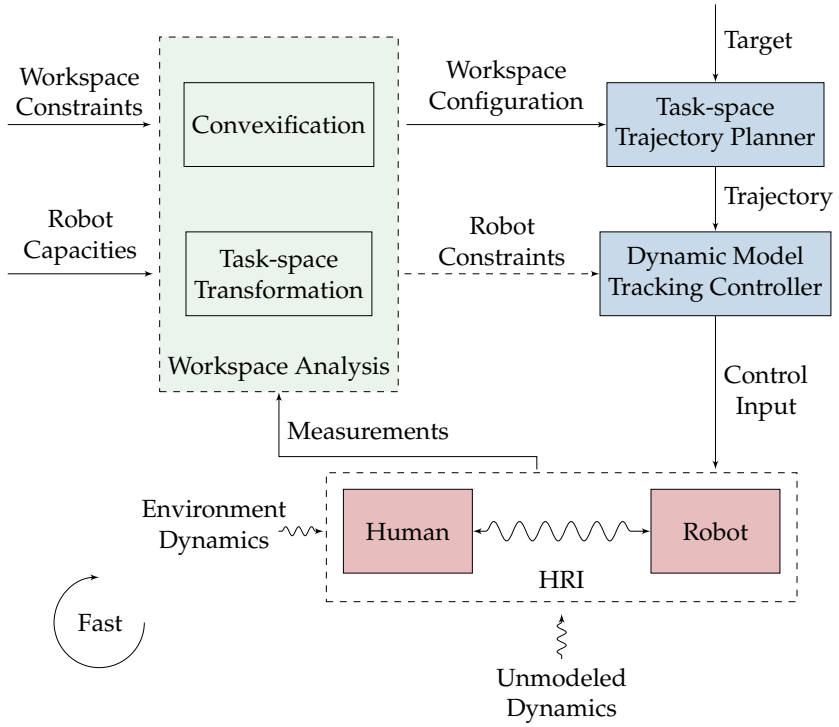


Figure 5.1: The proposed modular control architecture is based on a cascade loop composed of: 1) a task-space Model Predictive Controller (Model Predictive Control (MPC)) for online re-planning; 2) a constrained Task Space Inverse Dynamics solver to follow the planned trajectory. The whole control loop runs synchronously at the control rate. The Workspace Analysis block is in charge of sensing the environment, treating the information and transmitting the relevant information (workspace configuration and constraints) to the controller blocks. revisited ↑ ↓

point of view; and other times, due to the non-linear dynamics of the actuators, it may constitute an overvaluing. This last case could be considered potentially more dangerous as it breaks the underlying assumptions of the operating conditions, effectively increasing the risk of implicitely driving the system outside specifications *by design*.

Such situation is dangerous if it can put human lives at risk. Furthermore, inefficient use of the robot real capabilities also leads to dimensioning robots above the manufacturing task requirements, to compensate. In term, this means that potentially bigger and more dangerous [Joseph, 2018] robots are employed, provoking, yet again, an unnecessary safety risk.

5.1.2 Actuation-aware Control Framework ↑

This work focuses on the online motion adaptation of robot movements according to changing conditions and tasks in its environment. A concrete use case for this is the generation of trajectories safe for robot-human workspace sharing that are coherent with the robot capacities (feasible) and optimal with respect to some high-level criteria (like, for example, minimal energy expenditure).

The design principles underneath this control architecture remain consistent with the modular and task-centric point of view of this work. The modularity stems from

decoupling task-related behaviors and constraints from the actual execution in the robot, providing at least some level of independence from the robot architecture.

The proposal builds up on the previous chapter and is shown in Figure 5.1. The main focus of this chapter is the Workspace Analysis block that is generically named as a block in charge of measuring environment state and formulating it in a way that can be exploited by the control and planning blocks.

The task-centric design is revealed in the user configurability such as the ability to impose workspace constraints without being conditioned by the robot architecture or even the task. In other words, for example, safety constraints may be imposed as long as the re-planning algorithm exposes the related variables (for instance, in this case, the position and velocity). This means that task design, which can be defined with respect to the state of the collaborative assignment may be independent from the task constraints. Meanwhile, if the task constraints need to be enforced, there exists a mechanism that can limit the planned trajectories within these restrictions (along with other workspace constraints).

Another responsibility of this blocks is the convexification of the workspace constraints. For instance, the occupancy of the workspace may be defined by non-convex polyhedrons, yet a convex space definition is required for most optimization-based planning algorithms (such as the present MPC used). This convexification is actually application-dependent but here is generically named as a way to find the continuous space safe for the robot to perform a task without causing safety concerns. A concrete application of these is modeling the human body or the obstacles in the workspace, this way the robot can continuously consider them while performing its task.

Finally, the biggest focus of this chapter, is the consideration of the robot real capacitites. The proposal is to exploit a mathematical formalism to formulate the joint level constraints in the same space as the planning algorithm (the task space). This way, the optimization planning can actually have a more complete picture of the set of constraints that need to be considered in real-time. Not only does this help reduce or eliminate the guessing work of finding the fastest trajectory for a task-space task, but it also helps compatibility the planned trajectory with the actuation capabilities of the tracking controller.

5.2 From Joint-space to Task-space Constraints ↑

As mentioned in the introduction, one of the challenges of task-space planning is to correctly consider the actuation capacities that belong in the configuration space of the robot. Fortunately, some mathematical formalisms allow representing these capacities in the cartesian space.

The proposal in this work is to exploit the linear inequality mechanisms present in Quadratic Program (QP) optimization. This way, if the Joint-space Constraints (JSC) can somehow be represented as linear inequalities of task-space variables (i.e. those present in the Model Predictive Control (MPC)), they can be directly embedded into the problem constraints.

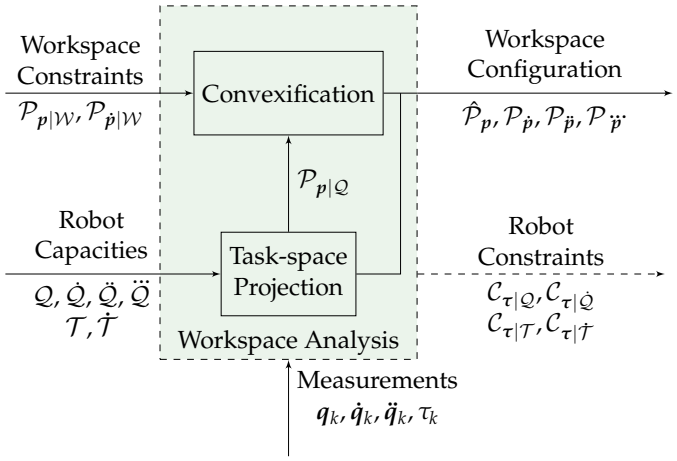


Figure 5.2: Zoom into the Workspace Analysis Block. It allows transforming user-defined workspace limits, joint-space capacitites and the task state into the "workspace configuration", which can be exploited as constraints and/or in the task function by the planner. revisited ↑ ↓

This section lies the foundations on how the JSC can be projected on to the Cartesian-space to provide the mathematical support needed actuation-aware task-space re-planning.

5.2.1 Task-space Feasible Sets as Convex Polyhedrons ↑

The JSC, once projected on to the Cartesian-space, define feasible sets. These sets relate the Cartesian-space motion quantities that are actually coherent with the actuation capabilities of the robot. The feasible sets that are introduced in this section all purposely share the convexity property, which makes it possible for them to take the form of a convex polyhedral through different methods [Gouttefarde et al., 2010]; [Skuric et al., 2021a]. These polyhedral can be defined in a generic way as:

$$\mathcal{P}_{x|y} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{b},\ \boldsymbol{y} \in [\boldsymbol{y}_m, \boldsymbol{y}_M]\} \tag{5.1}$$

where $\mathcal{P}$ defines a convex polyhedron[1] over $\boldsymbol{x}$ due to the bounds on $\boldsymbol{y}$ [Fukuda et al., 2004]. This definition implies that the interval domain of the $\boldsymbol{y}$ variable defines a projected constrained space on the $\boldsymbol{x}$ variable through the linear transformation operator $\boldsymbol{A}$. This constrained space is shifted from the origin by $\boldsymbol{b}$. The shape of the projected space is only affected by $\boldsymbol{A}$ and $\boldsymbol{b}$ determines affine spaces through a "bias" or displacement in the domain of $\boldsymbol{x}$.

The interest in these convex sets lies in that they admit half-plane representations (or H-rep for short). In other words, they can be reformulated as linear inequalities over the variable $\boldsymbol{x}$, allowing to effectively compress the domain information of $\boldsymbol{y}$ into the space of interest. The half-plane linear inequalities are defined as:

$$\boldsymbol{C}_x\boldsymbol{x} \leq \boldsymbol{d}_x \quad (\boldsymbol{x} \in \mathcal{P}_{x|y}) \tag{5.2}$$

A concrete application of this is the main focus of this chapter. Task-centric motion planning focuses on the Cartesian space. Meanwhile, the actuation capabilities of a

[1] Bounded convex polyhedron are also referred to as polytopes [Fukuda et al., 2004].

robot belong in the configuration space. The formula in (5.1) allows the expression of the actuation capabilities as convex sets in the Cartesian space.

Assume that the task-space domain is represented by the variable $\boldsymbol{x}$ and the joint-space by the variable $\boldsymbol{y}$. The major hypothesis for actuation-aware task-space planning is that if a trajectory defined in the domain of $\boldsymbol{x}(t)$ belongs in the set $\mathcal{P}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{y})$ (the Cartesian-space Constraints (CSC)), then it will also be compatible with the constraints on $\boldsymbol{y}$ (the JSC).

Should this hypothesis prove true, this constitutes a major advantage of the convex sets representations as it allows to represent the actual robot capacities and embed them in a QP for linear MPC as linear inequalities.

A comprehensive introduction to polyhedral sets applied to robotics and biomechanics can be found in Antun Skuric's Thesis[2], including a way to approximate the robot reachable space [Skuric et al., 2022c] and human-robot collaboration scenarios based on an estimate of the human wrench capabilities [Skuric et al., 2021b]; [Skuric et al., 2021a].

5.2.2 Robot Actuation Capacities ↑

The interest of this work lies in generating "smooth" and feasible trajectories. From a mathematical standpoint, this translates to generating $\mathcal{C}^2$-continuous joint trajectories $\boldsymbol{q}(t) \in \mathbb{R}^n$ (for an n-dof serial robot) such that:

$$\begin{aligned} \mathcal{Q} &= \{\boldsymbol{q}(t) \in \mathbb{R}^n \mid \boldsymbol{q} \in [\boldsymbol{q}_m, \boldsymbol{q}_M]\} \\ \dot{\mathcal{Q}} &= \{\dot{\boldsymbol{q}}(t) \in \mathbb{R}^n \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_m, \dot{\boldsymbol{q}}_M]\} \\ \ddot{\mathcal{Q}} &= \{\ddot{\boldsymbol{q}}(t) \in \mathbb{R}^n \mid \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_m, \ddot{\boldsymbol{q}}_M]\} \\ \dddot{\mathcal{Q}} &= \{\dddot{\boldsymbol{q}}(t) \in \mathbb{R}^n \mid \dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_m, \dddot{\boldsymbol{q}}_M]\} \end{aligned} \tag{5.3}$$

revisited ↑

where $\mathcal{Q}$ designates the set of feasible joint configurations; $\dot{\mathcal{Q}}$, $\ddot{\mathcal{Q}}$, $\dddot{\mathcal{Q}}$ respectively refer to velocity, acceleration and jerk constraints, that form the kinematic JSC. More specifically, the desired $\boldsymbol{q}(t)$ is jerk bounded and is smooth up to the second derivative $\ddot{\boldsymbol{q}}(t)$.

The robot motors are non-linear actuators that vary their behaviour according to the exerted joint velocities and torques. Modeling them requires a trade-off between complexity and correctly representing its behaviour. For instance, the dynamic robot model presented in Section 4.2.1 could be employed to bound the maximum joint accelerations from limits on the torque $\boldsymbol{\tau}$ and its derivative $\dot{\boldsymbol{\tau}}$.

Nevertheless, this work assumes constant joint acceleration and jerk capacities represented in $\ddot{\mathcal{Q}}$, $\dddot{\mathcal{Q}}$ respectively. These are conveniently announced by the manufacturer [Franka Emika, n.d.] Even though this constitutes an approximation, it provides an acceptable framework that improves feasibility by going up to a higher order [Gallant et al., 2018].

[2] https://auctus-team.gitlabpages.inria.fr/team-members/antunskuric/

5.2.3 From Joint-space Constraints to Task-space Feasible Motions ↑

The previous section presented the joint-space constraints modeling aspects of the robot. These constraints effectively determine a set of feasible motions that the robot is capable of realizing. Some mathematical formalizations are required in order to start connecting both quantities, which can be achieved through differential methods that project joint-space movements onto the Cartesian space.

A first step towards this objective is to extend the mathematical model introduced in previous chapters to relate robot reconfiguration trajectories to task-space motions. This can be achieved by revisiting the kinematic and dynamic ascpects from previous chapters, through the subsequent time derivatives of the twist:

$$\boldsymbol{\nu} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{5.4}$$ revisited ↑

$$\dot{\boldsymbol{\nu}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{5.5}$$ revisited ↑

$$\ddot{\boldsymbol{\nu}} = \boldsymbol{J}(\boldsymbol{q})\,\dddot{\boldsymbol{q}} + \ddot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\dot{\boldsymbol{q}} + 2\,\dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} \tag{5.6}$$

$$\boldsymbol{\nu},\dot{\boldsymbol{\nu}},\ddot{\boldsymbol{\nu}} \in \mathfrak{se}(3) \qquad\qquad \boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \in \mathbb{R}^n$$

where $\boldsymbol{J}$ refers to the robot analytical Jacobian and $\dot{\boldsymbol{J}}, \ddot{\boldsymbol{J}}$ to its time derivatives; moreover, $\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, \ddot{\boldsymbol{\nu}}$ respectively denote the twist, spatial acceleration and jerks generated by a given joint velocity $\dot{\boldsymbol{q}}$, acceleration $\ddot{\boldsymbol{q}}$ and jerk $\dddot{\boldsymbol{q}}$.

In order to differentiate between linear (as in positional displacement) and angular motions, a simple notation addendum denotes both parts of the Jacobian and its derivatives:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_l \\ \boldsymbol{J}_a \end{bmatrix} \qquad \dot{\boldsymbol{J}} = \begin{bmatrix} \dot{\boldsymbol{J}}_l \\ \dot{\boldsymbol{J}}_a \end{bmatrix} \qquad \ddot{\boldsymbol{J}} = \begin{bmatrix} \ddot{\boldsymbol{J}}_l \\ \ddot{\boldsymbol{J}}_a \end{bmatrix} \tag{5.7}$$

$$\boldsymbol{J}_l, \dot{\boldsymbol{J}}_l, \ddot{\boldsymbol{J}}_l, \boldsymbol{J}_a, \dot{\boldsymbol{J}}_a, \ddot{\boldsymbol{J}}_a \in \mathbb{R}^{3\times n}$$

which separate the upper and lower (in the convention used here) parts of the matrices.

Using the linear components of (5.4) to (5.6) along with the robot capacities as introduced in (5.3) allows for a first approach at relating the JSC to define the feasible sets of Cartesian variables of the same orders, i.e. velocity, acceleration and jerk, which in this case are denoted as $\dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}, \dddot{\boldsymbol{p}}$ respectively to make their relation to the position time derivative clear as well as making a distinction from the 6D pose motions variables. The sets are defined as:

$$\mathcal{P}_{\dot{\boldsymbol{p}}|\dot{\mathcal{Q}}}(\boldsymbol{q}_k) = \left\{\dot{\boldsymbol{p}} \in \mathbb{R}^3 \mid \dot{\boldsymbol{p}} = \boldsymbol{J}_l(\boldsymbol{q}_k)\dot{\boldsymbol{q}},\ \dot{\boldsymbol{q}} \in \dot{\mathcal{Q}}\right\} \tag{5.8}$$ revisited ↓

$$\mathcal{P}_{\ddot{\boldsymbol{p}}|\ddot{\mathcal{Q}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k) = \left\{\ddot{\boldsymbol{p}} \in \mathbb{R}^3 \mid \ddot{\boldsymbol{p}} = \boldsymbol{J}_l(\boldsymbol{q}_k)\ddot{\boldsymbol{q}} + \boldsymbol{B}_{\ddot{\boldsymbol{p}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k),\ \ddot{\boldsymbol{q}} \in \ddot{\mathcal{Q}}\right\} \tag{5.9}$$

$$\boldsymbol{B}_{\ddot{\boldsymbol{p}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k) = \dot{\boldsymbol{J}}_l(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k$$

$$\mathcal{P}_{\dddot{\boldsymbol{p}}|\dddot{\mathcal{Q}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k,\ddot{\boldsymbol{q}}_k) = \left\{\dddot{\boldsymbol{p}} \in \mathbb{R}^3 \mid \dddot{\boldsymbol{p}} = \boldsymbol{J}_l(\boldsymbol{q}_k)\,\dddot{\boldsymbol{q}} + \boldsymbol{B}_{\dddot{\boldsymbol{p}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k,\ddot{\boldsymbol{q}}_k),\ \dddot{\boldsymbol{q}} \in \dddot{\mathcal{Q}}\right\} \tag{5.10}$$

$$\boldsymbol{B}_{\dddot{\boldsymbol{p}}}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k,\ddot{\boldsymbol{q}}_k) = 2\dot{\boldsymbol{J}}_l(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k)\ddot{\boldsymbol{q}}_k + \ddot{\boldsymbol{J}}_l(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k,\ddot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k$$

where $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k$ denote the measured robot state at a time instant k; the

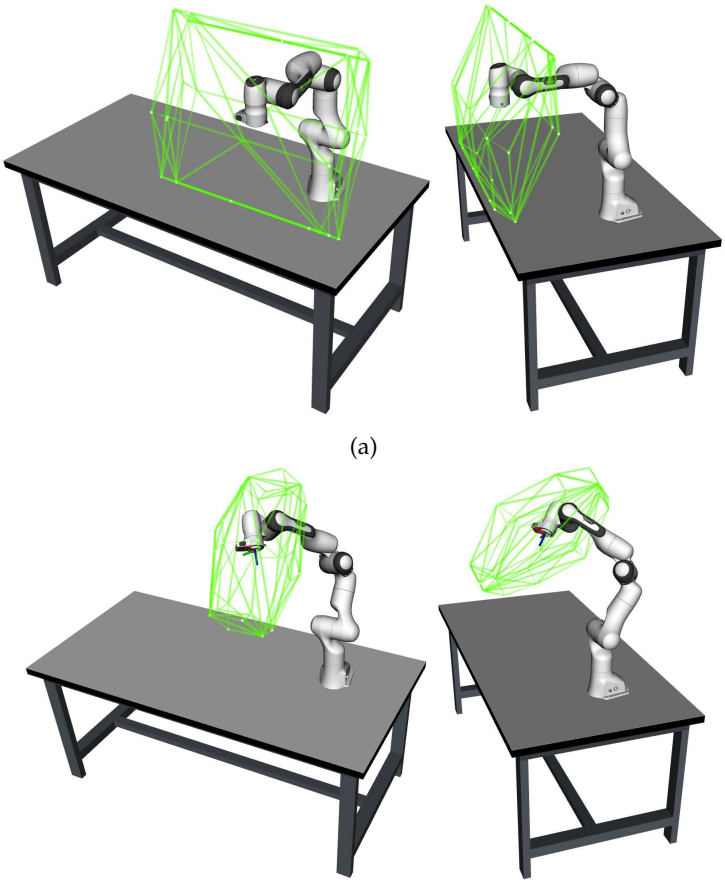


Figure 5.3: The translational velocity polyhedron. It visualizes the feasible velocity set $\mathcal{P}_{\dot{p}|\dot{\mathcal{Q}}}$ as defined in (5.8) once transformed into the "equivalently occupied space". Multiplying the set by a small time interval $\Delta t = 0.15$ corresponds to the space the end-effector would move to at a constant speed for the time period. This visualization showcases two distinct robot configurations: The first one represents a commonly used initial pose. The second configuration serves as an arbitrary example, illustrating how feasible velocities vary depending on the alignment of the robot actuators. This approach helps provide a clear depiction of the robot instantly possible velocities.

sets $\mathcal{P}_{\dot{p}|\dot{\mathcal{Q}}}, \mathcal{P}_{\ddot{p}|\ddot{\mathcal{Q}}}, \mathcal{P}_{\dddot{p}|\dddot{\mathcal{Q}}}$ respectively designate the feasible velocities, accelerations and jerks due to the joint-space capacities in velocities, accelerations and jerks in each case. The dependencies on the measured state were explicitly added to show that some terms do not affect the projection of the joint capacities onto the Cartesian space but rather act as a constant bias or displacement over the origin. These biases are denoted as $\boldsymbol{B}_{\ddot{p}}, \boldsymbol{B}_{\dddot{p}}$ for the acceleration and jerk, respectively.

Note that these biases, as displacements from the origin, they augment the risk of producing an incompatible optimization problem causing the QP solver to fail. In practice, the joint velocity and acceleration measures are also noisy, contributing to a degraded precision in the capacity sets. A strategy adopted in this work is to consider them approximately zero:

$$\boldsymbol{B}_{\ddot{p}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \approx 0 \qquad \boldsymbol{B}_{\dddot{p}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) \approx 0 \tag{5.11}$$

which a relatively good assumption at low to moderate speeds. When operating at higher velocities, eliminating these biases also allows to always include the origin in

the intersection set, increasing the chances of obtaining a compatible QP problem.

In order to visualize the feasible velocities set $\mathcal{P}_{\dot{p}|\dot{\mathcal{Q}}}$ defined in (5.8), a simple fashion consists in converting it to the "equivalently occupied space" of moving at constant speed for a small time delta. To do this, the set can be multiplied by a $\Delta t = 0.15$ to obtain the visualization in Figure 5.3, which is shown for two different robot configurations: the first one corresponds to a rather neutral pose commonly used as a starting configuration; the second configuration shows an arbitrary example of how the feasible velocities evolve according to the alignment of the actuators.

Note that (5.8) to (5.10) effectively take the same convex form as the definition in (5.1), making it possible to be re-formulated as linear inequalities.

Additionally, the sets in (5.8) to (5.10) model the attainable motions in the Cartesian space *without* constraining the orientation of the body. In other words, these formulations do not constraint the angular motion component, effectively leaving complete freedom for the body to change orientation during these motions. This might pose a problem when trying to plan a pure translational trajectory with a fixed orientation. In such case, these definitions need to further constraint the rotational motion to a null value:

$$\mathcal{L}_{\dot{p}|\dot{\mathcal{Q}}}(q_k) = \left\{\dot{p} \in \mathbb{R}^3 \mid \dot{p} = J_l(q_k)\dot{q},\ J_a(q_k)\dot{q} = \mathbf{0_3},\ \dot{q} \in \dot{\mathcal{Q}}\right\} \tag{5.12}$$

revisited ↑ ↓

$$\mathcal{L}_{\ddot{p}|\ddot{\mathcal{Q}}}(q_k, \dot{q}_k) = \left\{\ddot{p} \in \mathbb{R}^3 \mid \ddot{p} = J_l(q_k)\ddot{q} + B_{\ddot{p}},\ J_a(q_k)\ddot{q} = \mathbf{0_3},\ \ddot{q} \in \ddot{\mathcal{Q}}\right\} \tag{5.13}$$

$$\mathcal{L}_{\dddot{p}|\dddot{\mathcal{Q}}}(q_k, \dot{q}_k, \ddot{q}_k) = \left\{\dddot{p} \in \mathbb{R}^3 \mid \dddot{p} = J_l(q_k)\dddot{q} + B_{\dddot{p}},\ J_a(q_k)\dddot{q} = \mathbf{0_3},\ \dddot{q} \in \dddot{\mathcal{Q}}\right\} \tag{5.14}$$

where $\mathbf{0_3}$ designates a zero-valued 3-vector; the bias terms $B_{\ddot{p}}, B_{\dddot{p}}$ coincide with their previous definitions and their dependency on the measured states $q_k, \dot{q}_k, \ddot{q}_k$ was dropped for clarity. The added equality constraint on the angular component effectively restricts the joint motions to those that act on the kernel of the angular Jacobians and its derivatives $J_a, \dot{J}_a, \ddot{J}_a$. This way, the sets designated with $\mathcal{L}$ serve to clearly denote the purely translational feasible motions.

Finally, these definitions helped build an intuitive approach to the feasible sets while also drawing attention to the coupling effect of the model on the orientation and translation. However, the interest of this works lies in full pose trajectories planning. Thus, the same sets must be extended for 6D motions, which fortunately can be done in a straightforward way:

$$\mathcal{P}_{\nu|\dot{\mathcal{Q}}}(q_k) = \left\{\nu \in \mathfrak{se}(3) \mid \nu = J(q_k)\dot{q},\ \dot{q} \in \dot{\mathcal{Q}}\right\} \tag{5.15}$$

revisited ↑ ↓

$$\mathcal{P}_{\dot{\nu}|\ddot{\mathcal{Q}}}(q_k, \dot{q}_k) = \left\{\dot{\nu} \in \mathfrak{se}(3) \mid \dot{\nu} = J(q_k)\ddot{q} + B_{\dot{\nu}}(q_k, \dot{q}_k),\ \ddot{q} \in \ddot{\mathcal{Q}}\right\} \tag{5.16}$$

$$B_{\dot{\nu}}(q_k, \dot{q}_k) = \dot{J}(q_k, \dot{q}_k)\dot{q}_k$$

$$\mathcal{P}_{\ddot{\nu}|\dddot{\mathcal{Q}}}(q_k, \dot{q}_k, \ddot{q}_k) = \left\{\ddot{\nu} \in \mathbb{R}^3 \mid \ddot{\nu} = J(q_k)\dddot{q} + B_{\ddot{\nu}}(q_k, \dot{q}_k, \ddot{q}_k),\ \dddot{q} \in \dddot{\mathcal{Q}}\right\} \tag{5.17}$$

$$B_{\ddot{\nu}}(q_k, \dot{q}_k, \ddot{q}_k) = 2\dot{J}(q_k, \dot{q}_k)\ddot{q}_k + \ddot{J}(q_k, \dot{q}_k, \ddot{q}_k)\dot{q}_k$$

where most of the previous conclusions still stand but the notation should make it clear now that the full Jacobian matrix and its derivatives are used. This way, the full pose derivative variables can be constrained (i.e. the twist ν, spatial acceleration $\dot{\nu}$ and jerk $\ddot{\nu}$).

5.2.4 From Joint Bounds to Convex Reachable Workspace ↑

The reachable workspace due to the joint bounds $\mathcal{Q}$ is purposely omitted from the previous section because its consideration requires some special treatment. To begin, the reachable workspace may be modeled as the set of reachable positions or poses in the workspace. For a first approach, the focus is kept on the positional sense of the set.

For instance, the set of feasible positions that can be achieved by a robot end-effector can be defined as:

$$\mathcal{P}_{\boldsymbol{p}|\mathcal{Q}}(\boldsymbol{q}_k) = \left\{\boldsymbol{p} \in \mathbb{R}^3 \mid \boldsymbol{p} = \mathrm{FK}(\boldsymbol{q}),\ \boldsymbol{q} \in [\boldsymbol{q}_m, \boldsymbol{q}_M]\right\} \tag{5.18}$$

which, unlike the definitions from the previous section, employs a non-linear Forward Kinematics (FK) function[3]. This provokes that no convexity characteristics can be inferred from the set $\mathcal{P}_{\boldsymbol{p}|\mathcal{Q}}$, which is a problem for the intended use in this work (to be embedded into optimization algorithms).

To circumvent this problem, a convex set can be constructed through a variational approach, by employing a first order expansion on the position. Assuming a discretized position trajectory at a time step k, yields:

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + d\boldsymbol{p}_k \qquad d\boldsymbol{p}_k = \boldsymbol{J}_l d\boldsymbol{q}_k \tag{5.19}$$

where the main interest lies in computing $\boldsymbol{p}_{k+1}$ given a configuration differential $d\boldsymbol{q}$. Then, redefining this infinitely small variation to evaluate the available joint motion (signifiying it is no longer assumed to be small) yields a configuration delta $\Delta\boldsymbol{q}$. It can be computed to be the distance towards each of the joint bounds to define a set with respect to the current configuration:

$$\Delta\boldsymbol{q}_k \in [\Delta\boldsymbol{q}_m(\boldsymbol{q}_k), \Delta\boldsymbol{q}_M(\boldsymbol{q}_k)] \qquad \Delta\boldsymbol{q}_m(\boldsymbol{q}_k) = \boldsymbol{q}_m - \boldsymbol{q}_k \qquad \Delta\boldsymbol{q}_M(\boldsymbol{q}_k) = \boldsymbol{q}_M - \boldsymbol{q}_k \tag{5.20}$$

Finally, integrating these expressions allows defining a convex reachable space set as:

$$\begin{gathered}\hat{\mathcal{P}}_{\boldsymbol{p}|\mathcal{Q}}(\boldsymbol{q}_k) = \left\{\boldsymbol{p} \in \mathbb{R}^3 \mid\ \boldsymbol{p}(\Delta\boldsymbol{q}),\ \Delta\boldsymbol{q} \in [\Delta\boldsymbol{q}_m, \Delta\boldsymbol{q}_M]\right\} \\ \boldsymbol{p}(\Delta\boldsymbol{q}) \sim \boldsymbol{p}_k + \boldsymbol{J}_l(\boldsymbol{q}_k)\Delta\boldsymbol{q} \qquad\qquad \boldsymbol{p}_k = \mathrm{FK}(\boldsymbol{q}_k)\end{gathered} \tag{5.21}$$

revisited ↓

where this time around, the term $\boldsymbol{p}_k$ provides the constant bias for the convex set and can be computed by using the FK on the measured robot configuration $\boldsymbol{q}_k$. The definitions for $\Delta\boldsymbol{q}_m, \Delta\boldsymbol{q}_M$ are provided in (5.20).

