MASTER

Aim-Aware Collision Monitoring for Robot-Environment Edge Impacts

Proper, B.W.B.

Award date:
2021

Department of Mechanical Engineering
Dynamics and Control section

Aim-Aware Collision Monitoring for Robot-Environment Edge Impacts

MSc. Graduation Project - Project Phase Report
DC 2021.081

B.W.B. Proper (0959190)

Coaches:
ir. J.J. van Steen
ir. A. Kurdas TUM, Munich, Germany
dr. ir. S. Abdolshah TUM, Munich, Germany

Supervisor:
dr. ir. A. Saccon

Additional MSc. Committee Members:
prof. dr. ir. N. van de Wouw (Chair)
dr. ir. R. van de Molengraft

Eindhoven, September 27, 2021

Abstract

Currently, robots are experiencing a growing presence in the logistics sector due to the scarcity of labor and an increasing presence of e-commerce in day-to-day life. However, current robotic systems are not able to take advantage of impacts to reduce the cycle time of a pick-and-place operation due to the requirement of establishing contact with the environment at near-zero velocities to minimize the risk of damaging the robot or its environment. To improve the performance of robotic systems establishing contact with the environment, methods to predict, exploit, and classify impact behavior, which are all part of impact-aware manipulation, are required.

It has been shown that the rigid component of an oscillatory impact response can be predicted through a rigid impact map. This project aims at using this predicted post-impact velocity to generate a trajectory that serves as the expected and desired motion of the robot that can be used to classify the expectancy of post-impact behavior of a realistic system. Furthermore, the classification method focuses on geometric differences between the expected and real environment as the source of unexpected behavior. Due to the similarities to placing a box on a surface, another focus will be placed on simultaneous impact scenarios.

To develop this classification method, a planar approximation of the Franka Emika Panda is used to simulate impact behavior. The simulation has models with varying degrees of complexity, such as simulating the robotic manipulator as a flexible or rigid joint robotic manipulator and whether a rigid impact or an impact with compliant contact dynamics is simulated. For the impact simulations, a spherical end-effector is used to simulate single impact cases, whereas an edge end-effector is used to simulate simultaneous impact cases.

During this project, a method is proposed to generate the expected post-impact trajectory using the predicted post-impact velocity obtained by using a rigid impact map. Furthermore, this trajectory is shown to match the rigid component of a realistic post-impact response. Through the use of a novel filter design that generates an envelope around an oscillatory signal to approximate the location of the rigid component, the realistic post-impact response is classified using the expected post-impact trajectory. Finally, the classification method is applied to a set of single- and simultaneous impact cases, where the performance of the classification method in classifying the expectancy of the impact scenarios is shown to be satisfactory.

Overall, the results presented in this report achieve the goal that was laid out at the start. The designed classifier can classify expected single- and simultaneous impact scenarios and is also tunable based on the requirements set by the user. Currently, the classifier is only tested in simulations and the next step is to validate the obtained results on real hardware.

Acknowledgments

With the completion of my graduation project and being one step closer to graduating, I would like to take a moment and thank the people that helped me along the way with their guidance and support. These people have helped me not only during my graduation project but also during the rest of my studies and I would not be where I am today without them.

First of all, I would like to express my gratitude towards my supervisor, Alessandro Saccon, your valuable feedback has allowed me to steadily improve the quality of my work over the course of my graduation project. Despite the rough start of the project phase and the misunderstandings my writing style often created, you were adamant in guiding me through the project and making sure that I had a clear path to continue researching a solution to the problem. At several points during my graduation project, your insights were crucial in allowing me to make a lot of progress in a short time and this allowed me to achieve the goal that was set out at the start in the given time available for my project. I would also like to thank Alexander Kurdas, Jari van Steen, Maarten Jongeneel, and Saeed Abdolshah for providing additional support during my graduation project. You were always available when I encountered a problem that I could not solve on my own and the fruitful discussions helped me a great deal with solving the problems I faced.

I would also like to thank some of my fellow students Elise Verhees, Jelle Bevers, and Wouter Weekers for our biweekly Teams meetings during the first half of the project phase. These meetings were a welcome change of pace from the long periods of working from home due to the unusual times caused by the COVID-19 pandemic. Furthermore, these meetings also allowed for the exchange of ideas and to discuss the problems we faced throughout our projects. From these meetings I also learned that whatever doubts I had about my work were also experienced by others and that these feelings were only natural, which in turn helped me to deal with them.

Furthermore, I would like to thank the friends I made during my studies, both from university and middle school, for always being a source of companionship and support. Over the course of my studies and especially during the COVID-19 pandemic, your companionship made sure that I was able to continue my work without suffering from burnout. Your support has also helped me through some rough periods and I don't know how I would have dealt with that without your help.

Finally and most importantly, I would like to thank my parents, Tracey Lear and Johan Proper, for encouraging and supporting me throughout my life. Their support helped motivate me to always try my best regardless of how difficult the road ahead might be. Their support also allowed me to fully focus on my studies, yet they were also quick to remind me that I should be cautious and to avoid losing myself in my studies and that I shouldn't feel guilty for occasionally taking some time off for myself to avoid overworking.

Nomenclature

Greek symbols

α	End-effector impact approach angle with respect to the expected horizontal environment
$\boldsymbol{\alpha}$	State feedback reference trajectory
β	The rotation of environmental frame B around $\vec{z}_A$
β_i	Region of acceptable scenarios where the post-impact behavior should be classified as expected
γ	Task reference trajectory for task-based QP controller
Δt	Sample interval
δx	Target position at final system operation time t_f relative to the desired impact position and orientation ${}^A\mathbf{p}_{imp}$ on the $\vec{x}_A$ axis
δy	Target position at final system operation time t_f relative to the desired impact position and orientation ${}^A\mathbf{p}_{imp}$ on the $\vec{y}_A$ axis
δ_i	Contact point indentation at the i-th contact point
ϵ	Joint displacement error threshold
ζ	Maximum allowable time difference between detected and desired impact time
η	Hertz contact exponent
$\boldsymbol{\theta}$	$n \times 1$ vector with motor positions
κ_{ref}	$n_c \times 1$ vector of desired normal contact forces for each contact point
$\Lambda_{N,i}$	Normal impulsive contact force at the i-th contact point
$\lambda_{N,i}$	Normal contact force at the i-th contact point
$\boldsymbol{\mu}$	$m \times 1$ reference feed forward vector
μ_N	Newton's coefficient of restitution in normal direction
μ_{fr}	Friction coefficient
ν	Poisson's ratio of a material
ρ_i^-	Short-hand notation of $\frac{d_c}{k_c}\dot{\delta}_i^-$
$\boldsymbol{\tau}$	$n \times 1$ vector of applied joint torques
$\boldsymbol{\tau}_J$	$n \times 1$ vector spring joint torques of a flexible joint robotic manipulator
$\boldsymbol{\tau}_{act}$	$n \times 1$ vector of actuation torques resulting from the low-level torque controller
$\boldsymbol{\tau}_{co}$	$n \times 1$ vector of external joint torques as a consequence of contact with the environment
$\boldsymbol{\tau}_{ext}$	$n \times 1$ vector of externally applied torques
$\boldsymbol{\tau}_{fr}$	$n \times 1$ vector of external joint torques as a consequence of joint friction
Φ	Simplified notation of the rigid impact map
ϕ_{imp}	Desired end-effector orientation at time of impact

Latin symbols

$\mathbf{A}_{eq}$	$s \times v$ matrix of equality constraint components that are a function of the optimization variables
$\mathbf{A}_{in}$	$r \times v$ matrix of inequality constraint components that are a function of the optimization variables
a	Signal envelope decay rate
A, B, C ...	Coordinate frames
awl	Adaptive window length
$\mathbf{B}$	$n \times n$ reflected motor inertia matrix after the gear reduction
$\mathbf{B}_\theta$	$n \times n$ positive definite diagonal feedback gain matrix
$\mathbf{b}_{eq}$	$s \times 1$ vector of equality constraint components that are not a function of the optimization variables
$\mathbf{b}_{in}$	$r \times 1$ vector of inequality constraint components that are not a function of the optimization variables
$\mathcal{C}$	Weighted task cost function
$\mathbf{C}$	$n \times n$ matrix containing Coriolis and centripetal terms
$\mathbf{D}$	$n \times n$ diagonal joint damping matrix
$\mathbf{D}_s$	$n \times n$ positive diagonal feedback gain matrix
d_c	Contact point damping
$\mathbf{e}_{imp,j}$	$n \times 1$ vector of the joint displacement error of the j-th detected impact
$\mathbf{e}_{imp}$	$n \times 1$ vector of the joint displacement error
$\mathbf{e}_{vel}$	$n \times 1$ vector of the joint velocity error of the j-th detected impact
E	Young's modulus of a material
e_η	Signal envelope margin
${}^{L_i[B]}\mathbf{f}$	6×1 vector containing the contact wrench with respect to frame $L_i[B]$
FK	Forward kinematics function relating the joint displacements $\mathbf{q}$ to the position and rotation of frame L with respect to frame A
$\mathbf{g}$	$n \times 1$ gravity torque vector
$\mathbf{H}$	$v \times v$ cost function symmetric (semi)definite Hessian matrix
$\mathbf{I}_{a \times a}$	$a \times a$ identity matrix
IK	Inverse kinematics function relating the position and rotation of frame L with respect to frame A to the joint displacements q
${}^{L[A]}\mathbf{J}_{A,L}$	$6 \times n$ Jacobian that relates the joint velocities to the twist of the i-th contact point frame L with respect to environment frame A, expressed in mixed frame $L[A]$
${}^{L_i[B]}\mathbf{J}_{B,L_i}$	$6 \times n$ Jacobian that relates the joint velocities to the twist of the i-th contact point frame L_i with respect to environment frame B, expressed in mixed frame $L_i[B]$
$\mathbf{J}_i$	Short-hand notation for the Jacobian ${}^{L_i[B]}\mathbf{J}_{B,L_i}$
$\mathbf{J}_{i,N}$	$1 \times n$ vector consisting of the normal component of $\mathbf{J}_i$
j_f	Final detected impact during $t \in [t_0, t_f]$

λ_{ref}	Desired post-impact continuous contact force
$\mathcal{K}$	Effective stiffness in the (exponentially extended) Hunt-Crossley contact model
$\mathbf{K}$	$n \times n$ diagonal joint stiffness matrix
$\mathbf{K}_d$	$n \times n$ derivative gain matrix for QP controller
$\mathbf{K}_g$	$m \times m$ feedback gain matrix for state feedback control law
$\mathbf{K}_O$	$n \times n$ momentum observer gain matrix
$\mathbf{K}_p$	3×3 proportional gain matrix for QP controller
$\mathbf{K}_S$	$n \times n$ derivative torque feedback gain matrix for the low-level torque controller
$\mathbf{K}_T$	$n \times n$ proportional torque feedback gain matrix for the low-level torque controller
k_c	Contact point stiffness
l	Displacement of frame B along the $\vec{y}_A$ axis with respect to inertial frame A
$\mathbf{M}$	$n \times n$ joint space inertia matrix for a flexible joint robotic manipulator
$\mathbf{M}_r$	$n \times n$ equivalent joint space inertia matrix for a rigid joint robotic manipulator ($\mathbf{M}+\mathbf{B}_\theta$)
m	Signal envelope filter window length
$\mathcal{N}_\mathcal{T}$	Number of tasks
n	Degrees of freedom for the robotic manipulator
n_c	Number of edge end-effector contact points
n_{imp}	Expected maximum number of impacts in an impact sequence
$\boldsymbol{p}$	$n \times 1$ generalized momentum vector
${}^A\mathbf{p}_{a,0}$	6×1 vector of the initial ante-impact trajectory position and orientation expressed in frame A
${}^A\mathbf{p}_{a,f}$	6×1 vector of the final ante-impact trajectory position and orientation expressed in frame A
${}^A\mathbf{p}_{imp}$	6×1 vector with the desired end-effector impact location and orientation expressed in frame A
${}^A\mathbf{p}_{p,0}$	6×1 vector of the initial post-impact trajectory position and orientation expressed in frame A
${}^A\mathbf{p}_{p,f}$	6×1 vector of the final post-impact trajectory position and orientation expressed in frame A
$\mathbf{Q}$	$v \times 1$ cost function gradient vector
$\mathbf{q}$	$n \times 1$ vector of generalized coordinates/joint displacements
$\mathbf{q}_0$	$n \times 1$ vector of initial generalized coordinates/joint displacements
$\mathbf{q}_{imp,j}$	$n \times 1$ vector of the measured joint displacements at the j-th detected time of impact
$\mathbf{q}_{imp}$	$n \times 1$ vector of the measured joint displacements at the detected time of impact
$\mathbf{r}$	$n \times 1$ vector of estimated external joint torques residuals from the momentum observer
r	Radius of the spherical end-effector
r_b	The external joint torque detection bound
r_c	Distance from the edge end-effector frames to the edge

$\mathcal{T}_i$	Weighted task error of the i-th task
t_0	Starting time of system operation
t_f	Final time of system operation
t_k	Time at timestep k
t_N	Time when contact with environment is completed
t_ξ	Time difference between the detected impact time and final classification time
t_{delay}	Impact detection delay
t_{end}	Final classification time
t_{int}	Maximum allowable time difference between impacts
$t_{j,d}$	j-th desired impact time
t_j	j-th detected impact time
t_{switch}	Switching time after the first detected impact to switch to the post-impact phase for a simultaneous impact case
$\mathbf{u}$	$m \times 1$ input to a continuously evolving system
$\left({}^{L_i[B]}\mathbf{v}_{B,L_i}\right)_y$	velocity of contact frame L_i with respect to frame B expressed in mixed frame $L_i[B]$ in the $\vec{y}_B$ direction
${}^{L[A]}\mathbf{v}_{A,L}^{imp}$	6×1 desired end-effector twist of control frame L with respect to frame A expressed in mixed frame $L[A]$ at the desired time of impact
v_{abs}	Desired absolute impact velocity of the end-effector
$\mathbf{W}_i$	$v \times v$ symmetric weighting matrix
w	Edge end-effector width
$\mathcal{X}$	$v \times 1$ vector containing the optimization variables
$X_{\dot{\mathbf{q}}}$	$n \times 1$ vector of approximated additive white Gaussian noise for $\dot{\mathbf{q}}$
$X_{\dot{\boldsymbol{\theta}}}$	$n \times 1$ vector of approximated additive white Gaussian noise for $\dot{\boldsymbol{\theta}}$
$X_{\mathbf{q}}$	$n \times 1$ vector of approximated additive white Gaussian noise for $\mathbf{q}$
$X_{\boldsymbol{\theta}}$	$n \times 1$ vector of approximated additive white Gaussian noise for $\boldsymbol{\theta}$
x_{imp}	Desired impact location on the $\vec{x}_A$ axis
h	Gap function for the spherical end-effector contact point
h_i	Gap function for the i-th edge end-effector contact point
y_{imp}	Desired impact location on the $\vec{y}_A$ axis

Sub- and superscripts

$(\cdot)^+$	Post-impact quantity at the time of impact
$(\cdot)^-$	Ante-impact quantity at the time of impact
$(\cdot)_a$	Parameter associated with ante-impact trajectory
$(\cdot)_k$	Quantity at timestep k
$(\cdot)_p$	Parameter associated with post-impact trajectory

$(\cdot)_{lb}$	Lower bound
$(\cdot)_{max}$	Minimum value of a given quantity
$(\cdot)_{min}$	Minimum value of a given quantity
$(\cdot)_{noise}$	Quantity with noise added to the original value
$(\cdot)_{ref}$	Desired/Reference parameter
$(\cdot)_{ub}$	Upper bound
$\overline{(\cdot)}$	Extended trajectory using reference spreading

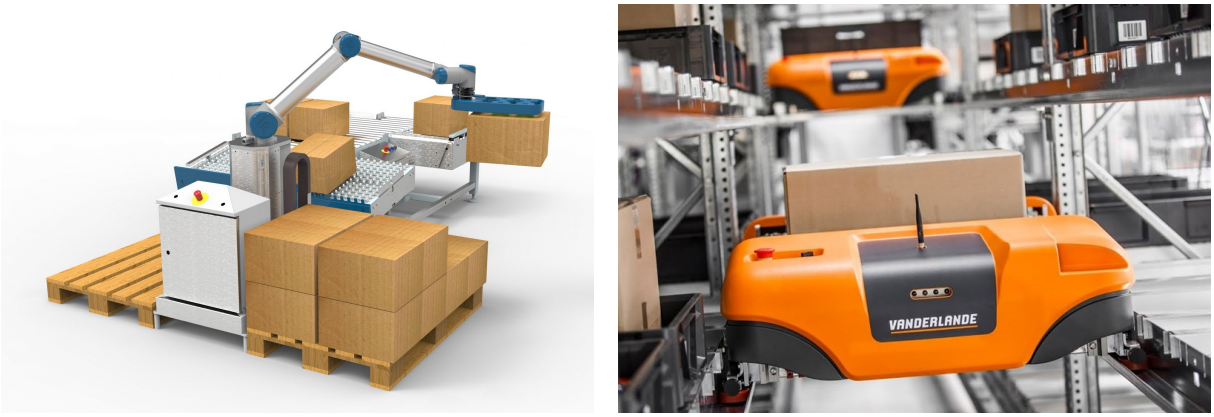
Contents

1 Introduction

Given the growing presence of e-commerce in day-to-day life [1], the number of parcels that must be sorted and delivered continues to increase. This growth is further accelerated due to the global 2020-2021 COVID-19 pandemic [2], which lead to further digitization of businesses across Europe. Given this growth in e-commerce, there is an increasing demand for labor in the logistics sector. However, due to the scarcity of manual labor, the logistics sector is not able to keep up with the demand. To solve this, repetitive tasks are being automated, which allows for human labor to be applied to more varied tasks, which increases overall throughput [3]. Examples of such automation include the Smart Palletizer by Smart Robotics and Vanderlande's Fastpick as seen in Figure 1.1.

Currently, robotic systems are not as dexterous as humans when handling packages. Humans can efficiently pick up and place items while being aware of the position and motion of the package and themselves in relation to their surroundings and can use impacts, which are defined as making contact with a surface at a nonzero velocity, to their advantage when placing a package. Currently, to minimize the risk of damaging the robot and its environment, robotic systems ensure that contact with the environment is established at near-zero velocities and can therefore not use impacts to their advantage [4]. Introducing the capability to exploit impacts with the environment into robotic systems can prove beneficial in a multitude of ways. For example, placing objects on a surface at a nonzero velocity or tossing an object to its destination would reduce the time needed for a pick-and-place operation, which would increase throughput. It is also possible to use impacts to decrease the air gap between packed objects, therefore increasing packing density.

To improve the capabilities of robotic systems to handle parcels quickly and safely while exploiting impacts, a method is required to predict and classify robot-environment and robot-object interactions. The robotic system will need to react appropriately to the impact to ensure the safety of the operators and to ensure that the parcel and its contents remain intact while finishing a pick-and-place operation as quickly as possible. Due to the complex dynamics of an object impacting a surface, where several impacts could occur over a small time frame before the impact has been concluded, current robotic systems cannot accurately predict or classify all robot-environment and robot-object interactions. However, it has been shown that a nonsmooth impact map can accurately predict the rigid component of the post-impact velocity for single impacts [5, 6]. This leaves the classification of robot-environment and robot-object interactions as an important continuation to improving the capabilities of robotic systems.



(a) (b)

Figure 1.1: Examples of automation in the logistics sector. In (a), the Smart Palletizer by Smart Robotics[2]. In (b), the Fastpick by Vanderlande[3].

[1]See https://i-am-project.eu/ for more information
[2]Image taken from: https://www.smart-robotics.nl/palletizing/
[3]Image taken from: https://www.vanderlande.com/warehousing/segments/general-merchandise/

The ability to predict, exploit, and classify intentional impacts in robotic systems are all part of *impact-aware manipulation*. Furthermore, the EU-funded I.AM. project[1] aims at developing impact-aware manipulation for robotic systems. The overall goal of this project is to make a contribution to the classification of robot-environment and robot-object interactions for impact-aware manipulation. To achieve this goal, it is important to understand all phases of an impact sequence, for which the collision event pipeline can be utilized. The collision event pipeline is a method to systematize the contact handling problem by dividing an impact sequence into distinct phases [7]. The collision event pipeline and its phases are investigated further in Section 1.1, which also investigates the state of the art for each of the phases, as well as what is missing for impact-aware manipulation.

1.1 Literature Review: The Collision Event Pipeline

A core problem of robotic manipulation is ensuring that possible human injury is minimized during physical human-robot interaction. To achieve this, unintended impacts should be avoided. However, since relative motions between humans and robots can be fast and unpredictable, traditional external sensors do not suffice. For this reason, to systematize the contact handling problem, a unified framework entitled the *collision event pipeline* is introduced by [7]. The collision event pipeline was designed with human-robot collision avoidance in mind and not with robot-environment collisions and impact-aware manipulation. However, the phases are defined generally enough such that it allows for using it as a basis for the context of impact-aware manipulation and to investigate where the pipeline is lacking. The collision event pipeline divides an impact event into seven distinct phases that clarify the desired information from the impact event to let the robot react appropriately. These seven phases are the pre- and post-collision, detection, isolation, identification, classification, and reaction phases. These phases and their ordering in the collision event pipeline are graphically depicted in Figure 1.2.

Each of the phases shown in Figure 1.2 can be separated into two categories. The first is monitoring signals, which are used to obtain information about an impact event. Furthermore, some phases are part of the context, which are dependent on internal and external factors such as the environment state or the current control objective. Furthermore, detection, isolation, and identification are often grouped when considering methods for implementing these phases. These methods are also called Fault Detection and Isolation (FDI) methods and are used to detect, isolate, and identify a fault in the response, which can take the form of an impact [8]. Classification methods are applied separately, where their implementation depends on the purpose of the robotic system. Finally, the reaction phase uses the information gathered in previous steps to adjust the control strategy to improve the performance of an operation. Overall, a wide variety of methods exist to obtain the information desired in each phase, some of these methods are discussed in the following sections.

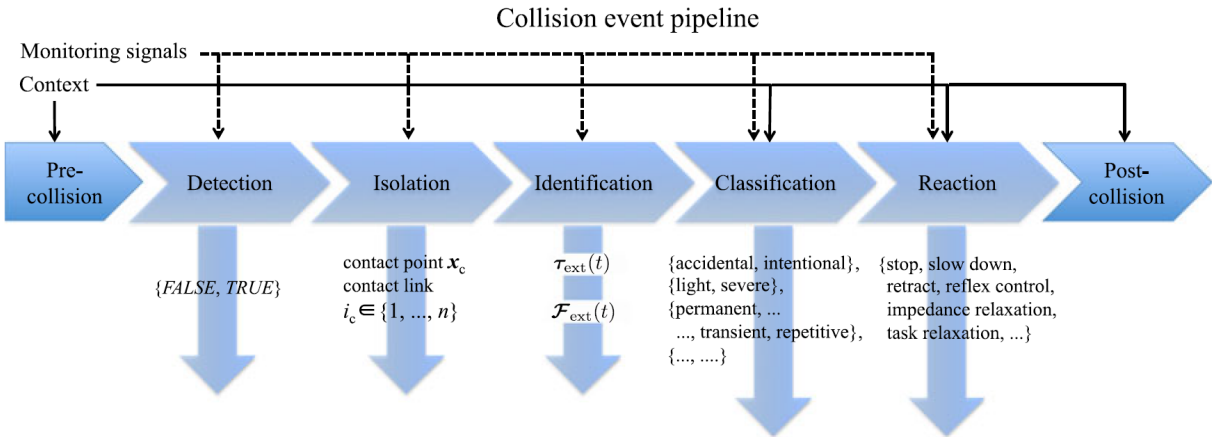


Figure 1.2: The Collision event pipeline [7]

Impact Detection

The impact detection phase is a phase that returns a binary output that states whether an impact has occurred or not. This phase is essential for all following phases as this indicates to the system that it is no longer in the pre-collision phase as shown in Figure 1.1. When considering a robotic system, an impact is characterized by the transmission of contact wrenches from the contact point to the robot. Furthermore, this contact wrench also results in a rapid change in the velocity and acceleration of the affected components at the time the contact wrench is applied. Impact detection algorithms aim to use information obtained from measurements to detect behavior that is indicative of impacts to conclude whether an impact has occurred or not.

Impact detection algorithms are based on the monitoring of signals obtained from the robotic system and uses certain metrics to determine whether an impact has taken place. Examples of such metrics are the measured currents in electrical drives, where fast transients can be analyzed as they could be indicative of a collision [9, 10]. Other impact detection methods use other metrics such as the commanded motor torques together with the expected motor torques, where any significant difference is attributed to unexpected external joint torques such as contact torques [11]. Finally, some impact detection methods use tactile sensors to determine whether an impact has occurred [12, 13]. However, these require additional sensors that not all systems have access to.

Some methods of impact detection rely on previous data points to predict the position of the robotic system after an action. For example, jump aware filtering employs a method where a prediction of the future position is made using the estimated velocity. If the measured position deviates from the predicted position past a predefined uncertainty bound, then the difference can be attributed to an impact [14]. Another common method of impact detection is through the use of estimations of the external joint torque. The external joint torque is estimated using sensor data and if the estimated external joint torque surpasses a predefined bound, it can be said that an impact has occurred [15, 16, 17, 18]. Finally, the external joint torque estimation methods that use generalized momenta [15, 16, 17] are usually used in combination with an external torque threshold to determine when an impact has taken place. However, it has also been shown to work with a neural network determining whether an impact has taken place [19].

While several impact detection methods are available, each with its benefits and drawbacks, the major practical problem of the implementation of an impact detection method is selecting the threshold of the monitored signal. If the threshold is too low, many false positives will occur. However, if the threshold is too high, the sensitivity of the impact detection method is lost. A balance must be struck to ensure that there are as few false positives as possible while maintaining a high sensitivity.

Impact Isolation

Impact isolation pertains to where on the robotic system an impact has occurred. It is desired that the collision isolation algorithm determines the precise contact point where the impact occurred. However, as this is not always feasible to achieve, isolating the link that has undergone an impact is commonly performed instead.

When considering methods for isolating the impact link of the robotic system, a detection method can be used that monitors the effects of an impact on individual links. Furthermore, the detection method for a link should be completely decoupled from the behavior of other links, i.e., dynamic effects that originate from impacts should not be spread to links that are not directly affected by the impact. For this reason, methods that require the inversion of mass matrices are not able to achieve reliable impact isolation [7]. If the detection method satisfies both of these demands, the impacting link can be isolated because only the links between the base and the impacting link show an effect of the impact.

Methods such as the monitoring of measured currents in robot electrical drives [9, 10], predicting joint displacements [14], and the estimation of external joint torques without mass matrix inversion [15, 16, 17, 18] satisfy the previously stated demands. The effect of the impact can vary in magnitude depending on the impact velocity, if the impact isolation method is not tuned sufficiently well enough, it is not guaranteed to isolate the impact link correctly for smaller impact velocities. However, assuming that the

impact isolation method has been tuned correctly, then the impact link can be isolated reliably.

Impact Identification

Impact identification pertains to the estimation of the directional information and intensity of the Cartesian contact wrench or the external joint torque. This information characterized the impact event [7], and all current methods for impact identification also provide a solution to the impact detection and isolation phases. Such methods, as described previously, are commonly referred to as Fault Detection and Isolation (FDI) methods.

Currently, the only methods that allow for impact identification are external joint torque estimators. These methods come in two forms, one that relies on generalized momenta [15, 16, 17] and another that relies on Inertial Measurement Units (IMUs) to provide additional information for direct estimation [18]. Furthermore, as explained previously, these solutions also provide results for the impact detection and isolation phase, meaning that these are also FDI methods.

Impact Classification

The impact classification phase is a broadly defined phase that uses the information obtained from sensors and previous phases to determine the properties of an impact. Impact classification can pertain to several properties of the impact, examples of which are whether it was accidental or intentional, whether it was an impact with a hard or soft object, and whether there is only a single impact or if there is continuous contact [20].

There are two main approaches to classifying impacts. The first approach is by using a threshold-based classifier, where certain properties can be classified based on a quantity surpassing a user-defined bound [21, 22]. These types of classifiers are not commonly used due to the rigidity of their classification method, i.e., a threshold solution typically only allows for the classification of a single property, and varying circumstances can affect the effectiveness of the classifier greatly. However, the main benefit of the threshold-based classifier is that it is simpler than the alternatives, resulting in improved ease of implementation for a variety of systems. Furthermore, threshold-based classifiers are also able to be tuned based on the requirements and have the capability to be adjusted if the accuracy of the classifier needs to be changed.

The second, more commonly used approach, is the neural network-based classifier [21, 23]. These classifiers use a neural network with a learning algorithm to determine the properties of an impact. Different learning algorithms include the nonlinear SVM supervised learning algorithms [24, 25, 26], deep learning Convolution Neural Networks (CNN) [27, 28], and Recurrent Neural Networks (RNN) [23]. Furthermore, besides training data, these classifiers can function without a priori information, resulting in a more versatile classifier. However, the drawback of these classifiers is that it requires a significant amount of training data, which is not always available.

Impact Reaction

The impact reaction phase is designated as the phase where the information from previous phases is used to determine a course of action in the post-collision phase. Reaction strategies can vary depending on the purpose of the robotic system. For example, if it is desired that no impact occurs, then a suitable reaction would be stopping the motion of the robotic system. Usually, the desired behavior of a robotic system that can undergo an impact includes the safety of people that are present near the system and the performance of the task the system should perform. From these desired behaviors, several reaction strategies can be devised.

One such strategy is a variable gain controller that lowers the gain based on the degree of deviation from the expected path, which results in a decreased chance of human injury [29]. Other reaction strategies seek to exploit the redundancy of a robotic manipulator to enhance the safety of the system while allowing the continuation of the execution of a Cartesian trajectory [30]. Another approach is the development of a set of standard collision reactions depending on different combinations of identification and classification results [31].

Conclusion

From the state of the art for each of the collision event pipeline phases, it is clear that there are a variety of solutions available for each. However, it is also clear that for applying the collision event pipeline to impact-aware manipulation, that there is an information gap. Currently, the pre-collision phase focuses on collision avoidance and anticipatory robot motion to minimize impact effects [7]. However, for impact-aware manipulation, impacts and the impact effects are desired to reduce the cycle time of an operation. Therefore, the pre-collision phase needs information about desired impacts and a prediction of the impact effects before it can be applied to impact-aware manipulation cases.

Furthermore, when considering the classification phase, most practical implementations of classification methods employ neural networks to obtain their results. However, these classifiers are trained with a set of training data, meaning that they do not use a priori information about a particular impact scenario. Differentiating between two impact scenarios, one desired and the other undesired, without a priori information, can prove to be difficult, as there is no baseline of what to compare the impact to.

Finally, the classifiers have all focused on single impact cases, where only one point makes contact with a surface. Considering that one of the purposes for which impact-aware manipulation is being developed is to handle packages, the classification of multiple impacts in a small time interval will need to be considered. The three points that were discussed in this section are further expanded upon in Section 1.2, where the focus and goal for the project will be laid out.

1.2 Research Goal

As was discussed in the conclusion of Section 1.1, three points were identified that were missing for the application of impact-aware manipulation in the logistics sector. These points are, the absence of a priori information in the collision event pipeline, most classification methods not being able to differentiate between a desired and undesired scenarios due to the absence of a priori information, and that classifiers mainly focus on single impact scenarios. Each of these points will be focused on in this project and the precise details will be expanded on below.

First of all, the pre-collision phase is missing information about desired impacts and the impact effects. However, as was discussed in the introduction to this chapter, impact dynamics are complex and it is not computationally feasible to obtain a realistic prediction quickly. However, as was shown in [5, 6] a simplified and computationally feasible prediction is available. Therefore, including and using such a computationally feasible method for impact classification will be a focus in this project.

Secondly, neural network-based classifiers do not use a priori information to classify impact behavior. As was discussed before, this project aims to use a priori information about the impact and the impact effects to classify its behavior, which is not possible for current neural network-based classifiers. Therefore, the focus will be placed on threshold-based classifiers instead. Furthermore, an additional benefit to using a threshold-based classifier is that they can be tuned based on the requirements and do not require training data, meaning that they can more easily be implemented on systems that do not have access to such data.

Furthermore, all discussed classifiers focused on single impact scenarios. However, when considering the application of impact-aware manipulation in the logistics sector, multiple impacts can occur over a small time interval when handling packages. Depending on the intent of the robotic manipulator, these impacts can be desired to occur simultaneously. As the name suggests, a simultaneous impact is an impact where at least two points make contact with the environment simultaneously. However, due to perturbations, these impacts are unlikely to be truly simultaneous. The unpredictable behavior due to the large number of possible impact responses that occur due to the loss of simultaneity complicates the classification of impact behavior and the methods that were discussed are not able to classify (nearly) simultaneous impacts for this reason.

However, the classification of simultaneous impacts is a broad subject. For example, the classification can relate to the intent of an impact, if it impacts a hard or soft surface, if there was only a single

impact or if continuous contact is maintained, or if it made contact with a static or dynamic object or a person. This project will focus on classifying the expectancy of the impact and post-impact behavior when impacting a flat surface, i.e., it will classify whether the impact and post-impact behavior were expected or unexpected based on the available and computationally feasible prediction made before operating the robotic manipulator. Furthermore, the classification will focus on geometric differences of the impact surface between an expected and unexpected scenario, i.e., differences between the position and slope of the impact surface are the leading cause of unexpected behavior.

Finally, the classification method will need to be tested to ensure that the proposed method functions as desired. When testing a new classification method, it is desired that a large set of scenarios is tested to gauge the performance of the method. This is harder to achieve in a reasonable time frame when testing on a real system, therefore, a simulation is used instead to test the performance of the classifier. With the focus of the project defined, the research goal can be defined as

Propose, design, and test a threshold-based classification method that employs an available and computationally feasible prediction of the impact behavior to classify the expectancy of a simultaneous impact scenario, focusing on the geometric differences of the impact surface as the leading cause of unexpected behavior.

To quantify the capabilities of the classifier, a set of requirements, preferences, and constraints are defined. These are defined as follows:

- Requirements
  - If the difference between the expected and measured impact location surpasses a user-defined threshold, then the impact is classified as an unexpected impact.
  - If the difference between the expected and measured post-impact behavior, such as velocity, surpasses a user-defined threshold, then it is classified as unexpected post-impact behavior.
  - The classifier can be tuned based on requirements set by the user.
- Preferences
  - The unexpected post-impact classification should occur as soon as possible. Preferably, unexpected post-impact behavior should be classified as soon as post-impact behavior can be measured or estimated.
  - The classification procedure should be implemented in real-time, and should therefore avoid computationally intensive methods that prevent the real-time implementation on real hardware.
  - The classifier should be presented in a manner that would allow it to be implemented into a real robotic manipulator straightforwardly.
- Constraints
  - The classifier only uses information typically available to a robotic manipulator, such as position measurements and velocity estimations. If information is not available, a method should be used to obtain this information that does not require changing the hardware of the robotic manipulator.
  - The classifier is not tailored to a specific robotic manipulator.
  - The classifier is designed to work for two- and three-dimensional simultaneous impacts.

1.3 Report Structure

Besides this introduction, this report is structured as follows. In Chapter 2, the preliminary information and simulation models will be discussed. Here, the notation and simultaneous impacts are discussed, as well as a detailed explanation of reference spreading and the momentum observer. Furthermore, the model types found in the simulation provided by [6] are explained.

Chapter 3 will focus on defining the expected and unexpected scenario, the methodology used to determine what is expected of the impact behavior, the reference that should be followed, and the control strategy

for simultaneous impacts. It will also discuss a method to detect multiple impacts using an existing impact detection method, and it will provide simulated behavior of the different model types, showing that they are consistent with each other.

Chapter 4 focuses on the development of the classifier. Initially, it will discuss the characteristic behavior of expected impacts, which is then followed by the development of a filter to obtain the necessary information from the response. This is then implemented into a single impact classifier, which is then expanded for simultaneous impacts. The performance of the classifiers will also be presented in this chapter.

Finally, in Chapter 5, the conclusion of the project will be given. This is then followed by a recommendation for future research and improvements to the used systems.

2 Preliminaries and Simulation Models

This chapter provides information on required preliminary theoretical knowledge and the used simulation models to understand the contribution made in Chapters 3 and 4. This includes the multibody dynamics notation that is used throughout the project, discussed in Section 2.1. Afterwards, the simulation models designed in [6] are described in Section 2.2, this discusses simulation frames, manipulator dynamics, impact dynamics, and quadratic programming (QP) robot control. A detailed explanation regarding what constitutes a simultaneous impact is given in Section 2.3. Section 2.4 discusses different implementations of reference spreading and Section 2.5 discusses the momentum observer.

2.1 Multibody Dynamics Notation

The multibody dynamics notation as proposed in [32] will be reviewed in this section and this notation will be used throughout the project. First of all, coordinate frames and points will be discussed. This is then followed by a method to formulate the velocity and acceleration vectors in these frames. Afterwards, a formulation for force covectors, otherwise known as wrenches, is presented. Finally, expressions are formulated to define geometric Jacobians in these frames.

2.1.1 Coordinate Frames and Points

A coordinate frame is defined as the combination of a point and orientation frame in 3D space and is typically denoted by a capital letter. Given an arbitrary frame W, the origin is defined as $\mathbf{o}_W$, and the orientation frame is defined as $[W]$. Frame W can then be formally written as $W = (\mathbf{o}_W, [W])$. Frames can move in time with respect to a given reference frame and are used to describe the position and orientation of a rigid body over time, as well as the wrench exchanged between two bodies and to describe a coordinate system used for a robot task.

For Newton's first law of motion to hold, the existence of an inertial frame is required. An inertial frame is a frame that is not undergoing an acceleration. For this project, an absolute inertial frame A is defined and is located at the base of the robotic manipulator. A mixed frame can also be defined, this is defined as a frame that has an origin that coincides with one frame, but has an orientation frame of another frame. For example, a mixed frame with the origin at $\mathbf{o}_W$ and the orientation frame $[A]$ can be formally written as $W[A] = (\mathbf{o}_W, [A])$.

For an arbitrary point $\mathbf{p}$, its coordinates with respect to frame A are collected in the coordinate vector ${}^A\mathbf{p}$. This coordinate vector represents the coordinates of the three-dimensional geometric vector $\vec{\boldsymbol{r}}_{\mathbf{o}_A,\mathbf{p}}$ connecting the origin of frame A with the point $\mathbf{p}$. This can be mathematically written as

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

where $\vec{x}_A$, $\vec{y}_A$, and $\vec{z}_A$ denote the unit vectors defining the orientation frame $[A]$. Given the frames A and W, a coordinate transformation from W to A can be performed using the homogeneous transformation matrix ${}^A\mathbf{H}_W \in \mathrm{SE}(3)$ given by

$$ {}^A\mathbf{H}_W := \begin{bmatrix} {}^A\mathbf{R}_W & {}^A\mathbf{o}_W \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{2.2} $$

where ${}^A\mathbf{R}_W \in \mathrm{SO}(3)$ is the rotation matrix from W to A and ${}^A\mathbf{o}_W$ is the position of origin $\mathbf{o}_W$ with respect to frame A. Given the point $\mathbf{p}$, the coordinate vector ${}^W\mathbf{p}$ can be transformed into ${}^A\mathbf{p}$ by

$$ {}^A\bar{\mathbf{p}} = {}^A\mathbf{H}_W {}^W\bar{\mathbf{p}} \tag{2.3} $$

where ${}^A\bar{\mathbf{p}}$ and ${}^W\bar{\mathbf{p}}$ are the homogeneous representations of the coordinate vectors ${}^A\mathbf{p}$ and ${}^W\mathbf{p}$, respectively. That is,

$$ {}^A\bar{\mathbf{p}} := \begin{bmatrix} {}^A\mathbf{p} \\ 1 \end{bmatrix} \quad \text{and} \quad {}^W\bar{\mathbf{p}} := \begin{bmatrix} {}^W\mathbf{p} \\ 1 \end{bmatrix} \tag{2.4} $$

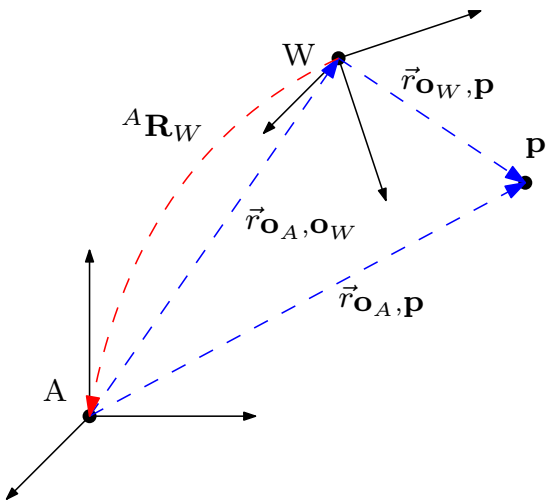


Figure 2.1: A visual representation of different frames, including their transformations. Showing the inertial frame A, an frame W which is placed and oriented arbitrarily in space, a point $\mathbf{p}$ that is placed arbitrarily in space, several vectors, and the rotation matrix ${}^{A}\mathbf{R}_{W}$. The parameters are defined according to the notation described in Section 2.1.1

These definitions are also shown in Figure 2.1.

2.1.2 Velocity and Acceleration Vectors

The velocity of coordinate frame W can be defined using the left trivialized velocity, the right trivialized velocity, and the mixed velocity. The left trivialized velocity is defined as the velocity of frame W with respect to frame A, expressed in frame W. This left trivialized velocity is given by

$$ {}^{W}\mathbf{v}_{A,W} := \begin{bmatrix} {}^{W}\boldsymbol{v}_{A,W} \\ {}^{W}\boldsymbol{\omega}_{A,W} \end{bmatrix} \in \mathbb{R}^6, \tag{2.5} $$

where

$$ {}^{W}\boldsymbol{v}_{A,W} := {}^{A}\mathbf{R}_{W}^{\top}\,{}^{A}\dot{\mathbf{o}}_{W}, \tag{2.6} $$

and

$$ {}^{W}\boldsymbol{\omega}_{A,W}^{\wedge} := {}^{A}\mathbf{R}_{W}^{\top}\,{}^{A}\dot{\mathbf{R}}_{W}. \tag{2.7} $$

In (2.5), (2.6), and (2.7), ${}^{W}\boldsymbol{v}_{A,W} \in \mathbb{R}^3$, ${}^{W}\boldsymbol{\omega}_{A,W} \in \mathbb{R}^3$, and the hat operator $^{\wedge}$ transforms a vector in $\mathbb{R}^3$ to a skew-symmetric matrix in $\mathfrak{so}(3)$. The acceleration vector associated with the left trivialized velocity is given by

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

The right trivialized velocity is defined as the velocity of frame W with respect to frame A, expressed in frame A. The right trivialized velocity is given by

$$ {}^{A}\mathbf{v}_{A,W} := \begin{bmatrix} {}^{A}\boldsymbol{v}_{A,W} \\ {}^{A}\boldsymbol{\omega}_{A,W} \end{bmatrix} \in \mathbb{R}^6, \tag{2.9} $$

where

$$ {}^{A}\boldsymbol{v}_{A,W} := {}^{A}\dot{\mathbf{o}}_{W} - {}^{A}\dot{\mathbf{R}}_{W}\,{}^{A}\mathbf{R}_{W}^{\top}\mathbf{o}_{W}, \tag{2.10} $$

and

$$ {}^{A}\boldsymbol{\omega}_{A,W}^{\wedge} := {}^{A}\dot{\mathbf{R}}_{W}\,{}^{A}\mathbf{R}_{W}^{\top}. \tag{2.11} $$

In (2.9), (2.10), and (2.11), ${}^{A}\boldsymbol{v}_{A,W} \in \mathbb{R}^3$ and ${}^{A}\boldsymbol{\omega}_{A,W} \in \mathbb{R}^3$. The acceleration vector associated with the right trivialized velocity is

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

The mixed velocity is defined as the velocity of frame W with respect to frame A, expressed in a mixed frame with the origin of frame W and the orientation of frame A, denoted as $W[A] = (\mathbf{o}_W, [A])$. This velocity is given by

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

The associated acceleration vector is given by

$$ {}^{W[A]}\dot{\mathbf{v}}_{A,W} := \begin{bmatrix} {}^{A}\ddot{\mathbf{o}}_W \\ {}^{A}\dot{\boldsymbol{\omega}}_{A,W} \end{bmatrix} \in \mathbb{R}^6, \tag{2.14} $$

These velocity vectors are also known as twists.

2.1.3 Force Covectors

The force covectors, also known as wrenches, $\mathbf{f}$ with respect to a given frame W are expressed as

$$ {}_W\mathbf{f} := \begin{bmatrix} {}_W\boldsymbol{f} \\ {}_W\boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^6, \tag{2.15} $$

where ${}_W\boldsymbol{f} \in \mathbb{R}^3$ is the force component and ${}_W\boldsymbol{\tau} \in \mathbb{R}^3$ is the torque component of the wrench. This wrench can be expressed in another frame A through

$$ {}_A\mathbf{f} = {}_A\mathbf{X}^W \, {}_W\mathbf{f} \tag{2.16} $$

where ${}_A\mathbf{X}^W \in \mathbb{R}^{6\times 6}$ is the wrench coordinate transformation. This wrench coordinate transformation is defined as

$$ {}_A\mathbf{X}^W := \begin{bmatrix} {}^{A}\mathbf{R}_W & \mathbf{0}_{3\times 3} \\ {}^{A}\mathbf{o}_W^{\wedge}\,{}^{A}\mathbf{R}_W & {}^{A}\mathbf{R}_W \end{bmatrix} \tag{2.17} $$

2.1.4 Jacobians

For multibody robotic systems such as robotic manipulators, Jacobians are essential tools in defining contact and constraint forces. They are also used to express velocity and force tasks in robot control and they serve a function in forward and inverse kinematics of velocity and acceleration. For a robotic manipulator, the coordinate frame $L = (\mathbf{o}_L, [L])$ represents the end-effector frame and can be expressed in the joint coordinates $\mathbf{q} \in \mathbb{R}^n$, where $n \in \mathbb{N}$ are the number of joints of the robotic manipulator. This can be expressed as

$$ \begin{bmatrix} {}^{A}\mathbf{o}_L \\ {}^{L}\mathbf{R}_A \end{bmatrix} = FK(\mathbf{q}), \tag{2.18} $$

where FK are the forward kinematics relating the joint displacements to the position and orientation of coordinate frame L. The velocity of frame L with respect to the inertial frame A can be expressed using various expressions for the velocity, the left trivialized velocity, the right trivialized velocity, and the mixed velocity. These velocities can be calculated using the Jacobian and the joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$. The left trivialized velocity can be calculated as

$$ {}^{L}\mathbf{v}_{A,L} = {}^{L}\mathbf{J}_{A,L}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.19} $$

where ${}^{L}\mathbf{J}_{A,L}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian relating the joint velocities to the twist of L with respect to A, expressed in L, as a function of the joint displacements $\mathbf{q}$. The associated acceleration is

$$ {}^{L}\dot{\mathbf{v}}_{A,L} = {}^{L}\mathbf{J}_{A,L}(\mathbf{q})\ddot{\mathbf{q}} + {}^{L}\dot{\mathbf{J}}_{A,L}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} \tag{2.20} $$

The right trivialized velocity is calculated as

$$ {}^{A}\mathbf{v}_{A,L} = {}^{A}\mathbf{J}_{A,L}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.21} $$

where ${}^{A}\mathbf{J}_{A,L}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian relating the velocity of L with respect to A, expressed in A, as a function of the joint displacements $\mathbf{q}$. The associated acceleration is

$$ {}^{A}\dot{\mathbf{v}}_{A,L} = {}^{A}\mathbf{J}_{A,L}(\mathbf{q})\ddot{\mathbf{q}} + {}^{A}\dot{\mathbf{J}}_{A,L}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} \tag{2.22} $$

The mixed velocity is calculated as

$$^{L[A]}\mathbf{v}_{A,L} = {}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.23}$$

where $^{L[A]}\mathbf{J}_{A,L}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian relating the velocity of L with respect to A, expressed in $L[A]$, as a function of the joint displacements $\mathbf{q}$. The associated acceleration is

$$^{L[A]}\dot{\mathbf{v}}_{A,L} = {}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\ddot{\mathbf{q}} + {}^{L[A]}\dot{\mathbf{J}}_{A,L}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} \tag{2.24}$$

2.2 Robotic manipulator simulation

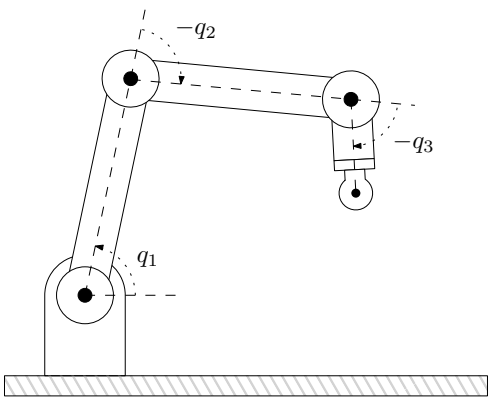
To test a classification method on a realistic response of a robotic manipulator impacting a surface, a set of numerical simulations are required. The original version of the simulation was designed by Wouter Weekers in [6] and is based on the system specifications of the Franka Emika Panda, which is a 7 degrees of freedom (DOF) robotic manipulator. However, to reduce the complexity of the model, rather than modeling the robotic manipulator in three-dimensional space, a two-dimensional approximation with 3-DOF is modeled instead.

