Impact aware robot manipulation via task-based reference spreading

Beumer, C.T.J.

Award date:
2019

Impact Aware Robot Manipulation via Task-Based Reference Spreading

Casper Beumer (0854071)

DC 2019.045

Master thesis

Project supervisor:	prof.dr. H. Nijmeijer
Daily supervisors:	dr.ir. A. Saccon dr.ir. A. Kheddar (CNRS-AIST JRL)
Committee:	prof.dr. H. Nijmeijer dr.ir. A. Saccon dr.ir. M.J.G. van de Molengraft dr.ir. A. Kheddar (CNRS-AIST JRL)

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control Group

Eindhoven, May 2019

Abstract

Robots that perform manipulation tasks usually slow down when they establish contacts with objects and their environment. In order to increase their performance, control strategies can be employed that take into account the effect of impacts, allowing for impact aware manipulation. A challenge in impact aware manipulation is that, in the presence of perturbations, a system can experience an impact at a different time than expected. This causes the perturbed system to reside in a different mode than described by the reference trajectory. In addition, when impacts are expected to occur simultaneously in multiple contact points, a perturbed system can enter modes that are not specified by the reference trajectory. Reference spreading provides a possible solution for these challenges, by extending reference trajectories beyond the intended impact times and using them to define a new tracking error. However, due to the state-feedback control approach that is used in reference spreading, this control strategy is not ideal for complex robots, like humanoid robots or dual arm manipulators. A common approach for the control of such robots is through task-based quadratic programming (QP) control. Therefore, this project poses the basis for extending reference spreading in the context of task-based robot control, in particular, by investigating how it could be merged with state-of-the-art task-based QP robot control.

In order to achieve this goal, an approach is proposed for the definition of ante- and post-impact reference trajectories that are compatible with the robot impact dynamics and suitable for task-based QP control. For this purpose, the ante-impact reference task trajectories have to implicitly define a unique joint state trajectory, such that unique impact dynamics are achieved. In this case, post-impact reference task trajectories can be specified that are compatible with these impact dynamics. Inspired by reference spreading, the reference task trajectories are extended beyond the intended jump times, in order to deal with perturbations. A procedure is proposed for the generation of such extended reference task trajectories for three specific QP tasks, selected for a dual arm dynamic box-lifting application. Finally, a task-based QP control framework is proposed, based on reference spreading, in order to achieve tracking of the extended reference task trajectories in the presence of perturbations. By means of a numerical simulation study it is shown how extended reference task trajectories can be generated for the dual arm box-lifting application, that are consistent with the impact dynamics.

Acknowledgements

I would like to take this opportunity to thank the people that have guided, helped, and supported me during my master graduation project and throughout the rest of my studies. I am very aware of the impact that these people have had on this work.

First of all, I would like to express my gratitude to my supervisors. Henk Nijmeijer, your passion for dynamics and control is inspiring and is one of the reasons that I got enthusiastic about this field as well. Thank you for your advice throughout my master, for helping me find a great internship in the US, and for your valuable feedback during my graduation project. Alessandro Saccon, your critical feedback and honest opinions have greatly improved the quality of my work. Your enthusiasm about your work and your interest in my project have inspired me and I have learned a lot from you during this project. Abder Kheddar, your vision and feedback has helped me get more insight in the subject of this project. Our collaboration allowed me to bring my theoretical knowledge to a real-life application, which I really enjoyed. Thanks also to Pierre Gergondet, for helping me understand and use the software framework. I appreciate that you were always willing to answer my questions. Mark Rijnen, it was nice to be able to ask you questions and have discussions when things were unclear. Even though you had your own thesis to worry about, you were never too busy to help me.

Then, I would like to thank my fellow students in the robotics lab for the support, lunch discussions, and lab activities. Being in the robotics lab has made my graduation project much more enjoyable. In particular, I would like to thank Menno, for introducing me to reference spreading and helping me conquer it.

I also want to thank my friends and family, for their support and welcome distraction from this graduation project. Special thanks to Marcel and Emil for teaming up with me throughout our studies and for making my student time unforgettable.

Finally, I want to thank my parents, brother and girlfriend for their support, encouragement and love. To my parents, without your support and motivation I could not have completed and enjoyed my studies in the way that I did. Thomas, you have helped me a lot with my studies, making choices, and putting things into perspective. Maaike, you have supported and motivated me every day during my graduation project and studies, for which I am very grateful. You make me very happy and I am looking forward to the fun times that are coming.

Casper Beumer
Eindhoven, April 2019

Contents

Nomenclature

Greek symbols

$\boldsymbol{\alpha}$	Reference state trajectory
γ	Guard function
Δt	Time step
δ	Time extension
θ	Angle
$\boldsymbol{\lambda}$	Vector of linearized friction cone base weights
λ	De Casteljau parameter
$\boldsymbol{\mu}$	Reference input
$\boldsymbol{\Omega}$	Skew-symmetric angular velocity matrix
${}^{C}\boldsymbol{\Omega}_{A,B}$	Skew-symmetric angular velocity matrix of B with respect to A, written in C
$\boldsymbol{\omega}$	Angular velocity
${}^{C}\boldsymbol{\omega}_{A,B}$	Angular velocity of B with respect to A, written in C
$\boldsymbol{\tau}$	Joint torque vector
τ	Intended impact time
Φ	Stacked contact force selection matrix
ϕ	Contact force selection matrix
$\boldsymbol{\chi}$	Vector of optimization variables

Roman symbols

$\boldsymbol{A}$	Constraint matrix
$A, B, \ldots$	Coordinate frames
$[A]$	Orientation frame associated with A
$\boldsymbol{B}$	Bézier curve
$\boldsymbol{b}$	Constraint vector

$\mathbf{C}$	Coriolis matrix
C	Control point
$\boldsymbol{c}$	Vector of constants
d	Polynomial degree
$\boldsymbol{E}$	Task error function
$\boldsymbol{e}$	Error
$\boldsymbol{F}$	Stacked contact force vector
$\mathbf{f}$	Contact force vector
$\mathbf{G}$	Potential force vector
$\boldsymbol{g}$	Jump map
$\boldsymbol{h}$	Forward kinematics
$\mathbf{J}$	Jacobian
${}^{C}\mathbf{J}_{A,B}$	Jacobian relating the velocity of frame B with respect to frame A, expressed in C
$\boldsymbol{K}$	Control gain matrix
$\boldsymbol{K}_d$	Derivative gain matrix
$\boldsymbol{K}_p$	Proportional gain matrix
k_d	Scalar derivative gain
k_p	Scalar proportional gain
$\mathbf{M}$	Mass matrix
$\mathbf{o}_B$	Origin of frame B
${}^{A}\mathbf{o}_B$	Coordinates of the origin of frame B, expressed in A
P	Polynomial
$\mathbf{p}$	Arbitrary point
${}^{A}\mathbf{p}$	Coordinates of $\mathbf{p}$, expressed in A
$\boldsymbol{Q}$	Hessian matrix
$\mathbf{q}$	Joint positions
$\mathbf{R}$	Rotation matrix
${}^{A}\mathbf{R}_B$	Rotation matrix from B to A
$\vec{r}$	Geometric vector
$\mathbf{S}$	Joint selection matrix
t	Time
$\boldsymbol{u}$	Input
$\mathbf{u}$	Direction vector

${}^{C}\boldsymbol{v}_{A,B}$	Linear velocity of B with respect to A, written in C
${}^{C}\mathbf{v}_{A,B}$	Twist expressing the velocity of B with respect to A, written in C
w	Task weight
$\boldsymbol{x}$	Joint state
$\boldsymbol{y}$	Reference task trajectory
$\boldsymbol{y}_{\epsilon}$	Perturbed task trajectory
$\mathbf{y}$	Task configuration

Sub- and superscripts

$\dot{(\cdot)}$	First time derivative
$\ddot{(\cdot)}$	Second time derivative
$\bar{(\cdot)}$	Extended version
${}^{a}(\cdot)$	Ante-impact
${}^{p}(\cdot)$	Post-impact
$(\cdot)^{\text{ref}}$	Reference quantity
$(\cdot)_0$	Quantity at initial time
$(\cdot)_\tau$	Quantity at intended impact time
$(\cdot)_f$	Quantity at final time
$(\cdot)_{\text{pos}}$	Quantity related to the position task
$(\cdot)_{\text{ori}}$	Quantity related to the orientation task
$(\cdot)_{\text{vec}}$	Quantity related to the direction task

Chapter 1

Introduction

1.1 Impact aware robot manipulation

Physical interaction with the environment is essential for manipulation tasks performed by robotic systems. As an example, in Figure 1.1a, the humanoid robot HRP-5P is shown lifting a drywall. For this task, the robot has to establish contacts with the drywall, in order to lift it. Furthermore, in assembly applications, robotic manipulators, like the KUKA LBR iiwa depicted in Figure 1.1b, make and break contact with objects repeatedly.

(a) HRP-5P by AIST [1] *(b) KUKA LBR iiwa [2]*

Figure 1.1: *Two examples of robots experiencing contact transitions during manipulation tasks.*

A common approach for handling the transition from free motion to contact motion is through the introduction of a transition phase [3]. During this transition phase, the velocity is modulated, such that the contact is established at almost zero velocity. However, for industrial applications in which throughput is of importance, such transition phases are not the ideal solution, as the robot has to slow down and speed up for every contact transition, which severely limits the performance. Motivated by this limitation, control strategies have been developed that are capable of dealing with dynamic contact transitions, in which contacts are made at nonzero velocity. Such control strategies take into

account the effect of the impacts that occur when contacts are made at nonzero velocity and are therefore referred to as impact aware. Impacts are characterized by short time intervals with high accelerations and decelerations. Due to the short duration of impacts, the associated velocity change is often assumed to be instantaneous, resulting in discontinuities in the velocity. The nonsmooth mechanics framework provides a modeling approach for systems with discontinuities in the velocity caused by impacts [4]. Well-established control strategies for smooth systems cannot be directly applied to nonsmooth mechanical systems. Therefore, the analysis and control of such nonsmooth systems has been an active area in research, recently.
In Section 1.2, existing approaches for the control of nonsmooth mechanical systems that are used for impact aware manipulation are detailed and classified on the basis of prespecified criteria. Based on this classification and the limitations of existing control approaches, the goal of this research is formulated in Section 1.3.

1.2 Existing control approaches for impact aware robot manipulation

Various control approaches exist for performing motion tasks involving intentional impacts using robotic systems. In this section, the existing approaches are reviewed. First, six criteria are specified, which are used to classify and compare the existing approaches. These criteria are detailed below.

Guaranteed convergence
First of all, it is important that a control approach is able to guarantee convergence to the reference signal after impact. When impacts occur, large impact forces are exerted on the robot, which may result in instability of the system. For example, a humanoid robot can easily tip over as a result of an impact with its environment. Due to the short duration of impacts, which is typically between 4 and 10 ms for a hard impact between a manipulator robot and a human head at 2 m/s [5], feedback control cannot be used to counteract the impact forces. Therefore, the control of systems experiencing impacts is complicated. Thereby, due to the nonsmooth nature of systems experiencing impacts, stability tools for smooth systems cannot be used to guarantee stability.

Noncoinciding jump times
In order for robots to perform motion tasks involving impacts in an uncertain environment, a control approach should be robust to perturbations of the initial conditions. These perturbations can cause impacts to occur at a different time than expected. An example of such a perturbation is the position of an object being different from the expected position. When a robot tries to grab the object, the impact and the associated state jump occur earlier or later than expected. This mismatch in jump times complicates the control of nonsmooth systems, as this causes the reference trajectory and the system to reside in different modes. For this purpose, the ability of dealing with noncoinciding jump times is considered for the control approaches.

Simultaneous impacts
In many manipulation tasks, robots are expected to make simultaneous impacts. Simultaneous impacts are impacts that occur at multiple contact points at the same time. An example of a manipulation task involving simultaneous impacts is a humanoid robot lifting a box through surface contacts between its hands and the box. In this case, the hands are expected to make impact with the box simultaneously in all contact points on both hands. Slight perturbations of the hands or the box cause loss of the simultaneity of the impacts, resulting in multiple impacts instead of a single simultaneous impact.

Experiencing more impacts than expected complicates the stabilization of the robot after impact. Therefore, it is investigated how the control approaches deal with simultaneous impacts.

Nonperiodic trajectories
Ideally, a control approach can be used for a large variety of motion tasks. Often, especially in the field of robotic locomotion, only periodic trajectories are considered, which severely limits the amount of different tasks the robot is able to perform. The fourth criterion that is considered for the control approaches is therefore the applicability to tasks involving nonperiodic trajectories.

Complex robots
Robots that are used for motion tasks with contact transitions can be very complex. For example, humanoid robots, like HRP-5P, which are very complex due to their floating base and many joints, are used for lifting tasks involving contact transitions, as shown in Figure 1.1a. Therefore, it is explored if the control approaches have been demonstrated on complex robots, as opposed to academic examples. In this research, all multibody robots with at least three joints are considered to be complex.

Output control
In many cases, especially for complex robots, it is desired to apply feedback control on a specific output of a robot, rather than its full state. For example, in order to lift a box using a dual arm manipulator, the manipulator should move its end effectors towards the box. In this case, it is much easier and intuitive to perform feedback control on the position of the end effectors, rather than the full state of the manipulator. For this purpose, the applicability of the control approaches to output feedback control is investigated, which is often referred to as task-based control in the field of robotics.

In the following subsections, five existing control approaches for motion tasks involving intentional impacts are reviewed and classified using the specified criteria. Based on the limitations of the existing control approaches, a new control approach is proposed in Section 1.2.6, which will be explored in this research. Table 1.1 shows an overview of the existing and proposed control approaches, and the criteria on which they are classified.

Table 1.1: *An overview of the classification of the existing and proposed control approaches for motion tasks involving impacts, based on six criteria.*

Control approach	Guaranteed convergence	Noncoinc. jump times	Simult. impacts	Nonperiodic trajectories	Complex robots	Output control
Hybrid zero dynamics [6–9]	✓	✓			✓	✓
Transition phase [10–12]	✓	✓	✓	✓		
Distance functions [13–15]	✓	✓		✓		
Reference spreading [16–19]	✓	✓	✓	✓	✓	
Task-based QP control [20, 21]	✓			✓	✓	✓
Task-based reference spreading	(✓)	✓	✓	✓	✓	✓

1.2.1 Hybrid zero dynamics

Using hybrid zero dynamics, an output feedback controller can be designed for dynamic systems performing periodic motion tasks with impacts, based on Poincaré's stability analysis, as detailed in [6]. For a given hybrid closed loop system, which consists of a robot, its environment and the feedback controller, it can be determined if asymptotically stable periodic orbits exist. Thereafter, the controller can be optimized in order to achieve the desired performance. Using this approach, asymptotic stability can be guaranteed, which means that convergence of the system to the reference trajectory after impact is guaranteed.
The best known application of systems that show periodic behavior and experience impacts is bipedal robot locomotion, in which the walking gait of the robot can be treated as a periodic trajectory, but juggling and hammering tasks can be considered as periodic as well. In [6], the hybrid zero dynamics are used to achieve asymptotically stable walking for bipedal robots. In [7], method has been experimentally shown to be applicable to complex bipedal robots as well. Also in [8] the hybrid zero dynamics are used, in combination with human data, in order to find a controller that results in stable robotic walking. This is validated using experiments on both an underactuated and a fully actuated bipedal robot.
The considered approaches do not explicitly mention simultaneous impacts. However, in some cases, simultaneous impacts are avoided by prescribing a sequence for the impacts, as is done for the walking gait in [9]. Each step consists of subsequently heel impact, toe impact, heel release and toe release.
The main limitation of the hybrid zero dynamics approach is that it is only applicable for periodic trajectories, which makes it unsuitable for many manipulation tasks.

1.2.2 Transition phase

In [10], an approach is proposed for tracking control of nonsmooth complementarity systems in order to perform motion tasks with both free and constraint motion phases. In order to achieve a stable transition from free to constraint motion involving impacts, a transition phase is introduced. During the transition phase, a specific controller is used, which takes into account Zeno behavior, such that a stable contact is established. This approach is applicable for nonperiodic trajectories. Also, in [11], a specific controller is introduced in order to ensure a stable transition between free and constraint motion phases. The transitions phases are robust to uncertainties in the position of the contact surface, which can cause impacts to occur earlier or later than expected. Therefore, the transition phases are capable of dealing with noncoinciding jump times. In [12], it is detailed how the transition phase proposed in [11] can be utilized to deal with uncertainties in orientation of the contact surface, which can cause a misalignment of the contact surfaces on a robot and the object it is making contact with. In this way, this approach can be used to perform motion tasks in which simultaneous impacts are expected to occur.
The considered control approaches have been validated for simple two-link manipulators, but have not been applied to more complex robots. Thereby, they have not been used in combination with output feedback control.

1.2.3 Distance functions

Another common approach for impact aware manipulation is through tracking control for hybrid systems. In order to deal with noncoinciding jump times, several approaches, based on hybrid trajectory tracking, have been proposed that introduce distance functions. Distance functions provide a novel notion of the error between two trajectories, which is unaffected by noincoinciding jump times. Using Lyapunov-based conditions, global asymptotic stability can be expressed in terms of these distance functions. This is achieved in [13], using a distance function that explicitly takes into account the hybrid nature of the system. Various other distance functions have been used to deal with noncoinciding jump times as well. In [14], an error notion induced by the quotient metric is used. When a system experiences an impact, the jump map is applied to the reference trajectory, such that the state of the system can be compared to a reference trajectory in the same mode. In [15], the minimum of the error between the current state and the reference trajectory before and after the impact is used as a distance function.
The considered approaches using distance functions are very recent. So far, none of these have been shown to be applicable to trajectories with simultaneous impacts. Also, only academic examples have been used to show the efficacy of the approaches. None of the approaches have been demonstrated on complex robots or have been used for output control.

1.2.4 Reference spreading

Similar to the approaches discussed in the previous subsection, reference spreading uses a novel error notion to deal with noncoinciding jump times, as seen in [16]. This error notion is based on extension of the ante- and post-impact trajectories, such that the state of the system can always be compared to an extended reference trajectory in the same mode. This error notion is used in a sensitivity analysis, which constructs a first order approximation of a nearby perturbed trajectory, resulting in a time-triggered linear system. In [17], it is shown that uniform asymptotic stability of the linearized system guarantees asymptotic stability of the associated state-triggered system.
Reference spreading is also applicable to trajectories with simultaneous impacts, as shown in [18]. Since, in this case, perturbations can cause the system to enter unspecified modes, the system has a multi-valued jump gain, which makes it impossible to linearize the system. Instead of the time-triggered linearization of [16], the positive homogenization is used to approximate the state-triggered trajectories. Reference spreading is applicable to hybrid systems with discontinuous, time-varying state trajectories and therefore applicable for nonperiodic trajectories. This control strategy has also shown to be applicable for complex robots. In [19], reference spreading is used for the stabilization of dynamic motion tasks involving impacts for a iCub humanoid robot. For this purpose, reference spreading makes use of a state-feedback control approach. So far, reference spreading has not been used in combination with output control.

1.2.5 Task-based QP control

Motion tasks involving impacts can also be performed using a task-based quadratic programming (QP) control approach. In task-based QP control, tasks can be specified for specific variables, like the position of an end effector or the center of mass of a humanoid robot. Therefore, task-based QP control allows for output feedback control. In [20], a task-based QP control approach is used to perform a Karate-chop

motion with a humanoid robot, utilizing impacts to break wooden planks. After impact, the stabilizing controller proposed in [21] is used in order for the humanoid robot to remain balanced. This task-based QP control approach has been validated using experiments. However, perturbations and simultaneous impacts have not been considered.

1.2.6 Task-based reference spreading

As seen in Table 1.1, several control approaches exist that guarantee convergence of a system to a nonperiodic reference signal after impacts have occurred and are capable of dealing with noncoinciding jump times. From these approaches, reference spreading is the only approach that both has been shown to be applicable to motion tasks involving simultaneous impacts and has been demonstrated on a complex robot. However, due to the state-feedback control approach that is used in reference spreading, it is not ideal for application on complex robotic systems. Task-based QP control is much better applicable to such systems, due to its output feedback control approach. However, existing approaches using task-based QP control have not been shown to be robust to perturbations causing noncoinciding jump times or a mismatch in the number of jumps when simultaneous impacts are expected to occur. In order to fill the gap, this research aims to pose the basis for a stable, robust control approach for impact aware manipulation, that is suitable for complex robots, by combining reference spreading with task-based QP control. As seen in Table 1.1, such a control approach, which is referred to as task-based reference spreading, meets all the specified criteria. This research does not provide a proof of guaranteed convergence after impact, which is why the check mark for this criterion is between parentheses.

1.3 Research goal

As detailed Section 1.2.6, this research aims to develop a control approach for impact aware manipulation, based on reference spreading in combination with task-based QP control. For this purpose, the current reference spreading control theory will be extended, such that it can be used for task-based QP control. Therefore, the goal of this research is formulated as:

Extending reference spreading in order to be applicable for task-based QP control.

Towards achieving this goal, four contributions will be made in this research.

- Defining reference trajectories for motion tasks with intentional impacts that are compatible with the impact dynamics and suitable for task-based QP control.
- Developing a trajectory generation and extension procedure for the reference trajectories associated with specific QP control tasks.
- Defining a task-based QP control framework, based on reference spreading, with which motion tasks involving simultaneous impacts can be performed, in the presence of perturbations.
- Demonstrating the extended reference trajectory generation procedure by means of a numerical simulation example.

1.4 Report outline

This report is structured as follows. Chapter 2 provides an overview of the essential background information for this research. Among other things, reference spreading and task-based QP control are reviewed in this chapter. Chapter 3 elaborates on the extension of reference spreading to task-based QP control. This chapter details the definition and generation of the task-based reference trajectories, as well as the QP control framework that allows to deal with perturbations. In Chapter 4, a numerical simulation study is performed, in which it is shown how task-based reference trajectories can be generated for a dual arm box-lifting application. Finally, conclusions are drawn on this research and recommendations are formulated for future research in Chapter 5.

Chapter 2

Preliminaries

This chapter provides an overview of the preliminary material that is relevant for this research. First of all, a notation is presented, that is used to describe the dynamics of multibody robotic systems. Thereafter, an overview is given of the essential background information about reference spreading. Then, task-based QP robot control is explained and the specific QP control implementation that is utilized in this work is presented. Furthermore, the dynamical system simulators that are used for the numerical simulation study are reviewed. Finally, an interpolation algorithm is detailed that can be used for trajectory generation.

2.1 Multibody dynamics notation

In this section, the multibody dynamics notation proposed in [22] is reviewed, which is used throughout this work. First, coordinate frames and points are introduced. Thereafter, expressions are formulated to describe the velocity and acceleration of these coordinate frames. Finally, joint states and Jacobians are considered, which are used to describe the body frames and the associated velocities and accelerations of fixed manipulators.

2.1.1 Coordinate frames and points

A coordinate frame is defined by the combination of a point and an orientation frame in the 3D space. Coordinate frames are indicated by a capital letter. For a given coordinate frame B, the origin is indicated by $\mathbf{o}_B$ and the orientation frame by $[B]$, such that $B = (\mathbf{o}_B, [B])$. In Figure 2.1, two coordinate frames W and B are depicted. In this work, W generally denotes a fixed inertial frame and B a moving body frame. Figure 2.1 also shows a point $\mathbf{p}$. The coordinates of this point with respect to the inertial frame W are given by the coordinate vector ${}^W\mathbf{p}$. Mathematically, the coordinate vector is given by

$$ {}^W\mathbf{p} := \begin{bmatrix} \vec{r}_{\mathbf{o}_W,\mathbf{p}} \cdot \vec{x}_W \\ \vec{r}_{\mathbf{o}_W,\mathbf{p}} \cdot \vec{y}_W \\ \vec{r}_{\mathbf{o}_W,\mathbf{p}} \cdot \vec{z}_W \end{bmatrix} \in \mathbb{R}^3, \tag{2.1} $$

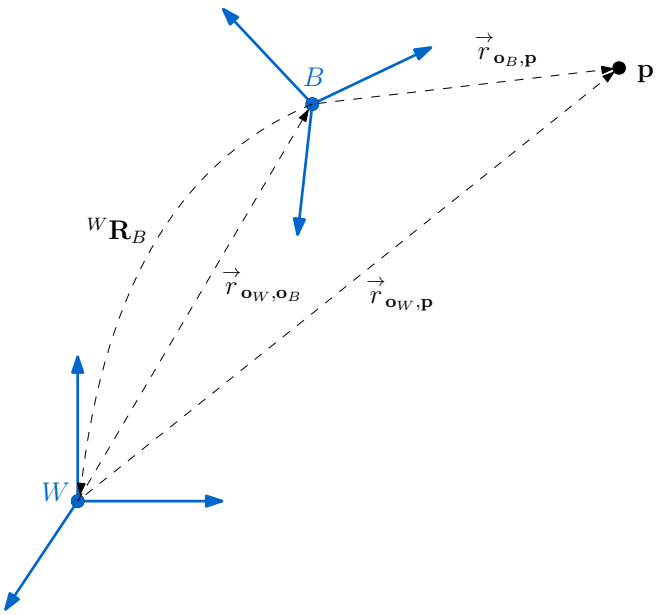


Figure 2.1: *Visual representation of an inertial frame W, a body frame B and a point $\mathbf{p}$. The dashed arrows represent the transformations between the point and the coordinate frames.*

where $\cdot$ denotes the scalar product between two vectors, $\vec{r}_{\mathbf{o}_W,\mathbf{p}}$ denotes the geometric vector which connects the origin of frame W with point $\mathbf{p}$, and $\vec{x}_W$, $\vec{y}_W$, $\vec{z}_W$, denote the unit vectors defining the orientation of the inertial frame $[W]$. The coordinates of the origin of frame B with respect to frame W, denoted by ${}^W\mathbf{o}_B$, can also be described by (2.1) for $\mathbf{p} = \mathbf{o}_B$. The orientation of frame B expressed with respect to frame W is given by ${}^W\mathbf{R}_B \in \mathrm{SO}(3)$. This coordinate transformation ${}^W\mathbf{R}_B$ only describes the relative orientation between the orientation frames $[W]$ and $[B]$ and is independent on the positions of $\mathbf{o}_W$ and $\mathbf{o}_B$.

2.1.2 Velocity vectors

The velocity of coordinate frame B can be expressed in multiple ways. The velocity of frame B with respect to frame W expressed in frame B, referred to as the left trivialized velocity, is given by

$$ {}^B\mathbf{v}_{W,B} := \begin{bmatrix} {}^B\boldsymbol{v}_{W,B} \\ {}^B\boldsymbol{\omega}_{W,B} \end{bmatrix} \in \mathbb{R}^6, \tag{2.2} $$

where

$$ {}^B\boldsymbol{v}_{W,B} := {}^W\mathbf{R}_B^T\, {}^W\dot{\mathbf{o}}_B, \tag{2.3} $$

$$ {}^B\boldsymbol{\omega}^{\wedge}_{W,B} := {}^W\mathbf{R}_B^T\, {}^W\dot{\mathbf{R}}_B, \tag{2.4} $$

with ${}^B\boldsymbol{v}_{W,B}$ and ${}^B\boldsymbol{\omega}_{W,B} \in \mathbb{R}^3$ and ${}^W\dot{\mathbf{o}}_B$ and ${}^W\dot{\mathbf{R}}_B$ the time derivatives of ${}^W\mathbf{o}_B$ and ${}^W\mathbf{R}_B$, respectively. The hat operator, denoted by $^\wedge$, transforms a vector in $\mathbb{R}^3$ to a skew-symmetric matrix in $\mathfrak{so}(3)$, such

that

$$w^{\wedge} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}^{\wedge} := \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{2.5}$$

The vee operator, denoted by $^{\vee}$, performs the inverse transformation, from a skew-symmetric matrix in $\mathfrak{so}(3)$ to a vector in $\mathbb{R}^3$, such that

$$W^{\vee} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}^{\vee} := \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3. \tag{2.6}$$

For the angular velocity holds that $\boldsymbol{\omega}^{\wedge} = \boldsymbol{\Omega} \in \mathfrak{so}(3)$.
The acceleration vector, associated with the left trivialized velocity, is given by

$${}^{B}\dot{\mathbf{v}}_{W,B} := \begin{bmatrix} {}^{B}\dot{\boldsymbol{v}}_{W,B} \\ {}^{B}\dot{\boldsymbol{\omega}}_{W,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.7}$$

The right trivialized velocity is used to describe the velocity of frame B with respect to frame W, expressed in frame W. The right trivialized velocity is given by

$${}^{W}\mathbf{v}_{W,B} := \begin{bmatrix} {}^{W}\boldsymbol{v}_{W,B} \\ {}^{W}\boldsymbol{\omega}_{W,B} \end{bmatrix} \in \mathbb{R}^6, \tag{2.8}$$

where

$${}^{W}\boldsymbol{v}_{W,B} := {}^{W}\dot{\mathbf{o}}_{B} - {}^{W}\dot{\mathbf{R}}_{B}\,{}^{W}\mathbf{R}_{B}^{T}\,{}^{W}\mathbf{o}_{B} \tag{2.9}$$

$${}^{W}\boldsymbol{\omega}_{W,B}^{\wedge} := {}^{W}\dot{\mathbf{R}}_{B}\,{}^{W}\mathbf{R}_{B}^{T} \tag{2.10}$$

with ${}^{W}\boldsymbol{v}_{W,B}$ and ${}^{W}\boldsymbol{\omega}_{W,B} \in \mathbb{R}^3$. The associated acceleration vector writes

$${}^{W}\dot{\mathbf{v}}_{W,B} := \begin{bmatrix} {}^{W}\dot{\boldsymbol{v}}_{W,B} \\ {}^{W}\dot{\boldsymbol{\omega}}_{W,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.11}$$

Finally, one can describe the velocity of frame B with respect to frame W with respect to a frame with the origin of frame B and the orientation of frame W, which is denoted by $B[W] := (\mathbf{o}_B, [W])$. This velocity, called the mixed velocity, is given by

$${}^{B[W]}\mathbf{v}_{W,B} := \begin{bmatrix} {}^{W}\dot{\mathbf{o}}_{B} \\ {}^{W}\boldsymbol{\omega}_{W,B} \end{bmatrix} \in \mathbb{R}^6, \tag{2.12}$$

where ${}^{W}\boldsymbol{\omega}_{W,B}$ is found by (2.10). The associated acceleration vector is given by

$${}^{B[W]}\mathbf{v}_{W,B} := \begin{bmatrix} {}^{W}\ddot{\mathbf{o}}_{B} \\ {}^{W}\dot{\boldsymbol{\omega}}_{W,B} \end{bmatrix} \in \mathbb{R}^6. \tag{2.13}$$

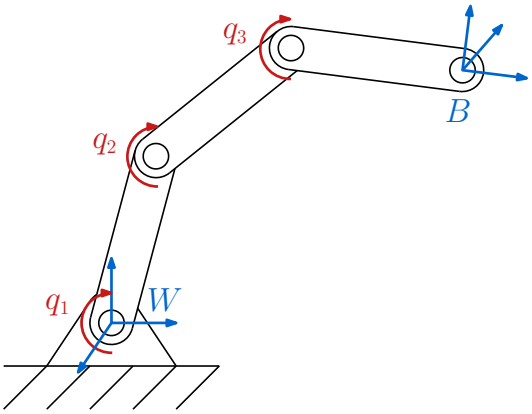


Figure 2.2: *A three-link manipulator with inertial frame W and end effector body frame B. The joint positions and velocities are denoted by q_i and $\dot{q}_i$, respectively, with $i \in \{1, 2, 3\}$.*

2.1.3 Joint states and Jacobians

For multibody robotic systems, the coordinate frames and the associated velocities and accelerations of the bodies can be expressed in terms of their joint states and Jacobians. In this work, only fixed base robot manipulators are considered, which are manipulators with a base that is rigidly attached to the world, as opposed to floating-base robotic systems, like humanoid or wheeled robots, which have a moving base. Consider such a fixed manipulator depicted in Figure 2.2. Here frame W denotes the fixed inertial frame, positioned in the base of the manipulator and B the moving body frame attached to the end effector. The coordinate frame B can be expressed in terms of the joint positions of the manipulator, given by $\mathbf{q} \in \mathbb{R}^{n_J}$, where n_J denotes the number of joints in the manipulator, such that

$$B = \boldsymbol{h}(\mathbf{q}), \tag{2.14}$$

where $\boldsymbol{h}$ denotes the forward kinematics relating the joint positions to the coordinate frame B. These forward kinematics are found using homogeneous matrix compositions with respect to a parametrization convention, such as the Denavit-Hartenberg convention [23].
The velocity of frame B can be expressed with respect to the inertial frame W, using the various expressions for the velocity detailed in the previous subsection. These velocities can also be expressed in terms of the joint velocities of the manipulator, denoted by $\dot{\mathbf{q}} \in \mathbb{R}^{n_J}$, using the Jacobian. The left trivialized velocity is computed by

$${}^{B}\mathbf{v}_{W,B} = {}^{B}\mathbf{J}_{W,B}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.15}$$

where ${}^{B}\mathbf{J}_{W,B}(\mathbf{q})$ denotes the Jacobian relating the velocity of B with respect to W expressed in B, which is dependent on the joint positions $\mathbf{q}$. The associated acceleration is found by

$${}^{B}\dot{\mathbf{v}}_{W,B} = {}^{B}\mathbf{J}_{W,B}(\mathbf{q})\ddot{\mathbf{q}} + {}^{B}\dot{\mathbf{J}}_{W,B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{2.16}$$

The right trivialized velocity ${}^{W}\mathbf{v}_{W,B}$ and mixed velocity ${}^{B[W]}\mathbf{v}_{W,B}$, and the associated accelerations can be computed by replacing ${}^{B}\mathbf{J}_{W,B}(\mathbf{q})$ in (2.15) and (2.16) by ${}^{W}\mathbf{J}_{W,B}(\mathbf{q})$ and ${}^{B[W]}\mathbf{J}_{W,B}(\mathbf{q})$, respectively.

2.2 Classical reference spreading tracking control

This section reviews the state-feedback control strategy for trajectory tracking with impacts, known as reference spreading [24]. In this work, this state-feedback control strategy is referred to as classical reference spreading. First, the challenge in trajectory tracking with impacts is highlighted. Thereafter, reference spreading for single impacts is reviewed, as proposed in [16]. Then, the complementary issue that occurs when simultaneous impacts are expected to occur is detailed, followed by the solution given by reference spreading for simultaneous impacts, as proposed in [18].

2.2.1 Trajectory tracking with impacts

For the particular class of tasks that are considered in this work, a mechanical system experiencing impacts can be represented as a hybrid system. A hybrid system consists of both continuous flows and discontinuous evolutions [25]. During the continuous flow modes, the state of the system is described by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \tag{2.17}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes the state, with n the state dimension, which can be different for each mode, $\boldsymbol{u} \in \mathbb{R}^m$ denotes the input, with m the input dimension and $t \in [t_0, t_f]$ the time. The system remains in the continuous mode, as long as a certain constraint is satisfied, given by $\gamma(\boldsymbol{x}, t) \geq 0$, where $\gamma : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ denotes the guard function. This guard function could, for example, describe the distance between two bodies. The guard is activated when $\gamma(\boldsymbol{x}, t) = 0$, which, in this case, corresponds to contact between the two bodies. When the system experiences an impact, due to a contact at nonzero velocity, an impulsive force is applied to the system, which results in a jump in the state $\boldsymbol{x}$. This jump is described by

$$\boldsymbol{x}^+ = \boldsymbol{g}(\boldsymbol{x}^-, t), \tag{2.18}$$

where $\boldsymbol{g} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n,\ (\boldsymbol{x}, t) \mapsto \boldsymbol{g}(\boldsymbol{x}^-, t)$ is the jump map and where $\boldsymbol{x}^-$ and $\boldsymbol{x}^+$ denote the left and right limit of the state at the time of the impact.
State-feedback control can be used in order to control such a hybrid system to a time-varying reference state trajectory $\boldsymbol{\alpha}(t) \in \mathbb{R}^n$. This reference trajectory satisfies (2.17), except at the reference event time τ, where the reference trajectory jumps according to (2.18). Local tracking of the reference trajectory can be achieved using the state-feedback control law

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{\mu}(t) - \boldsymbol{K}(t)(\boldsymbol{x} - \boldsymbol{\alpha}(t)), \tag{2.19}$$

where $\boldsymbol{\mu}(t) \in \mathbb{R}^m$ denotes the reference feedforward input and $\boldsymbol{K}(t) \in \mathbb{R}^{m \times n}$ the feedback gain. However, using this control law, an issue occurs when perturbations cause the state $\boldsymbol{x}$ to jump at a different time than $\boldsymbol{\alpha}(t)$. This is illustrated in Figure 2.3. The left plot shows state trajectory $\boldsymbol{x}$ and the reference state trajectory $\boldsymbol{\alpha}(t)$. The right plot shows the error $\boldsymbol{e}(t) = \|\boldsymbol{x} - \boldsymbol{\alpha}(t)\|$. Due to perturbations, the state jumps at $t = t_1$, where $t_1 \neq \tau$. The mismatch in jump times causes the perturbed state trajectory and the reference trajectory to reside in different modes on the time interval $t \in [t_1, \tau]$. On this time interval, the error $\boldsymbol{e}(t) = \|\boldsymbol{x} - \boldsymbol{\alpha}(t)\|$ becomes large. This phenomenon, which is called peaking, can result in poor tracking performance or even destabilization of the system, as the input computed by the control law (2.19) suddenly becomes large when the error peaks.