This definition constitutes a rough approximation of the instantaneously feasible space, because, in reality, this space is not necessarily convex. Yet, for configurations $\boldsymbol{q}$ relatively close to the joint bounds (small $\Delta\boldsymbol{q}$ values), the approximation becomes progressively better:

$$\lim_{\Delta\boldsymbol{q}\to 0} \hat{\mathcal{P}}_{\boldsymbol{p}|\mathcal{Q}}(\boldsymbol{q}_k) \approx \mathcal{P}_{\boldsymbol{p}|\mathcal{Q}}(\boldsymbol{q}_k) \tag{5.22}$$

Moreover, bigger $\Delta\boldsymbol{q}$ values imply that the joint bounds are far enough that even

[3] Prioritizing notation clarity over consistency, the Forward Kinematics function is denoted with the same name (as $\mathrm{FK}(\boldsymbol{q})$) both in the *position* sense (which outputs a 3-vector position) as well as in the full *pose* sense, which outputs a homogeneous transformation matrix in $\mathbb{SE}(3)$.

if the approximation is bad, their consideration is not urgent. In a worst case scenario that the robot motion tends towards the joint bounds, the increasingly better instantaneous precision of the approximation helps appropriately consider the joint bounds, provided that the convex set is re-computed frequently (which is an acceptable assumption in collaborative robotics scenarios that require reactivity).

The reachable space is visualized in Figure 5.4 for two configurations. The figure shows $\mathcal{P}_{p|\mathcal{Q}}$ from (5.21) with a Jacobian $\boldsymbol{J}_l$ multiplied by a factor of 0.15. This factor is necessary for visualization purposes because the approximation is rather coarse in extremes that correspond to the farthest from the joint bounds. The first configuration, shown in Figure 5.4(a), corresponds to a commonly used starting pose. This is a rather neutral configuration and it shows a symmetrical reachable space. In contrast, Figure 5.4(b) shows an extreme case of a twisted configuration where the robot was purposely put near the joint bounds. In such case, the approximation of the reachable space through the projections in the direction of the joint bounds becomes more precise and, as shown in the image, the robot is completely unable to move in this direction.

Just like with (5.8) to (5.10), the convex reachable space definition in (5.21) works under the assumption of an unrestricted orientation. In order to approximate the reachable space that can be attained without reorienting the body, the same principles introduced for (5.12) to (5.14) can be applied. This further restricts the joint configuration motions to the kernel of the angular component of the Jacobian through an extra equality constraint:

$$\mathcal{L}_{p|\mathcal{Q}}(\boldsymbol{q}_k) = \left\{ \boldsymbol{p} \in \mathbb{R}^3 \mid \boldsymbol{p}(\Delta\boldsymbol{q}),\ \boldsymbol{J}_a(\boldsymbol{q}_k)\Delta\boldsymbol{q} = \boldsymbol{0_3},\ \Delta\boldsymbol{q} \in [\Delta\boldsymbol{q}_m, \Delta\boldsymbol{q}_M] \right\} \tag{5.23}$$

revisited ↑ ↓

where $\Delta\boldsymbol{q}_m, \Delta\boldsymbol{q}_M$ are defined in (5.20) and $\boldsymbol{p}(\Delta\boldsymbol{q})$ is defined in (5.21).

The same definition in (5.18) can be made over the realizable poses of the robot as:

$$\mathcal{P}_{\mathcal{X}|\mathcal{Q}}(\boldsymbol{q}_k) = \{ \mathcal{X} \in \mathbb{SE}(3) \mid \mathcal{X} = \mathrm{FK}(\boldsymbol{q}),\ \boldsymbol{q} \in [\boldsymbol{q}_m, \boldsymbol{q}_M] \} \tag{5.24}$$

again without any potential inferences on the convexity properties of the set. However, the same expression in (5.21) cannot be straightforwardly extended for the poses. However, because this time the objective is to also represent the feasible orientations, the additional restrictions from (5.23) are not necessary. In fact, in order to define a convex set that represents the instantaneous reachable poses, it is required to go through an auxiliary variable: the exponential pose representations introduced in a general way in Chapter 2 and in a more applied way (to MPC) in Chapters 3 and 4.

To do so, the set is to be defined on to the logarithm of the pose $\boldsymbol{\xi} = \log(\mathcal{X})$. Then, revisiting the first-order algebraic expression in the tangent space one obtains:

$$\boldsymbol{\xi}_{k+1} = \log(e^{\boldsymbol{\xi}_k} e^{\Delta t \boldsymbol{\nu}_k}) \approx \boldsymbol{\xi}_k + \mathrm{dlog}_{\boldsymbol{\xi}_k} \Delta t \boldsymbol{\nu}_k \tag{5.25}$$

revisited ↑

that relates the pose logarithm $\boldsymbol{\xi}$ with the twist $\boldsymbol{\nu}$. Now, applying the same logic as in (5.21) is possible:

$$\begin{gathered} \hat{\mathcal{P}}_{\boldsymbol{\xi}|\mathcal{Q}}(\boldsymbol{q}_k) = \{ \boldsymbol{\xi} \in \mathfrak{se}(3) \mid \boldsymbol{\xi}(\Delta\boldsymbol{q}),\ \Delta\boldsymbol{q} \in [\boldsymbol{q}_m - \boldsymbol{q}_k, \boldsymbol{q}_M - \boldsymbol{q}_k] \} \\ \boldsymbol{\xi}(\Delta\boldsymbol{q}) \sim \boldsymbol{\xi}_k + \mathrm{dlog}_{\boldsymbol{\xi}_k} \boldsymbol{J} \Delta\boldsymbol{q} \qquad \boldsymbol{\xi}_k = \log(\mathcal{X}_k) \end{gathered} \tag{5.26}$$

revisited ↑ ↓

where $\mathcal{P}_{\boldsymbol{\xi}|\mathcal{Q}}$ defines a convex set over the pose logarithm. This set is defined on

the tangent space of the poses, $\mathfrak{se}(3)$. The interpretability of this expression is not directly translated for the space of the poses, $\mathbb{SE}(3)$. However, for optimization purposes, as the proposed approach in this work is based on using the pose tangent space, this effectively provides the necessary means to limit the solution space. Then, once the optimization is finished, the actual pose can be recovered by computing the exponential of the solution.

5.3 Actuation-aware Task-space Re-planning ↑

The current work is concerned with the online generation of trajectories subject to constraints originated from safety risks and/or feasibility restrictions, due to the limited actuation capacities of the robot.

The core contribution of this chapter concerns how to integrate the feasibility constraints introduced in Sections 5.2.3 and 5.2.4 into the task-space re-planning scheme of the proposed Model Predictive Control (MPC).

5.3.1 Some MPC notions and notations ↑

This section quickly revisits some definitions for generic MPC from Chapter 4 that are necessary for the contributions provided in the current chapter.

Assuming a d-sized state and m-sized input system, it can be described with a generic discrete time linear system (indexed with n) as

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{B}\boldsymbol{u}_n \qquad \boldsymbol{A} \in \mathbb{R}^{d\times d},\ \boldsymbol{B} \in \mathbb{R}^{d\times m} \tag{5.27}$$ revisited ↑

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent its state and input matrices whereas $\boldsymbol{x}$ and $\boldsymbol{u}$ are the state and input vectors. Smooth point to point pose motion generation can be achieved by generating trajectories at the acceleration level. Choosing the control input at the task level to be the acceleration, allows predicting the evolution of the end effector (system) state — pose and twist — over a short period of time to generate a jerk-bounded motion.

Discretizing a receding time window in h time steps of duration T starting at time t_n (total duration hT) yields the discretized time steps inside the horizon, defined as $t_k = t_n + k\mathrm{T}$ for $k = 0, 1, \ldots, \mathrm{h}$ and one can obtain $\boldsymbol{x}(t_k) = \boldsymbol{x}_{k|n}$, $\boldsymbol{u}(t_k) = \boldsymbol{u}_{k|n}$. A linear state propagation over the receding horizon expression is defined as:

$$\boldsymbol{x}_{k+1|n} = \boldsymbol{A}\boldsymbol{x}_{k|n} + \boldsymbol{B}\boldsymbol{u}_{k|n} \qquad \text{for } k = 0, 1, \ldots, \mathrm{h}-1 \tag{5.28}$$ revisited ↑

Subsequent sections drop the n subscript to simplify the notation, making $\boldsymbol{x}_{k|n}$ equivalent to $\boldsymbol{x}_k$. This propagation can be synthesized into a simple linear algebraic expression:

$$\overline{\boldsymbol{X}}(\boldsymbol{x}_0, \underline{\boldsymbol{U}}) = \hat{\boldsymbol{A}}\underline{\boldsymbol{U}} + \hat{\boldsymbol{B}}\boldsymbol{x}_0 \tag{5.29}$$ revisited ↑

where:

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_h \end{bmatrix}}_{\overline{X}} = \underbrace{\begin{bmatrix} B & & & & \\ AB & B & & & \\ A^2B & AB & B & & \\ \vdots & & & & \\ A^{h-1}B & A^{h-2}B & \dots & AB & B \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{h-1} \end{bmatrix}}_{\underline{U}} + \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^h \end{bmatrix}}_{\hat{B}} x_0 \tag{5.30}$$ revisited ↑

This formulation can be employed to define a pose motion generation problem as an optimization problem over the control input that minimizes the distance towards a target state. Because the state can be computed from a control input horizon, the optimization maintains the capability of recovering the state throughout the horizon, allowing to also consider state constraints while reducing the optimization variables (as it only needs to consider the control inputs), effectively reducing resolution times.

This problem is defined as a MPC control scheme that, thanks to the linear algebraic state propagation, can be formulated as a Quadratic Program (QP):

$$\min_{u} \left\| \overline{X}_t - \overline{X}(x_0, \underline{U}) \right\|_P^2 + \left\| \underline{U} \right\|_R^2 \tag{5.31}$$ revisited ↑

$$C_{x_m} \leq C_x x_k \leq C_{x_M} \tag{5.32}$$ revisited ↑

$$C_{u_m} \leq C_u u_k \leq C_{u_M} \tag{5.33}$$ revisited ↑

$$C_{\dot{u}_m} \leq C_{\dot{u}} \dot{u}_k \leq C_{\dot{u}_M} \tag{5.34}$$ revisited ↑

$$\text{with:} \quad \dot{u}_k = \frac{(u_{k+1} - u_k)}{\mathrm{T}}$$

optimizing over the control inputs u. Where P, R designate semi-positive definite matrices containing weighting terms for the input throughout the horizon and $\overline{X}_t$ corresponds to the target state vector repeated throughout the horizon.

Furthermore, (4.40) allows for an explicit expression of state bounds. Meanwhile, (4.41) allow a more generic expression of linear constraints over the control input. Note that in practice, for this formulation, (4.40) is also formulated as linear constraints over the control input, a consequence of (5.29). However, the versatility of this linear constraints is the main focus of the chapter, that is employed to achieve "actuation-aware" re-planning.

5.3.2 About the Linear Inequality Constraints ↑

The previous section presented a generic MPC formulation that can be applied for generalized systems. In the interest of motion generation with MPC in mind, the state and input of the system are specialized either for position control or for pose control.

In a collaborative robotics scenario, and contemplating the proposed control architecture, a common need from the control perspective is how to constraint the motions for needs that go beyond actuation limits. These may originate from safety needs (operating velocity), workspace limitations (for example, with safety zones or

virtual walls), task limitations (for example that explicitly require smooth motions), user preferences, etc. In fact, one can imagine any number of applications that impose arbitrary restrictions over the motion and these need to be considered by the planning algorithm continuously.

In order for these to be readily integrated into the task-space re-planning scheme proposed as an MPC, they need to be expressed as linear inequality constraints over either the state or the input of the system, as introduced in the most generic way that a QP admits in (5.32) to (5.34). Yet, they can also be used to formulate simple variable bounds or zonotopes [Bouchard et al., 2009]. For instance, the interval set of a variable defined as:

$$\mathcal{P}_{\boldsymbol{y}|\mathcal{Y}} = \{\boldsymbol{y} \in \mathbb{R}^3 \mid \forall \boldsymbol{y} \in [\boldsymbol{y}_m, \boldsymbol{y}_M]\} \tag{5.35}$$

where $\boldsymbol{y}$ generically represents the variable constrained within some set $\mathcal{Y}$ with limits $\boldsymbol{y}_m, \boldsymbol{y}_M$ (its upper and lower bounds), also admits a linear inequality representation as:

$$\boldsymbol{C}_l \leq \boldsymbol{C}_y \boldsymbol{y} \leq \boldsymbol{C}_u \tag{5.36}$$

or equivalently:

$$\begin{bmatrix} \boldsymbol{C}_y \\ -\boldsymbol{C}_y \end{bmatrix} \boldsymbol{y} \leq \begin{bmatrix} \boldsymbol{C}_u \\ -\boldsymbol{C}_l \end{bmatrix} \tag{5.37}$$

where $\boldsymbol{C}_y$ is an identity and $\boldsymbol{C}_l, \boldsymbol{C}_u$ contains the variable bounds. This last form shows that the linear inequality constraints can be used to embed complex constraints in the form of (5.2), because (5.35) is in fact a special case of (5.1).

The objective of this section is to show how this kind of formulation can be used to express Cartesian-space Constraints (CSC), either from the feasibility sets introduced in Section 5.2 or from other workspace/task restrictions.

5.3.3 Acceleration-based Position Re-planning ↑

For a pure translational motion re-planning (i.e. position control), the generic MPC formulations presented in Section 5.3.1 need to be specialized. More concretely, the system state and input need to be defined. Based on a second order expansion of the position:

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \mathrm{T}\, \dot{\boldsymbol{p}}_k + \frac{\mathrm{T}^2}{2} \ddot{\boldsymbol{p}}_k \tag{5.38}$$

$$\dot{\boldsymbol{p}}_{k+1} = \dot{\boldsymbol{p}}_k + \mathrm{T}\, \ddot{\boldsymbol{p}}_k \tag{5.39}$$

one can define a system based on the acceleration as the control input:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \dot{\boldsymbol{p}}_k \end{bmatrix} \qquad \boldsymbol{u}_k = \ddot{\boldsymbol{p}}_k \qquad \boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{p}_t \\ \boldsymbol{0} \end{bmatrix} \tag{5.40}$$

where $\boldsymbol{p}, \dot{\boldsymbol{p}}$ respectively designate the position and velocity and $\ddot{\boldsymbol{p}}$ denotes the acceleration. Meanwhile, $\boldsymbol{x}_t$ contains the target state: the target position and a null velocity. Then, to complete the state-space system definition, the expressions for

the $\boldsymbol{A}, \boldsymbol{B}$ matrices are required:

$$\boldsymbol{A}_k = \begin{bmatrix} \mathbf{I}_3 & \mathrm{T}\,\mathbf{I}_3 \\ \mathbf{0_3} & \mathbf{I}_3 \end{bmatrix} \qquad \boldsymbol{B}_k = \begin{bmatrix} \frac{\mathrm{T}^2}{2}\,\mathbf{I}_3 \\ \mathrm{T}\,\mathbf{I}_3 \end{bmatrix} \tag{5.41}$$

which provide all the necessary elements to complete the acceleration-based definition of the MPC for reactive position re-planning.

5.3.4 Position Re-planning Constraints ↑

In the interest of safety-aware planning, a multitude of approaches can be embedded as constraints. For instance, some strategies to increase safety through velocity modulation employ damage risks mitigation measures [Palleschi et al., 2021]; furthermore the technical specification for collaborative robotics contemplates velocity-limited operation zones [ISO/TS-15066, 2016]. In terms of position control, safety-related zones and velocity limits can be denoted with an $\mathcal{S}$ as:

$$\mathcal{P}_{\boldsymbol{p}|\mathcal{S}} = \{\boldsymbol{p} \in \mathbb{R}^3 \mid \forall \boldsymbol{v} \in [\boldsymbol{p}_{m|\mathcal{S}}, \boldsymbol{p}_{M|\mathcal{S}}]\} \tag{5.42}$$

$$\mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{S}} = \{\dot{\boldsymbol{p}} \in \mathbb{R}^3 \mid \forall \dot{\boldsymbol{p}} \in [\dot{\boldsymbol{p}}_{m|\mathcal{S}}, \dot{\boldsymbol{p}}_{M|\mathcal{S}}]\} \tag{5.43}$$

where $\mathcal{P}_{\boldsymbol{p}|\mathcal{S}}$ designates the space deemed safe for the robot to operate in; while $\mathcal{P}_{\boldsymbol{v}|\mathcal{S}}$ designates the maximum operating velocity.

Another way to effectively reduce the reachable geometrical workspace is through a time-varying version of $\mathcal{P}_{\boldsymbol{v}|\mathcal{S}}(t)$, by gradually reducing the maximum velocities when the robot approaches the limits of the allowed space. However, while some constraints must be time-varying, in this case, it is counter-productive as it adds unnecessary dynamics to constraints that can already be formulated explicitly, making the horizon re-planning more robust.

Meanwhile, manufacturers also provide task-space related maximum velocities, accelerations and jerks, which can be denoted with an $\mathcal{M}$ as:

$$\mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{M}} = \{\boldsymbol{v} \in \mathbb{R}^3 \mid \forall \dot{\boldsymbol{p}} \in [\dot{\boldsymbol{p}}_m, \dot{\boldsymbol{p}}_M]\} \tag{5.44}$$

$$\mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{M}} = \{\boldsymbol{v} \in \mathbb{R}^3 \mid \forall \ddot{\boldsymbol{p}} \in [\ddot{\boldsymbol{p}}_m, \ddot{\boldsymbol{p}}_M]\} \tag{5.45}$$

$$\mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{M}} = \{\boldsymbol{v} \in \mathbb{R}^3 \mid \forall \dddot{\boldsymbol{p}} \in [\dddot{\boldsymbol{p}}_m, \dddot{\boldsymbol{p}}_M]\} \tag{5.46}$$

These kind of constraints make sense from a usability point of view as they are easy to interpret and practical from a task-centric design proposed in this work. Yet, the linear inequalities form show that they could potentially be used to express more complex convex sets like the ones introduced in Section 5.2.1 that admit the same form as (5.1). For instance, this means that the shape of the allowed geometrical space in $\mathcal{P}_{\boldsymbol{p}|\mathcal{S}}$ does not need to be rectangular (as in virtual walls parallel to the world frame planes). These arbitrary and application-dependent sets are denoted with $\mathcal{A}$.

Altogether, the intersection of all set of CSC related to the task-space, are denoted as *workspace* constraints with a $\mathcal{W}$:

$$\mathcal{P}_{\boldsymbol{p}|\mathcal{W}} = \mathcal{P}_{\boldsymbol{p}|\mathcal{S}} \cap \mathcal{P}_{\boldsymbol{p}|\mathcal{M}} \cap \mathcal{P}_{\boldsymbol{p}|\mathcal{A}} \tag{5.47}$$

$$\mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{W}} = \mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{S}} \cap \mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{M}} \cap \mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{A}} \tag{5.48}$$

$$\mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{W}} = \mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{S}} \cap \mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{M}} \cap \mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{A}} \tag{5.49}$$

$$\mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{W}} = \mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{S}} \cap \mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{M}} \cap \mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{A}} \tag{5.50}$$

Finally, the focus of this section is on how to add actuation awareness onto the task-space re-planning, while also considering the workspace constraints. To complete the position control MPC problem that contemplates all these restrictions, this definition is necessary:

$$\boldsymbol{p}_k = \mathrm{FK}(\boldsymbol{q}_k) \tag{5.51}$$

$$\dot{\boldsymbol{p}}_k = \boldsymbol{J}_l \boldsymbol{q}_k \tag{5.52}$$

$$\boldsymbol{p} \in \mathcal{P}_{\boldsymbol{p}} \qquad \mathcal{P}_{\boldsymbol{p}} = \mathcal{P}_{\boldsymbol{p}|\mathcal{Q}}(\boldsymbol{q}_k) \cap \mathcal{P}_{\boldsymbol{p}|\mathcal{W}} \tag{5.53}$$

$$\dot{\boldsymbol{p}} \in \mathcal{P}_{\dot{\boldsymbol{p}}} \qquad \mathcal{P}_{\dot{\boldsymbol{p}}} = \mathcal{P}_{\dot{\boldsymbol{p}}|\dot{\mathcal{Q}}}(\boldsymbol{q}_k) \cap \mathcal{P}_{\dot{\boldsymbol{p}}|\mathcal{W}} \tag{5.54}$$

$$\ddot{\boldsymbol{p}} \in \mathcal{P}_{\ddot{\boldsymbol{p}}} \qquad \mathcal{P}_{\ddot{\boldsymbol{p}}} = \mathcal{P}_{\ddot{\boldsymbol{p}}|\ddot{\mathcal{Q}}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \cap \mathcal{P}_{\ddot{\boldsymbol{p}}|\mathcal{W}} \tag{5.55}$$

$$\dddot{\boldsymbol{p}} \in \mathcal{P}_{\dddot{\boldsymbol{p}}} \qquad \mathcal{P}_{\dddot{\boldsymbol{p}}} = \mathcal{P}_{\dddot{\boldsymbol{p}}|\dddot{\mathcal{Q}}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) \cap \mathcal{P}_{\dddot{\boldsymbol{p}}|\mathcal{W}} \tag{5.56}$$

where $\boldsymbol{p}_k, \dot{\boldsymbol{p}}_k$ respectively designate the initial position and velocity. These equations express the convex sets that need to be to be formulated as linear constraints as in (4.40) and (4.41). If fixed orientation is required along with position re-planning, the feasible sets may be replaced with their pure translation equivalent based on $\mathcal{L}$, as defined in Sections 5.2.3 and 5.2.4.

5.3.5 Position Re-planning Constraints Simulations ↑

In order to offer an intuitive comprehension of the effects of the constraints as formulated previously, this section presents a series of simulations. The simulations included contemplate 3 scenarios:

1. Zonotope constraints with zero initial conditions shown in Figure 5.5
2. Zonotope constraints with non-zero initial conditions shown in Figure 5.6
3. Polyhedral constraints with non-zero initial conditions conditions shown in Figure 5.7

In all cases, a single horizon with a fixed target position was computed and visualized. The horizon duration is hT=4s (as shown in the trajectories), with a configuration h=100 and T=40ms. The input weight is $\gamma = 5\frac{T^2}{2}$, and the terminal state weight is:

$$\boldsymbol{P}_T = \begin{bmatrix} \mathbf{I}_{3\times3}\ 100 & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \mathbf{I}_{3\times3}\ 100\mathrm{T} \end{bmatrix} \tag{5.57}$$

as defined in (4.29) and (4.30).

Furthermore, in all the scenarios, the target position is outside of the position constraints $\mathcal{P}_p$ to showcase what happens when the target is unreachable. In such case, the distance-minimizing cost function as formulated in the MPC drives the system towards it until it reaches the closest point within the constraints.

On the same note, it can also be seen in all velocity horizons that the ending velocity tends towards the origin (i.e. a zero value). This is a feature of the definition in (5.40) that drives the robot towards the target while also trying to inhibit the motion. This definition has a double effect: for one, it favors a stable convergence towards the target position (to arrive at a motionless state); the second effect is related to the fail safe mechanism: in case the MPC optimization fails, the robot will continue to execute the last successfully computed horizon. This target state ensures the horizon always converges to a more controllable (slower) state, which should in theory also increase the chances of spontaneously resolving the situation.

Another significant result that can be appreciated across all scenarios is the resulting acceleration curve: it approximately resembles an optimal planning algorithm based on a trapezoidal acceleration profile. This showcases that a close to optimal use of the available capacities (kinematic constraints in velocity and acceleration). In the case of polyhedral constraints, the same conclusion can be drawn regarding the saturation of the available capabilities. However, given the irregular shape of the polyhedron, the curves no longer seem to align with a trapezoidal acceleration profile. Yet, as can be appreciated in the 3D plots, the resulting horizon uses the maximum available capacities by aligning with the faces of the polyhedron.

The simulation 1, shown in Figure 5.5, constitutes the simplest scenario. The initial position is at the origin and the initial velocity is null. This result highlights two main results:

- The path taken constitutes shortest path towards the terminal position. The terminal position, as explained above, does not coincide with the target one but rather the closest possible, due to the imposed constraints.
- The resemblance to a trapezoidal acceleration profile is the most clear from all scenarios.

In simulation 2 (shown in Figure 5.5), the objective is to show a non trivial initial state. While the position still starts at the origin, the initial velocity is set to $\dot{\boldsymbol{p}} = [2\ 1.3\ 1]$. The resulting position trajectory highlights the effects of these conditions: due to the limited kinematics capacities, the optimal trajectory tends approximates a curved path towards the ending position. In a hypothetical infinite capacities scenario, the path would be swiftly corrected towards a straight line.

The relevance of this last result becomes even more prominent in an online replanning scenario like the collaborative tasks proposed in this work. In such case, the MPC would has to continuously compute new trajectories based on evolving initial conditions, closing the feedback loop. Furthermore, this also demonstrates that in a scenario where the task (cost function) is switched or time-varying, the obtained trajectories remain coherent with the available kinematic capacities, showcasing a smooth transition that takes into account the starting point.

Finally, Figure 5.7 shows simulation 3. The objective is to showcase the combined situations of the previous scenarios with a non trivial velocity convex set. For instance, this is analogous to employing the feasible velocity sets devised from the joint capacities. The result shows the same conclusions as before can be extrapolating, keeping in mind that the velocity capacities are now heavily restricted by a polyhedron: the resulting position shows a curvy path that smoothly transitions towards the ending position; while the velocity path saturates the available capacity by "sticking " to the faces of the polyhedron.

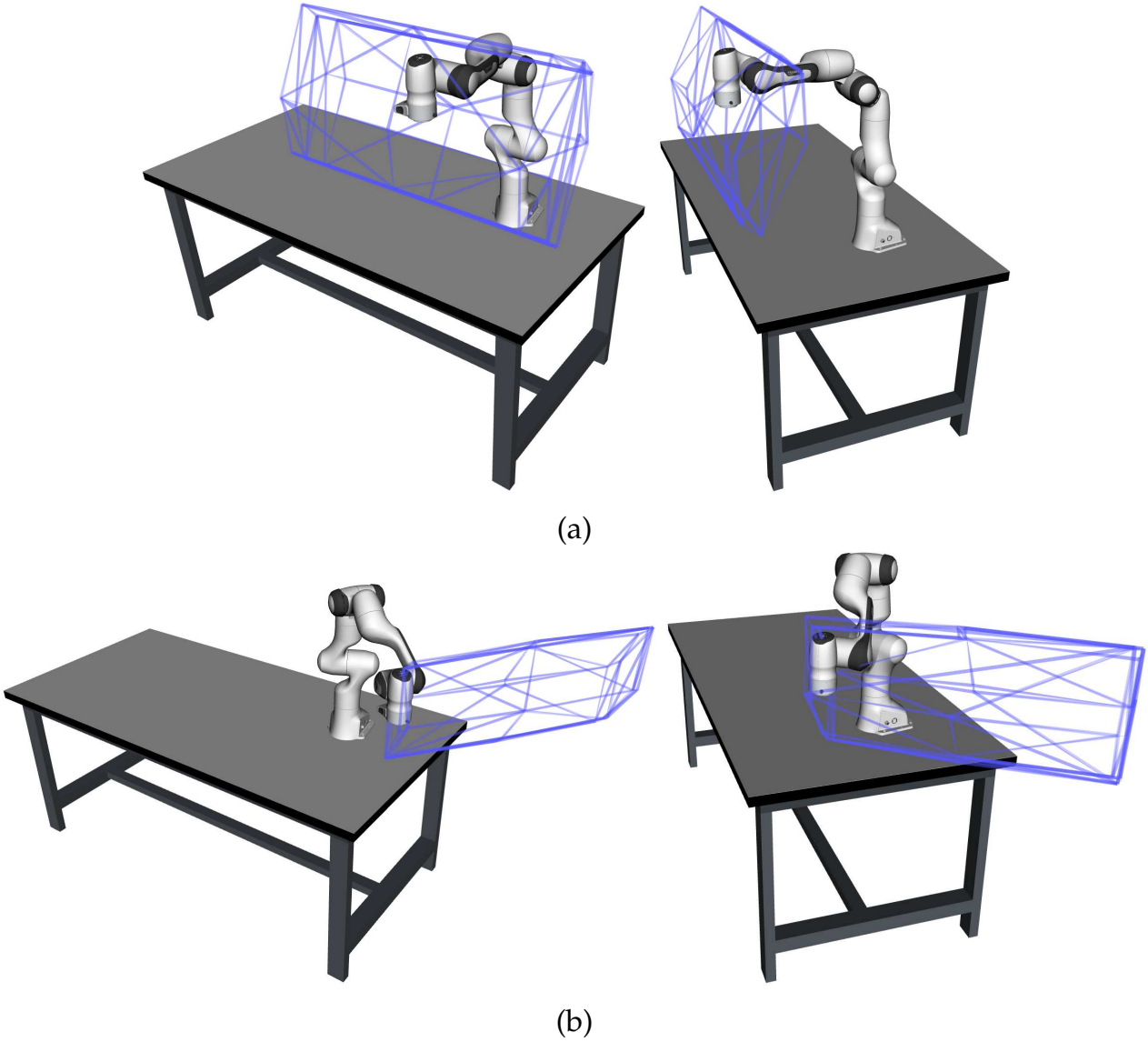


Figure 5.4: The translational convex reachable space approximation polyhedron. This illustrates the attainable space for two different configurations. It presents $\mathcal{P}_{\boldsymbol{p}|\mathcal{Q}}$ obtained from (5.21) with the multiplication of the Jacobian $\boldsymbol{J}_l$ by a scaling factor of 0.15. This scaling is essential for visualization since the approximation becomes coarse in extreme cases, particularly those farthest from the joint boundaries. The first configuration, displayed in Figure 5.4(a), represents a commonly used initial pose. This configuration is relatively neutral and displays a symmetrical attainable region. In contrast, Figure 5.4(b) illustrates an extreme scenario where the robot intentionally approaches the joint boundaries. In such instances, the approximation of the attainable region improves significantly in the projections aligned with the joint boundaries. As seen in the image, the robot mobility in this direction becomes severely restricted.