The simulation is designed to simulate the behavior of the Franka Emika Panda in its planar configuration, an example of which together with a schematic representation of this configuration can be found in Figure 2.2. The simulation consists of three distinct model types, where the robotic manipulator and the impact surface are modeled with varying degrees of complexity. These models are

- Model A: A robotic manipulator with flexible joints impacting an environment with compliant contact dynamics.
- Model B: A robotic manipulator with flexible joints impacting a rigid environment.
- Model C: A robotic manipulator with rigid joints impacting a rigid environment.

First of all, in Section 2.2.1 the coordinate frames that are used will be discussed. In Section 2.2.2, the robotic manipulator models are discussed, starting with a rigid joint model followed by an expanded flexible joint model. In Section 2.2.3, the impact surface modeling is discussed for the two different types of surfaces. Finally, in Section 2.2.4, the task-based QP controller is discussed.

(a)



(b)

Figure 2.2: Comparison between the Franka Emika Panda and a schematic planar representation. In (a), the Franka Emika Panda in its planar configuration from a left side view[4]. In (b), the schematic representation of the Franka Emika Panda from a right side view

[4]Image courtesy of Elise Verhees

2.2.1 Coordinate Frames

There are two sets of frames that are relevant for the simulation of the robotic manipulator. The first set contains the environment frames, which are inertial frames where Newton's first law of motion is valid. The second set of frames are the end-effector frames, which can move in time with respect to the environment frames, and are dependent on the joint configuration of the robotic manipulator. The environment frames are defined in Section 2.2.1, and the end-effector frames are defined in Section 2.2.1.

Environment Frame

The first frame of reference that is defined is frame $A = (\mathbf{o}_A, [A])$. This frame is located at the base of the simulated robotic manipulator and is also known as the absolute inertial frame. This frame is used to present any results of the simulation unless specified otherwise.

The second frame of reference is the environment frame, which is defined as $B = (\mathbf{o}_B, [B])$. This frame is related to frame A by rotating it around $\vec{z}_A$, followed by translating it along $\vec{y}_A$ to change the position of $\vec{x}_B$, which coincides with the surface of the environment. Figure 2.3 shows the relation between these frames. In Figure 2.3, $\beta \in \mathbb{R}$ represents the angle that frame B is rotated with respect to A, and $l \in \mathbb{R}$ is the translation of frame B with respect to A along $\vec{y}_A$.

End-Effector Frames

The position and orientation of the end-effector frames relative to the inertial frame A are defined using the joint configuration and a predefined location on the end-effector. However, to efficiently use these frames for impact modeling, their relation with the environment needs to be defined. Two specific end-effectors are considered for this simulation, namely, a spherical and edge end-effector.

The spherical end-effector is an end-effector that consists of an aluminum sphere and is attached to the end of the robotic manipulator where a gripper would normally be connected. In a two-dimensional simulation, this end-effector can be represented as a circle with a frame defined at its center. As described in Section 2.1.4, the location and orientation of this frame is dependent the joint configuration of the robotic manipulator. This frame, formally defined as $L = (\mathbf{o}_L, [L])$, is then rotated around $\vec{z}_L$ such that its orientation is identical to the environmental frame B. This results in mixed frame $L[B] = (\mathbf{o}_L, [B])$. These frames are also shown in Figure 2.3, where $r \in \mathbb{R}$ is the radius of the end-effector. Using the spherical end-effector frame and by defining ${}^B\mathbf{y}_B$ as the coordinate vector notation of unit vector $\vec{y}_B$ expressed in frame B, the gap function of the spherical end-effector can then be defined as

$$h = {}^B\mathbf{y}_B^\top {}^B\mathbf{o}_L - r, \tag{2.25}$$

with

$${}^B\mathbf{y}_B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{2.26}$$

The edge end-effector is designed as an aluminum half cylinder with rounded ends in three-dimensional space and is attached to the end of the robotic manipulator similarly to the spherical end-effector. In a two-dimensional simulation, this end-effector can be approximated as a rectangle with rounded corners. Similar to the spherical end-effector described previously, the edge end-effector frames are defined based on the joint configuration of the robotic manipulator. However, an edge making contact with the environment has multiple possible contact points that cannot easily be defined with a single frame. Therefore, a set of $n_c \in \mathbb{N}$ frames are defined along the edge to represent the contact points.

These contact point frames are placed at regular intervals over a predefined end-effector width $w \in \mathbb{R}_{>0}$ and can be defined as $L_i = (\mathbf{o}_{L_i}, [L_i])$ for $i \in [2, n_c]$. Each of these frames is then oriented so that the orientation is identical to the environment frame B, resulting in the mixed frames $L_i[B] = (\mathbf{o}_{L_i}, [B])$ for $i \in [2, n_c]$. Furthermore, to control the end-effector, a control frame is defined. This frame is defined as L_c and is located in the middle of the edge end-effector. A graphical representation of these edge end-effector frames can be seen in Figure 2.5, where $r_c \in \mathbb{R}$ is the smallest distance from each frame to the edge. Similarly to the spherical end-effector case, the gap function for the i-th contact point of the edge end-effector can be defined as

$$h_i = {}^B\mathbf{y}_B^\top {}^B\mathbf{o}_{L_i} - r_c, \tag{2.27}$$

with

$$ {}^B\mathbf{y}_B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{2.28}$$

From the similarities between (2.25) and (2.27), it can be seen that the gap function for each of the edge end-effector contact points is formulated as if the end-effector consists of a set of n_c spheres. The main interest of this project is the classification of behavior when the impact at the contact points did not occur simultaneously, which is possible with this approach to modeling the edge end-effector. Furthermore, there is no risk of the end-effector undergoing an impact that would be impossible for a continuous edge end-effector because the impact occurs on a flat surface.

2.2.2 Robotic Manipulator Model

For the robotic manipulator, two different approaches to modeling can be used for the simulation, one with rigid and the other with flexible joints. A rigid joint robotic manipulator is a manipulator where the transmission between the actuators and the joints is considered rigid, resulting in no internal dynamics in the joints and the joints being controlled directly. A flexible joint robotic manipulator controls the joints through a flexible transmission, which introduces additional dynamics into the system. In the following sections, the models for the rigid and flexible joint robotic manipulators are explained.

Rigid Joint Robotic Manipulator

The dynamic model of a robotic manipulator with rigid joints is given as

$$\mathbf{M}_r(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext}, \tag{2.29}$$

where $\mathbf{q} \in \mathbb{R}^n$ are the joint displacements, $\mathbf{M}_r(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite inertia matrix for a rigid robot, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is a matrix used to factorize the Coriolis and centripetal terms using Christoffel symbols, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity vector, and $\boldsymbol{\tau} \in \mathbb{R}^n$ and $\boldsymbol{\tau}_{ext} \in \mathbb{R}^n$ are the applied joint torques and the externally applied joint torques as a consequence of joint friction and contact forces, respectively. Considering that $\boldsymbol{\tau}_{ext}$ expresses both torques as a consequence of joint friction and contact with the environment, it is equivalent to

$$\boldsymbol{\tau}_{ext} = \boldsymbol{\tau}_{fr} + \boldsymbol{\tau}_{co}, \tag{2.30}$$

where $\boldsymbol{\tau}_{fr} \in \mathbb{R}^n$ is the torque as a consequence of joint friction, and $\boldsymbol{\tau}_{co} \in \mathbb{R}^n$ is the torque as a consequence from contact forces. $\boldsymbol{\tau}_{co}$ can be written as

$$\boldsymbol{\tau}_{co} = \sum_{i=1}^{n_c} \mathbf{J}_i^\top(\mathbf{q})_{L_i[B]}\mathbf{f}, \tag{2.31}$$

where $n_c \in \mathbb{N}$ are the number of contact points, $\mathbf{J}_i(\mathbf{q}) = {}^{L_i[B]}\mathbf{J}_{B,L_i}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian relating the joint velocity to the twist of the i-th contact point frame L_i with respect to inertial frame B, expressed in the mixed frame $L_i[B]$, the notation was shortened in equations to ensure readability, ${}_{L_i[B]}\mathbf{f} \in \mathbb{R}^6$ is the contact wrench with respect to mixed frame $L_i[B]$. These frames are discussed in Sections 2.2.1 and 2.2.1.

Flexible Joint Robotic Manipulator

The dynamic model of a robotic manipulator with flexible joints takes an approach that is similar to what is presented in the rigid joint case. However, the joint displacements $\mathbf{q}$ are connected to the motor positions $\boldsymbol{\theta} \in \mathbb{R}^n$ through a flexible transmission. The motor positions and joint displacements are related through $\boldsymbol{\tau}_J = \mathbf{K}(\boldsymbol{\theta} - \mathbf{q})$, where $\boldsymbol{\tau}_J \in \mathbb{R}^n$ is the spring torque and $\mathbf{K} = diag(k_i) \in \mathbb{R}^{n\times n}$ is the diagonal joint stiffness matrix. The full dynamics of a robotic manipulator with flexible joints is given as [33]

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) &= \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_{ext}, \\ \mathbf{B}\ddot{\boldsymbol{\theta}} + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_J &= \boldsymbol{\tau}_{act}, \end{aligned} \tag{2.32}$$

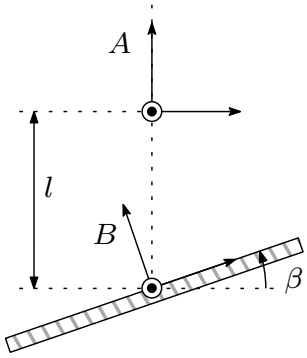


Figure 2.3: Visual representation of the relation between the absolute inertial frame A and environment frame B.

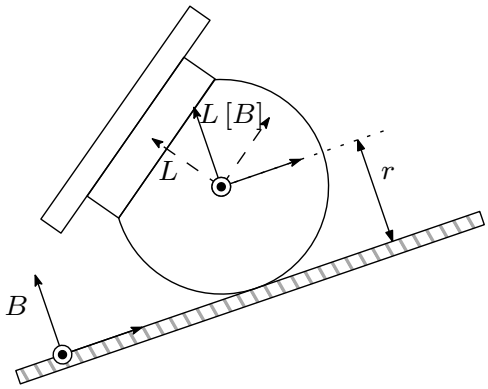
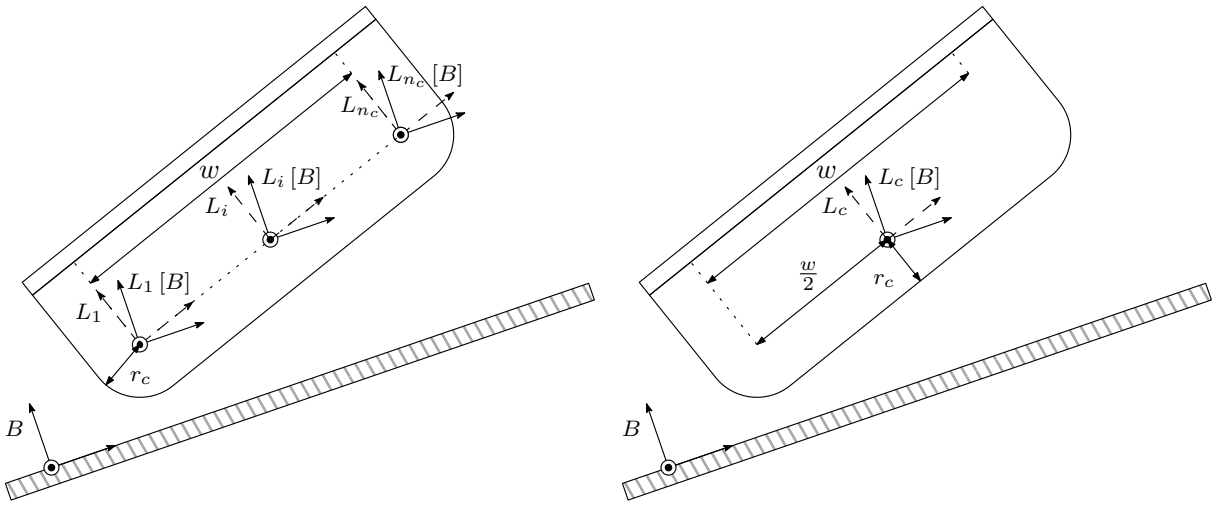


Figure 2.4: Visual representation of the relation between the environment frame B and the spherical end-effector frame L



(a) (b)

Figure 2.5: Visual representation of the relation between the environement frame B and the edge end-effector frames. In (a), the focus is placed on the location of the edge end-effector contact point frames L_i. In (b), the focus is placed on the edge end-effector control frame L_c.

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite inertia matrix for a rigid robot, $\mathbf{B} = diag(b_i) \in \mathbb{R}^{n\times n}$ is the reflected motor inertia matrix after the transmission, and $\mathbf{D} = diag(d_i) \in \mathbb{R}^{n\times n}$ is the diagonal joint damping matrix. $\boldsymbol{\tau}_{act} \in \mathbb{R}^n$ are the actuation torques obtained from a low-level torque controller [33] which is described as

$$\begin{aligned}\boldsymbol{\tau}_{act} &= \mathbf{B}\mathbf{B}_\theta^{-1}\boldsymbol{\tau} + \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J - \mathbf{B}\mathbf{B}_\theta^{-1}(\boldsymbol{\tau}_J + \mathbf{D}_s\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J)\\ &= \boldsymbol{\tau} + \mathbf{K}_T(\boldsymbol{\tau} - \boldsymbol{\tau}_J) - \mathbf{K}_S\dot{\boldsymbol{\tau}}_J,\end{aligned} \tag{2.33}$$

where $\mathbf{K}_T := \mathbf{B}\mathbf{B}_\theta^{-1} - \mathbf{I}_{n\times n}$ is a proportional torque feedback gain, $\mathbf{K}_S := \left(\mathbf{B}\mathbf{B}_\theta^{-1}\mathbf{D}_s - \mathbf{D}\right)\mathbf{K}^{-1}$ a derivative torque feedback gain, $\mathbf{B}_\theta = diag(b_{\theta,i}) \in \mathbb{R}^{n\times n}$ a positive definite diagonal matrix such that $b_{\theta,i} < b_i$, and $\mathbf{D}_s \in \mathbb{R}^{n\times n}$ is a positive diagonal feedback gain matrix.

As a consequence of this low-level torque controller. The inertia matrix for the rigid joint robotic manipulator $\mathbf{M}_r(\mathbf{q})$ and the inertia matrix for a robot with flexible joints $\mathbf{M}(\mathbf{q})$ are related to each other through the expression

$$\mathbf{M}_r(\mathbf{q}) = \mathbf{M}(\mathbf{q}) + \mathbf{B}_\theta. \tag{2.34}$$

2.2.3 Impact surface modeling

As was described in Section 2.2, for the impact simulations, two different approaches to modeling the impact surface are used. For model A, a compliant surface model is employed where the applied normal contact force $\lambda_{N,i}$ is a function of the contact point indentation δ_i and contact point indentation velocity $\dot{\delta}_i$. For models B and C, a rigid impact map is used to determine the post-impact velocity and the contact force is determined by using the Lagrange multipliers calculated using an acceleration constraint derived from the contact constraint. The compliant surface model and the derivation of the rigid impact map and unilateral constraints are described in the following sections.

Compliant Surface Model

To simulate a compliant surface, a modified Hunt-Crossley model is used. The original Hunt-Crossley model [34] calculates the normal contact force $\lambda_{N,i}$ for the i-th contact point as a function of the contact point indentation δ_i and indentation velocity $\dot{\delta}_i$ according to

$$\lambda_{N,i} = \begin{cases} 0, & \text{if } \delta_i < 0 \\ \mathcal{K}(\dot{\delta}_i)\delta_i^\eta & \text{if } \delta_i \geq 0 \end{cases}, \tag{2.35}$$

where $\mathcal{K}(\dot{\delta}_i) = k_c + d_c\dot{\delta}_i$ is the effective stiffness, with k_c the contact point stiffness, d_c the contact point damping, and η the Hertz contact exponent which is a geometry dependent contact parameter. However, (2.35) might result in negative contact forces upon forced separation of the objects in contact. To avoid this, the Hunt-Crossley model is exponentially extended [35]. The new effective stiffness proposed by the exponentially extended Hunt-Crossley model is given by

$$\mathcal{K}(\dot{\delta}_i) = \begin{cases} k_c + d_c\dot{\delta}_i, & \text{if } \dot{\delta}_i \geq -\mu_N\dot{\delta}_i^- \\ k_c(1+\rho_i^-\mu_N)\exp\left(\frac{\rho_i^-}{1-\rho_i^-\mu_N}\frac{\dot{\delta}_i+\mu_N\dot{\delta}_i^-}{\dot{\delta}_i^-}\right) & \text{if } \dot{\delta}_i < -\mu_N\dot{\delta}_i^- \end{cases}, \tag{2.36}$$

with

$$\rho_i^- = \frac{d_c}{k_c}\dot{\delta}_i^-, \tag{2.37}$$

where $\mu_N \in [0,1]$ is Newton's coefficient of restitution in the normal direction of the impact surface, i.e., in the $\vec{y}_B$ direction, with Newton's law of restitution defined as $\dot{\delta}_i^+ = -\mu_N\dot{\delta}_i^-$, and with $\dot{\delta}_i^+$ and $\dot{\delta}_i^-$ the post- and ante-impact indentation velocity in the $\vec{y}_B$ direction, respectively. Furthermore, inelastic impacts are assumed, i.e., $\mu_N = 0$, the effective stiffness $\mathcal{K}(\dot{\delta}_i)$ for inelastic impacts is given by

$$\mathcal{K}(\dot{\delta}_i) = \begin{cases} k_c + d_c\dot{\delta}_i, & \text{if } \dot{\delta}_i \geq 0 \\ k_c\exp\left(\rho_i^-\right) & \text{if } \dot{\delta}_i < 0 \end{cases}. \tag{2.38}$$

The contact point stiffness for a sphere can be calculated using the material properties of the impact surface and the end-effector. For a spherical end-effector with radius r impacting a plane, the equation is formulated as

$$k_c = \frac{4}{3(\sigma_i + \sigma_j)}\sqrt{r}, \tag{2.39}$$

where σ is a material parameter which is calculated using the poisson's ratio ν and elasticity modulus E for a material designated with i or j. In the case presented in material i is the end-effector material, and material j is the environment material. The general equation for σ is found to be

$$\sigma = \frac{1-\nu^2}{E}. \tag{2.40}$$

Furthermore, in [6], the contact point damping was chosen to be equal to $d_c = k_c \cdot 10^4$. To remain consistent with the findings presented in [6], the same calculation for the contact point damping is used in this project.

Rigid Impact Map and Unilateral Constraint Derivation

To simulate an impact with a rigid surface, nonsmooth mechanics [36] is used to model effects of the unilateral constraints applied to the end-effector. The unilateral position constraint that is applied to the system can be defined using the gap functions found in (2.25) and (2.27) as

$$h_i = {}^B\mathbf{y}_B^\top {}^B\mathbf{o}_{L_i} - r_c \geq 0, \tag{2.41}$$

with

$${}^B\mathbf{y}_B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{2.42}$$

When $h_i = 0$, an impact has occurred at the i-th contact point, and a state-jump takes place. This state jump can be calculated using a rigid impact map which can be derived using the manipulator dynamics and known properties of the impact surface. Furthermore, the rigid impact map will need to be derived for the rigid and flexible joint robotic manipulator separately. For the rigid joint robotic manipulator, substituting (2.31) into (2.30), which is then substituted into (2.29), the full expression for the rigid joint robotic manipulator dynamics can be obtained. This yields

$$\mathbf{M}_r(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{fr} + \sum_{i=1}^{n_c} \mathbf{J}_i^\top {}_{L_i[B]}\mathbf{f}. \tag{2.43}$$

By assuming a frictionless impact, the expression in (2.31) can be rewritten to

$$\boldsymbol{\tau}_{co} = \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top(\mathbf{q})\lambda_{N,i}, \tag{2.44}$$

where $\mathbf{J}_{i,N}(\mathbf{q}) \in \mathbb{R}^n$ is the normal component of $\mathbf{J}_i(\mathbf{q})$ and $\lambda_{N,i} \in \mathbb{R}$ is the normal contact force for the i-th contact point. Using (2.44) rather than (2.31), (2.43) can be expressed as

$$\mathbf{M}_r(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{fr} + \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top(\mathbf{q})\lambda_{N,i}. \tag{2.45}$$

Afterwards, (2.45) can be integrated over an infinitesimal impact duration $\Delta t \to 0$ to obtain

$$\lim_{\Delta t \to 0} \int_t^{t+\Delta t} \left(\mathbf{M}_r(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})\right) dt = \lim_{\Delta t \to 0} \int_t^{t+\Delta t} \left(\boldsymbol{\tau} + \boldsymbol{\tau}_{fr} + \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top(\mathbf{q})\lambda_{N,i}\right) dt. \tag{2.46}$$

Since the impact duration is very short compared to the time scale of the robot dynamics, it can be considered an instantaneous event. Furthermore, the controlled joint torque $\boldsymbol{\tau}$ does not change during the impact due to the sample rate of the controller, which is assumed to be 1 kHz. Finally, it is assumed that $\boldsymbol{\tau}_{fr} = \mathbf{0}_{n\times 1}$, i.e., there is no joint friction, that the joint velocities are bounded, and that the joint displacements are constant during the impact event. Applying these assumptions to (2.46) results in

$$\mathbf{M}_r(\dot{\mathbf{q}}^+ - \dot{\mathbf{q}}^-) = \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top \Lambda_{N,i}, \tag{2.47}$$

where $\Lambda_{N,i}$ denotes the normal impulsive contact force on the i-th contact point. The velocity of the i-th end-effector frame in the normal direction of the environment can be denoted as

$$\left({}^{L_i[B]}\mathbf{v}_{B,L_i}\right)_y = \mathbf{J}_{i,N}\dot{\mathbf{q}}. \tag{2.48}$$

Using Newton's law of restitution, the post-impact normal velocity can be defined as

$$\left({}^{L_i[B]}\mathbf{v}^+_{B,L_i}\right)_y = \mu_N \left({}^{L_i[B]}\mathbf{v}^-_{B,L_i}\right)_y, \tag{2.49}$$

Furthermore, it is assumed that the impact is inelastic, i.e., $\mu_N = 0$. By combining (2.48) and (2.49) with the assumption, it can be derived that

$$\mathbf{J}_{i,N}\dot{\mathbf{q}}^+ = \mu_N \mathbf{J}_{i,N}\dot{\mathbf{q}}^- = 0. \tag{2.50}$$

By using (2.47) and (2.50), an expression to calculate the post-impact joint velocities and impulsive forces for all points in contact with the environment can be derived to be

$$\begin{bmatrix} \dot{\mathbf{q}}^+ \\ \Lambda_{N,1} \\ \vdots \\ \Lambda_{N,n_c} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_r & -\mathbf{J}_{1,N}^\top & \dots & -\mathbf{J}_{n_c,N}^\top \\ \mathbf{J}_{1,N} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{n_c,N} & 0 & \dots & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{M}_r\dot{\mathbf{q}}^- \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.51}$$

This expression allows for the calculation of the post-impact joint velocity for single- or multi-contact impacts for a robotic manipulator with rigid joints.

For a robotic manipulator with flexible joints, a similar approach is taken to the method used to calculate (2.51). By substituting (2.44) into (2.30), which is then substituted together with (2.33) into (2.32), the dynamics for a flexible joint robotic manipulator can be rewritten to

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) &= \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_{fr} + \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top(\mathbf{q})\lambda_{N,i}, \\ \mathbf{B}_\theta\ddot{\boldsymbol{\theta}} + \mathbf{D}_s\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_J &= \boldsymbol{\tau}_{act}. \end{aligned} \tag{2.52}$$

Afterwards, (2.52) can be integrated over an infinitesimal impact duration $\Delta t \to 0$ to obtain

$$\begin{aligned} \lim_{\Delta t\to 0}\int_t^{t+\Delta t} (\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}))\, dt &= \lim_{\Delta t\to 0}\int_t^{t+\Delta t} \left(\boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_{fr} + \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top(\mathbf{q})\lambda_{N,i}\right) dt, \\ \lim_{\Delta t\to 0}\int_t^{t+\Delta t} \left(\mathbf{B}_\theta\ddot{\boldsymbol{\theta}} + \mathbf{D}_s\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_J\right) dt &= \lim_{\Delta t\to 0}\int_t^{t+\Delta t} (\boldsymbol{\tau}_{act})\, dt. \end{aligned} \tag{2.53}$$

This is followed by applying the same assumptions that were used for the rigid manipulator case. Namely, that the controlled joint torque $\boldsymbol{\tau}$ does not change during the impact due to the sample rate of the controller, there is no joint friction, the joint velocities are bounded, and the joint displacements are constant during the impact event. Applying these assumptions to (2.53) results in

$$\begin{aligned} \mathbf{M}(\dot{\mathbf{q}}^+ - \dot{\mathbf{q}}^-) &= \sum_{i=1}^{n_c} \mathbf{J}_{i,N}^\top \Lambda_{N,i}, \\ \mathbf{B}_\theta(\dot{\boldsymbol{\theta}}^+ - \dot{\boldsymbol{\theta}}^-) &= \mathbf{0}_{n\times 1}. \end{aligned} \tag{2.54}$$

From this, it can be concluded that the motor velocities remain constant during the impact, i.e., $\dot{\boldsymbol{\theta}}^+ = \dot{\boldsymbol{\theta}}^-$. It can also be concluded that, besides the mass matrix, the expressions (2.54) and (2.47) are identical. This implies that the derivation of the impact map also remains the same. This results in the following expressions to determine the post-impact joint and motor velocity.

$$\begin{bmatrix} \dot{\mathbf{q}}^+ \\ \Lambda_{N,1} \\ \vdots \\ \Lambda_{N,n_c} \end{bmatrix} = \begin{bmatrix} \mathbf{M} & -\mathbf{J}_{1,N}^\top & \dots & -\mathbf{J}_{n_c,N}^\top \\ \mathbf{J}_{1,N} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{n_c,N} & 0 & \dots & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{M}\dot{\mathbf{q}}^- \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.55}$$

$$\dot{\boldsymbol{\theta}}^+ = \dot{\boldsymbol{\theta}}^-. \tag{2.56}$$

However, note that (2.51) and (2.55) are only possible if the matrix containing the contact point Jacobians and mass matrix is not singular. For a two-dimensional model, a maximum of two normal Jacobians can be used in the impact map formulation and for a three-dimensional model this can be extended to a maximum of three normal Jacobians. This is equivalent to a maximum of two and three contact points that the impact map can be applied for at the same time, respectively. This is also known as a hyperstatic problem, where the system is statically indeterminate.

With the rigid impact maps defined, a method to calculate the normal contact force applied to the end-effector at each contact point will need to be derived. To achieve this, the first and second time derivatives are taken of (2.41) when it is active, i.e., $h_i = 0$, this yields

$$\dot{h}_i = \mathbf{J}_{i,N}(\mathbf{q})\dot{\mathbf{q}} = 0, \tag{2.57}$$

and

$$\ddot{h}_i = \mathbf{J}_{i,N}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{i,N}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = 0. \tag{2.58}$$

Together with the manipulator dynamics, the contact force and joint accelerations during persistent contact can be calculated as

$$\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda_{N,1} \\ \vdots \\ \lambda_{N,n_c} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_r & -\mathbf{J}_{1,N}^\top & \dots & -\mathbf{J}_{n_c,N}^\top \\ \mathbf{J}_{1,N} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{n_c,N} & 0 & \dots & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau} - \mathbf{C}\dot{\mathbf{q}} - \mathbf{g} \\ -\dot{\mathbf{J}}_{1,N}\dot{\mathbf{q}} \\ \vdots \\ -\dot{\mathbf{J}}_{n_c,N}\dot{\mathbf{q}} \end{bmatrix} \tag{2.59}$$

and

$$\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda_{N,1} \\ \vdots \\ \lambda_{N,n_c} \end{bmatrix} = \begin{bmatrix} \mathbf{M} & -\mathbf{J}_{1,N}^\top & \dots & -\mathbf{J}_{n_c,N}^\top \\ \mathbf{J}_{1,N} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{n_c,N} & 0 & \dots & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J - \mathbf{C}\dot{\mathbf{q}} - \mathbf{g} \\ -\dot{\mathbf{J}}_{1,N}\dot{\mathbf{q}} \\ \vdots \\ -\dot{\mathbf{J}}_{n_c,N}\dot{\mathbf{q}} \end{bmatrix} \tag{2.60}$$

Here, (2.59) and (2.60) calculate the normal contact force for each contact point for the rigid and flexible joint robotic manipulator, respectively. Similarly to (2.51) and (2.55), (2.59) and (2.60) are also statically indeterminate if there are too many contact points, which means that for the simulated two-dimensional case only two contact points can make contact at the same time.

2.2.4 Task-Based QP Robot Control

In the three models discussed in this section, the control inputs for the robotic manipulator are the actuation torques $\boldsymbol{\tau}$. The computation of these control inputs is subject to constraints such as kinematic and dynamic limits of the robotic manipulator. To determine the required joint torques for the robotic manipulator to complete a task, Quadratic Programming (QP) robot control is used. QP robot control uses a quadratic cost function and linear equality and inequality constraints to find the optimal values for the optimization parameters that minimize the cost function while satisfying the constraints. When formulating the cost function, a set of tasks can be defined that the robotic manipulator should perform, this is called task-based QP robot control. Examples of tasks are the task-space end-effector trajectory, joint-space trajectory, or contact task in the form of the desired contact wrench. The general formulation of the quadratic cost function and the (in)equality constraints is given as [37, 38, 39, 40]

$$\begin{aligned} \min_{\mathcal{X}} \quad & \frac{1}{2}\mathcal{X}^\top \mathbf{H}\mathcal{X} + \mathcal{X}^\top \mathbf{Q} \\ \text{s.t.} \quad & \mathbf{A}_{in}\mathcal{X} \leq \mathbf{b}_{in}, \\ & \mathbf{A}_{eq}\mathcal{X} = \mathbf{b}_{eq}, \\ & \mathcal{X}_{min} \leq \mathcal{X} \leq \mathcal{X}_{max}, \end{aligned} \tag{2.61}$$

with $\mathcal{X} \in \mathbb{R}^v$ the $v \in \mathbb{N}$ optimization variables, $\mathbf{H} \in \mathbb{R}^{v \times v}$ and $\mathbf{Q} \in \mathbb{R}^v$ the cost function's symmetric (semi)definite Hessian matrix and gradient vector, respectively. Furthermore, $\mathcal{X}_{min} \in \mathbb{R}^v$ and $\mathcal{X}_{max} \in \mathbb{R}^v$ are the lower- and upper bounds on the optimization variables, respectively, $\mathbf{A}_{in} \in \mathbb{R}^{r \times v}$ and $\mathbf{b}_{in} \in \mathbb{R}^r$ representing the $r \in \mathbb{N}$ inequality constraints, and $\mathbf{A}_{eq} \in \mathbb{R}^{s \times v}$ and $\mathbf{b}_{eq} \in \mathbb{R}^s$ representing the $s \in \mathbb{N}$ equality constraints.

When defining several tasks for task-based QP robot control, conflicting tasks may occur. There are two distinct approaches on how to handle these conflicting tasks, the *hierarchical approach* and the *weighted approach*. The hierarchical approach uses a strict order of importance to determine which task takes priority over another if conflicts arise. The weighted approach assigns task weights to each task, and a trade-off in task performance is made for conflicting tasks based on these weights. The robotic manipulator simulation employs a weighted approach for the QP controller, which reformulates the cost function presented in (2.61) to be the sum of $\mathcal{N}_\mathcal{T} \in \mathbb{N}$ weighted task errors $\mathcal{T}_i$. This weighted task cost function $\mathcal{C}$ is given as

$$\mathcal{C} = \sum_{i=1}^{\mathcal{N}_\mathcal{T}} \mathcal{T}_i, \tag{2.62}$$

where

$$\begin{aligned} \mathcal{T}_i \quad &:= \frac{1}{2}\left\|\mathbf{E}_i\mathcal{X} + \mathbf{e}_i\right\|^2_{\mathbf{W}_i} \\ &= \frac{1}{2}(\mathbf{E}_i\mathcal{X} + \mathbf{e}_i)^\top \mathbf{W}_i(\mathbf{E}_i\mathcal{X} + \mathbf{e}_i) \\ &= \frac{1}{2}\mathcal{X}^\top \mathbf{E}_i^\top \mathbf{W}_i \mathbf{E}\mathcal{X} + \mathbf{X}^\top \mathbf{E}_i^\top \mathbf{W}_i \mathbf{e}_i + \frac{1}{2}\mathbf{e}_i^\top \mathbf{e}_i. \end{aligned} \tag{2.63}$$

In (2.63), $\mathbf{W}_i$ is a symmetric weighting matrix that prioritizes one task over another in the case of conflicting tasks, $\mathbf{E}_i$ and $\mathbf{e}_i$ are task-dependent error matrices. Combining the cost function defined in (2.62) and (2.63) with the QP formulation found in (2.61), it is possible to define

$$\mathbf{H} := \sum_{i=1}^{\mathcal{N}_\mathcal{T}} \mathbf{E}_i^\top \mathbf{W}_i \mathbf{E}_i \quad \text{and} \quad \mathbf{Q} := \sum_{i=1}^{\mathcal{N}_\mathcal{T}} \mathbf{E}_i^\top \mathbf{W}_i \mathbf{e}_i. \tag{2.64}$$

As can be seen in the expression for $\mathbf{H}$ and $\mathbf{Q}$, $\mathbf{e}_i^\top \mathbf{e}_i$ is not used in the formulation of the cost function. Because $\mathbf{e}_i^\top \mathbf{e}_i$ is not a function of the optimization variables $\mathcal{X}$, it will therefore not influence the results from the QP problem and can safely be discarded. Using this optimization problem it is possible to determine the optimization variables that minimize the cost function at time t_k, where k is an integer representing the timestep, with controller sample interval $\Delta t = t_k - t_{k-1}$.

The optimization variables are chosen to be the joint accelerations $\ddot{\mathbf{q}}$, the applied joint torques $\boldsymbol{\tau}$, and the normal contact force for each contact point $\lambda_{N,i}$. Furthermore, two tasks are defined for the planar robotic manipulator, the end-effector pose task and the contact task. The end-effector pose task can be expressed in task-space or joint-space. These tasks will be explained further in the following sections. The optimization constraints will also be discussed below.

Task-Space End-Effector Pose Task

The task-space end-effector pose task aims to move the end-effector, such that the origin and orientation of the end-effector link frame L coincides with the desired position and orientation of the end-effector.

The task can be defined as

$$\begin{aligned}\mathcal{T}_{pos} &= \frac{1}{2}\left\| {}^{L[A]}\dot{\mathbf{v}}_{A,L} - {}^{L[A]}\dot{\mathbf{v}}^{d}_{A,L} \right\|^2_{\mathbf{W}_{pos}} \\ &= \frac{1}{2}\left\| {}^{L[A]}\mathbf{J}_{A,L}\ddot{\mathbf{q}} + {}^{L[A]}\dot{\mathbf{J}}_{A,L}\dot{\mathbf{q}} - {}^{L[A]}\dot{\mathbf{v}}^{d}_{A,L} \right\|^2_{\mathbf{W}_{pos}}\end{aligned} \tag{2.65}$$

with ${}^{L[A]}\dot{\mathbf{v}}_{A,L} \in \mathbb{R}^6$ the end-effector acceleration. Note that the Jacobian is defined with respect to the inertial frame A rather than environmental frame B, this is because the controller does not know the geometry of the environment and assumes that it is horizontal and that the elevation with respect to the base is 0, i.e., the controller assumes that $B = A$. Furthermore,

$${}^{L[A]}\dot{\mathbf{v}}^{d}_{A,L} = \begin{bmatrix} {}^{A}\ddot{\mathbf{p}}_{ref} + \mathbf{K}_d({}^{A}\dot{\mathbf{p}}_{ref} - {}^{A}\dot{\mathbf{o}}_L) + \mathbf{K}_p({}^{A}\mathbf{p}_{ref} - {}^{A}\mathbf{o}_L) \\ {}^{A}\dot{\boldsymbol{\omega}}_{A,L,ref} + k_d({}^{A}\boldsymbol{\omega}_{A,L,ref} - {}^{A}\boldsymbol{\omega}_{A,L}) + k_p({}^{A}\boldsymbol{\phi}_{ref} - {}^{A}\boldsymbol{\phi}_L) \end{bmatrix} \tag{2.66}$$

is the desired end-effector acceleration. In (2.66), ${}^{A}\mathbf{p}_{ref}$ is the desired end-effector position, $\mathbf{o}_L$ is the origin of the end-effector control frame L, ${}^{A}\boldsymbol{\phi}_{ref}$ is the reference orientation vector of the end-effector with respect to frame A, and ${}^{A}\boldsymbol{\phi}_L$ is the vector notation of the orientation of end-effector frame L with respect to frame A.

Joint-Space configuration Task

The joint-space configuration task aims to move the robotic manipulator from its initial configuration $\mathbf{q}_0$ to follow the defined task reference trajectory. The task is defined as

$$\mathcal{T}_{config} = \frac{1}{2}\left\| \ddot{\mathbf{q}} - \ddot{\mathbf{q}}^{d} \right\|^2_{\mathbf{W}_{config}} \tag{2.67}$$

where

$$\ddot{\mathbf{q}}^{d} = \ddot{\mathbf{q}}_{ref} + \mathbf{K}_d(\dot{\mathbf{q}}_{ref} - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_{ref} - \mathbf{q}), \tag{2.68}$$

The joint-space configuration task has a similar function to the task-space end-effector pose task. However, the joint-space configuration task always controls all degrees of freedom while the task-space end-effector pose task can control at most 6 degrees of freedom.

Contact Task

The contact task aims to ensure that the robotic manipulator maintains contact with the environment after the initial impact has occurred, regardless of if the geometry of the impact surface coincides exactly with the desired post-impact reference trajectory. This is a simple task that aims to control the normal contact force exerted on the environment. The i-th contact task is defined as

$$\mathcal{T}_{contact,i} = \frac{1}{2}\left\| \lambda_{N,i} - \left(\lambda_{N,i,ref} + k_p\left(\lambda_{N,i,ref} - \lambda_{N,i,meas}\right)\right) \right\|^2_{\mathbf{W}_{contact}}, \tag{2.69}$$

where $\lambda_{N,i,ref}$ and $\lambda_{N,i,meas}$ are the reference and measured normal contact force for the i-th contact point. Note that for most systems, no contact force estimation or measurements are available. In these cases, the proportional feedback gain k_p is considered to be 0.

Optimization Constraints

Due to the kinematic and dynamic limits of the robotic manipulator, the QP optimization problem presented in (2.61) is subjected to constraints. The constraints are formulated in terms of optimization variables and are heavily dependent on the controlled system. For the robotic manipulator discussed in Section 2.2, limits that should be considered lower- and upper-bound joint accelerations, applied joint torques, and normal contact forces. The optimization variables constraints for the robotic manipulator considered in this project as found in [6] are chosen to be

$$\mathcal{X}_{min} = \begin{bmatrix} -\infty \\ -\infty \\ -\infty \\ -87 \\ -87 \\ -12 \\ \mathbf{0}_{n_c\times 1} \end{bmatrix}, \quad \mathcal{X}_{max} = \begin{bmatrix} \infty \\ \infty \\ \infty \\ 87 \\ 87 \\ 12 \\ \boldsymbol{\infty}_{n_c\times 1} \end{bmatrix}, \tag{2.70}$$

Furthermore, the contact constraints are implemented similarly to how the contact forces were calculated in (2.59) and (2.60) when contact has been established. If no contact has been established, then the constraint $\lambda_{N,i} = 0$ is applied instead. Furthermore, when implementing the contact constraints into the optimization problem, it is assumed that the environment frame B is identical to the inertial frame A because the robot only has information about the expected geometry of the environment.

2.3 Defining Simultaneous Impacts

In section 1.2, a brief explanation was given to give an idea of what a simultaneous impact is. However, a more thorough explanation of simultaneous impacts is needed before a classifier can be developed for these types of impacts. As the name suggests, a simultaneous impact is an impact where multiple points make contact with the environment simultaneously. However, due to perturbations in the state trajectory, an impact is not guaranteed nor expected to be truly simultaneous, regardless of the intent of the impact. Assuming a planar inelastic impact where established contact is maintained, two possible scenarios can occur which are both shown in Figure 2.6.

The top example presented in Figure 2.6, shows a scenario where the entire edge of an object makes contact with the impact surface simultaneously. This is the desired behavior for a simultaneous impact and will be called an ideal simultaneous impact, where at least two gap functions close at the same time. In Figure 2.6, the detected impact time is defined as $t_1 \in [0, +\infty)$. Furthermore, considering that the ideal simultaneous impact occurs at the desired impact time, this impact is also said to occur at $t_{1,d} \in [0, +\infty)$, which is the first desired major impact event.

However, as explained before, due to perturbations the simultaneity of the impact is likely lost. In such a case, several single impacts occur after each other before contact along the entire edge or surface is completed, i.e., one gap function remains closed while other gap functions are closing. An example of such a case is shown in the bottom part of Figure 2.6, where the impact is detected at two different times, the first detected impact time $t_1 \in [0, +\infty)$ and the second detected impact time $t_2 \in (t_1, +\infty)$. Furthermore, Figure 2.6 represents a planar simultaneous impact, also known as an edge impact. In three-dimensional space, an additional impact $t_3 \in (t_2, +\infty)$ can occur and the resulting impact is known as a surface impact, however, the principle behind the impact remains the same. When discussing the

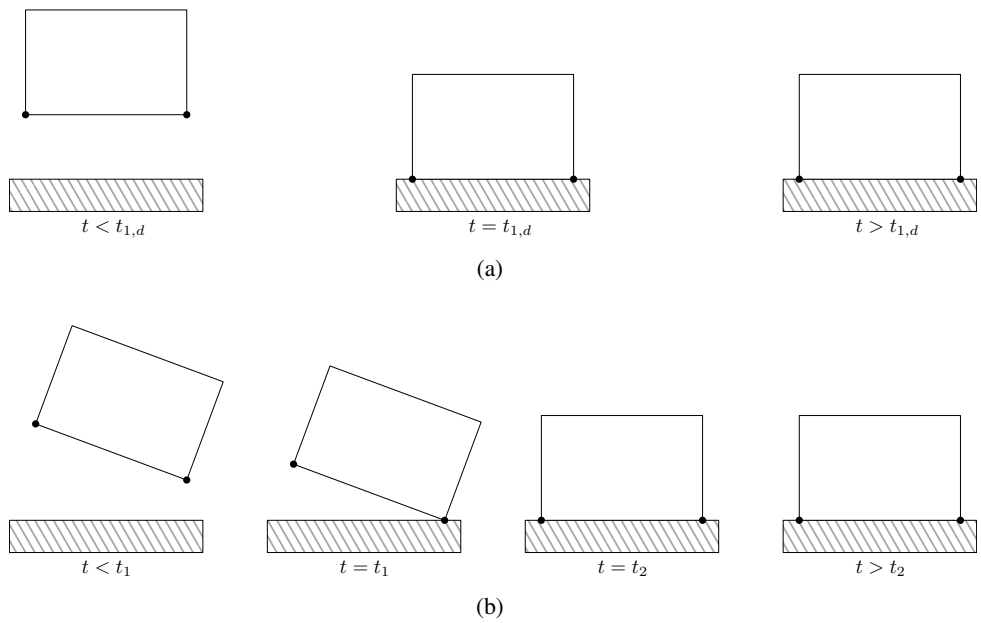


Figure 2.6: Simultaneous Impact Scenarios for an inelastic impact. In (a), an ideal simultaneous impact scenario where at least two points impact the environment at the same time at the desired impact time $t = t_{1,d}$. In (b), a non-ideal simultaneous impact where, due to perturbations, one point impacts the environment sooner at $t = t_1$ than the other at $t = t_2 > t_1$.

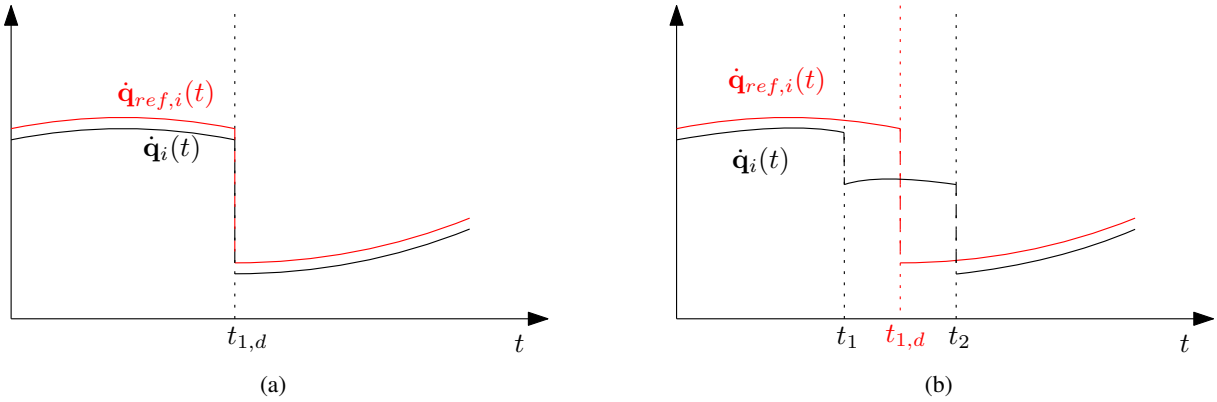


Figure 2.7: Comparison between the reference state trajectory and the state trajectory. In (a), the state trajectory of a rigid ideal simultaneous impact. In (b), the state trajectory of a rigid non-ideal simultaneous impact.

simultaneous impact case, both of these cases are considered unless specified otherwise.

When considering a single impact case, there is a clear ante- and post-impact phase, an ideal simultaneous impact also has a clear ante- and post-impact phase. However, when simultaneity is lost, an additional phase is introduced in the dynamics. This phase represents a situation where one point has made contact, but the contact has not been completed along the entire edge or surface. This takes place over the interval $t \in [t_1, t_2)$ for an edge impact and $t \in [t_1, t_3)$ for a surface impact. When only one point has made contact, a state transition occurs, which is followed by a second state transition when another point makes contact with the environment, this continues until contact has been completed along the entire edge or surface. This additional phase between state transitions that occurs due to the loss of simultaneity will be called the intermediate phase. Furthermore, the post-impact phase is the phase after contact has been completed, i.e., after all impacts have occurred.

This behavior is best shown in the schematics presented in Figure 2.7, which shows the differences between state transitions for a rigid ideal simultaneous impact, and a planar rigid simultaneous impact where simultaneity has been lost. Furthermore, a comparison between a reference joint velocity trajectory is made that shows the main difference for post-impact behavior classification for simultaneous impacts.