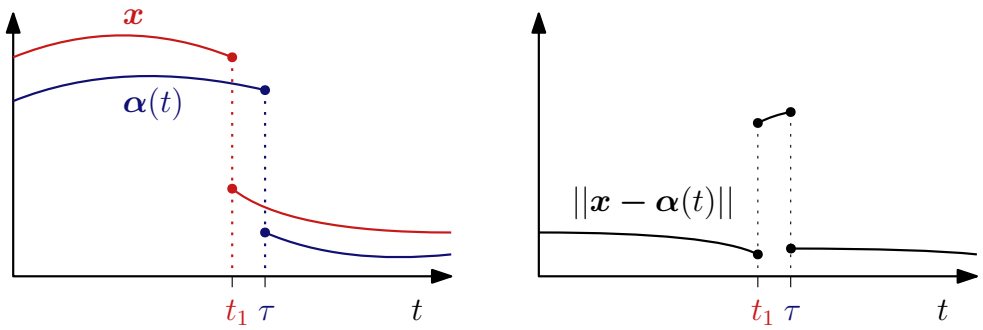


Figure 2.3: *An illustrative example of peaking in the error as a result of the mismatch in jump times between a perturbed state trajectory* $\boldsymbol{x}$ *and the reference state trajectory* $\boldsymbol{\alpha}(t)$*. The left figure shows* $\boldsymbol{x}$ *and* $\boldsymbol{\alpha}(t)$*, which jump at respectively* $t = t_1$ *and* $t = \tau$*. The right figure shows the error* $\boldsymbol{e}(t) = \|\boldsymbol{x} - \boldsymbol{\alpha}(t)\|$*, which peaks at the interval* $t \in [t_1, \tau]$*.*

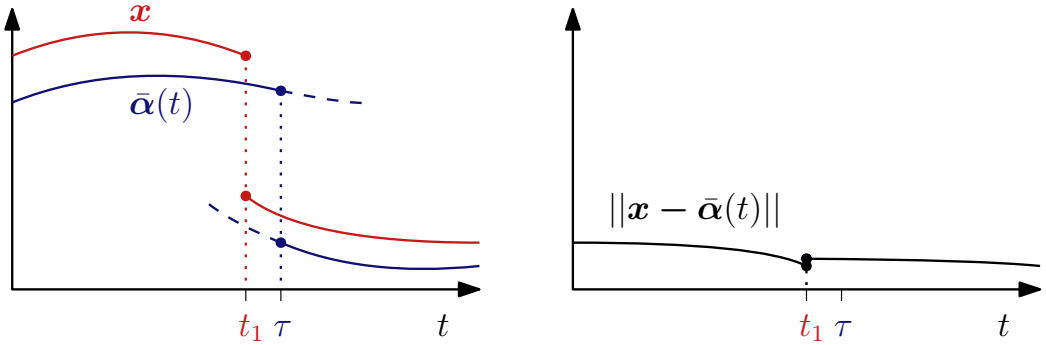


Figure 2.4: *Visual illustration of reference spreading applied to the example of Figure 2.3. The perturbed state trajectory* $\boldsymbol{x}$ *is compared to the extended reference state trajectory* $\bar{\boldsymbol{\alpha}}(t)$*, such that the error is defined as* $\boldsymbol{e}(t) = \|\boldsymbol{x} - \bar{\boldsymbol{\alpha}}(t)\|$*. This error, shown in the right figure, shows a single, small jump, at* $t = t_1$*, instead of a peak on the interval* $t \in [t_1, \tau]$*.*

2.2.2 Reference spreading for single impacts

Reference spreading, as introduced in [16], proposes a solution for the peaking in the error by extending the reference trajectory $\boldsymbol{\alpha}(t)$ and reference feedforward input $\boldsymbol{\mu}(t)$ beyond impact time τ and using them to define a new tracking error. The extended reference trajectory, denoted by $\bar{\boldsymbol{\alpha}}(t)$ is defined as

$$\bar{\boldsymbol{\alpha}}(t) = \begin{cases} {}^{a}\bar{\boldsymbol{\alpha}}(t), & t \in [t_0, \tau + \delta], \\ {}^{p}\bar{\boldsymbol{\alpha}}(t), & t \in [\tau - \delta, t_f], \end{cases} \tag{2.20}$$

where ${}^{a}\bar{\boldsymbol{\alpha}}(t)$ denotes the extended ante-impact trajectory and ${}^{p}\bar{\boldsymbol{\alpha}}(t)$ the extended post-impact trajectory and where δ is a sufficiently large, task dependent time extension. Similarly, the extended reference feedforward input $\boldsymbol{\mu}(t)$ is given by

$$\bar{\boldsymbol{\mu}}(t) = \begin{cases} {}^{a}\bar{\boldsymbol{\mu}}(t), & t \in [t_0, \tau + \delta], \\ {}^{p}\bar{\boldsymbol{\mu}}(t), & t \in [\tau - \delta, t_f], \end{cases} \tag{2.21}$$

Local tracking of the extended reference trajectory is achieved using the reference spreading control law

$$\boldsymbol{u}(\boldsymbol{x},t) = \begin{cases} {}^{a}\bar{\boldsymbol{\mu}}(t) - \boldsymbol{K}(t)(\boldsymbol{x} - {}^{a}\bar{\boldsymbol{\alpha}}(t)), & t \in [t_0, t_1], \\ {}^{p}\bar{\boldsymbol{\mu}}(t) - \boldsymbol{K}(t)(\boldsymbol{x} - {}^{p}\bar{\boldsymbol{\alpha}}(t)), & t \in (t_1, t_f]. \end{cases} \tag{2.22}$$

In the left plot of Figure 2.4, the same state trajectory $\boldsymbol{x}$ is shown as in Figure 2.3, with the extended reference state trajectory $\bar{\boldsymbol{\alpha}}(t)$. The right figure shows the error $\boldsymbol{e}(t) = \|\boldsymbol{x} - \bar{\boldsymbol{\alpha}}(t)\|$, which does not experience peaking, due to the extension of the reference state trajectory. Consequently, using control law (2.22), no sudden large changes occur in the input that could result in poor tracking performance or destabilization of the system.

2.2.3 Reference spreading for simultaneous impacts

In the previous subsection, impacts have been considered that are caused by contact in a single point, which are referred to as single impacts. However, in many applications contacts are expected to be established simultaneously between multiple contact points on a surface, which is referred to as a simultaneous impact. An example of such a situation, which is directly related to this work, is a dual arm manipulator dynamically lifting a box through surface contacts between the end effectors and the box. In addition to the peaking phenomenon which is observed for single impacts, there is another challenge that occurs when the simultaneity of the impacts is lost due to perturbations. This is illustrated in Figure 2.5 for a planar box making impact with the floor. In blue, the box is shown tracking the reference state trajectory $\boldsymbol{\alpha}(t)$. At the intended impact time $t = \tau$, the box makes impact with the floor, through a simultaneous impact of all contact points on the box. The box in red visualizes a perturbed trajectory $\boldsymbol{x}$. As a result of the perturbation, the simultaneity of the impact is lost, causing two impacts to occur, at $t = t_1$ and $t = t_2$.

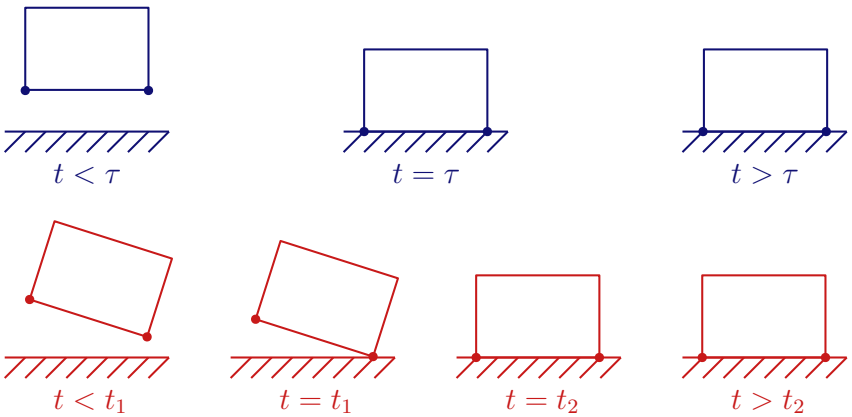


Figure 2.5: *Illustration of the loss of simultaneity as a result of perturbations. The blue box, tracking reference trajectory* $\boldsymbol{\alpha}(t)$*, makes a simultaneous impact with the ground at the intended impact time* $t = \tau$*. In red, a perturbed box with state trajectory* $\boldsymbol{x}$ *experiences two jumps at* $t = t_1$ *and* $t = t_2$*.*

Both trajectories $\boldsymbol{\alpha}(t)$ and $\boldsymbol{x}$ are depicted in Figure 2.6. It can be seen that the perturbed state trajectory $\boldsymbol{x}$ jumps twice, caused by the impacts at $t = t_1$ and $t = t_2$, whereas the reference state trajectory $\boldsymbol{\alpha}(t)$ jumps only once at $t = \tau$.

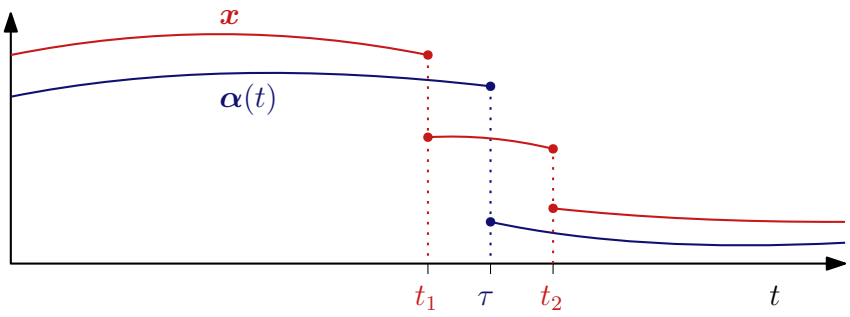


Figure 2.6: *Reference state trajectory $\boldsymbol{\alpha}(t)$ and perturbed state trajectory $\boldsymbol{x}$ associated with the motions illustrated in Figure 2.5. The reference state trajectory $\boldsymbol{\alpha}(t)$ jumps once, at $t = \tau$. The perturbed trajectory $\boldsymbol{x}$ shows two jumps at $t = t_1$ and $t = t_2$.*

Due to this mismatch in the number of jumps between the reference trajectory and the perturbed state trajectory, the perturbed state trajectory enters an unspecified mode in the time interval $t \in [t_1, t_2]$. This mode is called the intermediate unspecified mode, as the reference trajectory does not exist for the mode in which the box is in contact with the floor through only one contact point. In order to deal with the intermediate unspecified mode, the control law in (2.22) is modified into

$$\boldsymbol{u}(\boldsymbol{x}, t) = \begin{cases} {}^{a}\bar{\boldsymbol{\mu}}(t) - \boldsymbol{K}(\boldsymbol{x} - {}^{a}\bar{\boldsymbol{\alpha}}(t)), & t \in [t_0, t_1], \\ {}^{a}\bar{\boldsymbol{\mu}}(t), & t \in (t_1, t_\kappa], \\ {}^{p}\bar{\boldsymbol{\mu}}(t) - \boldsymbol{K}(\boldsymbol{x} - {}^{p}\bar{\boldsymbol{\alpha}}(t)), & t \in (t_\kappa, t_f], \end{cases} \tag{2.23}$$

where t_κ indicates the time at which the last impact is made, which is t_2 in case of the planar box. On the time interval $t \in [t_0, t_1]$, the extended ante-impact reference trajectory ${}^{a}\bar{\boldsymbol{\alpha}}(t)$ is tracked, which prevents peaking of the error in case $t_1 > \tau$. While the system is in the unspecified mode for $t \in (t_1, t_\kappa]$, the error is not defined, as no reference trajectory exists that prescribes the unspecified mode. For this reason, no feedback control is used during this mode. However, it is desired that the planned contact is eventually completed. In order to achieve this, the extended ante-impact feedforward input trajectory, related to the extended ante-impact reference state trajectory, is used as the input during the unspecified mode until the desired full contact is completed. Once this full contact is established, at $t = t_\kappa$, the extended post-impact trajectories are tracked with feedback control. This prevents the error from peaking if $t_\kappa < \tau$. Note that this approach is independent of the order in which the contacts are made, which does therefore not have to be known.

2.3 Single robot task-based QP control

In this section, task-based QP control is explained. First, the general theory of task-based QP control is detailed. Thereafter, the specific QP control implementation used in this research is reviewed, which is the multirobot QP controller proposed in [26]. The multirobot QP controller can be used to control multiple robots, interacting with each other and with unactuated mobile objects in the environment. This multirobot QP controller is an extension of the QP controller for a single robot, proposed in [27]. This section focuses on the single robot QP control implementation. The multirobot QP control implementation is detailed in the Section 2.4.

2.3.1 Task-based QP robot control

Using task-based control, the configuration of a robot body can be controlled in Cartesian space. Tasks can be specified to control a body to a desired pose. Thereby, tasks can be specified to control only specific degrees of freedom of a body, such as its Cartesian position or orientation. A task consists of an error function and a possibly time-varying reference signal. In general, an error function is given by

$$\boldsymbol{E} = \boldsymbol{e}_{\text{acc}} + \boldsymbol{K}_d\, \boldsymbol{e}_{\text{vel}} + \boldsymbol{K}_p\, \boldsymbol{e}_{\text{pose}}, \tag{2.24}$$

where $\boldsymbol{e}_{\text{pose}}$, $\boldsymbol{e}_{\text{vel}}$ and $\boldsymbol{e}_{\text{acc}}$ denote the error associated with the body pose, velocity and acceleration, respectively, expressed in task-space, and $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ denote the derivative and proportional control gains. In order to control the body to the reference signal, the error function has to be minimized, which is done using a QP controller. The general formulation of a QP control problem is given by

$$\min_{\boldsymbol{\chi}} \qquad \frac{1}{2}\boldsymbol{\chi}^T \boldsymbol{Q}\boldsymbol{\chi} + \boldsymbol{c}^T\boldsymbol{\chi}, \tag{2.25a}$$

s.t.

$$\boldsymbol{A}_1\boldsymbol{\chi} = \boldsymbol{b}_1, \tag{2.25b}$$

$$\boldsymbol{A}_2\boldsymbol{\chi} \geq \boldsymbol{b}_2, \tag{2.25c}$$

where $\boldsymbol{\chi} \in \mathbb{R}^q$ denotes the vector of optimization variables, with q the optimization variable state dimension, $\boldsymbol{Q} \in \mathbb{R}^{q\times q}$ the symmetric Hessian matrix, $\boldsymbol{c} \in \mathbb{R}^q$ a vector of constants, $\boldsymbol{A}_1 \in \mathbb{R}^{r\times q}$ and $\boldsymbol{b}_1 \in \mathbb{R}^r$ define the r equality constraints and $\boldsymbol{A}_2 \in \mathbb{R}^{s\times q}$ and $\boldsymbol{b}_2 \in \mathbb{R}^s$ the s inequality constraints [28]. The goal of the QP controller is to find the values of $\boldsymbol{\chi}$, for which cost function (2.25a) is minimized. If task error function (2.24) is used as the cost function, the QP controller computes the values of $\boldsymbol{\chi}$ for which the error is minimized. In order to be used in the QP controller, the error function has to be reformulated in the form of (2.25a). Usually, only the acceleration error $\boldsymbol{e}_{\text{acc}}$ is linearly dependent on the optimization variable $\boldsymbol{\chi}$, which means that (2.24) can also be written as

$$\boldsymbol{E} = \boldsymbol{H}\boldsymbol{\chi} + \boldsymbol{\eta}, \tag{2.26}$$

where $\boldsymbol{H}\boldsymbol{\chi} = \boldsymbol{e}_{\text{acc}}$ and $\boldsymbol{\eta} = \boldsymbol{K}_d\, \boldsymbol{e}_{\text{vel}} + \boldsymbol{K}_p\, \boldsymbol{e}_{\text{pose}}$. In order to fit the form of (2.25a), (2.26) is reformulated as

$$\boldsymbol{E} = \frac{1}{2}\boldsymbol{\chi}^T\boldsymbol{Q}\boldsymbol{\chi} + \boldsymbol{c}^T\boldsymbol{\chi} + \frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{\eta} = \frac{1}{2}\left\|\boldsymbol{H}\boldsymbol{\chi} + \boldsymbol{\eta}\right\|^2, \tag{2.27}$$

with $\boldsymbol{Q} = \boldsymbol{H}^T\boldsymbol{H}$ and $\boldsymbol{c}^T = \boldsymbol{\eta}^T\boldsymbol{H}$. Note that the term $\frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{\eta}$ is ignored in the QP problem (2.25), as it does not contain the optimization variables. Equality constraints (2.25b) and inequality constraints (2.25c) can be used to set constraints and bounds on the optimization variables $\boldsymbol{\chi}$, such as contact constraints and joint bounds.

2.3.2 Single robot task-based QP control without contacts

In the QP control implementation proposed in [27], a weighted, multi-task cost function is used, in order to perform multiple tasks simultaneously using a single robot. The multi-task cost function is formulated as

$$\boldsymbol{E}_{\text{sum}} = \frac{1}{2}\boldsymbol{\chi}^T\boldsymbol{Q}\boldsymbol{\chi} + \boldsymbol{c}^T\boldsymbol{\chi} + \frac{1}{2}\boldsymbol{c}^T\boldsymbol{c} = \sum_{k=1}^{n_T} w_k\boldsymbol{E}_k, \tag{2.28}$$

where $\boldsymbol{E}_k$ denotes a task of the form (2.27) with corresponding task weight w_k and n_T the total number of tasks. Using the task weights w_k, priority can be given to specific tasks over other tasks. The QP control problem for motion tasks without contacts, for a single robot is given by

$$\min_{\ddot{\mathbf{q}}} \quad \sum_{k=1}^{n_T} w_k \boldsymbol{E}_k, \tag{2.29a}$$

s.t.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\,\boldsymbol{\tau}, \tag{2.29b}$$

The optimization variables are the joint accelerations of the manipulator $\ddot{\mathbf{q}} \in \mathbb{R}^{n_J}$. The QP problem is subject to the equation of motion for the robot, where $\mathbf{M} \in \mathbb{R}^{n_J \times n_J}$ denotes the mass matrix, $\mathbf{C} \in \mathbb{R}^{n_J \times n_J}$ the Coriolis matrix, $\mathbf{G} \in \mathbb{R}^{n_J}$ the potential force vector and $\mathbf{S} \in \mathbb{R}^{n_J \times m}$ the joint selection matrix, which is equal to the identity matrix in case of a fully actuated manipulator, and $\boldsymbol{\tau} \in \mathbb{R}^m$ the joint torques. In addition to the equation of motion, the QP problem is subject to other constraints, such as joint position, velocity and torque bounds. The definition of these constraints is detailed in [29] and will not be further discussed in this work. Here, only the constraints that are relevant for this work are elaborated.
Using QP problem (2.29), the joint torques are computed that are necessary for a robot to perform a desired motion specified by the QP tasks with error function $\boldsymbol{E}_k$. These joint torques are sent to the robot at a fixed rate. Therefore, the QP problem has to be solved at every time step, using the current values of $\mathbf{q}$ and $\dot{\mathbf{q}}$ at this time step. The QP controller computes the joint accelerations $\ddot{\mathbf{q}}$ for which the cost function (2.29a) is minimized, while satisfying the equation of motion (2.29b) and other constraints. The joint torques, that are used as the input for the robot are found using the computed joint accelerations $\ddot{\mathbf{q}}$ and (2.29b).

2.3.3 Single robot QP control with contacts

When a robot is in contact with a fixed environment, the QP problem given by (2.29) is modified. This extended QP problem is only valid for sticking contacts. The full QP problem for a robot in sticking contact with a fixed environment is given by

$$\min_{\ddot{\mathbf{q}}, \boldsymbol{\lambda}} \quad \sum_{k=1}^{n_T} w_k \boldsymbol{E}_k + \|\boldsymbol{\lambda}\|^2, \tag{2.30a}$$

s.t.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\,\boldsymbol{\tau} + \sum_{c \in \mathcal{I}_C} \mathbf{J}_c^T(\mathbf{q})\ \mathbf{f}_c, \tag{2.30b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \tag{2.30c}$$

$$\left(\mathbf{J}_c(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_c(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}\right) \Delta t + \mathbf{J}_c(\mathbf{q})\dot{\mathbf{q}} + \frac{\mathrm{Err}}{\Delta t} = \mathbf{0}, \tag{2.30d}$$

where $\boldsymbol{\lambda}$ denotes the vector of linearized friction cone base weights, $\mathcal{I}_C$ the set of closed contacts, $\mathbf{f}_c$ the contact forces at the c-th contact point and $\mathbf{J}_c$ the geometric Jacobian associated to the frame that is rigidly attached to this contact point, Δt the time step of the QP controller and Err an error function comparing the current pose of a frame to its desired pose. An explanation of cost function

(2.30a), constraints (2.30b) - (2.30d) and the aforementioned variables is given in the remainder of this subsection.
First, consider the equation of motion given by (2.30b), which is found by extending (2.29b) with a term for the contact forces. The contact forces $\mathbf{f}_c$ in each contact point $c \in \mathcal{I}_C$ can be written as a linear combination of vectors $\{\vec{r}_1, \ldots, \vec{r}_\nu\}$ which construct the linearized friction cone, such that

$$\mathbf{f}_c = \sum_{\mu=1}^{\nu} \lambda_{c,\mu} \vec{r}_{c,\mu}, \tag{2.31}$$

where $\lambda_{c,\mu}$ is a linearized friction cone base weight. The vector of linearized friction cone base weights for each contact point is denoted by $\boldsymbol{\lambda}_c$. The vector of linearized friction cone base weights for all contact points, given by

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_{n_C} \end{bmatrix}, \tag{2.32}$$

with n_C the number of contact points, is used as minimization variable, in addition to the joint accelerations $\ddot{\mathbf{q}}$. For regularization purposes, $\boldsymbol{\lambda}$ is added to the QP cost function (2.30a) in quadratic form. The base weights have to be positive in order for the contacts to be unilateral and inside the friction cone. For this reason, the QP problem is subject to (2.30c).
The equality constraint given by (2.30d), represents a contact constraint, which serves to maintain a contact between a robot and its environment. This constraint is explained using the example in Figure 2.7. Suppose the manipulator is in contact with the fixed environment through a contact of

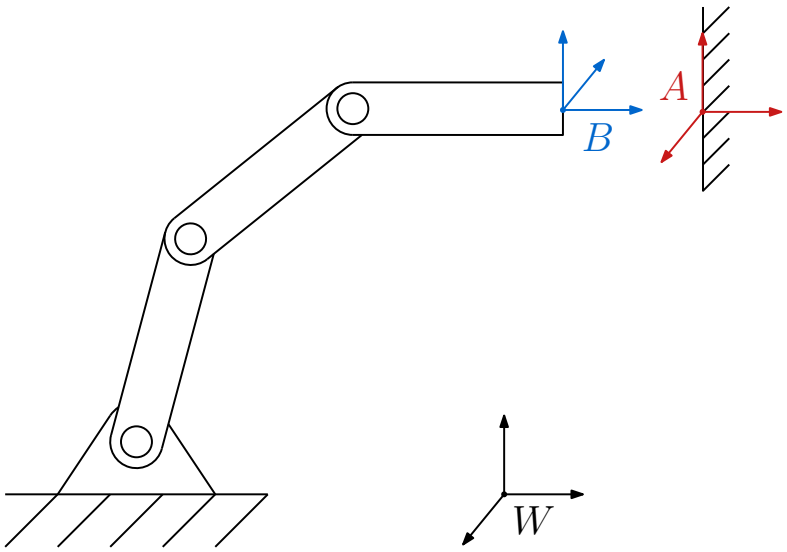


Figure 2.7: *Illustrative example of a manipulator making contact with a fixed environment. Frame B is the contact frame rigidly attached to the manipulator. Frame A is a fixed frame in the contact point on the environment.*

the origin of frame B, which is rigidly attached to the manipulator in the contact point, and the origin of frame A, which is the fixed frame attached to the environment. A contact constraint for the

manipulator can be formulated by constraining the velocity of frame B to zero by

$$ {}^{W}\mathbf{J}_{W,B}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0}. \tag{2.33} $$

In order to be compatible with the QP problem, this constraint is differentiated with respect to time, resulting in

$$ {}^{W}\mathbf{J}_{W,B}(\mathbf{q})\ddot{\mathbf{q}} + {}^{W}\dot{\mathbf{J}}_{W,B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{0}. \tag{2.34} $$

However, this formulation of the contact constraint imposes issues when the velocity is not exactly zero at the moment the contact constraint is applied or when the velocity diverges from zero as a result of numerical integration. Therefore, the velocity is included in the contact constraint by

$$ \left({}^{W}\mathbf{J}_{W,B}(\mathbf{q})\ddot{\mathbf{q}} + {}^{W}\dot{\mathbf{J}}_{W,B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right)\Delta t + {}^{W}\mathbf{J}_{W,B}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0}, \tag{2.35} $$

where Δt denotes the time step of the QP controller. However, this constraint does not yet guarantee that the contact is maintained, as small perturbations can cause the manipulator pose to diverge from the desired pose. For this purpose, the constraint is modified into

$$ \left({}^{W}\mathbf{J}_{W,B}(\mathbf{q})\ddot{\mathbf{q}} + {}^{W}\dot{\mathbf{J}}_{W,B}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right)\Delta t + {}^{W}\mathbf{J}_{W,B}(\mathbf{q})\dot{\mathbf{q}} + \begin{bmatrix} {}^{B}\mathbf{o}_{A}^{\mathrm{ref}} - {}^{B}\mathbf{o}_{A} \\ \log({}^{A}\mathbf{R}_{B}^{\mathrm{ref}}\,{}^{B}\mathbf{R}_{A})^{\vee} \end{bmatrix} \frac{1}{\Delta t} = \mathbf{0}. \tag{2.36} $$

where ${}^{B}\mathbf{o}_{A}^{\mathrm{ref}}$ denotes the desired distance between the origins of frame A and B and ${}^{A}\mathbf{R}_{B}^{\mathrm{ref}}$ the desired rotation between the orientations $[A]$ and $[B]$. The reference distance and rotation are equal to the distance and rotation at the moment the contact constraint is initially applied. Since the Jacobian relating the velocity of frame B with respect to the world frame ${}^{W}\mathbf{J}_{W,B}(\mathbf{q})$ is the same as $\mathbf{J}_{c}(\mathbf{q})$, the contact constraint can be simplified to the (2.30d), where

$$ \mathrm{Err} = \begin{bmatrix} {}^{B}\mathbf{o}_{A}^{\mathrm{ref}} - {}^{B}\mathbf{o}_{A} \\ \log({}^{A}\mathbf{R}_{B}^{\mathrm{ref}}\,{}^{B}\mathbf{R}_{A})^{\vee} \end{bmatrix}. \tag{2.37} $$

This concludes the definition of the QP problem for a single robot in sticking contact, given by (2.30).

2.4 Multirobot QP control

The multirobot QP controller proposed in [26] allows to control multiple interacting robots, which are in contact through sticking contacts, using a single centralized controller. The multirobot controller combines the equations of motion, tasks and interaction constraints of all robots in a single QP problem. In addition to actuated robots, the controller considers unactuated mobile objects to be robots as well. In this section, first the combined equation of motion is presented. Thereafter, contact constraints between interacting robots are detailed. Then, the formulation of the multirobot QP control problem is given. Finally, an overview is given of the control framework that is used to implement the multirobot QP controller on a scene of multiple robots and objects.

2.4.1 Multirobot equation of motion

Each robot, considered in the multirobot QP controller has an equation of motion, given by (2.30b). In this research, only contacts between robots are considered. In this case, the contact forces $\mathbf{f}_c$ appearing in the equation of motion of one robot are applied by another robot. Newton's third law implies that the contact forces on one robot appear with an opposite sign in the equation of motion of another robot. Consider n_R interacting robots numbered by $i \in \{1, \ldots, n_R\}$. For robot i, the summation of contact forces over all contact points in (2.30b) can be rewritten as

$$\sum_{c \in \mathcal{I}_C} \mathbf{J}_c^T(\mathbf{q})\ \mathbf{f}_c = \mathbf{J}_{i,1}^T(\mathbf{q}_i)\ \mathbf{f}_{i,1} - \mathbf{J}_{i,2}^T(\mathbf{q}_i)\ \mathbf{f}_{i,2}, \tag{2.38}$$

where $\mathbf{f}_{i,1}$ denotes all contact forces applied by robots $j \in \{1, \ldots, i-1\}$ on robot i, with associated Jacobian $\mathbf{J}_{i,1}(\mathbf{q}_i)$ and $\mathbf{f}_{i,2}$ the contact forces that are applied by robot i on robots $j \in \{i+1, \ldots, n_R\}$, with associated Jacobian $\mathbf{J}_{i,2}(\mathbf{q}_i)$. The equation of motion for each of the robots, given by (2.30b), can therefore be rewritten as

$$\mathbf{M}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \mathbf{C}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i + \mathbf{G}_i(\mathbf{q}_i) = \mathbf{S}_i\,\boldsymbol{\tau}_i + \mathbf{J}_{i,1}^T(\mathbf{q}_i)\ \mathbf{f}_{i,1} - \mathbf{J}_{i,2}^T(\mathbf{q}_i)\ \mathbf{f}_{i,2}. \tag{2.39}$$

Let $\boldsymbol{F}_1$ denote the stacked vectors of contact forces $\mathbf{f}_{i,1}$, such that $\boldsymbol{F}_1 = (\mathbf{f}_{1,1}, \ldots, \mathbf{f}_{n_R,1})^T$. As a result of Newton's third law, all contact forces $\mathbf{f}_{i,2}$ appear at some position in $\boldsymbol{F}_1$. Therefore, these contact forces can be written as

$$\mathbf{f}_{i,2} = \boldsymbol{\phi}_i \boldsymbol{F}_1, \tag{2.40}$$

where $\boldsymbol{\phi}_i$ is a selection matrix reordering the contact forces in $\boldsymbol{F}_1$ to contact forces $\mathbf{f}_{i,2}$. Consequently, (2.39) can be rewritten to

$$\mathbf{M}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \mathbf{C}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i + \mathbf{G}_i(\mathbf{q}_i) = \mathbf{S}_i\,\boldsymbol{\tau}_i + \mathbf{J}_{i,1}^T(\mathbf{q}_i)\ \mathbf{f}_{i,1} - \mathbf{J}_{i,2}^T(\mathbf{q}_i)\ \boldsymbol{\phi}_i \mathbf{F}_1. \tag{2.41}$$

The combined equation of motion for all robots is given by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\,\boldsymbol{\tau} + \left(\mathbf{J}_1^T(\mathbf{q}) - \mathbf{J}_2^T(\mathbf{q})\boldsymbol{\Phi}\right)\mathbf{F}_1, \tag{2.42}$$

where

$$\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{n_R})^T, \tag{2.43a}$$
$$\mathbf{M}(\mathbf{q}) = \operatorname{diag}\left(\mathbf{M}_1(\mathbf{q}_1), \ldots, \mathbf{M}_{n_R}(\mathbf{q}_{n_R})\right), \tag{2.43b}$$
$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \operatorname{diag}\left(\mathbf{C}_1(\mathbf{q}_1, \dot{\mathbf{q}}_1), \ldots, \mathbf{C}_{n_R}(\mathbf{q}_{n_R}, \dot{\mathbf{q}}_{n_R})\right), \tag{2.43c}$$
$$\mathbf{G}(\mathbf{q}) = (\mathbf{G}_1(\mathbf{q}_1), \ldots, \mathbf{G}_{n_R}(\mathbf{q}_{n_R}))^T, \tag{2.43d}$$
$$\mathbf{S} = \operatorname{diag}\left(\mathbf{S}_1, \ldots, \mathbf{S}_{n_R}\right), \tag{2.43e}$$
$$\boldsymbol{\tau} = (\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{n_R})^T, \tag{2.43f}$$
$$\mathbf{J}_1(\mathbf{q}) = \operatorname{diag}\left(\mathbf{J}_{1,1}(\mathbf{q}_1), \ldots, \mathbf{J}_{n_R,1}(\mathbf{q}_{n_R})\right), \tag{2.43g}$$
$$\mathbf{J}_2(\mathbf{q}) = \operatorname{diag}\left(\mathbf{J}_{1,2}(\mathbf{q}_1), \ldots, \mathbf{J}_{n_R,2}(\mathbf{q}_{n_R})\right), \tag{2.43h}$$
$$\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{n_R})^T. \tag{2.43i}$$

The combined equation of motion (2.42) is implemented as an equality constraint in the multirobot QP controller, which is detailed in Section 2.4.3.