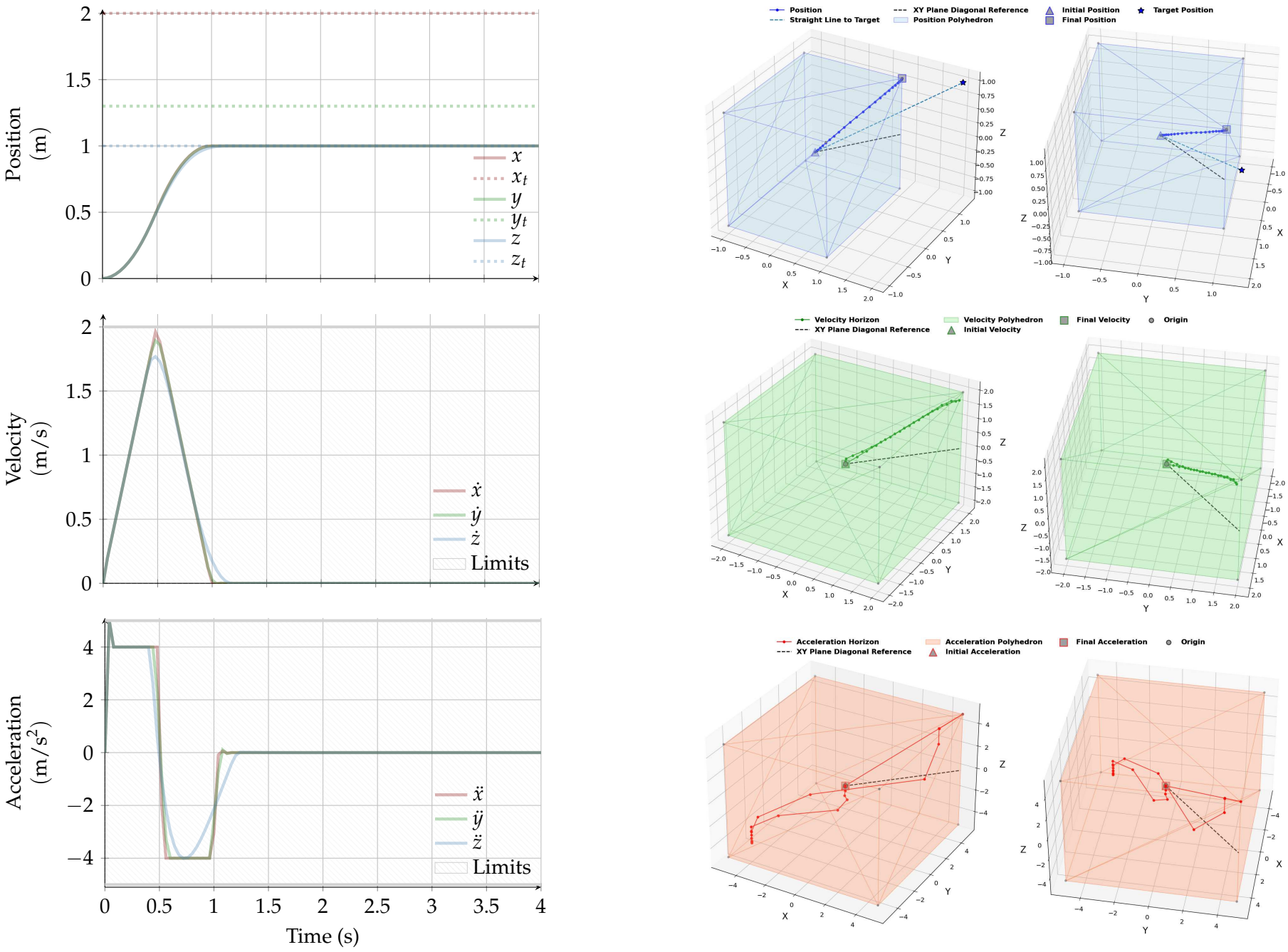


Figure 5.5: Example horizon with zonotope constraints and zero initial conditions: The simulation begins at the origin with zero velocity, illustrating the system response to constraints. The trajectory demonstrates the shortest path towards the nearest feasible point within the constraints, since the target position is unreachable due to the imposed limitations. Notably, the trajectory closely resembles an optimal planning algorithm utilizing a trapezoidal acceleration profile, showcasing efficient utilization of available kinematic resources.

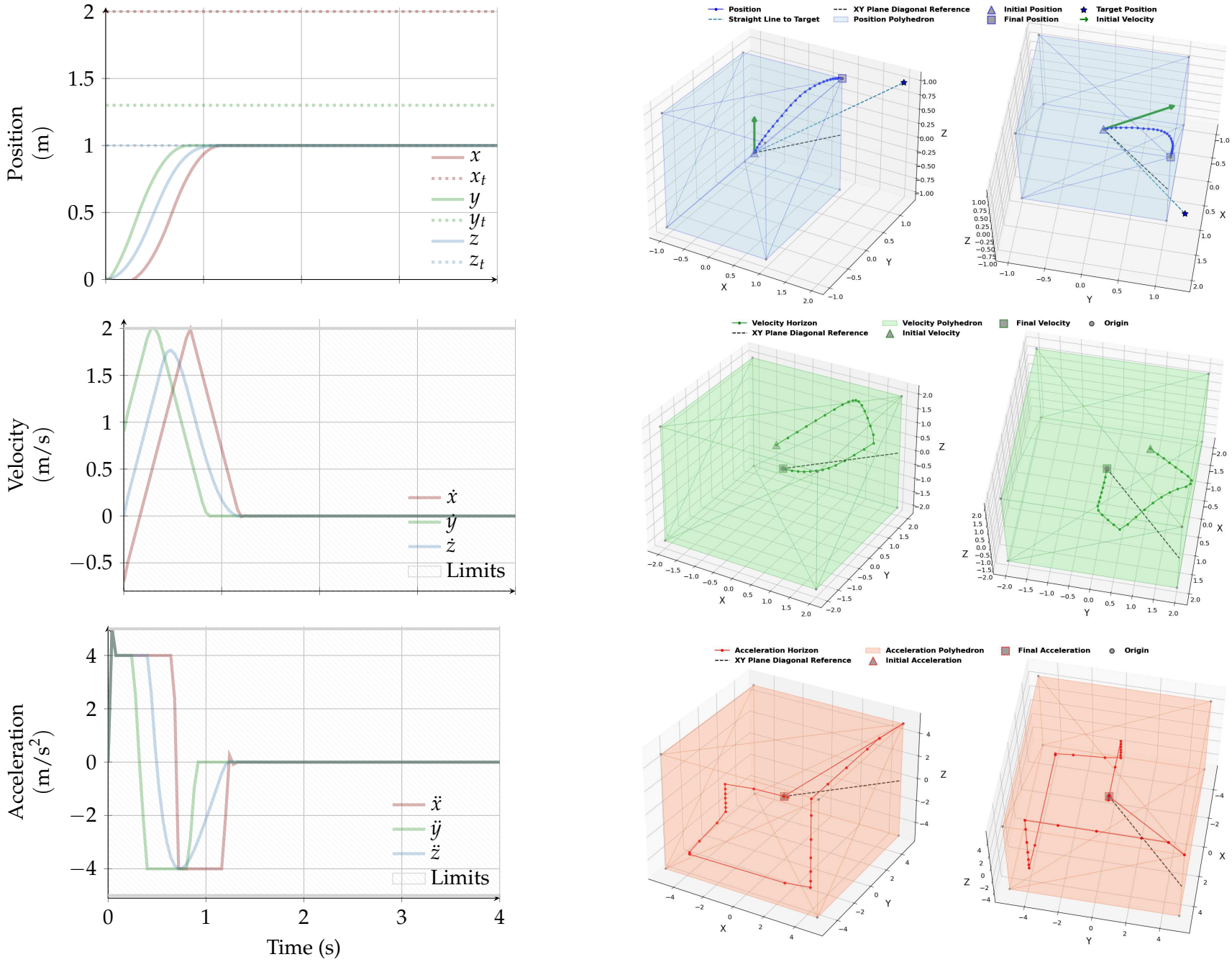


Figure 5.6: Example horizon with zonotope constraints and non-zero initial velocity: The simulation highlights the impact of the initial velocity on the trajectory. Due to limited kinematic capacities, the optimal trajectory takes on a curved path towards the target. This observation is particularly relevant in online re-planning scenarios, where the system must continuously compute new trajectories based on evolving initial conditions, demonstrating the capability for smooth transitions that consider the starting point in a loop.

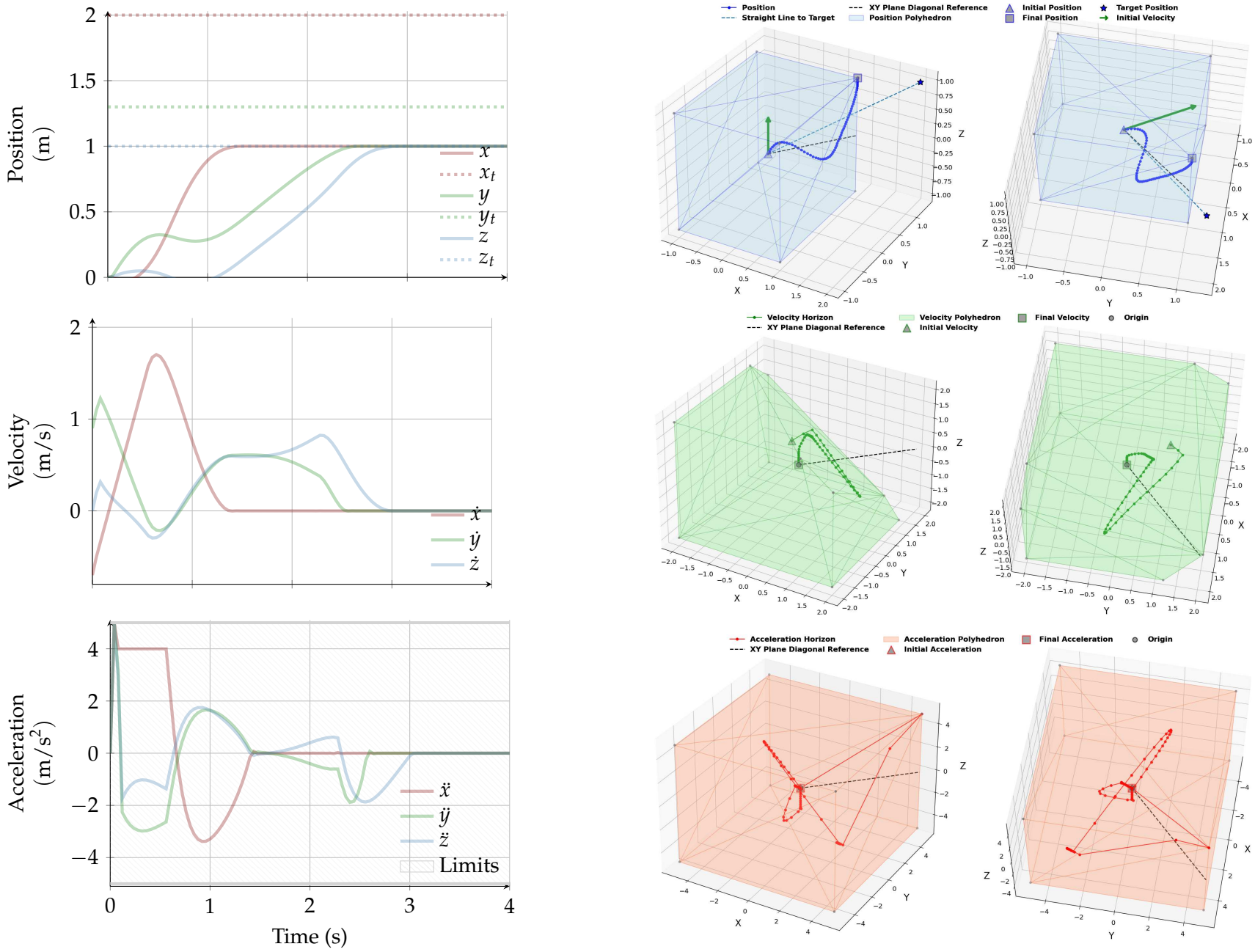


Figure 5.7: Example horizon with polyhedral constraints, non-zero initial velocity: This third simulation implements non-trivial polyhedral velocity constraints. This combines elements from the previous scenarios while mimicking the constraints derived from joint capabilities. The resulting trajectory exhibits a curved path that gradually transitions toward the target position. It is worth noting that the trajectory saturates the available velocity capacity by closely following the edges of the polyhedron, demonstrating how the system optimally utilizes constrained kinematic resources to approach the goal.

5.3.6 Acceleration-based Pose Re-planning ↑

Full pose motion re-planning (i.e. position and orientation control) requires some more work than pure positional control. The mathematical foundations that allow to perform optimizations for constrained motion generation with an MPC were first introduced in Sections 3.3.3 and 3.3.4. This section quickly revisits these definitions for pose control based on the spatial acceleration as introduced in Chapter 4. The re-introduction is necessary to construct the integration of actuation aware constraints that, combined to the computational advantages of the approach, aid in achieving re-planning at the control-rate.

A procedure similar to the previous section can be used to achieve full pose motion control. A first step consists in revisiting the expressions analogous to (5.38) and (5.39):

$$\mathcal{X}_{k+1} = \mathcal{X}_k e^{\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k} \tag{5.58}$$ revisited ↑

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + \mathrm{T}\dot{\boldsymbol{\nu}}_k \tag{5.59}$$ revisited ↑

Consider the first-order integration of a pose in $\mathbb{SE}(3)$ subject to an infinitesimal body twist increment for a single time step T shown in (5.58). This propagation shows the pose trajectories as the successive group actions (exponentials of the twist) on the Lie manifold. The objective is to include the pose and the twist into the state $\boldsymbol{x}$ of the system, and use the spatial acceleration $\dot{\boldsymbol{\nu}}$ as a decision variable or control input $\boldsymbol{u}$. This way, (5.58) and (5.59) can be expressed in the form of (5.28) to be used for linear MPC with (4.39).

By using the lift function introduced in (4.47), one can map $\mathcal{X}_k$ to a vector $\boldsymbol{\xi}_k$. The lift function maps elements from the $\mathbb{SE}(3)$ manifold $\mathcal{M}$ to its tangent space at $\mathcal{X}_0$: $\boldsymbol{\psi} : \mathcal{M} \to \mathcal{T}_{\mathcal{X}_0}\mathcal{M}$. The function $\boldsymbol{\psi}^{-1}$ is referred to as the retract function.

With the MPC being formulated in $\mathfrak{se}(3)$, the lift operator $\boldsymbol{\psi}$ is used to map the current state and the desired pose into $\mathfrak{se}(3)$ to perform the optimization. Meanwhile the retract operator $\boldsymbol{\psi}^{-1}$ is used to map the computed trajectory back to $\mathbb{SE}(3)$. This enables directly embedding the *lifted* pose $\boldsymbol{\xi} = \boldsymbol{\psi}(\mathcal{X})$ into the discretized state of the system $\boldsymbol{x}$, along with the body twist $\boldsymbol{\nu}$. Then, choosing the spatial acceleration $\dot{\boldsymbol{\nu}}$ as the input:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\nu}_k \end{bmatrix} \qquad \boldsymbol{u}_k = \dot{\boldsymbol{\nu}}_k \tag{5.60}$$ revisited ↑

where the reader is reminded that the k sub-index symbolizes the discretized step within the horizon. In other words, with this notation, $\boldsymbol{x}_k$ corresponds to the state at the step k in the horizon starting at time t_n, or equivalently, denoted as $\boldsymbol{x}_{k|n}$. This expression has a first-order approximation in its Lie algebra (see Equations 68-74 in [Solà et al., 2018]):

$$\boldsymbol{\xi}_{k+1} = \log(e^{\boldsymbol{\xi}_k} e^{\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k}) \approx \boldsymbol{\xi}_k + \mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k)(\mathrm{T}\boldsymbol{\nu}_k + \frac{\mathrm{T}^2}{2}\dot{\boldsymbol{\nu}}_k) \tag{5.61}$$ revisited ↑

where $\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k)$ is the right-trivialized Jacobian of the logarithmic map, evaluated at $\boldsymbol{\xi}_k$. A closed-form expression can be found in [Barfoot et al., 2014]; [Solà et al., 2018] and is implemented in the Pinocchio library [Carpentier et al., 2019]. It relates the additive increments in $\mathcal{T}_0\mathcal{M}$ (the tangent map at the origin) to the right-multiplied increments in $\mathbb{SE}(3)$.

Equation (4.52) offers the missing piece to have a concrete definition for (3.27) when considering the acceleration $\dot{\boldsymbol{v}}_k$ as the control input:

$$\boldsymbol{A}_k = \begin{bmatrix} \mathbf{I}_6 & \mathrm{T}\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k) \\ \mathbf{0}_6 & \mathbf{I}_6 \end{bmatrix} \qquad \boldsymbol{B}_k = \begin{bmatrix} \frac{\mathrm{T}^2}{2}\mathrm{J}_{\mathrm{R}}^{\log}(\boldsymbol{\xi}_k) \\ \mathrm{T}\mathbf{I}_6 \end{bmatrix} \tag{5.62}$$

revisited ↑

finally offering a concrete definition for the propagation of the state of the system as in (5.29).

Assuming the target state $\boldsymbol{x}_t$ contains the lifted target pose and a null (zero) body twist:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{\xi}_t \\ \mathbf{0} \end{bmatrix} \tag{5.63}$$

revisited ↑

5.3.7 Pose Re-planning Constraints ↑

Just as in Section 5.3.4, the objective of this section is to formulate the set of constraints that model workspace-related restrictions along with actuation capabilities.

Most of the constraints introduced for position control can be straightforwardly extended for the pose, except for the safety zone. The same principles used in Section 5.2.4 for the formulation of the convex reachable space for the pose in (5.26) can be employed for this case. The main caveat is that, because the MPC problem was formulated in the body frame, the reachable space also must be expressed in the same frame (i.e. a moving frame aligned with the body). In other words, instead of defining the allowed "workspace", it must be formulated as the allowed "space displacement".

The formulation can be constructed by analogy from a position displacement zonotope constraint:

$$\boldsymbol{p}_k + \Delta\boldsymbol{p} \leq \boldsymbol{p}_M \rightarrow \Delta\boldsymbol{p} \leq \Delta\boldsymbol{p}_M \qquad \Delta\boldsymbol{p}_M = \boldsymbol{p}_M - \boldsymbol{p}_k \tag{5.64}$$

that can also be formulated as a polylhedron:

$$\boldsymbol{C}_i\Delta\boldsymbol{p} \leq \boldsymbol{C}_i\Delta\boldsymbol{p}_M \tag{5.65}$$

where $\boldsymbol{C}_i$ denotes the first row of some linear inequality constraint matrix $\boldsymbol{C}$. For the zonotope case, it is implied that $\boldsymbol{C}_i = [1\ 1\ 1]^T$. In that case, the variable $\Delta\boldsymbol{p}$ and $\boldsymbol{C}_i$ are both in the same frame (implicitly, the local world-aligned frame). Assuming the constrained variable $\Delta\boldsymbol{p}$ is now in the body frame (i.e. rotated), the inequality needs to be brought into the same frame, by applying the current orientation $\boldsymbol{R}_k$:

$$\boldsymbol{R}_k^T\boldsymbol{C}_i\Delta\boldsymbol{p} \leq \boldsymbol{R}_k^T\boldsymbol{C}_i\Delta\boldsymbol{p}_M \tag{5.66}$$

The same procedure can be followed for the case of pose constraints with:

$$\mathbf{Ad}_{\mathcal{X}_k}{}^{-1}\boldsymbol{C}_i\boldsymbol{C}_l \leq \mathbf{Ad}_{\mathcal{X}_k}{}^{-1}\boldsymbol{C}_i\mathrm{dlog}_{\boldsymbol{\xi}_k}\boldsymbol{\xi} \leq \mathbf{Ad}_{\mathcal{X}_k}{}^{-1}\boldsymbol{C}_i\boldsymbol{C}_u \tag{5.67}$$

where $\boldsymbol{C}_l, \boldsymbol{C}_u$, that respectively denote the original lower and upper bounds, must be continuously updated to account for the movement of the end-effector, just like with $\Delta\boldsymbol{p}_M$. The matrix $\mathbf{Ad}_{\mathcal{X}_k}{}^{-1}$ corresponds to the adjoint map that allows changing

the frame of quantities in the tangent space.

And now, the safety-related constraints denoted with an $\mathcal{S}$ can be formulated:

$$\mathcal{P}_{\boldsymbol{\xi}|\mathcal{S}} = \{\boldsymbol{\xi} \in \mathfrak{se}(3) \mid \forall \boldsymbol{v} \in [\boldsymbol{\xi}_{m|\mathcal{S}}, \boldsymbol{\xi}_{M|\mathcal{S}}]\} \tag{5.68}$$

$$\mathcal{P}_{\boldsymbol{\nu}|\mathcal{S}} = \{\boldsymbol{\nu} \in \mathfrak{se}(3) \mid \forall \dot{\boldsymbol{p}} \in [\boldsymbol{\nu}_{m|\mathcal{S}}, \boldsymbol{\nu}_{M|\mathcal{S}}]\} \tag{5.69}$$

where now the bounds are expressed in the body frame. So, for instance, $\boldsymbol{\nu}_{M|\mathcal{S}}$ denotes the maximum body twist.

The case of the manufacturer-provided task-space limits is even more direct and are denoted with an $\mathcal{M}$ as:

$$\mathcal{P}_{\boldsymbol{\nu}|\mathcal{M}} = \{\boldsymbol{\nu} \in \mathfrak{se}(3) \mid \forall \boldsymbol{\nu} \in [\boldsymbol{\nu}_m, \boldsymbol{\nu}_M]\} \tag{5.70}$$

$$\mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{M}} = \{\dot{\boldsymbol{\nu}} \in \mathfrak{se}(3) \mid \forall \dot{\boldsymbol{\nu}} \in [\dot{\boldsymbol{\nu}}_m, \dot{\boldsymbol{\nu}}_M]\} \tag{5.71}$$

$$\mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{M}} = \{\ddot{\boldsymbol{\nu}} \in \mathfrak{se}(3) \mid \forall \ddot{\boldsymbol{\nu}} \in [\ddot{\boldsymbol{\nu}}_m, \ddot{\boldsymbol{\nu}}_M]\} \tag{5.72}$$

again with the same clarifications about the frame. Then, the set of *workspace* constraints can be formulated (denoted with $\mathcal{W}$):

$$\mathcal{P}_{\boldsymbol{\xi}|\mathcal{W}} = \mathcal{P}_{\boldsymbol{\xi}|\mathcal{S}} \cap \mathcal{P}_{\boldsymbol{\xi}|\mathcal{M}} \cap \mathcal{P}_{\boldsymbol{\xi}|\mathcal{A}} \tag{5.73}$$

$$\mathcal{P}_{\boldsymbol{\nu}|\mathcal{W}} = \mathcal{P}_{\boldsymbol{\nu}|\mathcal{S}} \cap \mathcal{P}_{\boldsymbol{\nu}|\mathcal{M}} \cap \mathcal{P}_{\boldsymbol{\nu}|\mathcal{A}} \tag{5.74}$$

$$\mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{W}} = \mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{S}} \cap \mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{M}} \cap \mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{A}} \tag{5.75}$$

$$\mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{W}} = \mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{S}} \cap \mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{M}} \cap \mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{A}} \tag{5.76}$$

where the sub-index $\mathcal{A}$ denotes arbitrary and application-dependent sets.

Finally, the focus of this section is on how to add actuation awareness onto the task-space re-planning, while also considering the workspace constraints. To complete the pose control MPC problem that contemplates all these restrictions, this definition is necessary:

$$\mathcal{X}_k = \mathrm{FK}(\boldsymbol{q}_k) \tag{5.77}$$

$$\boldsymbol{\nu}_k = \boldsymbol{J}_l \boldsymbol{q}_k \tag{5.78}$$

$$\boldsymbol{\xi} \in \mathcal{P}_{\boldsymbol{p}} \qquad \mathcal{P}_{\boldsymbol{\xi}} = \mathcal{P}_{\boldsymbol{\xi}|\mathcal{Q}}(\boldsymbol{q}_k) \cap \mathcal{P}_{\boldsymbol{\xi}|\mathcal{W}} \tag{5.79}$$

$$\boldsymbol{\nu} \in \mathcal{P}_{\boldsymbol{\nu}} \qquad \mathcal{P}_{\boldsymbol{\nu}} = \mathcal{P}_{\boldsymbol{\nu}|\dot{\mathcal{Q}}}(\boldsymbol{q}_k) \cap \mathcal{P}_{\boldsymbol{\nu}|\mathcal{W}} \tag{5.80}$$

$$\dot{\boldsymbol{\nu}} \in \mathcal{P}_{\dot{\boldsymbol{\nu}}} \qquad \mathcal{P}_{\dot{\boldsymbol{\nu}}} = \mathcal{P}_{\dot{\boldsymbol{\nu}}|\ddot{\mathcal{Q}}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \cap \mathcal{P}_{\dot{\boldsymbol{\nu}}|\mathcal{W}} \tag{5.81}$$

$$\ddot{\boldsymbol{\nu}} \in \mathcal{P}_{\ddot{\boldsymbol{\nu}}} \qquad \mathcal{P}_{\ddot{\boldsymbol{\nu}}} = \mathcal{P}_{\ddot{\boldsymbol{\nu}}|\dddot{\mathcal{Q}}}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) \cap \mathcal{P}_{\ddot{\boldsymbol{\nu}}|\mathcal{W}} \tag{5.82}$$

where $\mathcal{X}_k, \boldsymbol{\nu}_k$ respectively designate the initial pose and twist. These equations expresses the convex sets that need to be to be formulated as linear constraints as in (4.40) and (4.41).

Objective (Section 5.4.4)	Setup (Section 5.4.2)	Scenarios		
		Motionless Planning (Scenario 1)	Back & Forth (Scenario 2)	Interactive (Scenario 3)
Joint Bounds Modulation (Section 5.4.6)	Simulation	Figure 5.11		
	Real Robot		Figure 5.12	
Joint Velocity Modulation (Section 5.4.7)	Simulation		Without Re-orientation Figure 5.13	
	Simulation		With Re-orientation Figure 5.14	
	Real Robot		Without Re-orientation Figure 5.15	
	Real Robot			Figure 5.16

Table 5.1: Overview of the experimental validation, showing the experimental objectives, scenarios and setups used in each case.

5.4 Experimental Validation ↑

Section 5.3.5 offered an intuitive way to comprehend the effects of using an Model Predictive Control (MPC) for position planning while subject to polyhedral constraints. The objective was to provide an analysis in simple scenarios without actual re-planning. This section proposes a series of experimental settings that showcase the full reactive capabilities of the controller proposed in this manuscript. Also, unlike the previous experiments that focused on position planning (for the interpretability), this section provides a full pose control demonstration.

The main purpose is to provide an experimental validation of the claimed features. In this context, the focus of these experiments is to highlight the actuation-awareness functionality described in Section 5.1.2 and subsequently formulated in Sections 5.2 and 5.3. In particular, the scenarios provide the bases to analyze the effects of re-planning while considering: the joint velocities; and the joint limits.

Refer to Table 5.1 for an overview of the experimental validations performed.