2.4 Reference Spreading

A robotic manipulator impacting a surface can be represented as a hybrid system, a system containing both continuous flows and discontinuous jumps [41]. If the state of the robotic manipulator at time $t \in [t_0, t_f]$ is represented by a state vector $\mathbf{x} \in \mathbb{R}^n$ the continuous flow evolves according to

$$\dot{\mathbf{x}} = \boldsymbol{k}(\mathbf{x}, \mathbf{u}, t), \tag{2.71}$$

with $\mathbf{u} \in \mathbb{R}^m$ the input, $m \in \mathbb{N}$ the number of inputs, t_0 the initial time of operation, and t_f the final time of operation. Furthermore, (2.71) is the general form of the robotic manipulator model discussed in Section 2.2 if $\mathbf{x} = \left[\mathbf{q}^\top, \dot{\mathbf{q}}^\top\right] \in \mathbb{R}^{2n}$ is used instead. Evolution according to (2.71) is only possible as long as certain constraints are satisfied. In the case of a robotic manipulator impacting a surface, these constraints are the position constraints $h_i \geq 0$ for each contact point. When contact with the environment has been made and the boundary of the position constraint has been reached, an impulsive force is applied to the system which results in a jump in the state vector formally defined as

$$\mathbf{x}^+ = \boldsymbol{g}(\mathbf{x}^-, t), \tag{2.72}$$

When controlling a system that can undergo these state-triggered jumps, a control strategy needs to be devised that allows for increased performance when the state-triggered jump time is not certain. In [42, 43], a method is proposed that extends the ante- and post-jump reference trajectories past the desired jump time. This method is referred to as *reference spreading control*. The standard implementation of

reference spreading control as described by [42, 43] is described in Section 2.4.1. However, for the implementation of reference spreading in operational space such as for a QP controller, a different approach needs to be taken to extend the references as shown in [44]. This approach is expanded in Section 2.4.2.

2.4.1 Standard Reference Spreading Control

In [42, 43], a time-varying state feedback control law is considered to be

$$\mathbf{u}(\mathbf{x}, t) = \boldsymbol{\mu}(t) - \mathbf{K}_g(\mathbf{x} - \boldsymbol{\alpha}(t)), \tag{2.73}$$

with $\boldsymbol{\mu} \in \mathbb{R}^m$ the reference feedforward and $\mathbf{K}_g \in \mathbb{R}^{m\times n}$ a matrix gain, and $\boldsymbol{\alpha} \in \mathbb{R}^n$ the reference trajectory. This control law allows for the local tracking of a set of trajectories designed around the desired state-triggered jump times $t_{j,d}$, with $j \in \mathbb{N}_{\geq 1}$ representing the j-th desired major impact event. However, due to the possible mismatch of the desired jump times $t_{j,d}$ and the actual jump times t_j due to perturbations, a phenomenon called peaking occurs. With peaking, the error $\mathbf{x} - \boldsymbol{\alpha}$ suddenly increases because the reference state trajectory and perturbed state trajectory both reside in different modes. This behavior is shown in Figure 2.8. How to handle the peaking phenomenon is expanded upon in the following sections for the single- and simultaneous impacts.

Reference Spreading for Single Impacts

As proposed in [42, 43], for single impacts, the reference state trajectories are extended past the desired impact time $t_{j,d}$ by extending the reference feedforward $\boldsymbol{\mu}$ and reference trajectory $\boldsymbol{\alpha}$. First of all, (2.73) is adapted to accommodate the extended feedforward and reference trajectories and is considered to be

$$\mathbf{u}(\mathbf{x}, t) = \overline{\boldsymbol{\mu}}(t) - \mathbf{K}_g(\mathbf{x} - \overline{\boldsymbol{\alpha}}(t)), \tag{2.74}$$

where $\overline{\boldsymbol{\mu}} \in \mathbb{R}^m$ the extended reference feedforward and $\overline{\boldsymbol{\alpha}} \in \mathbb{R}^n$ the extended reference state trajectory. The extended reference feedforward is designed and structured as

$$\overline{\boldsymbol{\mu}}(t) = \begin{cases} \overline{\boldsymbol{\mu}}_a(t), & t \in [t_0, t_{1,d} + \delta], \\ \overline{\boldsymbol{\mu}}_p(t), & t \in (t_{1,d} - \delta, t_f], \end{cases} \tag{2.75}$$

where subscripts $(\cdot)_a$ and $(\cdot)_p$ represent the ante- and post-impact phase, respectively, and δ is the time extension. Using this reference feedforward, the extended reference state can be determined through forward and backward integration of the dynamics with the feedforward reference as input. This results in the extended reference state trajectories defined as

$$\overline{\boldsymbol{\alpha}}(t) = \begin{cases} \overline{\boldsymbol{\alpha}}_a(t), & t \in [t_0, t_{1,d} + \delta] \\ \overline{\boldsymbol{\alpha}}_p(t), & t \in (t_{1,d} - \delta, t_f] \end{cases}. \tag{2.76}$$

Substituting (2.75) and (2.76) into (2.74) and introducing the switching condition the reference spreading feedback law can be defined as

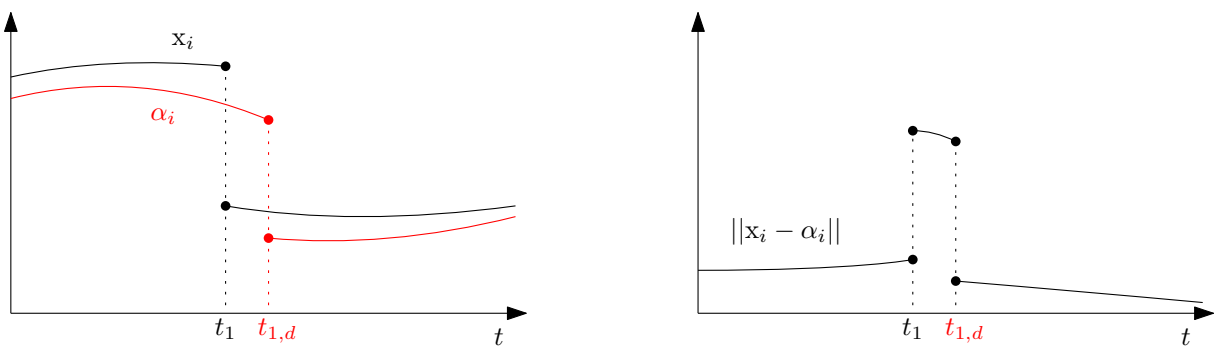


Figure 2.8: A schematic representation of peaking in the error that occurs due to the mismatch between jump times t_1 and $t_{1,d}$. In the example shown in the figure, during the period $t \in (t_1, t_{1,d}]$, the error between the perturbed state trajectory $\mathbf{x}_i$ and reference state trajectory $\boldsymbol{\alpha}_i$ peaks.

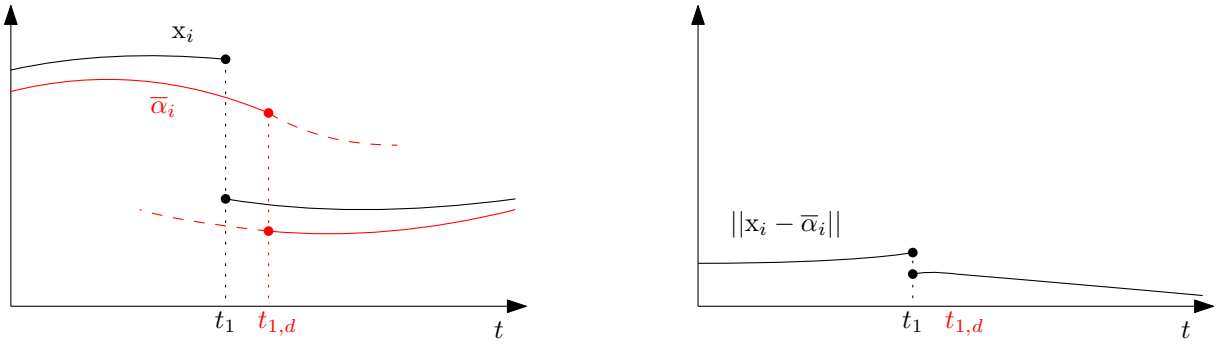


Figure 2.9: A schematic representation of the perturbed state trajectory $\mathbf{x}_i$ and extended reference state trajectory $\overline{\boldsymbol{\alpha}}_i$. Note that peaking in the error no longer occurs as the followed reference state trajectory and the perturbed state trajectory always reside in the same mode.

$$\mathbf{u}(\mathbf{x}, t) = \begin{cases} \overline{\boldsymbol{\mu}}_a(t) - \mathbf{K}_g(\mathbf{x} - \overline{\boldsymbol{\alpha}}_a(t)), & t \in [t_0, t_1] \\ \overline{\boldsymbol{\mu}}_p(t) - \mathbf{K}_g(\mathbf{x} - \overline{\boldsymbol{\alpha}}_p(t)), & t \in (t_1, t_f] \end{cases}. \tag{2.77}$$

A schematic representation of this new control law can be seen in Figure 2.9.

Reference Spreading for Simultaneous Inelastic Impacts

It is important to consider the implementation of reference spreading for systems undergoing simultaneous impacts. Recall the explanation regarding simultaneous impacts given in Section 2.3. Due to the possibility of two impact times, the control law presented in 2.4.1 is no longer valid. With simultaneous impacts, the intermediate phase is a phase where neither the ante- nor post-impact trajectory can be followed, as it would always result in the peaking phenomenon. This absence of trajectory that can be followed is expressed in Figure 2.10 by a red box within that region defined as the intermediate phase.

Under the assumption that contact is maintained until the post-impact phase, a control strategy can be designed for the intermediate phase, which occurs in the interval $t \in (t_1, t_N]$, with t_N being the final detected impact time. This can be done by reducing the feedback gain to 0 during the intermediate phase and remain as such until contact along the entire edge or surface has been completed. This results in the simultaneous impact reference spreading control law defined as [43, 45]

$$\mathbf{u}(\mathbf{x}, t) = \begin{cases} \overline{\boldsymbol{\mu}}_a(t) - \mathbf{K}_g(\mathbf{x} - \overline{\boldsymbol{\alpha}}_a(t)), & t \in [t_0, t_1], \\ \overline{\boldsymbol{\mu}}_a(t), & t \in (t_1, t_N], \\ \overline{\boldsymbol{\mu}}_p(t) - \mathbf{K}_g(\mathbf{x} - \overline{\boldsymbol{\alpha}}_p(t)), & t \in (t_N, t_f], \end{cases} \tag{2.78}$$

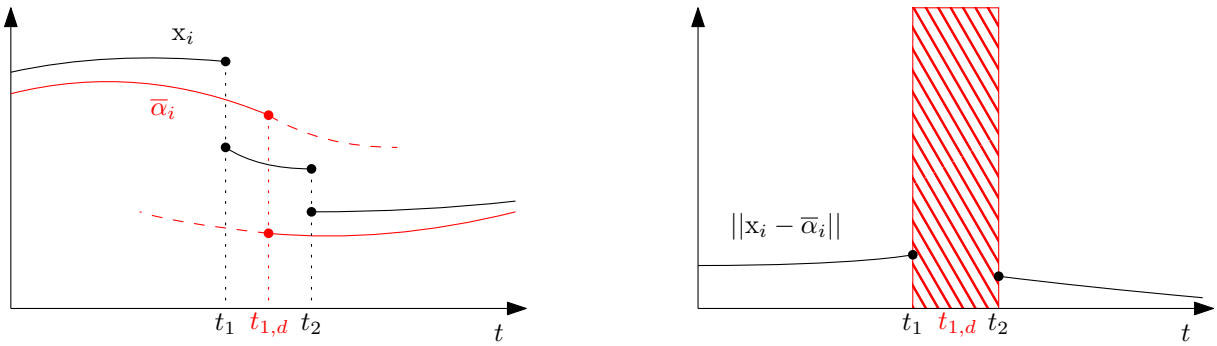


Figure 2.10: A schematic representation of the perturbed state trajectory $\mathbf{x}_i$ and extended reference state trajectory $\overline{\boldsymbol{\alpha}}_i$ for a simultaneous impact case. Note that in the period $t \in (t_1, t_2]$, it is not possible to determine whether the ante- or post-impact reference state trajectory should be followed, which is indicated by the red box where the error is unknown.

The switching from the ante-impact phase to the intermediate phase occurs when the first impact is detected at t_1, and where it will switch to the post-impact phase when contact completion has been detected at t_N, which concludes the simultaneous impact.

2.4.2 Expanding Reference Spreading Control for Task-Based QP Control

The reference spreading control law described in Section 2.4.1 only works for a system that is controlled through a feedback control law with a known feedforward. However, this cannot be implemented in the same way for a system where the input to the system is determined through a QP optimization problem. The reference spreading approach described in Section 2.4.1 uses a designed extended reference feedforward to determine the extended reference state trajectories, however, no reference feedforward is available when using a QP robot controller. Furthermore, QP robot control also uses task trajectories rather than reference state trajectories. This complicates the forward and backward integration to obtain the extended reference state trajectories.

To solve this, a method where the task trajectories are extended directly is considered. This method is explained further in the following sections for single- and simultaneous impacts.

Task-Based Reference Spreading for Single Impacts

When designing the extended task trajectories for QP robot control, it is important to start with defining the different task trajectories. This task reference trajectory before reference spreading is applied is known as $\boldsymbol{\gamma}(t)$ and contains the task-space pose or the joint configuration task reference trajectories, as well as the normal contact force task reference. Similarly to how it is presented in Section 2.4.1, this task reference trajectory takes into account the state-triggered jump at $t_{j,d}$. This also introduces a similar problem as presented in Section 2.4.1, where due to perturbations, the perturbed task trajectory will trigger the state-triggered jump at t_j which is not guaranteed to coincide with $t_{j,d}$.

To solve this, two trajectories are defined, the ante- and post-impact task reference trajectories. The task reference trajectory will then become

$$\overline{\boldsymbol{\gamma}}(t) = \begin{cases} \overline{\boldsymbol{\gamma}}_a(t), & t \in [t_0, t_1], \\ \overline{\boldsymbol{\gamma}}_p(t), & t \in (t_1, t_f]. \end{cases} \tag{2.79}$$

Extending $\overline{\boldsymbol{\gamma}}_a(t)$ and $\overline{\boldsymbol{\gamma}}_p(t)$ past $t_{1,d}$ is no longer done through forward or backward integration. Instead, the ante-impact task reference trajectory is extended by continuing its motion past $t_{1,d}$ until it has reached a target defined as $\overline{\boldsymbol{\gamma}}_a(t_{1,d} + \delta)$, whereas the post-impact task reference trajectory starts at $\overline{\boldsymbol{\gamma}}_p(t_{1,d} - \delta)$ and a reference is defined to connect it to $\overline{\boldsymbol{\gamma}}_p(t_{1,d})$. The points $\overline{\boldsymbol{\gamma}}_a(t_{1,d} + \delta)$ and $\overline{\boldsymbol{\gamma}}_p(t_{1,d} - \delta)$ are defined based on the limits and desired motions of the system, and is dependent on the function of the individual tasks. Furthermore, the task reference trajectories should be designed to be continuous, to ensure a smooth transition from the extended part of the reference to the original part of the reference.

Task-Based Reference Spreading for Simultaneous Inelastic Impacts

Expanding the task-based reference spreading for single impacts to a simultaneous impact case can be done using a similar approach to how it was performed for the reference spreading control law discussed in Section 2.4.1, i.e., the feedback gains are reduced to zero. However, before reducing the feedback gain to zero for the intermediate phase, a condition needs to be satisfied first. This condition is that the error formulation should not contain the state variables that are desired to be omitted by reducing the feedback gains to zero.

As an example, consider the task-space end-effector pose task presented in (2.65) and (2.66). When the feedback gains are reduced to zero, the error formulation will still contain $\dot{\mathbf{q}}$. However, when considering a joint-space configuration task as presented in (2.67), reducing the feedback gains to zero removes $\dot{\mathbf{q}}$ from the error formulation. Therefore, one possible solution for implementing reference spreading for a QP-controlled system that can undergo simultaneous impacts is to formulate the tasks in joint-space. Furthermore, when considering a robotic manipulator such as what is considered in Section 2.2, it is not required to reduce the proportional feedback gain $\mathbf{K}_p$ to zero as these can be used to ensure to increase the performance of the intermediate phase. Only the derivative feedback gain $\mathbf{K}_d$ should be reduced to

zero because the estimation of the joint velocities is unreliable when the system is undergoing an impact. This leads to the task reference trajectory for a simultaneous impact case to be defined as

$$\overline{\boldsymbol{\gamma}}(t) = \begin{cases} \overline{\boldsymbol{\gamma}}_a(t), & t \in [t_0, t_1], \\ \overline{\boldsymbol{\gamma}}_a(t), & t \in (t_1, t_N], \\ \overline{\boldsymbol{\gamma}}_p(t), & t \in (t_N, t_f]. \end{cases} \tag{2.80}$$

The derivative feedback gain strategy can then be defined as

$$\begin{aligned} &\text{if } t \in [t_0, t_1], && \mathbf{K}_d \neq 0, \\ &\text{if } t \in (t_1, t_N], && \mathbf{K}_d = 0, \\ &\text{if } t \in (t_N, t_f], && \mathbf{K}_d \neq 0. \end{aligned} \tag{2.81}$$

2.5 Momentum Observer

When classifying impact behavior, it is important to have an impact detection method in place to ensure that the impacts that will be classified are detected. Impact detection is possible through several methods, which were briefly discussed in Section 1.1. For this project, the focus will be put on external torque estimators, namely the momentum observer, as this method does not require additional sensors [15] and also provides a solution to the isolation and identification step.

Before the momentum observer can be formulated, several preliminaries need to be defined first. The first is the robotic manipulator dynamics, which can be defined as (2.29) and (2.32) for a rigid and flexible joint robotic manipulator, respectively. Furthermore, from the skew-symmetry of matrix $\dot{\mathbf{M}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ it follows that

$$\dot{\mathbf{M}}(\mathbf{q}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{C}^\top(\mathbf{q}, \dot{\mathbf{q}}). \tag{2.82}$$

Furthermore, the generalized momentum can be defined as

$$\boldsymbol{p} = \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.83}$$

with its time derivative

$$\dot{\boldsymbol{p}} = \dot{\mathbf{M}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}. \tag{2.84}$$

Substituting (2.82) and either (2.29) or (2.32) into (2.84) results in

$$\dot{\boldsymbol{p}} = \boldsymbol{\tau}_{tot} + \mathbf{C}^\top(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{g}(\mathbf{q}), \tag{2.85}$$

where

$$\boldsymbol{\tau}_{tot} = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{2.86}$$

for a rigid joint robotic manipulator, and

$$\boldsymbol{\tau}_{tot} = \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J + \boldsymbol{\tau}_{ext} \tag{2.87}$$

for flexible joint robotic manipulators. Using the integral of (2.85), it is possible to derive that is must hold that

$$\boldsymbol{p}(t) - \int_0^t \dot{\boldsymbol{p}} ds - \boldsymbol{p}(0) = 0 \tag{2.88}$$

Using (2.88), (2.85), and either (2.87) or (2.86) for a flexible or rigid joint robotic manipulator, respectively, it is possible to derive a residual that will converge to the external joint torque. This residual takes the form of

$$\mathbf{r}(t) = \mathbf{K}_O \left[\boldsymbol{p}(t) - \int_0^t (\boldsymbol{\tau}_{in} + \mathbf{C}^\top(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{r}(t)) ds - \boldsymbol{p}(0) \right], \tag{2.89}$$

where $\mathbf{r}(t) \in \mathbb{R}^n$ is the estimated external joint torque for a robotic manipulator, $\mathbf{K}_O$ is the diagonal observer gain matrix, $\boldsymbol{\tau}_{in} = \boldsymbol{\tau}$ for rigid joint robotic manipulators, and $\boldsymbol{\tau}_{in} = \boldsymbol{\tau}_J + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_J$ for flexible joint robotic manipulators. The diagonal observer gain matrix $\mathbf{K}_O$ is usually taken to be between $10\mathbf{I}_n$ and $200\mathbf{I}_n$ [15, 18] and is based on the trade-off between the impact detection delay and the noise and accuracy of the external joint torque estimation. By increasing $\mathbf{K}_O$, the impact detection delay will be

shorter, but the noise will increase and the accuracy of the external joint torque estimation will decrease. For the purposes of this project, the observer gain matrix is chosen to be $\mathbf{K}_O = 30\mathbf{I}_n$. Taking the time derivative of (2.89) and substituting (2.84) into it, the dynamics of the momentum observer can be derived to be

$$\dot{\mathbf{r}} = \mathbf{K}_O \left(\boldsymbol{\tau}_{ext} - \mathbf{r} \right). \tag{2.90}$$

The dynamics presented in (2.90) show that, when a change in $\boldsymbol{\tau}_{ext}$ occurs, $\mathbf{r}$ will converge to $\boldsymbol{\tau}_{ext}$. Assuming that the external joint torque is bounded in free motion, an impact is detected if there exists an $i \in [1, n]$ such that

$$|\mathbf{r}_i| \geq r_b, \tag{2.91}$$

where $r_b \in \mathbb{R}$ is the external joint torque detection bound. Furthermore, the momentum observer allows for the isolation of the impact link, as the residual for all links past the impact link will not be affected by the impact. This method also provides a solution to the identification phase of the collision event pipeline as it estimates the external joint torque.

2.6 Conclusion

In this chapter, the notation, simulation models, simultaneous impacts, reference spreading, and the momentum observer were discussed in detail. However, the information provided here serves as a basis for Chapters 3 and 4, where the contents of this chapter will be used to build an impact-aware classification method to classify impacts. Furthermore, some of the subjects discussed in this chapter will be expanded upon in the coming chapters to achieve the research goal defined in Section 1.

3 Defining and Simulating Impact Scenarios

With the preliminary theoretical knowledge and the specifics of the used simulation models explained in the previous chapter, the impact scenarios for which the classifier will be designed need to be defined and simulated. This chapter will discuss what constitutes an expected and unexpected scenario, how multi-impact detection is implemented, further adjustments to the simulation to introduce non-ideal behavior into the simulation, how to obtain a prediction of the impact response and how this is used to obtain the task reference trajectory, and discussing simulation results.

In Section 3.1, the different model scenarios are discussed. In Section 3.2, a method to detect multiple impacts using existing impact detection methods is discussed. Section 3.3 discusses further adjustments to the simulation to introduce non-ideal behavior into the simulations. These adjustments are the inclusion of a delay in the impact detection algorithm, the control strategy, and the differences between task references for the single- and simultaneous impact case. Section 3.4 discusses how the prediction is obtained and how it is used to generate a trajectory. Finally, in Section 3.5, simulations are performed to show the impact response of the different model types for the single- and simultaneous impact case.

3.1 Model Scenarios

As was discussed in Section 2.2, the simulation that is used throughout the project is the planar approximation of the Franka Emika Panda developed in [6]. The planar approximation of the Franka Emika Panda is a 3-DOF robotic manipulator and a schematic illustration of such a system is shown in Figure 3.1. For the single impact scenarios, a spherical end-effector is used, but for simultaneous impact scenarios, an edge end-effector is used. In the following sections it is discussed what constitutes an expected scenario, and how an unexpected scenario differs from the expected scenario. This is done in Sections 3.1.1 and 3.1.2, respectively.

3.1.1 Expected Scenario

The expected scenario is dependent on two sets of information, information detailing the expected impact surface and user-defined trajectory parameters. The information detailing the expected impact surface refers to its material properties and geometry. The user-defined trajectory parameters are used to detail the task reference trajectories, examples of these parameters include the desired end-effector approach angle, impact location, impact velocity, and impact orientation and are partially based on the expected impact surface.

The expected impact surface is defined based on its material properties and geometry. However, as was discussed in Chapter 1, this classifier will focus on the geometric part of the impact classification problem. Therefore, the material properties of the impact surface will remain constant between expected and unexpected impact scenarios. For the material properties, it is assumed that the impact is inelastic

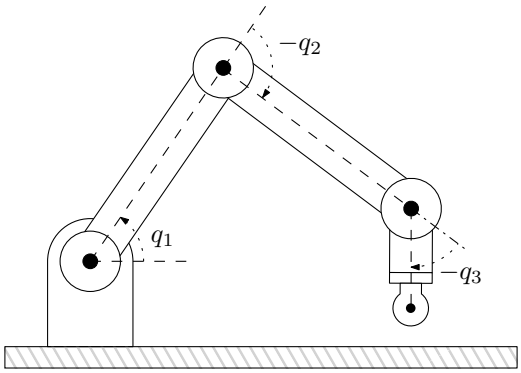


Figure 3.1: Schematic illustration of a 3-DOF robotic manipulator

Table 3.1: Tables showing expected impact surface parameters.

Parameter	Value	Unit
l	0	m
β	0	deg
k_c	$1.9657 \cdot 10^9$	N/m
d_c	$1.9657 \cdot 10^{13}$	Ns/m
μ_{fr}	0	$[-]$
μ_N	0	$[-]$

Table 3.2: Tables showing the user-defined trajectory parameters.

Parameter	Value	Unit
v_{imp}	0.2	m/s
x_{imp}	0.3	m
y_{imp}	0.02	m
ϕ_{imp}	-90	deg
α	120	deg
δx	0.1	m
δy	0.1	m
$t_{1,d}$	1	s
$\mathbf{q}_0$	$[45, -90, -45]^\top$	deg
λ_{ref}	30	N

Table 3.3: Tables showing end-effector parameters used in the simulation.

Parameter	Value	Unit
r	0.02	m
r_c	0.02	m
w	0.2	m
n_c	2	$-$

and frictionless, i.e., Newton's coefficient of restitution $\mu_N = 0$ and the friction coefficient $\mu_{fr} = 0$, and that the contact point stiffness k_c and damping d_c are calculated with an aluminum sphere impacting a wooden plane as explained in Section 2.2.3.

The geometry of the environment can be separated into two parameters. The first parameter is the angle β, which is the angle between $\vec{x}_B$ and $\vec{x}_A$ and represents the angle with which the impact surface has been rotated around $\vec{z}_A$. The second parameter is the elevation of the impact surface along the $\vec{y}_A$ axis, which is represented as l. In Section 2.2.1, additional details on how these parameters relate to the impact surface are presented.

The expected scenario assumes that the impact surface is horizontal and that it is placed at the same height as the base of the robotic manipulator. Therefore, the environment parameters used to define the expected impact surface can be defined as shown in Table 3.1.

The user-defined trajectory parameters are defined based on the requirements of the robotic manipulator, however, the desired impact location on the $\vec{x}_A$ and $\vec{y}_A$ axis, x_{imp} and y_{imp} are based on the known dimensions of the end-effector and the expected location of the impact surface. However, the initial configuration, end-effector impact approach angle, velocity, orientation, and time, can be selected freely. These parameters will also be the main focus when determining the desired trajectory, which will be explained further in Section 3.4. The user-defined trajectory parameters can be defined arbitrarily, however, unless stated otherwise, the parameters presented in Table 3.2 are used for the figures in this report. Finally, for completeness, the values used for the spherical end-effector radius r, distance from the origin of the edge end-effector contact point frame to the edge r_c, edge end-effector width w, and the number of contact points n_c are found in Table 3.3.

In Table 3.2, $\mathbf{q} = [q_1, q_2, q_3]^\top \in \mathbb{R}^3$ are the generalized coordinates of the planar robotic manipulator, $x_{imp} \in \mathbb{R}$ is the desired impact location on the $\vec{x}_A$ axis, $y_{imp} \in \mathbb{R}$ is the desired impact location on the $\vec{x}_A$ axis, and $\phi_{imp} = \sum_{i=1}^{3} \mathbf{q}_i(t_{1,d}) \in \mathbb{R}$ is the desired end-effector orientation at the desired time of impact $t_{1,d}$. Furthermore, x_{imp}, y_{imp}, and ϕ_{imp} can be used to formulate the desired impact position and orientation vector as

$$ {}^{A}\mathbf{p}_{imp} = \begin{bmatrix} x_{imp} \\ y_{imp} \\ 0 \\ 90 \\ 0 \\ \phi_{imp} \end{bmatrix}, \tag{3.1} $$

where the additional rotation around $\vec{x}_A$ is implemented to ensure that the end-effector is not positioned out of plane. Furthermore, $\alpha \in \mathbb{R}$ is the approach angle which is defined as the angle between the desired ante-impact velocity ${}^{L[A]}\boldsymbol{v}_{A,L}^{imp}$ and $\vec{x}_A$, furthermore, ${}^{L[A]}\boldsymbol{v}_{A,L}^{imp}$ is calculated using α and the desired absolute impact velocity v_{imp}. With the desired end-effector angular velocity defined as $\mathbf{0}_{3\times1}$, the desired end-effector twist at the time of impact is defined as

$$ {}^{L[A]}\mathbf{v}_{A,L}^{imp} = \begin{bmatrix} -v_{imp}\cos\alpha \\ -v_{imp}\sin\alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.2} $$

The parameter δx is the target position relative to the desired impact position and orientation ${}^{A}\mathbf{p}_{imp}$ in the $\vec{x}_A$ direction at the final simulation time t_f, δy is the target position relative to the impact position and orientation ${}^{A}\mathbf{p}_{imp}$ in the $\vec{y}_A$ direction at the final simulation time t_f for the ante-impact trajectory, and $\lambda_{ref} \in \mathbb{R}_{\geq 0}$ is the desired post-impact continuous contact force.

For a simultaneous impact case, the user-defined trajectory parameters can remain the same as those presented in Table 3.1. However, for parameters that differ from the parameters presented in the table, an additional restriction is applied. This restriction is that $\phi_{imp} = -90\ deg$, i.e., the end-effector orientation is perpendicular to the expected environment. This restriction is in place to ensure that if there are no perturbations and the end-effector is controlled precisely as is desired then the impact is an ideal simultaneous impact. Schematic representations of the expected single- and simultaneous impact scenarios are found in Figure 3.2.

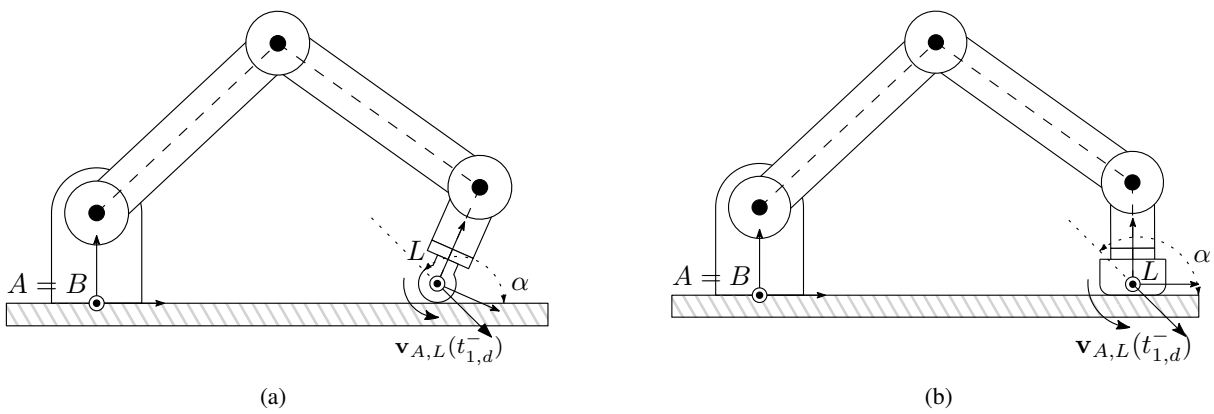


Figure 3.2: Schematic representations of two expected impact scenarios. In (a), an example of an expected single impact scenario. In (b), an example of an expected simultaneous impact scenario

3.1.2 Unexpected Scenarios

When considering an unexpected scenario, two different types of scenarios present themselves. The first unexpected scenario is a scenario where the impact occurs at a different location and time than was expected. The second unexpected scenario is a scenario where the impact location and time were expected, but the dynamics following the impact is not, i.e., the post-impact behavior is unexpected. Each of these scenarios can be represented by adjusting the elevation $l \in \mathbb{R}$ and rotation $\beta \in \mathbb{R}$ of the environment. These variables for defining the environment were further elaborated on in Section 2.2.1. Both scenarios, as well as how l and β can be used to define these scenarios, are explained below.

The most common cause for a difference between the expected and measured impact location and time is due to an object that obstructs the desired trajectory of the end-effector. In the simulation, this is represented by changing the elevation l of the environment without adjusting the desired trajectory. This principle is shown in Figure 3.3 (a) and (b), where two examples of adjusting the elevation of the environment can be seen, one where the environment is elevated and one where it is lowered.

An unexpected post-impact scenario occurs when the impact location was expected, but the geometry of the environment is different than expected. In the simulation, this is represented by the rotation of the impact surface β but adjusting l to ensure that the impact location remains the same. Using basic trigonometry, the elevation l that must be chosen to ensure that the impact location remains the same when regardless of the rotation of the environment is found to be

$$l = -\tan(\beta)^{A}\mathbf{x}_{A}^{\top A}\mathbf{p}_{imp}, \tag{3.3}$$

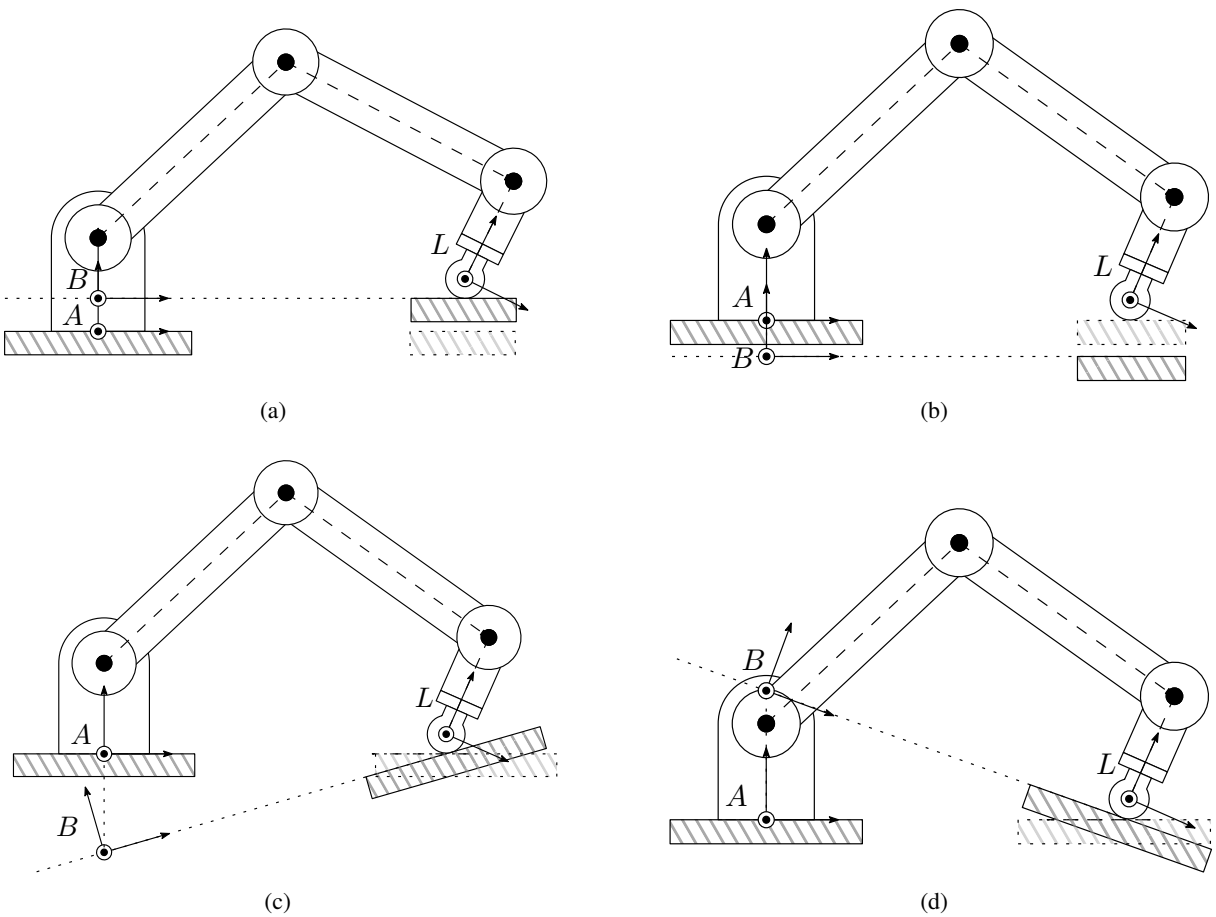


Figure 3.3: Schematic representation of four unexpected impact scenarios. In (a), where the impact surface is placed higher than expected, i.e., $l > 0$. In (b), where the impact surface is placed lower than expected, i.e., $l < 0$. In (c), where the environment is sloped upwards because $\beta > 0$. In (d), where the environment is sloped downwards because $\beta < 0$.

where ${}^{A}\mathbf{x}_A = [1, 0, 0, 0, 0, 0]^\top$ is the coordinate vector notation of unit vector $\vec{x}_A$ extended by $\mathbf{0}_{3\times1}$ to ensure that the equation is mathematically sound. Scenarios where $\beta \neq 0$ but where the impact location remains the same are also shown schematically in Figure 3.3 (c) and (d), where a scenario is shown where $\beta > 0$ and $\beta < 0$ are shown.

3.2 Multi-Impact Detection

Before developing the classifier, a method to detect impacts should be in place that works for both single- and simultaneous impacts. The most common method of impact detection is through the use of external joint torque estimators. Examples of these estimators include the direct and observer-extended direct method [18] and the momentum observer [15, 16, 17], where an impact is detected if the estimated external joint torque surpasses a user-defined threshold.

When considering which impact detection method will be used for the simulation, the direct method of external joint torque estimation can be disregarded due to the noisy estimation that results from this method [18]. Furthermore, the observer-extended direct is also disregarded as it requires Inertial Measurement Units (IMUs) to function, which are absent on most robotic manipulators. This leaves the momentum observer for the external joint torque estimation method as this method only relies on joint position measurements, joint velocity estimation, and applied joint torque calculation, which are commonly available on robotic manipulators. The derivation and equations of the momentum observer are explained in detail in Section 2.5.

However, the momentum observer is insufficient for detecting a set of impacts where contact with the environment is maintained after the first impact. Consider Figure 3.4, where at t_1 the absolute residual $|\mathbf{r}|$ obtained from the momentum observer surpasses $\mathbf{r}_b$ as a consequence of an impact. Assuming that contact is maintained between impacts, $|\mathbf{r}|$ will remain above $\mathbf{r}_b$. When a second impact occurs at t_2, a sudden change in the residual occurs. However, when estimating the residual, it is not guaranteed that the original residual bound is surpassed again, preventing the second impact from being detected.

Before constructing a method that uses the momentum observer to detect multiple impacts, some specifics of the momentum observer need to be discussed first. When considering the dynamics of the momentum observer found in (2.90), it can be concluded that the momentum observer is a first-order filter that is applied to the impulsive difference between the external joint torque and the residual. When estimating the residual, this impulsive difference between the external joint torque and the residual is observed as a sudden change in $\mathbf{r}$, which converges to the true $\boldsymbol{\tau}_{ext}$. This is also observed in [18], which compares the estimation of the direct method, the observer-extended direct method, and the momentum observer. This sudden increase in the residual can be used to detect multiple impacts, as this behavior is observed regardless of if contact has already been established or not.

Ideally, the rate of change presented in (2.90) should be monitored to detect an impact irrespective of the current contact state of the robotic system. However, considering that the true value of τ_{ext} is unknown, (2.90) cannot be used directly to determine the rate of change in the residual. Therefore, to approximate

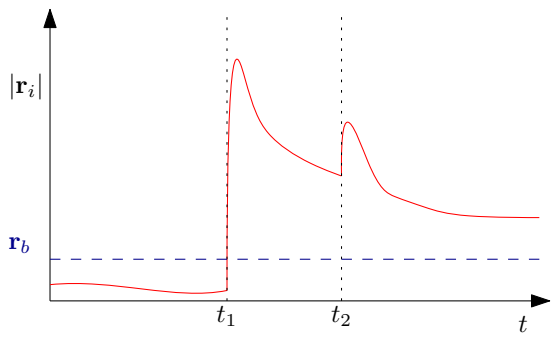


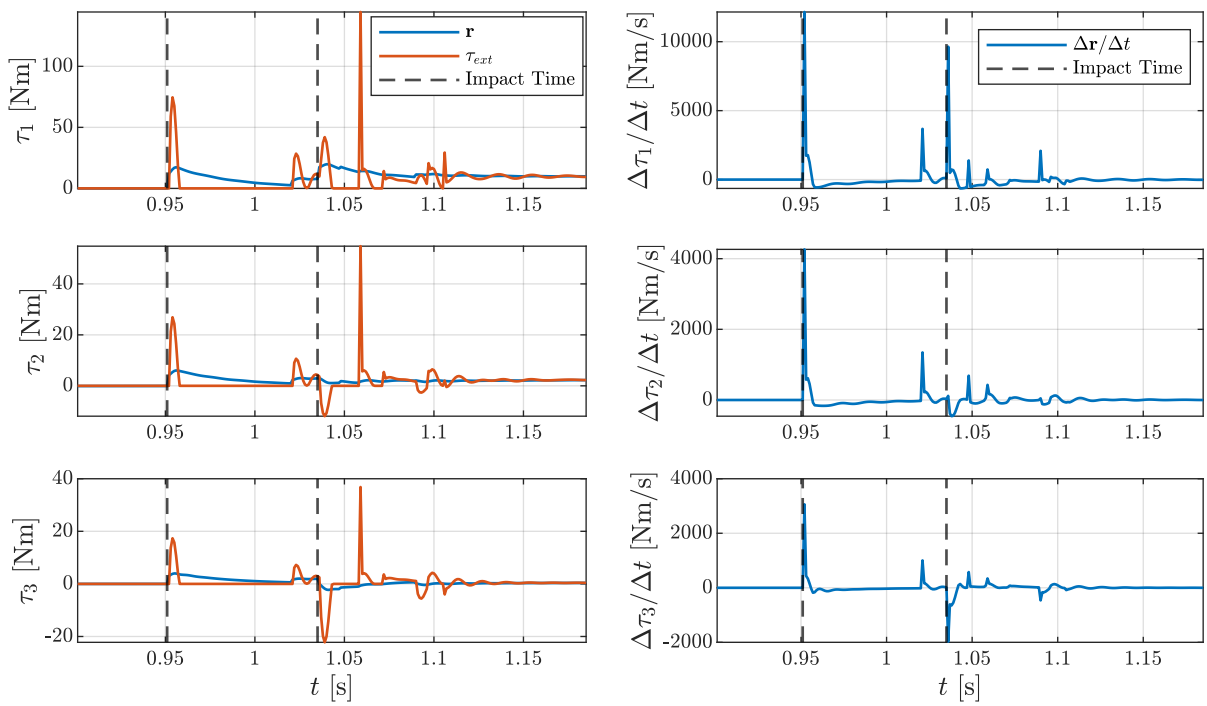
Figure 3.4: Schematic illustration of the residual when the robotic manipulator is undergoing multiple impacts while maintaining contact between them.

the rate of change, numerical differentiation is performed on the residual obtained from the momentum observer. This is expressed as

$$\frac{\Delta \mathbf{r}_k}{\Delta t} = \frac{\mathbf{r}_k - \mathbf{r}_{k-1}}{t_k - t_{k-1}}, \tag{3.4}$$

for each timestep k. In a simulated noiseless environment, this method would provide a suitable method for impact detection. Consider Figure 3.5, where the momentum observer and (3.4) are applied to the simulation. In this simulation, to simulate a non-ideal simultaneous impact, the environment is rotated by $\beta = 5\ deg$, and the elevation of the environment is changed to be $l = -\tan(\beta)^A\mathbf{x}_A^\top{}^A\mathbf{p}_{imp}$ to ensure that the impact location remains the same. Besides these changes in the environment parameters, the remaining parameters are chosen to be as shown in Tables 3.1 and 3.2. In (a), the residual is compared to the true external joint torque. In (b), the time derivative of the residual estimated via (3.4) is shown.

From Figure 3.5, some interesting behavior can be observed. During the intermediate phase, which is the interval between the two impact times, the external joint torque returns to $\mathbf{0}_{n\times 1}$, and after contact along the edge is established smaller jumps in external joint torque can be observed. The reason why the external joint torque returns to $\mathbf{0}_{n\times 1}$ in the intermediate phase can be explained by a property of model A, namely that contact release is still possible. While the control strategy for the intermediate phase is designed to complete contact along the edge of the end-effector, it has not been designed to maintain the contact that has already been established due to the uncertainty of what points have made contact with the environment. This leads to the behavior where after an impact has occurred, a small bounce occurs where contact is briefly lost but then reestablished. This also explains the small peaks in the estimated external joint torque after contact has been completed. This behavior can also be seen in Figure 3.6, which show the gap function for each contact point for the scenario shown in Figure 3.5 as defined in (2.27) and represents the distance until contact has been made. Furthermore, the fact that the true external joint torque returns to $\mathbf{0}_{n\times 1}$ as shown in Figure 3.5 does not guarantee that the residual was able to converge to $\mathbf{0}_{n\times 1}$ before the second impact occurs. Therefore, the behavior presented in Figure 3.4 still occurs and the multi-impact detection method is still necessary.



(a) (b)

Figure 3.5: Momentum observer results obtained from a noiseless simulation. In (a), a comparison is shown between the estimated residual $\mathbf{r}$ and the true external joint torque $\boldsymbol{\tau}_{ext}$. In (b), the time derivative of the residual is shown as calculated in (3.4).

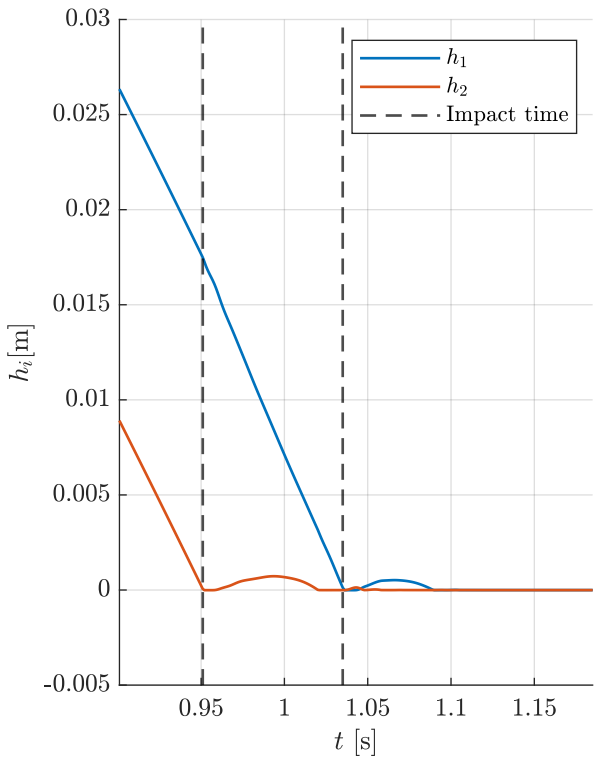


Figure 3.6: The distance from each contact point to the environment for an edge end-effector for the scenario shown in Figure 3.5, showing the bounce that occurs after an impact occurs.

From Figure 3.5, it is clear that multi-impact detection through numerical differentiation should be possible. However, due to the uncertainty in the magnitude of the response, no isolation of the impact link can be guaranteed with this method. This is not a problem as only the detection of multiple impacts is necessary for the current case. It should be noted that while this behavior can be observed in a noiseless simulated environment, it is not guaranteed to work when implemented on a real system with quantization noise and estimation errors. To test if this method of impact detection is possible on a real system, a simple measurement was performed on the Franka Emika Panda to obtain an approximation of the noise. Four sets of information are relevant for this, the joint displacement and velocity, and the motor position and velocity. This data is presented in Figure 3.7. Using Figure 3.7, the noise can be approximated as additive white Gaussian noise. This noise is applied to each relevant parameter and is expressed as

$$\begin{aligned}
\mathbf{q}_{noise,k} &= \mathbf{q}_k + X_{\mathbf{q},k}, \quad \text{with } X_{\mathbf{q}} \in [-7.5\times10^{-4}, 7.5\times10^{-4}]^n, \\
\dot{\mathbf{q}}_{noise,k} &= \dot{\mathbf{q}}_k + X_{\dot{\mathbf{q}},k}, \quad \text{with } X_{\dot{\mathbf{q}}} \in [-5\times10^{-3}, 5\times10^{-3}]^n, \\
\boldsymbol{\theta}_{noise,k} &= \boldsymbol{\theta}_k + X_{\boldsymbol{\theta},k}, \quad \text{with } X_{\boldsymbol{\theta}} \in [-1.5\times10^{-5}, 1.5\times10^{-5}]^n, \\
\dot{\boldsymbol{\theta}}_{noise,k} &= \dot{\boldsymbol{\theta}}_k + X_{\dot{\boldsymbol{\theta}},k}, \quad \text{with } X_{\dot{\boldsymbol{\theta}}} \in [-2\times10^{-2}, 2\times10^{-2}]^n,
\end{aligned} \tag{3.5}$$

where $\mathbf{q}_{noise,k} \in \mathbb{R}^n$, $\dot{\mathbf{q}}_{noise,k} \in \mathbb{R}^n$, $\boldsymbol{\theta}_{noise,k} \in \mathbb{R}^n$, and $\dot{\boldsymbol{\theta}}_{noise,k} \in \mathbb{R}^n$ are the joint displacements and velocities and motor positions and velocities with noise at timestep k, respectively. Furthermore, $X_{\mathbf{q},k} \in \mathbb{R}^n$, $X_{\dot{\mathbf{q}},k} \in \mathbb{R}^n$, $X_{\boldsymbol{\theta},k} \in \mathbb{R}^n$, and $X_{\dot{\boldsymbol{\theta}},k} \in \mathbb{R}^n$ are the randomly sampled noises for their respective parameter at timestep k. In Figure 3.8, an estimate of the external joint torque obtained from the momentum observer where the additive white Gaussian noise is added to the input signals as defined in (3.4).