2.4.2 Multirobot contact constraints

The multirobot QP controller uses contact constraints to maintain the contact between two robots. As opposed to the contact constraints discussed in Section 2.3.3, the constraint is applied between two frames that are both allowed to move. This contact constraint is explained using the example of two manipulator arms lifting a box, depicted in Figure 2.8. Suppose the manipulator arms are in contact

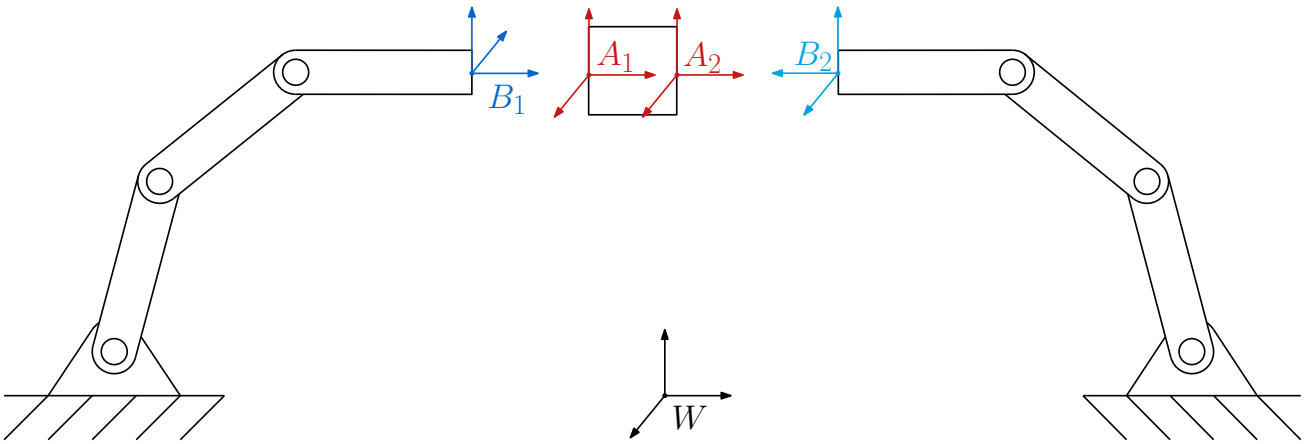


Figure 2.8: *Illustrative example of two manipulators interacting with a box. Frame B_1 and B_2 are the contact frames rigidly attached to the manipulators on the left and right, respectively. Frame A_1 and A_2 are the contact frames on the left and right side of the box, respectively.*

with the box, such that the origins of frames B_1 and A_1 coincide and the origins of frame B_2 and A_2 coincide. In order to assure that the origins of the frames remain coincident and to prevent the frames from rotating with respect to each other, contact constraints can be defined. The contact constraints are formulated as

$$ {}^{W}\mathbf{J}_{W,B_1}(\mathbf{q}_1)\dot{\mathbf{q}}_1 - {}^{W}\mathbf{J}_{W,A_1}(\mathbf{q}_{\text{box}})\dot{\mathbf{q}}_{\text{box}} = \mathbf{0}, \tag{2.44} $$

$$ {}^{W}\mathbf{J}_{W,B_2}(\mathbf{q}_2)\dot{\mathbf{q}}_2 - {}^{W}\mathbf{J}_{W,A_2}(\mathbf{q}_{\text{box}})\dot{\mathbf{q}}_{\text{box}} = \mathbf{0}, \tag{2.45} $$

where $\mathbf{q}_1$ and $\mathbf{q}_2$ denote the joint positions of the left and right manipulator arm respectively and $\mathbf{q}_{\text{box}}$ denotes the generalized coordinates of the box. In order to improve readability, the explicit dependency of the Jacobian on the joint positions is dropped in the remainder of this section. In order to be compatible with the QP controller, the contact constraints are differentiated with respect to time, resulting in

$$ {}^{W}\mathbf{J}_{W,B_1}\ddot{\mathbf{q}}_1 + {}^{W}\dot{\mathbf{J}}_{W,B_1}\dot{\mathbf{q}}_1 - \left({}^{W}\mathbf{J}_{W,A_1}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_1}\dot{\mathbf{q}}_{\text{box}}\right) = \mathbf{0}, \tag{2.46} $$

$$ {}^{W}\mathbf{J}_{W,B_2}\ddot{\mathbf{q}}_2 + {}^{W}\dot{\mathbf{J}}_{W,B_2}\dot{\mathbf{q}}_2 - \left({}^{W}\mathbf{J}_{W,A_2}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_2}\dot{\mathbf{q}}_{\text{box}}\right) = \mathbf{0}. \tag{2.47} $$

Similar to the contact constraint in Section 2.3.3, the contact constraints are extended to prevent the relative velocities of the constrained frames from diverging from zero. This extends the constraint to

$$\begin{aligned}\left({}^{W}\mathbf{J}_{W,B_1}\ddot{\mathbf{q}}_1 + {}^{W}\dot{\mathbf{J}}_{W,B_1}\dot{\mathbf{q}}_1\right)\Delta t + {}^{W}\mathbf{J}_{W,B_1}\dot{\mathbf{q}}_1 \\ - \left({}^{W}\mathbf{J}_{W,A_1}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_1}\dot{\mathbf{q}}_{\text{box}}\right)\Delta t - {}^{W}\mathbf{J}_{W,A_1}\dot{\mathbf{q}}_{\text{box}} = \mathbf{0},\end{aligned} \tag{2.48}$$

$$\begin{aligned}\left({}^{W}\mathbf{J}_{W,B_2}\ddot{\mathbf{q}}_2 + {}^{W}\dot{\mathbf{J}}_{W,B_2}\dot{\mathbf{q}}_2\right)\Delta t + {}^{W}\mathbf{J}_{W,B_2}\dot{\mathbf{q}}_2 \\ - \left({}^{W}\mathbf{J}_{W,A_2}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_2}\dot{\mathbf{q}}_{\text{box}}\right)\Delta t - {}^{W}\mathbf{J}_{W,A_2}\dot{\mathbf{q}}_{\text{box}} = \mathbf{0},\end{aligned} \tag{2.49}$$

with Δt the time step of the QP controller. Finally, a term is added to prevent the relative position and orientation of the constraint frames from changing, such that

$$\begin{aligned}\left({}^{W}\mathbf{J}_{W,B_1}\ddot{\mathbf{q}}_1 + {}^{W}\dot{\mathbf{J}}_{W,B_1}\dot{\mathbf{q}}_1\right)\Delta t + {}^{W}\mathbf{J}_{W,B_1}\dot{\mathbf{q}}_1 - \left({}^{W}\mathbf{J}_{W,A_1}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_1}\dot{\mathbf{q}}_{\text{box}}\right)\Delta t \\ - {}^{W}\mathbf{J}_{W,A_1}\dot{\mathbf{q}}_{\text{box}} + \begin{bmatrix} {}^{B_1}\mathbf{o}^{\text{ref}}_{A_1} - {}^{B_1}\mathbf{o}_{A_1} \\ \log({}^{A_1}\mathbf{R}^{\text{ref}}_{B_1}\,{}^{B_1}\mathbf{R}_{A_1})^{\vee}\end{bmatrix}\frac{1}{\Delta t} = \mathbf{0},\end{aligned} \tag{2.50}$$

$$\begin{aligned}\left({}^{W}\mathbf{J}_{W,B_2}\ddot{\mathbf{q}}_2 + {}^{W}\dot{\mathbf{J}}_{W,B_2}\dot{\mathbf{q}}_2\right)\Delta t + {}^{W}\mathbf{J}_{W,B_2}\dot{\mathbf{q}}_2 - \left({}^{W}\mathbf{J}_{W,A_2}\ddot{\mathbf{q}}_{\text{box}} + {}^{W}\dot{\mathbf{J}}_{W,A_2}\dot{\mathbf{q}}_{\text{box}}\right)\Delta t \\ - {}^{W}\mathbf{J}_{W,A_2}\dot{\mathbf{q}}_{\text{box}} + \begin{bmatrix} {}^{B_2}\mathbf{o}^{\text{ref}}_{A_2} - {}^{B_2}\mathbf{o}_{A_2} \\ \log({}^{A_2}\mathbf{R}^{\text{ref}}_{B_2}\,{}^{B_2}\mathbf{R}_{A_2})^{\vee}\end{bmatrix}\frac{1}{\Delta t} = \mathbf{0}.\end{aligned} \tag{2.51}$$

Using the notation of (2.39), a contact constraint between robot i and j can be written as

$$\begin{aligned}\left(\mathbf{J}_{1,i}(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \dot{\mathbf{J}}_{1,i}(\mathbf{q}_i,\dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i\right)\Delta t + \mathbf{J}_{1,i}(\mathbf{q}_i)\dot{\mathbf{q}}_i \\ - \left(\mathbf{J}_{2,j}(\mathbf{q}_j)\ddot{\mathbf{q}}_j + \dot{\mathbf{J}}_{2,j}(\mathbf{q}_j,\dot{\mathbf{q}}_j)\dot{\mathbf{q}}_j\right)\Delta t - \mathbf{J}_{2,j}(\mathbf{q}_j)\dot{\mathbf{q}}_j + \frac{\text{Err}}{\Delta t} = \mathbf{0},\end{aligned} \tag{2.52}$$

where

$$\text{Err} = \begin{bmatrix} {}^{B_{1,i}}\mathbf{o}^{\text{ref}}_{B_{2,j}} - {}^{B_{1,i}}\mathbf{o}_{B_{2,j}} \\ \log({}^{B_{2,j}}\mathbf{R}^{\text{ref}}_{B_{1,i}}\,{}^{B_{1,i}}\mathbf{R}_{B_{2,j}})^{\vee}\end{bmatrix}, \tag{2.53}$$

with $B_{1,i}$ and $B_{2,j}$ the frames rigidly attached to the contact points on robot i and j, respectively. These contact constraints are added to the QP problem presented in the following subsection.

2.4.3 Multirobot QP problem

Given the formulation of the combined equation of motion presented in Section 2.4.1 and the contact constraints between the robots presented in 2.4.2, the multirobot QP controller is given by

$$\min_{\ddot{\mathbf{q}},\boldsymbol{\lambda}} \quad \sum_{k=1}^{n_T} w_k \boldsymbol{E}_k + \|\boldsymbol{\lambda}\|^2, \tag{2.54a}$$

s.t.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\,\boldsymbol{\tau} + \left(\mathbf{J}_1^T(\mathbf{q}) - \mathbf{J}_2^T(\mathbf{q})\boldsymbol{\Phi}\right)\mathbf{F}_1, \tag{2.54b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \tag{2.54c}$$

$$\begin{aligned}\Big(\mathbf{J}_{1,i}(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \dot{\mathbf{J}}_{1,i}(\mathbf{q}_i,\dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i\Big)\,\Delta t + \mathbf{J}_{1,i}(\mathbf{q}_i)\dot{\mathbf{q}}_i \\ - \left(\mathbf{J}_{2,j}(\mathbf{q}_j)\ddot{\mathbf{q}}_j + \dot{\mathbf{J}}_{2,j}(\mathbf{q}_j,\dot{\mathbf{q}}_j)\dot{\mathbf{q}}_j\Big)\,\Delta t - \mathbf{J}_{2,j}(\mathbf{q}_j)\dot{\mathbf{q}}_j + \frac{\mathrm{Err}}{\Delta t} = \mathbf{0},\end{aligned} \tag{2.54d}$$

where the minimization variable $\boldsymbol{\lambda}$ denotes the stacked vector of linearized friction cone base weights $\boldsymbol{\lambda}_i$, such that $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{n_R})^T$. Note that the tasks with cost functions $\boldsymbol{E}_k$, can be tasks for any of the robots i.

2.4.4 Control framework

The implementation of the multirobot QP controller proposed in [26] on a scene with multiple robots and objects is represented by the block diagram in Figure 2.9. The control framework uses an overarching

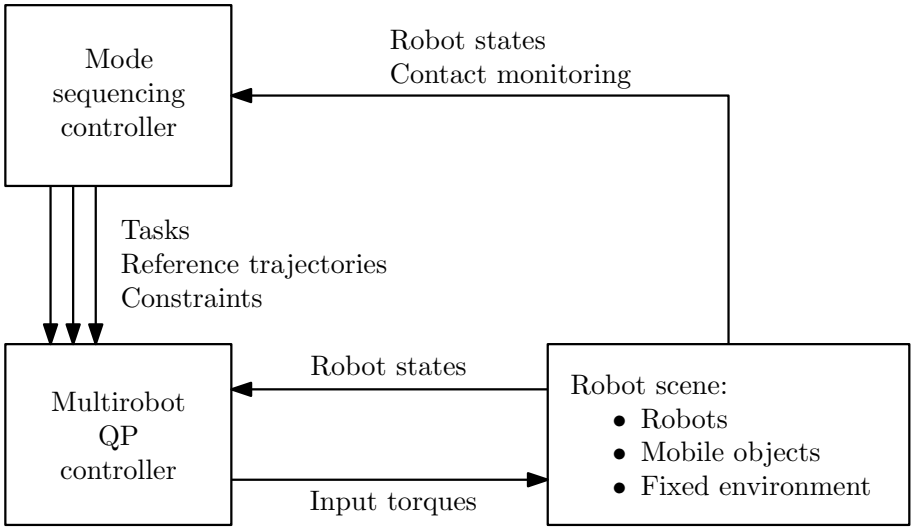


Figure 2.9: *Schematic overview of the control framework that is used to implement the multirobot QP controller on multiple physical robots or simulation models.*

controller, referred to as the mode sequencing controller, in order to specify the tasks, reference task trajectories and constraints for the multirobot QP controller. This controller can reside in different modes, which can differentiate in task and reference specification, or impose different constraints.

For example, separate modes can be specified for free motion, in which the robots do not interact, and constrained motion with interaction between the robots. Switching rules can be defined in the controller to switch between the modes and in this way a mode sequence can be specified. These switching rules can be based on the robot states, contact forces or other sensor data, which are obtained from the robot scene.
The robot scene represents the collection of robots and mobile objects that are controlled using the multirobot QP controller and any additional sensors that are used. The scene can consist of physical robots and objects or simulated models in a dynamical system simulator. Such a dynamical system simulator is discussed in the next section.
The multirobot QP controller computes the joint torques for all actuated robots in the robot scene, as discussed in the previous subsections. For this purpose, the QP controller obtains the robot states from the robot scene.

2.5 Dynamical systems simulators

In this work, numerical experiments are performed on robot models in a simulation environment. In this case, the robot scene in Figure 2.9, can be visualized schematically by Figure 2.10. The robot

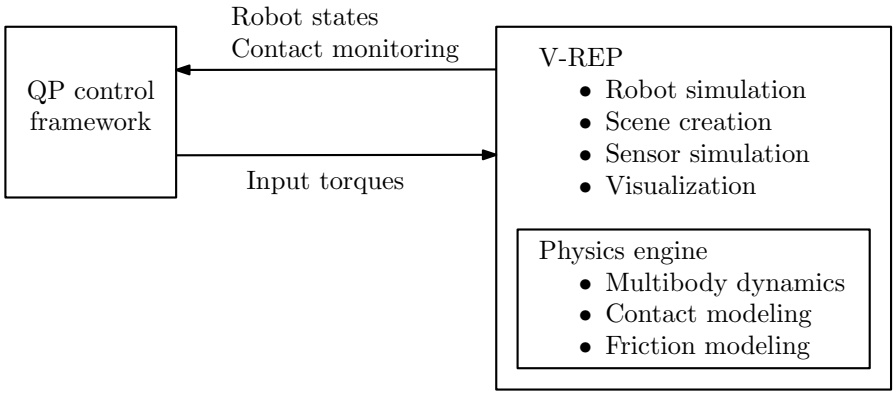


Figure 2.10: *Schematic overview of the simulated robot scene consisting of the V-REP robot simulator and an integrated physics engine. The bulleted lists summarize the main features of each framework.*

simulator V-REP, depicted in Figure 2.11, is used to create a virtual experimentation environment, consisting of controllable robots and unactuated objects [30]. In addition, sensors, like proximity and force sensors can be included in the simulation scene, which can for example be used to detect contacts between robots. V-REP provides tools for visualization of the simulation scene as well.
Integrated in V-REP are multiple physics engines. A physics engine is used to simulate the multibody dynamics of the robots and objects in the simulation environment. In addition, the interaction between the robots and objects is simulated by the physics engine. For this purpose, the physics engine models the response to impacts, contacts and friction. In this research, the Vortex Dynamics physics engine is used [31]. The Vortex Dynamics physics engine focuses on high performance and high precision simulations. In addition, Vortex Dynamics allows to set many physical properties, like material properties, such that near-to-real physics behavior is achieved.

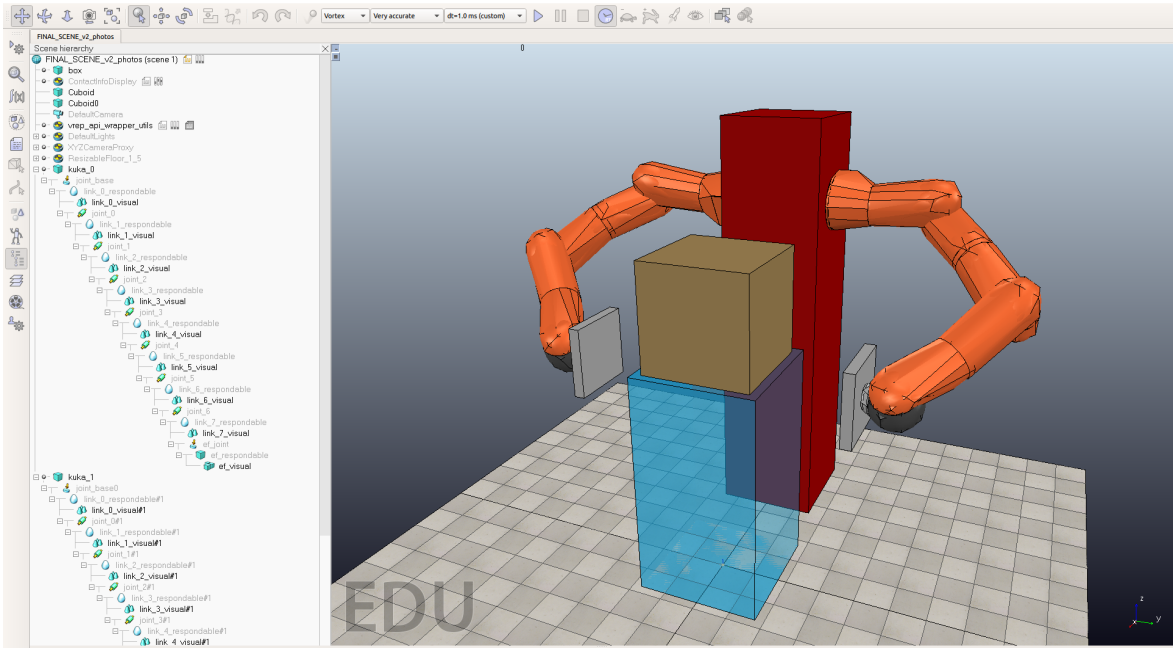


Figure 2.11: *The V-REP simulation environment used for numerical experiments.*

2.6 Interpolation using the De Casteljau algorithm

In this research, the De Casteljau algorithm is used to generate trajectories that are subject to boundary conditions. The De Casteljau algorithm is a recursive approach for computing a $\mathcal{C}^2$ smooth polynomial, between two end points [32]. The algorithm can be applied on end points in various manifolds, including $\mathbb{R}^3$, SO(3) and S^2. Regardless of the considered manifold, the algorithm uses the same procedure to compute the polynomial. In this section, this procedure is explained for the general case. In Section 3.3, the manifold-specific aspects of the De Casteljau algorithm are discussed.
The De Casteljau algorithm for computing a polynomial of degree d is defined as follows:

1. Specify $d+1$ control points, denoted by C_i with $i \in \{1, \ldots, d+1\}$, where the C_1 and C_{d+1} specify the end points of the polynomial and the intermediary control points define the shape of the polynomial.
2. Compute the first order interpolation between all consecutive control points. Let $P_i^1(\lambda)$ denote the function moving along the first order interpolation from control point C_i to C_{i+1} for increasing $\lambda \in [0, 1]$, such that $P_i^1(0) = C_i$ and $P_i^1(1) = C_{i+1}$.
3. Define new control points C_i^1 on $P_i^1(\lambda)$ evaluated at λ.
4. Compute the first order interpolation between the consecutive control points C_i^1 in order to find $P_i^2(\lambda)$.
5. Repeat step 3 and 4 for $P_i^j(\lambda)$, resulting in new control points C_i^j, where j is increased by one for every repetition. Finally, the function $P_1^d(\lambda)$ defines the interpolation of degree d between C_1 and C_d.

As an illustrative example, a polynomial $P_1^2(\lambda) \in \mathbb{R}^2$ of degree 2 is shown in Figure 2.12, which is computed using the De Casteljau algorithm.

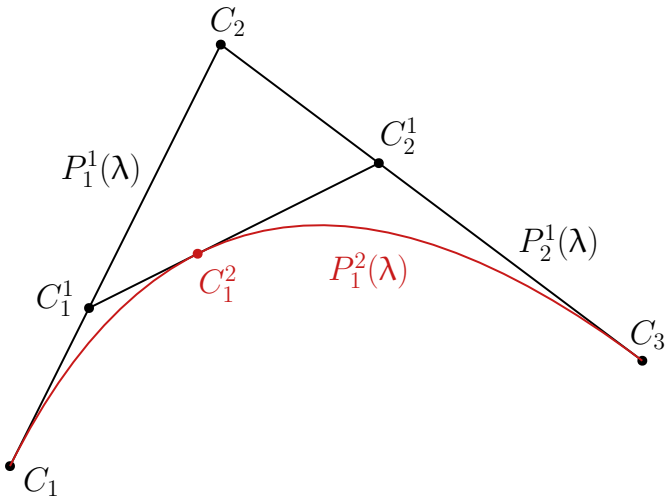


Figure 2.12: *A polynomial $P_1^2(\lambda)$ of degree two, computed using the De Casteljau algorithm on $\mathbb{R}^2$.*

The formulation of the first order interpolation between two consecutive control points $P_i^j(\lambda)$ depends on the considered manifold. These functions are detailed for three different manifolds in Section 3.3. By choosing the control points C_i in a specific way, the first and second order derivative of the computed polynomial at the boundaries can be specified. This is useful for the generation of reference trajectories with boundary conditions on the velocity and acceleration. The generation of such reference trajectories and the specific choices for the control points C_i are also presented in Section 3.3.

2.7 Summary

In this chapter, the essential background information for this research has been detailed. First, the multibody dynamics notation, which is used in the remainder of this research, has been presented. Thereafter, reference spreading and a specific task-based QP control implementation have been reviewed, which are combined in Chapter 3 in order to obtain a task-based QP control strategy for impact aware manipulation. Then, the software that is used for the numerical simulation study in Chapter 4 has been introduced. Finally, an interpolation procedure called the De Casteljau algorithm has been detailed, which is used for trajectory generation.

Chapter 3

Task-Based Reference Spreading for Impact Aware Manipulation

In this chapter, reference spreading is combined with task-based QP control, resulting in a QP control paradigm that allows for impact aware manipulation, called task-based reference spreading. First, it is shown how task-based reference trajectories have to be defined, in order to be compatible with impact dynamics. Then, inspired by classical reference spreading, extended reference trajectories are used in combination with a specific QP control framework, in order to deal with perturbations. The generation procedure for these extended reference task trajectories and the QP control framework are dependent on the application and robot that are considered. For this purpose, a dynamic box-lifting application for a dual arm manipulator is introduced, for which three QP tasks are defined. Thereafter, for each task, trajectory generation procedures are presented. Finally, the QP control framework for the considered application is detailed.

3.1 Task-based reference spreading

In this section, the approach for combining reference spreading with task-based QP control is presented. In classical reference spreading, a state-feedback approach is used to track reference state trajectories. However, in task-based QP control, reference task trajectories are used to prescribe a desired motion. Therefore, in order to combine reference spreading with task-based QP control, reference spreading should be applied on reference task trajectories. Whereas reference state trajectories can be used to prescribe desired motions that involve intentional impacts, this is not possible using reference task trajectories, in case of task redundancy in the robot. It is shown that using additional tasks, the task redundancy in the manipulator can be removed. In this case, the reference task trajectories, associated with these tasks, implicitly define a unique reference state trajectory. This allows to prescribe motions with intentional impacts using reference task trajectories. Finally, reference spreading is applied to these reference task trajectories by extending the trajectories beyond the intended impact time.

3.1.1 Reference state trajectories vs. reference task trajectories

In classical reference spreading, as reviewed in Section 2.2, a state-feedback controller is used to track reference state trajectories that prescribe motion tasks with intentional impacts. As an illustrative example of such a motion task, consider the three-link manipulator, depicted in Figure 3.1, making impact with the wall and moving along the wall after the impact has occurred. The reference state trajectory for this motion, denoted by $\boldsymbol{\alpha}(t)$, prescribes the reference positions and velocities for each joint of the manipulator in time, namely

$$\boldsymbol{\alpha}(t) = \begin{bmatrix} \mathbf{q}^{\text{ref}}(t) \\ \dot{\mathbf{q}}^{\text{ref}}(t) \end{bmatrix}, \tag{3.1}$$

where $\mathbf{q}^{\text{ref}}(t),\ \dot{\mathbf{q}}^{\text{ref}}(t) \in \mathbb{R}^{n_J}$ denote the reference joint positions and velocities, respectively, with n_J the number of joints in the manipulator. The reference state trajectory $\boldsymbol{\alpha}(t)$ is shown in Figure 3.2.

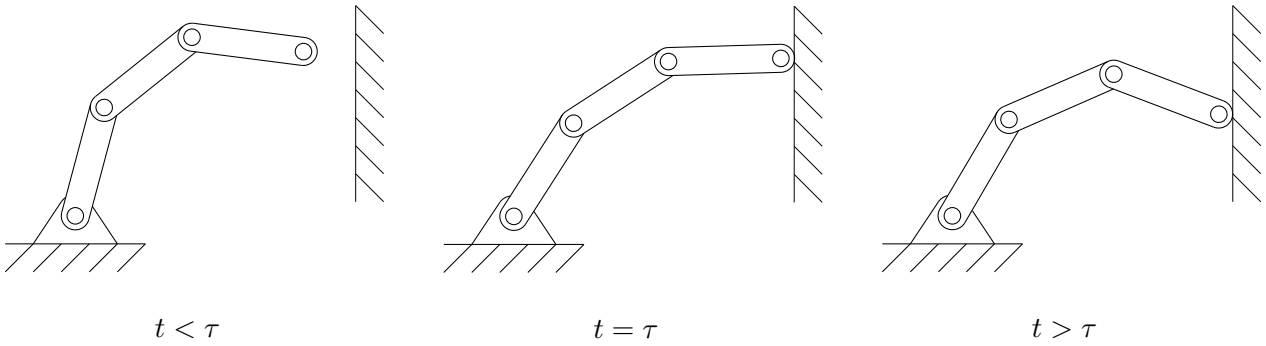


Figure 3.1: *A three-link manipulator performing a motion task involving an impact. Before impact time, for $t < \tau$, the manipulator end effector moves towards the wall in order to make impact. The impact occurs at the intended impact time $t = \tau$. After the impact, for $t > \tau$, the manipulator end effector moves along the wall.*

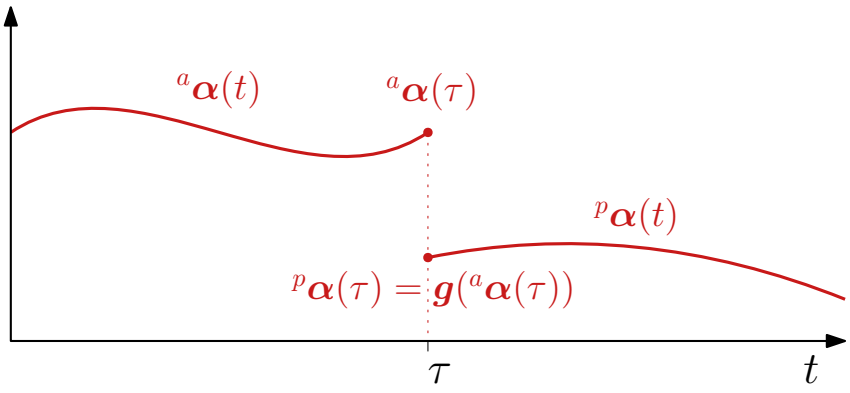


Figure 3.2: *The reference state trajectory $\boldsymbol{\alpha}(t)$ associated with the motion task depicted in Figure 3.1. At impact time τ, the post-impact reference state ${}^{p}\boldsymbol{\alpha}(\tau)$ is compatible with the state jump associated with ante-impact state ${}^{a}\boldsymbol{\alpha}(\tau)$.*

Reference state trajectory $\boldsymbol{\alpha}(t)$ is divided in the ante-impact trajectory ${}^{a}\boldsymbol{\alpha}(t)$ and the post-impact trajectory ${}^{p}\boldsymbol{\alpha}(t)$, where

$$\boldsymbol{\alpha}(t) = \begin{cases} {}^{a}\boldsymbol{\alpha}(t), & t \in [t_0, \tau], \\ {}^{p}\boldsymbol{\alpha}(t), & t \in (\tau, t_f], \end{cases} \tag{3.2}$$

with t_0 the initial time, τ the intended impact time and t_f the final time. The post-impact reference state trajectory is compatible with the impact dynamics, in the sense that

$${}^{p}\boldsymbol{\alpha}(\tau) = \boldsymbol{g}({}^{a}\boldsymbol{\alpha}(\tau)), \tag{3.3}$$

such that the post-impact trajectory at impact time ${}^{p}\boldsymbol{\alpha}(\tau)$ coincides with the state jump with jump map $\boldsymbol{g}$ from ante-impact state ${}^{a}\boldsymbol{\alpha}(\tau)$.
Task-based QP control, as reviewed in Section 2.3, allows to control a manipulator body to a desired reference task trajectory, defined in Cartesian space. For example, a reference task trajectory can be defined for the position and velocity of the end effector. Let $\boldsymbol{y}_{\text{pos}}(t)$ denote such a reference task trajectory, given by

$$\boldsymbol{y}_{\text{pos}}(t) = \begin{bmatrix} \mathbf{p}^{\text{ref}}(t) \\ \dot{\mathbf{p}}^{\text{ref}}(t) \end{bmatrix}, \tag{3.4}$$

where $\mathbf{p}^{\text{ref}}(t) \in \mathbb{R}^2$ and $\dot{\mathbf{p}}^{\text{ref}}(t) \in \mathbb{R}^2$ denote the reference position and velocity of the end effector, respectively. Using this reference task trajectory, the same end effector positions and velocities can be prescribed, as defined by the reference state trajectory $\boldsymbol{\alpha}(t)$. In this case, the reference task trajectory $\boldsymbol{y}_{\text{pos}}(t)$ corresponds to

$$\boldsymbol{y}_{\text{pos}}(t) = \boldsymbol{h}(\boldsymbol{\alpha}(t)), \tag{3.5}$$

where $\boldsymbol{h}$ denotes forward kinematics, transforming the joint state trajectory $\boldsymbol{\alpha}(t)$ into reference task trajectory $\boldsymbol{y}_{\text{pos}}(t)$ for the end effector. The difference between this reference task trajectory $\boldsymbol{y}_{\text{pos}}(t)$ and reference state trajectory $\boldsymbol{\alpha}(t)$ is that $\boldsymbol{y}_{\text{pos}}(t)$ does not define the reference positions and velocities for each joint in the manipulator. Generally, this also means that a reference task trajectory does not implicitly define a unique reference state trajectory. This is illustrated visually in Figure 3.3, in which the three-link manipulator is shown, tracking the reference task trajectory $\boldsymbol{y}_{\text{pos}}(t)$. Aside from the solution given by reference state trajectory $\boldsymbol{\alpha}(t)$, shown in solid lines, another solution to $\boldsymbol{y}_{\text{pos}}(t)$ is shown in dashed lines. In fact, infinitely many solutions exist, in terms of joint states, for the same reference task trajectory. This is known as self-motion, which is caused by task redundancy in manipulators. This means that the manipulator has more degrees of freedom (DOFs) than required to perform a given task. The manipulator in Figure 3.3 has three DOFs, as it consists of three revolute joints. However, the reference task trajectory $\boldsymbol{y}_{\text{pos}}(t)$ only requires two DOFs to be performed.
In many cases, task redundancy of the manipulator is desired, like in an environment with obstacles. In this case, redundancy allows to choose a solution in which the manipulator does not collide with the obstacles. However, for motion tasks involving impacts, self-motion needs to be carefully taken care of. In case of self-motion, infinitely many state trajectories exist that correspond to the same reference position task trajectory. This means that, in addition to ${}^{a}\boldsymbol{\alpha}(t)$, another ante-impact reference state trajectory ${}^{a}\tilde{\boldsymbol{\alpha}}(t)$ exists that corresponds to the same ante-impact reference task trajectory ${}^{a}\boldsymbol{y}_{\text{pos}}(t)$ in the sense of (3.5). However, the post-impact reference state ${}^{p}\tilde{\boldsymbol{\alpha}}(\tau)$ associated with ${}^{a}\tilde{\boldsymbol{\alpha}}(\tau)$, which is

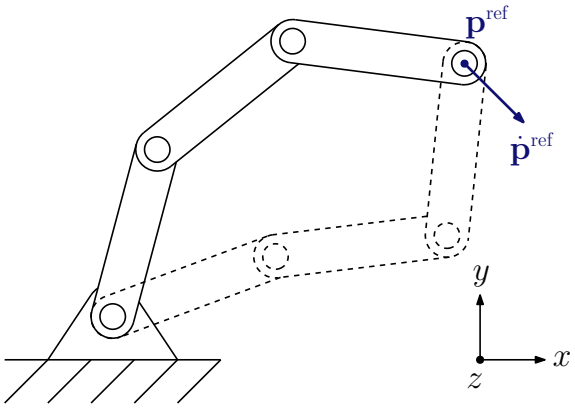


Figure 3.3: *A task redundant, three-link manipulator performing a 2DOF position task for the end effector. In solid lines, the solution is shown corresponding to reference joint state trajectory $\boldsymbol{\alpha}(t)$. The manipulator in dashed lines shows one of the infinite other solutions to the same position task.*

found through (3.3), does generally not correspond to ${}^{p}\boldsymbol{y}_{\text{pos}}(\tau)$ in the sense of (3.5). This is caused by the fact that the jump map in (3.3) strictly depends on the joint state of the manipulator. The result is that infinitely many post-impact task states ${}^{p}\boldsymbol{y}_{\text{pos}}(\tau)$ can be found, that correspond to a single reference ante-impact reference task trajectory ${}^{a}\boldsymbol{y}_{\text{pos}}(t)$. Therefore, it is not possible to define a unique reference position task trajectory $\boldsymbol{y}_{\text{pos}}(t)$ describing a motion task involving impacts. Generally speaking, it is not possible to define a unique reference task trajectory for motions with intentional impacts using a task that requires less DOFs than the manipulator has.

3.1.2 Unique reference task trajectories

In order to define unique reference task trajectories that take jumps caused by intentional impacts into account, the task redundancy in the manipulator has to be removed. For non-redundant manipulators, the number of DOFs required to perform a task is equal to the number of DOFs in the manipulator. This means that the reference task trajectories associated with this task implicitly define a unique reference state trajectory. As a result, ante-impact reference task trajectories can be defined that imply a unique ante-impact reference state trajectory ${}^{a}\boldsymbol{\alpha}(t)$, and through (3.3) a unique post-impact state ${}^{p}\boldsymbol{\alpha}(\tau)$ as well. This eliminates the problem that infinitely many post-impact states ${}^{p}\boldsymbol{\alpha}(\tau)$ can be found, that correspond the same ante-impact reference task trajectories, which was seen in the previous subsection. In this case, post-impact reference task trajectories can be formulated that correspond to the unique post-impact state ${}^{p}\boldsymbol{\alpha}(\tau)$ at $t = \tau$ and therefore account for the state jump caused by the intentional impact.
As an illustrative example, consider again the three-link manipulator in Figure 3.3. In order to remove the task redundancy in this manipulator, a 3DOF task has to be specified. Such a task can be constructed by combining the 2DOF position task, with reference trajectory $\boldsymbol{y}_{\text{pos}}(t)$, and a 1DOF

orientation task, with reference trajectory $\boldsymbol{y}_{\text{ori}}(t)$. The reference task trajectory $\boldsymbol{y}_{\text{ori}}(t)$, is given by

$$\boldsymbol{y}_{\text{ori}}(t) = \begin{bmatrix} \theta^{\text{ref}}(t) \\ \dot{\theta}^{\text{ref}}(t) \end{bmatrix}, \tag{3.6}$$

where $\theta^{\text{ref}}(t)$ denotes the desired angle of the end effector with respect to the inertial x-axis and $\dot{\theta}^{\text{ref}}(t)$ denotes the angular velocity of the end effector. In Figure 3.4, the three-link manipulator is shown again, tracking both $\boldsymbol{y}_{\text{pos}}(t)$ and $\boldsymbol{y}_{\text{ori}}(t)$. In this case, a unique solution is found that satisfies both

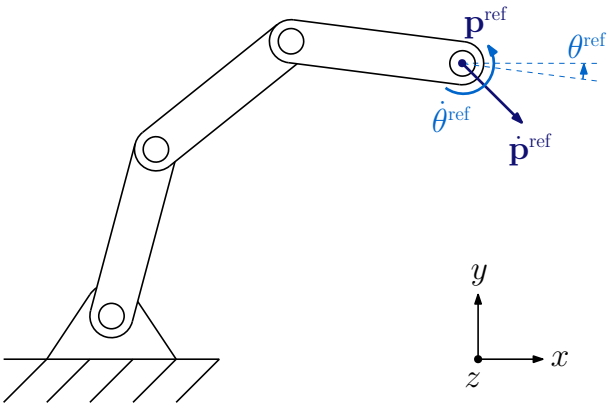


Figure 3.4: *A three-link manipulator performing a combined position and orientation task. Since the minimum required amount of DOFs for the combined tasks equals the amount of DOFs in the manipulator, a unique joint state corresponds to this task.*

reference task trajectories. This means that reference task trajectories $\boldsymbol{y}_{\text{pos}}(t)$ and $\boldsymbol{y}_{\text{ori}}(t)$ implicitly define a reference state trajectory $\boldsymbol{\alpha}(t)$. Therefore, a unique ante-impact reference state trajectory ${}^{a}\boldsymbol{\alpha}(t)$ is implied by defining ante-impact reference task trajectories ${}^{a}\boldsymbol{y}_{\text{pos}}(t)$ and ${}^{a}\boldsymbol{y}_{\text{ori}}(t)$. The post-impact state ${}^{p}\boldsymbol{\alpha}(\tau)$, associated with ${}^{a}\boldsymbol{\alpha}(t)$ through (3.3), is therefore also uniquely associated with ${}^{a}\boldsymbol{y}_{\text{pos}}(t)$ and ${}^{a}\boldsymbol{y}_{\text{ori}}(t)$. This means that post-impact reference task trajectories that correspond to ${}^{p}\boldsymbol{\alpha}(\tau)$ at $t = \tau$, are consistent with ${}^{a}\boldsymbol{y}_{\text{pos}}(t)$ and ${}^{a}\boldsymbol{y}_{\text{ori}}(t)$, and the associated impact dynamics. This shows that it is possible to prescribe a motion with intentional impacts, using reference task trajectories, when enough tasks are used to remove the task redundancy in the manipulator.

Remark 1: It is not required that the same tasks are used after the impact as before the impact. Any tasks can be used, as long as the associated reference task trajectories correspond to the post-impact state ${}^{p}\boldsymbol{\alpha}(\tau)$ at $t = \tau$. Thereby, the post-impact reference task trajectories do not necessarily have to implicitly define the post-impact reference task trajectories, in case it is not intended to make another impact. In this case, task redundancy is allowed after impact. However, in the remainder of this work, the task redundancy is removed after impact as well, such that it is possible to perform another impact, if desired.

Remark 2: For the sake of clarity, a general reference task trajectory $\boldsymbol{y}(t)$ is introduced, which is used in the remainder of this work to represent the reference trajectories for all considered tasks combined.