All in all, the main conclusions of these experiences are:

- Trajectory planning from a purely task-space centric point of view while considering joint level actuation limits is achieved.
- It does so while performing a reactive /collaborative task (i.e. online).
- The trajectory also takes into account extra workspace constraints.
- It retains the features highlighted in previous chapters.

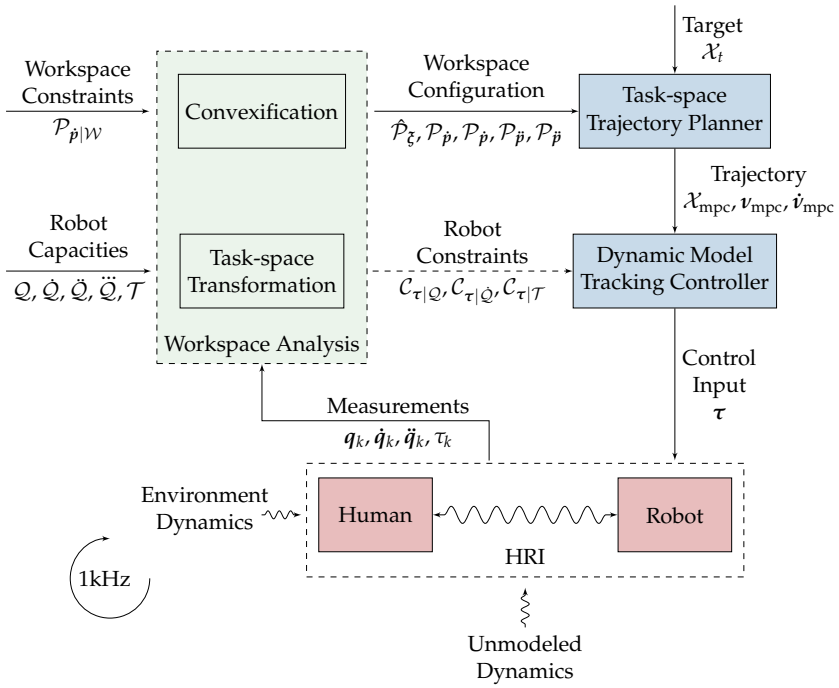


Figure 5.8: The implemented control architecture is based on the one proposed in Section 4.1.2. It is composed of a cascade loop: 1) a task-space Model Predictive Controller for online re-planning (described in Section 4.3); 2) a constrained Task Space Inverse Dynamics solver to follow the planned trajectory (described in Section 4.3). The feedback is closed with the current state of the robot. revisited ↑ ↓

5.4.1 Control Architecture Implementation ↑

The control architecture resembles the experimental setup in Chapter 4. However, the control architecture is extended to include the feasibility sets. The differences are highlighted in Figure 5.8. The main changes lie in the workspace analysis features, which now considers both the workspace constraints with the joint actuation limits. The geometric workspace available is not included though, as they are analyzed broadly in the position control MPC whose example simulations are shown in Section 5.3.5. Roughly speaking, this architecture is composed of an MPC block

Name	Symbol	Value
Horizon Steps	h	5
Step Duration	T	100ms
Horizon Duration	hT	0.5s
Terminal Weight	w_T	1, T
Terminal Weight Matrix	$\boldsymbol{P}_T$	$\begin{bmatrix} \mathbf{I}_{6\times6} & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & \mathbf{I}_{6\times6}\,\mathrm{T} \end{bmatrix}$
Regularization Weight	γ	10^{-2}T/2

Table 5.2: MPC parameters used in this implementation. In the case of w_T, the terminal weight is homogeneized to account for the dimensional difference in the state (i.e. positions and velocities) as defined in (5.60).

for online pose re-planning while a second inverse dynamic solver executes the resulting trajectories. This loop runs at a 1kHz in sync with the actual robot control

rate.

The parameters employed for the MPC configuration are show in Table 5.2. The block is configured for a 0.5s horizon duration to achieve under 1ms computation times while leaving some time for the inverse dynamics block resolution, respecting the robot control rate.

The actuation aware constraints allow the MPC to plan a task-space trajectory that actually accounts for the joint capacities and current state of the robot. During the experimental scenarios, these constraints are manipulated to highlight their effects on the resulting behavior. The Cartesian-space Constraints (CSC) and Joint-space Constraints (JSC) provided by the manufacturer and employed in this experiment are shown in Tables 5.3 and 5.4, respectively.

Translation		**Rotation**	
v_M	1.7 m/s	ω	2.5 rad/s
$\dot{v}_M$	13 rad/s^2	$\dot{\omega}$	25 rad/s^2
$\ddot{v}_M$	6500 rad/s^3	$\ddot{\omega}$	12500 rad/s^3

Table 5.3: Franka Emika Cartesian trajectory requirements, as specified on their site: https://frankaemika.github.io/docs/control_parameters.html.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	Unit
q_M	2.8973	1.7628	2.8973	-0.0698	2.8973	3.7525	2.8973	rad
q_m	-2.8973	-1.7628	-2.8973	-3.0718	-2.8973	-0.0175	-2.8973	rad
$\dot{q}_M$	2.1750	2.1750	2.1750	2.1750	2.6100	2.6100	2.6100	rad/s
$\ddot{q}_M$	15	7.5	10	12.5	15	20	20	rad/s^2
$\dddot{q}_M$	7500	3750	5000	6250	7500	10000	10000	rad/s^3
τ_M	87	87	87	87	12	12	12	Nm
$\dot{\tau}_M$	1000	1000	1000	1000	1000	1000	1000	Nm/s

Table 5.4: Franka Emika joint trajectory requirements, as specified on their site: https://frankaemika.github.io/docs/control_parameters.html.

5.4.2 Experimental Setups & Software Infrastructure ↑

This section delves into some details about the deployment aspects of the experiments. It addresses the experimental hardware were the experiments were executed and offers some specifics about the software design and choices that support it.

The experiences were conducted on a 7-Degree of Freedom (DoF) serial robot manipulator Franka Emika Panda controlled through its joint torques with Franka's open-source libraries[4].

For the experiments proposed in this chapter, Two experimental setups were used:

- **Simulation:** The simulation setup is based on the Gazebo [Koenig et al., 2004] physics engine (interfaced through Franka's libraries).

[4] https://frankaemika.github.io/docs/franka_ros.html

- **Real Robot:** The hardware experimental setup is the same one used in Figure 4.4. When used, the plastic pick in the scene is detected in real time via an infra-red Optitrack camera array.

Both setups are based on the ROS [Quigley et al., 2009] framework for the general communication and visualization. In order to compute the robot modeling aspects, the Pinocchio library [Carpentier et al., 2019] is employed.

On an implementation note, the formulated MPC grows in complexity quite fast as the horizon steps are increased (in order to cover a longer duration) because the number of optimization variables is increased too. Being able to consider longer horizons while remaining computationally efficient is crucial need to achieve high frequency re-planning (i.e. at the control rate).

This chapter introduced a series of polyhedral inequalities that further scale the computation requirements as they need to be applied to every step of the horizon. This puts a heavy responsibility on finding an appropriate Quadratic Program (QP) solver that can find solutions at the control rate. One feature that helps with this problem is the possibility to limit resolution times. This implies that the QP solver can at least provide an approximate solution at the control rate. Given a high enough control rate, this solution is progressively refined and limits the undesired effects of non-optimal solutions in the overall behavior, without breaking the real-time requirements of the controller. Multiple options were explored including:

- ProxQP [Bambade et al., 2022]: A relatively recent library optimized with performance in mind. In these tests, the computation time was not up to the level of other alternative. However, more importantly, at the time of writing, it does not feature a computation time limit, necessary for the real time requirements of the controller.

- OSQP [Stellato et al., 2020]: A sparse solver implementation. Due to the sparsity of the problem, it should offer some optimization advantages. In practice though, the convergence seemed more erratic, although it does feature a computation time limitation.

- qpMAD[5]: This library implements the Goldfarb-Idnani dual active set algorithm for QP [Goldfarb et al., 1983] with some added optimizations to improve computation times. It often yielded the fastest resolution times. However, in some rare cases, it seemed to have a hard time dealing with too many linear inequalities (a consequence of polyhedral constraints in a horizon), leading to computation times over the control rate period. It also does not feature a computation time limit and so it was discarded in favor of the last one.

- qpOASES [Ferreau et al., 2014]: This library offered the most robust behavior as well as the required capability to limit computation times. It performed consistently and either equally or on pair with qpMAD.

In the end, qpOASES was retained for the MPC and qpMAD for the inverse dynamics solver.

Though it remains an experimental implementation (not intended for production

[5] https://github.com/asherikov/qpmad

setups), more details may be found in the open-source implementation library[6]. A workspace is also readily available for experimentation purposes[7].

5.4.3 Experimental Scenarios ↑

The experiments validate the task-space planning while taking into account the joint constraints. Concretely, this entails obtaining full pose Cartesian trajectories while retaining awareness of the joint level constraints. An overview of the experiences is presented in Table 5.1.

The objective is to validate both constraints independently. Because the architecture is tailored towards online re-planning, two types of reactive scenarios are proposed:

1. **Motionless Planning**: In this scenario, the robot is static. The only purpose is to show the effects of the joint bound constraints in the planned horizon.

2. **Back and forth scenario**: In this scenario, the robot is instructed to alternate between two predefined poses. However, these are provided as target poses on the fly, so the robot is not able to anticipate a change in target. Yet, this shows the transition capabilities during instant target changes of the architecture while providing a rather repeatable scenario that can be analyzed in more details in the resulting curves.

3. **Interactive scenario**: This scenario entails tracking a pose in real-time. The robot is instructed to follow a pose in the workspace sensed in real-time, this highlights the reactive capabilities of the architecture. The objective is to showcase the properties highlighted individually in other scenarios are maintained while in more random situations, representative to what is expected of a collaborative scenario.

For the case of the real hardware setup, the scenario 2 is shown in Figure 5.9 for the real setup. In the example shown, the back and forth poses entail only a translational displacement (i.e. they maintain the same orientation). The same situation is shown during the simulation experiments, though in one example a change in orientation is also added.

The interactive scenario 3 is shown in Figure 5.10. In this case, a plastic piece is presented in the robot workspace. Its pose is tracked using the Optitrack camera array and robot is instructed to follow it at a distance projected orthogonally over the Z axis. Although this exact scenario is not shown in simulation, a similar one is in fact devised to intuitively showcase the joint bound awareness while tracking an interactive pose in the simulation.

5.4.4 Experimental Design & Objectives ↑

The previous section detailed the scenarios employed. This section delves into the hypothesis to be tested in those scenarios.

As stated before, the objective is to showcase the effect of task-pace planning while accounting for joint level constraints. To maintain a certain level of simplicity and

[6] https://gitlab.inria.fr/auctus-team/people/nicolas-torres/lmpcpoly
[7] https://gitlab.inria.fr/auctus-team/people/nicolas-torres/lmpcpoly-ws

video/link

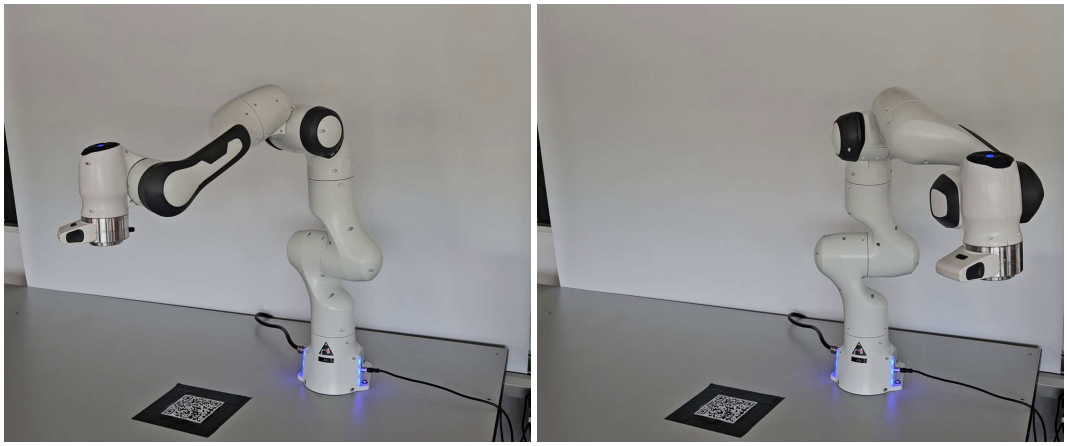


Figure 5.9: An example of the back and forth scenario (as defined in Section 5.4.3) used for the experiments, showing the starting and ending poses on the real setup. This scenario is also used in simualtions but the video shows one of the experiments being executed on the robot.

video/link

Figure 5.10: Showcases the interactive scenario (as defined in Section 5.4.3) used for the experiments, showing the robot tracking the pose of a plastic piece in real time. The video shows the experiment.

intuitiveness in the experiences and results, only the joint bounds and velocities are taken into account for these objectives, leaving the JSC for the acceleration and jerk joint activated but not shown in the results. Concretely, this entails obtaining full pose Cartesian trajectories while considering joint level constraints $\mathcal{Q}$, $\dot{\mathcal{Q}}$ introduced in (5.3).

In order to test these constraints independently, two types of experiments were devised:

1. **Joint bounds modulation experiments**: This type of experiments looks to highlight the effects of considering the convex reachable space approximation $\hat{\mathcal{P}}_{\xi|\mathcal{Q}}$ introduced in Section 5.2.4. This means that the robot should not plan to go towards unreachable spaces.

2. **Joint velocity modulation experiments**: In this case, the experiment looks to showcase the effects of $\mathcal{P}_{\nu|\dot{\mathcal{Q}}}$ introduced in Section 5.2.3.

Displaying Type 1 on the actual robot is somewhat intricate as it necessitates positioning it in a configuration close to its limits. This results in rather unpredictable behaviors, a result of the robot own internal mechanism to avoid damage interacting with the controller. To avoid this situation, the joint bounds were artificially reduced to show case the robot acting as if its joint range was smaller.

Conversely, Type 2 presents its own set of challenges to showcase. For instance, the expected result is that the planned trajectories should respect the allowed joint velocities. However, the robot redundant motion capabilities means that it can perform a trajectory while also re-configuring. In other words, the robot may perform a joint trajectory that follows the desired Cartesian motion while also generating a change in its configuration that does not result in an end-effector motion. This is mathematically understood as a movement in the kernel of the Jacobian. To bypass this problem, only the joint velocity motion that do not act on this kernel need to be accounted for (as those are responsible for the actual end-effector motion). This is explained in more details while addressing the results.

5.4.5 About the Measured Result Quantities ↑

This section summarizes the quantities employed in the result curves, offering their mathematical formulation, in the cases they are included:

- **Position**: Refers to the actual position of the end-effector in the workspace $\boldsymbol{p} = [x\ y\ z]^T$, expressed in the world frame. It is shown decomposed for its three components. Meanwhile, the target pose may also be shown with a dotted curve and with an added sub-index t in the legends: $\boldsymbol{p}_t = [x_t, y_t, z_t]^T$.

- **Position Error**: Refers to the error between the end-effector position and the target position (the translation component of the target pose $\mathcal{X}_t$ shown in Figure 5.8). Not to be confused with the "desired error" which would correspond to the position error between the one *planned* by the MPC (the position component of $\mathcal{X}_{\text{mpc}}$) and the current position of the end-effector (the current pose is denoted $\mathcal{X}$). This is wrongfully called desired because it represents the servoed trajectory that actually accounts for the planning. The way to compute this

error is:

$$\boldsymbol{e} = \boldsymbol{p}_t - \boldsymbol{p} \qquad \mathcal{X}_t = (\boldsymbol{R}_t, \boldsymbol{p}_t) \qquad \mathcal{X} = (\boldsymbol{R}, \boldsymbol{p}) \tag{5.83}$$

- **Orientation Error**: Similar to the position error shown above, this corresponds to the orientation error between the end-effector and the target orientation (the orientation component of the $\mathcal{X}_t$ shown in Figure 5.8). Not to be confused with the "desired error" which would correspond to the orientation error between one *planned* by the MPC (the orientation component of $\mathcal{X}_{\text{mpc}}$) and the current orientation of the end-effector. This is wrongfully called desired because it represents the servo-ed trajectory that actually accounts for the planning.

$$\boldsymbol{\phi} = \log(\boldsymbol{R}_t^{-1}\boldsymbol{R})) \qquad \mathcal{X}_t = (\boldsymbol{R}_t, \boldsymbol{p}_t) \qquad \mathcal{X} = (\boldsymbol{R}, \boldsymbol{p}) \tag{5.84}$$

- **Joint Configuration**: This shows the raw values of the joint positional configurations for each of the joints contained in the configuration vector $\boldsymbol{q}$.

- **Normalized Joint Bound Distance**: This shows the joint configuration as normalized against the joint bound (the maximum) specified by the manufacturer (see Table 5.4). It indicates how far or close the configuration is to the respective joint bound (for a value of 1, the configuration is near its limit). Note that given the asymmetrical (as in $\boldsymbol{q}_m \neq -\boldsymbol{q}_M$) joint bounds of joints 4 and 6 start at zero, the definition for each joint i as follows still yields a comparable quantity:

$$\hat{q}_i = |q_i| / q'_{M,i} \qquad q'_{M,i} = \max(|q_{m,i}|, |q_{M,i}|) \tag{5.85}$$

- **Joint Velocities**: This shows the raw values for the joint velocities as measured by the robot. They are computed with a vector $\dot{\boldsymbol{q}}$.

- **Normalized Joint Velocities**: This shows the joint velocities as normalized against the maximum joint velocities specified by the manufacturer (see Table 5.4). For each joint i, they are computed as:

$$\hat{\dot{q}}_i = |\dot{q}_i| / \dot{q}_{M,i} \tag{5.86}$$

- **Normalized Non-kernel Joint Velocities**: Similar to the previous normalized definition, this shows the joint velocities as normalized against the maximum joint velocities specified by the manufacturer (see Table 5.4). However, in this case, only the components not corresponding to the movements in the kernel of the Jacobian are accounted for. The joint velocities component projected in the kernel are denoted as $\dot{\boldsymbol{q}}_{\text{ker}}$ and the normalized joint velocities, once they are removed, are denoted with $\dot{\boldsymbol{q}}'$. For each joint i, they are computed as:

$$\dot{\boldsymbol{q}}_{\text{ker}} = (\mathbf{I}_n - \boldsymbol{J}^+\boldsymbol{J})\dot{\boldsymbol{q}} \qquad \boldsymbol{J}^+ = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1} \tag{5.87}$$

$$\hat{\dot{q}}'_i = |(\dot{q}_i - \dot{q}_{\text{ker},i})| / \dot{q}_{M,i} \tag{5.88}$$

where $\boldsymbol{J}^+$ corresponds to the right pseudo inverse (or right Moore-Penrose inverse) of the Jacobian.

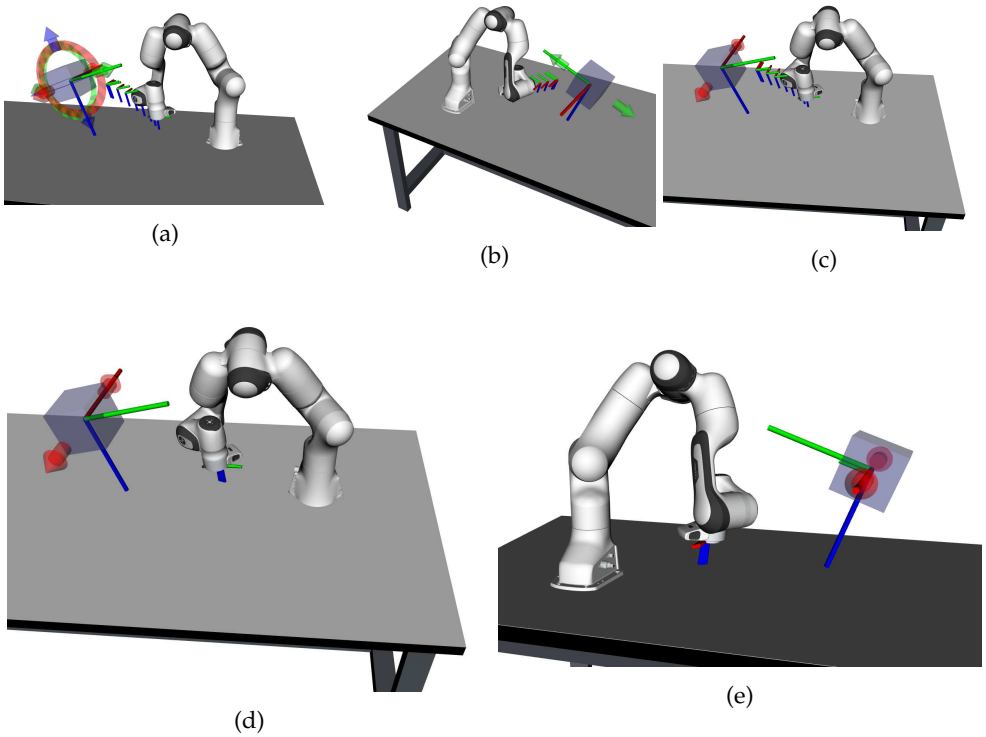


Figure 5.11: This image showcases the results of a joint-bounds aware task-space planning addressed in Section 5.4.6. The experiment corresponds to a simulation of a motionless planning scenario (as defined in Section 5.4.3) in which the robot is driven to a configuration near its joint bounds so that it cannot move towards the target cube. When the joint bounds limits are disabled (in (a), (b) and (c)), the robot plans a horizon path through the geodesic towards the target. Conversely, once the joint bounds limits are employed in (d) and (e), the horizon becomes very small and tries to find a path that minimizes the distance to the target without violating the joint bounds constraints.

5.4.6 Results: Joint Bounds Aware Planning ↑

A first approach to this type of experiment is performed in a simulation scenario. The robot is driven to a configuration near its joint bounds, a situation unsafe for execution in the real robot. The expectation is that the planned trajectory should not try to drive the robot towards unreachable poses.

Such setting is shown in Figure 5.11. The robot is instructed to follow the pose of a gray cube floating in space. The images show the robot in a configuration near its limits. The horizon trajectory is also showed. However, because horizon tries to drive the robot in an impossible direction, the robot is stuck at a local minima. The horizon corresponds to the geodesic path towards the target pose, as shown in Figures 5.11(a) to 5.11(c). Once the convex reachable space constraints are activated, the horizon no longer tries to take this path. In fact, it stays near the end-effector, as it corresponds to the closest possible configuration achievable from this configuration, as shown in Figures 5.11(d) and 5.11(e). A consequence of using a linearized approximation of the reachable space is that these constraints is that the MPC, while still able to consider them, is also prone to local minima (as the target poses are actually reachable by the robot in alternative configurations).

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	Unit
$\boldsymbol{q}'_M$	1.7384	1.0577	1.7384	-0.0698	1.7384	2.2515	1.7384	rad
$\boldsymbol{q}'_m$	-1.7384	-1.0577	-1.7384	-1.8431	-1.7384	-0.0175	-1.7384	rad

Table 5.5: Artifitially limiting the joint bounds to 60% of the range for experimental valudation yields the limits shown here. The full range are shown in Table 5.4.

Experimental Validation on the Robot

To showcase the effects in a dynamic setting, a "back and forth" experiment (scenario 2) was executed on the robot. In order to operate the robot in undesirable conditions (i.e. near its joint bounds), the joint bounds constraints were set to an artificially constrained value (smaller than the real values). Instead of using $\hat{\mathcal{P}}_{\xi|\mathcal{Q}}$ as indicated in Figure 5.8, a new feasible reachable space was constructed with:

$$\begin{gathered}\hat{\mathcal{P}}_{\xi|\mathcal{Q}'}(\boldsymbol{q}_k) = \{\boldsymbol{\xi} \in \mathfrak{se}(3) \mid \boldsymbol{\xi}(\Delta\boldsymbol{q}),\ \Delta\boldsymbol{q} \in [\boldsymbol{q}'_m - \boldsymbol{q}_k, \boldsymbol{q}'_M - \boldsymbol{q}_k]\} \\ \boldsymbol{\xi}(\Delta\boldsymbol{q}) \sim \boldsymbol{\xi}_k + \mathrm{dlog}_{\boldsymbol{\xi}_k} \boldsymbol{J}\Delta\boldsymbol{q} \qquad \boldsymbol{\xi}_k = \log(\mathcal{X}_k)\end{gathered} \tag{5.89}$$

revisited ↑

$$\mathcal{Q}' = \{\boldsymbol{q}(t) \in \mathbb{R}^n \mid \boldsymbol{q} \in [\boldsymbol{q}'_m, \boldsymbol{q}'_M]\}$$

revisited ↑

where the artificial joint bounds are provided by $\boldsymbol{q}'_m, \boldsymbol{q}'_M$ and computed using a factor $R_q \in [0, 1]$. Each element of these bounds (for each joint) is defined as:

$$\begin{aligned}
q'_{m,1} &= R_q q_{m,1} & q'_{M,1} &= R_q q_{M,1} \\
q'_{m,2} &= R_q q_{m,2} & q'_{M,2} &= R_q q_{M,2} \\
q'_{m,3} &= R_q q_{m,3} & q'_{M,3} &= R_q q_{M,3} \\
q'_{m,4} &= R_q q_{m,4} & q'_{M,4} &= q_{M,4} \\
q'_{m,5} &= R_q q_{m,5} & q'_{M,5} &= R_q q_{M,5} \\
q'_{m,6} &= q_{m,6} & q'_{M,6} &= R_q q_{M,6} \\
q'_{m,7} &= R_q q_{m,7} & q'_{M,7} &= R_q q_{M,7}
\end{aligned} \tag{5.90}$$

This helps to explicitly show the range of the joint angles with respect to their bounds, by using the normalized joint bounds defined in (5.85). The joints with asymmetrical bounds (joints 4 and 6) are the ones that render this special definition required but because both have one of their bounds approximately near zero, the "normalizations" are still comparable between all of them.

For the results of this experimental validation, refer to Figure 5.12. In this experiment, the robot starts at a neutral pose, which corresponds to the configuration at the middle between the bounds shown in (4.13) to ensure $\boldsymbol{q}$ belongs to $\mathcal{Q}'$. Then, the robot is instructed to alternate between two poses until the end of the experience, as shown in Figure 5.9, where a link to the corresponding video is also included. The first peak value in the position and orientation error (approximately before 1s) corresponds to the moment the back and forth task begins (switching from the starting neutral pose to the first pose in the cycle).

The experience shows two stages: a first stage where $R_q = 0.6$ (which yields the joint bound limits shown in Table 5.5) with the resulting constrained movement effects; a

second stage were the full range (meaning $R_q = 1$) is enabled to highlight the difference in behavior. The split between both moments is marked with a vertical black dashed line and it can be seen that it also coincides with the increase in the value of R_q, shown in the same curves as the normalized joint bounds distance.

The first stage shows the results of the planned trajectory when the bounds are reduced. It can be seen that the robot is not able to reach the target pose and thus the position and orientation errors never really converge to zero. In contrast, in the second stage, when the full joint range is enabled, the robot is able to minimize both errors.

Furthermore, the joint configuration trajectory changes at the moment of the switch in the R_q value. It can be appreciated in the joints 4 and 6, in the joint configuration graph, where they suddenly surpass artificially imposed limits. This also leads to reducing the position and orientation target error towards zero (until the next change in target, from the back and forth scenario).

5.4.7 Results: Joint Velocity Modulation Task-space Planning ↑

These experiments entail a combination of back and forth and interactive scenarios. In order to showcase the velocity modulation capabilities of the feasible velocities constraint using $\hat{\mathcal{P}}_{\boldsymbol{\xi}|\dot{\mathcal{Q}}}$ as indicated in Figure 5.8, a new feasible set with modulated capacities was defined:

$$\mathcal{P}_{\boldsymbol{\nu}|\dot{\mathcal{Q}}'}(\boldsymbol{q}) = \left\{\boldsymbol{\nu} \in \mathfrak{se}(3) \mid \boldsymbol{\nu} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}},\ \dot{\boldsymbol{q}} \in \dot{\mathcal{Q}}'\right\} \tag{5.91}$$ revisited ↑

$$\dot{\mathcal{Q}}' = \{\dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \dot{\boldsymbol{q}} \in [R_{\dot{q}}\dot{\boldsymbol{q}}_m, R_{\dot{q}}\dot{\boldsymbol{q}}_M]\} \tag{5.92}$$ revisited ↑

$$R_{\dot{q}} \in [0, 1]$$

where $R_{\dot{q}}$ designates a factor between 0 and 1 that reduces the maximum joint velocity allowed.

It is important to note that while a new $\dot{\mathcal{Q}}'$ is employed for the MPC re-planning, the original $\dot{\mathcal{Q}}$ is still used in the inverse dynamics controller. It would be straightforward to replace one for the other and could certainly be considered an improvement to the overall guarantees of respecting the actual constraints. However, the objective of the experience is to showcase the effects of modulating the task-space trajectory without direct access to the joint constraints. This way, the only thing that can impact the resulting trajectory is in fact the planned horizon. This helps provide a stronger argument for the capabilities of the actuation aware MPC.