As can be seen in Figure 3.8, the proposed method for multi-impact detection should work even with a noisy signal. However, due to the noise, it is expected that small impact velocities will be significantly more difficult to detect, as the noise might be larger than the effect an impact might have. For this reason, another simulation is performed with identical trajectory parameters as used to obtain Figure 3.8, but with $v_{imp} = 0.01\ m/s$ used for the impact velocity. Consider Figure 3.9, where the residual and time derivative of the residual for a low-velocity impact are shown.

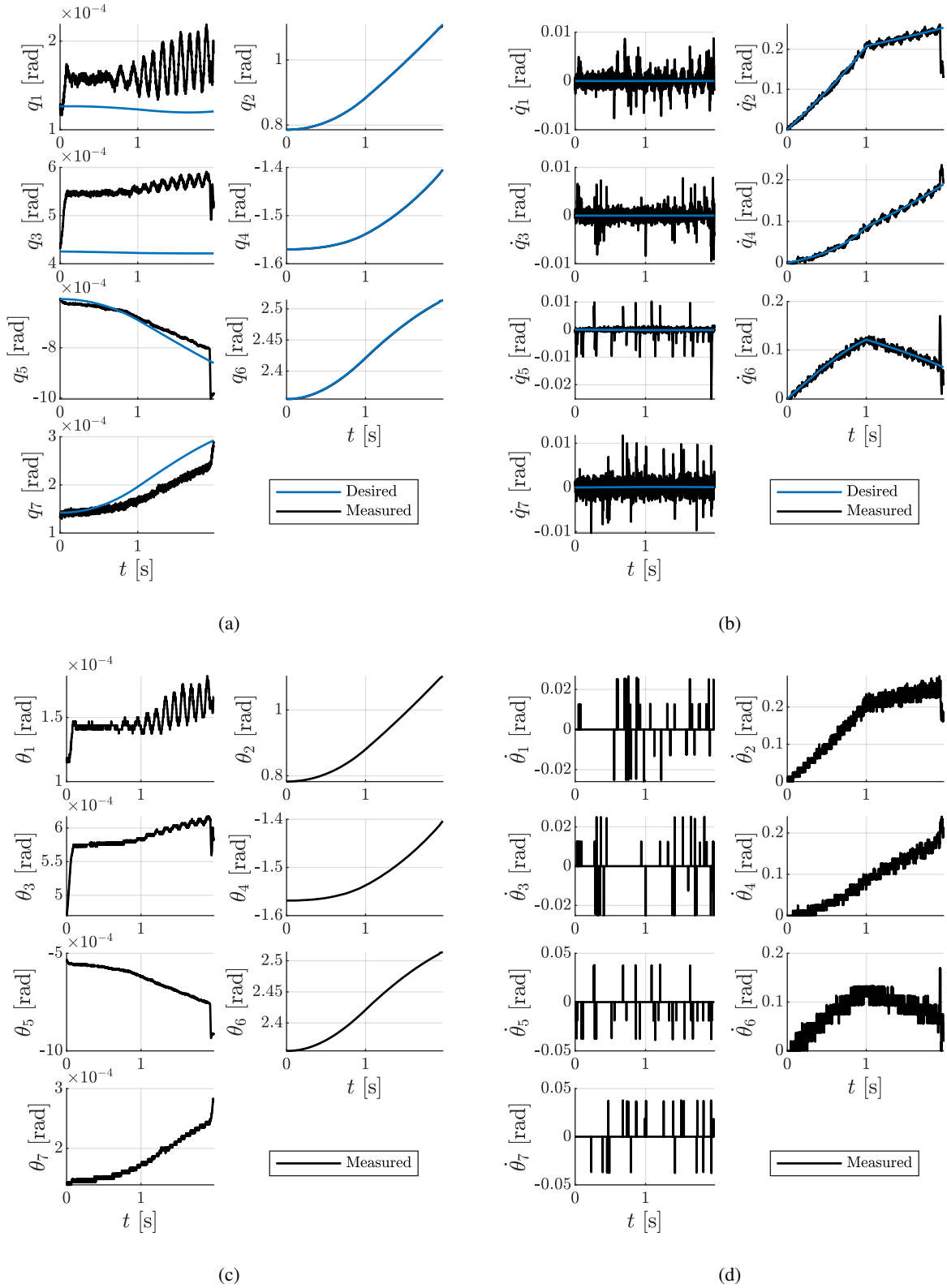
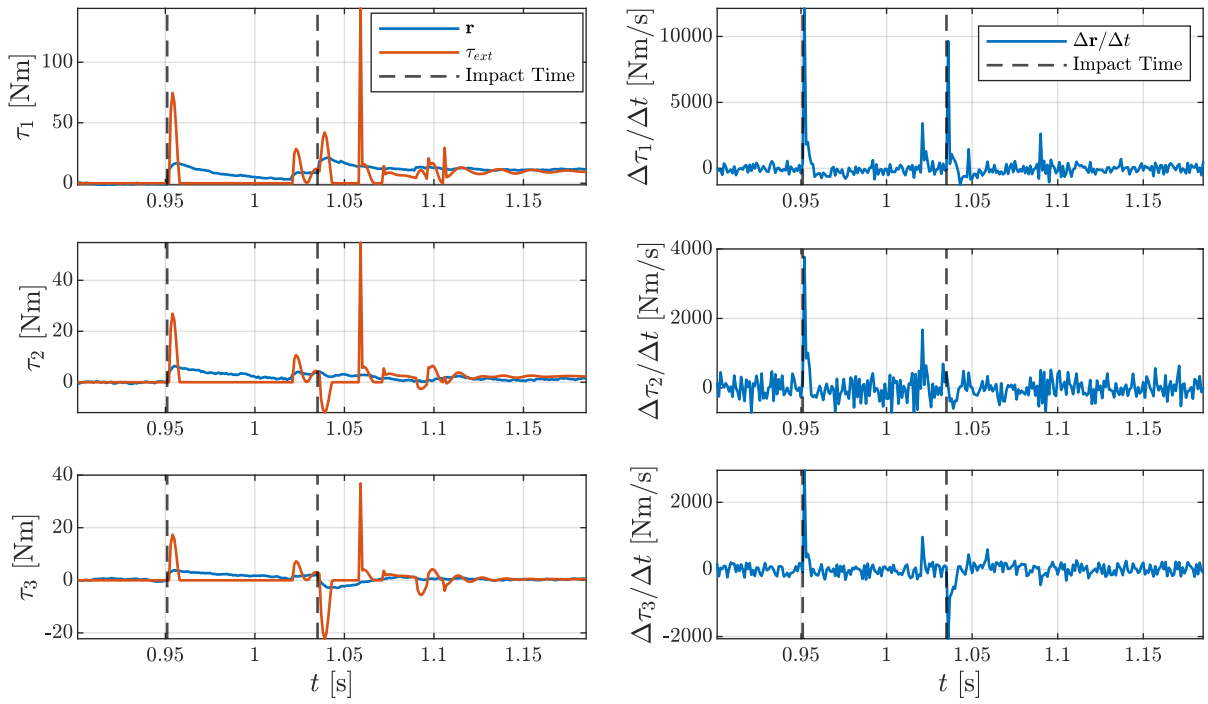
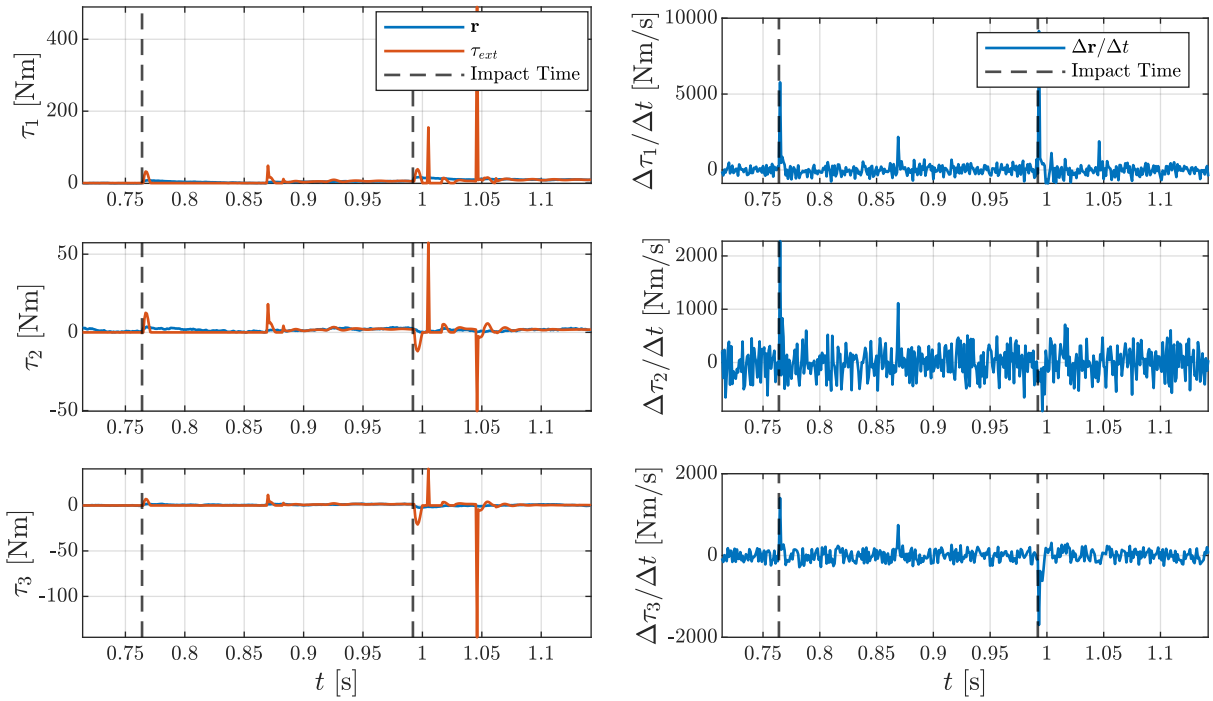


Figure 3.7: Measurement results obtained from an impact experiment performed on the Franka Emika Panda. In (a), the desired and measured joint displacements. In (b), the desired and estimated joint velocities. In (c), the measured motor velocities. In (d), the measured motor velocities.



(a) (b)

Figure 3.8: Momentum observer results obtained from a simulation with additive white Gaussian noise defined in (3.5). In (a), a comparison is shown between the estimated residual $\mathbf{r}$ and the true external joint torque $\boldsymbol{\tau}_{ext}$. In (b), the time derivative of the residual is shown as calculated in (3.4).



(a) (b)

Figure 3.9: Momentum observer results obtained from a simulation with $v_{imp} = 0.01\ m/s$ and the additive white Gaussian noise defined in (3.5). In (a), a comparison is shown between the estimated residual $\mathbf{r}$ and the true external joint torque $\boldsymbol{\tau}_{ext}$. In (b), the time derivative of the residual is shown as calculated in (3.4).

As can be seen in Figure 3.9, there is still a noticeable peak in the time derivative of the estimated residual. Therefore, this shows that assuming that this noise is representative of the noise present on a real system, then this method allows for the detection of multi-impacts if contact is maintained after the first impact. However, it is not guaranteed to be able to isolate the link where the second impact occurs. It should also be noted that a uniform bound is not possible as the magnitude of the residual differs depending on how many joints are between the joint and the impact location.

These tests were all done using numerical simulations, therefore, proving that the multi-impact detection method works in these simulations does not guarantee that it will work when implemented on a real system, regardless of the accuracy of the additive white Gaussian noise. To provide additional evidence that the multi-impact detection method will work on a real system, real impact experiments are investigated to check if the impact is observable. The relevant sets of data that will need to be checked are the joint displacements $\mathbf{q}$, the joint velocities $\dot{\mathbf{q}}$, the spring torque $\boldsymbol{\tau}_J$ that originate from the flexible joints defined as $\boldsymbol{\tau}_J = \mathbf{K}(\boldsymbol{\theta} - \mathbf{q})$, and the time derivative of the spring torque $\dot{\boldsymbol{\tau}}_J$. The data was obtained from impact measurements performed on the Franka Emika Panda by Wouter Weekers in [6], and the plotted data can be found in Figure 3.10.

In Figure 3.10, the impact time denoted by the dashed black line is determined as part of the post-processing. As can be seen in the figures, there is a clear jump in the joint velocities, spring torques, and the time derivative of the spring torque. This is expected behavior when an impact occurs. Furthermore, by only taking the parameters for the even-numbered joints, and using this together with the derived realistic planar dynamics detailed in [6], a rough external joint torque estimation can be performed through a momentum observer[5]. In Figure 3.11, the estimation of the residual was performed along with the numeric differentiation found in (3.4).

As can be seen in Figure 3.11, the external joint torque that is estimated by the momentum observer can be deemed inaccurate. This is because during the ante-impact phase, the external joint torque is not approximately $\mathbf{0}_{n\times 1}$. This is explained by the fact that the dynamics from the uneven joints of the robotic manipulator are discarded to allow for using an available planar model for the momentum observer, whereas the effects from these degrees of freedom are either discarded or spread out over the remaining degrees of freedom. However, while the external joint torque estimation can be deemed inaccurate, the effect of the impact can be seen in the numerically differentiated estimated external joint torque. Additionally, it can be observed that the numerically differentiated estimated external joint torque peaks before the impact time that was determined through post-processing. However, regardless of whether the impact time determined through post-processing or the proposed multi-impact detection method is slightly inaccurate, the two methods still show similar results. Therefore, the results in Figure 3.11 show that numerically differentiating the estimated residual could be a valid method for the detection of multiple impacts regardless of the current contact state. For this reason, this impact detection method is used throughout the rest of the project on simulated responses.

To finalize the multi-impact detection algorithm, a threshold can be applied to the time derivative of $\mathbf{r}$ to detect an impact. This is reminiscent of the standard momentum observer impact detection method described in [15], where a threshold is applied to $\mathbf{r}$ directly. Furthermore, an additional condition is included to ensure that if the rate of change is lower than the previous timestep but still surpasses the designated bound for longer than one timestep, it assumes that it is part of the same impact. The impact detection condition can then be formulated as follows:

The time instant $t_k = k\Delta t$ is considered an impact time if and only if there exists an $i \in [1, n]$ such that

$$\left|\frac{\Delta \mathbf{r}_{k,i}}{\Delta t}\right| \geq \mathbf{r}_{b,i}, \tag{3.6}$$

and

$$\left|\frac{\Delta \mathbf{r}_{k,i}}{\Delta t}\right| \geq \left|\frac{\Delta \mathbf{r}_{k-1,i}}{\Delta t}\right|. \tag{3.7}$$

[5]This estimation does not accurately estimate the external joint torques, as only three out of seven degrees of freedom are considered. The effects from the ignored degrees of freedom are either discarded or spread out over the remaining three degrees of freedom.

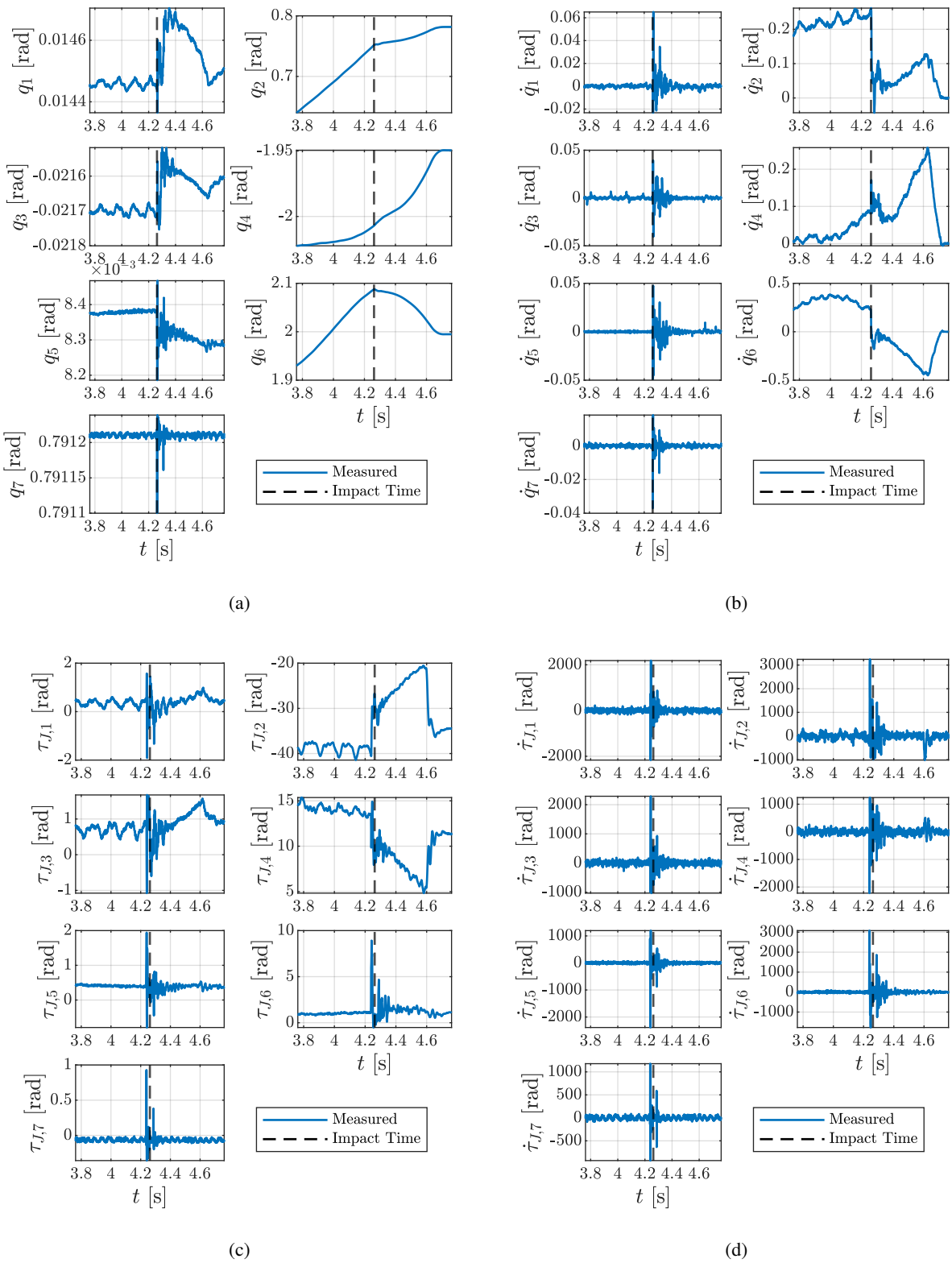
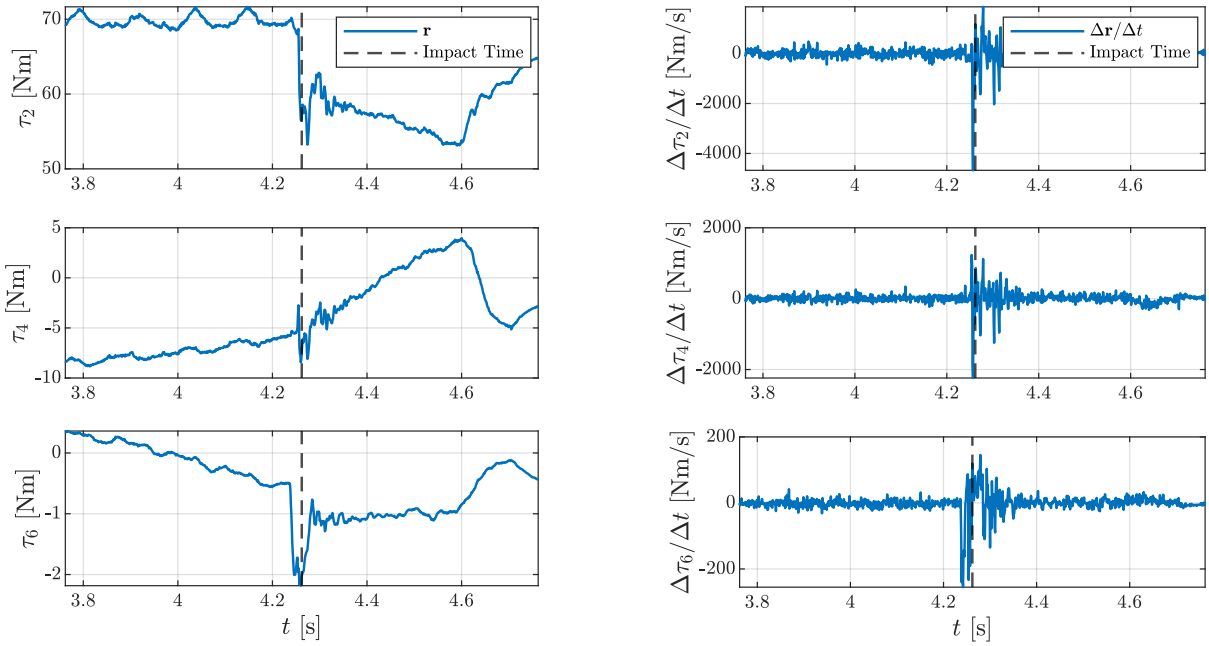


Figure 3.10: Measurements obtained from impact experiment performed on the Franka Emika Panda in [6]. In (a), the link side joint displacements. In (b), the link side joint velocities. In (c), the spring torques. In (d), the time derivative of the spring torques.



(a) (b)

Figure 3.11: Momentum observer estimation results using a planar momentum observer and measurements from an impact experiment performed in [6]. In (a), the estimated residual is found. In (b), the time derivative of the residual is found.

Here, $\mathbf{r}_b = [r_{b,1}, \ldots, r_{b,n}]^\top \in \mathbb{R}^n$ represents the user-defined bound for when an impact is detected. where $r_{b,i}$ is the external joint torque rate of change bound for joint i. After some testing, the impact detection bound is chosen to be $\mathbf{r}_b = [4500, 2500, 1450]^\top$ as this provided a suitable sensitivity to detect multiple impacts while ignoring the noise.

3.3 Adjustments to the Simulation

The classifier that will be discussed in Chapter 4 is designed based on simulated results. To ensure that this classifier also functions on a real system, several adjustments are made to improve the realism of the simulation and to prepare it for simultaneous impact simulations. These adjustments include an impact detection delay, the impact control strategy, and adjustments to the control tasks for the simultaneous impact case. These will be discussed in Section 3.3.1, 3.3.2, and 3.3.3, respectively.

3.3.1 Impact Detection Time Delay

When implementing an impact detection algorithm in simulations, the impact can be promptly detected in the timestep after the impact has occurred. However, in real systems, a larger delay is to be expected due to the delay of processing the measurements and drawing the conclusion that the measured response is indicative of an impact. Depending on the impact detection method, this delay can vary in length. For momentum observers, the typical delay can range from $6 - 281$ ms depending on observer gains and placement of the detection threshold [18].

These delays relate to the rate of convergence of $\mathbf{r}$ to the external joint torque, where lower rates of convergence lead to a larger interval between when the impact occurs and when it is detected. Furthermore, as was discussed in [18], increasing $\mathbf{K}_O$ introduces noise and decreases the accuracy of the external joint torque estimation, but it will also decrease the impact detection delay due to the rate of convergence increasing. Normally, a compromise in the size of $\mathbf{K}_O$ needs to be made between observer estimation accuracy and the detection delay. However, when considering the impact detection method proposed in

Section 3.2, only the time derivative of $\mathbf{r}$ is of importance. This allows for tuning the observer gain matrix $\mathbf{K}_O$ larger than for cases where an accurate estimation of the external joint torque is required.

As stated in Section 2.5, $\mathbf{K}_O = 30\mathbf{I}_n$ is used for the observer gain for the momentum observer. Furthermore, in [18] the lowest impact detection delay associated with the momentum observer is 6 ms with an observer gain matrix of $\mathbf{K}_O = 10\mathbf{I}_n$. Considering the differences between the designed impact detection method and the higher observer gain matrix, the impact detection delay was chosen to be $t_{delay} = 0.002$ s.

3.3.2 Control Strategy

As was discussed in Section 2.2.4, the simulation designed in [6] is controlled using a task-based QP robot controller. Due to the availability of the QP controller for the simulation, this controller will also be used for the simulations in this project. As was discussed in Section 2.4, when controlling a system that is expected to undergo state triggered jumps, a control strategy needs to be used that allows for increased tracking performance for differences between the expected and detected jump time. This leads to the inclusion of an ante- and post-impact control strategy for the single- and simultaneous impact case. Furthermore, for the simultaneous impact case, the intermediate phase also requires a control strategy.

The robotic manipulator for this system is controlled by a task-based QP controller. Therefore, when implementing a control strategy based on reference spreading, a different approach needs to be taken than for a state feedback control law. This approach is detailed in Section 2.4.2, which discusses the implementation of reference spreading on a task-based QP controller for single- and simultaneous impacts. This leads to an important decision concerning the task reference trajectories. Namely, for the control strategy for the intermediate phase of a simultaneous impact case, that the pose of the end-effector should be controlled through joint-space configuration tasks. Therefore, when implementing reference spreading, all pose tasks are formulated as joint-space configuration tasks, which is done for both the single- and simultaneous impact case. The base reference spreading control strategy that will be used for this case is formulated in Section 2.4.2. This strategy concerns what task reference trajectory to follow for each impact phase, and for the single impact case is formulated as

$$\overline{\boldsymbol{\gamma}}(t) = \begin{cases} \overline{\boldsymbol{\gamma}}_a(t), & t \in [t_0, t_1], \\ \overline{\boldsymbol{\gamma}}_p(t), & t \in (t_1, t_f]. \end{cases} \tag{3.8}$$

For the simultaneous impact case it is formulated as

$$\overline{\boldsymbol{\gamma}}(t) = \begin{cases} \overline{\boldsymbol{\gamma}}_a(t), & t \in [t_0, t_1], \\ \overline{\boldsymbol{\gamma}}_a(t), & t \in (t_1, t_N], \\ \overline{\boldsymbol{\gamma}}_p(t), & t \in (t_N, t_f], \end{cases} \tag{3.9}$$

with the derivative feedback gain strategy defined as

$$\begin{aligned} &\text{if } t \in [t_0, t_1], && \mathbf{K}_d \neq 0, \\ &\text{if } t \in (t_1, t_N], && \mathbf{K}_d = 0, \\ &\text{if } t \in (t_N, t_f], && \mathbf{K}_d \neq 0, \end{aligned} \tag{3.10}$$

where t_0 is the starting time, t_1 is the time when the first impact is detected, t_N is the time when the final impact is detected, and t_f is the final operation time. For implementation purposes, a unifying control strategy needs to be designed that uses conditions to switch between strategies that are applicable to a real system. This is done to ensure that the switching behavior can be replicated on a experimental setup to validate the results. Furthermore, this unifying control strategy should be a general formulation that allows for the implementation of reference spreading on a QP-controlled robotic manipulator, regardless of the impact case.

To unify these control strategies, a new parameter called $n_{imp} \in \mathbb{N}$, also known as the expected maximum number of impacts, is introduced. This parameter can be defined to be $n_{imp} = 1$ for the single impact case, and $n_{imp} = 2$ for the two-dimensional simultaneous impact case and signifies the maximum number of impacts that are expected until contact between the end-effector and the environment has been completed.

This is then combined with the detected number of impacts j, where if $j = 0$ no impact has taken place, if $j \neq 0$ and $j < n_{imp}$ then an impact has occurred but contact has not been completed, and if $j \geq n_{imp}$ then contact has been completed.

It should be noted, that when using a switching strategy that relies on counting the number of detected impacts, that there is no guarantee that contact has been completed when $j \geq n_{imp}$. For example, the first impact followed by loss of contact followed by regaining contact can result in a second impact detection that would result in $j = 2$. For the simultaneous impact case, this would result in switching to the post-impact strategy before contact has been completed. Therefore, it is recommended that these conditions are replaced by a method that allows for detecting whether contact has been completed or not, the method of which could be a subject for follow-up research.

Furthermore, an additional condition needs to be included in the case that an ideal simultaneous impact occurs. In such a case, the expected maximum number of impacts is $n_{imp} = 2$, however, for an ideal simultaneous impact only a single impact is detected, i.e., $j = 1$. By adding the additional condition that the intermediate phase strategy can only remain active for t_{switch} s, then an ideal simultaneous impact will switch to the post-impact phase when $t - t_1 \geq t_{switch}$. All of these new conditions can be compiled in the following task reference trajectory switching strategy

$$\overline{\gamma}(t) = \begin{cases} \overline{\gamma}_a(t), & \text{if } j = 0, \\ \overline{\gamma}_a(t), & \text{if } j \neq 0 \text{ and } j < n_{imp} \text{ and } t - t_1 < t_{switch}, \\ \overline{\gamma}_p(t), & \text{if } j \geq n_{imp} \text{ or } t - t_1 \geq t_{switch}, \end{cases} \tag{3.11}$$

with the derivative feedback gain switching strategy defined as

$$\begin{aligned} &\text{if } j = 0, && \mathbf{K}_d \neq 0, \\ &\text{if } j \neq 0 \text{ and } j < n_{imp} \text{ and } t - t_1 < t_{switch}, && \mathbf{K}_d = 0, \\ &\text{if } j \geq n_{imp} \text{ or } t - t_1 \geq t_{switch}, && \mathbf{K}_d \neq 0. \end{aligned} \tag{3.12}$$

This unifying control strategy is reminiscent of the (3.8) for a single impact case, i.e., when $n_{imp} = 1$. Likewise, this control strategy is also reminiscent of (3.9) and (3.10) for a simultaneous impact case, i.e., when $n_{imp} > 1$.

3.3.3 Task Adjustment for the Simultaneous Impact Case

The only task adjustment required for the simultaneous impact case concerns the contact task. To ensure that the entire edge maintains contact with the environment for a planar simultaneous impact case, multiple contact tasks are implemented into the optimization problem, one for each contact point of the end-effector. Furthermore, the contact task reference is adjusted to account for multiple contact points, where the desired contact force for each contact point is the total desired contact force divided by the total number of contact points.

3.4 Defining Impact Task

As was discussed in Section 3.1, using user-defined trajectory parameters and the expected environment parameters defined in Tables 3.1 and 3.2, a trajectory can be generated for the robotic manipulator to follow. Furthermore, this trajectory is generated using a prediction of the post-impact velocity to increase the tracking performance after the impact. In Section 3.3.2, a control strategy was discussed that uses reference spreading to reduce the peaking phenomenon when the detected and expected impact time are different, which further improves tracking performance. Therefore, when defining the task trajectory of the robotic manipulator, two trajectories need to be generated, the ante- and post-impact trajectory. For the reference spreading method discussed in Section 3.3.2, joint-space tasks are desired. However, when generating the trajectories in joint-space, the motion of the end-effector is less intuitive and it becomes more difficult to take into account the expected location of the environment. This problem does not exist when designing the trajectories in task-space. Therefore, for this section, the trajectories will be generated in task-space first and will be converted to joint-space in Section 3.4.1. The general structure of the task-space trajectories is defined as

$$\boldsymbol{\gamma}(t) = \begin{bmatrix} {}^{A}\mathbf{p}_{ref}(t) \\ {}^{A}\dot{\mathbf{p}}_{ref}(t) \\ {}^{A}\ddot{\mathbf{p}}_{ref}(t) \\ \kappa_{ref}(t) \end{bmatrix}, \tag{3.13}$$

where ${}^{A}\mathbf{p}_{ref}(t) \in \mathbb{R}^6$, ${}^{A}\dot{\mathbf{p}}_{ref}(t) \in \mathbb{R}^6$, and ${}^{A}\ddot{\mathbf{p}}_{ref}(t) \in \mathbb{R}^6$ are the desired end-effector position and orientation, twist, and acceleration, respectively, and $\kappa_{ref}(t) \in \mathbb{R}^{n_c}$ are the desired normal contact forces for each contact point. Splitting these into the ante- and post-impact trajectories results in

$$\boldsymbol{\gamma}_a(t) = \begin{bmatrix} {}^{A}\mathbf{p}_{a,ref}(t) \\ {}^{A}\dot{\mathbf{p}}_{a,ref}(t) \\ {}^{A}\ddot{\mathbf{p}}_{a,ref}(t) \\ \mathbf{0}_{n_c\times 1} \end{bmatrix}, \qquad \boldsymbol{\gamma}_p(t) = \begin{bmatrix} {}^{A}\mathbf{p}_{p,ref}(t) \\ {}^{A}\dot{\mathbf{p}}_{p,ref}(t) \\ {}^{A}\ddot{\mathbf{p}}_{p,ref}(t) \\ \frac{\lambda_{ref}}{n_c}\mathbf{1}_{n_c\times 1} \end{bmatrix}, \tag{3.14}$$

where the subscript $(\cdot)_a$ denotes quantities associated with the ante-impact trajectory, while the subscript $(\cdot)_p$ denotes quantities associated with the post-impact trajectory. This should not be confused with the superscripts $(\cdot)^-$ and $(\cdot)^+$, as these denote ante- and post-impact quantities at the time of impact.

With the general form of the ante- and post-impact task trajectories defined, the trajectories can be constructed. The ante-impact task trajectory is constructed so that it moves the end-effector from its initial position and orientation ${}^{A}\mathbf{p}_{a,0} = FK(\mathbf{q}_0)$ to the desired impact location and orientation ${}^{A}\mathbf{p}_{imp}$ with desired ante-impact twist ${}^{L[A]}\mathbf{v}_{A,L}^{imp}$, these parameters can also be found in Section 3.1.1. Furthermore, the post-impact trajectory is designed to move the end-effector from its desired impact position and orientation ${}^{A}\mathbf{p}_{imp}$ to a final position and orientation defined as

$${}^{A}\mathbf{p}_{p,f} = {}^{A}\mathbf{p}_{imp} + \begin{bmatrix} sgn\left(\left({}^{L[A]}\boldsymbol{v}_{A,L}^{imp}\right)_x\right)\delta x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.15}$$

The final end-effector position and orientation defined in (3.15) also ensures that the end-effector is realigned to the desired end-effector orientation because of changes in joint velocity that might occur as a consequence of an impact. To improve the tracking performance when switching between the ante- and post-impact task trajectories, the post-impact end-effector twist ${}^{A}\dot{\mathbf{p}}_{p,ref}(t_{1,d})$ is predicted by applying the rigid impact map derived in Section 2.2.3 to the desired ante-impact end-effector twist ${}^{A}\dot{\mathbf{p}}_{a,ref}(t_{1,d})$. This prediction can be represented as

$${}^{A}\dot{\mathbf{p}}_{p,ref}(t_{1,d}) = {}^{L[A]}\mathbf{J}_{A,L}(IK({}^{A}\mathbf{p}_{imp}))\dot{\mathbf{q}}^+, \tag{3.16}$$

with

$$\dot{\mathbf{q}}^+ = \Phi(\dot{\mathbf{q}}^-), \tag{3.17}$$

and

$$\dot{\mathbf{q}}^- = \left({}^{L[A]}\mathbf{J}_{A,L}(IK({}^{A}\mathbf{p}_{imp}))\right)^{-1}\,{}^{A}\dot{\mathbf{p}}_{a,ref}(t_{1,d}). \tag{3.18}$$

In (3.16), (3.17), and (3.18), $\mathbf{q} = IK({}^{A}\mathbf{p})$ represents the inverse kinematics that calculates the joint positions from the end-effector position and orientation, while $\dot{\mathbf{q}}^+ = \Phi(\dot{\mathbf{q}}^-)$ is a simplified notation of the impact map derived in Section 2.2.3.

Furthermore, when applying the rigid impact map derived in Section 2.2.3 to the desired ante-impact end-effector twist, the expected number of contact points should be taken into account. This can be done by adding the Jacobians for each expected contact point to the impact map as is described in Section 2.2.3. For a single impact case, when an impact occurs, the single contact point allows for a change in the task-space velocity as well as the angular velocity of the end-effector because the end-effector can pivot around the contact point. However, for a simultaneous impact case, two points on the end-effector make contact with the environment simultaneously. With two contact points, the end-effector can no longer pivot around the contact points. This results in only the task-space velocity changing, but the angular velocity remaining constant.

As was stated in Section 3.3.2, due to perturbations or incorrect information about the environment an impact can occur at a time $t_1 \neq t_{1,d}$ and poor tracking performance is observed due to peaking. The peaking problem can be solved by extending the ante- and post-impact trajectories past $t_{1,d}$. Furthermore, to further decrease the effects of peaking the trajectories should match as much as possible. For the ante-impact trajectory, this results in the final end-effector position and orientation that acts as the ante-impact counterpart of ${}^{A}\mathbf{p}_{p,f}$, this ante-impact final position and orientation is defined as

$$ {}^{A}\mathbf{p}_{a,f} = {}^{A}\mathbf{p}_{imp} + \begin{bmatrix} sgn\left(\left({}^{L[A]}\boldsymbol{v}_{A,L}^{imp}\right)_x\right)\delta x \\ sgn\left(\left({}^{L[A]}\boldsymbol{v}_{A,L}^{imp}\right)_y\right)\delta y \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.19} $$

For the post-impact trajectory, a new starting position and orientation is defined, which acts as the post-impact counterpart of ${}^{A}\mathbf{p}_{a,0}$. This post-impact starting position and orientation can be defined as

$$ {}^{A}\mathbf{p}_{p,0} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} {}^{A}\mathbf{p}_{a,0}, \tag{3.20} $$

As can be seen in (3.15), (3.19), and (3.20), the ante- and post-impact starting and final position and orientation are similar to each other. The only difference that is present is the motion in the $\vec{y}_A$ direction, where the post-impact starting and final positions and orientations are designed to always be in contact with the expected environment. Furthermore, the ante-impact final position is designed to ignore the expected environment and to continue its motion towards a point located at the predefined distance from the desired impact position. The reason for this is twofold, firstly, if the environment is placed further away from the desired impact location than expected, then contact can still be made if it is on the path to the final position. Secondly, if the environment is placed too far away, or is unreachable, then the robotic manipulator will stop moving before it reaches a singular configuration.

With the ante- and post-impact starting and final positions and orientations defined ante- and post-impact trajectories can be defined that satisfy the user-defined trajectory parameters where Table 3.2 presents an example of such a set of parameters. To summarize, for the ante-impact trajectory, the trajectory is designed to start from its initial position ${}^{A}\mathbf{p}_{a,0}$ to the desired impact location ${}^{A}\mathbf{p}_{imp}$ with the desired ante-impact end-effector twist ${}^{L[A]}\mathbf{v}_{A,L}^{imp}$. The trajectory is then designed to continue towards ${}^{A}\mathbf{p}_{a,f}$ where it will come to a stop. For the post-impact trajectory, the trajectory is designed to start at ${}^{A}\mathbf{p}_{p,0}$ and will move towards the desired impact location ${}^{A}\mathbf{p}_{imp}$ with the post-impact velocity calculated in (3.16). Afterwards, the trajectory is designed to continue until it has reached ${}^{A}\mathbf{p}_{p,f}$, where it will stop. The full extended trajectories are then defined as

$$ \overline{\boldsymbol{\gamma}}_a(t) = \begin{bmatrix} {}^{A}\overline{\mathbf{p}}_{a,ref}(t) \\ {}^{A}\dot{\overline{\mathbf{p}}}_{a,ref}(t) \\ {}^{A}\ddot{\overline{\mathbf{p}}}_{a,ref}(t) \\ \mathbf{0}_{n_c\times 1} \end{bmatrix}, \qquad \overline{\boldsymbol{\gamma}}_p(t) = \begin{bmatrix} {}^{A}\overline{\mathbf{p}}_{p,ref}(t) \\ {}^{A}\dot{\overline{\mathbf{p}}}_{p,ref}(t) \\ {}^{A}\ddot{\overline{\mathbf{p}}}_{p,ref}(t) \\ \frac{\lambda_{ref}}{n_c}\mathbf{1}_{n_c\times 1} \end{bmatrix}, \tag{3.21} $$

Using the trajectory parameters defined in Table 3.2, two sets of trajectories are defined. These trajectories are for the single- and simultaneous impact case. In Figure 3.12, the task-space task reference trajectories are shown for the single impact case. Additionally, in Figure 3.13, the task-space task reference trajectories are shown for the simultaneous impact case. For completeness, in Figure 3.14 the planar representation of the path of the end-effector is shown for the single- and simultaneous impact case. These figures show the different components of the task-space task reference trajectories as denoted by (3.14). Furthermore, observe the behavior in Figures 3.12 (b) and 3.13 (b), wherein the angular velocity of the end-effector at the time of impact, the deviation due to the pivoting of the end-effector can be observed. Furthermore, for the simultaneous impact case, this deviation is also observed contrary to the observations that were made earlier. However, the deviation is observed to be of an order of magnitude 10^{-13} lower than for the single impact case, this difference in the order of magnitude shows that the single impact case is more sensitive to the pivoting of the end-effector after an impact.

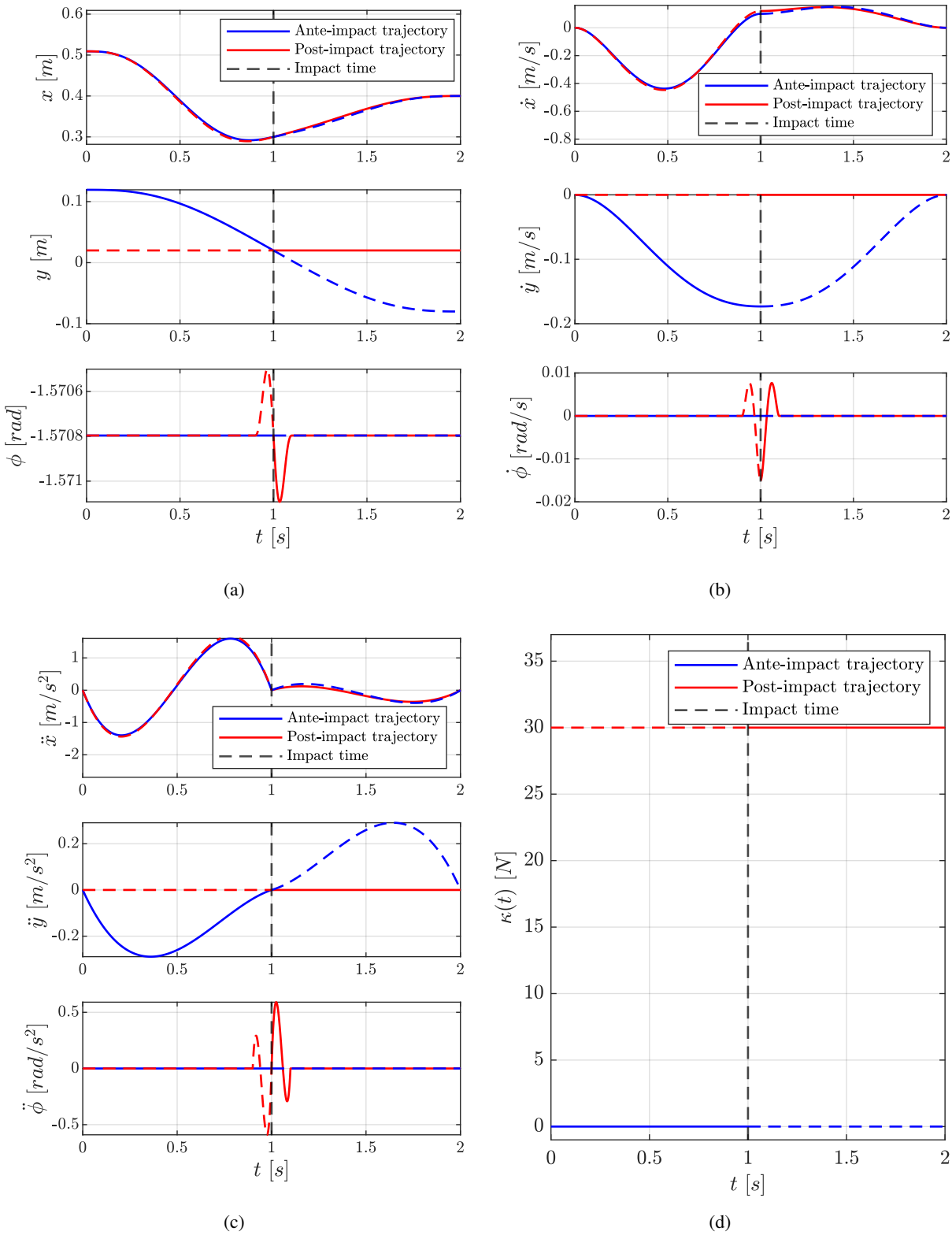


Figure 3.12: Task-space task reference trajectories for the single impact case. In (a), the task-space position task. In (b), the task-space velocity task. In (c), the task-space acceleration task. In (d), the contact task.

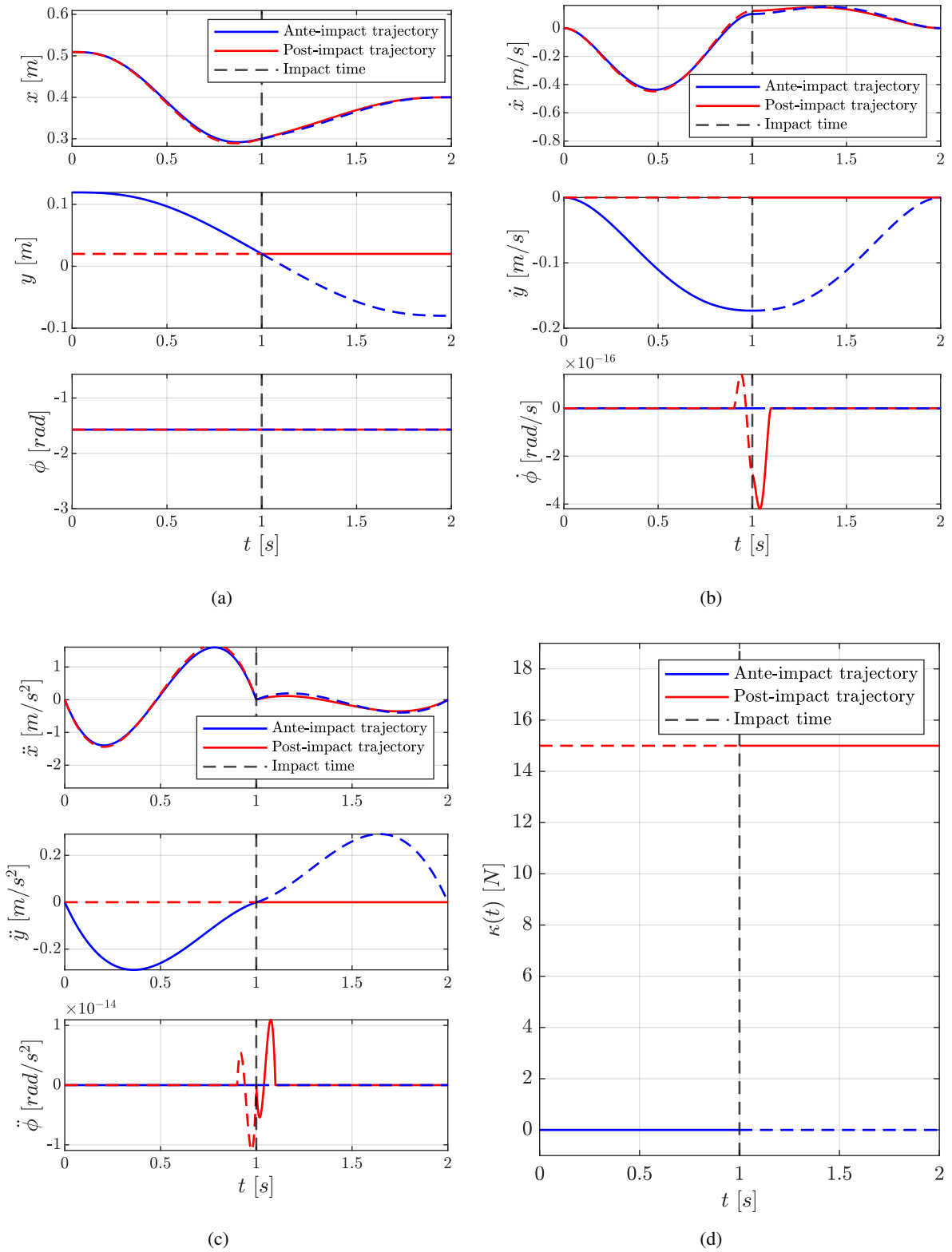


Figure 3.13: Task-space task reference trajectories for the simultaneous impact case. In (a), the task-space position task. In (b), the task-space velocity task. In (c), the task-space acceleration task. In (d), the contact task.

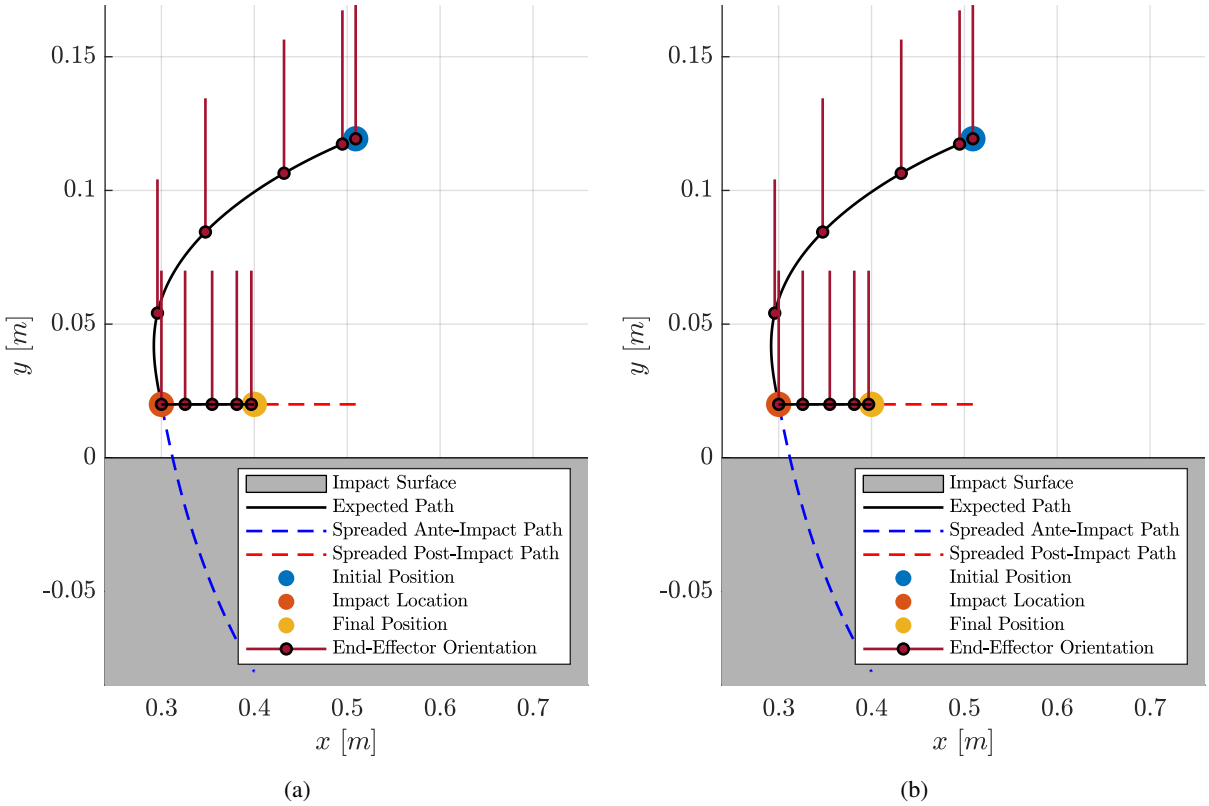


Figure 3.14: Planar representation of the task-space reference path. In (a), for a single impact case. In (b), for a simultaneous impact case. Note, both figures are almost identical due to velocity being the main difference between the single- and simultaneous impact case trajectories, which are not shown in this figure.

This section focused on designing the ante- and post-impact task reference trajectories in task-space. This trajectory can be used to control a QP-controlled robotic manipulator with task-space end-effector pose tasks for single impact cases. However, as stated earlier, to implement reference spreading on a QP-controlled robotic manipulator undergoing simultaneous impacts, a joint-space configuration task is needed instead. Furthermore, joint-space behavior will also be used for the classification performed in Chapter 4. Therefore, in Section 3.4.1, the task-space task reference trajectories are converted to joint-space for control and classification purposes.

3.4.1 Converting to Joint-Space Reference

As stated in Section 3.4 for classification purposes and use as a joint-space configuration task, the ante- and post-impact task-space task reference trajectories are converted to joint-space task reference trajectories. To transform the ante- and post-impact task trajectories into joint-space, inverse kinematics can be applied. The inverse kinematics are derived through the use of the forward kinematics, the forward kinematics on the position level is dependent on the robotic manipulator, and can be represented as the function

$$\begin{bmatrix} {}^{A}\mathbf{o}_{L} \\ {}^{L}\mathbf{R}_{A} \end{bmatrix} = FK(\mathbf{q}), \tag{3.22}$$

By converting ${}^{L}\mathbf{R}_{A}$ to a set of roll-pitch-yaw angles that can be used to obtain this rotation matrix, it is possible to write (3.22) as

$${}^{A}\mathbf{p} = FK(\mathbf{q}), \tag{3.23}$$

where ${}^{A}\mathbf{p} \in \mathbb{R}^{6}$ represents the task-space position and orientation of the end-effector. By taking the time derivative of (3.23), the forward kinematics on velocity level can be obtained. As was discussed in Section 2.1.4, for the mixed velocity, this leads to

$$^{L[A]}\mathbf{v}_{A,L} = {}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\dot{\mathbf{q}}. \tag{3.24}$$

Taking the time derivative of (3.24), the forward kinematics for the mixed acceleration can be defined as

$$^{L[A]}\dot{\mathbf{v}}_{A,L} = {}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\ddot{\mathbf{q}} + {}^{L[A]}\dot{\mathbf{J}}_{A,L}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}, \tag{3.25}$$

Finally, starting with (3.23), the forward kinematics can be inverted to obtain the inverse kinematics. The inverse kinematics on the position level can then be defined as

$$\mathbf{q} = IK({}^{A}\mathbf{p}). \tag{3.26}$$

Furthermore, inverting (3.24) to obtain an equation for $\dot{\mathbf{q}}$ results in

$$\dot{\mathbf{q}} = \left({}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\right)^{-1} {}^{L[A]}\mathbf{v}_{A,L}. \tag{3.27}$$

Finally, by inverting (3.25), the inverse kinematics to obtain $\ddot{\mathbf{q}}$ can be defined as

$$\ddot{\mathbf{q}} = \left({}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})\right)^{-1} \left({}^{L[A]}\dot{\mathbf{v}}_{A,L} - {}^{L[A]}\dot{\mathbf{J}}_{A,L}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right). \tag{3.28}$$

In these sets of equations, $FK(\mathbf{q}) \in \mathbb{R}^6$, $IK({}^{A}\mathbf{p}) \in \mathbb{R}^n$ represents the forward and inverse kinematics functions, respectively, ${}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the end-effector Jacobian relating the joint velocities to the twist of L with respect to A, expressed in mixed frame $L[A]$, $\mathbf{q} \in \mathbb{R}^n$, $\dot{\mathbf{q}} \in \mathbb{R}^n$, and $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are the joint displacements, velocities, and accelerations, respectively.

However, before these inverse kinematics are used, several remarks need to be made. First of all, the inverse of the Jacobian ${}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})$ serves as the crux of using the inverse kinematics. However, ${}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})$ is not guaranteed to be a square matrix, meaning that it is not guaranteed that the matrix can be inverted using standard methods. Furthermore, the Franka Emika Panda is a 7 Degrees of Freedom robotic manipulator, which means that the Jacobian for this robot has the dimensions of 6×7, which is not square. This shows that using this method to determine the inverse kinematics should not be possible. However, the simulation is a 3-DOF planar approximation of the Franka Emika Panda. Combining this fact with the fact that out of plane motion can be ignored when deriving the ${}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})$, it can be reduced to a 3×3 matrix, which is square. Furthermore, inverse kinematics can only be performed when ${}^{L[A]}\mathbf{J}_{A,L}(\mathbf{q})$ is not singular. However, under the assumption that singular configurations are avoided, no further attention is required for this problem.

Finally, when performing the inverse kinematics to obtain the joint configuration on a planar 3-DOF robotic manipulator, if the desired position and orientation of the end-effector is feasible and does not result in a singular configuration, two possible joint configurations are possible. These positions are the elbow-up and elbow-down positions. For a planar robotic manipulator, the elbow-up configuration is defined as

$$q_2 < 0, \tag{3.29}$$

in whereas the elbow-down configuration is defined as

$$q_2 > 0, \tag{3.30}$$

with $q_2 \in \mathbb{R}$ the displacement of the second joint, which is shown in Figure 3.1. Note that $q_2 = 0$ results in a singular configuration, and is excluded from the definitions found in (3.29) and (3.30). The type of configuration at the time of impact is dependent on the initial configuration. This can be explained by observing that the elbow-up and elbow-down configurations are separated by a singular configuration. Therefore, to move from the elbow-up to the elbow-down configuration, the singular configuration needs to be passed first. Considering the assumption that singular configurations are avoided, this implies that the type of configuration that the robotic manipulator starts in will also be the type of configuration that it will remain in. Furthermore, when deriving the rigid impact map in Section 2.2.3, it was assumed that when an impact occurs the ante- and post-impact joint displacements remain constant, i.e., $\mathbf{q}^-(t_{1,d}) = \mathbf{q}^+(t_{1,d})$. Therefore, the type of configuration will also remain the same when the system is undergoing an impact. This leads to the conclusion that the system will remain in the same type of configuration that it started in.

Finally, as discussed earlier, the calculated joint displacements, velocities, and accelerations also serve as the task reference trajectories for the QP controller tasks used for controlling the robotic manipulator. Using the inverse kinematics derived in (3.26), (3.27), and (3.28) combined with the remarks explained above, the task-space task reference trajectories can be converted to joint-space task reference trajectories. To obtain the joint-space ante-impact trajectory, ${}^{A}\mathbf{p}$ is substituted with ${}^{A}\overline{\mathbf{p}}_{a,ref}(t)$, ${}^{L[A]}\mathbf{v}_{A,L}$ is substituted with $\overline{\dot{\mathbf{p}}}_{a,ref}(t)$, and ${}^{L[A]}\dot{\mathbf{v}}_{A,L}$ is substituted with $\overline{\ddot{\mathbf{p}}}_{a,ref}(t)$. Furthermore, to obtain the joint-space post-impact trajectory, ${}^{A}\mathbf{p}$ is substituted with ${}^{A}\overline{\mathbf{p}}_{p,ref}(t)$, ${}^{L[A]}\mathbf{v}_{A,L}$ is substituted with $\overline{\dot{\mathbf{p}}}_{p,ref}(t)$, and ${}^{L[A]}\dot{\mathbf{v}}_{A,L}$ is substituted with $\overline{\ddot{\mathbf{p}}}_{p,ref}(t)$. Therefore, the ante- and post-impact joint-space task reference trajectories can be defined as

$$\overline{\boldsymbol{\gamma}}_{a,ref}(t) = \begin{bmatrix} \overline{\mathbf{q}}_{a,ref}(t) \\ \overline{\dot{\mathbf{q}}}_{a,ref}(t) \\ \overline{\ddot{\mathbf{q}}}_{a,ref}(t) \\ \mathbf{0}_{n_c \times 1} \end{bmatrix}, \qquad \overline{\boldsymbol{\gamma}}_{p,ref}(t) = \begin{bmatrix} \overline{\mathbf{q}}_{p,ref}(t) \\ \overline{\dot{\mathbf{q}}}_{p,ref}(t) \\ \overline{\ddot{\mathbf{q}}}_{p,ref}(t) \\ \frac{\lambda_{ref}}{n_c}\mathbf{1}_{n_c \times 1} \end{bmatrix}, \tag{3.31}$$

where $\overline{\mathbf{q}}_{a,ref} \in \mathbb{R}^n$, $\overline{\dot{\mathbf{q}}}_{a,ref} \in \mathbb{R}^n$, and $\overline{\ddot{\mathbf{q}}}_{a,ref} \in \mathbb{R}^n$ are the ante-impact reference joint displacement, velocity, and acceleration, respectively, and $\overline{\mathbf{q}}_{p,ref} \in \mathbb{R}^n$, $\overline{\dot{\mathbf{q}}}_{p,ref} \in \mathbb{R}^n$, and $\overline{\ddot{\mathbf{q}}}_{p,ref} \in \mathbb{R}^n$ are the post-impact reference joint displacement, velocity, and acceleration, respectively. Furthermore, the joint displacements, velocities, and accelerations associated with the task-space task reference trajectories shown in Figures 3.12 and 3.13 are shown in Figures 3.15 and 3.16, respectively.

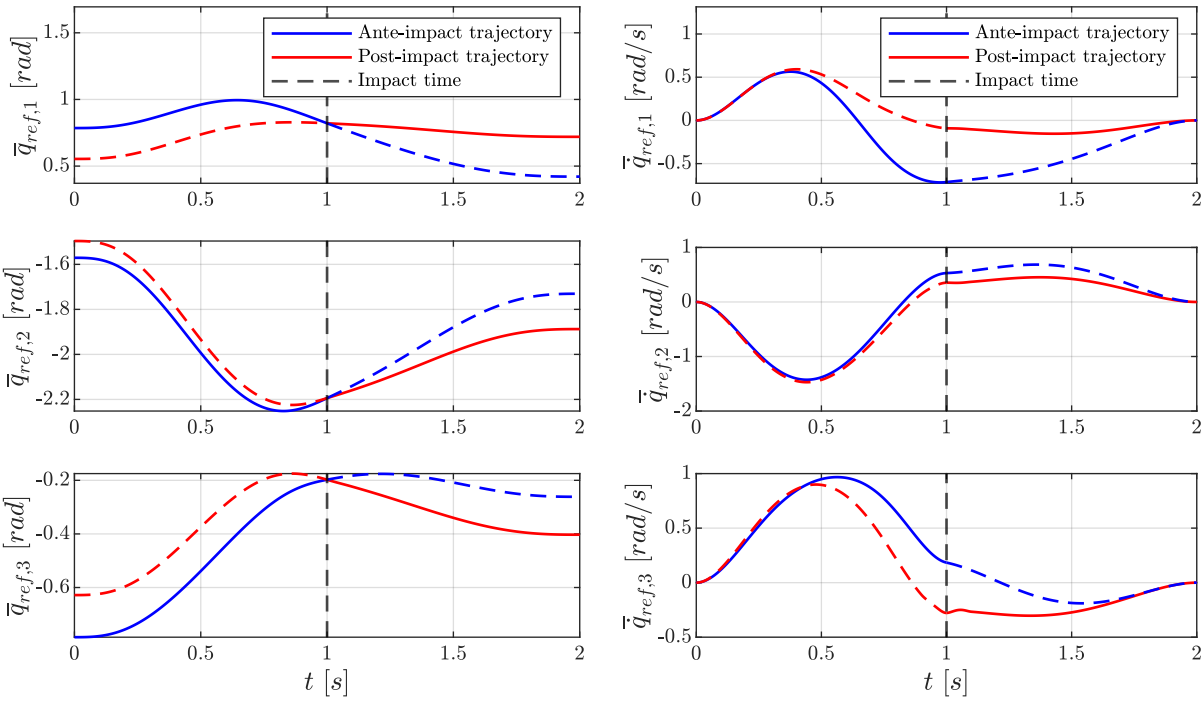
It should be noted that the inverse kinematics expressed in (3.26), (3.27), and (3.28) assume that the joints of the robotic manipulator are rigid. This assumption simplifies the inverse kinematics and ensures that no in-depth knowledge regarding the joints is necessary for determining the joint-space trajectories which is beneficial if it is going to be implemented into a real system. Similarly, as shown in (3.17), to simplify the link between the ante- and post-impact trajectory, it is also assumed that the impact is rigid. The difference between the rigid prediction and the simulated flexible joint and compliant impact behavior is what will need to be analyzed to properly classify the post-impact behavior and will be further detailed in Chapter 4.

3.5 Numerical Simulation Results

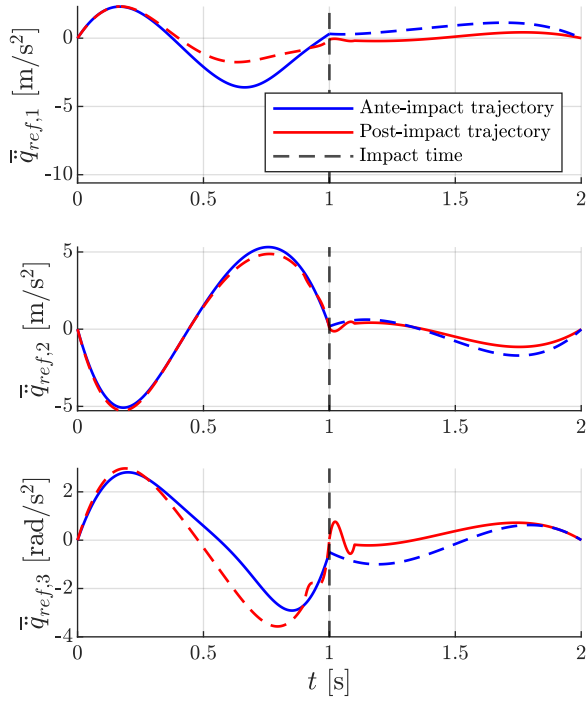
In previous sections, the impact scenarios, a multi-impact detection method, a control strategy that allows for switching between different impact phases, and the extended task reference trajectories were defined. In this section, simulations will be performed to ensure consistency between the results obtained from simulations using different model types. In this context, consistency relates to the differences in behavior between different model types and whether these differences can be explained through the differences in the models, i.e., a set of models are consistent if the differences between the model dynamics allow for the observed differences in behavior. This consistency is checked because if two models are found to be inconsistent, then that would either indicate that there is an error in the model or that a simplified model cannot be used to predict the behavior of a complex model, which would show that classification of the impact behavior based on a simple prediction is impossible. These simulations are performed for a single- and simultaneous impact case with a task-based QP controller that uses joint-space configuration and contact tasks to define the desired motion of the robotic manipulator. When investigating the consistency between the results from different model types, the primary focus is to check whether the rigid impact with rigid joints case provides accurate impact dynamics when compared to the compliant impact and flexible joint cases. For further information regarding the model types, Section 2.2 contains a detailed explanation of each model. Furthermore, as a brief refresher, the overview of the different model types is as follows:

- Model A: A robotic manipulator with flexible joints impacting an environment with compliant contact dynamics.
- Model B: A robotic manipulator with flexible joints impacting a rigid environment.
- Model C: A robotic manipulator with rigid joints impacting a rigid environment.

For illustrative purposes to show the differences in behavior for an expected and unexpected impact scenario for both the single- and simultaneous impact case, an expected and unexpected scenario will be

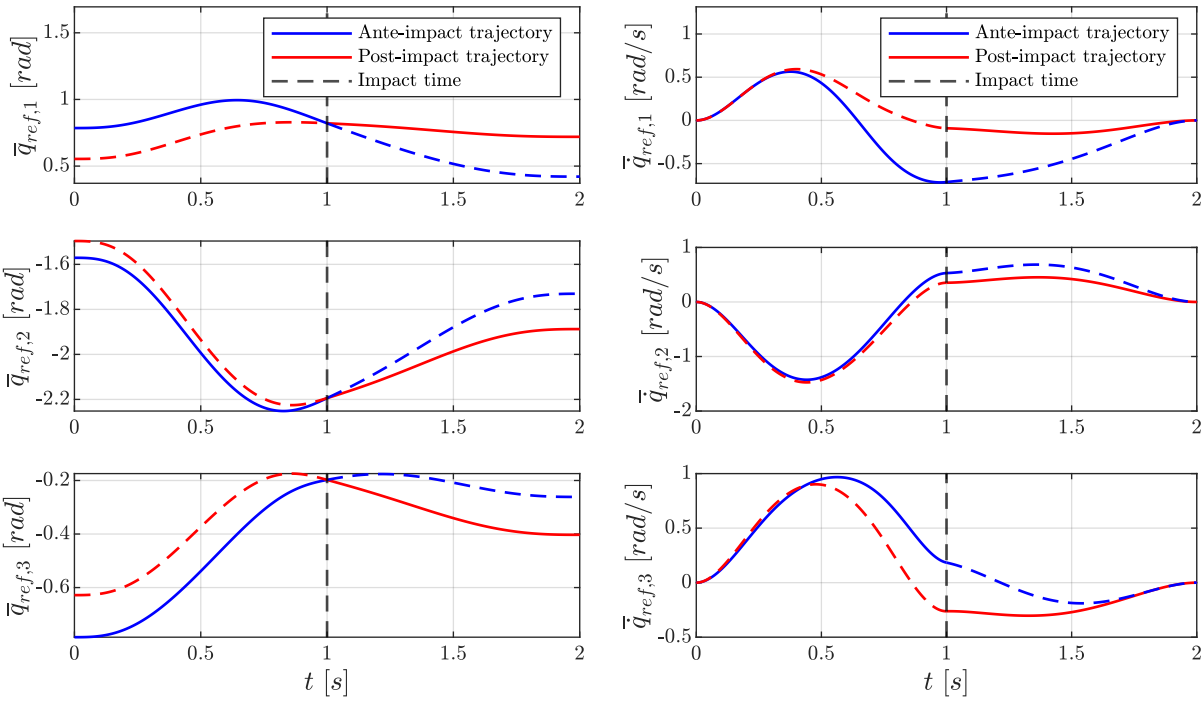


(a) (b)

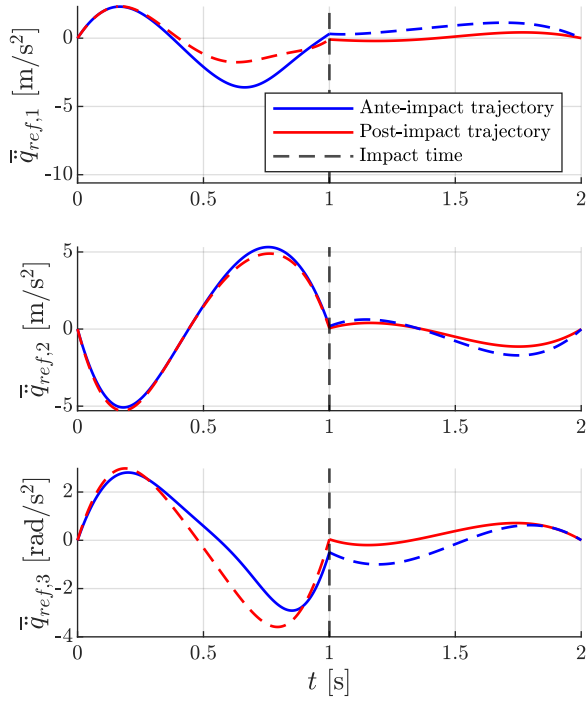


(c)

Figure 3.15: Corresponding joint-space task reference trajectories for the single impact case (See Figure 3.12 for task-space task reference trajectories). In (a), the joint-space displacement task. In (b), the joint-space velocity task. In (c), the joint-space acceleration task.



(a) (b)



(c)

Figure 3.16: Corresponding joint-space task reference trajectories for the simultaneous impact case (See Figure 3.13 for task-space task reference trajectories). In (a), the joint-space displacement task. In (b), the joint-space velocity task. In (c), the joint-space acceleration task.

simulated with each model type. For the expected scenario, the expected and real environment parameters and trajectory parameters are as presented in Tables 3.1 and 3.2. For the unexpected case, most of the parameters presented in Tables 3.1 and 3.2 remain the same, however, the real environment is rotated by $\beta = 5\ deg$, and the elevation of the environment is changed to ensure that the impact location remains the same. To achieve this, the elevation of the environment is changed according to the derived relation

$$l = -\tan(\beta)\,{}^{A}\mathbf{x}_A^{\top}\,{}^{A}\mathbf{p}_{imp}, \tag{3.32}$$

where ${}^{A}\mathbf{x}_A = [1, 0, 0, 0, 0, 0]^{\top}$ is the vector notation of the unit vector $\vec{x}_A$ extended by $\mathbf{0}_{3\times 1}$ to ensure that the equation is mathematically sound.

3.5.1 Single Impact Case

First of all, a single impact case is considered to test the behavior of the different model types. As described in the introduction of this section, two scenarios are considered: an expected and unexpected post-impact scenario. In Figure 3.17 the position and velocity of the expected scenario are presented in joint- and task-space. Furthermore, as the impact behavior is of interest, this figure is zoomed in on the behavior in the time interval surrounding the impact.

Before discussing the results, it is important to discuss the defining characteristic of each model type. Between the three models discussed in Section 2.2, two components contribute to the characteristic behavior in each model. These components are what robotic manipulator model is used and what method is used to simulate contact with the environment. When considering the robotic manipulator model, two different types were discussed. These model types are the rigid- and flexible joint robotic manipulators which are explained in detail in Section 2.2.2. The main difference between these models is how the robotic manipulator is actuated, where on one hand the rigid joint manipulator assumes that the transmission between the motor and the joint is rigid, whereas, on the other hand, the flexible joint manipulator assumes that the transmission is flexible. Due to this flexible transmission, it is to be expected that when undergoing an impact, oscillations will occur that will damp out after a short period. For the rigid joint case, it is to be expected that no such vibrations will occur.

When considering a method to simulate contact with the environment, two different methods were discussed in Section 2.2.3. These methods are the compliant surface model and the rigid impact map, where the first method determines the contact force as a function of the indentation of the contact point on the environment, whereas the second uses a rigid impact map to determine the post-impact velocities and a bilateral constraint to determine the contact force. When comparing the simulations, it is expected that the main difference will be that for the compliant surface model, the change in velocity will not be instantaneous. Furthermore, the compliant surface model also allows for contact with the environment to cease, which would also result in the possibility of the end-effector bouncing, this contact model also allows for further indentation of the environment, which can result in small changes in velocity. This behavior is not possible for the rigid environment, as the constraint used to ensure that the end-effector does not pass through the environment is implemented as a bilateral constraint on the position of the contact point in the $\vec{y}_B$ direction.

Consider again Figure 3.17, here, an expected single impact scenario is simulated for the three models discussed in Section 2.2. It can be observed that the behavior of all models converges to the same response. Furthermore, model C exhibits a response without oscillations, while models A and B both show this behavior due to the flexible joints in those models. Furthermore, models B and C have an instantaneous jump in velocity while model A has a fast, but not instantaneous, change in velocity. Finally, model A shows small differences in the post-impact velocity. These differences can be attributed to contact between the environment and the end-effector ceasing for a brief period. This conclusion can be drawn by observing the vertical velocity of the end-effector, which undergoes an increase followed by a decrease after the impact has occurred. It should also be noted that model C follows the trajectory perfectly, showing that the method used to predict post-impact velocity used to generate the task reference trajectory in Section 3.4 is correct.

Furthermore, it should be noted that in the figure showing the end-effector orientation, the three models show different behavior from each other. However, it should be emphasized that these differences are in the order of magnitude of $10^{-4}\ rad$, which is negligible. The differences can be explained due to the

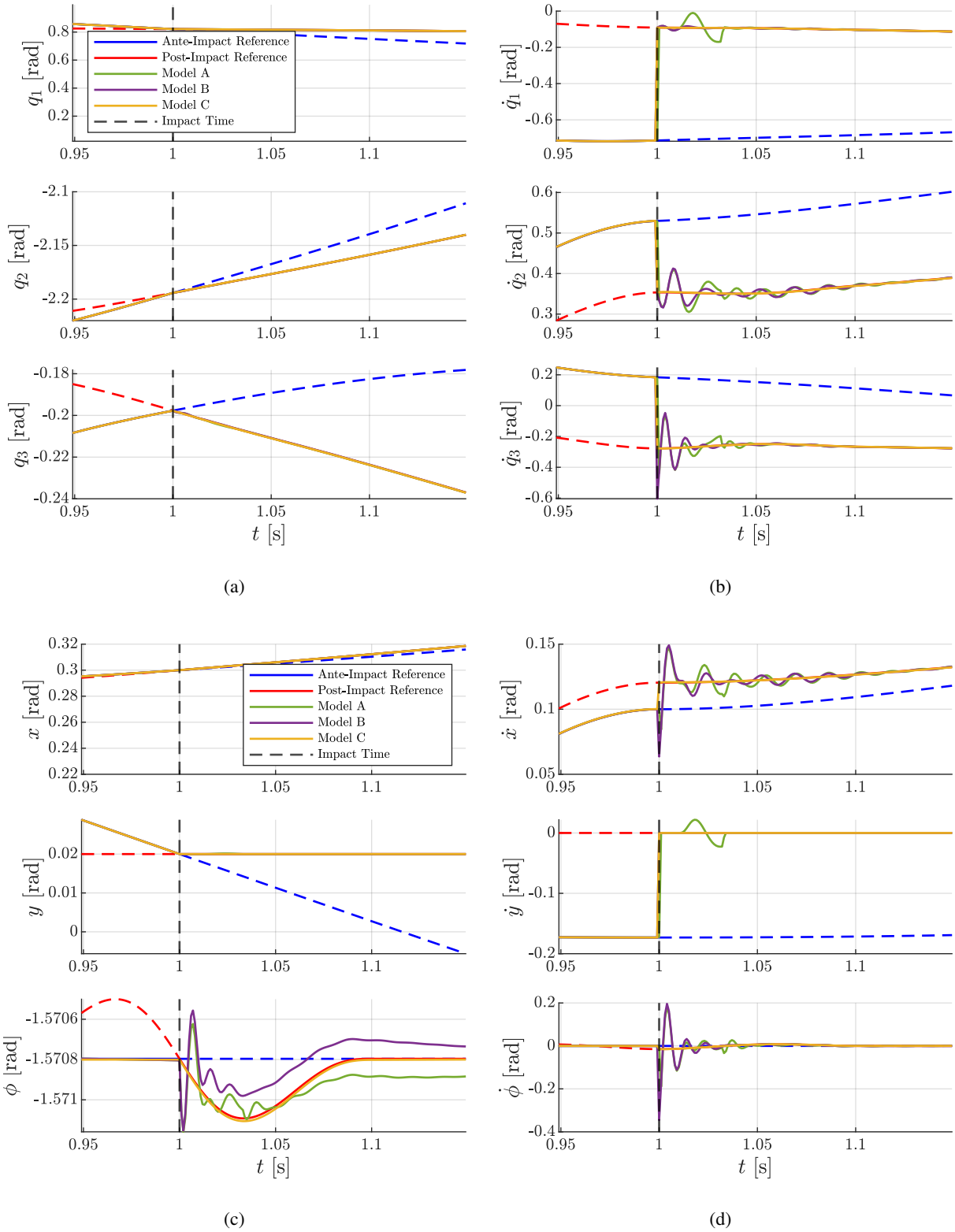


Figure 3.17: Comparison between the simulated response of model A, B, and C, and the ante- and post-impact trajectories for an expected single impact. In (a) and (b), the joint-space behavior. In (c) and (d), the task-space behavior.

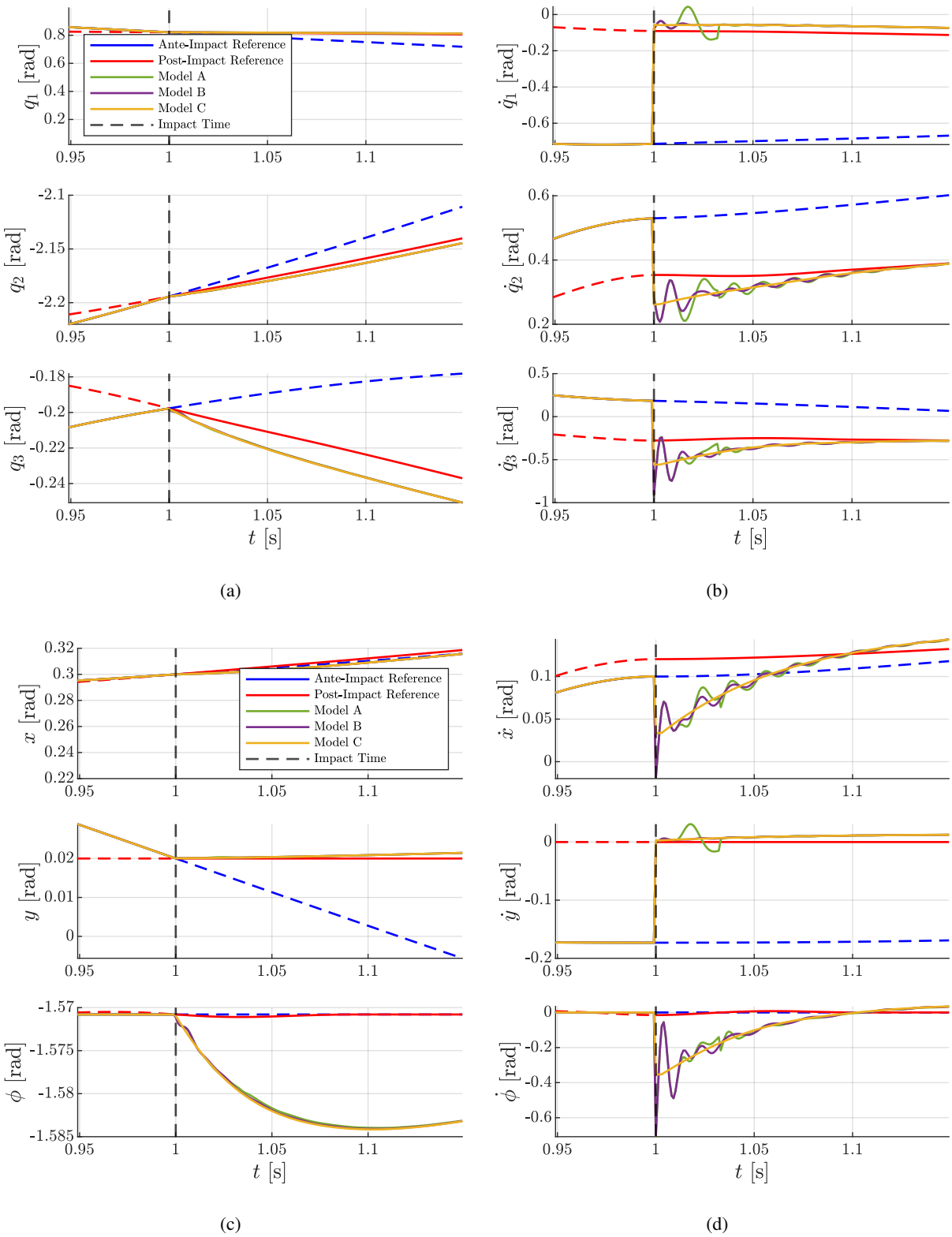


Figure 3.18: Comparison between the simulated response of model A, B, and C, and the ante- and post-impact trajectories for an unexpected single impact. In (a) and (b), the joint-space behavior. In (c) and (d), the task-space behavior.

difference between the model used to control the robotic manipulator and the model that is controlled. The QP controller assumes that there are rigid joints, which leads to model C producing an accurate tracking of the task reference. Both models A and B use a different manipulator model, which explains why the response obtained from these models does not track the task reference as well as model C does. Furthermore, the difference between models A and B can be explained by the difference in how the impact is modeled.

The single impact case is then tested with an unexpected impact scenario, the results of which can be found in Figure 3.18. This figure presents the behavior in joint-space and task-space. When investigating the unexpected scenario response, it is clear that the response of each model is different than was expected. Furthermore, by comparing the responses of each model type, similar observations can be made concerning the differences between the model types. Namely, that model C does not exhibit vibrations, but models A and B do and that the change in velocity is instantaneous for models B and C, but model A has a fast, but not instantaneous, change in velocity. Furthermore, the behavior that is assumed to originate from the end-effector ceasing contact with the environment for a brief period in model A is also observed.

Finally, to verify that the cause for the observed differences between models A and B are caused by contact that is briefly lost, the gap functions for these scenarios are plotted. These gap functions can be found in Figure 3.19 for the expected and unexpected impact scenario. From this figure, it can be seen that contact is lost briefly at the same time as to where the differences between models A and B occur, verifying the earlier drawn conclusions. Therefore, with these observations verified, it can be concluded that the three models are consistent with each other for the single impact case.

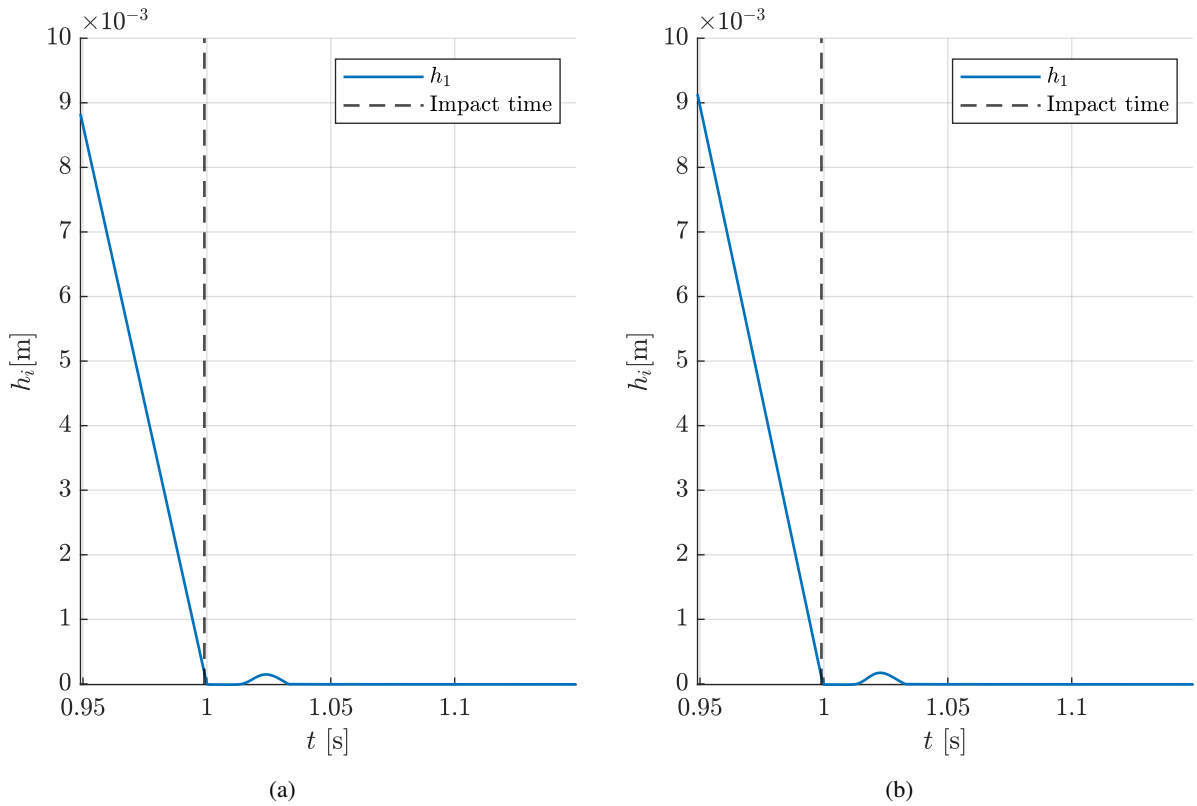


Figure 3.19: Gap functions for the single impact case. In (a), for the expected impact scenario in Figure 3.17. In (b), for the unexpected impact scenario in Figure 3.18.

3.5.2 Simultaneous Impact Case

With the single impact case tested and checked for consistency between models, the simultaneous impact case is tested next. All of the results obtained from these tests can be found in Figures 3.20, 3.21, 3.22, 3.23, and 3.24. For formatting purposes, these figures are shown on the following pages in this order and will be elaborated on individually in this section.

The results of the first test for the simultaneous impact case are shown in Figure 3.20, where the impact response for an expected simultaneous impact case is shown. In this figure, some interesting behavior can be observed. Namely, that the response of model A is significantly different from the response of models B and C. The responses for models B and C are similar to each other, where model C does not oscillate, but model B does. However, the response of model A converges to the post-impact trajectory sooner than models B and C.

This behavior can be explained by the switching strategy used when switching from the intermediate phase to the post-impact phase. For a (nearly) ideal simultaneous impact, such as what is shown in Figure 3.20, only one impact will be detected, which means that the control strategy switches to the intermediate phase. After $t_{switch} = 0.15\ s$ have passed, the strategy for models B and C will switch to the post-impact phase, whereas, for model A the switch to the post-impact phase occurs at $t \approx 1.07$. Observe that at $t \approx 1.07$ there is behavior that indicates that a second impact might have occurred. This can be verified using Figure 3.23, which shows that the impact detection condition discussed at the end of Section 3.2 is satisfied a second time, which is caused by the reestablished of contact after it has been briefly lost. When this impact gets detected, the switching conditions discussed in Section 3.3.2 cause a switch in the control strategy from the intermediate impact phase to the post-impact phase. As explained in Section 3.5.1, Models B and C do not have the capability for contact release, which means that a second impact cannot occur, which explains why these models show different behavior than model A.

By adjusting the original detection bound defined in Section 3.2 to $\mathbf{r}_b = [6000, 2500, 1450]^\top$, the second impact is no longer detected. This results in the response found in Figure 3.21, which shows that the three responses converge and then switch to the post-impact strategy at $t = 1.15\ s$. The updated impact detection bounds are only used for Figure 3.21 because the original impact detection bounds were chosen based on their performance of detecting multiple impacts for a large set of scenarios. Furthermore, while the behavior of model A shown in Figure 3.20 is different from the behavior of models B and C, it will not affect the goal of this project as it can be used as an additional source of unpredictable behavior that the classification method should be able to handle. A possible solution to this problem that is scale-able for multiple scenarios would be to develop a method to confirm whether contact has been completed, however, this project will continue with the current method of switching between strategies.

The second test for the simultaneous impact case concerns the unexpected impact scenario, which results in the loss of simultaneity of the impact. The results of this test can be found in Figure 3.22, where the response of the unexpected impact scenarios is presented in joint- and task-space. When looking at Figure 3.22, it is clear that all models A, B, and C show behavior that are similar to each other. Furthermore, all models switch to the intermediate- and post-impact phase at the same time, avoiding the problem that was observed in Figure 3.20. Furthermore, it can also be observed that contact is briefly lost before it is reestablished, which causes the main differences between the response of models A and B. This detachment after each impact is also shown in Figure 3.24, which shows the gap function for each contact point.

With the results presented in this section, all differences between the impact responses for the simultaneous impact case for each model can be explained by the differences in the modeling of the behavior. Therefore, it can be concluded that the different models are consistent with each other for the simultaneous impact case. Furthermore, while differences between the post-impact trajectory and the response of model A were observed for an ideal simultaneous impact in Figure 3.20, it will still be suitable for classification purposes due to the requirement that the classification method should be able to deal with unexpected behavior that should not be classified as unexpected.

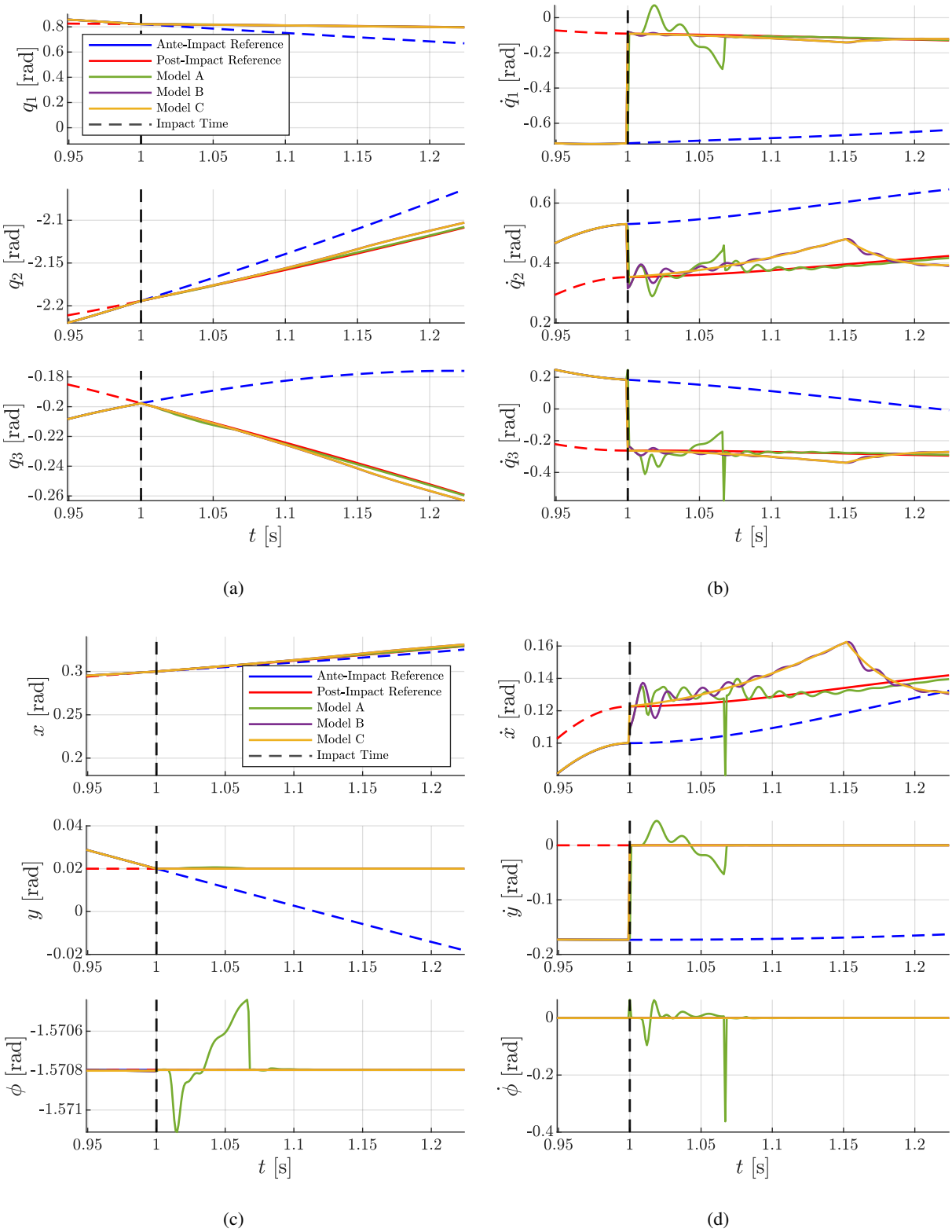


Figure 3.20: Comparison between the simulated response of model A, B, and C, and the ante- and post-impact trajectories for an expected simultaneous impact. In (a) and (b), the joint-space behavior. In (c) and (d), the task-space behavior.

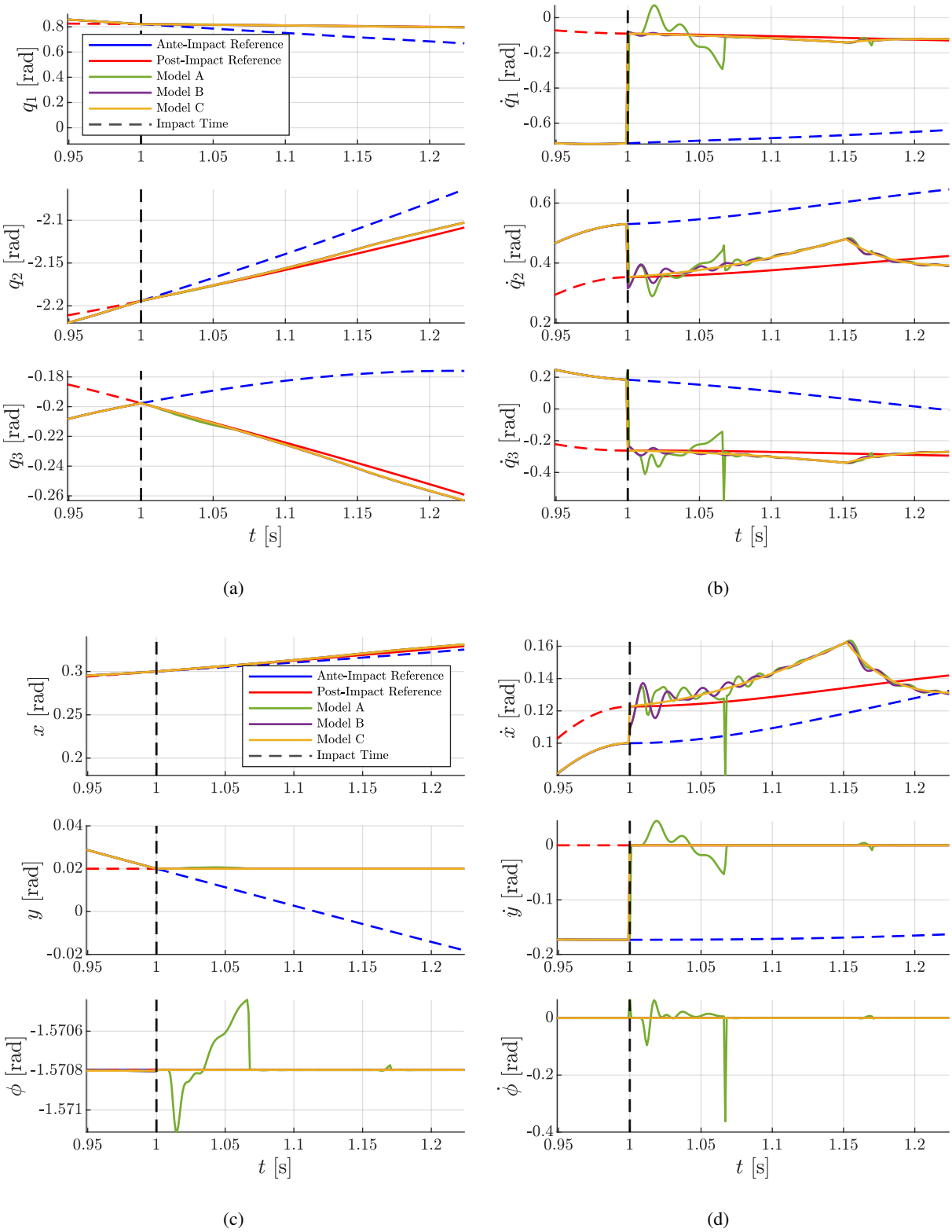


Figure 3.21: Comparison between the simulated response of model A, B, and C, and the ante- and post-impact trajectories for an expected simultaneous impact with adjusted impact detection bounds. In (a) and (b), the joint-space behavior. In (c) and (d), the task-space behavior.

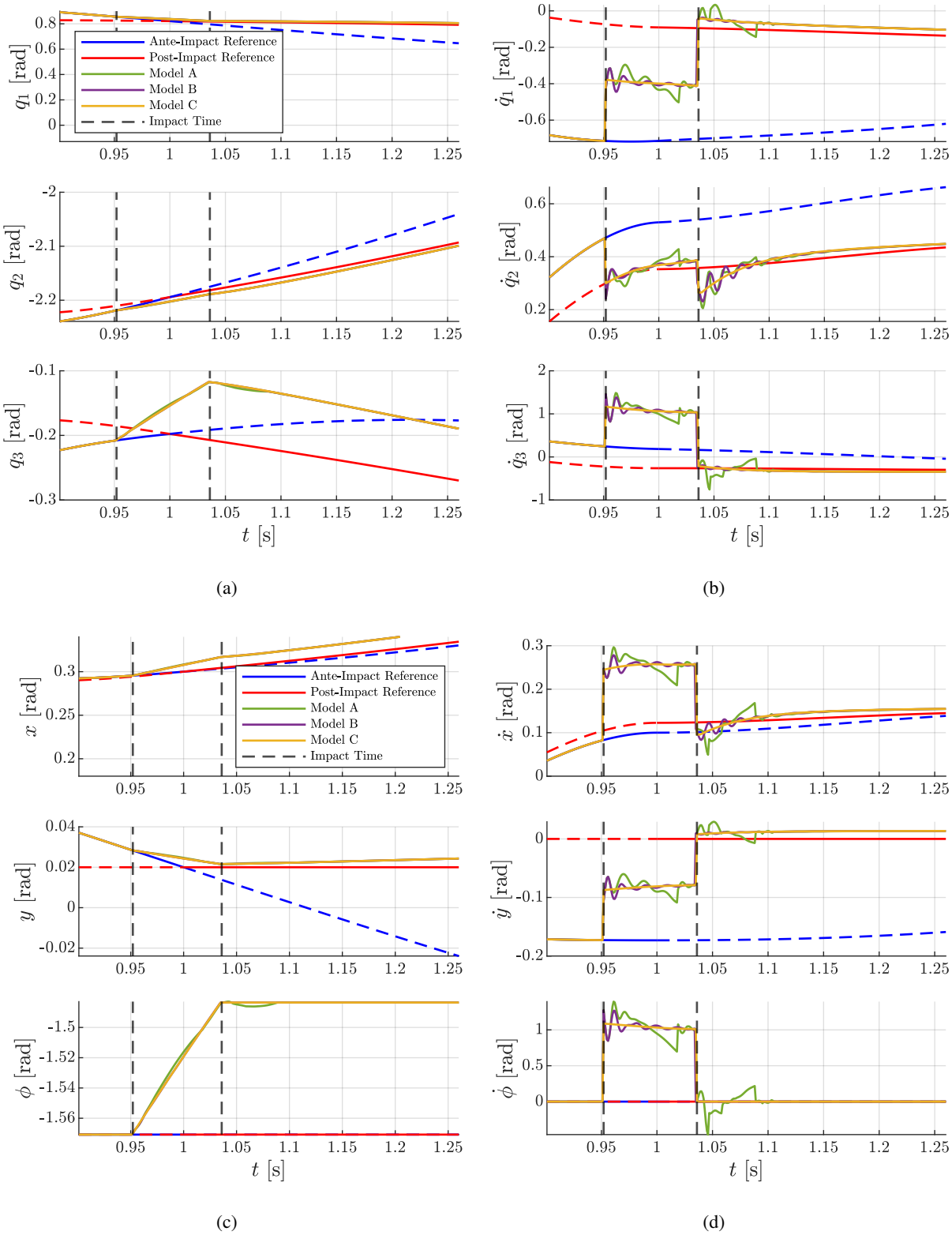
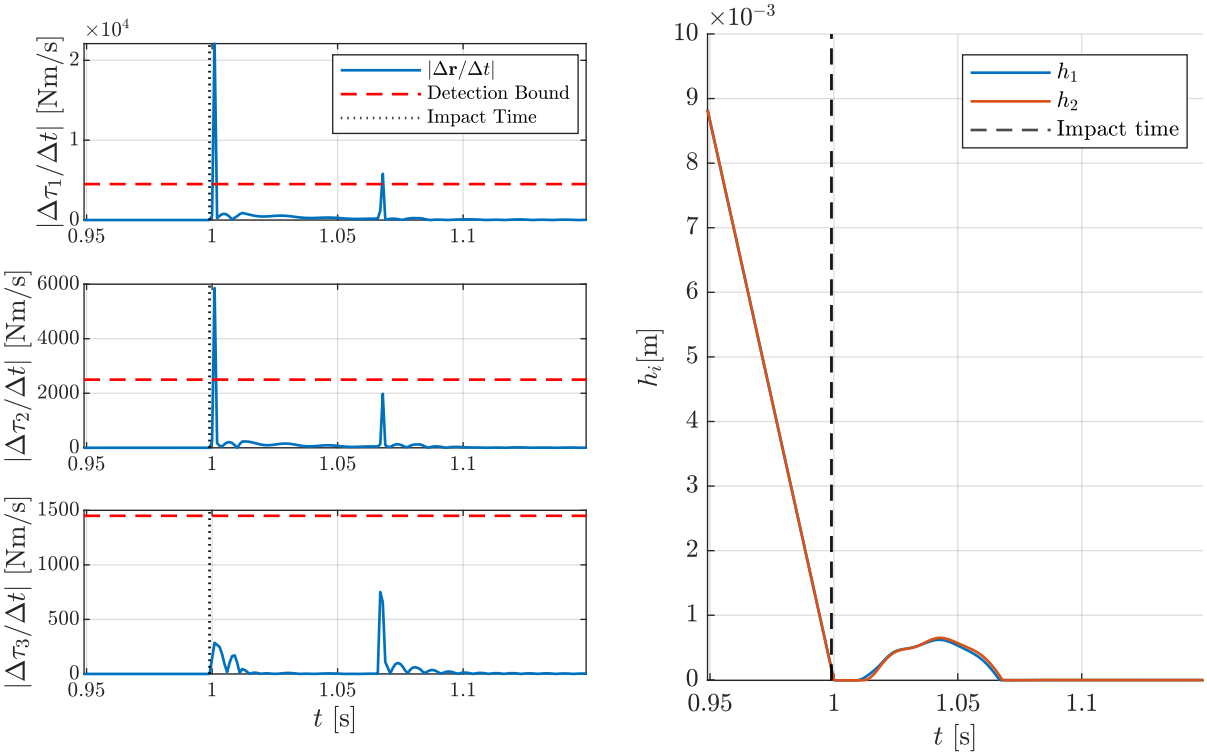


Figure 3.22: Comparison between the simulated response of model A, B, and C, and the ante- and post-impact trajectories for an unexpected simultaneous impact. In (a) and (b), the joint-space behavior. In (c) and (d), the task-space behavior.



(a) (b)

Figure 3.23: Impact detection for the ideal simultaneous impact case in Figure 3.20. In (a), the impact detection condition. In (b), the gap function.

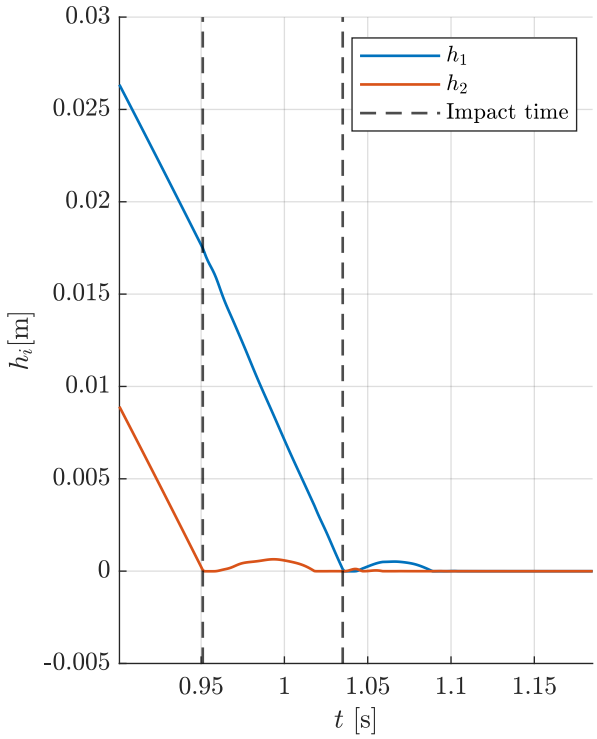


Figure 3.24: Gap function for the simultaneous impact case for the unexpected scenario in Figure 3.22.

3.6 Reflection

When investigating the results obtained during this chapter, it is important to reflect on the unexpected behavior that was observed. This reflection pertains to whether the unexpected behavior observed in this chapter still allows for the development of a classification method to classify the expectancy of simultaneous impacts. The points of reflection are, the validity of the switching strategy discussed in Section 3.3.2 and the brief moments of contact point detachment after an impact observed in model A. This is then concluded by discussing whether the simulation discussed in this chapter can be used as a representation of a real system and if there is evidence to suggest that classification of single- and simultaneous impact scenarios is possible.

Validity of the Switching Strategy

As was shown in Figure 3.20, during an ideal simultaneous impact, it is possible for the brief loss and subsequent reestablishment of contact to result in a second detected impact. As a consequence, an early transition from the intermediate- to post-impact phase with the strategy designed in Section 3.3.2 occurs. As was seen in Figure 3.20 and Figure 3.21, this behavior is unique to model A, and can be mitigated through small changes in the impact detection bounds or an improved switching strategy. The current impact detection bounds were tuned to allow for multi-impact detection in a variety of unexpected impact scenarios. Therefore, changing the impact detection bounds to be less sensitive risks other scenarios from being detected.

Therefore, the other method is finding another solution for when to switch to the post-impact phase. This solution can take a variety of forms, namely, as a method that detects whether contact has been completed or by waiting until no impact is detected for a predefined time frame after the last impact. Currently, no method exists that detects the completion of contact, furthermore, alternate switching conditions based on the available information are likely to exhibit problems intrinsic to that method that might not occur with the method used in this project. Developing a better switching strategy, either through a contact completion detection algorithm or through alternate switching conditions, is left for a possible follow-up project.

Finally, with the current switching strategy in place, behavior that is different from models B and C can occur. However, a real system will always undergo unexpected behavior, which can range from small effects such as noise to large effects such as incorrect parameters when obtaining the prediction of the post-impact velocity for the expected scenario. This unexpected behavior is inevitable and the classifier is therefore required to be able to deal with this kind of unexpected behavior, which includes the detection of additional impacts. Therefore, it is not believed that this behavior will hinder the development of the classification method in Chapter 4.

Contact Point Detachment

As was discussed in Section 3.5, when the system is undergoing an impact, loss of contact occurs, after which contact is reestablished. This brief loss of contact results in the dynamics that differentiate the behavior of model A from models B and C. Furthermore, as was observed in Figure 3.23, the reestablishment of contact can result in the detection of an impact, which can result in the control strategy switching from the intermediate- to post-impact phase. It is unlikely that this behavior can be solved through adjustments to the controller, however, the mitigation of bounces could be a possible avenue for further research. However, regardless of whether this behavior is avoidable, model A is still representative of a real system. As was explained earlier, a real system always has unexpected behavior, and the classification method should be able to deal with small unexpected occurrences in the dynamics.

Representation for Realistic System

With the points discussed earlier brought forward, it is important to conclude whether the current implementation of model A is representative of a realistic system. The differences between the responses obtained in Section 3.5 can be explained by the differences between the models, which indicates that

model A is consistent with models B and C. Therefore, in particular model C can act as a simplified model for model A. As was brought up earlier, the unexpected behavior that occurs due to the brief loss of contact does not negate the fact that model A is representative of a real system, as this unexpected behavior can also occur on a real system. Finally, considering that a simplified model exists for model A that allows for the prediction of the post-impact velocity suggests that it is possible to classify impact scenarios using this prediction and the subsequently generated trajectory. This is expanded on further in Chapter 4.

4 Impact-Aware Classifier for Single- and Simultaneous Impacts

In the previous chapter, a joint-space trajectory was generated that employs a prediction of the post-impact velocity to improve the tracking performance after an impact occurs. Furthermore, this trajectory was compared to the simulated responses of models A, B, and C and this comparison resulted in the observations that for an expected impact scenario, model C serves as an accurate simplified model when compared to model A for single- and simultaneous impact cases. This chapter focuses on developing a classification method that uses the rigid prediction and the subsequently generated joint-space trajectory to classify whether a single- or simultaneous impact case was expected or unexpected, i.e., to classify the expectancy of an impact scenario. Before a classification method can be developed, the characteristic behavior of expected impact scenarios will be investigated, which is behavior that is indicative that an expected impact has occurred.

The characteristic behavior is discussed in Section 4.1. In Section 4.2 a signal envelope filter will be designed for the classification of the post-impact behavior. This is then followed by designing and testing the classifier for a single impact case in Section 4.3, which is then adjusted and tested for a simultaneous impact case in Section 4.4.

4.1 Characteristic Behavior for Expected Impact Scenarios

When considering the classification of an impact scenario, two distinct challenges can be identified. The first challenge is the classification of the impact time and location. The second challenge is the classification of the post-impact behavior, where the response as a consequence of the impact is of importance. A brief refresher regarding what constitutes an expected and unexpected impact scenario is given in Section 4.1.1. Furthermore, in Sections 4.1.2 and 4.1.3, the challenges for classifying impact location and time and post-impact behavior are discussed, respectively. Furthermore, both Sections 4.1.2 and 4.1.3 will initially focus on the classification of single impact scenarios which is subsequently expanded to the simultaneous impact case.

4.1.1 Expected and Unexpected Impact Scenarios

As was originally discussed Section 3.1, two types of impact scenarios can be identified, namely, an expected or unexpected impact scenario. The type of impact scenario that occurs is dependent on the accuracy of the known information about the environment compared to the real environment. This set of known information about the environment is used to define the expected environment. The expected environment is then used together with a set of user-defined trajectory parameters to define an ante-impact trajectory with the aim of the robotic manipulator impacting the environment at a specific time and location. The post-impact trajectory is then designed to maintain contact with the environment and is based on the expected geometry of the environment and the predicted post-impact velocity. For this project, the expected environment was defined with the information presented in Table 3.1 and unless specified otherwise, the user-defined trajectory parameters used are defined in Table 3.2. Consider Figure 4.1 (a), where an example of an expected impact scenario using these parameters is shown, where the robotic manipulator impacts the environment at the expected time and location.

When considering an unexpected impact scenario, two distinct types of unexpected scenarios can be identified. Namely, a scenario where the impact occurs at a different time or location than was expected and a scenario where the impact location and time were expected, but the post-impact behavior was unexpected. The first type of unexpected scenario can be caused by an object obstructing the desired trajectory or if the location of the environment was not as expected. For the simulation, this can be implemented by changing the elevation $l \in \mathbb{R}$ of the environment, but not changing the expected elevation of the environment. This method of implementing an unexpected impact is also shown in Figure 4.1 (b) and (c). The second type of unexpected scenario can be caused by a different shape of the environment, such as the environment being rounded or sloped when it was expected that the environment was flat and

horizontal. In the simulation, this can be implemented by adjusting the rotation $\beta \in \mathbb{R}$ of the environment and adjusting the elevation of the environment by using the trigonometric relation $l = -\tan(\beta)^{A}\mathbf{x}_{A}^{\top}{}^{A}\mathbf{p}_{imp}$ to ensure that the impact location remains the same. This method of implementing an unexpected post-impact scenario can be found in Figure 4.1 (d) and (e).

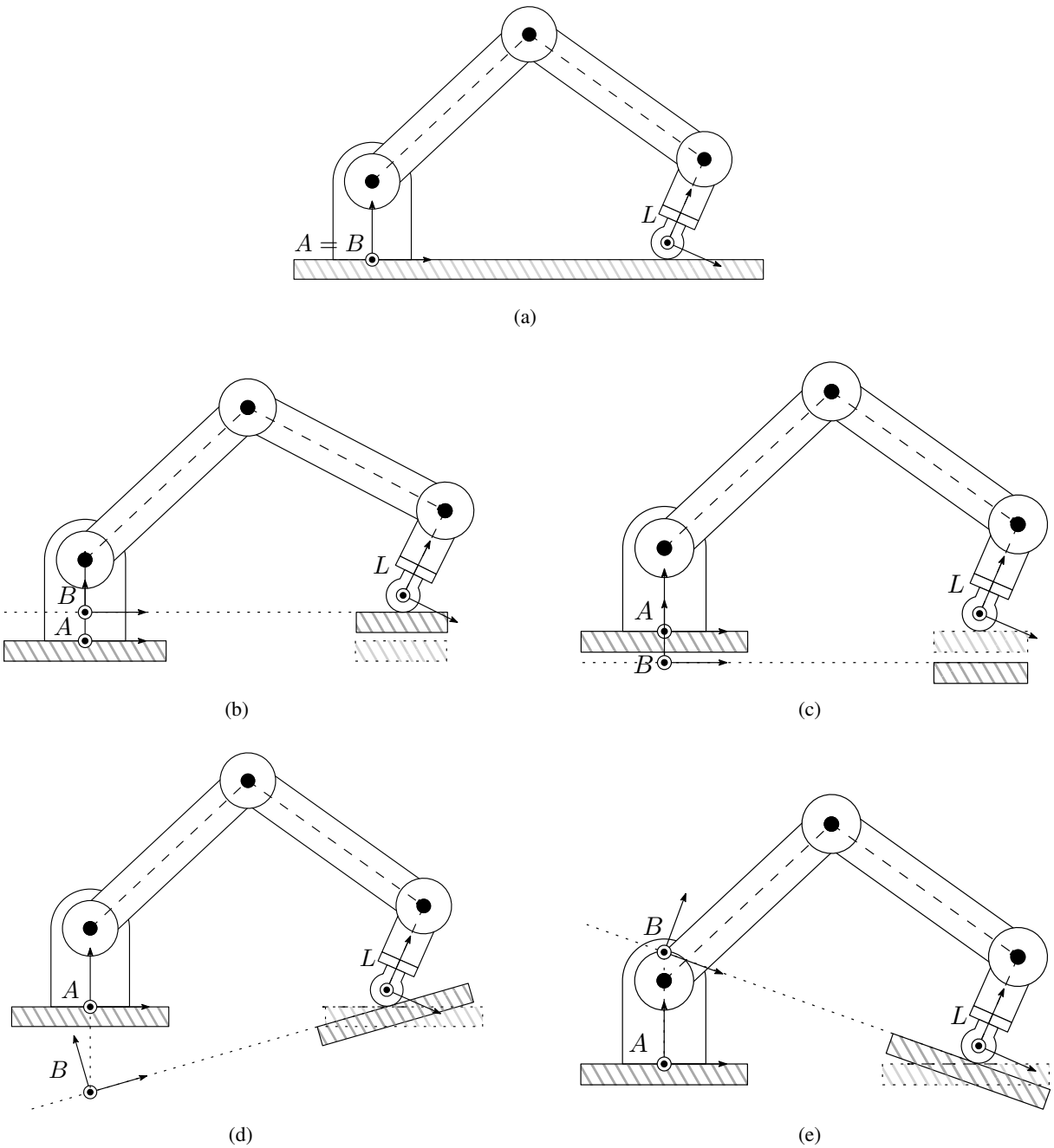


Figure 4.1: Schematic representation of five impact scenarios. In (a), an expected impact scenario where $l = 0$ and $\beta = 0$. In (b), an unexpected impact scenario where the impact surface is placed higher than expected, i.e., $l > 0$. In (c), an unexpected impact scenario where the impact surface is placed lower than expected and no impact occurs, i.e., $l < 0$. In (d), an unexpected post-impact scenario where the environment is sloped upwards because $\beta > 0$. In (e), an unexpected post-impact scenario where the environment is sloped downwards because $\beta < 0$.

4.1.2 Classification of Impact Location and Time

When considering an impact, there are two properties that are relevant for the classification of its expectancy, which are the impact time and impact location. Additionally, the impact time is normally only relevant for time-based tasks, however, for completeness, the impact time will be considered for the classification method. Classification based on impact time for a single impact case can be performed using two variables, the desired impact time $t_{1,d}$ and the detected impact time t_1. Ideally, the desired and detected impact time should be equal. However, due to small errors when controlling the robotic manipulator, delays in the detection of an impact, the existence of small unpredictable perturbations in the state trajectory, and due to bad recognition of the robot's environment, this condition is too strict to apply to a real system. By using an adjustable threshold the concept behind such a method can still be implemented. For a single impact case, this results in a simple classification condition for an expected impact defined as

$$\|t_1 - t_{1,d}\| \leq \zeta, \tag{4.1}$$

where $\zeta \in \mathbb{R}_{\geq 0}$ is the user-defined maximum allowable time difference between the detected and desired impact time and acts as a time region where if impacts occur within ζ s of the desired impact time, then the impact time was expected. Furthermore, ζ is tuned based on the maximum allowable time difference between the desired and detected impact time for a time-based operation and can vary significantly based on the purpose of the controlled system. To include simultaneous impacts into the classification condition, (4.1) needs to be expanded for an arbitrary amount of impacts. In such a case, there is still only one desired impact time $t_{1,d}$, however, multiple impact times can be detected and are denoted by t_j, where $j \in \mathbb{N}$ represents the j-th detected impact. Using the same principle that was used to construct (4.1), the updated condition for a simultaneous impact case can be defined as

$$\|t_j - t_{1,d}\| \leq \zeta, \quad \text{for } j \in [1, j_f)\,. \tag{4.2}$$

To account for the possibility of bounces, i.e., the brief detachment of contact points observed in Section 3.5.2 which can result in an additional impact detection, it was decided to not define a strict limit to the number of impacts that can be detected around the desired impact time. Instead, $j_f \in \mathbb{N}$ represents the final detected impact during $t \in [t_0, t_f]$, where t_0 is the starting system operation time and t_f is the final system operation time. Furthermore, tuning ζ for a simultaneous impact case functions in the same way as for the single impact case, however, depending on the time-based task it might need to be tuned differently to account for the possibility of multiple impacts.

Furthermore, for simultaneous impacts, the time difference between two impacts is also relevant for the classification, i.e., if the time between two impacts is larger than desired, then the impact should also be classified as unexpected. When the simultaneity of an ideal simultaneous impact is lost several impacts are detected and this time difference is indicative of how large the difference is between the expected and real scenario, i.e., the time between impacts become larger the further the real environment deviates from the expected environment for which the trajectory was generated. Using a user-defined threshold, a condition can be written that classifies the impact based on the maximum allowable time difference between impacts. This condition is written as

$$\|t_j - t_{j-1}\| \leq t_{int}, \quad \text{for } j \in [2, j_f)\,, \tag{4.3}$$

where t_{int} is the user-defined maximum allowable time difference between impacts. Similarly to ζ, t_{int} is tuned based on the allowable difference between the expected and real scenario. However, while ζ is tuned for the allowable difference between the desired and detected impact time, t_{int} is tuned on how far the impact can deviate from an ideal simultaneous impact. In other words, t_{int} should be tuned lower if it is desired that the impact is as close to an ideal simultaneous impact as possible. With the conditions presented in (4.2) and (4.3), the classification of an impact using the detected impact times can be performed.

With a condition to classify an impact based on the detected impact time defined, the classification based on the impact location will be defined next. The impact location can be defined in two separate ways, the task-space impact location and the joint-space impact configuration. The task-space impact location is the position of the end-effector in task-space space at the time of impact, which is calculated using the measured joint displacement at the time of impact through forward kinematics. The joint-space impact

configuration is the measured joint displacements at the time of impact. For robotic manipulators with at least 2 degrees of freedom, every feasible task-space end-effector position and orientation that does not result in a singular configuration can be achieved with two possible joint configurations. Therefore, to reduce the amount of false-positive impact classifications, the classification of the impact location is done using the joint-space impact configuration. For a single impact case, two quantities are defined, the measured joint displacements at the detected time of impact $\mathbf{q}_{imp} = \mathbf{q}(t_1) \in \mathbb{R}^n$ and the desired joint displacements at the desired time of impact $\overline{\mathbf{q}}_{ref,imp} = \overline{\mathbf{q}}_{a,ref}(t_{1,d}) = \overline{\mathbf{q}}_{p,ref}(t_{1,d}) \in \mathbb{R}^n$. These quantities can be used to define the impact joint displacement error $\mathbf{e}_{imp} \in \mathbb{R}^n$ as

$$\mathbf{e}_{imp} = \mathbf{q}_{imp} - \overline{\mathbf{q}}_{ref,imp}. \tag{4.4}$$

Similarly to what was discussed for the classification condition with the impact time, ideally, an impact should be classified as expected when

$$\mathbf{e}_{imp} = \mathbf{0}_{n\times 1}. \tag{4.5}$$

However, similarly to how the conditions (4.1) and (4.2) were formulated, a threshold needs to be used when enforcing (4.5), because this condition is also too strict to be applied to a real robotic manipulator. Therefore, by taking the Euclidean norm of $\mathbf{e}_{imp}$ and setting this against a user-defined threshold, the classification condition for the joint-space impact configuration can be defined as

$$\|\mathbf{e}_{imp}\| \leq \epsilon, \tag{4.6}$$

where $\epsilon \in \mathbb{R}_{>0}$ is the user-defined error threshold. This threshold is tuned based on the amount of noise, the delay in impact detection, and the allowable difference between the measured and expected impact configuration, where the lower each of these are required to be, the lower ϵ is tuned. Condition (4.6) is defined for the single impact case, expanding this to include simultaneous impacts requires redefining the impact joint displacement error as

$$\mathbf{e}_{imp,j} = \mathbf{q}_{imp,j} - \overline{\mathbf{q}}_{ref,imp}, \quad \text{for } j \in [1, j_f)\,, \tag{4.7}$$

where $\mathbf{q}_{imp,j} = \mathbf{q}(t_j) \in \mathbb{R}^n$ are the detected impact joint displacements for the j-th impact. The classification condition for a simultaneous impact case can then be expanded to include the new error formulation presented in (4.7), resulting in

$$\|\mathbf{e}_{imp,j}\| \leq \epsilon, \quad \text{for } j \in [1, j_f)\,. \tag{4.8}$$

The classification conditions presented in this section are relatively straightforward to implement. However, for the classification of a simultaneous impact, it is important to discuss how the classification algorithm needs to be structured. For both the single- and simultaneous impact case, the implementation will be discussed in Sections 4.3 and 4.4, respectively.

4.1.3 Classification of Post-Impact Behavior

For the classification of post-impact behavior, the immediate response of the impact will need to be analyzed and compared to the post-impact trajectory. Consider how the post-impact trajectory is generated through the use of a post-impact velocity prediction in Section 3.4 and what the rigid impact map used for the post-impact velocity entails in Section 2.2.3. Here, at the time of impact, it is assumed that the position remains constant, but an instantaneous change in the velocity takes place, both of which can be verified using the results from Section 3.5. Furthermore, as can be observed in Section 3.5, in the simulated unexpected scenarios, the post-impact position will diverge from the expected position after a period of time. Therefore, to ensure that post-impact classification can occur soon after the impact has occurred, only the joint velocity is considered for the post-impact classification.

Finally, the classification of the impact location and time only relied on the desired impact joint displacements and time, which are unaffected by the differences between the used trajectory and the characteristics of the system that is classified. However, when investigating post-impact behavior classification, the differences between the generated trajectory and the post-impact behavior of the system become more pronounced. As was explained in Section 3.4, the post-impact velocity prediction assumes that the robotic manipulator impacts a rigid surface, furthermore, the subsequently generated trajectory also assumes that the robotic manipulator has rigid joints. However, for a real robotic manipulator, the impact occurs on a compliant surface and the joints are flexible. As shown in Section 3.5, modeling an impact via

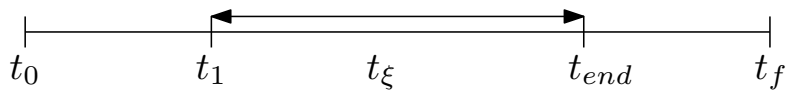


Figure 4.2: Schematic illustration showing the relation between t_0, t_1, t_ξ, t_{end}, and t_f.

a compliant contact model results in a non-instantaneous velocity change. The duration of such a hard robot-surface impact will be approximately 5-10 ms [46, 47], and the impact impulse propagates through the flexible joint manipulator and introduces vibrations in the joints. After the transient phase, which is the period where the vibrations as a consequence of the impact are clearly visible, the steady-state behavior will converge to the generated post-impact trajectory if the expected and real scenario are similar [5]. The duration of the transient phase has been shown to be approximately 100 ms for the Franka Emika Panda and the KUKA LWR IV+ [6, 5]. This transient phase and the duration can also be observed in the figures presented in Section 3.5. In other words, if the post-impact behavior was expected, then for a single impact case, it should satisfy

$$\mathbf{e}_{vel}(t) \to \mathbf{0}_{n\times 1}, \quad \text{for } t \in (t_1, t_{end}], \tag{4.9}$$

where

$$\mathbf{e}_{vel}(t) = \dot{\mathbf{q}}(t) - \overline{\dot{\mathbf{q}}}_{ref}(t). \tag{4.10}$$

Here, $\mathbf{e}_{vel}(t) \in \mathbb{R}^n$ is the error between the measured joint velocity $\dot{\mathbf{q}}(t) \in \mathbb{R}^n$ and the expected joint velocity $\overline{\dot{\mathbf{q}}}_{ref}(t) \in \mathbb{R}^n$. Furthermore, $\overline{\dot{\mathbf{q}}}_{ref}(t)$ is constructed from the extended ante- and post-impact joint velocities, as

$$\overline{\dot{\mathbf{q}}}_{ref}(t) = \begin{cases} \overline{\dot{\mathbf{q}}}_{a,ref}(t), & \text{for } t \in [t_0, t_1], \\ \overline{\dot{\mathbf{q}}}_{p,ref}(t), & \text{for } t \in (t_1, t_f], \end{cases} \tag{4.11}$$

where $t_0 \in [0, \infty)$ is the starting operation time, $t_f \in (t_{end}, +\infty)$ is the final system operation time. Furthermore, in (4.9), $t_{end} \in (t_1, +\infty)$ is the time when the post-impact behavior should have been classified and can be defined based on the requirements. Formally, for a single impact case, this can be defined as

$$t_{end} = t_1 + t_\xi, \tag{4.12}$$

where $t_\xi \in (0, +\infty)$ is the time difference between the detected impact time and final classification time, where if $t \geq t_1 + t_\xi$ and the impact has not been classified as unexpected, then the impact is classified as expected. For clarity, a schematic illustration showing the relation between t_0, t_1, t_ξ, t_{end}, and t_f is shown in Figure 4.2. Finding a suitable value for t_ξ is dependent on the type of system and is chosen based on the interval where the transient phase of the response is clearly present. As was stated earlier, for the Franka Emika Panda and the KUKA LWR IV+ this transient phase takes 100 ms [6, 5].

Expanding these conditions for a simultaneous impact case can be performed in a similar manner to how it was handled in the impact location and time classification conditions. The expected post-impact behavior should satisfy

$$\mathbf{e}_{vel}(t) \to \mathbf{0}_{n\times 1}, \quad \text{for } t \in \left(t_{j_f}, t_{end,j_f}\right], \tag{4.13}$$

where

$$t_{end,j_f} = t_{j_f} + t_\xi, \tag{4.14}$$

where j_f is the final detected impact that signals that contact has been completed. The reason why this method specifically uses the final detected impact j_f is of the uncertainty of the response in the intermediate phase. In other words, due to the uncertainty of the perturbation that results in the loss of simultaneity, the behavior of the intermediate phase is inherently unpredictable. Therefore, for a simultaneous impact case, only after all impacts have been concluded can a post-impact classification be performed.

With the conditions to classify post-impact behavior detailed, a method needs to be found to analyze the impact response of the system. As was discussed earlier, due to the flexible joints, a realistic system suffers from vibrations at the joint level, this is different from the generated post-impact trajectory which assumes that there are rigid joints. A method should be developed to analyze the vibrations to see if the rigid response is similar to the expected post-impact joint velocities. This can be achieved through the use of a signal envelope filter, which is proposed in Section 4.2.

4.2 Signal Envelope Filter

The impact response of a realistic system consists of two components: a rigid component and an underdamped oscillatory component. In [6, 5] and Section 3.5, it was shown that it is possible to use a rigid joint robotic manipulator model undergoing an impact with a rigid surface to obtain the rigid component of the impact response. Furthermore, as was performed in Section 3.4 and shown in Section 3.5, a post-impact velocity prediction can be used to generate a trajectory that matches the rigid component of the impact response. It was also observed that the realistic impact response oscillates around the rigid component. Therefore, to classify the post-impact behavior of a realistic system using the rigid component of an expected impact response a method needs to be developed to generate a region where the rigid response of the realistic system is located. To determine the region where the rigid response of the impact response is located a signal envelope filter is developed.

The goal of the signal envelope filter is to generate an envelope around an underdamped oscillatory impact response that occurs on a flexible joint robotic manipulator. The signal envelope filter is designed to allow for approximating the location of the rigid component of the response. This filter generates a bound that contains the rigid component of the response, and as the oscillatory response is further damped out, the bounds narrow and will approximate the rigid component of the response more closely. This bound can then be used to classify the post-impact behavior using the generated post-impact trajectory which was based on a prediction of the post-impact velocity.

This section focuses on explaining the signal envelope filter, as well as applying it to the system to test its performance and making adjustments as necessary. The filter is explained in detail in Section 4.2.1. This filter is then applied to the joint velocity and the joint velocity error in Sections 4.2.2 and 4.2.3, respectively. In these sections, the filter is used in the context of post-impact classification, showing its effectiveness. For simplicity, the classification performed in this section focuses on the single impact case, the simultaneous impact case will be expanded on in Section 4.4.

4.2.1 Proposed Filter

The goal of the proposed filter is to generate an envelope around an oscillatory signal to approximate where the rigid component of the response is located. The concept of the signal envelope filter applied to an underdamped oscillatory signal can be seen in Figure 4.3. In Figure 4.3, the signal envelope consists of a region defined by an upper- and lower bound, which envelopes the oscillatory response and for an expected scenario also envelopes the expected rigid response, which takes the form of the generated post-impact trajectory defined in Section 3.4. To implement the concept found in Figure 4.3 for a real system, a set of bounds need to be defined using the measurement data of an oscillatory response. The first step to defining these bounds is by finding the extrema of the oscillatory response over a moving window length m. Here, the filter uses the previous m timesteps to determine the extrema over the measured response. This can be formulated as

$$\begin{aligned} \mathbf{x}_{min,k} = &\quad \min\left\{\mathbf{x}_{k-m}, \ldots, \mathbf{x}_{k-j}, \ldots, \mathbf{x}_k\right\}, \\ \mathbf{x}_{max,k} = &\quad \max\left\{\mathbf{x}_{k-m}, \ldots, \mathbf{x}_{k-j}, \ldots, \mathbf{x}_k\right\}, \end{aligned} \tag{4.15}$$

where $\mathbf{x}_{min,k} \in \mathbb{R}^n$ and $\mathbf{x}_{max,k} \in \mathbb{R}^n$ are defined as a vector containing the minimum and maximum values of a measured response $\mathbf{x} \in \mathbb{R}^n$ at timestep k, determined from data from timestep $k-m$ to timestep k, every entry in vectors $\mathbf{x}_{min,k}$ and $\mathbf{x}_{max,k}$ contains the minimum or maximum value of that entry in $\mathbf{x}$ over the designated window length. To ensure that the generated bounds more closely follows the signal, a decaying bound is formulated using the calculated minima and maxima. The decaying function can be

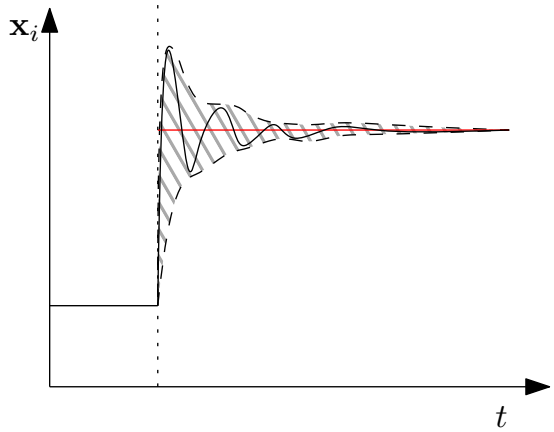
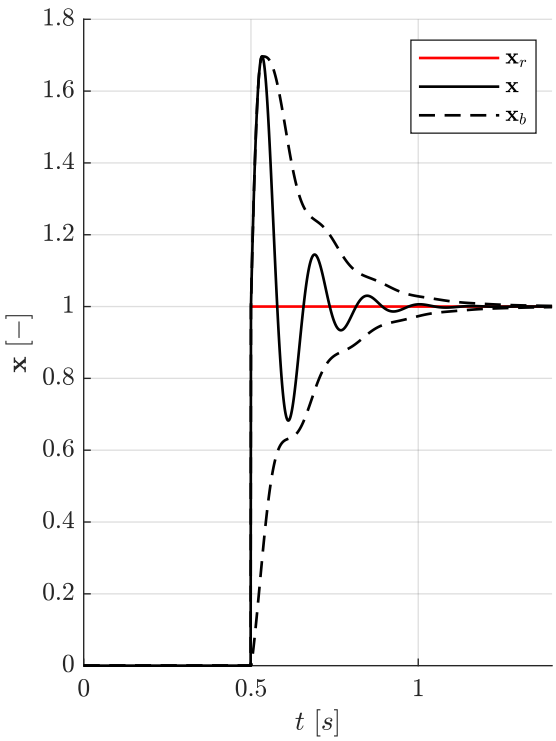


Figure 4.3: Concept of the proposed signal envelope filter. Here, the solid black line represents the measured oscillatory response, the red line represent the rigid component of the response, and the dashed lines enclose a region that envelop the measured response, which has been hatched with dark gray lines.



(a)

Tuning Parameter	Symbol	Value	Unit
Window Length	m	3	−
Decay Rate	a	7	$1/s$

(b)

Figure 4.4: Signal envelope filter applied to an underdamped sinusoidal oscillation. In (a), the figure showing the functioning of the signal envelope filter where the solid black line represents the measured signal, the red line the rigid component of the measured signal, and the dashed black line represents the generated signal envelope. In (b), the parameters used to obtain the results found in (a).

derived from the equation

$$\frac{\Delta \mathbf{x}_b}{\Delta t} = a\left(\mathbf{x} - \mathbf{x}_b\right), \tag{4.16}$$

where $a \in \mathbb{R}_{>0}$ is the decaying constant and $\mathbf{x}_b \in \mathbb{R}^n$ is the current value of the decaying function. The measured signal $\mathbf{x}$ is used as an offset to ensure that the bound more closely follows the shape of the measured signal. Incorporating (4.15) into (4.16) results in the general expression for a decaying bound

$$\frac{\Delta \mathbf{x}_{b,k}}{\Delta t} = a\left(\mathbf{x}_{ex,k} - \mathbf{x}_{b,k-1}\right). \tag{4.17}$$

This results in

$$\mathbf{x}_{b,k} - \mathbf{x}_{b,k-1} = a\Delta t\left(\mathbf{x}_{ex,k} - \mathbf{x}_{b,k-1}\right), \tag{4.18}$$

or equivalently

$$\mathbf{x}_{b,k} = \left(1 - a\Delta t\right)\mathbf{x}_{b,k-1} + a\Delta t\,\mathbf{x}_{ex,k} \tag{4.19}$$

where $\mathbf{x}_{b,k} \in \mathbb{R}^n$ is the bound at timestep k, $\mathbf{x}_{ex,k} \in \mathbb{R}^n$ are the extrema found using (4.15), and can represent either the minima or maxima, and $\Delta t = t_k - t_{k-1}$ is the sampling time for measuring the signal. The function described in (4.19) allows for the gradual decay of the bound as the measured oscillatory response converges to the rigid response. However, this does not account for a sudden increase or decrease of the measured signal which requires that the bound increases in size. To achieve this, the function found in (4.19) is compared to the parameter found in (4.15), and either the minimum or maximum of these parameters is chosen for the bound at timestep k. This results in the full expression for the bounds

$$\begin{aligned} \mathbf{x}_{lb,k} &= \min\left\{\mathbf{x}_{min,k}, \left(1 - a\Delta t\right)\mathbf{x}_{lb,k-1} + a\Delta t\,\mathbf{x}_{min,k}\right\}, \\ \mathbf{x}_{ub,k} &= \max\left\{\mathbf{x}_{max,k}, \left(1 - a\Delta t\right)\mathbf{x}_{ub,k-1} + a\Delta t\,\mathbf{x}_{max,k}\right\}, \end{aligned} \tag{4.20}$$

where the $\mathbf{x}_{lb} \in \mathbb{R}^n$ and $\mathbf{x}_{ub} \in \mathbb{R}^n$ for the lower- and upper bound of the envelope, respectively. Using this filter, an arbitrary oscillatory signal $\mathbf{x}$ can be filtered to produce an envelope that encompasses the signal. When applying the filter presented in this section to a step response with an underdamped sinusoidal oscillation, the results and the parameters used to obtain the results are shown in Figure 4.4 can be obtained. As can be seen in the figure, the filter works as was originally shown in the proposal presented in Figure 4.3, where a lower- and upper bound are defined that encloses the measured oscillatory signal and the rigid component.

4.2.2 Filtering the Joint Velocities

With the signal envelope filter defined, the next step is applying the filter to a realistic system to test its performance. Recall that in the introduction of this section and in Section 4.1.3, it was described that the response of a realistic system is expected to oscillate around the rigid component of the response, which was shown to occur when investigating the results in Section 3.5 and was also investigated in [5]. Recall also that in Section 3.4 it was noted that to obtain the post-impact trajectory, it was assumed that the robotic manipulator has rigid joints and that it undergoes a rigid impact on the environment. Furthermore, in Section 3.5, it was shown that the post-impact trajectory matched the post-impact response of model C, which modeled a rigid joint robotic manipulator undergoing a rigid impact with the environment. Therefore, by applying the filter as described in Section 4.2.1, to the response of model A, which modeled a flexible joint robotic manipulator with compliant contact dynamics and was used as a realistic model of a robotic manipulator in Section 3.5, the generated envelope is expected to encompass the post-impact trajectory in an expected scenario.

With this information, a classifier can be proposed based on the joint velocity measurement and the signal envelope filter. If the generated signal envelope obtained from the signal envelope filter encompasses the reference post-impact joint velocities, then an expected post-impact scenario has occurred. This proposed classifier can be seen in Figure 4.5, which shows a schematic illustration of the filter applied to an expected and unexpected post-impact scenario. As can be seen in Figure 4.5, the velocity signals are processed via the envelope filter to define the region that is expected to contain the generated post-impact trajectory for an expected scenario. With the signal envelope filter and the reference post-impact joint velocities, the measured joint velocities can be filtered to see if the real scenario went as expected. The filter is reformulated using the relevant parameters and is now defined as

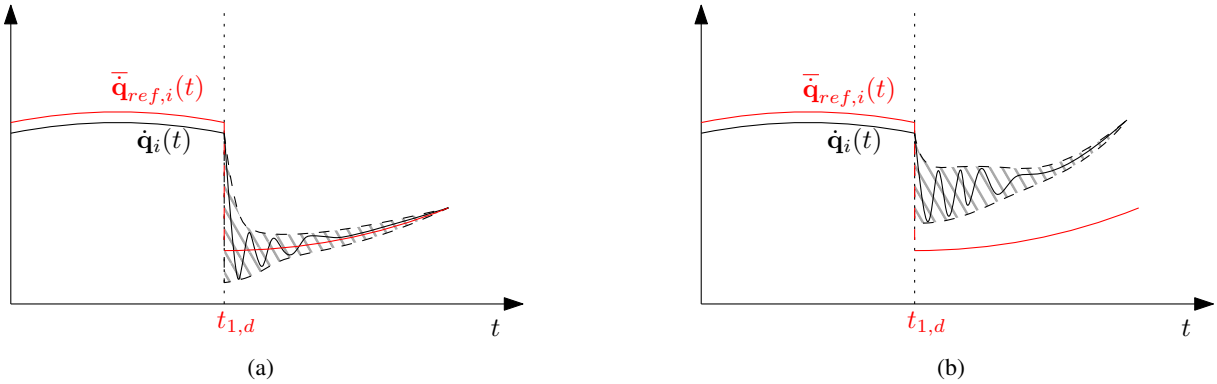
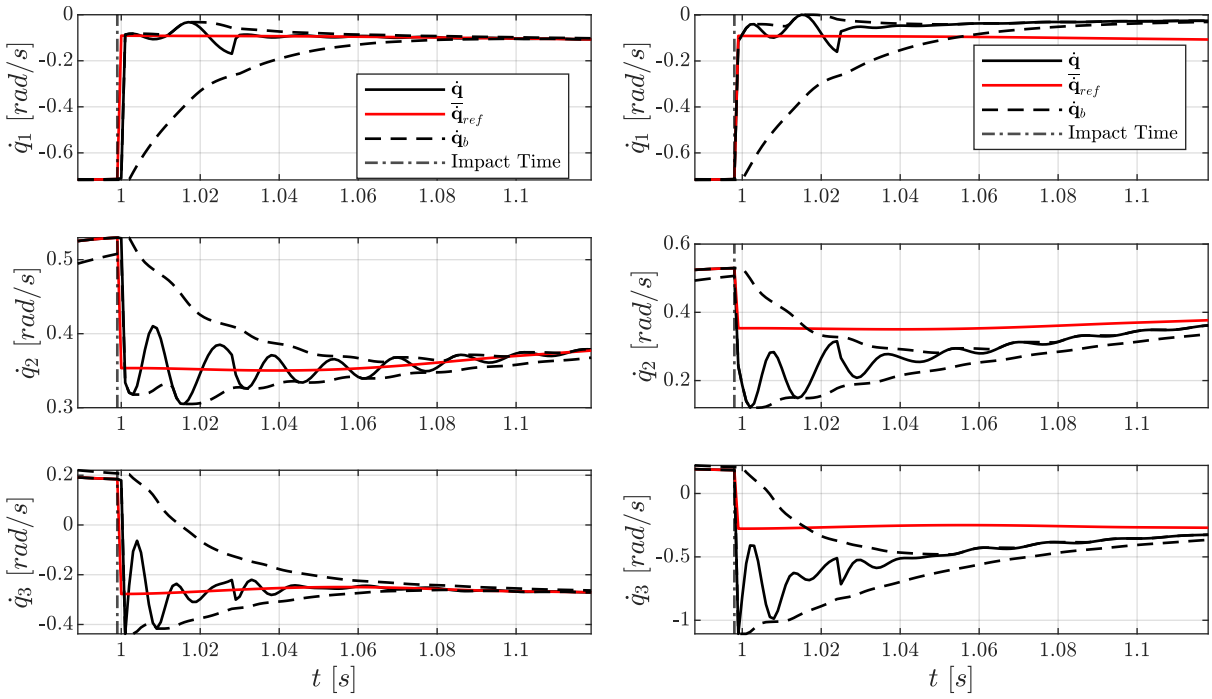


Figure 4.5: Preliminary proposed classification method by using the signal envelope filter to filter the joint velocities. Here, the solid black line $\dot{\mathbf{q}}_i$ is the measured joint velocity of the i-th joint, the red line $\bar{\dot{\mathbf{q}}}_{ref,i}$ represents the generated trajectory as defined in (4.11) and serves as the expected rigid component of the response, and the hatched dark gray region is the envelope of the joint velocities defined by the lower- and upper bound generated by signal envelope filter. In (a), the envelope filter is applied to an expected scenario. In (b), the envelope filter is applied to an unexpected scenario.



Tuning Parameter	Symbol	Value	Unit
Window Length	m	3	–
Decay Rate	a	50	$1/s$

(c)

Figure 4.6: Signal envelope filter applied to the joint velocity. In (a), applied to the joint velocity of an expected scenario where $\beta = 0\ deg$. In (b), applied to the joint velocity of an unexpected scenario where $\beta = 10\ deg$. In (c), the parameters used of the signal envelope filter. Note that in both (a) and (b), (4.23) is not satisfied briefly after the impact occurs.

$$\begin{aligned} \dot{\mathbf{q}}_{min,k} &= \min\left\{\dot{\mathbf{q}}_{k-m}, \ldots, \dot{\mathbf{q}}_{k-j}, \ldots, \dot{\mathbf{q}}_{k}\right\}, \\ \dot{\mathbf{q}}_{max,k} &= \max\left\{\dot{\mathbf{q}}_{k-m}, \ldots, \dot{\mathbf{q}}_{k-j}, \ldots, \dot{\mathbf{q}}_{k}\right\}, \end{aligned} \tag{4.21}$$

$$\begin{aligned} \dot{\mathbf{q}}_{lb,k} &= \min\left\{\dot{\mathbf{q}}_{min,k}, (1-a\Delta t)\,\dot{\mathbf{q}}_{lb,k-1} + a\Delta t\,\dot{\mathbf{q}}_{min,k}\right\}, \\ \dot{\mathbf{q}}_{ub,k} &= \max\left\{\dot{\mathbf{q}}_{max,k}, (1-a\Delta t)\,\dot{\mathbf{q}}_{ub,k-1} + a\Delta t\,\dot{\mathbf{q}}_{max,k}\right\}, \end{aligned} \tag{4.22}$$

where $\dot{\mathbf{q}}_{min,k} \in \mathbb{R}^n$ and $\dot{\mathbf{q}}_{max,k} \in \mathbb{R}^n$ are the minimum and maximum joint velocities over window length m, respectively, and $\dot{\mathbf{q}}_{lb} \in \mathbb{R}^n$ and $\dot{\mathbf{q}}_{ub} \in \mathbb{R}^n$ are the lower- and upper bound of the envelope around the measured joint velocities, respectively. Using these bounds, a classifier can be formulated such that an expected post-impact scenario has occurred if

$$\dot{\mathbf{q}}_{lb}(t) \leq \overline{\dot{\mathbf{q}}}_{ref}(t) \leq \dot{\mathbf{q}}_{ub}(t), \quad \forall t \in (t_1, t_{end}]\,. \tag{4.23}$$

This results in the plots shown in Figure 4.6, which show an expected and unexpected post-impact scenario where the impact surface was rotated by $\beta = 10\ deg$. Interestingly and counter-intuitively, neither of the scenarios presented in Figure 4.6 result in an expected post-impact classification with this proposed method, as the post-impact behavior is classified as unexpected immediately after the impact occurs. While this might seem contradictory to the statements made at the start of this section, the reason for this behavior is due to the nature of the type of impact. Recall that, in Section 4.1.3, it was described that a hard robot-surface impact has a duration of approximately 5-10 ms [46, 47], as well as that the predicted post-impact joint velocities are determined using the rigid impact map. Furthermore, to ensure that the envelope generated by the filter encompasses the measured signal, the extrema over the window length are compared to the decaying bound and will take precedence if they surpass the decaying bound. However, the signal envelope filter can only generate a bound after a measurement has been analyzed, it cannot anticipate post-impact behavior. Therefore, by combining these observations with Figure 4.6, it is clear that the problem occurs when the expected behavior switches to the generated post-impact trajectory at the time of impact and the measured joint velocity only gradually changes to the post-impact velocity over a short time interval. Considering that the signal envelope filter cannot anticipate the post-impact behavior, it takes several timesteps until the generated envelope encompasses the post-impact trajectory. During the time where the envelope does not encompass $\overline{\dot{\mathbf{q}}}_{ref}(t)$, (4.23) is not satisfied and the post-impact behavior is classified as unexpected.

There are two approaches to solving this problem, the most straightforward choice being to introduce a delay in the classification. For this solution, a user-defined delay in the classification can be introduced after the impact has occurred that can ensure that the joint velocity has time to converge to the rigid component of the impact response before classification is performed. However, this solution would result in the decreased performance of the classifier, which is something that goes against one of the preferences set in Section 1.2, which states that the classifier should classify an unexpected post-impact scenario as soon as possible. Furthermore, introducing an additional tuning parameter to solve a problem can complicate tuning for a real system and should therefore be avoided if possible. The second solution to this problem is reformulating the filtered signal. The desire of reformulating the filtered signal is to ensure that the signal that the envelope filter is applied to always oscillates around the same value for an expected scenario. This is explained further in the following section.

4.2.3 Filter Joint Velocity Error

As described in Section 4.2.2, a problem with filtering the measured joint velocities is that at the time of impact for an expected scenario, the post-impact behavior is always classified as unexpected. This section seeks to provide a solution to this problem by reformulating the needed filtered signal for a correct classification.

First of all, by using the same set of information used for the classifier shown in Section 4.2.2, a new signal can be created. This signal is known as the joint velocity error (see Section 4.1.3, equation (4.10)). For ease of readability, we recall that this error is defined as

$$\mathbf{e}_{vel}(t) = \dot{\mathbf{q}}(t) - \overline{\dot{\mathbf{q}}}_{ref}(t) \tag{4.24}$$

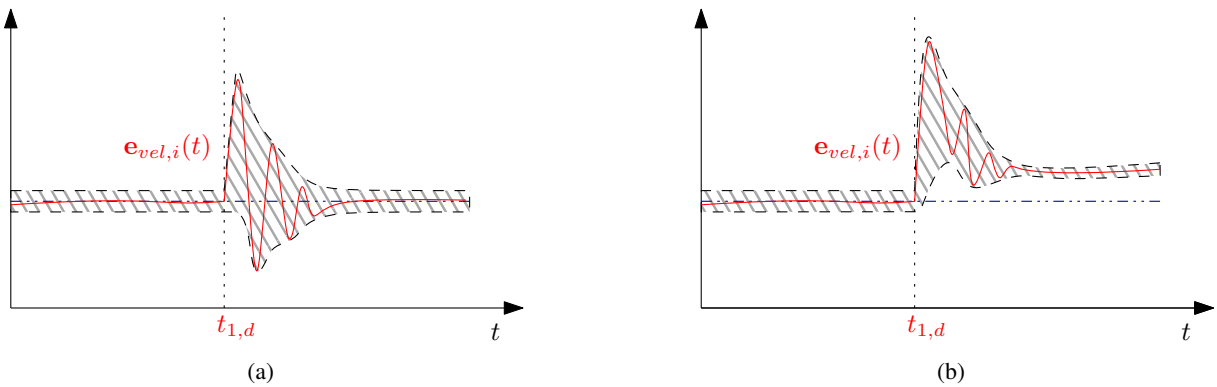
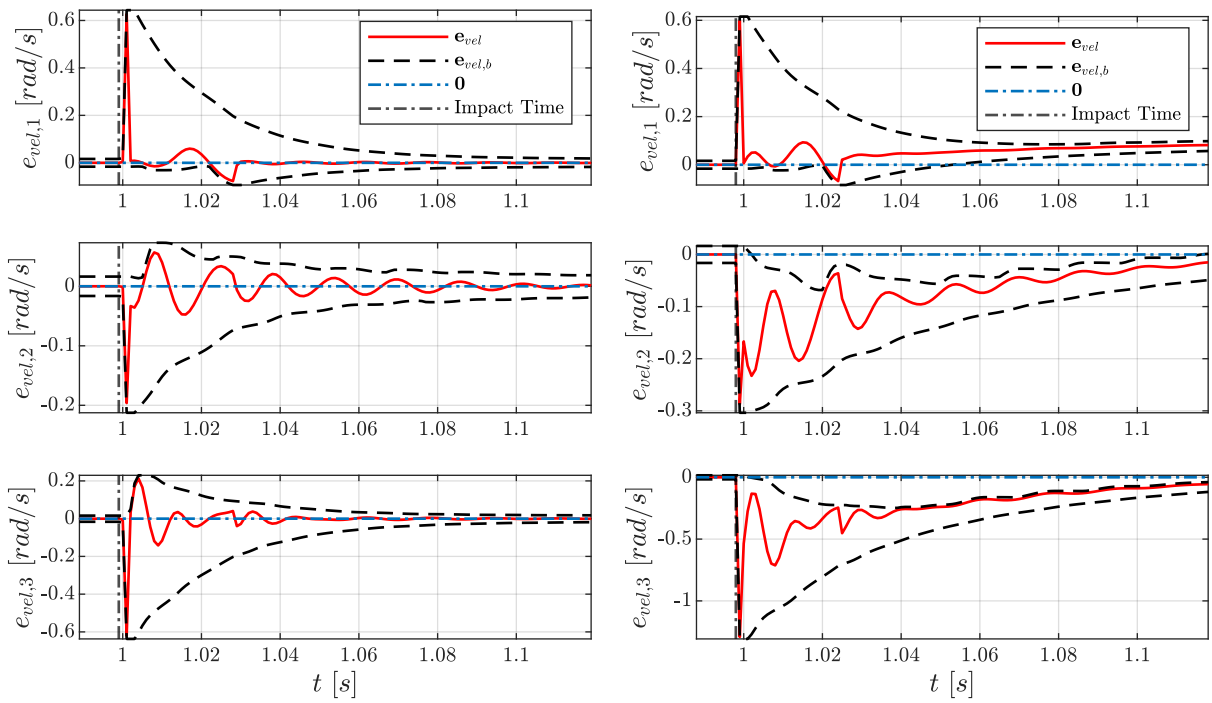


Figure 4.7: Newly proposed classification method by using the signal envelope filter to filter the joint errors. This classification method is to be compared against the original method shown in Figure 4.5. Here, the red line $\mathbf{e}_{ref,i}$ is the joint velocity error of the i-th joint as defined in (4.24), the dashed-dotted blue line represents 0, and the hatched dark gray region is the envelope of the joint velocity error defined by the lower- and upper bound generated by signal envelope filter. In (a), the envelope filter is applied to an expected scenario. In (b), the envelope filter is applied to an unexpected scenario.



Tuning Parameter	Symbol	Value	Unit
Window Length	m	3	–
Decay Rate	a	50	$1/s$
Envelope Margin	e_η	0.01	rad

(c)

Figure 4.8: Signal envelope filter applied to the joint velocity error. In (a), applied to the joint velocity error of an expected scenario where $\beta = 0\ deg$ where it satisfies (4.27). In (b), applied to the joint velocity error of an unexpected scenario where $\beta = 10\ deg$ where it violates (4.27). In (c), the parameters used of the signal envelope filter.

By using this joint velocity error as the input for the envelope filter of Section 4.2.2, the post-impact velocity response and generated signal bounds will converge to $\mathbf{0}_{n\times 1}$ for an expected scenario. Furthermore, for a well-controlled system $\mathbf{e}_{vel} \approx \mathbf{0}_{n\times 1}$ in the ante-impact phase and the signal envelope filter will ensure that the bound is held constant for a total of m timesteps before decaying if a sudden change in the measured signal occurs. Therefore, after an impact occurs the generated signal bound will encompass $\mathbf{0}_{n\times 1}$ for at least m timesteps. This gives the post-impact velocity response time to converge to $\mathbf{0}_{n\times 1}$. By combining this adjustment to the input signal of the envelope filter with a new classification condition where for an expected scenario the generated bounds should encompass $\mathbf{0}_{n\times 1}$, the problem of incorrect unexpected post-impact classification can be avoided. A schematic illustration of this concept is shown in Figure 4.7, where the signal envelope encompasses $\mathbf{0}_{n\times 1}$ for an expected scenario. Using the joint velocity error as the input to the signal envelope filter results in the formulation of the envelope filter as

$$\begin{aligned} \mathbf{e}_{vel,min,k} &= \min\left\{\mathbf{e}_{vel,k-m},\ldots,\mathbf{e}_{vel,k-j},\ldots,\mathbf{e}_{vel,k}\right\}, \\ \mathbf{e}_{vel,max,k} &= \max\left\{\mathbf{e}_{vel,k-m},\ldots,\mathbf{e}_{vel,k-j},\ldots,\mathbf{e}_{vel,k}\right\}, \end{aligned} \tag{4.25}$$

$$\begin{aligned} \mathbf{e}_{vel,lb,k} &= \min\left\{\mathbf{e}_{vel,min,k}, (1-a\Delta t)\,\mathbf{e}_{vel,lb,k-1} + a\Delta t\,\mathbf{e}_{vel,min,k}\right\}, \\ \mathbf{e}_{vel,ub,k} &= \max\left\{\mathbf{e}_{vel,max,k}, (1-a\Delta t)\,\mathbf{e}_{vel,ub,k-1} + a\Delta t\,\mathbf{e}_{vel,max,k}\right\}, \end{aligned} \tag{4.26}$$

where $\mathbf{e}_{vel,min,k} \in \mathbb{R}^n$ and $\mathbf{e}_{vel,max,k} \in \mathbb{R}^n$ are the minimum and maximum joint velocity errors over window length m, respectively, and $\mathbf{e}_{vel,lb} \in \mathbb{R}^n$ and $\mathbf{e}_{vel,ub} \in \mathbb{R}^n$ are the lower- and upper bound of the envelope around the joint velocity error, respectively. An additional change to the originally proposed classifier is made to take into account potential noise, the steady-state tracking error, and disturbances that should not result in an unexpected post-impact behavior classification. This change is the introduction of a signal envelope margin $e_\eta \in \mathbb{R}_{\geq 0}$, which is a user-defined parameter that can be changed depending on the noise level, the tracking error, and other user-defined requirements. This results in a new classification condition where the post-impact scenario is labeled as expected if

$$\mathbf{e}_{vel,lb}(t) - e_\eta \leq \mathbf{0}_{n\times 1} \leq \mathbf{e}_{vel,ub}(t) + e_\eta, \quad \forall t \in (t_1, t_{end}]\,. \tag{4.27}$$

This classification condition can then be implemented and applied to the same scenario used to test the classifier in Section 4.2.2. This results in the plots shown in Figure 4.8. As can be seen in the plots, the classification of expected and unexpected scenarios works as desired; the expected post-impact scenario is classified as expected because it always satisfies (4.27) and the unexpected post-impact scenario is classified as unexpected because it violates (4.27). This shows the effectiveness of the signal envelope filter for classification purposes when it is applied to a realistic post-impact case.

4.3 Single Impact Classifier

Combining the approaches presented in Sections 4.1.2 and 4.2.3, a classifier can be constructed that can classify the impact time and location and post-impact velocity for single impact cases. The prerequisites for the single impact classifier are: (i) the ability to detect impacts, (ii) joint displacement measurements, (iii) joint velocity measurements or estimations, (iv) the desired link configuration at the time of impact, (v) the desired impact time, and (vi) the expected post-impact velocity. If the system is controlled through a joint-space configuration task, then the expected and reference post-impact velocity are equivalent.

A schematic implementation of a single loop of the single impact classifier can be found in Algorithm 1, this loop is repeated for every timestep. This implementation always has the signal envelope filter active as defined in (4.25) and (4.26), but the generated envelope will only be used for classification after an impact has been detected. Furthermore, besides an expected and unexpected classification, an unknown classification is included to ensure that the classifier knows whether the behavior has already been classified. When an impact has been detected the impact location and time will be classified first by using (4.6) and (4.1). Additionally, when the impact location and time is unexpected, this implies that the post-impact behavior is also unexpected. Therefore, the classifier will immediately classify the post-impact behavior as unexpected if the impact time and location are classified as unexpected. If the impact location and time were classified as expected, then (4.27) is applied for the post-impact classification. If during $t \in [t_1, t_1 + t_\xi]$, (4.27) is violated, the post-impact behavior is classified as unexpected, otherwise it is classified as expected.

Algorithm 1: Single impact classification algorithm

```
input     : The joint velocity error e_vel from timesteps k − m to k.
            Current timestep k.
            Current time t_k.
            Sample time Δt.
            Signal envelope for previous timestep, e_vel,ub,k−1 and e_vel,lb,k−1.
            The current impact location and time classification result, Impact.
            The current post-impact behavior classification result, Post-Impact.
            Impact detection Status.
            If an impact has been detected, the impact configuration error e_imp and time t_1.
output    : Updated impact location and time classification result, Impact.
            Updated post-impact behavior classification result, Post-Impact.
            Signal envelope for current timestep, e_vel,ub,k and e_vel,lb,k.
parameter: Window length m.
            Signal envelope decay rate a.
            Maximum impact configuration error ϵ.
            Permissible time difference between the desired and detected impact time ζ.
            Signal envelope margin e_η.
            Post-impact classification time t_ξ.

e_vel,min,k = min {e_vel,k−m, ..., e_vel,k−j, ... e_vel,k};
e_vel,max,k = max {e_vel,k−m, ..., e_vel,k−j, ..., e_vel,k};
e_vel,lb,k = min {e_vel,min,k, (1 − aΔt) e_vel,lb,k−1 + aΔt e_vel,min,k};
e_vel,ub,k = max {e_vel,max,k, (1 − aΔt) e_vel,ub,k−1 + aΔt e_vel,max,k};
if Impact detected then
    if Impact = Unknown then
        if ‖e_imp‖ > ϵ or ‖t_1 − t_1,d‖ > ζ then
            Impact = Unexpected;
            Post-Impact = Unexpected;
        else
            Impact = Expected;
            Post-Impact = Unknown;
        end
    end
    if Post-Impact = Unknown then
        if e_vel,lb,k − e_η > 0_n×1 or e_vel,ub(t) + e_η < 0_n×1 then
            Post-impact behavior = Unexpected;
        else
            if t_k − t_1 > t_ξ then
                Post-impact behavior = Expected;
            end
        end
    end
end
```

There are some remarks about this classifier that need to be made. First of all, it was described in Section 1.2 that the classifier should classify post-impact behavior as soon as possible, however, there is always a fixed time delay in the post-impact classification. This is due to how the filter operates in combination with the user-defined window length. The earliest possible classification time is after the first m timesteps following an impact and the theoretical minimum of post-impact behavior classification time is $m\Delta t$ s after the impact occurred. In practice, formulating the input signal of the signal envelope filter as the error shows similar behavior to the proposed artificial delay of the classification time discussed in Section 4.2.2. However, the current implementation of the classifier provides a cleaner solution than the discussed artificial delay.

The second remark is regarding the reasoning of why for the post-impact behavior classification, each

joint is analyzed separately rather than analyzing the combined response as is done with the impact location classification defined in (4.6). Currently, the classifier will use each joint velocity error in the classification, meaning that if only one joint velocity error deviates too far from 0, then the post-impact behavior is classified as unexpected. The alternative would be using the Euclidean norm on the error $\mathbf{e}_{vel}(t)$, however, this would not work as the Euclidean norm is always non-negative, and applying the filter would result in a bias in the positive direction. Furthermore, when an impact occurs, this error will peak in the positive direction, risking the bounds no longer encompassing 0 regardless of whether the impact scenario was expected or not. Furthermore, a benefit of filtering each joint velocity separately is that the performance of the classifier also improves, as only one deviation results in the classification of an unexpected post-impact scenario.

The final remark concerns the user-defined tuning parameters for post-impact classification, i.e., the window length m, the decay rate a, and the signal envelope margin e_η. These parameters are dependent on the requirements set by the user and systems dynamics and can be tuned independently from each other. The precision of tuning these parameters affects the performance of the classifier, where leniently tuned parameters will result in slower classifications and tuning the boundary between expected and unexpected classifications. Whereas when parameters are tuned too strictly will result in faster classifications and false-positive unexpected classifications. Furthermore, expected and unexpected scenarios are dependent on the application, however, the limits of what can be classified are dependent on the noise level. For example, assuming that the tuned parameters allow for accurate classification of a set of scenarios, then depending on the level of noise, unexpected scenarios that are very similar to the expected scenario will also be classified as expected. The effect of noise on the accuracy of the classifier is something that needs to be considered when implementing this classification method on a system where high precision is desirable, i.e., noise levels should be kept to a minimum for high precision systems that require a classification method such as what is presented in this section.

4.3.1 Tuning the Classifier

With the classifier described in Section 4.3, the different tuning parameters are explained and how they affect the classification. We recall that these tuning parameters are (i) the moving window length m, (ii) the bound decay rate a, (iii) the signal envelope margin e_η, (iv) the maximum impact configuration error ϵ, (v) the post-impact classification time t_ξ, and (vi) the permissible time difference between the desired and detected impact time ζ. Each of these parameters have their own purpose in the classification, and can be tuned separately from each other to classify scenario as expected or unexpected based on the user requirements.

Before tuning each classification parameter, the scenario for which the classifier will be tuned needs to be discussed. Several considerations need to be taken into account when tuning, such as the desired impact velocity v_{imp}, impact approach angle α, and impact end-effector position and orientation ${}^{A}\mathbf{p}_{imp}$. Furthermore, the post-impact classification can be tuned to consider a range of $\beta \in [\beta_{lb}, \beta_{ub}]$ that should be classified as expected. The classifier can be tuned for a range of these parameters, however, the more parameters that need to be taken into account, the worse the performance of the classifier will be.

Tuning a classifier for every combination of the above-mentioned scenario parameters will result in a classifier that has been tuned for a set of scenarios that is too broad. When the classifier is tuned for a broad set of scenarios it causes a problem where the gray area, which is a region where the unexpected impact scenarios are not guaranteed to be classified as expected or unexpected, is significantly larger than for a classifier tuned for a focused set of scenarios. Therefore, before tuning the classifier, as many fixed parameters as possible need to be selected to improve the classification performance. For example, because impacting the environment with an end-effector that is not perpendicular to the environment results in a large change in the joint velocity upon impact, the end-effector should be oriented perpendicular to the environment when contact is made. This parameter can also be controlled directly and can therefore be taken into account when using a tuned classifier on the real system. Similar approaches should be taken for other parameters, the fewer scenarios the classifier needs to be tuned for, the better the performance will be.

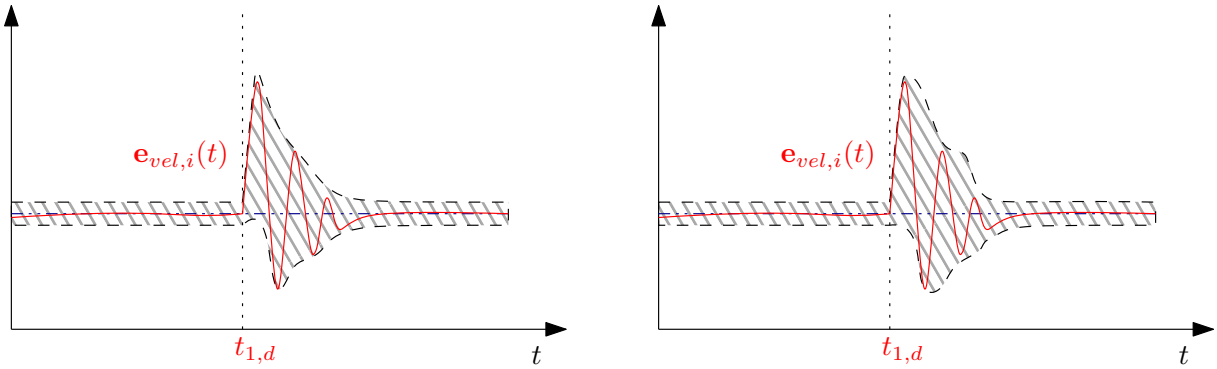


Figure 4.9: Schematic illustrating the effect of a lower window length m (left) and a higher window length m (right) on the generated signal bounds

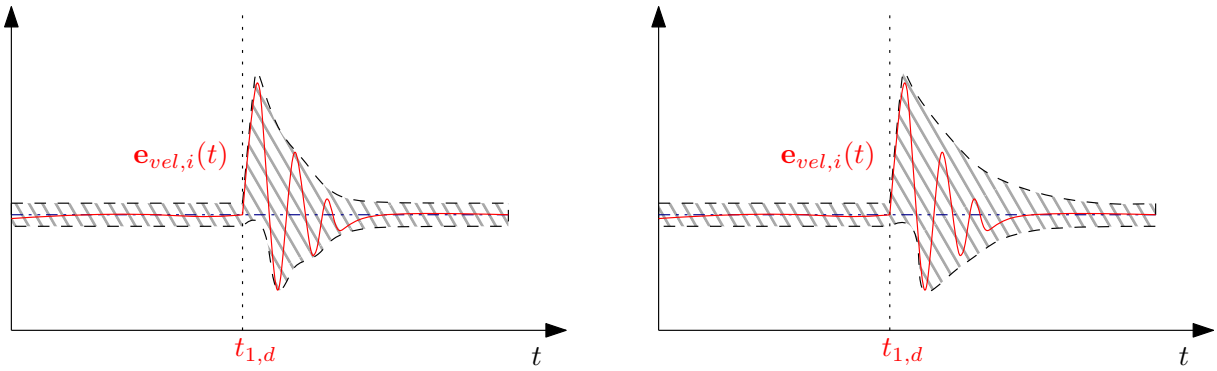


Figure 4.10: Schematic illustrating the effect of a high decay rate a (left) and a low decay rate a (right) on the generated signal bounds

Effect of the Window Length m on the Classification

The moving window length m, as originally defined in Section 4.2.1, dictates the range of timesteps where the minimum or maximum values will be chosen in the first step of the filter. This allows for holding the bound at a certain minimum or maximum even if there is a sudden change in $\mathbf{e}_{vel}(t)$ in the opposite direction. This serves two distinct purposes, the first is to counteract the discontinuity in $\mathbf{e}_{vel}(t)$ that occurs due to the difference between the time of impact and the time when the impact is detected. Furthermore, the window length also serves the purpose of ensuring that the filter does not generate a signal envelope too close behind the measured signal during the transient phase. Instead, for higher values of m the signal envelope will lag behind the measured signal more than for lower values of m. Which would otherwise result in the signal envelope not containing the rigid component of the response. An example of the effect of changing the window length can be seen in the schematic illustration presented in Figure 4.9.

Effect of the Signal Envelope Decay Rate a on the Classification

The decay rate of the signal bound generated by the filter a, as originally defined in Section 4.2.1, is a parameter that dictates how closely the bound follows the filtered signal. This parameter should be tuned to ensure that it closely follows the decaying optima of the transient response. The higher the decay rate is chosen, the closer it follows the measured signal. An example of changing the decay rate is presented in Figure 4.10.

When considering scenarios that are similar to the expected scenario $\beta = 0$ that should also be classified as expected, then the tuning of the decay rate should be approached differently. For scenarios where $\beta \in [\beta_{lb}, \beta_{ub}]$, the immediate post-impact behavior can differ greatly from the prediction while still converging to roughly the same behavior as for the originally expected scenario. To solve the problem of classifying these scenarios, the decay rate should be tuned to be lower, the larger the deviation of β_{lb} and

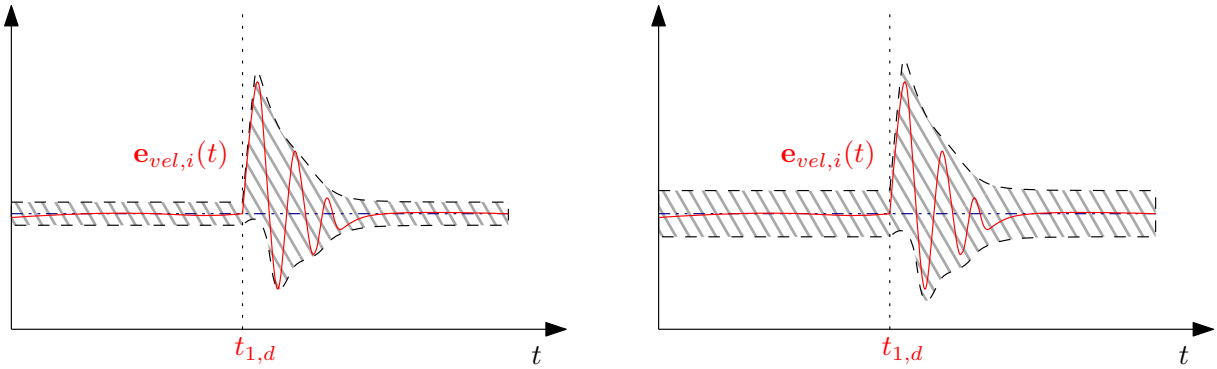


Figure 4.11: Schematic illustrating the effect of a lower margin e_η (left) and a higher margin e_η (right) on the generated signal bounds

β_{ub} from $\beta = 0$, the lower the decay rate should be when compared to how it was tuned just for $\beta = 0$.

Effect of the Signal Envelope Margin e_η on the Classification

The signal envelope margin e_η, which was originally introduced in Section 4.2.3, is implemented to account for any small deviations from the expected joint trajectory. For scenarios such as $\beta = 0$, the parameter can be tuned to be relatively small, as the measured behavior should match the expected behavior closely. However, when the range of scenarios that should be classified as expected is extended to $\beta \in [\beta_{lb}, \beta_{ub}]$, then the margin of error can be used to contain the deviations from the expected trajectory. This parameter serves a similar function to the decay rate a, these parameters complement each other and should be tuned together to ensure that the tuning is as close to the expected behavior as possible. The effect of increasing the margin is shown in the schematic presented in Figure 4.11.

Effect of the Permissible Impact Configuration Error ϵ on the Classification

The permissible impact configuration error ϵ, as originally defined in Section 4.1.2, is the main contributor for unexpected impact classification. The tuning of the parameter is influenced by two components of the system. The first and lesser influence on the permissible impact configuration error is the difference between the desired trajectory and the followed trajectory. As explained in Section 4.1.2, a small deviation between these trajectories will always be present and cannot be avoided, resulting in the need that $\epsilon > 0$. The second and greater influence is the delay between the impact and when it is detected. Tuning this is mostly trial and error, however, a general rule of thumb is to consider how much the end-effector will move in the time between when an impact occurs and when it is detected, and using this to determine the permissible impact configuration error.

Effect of the Post-Impact Classification Time t_ξ and Permissible Impact Time Error ζ on the Classification

The expected post-impact classification time t_ξ, originally defined in Section 4.1.3, and permissible time difference between the desired and detected impact time ζ, as originally defined in Section 4.1.2, are mostly based on the requirements set by the user. A general recommendation for tuning for t_ξ is to let it be at least the length of the transient phase of a response for the given system. Furthermore, ζ should be tuned to be larger than the average impact detection time delay.

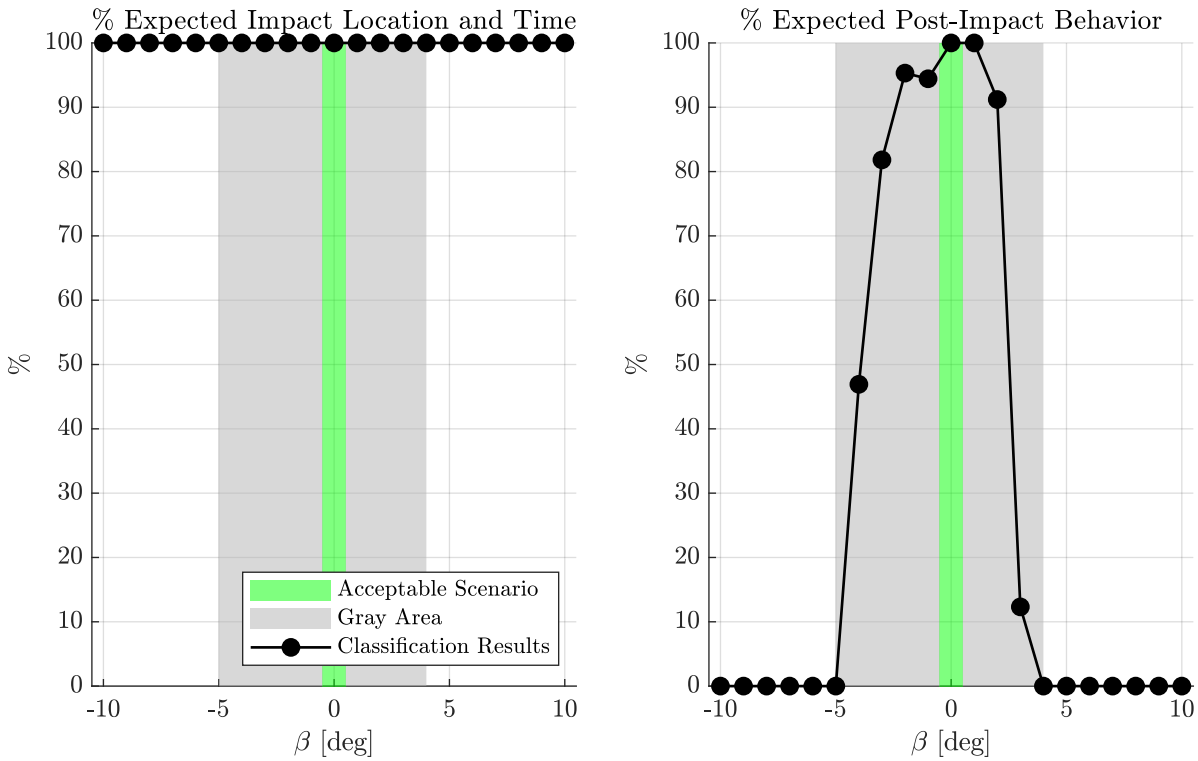
4.3.2 Testing the Single Impact Classifier

With the single impact classifier fully defined and an explanation regarding the tuning of the parameters given, the classifier is tuned and tested for a set of scenarios to test its performance. For these tests, the general parameters found in Table 3.2 are used. However, the approach angle α and desired impact location along the $\vec{x}_A$ axis $\left({}^A\mathbf{p}_{imp}\right)_x$ will vary over a set interval of parameters, which can be found in Table 4.1. This allows for testing the performance of the classifier by simulating each combination of these parameters.

Table 4.1: Table containing the different parameters used to test the classifier performance

Parameter	Lowest Value	Interval	Highest Value	Unit
α	60	2	120	deg
$\left({}^{A}\mathbf{p}_{imp}\right)_x$	0.2	0.02	0.4	m

To test the tuning capabilities of the classifier, three sets of simulations are performed which are based on requirements set by an imaginary user. These requirements pertain to the deviation from the expected environment that should be classified as an expected impact scenario. This leads to the distinction between two different terms. The first term is the *expected environment*, which pertains to the expectation of the environment used to obtain the prediction of the post-impact velocity and subsequently generated trajectory. In the case discussed in this project, the expected environment is an environment where $\beta = 0$ and $l = 0$. The second term is the *acceptable scenario*, which is the set of environments $\beta \in [\beta_{lb}, \beta_{ub}]$ that should be classified as expected. Furthermore, the region outside of the acceptable scenarios where the scenario classification is not strictly unexpected is called the *gray area*. This area is unavoidable when dealing with post-impact classification due to the nonlinearity of the robotic manipulator. The performance of the classifier can be judged by the gray area, where smaller gray areas indicate better



(a) (b)

Tuning Parameter	Symbol	Value	Unit
Window Length	m	8	–
Decay Rate	a	30	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.0163	rad
Impact Configuration Error	ϵ	0.01	rad
Impact Time Difference	ζ	0.1	s

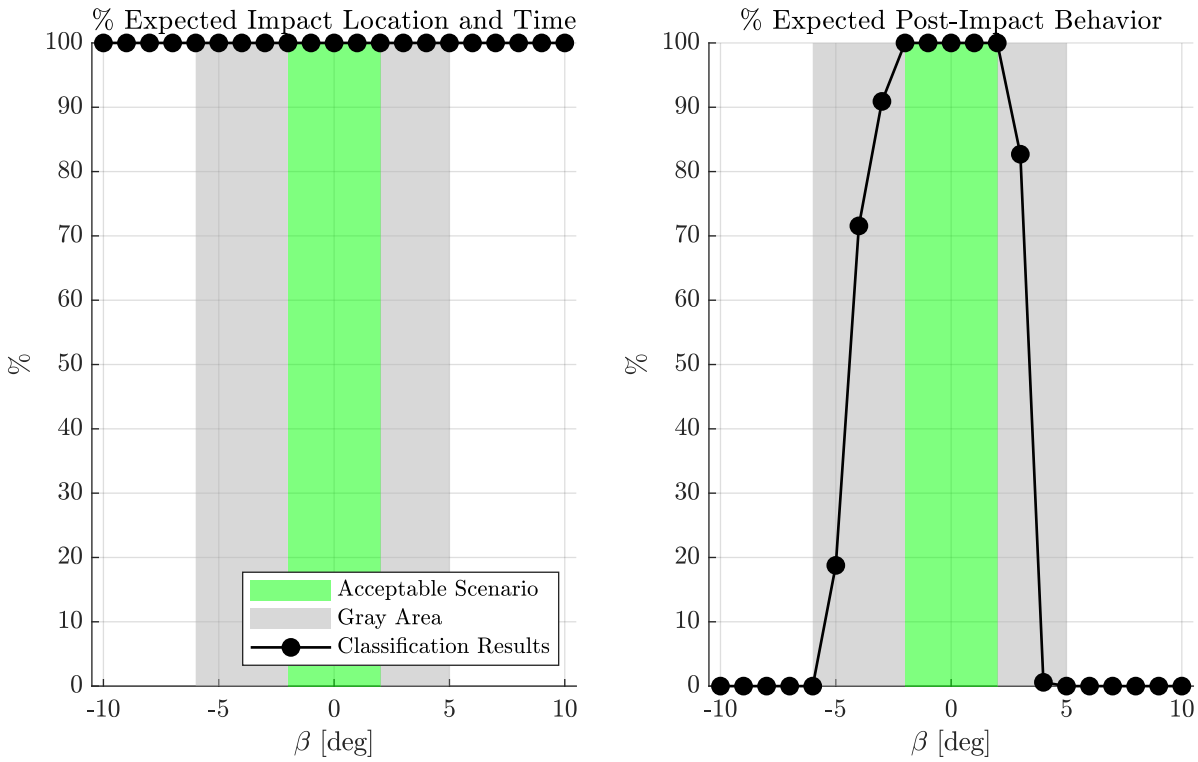
(c)

Figure 4.12: Results for the single impact classifier tuned for the acceptable scenario of β_1. For clarity, the acceptable scenario has been marked to be slightly wider than only β_1. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

performance. For testing the classifier, three acceptable scenarios are considered, namely, $\beta_1 = 0$, $\beta_2 \in [-2, 2]$, and $\beta_3 \in [-4, 4]$ deg. Furthermore, the elevation of the environment is changed according to $l = -\tan(\beta)^A\mathbf{x}_A^{\top A}\mathbf{p}_{imp}$ to ensure that the desired and detected impact location are the approximately the same. Additionally, considering that the impact classification discussed in Section 4.1.2 is relatively simple and not that interesting, the focus will be placed on the post-impact behavior classification.

The first set of results when tuning for β_1 and the tuning parameters used to obtain these results can be found in Figure 4.12. From these results, some interesting behavior can be observed. Overall, a clear trend can be observed where the further the tested scenario deviates from the acceptable scenario, the more common unexpected post-impact classification becomes. This is desired behavior and indicates that after a certain deviation from the acceptable scenario, reliable post-impact classification can occur where all post-impact behavior is classified as unexpected.

Curiously, the direction in which the scenario deviates from the acceptable scenario affects the post-impact classification performance, where $\beta < 0$ has overall worse performance than when $\beta > 0$. However, this behavior can be explained by the nonlinearity of the system, meaning that a linear change in the environment does not result in a linear difference between the post-impact responses. Due to this nonlinearity, one side will exhibit a response that is further removed from the expected behavior, which increases the chance of an unexpected post-impact classification.



Tuning Parameter	Symbol	Value	Unit
Window Length	m	8	$-$
Decay Rate	a	29	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.027	rad
Impact Configuration Error	ϵ	0.01	rad
Impact Time Difference	ζ	0.1	s

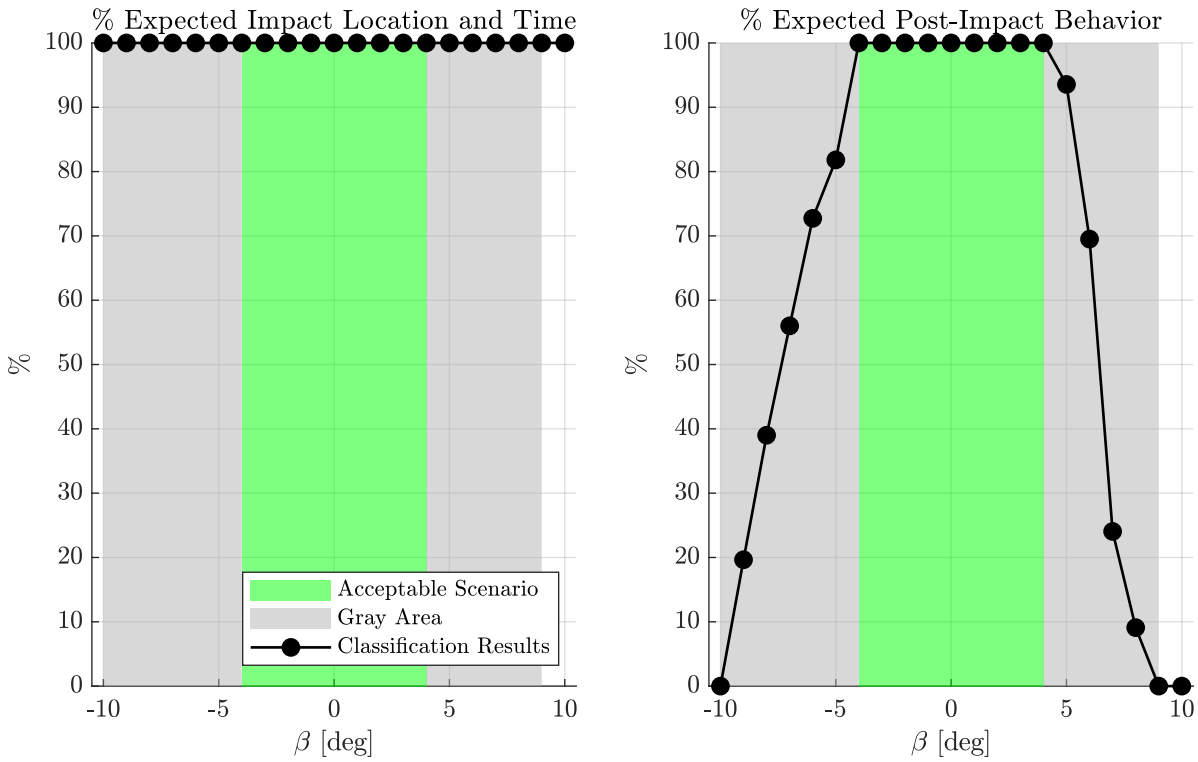
(c)

Figure 4.13: Results for the single impact classifier tuned for the acceptable scenario of β_2. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

There also seems to be a small deviation at $\beta = -2$, where the post-impact behavior is classified as expected more frequently than for $\beta = -1$. This shows that the post-impact behavior classification is not strictly decreasing in expected post-impact classification when deviating further from the acceptable scenario. However, considering that the remainder of the scenarios still undergoes a decreasing expected post-impact classification when deviating further from the acceptable scenario, a clear trend can still be observed.

The second set of results when tuning for β_2 and the tuning parameters used to obtain these results can be found in Figure 4.13. The second set of results show much of the same behavior as for the acceptable scenario β_1. However, another observation can be made where the classifier has trouble with classifying for small acceptable scenarios, i.e., in Figure 4.12, $\beta = 1$, which is a scenario outside the acceptable scenario β_1, was always classified as expected, however, no scenarios outside the acceptable scenarios are classified as expected for β_2 as can be seen in Figure 4.13. Furthermore, the gray area for the acceptable scenario of β_2 is smaller than for β_1, indicating that the performance for this case is also better.

The final set of results when tuning for β_3 and the tuning parameters used to obtain these results can be found in Figure 4.14. The results shown in Figure 4.14 show similar results to the other cases. However, it is worth noting that the gray area for this scenario has increased, which indicates that the performance is worse. Besides this observation, no other peculiar behavior can be observed. With the results for



Tuning Parameter	Symbol	Value	Unit
Window Length	m	8	$-$
Decay Rate	a	19	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.085	rad
Impact Configuration Error	ϵ	0.01	rad
Impact Time Difference	ζ	0.1	s

(c)

Figure 4.14: Results for the single impact classifier tuned for the acceptable scenario of β_3. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

tuning and testing the classifier on three acceptable scenarios shown and discussed, it is clear that the single impact classifier is able to be tuned based on the requirements set by the user. Furthermore, in all of these cases, the gray area is also sufficiently small so that large deviations from the acceptable scenario can be correctly classified as unexpected. This classifier was designed with single impacts in mind. To improve on the design to allow for the classification of simultaneous impacts, several adjustments are required. These adjustments and the subsequent testing of the simultaneous impact classifier will be discussed in Section 4.4.

4.4 Simultaneous Impact Classifier

When considering a classification algorithm for simultaneous impacts, the prominent differences between the single- and simultaneous impact cases need to be investigated to see what adjustments are needed to facilitate simultaneous impact classification. The main difference between the single- and simultaneous impact cases is that the simultaneous impact case has two possible impact types and are both described in Section 2.3. The first type of impact is the ideal simultaneous impact, where several points make contact with the environment simultaneously. The second type is a non-ideal simultaneous impact where there is a time difference between the first impact and when contact has been completed. This leads to the main challenge of simultaneous impact classification, the classifier should work for both types of simultaneous impacts.

When considering the classification of an ideal simultaneous impact, the classification procedure can remain the same as the single impact case. However, when simultaneity is lost and the impact becomes a non-ideal simultaneous impact, it becomes important to know when an impact has concluded because the intermediate phase cannot be classified due to the unpredictability of the order of impacts and the subsequent intermediate impact behavior.

However, ignoring the intermediate phase of the multi-impact while still classifying an ideal simultaneous impact correctly is difficult, as it requires information that is not available to the robotic manipulator using current methods, namely, if contact with the environment has been completed. Furthermore, due to the existence of bounces, as was shown in Section 3.5, the maximum number of impacts before contact is completed cannot be guaranteed. However, after contact has been completed, a classification scheme similar to the single impact case can be applied to classify the impact.

The most obvious solution to this challenge would be using a method to detect when contact has been completed, however, no such method is currently available. The method used in the switching strategy in Section 3.3.2 could be a viable alternative, but due to the possibility of bounces faulty classification can occur where the behavior is classified before contact has been completed. Instead, the proposed method is to reset the classifier when a new impact is detected within a certain time after the previous impact.

The concept behind this principle is that the classifier is turned on when an impact is detected. It will first determine if the impact location and time were expected while taking into account broader margins for ζ and ϵ due to the increased likelihood that the impact occurs at a different time and joint configuration than desired. Afterwards, the post-impact classifier is turned on. The classifier will then continue to classify the post-impact behavior. If a second impact occurs within t_ξ s of the previous impact, the new impact is classified and the post-impact classifier is reset, which is done by discarding every post-impact classification result it has obtained and resetting the signal envelope to only use information from the current timestep onward. This will continue until no impacts have been detected in a predefined time interval t_ξ after the final detected impact, after which the results of the classification are given.

The benefits of this method are twofold. The first benefit is that this method solves the simultaneous impact classification challenge discussed in this chapter. The second benefit is that this can be extended to three dimensions without further adjustments. Furthermore, by keeping in mind the possibility that a method will exist in the future to detect whether contact has been completed, the simultaneous impact classifier can easily be changed by only starting the classifier after contact has been completed.

Several additional changes need to be made to the post-impact classifier for it to function as desired. First of all, an adaptable window length awl needs to be implemented that will reset to 1 every time an

Algorithm 2: Simultaneous impact classification algorithm

input : The joint velocity error $\mathbf{e}_{vel}$ from timesteps $k-m$ to k.
Current timestep k.
Current time t_k.
Sample time Δt.
Signal envelope for previous timestep, $\mathbf{e}_{vel,ub,k-1}$ and $\mathbf{e}_{vel,lb,k-1}$.
The current impact location and time classification result, Impact.
The current post-impact behavior classification result, Post-Impact.
Impact detection status.
For each detected impact, the impact configuration error $\mathbf{e}_{imp,j}$ and detection time t_j.
The number of detected impacts j.
The adaptive window length of the previous iteration awl.

output : Updated impact location and time classification result, Impact.
Updated post-impact behavior classification result, Post-Impact.
Signal envelope for current timestep, $\mathbf{e}_{vel,ub,k}$ and $\mathbf{e}_{vel,lb,k}$.
Updated adaptive window length awl.

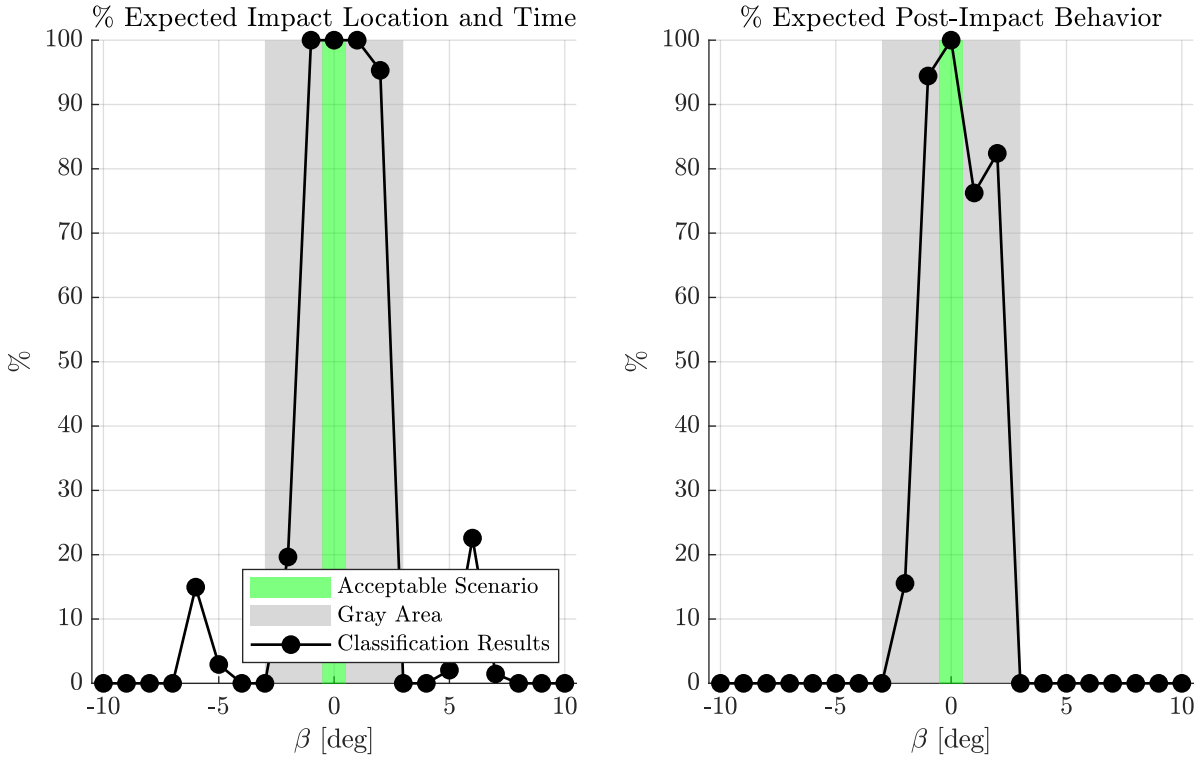
parameter: Window length m.
Signal envelope decay rate a.
Maximum impact configuration error ϵ.
Permissible time difference between the desired and detected impact time ζ.
Signal envelope margin e_η.
Post-impact classification time t_ξ.
Maximum allowable time difference between impacts t_{int}.

```
if Impact detected at current time t_k then
    awl = 1;
    if ‖e_imp,j‖ > ε or ‖t_j − t_1,d‖ > ζ then
        Impact = Unexpected;
        Post-Impact = Unexpected;
    else
        Impact = Expected;
        Post-Impact = Unknown;
    end
    e_vel,lb,k = e_vel,k ;
    e_vel,ub,k = e_vel,k ;
else
    e_vel,min,k = min {e_vel,k−awl, ..., e_vel,k−l, ..., e_vel,k};
    e_vel,max,k = max {e_vel,k−awl, ..., e_vel,k−l, ..., e_vel,k};
    e_vel,lb,k = min {e_vel,min,k, (1 − aΔt) e_vel,lb,k−1 + aΔt e_vel,min,k};
    e_vel,ub,k = max {e_vel,max,k, (1 − aΔt) e_vel,ub,k−1 + aΔt e_vel,max,k};
end
if Post-Impact = Unknown then
    if (e_vel,lb,k − e_η > 0_n×1 or e_vel,ub(t) + e_η < 0_n×1 or t_j − t_j−1 > t_int ) and awl = m then
        Post-Impact = Unexpected;
    else
        if t_k − t_j > t_ξ then
            Post-Impact = Expected;
        end
    end
end
if awl < m then
    awl = awl + 1;
end
if Impact ≠ Unknown and Post-Impact ≠ Unknown and t_k − t_j > t_ξ then
    Deactivate Classifier;
end
```

impact is detected, which will then increase by 1 for every loop of the classification algorithm until it reaches the desired window length m. Furthermore, the post-impact classification should only start when the window has reached its full length, ensuring that incorrect classifications such as what occurred in Section 4.2.2 do not occur. An additional parameter is included that allows for the classification based on the time between impacts as seen in (4.3). Finally, the impact location and time classification conditions are changed to the equivalent simultaneous impact conditions found in (4.8) and (4.2), respectively. It should be noted that the simultaneous impact classifier requires a method to detect impacts regardless of the contact state. For this project, the multi-impact detection method described in Section 3.2 is used to detect the impacts. The overall structure of a single loop of the simultaneous impact classification algorithm can be found in Algorithm 2. This classifier will be tested in Section 4.4.1.

4.4.1 Testing the Simultaneous Impact Classifier

With the classifier adapted to deal with simultaneous impacts, it will need to be tested to ensure that the performance is satisfactory. When testing the simultaneous impact classifier, two cases will be tested. The first and most important case that is considered is the simultaneous impact case. The second case



Tuning Parameter	Symbol	Value	Unit
Window Length	m	10	–
Decay Rate	a	30	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.033	rad
Impact Configuration Error	ϵ	0.042	rad
Impact Time Difference	ζ	0.1	s
Intermediate Impact Time Difference	t_{int}	0.1	s

(c)

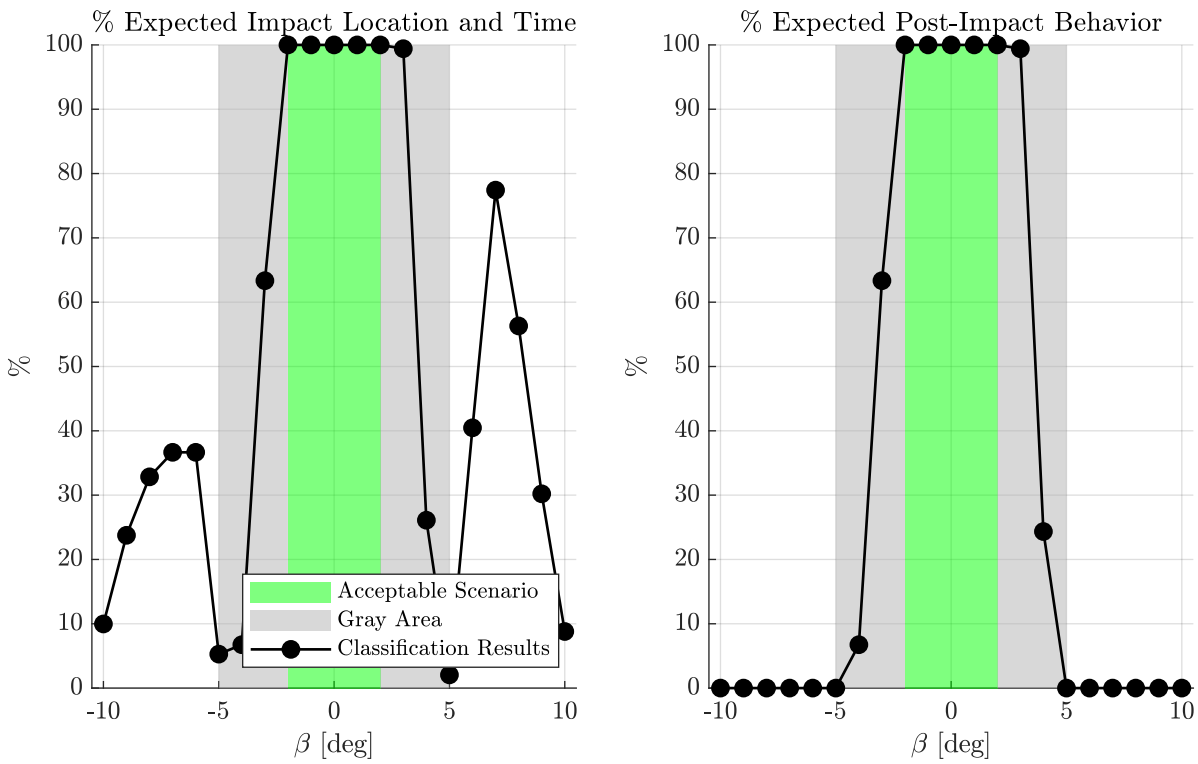
Figure 4.15: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_1 for a simultaneous impact case. For clarity, the acceptable scenario has been marked to be slightly wider than only β_1. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

that is considered is the single impact case because if it works for the single impact case, a single classifier can be used for classification regardless of the number of contact points. Finally, similarly to Section 4.3.2, $\beta_1 = 0$, $\beta_2 \in [-2, 2]$, and $\beta_3 \in [-4, 4]$ *deg* are considered as the acceptable scenarios when tuning.

Testing for Simultaneous Impact Case

The first set of results when tuning for β_1 and the tuning parameters used to obtain these results for the simultaneous impact case can be found in Figure 4.15. With the first set of results, some interesting behavior can be observed. First of all, it can be observed that the gray area is smaller than for the single impact classifier in Figure 4.12, indicating that the performance of the simultaneous impact classifier is better than the single impact classifier. Furthermore, the same behavior can be observed for the single impact classifier with β_1 as the acceptable scenario, where at $\beta = 2$, the expected post-impact classification occurs more often than for $\beta = 1$. This behavior itself is not desirable, however, after $\beta = 2$ the gray area ends, meaning that the overall performance is still better than the single impact classifier for this acceptable scenario.

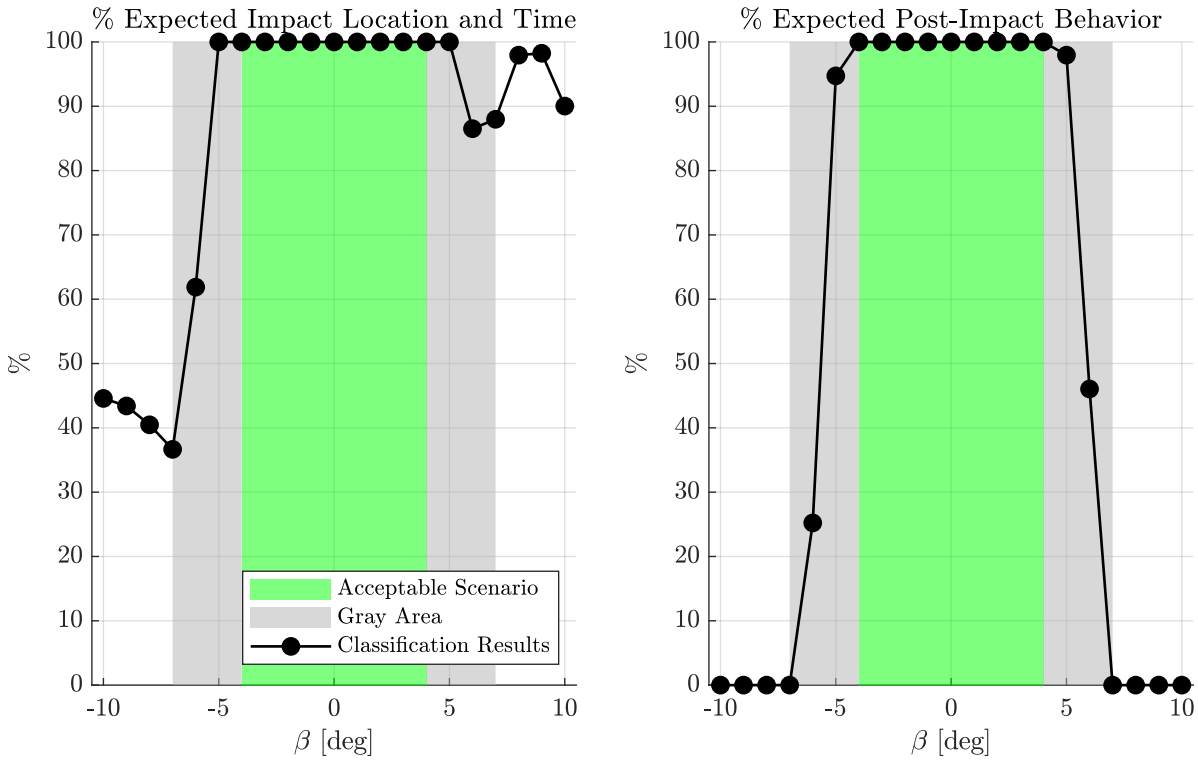
The second set of results when tuning for β_2 and the tuning parameters used to obtain these results for the simultaneous impact case can be found in Figure 4.16. Overall, much of the same behavior can be



Tuning Parameter	Symbol	Value	Unit
Window Length	m	15	–
Decay Rate	a	18	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.1	rad
Impact Configuration Error	ϵ	0.067	rad
Impact Time Difference	ζ	0.1	s
Intermediate Impact Time Difference	t_{int}	0.1	s

(c)

Figure 4.16: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_2 for a simultaneous impact case. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.



Tuning Parameter	Symbol	Value	Unit
Window Length	m	15	–
Decay Rate	a	18	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.236	rad
Impact Configuration Error	ϵ	0.115	rad
Impact Time Difference	ζ	0.11	s
Intermediate Impact Time Difference	t_{int}	0.1	s

(c)

Figure 4.17: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_3 for a simultaneous impact case. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

observed for the acceptable scenario of β_1. Furthermore, the gray area of this region is the same size as for the previous acceptable scenario. Other than the previously mentioned behavior, no notable observations can be made about these results.

The third set of results when tuning for β_3 and the tuning parameters used to obtain these results for the simultaneous impact case can be found in Figure 4.17. With the final set of figures, all of the behavior is similar to the previous cases. It should be noted that the size of the gray area remained consistent between different acceptable scenarios, this indicates that the simultaneous impact classifier can be tuned for a variety of requirements while maintaining its performance. This is different from the single impact classifier, which had fluctuating performance between different acceptable scenarios.

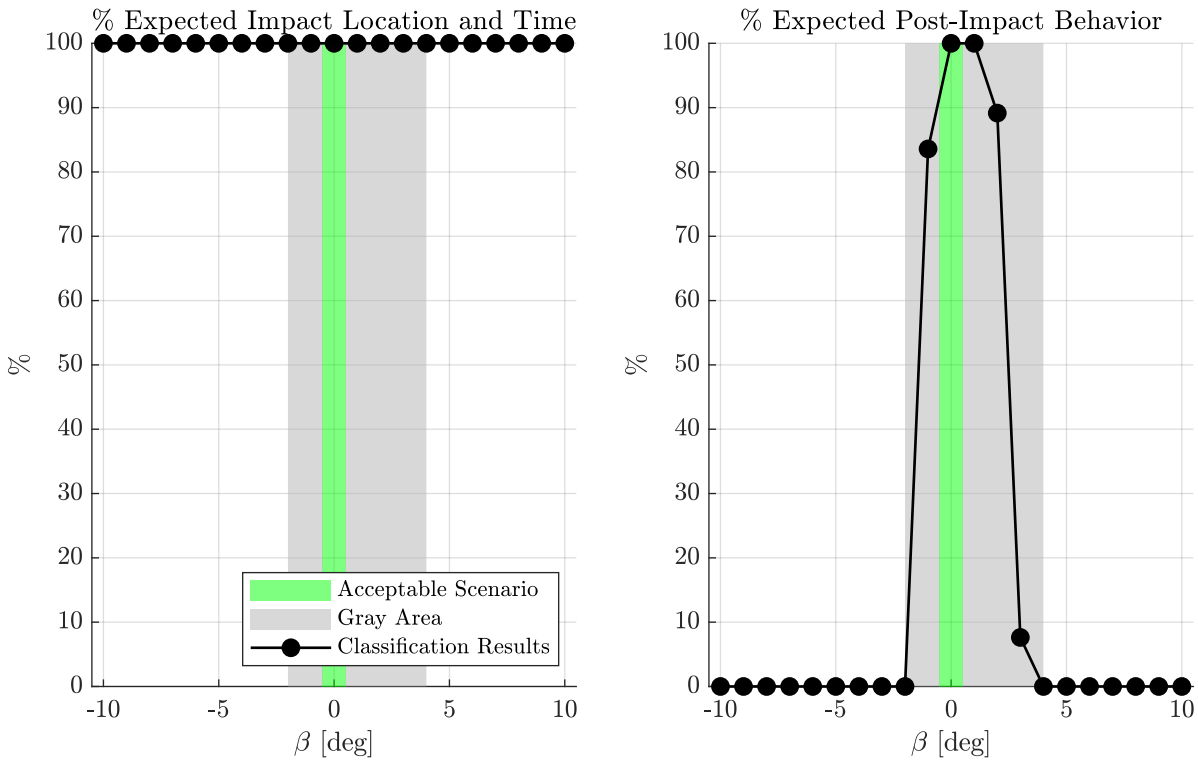
Overall, from the figures presented in this section, it is clear that the classification of post-impact behavior is possible for simultaneous impacts. Furthermore, the classification method also shows a consistent performance between different tested scenarios. Therefore, it can be concluded that the simultaneous impact classifier functions as desired for the simultaneous impact case.

Testing for Single Impact Case

To test the performance of the simultaneous impact classifier on the single impact case, the simultaneous impact classifier is tuned for the same three different acceptable scenarios as for the previous sets of simulation results for the simultaneous impact case. The first set of results when tuning the simultaneous impact classifier for β_1 and the tuned parameters used to obtain these results for the single impact case can be found in Figure 4.18.

With the first set of results in Figure 4.18, similar performance can be observed to the simultaneous impact case. Interestingly, the performance for this case is better than for the single impact classifier when tuned for the acceptable scenario of β_1, as seen in Figure 4.12. However, the results in Figure 4.18 suffer from the same problem where $\beta = 1$ is always classified as expected, which also occurs for the single impact classifier for β_1, as observed in Figure 4.12.

The second set of results when tuning the simultaneous impact classifier for $\beta =\in [-2, 2]$ and the tuned parameters used to obtain these results for the single impact case can be found in Figure 4.19. For the second set of results, interesting behavior can be observed in the form of the size of the gray area. First of all, when compared to Figure 4.13, for $\beta < -2$, the gray area is similar in size to the single

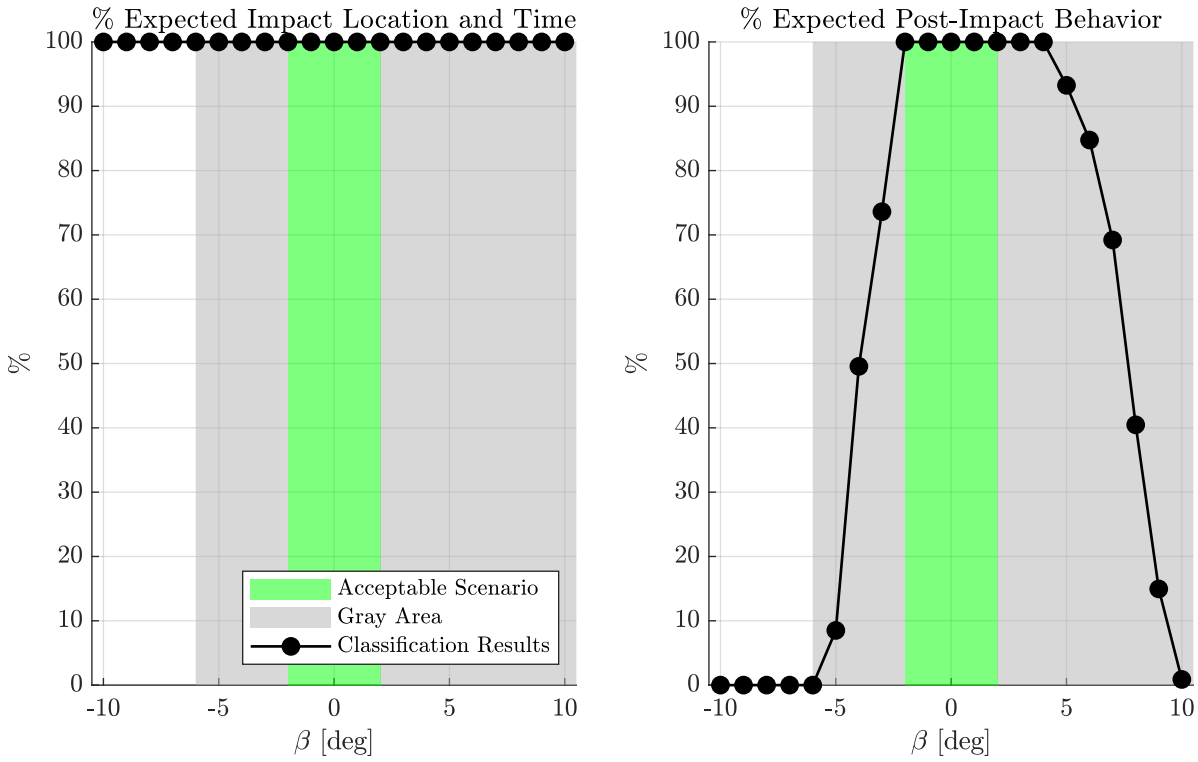


(a) (b)

Tuning Parameter	Symbol	Value	Unit
Window Length	m	10	$-$
Decay Rate	a	30	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.01	rad
Impact Configuration Error	ϵ	0.025	rad
Impact Time Difference	ζ	0.1	s

(c)

Figure 4.18: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_1 for a single impact case. For clarity, the acceptable scenario has been marked to be slightly wider than only β_1. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.



(a) (b)

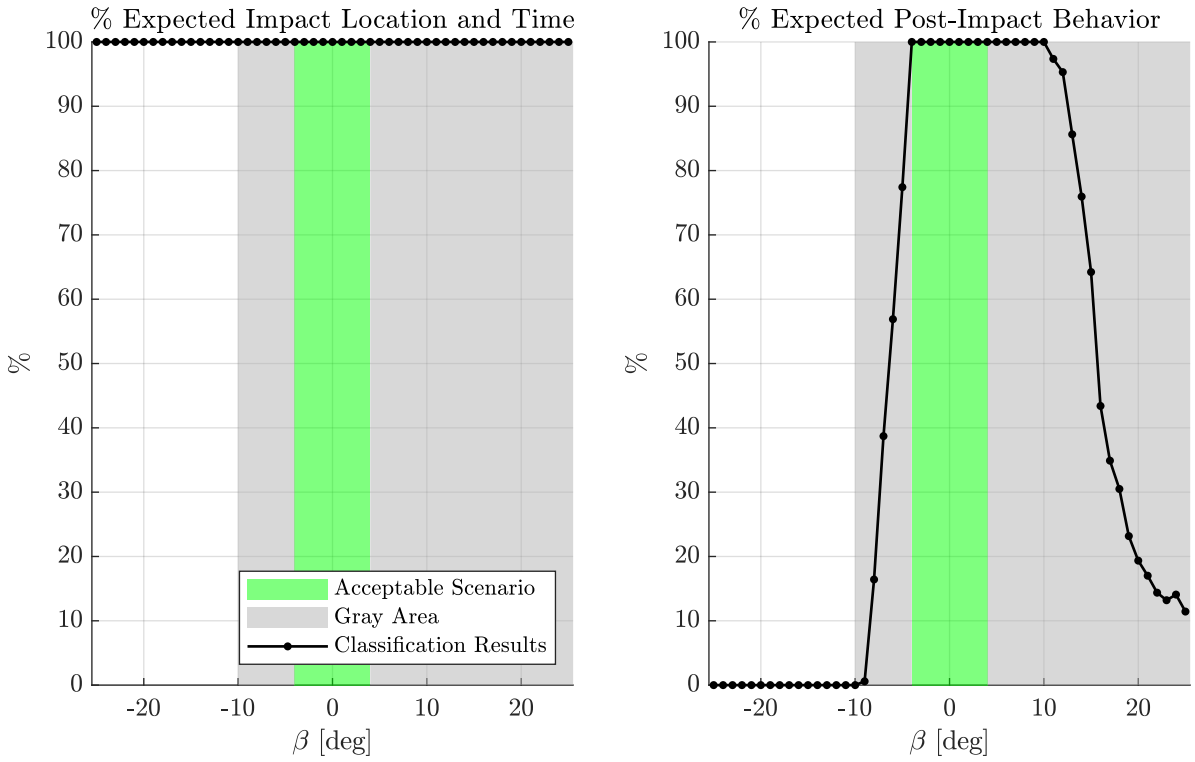
Tuning Parameter	Symbol	Value	Unit
Window Length	m	10	$-$
Decay Rate	a	25	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.126	rad
Impact Configuration Error	ϵ	0.025	rad
Impact Time Difference	ζ	0.1	s

(c)

Figure 4.19: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_2 for a single impact case. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

impact classifier for the same acceptable scenario. However, $\beta > 2$ has a significantly larger gray area, showing that the overall performance is significantly worse. Additionally, the gray area for $\beta > 2$ does not end for the tested scenarios, another indication that the performance of the simultaneous impact classifier applied to the single impact case is worse than a dedicated single impact classifier. Finally, the simultaneous impact classifier applied to a single impact case also suffers from the same problem as for the single impact classifier where the acceptable scenario was set to β_1, namely, that there are scenarios that should be classified as unexpected but are instead strictly classified as expected.

The third set of results when tuning the simultaneous impact classifier for $\beta =\in [-4, 4]$ and the tuned parameters used to obtain these results for the single impact case can be found in Figure 4.20. For the final set of results, an extended region was selected to simulate. As can be seen in the figure, the gray area for $\beta < -4$ is smaller than for the single impact classifier as found in Figure 4.14. However, for $\beta > 4$ the gray area continued indefinitely until it was decided that no more simulations would be run. This shows how the performance of the simultaneous impact classifier applied to the single impact case is unsatisfactory for large acceptable scenarios. Furthermore, it should be noted that for small acceptable scenarios, the performance is better, meaning that it can still be applied if careful attention is given to what type of requirements the classifier is tuned for.



(a) (b)

Tuning Parameter	Symbol	Value	Unit
Window Length	m	12	–
Decay Rate	a	20	$1/s$
Classification Time	t_ξ	0.1	s
Envelope Margin	e_η	0.3	rad
Impact Configuration Error	ϵ	0.025	rad
Impact Time Difference	ζ	0.1	s

(c)

Figure 4.20: Results for the simultaneous impact classifier tuned for the acceptable scenario of β_3 for a single impact case. In (a), the impact location and time classification results are shown. In (b), the post-impact behavior classification results are shown. In (c), the tuning parameters used to obtain the results are shown.

4.5 Reflection

With the results for the single- and simultaneous impact classifier obtained and reviewed in Sections 4.3.2 and 4.4.1, it is important to reflect on the unexpected behavior that was observed, what the source of this behavior is, and to draw a conclusion regarding the overall functioning of the classifier as well as recommendations on how to use the classifier. The subjects that will be discussed are the sudden increase in expected post-impact classification for increasing deviation from the acceptable scenario observed in Figures 4.12 and 4.15 and the size of the gray area. This is concluded by discussing the overall results of the classification methods proposed in this chapter, focusing on if the classification methods provide satisfactory performance for classifying impacts.

Increased Expected Classification

As was observed in Figures 4.12 and 4.15, two instances are found where a scenario has a higher expected post-impact behavior classification than the scenario preceding it. This indicates that when the scenario deviates from the acceptable scenario, it does not guarantee that the unexpected post-impact behavior classification will occur more frequently. Furthermore, it should be noted that this behavior is also observed in Figure 4.20. However, because the overall results presented in this figure were unsatisfactory,

no further attention was given to this fact.

The precise cause of this behavior is unknown, as it was expected that the further the scenario deviates from the acceptable scenario, unexpected post-impact classifications would become more common. However, two possible sources of this problem can be identified. The first possibility is the brief loss of contact that was discussed in Section 3.6, which introduces an additional nonlinearity into the system. Furthermore, for the simultaneous impact case, the problem can be exacerbated due to the possible detection of these additional impacts, which resets the classifier. Therefore, if the problems presented in Section 3.6 were to be solved, these tuning methods should be tested again to see whether this unexpected behavior is still present.

When considering that this phenomenon occurred only twice in the results presented in Sections 4.3.2 and 4.4.1, it can be concluded that a clear trend still exists. The further the scenario deviates from the acceptable scenario for which it is tuned, eventually, it will reach a scenario where all post-impact behavior is classified as unexpected. Furthermore, none of the scenarios in the acceptable scenario were classified as unexpected, and none of the scenarios outside both the gray area and the acceptable scenarios were classified as expected.

Size of the Gray Area

As can be observed in the results presented in Sections 4.3.2 and 4.4.1, the size of the gray area varies depending on the type of classifier and the selection of the acceptable scenarios. For the successful functioning of the classifier, it is required that the gray areas are finite and it is desirable that the gray areas are as small as possible.

For the single impact case, both the single- and simultaneous impact classifier were tested. For the single impact classifier results presented in Section 4.3.2, it can be observed that for each set of results the gray area is finite. However, the size of the gray area can vary by a lot, where Figure 4.14 has a significantly larger gray area than Figure 4.13. However, when comparing these results with the simultaneous impact classifier applied to the single impact case results presented in Section 4.4.1, the difference in performance is significant. For example, in Figures 4.19 and 4.20 the gray area does not end in the scenarios that were simulated, where Figure 4.20 tests an extended set of scenarios. However, Figure 4.18 has a smaller gray area than Figure 4.12, but this does not counteract the significantly worse performance overall that the simultaneous impact classifier has for the single impact case than the single impact classifier.

The simultaneous impact classification results presented in 4.4.1 show a consistent performance. First of all, for each of the figures, the size of the gray area is the same. Furthermore, while the shape of the gray area is not symmetrical, the size of the gray area on either side of the acceptable scenario is symmetrical. Finally, none of the scenarios outside of the acceptable scenario has a 100 % expected post-impact classification. All of these properties show that the simultaneous impact provides good scalability for a variety of acceptable scenarios while maintaining good overall performance for classifying unexpected scenarios.

Classifier Conclusion

With the results presented in Sections 4.3.2 and 4.4.1, and the points of reflection brought up in the sections above, a conclusion can be drawn regarding the classification methods designed in this chapter. For single impact classification, two solutions were proposed, a method designed for single impacts, and the method designed for simultaneous impacts but applied to a single impact case. From the results, it is clear that for single impact classification, a tunable classifier designed specifically for this case provides the best results. This is because when using a classifier designed for the single impact case, the gray areas are finite, and sufficiently large deviations from the acceptable scenario guarantee an unexpected post-impact classification. Furthermore, the simultaneous impact classifier applied to the single impact case has significantly worse performance for most tested acceptable scenarios.

For the simultaneous impact case, the simultaneous impact classifier provides a solution that has a consistent performance between different acceptable scenarios. Furthermore, the gray areas are also small, and no scenarios outside of the acceptable scenarios are always classified as expected for the tested

scenarios. These properties were all desirable and provide a solution to the goal set out in Chapter 1, which was to develop a classification method to classify unexpected behavior of simultaneous impacts that focuses on the geometric differences between the expected and real environment as the leading cause of unexpected behavior.

5 Conclusion and Recommendations

With the classifier constructed and tested in Chapter 4, this chapter will conclude the results obtained during the project. This is then followed by a set of recommendations that briefly discuss possible improvements and suggestions for follow-up research and projects. The research objective, as was outlined in Chapter 1, was defined as

Propose, design, and test a threshold-based classification method that employs an available and computationally feasible prediction of the impact behavior to classify the expectancy of a simultaneous impact scenario, focusing on the geometric differences of the impact surface as the leading cause of unexpected behavior.

The simplified prediction was obtained through the use of a rigid impact map and inverse kinematics that is standard and available in software simulators and was explained in detail in Chapter 3. Furthermore, a novel signal envelope filter was developed that allowed for the analysis of a post-impact velocity response of an impact on a flexible joint robotic manipulator. Finally, this signal envelope filter was used to develop both a single- and simultaneous impact classifier, which were both tested. The conclusions regarding this project will be drawn in Section 5.1, and the recommendations will be given in Section 5.2.

5.1 Conclusion

The conclusion of this project consists of two parts. The first part is whether the research objective was achieved and the second part is whether the solution found for the research objective satisfies the requirements, preferences, and constraints defined in Section 1.2.

Starting with the research objective, the threshold-based classification method that was proposed, designed, and tested in Chapter 4 was able to classify the impact behavior of a realistic robotic manipulator using an available and computationally feasible prediction. Furthermore, two classifiers were proposed that use the same underlying principle of classification, one for single impact cases and the other for simultaneous impact cases. Unexpectedly, reformulating the joint velocity as the joint velocity error allowed for a simplified implementation of the classification method without introducing additional variables and sacrificing the performance in ways alternate solutions to a problem encountered in Section 4.2.2 would. Finally, the classification method proposed in Chapter 4 focuses on the impact behavior as a consequence of the geometric properties of the environment. Each part of the research objective for this project has been solved with the classification method proposed in this report, which leads to the overall conclusion that the research objective has been achieved.

When quantifying how well the classifier performs, the requirements, preferences, and constraints presented in Section 1.2 are consulted. The requirements that were set refer to the basic functioning of the classifier, namely, that if the difference between the expected and measured behavior surpasses a threshold, that the scenario is classified as unexpected. Furthermore, a requirement was defined where the classifier can be tuned based on user-defined requirements. When investigating the results presented in Sections 4.3.2 and 4.4.1, it is clear that these requirements are all satisfied, namely, that the classifier thresholds that classify different impact scenarios can be tuned according to the requirements.

The first preference discusses the desire that unexpected post-impact classification occurs as soon as possible, preferably the moment the post-impact behavior can be measured or estimated. For the single impact classifier, the minimum post-impact classification time is found to be $m\Delta t$ s. However, for the simultaneous impact classifier the minimum post-impact classification time is t_ξ s after the final impact is detected due to the uncertainty of whether contact has been completed or not. If a method were to be implemented so that it can be confirmed that contact has been completed, a similar post-impact classification speed can be achieved as for the single impact classifier.

For the second preference, the only computationally intensive calculation that is performed is the generation of the signal envelope. However, this does not rely on complex calculations, instead, the intensity of the calculation is dependent on the window length, where for longer window lengths it will take longer

to determine the minimum and maximum values of the response. If attention is given to ensure that the window length is not excessively large, then the classifier itself is a lightweight solution to the classification problem which lends itself to implementation on real hardware. The final preference concerns the presentation and ease of implementation of the classifier. This preference is satisfied when looking at Algorithms 1 and 2, which show simplified representations of the classifiers that can be followed for implementation on a variety of systems.

For the constraints, the first two points concern the fact that the classifier should only use the information available to a typical robotic manipulator, and that it should not be tailored to a specific robotic manipulator. This was achieved for the classifier, as special attention was given to only using information that was available to a generic robotic manipulator as well as the design of a multi-impact detection method that uses this information. This automatically leads to a classifier design that does not rely on properties of a specific robotic manipulator.

The final constraint discusses the necessity that the classifier should work for a three-dimensional simultaneous impact, i.e., an impact where at least three points in three-dimensional space impact the environment at the same time. This final constraint remains untested due to the limitations of the simulation. However, as was discussed in Section 4.4, the design of the simultaneous impact classifier was made to facilitate three-dimensional simultaneous impact classifications. This should be tested in a follow-up project to ensure that the classifier does work for a three-dimensional case.

Overall, the classifier satisfies most of the requirements, preferences, and constraints laid out. The only points of improvement for the proposed classification method are the speed of classification for the simultaneous impact classifier and testing the simultaneous impact classifier in a three-dimensional simultaneous impact case. For these remaining points that were not satisfied, there is a clear path to follow to ensure that the classifier does satisfy them. However, considering that achieving these points still requires a considerable amount of work, these are best left for follow-up research and projects.

5.2 Recommendations

When considering the conclusion presented in the previous section, it is clear that the overall research objective has been achieved and that the requirements, preferences, and constraints have been mostly satisfied. However, while developing the classifier, some problems were encountered that would benefit from a project dedicated to them. Furthermore, there are also recommendations regarding the continuations of this project. Each of these recommendations will be discussed in the following sections.

Validate the Classifier on a Real System

Currently, the classifier has only been tested in simulations using a model designed to resemble a realistic scenario. Furthermore, some components such as the control switching strategy and impact detection methods are based on the understanding of the model, and there is no guarantee that these work for a real system unless these are validated on such a system. This leads to the necessity of validating the results obtained in this report on a real robotic manipulator. This should start with validating the individual components such as the multi-impact detection method and the switching strategy. This is then followed by implementing the single- and simultaneous impact classifier on this system and testing if the classification works by performing several impact experiments.

Classification of Three-Dimensional Simultaneous Impacts

In Section 4.4, it was discussed that resetting the classifier when a new impact is detected allows for classifying the post-impact behavior when it is unknown if contact has been completed. This method was designed with three-dimensional simultaneous impacts in mind. This should therefore be tested using either a three-dimensional model or when validating the classification method on a real system.

Control Strategy Adjustments

The current control strategy to handle simultaneous impacts is an extension of reference spreading for a QP control implementation. However, as was discussed in Section 3.3.2, this can only be implemented on a QP controller that uses joint-space configuration tasks to control the robotic manipulator. However, in real systems, task-space position and orientation tasks are more commonly used to control a robotic manipulator. Whether using a task-space controller or joint-space controller with trajectories based on a task-space trajectory affects the results by a significant margin remains to be seen. Regardless, improving the reference spreading formulation for QP control with task-space trajectories is a topic that can be investigated further.

Furthermore, in Section 3.5, it was observed that a loss of contact could occur. This loss of contact could result in a second impact being detected, which would affect the control strategy that the QP controller is following. Furthermore, this loss of contact is observed after every impact, showing that this problem is not unique to the intermediate phase. Preventing these bounces entirely in scenarios where they are undesired is unlikely to be possible, however, improving the switching strategy to ensure that early switching to the post-impact phase is avoided is another possibility for a follow-up project. Examples of possible solutions to this problem are by assuming a fixed maximum interval between impacts, where switching to the post-impact phase occurs after this interval regardless of the detected number of impacts. Another possibility is using the post-impact classification method to determine that for an expected post-impact scenario, the impact has concluded.

Contact Completion Detection

Another problem that was encountered during the project was the inability to detect when contact with the environment has been completed. The proposed solution that allows for switching strategies and classifying simultaneous impacts was designed based on the assumption that no additional impacts occur. However, as was discussed in Section 3.5, additional impacts do occur which throw off the current control strategy and can affect the classification method. If a solution were to be designed that can detect when contact has been completed, the control strategy can be refined and the simultaneous impact classifier performance can be improved by only turning it on when contact is completed. Investigating this possibility is another opportunity for a follow-up project.

Bibliography

[1] Sara Lone, Isabela Fàvero, Ludovica Quaglieri, and Shaun Packiaraja. European ecommerce report. Technical report, Ecommerce Europe, 2019.

[2] Ecommerce Europe. Impact of the coronavirus on e-commerce. Technical report, Ecommerce Europe, 2021.

[3] Richard P. Smith. Boredom: A review. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 23(3):329–340, 1981.

[4] Syed Sina Mirrazavi Salehian and Aude Billard. A dynamical-system-based approach for controlling robotic manipulators during noncontact/contact transitions. *IEEE Robotic and Automation Letters*, 3(4):2738–2745, October 2018.

[5] Ilias Aouaj, Vincent Padois, and Alessandro Saccon. Predicting the post-impact velocity of a robotic arm via rigid multibody models: an experimental study. *CoRR*, abs/2010.08220, 2020.

[6] Wouter Weekers. Validation of nonsmooth impact maps in robot impact experiments. Master's thesis, TU/e: Department of Mechanical Engineering, April 2021.

[7] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, December 2017.

[8] S. Khelouat, T.M. Laleg-Kirati, A. Benalia, D. Boukhetala, and M. Djemai. A geometric approach to nonlinear fault detection and isolation in a hybrid three-cell converter. *International Journal of Systems Science*, 50(5):1069–1088, 2019.

[9] Yoji Yamada, Yasuhiro Hirasawa, Shengyang Huang, Yoji Umetani, and Kazutsugu Suita. Human–robot contact in the safeguarding space. *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, 2(4):230–236, December 1997.

[10] Kazutsugu Suita, Yoji Yamada, Nuio Tsuchida, Koji Imai, Hiroyasu Ikeda, and Noboru Sugimoto. A failure-to-safety 'kyozon' system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5163–5170, May 1995.

[11] Shinji Takakura, Toshiyuki Murakami, and Kouhei Ohnishi. An approach to collision detection and recovery motion in industrial robot. In *Conference of the IEEE Industrial Electronics Society*, pages 421–426, 1989.

[12] Daolin Ma, Siyuan Dong, and Alberto Rodriguez. Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements. *CoRR*, abs/2103.08108, 2021.

[13] G. De Maria, C. Natale, and S. Pirozzi. Force/tactile sensor for robotic applications. *Sensors and Actuators A: Physical*, 175(1):60–72, December 2011.

[14] Mark Rijnen, Alessandro Saccon, and Henk Nijmeijer. Motion signals with velocity jumps: Velocity estimation employing only quantized position data. *IEEE Robotics and Automation Letters*, 3(3):1498–1505, 2018.

[15] Alin Albu-Schäffer, Sami Haddadin, Gerd Hirzinger, and Alessandro De Luca. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *International Conference on Intelligent Robots and Systems*, pages 1623–1630, 2006.

[16] Alessandro De Luca and Raffaella Mattone. Actuator failure detection and isolation using generalized momenta. In *2003 IEEE International Conference on Robotics and Automation*, pages 634–639, September 2003.

[17] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *2005 IEEE International Conference on Robotics and Automation*, pages 999–1004, April 2005.

[18] Seyed Ali Baradaran Birjandi, Johannes Kühn, and Sami Haddadin. Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing. *IEEE Robotics and Automation Letters*, 5(2):954–961, 2020.

[19] Kyu Min Park, Jihwan Kim, Jinhyuk Park, and Frank C. Park. Learning-based real-time detection of robot collisions without joint torque sensors. *IEEE Robotics and Automation letters*, 6(1):103–110, January 2021.

[20] Dmitry Popov, Alexandr Klimchik, and Nikolaos Mavridis. Collision detection, localization & classification for industrial robots with joint torque. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 838–843, 2017.

[21] Dmitry Popov, Alexandr Klimchik, and Nikolaos Mavridis. Collision detection, localization & classification for industrial robots with joint torque. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 838–843, 2017.

[22] Stanislav Mikhel, Dmitry Popov, Shamil Mamedov, and Alexandr Klimchik. Development of typical collision reactions in combination with algorithms for external impacts identification. In *9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019*, pages 253–258, 2019.

[23] Martin Karlsson, Anders Robertsson, and Rolf Johansson. Detection and control of contact force transients in robotic manipulation without a force sensor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4091–4096, 2018.

[24] Saskia Golz, Christian Osendorfer, and Sami Haddadin. Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3788–3794, 2015.

[25] Christopher J.C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Kluwer Academic Publishers, second edition, 1998.

[26] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):Article 27, 27 pages, 2011.

[27] Mohammad Anvaripour and Mehrdad Saif. Collision detection for human-robot interaction in an industrial setting using force myography and a deep learning approach. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2149–2154, 2019.

[28] Nolwenn Briquet-Kerestedjian, Arne Wahrburg, Mathieu Grossard, Maria Makarov, and Pedro Rodriguez-Ayerbe. Using neural networks for classifying human-robot contact situations. In *18th European Control Conference (ECC)*, pages 3279–3285, 2019.

[29] Yiannis Karayiannidis, Leonidas Droukas, Dimitrios Papageorgiou, and Zoe Doulgeri. Robot control for task performance and enhanced safety under impact. *Frontiers in Robotics and AI*, 34(2):1–12, December 2015.

[30] Alessandro De Luca and Lorenzo Ferrajoli. Exploiting robot redundancy in collision detection and reaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3299–3305, September 2008.

[31] Stanislav Mikhel, Dmitry Popov, Shamil Mamedov, and Alexandr Klimchik. Development of typical collision reactions in combination with algorithms for external impacts identification. *IFAC PapersOnLine*, 52(13):253–258, 2019.

[32] S. Traversaro and A. Saccon. *Multibody Dynamics Notation (version 2)*. Technische Universiteit Eindhoven, 2019.

[33] Christian Ott, Gerd Hirzinger, and Alin Albu-Schäffer. A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *The International Journal of Robotics Research*, 26(1):23–29, 2007.

[34] K.H. Hunt and F.R.E. Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics*, 42(2):440–445, June 1975.

[35] André S. Carvalho and Jorge M. Martins. Exact restitution and generalizations for the hunt–crossley contact model. *Mechanism and Machine Theory*, 139(1):174–194, March 2019.

[36] Bernard Brogliato. *Nonsmooth Mechanics: Models, Dynamics and Control*. Springer, third edition, 2016.

[37] Karim Bouyarmane and Abderrahmane 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*, pages 4414–4419, September 2011.

[38] Joseph Salini and Vincent Padoisand Philippe Bidaud. Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *2011 International Conference on Robotics and Automation*, pages 1283–1290, May 2011.

[39] Karim Bouyarmane, Kevin Chappellet, Joris Vaillant, and Abderrahmane Kheddar. Quadratic programming for multirobot and task-space force control. *IEEE Transactions on Robotics*, 35(1):64–77, February 2019.

[40] Joris Vaillant, Karim Bouyarmane, and Abderrahmane Kheddar. Multi-character physical and behavioral interactions controller. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1650–1662, June 2017.

[41] R. Goebel, R.G. Sanfelice, and A.R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, April 2009.

[42] Alessandro Saccon, Nathan van de Wouw, and Henk Nijmeijer. Sensitivity analysis of hybrid systems with state jumps with application to trajectory tracking. In *IEEE Conference on Decision and Control*, pages 3065–3070, December 2014.

[43] Mark Rijnen. *Enabling Motions with Impacts in Robotic and Mechatronic Systems*. PhD thesis, TU/e: Department of Mechanical Engineering, 2018.

[44] Casper Beumer. Impact aware robot manipulation via task-based reference spreading. Master's thesis, TU/e: Department of Mechanical Engineering, April 2019.

[45] Mark Rijnen, Hao Liang Chen, Nathan van de Wouw, Alessandro Saccon, and Henk Nijmeijer. Sensitivity analysis for trajectories of nonsmooth mechanical systems with simultaneous impacts: a hybrid systems perspective. In *American Control Conference*, pages 3623–3629, 2019.

[46] Alin Albu-Schäffer, Gerd Hirzinger, Alessandro De Luca, and Sami Haddadin. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363, September 2008.

[47] Sami Haddadin, Ali Albu-Schäffer, and Gerd Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *The International Journal of Robotics Research*, 28(11-12):1507–1527, 2009.

A | Code of Scientific Conduct

Declaration concerning the TU/e Code of Scientific Conduct

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

In carrying out research, design and educational activities, I shall observe the five central values of scientific integrity, namely: trustworthiness, intellectual honesty, openness, independence and societal responsibility, as well as the norms and principles which follow from them.

Date

27-09-2021

Name

B.W.B. Proper

ID-number

0959190

Signature

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

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

February 21, 2020