The general reference task trajectory is given by

$$\boldsymbol{y}(t) = \begin{bmatrix} \mathbf{y}^{\mathrm{ref}}(t) \\ \dot{\mathbf{y}}^{\mathrm{ref}}(t) \end{bmatrix}, \tag{3.7}$$

where $\mathbf{y}^{\mathrm{ref}}(t)$ and $\dot{\mathbf{y}}^{\mathrm{ref}}(t)$ denote the time-varying reference task configuration and velocity, respectively. Note that post-impact reference task trajectory ${}^{p}\boldsymbol{y}(t)$ does not necessarily have to describe the reference task trajectory for the same tasks as the ante-impact reference task trajectory ${}^{a}\boldsymbol{y}(t)$.

3.1.3 Reference spreading for task trajectories

Using the method presented in Section 3.1.2, unique reference task trajectories can be defined that describe motions involving intentional impacts, which can be used in task-based QP control. In order to use the reference spreading methodology in task-based QP control, the reference task trajectories have to be extended about their event-times, similarly to classical reference spreading discussed in Section 2.2. It is desired to apply reference spreading on reference task trajectories, as the tracking of the reference task trajectories using QP control encounters similar challenges as encountered in state-feedback control, discussed in Section 2.2.1. Perturbations can cause the system to make impact before or after the intended impact time. As a result of the mismatch in jump times, the reference trajectories describe a different mode than the mode that the system is in. The task error, which is generally defined as (2.27), can therefore not be used. By extending the reference task trajectories beyond the intended impact time, the system can, at all times, be compared to a reference trajectory that describes the same mode. This is illustrated visually in Figure 3.5 for an extended reference task trajectory $\bar{\boldsymbol{y}}(t)$, depicted in the bottom plot. The extended reference task trajectory implicitly defines

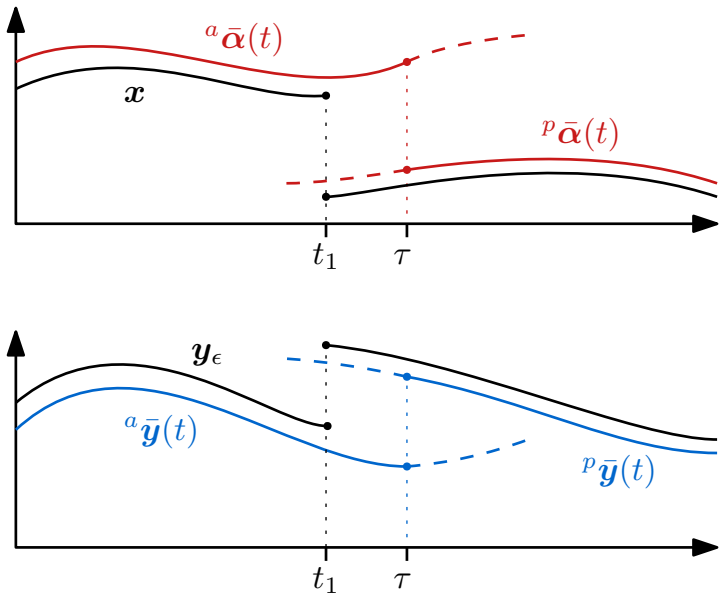


Figure 3.5: *Reference spreading applied on reference task trajectories. The bottom plot shows an extended reference task trajectory $\bar{\boldsymbol{y}}(t)$, which implicitly defines $\bar{\boldsymbol{\alpha}}(t)$ depicted in the top plot. A perturbed state trajectory $\boldsymbol{x}$ and perturbed task trajectory $\boldsymbol{y}_\epsilon$ are depicted in the top and bottom plot respectively.*

the extended reference state trajectory $\bar{\boldsymbol{\alpha}}(t)$. A perturbed state trajectory $\boldsymbol{x}$ is depicted in Figure 3.5 as well. Perturbed task state trajectory $\boldsymbol{y}_\epsilon$ describes the task state corresponding to $\boldsymbol{x}$, associated with reference task trajectory $\bar{\boldsymbol{y}}(t)$. On time interval $t \in [t_1, \tau]$, task error $\boldsymbol{e} = \|\boldsymbol{y}_\epsilon - {}^a\bar{\boldsymbol{y}}(t)\|$ cannot be used, as a jump has occurred in $\boldsymbol{y}_\epsilon$, but not in ${}^a\bar{\boldsymbol{y}}(t)$, which means that both trajectories are in a different mode. This is solved by tracking the extended post-impact reference task trajectory ${}^p\bar{\boldsymbol{y}}(t)$ for $t > t_1$, using task error $\boldsymbol{e} = \|\boldsymbol{y}_\epsilon - {}^p\bar{\boldsymbol{y}}(t)\|$.
In case of simultaneous impacts, the same challenge occurs as detailed in Section 2.2.3. The system can enter a mode that is not specified by the ante- or post-impact reference task trajectories. Similar to the control strategy that is used in Section 2.2.3, a controller has to be specified consisting of three modes: a mode for tracking of the extended ante-impact reference task trajectories, an intermediate mode to deal with unspecified modes and a mode for tracking of the extended post-impact reference task trajectories. The exact definition of these modes in a task-based QP control framework depends on the considered manipulator and application, as different tasks and different constraints are used in the QP problem for different applications. Similarly, the generation of the reference task trajectories depends on the considered manipulator and application. Therefore, in the remainder of this work, a specific manipulator and application are considered, for which task-based reference spreading is demonstrated. This specific manipulator and application are introduced in Section 3.2.

3.2 QP tasks for a dual arm dynamic box-lifting application

Since a general definition of task-based reference spreading is complicated, due to the dependence on the manipulator and application, a specific manipulator and application are considered in the remainder of this work, for which task-based reference spreading is demonstrated. In this section, the considered manipulator and application are introduced. Thereafter, a set of tasks is specified, with which the desired motion can be performed and the task redundancy of the manipulator is removed. By evaluation of the cost function of each task, it is seen how a reference task trajectory can be formulated for this task. The generation of these reference task trajectories is considered in Section 3.3.

3.2.1 Dynamic box-lifting using a dual arm manipulator

In this research, a dual arm manipulator is considered, consisting of two identical, torque controlled 7DOF KUKA LWR robot arms, attached to a rigid torso anchored to the ground. Dual arm manipulators are suitable for grabbing larger objects, which cannot be picked using a single gripper. An example of such an application is grabbing a box from a stockpile. In this research, the specific scenario is considered in which a box is grabbed from the top of the stockpile and moved to another location. In Figure 3.6, the considered scenario is illustrated in the V-REP robot simulator environment, presented in Section 2.5.
The desired motion for this box-lifting application consists of two parts. First, the manipulator has to establish a no-slip surface contact between the end effectors and the box. Thereafter, the manipulator has to move the box to a desired location. This application is a typical example of a motion task which involves simultaneous impacts, as it is desired that contacts are established at nonzero velocity for all contact points on both end effector surfaces simultaneously.
The end effectors of the manipulator consist of a square shaped body, which is rigidly attached to the final link of the manipulator arms in order to enlarge the contact surface between the box and the

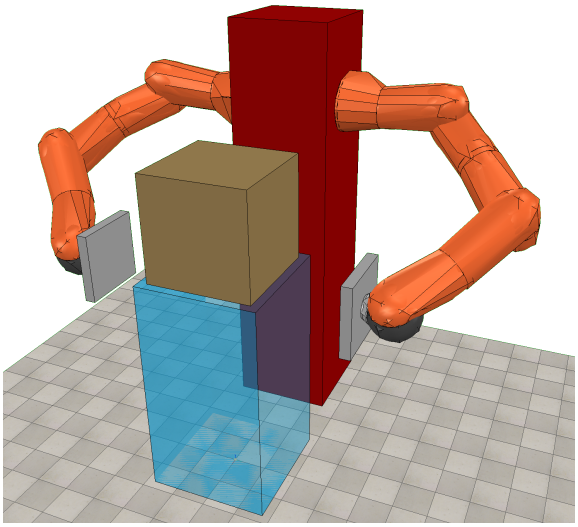


Figure 3.6: *Visualization of a box-picking application using a dual arm manipulator in V-REP. The dual arm manipulator is attached to a rigid torso, depicted in red, which is anchored to the ground. The box is placed on a platform in front of the manipulator and can be moved freely.*

manipulator. As a result, the box is less likely to rotate with respect to the end effectors, which would result in loss of the surface contact.
In order to describe the desired motion for the manipulator arms and the box, three different QP tasks are used. First of all, a position task is used, which allows to specify a possibly time-varying reference position, linear velocity and linear acceleration for a point in Cartesian space. For the considered application, the position task is used to control a point on the end effector surface to the box surface in order to establish contact. When the no-slip contact has been established, all DOFs of the end effectors are constrained by the DOFs of the box. For this reason, a position task for the box is used after the contact has been established, with which the desired position of the box can be prescribed.
Secondly, an orientation task is used. The orientation task can be used to prescribe a possibly time-varying desired orientation, angular velocity and angular acceleration for a body frame. This task is used to align the end effector surfaces with the box surfaces, such that contact is made through simultaneous impact. In addition, the orientation task is used to specify the desired orientation of the box, after the contact has been established.
The position task and orientation task both require three DOFs to be performed. This means that when only the position and orientation task are applied on the 7DOF manipulator arms, the manipulator is task redundant. As discussed in Section 3.1.1, it is not possible to prescribe reference task trajectories with impacts for task redundant manipulators. Conform to the approach discussed in Section 3.1.2, an additional task is specified to remove the task redundancy. For this purpose, a direction task is used, which prescribes a possibly time-varying reference direction, velocity and acceleration for a vector on one of the bodies of the manipulator arms. In this case, the direction task cannot be used on a vector on the end effector, as the orientation of the end effector body is already prescribed by the orientation task. Therefore, the direction task is used for a vector on the fourth link of the manipulator arms, both before and after the contact has been established.
In the following subsections, the cost function of each QP task is specified. By evaluating the cost

functions, it is shown how the reference task trajectories that are associated with each task have to be defined.

3.2.2 Position task

The position task is used to control the Cartesian position of a point $\mathbf{p}$, which is rigidly attached to a body with coordinate frame B. In Figure 3.7, this is illustrated visually for a manipulator arm, where $\mathbf{p}$ is a point on the end effector surface and frame B is the body frame of the end effector. The reference position is given by $\mathbf{p}^{\text{ref}}$.

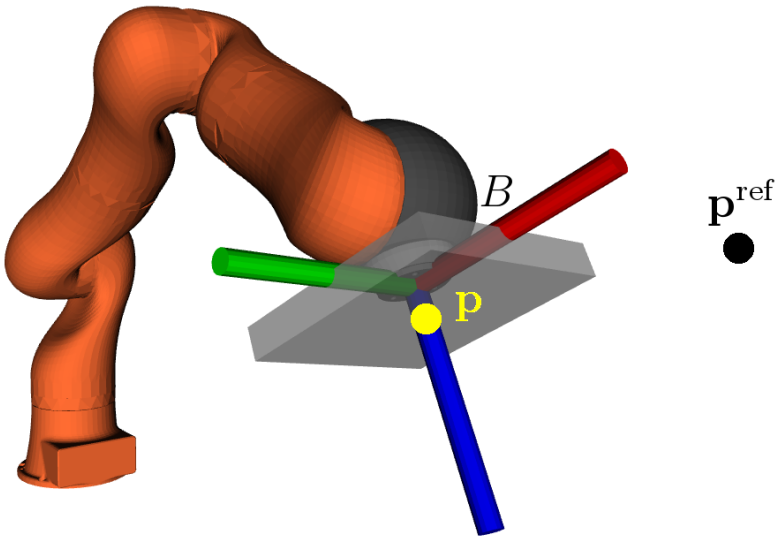


Figure 3.7: *Visualization of the position task for a manipulator arm. The goal of the position task is to control a point* $\mathbf{p}$ *to a reference position* $\mathbf{p}^{\text{ref}}$*, where* $\mathbf{p}$ *is rigidly attached to a body with coordinate frame* B*.*

The cost function for the position task is formulated in quadratic form, such that it fits the QP problem discussed in Section 2.3. This cost function, denoted by $\boldsymbol{E}_{\text{pos}}$, is given by

$$\boldsymbol{E}_{\text{pos}} = \frac{1}{2}\left\| {}^{W}\ddot{\mathbf{p}} - \left({}^{W}\ddot{\mathbf{p}}^{\text{ref}} + \boldsymbol{K}_d({}^{W}\dot{\mathbf{p}}^{\text{ref}} - {}^{W}\dot{\mathbf{p}}) + \boldsymbol{K}_p({}^{W}\mathbf{p}^{\text{ref}} - {}^{W}\mathbf{p}) \right) \right\|^2, \tag{3.8}$$

where ${}^{W}\mathbf{p} \in \mathbb{R}^3$ denotes the current position of point $\mathbf{p}$ expressed in an inertial frame W and ${}^{W}\mathbf{p}^{\text{ref}} \in \mathbb{R}^3$ the reference position for this point, also expressed in the inertial frame. Here, $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are positive-definite diagonal matrices, denoting the proportional and derivative control gains, respectively. The current position of the point is given by

$${}^{W}\mathbf{p} = {}^{W}\mathbf{o}_B + {}^{W}\mathbf{R}_B\, {}^{B}\mathbf{p}, \tag{3.9}$$

where ${}^{W}\mathbf{o}_B$ denotes the coordinates of the origin of frame B, expressed in inertial frame W, ${}^{W}\mathbf{R}_B$ the orientation of frame B with respect to the inertial frame and ${}^{B}\mathbf{p}$ the position of point $\mathbf{p}$ expressed in

body frame B. The time derivative of the position of point $\mathbf{p}$ is computed by

$$\begin{aligned} {}^{W}\dot{\mathbf{p}} &= {}^{W}\dot{\mathbf{o}}_B + {}^{W}\dot{\mathbf{R}}_B {}^{B}\mathbf{p}, \\ &= {}^{W}\dot{\mathbf{o}}_B + {}^{W}\mathbf{R}_B \left({}^{B}\boldsymbol{\omega}_{W,B} \times {}^{B}\mathbf{p}\right), \\ &= {}^{W}\mathbf{R}_B \left({}^{B}\boldsymbol{v}_{W,B} + {}^{B}\boldsymbol{\omega}_{W,B} \times {}^{B}\mathbf{p}\right). \end{aligned} \tag{3.10}$$

Note that, since the point is rigidly attached to the body, ${}^{B}\dot{\mathbf{p}} = 0$. The linear and angular velocity of the body frame B with respect to the inertial frame, denoted with ${}^{B}\boldsymbol{v}_{W,B}$ and ${}^{B}\boldsymbol{\omega}_{W,B}$, respectively, can be found by

$${}^{B}\mathbf{v}_{W,B} = \begin{bmatrix} {}^{B}\boldsymbol{v}_{W,B} \\ {}^{B}\boldsymbol{\omega}_{W,B} \end{bmatrix} = {}^{B}\mathbf{J}_{W,B}\dot{\mathbf{q}}, \tag{3.11}$$

with ${}^{B}\mathbf{J}_{W,B}$ the Jacobian relating the velocity of the body frame with respect to the inertial frame, expressed in the body frame.
The second order time derivative of ${}^{W}\mathbf{p}$ can be found by differentiation of (3.10), which results in

$$\begin{aligned} {}^{W}\ddot{\mathbf{p}} &= {}^{W}\dot{\mathbf{R}}_B \left({}^{B}\boldsymbol{v}_{W,B} + {}^{B}\boldsymbol{\omega}_{W,B} \times {}^{B}\mathbf{p}\right) + {}^{W}\mathbf{R}_B \left({}^{B}\dot{\boldsymbol{v}}_{W,B} + {}^{B}\dot{\boldsymbol{\omega}}_{W,B} \times {}^{B}\mathbf{p}\right), \\ &= {}^{W}\mathbf{R}_B \left({}^{B}\boldsymbol{\omega}_{W,B} \times \left({}^{B}\boldsymbol{v}_{W,B} + {}^{B}\boldsymbol{\omega}_{W,B} \times {}^{B}\mathbf{p}\right)\right) + {}^{W}\mathbf{R}_B \left({}^{B}\dot{\boldsymbol{v}}_{W,B} + {}^{B}\dot{\boldsymbol{\omega}}_{W,B} \times {}^{B}\mathbf{p}\right), \end{aligned} \tag{3.12}$$

where

$${}^{B}\dot{\mathbf{v}}_{W,B} = \begin{bmatrix} {}^{B}\dot{\boldsymbol{v}}_{W,B} \\ {}^{B}\dot{\boldsymbol{\omega}}_{W,B} \end{bmatrix} = {}^{B}\mathbf{J}_{W,B}\ddot{\mathbf{q}} + {}^{B}\dot{\mathbf{J}}_{W,B}\dot{\mathbf{q}}. \tag{3.13}$$

Let $\bar{\boldsymbol{y}}_{\text{pos}}(t)$ denote the extended reference task trajectory for the position task, such that

$$\bar{\boldsymbol{y}}_{\text{pos}}(t) = \begin{bmatrix} \bar{\mathbf{p}}^{\text{ref}}(t) \\ \dot{\bar{\mathbf{p}}}^{\text{ref}}(t) \end{bmatrix} \in \mathbb{R}^6, \tag{3.14}$$

where $\bar{\mathbf{p}}^{\text{ref}}(t)$ represents the trajectory of reference positions ${}^{W}\mathbf{p}^{\text{ref}} \in \mathbb{R}^3$, and $\dot{\bar{\mathbf{p}}}^{\text{ref}}(t)$ represents the trajectory of reference velocities ${}^{W}\dot{\mathbf{p}}^{\text{ref}} \in \mathbb{R}^3$, both expressed with respect to the inertial frame W. In addition, a reference acceleration trajectory $\ddot{\bar{\mathbf{p}}}^{\text{ref}}(t)$ can be defined, which prescribes the reference acceleration ${}^{W}\ddot{\mathbf{p}}^{\text{ref}} \in \mathbb{R}^3$ at each point in time. The acceleration trajectory represents the feedforward.

3.2.3 Orientation task

The orientation task is used to control the orientation of frame B, denoted by $[B]$, to a possibly time-varying desired orientation frame $[D]$. In Figure 3.8, this is illustrated for a manipulator arm, where $[B]$ denotes the orientation of the end effector body frame. The quadratic cost function for the orientation task is denoted by $\boldsymbol{E}_{\text{ori}}$. The derivation of this cost function, based on the PD controller on SO(3) proposed in [33], can be found in Appendix A. Here, only the cost function itself is given, which is formulated as

$$\begin{aligned} \boldsymbol{E}_{\text{ori}} = \frac{1}{2}\Big|\Big|{}^{W}\dot{\boldsymbol{\omega}}_{W,B} - \big({}^{W}\dot{\boldsymbol{\omega}}_{W,D} - {}^{W}\boldsymbol{\omega}_{W,B} \times \big({}^{W}\boldsymbol{\omega}_{W,D} - {}^{W}\boldsymbol{\omega}_{W,B}\big) \\ + k_d \big({}^{W}\boldsymbol{\omega}_{W,D} - {}^{W}\boldsymbol{\omega}_{W,B}\big) + k_p {}^{W}\mathbf{R}_B(\log({}^{B}\mathbf{R}_D))^{\vee}\big)\Big|\Big|^2, \end{aligned} \tag{3.15}$$

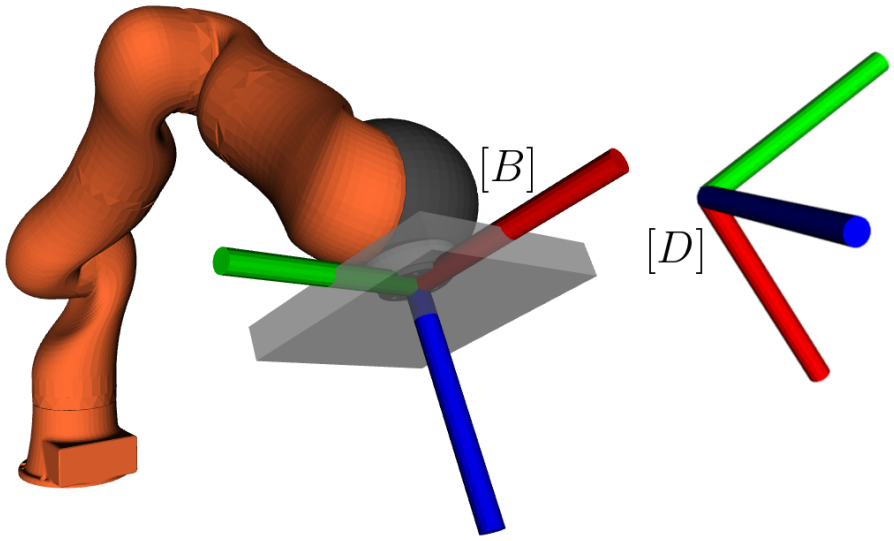


Figure 3.8: *Visualization of the orientation task for a manipulator arm. The goal of the orientation task is to control the body frame orientation* $[B]$ *to reference orientation given by* $[D]$*. Note that the orientation task is independent of the position origin of frames* B *and* D*, as only the orientation is controlled.*

where ${}^{W}\boldsymbol{\omega}_{W,B} \in \mathbb{R}^3$ and ${}^{W}\boldsymbol{\omega}_{W,D} \in \mathbb{R}^3$ denote the angular velocity of respectively frame B and D with respect to inertial frame W, expressed in the inertial frame. In this case, $k_p, k_d \in \mathbb{R}$ denote scalar proportional and derivative gains. The angular velocity and its time derivative are computed with

$$ {}^{W}\boldsymbol{\omega}_{W,B} = {}^{W}(\mathbf{J}_{\omega})_{W,B}(\mathbf{q})\,\dot{\mathbf{q}}, \tag{3.16} $$

$$ {}^{W}\dot{\boldsymbol{\omega}}_{W,B} = {}^{W}(\dot{\mathbf{J}}_{\omega})_{W,B}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + {}^{W}(\mathbf{J}_{\omega})_{W,B}(\mathbf{q})\,\ddot{\mathbf{q}}, \tag{3.17} $$

where ${}^{W}(\mathbf{J}_{\omega})_{W,B}$ represents the angular part of the Jacobian, relating the angular velocity of the body frame with respect to the inertial frame, expressed in the inertial frame. The error between the orientations of frame B and D is defined as the matrix logarithm of the orientation of frame D with respect to frame B, as discussed in [33]. This orientation, denoted by ${}^{B}\mathbf{R}_{D}$, is found by

$$ {}^{B}\mathbf{R}_{D} = {}^{W}\mathbf{R}_{B}^{T}\,{}^{W}\mathbf{R}_{D}. \tag{3.18} $$

Let $\bar{\boldsymbol{y}}_{\text{ori}}(t)$ denote the extended reference task trajectory for the orientation task, which is defined as

$$ \bar{\boldsymbol{y}}_{\text{ori}}(t) = \left(\bar{\mathbf{R}}^{\text{ref}}(t), \bar{\boldsymbol{\omega}}^{\text{ref}}(t)\right) \in \left(\text{SO}(3), \mathbb{R}^3\right), \tag{3.19} $$

where $\bar{\mathbf{R}}^{\text{ref}}(t)$ is the time-varying orientation of the desired frame with respect to the inertial frame ${}^{W}\mathbf{R}_{D} \in \text{SO}(3)$ and $\bar{\boldsymbol{\omega}}^{\text{ref}}(t)$ the reference angular velocity of the desired frame ${}^{W}\boldsymbol{\omega}_{W,D} \in \mathbb{R}^3$. The angular acceleration reference trajectory $\dot{\bar{\boldsymbol{\omega}}}^{\text{ref}}(t)$ is given in terms of ${}^{W}\dot{\boldsymbol{\omega}}_{W,D} \in \mathbb{R}^3$.

3.2.4 Direction task

Using the direction task, a time-varying reference direction can be prescribed for a vector $\mathbf{u} \in S^2$, with S^2 the unit sphere, which is rigidly attached to a body. For the considered application, the direction task is applied on a vector on the fourth link of the manipulator arms. In Figure 3.9 the direction task is visualized for a vector $\mathbf{u}$ on this link, with the desired direction given by $\mathbf{u}^{\text{ref}} \in S^2$.

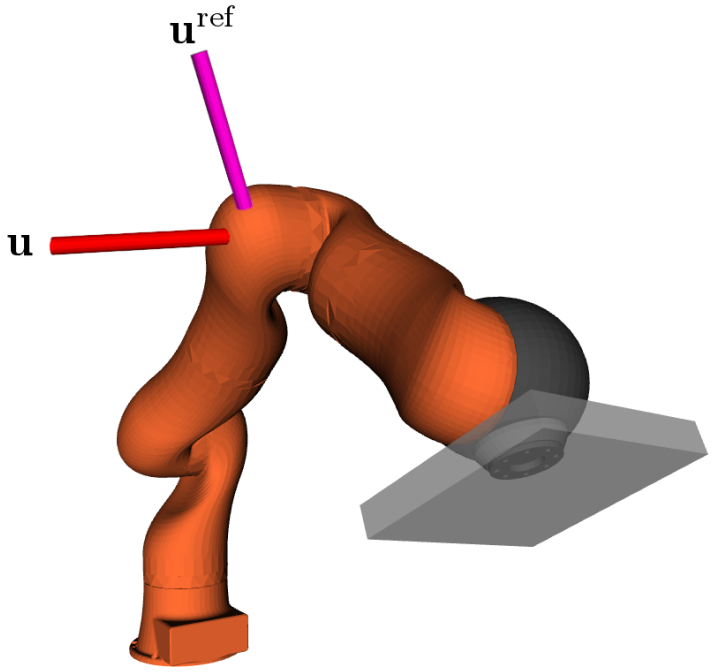


Figure 3.9: *Visualization of the direction task. The goal of the direction task is to control a vector* $\mathbf{u}$ *to a reference direction vector* $\mathbf{u}^{\text{ref}}$.

The quadratic cost function for the direction task $\boldsymbol{E}_{\text{vec}}$ is formulated as

$$\boldsymbol{E}_{\text{vec}} = \frac{1}{2} \left\| {}^W\ddot{\mathbf{u}} - \left({}^W\ddot{\mathbf{u}}^{\text{ref}} + k_d({}^W\dot{\mathbf{u}}^{\text{ref}} - {}^W\dot{\mathbf{u}}) + k_p({}^W\mathbf{u}^{\text{ref}} - {}^W\mathbf{u}) \right) \right\|^2, \tag{3.20}$$

where ${}^W\mathbf{u}$ denotes the vector to be controlled, expressed in the world frame. This vector is given by

$${}^W\mathbf{u} = {}^W\mathbf{R}_B {}^B\mathbf{u}, \tag{3.21}$$

with ${}^B\mathbf{u}$ the vector expressed in body frame B, which, in this case, represents the body frame of the fourth link. Assuming that ${}^B\dot{\mathbf{u}} = 0$, the first and second order derivatives of ${}^B\mathbf{u}$ are found by

$${}^W\dot{\mathbf{u}} = {}^W\mathbf{R}_B \left({}^B\boldsymbol{\omega}_{W,B} \times {}^B\mathbf{u} \right), \tag{3.22}$$

$${}^W\ddot{\mathbf{u}} = {}^W\mathbf{R}_B \left({}^B\boldsymbol{\omega}_{W,B} \times \left({}^B\boldsymbol{\omega}_{W,B} \times {}^B\mathbf{u} \right) + {}^B\dot{\boldsymbol{\omega}}_{W,B} \times {}^B\mathbf{u} \right), \tag{3.23}$$

where the angular velocities and accelerations are found by (3.11) and (3.13), respectively.
Let the extended reference task trajectory of the direction task be denoted by $\bar{\boldsymbol{y}}_{\text{vec}}(t)$, such that

$$\bar{\boldsymbol{y}}_{\text{vec}}(t) = \begin{bmatrix} \bar{\mathbf{u}}^{\text{ref}}(t) \\ \dot{\bar{\mathbf{u}}}^{\text{ref}}(t) \end{bmatrix} \in \mathbb{R}^6, \tag{3.24}$$

where $\bar{\mathbf{u}}^{\text{ref}}(t)$ denotes the trajectory of direction vectors ${}^{W}\mathbf{u}^{\text{ref}} \in S^2$ and $\dot{\bar{\mathbf{u}}}^{\text{ref}}(t)$ denotes the trajectory of reference velocities ${}^{W}\dot{\mathbf{u}}^{\text{ref}} \in \mathbb{R}^3$, both expressed with respect to the inertial frame. Note that reference velocity $\dot{\bar{\mathbf{u}}}^{\text{ref}}(t)$ has to be orthogonal to the reference direction vector $\bar{\mathbf{u}}^{\text{ref}}(t)$. In addition, the reference acceleration trajectory $\ddot{\bar{\mathbf{u}}}^{\text{ref}}(t)$ is defined in terms of ${}^{W}\ddot{\mathbf{u}}^{\text{ref}} \in \mathbb{R}^3$, which has to be orthogonal to $\bar{\mathbf{u}}^{\text{ref}}(t)$ as well.

3.3 Task trajectory generation

In the previous section, it is shown how reference trajectories are defined for the position, orientation, and direction task, which are used to describe the desired motion for the dual arm dynamic box-lifting application. In this section, an approach is proposed for the generation of the extended reference trajectories for each of these tasks. Using this approach, conditions can be specified that the trajectories have to satisfy, which allows to generate trajectories that are compatible with the impact dynamics, as discussed in Section 3.1.2. The specific conditions that are chosen for the reference task trajectories and their extensions are similar for all considered tasks. Therefore, these are listed and explained for a general reference task trajectory, first. Thereafter, the De Casteljau interpolation algorithm, as reviewed in Section 2.6, is used to compute extended reference task trajectories for the position, orientation, and direction task that satisfy the specified conditions. The implementation of the De Casteljau algorithm depends on the considered task and is therefore discussed separately for each task. In Chapter 4, the trajectory generation procedures are demonstrated in a numerical simulation study on the dual arm dynamic box-lifting application.

3.3.1 Conditions on extended reference task trajectories

It is chosen to generate the extended task trajectories on the basis of three conditions on both the extended ante- and post-impact reference task trajectory. In Figure 3.10, these conditions are depicted for general extended task trajectory $\bar{\boldsymbol{y}}(t)$.

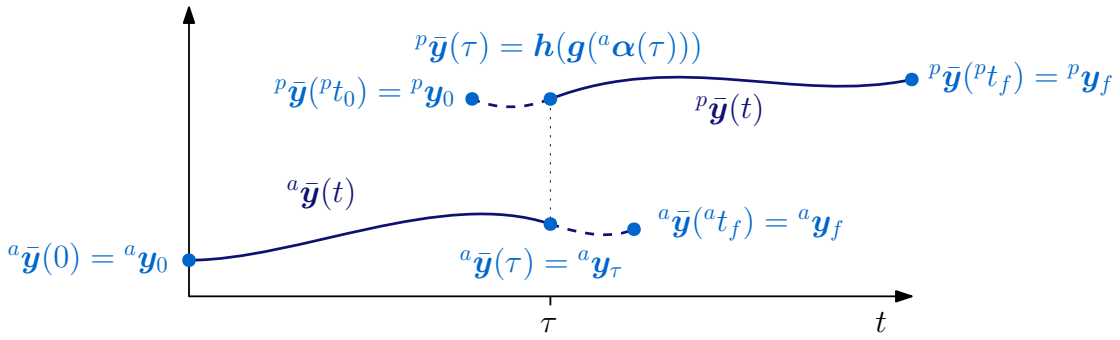


Figure 3.10: *The conditions on the reference task trajectory, shown for the general extended reference task trajectory* $\bar{\boldsymbol{y}}(t)$.

The conditions on the extended ante- and post-impact reference task trajectories are formulated as:

- ${}^{a}\bar{\boldsymbol{y}}(0) = {}^{a}\boldsymbol{y}_0$, which allows to specify the initial conditions of the ante-impact reference task trajectory. Here, ${}^{a}\boldsymbol{y}_0$ denotes the desired initial task state of the manipulator.

- ${}^a\bar{\boldsymbol{y}}(\tau) = {}^a\boldsymbol{y}_\tau$, which sets conditions on the ante-impact reference task trajectories at the intended impact time, where ${}^a\boldsymbol{y}_\tau$ is the desired task state at $t = \tau$. This allows to specify how the impact should be made. For example, the orientation of the end effectors can be specified, which is required in order to achieve a simultaneous impact.
- ${}^a\bar{\boldsymbol{y}}({}^at_f) = {}^a\boldsymbol{y}_f$, which is used to extend the ante-impact reference task trajectories beyond the intended impact time. Here, ${}^a\boldsymbol{y}_f$ denotes an arbitrary final task state and ${}^at_f$ the final time of the ante-impact reference trajectories. The final time should be chosen sufficiently large, in order for the extended ante-impact reference task trajectories to be able to account for all perturbations.
- ${}^p\bar{\boldsymbol{y}}({}^pt_0) = {}^p\boldsymbol{y}_0$, which is used to specify the backward extension of the reference task trajectories before $t = \tau$, with ${}^p\boldsymbol{y}_0$ the arbitrary initial conditions on the extended post-impact reference task trajectories. The initial time ${}^pt_0$ should be chosen sufficiently small, such that all perturbations are accounted for by the extended post-impact reference task trajectories.
- ${}^p\bar{\boldsymbol{y}}(\tau) = \boldsymbol{h}(\boldsymbol{g}({}^a\boldsymbol{\alpha}(\tau)))$, which implies that the post-impact reference task trajectories have to be compatible with the ante-impact reference task trajectories and the associated impact dynamics, as discussed in Section 3.1.2. Here, ${}^a\boldsymbol{\alpha}(t)$ denotes the ante-impact reference state trajectory associated with ${}^a\bar{\boldsymbol{y}}(t)$, $\boldsymbol{g}$ the jump map and $\boldsymbol{h}$ the forward kinematics transforming the reference state trajectory $\boldsymbol{\alpha}(t)$ into reference task trajectory $\bar{\boldsymbol{y}}(t)$.
- ${}^p\bar{\boldsymbol{y}}({}^pt_f) = {}^p\boldsymbol{y}_f$, which allows to set desired final conditions on the reference task trajectories. Here, ${}^p\boldsymbol{y}_f$ and ${}^pt_f$ denote the desired final task state and time of the post-impact trajectory, respectively.

In the following subsections, the task-specific trajectory generation procedures are presented, which are used to generate extended reference task trajectories that satisfy these conditions.