Because of the robot redundancy, part of the joint motion is performed without net end-effector movement (i.e. into the Jacobian kernel). This is why two joint velocities are shown in these experiments: the raw joint velocity; a normalized joint velocity corresponding to the components not projecting to the kernel, as defined in (5.87).

To showcase the effects in a dynamic setting while modulating $R_{\dot{q}}$, four experiments were performed:

1. **Simulation without reorientation**: A back and forth between poses without changing the targets orientation is shown in Figure 5.13.

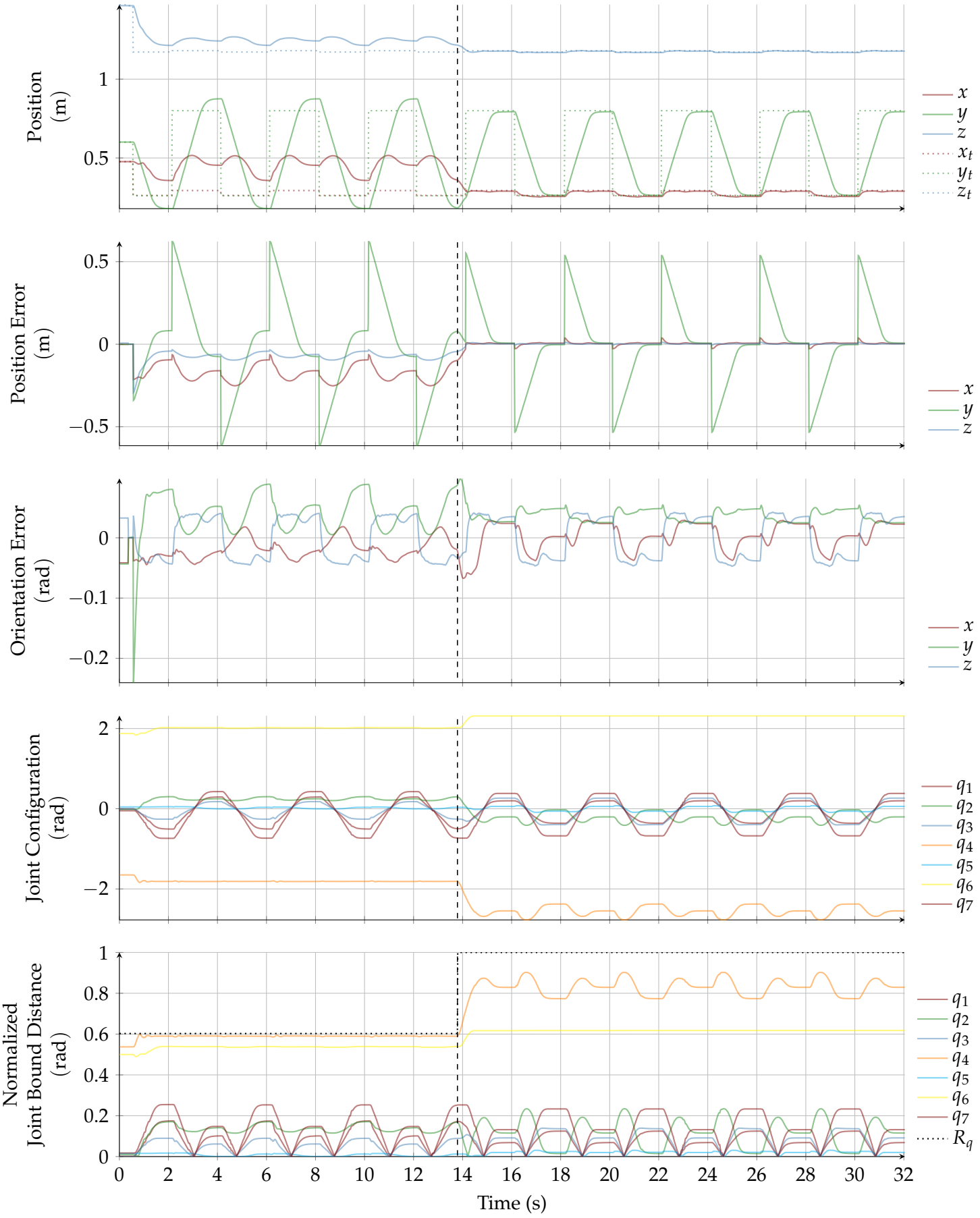


Figure 5.12: Real-world Experiment: Showcases a back and forth task scenario between two poses that have the same orientation. The objective is to show the robot behavior while the allowed joint range is artificially reduced through a factor R_q beween 0 and 1 (also shown in the plot) to demonstrate the MPC capability to achieve a task-space trajectory that takes into account joint level constraints. The passage from $R_q = 0.6$ to $R_q = 1$ is highlighted with a dashed vertical line.

2. **Simulation with reorientation**: A back and forth between poses with change in the target orientations is shown in Figure 5.14.

3. **Real Robot without reorientation**: A back and forth between poses with change in the target orientations is shown in Figure 5.15 and executed on the real hardware.

4. **Real Robot interactive**: An interactive setting (scenario 3) representative of collaborative task, where the robot is instructed to follow the pose of a plastic piece in real-time is shown in Figure 5.10 (with its corresponding results shown in Figure 5.16).

In all these settings the robot begins at a neutral pose and then is instructed to begin. In all experiments, while the robot is executing the task, the value of $R_{\dot{q}}$ is changed to modulate the available joint velocities. The effects are highlighted in the "non-kernel" normalized joint velocities. The values of $R_{\dot{q}}$ are also plotted in these graphs, allowing for a direct comparison (thanks to the normalized velocities).

It can be seen that in some situations, there are peaks that surpass the allowed joint velocities, even though these correspond to the "non-kernel" components. This is in part a consequence of not enforcing the same joint level constraints in the inverse dynamics controller; and in part is due to the high velocity movements requiring more servoing from the Proportional Derivative (PD) controller, which is defined in (4.9).

The general results help make an argument for the consideration of JSC into the MPC, as the joint velocities roughly follow the same pattern as the modulation set with $R_{\dot{q}}$.

Furthermore, the interactive scenario with the real robot showcases the capability of the approach to adapt to completely unknown situation in an online fashion, a crucial requisite for collaborative tasks.

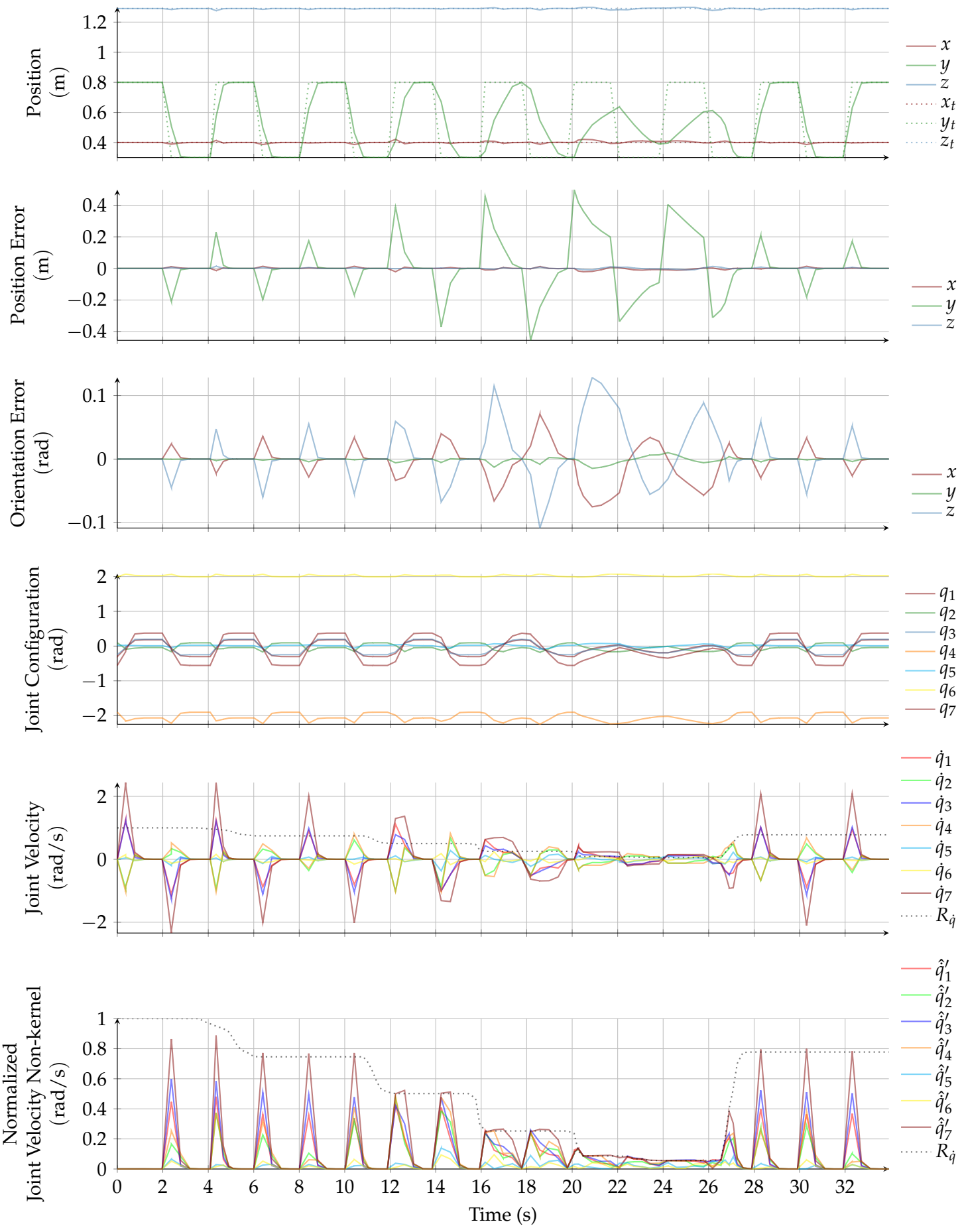


Figure 5.13: Simulation: Showcases a back and forth task scenario between two poses without changing the orientation, while changing the maximum allowed $\dot{\boldsymbol{q}}$ (showed with $R_{\dot{q}}$) to demonstrate the MPC capability to achieve a task-space trajectory that takes into account joint level constraints.

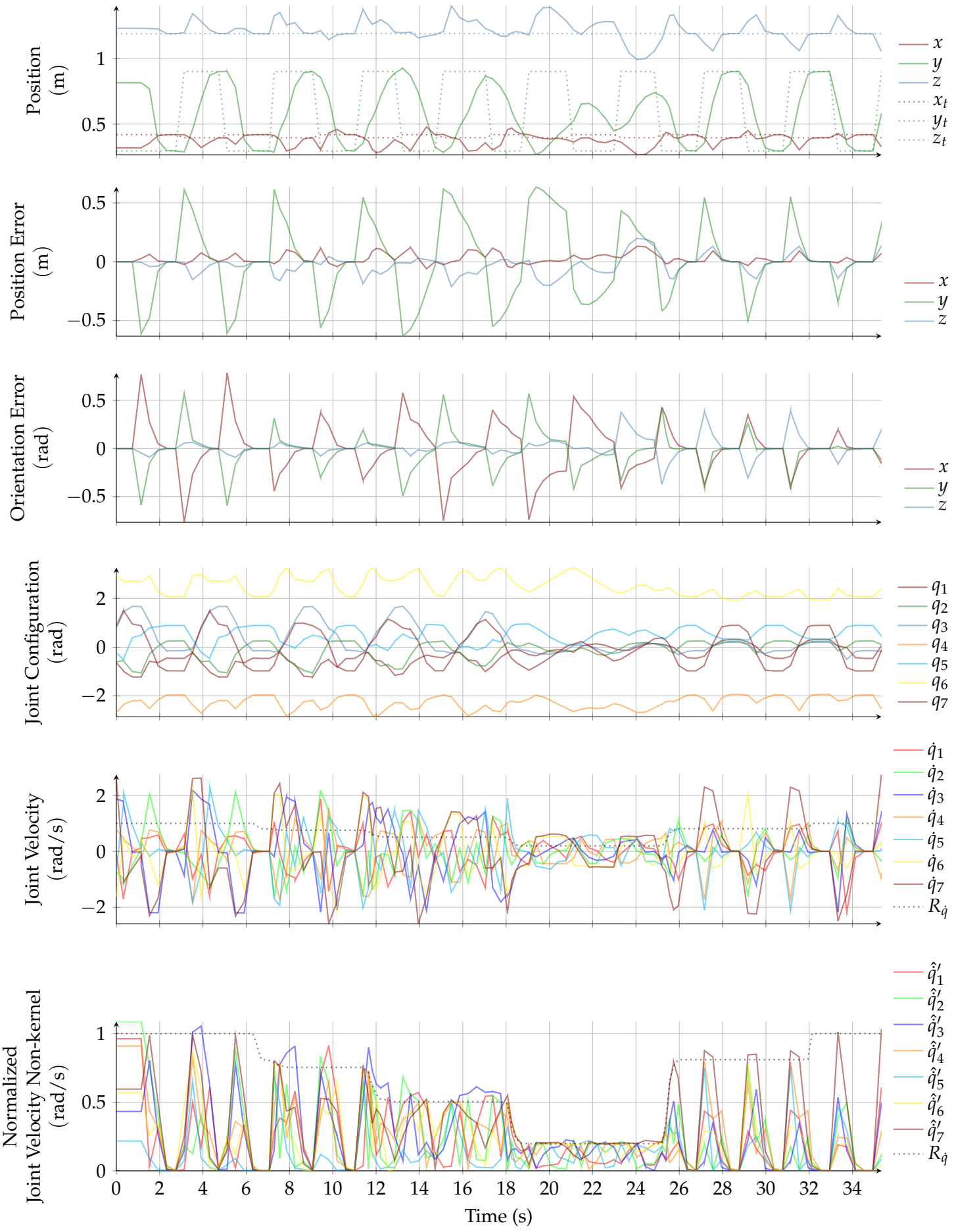


Figure 5.14: Simulation: Showcases a back and forth task scenario between two poses that have different orientations, while changing the maximum allowed $\dot{\boldsymbol{q}}$ (showed with $R_{\dot{q}}$) to demonstrate the MPC capability to achieve a task-space trajectory that takes into account joint level constraints.

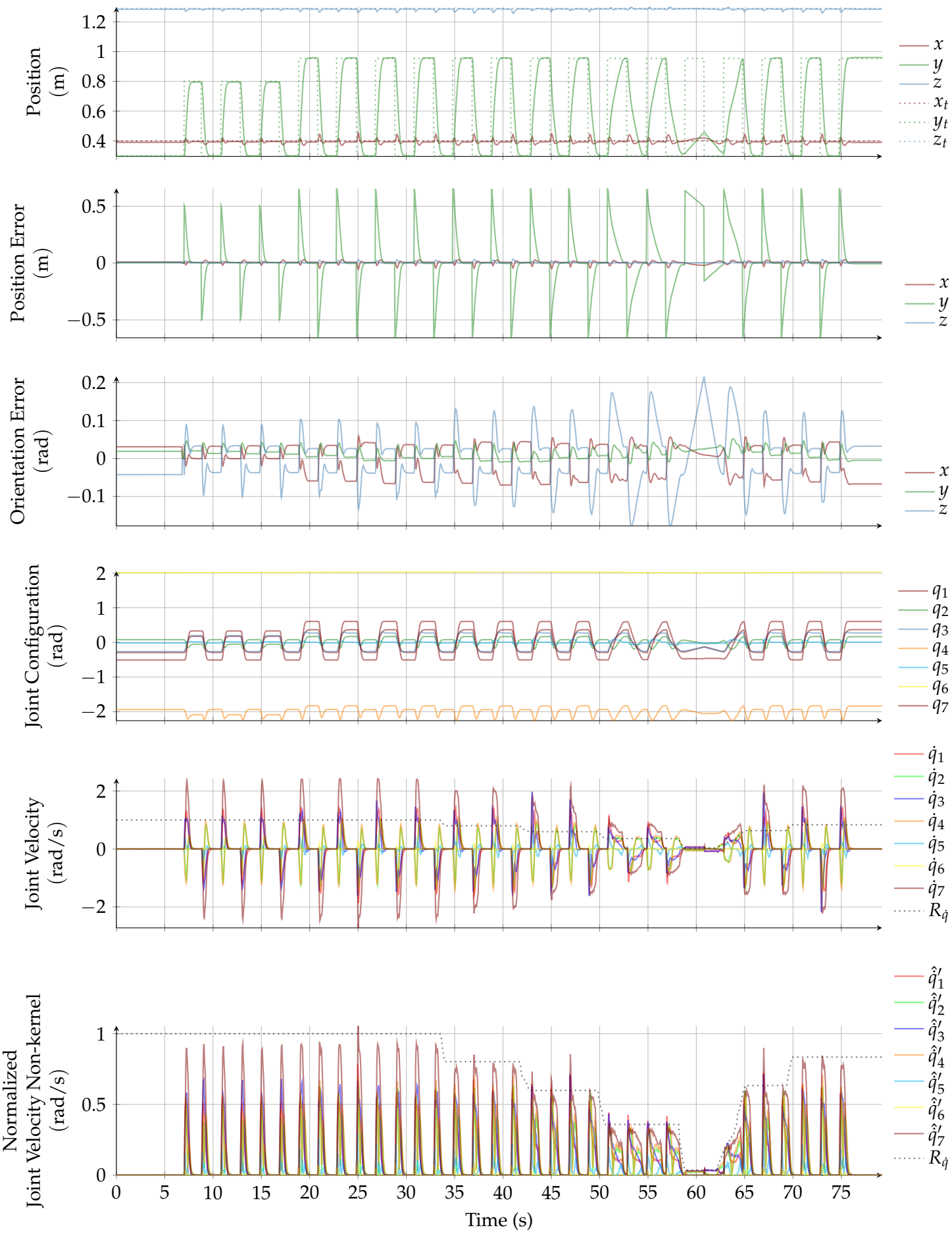


Figure 5.15: Real-world experiment: Showcases a back and forth task scenario between two poses that have the same orientation, while changing the maximum allowed $\dot{\boldsymbol{q}}$ (showed with $R_{\dot{q}}$) to demonstrate the MPC capability to achieve a task-space trajectory that takes into account joint level constraints.

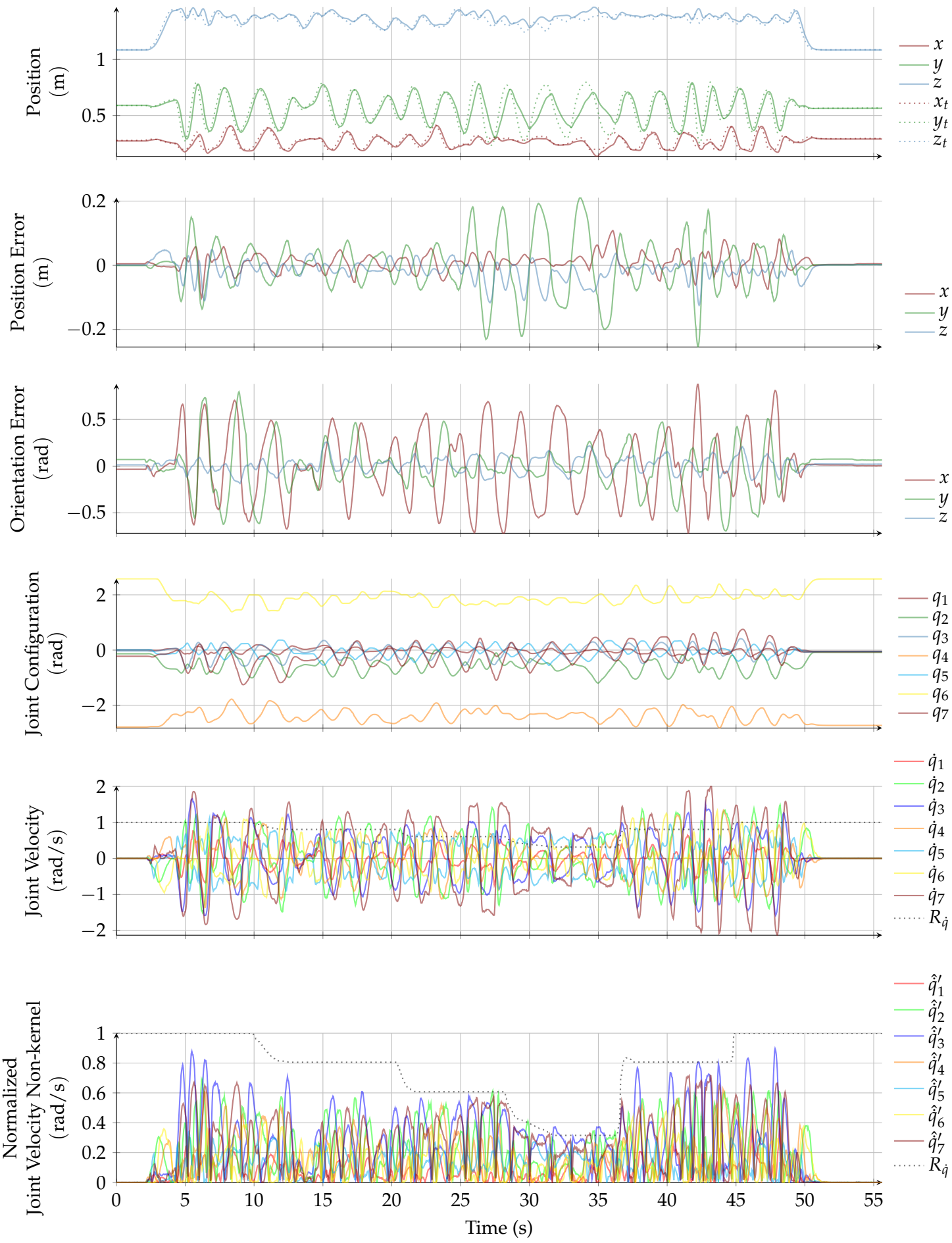


Figure 5.16: Real-world experiment: Showcases an interactive task scenario where the robot tracks a pose captured in real-time while changing the maximum allowed $\dot{\boldsymbol{q}}$ (showed with $R_{\dot{q}}$) to demonstrate the MPC capability to achieve a task-space trajectory that takes into account joint level constraints in an online fashion. The corresponding video is shown in Figure 5.10.

5.4.8 Discussion ↑

While the results helped strongly showcase the task-space centric re-planning for joint bounds and velocity modulation, the same cannot be said of the JSC for the acceleration and jerks. For one, the measurements are noisy enough as is and would make it difficult to obtain reliable results. Furthermore, this problem would also translate to a worse estimation of the feasible motion set (for the spatial accelerations and jerk), further degrading the capabilities of the MPC to achieve the desired modulation. And finally, a bad estimation of biases in these polyhedral constraints would also contribute to incompatible constraints in the MPC, provoking feasibility issues.

An important issue when dealing with constraints is finding a way to always produce trajectories that consistently stay within limits. This issue is exacerbated by two situations: for one, the trajectory constraints are originated from different sources (i.e. Cartesian vs joint-level) but they still need to be expressed in the same space to be considered at the same time; secondly, the nature of collaborative tasks further complexifies planning because of the limited computation time and not predefined starting conditions. While the present control architecture helps address these issues by using horizon optimization, it cannot really provide theoretical guarantees to the viability of the solution (viability as in not going to lead to undesirable situations, as defined in [Prete, 2018]). With respect to the viability of JSC, a proposal is made in Section 5.5.

Another a big problem in all optimization approaches with hard constraints is dealing with incompatible problems / constraints. For instance, this means that the if the problem is unsolvable, the robot will enter an inconsistent state as no control input can be defined. The present approach has two mechanisms that help avoid these kind of situations: first, the weight on the terminal state helps ensure the end of the horizon tends towards a safe state (in this case, motionless); secondly, if ever the MPC cannot solve the optimization, the controller will continue to use the last horizon successfully calculated.

Finally, these results highlight the potential of a modular architecture separating tasks-space and joint-space aspects: for one it is generic enough for the application to other robot architectures; secondly, it separates the motion problem formulation while still allowing some aspects of the robot model to be considered at the task level. One potential field that could benefit of this approach is humanoid robot control, where the trajectory planning is often made at the task-space level [Lober et al., 2020] and considering the actual actuators capabilities is crucial for stable motions like walking.

5.5 Towards Viable Joint Constraints ↑

As it was briefly mentioned in this chapter, an unresolved issue regarding constraints is on the future states compatibility. As referenced in [Rubrecht et al., 2012], a set of constraints may be instantly feasible but incompatible in the short-medium term, as they might lead to inevitable constraint violations. The assumption of bounded jerk imposes non instantaneous changes to the accelerations. Then, the assumptions in (5.3) imply that there is a minimal joint displacement required for the robot to decelerate completely (to reach a stopped state $\dot{\boldsymbol{q}} = 0, \ddot{\boldsymbol{q}} = 0$) before hitting the joint limits described by $\mathcal{Q}$ (the jerk is not part of the stopped state because, under the current assumptions of not necessarily continuous jerk, it can be instantly set to zero). The set of states that do not lead to inevitable violations can be referred to as *viable* states [Prete, 2018].

As an example of a state leading to constraint violations, assume the joint states $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k, \dddot{\boldsymbol{q}}_k$ of the robot can be measured for a time step. The robot state may be instantly valid (i.e. $\boldsymbol{q}_k \in \mathcal{Q}$, $\dot{\boldsymbol{q}}_k \in \dot{\mathcal{Q}}$, $\ddot{\boldsymbol{q}}_k \in \ddot{\mathcal{Q}}$, $\dddot{\boldsymbol{q}}_k \in \dddot{\mathcal{Q}}$) but for some high enough $\dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k$, future states of the robot could end up outside the joint bounds $\boldsymbol{q}_n \notin \mathcal{Q}$ (for some time step n>k). The same logic can be applied to a future velocity $\dot{\boldsymbol{q}}_n$ (but not for the acceleration $\ddot{\boldsymbol{q}}_n$, because, under the assumptions mentioned in (5.3), the decision variable $\dddot{\boldsymbol{q}}_k$ can be changed instantly, and thus even if $\ddot{\boldsymbol{q}}_k$ is near its limit, a maximum deceleration jerk can start reducing it immediately).

A worst-case scenario[8] is shown in Figure 5.17 assuming the robot starts braking at the start of the trajectory: when the acceleration is at a maximum value and in the same sense as the current velocity (so it is driving the velocity towards its limit). Because of the smooth stopping requirements imposed, instant changes in acceleration are forbidden and will require reverting the acceleration at maximum jerk (for the acceleration to become opposite and start reducing the velocity) for a duration that depends on the jerk limits and the starting acceleration; during this time, the velocity will continue to increase towards its limit.

This phenomenon highlights an implicit incompatibility problem between constraints because they hide a potentially risky situation that may arrive from edge-case scenarios. The initial velocity shown in Figure 5.17 should not have been allowed from the start to avoid this situation. This implies that a contraction on the maximum velocity constraints caused by the higher order dynamics can help avoid inevitable violation states.

A constraint "compatibilization"[9] strategy in this sense that coherently compresses the limits can be proposed by using some worst-case scenario analysis. A theoretical proposal is presented without an experimental validation.

The higher order constraints impose contractions on the lower-order constraints. A trapezoidal acceleration profile (like the one shown in Figure 5.18) is employed for the generation of smooth stopping trajectories.

The proposal for this kind of situation is analyzed in Figure 5.19: avoiding instant

[8] not *the* worst case because the initial velocity is not maxed out, in which case the trajectory quickly surpasses the constraints

[9] As in long term compatibility, so it is rather a viability analysis.

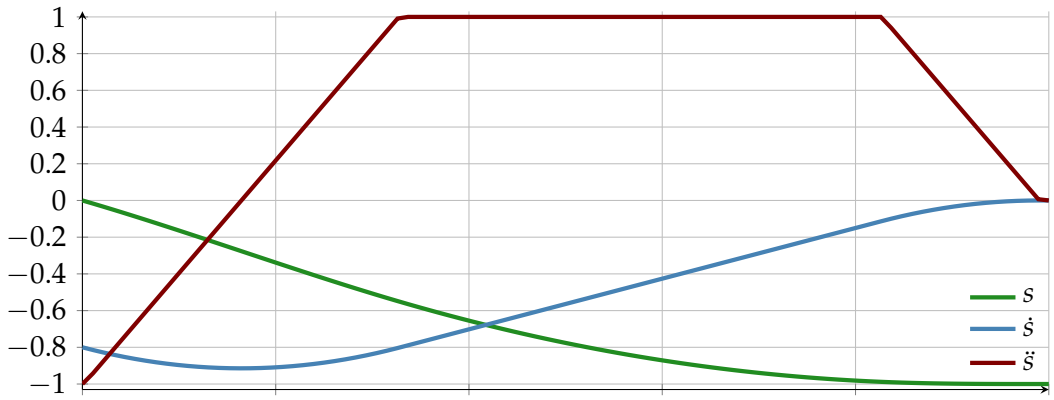


Figure 5.17: A smooth jerk-bounded (TAP) stopping trajectory example for a worst-case scenario: $v_0 = -0.7v_M, a_0 = -a_M$ (acceleration starts at maximum value and not inversed to velocity while velocity starts near its maximum value).