3.3.2 Position task trajectory generation

As detailed in Section 3.2.1, a position task is used to describe the desired motion of the manipulator end effectors before impact. For this purpose, an extended ante-impact reference task trajectory ${}^a\bar{\boldsymbol{y}}_{\text{pos}}(t)$ has to be generated for both manipulator arms. After impact, the position task is used to prescribe the desired motion of the box, for which an extended post-impact reference task trajectory ${}^p\bar{\boldsymbol{y}}_{\text{pos}}(t)$ has to be generated. Both the ante- and post-impact reference position task trajectories are subject to conditions on the reference position and velocity at the initial time, at impact time and at the final time of the trajectory, as explained in Section 3.3.1. The ante- and post-impact reference task trajectories can therefore be generated using the same procedure. Consider a general reference position trajectory $\mathbf{p}^{\text{ref}}(t) \in \mathbb{R}^3$, which could represent either the extended ante- or post-impact position task trajectory. The conditions on $\mathbf{p}^{\text{ref}}(t)$ are given by

$$\mathbf{p}^{\text{ref}}(t_0) = \mathbf{p}_0, \tag{3.25}$$

$$\mathbf{p}^{\text{ref}}(\tau) = \mathbf{p}_\tau, \tag{3.26}$$

$$\mathbf{p}^{\text{ref}}(t_f) = \mathbf{p}_f, \tag{3.27}$$

where $\mathbf{p}_0$, $\mathbf{p}_\tau$ and $\mathbf{p}_f$, denote the conditions on the position at initial time, impact time and final time, respectively. Similarly, conditions are specified on the velocity reference trajectory $\dot{\mathbf{p}}^{\text{ref}}(t) \in \mathbb{R}^3$, given

by

$$\dot{\mathbf{p}}^{\text{ref}}(t_0) = \dot{\mathbf{p}}_0, \tag{3.28}$$

$$\dot{\mathbf{p}}^{\text{ref}}(\tau) = \dot{\mathbf{p}}_\tau, \tag{3.29}$$

$$\dot{\mathbf{p}}^{\text{ref}}(t_f) = \dot{\mathbf{p}}_f, \tag{3.30}$$

where $\dot{\mathbf{p}}_0$, $\dot{\mathbf{p}}_\tau$ and $\dot{\mathbf{p}}_f$ denote the conditions on the velocity at initial time, impact time and final time, respectively.
Using the De Casteljau algorithm on $\mathbb{R}^3$, $\mathcal{C}^2$ smooth trajectories can be generated, which take into account boundary conditions at both ends of the trajectory. Since, in this case, trajectory has to satisfy conditions at three time instances, the De Casteljau algorithm is used to compute two segments of the trajectory. One segment is computed for the time interval $t \in [t_0, \tau]$, which satisfies the conditions at $t = t_0$ and $t = \tau$ and one segment is computed for $t \in [\tau, t_f]$, which satisfies the conditions at $t = \tau$ and $t = t_f$. Both segments are combined to obtain the full trajectory. In order for the full trajectory to remain $\mathcal{C}^2$ smooth over the entire interval $t \in [t_0, t_f]$, the acceleration at impact time should be equal for both segments. Therefore, an additional condition is formulated, given by

$$\ddot{\mathbf{p}}^{\text{ref}}(\tau) = \ddot{\mathbf{p}}_\tau, \tag{3.31}$$

with $\ddot{\mathbf{p}}_\tau$ the desired acceleration at impact time. In order to satisfy all conditions, a minimum of five control points has to be used in the De Casteljau algorithm, for each segment. The conditions are taken into account by choosing the control points in a specific way. The derivation of the choices for these control points can be found in Appendix B. Here, only the control points themselves are given. The control points for the first segment, ranging in $t \in [t_0, \tau]$, are denoted by $C_i \in \mathbb{R}^3$ and for the second segment, ranging in $t \in [\tau, t_f]$, by $C_i^* \in \mathbb{R}^3$ for $i \in \{1, 2, \ldots, 5\}$.
Using the first and last control point, the initial and final position of the trajectory can be specified, in order to satisfy (3.25) - (3.27). For the first segment, these are therefore chosen as

$$C_1 = \mathbf{p}_0, \tag{3.32}$$

$$C_5 = \mathbf{p}_\tau, \tag{3.33}$$

and for the second segment as

$$C_1^* = \mathbf{p}_\tau, \tag{3.34}$$

$$C_5^* = \mathbf{p}_f. \tag{3.35}$$

The initial and final velocity of each segment, given by (3.28) - (3.30), can be specified by choosing the second and fourth control point respectively, such that

$$C_2 = C_1 + \frac{1}{4}\dot{\mathbf{p}}_0(\tau - t_0), \tag{3.36}$$

$$C_4 = C_5 - \frac{1}{4}\dot{\mathbf{p}}_\tau(\tau - t_0), \tag{3.37}$$

for the first segment and

$$C_2^* = C_1^* + \frac{1}{4}\dot{\mathbf{p}}_\tau(t_f - \tau), \tag{3.38}$$

$$C_4^* = C_5^* - \frac{1}{4}\dot{\mathbf{p}}_f(t_f - \tau), \tag{3.39}$$

for the second segment. In order to satisfy (3.31), the final acceleration of the first segment can be set by choosing the third control point equal to

$$C_3 = \frac{1}{12}(\tau - t_0)^2 \ddot{\mathbf{p}}_\tau - C_5 + 2C_4. \tag{3.40}$$

The initial acceleration of the second segment, is specified by choosing the third control point equal to

$$C_3^* = \frac{1}{12}(t_f - \tau)^2 \ddot{\mathbf{p}}_\tau - C_1^* + 2C_2^*. \tag{3.41}$$

Using these control points and the procedure presented in Section 2.6, a position reference trajectory $\mathbf{p}^{\text{ref}}(t)$ can be computed. In order to compute the interpolation between consecutive control points, the De Casteljau algorithm on $\mathbb{R}^3$ uses the function given by

$$P_i(\lambda) = C_i + (C_{i+1} - C_i)\lambda. \tag{3.42}$$

The first and second order time derivative of reference trajectory $\mathbf{p}^{\text{ref}}(t)$ are used as the reference velocity trajectory $\dot{\mathbf{p}}^{\text{ref}}(t)$ and acceleration trajectory $\ddot{\mathbf{p}}^{\text{ref}}(t)$. As detailed in Section 3.2.2, $\mathbf{p}^{\text{ref}}(t)$, $\dot{\mathbf{p}}^{\text{ref}}(t)$ and $\ddot{\mathbf{p}}^{\text{ref}}(t)$ have to be defined in terms of ${}^W\mathbf{p}^{\text{ref}}$, ${}^W\dot{\mathbf{p}}^{\text{ref}}$ and ${}^W\ddot{\mathbf{p}}^{\text{ref}}$, respectively. Therefore, the conditions (3.25) - (3.31) have to be specified in the same way.
As an illustrative example, in Figure 3.11, a reference position trajectory is shown for arbitrary boundary conditions. In blue, the control points are shown that prescribe the positions at the boundaries, as given by (3.32) - (3.35). It can be seen that the trajectory crosses these control points. The red control points, which correspond to (3.36) - (3.41), prescribe the velocities and accelerations at the boundaries. The trajectory is $\mathcal{C}^2$ smooth at the boundary position given by C_5 and C_1^* due to the constraint on the velocity imposed by C_4 and C_2^* and the constraint on the acceleration imposed by C_3 and C_3^*.

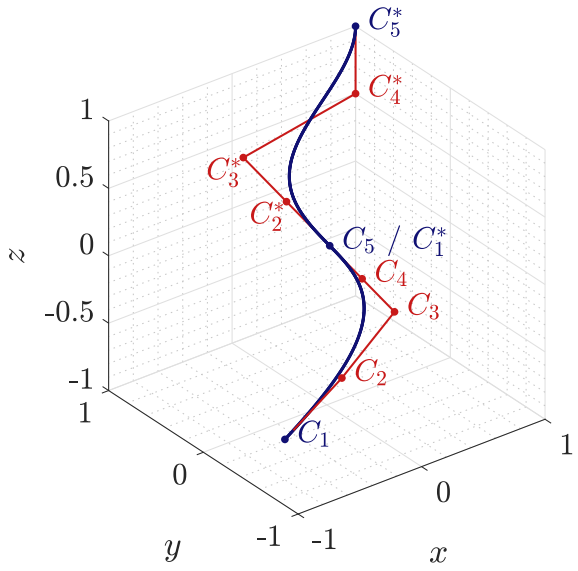


Figure 3.11: *An example position trajectory, consisting of two segments computed using the De Casteljau algorithm on $\mathbb{R}^3$. The blue control points define the position at the boundaries of the segments. Using the red control points, the velocities and accelerations at the boundaries are specified.*

3.3.3 Orientation task trajectory generation

The orientation task is used to prescribe the desired orientation of the manipulator end effectors before impact and the desired orientation of the box after impact, as explained in Section 3.2.1. For this purpose, an extended ante-impact reference task trajectory ${}^{a}\bar{\boldsymbol{y}}_{\text{ori}}(t)$ has to be generated for both manipulator arms and an extended post-impact reference task trajectory ${}^{p}\bar{\boldsymbol{y}}_{\text{ori}}(t)$ has to be generated for the box. A similar approach as for the position task trajectory is used for generating the ante- and post-impact orientation task trajectories. Consider the reference orientation trajectory $\mathbf{R}^{\text{ref}}(t) \in \text{SO}(3)$ representing either an extended ante- or post-impact trajectory. The conditions on this trajectory are given by

$$\mathbf{R}^{\text{ref}}(t_0) = \mathbf{R}_0, \tag{3.43}$$
$$\mathbf{R}^{\text{ref}}(\tau) = \mathbf{R}_\tau, \tag{3.44}$$
$$\mathbf{R}^{\text{ref}}(t_f) = \mathbf{R}_f, \tag{3.45}$$

where $\mathbf{R}_0$, $\mathbf{R}_\tau$ and $\mathbf{R}_f$ denote the conditions on the orientation at the initial, impact and final time. The conditions on the angular velocity reference trajectory $\boldsymbol{\omega}^{\text{ref}}(t) \in \mathbb{R}^3$ are given by

$$\boldsymbol{\omega}^{\text{ref}}(t_0) = \boldsymbol{\omega}_0, \tag{3.46}$$
$$\boldsymbol{\omega}^{\text{ref}}(\tau) = \boldsymbol{\omega}_\tau, \tag{3.47}$$
$$\boldsymbol{\omega}^{\text{ref}}(t_f) = \boldsymbol{\omega}_f, \tag{3.48}$$

with $\boldsymbol{\omega}_0$, $\boldsymbol{\omega}_\tau$ and $\boldsymbol{\omega}_f$ the conditions on the angular velocity at the initial, impact and final time.
Similar to the position task trajectory, the orientation task trajectory is obtained by combining two segments, ranging from $t \in [t_0, t_\tau]$ and $t \in [t_\tau, t_f]$. Both segments are computed using the De Casteljau algorithm on SO(3), as proposed in [34]. An additional condition is introduced on the angular acceleration at impact time, in order for the trajectory to be $\mathcal{C}^2$ smooth on the entire time interval $t \in [t_0, t_f]$. The De Casteljau algorithm on SO(3) only allows to set the angular acceleration equal to zero, so the condition is formulated as

$$\dot{\boldsymbol{\omega}}^{\text{ref}}(\tau) = \mathbf{0}. \tag{3.49}$$

Five control points are used to compute a trajectory that satisfies all conditions. The control points are denoted by $C_i \in \text{SO}(3)$ and $C_i^* \in \text{SO}(3)$ for $i \in \{1, 2, \ldots, 5\}$, for the first and second segment respectively. For the derivation of the choices of these control points, the reader is referred to [34]. The first and last control point of each segment are chosen equal to the desired initial and final orientation of this segment, in order to satisfy (3.43) - (3.45). For the first segment this results in

$$C_1 = \mathbf{R}_0, \tag{3.50}$$
$$C_5 = \mathbf{R}_\tau, \tag{3.51}$$

and for the second segment in

$$C_1^* = \mathbf{R}_\tau, \tag{3.52}$$
$$C_5^* = \mathbf{R}_f. \tag{3.53}$$

The initial and final angular velocities of the segments, given by (3.46) - (3.48), are specified using the second and fourth control point. These are set equal to

$$C_2 = \exp\left(\frac{1}{4}(\tau - t_0)\boldsymbol{\omega}_0^\wedge\right) C_1, \tag{3.54}$$

$$C_4 = \exp\left(-\frac{1}{4}(\tau - t_0)\boldsymbol{\omega}_\tau^\wedge\right) C_5, \tag{3.55}$$

for the first segment, where exp denotes the matrix exponential. For the second segment, the control points are

$$C_2^* = \exp\left(\frac{1}{4}(t_f - \tau)\boldsymbol{\omega}_\tau^\wedge\right) C_1^*, \tag{3.56}$$

$$C_4^* = \exp\left(-\frac{1}{4}(t_f - \tau)\boldsymbol{\omega}_f^\wedge\right) C_5^*. \tag{3.57}$$

The angular acceleration of zero at impact time, as imposed by (3.49), is achieved by setting

$$C_3 = \exp\left(-\frac{1}{4}(\tau - t_0)\boldsymbol{\omega}_\tau^\wedge\right) C_4, \tag{3.58}$$

for the first segment and

$$C_3^* = \exp\left(\frac{1}{4}(t_f - \tau)\boldsymbol{\omega}_\tau^\wedge\right) C_2^*, \tag{3.59}$$

for the second segment.
These control points are used to compute the reference orientation trajectory $\mathbf{R}^{\text{ref}}(t)$ using the procedure presented in Section 2.6. In the De Casteljau algorithm on SO(3), the interpolation between two consecutive control points is computed by

$$P_i(\lambda) = C_i \exp(\lambda \log(C_i^T C_{i+1})). \tag{3.60}$$

As seen in Section 3.2.3, the reference trajectory $\mathbf{R}^{\text{ref}}(t)$ should be given in terms of the orientation of the desired frame with respect to the inertial frame ${}^W\mathbf{R}_D$. Boundary conditions (3.43) - (3.45), should therefore be specified in the same way. The reference velocity conditions, given by (3.46) - (3.48), should be given in terms of the angular velocity of the desired frame with respect to the world frame, expressed in the world frame ${}^W\boldsymbol{\omega}_{W,D}$. The angular velocity reference trajectory $\boldsymbol{\omega}^{\text{ref}}(t)$ can be computed using the orientation trajectory $\mathbf{R}^{\text{ref}}(t)$, according to

$${}^W\boldsymbol{\omega}_{W,D}^\wedge = {}^W\dot{\mathbf{R}}_D {}^W\mathbf{R}_D^T. \tag{3.61}$$

The angular acceleration reference trajectory $\dot{\boldsymbol{\omega}}^{\text{ref}}(t)$ is found by differentiation of the reference velocity trajectory with respect to time.
As an illustrative example, in Figure 3.12, an orientation task trajectory, computed using the De Casteljau algorithm on SO(3) for arbitrary boundary conditions, is visualized using a block. The orientations of the blue blocks correspond to the boundary conditions imposed by (3.50) - (3.53). The red blocks have the orientations specified by the control points that define the angular velocities and accelerations of the block, given by (3.54) - (3.59). The gray blocks show the sequence of orientations computed with the De Casteljau algorithm, representing a reference trajectory $\mathbf{R}^{\text{ref}}(t)$.

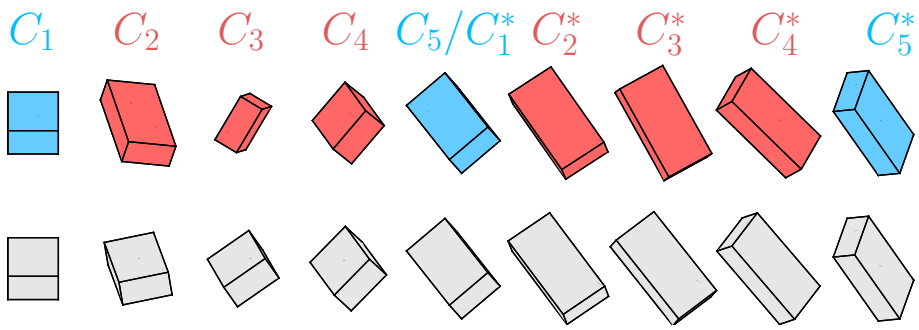


Figure 3.12: *An example of an orientation trajectory, consisting of two segments computed using the De Casteljau algorithm on $SO(3)$. The blocks show a sequence of orientations. The blue blocks represent the orientations at the boundaries of the segments. The red blocks show the orientations given by the control points that define the angular velocities and accelerations of the block. The orientations of the gray blocks are computed by the algorithm.*

3.3.4 Direction task trajectory generation

The direction task is used to prescribe the direction of a body vector on the fourth link of both manipulator arms, both before and after the impact, as detailed in Section 3.2.1. For this purpose, extended ante- and post-impact reference task trajectories ${}^a\bar{\boldsymbol{y}}_{\text{vec}}(t)$ and ${}^p\bar{\boldsymbol{y}}_{\text{vec}}(t)$ have to be generated. The approach for the generation of the direction task trajectories is very similar to the other tasks. Consider a general reference direction trajectory $\mathbf{u}^{\text{ref}}(t) \in S^2$, which represents either the extended ante- or post-impact reference direction trajectory. The trajectory is subject to

$$\mathbf{u}^{\text{ref}}(t_0) = \mathbf{u}_0, \tag{3.62}$$
$$\mathbf{u}^{\text{ref}}(\tau) = \mathbf{u}_\tau, \tag{3.63}$$
$$\mathbf{u}^{\text{ref}}(t_f) = \mathbf{u}_f, \tag{3.64}$$

with $\mathbf{u}_0$, $\mathbf{u}_\tau$ and $\mathbf{u}_f$ the conditions at the initial, impact and final time, respectively. The conditions on reference velocity trajectory $\dot{\mathbf{u}}^{\text{ref}}(t) \in \mathbb{R}^3$ are given by

$$\dot{\mathbf{u}}^{\text{ref}}(t_0) = \dot{\mathbf{u}}_0, \tag{3.65}$$
$$\dot{\mathbf{u}}^{\text{ref}}(\tau) = \dot{\mathbf{u}}_\tau, \tag{3.66}$$
$$\dot{\mathbf{u}}^{\text{ref}}(t_f) = \dot{\mathbf{u}}_f, \tag{3.67}$$

where $\dot{\mathbf{u}}_0$, $\dot{\mathbf{u}}_\tau$ and $\dot{\mathbf{u}}_f$ denote the conditions on the velocity at initial, impact and final time, respectively. Two segments, for $t \in [t_0, \tau]$ and $t \in [\tau, t_f]$, are computed using the De Casteljau algorithm for S^2 spheres, as proposed in [34]. In order to achieve $\mathcal{C}^2$ smoothness of the trajectory on the entire interval $t \in [t_0, t_f]$, a condition on the acceleration of the direction vector at impact time $\ddot{\mathbf{u}}^{\text{ref}}(\tau)$ is set. The De Casteljau algorithm on S^2 only allows to specify an acceleration that corresponds to an angular acceleration of the vector that is equal to zero.
Control points $C_i \in S^2$ and $C_i^* \in S^2$ for $i \in \{1, 2, \ldots, 5\}$ are used for the first and second segment respectively, in order to satisfy all conditions. The derivation of the choices for the control points is detailed in [34]. The first and last control point are chosen equal to the initial and final direction vector

of the segments, given by (3.62) - (3.64), such that

$$C_1 = \mathbf{u}_0, \tag{3.68}$$

$$C_5 = \mathbf{u}_\tau, \tag{3.69}$$

and

$$C_1^* = \mathbf{u}_\tau, \tag{3.70}$$

$$C_5^* = \mathbf{u}_f. \tag{3.71}$$

In order to achieve the desired initial velocity and final velocity, given by (3.65) - (3.67), C_2 and C_4 are chosen as

$$C_2 = \exp\left(\frac{1}{4}(\tau - t_0)(C_1 \times \dot{\mathbf{u}}_0)^\wedge\right) C_1, \tag{3.72}$$

$$C_4 = \exp\left(-\frac{1}{4}(\tau - t_0)(C_5 \times \dot{\mathbf{u}}_\tau)^\wedge\right) C_5. \tag{3.73}$$

and C_2^* and C_4^* as

$$C_2^* = \exp\left(\frac{1}{4}(t_f - \tau)(C_1^* \times \dot{\mathbf{u}}_\tau)^\wedge\right) C_1^*, \tag{3.74}$$

$$C_4^* = \exp\left(-\frac{1}{4}(t_f - \tau)(C_5^* \times \dot{\mathbf{u}}_f)^\wedge\right) C_5^*. \tag{3.75}$$

Finally, control point C_3 and C_3^* are chosen such that the angular acceleration of the direction vector is equal to zero at impact time, such that

$$C_3 = \exp\left(-\frac{1}{4}(\tau - t_0)(C_5 \times \dot{\mathbf{u}}_\tau)^\wedge\right) C_4, \tag{3.76}$$

and

$$C_3^* = \exp\left(\frac{1}{4}(t_f - \tau)(C_1^* \times \dot{\mathbf{u}}_\tau)^\wedge\right) C_2^*. \tag{3.77}$$

Using these control points and the procedure presented in Section 2.6, the reference direction task trajectory $\mathbf{u}^{\text{ref}}(t)$ is computed. In the De Casteljau algorithm on S^2, the interpolation between consecutive control points is given by

$$P_i(\lambda) = \begin{cases} \exp\left(\lambda \cos^{-1}\left(C_i \cdot C_{i+1}\right)\left(\frac{C_i \times C_{i+1}}{\|C_i \times C_{i+1}\|}\right)\right) C_i, & \text{if } \|C_i \times C_{i+1}\| \neq 0, \\ C_i, & \text{if } \|C_i \times C_{i+1}\| = 0. \end{cases} \tag{3.78}$$

The first and second order time derivative of $\mathbf{u}^{\text{ref}}(t)$ are used as the reference velocity trajectory $\dot{\mathbf{u}}^{\text{ref}}(t)$ and acceleration trajectory $\ddot{\mathbf{u}}^{\text{ref}}(t)$. Since $\mathbf{u}^{\text{ref}}(t)$, $\dot{\mathbf{u}}^{\text{ref}}(t)$ and $\ddot{\mathbf{u}}^{\text{ref}}(t)$ have to be defined in terms of ${}^W\mathbf{u}^{\text{ref}}$, ${}^W\dot{\mathbf{u}}^{\text{ref}}$ and ${}^W\ddot{\mathbf{u}}^{\text{ref}}$, respectively, as detailed in Section 3.2.4, the boundary conditions (3.62) - (3.67) have to be specified in the same way.

In Figure 3.13, an example of a reference direction task trajectory is shown for arbitrary boundary conditions. It can be seen that the trajectory prescribes a path on the unit sphere. The control points in blue, given by (3.68) - (3.71), define the direction vector at the boundaries and therefore lie on the computed trajectory. Using the red control points, which correspond to (3.72) - (3.77), the velocities and accelerations at the boundaries are specified. The conditions on the velocity imposed by control points C_4 and C_2^* and the condition on the acceleration imposed by C_3 and C_3^* cause the trajectory to be $\mathcal{C}^2$ smooth at the control point given by C_5 and C_1^*.

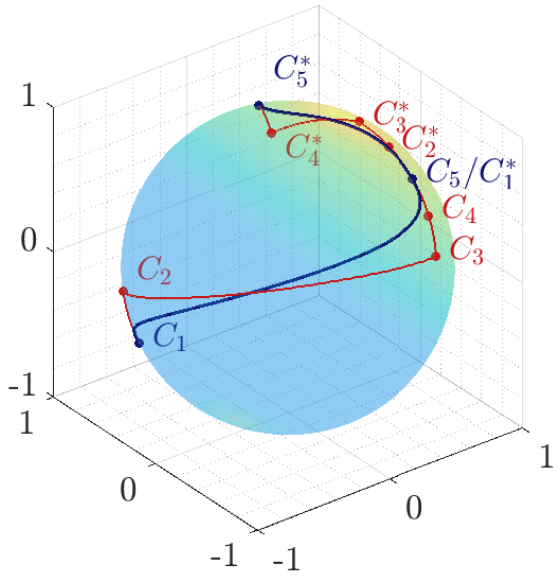


Figure 3.13: *An example trajectory for the direction task, consisting of two segments computed using the De Casteljau algorithm on S^2. The blue control points define the direction vector at the boundaries. Using the red control points the velocities and accelerations at the boundaries are specified.*

3.4 Task-based reference spreading QP controller

In this section, the QP control framework for task-based reference spreading, proposed in Section 3.1.3, is detailed for the considered dual arm dynamic box-lifting application. This framework allows to track the extended reference task trajectories, such that the system is always compared to a reference trajectory that describes the same mode, even in the presence of perturbations. Thereby, it provides a control strategy during intermediate unspecified modes, which occur when the simultaneity of the impact between the end effectors of the manipulator and the box is lost. For this purpose, an overarching controller, called the task-based reference spreading controller, is specified, which operates in three sequential modes, being the ante-impact mode, the intermediate mode and the post-impact mode. First, the different modes are explained. Thereafter, the QP control strategy during each of the modes is detailed.

3.4.1 Controller modes

The controller starts in the ante-impact mode, in which the extended ante-impact reference task trajectories are tracked. As explained in Section 3.2.1, the reference trajectories are defined such that contact between the end effector surfaces and the box is established through simultaneous impacts. Therefore, the reference task trajectory $\bar{\boldsymbol{y}}(t)$, which represents the combined reference task trajectory for all tasks, jumps once at $t = \tau$, as seen in Figure 3.14. However, in the presence of perturbations, the simultaneity of the impact is generally lost. Since the end effector surfaces of the manipulator are square, it is possible that, instead of a surface contact, the end effectors first make a point contact, then a line contact and finally a surface contact, which is illustrated in Figure 3.15. Therefore, the end

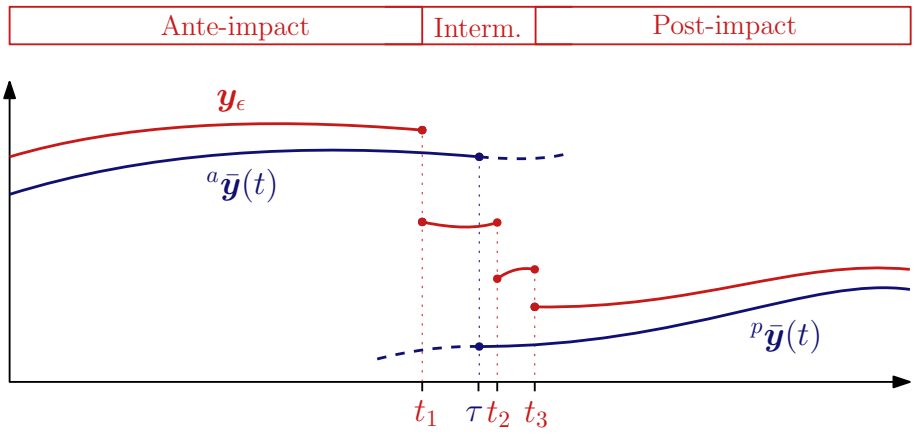


Figure 3.14: *Visual illustration of the three sequential modes of the reference spreading controller. In blue, the extended reference task trajectory $\bar{\boldsymbol{y}}(t)$ is shown. In red, the perturbed task state trajectory $\boldsymbol{y}_\epsilon$ is shown. As a result of a perturbation, the task state trajectory jumps three times, at $t = t_1$, $t = t_2$ and $t = t_3$, as opposed to the reference task trajectory, which only jumps at $t = \tau$. The bar in the top of the figure shows the current mode of the controller.*

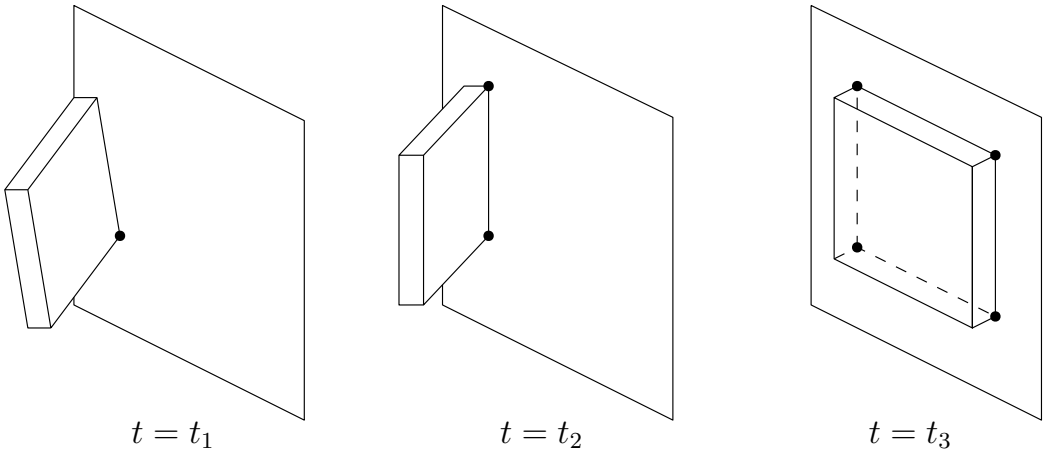


Figure 3.15: *Visual representation of the loss of simultaneity for an impact between a square end effector and a box surface. It is seen that the end effector establishes a point contact, followed by a line contact and eventually a surface contact. Therefore, a perturbed task state trajectory $\boldsymbol{y}_\epsilon$ jumps three times, at $t = t_1$, $t = t_2$ and $t = t_3$. Note that, for the sake of clarity, a very large perturbation is considered in this figure. Typically, perturbations are much smaller.*

effectors can experience three impacts, at $t = t_1$, $t = t_2$ and $t = t_3$. This is seen for the perturbed task trajectory $\boldsymbol{y}_\epsilon$ in Figure 3.14, which jumps three times. On the time interval $t \in [t_1, t_3]$, the manipulator is in an intermediate unspecified mode, as the reference trajectory is not specified for the mode in which the end effectors are in contact with the box through point or line contacts. Therefore, normal feedback control cannot be used during the unspecified mode. However, it is important that the full surface contact is eventually established, in order to perform the motion prescribed by the post-impact reference task trajectories. For this purpose, the intermediate mode is used. The intermediate mode

ideally lasts from the moment the first impact is made between the manipulator and the box until the moment the last impact has been completed, which is in this case from $t = t_1$ until $t = t_3$. After the intermediate mode, the controller switches to the post-impact mode. In the post-impact mode, the manipulator and the box perform a constrained motion, described by the post-impact reference task trajectories. During the manipulation of the box, the manipulator end effectors have to maintain the surface contact with the box.
In Figure 3.14, the current mode of the controller for the perturbed trajectory is shown in the bar at the top of the figure. The QP control strategy in each mode is explained in the following subsections.

3.4.2 Ante-impact mode

During the ante-impact mode, the standard multirobot QP controller is used, as reviewed in Section 2.4, without contact constraints. Position, orientation, and direction tasks are used for both manipulator arms, as specified in Section 3.2.1. Therefore, the QP cost function is formulated as

$$\min_{\ddot{\mathbf{q}}} \quad w_{\text{pos},1}\boldsymbol{E}_{\text{pos},1} + w_{\text{ori},1}\boldsymbol{E}_{\text{ori},1} + w_{\text{vec},1}\boldsymbol{E}_{\text{vec},1} + w_{\text{pos},2}\boldsymbol{E}_{\text{pos},2} + w_{\text{ori},2}\boldsymbol{E}_{\text{ori},2} + w_{\text{vec},2}\boldsymbol{E}_{\text{vec},2}, \tag{3.79}$$

where $\boldsymbol{E}_{\text{pos},i}$, $\boldsymbol{E}_{\text{ori},i}$ and $\boldsymbol{E}_{\text{vec},i}$ denote the cost functions for the position, orientation, and direction task, respectively, as defined in Section 3.2, for manipulator arm i with $i \in \{1, 2\}$ and $w_{\text{pos},i}$, $w_{\text{ori},i}$ and $w_{\text{vec},i}$ denote the associated task weights.
In Figure 3.16, a block diagram is shown of the QP control framework during the ante-impact mode. The extended ante-impact reference task trajectories are specified in the task-based reference spreading

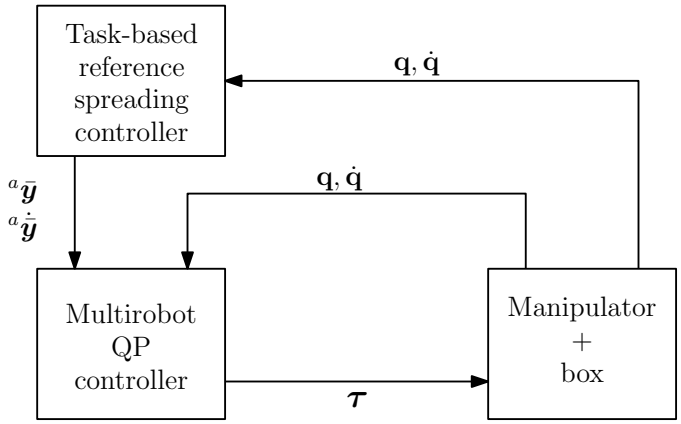


Figure 3.16: *Block diagram of the control framework during the ante-impact mode.*

controller and sent to the QP controller. The QP controller computes the torques that are necessary to track the reference trajectories, using the actual joint positions and velocities of the manipulator. The task-based reference spreading controller switches to the intermediate mode when an impact is detected. The first impact, which triggers the intermediate mode, can be detected using an impact detection method based on the joint states, like the method proposed in [24, Ch. 7]. For this purpose, the current joint states of the manipulator are sent to the task-based reference spreading controller.

3.4.3 Intermediate mode

During the intermediate mode, closed loop control, like in the ante-impact mode, cannot be used, as the reference task trajectories are not specified for the mode in which the end effectors are in contact with the box through point or line contacts. However, it is desired that surface contacts between the end effector surfaces and the box surfaces are established, in order to perform the motion described by the post-impact reference task trajectories. In classical reference spreading, it is proposed to switch off the feedback during the intermediate unspecified mode, as explained in Section 2.2.3. However, this is not straightforward to implement in task-based QP control. To illustrate why this is the case, consider the position task cost function in (3.8). Removing the feedback terms simplifies the cost function to

$$\boldsymbol{E}_{\text{pos}} = \frac{1}{2} \left\| {}^{W}\ddot{\mathbf{p}} - {}^{W}\ddot{\mathbf{p}}^{\text{ref}} \right\|^2 . \tag{3.80}$$

The problem is that for the computation of ${}^{W}\ddot{\mathbf{p}}$, the current joint velocities of the manipulator $\dot{\mathbf{q}}$ are used, as seen in (3.12). However, these joint velocities are likely unreliable as state jumps occur when impact is made. Therefore, removing the feedback may not result in the desired behavior. In order to solve this, the QP control framework should operate in open loop during the intermediate mode. This means that the QP controller computes the joint torques for the robot using the reference joint positions $\mathbf{q}^{\text{ref}}$ and velocities $\dot{\mathbf{q}}^{\text{ref}}$, found in the previous time step, instead of the actual joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ that are measured on the robot. Using the extended ante-impact reference task trajectories as the reference signal, the QP controller computes the joint torques that would result in tracking of these trajectories if the impact would not have occurred. The computed joint torques are used as the input for the robot in order to complete the surface contact. The open loop QP control framework for the intermediate mode is shown in Figure 3.17. The extended ante-impact reference task

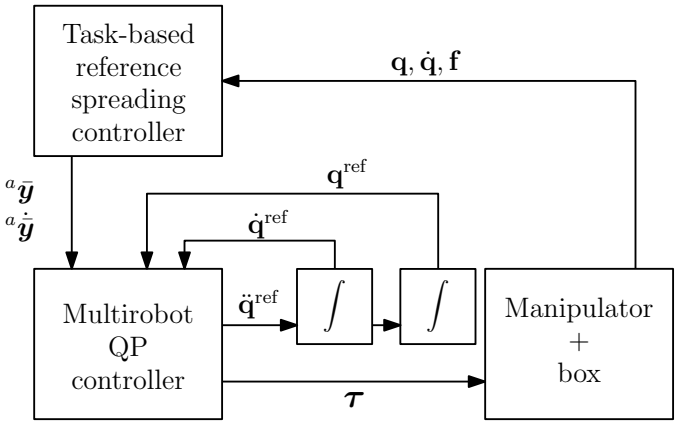


Figure 3.17: *Block diagram of the open loop control framework during the intermediate mode.*

trajectories are sent to the QP controller by the task-based reference spreading controller, as is also done in the ante-impact mode. The QP controller computes the joint torques for the robot based on the same model and cost function as the ante-impact mode, but using the reference joint positions and velocities. These reference joint positions and velocities are obtained by integration of the reference joint accelerations that are computed by the QP controller. The controller remains in the intermediate mode until all contact points of the end effector surfaces are in contact with the surface of the box. For this purpose, an impact detection method has to be used that is able to detect the last impact.

However, the impact detection method used in the ante-impact mode is not suitable for this. Therefore, an alternative impact detection method has to be used, for example using force sensors in the contact points. In simulation, the impact detection can be done by measuring the force in the contact points. For this purpose, the contact forces are sent to the task-based reference spreading controller. The impact detection method on a real robot is beyond the scope of this research.
In addition to the open loop control approach, a second control approach can be explored for the intermediate mode. Since the actual joint positions of the robot are not affected by the impact, these are still reliable after the impact has occurred. Therefore, another option for the intermediate mode is to use the actual joint positions in the QP controller, in combination with the reference joint velocities. This has the advantage that closed loop control is possible on the body pose of the end effectors, which may result in faster completion of the desired surface contact. However, since this approach imposes an incoherency in the robot model between the used joint positions $\mathbf{q}$ and joint velocities $\dot{\mathbf{q}}^{\text{ref}}$, this approach has to be carefully investigated. The QP control framework associated with this approach is shown in Figure 3.18. This framework is identical to the framework shown in Figure 3.18, except for the joint positions and velocities that are used by the QP controller.