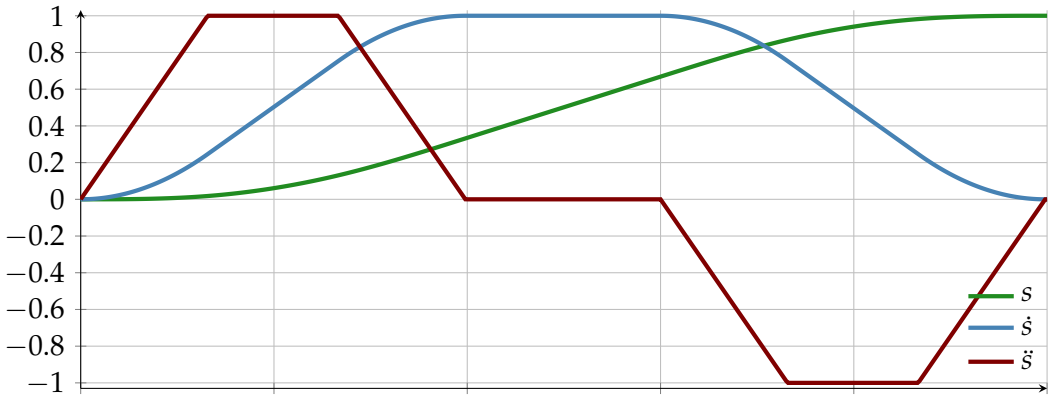


Figure 5.18: Normalized reference TAP trajectory generated with [Berscheid et al., 2021b].

changes in acceleration imposes a 2-stage stopping trajectory: an *acceleration reversing* stage; and a *deceleration stage* that continuously reduces the velocity up to a point where both acceleration and velocity reach zero, as is proposed in [Joseph et al., 2020].

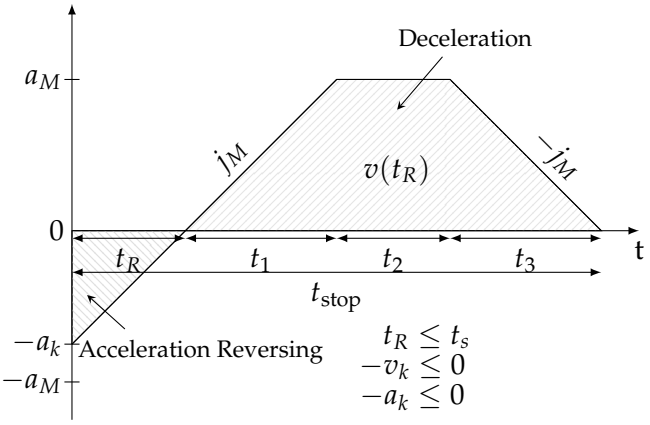


Figure 5.19: Smooth stopping trajectory with acceleration reversing, where the initial velocity and acceleration have the same direction, requiring a max-jerk trajectory to inverse the acceleration.

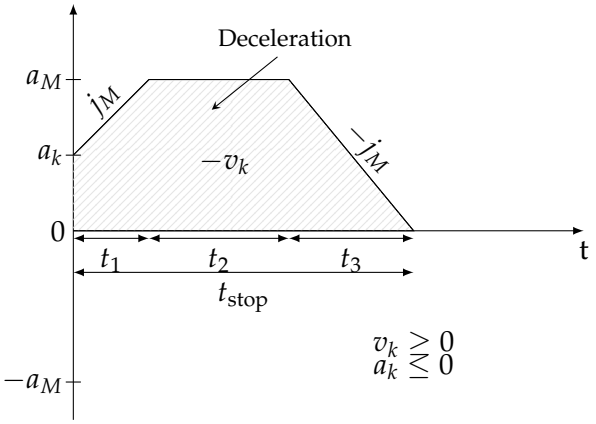


Figure 5.20: When the initial acceleration and velocity have opposed sense, the acceleration reversing stage is not necessary.

In order to find the viable set of constraints (that will avoid inevitable violation states [Rubrecht et al., 2010]), the smooth (as in jerk-bounded) stopping trajectories need to be computed. The way to do this is described in Appendix E. Applying this enables computing the worst-case scenario displacement for $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$:

$$\Delta \boldsymbol{q}_{stop} = \Delta \boldsymbol{q}_R + \Delta \boldsymbol{q}_D \tag{5.93}$$

$$\Delta \dot{\boldsymbol{q}}_{stop} = \Delta \dot{\boldsymbol{q}}_R + \Delta \dot{\boldsymbol{q}}_D \tag{5.94}$$

where the subdindices R, D refer to the *reversing* and *deceleration* stages, respectively. Furthermore, $\Delta \boldsymbol{q}_R = 0, \Delta \dot{\boldsymbol{q}}_R = 0$ for the stopping trajectories that require only a deceleration stage, as identified in Table 5.6 according to the initial velocity and acceleration. Extending the assumptions mentioned in (5.3) yields the *viable* Joint-space

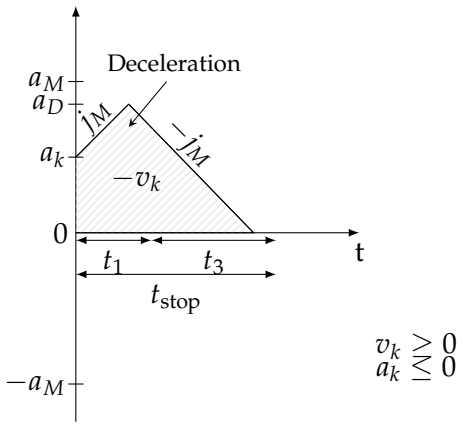


Figure 5.21: When the initial acceleration and velocity have opposed sense, but the initial velocity is low enough that the Deceleration stage does not require a constant acceleration part.

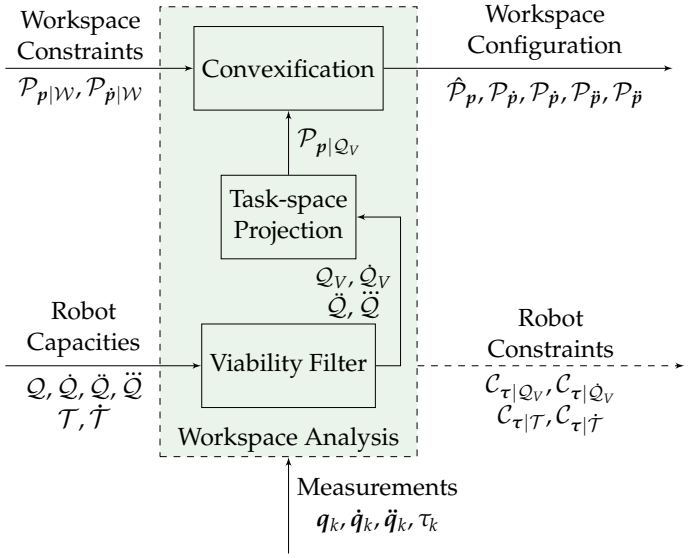


Figure 5.22: Polytope constraints from compatibilized joint-space constraints. revisited ↑

Constraints (JSC)[Prete, 2018][Rubrecht et al., 2012]:

$$\begin{aligned}\mathcal{Q}_V &= \{\boldsymbol{q}(t) + \Delta\boldsymbol{q}_{stop} \in \mathbb{R}^n \mid \boldsymbol{q} \in [\boldsymbol{q}_m, \boldsymbol{q}_M]\} \\ \dot{\mathcal{Q}}_V &= \{\dot{\boldsymbol{q}}(t) + \Delta\dot{\boldsymbol{q}}_{stop} \in \mathbb{R}^n \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_m, \dot{\boldsymbol{q}}_M]\}\end{aligned} \tag{5.95}$$

Finally, these new sets may be used to extend the feasible sets defined in Section 5.2.3. Having ensured that the joint level constraints are viable, theoretically ensures that the polyhedral feasible sets are also viable. So, these new sets can be defined by replacing $\mathcal{Q}_V, \dot{\mathcal{Q}}_V$ with $\mathcal{Q}, \dot{\mathcal{Q}}$, respectively.

		a_0 -	a_0 +
v_0	+	**D**	**R**
	-	**R**	**D**

Legends	
D	Only deceleration stage
R	Acceleration reversing and deceleration stage

Table 5.6: Worst-case scenario stages required for the stopping trajectory according to the initial velocity v_0 and acceleration a_0.

5.6 Conclusion ↑

Being able to quickly design and deploy collaborative cells without being constrained by the robot architecture is a fundamental requirement for agile manufacturing solutions. The present chapter presented a way extend the task-centric architecture proposal by considering joint level constraints at the task space. This is achieved by reformulating Joint-space Constraints (JSC) into the task space.

Not only does this help with the modularity of the control architecture, but it also helps fully utilize the robot capabilities, which leads to better dimensioning robots for the task required, potentially reducing the sizes and thus increasing the working cell safety and efficiency.

This work demonstrates some of the possibilities offered by an Model Predictive Control (MPC)-based architecture, showcasing the flexibility of the approach. Yet, the technique is quite open to further adaptations that can contemplate more complex scenarios. For instance, the human factor in manufacturing traditionally does not take into account the cognitive or ergonomic aspects of collaborative tasks, and these can be integrated into straightforwardly in the current form in many ways. For instance, constraints would allow expressing the human capabilities and exhaustion while tasks could be employed to optimize for better postures that minimize musculoskeletal lesions.

Furthermore, the MPC approach used here is defined for mathematically designed cost functions (i.e. minimizing a distance). Yet, one could imagine more complex tasks being devised or learned through machine learning approaches that can be readily be integrated with the same horizon optimization.

Chapter 6

Conclusion

This manuscript constructs some theoretical notions towards truly collaborative control approaches. Firstly, it contributes to the formulation of reactive motions trajectories. Secondly, it shows how these formulations can be exploited for online optimization-based control, achieving real-time motion adaptation. It proposes a way to integrate these high-level aspects of the tasks (behaviors) with the low level aspects of the robot architecture (like the actuators).

A first step into truly reactive motion adaptation is correctly parameterizing full pose trajectories (i.e. position and orientation). The mathematical foundations that enable formulating pose motions of free rigid body in space on the $\mathbb{SE}(3)$ manifold are addressed in Chapter 2. These foundations were employed to formalize an integration strategy based on a linear algebraic expression. This is used to efficiently approximate an integration scheme for fast pose estimations in gliding time horizons. The importance of this strategy for pose estimation is crucial in predictive and optimization-based control.

A second step towards reactive motion adaptation is exploiting this information for decision making. Chapter 3 presents a first approach to an optimization scheme exploiting the pose estimation strategy on the manifold. It addresses the challenge of efficient workspace sharing between collaborative robots and humans by proposing a Linear Model Predictive Control (MPC) adapting to changing workspace constraints and interactive task transitions in the Cartesian space. The controller enables online re-planning of position and orientation and is demonstrated through experiments with the Franka Emika robot.

In Chapter 4, the approach is further extended to obtain smoother behaviors, by working at the dynamics level. It focuses on optimizing the formulation to balance task achievement and computation time. This architecture dissociates planning from execution, utilizing a linear formalization in $\mathbb{SE}(3)$ to achieve task re-planning at a high-frequency closed-loop control rate. Experiments with the Franka Emika robot showcase the effectiveness of this approach in scenarios with dynamic task modifications.

Finally, Chapter 5 links back the task-space to the actuation level, introducing an enhancement to the control strategy by incorporating joint space constraints to better describe the robot actual actuation capabilities. This chapter outlines a method for transitioning from task space considerations to include these constraints, resulting in a reactive control strategy. This approach allows for the adjustment of task-space

movements while incorporating "actuation awareness" to ensure the robot operates within its capacities.

6.1 Perspectives ↑

The pertinence of the proposal in this manuscript create opportunities for potential future improvements in various aspects:

Mathematical Formulation

The approach presented in this manuscript exploits some mathematical structures in order to arrive at a convenient propagation scheme for pose motions. This enabled including the orientation in the planning algorithm. However, describing motions in $\mathbb{SE}(3)$ and/or intricate constraints in the workspace can often be nonintuitive. At the same time, while the current approach shows it is possible to generate motions in a horizon at the control rate, it did so by resorting to interpolation, which remains a rather coarse approach. These aspects are fundamental for designing more complex tasks, to formulate constraints that better adapt to collaborative scenarios and to improve the responsiveness.

While the approach shows a promising potential, the algebra employed by the algorithms can be generalized to a formulation based on geometric algebra. This approach offers the mechanisms for potentially more interpretable expressions with some efficiency gains [Löw et al., 2023].

On the constraints front, the presented approach paves the way for the study of task-specific polytope-type of constraints on manifolds that can better represent the workspace information/restrictions. One example of this would be the expression of richer orientation constraints. Beyond that, it can be used to better represent the available workspace for the robot to move (free of obstacles and humans) with an arbitrary precision. In particular, this type of polyhedrals on the manifold are known as orbitopes [Sanyal et al., 2011] and remain an open research topic for the robotics community.

Control Architecture

Furthermore, another unexplored question in the current manuscript pertains to the use of Model Predictive Control (MPC) at the joint level. From a performance perspective, these kind of approaches based explicitly on the joint states [Krämer et al., 2020] could deliver an improvement provided they can be implemented with approximately the same computational efficiency.

In fact, the use of this kind of approaches could be applied at two levels in the current architecture: for the task-space planning, in which case the validity of the model compared to the current approach remains an unexplored question but an interesting one; and secondly on the low level task-tracking controller.

Indeed, while the task-space planning algorithm is capable of modulating its behaviors according to a gliding time horizon, as it is now, the low level control lacks this capability and could benefit from optimizing the joint trajectory from the whole task-space horizon (even though some level of actuation awareness is already embedded in the generated horizon trajectories, in the form proposed in Chapter 5). Such an

approach presents an opportunity to improve the constraint viability through some efficient optimization approaches like differential dynamic programming [Tassa et al., 2014].

Another aspect unexplored in this work that merits exploration is the treatment of tasks that require manipulation [Gold et al., 2023] and contact forces [Kleff et al., 2022] from the same perspective of decoupling task and joint spaces. From a performance point of view, it presents an interesting line of questioning. Beyond that, it would better align to the modularity properties desired for collaborative robots. Furthermore, from an application standpoint, this would enable evaluating the presented actuation awareness approach while exploiting other aspects of the collaborative task, like the wrench capacities of the robot and the human [Skuric et al., 2021b]; [Skuric et al., 2021a].

Finally, the current manuscript focuses on collaborative robotics and more specifically, for the case of serial manipulators. However, many of the principles and results can be straightforwardly extended to other robot architectures. For instance, some control approaches for humanoid robots employ optimization methods to generate task-space reference trajectories [Lober et al., 2020] because doing so in the actuator space is too computationally intensive. This application could benefit from the actuation-aware task-space planning proposal to generate a-priori feasible trajectories.

Hybrid Learning Algorithms

On a more long term perspective, MPC algorithms in general offer a deterministic approach to modulating systems behaviors. However, the same mathematical principles that render them deterministic also limit their flexibility in terms of evolving behaviors: once deployed, these algorithms behave consistently and do not offer improving or fine tuning to the situation. This offers an opportunity for hybrid learning approaches that can leverage the model-based knowledge with adaptive algorithms [Bechtle et al., 2021] or equip the inherently non-deterministic control of learning approaches with some safety awareness [Fisac et al., 2019].

Furthermore, the tasks enabled by optimization-based controllers are limited by the design of the cost functions. In other words, the behaviors that are achievable by these approaches are limited to what can be described mathematically. Sometimes, finding a formulation that performs consistently proves to be a complex endeavor. This ultimately hinders the complexity of the tasks that can be obtained and opens doors for the exploration of learning algorithms that can better adjust their behaviors with more finely tuned tasks [Bogdanovic et al., 2019]; [Calinon, 2020].

Human Factor Considerations

Finally, a fundamental aspect of collaborative robotics is the human-centric perspective. This means that the human factor must be accounted for within the control schemes. Integrating these kind of aspects remains an open scientific question. However, the proposed control architecture offers the infrastructure for the inclusion of generic and human-aware constraints that can potentially express complex information about the human state such as the fatigue level [Savin et al., 2016].

Other human aspects that offer opportunities for richer human-robot collaboration

aspects stem from evaluating posture stress in real time (ergonomic aspects), the cognitive state [Camblor et al., 2022] and the expertise of the operators [Benhabib et al., 2020].

In the same way, the MPC can straightforwardly be formulated to follow a previously generated trajectory instead of a target pose. This would allow for path planning algorithms to be plugged right before the controller, integrating unexplored behaviors like, for example, human-aware collision avoidance.

6.2 Closing Remarks ↑

Collaborative robots constitute a fundamental piece in the ongoing modern industrial transformation towards human-centric and sustainable manufacturing. As such, the market demand for cobots has seen an increasing growth trend recently in market cap and is projected to even accelerate development for the future [Gambao, 2023]. Even though the cobot industry represents a small part of the overall robotics market cap, it still constitutes a young market with immense potential in the world and European market. Notably, Europe is the second main consumer of cobots in the world (after Asia) and is home to many of the biggest manufacturers.

Remarkably, the versatility and rapid adoption of cobots (as opposed to industrial robots) shows the potential to accelerate the development of multiple industrial domains. In this context, the continued investment in innovative ways to integrate cobots into every industry should be considered a strategic priority for industrial development.

Since their inception, cobots marked a hiatus from previous robot design trends to prioritize the new and (thus far) unconceived scenario of collaboration between humans and robots. These promises, while ambitious and enticing, have been ahead of the control strategies required to actually enable true collaboration.

The present work constitutes an effort towards showcasing and extending cobots flexibility in collaborative and dynamic environments. It shows that the promises of true human-robot collaboration demand a significant design effort. From the hardware point of view, robots are required to evolve (as introduced with cobots). From a control perspective, human-centric strategies demand an equally significant departure from classical control approaches. In the end, these are fundamental steps towards the true collaboration that will enable augmenting the human-robot synergy.

Appendix A

Quadratic Programming

Quadratic Programming is a technique that permits optimizing some vector variable $\boldsymbol{x}$ with respect to some second order cost function $f(\boldsymbol{x})$ and some linear equality and inequality constraints (lower/upper bounds).

The canonical QP optimization problem is formulated as:

$$\begin{aligned} \boldsymbol{x}^{opt} = \arg\min_{\boldsymbol{x}} \quad & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + r \\ \text{s.t.} \quad & \boldsymbol{C}_l \leq \boldsymbol{C}_i\boldsymbol{x} \leq \boldsymbol{C}_u \\ & \boldsymbol{b}_l \leq \boldsymbol{x} \leq \boldsymbol{b}_u \\ & \boldsymbol{C}_e\boldsymbol{x} = \boldsymbol{d} \end{aligned} \tag{A.1}$$

where:

- $\boldsymbol{x} \in R^{n\times 1}$ designates the optimizing variable (also referred to as the decision variable).
- $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ contains the quadratic cost function coefficients.
- $\boldsymbol{g} \in \mathbb{R}^{1\times n}$ refers to the linear coefficients, also referred to as the gradient vector.
- $r \in \mathbb{R}$ refers to a constant cost term, independent of the decision variable, and so it can be neglected, as it doesn't affect the result of the optimization.
- The *inequality constraints* are expressed as a linear system described with the lower and upper bounds vectors $\boldsymbol{C}_l, \boldsymbol{C}_u$ and the coefficients matrix $\boldsymbol{A}$ that relates them to the optimizing variable. This constitutes the most general set of constraints and is often used to formulate the other onces. Some solvers only offer this alternative (and equivalent) formulation:

$$\begin{bmatrix} \boldsymbol{C}_i \\ -\boldsymbol{C}_i \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} \boldsymbol{C}_u \\ -\boldsymbol{C}_l \end{bmatrix} \tag{A.2}$$

- The *equality constraints* are expressed as a linear system described with the vector $\boldsymbol{d}$ and the coefficients matrix $\boldsymbol{C}$ that relates them to the optimizing variable. They can be embedded into the inequality constraints by employing $\boldsymbol{d}$ both as the lower and upper bounds.

- Finally, the *variable bounds* express the decision variable lower $\boldsymbol{b}_l$ and upper $\boldsymbol{b}_u$ bounds. These can be included into the inequality constraints by adding an identity matrix to $\boldsymbol{C}$.

Embedded Constraints

Often, Quadratic Program (QP) solvers offer only the linear inequality constraints, as the others can be embedded in them for simplification:

$$\begin{bmatrix} \boldsymbol{b}_l \\ \boldsymbol{A}_l \\ \boldsymbol{d} \end{bmatrix} \leq \begin{bmatrix} \mathbf{I}_n \\ \boldsymbol{A} \\ \boldsymbol{C}_e \end{bmatrix} \boldsymbol{x} \leq \begin{bmatrix} \boldsymbol{b}_u \\ \boldsymbol{A}_u \\ \boldsymbol{d} \end{bmatrix} \tag{A.3}$$

Simplified formulation

Then, it is possible to arrive at a simplified canonical formulation:

$$\begin{aligned} \boldsymbol{x}^{opt} = \arg\min_{\boldsymbol{x}} \quad & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}_l \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{A}_u \end{aligned} \tag{A.4}$$

A.1 Weighted Least Square ↑

The following least-square optimization problem can be reformulated in a standard quadratic problem form as in (A.1):

$$\arg\min_{\boldsymbol{x}} \quad ||\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b}||^2_{\boldsymbol{W}} \tag{A.5}$$

where $\boldsymbol{W}$ is a positive semi-definite matrix containing weights the optimization variable.

Given:

$$\|\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b}\|^2_{\boldsymbol{W}} = (\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b})^T\boldsymbol{W}(\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b}) \tag{A.6}$$

$$= \boldsymbol{x}^T\boldsymbol{E}^T\boldsymbol{W}\boldsymbol{E}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{W}\boldsymbol{E}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b} \tag{A.7}$$

then we can express (A.5) in the initial formulation of the QP optimization (A.1):

$$\boldsymbol{Q} = 2\boldsymbol{E}^T\boldsymbol{W}\boldsymbol{E}, \qquad \boldsymbol{g}^T = -2\boldsymbol{b}^T\boldsymbol{W}\boldsymbol{E}, \qquad r = \boldsymbol{b}^T\boldsymbol{b} \tag{A.8}$$

A commonly used case of this least squares optimization is for when the weight matrix is an identity $\boldsymbol{W} = \mathbf{I}$ (giving equal weight to all elements in the decision variable):

$$\arg\min_{\boldsymbol{x}} \quad ||\boldsymbol{E}\boldsymbol{x} - \boldsymbol{b}||^2 \tag{A.9}$$

Appendix B

Discretization of Continuous LTI systems

A dynamic system can be parameterized with state $\boldsymbol{x} \in \mathbb{R}^n$ and input $\boldsymbol{u} \in \mathbb{R}^m$ vectors (where n is the size of the state and m the size of the input variables). The state vector is the smallest subset of system variables that contains all the information required to describe the system at some time instant. The input variable corresponds to the decision variable used to drive the system to some desired or target state.

Given some non-linear system dynamics described with $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, the continuous state-space system representations is:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_c(t)\boldsymbol{x}(t) + \boldsymbol{B}_c(t)\boldsymbol{u}(t) \tag{B.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) \ + \boldsymbol{D}(t)\boldsymbol{u}(t) \tag{B.2}$$

with:

$$\boldsymbol{A}_c(t) = \frac{\delta \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\delta \boldsymbol{x}} \qquad \boldsymbol{B}_c(t) = \frac{\delta \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\delta \boldsymbol{u}} \tag{B.3}$$

where:

- $\dot{\boldsymbol{x}}, \boldsymbol{x}$: represent the system state and its derivative
- $\boldsymbol{u}$: represents the system input
- $\boldsymbol{y}$: represents the system observed state
- $\boldsymbol{A}_c, \boldsymbol{B}_c$: LTI matrices that express the relation between the system state derivative and its state and input
- $\boldsymbol{C}, \boldsymbol{D}$: LTI matrices that express the relation between the system *observed* state and its state and input

The solution to the differential equation is:

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}_c t}\boldsymbol{x}(0) + \int_0^t e^{\boldsymbol{A}_c(t-\tau)}\boldsymbol{B}_c(\tau)\delta\tau \tag{B.4}$$

B.1 Discrete-time State-space ↑

The time-discrete equivalent of (B.1) and (B.2) are:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \boldsymbol{B}_d \boldsymbol{u}_k \tag{B.5}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k \tag{B.6}$$

Note that for the general case of a non-linear system, all these matrices also depend on the state. This means that, the system needs to be linearized at some time instant t_n and then $\boldsymbol{A}_{d,n}$ can be assumed constant for a limited time window. The same conclusions can be applied for $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$.

Using (B.4) with a sampling period of T_s, we arrive at the exact solutions:

$$\boldsymbol{A}_d = e^{\boldsymbol{A}_c T_s} \tag{B.7}$$

$$\boldsymbol{B}_d = \int_0^{T_s} e^{\boldsymbol{A}_c \tau} \delta\tau \boldsymbol{B}_c = \boldsymbol{K}\boldsymbol{B}_c \tag{B.8}$$

$$\text{with } \boldsymbol{K} = \boldsymbol{A}_c^{-1}(e^{\boldsymbol{A}_c T_s} - \mathbf{I}_n) \tag{B.9}$$

It is important to note that (B.8) is defined even if $\boldsymbol{A}_c^{-1}$ does not ($\boldsymbol{A}_c$ not reversible), as shown by developing the expressions though their taylor series:

$$\boldsymbol{A}_d = \mathbf{I}_n + \boldsymbol{A}_c T_s + \frac{(\boldsymbol{A}_c T_s)^2}{2} \ldots = \sum_{n=0}^{\infty} \frac{(\boldsymbol{A}_c T_s)^n}{n!} \tag{B.10}$$

$$\boldsymbol{K} = \boldsymbol{A}_c^{-1}(\sum_{n=0}^{\infty} \frac{(\boldsymbol{A}_c T_s)^n}{n!} - \mathbf{I}_n) \tag{B.11}$$

$$= \boldsymbol{A}_c^{-1}(\cancel{\mathbf{I}_n} + \boldsymbol{A}_c T_s + \frac{(\boldsymbol{A}_c T_s)^2}{2} \ldots - \cancel{\mathbf{I}_n}) \tag{B.12}$$

$$= \boldsymbol{A}_c^{-1} \sum_{n=1}^{\infty} \frac{(\boldsymbol{A}_c T_s)^n}{n!} \tag{B.13}$$

$$= \sum_{n=1}^{\infty} \frac{\boldsymbol{A}_c^{n-1} T_s^n}{n!} \tag{B.14}$$

Appendix C

Summary: Velocity-based Pose Tracking as a Quadratic Program

The inverse kinematic problem consists of finding the robot configuration satisfying some operational-space position and orientation. This section introduces a way to solve this problem for a robot that admits being controlled by its joint velocities.

This controller is similar to the one in [Joseph et al., 2020]. It is formulated as a linearly constrained Quadratic Program (QP) that optimizes for the joint velocity; the main difference being the added generic linear constraints in (C.5) that will be detailed shortly. The resulting QP formulation is:

$$\dot{\boldsymbol{q}}^{\text{opt}} = \arg\min_{\dot{\boldsymbol{q}}} \; f_{\text{main}}(\dot{\boldsymbol{q}}) + w_{\text{reg}}\, f_{\text{reg}}(\dot{\boldsymbol{q}}) \tag{C.1}$$

s.t.

$$\dot{\boldsymbol{q}}_m \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_M \tag{C.2}$$

$$\dot{\boldsymbol{q}} \in \boldsymbol{C}_q \tag{C.3}$$

$$\dot{\boldsymbol{q}} \in \boldsymbol{C}_{\ddot{q}} \tag{C.4}$$

$$\boldsymbol{A}_l \leq \boldsymbol{A}\dot{\boldsymbol{q}} \leq \boldsymbol{A}_u \tag{C.5}$$

$$\dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_m, \dot{\boldsymbol{q}}_M \in \mathbb{R}^n$$

with:

$$\boldsymbol{C}_q = \left\{ \dot{\boldsymbol{q}} \in \mathbb{R}^n \;\middle|\; \dot{\boldsymbol{q}} \in \left[\frac{\boldsymbol{q}_m - \boldsymbol{q}_k}{\Delta t}, \frac{\boldsymbol{q}_M - \boldsymbol{q}_k}{\Delta t} \right] \right\} \tag{C.6}$$

$$\boldsymbol{C}_{\ddot{q}} = \{ \dot{\boldsymbol{q}} \in \mathbb{R}^n \mid \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_k + \ddot{\boldsymbol{q}}_m \Delta t, \dot{\boldsymbol{q}}_k + \ddot{\boldsymbol{q}}_M \Delta t] \} \tag{C.7}$$

$$f_{\text{main}}(\dot{\boldsymbol{q}}) = ||{}^B\boldsymbol{v}^* - {}^B\boldsymbol{J}(\boldsymbol{q}_k)\dot{\boldsymbol{q}}||_2^2 \tag{C.8}$$

$$\begin{aligned} {}^B\boldsymbol{v}^* &= \boldsymbol{K}_p\, {}^B\boldsymbol{e} + {}^B\boldsymbol{v}^r \qquad & \boldsymbol{e}, \boldsymbol{v} &\in \mathfrak{se}(3) \\ {}^B\boldsymbol{e} &= \log(\mathcal{X}_k^{-1}\mathcal{X}^r) \qquad & \mathcal{X} &\in \mathbb{SE}(3) \end{aligned} \tag{C.9}$$

$$\begin{aligned} f_{\text{reg}}(\dot{\boldsymbol{q}}) &= ||\dot{\boldsymbol{q}}_{\text{reg}} - \dot{\boldsymbol{q}}||_2^2 \\ \dot{\boldsymbol{q}}_{\text{reg}} &= \boldsymbol{K}_q(\boldsymbol{q}_{\text{reg}} - \boldsymbol{q}) \\ \boldsymbol{q}_{\text{reg}} &= \tfrac{(\boldsymbol{q}_M - \boldsymbol{q}_m)}{2} \end{aligned} \tag{C.10}$$

where B implies the corresponding quantity is expressed in the *body frame* (see Section 3.2.4). Additionally:

- $\dot{\boldsymbol{q}}^{\text{opt}}$ corresponds to the optimal $\dot{\boldsymbol{q}}$ that minimizes the cost function; this implies it will be applied in the next step as $\dot{\boldsymbol{q}}_{k+1}$. On the other hand, the current states of the robot referred to as $\mathcal{X}_k, \boldsymbol{q}_k, \dot{\boldsymbol{q}}_k$ and are used in (C.6), (C.7), (C.8), (C.9) and (C.9).

- $f_{\text{main}}(\dot{\boldsymbol{q}})$ in (C.8) designates the main task cost function. In this case, it corresponds to a pose tracking task that minimizes the distance between some desired twist $\boldsymbol{\nu}^*$ (see (C.9) explained below) and the robot current twist, which is computed as a function of the optimizing variable (the joint velocities) through the robot Jacobian $\boldsymbol{J}(\boldsymbol{q})$.

- $\boldsymbol{\nu}^*$ from (C.8) is defined in (C.9) as a Proportional Derivative (PD) from a trajectory providing a reference pose $\mathcal{X}^r$ and twist $\boldsymbol{\nu}^r$. The proportional gain $\boldsymbol{K}_p$ provides a way to modulate the correction from the error term. Finally, $\boldsymbol{e}$ is constructed as the desired twist displacement required to correct the robot pose from $\mathcal{X}$ to $\mathcal{X}^r$.

- $f_{\text{reg}}(\dot{\boldsymbol{q}})$ is a regularisation task or function that helps uniquely determine an optimal solution in redudant robots by acting on the extra degrees of freedom. In this case, the task tries to maintain a configuration close to the robot mean position relative to its bounds. The details can be found in [Joseph et al., 2020]. Then w_{reg} is chosen small enough not to affect the main task.

- $\boldsymbol{C}_q$ in (C.6) offers one way to implement (C.3) via the first order taylor expansion on the current robot configuration $\boldsymbol{q}_k$, establishing a relation between the joint bound limits $(\boldsymbol{q}_m, \boldsymbol{q}_M)$ and the joint velocities.

- Similarly, $\boldsymbol{C}_{\ddot{q}}$ in (C.7) is one possible implementation of (C.4); in this case, it exploits the first order taylor expansion of the current joint velocity $\dot{\boldsymbol{q}}_k$ to incorporate the joint acceleration limits $(\ddot{\boldsymbol{q}}_m, \ddot{\boldsymbol{q}}_M)$ as a function of the decision variable, the joint velocities.

- Finally, (C.5) offers a way to incorporate linear constraints. For instance, it can be used to express workspace-related limits like the geometric space or, for example, the twist constraints can be formulated as:

$$\boldsymbol{\nu}_{b_l} \leq \boldsymbol{J}\dot{\boldsymbol{q}} \leq \boldsymbol{\nu}_{b_u} \tag{C.11}$$

Appendix D

Summary: Torque-based Pose Tracking as a Quadratic Program

An inverse kinematics solver as a Quadratic Program (QP) is presented in Appendix C. This section extends the previous strategy to formulate an inverse dynamics solver. While both problems involve working backwards to solve for unknowns, inverse kinematics focuses on the geometric view of the problem, solving for the joint angles necessary to achieve a certain end-effector pose, ignoring the required forces or torques. In contrast, inverse dynamics deals with the problem of finding the torques and forces required for an end-effector motion. This accounts for the system dynamic properties such as the mass, inertia, external forces, etc.

The controller presented here is similar to the one in [Joseph et al., 2018b]. It is designed for torque-controlled robots. By exploiting the robot dynamics model (see Section 4.2.1), we can formulate the joint accelerations as a function of the joint torques. Furthermore, the Cartesian spatial acceleration can then be related to the

joint accelerations through the robot Jacobian, enabling a QP formulation of the task-space tracking problem:

$$\boldsymbol{\tau}^{\text{opt}} = \arg\min_{\boldsymbol{\tau}} \; f_{\text{main}}(\boldsymbol{\tau}) + w_{\text{reg}} \, \boldsymbol{f}_{\text{reg}}(\boldsymbol{\tau}) \tag{D.1}$$

s.t.

$$\boldsymbol{\tau}_m \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_M \tag{D.2}$$

$$\boldsymbol{\tau} \in \boldsymbol{C}_{\boldsymbol{q}} \tag{D.3}$$

$$\boldsymbol{\tau} \in \boldsymbol{C}_{\dot{\boldsymbol{q}}} \tag{D.4}$$

$$\boldsymbol{\tau} \in \boldsymbol{C}_{\dot{\boldsymbol{\tau}}} \tag{D.5}$$

$$\begin{gathered} \boldsymbol{A}_{\boldsymbol{l}} \leq \boldsymbol{A}\dot{\boldsymbol{q}} \leq \boldsymbol{A}_{\boldsymbol{u}} \\ \boldsymbol{\tau}, \boldsymbol{\tau}_m, \boldsymbol{\tau}_M \in \mathbb{R}^n \end{gathered} \tag{D.6}$$

with:

$$\ddot{\boldsymbol{q}}(\boldsymbol{\tau}) = \boldsymbol{M}(\boldsymbol{q}_k)^{-1}(\boldsymbol{\tau} + \boldsymbol{b}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)) \tag{D.7}$$

$$\boldsymbol{C}_{\boldsymbol{q}} = \left\{ \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \mathbb{R}^n \mid \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \left[2\frac{(\boldsymbol{q}_m - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k \Delta t)}{\Delta t^2}, 2\frac{(\boldsymbol{q}_M - \boldsymbol{q}_k - \dot{\boldsymbol{q}}_k \Delta t)}{\Delta t^2} \right] \right\} \tag{D.8}$$

$$\boldsymbol{C}_{\dot{\boldsymbol{q}}} = \left\{ \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \mathbb{R}^n \mid \ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \in \left[\frac{(\dot{\boldsymbol{q}}_m - \dot{\boldsymbol{q}}_k)}{\Delta t}, \frac{(\dot{\boldsymbol{q}}_M - \dot{\boldsymbol{q}}_k)}{\Delta t} \right] \right\} \tag{D.9}$$

$$\boldsymbol{C}_{\dot{\boldsymbol{\tau}}} = \{ \boldsymbol{\tau} \in \mathbb{R}^n \mid \boldsymbol{\tau} \in [\boldsymbol{\tau}_k + \dot{\boldsymbol{\tau}}_m \Delta t, \boldsymbol{\tau}_k + \dot{\boldsymbol{\tau}}_M \Delta t] \} \tag{D.10}$$

$$\begin{gathered} f_{\text{main}}(\boldsymbol{\tau}) = ||{}^B\dot{\boldsymbol{\nu}}^* - \dot{\boldsymbol{\nu}}(\boldsymbol{\tau})||_2^2 \\ \dot{\boldsymbol{\nu}}(\boldsymbol{\tau}) = \boldsymbol{J}(\boldsymbol{q}_k)\ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \end{gathered} \tag{D.11}$$

$$\begin{gathered} {}^B\boldsymbol{\nu}^* = \boldsymbol{K}_p {}^B\boldsymbol{e} + \boldsymbol{K}_d {}^B\dot{\boldsymbol{e}} + {}^B\dot{\boldsymbol{\nu}}^r \qquad \boldsymbol{e}, \dot{\boldsymbol{e}}, \boldsymbol{\nu}, \dot{\boldsymbol{\nu}} \in \mathfrak{se}(3) \\ {}^B\boldsymbol{e} = \log(\mathcal{X}_k^{-1}\mathcal{X}^r) \qquad \mathcal{X} \in \mathbb{SE}(3) \\ {}^B\dot{\boldsymbol{e}} = {}^B\boldsymbol{\nu}^r - {}^B\boldsymbol{\nu}_k \end{gathered} \tag{D.12}$$

$$f_{\text{reg}}(\boldsymbol{\tau}) = ||\boldsymbol{\tau}_{\text{reg}} - \boldsymbol{\tau}||_2^2 \tag{D.13}$$

where B implies the corresponding quantity is expressed in the *body frame* (see Section 3.2.4). Additionally:

- $\boldsymbol{\tau}^{\text{opt}}$ corresponds to the optimal $\boldsymbol{\tau}$ that minimizes the cost function; this implies it will be applied in the next step as $\boldsymbol{\tau}_{k+1}$. On the other hand, the current states of the robot referred to as $\mathcal{X}_k, \boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{\tau}_k$ and are used in (D.7), (D.8), (D.10), (D.11) and (D.12).

- $f_{\text{main}}(\boldsymbol{\tau})$ in (D.11) designates the main task cost function. In this case, it corresponds to a pose tracking task that minimizes the distance between some desired twist derivative $\dot{\boldsymbol{\nu}}^*$ (analogous to a rigid body motion acceleration, see (D.12) explained below) and the robot current spatial acceleration, which is computed as a function of the optimizing variable (the joint torques) through the robot dynamics model in (D.7).

- $\boldsymbol{\nu}^*$ from (D.11) is defined in (D.12) as a Proportional Derivative (PD) term from

a trajectory providing a reference pose $\mathcal{X}^r$, a twist $\boldsymbol{\nu}^r$ and a spatial acceleration $\dot{\boldsymbol{\nu}}^r$ (the latter used as a feed forward term). The proportional gain $\boldsymbol{K}_p$ provides a way to modulate the correction from the error term $\boldsymbol{e}$, which is constructed as the desired twist displacement required to correct the robot pose from $\mathcal{X}$ to $\mathcal{X}^r$; in the same way, the derivative gain $\boldsymbol{K}_d$ modulates the twist error $\dot{\boldsymbol{e}}$.

- $f_{\text{reg}}(\boldsymbol{\tau})$ is a regularisation task or function that helps uniquely determine an optimal solution in redudant robots by acting on the extra degrees of freedom. In this case, the task tries to maintain a configuration close to the robot mean position relative to its bounds without affecting the main task (because w_{reg} is chosen small enough not to affect it). Then $\boldsymbol{\tau}_{\text{reg}}$ can be chosen in different ways depending on the targeted application; here, it is chosen to ensure the convergence of the robot configuration to a neutral configuration $\boldsymbol{q}_{\text{reg}}$. The servoing of this configuration can be expressed at the joint torque level using a PD controller such that:

$$\boldsymbol{\tau}_{\text{reg}} = \boldsymbol{K}_q(\boldsymbol{q}_{\text{reg}} - \boldsymbol{q}) - \boldsymbol{K}_{\dot{q}}\dot{\boldsymbol{q}} \tag{D.14}$$

  with $\boldsymbol{K}_q$ and $\boldsymbol{K}_{\dot{q}}$ positive proportional and derivative gains.

- $\boldsymbol{C}_q$ in (D.8) offers one way to implement (D.3) via the second order taylor expansion on the current robot configuration $\boldsymbol{q}_k$, establishing a relation between the joint bound limits $(\boldsymbol{q}_m, \boldsymbol{q}_M)$ and the joint accelerations (and indirectly, the joint torques).

- Similarly, $\boldsymbol{C}_{\dot{\tau}}$ in (D.10) is one possible implementation of (D.5); in this case, it exploits the first order taylor expansion of the current joint torque $\boldsymbol{\tau}_k$ to incorporate the joint torque derivative limits $(\dot{\boldsymbol{\tau}}_m, \dot{\boldsymbol{\tau}}_M)$ as a function of the decision variable, the joint torque.

- Finally, (D.6) offers a way to incorporate linear constraints. For instance, it can be used to express workspace-related limits like the geometric space or, for example, the spatial acceleration constraints can be formulated as:

$$\dot{\boldsymbol{\nu}}_{b_l} \leq \boldsymbol{J}\ddot{\boldsymbol{q}}(\boldsymbol{\tau}) \leq \dot{\boldsymbol{\nu}}_{b_u} \tag{D.15}$$

Given the interpolated first step of the trajectory computed by the Model Predictive Control (MPC), task space inverse dynamics has to be performed to compute the input joint torque for the robot.

Feeding the inverse dynamics solver with the desired acceleration computed by the MPC may sound tempting. Nevertheless, the feedback provided by the closed-loop MPC is, partly due to the interpolated output, not efficient enough to properly reject tracking errors related to the inaccuracies in the model of the robot.

These inaccuracies are mostly related to dry friction at the joint level, imperfectly rejected by the lower-level torque control loop in most robots. As a consequence, a PD controller including a feed-forward term in acceleration is used to compute a corrected desired acceleration.

For convenience and to further clarify how the different components are connected, this Section revisits some of the concepts previously introduced in Section 4.2.2:

$$\dot{\boldsymbol{\nu}}^* = \boldsymbol{K}_p \log(\mathcal{X}^{-1}\mathcal{X}_{\text{mpc}}) + \boldsymbol{K}_d(\boldsymbol{\nu}_{\text{mpc}} - \boldsymbol{\nu}) + \dot{\boldsymbol{\nu}}_{\text{mpc}} \tag{D.16}$$

where $\mathcal{X}_{\text{mpc}}, \boldsymbol{\nu}_{\text{mpc}}, \dot{\boldsymbol{\nu}}_{\text{mpc}}$ are respectively the interpolated desired pose, twist and acceleration outputted by the MPC; $\mathcal{X}, \boldsymbol{\nu}$ are the measured pose and twist of the robot and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are positive proportional and derivative gains.

Given this control acceleration, one can solve task-space inverse dynamics under constraints through a QP formulation, similar to the one in [Joseph et al., 2018b].

Appendix E

Smooth stopping trajectory

The smooth stopping analysis contemplates two stages in the trajectory, as depicted in Figure 5.19 and Table 5.6.

E.1 Acceleration Reversing Stage ↑

When the initial acceleration and velocity are in the same sens. This considers the two following starting conditions:

$$① = \begin{cases} 0 \leq a_k \leq a_M \\ 0 \leq v_k \leq v_M \end{cases}, \qquad ② = \begin{cases} a_M \leq a_k \leq 0 \\ v_M \leq v_k \leq 0 \end{cases} \tag{E.1}$$

yielding:

$$j_R(t) = \begin{cases} 0 & t = 0 \\ j_M & 0 < t < t_R \\ 0 & t = t_R \end{cases} \tag{E.2}$$

$$a_R(t) = \begin{cases} -a_k & t = 0 \\ -a_k + j_M t & 0 < t < t_R \\ 0 & t = t_R \end{cases} \tag{E.3}$$

$$v_R(t) = \begin{cases} -v_k & t = 0 \\ -v_k - a_k t + \frac{j_M t^2}{2} & 0 < t \leq t_R \end{cases} \tag{E.4}$$

$$\Delta p_R(t) = \begin{cases} 0 & t = 0 \\ -v_k t - \frac{a_k t^2}{2} + \frac{j_M t^3}{6} & 0 < t \leq t_R \end{cases} \tag{E.5}$$

and finally:

$$t_R = \frac{a_k}{j_M}, \quad v_R(t_R) = -v_k - \frac{a_k^2}{2j_M} \tag{E.6}$$

$$\rightarrow \Delta v_R(t_R) = v_k - v_R(t_R) \tag{E.7}$$

$$\rightarrow \Delta p_R(t_R) = -v_k \frac{a_k}{j_M} - \frac{2a_k^3}{6j_M^2} \tag{E.8}$$

E.2 Deceleration Stage ↑

When the initial acceleration is zero or in the opposite sens of the initial velocity. Note that if an acceleration reversing stage was necessary, then $a_k = 0, v_k = v_R(t_R)$.

$$j_D(t) = \begin{cases} 0 & t = 0 \\ j_M & 0 < t < t_1 \\ 0 & t_1 \leq t < t_{12} \\ -j_M & t_{12} \leq t < t_{123} \\ 0 & t_{123} \leq t \end{cases} \tag{E.9}$$

$$a_D(t) = \begin{cases} a_k & t = 0 \\ a_k + j_M t & 0 < t < t_1 \\ a_M & t_1 \leq t \leq t_{12} \\ a_M - j_M t & t_{12} < t < t_{123} \\ 0 & t_3 \leq t \end{cases} \tag{E.10}$$

$$v_D(t) = \begin{cases} -v_k & t = 0 \\ -v_k + a_k t + \frac{j_M t^2}{2} & 0 < t \leq t_1 \\ v_D(t_1) + a_M t & t_1 < t \leq t_{12} \\ v_D(t_{12}) + a_M t - \frac{j_M t^2}{2} & t_{12} < t < t_{123} \\ 0 & t_{123} \leq t \end{cases} \tag{E.11}$$

$$\Delta p_D(t) = \begin{cases} 0 & t = 0 \\ -v_k t - \frac{j_M t^3}{6} & 0 < t \leq t_1 \\ \Delta p_D(t_1) + v_D(t_1) t + \frac{a_M t^2}{2} & t_1 < t \leq t_{12} \\ \Delta p_D(t_{12}) + v_D(t_{12}) t + \frac{a_M t^2}{2} - \frac{j_M t^3}{6} & t_{12} < t \leq t_{123} \\ \Delta p_D(t_{123}) & t_{123} < t \end{cases} \tag{E.12}$$

where:

$$t_{12} = t_1 + t_2, \quad t_{123} = t_1 + t_2 + t_3, \quad t_1 = \frac{a_M - a_k}{j_M}, \quad t_3 = \frac{a_M}{j_M} \tag{E.13}$$

$$\Rightarrow \Delta v_D(t_{123}) = \int_0^{t_{123}} a_D(t)\, dt - v_k \tag{E.14}$$

$$= a_k t_1 + \frac{j_M t_1^2}{2} + a_M t_{12} + a_M t_{123} - \frac{j_M t_{123}^2}{2} \tag{E.15}$$

$$\Rightarrow \Delta p_D(t_{123}) = \Delta p_D(t_{12}) + v_D(t_{12}) t_{123} + \frac{a_M t_{123}^2}{2} - \frac{j_M t_{123}^3}{6} \tag{E.16}$$

$$= \underbrace{\Delta p_D(t_1) + v_D(t_1) t_{12} + \frac{a_M t_{12}^2}{2}}_{\Delta p_D(t_{12})} + \underbrace{(v_D(t_1) + a_M t_{12})}_{v_D(t_{12})} t_{123}$$

$$+ \frac{a_M t_{123}^2}{2} - \frac{j_M t_{123}^3}{6} \tag{E.17}$$

$$= \underbrace{-v_k t_1 - \frac{j_M t_1^3}{6}}_{\Delta p_D(t_1)} + \underbrace{(v_D(t_1) + a_M t_1)}_{v_D(t_1)} t_{12} + \frac{a_M t_{12}^2}{2}$$

$$+ \frac{a_M t_{123}^2}{2} - \frac{j_M t_{123}^3}{6} + \underbrace{(-v_k + a_k t_1 + \frac{j_M t_1^2}{2}}_{v_D(t_1)} + a_M t_{12}) t_{123} \tag{E.18}$$

$$= -v_k t_1 - \frac{j_M t_1^3}{6} + \underbrace{(-v_k t_1 - \frac{j_M t_1^3}{6}}_{v_D(t_1)} + a_M t_1) t_{12} + \frac{a_M t_{12}^2}{2}$$

$$+ \frac{a_M t_{123}^2}{2} - \frac{j_M t_{123}^3}{6} + (-v_k + a_k t_1 + \frac{j_M t_1^2}{2} + a_M t_{12}) t_{123} \tag{E.19}$$

Two cases considered:

$$\text{①} \quad 0 \le a_k \le a_m, \quad v_k \le \frac{2a_m^2 - a_k^2}{2j_M} \tag{E.20}$$

$$t_3 = \frac{a_m}{j_m}, \quad t_1 = \frac{a_m - a_k}{j_m} \tag{E.21}$$

Only a triangular trajectory is required.

$$\text{②} \quad 0 \le a_k, \quad v_k > \frac{2a_m^2 - a_k^2}{2j_M} \tag{E.22}$$

A full TAP trajectory is required.

(E.23)

Bibliography

Stellantis Communication (2023). *Kokomo factory report (April 2023)*. Folders : CORPORATE / Highlights / Group presentation / Our teams / Employee / Factory. URL: https://medialibrary-corporate.stellantis.com/home.
(Cited on page 4)

Julien Cresp Groupe PSA, Direction de la communication (2017a). *SITE DE PRODUCTION USINE DE MULHOUSE 2017*. Folders : CORPORATE / Highlights / Expertises / Sites & Buildings / Body assembly. URL: https://medialibrary-corporate.stellantis.com/home.
(Cited on page 4)

Wikipedia (2023). *Car — Wikipedia, The Free Encyclopedia*. https://web.archive.org/web/20230831211641/https://en.wikipedia.org/wiki/Car. [Online; accessed 31-August-2023].
(Cited on page 3)

Stellantis communication (2023). *Press release - First Half 2023 Results - English version*. Folders : CORPORATE / Highlights / Key Events / Events, CORPORATE / Current Events / First Half 2023 Results / Press release, CORPORATE / Highlights / Finance / Financial Results / First Half 2023 Results / Press release. URL: https://medialibrary-corporate.stellantis.com/home.
(Cited on page 4)

Beauville Dit Eynaud, Amélie (June 2020). "Démarche de conception d'un système de production industriel reconfigurable, dans un contexte de fortes variations de marché en volume et en diversité". Theses. HESAM Université.
(Cited on pages 4, 5)

Voilqué, Anthony (July 2020). "Formalisation du problème de conception d'un exosquelette industriel et application à l'assistance aux efforts des membres supérieurs". Theses. Université Clermont Auvergne [2017-2020].
(Cited on page 4)

Ghobakhloo, Morteza (2020). "Industry 4.0, digitization, and opportunities for sustainability". In: *Journal of cleaner production* 252, p. 119869.
(Cited on page 5)

Gunasekaran, A (1999). "Agile manufacturing: A framework for research and development". In: *International Journal of Production Economics* 62.1, pp. 87–105.
(Cited on page 5)

Demir, Kadir Alpaslan, Gözde Döven, and Bülent Sezen (2019). "Industry 5.0 and Human-Robot Co-working". In: *Procedia Computer Science* 158. 3rd WORLD CONFERENCE ON TECHNOLOGY, INNOVATION AND ENTREPRENEURSHIP"INDUSTRY 4.0 FOCUSED INNOVATION, TECHNOLOGY, ENTREPRENEURSHIP AND MANUFACTURE" June 21-23, 2019, pp. 688–695.
(Cited on page 5)

Lu, Yuqian et al. (2022). "Outlook on human-centric manufacturing towards Industry 5.0". In: *Journal of Manufacturing Systems* 62, pp. 612–627.
(Cited on page 5)

Colgate, James E. and Michael A. Peshkin (U.S. Patent 5952796A, Feb. 1996). *Cobots*.

(Cited on page 6)

Groupe PSA Communication Metz Tremery (2018). *Site PSA Group Metz: Production of gearboxes*. Folders : CORPORATE / Highlights / Expertises / Sites & Buildings. URL: https://medialibrary-corporate.stellantis.com/home.

(Cited on page 6)

Stellantis Communication (2022a). *Stellantis Metz : e-Transmissions production*. Folders : CORPORATE / Highlights / Group presentation / Our teams / Employee / Factory. URL: https://medialibrary-corporate.stellantis.com/home.

(Cited on page 6)

Julien Cresp Groupe PSA, Direction de la communication (2017b). *SITE DE PRODUCTION USINE DE MULHOUSE 2017*. Folders : CORPORATE / Highlights / Expertises / Sites & Buildings / Assembly line. URL: https://medialibrary-corporate.stellantis.com/home.

(Cited on page 7)

"Cobot architecture" (2001). In: *IEEE Transactions on Robotics and Automation* 17.4, pp. 377–390.

(Cited on page 6)

Proia, Silvia et al. (2022). "Control Techniques for Safe, Ergonomic, and Efficient Human-Robot Collaboration in the Digital Industry: A Survey". In: *IEEE Transactions on Automation Science and Engineering* 19.3, pp. 1798–1819.

(Cited on pages 7, 16)

Stellantis Communication (2022b). *Trémery factory : assembly of electric powertrain*. Folders : CORPORATE / Highlights / Group presentation / Our teams / Employee / Factory. URL: https://medialibrary-corporate.stellantis.com/home.

(Cited on page 8)

julien cresp/Within (2023). *Poissy factory : Assembly shop production step 25*. Folders : CORPORATE / Highlights / Group presentation / Our teams / Employee / Factory. URL: https://medialibrary-corporate.stellantis.com/home.

(Cited on page 8)

Haddadin, Sami, Alin Albu-Schäffer, and Gerd Hirzinger (2009). "Requirements for safe robots: Measurements, analysis and new insights". In: *The International Journal of Robotics Research* 28.11-12, pp. 1507–1527.

(Cited on page 7)

Robla-Gómez, S. et al. (2017). "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments". In: *IEEE Access* 5, pp. 26754–26773.

(Cited on page 7)

Fryman, Jeff and Bjoern Matthias (2012). "Safety of Industrial Robots: From Conventional to Collaborative Applications". In: *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–5.

(Cited on page 7)

Rosenstrauch, Martin J. and Jörg Krüger (2017). "Safe human-robot-collaboration-introduction and experiment using ISO/TS 15066". In: *3rd International Conference on Control, Automation and Robotics*.

(Cited on page 7)

Hanna, Atieh et al. (2022). "Deliberative safety for industrial intelligent human–robot collaboration: Regulatory challenges and solutions for taking the next step towards industry 4.0". In: *Robotics and Computer-Integrated Manufacturing* 78, p. 102386.