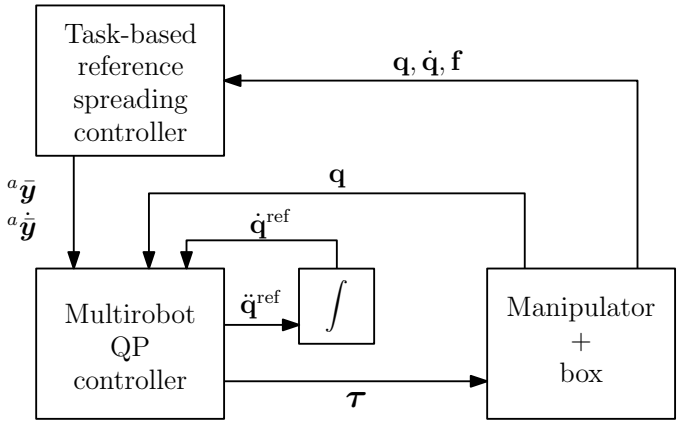


Figure 3.18: *Block diagram of the partially open loop control framework during the intermediate mode.*

3.4.4 Post-impact mode

During the post-impact mode, the standard multirobot QP controller with contact constraints is used, as discussed in Section 2.4. The manipulator end effector surfaces and the box surfaces are constrained to each other, using contact constraints, as discussed in Section 2.4.2. Position and orientation tasks are used for the box, as well as direction tasks for both manipulator arms, as detailed in Section 3.2.1. In addition to these tasks, a force task is used in order to control the contact forces between the end effectors and the box, such that contact is maintained. The exact implementation of the force task is presented in [35]. The cost function of the force task is formulated as

$$\boldsymbol{E}_{\text{force}} = \frac{1}{2} \left\| \mathbf{f} - \mathbf{f}^{\text{ref}} \right\|^2, \tag{3.81}$$

where $\mathbf{f}$ denotes the vector of contact forces and $\mathbf{f}^{\text{ref}}$ denotes the vector of desired contact forces. Eventually, the constrained set of robots is controlled using a single QP problem, which is formulated

as (2.54), with cost function

$$\min_{\ddot{\mathbf{q}},\boldsymbol{\lambda}} \quad w_{\text{vec},1}\boldsymbol{E}_{\text{vec},1} + w_{\text{vec},2}\boldsymbol{E}_{\text{vec},2} + w_{\text{pos,box}}\boldsymbol{E}_{\text{pos,box}} + w_{\text{ori,box}}\boldsymbol{E}_{\text{ori,box}} + w_{\text{force}}\boldsymbol{E}_{\text{force}}, \tag{3.82}$$

where $\boldsymbol{E}_{\text{pos,box}}$ and $\boldsymbol{E}_{\text{ori,box}}$ denote the cost functions for the position and orientation task for the box, respectively, as defined in Section 3.2, $w_{\text{pos,box}}$ and $w_{\text{ori,box}}$ denote the associated task weights, and w_{force} the weight of the force task given by (3.81).
An overview of the control framework during the post-impact mode is shown in Figure 3.19. The extended post-impact reference task trajectories and the desired contact forces are specified in the task-based reference spreading controller and sent to the multirobot QP controller. In addition, the task-based reference spreading controller imposes contact constraints on the multirobot QP controller. The QP controller computes the joint torques for the manipulator based on the current state of the box and the manipulator.

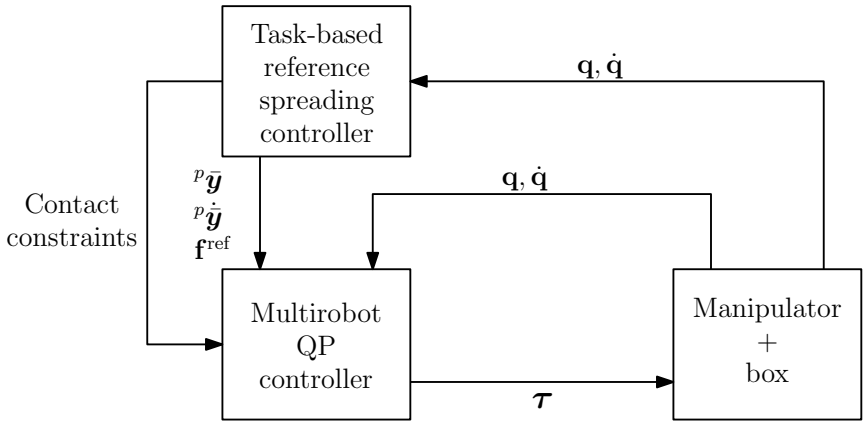


Figure 3.19: *Block diagram of the control framework during the post-impact mode.*

3.5 Summary

In this chapter, a task-based QP control approach for impact aware manipulation has been presented, based on reference spreading. In order to apply reference spreading on task-based QP control, ante- and post-impact reference trajectories have been defined that are compatible with the robot impact dynamics and suitable for task-based QP control. For this purpose, the ante-impact reference task trajectories have to define a unique state trajectory implicitly, such that unique impact dynamics are implied as well. This has been achieved by removing the task redundancy in the robot. Post-impact reference task trajectories have been specified such that they are compatible with the impact dynamics corresponding to the ante-impact reference task trajectories. Inspired by reference spreading, the reference task trajectories have been extended beyond the intended jump times, in order to deal with perturbations. The generation of such extended reference task trajectories and the implementation of task-based reference spreading in a QP control framework are dependent on the considered application and robot. For this purpose, a dynamic box-lifting application for a dual arm manipulator has been introduced. Three QP tasks have been defined, which are used to specify the desired motion of the manipulator for this application. A trajectory generation procedure has been proposed, which can be

used to generate extended task trajectories for each of the defined tasks, that are compatible with the impact dynamics. A QP control framework for task-based reference spreading has been proposed for the dual arm dynamic box-lifting application. For this purpose, an overarching controller has been presented, consisting of three sequential modes. The ante- and post-impact mode are used to track the extended ante- and post-impact reference task trajectories, allowing the system to deal with noncoinciding jump times. An intermediate mode has been introduced in order to deal with the intermediate unspecified mode, that occurs when the simultaneity of the impact is lost.

Chapter 4

Numerical Simulation Study

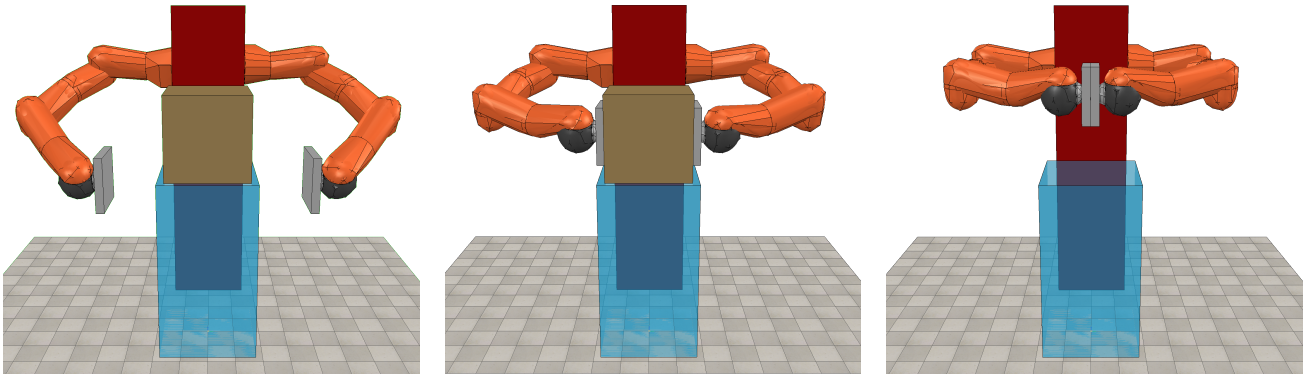
In this chapter, the first step is taken towards numerically validating the task-based QP control approach for impact aware manipulation, proposed in Chapter 3. For this purpose, the trajectory generation procedure for task-based reference spreading, presented in Section 3.3, is demonstrated by means of a numerical simulation study on the dual arm dynamic box-lifting application, presented in Section 3.2.1, in the robot simulator V-REP, introduced in Section 2.5.
First, extended ante-impact reference trajectories are generated for the position, orientation, and direction tasks for the manipulator arms, resulting in the desired simultaneous impacts between the manipulator end effectors and the box. Thereafter, simulations are performed, in order to determine the post-impact task states, which result from the simultaneous impacts that occur when the extended ante-impact reference task trajectories are tracked. These post-impact task states are then used to generate extended post-impact reference trajectories for the position and orientation task of the box, and the direction tasks for both manipulator arms, which are compatible with the ante-impact reference task trajectories and the associated impact dynamics.

4.1 Ante-impact reference task trajectories

In this section, extended ante-impact reference task trajectories are generated for the position, orientation, and direction task for the manipulator arms, such that the desired simultaneous impacts between the end effectors of the manipulator and the box are established. Thereafter, the control gains and weights for each task are tuned, in order to achieve accurate tracking of these trajectories.

4.1.1 Trajectory generation

The extended ante-impact reference task trajectories for the position, orientation, and direction task for the manipulator arms are computed such that the desired motion depicted in Figure 4.1 is achieved. The left figure shows the manipulator in its initial state, at $t_0 = 0$ s. The desired state of the manipulator at impact time $\tau = 1$ s is shown in the middle figure. After impact time, the ante-impact trajectories are extended to the state shown in the right figure, at $t_f = 1.33$ s, which is the final time of the extended ante-impact reference task trajectories. Note that the box is not shown in the final state, as the extension of the ante-impact trajectories represents a motion through the box.



$t = 0$ s $\quad$ $t = 1$ s $\quad$ $t = 1.33$ s

Figure 4.1: *Visual illustration of the desired motion described by the extended ante-impact reference task trajectories. On the left, the desired initial state of the manipulator at $t = 0$ is shown. The middle figure shows the desired state at impact time $t = 1$ s. The right figure shows the final state of the manipulator after extension of the ante-impact reference task trajectories, at $t = 1.33$ s. The box is not shown in the latter figure, as the extensions prescribe a motion through the box.*

First of all, extended reference task trajectories are generated for the position task for both manipulator arms, such that the impacts between the end effectors and the box occur at the desired position and with the desired velocity and acceleration. A reference position trajectory ${}^{a}\bar{\mathbf{p}}^{\mathrm{ref}}(t)$, velocity trajectory ${}^{a}\dot{\bar{\mathbf{p}}}^{\mathrm{ref}}(t)$ and acceleration trajectory ${}^{a}\ddot{\bar{\mathbf{p}}}^{\mathrm{ref}}(t)$ are defined as stated in Section 3.2.2, for points $\mathbf{p}_1$ and $\mathbf{p}_2$, which denote points on the end effector surfaces of KUKA 1 and KUKA 2, respectively. The left arm of the manipulator, which is the arm shown right in the snapshots in Figure 4.1, is denoted as KUKA 1, and the right arm, shown left, is denoted as KUKA 2. The points $\mathbf{p}_1$ and $\mathbf{p}_2$ are illustrated in Figure 4.2 on a schematic representation of the manipulator end effectors in the configuration shown in Figure 4.1 at $t = 1$ s. The points are specified on the center of the square end effector surfaces. Figure 4.2 also shows the orientation of inertial frame W, which is positioned in the center of the ground plane of the blue platform shown in Figure 4.1.

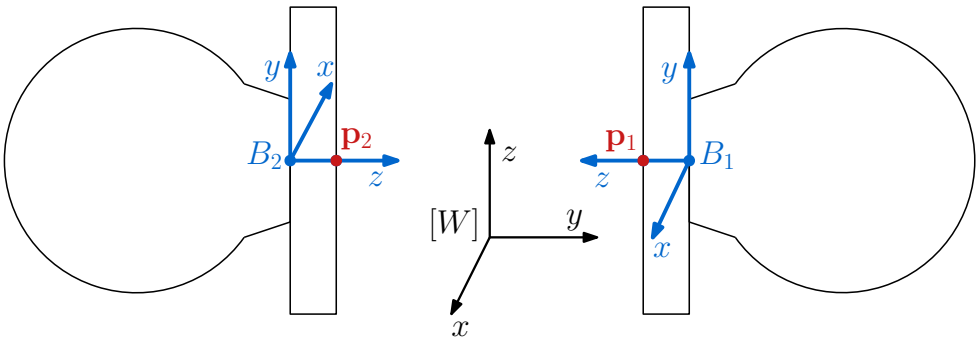


Figure 4.2: *Schematic representation of the end effectors illustrating points $\mathbf{p}_1$ and $\mathbf{p}_2$ on the center of the end effector surfaces, controlled by the position task and body frames B_1 and B_2 controlled by the orientation task. The orientation of the inertial frame is given by $[W]$.*

Table 4.1: *Conditions on the ante-impact reference position task trajectories for both KUKA arms. All conditions are expressed with respect to the inertial frame W.*

		${}^a\bar{\mathbf{p}}^{\text{ref}}(0)$	${}^a\bar{\mathbf{p}}^{\text{ref}}(1.00)$	${}^a\bar{\mathbf{p}}^{\text{ref}}(1.33)$	${}^a\dot{\bar{\mathbf{p}}}^{\text{ref}}(0)$	${}^a\dot{\bar{\mathbf{p}}}^{\text{ref}}(1.00)$	${}^a\dot{\bar{\mathbf{p}}}^{\text{ref}}(1.33)$	${}^a\ddot{\bar{\mathbf{p}}}^{\text{ref}}(1.00)$
KUKA 1	x	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	y	0.37	0.15	0.04	0.00	-0.33	-0.33	0.00
	z	0.63	0.80	0.90	0.00	0.30	0.30	0.00
KUKA 2	x	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	y	-0.37	-0.15	-0.04	0.00	0.33	0.33	0.00
	z	0.63	0.80	0.90	0.00	0.30	0.30	0.00

In Table 4.1, all conditions on the reference position task trajectories are summarized. Taking into account these conditions, the position task trajectories are computed for both arms, using the approach discussed in Section 3.3.2. The generated trajectories for both arms are shown in Figure 4.3.

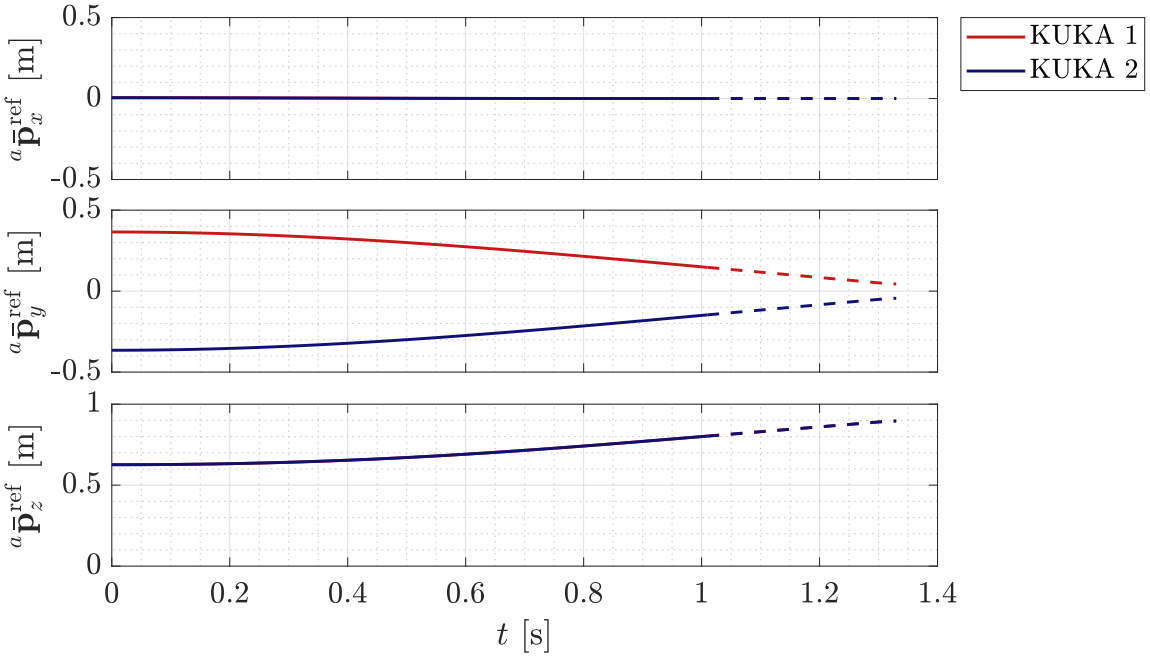


Figure 4.3: *The extended ante-impact reference position task trajectory for both manipulator arms. Subscripts x, y and z denote the x-, y- and z-components of the position vector.*

For the orientation task, extended ante-impact reference orientation trajectories ${}^a\bar{\mathbf{R}}^{\text{ref}}(t)$ are generated, which prescribe the desired orientation of frames B_1 and B_2, which are shown in Figure 4.2, with respect to the inertial frame W, as defined in Section 3.2.3. The desired orientations are chosen to be constant, such that the end effector surfaces are aligned with the box surfaces at all times. The reference orientation trajectory for the end effector of KUKA 1 is therefore given by

$$ {}^a\bar{\mathbf{R}}^{\text{ref}}(t) = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}, \tag{4.1} $$

and by

$$ {}^{a}\bar{\mathbf{R}}^{\mathrm{ref}}(t) = \begin{bmatrix} -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}, \tag{4.2} $$

for the end effector of KUKA 2. The angular velocity and acceleration reference trajectories, denoted by ${}^{a}\bar{\boldsymbol{\omega}}^{\mathrm{ref}}(t)$ and ${}^{a}\dot{\bar{\boldsymbol{\omega}}}^{\mathrm{ref}}(t)$, respectively, are equal to zero for the entire extended ante-impact trajectory. For the direction task for each manipulator arm, extended ante-impact reference direction task trajectory ${}^{a}\bar{\mathbf{u}}^{\mathrm{ref}}(t)$ and its derivatives ${}^{a}\dot{\bar{\mathbf{u}}}^{\mathrm{ref}}(t)$ and ${}^{a}\ddot{\bar{\mathbf{u}}}^{\mathrm{ref}}(t)$, are defined as specified in Section 3.2.4. These trajectories are used to describe the desired direction, velocity and acceleration of body vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, depicted in Figure 4.4. The reference task trajectories are generated using the algorithm presented in Section 3.3.4, using the conditions summarized in Table 4.2. The generated extended trajectories for both arms are shown in Figure 4.5.

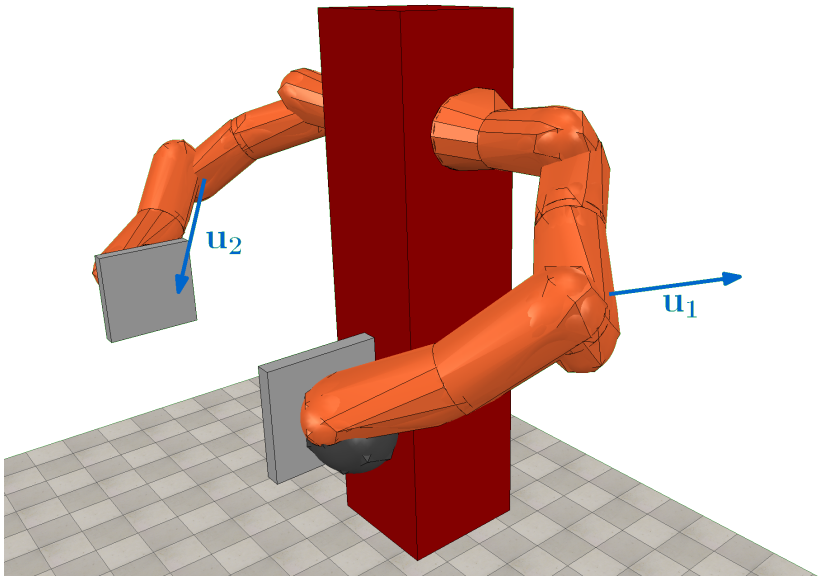


Figure 4.4: *Illustration of the body vectors* $\mathbf{u}_1$ *and* $\mathbf{u}_2$ *on the fourth link of the manipulator arms, which are controlled by the direction task.*

Table 4.2: *Conditions on the ante-impact reference direction task trajectories.*

		${}^{a}\bar{\mathbf{u}}^{\mathrm{ref}}(0)$	${}^{a}\bar{\mathbf{u}}^{\mathrm{ref}}(1.00)$	${}^{a}\bar{\mathbf{u}}^{\mathrm{ref}}(1.33)$	${}^{a}\dot{\bar{\mathbf{u}}}^{\mathrm{ref}}(0)$	${}^{a}\dot{\bar{\mathbf{u}}}^{\mathrm{ref}}(1.00)$	${}^{a}\dot{\bar{\mathbf{u}}}^{\mathrm{ref}}(1.33)$
KUKA 1	x	-0.49	-0.34	-0.18	0.00	0.47	0.47
	y	0.79	0.93	0.98	0.00	0.23	0.23
	z	0.37	0.15	0.02	0.00	-0.37	0.37
KUKA 2	x	0.49	0.34	0.18	0.00	-0.47	-0.47
	y	0.79	0.93	0.98	0.00	0.23	0.23
	z	-0.37	-0.15	-0.02	0.00	0.37	0.37

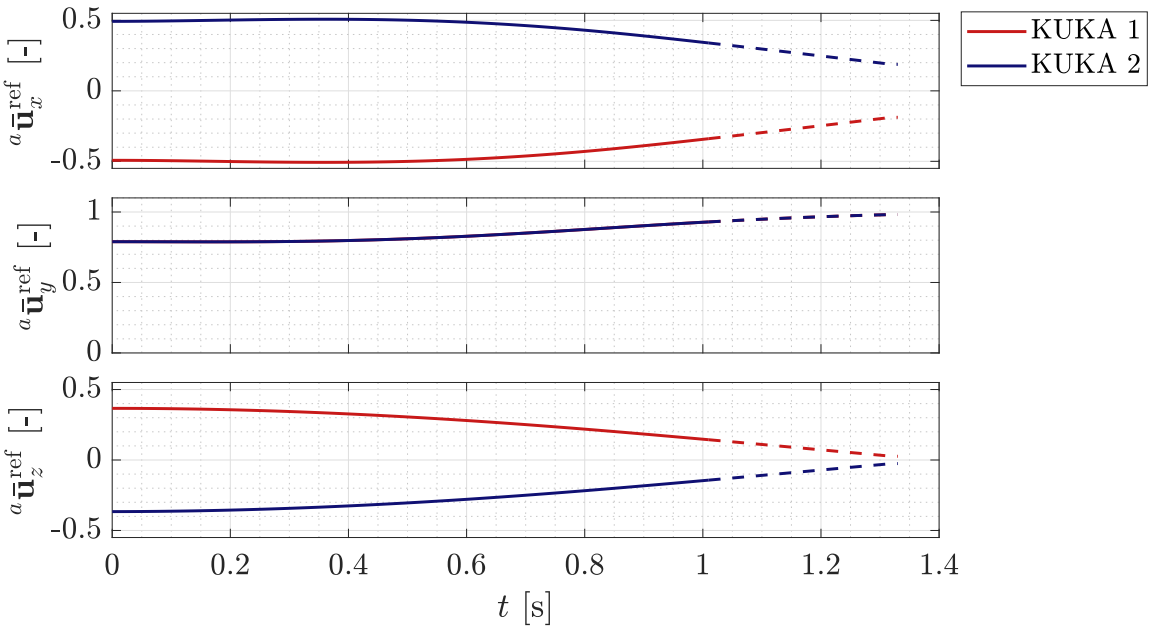


Figure 4.5: *The extended ante-impact reference direction task trajectory for both manipulator arms. Subscripts x, y and z denote the x-, y- and z-components of the direction vector.*

4.1.2 Tracking performance

In order to achieve accurate tracking of the generated ante-impact reference task trajectories, the proportional and derivative control gains for each task, presented in Section 3.2, have to be tuned, as well as the task weights. The control gains are chosen to be scalar for all tasks, as no distinction is made in the control action for different directions. In the remainder of this chapter, the control gains are therefore denoted by lowercase letters for all tasks. The derivative gain k_d is chosen such that $k_d = 2\sqrt{k_p}$, as this results in critical damping of the task errors.
In order to find the control gains and task weights, various perturbed initial configurations of the manipulator are considered, which are obtained by perturbing the initial joint positions. The considered initial configurations are summarized in Appendix C. The control gains and task weights are tuned such that the manipulator achieves accurate tracking of all reference task trajectories, before impact time, for all considered perturbed initial configurations. This ensures that the impacts occur at the desired position, with the desired velocity and orientation. Thereby, if the manipulator tracks all reference task trajectories, this implies that the joint states of the manipulator track the joint state reference trajectory that is implicitly defined by the reference task trajectories, as explained in Section 3.1.1. The task weights have to be chosen such that all reference task trajectories are tracked accurately, as choosing the weight of one of the tasks relatively high with respect to the other tasks causes lower tracking performance of the other tasks. An overview of the gains and weights used in this simulation is given in Table 4.3. Figures 4.6 - 4.8 show the tracking behavior of the manipulator in simulation for each task, using these control gains and weights, for the perturbed initial configurations given in Appendix C. For all tasks, it can be seen that the manipulator achieves accurate tracking of the

Table 4.3: *Control gains and task weights for all tasks.*

Position task			Orientation task			Direction task		
k_p	k_d	w	k_p	k_d	w	k_p	k_d	w
200	28.3	5000	200	28.3	5000	200	28.3	200

reference trajectory within the first second, before impact occurs. Since all reference task trajectories are tracked, the implicitly defined reference joint state trajectories are tracked as well. In Appendix D, it is shown that for all perturbed systems, the joint states converge to the same state trajectory. It is also shown that the joint states do not converge to the same trajectory if the direction task is not used.

Remark 3: The control gains and task weights are tuned manually and do therefore not result in optimal tracking behavior. However, the goal of this research is not to achieve optimal tracking, but to demonstrate task-based reference spreading control. For this purpose, it is sufficient, for now, if the perturbed systems achieve accurate tracking of the reference task trajectories before impact occurs.

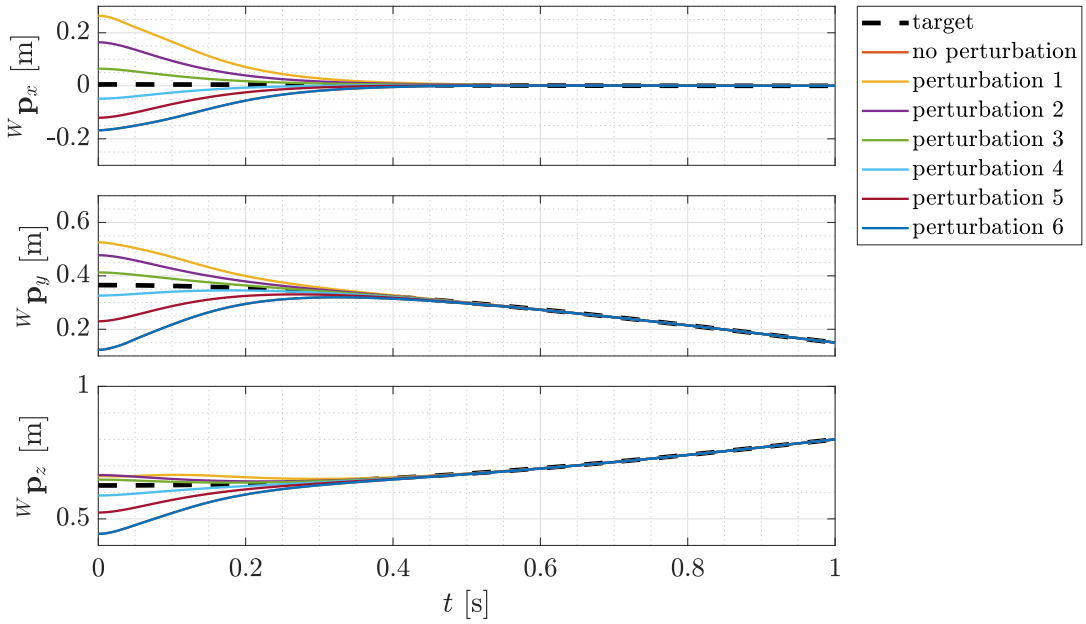


Figure 4.6: *Tracking of the ante-impact reference position trajectory by the manipulator with an unperturbed and various perturbed initial configurations. Accurate tracking of the reference trajectory is achieved within the first second, before impact occurs, for all trajectories. Subscripts x, y and z denote the x-, y- and z-components of the position vector.*

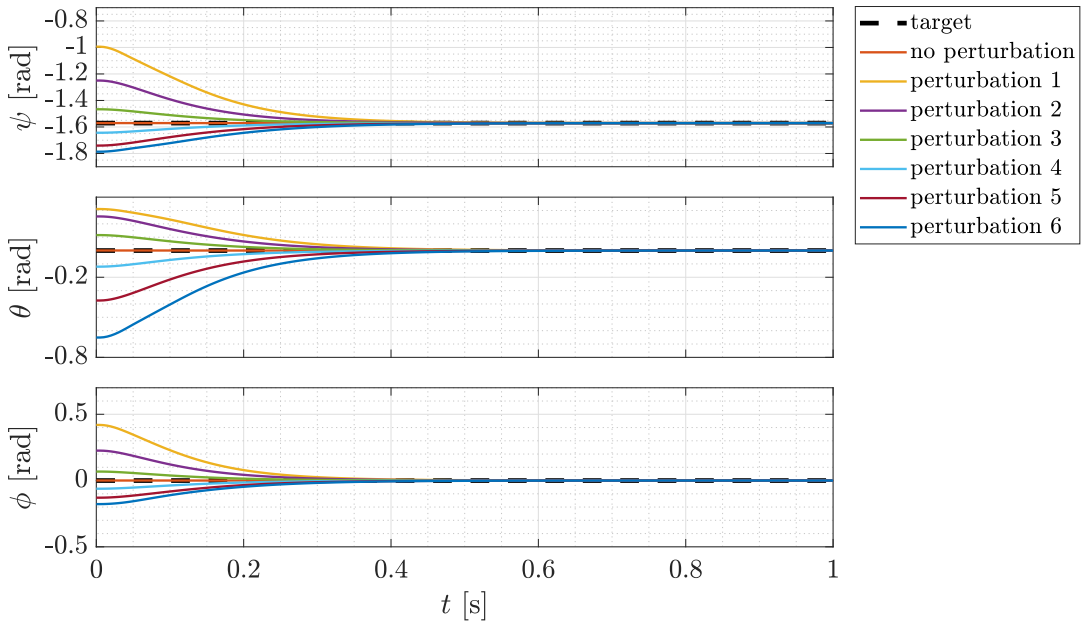


Figure 4.7: *Tracking of the ante-impact reference orientation trajectory by the manipulator with an unperturbed and various perturbed initial configurations. Accurate tracking of the reference trajectory is achieved within the first second, before impact occurs, for all trajectories. Note that, for visualization purposes, the orientation is plotted in terms of roll, pitch and yaw angles ψ, θ and ϕ, which define rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$ describing the rotation about the x-, y- and z-axes respectively.*

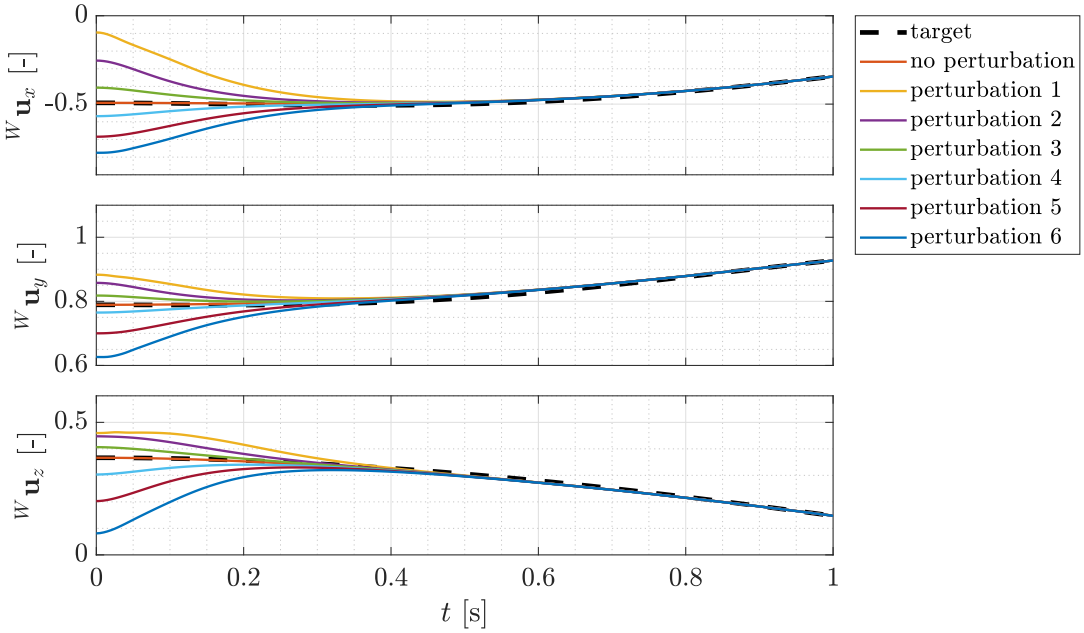


Figure 4.8: *Tracking of the ante-impact reference direction task trajectory by the manipulator with an unperturbed and various perturbed initial configurations. Accurate tracking of the reference trajectory is achieved within the first second, before impact occurs, for all trajectories. Subscripts x, y and z denote the x-, y- and z-components of the direction vector.*

4.2 Determining impact dynamics

Given the ante-impact reference task trajectories generated in Section 4.1.1, post-impact reference task trajectories have to be generated that are compatible with the ante-impact trajectories and the associated impact dynamics. In other words, at the intended impact time τ, the post-impact task trajectories have to be consistent with joint state

$$^{p}\bar{\boldsymbol{\alpha}}(\tau) = \boldsymbol{g}(^{a}\bar{\boldsymbol{\alpha}}(\tau)), \tag{4.3}$$

where $^{a}\bar{\boldsymbol{\alpha}}(t)$ is the ante-impact reference state trajectory implied by the ante-impact reference task trajectories given in Section 4.1.1. The reference post-impact task states at impact time, corresponding to $^{p}\bar{\boldsymbol{\alpha}}(\tau)$, are determined using simulations.
Ideally, one would be able to access a physics engine directly, in order to find the post-impact states that correspond to a specific ante-impact state, but this is not possible in V-REP. Thereby, V-REP does not allow to initialize robot models with a velocity other than zero. Therefore, in order to find the post-impact task states that correspond to a specific ante-impact state, simulations have to be performed in which the manipulator is controlled to this ante-impact state. For this purpose, the extended ante-impact reference task trajectories are tracked by the manipulator, starting from the unperturbed initial configuration. Since the control gains and task weights have been tuned such that the reference trajectories are all tracked accurately, the implicitly defined reference state trajectory $^{a}\bar{\boldsymbol{\alpha}}(t)$ is tracked as well. In simulation, the manipulator will therefore make impact with the box with an ante-impact joint state equal to $^{a}\bar{\boldsymbol{\alpha}}(\tau)$. The post-impact task states, associated with the post-impact joint states given by (4.3), can therefore be found by evaluating the task states right after the impact has occurred. In these simulations, unless specified otherwise, the box has a mass of 1.5 kg with uniform density distribution and the end effectors have sides of 20 cm.
In this section, first the feasibility of the post-impact states, obtained through simulations, is considered. Thereafter, the post-impact task states are obtained, which are used for post-impact trajectory generation in Section 4.3.

4.2.1 Feasibility of the impact dynamics

In the simulations that are used to determine the post-impact task states, the Vortex physics engine is used, which is reviewed in Section 2.5. Since this physics engine can be used for a large variety of applications, many properties can be set to the preference of the user. For the determination of the post-impact task states, it is especially important that the material properties of the box and the end effector are chosen correctly, such that feasible impact dynamics are found. In this research, only inelastic impacts are considered. However, by default, the physics engine assumes objects to have a so-called skin thickness [36]. This skin thickness defines a small volume below the surface of an object, in which the object is softer. This is used in simulation in order to make it easier for robots to grasp the object. However, due to this skin thickness, the end effector surfaces and the box can rotate with respect to each other while in contact, which is not feasible. Therefore, infeasible post-impact task states are found, when simulations are performed using a box with a skin thickness. This is shown in Figure 4.9 for a simulation in which smaller end effector shapes have been used, with sides of 10 cm, in order for the effect of the skin thickness to be more apparent. It can be seen that, as a result of the impact, which occurs at 1.00 s, the angular velocities of the end effectors and the box are different, which is infeasible for an inelastic impact without restitution. The end effectors have an angular

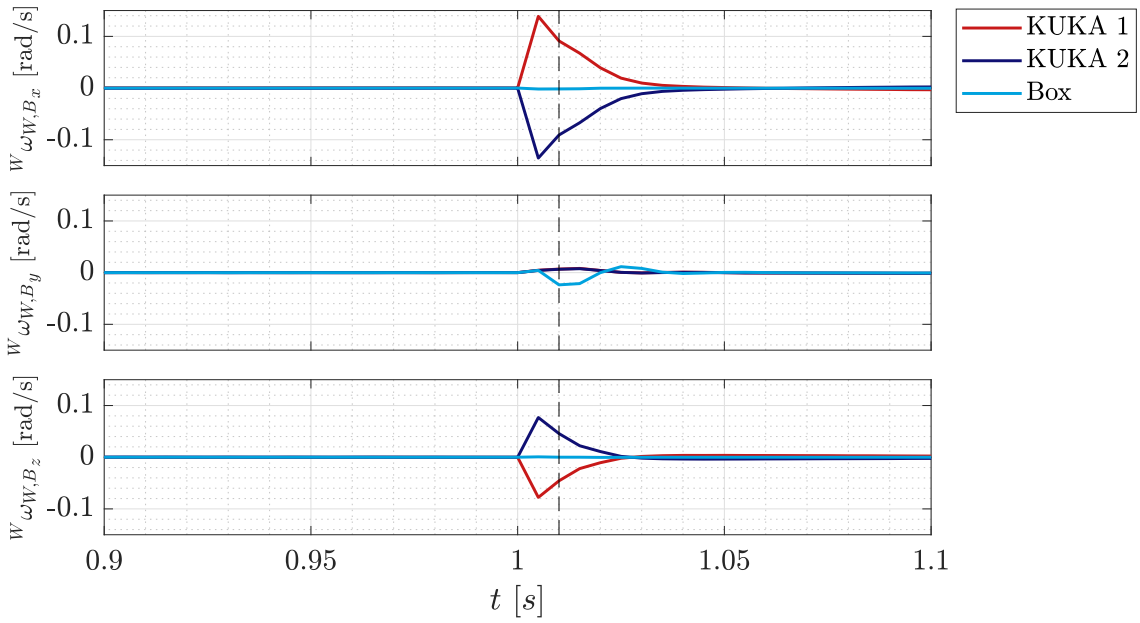


Figure 4.9: *The angular velocity of the end effector frame for both KUKA arms and the box around impact time, where the box has a skin thickness of 2 mm in Vortex. End effector shapes are used with sides of 10 cm. After the impact, the end effectors have an angular velocity in opposite direction, which is not equal to the angular velocity of the box.*

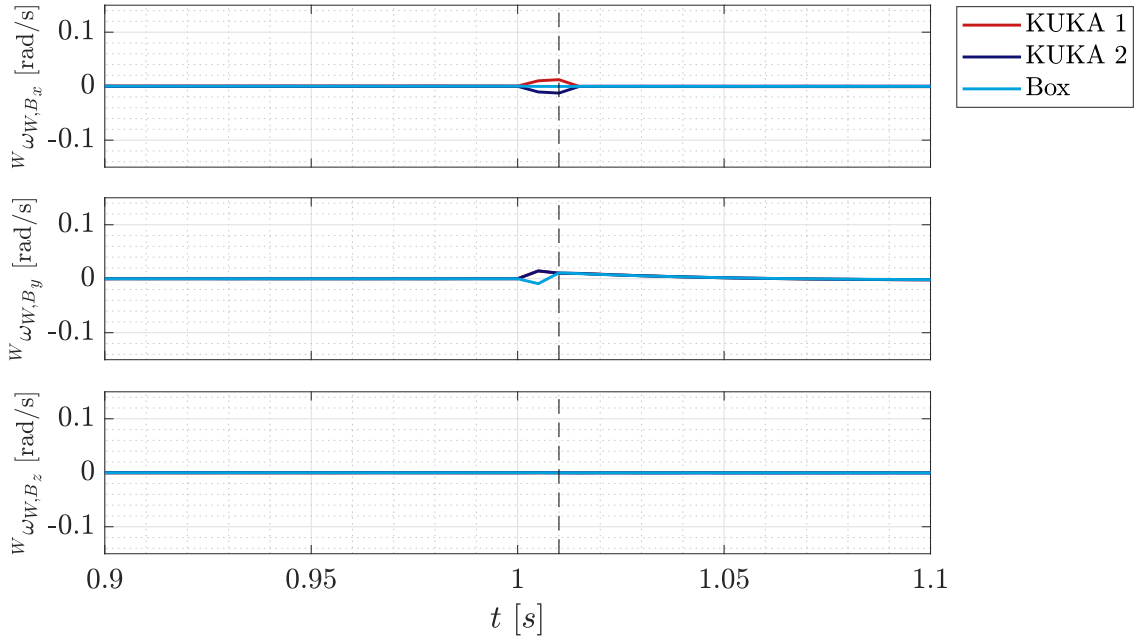


Figure 4.10: *The angular velocity of the end effector frame for both KUKA arms and the box around impact time, where the box has no skin thickness in Vortex. End effector shapes are used with sides of 10 cm. Shortly after the impact, the angular velocities of the end effector and the box are the same.*

velocity in opposite direction about the inertial x- and z-axes, which means that the surfaces cannot both be aligned with the box surfaces after impact. However, this is required, in order to apply contact constraints between the end effectors and the box. In order to achieve feasible post-impact velocities and for the end effector surfaces to be aligned with the box surfaces after impact, the skin thickness has to be removed. In Figure 4.10, it can be seen that, in this case, the angular velocities are equal after the impact has occurred, apart from a small time window, in which the velocities are slightly different. This can be caused by the impact not being perfectly simultaneous or by numerical instabilities. Since the post-impact states are feasible after $t = 1.015$ s, these task states can be used for the generation of the post-impact reference task trajectories and allow contact constraints to be applied.

4.2.2 Determining the post-impact task states

Simulations are performed, in which the manipulator tracks the extended ante-impact reference task trajectories, in order to find the corresponding post-impact task states. In Figures 4.11 - 4.13, the velocities that are associated with the position, orientation, and direction task are plotted around impact time. In Figure 4.12, it can be seen that the angular velocities become feasible at $t = 1.01$ s, as the angular velocities of the end effectors and the box are the same after this time. At this time, also the impulsive change in the linear velocities of the end effectors and the box, and the velocity of the direction vector has stopped, which indicates that impact has been completed. For these reasons, it has been decided to determine the post-impact task states at $t = 1.01$ s. In Figures 4.11 - 4.13 this time instance has been indicated with a dashed line. The determined post-impact states are used for the generation of the extended post-impact reference task trajectories, in Section 4.3.

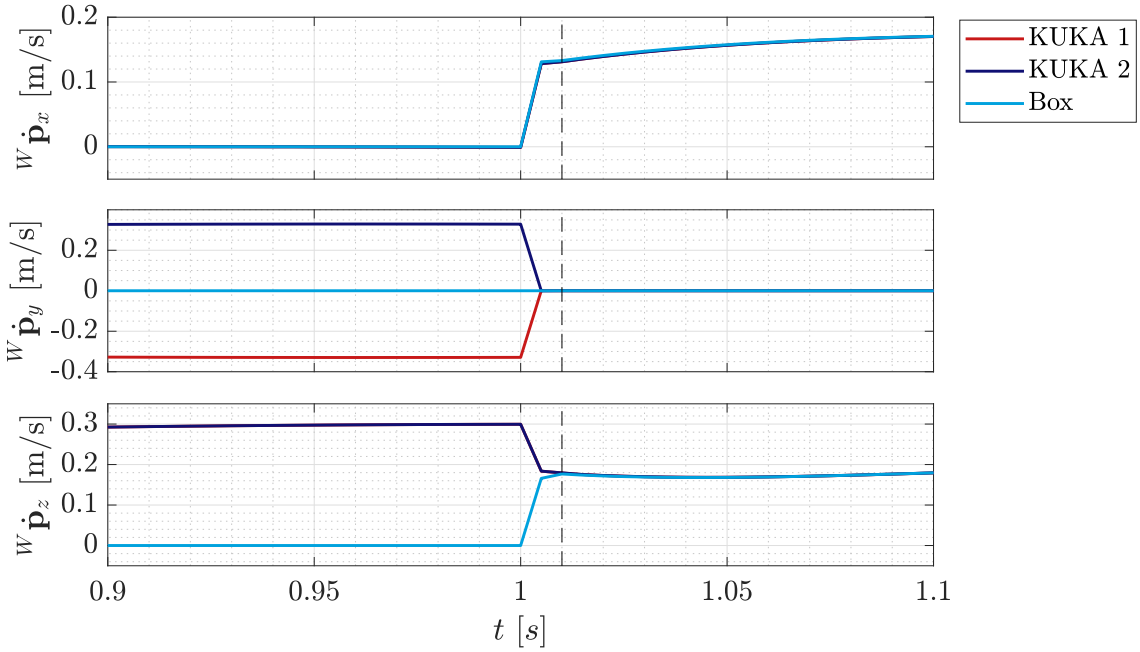


Figure 4.11: *The linear velocity of the point on the end effector both KUKA arms around impact time. End effector shapes are used with sides of 20 cm. The velocities jump as a result of the impact with the box.*

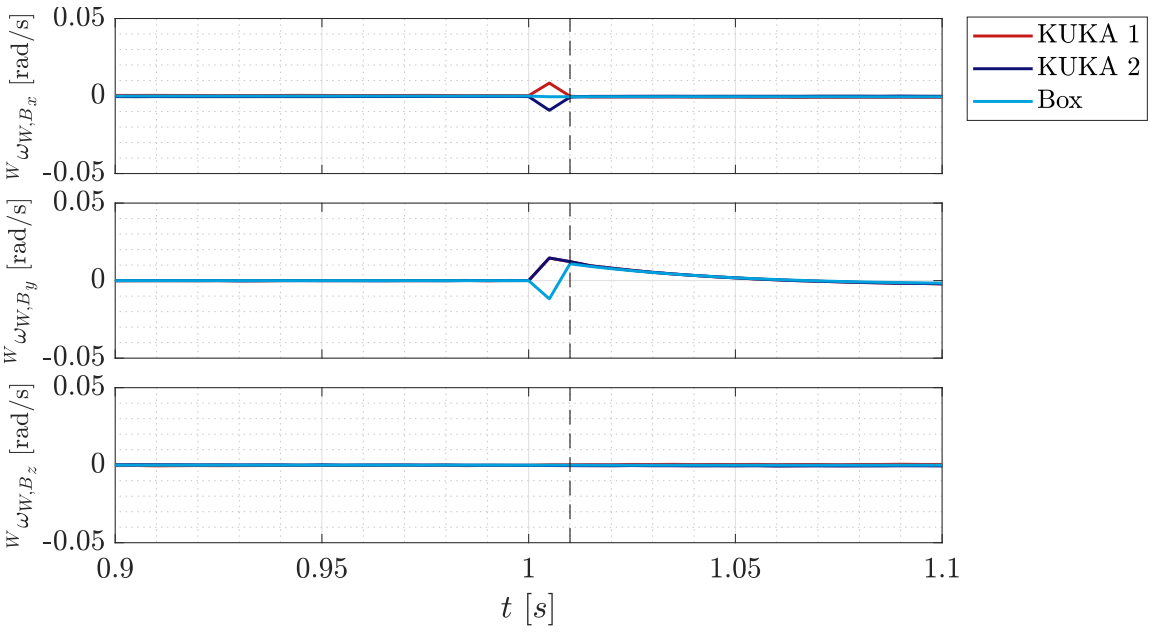


Figure 4.12: *The angular velocity of the end effector frame for both KUKA arms and the box around impact time. End effector shapes are used with sides of 20 cm. The velocities show small jumps as a result of the impact with the box.*

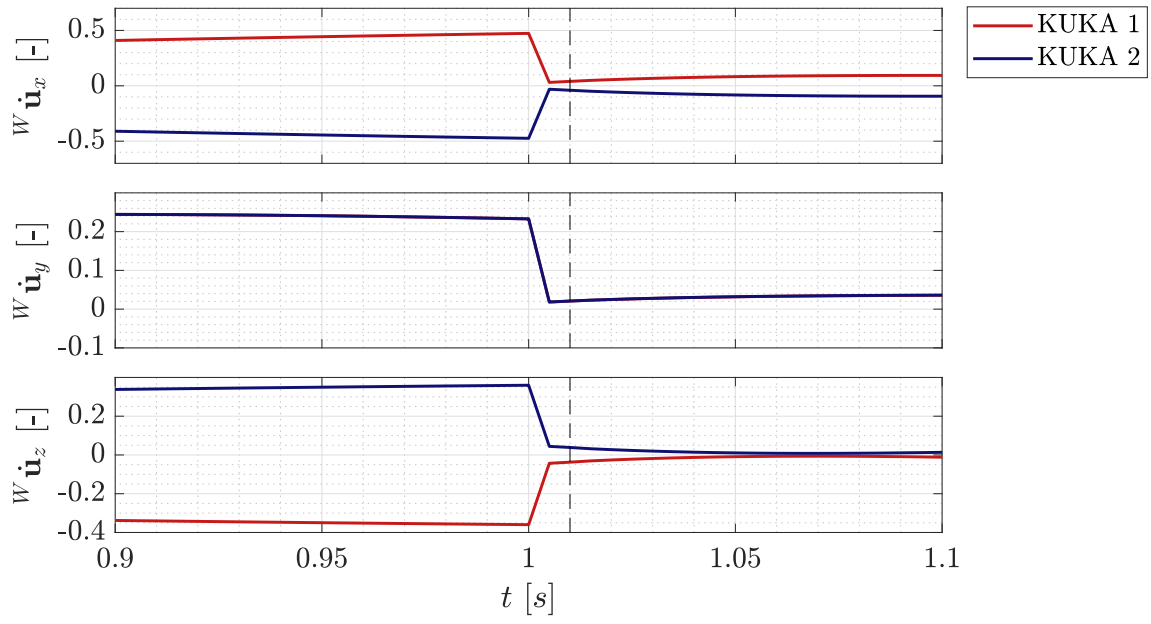
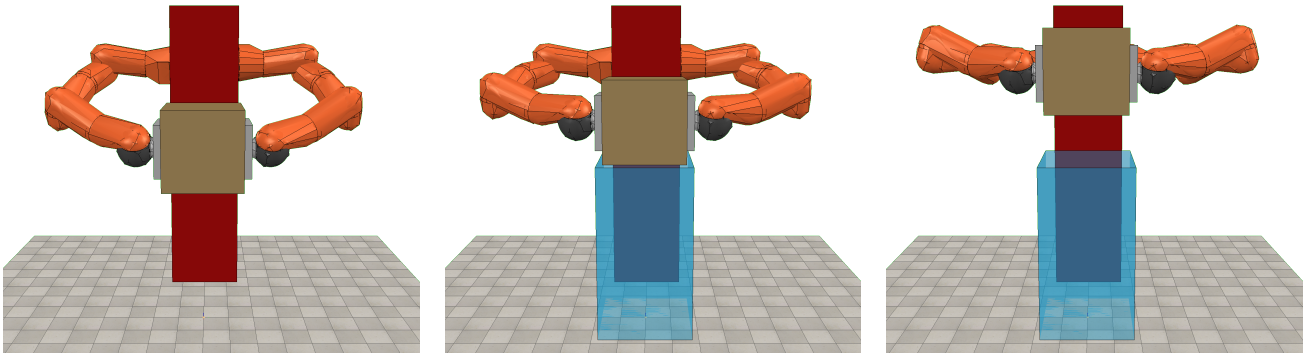


Figure 4.13: *The velocity of the body vectors for both KUKA arms around impact time. The velocities jump as a result of the impact with the box.*

4.3 Post-impact trajectory generation

Using the post-impact task states determined in Section 4.2.2, extended post-impact reference task trajectories can be generated that are compatible with the ante-impact reference task trajectories generated in Section 4.1.1 and the associated impact dynamics. The desired motion that has to be described by the extended post-impact reference task trajectories is illustrated visually in Figure 4.14. The left figure shows the configuration of the manipulator and the box at $t = 0$ s, which is the start of the extended post-impact reference task trajectories. In this configuration, the box is inside the platform, which is not shown for this reason, as the initial configuration represents the backwards extension of the post-impact reference task trajectories. The middle figure shows the configuration at $t = 1.01$ s, which corresponds to the post-impact task states found in Section 4.2.2. The desired final state at $t = 2.00$ s, in which the box has been lifted by the manipulator, is shown in the right figure.



$t = 0$ s $\quad$ $t = 1.01$ s $\quad$ $t = 2.00$ s

Figure 4.14: *Visual illustration of the desired motion described by the extended post-impact reference task trajectories. The left figure shows the initial configuration of the manipulator and the box at $t = 0$ s, which represents the backwards extension of the post-impact reference task trajectories. The middle figure shows the configuration right after impact time at $t = 1.01$ s. In the right figure, the desired final configuration at $t = 2.00$ s is shown.*

As detailed in Section 3.2.1, a position and orientation task for the box, and a direction task for both manipulator arms are used to describe this desired motion. The extended reference trajectories for these tasks are generated in the same way as the ante-impact reference task trajectories, in Section 4.1.1. For the position task for the box, an extended post-impact reference position trajectory ${}^{p}\bar{\mathbf{p}}^{\mathrm{ref}}(t)$, velocity trajectory ${}^{p}\dot{\bar{\mathbf{p}}}^{\mathrm{ref}}(t)$ and acceleration trajectory ${}^{p}\ddot{\bar{\mathbf{p}}}^{\mathrm{ref}}(t)$ are defined. The conditions for these reference trajectories are summarized in Table 4.4. The position trajectory, computed using the procedure

Table 4.4: *Conditions on the post-impact reference position task trajectories for the box.*

	${}^{p}\bar{\mathbf{p}}^{\mathrm{ref}}(0)$	$\bar{\mathbf{p}}^{\mathrm{ref}}(1.01)$	${}^{p}\bar{\mathbf{p}}^{\mathrm{ref}}(2.00)$	${}^{p}\dot{\bar{\mathbf{p}}}^{\mathrm{ref}}(0)$	${}^{p}\dot{\bar{\mathbf{p}}}^{\mathrm{ref}}(1.01)$	${}^{p}\dot{\bar{\mathbf{p}}}^{\mathrm{ref}}(2.00)$	${}^{p}\ddot{\bar{\mathbf{p}}}^{\mathrm{ref}}(1.01)$
x	0.00	0.00	0.00	0.00	0.13	0.00	0.00
y	0.00	0.00	0.00	0.00	0.00	0.00	0.00
z	0.70	0.80	1.00	0.00	0.18	0.00	0.00

presented in Section 3.3.2, is shown in Figure 4.15. In Figure 4.16, it is shown that, at $t = 1.01$ s, the extended post-impact reference velocity trajectory ${}^{p}\dot{\bar{\mathbf{p}}}^{\text{ref}}(t)$ coincides with the post-impact velocities found in Section 4.2.2. This shows that the post-impact reference position task is compatible with the impact dynamics that are associated with the extended ante-impact reference task trajectories generated in Section 4.1.1.
In a similar fashion, the extended post-impact reference orientation task trajectory ${}^{p}\bar{\mathbf{R}}^{\text{ref}}(t)$ for the box is generated. At $t = 1.01$ s, this trajectory has to satisfy

$$ {}^{p}\bar{\mathbf{R}}^{\text{ref}}(1.01) = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{4.4} $$

The initial and final orientation are chosen to be the same as the orientation at impact time. The conditions on the reference angular velocity trajectory ${}^{p}\bar{\boldsymbol{\omega}}^{\text{ref}}(t)$ are given in Table 4.5. The generated

Table 4.5: *Conditions on the post-impact reference angular velocity trajectory for the box.*

	${}^{p}\bar{\boldsymbol{\omega}}^{\text{ref}}(0)$	${}^{p}\bar{\boldsymbol{\omega}}^{\text{ref}}(1.01)$	${}^{p}\bar{\boldsymbol{\omega}}^{\text{ref}}(2.00)$
x	0.00	0.00	0.00
y	0.00	0.01	0.00
z	0.00	0.00	0.00

trajectory is shown in Figure 4.17. Note that, for visualization purposes, the desired orientation of the box is expressed in roll, pitch and yaw angles, expressing the desired orientation in the inertial frame. In Figure 4.18, it can be seen that at $t = 1.01$ s, the extended post-impact reference trajectory for the angular velocity coincides with the angular velocity of the box right after impact has occurred, found in the simulation presented in Section 4.2.2.
The conditions on the post-impact reference direction task trajectory ${}^{p}\bar{\mathbf{u}}^{\text{ref}}(t)$ and its derivative ${}^{p}\dot{\bar{\mathbf{u}}}^{\text{ref}}(t)$ for both KUKA arms are shown in Table 4.6. The generated extended post-impact direction trajectory is shown in Figure 4.19. Figure 4.20 shows that the associated extended post-impact reference velocity trajectory for the direction task coincides with the velocity of the direction vector at $t = 1.01$ s in the simulation detailed in Section 4.2.2.

Table 4.6: *Conditions on the post-impact reference direction task trajectories.*

		${}^{p}\bar{\mathbf{u}}^{\text{ref}}(0)$	$\bar{\mathbf{u}}^{\text{ref}}(1.01)$	${}^{p}\bar{\mathbf{u}}^{\text{ref}}(2.00)$	${}^{p}\dot{\bar{\mathbf{u}}}^{\text{ref}}(0)$	${}^{p}\dot{\bar{\mathbf{u}}}^{\text{ref}}(1.01)$	${}^{p}\dot{\bar{\mathbf{u}}}^{\text{ref}}(2.00)$
Kuka 1	x	-0.11	-0.34	-0.34	0.00	0.04	0.00
	y	0.98	0.93	0.93	0.00	0.02	0.00
	z	0.18	0.15	0.15	0.00	-0.04	0.00
Kuka 2	x	0.11	0.34	0.34	0.00	-0.04	0.00
	y	0.98	0.93	0.93	0.00	0.02	0.00
	z	-0.18	-0.15	-0.15	0.00	0.04	0.00

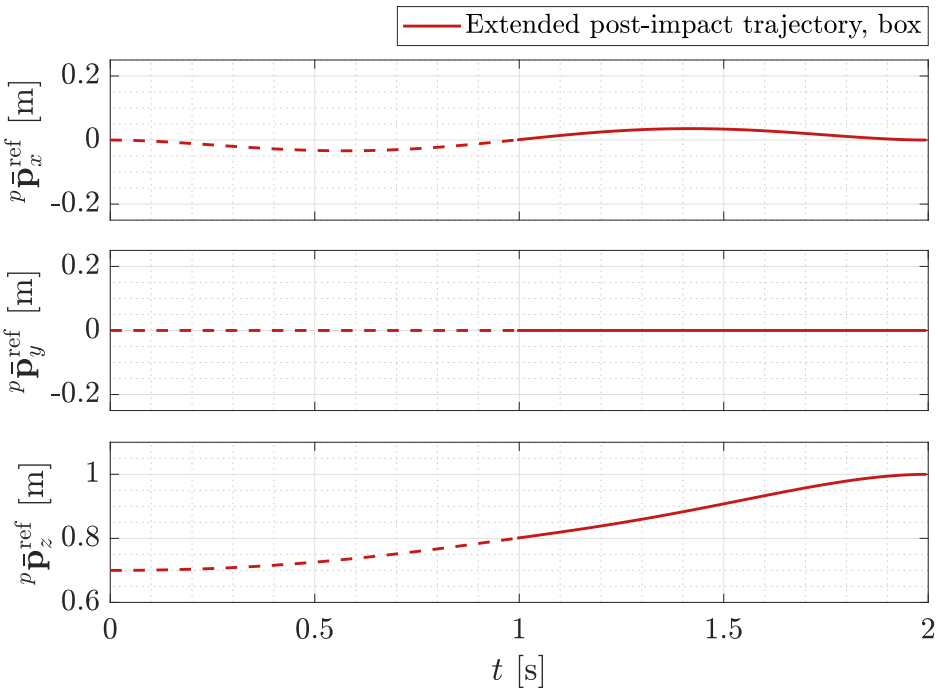


Figure 4.15: *The extended post-impact reference position trajectory for the box in x-, y- and z-direction.*

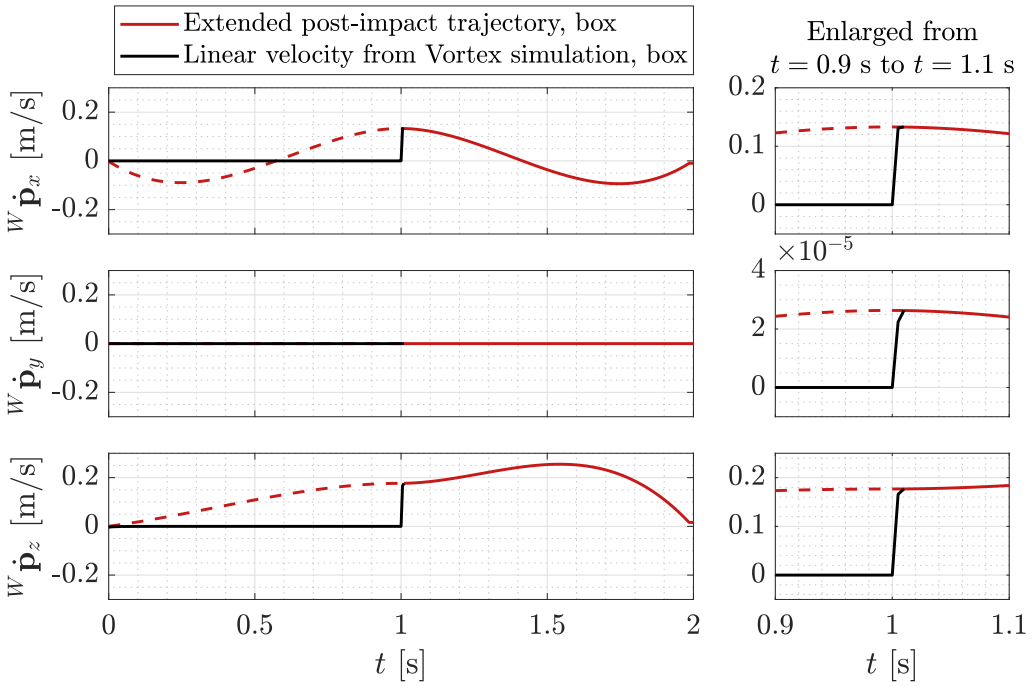


Figure 4.16: *The extended post-impact reference velocity trajectory for the position task in x-, y- and z-direction, depicted in red. At $t = 1.01$ s, the reference trajectory coincides with the velocity of the box obtained from the Vortex simulation, which is plotted in black.*

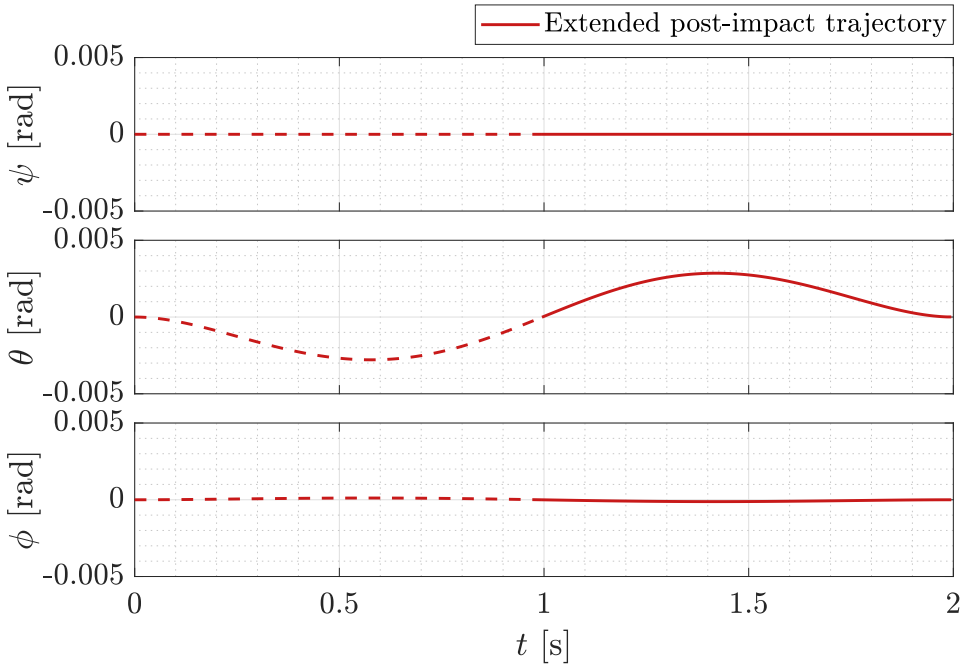


Figure 4.17: *The extended post-impact reference orientation trajectory for the box, in roll, pitch and yaw angles ψ, θ and ϕ, which define rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$, expressing the orientation of the desired frame with respect to the inertial frame in terms of the rotation about the x-, y- and z-axes.*

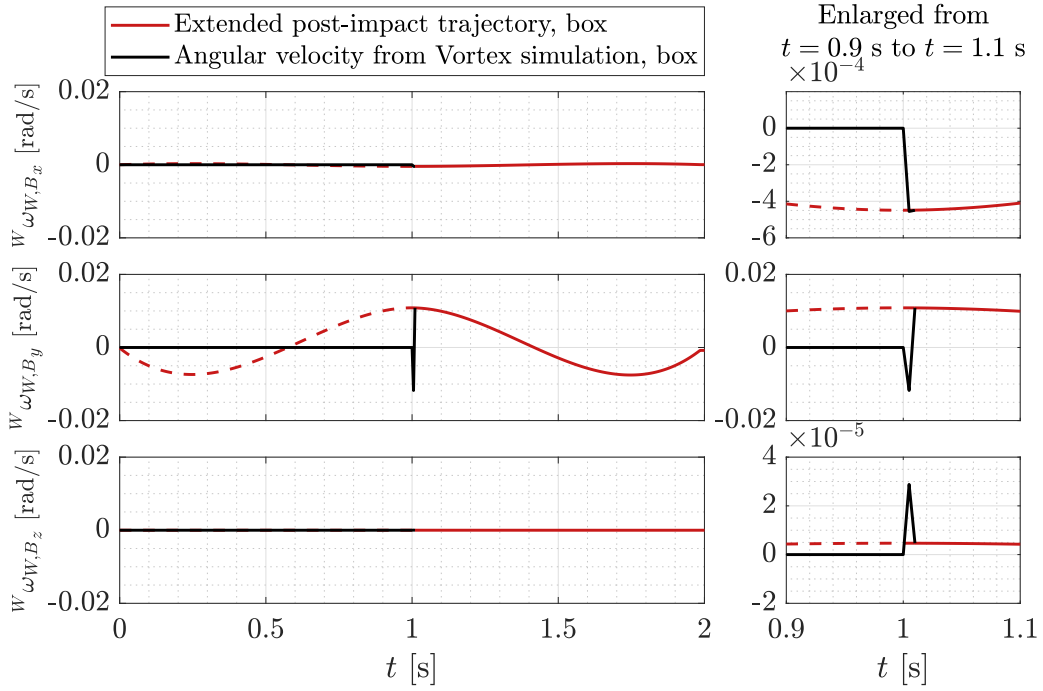


Figure 4.18: *The extended post-impact reference trajectories for the angular velocities about the inertial x-, y- and z-axis, depicted in red. At $t = 1.01$ s, the reference trajectory coincides with the angular velocity of the box obtained from the Vortex simulation, which is plotted in black.*

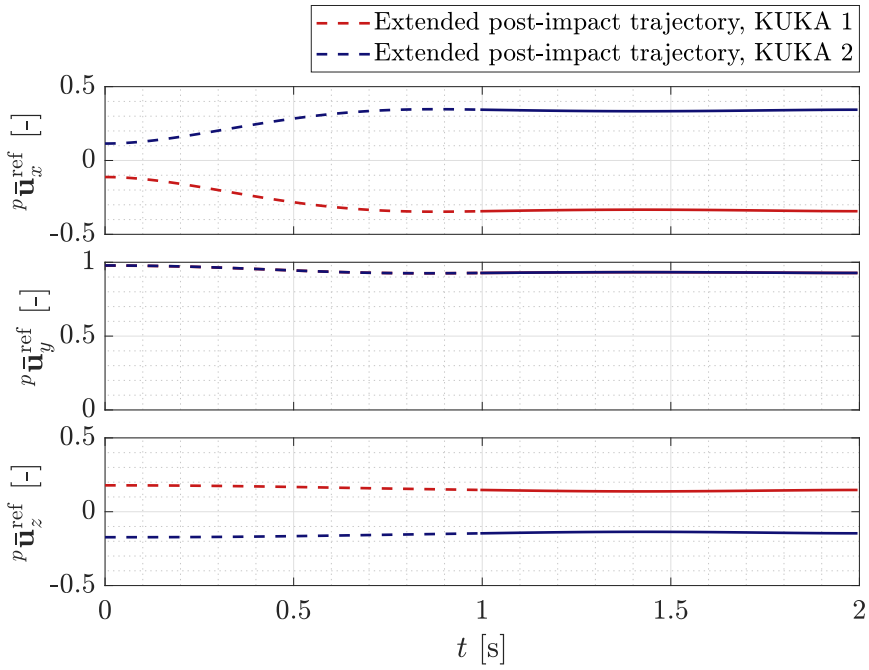


Figure 4.19: *The extended post-impact reference direction trajectory for both manipulator arms. The trajectories define the x-, y- and z-component of the direction vector.*

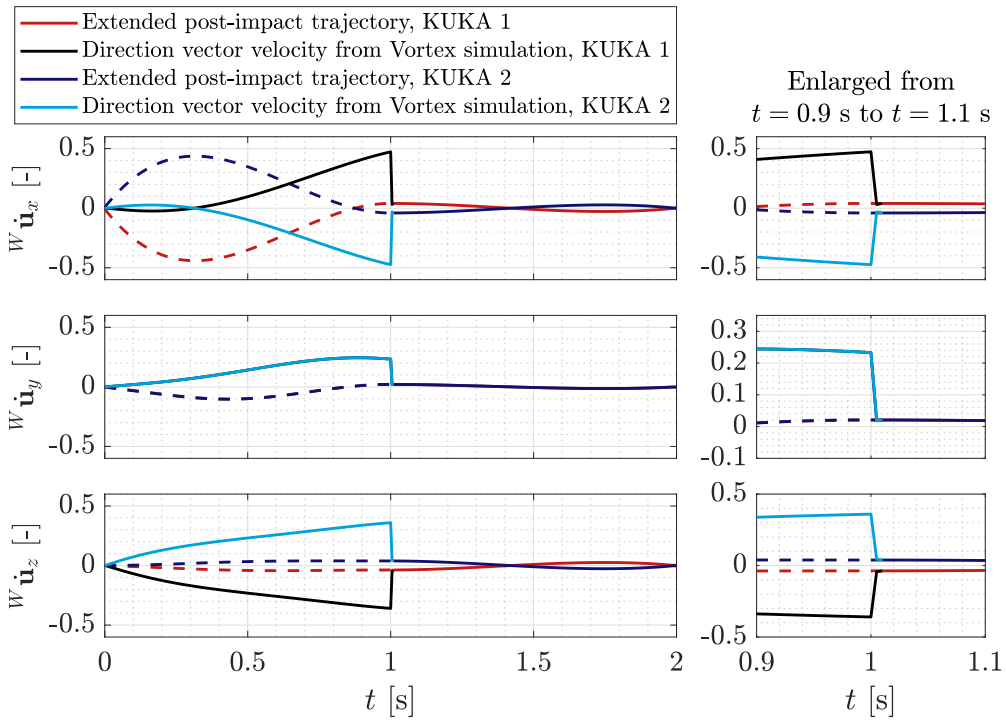


Figure 4.20: *The extended post-impact reference trajectories for the direction vector velocities, depicted in red and dark blue for KUKA 1 and KUKA 2, respectively. At $t = 1.01$ s, the reference trajectories coincide with the actual velocities of the direction vectors, which are plotted in black and cyan for KUKA 1 and KUKA 2, respectively.*

4.4 Summary

In this chapter, extended ante- and post-impact reference task trajectories have been generated for a dual arm dynamic box-lifting application, which are compatible with the robot impact dynamics. For this purpose, extended ante-impact reference task trajectories have been generated that result in simultaneous impacts between the end effectors of the dual arm manipulator and the box. The control gains and task weights have been tuned, such that accurate tracking of all ante-impact reference task trajectories is achieved. It has been shown that the ante-impact reference task trajectories implicitly define a unique state trajectory. Thereafter, simulations have been performed using the Vortex physics engine, in which the extended ante-impact reference task trajectories are tracked, in order to determine the post-impact task states that result from the simultaneous impact between the end effectors and the box. It is found that the settings of the Vortex physics engine have to be carefully chosen, as choosing the wrong settings can cause the impact dynamics to become infeasible. The post-impact task states that are found in the simulations have been used to generate extended post-impact reference task trajectories. It has been shown that these trajectories are compatible with the generated ante-impact reference task trajectories and the associated impact dynamics.

Chapter 5

Conclusions and Recommendations

Performing motion tasks with intentional impacts is complicated due to the velocity jumps that occur when contact is established at nonzero velocity. In the presence of perturbations, it cannot be assumed that the velocity jumps occur at the intended time. As a result of the mismatch in jump times, the system can reside in a different mode than the reference trajectory. Comparing the state of the system to a reference state in a different mode, in feedback control, can result in poor tracking performance or destabilization of the system. Thereby, when it is expected to perform simultaneous impacts, a perturbed system can experience more jumps than the reference trajectory, which causes intermediate unspecified modes to occur. Reference spreading provides a possible solution for the mismatch in jump times and the number of jumps, by extending the ante-impact reference trajectory beyond the intended impact time and the post-impact reference trajectory before the intended impact time, and using these trajectories to define a new tracking error. So far, reference spreading has made use of reference state trajectories, which are tracked using a state-feedback controller. However, an output feedback control approach is much better suitable for the control of complex robots, such as dual arm manipulators, as specific variables of interest can be controlled, like end effector positions. A control approach that is commonly used for such complex robots, is task-based quadratic programming (QP) control. Therefore, this research has aimed to extend reference spreading such that it can be used in task-based QP control, creating a robust QP control approach for impact aware manipulation that is suitable for the control of complex robots. Formally, the goal of this research was formulated as:

Extending reference spreading in order to be applicable for task-based QP control.