(Cited on page 7)

Fryman, Jeff (2014). "Updating the industrial robot safety standard". In: *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE, pp. 1–4.
(Cited on page 7)

Chemweno, Peter, Liliane Pintelon, and Wilm Decre (2020). "Orienting safety assurance with outcomes of hazard analysis and risk assessment: A review of the ISO 15066 standard for collaborative robot systems". In: *Safety Science* 129, p. 104832.
(Cited on page 7)

Wang, L. et al. (2019). "Symbiotic human-robot collaborative assembly". In: *CIRP Annals* 68.2, pp. 701–726.
(Cited on page 7)

Tsarouchi, Panagiota, Sotiris Makris, and George Chryssolouris (2016). "Human–robot interaction review and challenges on task planning and programming". In: *International Journal of Computer Integrated Manufacturing* 29.8, pp. 916–931.
(Cited on page 7)

Skuric, Antun, Vincent Padois, and David Daney (May 2021a). "On-line force capability evaluation based on efficient polytope vertex search". In: *IEEE International Conference on Robotics and Automation*. Xi'an, China.
(Cited on pages 9, 100, 106, 107, 152)

Joseph, Lucas, Vincent Padois, and Guillaume Morel (Nov. 2018a). "Experimental validation of an energy constraint for a safer collaboration with robots". In: *International Symposium on Experimental Robotics*. Buenos Aires, Argentina.
(Cited on pages 9, 66, 99)

ISO/TS-15066 (2016). *Robots and robotic devices - Collaborative robots*. International Organization for Standardization. Geneva, Switzerland.
(Cited on pages 10, 66, 116)

Raiola, Gennaro et al. (2018). "Development of a Safety- and Energy-Aware Impedance Controller for Collaborative Robots". In: *IEEE Robotics and Automation Letters* 3.2, pp. 1237–1244.
(Cited on pages 10, 66)

Meguenani, Anis, Vincent Padois, and Philippe Bidaud (2015). "Control of robots sharing their workspace with humans: An energetic approach to safety". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4678–4684.
(Cited on pages 10, 66)

Joseph, Lucas et al. (Oct. 2020). "Online velocity constraint adaptation for safe and efficient human-robot workspace sharing". In: *International Conference on Intelligent Robots and Systems*. Las Vegas, United States.
(Cited on pages 10, 47, 55, 71, 78, 147, 158, 159)

Rubrecht, Sébastien et al. (Oct. 2010). "Constraints Compliant Control: constraints compatibility and the displaced configuration approach". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei, Taiwan, pp. 677–684.
(Cited on pages 10, 47, 55, 59, 78, 84, 99, 147)

Prete, Andrea Del (2018). "Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators". In: *IEEE Robotics and Automation Letters* 3.1, pp. 281–288.
(Cited on pages 10, 47, 78, 82, 100, 144, 145, 148)

Kleff, Sébastien et al. (May 2021). "High-Frequency Nonlinear Model Predictive Control of a Manipulator". In: *IEEE International Conference on Robotics and Automation*. Xi'an, China.
(Cited on pages 10, 16, 59, 68, 79)

Gold, Tobias, Andreas Völz, and Knut Graichen (2023). "Model Predictive Interaction Control for Robotic Manipulation Tasks". In: *IEEE Transactions on Robotics* 39.1, pp. 76–89.
(Cited on pages 10, 59, 152)

Forster, Christian et al. (2015). "On-Manifold Preintegration Theory for Fast and Accurate Visual-Inertial Navigation". In: *CoRR* abs/1512.02363.
(Cited on pages 11, 31, 33–35, 62, 63, 68, 89)

Solà, Joan, Jérémie Deray, and Dinesh Atchuthan (2018). "A micro Lie theory for state estimation in robotics". In: *CoRR* abs/1812.01537.
(Cited on pages 11, 24, 31, 33, 34, 37, 63, 89, 90, 124)

Torres Alberto, Nicolas et al. (June 2022a). "A linearization method based on Lie algebra for pose estimation in a time horizon". In: *18th International Symposium on Advances in Robot Kinematics*. Bilbao, Spain.
(Cited on pages 11, 33, 63, 68, 89)

Skuric, Antun et al. (Sept. 2022a). "Online task-space trajectory planning using real-time estimations of robot motion capabilities". working paper or preprint.
(Cited on page 12)

Torres Alberto, Nicolas et al. (Apr. 2023). "Model Predictive Control for robots adapting their task space motion online". working paper or preprint.
(Cited on page 12)

– (Sept. 2022b). "Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces". working paper or preprint.
(Cited on page 12)

Esquerre-Pourtère, Arthur, Nicolas Torres Alberto, and Vincent Padois (Apr. 2022). "From a trapezoidal acceleration profile to a learnt time optimal control policy for robot braking". working paper or preprint.
(Cited on page 12)

Ajoudani, Arash et al. (2018). "Progress and prospects of the human–robot collaboration". In: *Autonomous Robots* 42.5, pp. 957–975.
(Cited on page 16)

Proia, Silvia et al. (2021). "A Literature Review on Control Techniques for Collaborative Robotics in Industrial Applications". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 591–596.
(Cited on page 16)

Escande, Adrien, Nicolas Mansard, and Pierre-Brice Wieber (2014). "Hierarchical quadratic programming: Fast online humanoid-robot motion generation". In: *The International Journal of Robotics Research* 33.7, pp. 1006–1028.
(Cited on page 16)

Ibanez, Aurélien, Philippe Bidaud, and Vincent Padois (2019). "Optimization-Based Control Approaches to Humanoid Balancing". In: *Humanoid Robotics: A Reference*. Ed. by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, pp. 1541–1567.
(Cited on page 16)

Müller, Andreas (Feb. 2018). "Screw and Lie group theory in multibody dynamics: Recursive algorithms and equations of motion of tree-topology systems". en. In: *Multibody System Dynamics* 42.2, pp. 219–248.
(Cited on page 16)

Huber, Gerold and Dirk Wollherr (2020). "An Online Trajectory Generator on SE(3) for Human–Robot Collaboration". In: *Robotica* 38.10, pp. 1756–1777.
(Cited on page 16)

Casas, Fernando and Arieh Iserles (May 2006). "Explicit Magnus expansions for nonlinear equations". In: *J. Phys. A: Math. Gen* 39, pp. 5445–5461.
(Cited on page 16)

Lynch, Kevin M and Frank C Park (2017). "Modern Robotics: Mechanics, Planning, and Control". In: Cambridge University Press.
(Cited on pages 18, 24, 27, 28, 31, 38, 53, 62, 81)

Siciliano, Bruno and Oussama Khatib, eds. (2008). *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer.
(Cited on pages 18, 24, 53, 81)

Craig, John J. (1986). *Introduction to Robotics : Mechanics & Control*. Includes bibliographies and index. Reading, Mass.: Addison-Wesley Pub. Co.,
(Cited on page 18)

Murray, Richard M., S. Shankar Sastry, and Li Zexiang (1994). *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc.
(Cited on pages 18, 24)

Brockett, Roger W. (1984). "Robotic manipulators and the product of exponentials formula". In.
: (cited on pages 19, 32)

Merker, Joel (2010). *Theory of Transformation Groups, by S. Lie and F. Engel (Vol. I, 1888). Modern Presentation and English Translation*.
(Cited on page 20)

Lang, Serge (2002). *Algebra*. New York, NY: Springer.
(Cited on page 20)

Euler, Leonhard (1765). *Theoria motus corporum solidorum seu rigidorum (etc.)(Cum tabulis aeneis.)* AF Röse.
(Cited on page 21)

Rodrigues, Olinde (1840). "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire". fr. In: *Journal de Mathématiques Pures et Appliquées* 1e série, 5.
(Cited on pages 21, 25)

Gogu, Grigore and Philippe Coiffet (Jan. 1996). *Représentation du mouvement des corps solides*.
(Cited on page 22)

Hamano, Fumio (2013). "Derivative of Rotation Matrix Direct Matrix Derivation of Well Known Formula". In: *CoRR* abs/1311.6010.
(Cited on page 22)

Zhao, Shiyu (2016). "Time Derivative of Rotation Matrices: A Tutorial". In: *CoRR* abs/1609.06088.
(Cited on page 22)

Grassia, F. Sebastian (1998). "Practical Parameterization of Rotations Using the Exponential Map". In: *Journal of Graphics Tools* 3.3, pp. 29–48.
(Cited on page 24)

Chirikjian, Gregorg S. (2005). "Rigid-Body Kinematics". In: *Robotics and Automation Handbook*. Ed. by T. R. Kurfess. CRC, Boca Raton.
(Cited on page 24)

Hairer, Ernst, Christian Lubich, and Gerhard Wanner (2006). *Geometric numerical integration*. Second. Vol. 31. Springer Series in Computational Mathematics. Structure-preserving algorithms for ordinary differential equations. Springer-Verlag, Berlin, pp. xviii+644.
(Cited on pages 24, 28, 36)

Chasles, Michel (1830). "Note sur les propriétés génerales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre [A note on the general properties of a system of two similar bodies arbitrarily positioned in space; and on the finite or infinitely small displacement of an unconstrained solid body]". In: *Bulletin des Sciences Mathématiques, Férussac* 14, pp. 321–26.

(Cited on page 27)

Davidson, Joseph K, Kenneth H Hunt, and Gordon R Pennock (2004). "Robots and screw theory: applications of kinematics and statics to robotics". In: *J. Mech. Des.* 126.4, pp. 763–764.

(Cited on page 27)

Traversaro, S. and Alessandro Saccon (Nov. 2019). *Multibody dynamics notation (version 2)*. English. Tech. rep. Dept. of Mechanical Engineering. Report locator DC 2019.100.

(Cited on pages 31, 56)

Absil, P.-A., R. Mahony, and R. Sepulchre (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press, pp. xvi+224.

(Cited on page 34)

Saccon, Alessandro, John Hauser, and A. Pedro Aguiar (Sept. 2013). "Optimal Control on Lie Groups: The Projection Operator Approach". In: *Automatic Control, IEEE Transactions on* 58.9.

(Cited on pages 34, 62, 68)

Boumal, Nicolas (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press.

(Cited on pages 34, 64)

Lee, John M. (2013). *Introduction to smooth manifolds*. Graduate Texts in Mathematics. New York: Springer International Publishing.

(Cited on page 34)

Boothby, William Munger (1986). *An introduction to differentiable manifolds and Riemannian geometry; 2nd ed.* Pure and applied mathematics (Elsevier). Orlando, FL: Academic Press.

(Cited on page 34)

Magnus, Wilhelm (1954). "On the exponential solution of differential equations for a linear operator". In: *Communications on Pure and Applied Mathematics* 7.4, pp. 649–673.

(Cited on pages 35, 36)

Zanna, Antonella (June 1999). "Collocation and Relaxed Collocation for the Fer and the Magnus Expansions". In: *Siam Journal on Numerical Analysis - SIAM J NUMER ANAL* 36.

(Cited on page 36)

Iserles, Arieh et al. (2000). "Lie-group methods". In: *Acta Numerica* 9, pp. 215–365.

(Cited on page 36)

Iserles, Arieh and Syvert Norsett (Apr. 1999). "On the solution of linear differential equations in Lie groups". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 357.

(Cited on page 36)

Blanes, Sergio et al. (Nov. 2015). "Fer and Magnus Expansions". In: pp. 508–513.

(Cited on page 36)

Schur, Friedrich (1891). "Zur Theorie der endlichen Transformationsgruppen". und. In: *Mathematische Annalen* 38, pp. 263–286.

(Cited on page 36)

Rossmann, Wulf (2006). *Lie Groups: An Introduction Through Linear Groups*. en. Oxford University Press.

(Cited on page 36)

Blanes, Sergio et al. (Nov. 2008). "The Magnus expansion and some of its applications. Physics Reports, 470(5-6), 151-238". In: *Physics Reports* 470, pp. 151–238.

(Cited on page 36)

Berscheid, Lars and Torsten Kröger (2021a). "Jerk-limited Real-time Trajectory Generation with Arbitrary Target States". In: *Robotics: Science and Systems XVII*.

(Cited on page 41)

Haddadin, Sami and Elizabeth Croft (2016). "Physical Human–Robot Interaction". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, pp. 1835–1874.
(Cited on page 46)

Gualtieri, Luca, Erwin Rauch, and Renato Vidoni (2021). "Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review". In: *Robotics and Computer-Integrated Manufacturing* 67, p. 101998.
(Cited on page 46)

Liu, Mingxing, Yang Tan, and Vincent Padois (2015). "Generalized hierarchical control". In: *Autonomous Robots* 40.1.
(Cited on pages 47, 78, 79)

Salini, Joseph, Vincent Padois, and Philippe Bidaud (May 2011). "Synthesis of Complex Humanoid Whole-Body Behavior: a Focus on Sequencing and Tasks Transitions". In: *IEEE International Conference on Robotics and Automation*. Shanghai, China, pp. 1283–1290.
(Cited on pages 47, 51)

Kurniawati, Hanna, Tirthankar Bandyopadhyay, and Nicholas Patrikalakis (Jan. 2011). "Global Motion Planning under Uncertain Motion, Sensing, and Environment Map." In: vol. 33.
(Cited on page 47)

Rubrecht, Sébastien et al. (Apr. 2012). "Motion safety and constraints compatibility for multibody robots". In: *Autonomous Robots* 32.3, pp. 333–349.
(Cited on pages 47, 145, 148)

Spaa, Linda van der et al. (2020). "Predicting and Optimizing Ergonomics in Physical Human-Robot Cooperation Tasks". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1799–1805.
(Cited on page 49)

Du, Guanglong et al. (2018). "Active collision avoidance for human-robot interaction with ukf, expert system, and artificial potential field method". In: *Frontiers in Robotics and AI* 5, p. 125.
(Cited on page 49)

Mohammed, Abdullah, Bernard Schmidt, and Lihui Wang (2017). "Active collision avoidance for human–robot collaboration driven by vision sensors". In: *International Journal of Computer Integrated Manufacturing* 30.9, pp. 970–980.
(Cited on page 49)

Safeea, Mohammad, Pedro Neto, and Richard Bearee (Sept. 2019). "On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case". In: *Robotics and Autonomous Systems* 119, pp. 278–288.
(Cited on page 49)

Halme, Roni-Jussi et al. (2018). "Review of vision-based safety systems for human-robot collaboration". In: *Procedia Cirp* 72, pp. 111–116.
(Cited on page 49)

Krämer, Maximilian et al. (2020). "Model predictive control of a collaborative manipulator considering dynamic obstacles". en. In: *Optimal Control Applications and Methods* 41.4.
(Cited on pages 49, 151)

Spahn, Max, Martijn Wisse, and Javier Alonso-Mora (2023). "Dynamic Optimization Fabrics for Motion Generation". In: *IEEE Transactions on Robotics*, pp. 1–16.
(Cited on page 49)

Bajcsy, Andrea et al. (2021). "A Robust Control Framework for Human Motion Prediction". In: *IEEE Robotics and Automation Letters* 6.1, pp. 24–31.
(Cited on page 49)

Weitschat, Roman and Harald Aschemann (2018). "Safe and Efficient Human–Robot Collaboration Part II: Optimal Generalized Human-in-the-Loop Real-Time Motion Generation". In: *IEEE Robotics and Automation Letters* 3.4, pp. 3781–3788.
(Cited on page 49)

Tan, Yang et al. (2015). "Minimization of the rate of change in torques during motion and force control under discontinuous constraints". In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2621–2628.
(Cited on page 50)

Benzi, Federico and Cristian Secchi (2021). "An optimization approach for a robust and flexible control in collaborative applications". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3575–3581.
(Cited on page 50)

Wang, Lihui, Bernard Schmidt, and Andrew Y.C. Nee (2013). "Vision-guided active collision avoidance for human-robot collaborations". In: *Manufacturing Letters* 1.1, pp. 5–8.
(Cited on page 50)

Scalera, Lorenzo et al. (2022). "Enhancing fluency and productivity in human-robot collaboration through online scaling of dynamic safety zones". In: *The International Journal of Advanced Manufacturing Technology* 121.9-10, pp. 6783–6798.
(Cited on page 50)

Wieber, Pierre-brice (2006). "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations". In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pp. 137–142.
(Cited on page 50)

Lober, Ryan, Olivier Sigaud, and Vincent Padois (2020). "Task Feasibility Maximization Using Model-Free Policy Search and Model-Based Whole-Body Control". In: *Frontiers in Robotics and AI* 7.
(Cited on pages 50, 68, 144, 152)

Joseph, Lucas (Dec. 2018). "An energetic approach to safety in robotic manipulation". Theses. Sorbonne Université.
(Cited on pages 51, 55, 104)

Joseph, Lucas, Vincent Padois, and Guillaume Morel (May 2019). "Online minimization of the projected mass of a robot for safe workspace sharing with a human". In: *Workshop on "Human movement science for physical human-robot collaboration" at the IEEE International Conference on Robotics and Automation*. Montreal, Canada.
(Cited on pages 55, 99)

Carpentier, Justin et al. (2019). "The Pinocchio C++ library, a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives". In: *IEEE International Symposium on System Integrations*.
(Cited on pages 56, 70, 90, 124, 130)

Featherstone, Roy (2008). *Rigid Body Dynamics Algorithms*. Springer US.
(Cited on page 56)

Rawlings, J., D.Q. Mayne, and Moritz Diehl (Jan. 2017). *Model Predictive Control: Theory, Computation, and Design*.
(Cited on page 58)

Borrelli, Francesco, Alberto Bemporad, and Manfred Morari (2017). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press.
(Cited on page 58)

Rossiter, John (Jan. 2003). *Model-Based Predictive Control: A Practical Approach*.
(Cited on page 58)

Kress-Gazit, Hadas et al. (Aug. 2021). "Formalizing and guaranteeing human-robot interaction". In: *Communications of the ACM* 64.9, pp. 78–84.
(Cited on page 59)

Matschek, Janine, Johanna Bethge, and Rolf Findeisen (2023). "Safe Machine-Learning-Supported Model Predictive Force and Motion Control in Robotics". In: *IEEE Transactions on Control Systems Technology*, pp. 1–13.
(Cited on page 59)

Bednarczyk, Maciej, Hassan Omran, and Bernard Bayle (2020). "Model Predictive Impedance Control". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4702–4708.
(Cited on page 59)

Oleinikov, Artemiy et al. (2021). "Safety-Aware Nonlinear Model Predictive Control for Physical Human-Robot Interaction". In: *IEEE Robotics and Automation Letters* 6.3, pp. 5665–5672.
(Cited on page 59)

Terze, Zdravko, Andreas Mueller, and Dario Zlatar (July 2015). "Lie-Group Integration Method for Constrained Multibody Systems in State Space". In: *Multibody System Dynamics* 33, pp. 1–33.
(Cited on page 59)

Knyazev, Andrew and Alexander Malyshev (2015). "Continuation model predictive control on smooth manifolds". In: *IFAC-PapersOnLine* 48.25. arXiv: 1509.02848, pp. 126–131.
(Cited on page 59)

Chang, Dong Eui, Karmvir Singh Phogat, and Jongeun Choi (July 2020). "Model Predictive Tracking Control for Invariant Systems on Matrix Lie Groups via Stable Embedding into Euclidean Spaces". In: *IEEE Transactions on Automatic Control* 65.7. arXiv: 1910.05669, pp. 3191–3198.
(Cited on page 59)

Tan, Yang (Mar. 2016). "Continuous tasks and constraints transitions for the control of robots". Theses. Université Pierre et Marie Curie - Paris VI.
(Cited on page 62)

Mayne, D.Q. et al. (2000). "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6, pp. 789–814.
(Cited on page 62)

Bard, Yonathan (1974). *Nonlinear parameter estimation*. New York: Academic Press.
(Cited on page 62)

Alexis, K., G. Nikolakopoulos, and A. Tzes (Aug. 2012). "Model predictive quadrotor control: Attitude, altitude and position experimental studies". English (US). In: *IET Control Theory and Applications* 6.12, pp. 1812–1827.
(Cited on page 62)

Bemporad, Alberto et al. (2002). "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1, pp. 3–20.
(Cited on page 62)

Lee, John M. (2019). *Introduction to Riemannian Manifolds*. Graduate Texts in Mathematics. Springer International Publishing.
(Cited on page 64)

Jayasumana, Sadeep et al. (2014). "Kernel Methods on the Riemannian Manifold of Symmetric Positive Definite Matrices". In: *CoRR* abs/1412.4172.
(Cited on pages 64, 90)

Novelia, Alyssa and Oliver M O'Reilly (2015). "On geodesics of the rotation group SO (3)". In: *Regular and Chaotic Dynamics* 20, pp. 729–738.
(Cited on page 64)

Marthinsen, Arne (Apr. 1999). "Interpolation in Lie Groups and Homogeneous Spaces". In.
: (cited on page 64)

Amaya-Mejía, Lina María et al. (2022). "Vision-Based Safety System for Barrierless Human-Robot Collaboration". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7331–7336.
(Cited on page 66)

Long, Philip et al. (2018). "An industrial security system for human-robot coexistence". In: *Industrial Robot: An International Journal* 45.2, pp. 220–226.
(Cited on page 66)

– (2017). "An industrial security system for human-robot coexistence". In: *Industrial Robot: An International Journal*.
(Cited on page 67)

Mainprice, Jim, Rafi Hayne, and Dmitry Berenson (2016). "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces". In: *Transactions on Robotics* 32.4.
(Cited on page 67)

Eckhoff, Moritz et al. (2022). "An MPC Framework For Planning Safe And Trustworthy Robot Motions". In: *International Conference on Robotics and Automation*.
(Cited on pages 67, 77, 79)

Sathya, Ajay Suresha et al. (2020). "Real-time Robot Arm Motion Planning and Control with Nonlinear Model Predictive Control using Augmented Lagrangian on a First-Order Solver". In: *European Control Conference*.
(Cited on page 68)

Nicolis, Davide, Fabio Allevi, and Paolo Rocco (Aug. 2020). "Operational Space Model Predictive Sliding Mode Control for Redundant Manipulators". In: *Transactions on Robotics* 36.4.
(Cited on page 68)

Incremona, Gian Paolo, Antonella Ferrara, and Lalo Magni (2017). "MPC for Robot Manipulators With Integral Sliding Modes Generation". In: *IEEE/ASME Transactions on Mechatronics* 22.3.
(Cited on page 68)

Berscheid, Lars and Torsten Kröger (2021b). "Jerk-limited Real-time Trajectory Generation with Arbitrary Target States". In: *Robotics: Science and Systems XVII*.
(Cited on pages 69, 70, 146)

Stellato, B. et al. (2020). "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4.
(Cited on pages 70, 130)

Ferreau, Joachim et al. (Dec. 2014). "qpOASES: A parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.
(Cited on pages 70, 95, 130)

Liu, Zhihao et al. (2020). "Dynamic risk assessment and active response strategy for industrial human-robot collaboration". In: *Computers & Industrial Engineering* 141, p. 106302.
(Cited on page 71)

Mayyas, Mohammad, Sai P Vadlamudi, and Muhammed A Syed (2020). "Fenceless obstacle avoidance method for efficient and safe human–robot collaboration in a shared work space". In: *International Journal of Advanced Robotic Systems* 17.5, p. 1729881420959018.
(Cited on page 71)

Mansfeld, Nico et al. (2017). "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
(Cited on pages 71, 99)

Makinson, B John (1971). *Research and development prototype for machine augmentation of human strength and endurance: Hardiman I project*. Tech. rep. Schenectady, NY, USA: General Electric Company.
(Cited on page 77)

Colgate, J Edward, Michael Peshkin, and Stephen H Klostermeyer (2003). "Intelligent assist devices in industrial applications: a review". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3.
(Cited on page 77)

Maurice, Pauline et al. (2017). "Human-oriented design of collaborative robots". In: *International Journal of Industrial Ergonomics* 57.
(Cited on page 77)

Schoose, C. et al. (2022). "Evolution of the biomechanical dimension of the professional gestures of grinders when using a collaborative robot". In: *International Journal of Occupational Safety and Ergonomics*.
(Cited on page 77)

Mainprice, Jim, Rafi Hayne, and Dmitry Berenson (2015). "Predicting human reaching motion in collaborative tasks using Inverse Optimal Control and iterative replanning". In: *IEEE International Conference on Robotics and Automation*.
(Cited on page 77)

Chen, Jessie Y.C. and Michael J. Barnes (2021). "Handbook Of Human Factors And Ergonomics". In: John Wiley & Sons, Ltd. Chap. Human-Robot Interaction.
(Cited on page 77)

Merckaert, Kelly et al. (2022). "Real-time motion control of robotic manipulators for safe human–robot coexistence". In: *Robotics and Computer-Integrated Manufacturing* 73.
(Cited on page 77)

Palleschi, Alessandro et al. (2021). "Fast and Safe Trajectory Planning: Solving the Cobot Performance/Safety Trade-Off in Human-Robot Shared Environments". In: *IEEE Robotics and Automation Letters* 6.3, pp. 5445–5452.
(Cited on pages 78, 116)

Coleman, David et al. (May 2014). "Reducing the barrier to entry of complex robotic software: a moveit! case study". In: *Journal of Software Engineering for Robotics* 5.1.
(Cited on page 78)

Pham, Hung and Quang Cuong Pham (June 2018). "A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis". In: *IEEE Transactions on Robotics* 34.
(Cited on page 78)

Ghazaei Ardakani, M. Mahdi et al. (2019). "Model Predictive Control for Real-Time Point-to-Point Trajectory Generation". In: *Transactions on Automation Science and Engineering* 16.2.
(Cited on page 78)

Meguenani, Anis (2017). "Safe control of robotic manipulators in dynamic contexts". PhD thesis. Université Pierre et Marie Curie-Paris VI.
(Cited on page 78)

Kouzoupis, D. et al. (2015). "Towards proper assessment of QP algorithms for embedded model predictive control". In: *2015 European Control Conference (ECC)*, pp. 2609–2616.
(Cited on page 78)

Kouzoupis, Dimitris et al. (2018). "Recent advances in quadratic programming algorithms for nonlinear model predictive control". In: *Vietnam Journal of Mathematics* 46.4, pp. 863–882.
(Cited on page 78)

Mastalli, Carlos et al. (2020). "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control". In: *IEEE International Conference on Robotics and Automation*.
(Cited on page 79)

Faroni, Marco, Manuel Beschi, and Nicola Pedrocchi (2019). "An MPC Framework for Online Motion Planning in Human-Robot Collaborative Tasks". In: *24th IEEE International Conference on Emerging Technologies and Factory Automation*.
(Cited on page 79)

Joseph, Lucas, Vincent Padois, and Guillaume Morel (2018b). "Towards X-ray medical imaging with robots in the open: safety without compromising performances". In: *IEEE International Conference on Robotics and Automation*.
(Cited on pages 82, 160, 163)

Ferreau, Joachim (Jan. 2011). "Model Predictive Control Algorithms for Applications with Millisecond Timescales". PhD thesis.
(Cited on page 88)

Barfoot, Tim D. and Paul Timothy Furgale (2014). "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems". In: *Transactions on Robotics* 30.
(Cited on pages 90, 124)

Skuric, Antun et al. (2022b). "On-line feasible wrench polytope evaluation based on human musculoskeletal models: an iterative convex hull method". In: *IEEE Robotics and Automation Letters*.
(Cited on page 99)

Camblor, Benjamin et al. (2022). "Task-Consistent Signaling Motions for Improved Understanding in Human-Robot Interaction and Workspace Sharing". In: *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.
(Cited on pages 99, 153)

Gouttefarde, Marc and Sébastien Krut (June 2010). "Characterization of Parallel Manipulator Available Wrench Set Facets". In: *Advances in Robots Kinematics: Motion in Man and Machine*. Piran-Portorož, Slovenia: Springer, pp. 475–482.
(Cited on page 106)

Fukuda, Komei et al. (2004). "Frequently asked questions in polyhedral computation". In: *ETH, Zurich, Switzerland* 85.
(Cited on page 106)

Skuric, Antun, Vincent Padois, and David Daney (Sept. 2022c). "Approximating robot reachable space using convex polytopes". In: *15th International Workshop on Human-Friendly Robotics*. Delft, Netherlands.
(Cited on page 107)

Skuric, Antun et al. (Oct. 2021b). "Common wrench capability evaluation of a human-robot collaborative system". In: *46ème Congrès Société Biomécanique*. Vol. 24. sup1. Saint Etienne, France: Taylor & Francis, S242–S245.
(Cited on pages 107, 152)

Franka Emika (n.d.). *Robot and interface specifications*. `https://frankaemika.github.io/docs/control_parameters.html`. [Online; accessed July-2023].
(Cited on page 107)

Gallant, Andre and Clément Gosselin (Mar. 2018). "Extending the capabilities of robotic manipulators using trajectory optimization". In: *Mechanism and Machine Theory* 121, pp. 502–514.
(Cited on page 107)

Bouchard, Samuel, Clément Gosselin, and Brian Moore (Dec. 2009). "On the Ability of a Cable-Driven Robot to Generate a Prescribed Set of Wrenches". In: *Journal of Mechanisms and Robotics* 2.1, p. 011010.
(Cited on page 115)

Koenig, Nathan and Andrew Howard (2004). "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE, pp. 2149–2154.
(Cited on page 129)

Quigley, Morgan et al. (2009). "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan, p. 5.
(Cited on page 130)

Bambade, Antoine et al. (June 2022). "PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond". In: *RSS 2022 - Robotics: Science and Systems*. New York, United States.
(Cited on page 130)

Goldfarb, Donald and Ashok Idnani (1983). "A numerically stable dual method for solving strictly convex quadratic programs". In: *Mathematical programming* 27.1, pp. 1–33.
(Cited on page 130)

Löw, Tobias and Sylvain Calinon (2023). "Geometric Algebra for Optimal Control With Applications in Manipulation Tasks". In: *IEEE Transactions on Robotics*, pp. 1–15.
(Cited on page 151)

Sanyal, Raman, Frank Sottile, and Bernd Sturmfels (June 2011). "ORBITOPES". In: *Mathematika* 57.2, pp. 275–314.
(Cited on page 151)

Tassa, Yuval, Nicolas Mansard, and Emo Todorov (May 2014). "Control-limited differential dynamic programming". In.
: (cited on page 152)

Kleff, Sébastien et al. (2022). "Introducing Force Feedback in Model Predictive Control". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13379–13385.
(Cited on page 152)

Bechtle, Sarah et al. (2021). "Leveraging Forward Model Prediction Error for Learning Control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4445–4451.
(Cited on page 152)

Fisac, Jaime F. et al. (2019). "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems". In: *IEEE Transactions on Automatic Control* 64.7, pp. 2737–2752.
(Cited on page 152)

Bogdanovic, Miroslav and Ludovic Righetti (2019). "Learning to Explore in Motion and Interaction Tasks". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE Press, pp. 2686–2692.
(Cited on page 152)

Calinon, Sylvain (2020). "Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control". In: *IEEE Robotics & Automation Magazine* 27.2, pp. 33–45.
(Cited on page 152)

Savin, Jonathan et al. (2016). "Movement Variability and Digital Human Models: Development of a Demonstrator Taking the Effects of Muscular Fatigue into Account". In: *Advances in Applied Digital Human Modeling and Simulation: Proceedings of the AHFE 2016 International Conference on Digital Human Modeling and Simulation*. Ed. by Vincent G. Duffy. Vol. 481. Advances in Intelligent Systems and Computing. Springer, pp. 169–179.
(Cited on page 152)

Benhabib, Nassim, Vincent Padois, and David Daney (May 2020). "Securing Industrial Operators with Collaborative Robots: Simulation and Experimental Validation for a Carpentry task". In: *ICRA 2020 - IEEE International Conference on Robotics and Automation*. IEEE ICRA Best Paper Award in Automation. Paris, France.
(Cited on page 153)

Gambao, Ernesto (2023). *Analysis exploring risks and opportunities linked to the use of collaborative industrial robots in Europe*. Tech. rep. Panel for the Future of Science, Technology (STOA), and managed by the Scientific Foresight Unit, within the Directorate-General for Parliamentary Research Services (EPRS) of the Secretariat of the European Parliament.
(Cited on page 153)