The following section concludes on the contributions that have been made in order to achieve this goal. Thereafter, recommendations for future research are formulated, based on the findings in this work.

5.1 Conclusions

Reference task trajectories, as opposed to reference state trajectories, might not prescribe a unique reference trajectory for each joint in a manipulator. This is the case when a task requires less degrees of freedom to perform than there are degrees of freedom in the manipulator, which implies that the manipulator is task redundant. For task redundant manipulators, infinitely many state trajectories can be found that result in completion of the same reference task trajectory. For motion tasks involving

intentional impacts this causes problems, as the jump map that is associated with an impact is typically strictly dependent on the joint state of the manipulator at the moment of impact. This means that the joint state after impact is dependent on the joint state before impact. Consequently, infinitely many post-impact task states exist that correspond to the same ante-impact reference task trajectory. Therefore, it is not possible to define unique post-impact reference task trajectories, that are compatible with the ante-impact reference task trajectories and the associated impact dynamics. It has been shown that by using additional tasks, the task redundancy in the manipulator can be removed. The combination of reference task trajectories, associated with these tasks, implicitly defines a unique joint state trajectory. In this case, also the impact dynamics corresponding to these reference task trajectories are unique. This allows to define post-impact reference task trajectories that are compatible with these impact dynamics. This concludes the first contribution that has been made in this research, which was formulated as:

> *Defining reference trajectories for motion tasks with intentional impacts that are compatible with the impact dynamics and suitable for task-based QP control.*

Using the concept of reference spreading, the reference task trajectories can be extended beyond the intended impact time. In this way, it is possible to prevent the task errors from comparing the system to a reference trajectory in a different mode, when perturbations cause the system to jump at a different time than expected. A QP control framework had to be developed, in order to switch between the extended ante- and post-impact reference task trajectories and in order to deal with the intermediate unspecified mode, which occurs when the simultaneity of an impact is lost due to perturbations. The generation of the extended reference task trajectories and the QP control framework are dependent on the robot and the application that are considered. Therefore, a dynamic box-lifting application for a specific dual arm manipulator has been introduced. Three tasks and the associated reference task trajectories have been detailed, in order to specify the desired motion for this application. A position and orientation task have been used to specify the time-varying desired pose of the end effectors of the manipulator before impact. After contact has been established, the position and orientation task are used to specify the time-varying desired pose of the box. In addition, a direction task has been used on the fourth link of both manipulator arms, in order to remove the task redundancy in the manipulator. An approach for the generation of extended reference task trajectories, based on the De Casteljau algorithm, has been proposed for each of the selected tasks. This approach allows to generate ante-impact reference task trajectories that result in the desired impact and post-impact reference task trajectories that are compatible with the associated impact dynamics. Thereby, the ante- and post-impact reference task trajectories can be extended beyond the impact time. This concludes the second contribution of this research, which was formulated as:

> *Developing a trajectory generation and extension procedure for the reference trajectories associated with specific QP control tasks.*

A QP control framework has been proposed, based on reference spreading, with which the dynamic box-lifting application can be performed by the dual arm manipulator, in the presence of perturbations. For this purpose, a controller has been defined consisting of three sequential modes. In the first mode, the ante-impact mode, the manipulator tracks the generated extended ante-impact reference task trajectories, until an impact is detected. In case perturbations cause the simultaneity of the impact to be lost, the controller switches to the intermediate mode. In the intermediate mode, the extended ante-impact reference task trajectories are tracked as well, in order to complete the full contact. However, the joint velocities of the manipulator, which have jumped as a result of the impact,

are likely not reliable and cannot be used for feedback control in this mode. Therefore, an open loop control strategy has been proposed, which uses the reference joint positions and velocities for the computation of the joint torques in the QP controller, instead of the actual joint states measured on the robot. A second approach has been proposed, in which the actual joint positions are used in the QP controller, in combination with the reference joint velocities, as the joint positions are not affected by impacts. However, this option should be carefully investigated, since this approach imposes an incoherency between the joint positions and velocities in the robot model. Once the surface contact has been completed, the controller switches to the post-impact mode. In the post-impact mode, the manipulator arms and the box perform a constrained motion, prescribed by the extended post-impact reference task trajectories. A force task has been used to specify the desired contact forces between the end effectors and the box, such that the contact is maintained. The proposed controller provides a possible control strategy for performing the dual arm dynamic box-lifting application, which concludes the third contribution:

> *Defining a task-based QP control framework, based on reference spreading, with which motion tasks involving simultaneous impacts can be performed, in the presence of perturbations.*

To conclude, an extension of reference spreading has been proposed, which can be implemented in a task-based QP control framework. Herewith, the goal of this research, formulated in Section 1.3 is achieved.

A numerical simulation study has been performed on the dynamic box-lifting application for a dual arm manipulator in a robotic simulator environment. It has been shown how extended reference task trajectories can be generated for this particular application, which are compatible with the robot impact dynamics.
First, extended ante-impact reference task trajectories have been generated, which prescribe the desired motion for the dual arm manipulator. It has been shown that when the manipulator achieves tracking of the reference task trajectories, its joint states track the implicitly defined state trajectory. This is required in order to find post-impact reference task trajectories that are compatible with the ante-impact trajectories and the associated impact dynamics.
Thereafter, simulations have been performed in order to determine the post-impact task states that correspond to the generated ante-impact trajectories, using the Vortex physics engine. It was seen that for the determination of the post-impact states, it is important that the correct settings for the physics engine are used. Using the wrong settings can result in infeasible post-impact states.
Extended post-impact reference task trajectories have been generated, which coincide with the determined post-impact states, at impact time. This shows that these post-impact trajectories are compatible with the ante-impact trajectories and the associated impact dynamics. Since the ante- and post-impact reference task trajectories have been extended beyond the extended impact time, they are suitable to be used in the proposed QP control framework. This concludes the last contribution, formulated as:

> *Demonstrating the extended reference trajectory generation procedure by means of a numerical simulation example.*

Due to limited time, the proposed QP control framework has not been fully implemented and validated in the numerical simulation study. This remains a topic for future research.

5.2 Recommendations

As detailed in the previous concluding section, the goal of this research has been achieved. However, the proposed QP control framework for task-based reference spreading has not been fully validated using numerical simulations. Thereby, several interesting topics for future research can be thought of, based on the findings in this work. Therefore, recommendations for future research have been formulated, which are listed below.

Implementation of task-based QP control based on reference spreading
The numerical simulation study performed in this work has shown how extended ante- and post-impact reference task trajectories, compatible with robot impact dynamics, can be generated by means of the trajectory generation procedure presented in this work, based on the De Casteljau algorithm. However, the proposed QP control framework for task-based reference spreading has not been fully validated with numerical simulations. So far, only the ante-impact mode of the proposed controller has been fully implemented in the software framework. Therefore, future research should focus on the theoretical and software related challenges that come with the implementation of the intermediate and post-impact mode.
First, the post-impact mode has to be implemented, in order to validate that the desired motion described by the extended post-impact reference task trajectories can be performed by the manipulator. First steps have been made in the implementation of this mode. However, due to software related issues, the contact constraints between the end effectors and the box do not result in the desired outcome when used in combination with a torque controlled manipulator.
Once the post-impact mode has been fully implemented and the manipulator is capable of tracking the extended post-impact reference task trajectories, the intermediate mode can be implemented. Ideally, both proposed approaches for the intermediate mode should be implemented, validated, and compared. Having all modes implemented allows to perform simulations of the full box-lifting application in the presence of perturbations, which can be used to validate the proposed task-based QP control framework based on reference spreading.

Numerical simulation study on performance
Once the task-based QP control framework is fully implemented and validated, a numerical simulation study should be done on the performance of the controller in terms of execution time and robustness to perturbations. For this purpose, task-based reference spreading should be compared to existing control strategies for task-based QP control with contact transitions.

Stability proof for task-based reference spreading
Even if the task-based QP control implementation is fully validated using numerical simulations, it cannot be guaranteed that a robot converges to the post-impact reference task trajectories. For this purpose, a stability analysis, similar to [17], could be performed for systems with simultaneous impacts, controlled using task-based QP control.

Task-based reference spreading for different robots
This research has focused specifically on an application for a torque controlled dual arm manipulator. However, the same conceptual approach of task-based reference spreading could be used for other robots as well. Interesting robots that can be considered in future research are floating-base robots like humanoids and robots that are controlled using position or velocity control loops. The main reason why this research has focused specifically on torque controlled robots, even though the proposed approach is straightforwardly applicable to position controlled robots as well, is due to the incorrect implementation

of position control in the V-REP simulator. Therefore, in order to apply task-based reference spreading on position controlled robots in numerical simulations, the position control implementation of V-REP has to be changed or another robot simulator has to be used.

Physical experiments
Once task-based reference spreading has been fully validated using numerical simulations, physical experiments can be performed. Ideally, experiments should be performed on an box-lifting application using a torque controlled dual arm manipulator, as discussed in this research, in order to validate the approach proposed in this research. However, if this is not possible, task-based reference spreading could be applied to other applications and manipulators as well, using a similar approach.

Impact detection method
The impact detection methods that are proposed in this research, in order to detect the start and end of the intermediate mode, are suitable for numerical simulations, but not directly applicable to physical experiments. Therefore, future research should focus on developing an impact detection method that is capable of detecting the beginning and end of the impact phase, in order to switch between the controller modes.

Physics engine comparison
In this work, the Vortex physics engine is used to simulate the impact dynamics. Apart from Vortex, multiple other physics engines are available in V-REP and other dynamic system simulators. An interesting topic for future research is to investigate the differences in impact modeling between the physics engines and to perform a comparison using numerical simulations.

Simulating impact dynamics
In the simulation environment used in this work, it is not possible to initialize robots or objects with a velocity different from zero. Therefore, in order to simulate the impact dynamics for an impact at a desired velocity, first the manipulator has to be controlled to the desired velocity. However, this approach is very time-consuming when a large variety of velocities is considered. In this case, it would be desireable to be able to define the initial velocity of the manipulator, such that many different impact velocities can be quickly evaluated. Ideally, one would even be able to access the physics engine directly in order to compute the impact dynamics corresponding to an impact with a specific robot state.

Bibliography

[1] AIST, "Development of a Humanoid Robot Prototype, HRP-5P, Capable of Heavy Labor." https://www.aist.go.jp/aist_e/list/latest_research/2018/20181116/en20181116.html, September 2018.

[2] KUKA, "KUKA ready2 load." https://www.kuka.com/hu-hu/term%C3%A9kek-szolg%C3%A1ltat%C3%A1sok/robotrendszerek/ready2_use/kuka-ready2_load, March 2019.

[3] S. Sina, M. Salehian, and A. Billard, "A Dynamical System Based Approach for Controlling Robotic Manipulators During Non-contact / Contact Transitions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018.

[4] B. Brogliato, *Nonsmooth Mechanics*. London: Springer-Verlag, 3rd ed., 1999.

[5] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3356–3363, Sep. 2008.

[6] J. W. Grizzle and E. R. Westervelt, "Hybrid zero dynamics of planar bipedal walking," in *Analysis and Design of Nonlinear Control Systems*, pp. 223–237, Springer, 2008.

[7] B. G. Buss, K. A. Hamed, B. A. Griffin, and J. W. Grizzle, "Experimental Results for 3D Bipedal Robot Walking Based On Systematic Optimization of Virtual Constraints," *American Control Conference*, pp. 4785–4792, 2016.

[8] A. D. Ames, "Human-Inspired Control of Bipedal Walking Robots," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1115–1130, 2014.

[9] H. Zhao, J. Horn, J. Reher, V. Paredes, and A. D. Ames, "A Hybrid Systems and Optimization-Based Control Approach to Realizing Multi-Contact Locomotion on Transfemoral Prostheses," *IEEE Conference on Decision and Control*, pp. 1607–1612, 2015.

[10] I. C. Morarescu and B. Brogliato, "Trajectory tracking control of multiconstraint complementarity lagrangian systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 6, pp. 1300–1313, 2010.

[11] P. R. Pagilla and B. Yu, "A stable transition controller for constrained robots," *IEEE/ASME transactions on mechatronics*, vol. 6, no. 1, pp. 65–74, 2001.

[12] B. Yu and P. R. Pagilla, "Design and implementation of a robust switching control scheme for a class of constrained robot tasks," *International Journal of Systems Science*, vol. 37, no. 5, pp. 303–321, 2006.

[13] J. B. Biemond, W. M. H. Heemels, R. G. Sanfelice, and N. van de Wouw, "Distance function design and lyapunov techniques for the stability of hybrid trajectories," *Automatica*, vol. 73, pp. 38–46, 2016.

[14] M. Baumann, J. J. B. Biemond, R. I. Leine, and N. V. D. Wouw, "Synchronization of impacting mechanical systems with a single constraint," *Physica D-Nonlinear Phenomena*, vol. 362, pp. 9–23, 2018.

[15] T. Yang, F. Wu, and U. States, "Tracking Control of Hybrid Systems with State-Triggered Jumps and stochastic events and its application," *IET Control Theory & Applications*, vol. 11, no. January, pp. 1024–1033, 2017.

[16] A. Saccon, N. van de Wouw, and H. Nijmeijer, "Sensitivity analysis of hybrid systems with state jumps with application to trajectory tracking," *IEEE Conference on Decision and Control*, vol. 53, pp. 3065–3070, 2014.

[17] M. W. Rijnen, J. J. Biemond, A. Saccon, N. V. D. Wouw, and H. Nijmeijer, "Hybrid Systems with State-Triggered Jumps: Sensitivity-Based Stability Analysis with Application to Trajectory Tracking," *Under review for IEEE Transactions on Automatic Control*, 2018.

[18] M. Rijnen, H. L. Chen, N. Van De Wouw, A. Saccon, and H. Nijmeijer, "Sensitivity Analysis for Trajectories of Nonsmooth Mechanical Systems with Simultaneous Impacts: A Hybrid Systems Perspective," *American Control Conference (ACC)*, accepted for publication, 2019.

[19] M. Rijnen, E. De Mooij, S. Traversaro, F. Nori, N. Van De Wouw, A. Saccon, and H. Nijmeijer, "Control of humanoid robot motions with impacts: Numerical experiments with reference spreading control," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4102–4107, 2017.

[20] A. Konno, T. Myojin, T. Matsumoto, T. Tsujita, and M. Uchiyama, "An impact dynamics model and sequential optimization to generate impact motions for a humanoid robot," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1596–1608, 2011.

[21] T. Matsumoto, A. Konno, L. Gou, and M. Uchiyama, "A humanoid robot that breaks wooden boards applying impulsive force," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5919–5924, IEEE, 2006.

[22] S. Traversaro and A. Saccon, "Multibody dynamics notation, revision 2." Available online at `research.tue.nl`, 2019.

[23] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215–221, 1955.

[24] M. W. L. M. Rijnen, *Enabling motions with impacts in robotic and mechatronic systems.* PhD thesis, Eindhoven University of Technology, Dynamics and Control group, November 2018.

[25] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems," *IEEE Control Systems Magazine*, vol. 29, no. 2, pp. 28–93, 2009.

[26] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-character physical and behavioral interactions controller," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 6, pp. 1650–1662, 2017.

[27] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4414–4419, IEEE, 2011.

[28] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[29] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet, E. Yoshida, S. Kajita, and F. Kanehiro, "Multi-contact vertical ladder climbing with an HRP-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.

[30] Copellia, "V-REP virtual robot experimentation platform." `http://www.coppeliarobotics.com/`.

[31] cm labs, "Vortex Studio." `https://www.cm-labs.com/vortex-studio/`.

[32] J. Gravesen, "Differential geometry and design of shape and motion," *Department of Mathematics, Technical University of Denmark*, 2002.

[33] F. Bullo and R. M. Murray, "Proportional derivative (pd) control on the euclidean group," in *European Control Conference*, vol. 2, pp. 1091–1097, 1995.

[34] P. Crouch, G. Kun, and F. S. Leite, "The de casteljau algorithm on lie groups and spheres," *Journal of Dynamical and Control Systems*, vol. 5, no. 3, pp. 397–429, 1999.

[35] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, 2019.

[36] cm labs, "Coppelia Robotics." `http://www.coppeliarobotics.com/helpFiles/en/shapeDynamicEngineProperties.htm`.

Appendix A

PD Control on SO(3)

The cost function for the orientation task can be derived from the PD controller for the stabilization of the origin of SO(3), proposed in [33]. This PD controller is given by

$$\dot{\mathbf{E}} = \boldsymbol{\Omega}_E \mathbf{E} \tag{A.1}$$

$$\dot{\boldsymbol{\omega}}_E = -\boldsymbol{K}_d \boldsymbol{\omega}_E - \boldsymbol{K}_p (\log(\mathbf{E}))^\vee, \tag{A.2}$$

with $\mathbf{E} \in \mathrm{SO}(3)$, and $\boldsymbol{\Omega} = \boldsymbol{\omega}^\wedge \in \mathbb{R}^{3\times 3}, \boldsymbol{\Omega} + \boldsymbol{\Omega}^T = 0$ ($\boldsymbol{\Omega}$ is skew symmetric). The rotation error $\mathbf{E}$ is defined as the rotation matrix from the desired frame D to the current body frame B, so

$$\mathbf{E} := {}^B\mathbf{R}_W \; {}^W\mathbf{R}_D. \tag{A.3}$$

The angular velocity $\boldsymbol{\Omega}_E$ denotes the angular velocity of the desired frame with respect to the current frame, expressed in the current frame, such that

$$\boldsymbol{\Omega}_E = {}^B\boldsymbol{\Omega}_{B,D}. \tag{A.4}$$

For the sake of clarity, let $\mathbf{D} := {}^W\mathbf{R}_D$, $\mathbf{B} := {}^W\mathbf{R}_B$, $\mathbf{E} := \mathbf{R}^T\mathbf{D}$, $\boldsymbol{\Omega}_D := {}^W\boldsymbol{\Omega}_{W,D}$ and $\boldsymbol{\Omega}_B := {}^W\boldsymbol{\Omega}_{W,B}$. From (A.1), it follows that

$$\begin{aligned}
\dot{\mathbf{B}}^T\mathbf{D} + \mathbf{B}^T\dot{\mathbf{D}} &= \boldsymbol{\Omega}_E\mathbf{B}^T\mathbf{D}, \\
\dot{\mathbf{B}}^T + \mathbf{B}^T\dot{\mathbf{D}}\mathbf{D}^T &= \boldsymbol{\Omega}_E\mathbf{B}^T, \\
\dot{\mathbf{B}}^T\mathbf{B} + \mathbf{B}^T\dot{\mathbf{D}}\mathbf{D}^T\mathbf{B} &= \boldsymbol{\Omega}_E, \\
-\mathbf{B}^T\boldsymbol{\Omega}_B\mathbf{B} + \mathbf{B}^T\boldsymbol{\Omega}_D\mathbf{B} &= \boldsymbol{\Omega}_E, \\
\mathbf{B}^T\left(\boldsymbol{\Omega}_D - \boldsymbol{\Omega}_B\right)\mathbf{B} &= \boldsymbol{\Omega}_E.
\end{aligned} \tag{A.5}$$

This equation can be transformed from skew-symmetric angular velocity matrices to angular velocity vectors using $\boldsymbol{\omega} = \boldsymbol{\Omega}^\vee$, which results in

$$\boldsymbol{\omega}_E = \mathbf{B}^T\left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right). \tag{A.6}$$

Then, an expression can be found for $\dot{\boldsymbol{\omega}}_E$ by taking the derivative of this equation

$$\begin{aligned}
\dot{\boldsymbol{\omega}}_E &= \dot{\mathbf{B}}^T \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \mathbf{B}^T \left(\dot{\boldsymbol{\omega}}_D - \dot{\boldsymbol{\omega}}_B\right), \\
\dot{\boldsymbol{\omega}}_E &= -\mathbf{B}^T \boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \mathbf{B}^T \left(\dot{\boldsymbol{\omega}}_D - \dot{\boldsymbol{\omega}}_B\right), \\
\dot{\boldsymbol{\omega}}_E &= -\mathbf{B}^T \left[\boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \dot{\boldsymbol{\omega}}_D - \dot{\boldsymbol{\omega}}_B\right], \\
\mathbf{B}\dot{\boldsymbol{\omega}}_E &= -\boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \dot{\boldsymbol{\omega}}_D - \dot{\boldsymbol{\omega}}_B. \qquad \text{(A.7)}
\end{aligned}$$

This equation can be rewritten into

$$\dot{\boldsymbol{\omega}}_D = \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \mathbf{B}\dot{\boldsymbol{\omega}}_E. \qquad \text{(A.8)}$$

Implementing the control law (A.2) gives

$$\dot{\boldsymbol{\omega}}_D = \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) + \mathbf{B}\left(-\boldsymbol{K}_d \boldsymbol{\omega}_E - \boldsymbol{K}_p (\log(\mathbf{E}))^\vee\right), \qquad \text{(A.9)}$$

which, by substitution of (A.6), is equal to

$$\dot{\boldsymbol{\omega}}_D = \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) - \mathbf{B}\boldsymbol{K}_d \mathbf{B}^T \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) - \mathbf{B}\boldsymbol{K}_p \mathbf{B}^T \mathbf{B} (\log(\mathbf{E}))^\vee. \qquad \text{(A.10)}$$

If the gains $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are scalar, so $\boldsymbol{K}_d = k_d$ and $\boldsymbol{K}_p = k_p$ then

$$\mathbf{B}\boldsymbol{K}_d \mathbf{B}^T = k_d, \qquad \text{(A.11)}$$

$$\mathbf{B}\boldsymbol{K}_p \mathbf{B}^T = k_p, \qquad \text{(A.12)}$$

such that (A.10) equals

$$\dot{\boldsymbol{\omega}}_D = \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) - k_d \left(\boldsymbol{\omega}_D - \boldsymbol{\omega}_B\right) - k_p \mathbf{B} (\log(\mathbf{E}))^\vee. \qquad \text{(A.13)}$$

This equation can be used as the cost function for the orientation task.

Appendix B

Position Task Trajectory Control Points

Using the De Casteljau algorithm, polynomials are computed known as Bézier curves. The general formulation of a Bézier curve on $\mathbb{R}^3$ of degree d, denoted by $\boldsymbol{B}(\lambda)$, is given by

$$\boldsymbol{B}(\lambda) = \sum_{i=0}^{d} \binom{d}{i} (1-\lambda)^{d-i} \lambda^i C_{i+1}, \tag{B.1}$$

where $\binom{d}{i}$ denotes the binomial coefficient, $\lambda \in [0,1]$ and $C_{i+1} \in \mathbb{R}^3$ the control points. A polynomial on the time interval $t \in [t_0, t_f]$ is found by choosing

$$\lambda = \frac{t - t_0}{t_f - t_0}. \tag{B.2}$$

As follows from (B.1), the Bézier polynomial with $d = 4$ is given by

$$\boldsymbol{B}(\lambda) = (1-\lambda)^4 C_1 + 4\lambda(1-\lambda)^3 C_2 + 6\lambda^2(1-\lambda)^2 C_3 + 4\lambda^3(1-\lambda) C_4 + \lambda^4 C_5. \tag{B.3}$$

The first order derivative of this polynomial with respect to t writes

$$\boldsymbol{B}'(\lambda) = \frac{4(1-\lambda)^3 (C_2 - C_1) + 12\lambda(1-\lambda)^2 (C_3 - C_2) + 12\lambda^2(1-\lambda) (C_4 - C_3) + 4\lambda^3 (C_5 - C_4)}{t_f - t_0}, \tag{B.4}$$

and the second order derivative

$$\boldsymbol{B}''(\lambda) = \frac{12(1-\lambda)^2 (C_3 - 2C_2 + C_1) + 24\lambda(1-\lambda) (C_4 - 2C_3 + C_2) + 12\lambda^2 (C_5 - 2C_4 + C_3)}{(t_f - t_0)^2}. \tag{B.5}$$

Using these definitions, the choices for control points C_{i+1} can be derived, which result in a Bézier curve that satisfies boundary conditions at $t = t_0$ and $t = t_f$, for which $\lambda = 0$ and $\lambda = 1$ hold respectively. Such boundary conditions are given by

$$\boldsymbol{B}(0) = \mathbf{p}_0, \tag{B.6a}$$
$$\boldsymbol{B}(1) = \mathbf{p}_f, \tag{B.6b}$$
$$\boldsymbol{B}'(0) = \dot{\mathbf{p}}_0, \tag{B.6c}$$
$$\boldsymbol{B}'(1) = \dot{\mathbf{p}}_f, \tag{B.6d}$$
$$\boldsymbol{B}''(1) = \ddot{\mathbf{p}}_f, \tag{B.6e}$$

for a trajectory with a boundary condition on the acceleration at $t = t_f$. Evaluating (B.3) - (B.5) at $\lambda = 0$ and $\lambda = 1$ results in

$$\boldsymbol{B}(0) = C_1, \tag{B.7a}$$
$$\boldsymbol{B}(1) = C_5, \tag{B.7b}$$
$$\boldsymbol{B}'(0) = \frac{4\,(C_2 - C_1)}{t_f - t_0}, \tag{B.7c}$$
$$\boldsymbol{B}'(1) = \frac{4\,(C_5 - C_4)}{t_f - t_0}, \tag{B.7d}$$
$$\boldsymbol{B}''(1) = \frac{12\,(C_5 - 2C_4 + C_3)}{(t_f - t_0)^2}. \tag{B.7e}$$

Combining (B.6) and (B.7) gives the choices for the control points

$$C_1 = \mathbf{p}_0, \tag{B.8a}$$
$$C_2 = C_1 + \frac{1}{4}\dot{\mathbf{p}}_0(\tau - t_0), \tag{B.8b}$$
$$C_3 = \frac{1}{12}(\tau - t_0)^2\ddot{\mathbf{p}}_f - C_5 + 2C_4, \tag{B.8c}$$
$$C_4 = C_5 - \frac{1}{4}\dot{\mathbf{p}}_f(\tau - t_0), \tag{B.8d}$$
$$C_5 = \mathbf{p}_f. \tag{B.8e}$$

By choosing the control points as (B.8), a polynomial is computed using the De Casteljau algorithm that satisfies the conditions given by (B.6).
In case a trajectory has a boundary condition on the acceleration at $t = t_0$, (B.6e) is replaced by

$$\boldsymbol{B}''(0) = \ddot{\mathbf{p}}_0. \tag{B.9}$$

Evaluation of (B.5) at $\lambda = 0$ gives

$$\boldsymbol{B}''(0) = \frac{12\,(C_3 - 2C_2 + C_1)}{(t_f - t_0)^2}, \tag{B.10}$$

such that the third control point is given by

$$C_3 = \frac{1}{12}(t_f - t_0)^2\ddot{\mathbf{p}}_0 - C_1 + 2C_2. \tag{B.11}$$

Appendix C

Perturbed Initial Configurations

In order to achieve accurate tracking of the ante-impact reference task trajectories, the control gains and weights for each task are tuned on the basis of simulations with perturbed initial configurations for the dual arm manipulator. The joint positions for the unperturbed and perturbed initial configurations of both KUKA arms are summarized in Tables C.1 and C.2.

Table C.1: *Initial joint positions associated with the unperturbed and perturbed configurations for KUKA 1.*

	No perturbation	1	2	3	4	5	6
Joint 0	6.68	20	15	10	3	-2	-7
Joint 1	-52.11	-65	-60	-55	-50	-45	-40
Joint 2	0.12	13	8	3	-3	-8	-13
Joint 3	-74.56	-88	-83	-78	-72	-67	-62
Joint 4	-0.11	-13	-8	-3	3	8	13
Joint 5	-53.36	-66	-61	-56	-50	-45	-40
Joint 6	36.57	50	45	40	33	28	23

Table C.2: *Initial joint positions associated with the unperturbed and perturbed configurations for KUKA 2.*

	No perturbation	1	2	3	4	5	6
Joint 0	-6.68	-20	-15	-10	-3	2	7
Joint 1	52.11	65	60	55	50	45	40
Joint 2	-0.12	-13	-8	-3	3	8	13
Joint 3	74.56	88	83	78	72	67	62
Joint 4	0.11	13	8	3	-3	-8	-13
Joint 5	53.36	66	61	56	50	45	40
Joint 6	-36.57	-50	-45	-40	-33	-28	-23

Appendix D

Joint State Trajectory Tracking

Figure D.1 shows that the joints converge to the same joint state trajectory within the first second for all considered initial configurations, as a result of the position, orientation, and direction task. In case the direction task is not used, the joints do not converge to the same joint states as seen in Figure D.2.

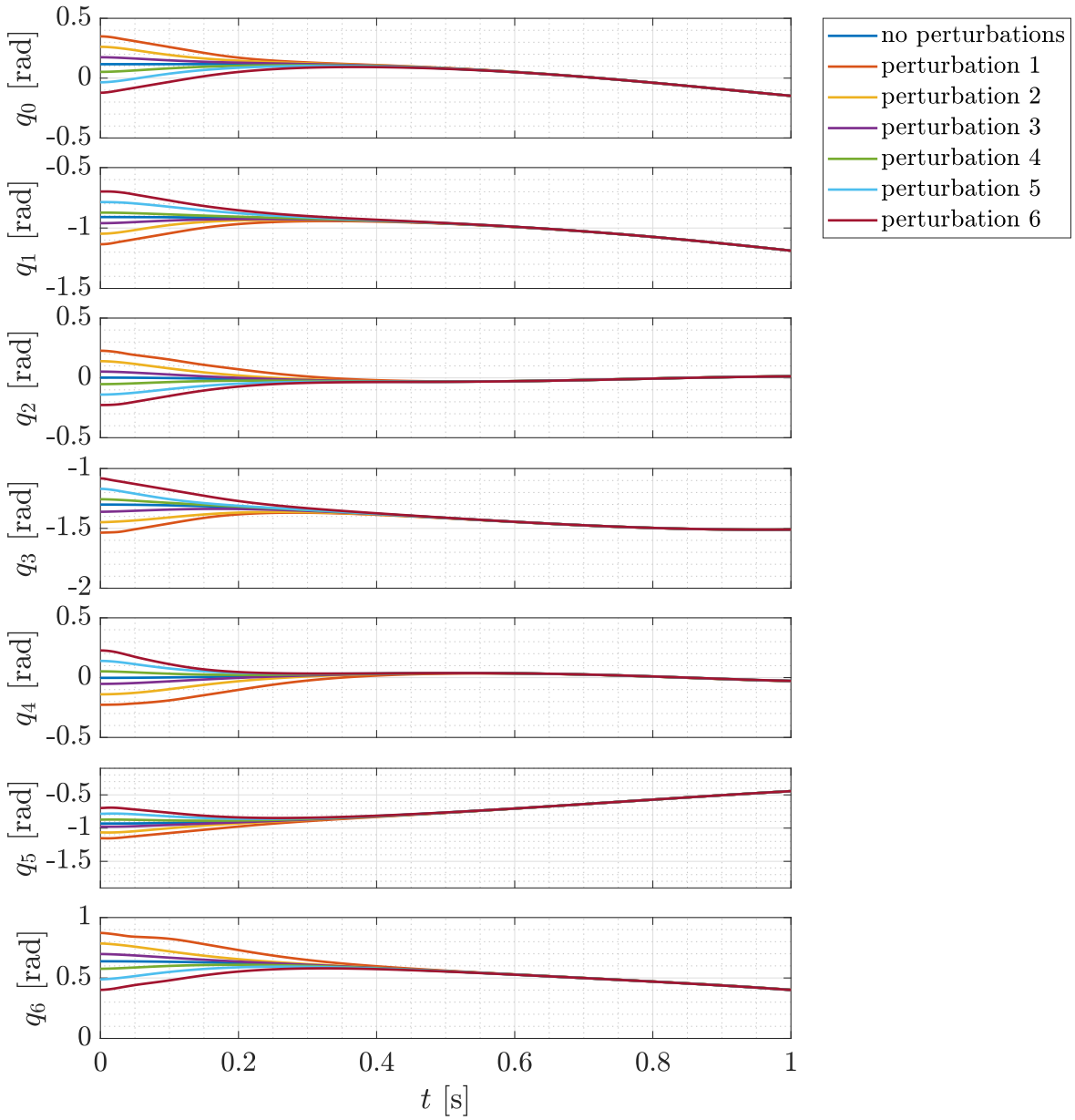


Figure D.1: *Convergence of the joint positions of all 7 joints in KUKA 1 to a unique state trajectory as a result of the position, orientation, and direction task.*

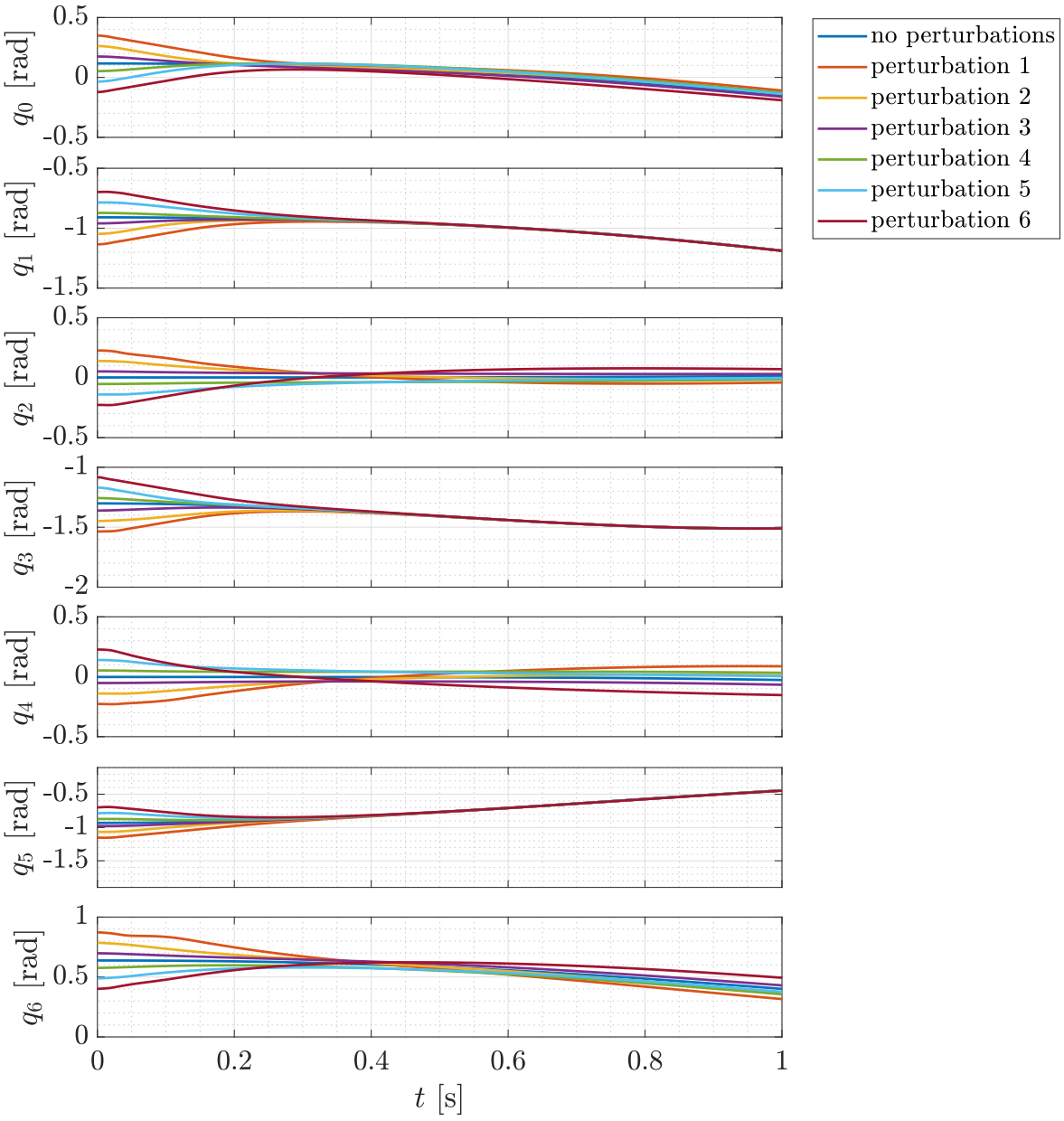


Figure D.2: *The joint positions of the 7 joints in KUKA 1 do not all converge to a unique state trajectory when only a position and orientation task are used.*

Technische Universiteit
Eindhoven
University of Technology

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

01-05-2019

Name

Casper Beumer

ID-number

0854071

Signature

Beumer

Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU