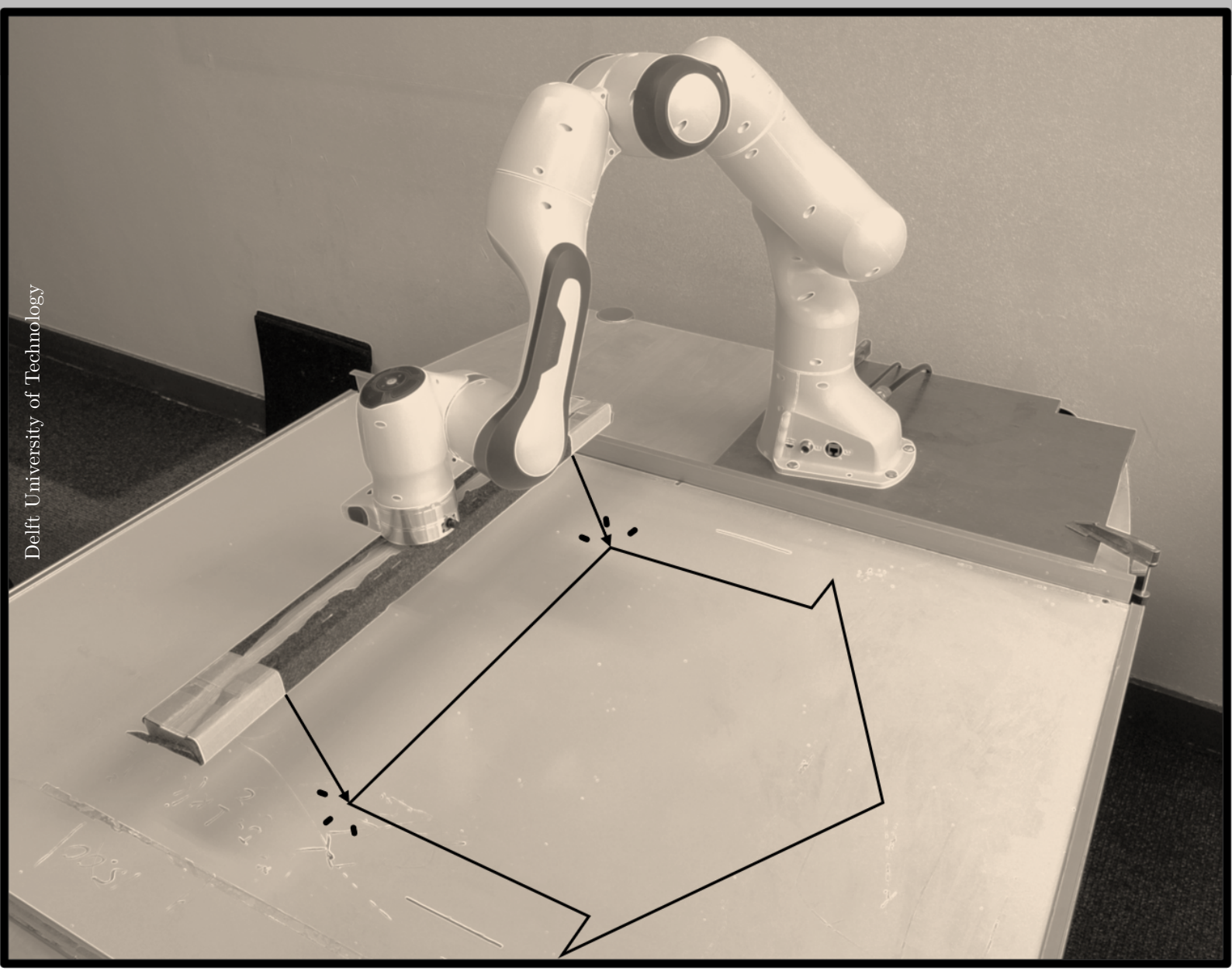
MSc Thesis

Teaching Robots Impact Tasks by Performing Demonstrations

S.J. Uitendaal



TUDelft

TU/e

Teaching Robots Impact Tasks by Performing Demonstrations

by

S.J. Uitendaal

to obtain the degree of

Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday March 31, 2022 at 14:00

Student Number:	4478320	
Thesis committee:	Dr. Ing. J. Kober	TU Delft - Cognitive Robotics, supervisor
	Dr. Ing. A. Saccon	TU Eindhoven - Dynamics and Control, supervisor
	Dr. Ing. C. Della Santina	TU Delft - Cognitive Robotics

Preface

Since I started as a bachelor student at the TU Delft, my studies have been a great part of my life. I am both proud and happy to complete it with this master's thesis. I would like to take the opportunity to thank the people that helped me during my studies and with getting my master's degree.

First of all I would like to thank my supervisors Jens and Alessandro. Jens, thank you for your feedback and your valuable questions about the approaches I talked about during our meetings. Alessandro, thank you for your critical view on my work and for making time for me whenever I needed a discussion.

Furthermore I am very grateful to Giovanni Franzese for his technical support with using the robot, our useful discussions, and his willingness to always make time for me when I asked for it. He really helped me further in operating the robot and in doing this research. I would also like to thank Linda van der Spaa for our helpful discussions and her useful suggestions.

I would like to thank my friends who supported me in doing this research. Anna and Armin, thanks for making my time during the thesis pleasant.

I am very grateful to my parents, who supported me both mentally and financially during my entire studies. Pap and mam, thank you for everything. I would like to thank Renske for encouraging me to get a master's degree, and for her love and support during my studies. Doing this thesis would not be possible without her.

I would like to thank both my mother and Petra for thinking along with me about impact dampening materials, and for providing these materials. And pap, thank you for reading my thesis multiple times before handing in, to provide feedback about my English grammar.

Lastly, I would like to thank you, the reader, for taking the time to read this thesis.

S.J. Uitendaal
Delft, March 2022

Contents

1

Paper

Teaching Robots Impact Tasks by Performing Demonstrations

Sven Uitendaal
Supervisors: Jens Kober, and Alessandro Saccon

***Abstract*—While robots execute many tasks where physical interaction with the environment is required, it is still challenging to control a robot that deliberately makes contact at a non-zero velocity, especially with multiple contact points that are impacted simultaneously. When there is a mismatch between planned and actual impact time, the robot typically does not respond as desired. In this paper, we demonstrate that an Impact-Aware Learning from Demonstration (IA-LfD) framework, that is based on Reference Spreading, can be developed and validated by physical experiments on real robots. The proposed IA-LfD framework is based on the following key aspects: (a) generating suitable ante-impact and post-impact tracking references from demonstrations; (b) development and validation of an impact detection mechanism to identify the contact transition, typically consisting of multiple impacts. The validation of the approach shows in particular the advantage of using an intermediate phase controller in reducing peak contact forces and oscillations during the dynamic contact transition, when compared to baseline approaches not using this controller. In addition, the validation highlights the role played by active/physical contact damping during the contact transition to improve execution performance.**

***Index Terms*—Learning from Demonstration, Reference Spreading, Impact.**

I. Introduction

SINCE the development of robots gets more advanced, robots take more place in our daily life. This can be seen in household machines, e.g. automatic vacuum cleaners, or in warehouse robots. Many tasks require physical interaction with the environment, e.g., running or picking up packages. Where robots used to be programmed manually to execute these skills, nowadays a more intuitive methodology to teach skills to robots is Learning from Demonstration (LfD) [1]. LfD refers to the process of transferring skills from a human teacher to a robot by performing demonstrations. Robots can learn skills directly from demonstrations, e.g., by fitting Movement Primitives (MPs) through the trajectories that a robot has carried out during demonstrations.

When a robot makes contact with its environment at a non-zero velocity, one or more impacts take place. The jumps in the velocity and peaks in the forces in the robot's trajectory, that result from the impacts, can cause a closed-loop controller to destabilize, due to an improper definition of the tracking error [2], [3]. In the case of a planned trajectory with jumps, the destabilization is caused by the unavoidable time mismatch between the nominal impact and the actual impact, causing a peak in the velocity error and in the difference between contact force and desired contact force, both causing a peak in the closed-loop commands. This destabilization is also referred as the 'peaking phenomenon' [2] and is visualized in Fig. 1a and 1c. This phenomenon can be avoided by establishing contact with a relative velocity of almost zero [4], but using a low velocity limits the performance. In tasks like stamping and hitting, impacts are required. In other tasks, like running, picking up objects, and poking objects, slowing down the robot will result in a longer execution time, and is contradictory to how humans naturally execute these kind of tasks.

Many tasks require not only to make an impact, but ideally to make it happen at different locations simultaneously. In practice, the ideal simultaneous impacts will not take place at exactly the same time, but slightly after another. Whereas for cases with single impacts, methods exist to predict the post-impact state [5], the state cannot be predicted during the time interval in which the impacts take place, since identifying the order of impacts is impossible to predict [6]. Since in reality one has to deal with 'practical-simultaneous' impacts, controlling a robotic system leads to further challenges to define the tracking error and to decide what type of control action one should employ.

In this paper, the Impact-Aware Learning from Demonstration (IA-LfD) framework, that is described and validated with a numerical simulation in [7], is tested on a real robot setup, by executing tasks that involve practical-simultaneous impacts. This framework makes use of an impact-aware control approach, called Reference Spreading (RS), which is described in more detail in Sec. II-C. To our knowledge this is the first research that demonstrates RS for ideally simultaneous impact tasks on a multi-DoF robotic arm in an experimental setting. In this approach, a trajectory is divided in an ante-impact, an interim, and a post-impact phase, and each phase has a different control strategy. Using RS in a real robot setup requires online detection of impacts to switch phases. A scenario is developed where the robot moves freely during the ante-impact phase, then impacts its environment and finally stays in contact during the post-impact phase. Two different tasks have been taught to the robot within this scenario. Demonstrations are performed via teleoperation, thus not distorting the impact dynamics (i.e. the natural inertia and damping parameters of the system) that would result from physically guiding a robot. Experiments are executed on a Franka Emika Panda 7-DoF robotic arm, and

S. Uitendaal and J. Kober are with the Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands. A. Saccon is with the Group of Dynamics and Control, Eindhoven University of Technology, Groene Loper 5, 5612 AE Eindhoven, The Netherlands. (email: s.j.uitendaal@student.tudelft.nl, J.Kober@tudelft.nl, A.Saccon@tue.nl).

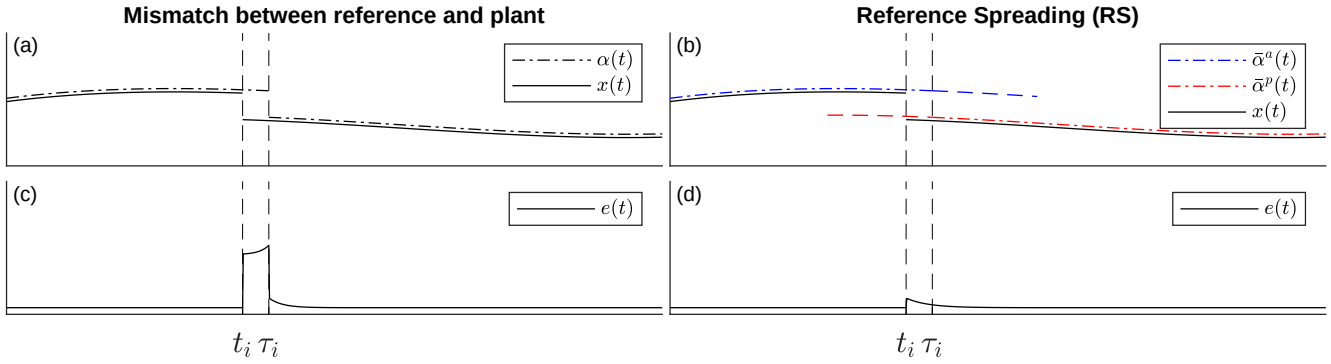


Fig. 1. A mismatch between nominal jump time τ_i in the reference α and jump time t_i of the plant x causes a peak in the tracking error $e(t) = \alpha(t) - x(t)$. RS solves this by using extended ante-impact and post-impact trajectories $\bar{\alpha}_a$ and $\bar{\alpha}_p$.

demonstrations are performed by recording the desired pose of the end-effector using an HTC VIVE Pro Controller 2.0, and by using this pose as reference for the robot.

In summary, this paper contributes to the development of the combination of LfD with RS by: 1) validating the implementation of the IA-LfD framework from [7] on a practical robot setup and showing the improvement in performance of using a separate interim phase strategy; 2) showing the beneficial effects of additional active damping during the interim phase to absorb vibrations, when damping is not naturally present in the physical structure of the robot and the environment; 3) developing a suitable impact-detection method to identify the beginning and estimate the end of the time interval in which practical-simultaneous impacts take place.

This paper is organized as follows: in Sec. II, the robot dynamics are described, and background information about RS and impact detection methods are described. In Sec. III the developed control approach and impact detection method are presented. Sec. IV shows our experiments, and our conclusions and recommendations are provided in Sec. V.

II. Background and Preliminaries

In this section, background information is provided about the robot dynamics, the IA-LfD control framework, and the state of the art of impact detection methods.

A. Robot Dynamics and Control

The dynamics of a rigid robotic arm with n joints are modeled as follows:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}_m - \boldsymbol{\tau}_{ext}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ are the joint positions, $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ the inertia matrix, $\boldsymbol{C} \in \mathbb{R}^n$ the combined Coriolis and centrifugal forces, $\boldsymbol{g} \in \mathbb{R}^n$ the gravity, $\boldsymbol{\tau}_m \in \mathbb{R}^n$ the motor torques and $\boldsymbol{\tau}_{ext} \in \mathbb{R}^n$ are the joint torques caused by the external forces acting on the end-effector. In free motion $\boldsymbol{\tau}_{ext} = \boldsymbol{0}$. The external Cartesian wrench $\boldsymbol{F}_{ext}$ (i.e. the forces and torques) that is acting on the end-effector can be calculated using the external joint torques by

$$\boldsymbol{F}_{ext} = (\boldsymbol{J}^\top)^\dagger \boldsymbol{\tau}_{ext}, \tag{2}$$

with $\boldsymbol{J}$ the (geometric) Jacobian, and $(\boldsymbol{J}^\top)^\dagger$ denoting the Moore-Penrose pseudoinverse of $\boldsymbol{J}^\top$.

With a robotic arm that is torque-controlled, the motor torques can compensate for gravity and Coriolis and centrifugal forces by choosing

$$\boldsymbol{\tau}_m = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{cmd}, \tag{3}$$

where $\boldsymbol{\tau}_{cmd}$ are the commanded joint torques. If a control equation is defined in operational space, the robotic arm can be controlled via

$$\boldsymbol{\tau}_{cmd} = \boldsymbol{J}^\top \boldsymbol{F}_{cmd} + \boldsymbol{\tau}_{ns}, \tag{4}$$

with $\boldsymbol{F}_{cmd}$ the commanded wrench in operational space, and $\boldsymbol{\tau}_{ns}$ the nullspace joint torques, that can be used to control the redundant degrees of freedom.

B. Impedance Control

Impedance control [8] is a control method that can be used in situations where a physical interaction with the environment is made. In addition, impedance control is able to absorb impacts [9]. The impedance control equation is built up in such a way that the end-effector is attached to a virtual spring-damper system, such that is pulled towards a goal position.

Using a combination of the formulations of [10] and [11], the control equation is defined as

$$\boldsymbol{F}_{cmd} = \boldsymbol{K}(\boldsymbol{x}_d - \boldsymbol{x}) - \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{F}_d, \tag{5}$$

which is impedance control with an additional feedforward term $\boldsymbol{F}_d$. The result is used in (4). It uses position and wrench references $\boldsymbol{x}_d$ and $\boldsymbol{F}_d$, and it solves the static spring-damper system with stiffness $\boldsymbol{K}$ and damping $\boldsymbol{D}$, and $\boldsymbol{x}$ is the current position of the end-effector. When there is a mismatch between desired wrench $\boldsymbol{F}_d$ and actual external wrench $\boldsymbol{F}_{ext}$, the equation is stabilized by the pose and velocity feedback, with a position offset dependent on the mismatch in wrenches.

C. Impact-Aware Control Framework

Current methods that can control a system that is exposed to impacts are changing the control parameters, for example by slowing down the system before impact to a relative velocity of (almost) zero [4], or to a maximum feasible ante-impact velocity [3], or by adapting the control gains [12], [13]. Other methods require a case specific 'glueing' function design [14], or are only suited for periodic impacts [15], e.g., a bouncing ball. Impedance control is suited for absorbing impacts, but is not impact-aware [9]. Then there are also hybrid control approaches, which use different strategies based on impacts that have or have not taken place. A method that mirrors a reference trajectory on an impacted surface requires an impact formulation of the environment [16]. Methods that do not need such a formulation of the environment use a different formulation for the tracking error [2], or make use of RS [7], [17]–[19].

The core idea of RS is to divide the reference trajectory into an ante-impact and a post-impact phase, based on the nominal impact time. The reference trajectories for both the ante-impact and the post-impact phase are extended in time, so that they partially overlap. As can be seen in Fig. 1, peaking is prevented by switching from ante-impact to post-impact reference trajectory directly when an impact has taken place. In case of practical-simultaneous impacts, an additional strategy can be used for the interim phase. In [6] and [7], an interim control strategy is used that only uses pure feedforward control during the interim phase, and [19] proposes the additional use of positional feedback.

D. Learning from Demonstration

Tasks with impacts can be taught to the robot by performing demonstrations. In [7], the trajectories of the end-effector in the demonstrations are used to create Probabilistic Movement Primitives (ProMPs) [20]. Before fitting ProMPs on the data, the trajectories are first extended around their nominal impact time. When executing the learned task, the ProMPs are used as reference trajectories.

E. Impact Detection Methods

The detection of impacts can be a challenging process. Even at low velocities, impacts take only a couple of milliseconds [3], so in order to gather sufficient information to detect impacts, data acquisition has to happen with a high sampling frequency, and the algorithm to detect impacts has to be fast to be able to process all this information and respond in time. Current methods detect impacts based on jumps in the velocity or based on force data.

Impacts cause abrupt changes in velocity signals. By identifying these changes, impacts can be detected. In [21], an impact detection method is described that puts a threshold on the difference between consecutive velocity datapoints. When this threshold is exceeded, an impact is detected. The method described in [22] detects jumps in the velocity by using position data and is called Jump-Aware (JA) filter. This method uses a window with an adaptive window length, and it uses the position measurements within this window to make a prediction for the next position value. An impact is detected if the difference between the prediction and the measured value of the next datapoint exceeds a certain threshold.

As explained in Sec. I, impacts are collisions with the environment, resulting in peaks in the forces that are acting on the robot. In [23], a walking robot is trained to detect impacts based on the Z component of the Cartesian force, calculated using the measured joint torques. First, a nominal force profile is recorded by executing the walking motion, while hanging above the ground. When the robot is walking on the ground, an impact is detected when the difference between measured and nominal force exceeds a certain threshold. Although this method is simple and model-free, it requires a large data storage, and it is situation dependent, as per type of movement a nominal force profile needs to be recorded. Several collision detection methods discussed in [24], are based on the external joint torques $\boldsymbol{\tau}_{ext}$ in (1). Since direct sensing of $\boldsymbol{\tau}_{ext}$ is usually not available on robotic arms, an estimation of $\boldsymbol{\tau}_{ext}$ can be made by using a momentum observer. The momentum observer described in [24] provides a first-order estimation of $\boldsymbol{\tau}_{ext}$. A robot is in physical contact with its environment if $|\boldsymbol{\tau}_{ext}| > \mathbf{0}$, with $|.|$ indicating the absolute value of each of the vector's indices, however due to disturbances in the external torque estimation, a threshold $\boldsymbol{\epsilon}$ can be used, such that a collision is detected if $|\hat{\boldsymbol{\tau}}_{ext}| > \boldsymbol{\epsilon}$, with $\hat{\boldsymbol{\tau}}_{ext}$ denoting the estimation of $\boldsymbol{\tau}_{ext}$, which is the output of the observer.

The author of [25] points out that such a method does not detect multiple impacts, when contact with the environment is maintained after the first impact. He uses the fact that since $\hat{\boldsymbol{\tau}}_{ext}$ is a first-order estimation, a peak in the external joint torques $\boldsymbol{\tau}_{ext}$ results in a step in the estimation $\hat{\boldsymbol{\tau}}_{ext}$, which is equivalent to a peak in the estimation rate $\dot{\hat{\boldsymbol{\tau}}}_{ext}$, thus an impact is detected when $|\dot{\hat{\boldsymbol{\tau}}}_{ext}| > \boldsymbol{\epsilon}$. Although the true value of $\dot{\hat{\boldsymbol{\tau}}}_{ext}$ is unknown, Euler differentiation of $\hat{\boldsymbol{\tau}}_{ext}$ can be used to estimate $\dot{\hat{\boldsymbol{\tau}}}_{ext}$. This is a similar approach to high-pass filtering $\hat{\boldsymbol{\tau}}_{ext}$, which deals with dynamic modelling errors, and provides more sensitive impact detection according to [24].

III. Method

Our method consists of an offline part and an online part. In the online part of the method, the robot is executing the task using impedance control, with a feedforward term during contact. The offline part is very similar to the RS framework in a LfD setting of [7]. Reference trajectories of ante-impact and post-impact phase are created using recorded demonstration data. The data of demonstrations are segmented into ante-impact and post-impact phases, based on the times of impact. Whereas [7] was assuming to know the times of impacts, since it was implemented in a simulation, our method relies on an impact detection method. As mentioned in Sec. II-C, reference trajectories are made by fitting ProMPs on the segmented demonstration data.

The motion of the robot is described as the position and orientation of the end-effector in the ante-impact phase $\boldsymbol{x}^a$ and $\boldsymbol{\theta}^a$, as well as the position, orientation, and contact wrench in

the post-impact phase $\boldsymbol{x}^p$, $\boldsymbol{\theta}^p$, and $\boldsymbol{F}^p$. Since the robot is freely moving in the ante-impact phase, it is assumed that the contact wrench $\boldsymbol{F}^a = \mathbf{0}$.

A. Segmentation and Alignment of Demonstration Data

In order to use RS, the demonstration datasets need to be segmented into the different phases. For creating ProMPs of each phase, the time domain of each dataset needs to cover the same time domain. Therefore the data of each phase is aligned in time, and to make the time domains of equal length, the datasets are trimmed. It is assumed that all the demonstrations are executed with more or less the same velocity.

1) Segmentation: The ante-impact phase starts at the beginning of the recording and ends one datapoint before the first impact, such that it is free of impacts. The post-impact phase starts at the time of the last impact of the practical-simultaneous impacts. This has the consequence that post-impact dynamics are still present in the post-impact phase, but this can be compensated by post-processing the demonstration data. The datapoints in the interim phase are not used in creating reference trajectories. The segmentation of data can be viewed in Fig. 2.

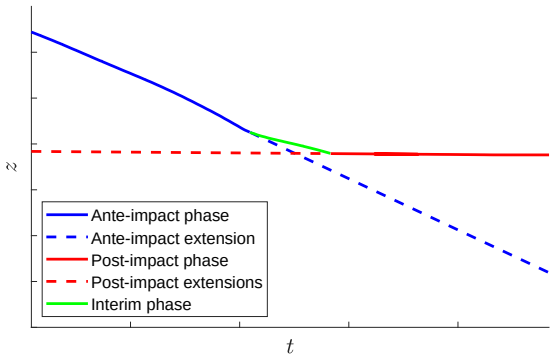


Fig. 2. Demonstration data is segmented into ante-impact, interim, and post-impact phase. The data of the ante-impact and post-impact are then extended around the nominal impact times.

2) Time-alignment: Whereas in the numerical simulation study of [7], information from the simulations could be used to align demonstrations, in a practical study we have to look for a common point in the recorded demonstrations. The demonstration datasets of both the ante-impact and post-impact phases all have one common point: the point in time when (initial/full) contact is established. As this is easy to implement, these points in time are used to align the datasets. Since the ante-impact phase ends when initial contact between the robot and environment is established, the data from the ante-impact phase are aligned such that the last datapoint of each dataset has the same timestamp. In the post-impact phase, full contact is established at the start of the phase, so datasets are aligned at the first datapoint. However, the alignment of demonstrations could be further analyzed, since the approach used does not take into account different contact locations and multiple ante-impact and post-impact phases in case of multiple contact establishments throughout the task.

3) Trimming Datasets: In order to fit ProMPs on the datasets, the time domain of each dataset should be of equal length. Time modulation techniques, such as introducing a phase variable [20] or Dynamic Time Warping [26], cannot be used, since the modulation of time distorts the velocity, and the velocity strongly influences the behaviour of the robot when establishing contact. Instead, the datasets are trimmed. Unnecessary datapoints are removed, such that the time domains from datasets of different demonstrations have equal length. For the ante-impact phase, the datapoints at the start of the trajectory are discarded, until each demonstration covers the same time domain. The post-impact phase is trimmed at the end of the demonstration data.

B. Extending Trajectories

To apply RS, the ante-impact and post-impact trajectories need to be extended around the nominal impact time for a sufficiently large time interval that accounts for the expected impact time uncertainty. Similar to [7], the wrench trajectories are extended using a zero order hold, while the position and orientation trajectories are extended using a first order hold. A first order hold position extension corresponds to a constant velocity extension, which encourages the system to impact with the same velocity as planned. It also ensures the establishment of contact. The same counts for the orientation extension. Similar reasoning is used for the extension of the wrench. Extending with a constant wrench encourages the system to apply the same wrench as planned after the impact has taken place. Additionally, whereas a linear wrench extension would lead to extremely high/low wrench commands when the mismatch in impact time is large and the wrench gradient is nonzero, this will not be the case with a constant wrench extension.

The data is extended by keeping the time step between extended datapoints equal to the average sampling period T_s of the recorded data. This is done to prevent the ProMP from overfitting on the extended data. The extension times can be described as

$$\bar{t}_k = t_{ext} + k \cdot T_s, \tag{6}$$

with for the ante-impact phase $t_{ext} = t_{L-1}$ is the last time step of the recorded demonstration, and $k \in \{1, 2, ..., N_{ext}\}$, with N_{ext} denoting the number of extended datapoints. For the post-impact phase $t_{ext} = t_0$ and $k \in \{-N_{ext}, -N_{ext} + 1, ..., 1\}$.

Whereas the position and orientation data of the ante-impact phase are extended using the constant velocity value at the end of the ante-impact trajectory, this cannot be done in the post-impact trajectory, since the velocity at the start of the post-impact trajectory is heavily influenced by post-impact vibrations, resulting in oscillatory behaviour in the velocity signal, as is pointed out in [5]. To eliminate these oscillations in the velocity signal, the velocity used for extending the post-impact trajectories is the approximated post-impact velocity

according to the method of [5], which uses around 3 oscillation periods of the post-impact velocity to make the approximation. The velocity data that is used for the approximation is then replaced by the approximated post-impact velocity.

C. Detecting Impacts

As mentioned in Sec. I, impacts take place when the robot collides with its environment, resulting in peaks in the contact force. In Sec. II-E, it is described that the external joint torque estimation by a momentum observer can be used to detect these peaks, by taking the Euler differentiation of the estimated external joint torques

$$\dot{\hat{\boldsymbol{\tau}}}_{ext,k} \approx \frac{\hat{\boldsymbol{\tau}}_{ext,k} - \hat{\boldsymbol{\tau}}_{ext,k-1}}{t_k - t_{k-1}}, \tag{7}$$

with t_k denoting the k-th time instant. At times when the estimation rate exceeds a threshold $\boldsymbol{\epsilon}$, an impact at joint j is detected if

$$\frac{\hat{\tau}^j_{ext,k} - \hat{\tau}^j_{ext,k-1}}{t_k - t_{k-1}} > \epsilon_j. \tag{8}$$

Although this method gives insight about which joints are affected by the impact, making a good choice for $\boldsymbol{\epsilon}$ can be challenging, since a value ϵ_j for each joint has to be determined, and tuning these values is not very intuitive, because the value for each joint depends on the robot's configuration. A more intuitive approach is to detect impacts in the operational space, since impacts take place in the robot's operational space. Given that $\boldsymbol{F}_{ext} = [\boldsymbol{f}^\top_{ext}\ \boldsymbol{m}^\top_{ext}]^\top$, with $\boldsymbol{f}_{ext}$ and $\boldsymbol{m}_{ext}$ the external force and torque vectors, and $\boldsymbol{F}_{ext}$ coming from (2), we propose to change (8) to

$$\frac{\|\boldsymbol{f}_k - \boldsymbol{f}_{k-1}\|}{t_k - t_{k-1}} > \epsilon, \tag{9}$$

with $\boldsymbol{f}_{ext}$ written as $\boldsymbol{f}$ for the sake of readability, and ϵ a bounding constant. By taking the magnitude of $\boldsymbol{f}$, the detection of impacts is direction independent. Since impacts acting on the robot only result in forces that are applied to the robot, the estimation rate of the external force magnitude $\dot{f}$, with $f = \|\boldsymbol{f}\|$, is positive when an impact takes place. Thus, in order to reduce the number of false positive impact detections, we can formulate that an impact is only detected under the condition of (9) and

$$\|\boldsymbol{f}_k\| > \|\boldsymbol{f}_{k-1}\|. \tag{10}$$

A drawback of the method (9), (10), is that the detection of impacts relies on only two consecutive datapoints, thus making it not robust against missing datapoints and noise. A solution is offered by the JA filter, which, as mentioned in Sec. II-E, uses an adaptive window of multiple datapoints for the detection of impacts. Although the JA filter is introduced as a method to find changes in the slope of the position signal, and thus steps in the velocity signal, we show that it can also be used to find jumps in the external force signal. In Fig. 3, it can be seen that when an impact takes place, a peak in the external force appears. When contact is then established, the external force stays constant. Since the momentum observer described in Sec. II-E provides a first-order external force estimation, the impulse results in a step response, after which the observer output converges to the constant external force value. Therefore, the JA filter can be used to detect jumps in the external force signal. In this method, a prediction for the current datapoint is made by using an adaptive window of previous datapoints. If the actual value differs too much from the prediction, a jump is detected. Namely, a jump in signal q at timestep k is detected when

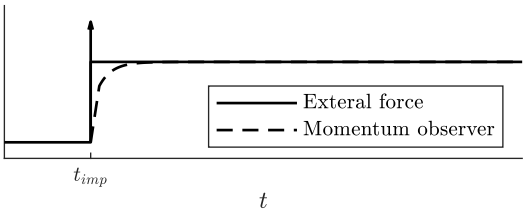


Fig. 3. While the actual external force is an impulse when an impact takes place at $t = t_{imp}$, the momentum observer output is a step, since it is a first order estimation.

$$\|q_k - p(\underline{q}_k)\| > b(\underline{q}_k), \tag{11}$$

with p and b the prediction and bounding functions, and $\underline{q}_k = [q_{k-m_k}, q_{k-m_k+1}, ..., q_{k-1}]$, where m_k is the current window length. The prediction $p(\underline{q}_k)$ is an extrapolation of $\underline{q}_k$, which can be retrieved by e.g. a polynomial fit. The window length updates each step according to

$$m_{k+1} \quad = \begin{cases} 0, & (11), \\ \min(M, m_k + 1), & \textit{otherwise}, \end{cases} \tag{12}$$

where M denotes the maximum window length. The detection of impacts using this method is visualized in Fig. 4.

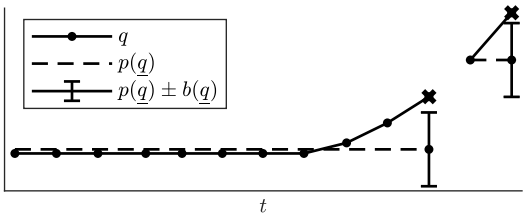


Fig. 4. JA filter: an impact is detected when the difference between datapoint q and prediction $p(\underline{q})$ exceeds the value of bound $b(\underline{q})$. Since an impact can take a few milliseconds, using the JA filter directly after a detection can result in a second detection of the same impact.

In (9), $\boldsymbol{f}_{k-1}$ can be used as a prediction p, whereas $[t_k - t_{k-1}]\epsilon$ can be used as a bound b. This method can incorporate multiple datapoints by choosing prediction function

$$p(\underline{\boldsymbol{f}}_k) = \bar{\underline{\boldsymbol{f}}}_k, \tag{13}$$

where $\bar{\underline{\boldsymbol{f}}}_k$ stands for the mean force vector of $\underline{\boldsymbol{f}}_k$, and bounding function

$$b(\underline{\boldsymbol{f}}_k) = \frac{t_k - t_{k-m_k}}{m_k}\epsilon, \tag{14}$$

with ϵ a bounding constant, and m_k is the window length as described in (12). The positive estimation rate can be incorporated by changing (10) to

$$\|\boldsymbol{f}_k\| > \|p(\underline{\boldsymbol{f}}_k)\|. \tag{15}$$

As mentioned in Sec. II-E, an impact can take a couple of milliseconds. As a consequence, the force rate can exceed the threshold for multiple consecutive timesteps, which can result in multiple detections of the same impact, as can be seen in Fig. 4. To prevent this, the JA filter can be turned off for a certain time interval after an impact detection. In order to achieve this, the window length m_k in (12) stays at 0 when

$$t_k - t_{imp} < T_{imp}, \tag{16}$$

where t_{imp} is the timestamp of the last detected impact, and T_{imp} is a time interval in which the JA filter is not used, because the previous detected impacts dynamics can still be measured in the data. In our JA filter, the window length update is defined by modifying (12) to

$$m_{k+1} = \begin{cases} 0 & \text{[(11) and (15)] or [(16)]} \\ \min(M, m_k + 1), & \textit{otherwise}. \end{cases} \tag{17}$$

D. Controller Structure

Since in the different phases (ante-impact, interim, and post-impact) the robot has a different contact state (free movement, contact partially established, and full contact), for each of the phases a different control strategy is proposed. Each of these strategies use the impedance control formulation of (5). Additional to the phase strategies, a strategy that determines when to switch from one phase to the next phase is proposed. In summary, the controller consists of a strategy for the 1) ante-impact phase; 2) interim phase; 3) post-impact phase; and it also contains 4) a strategy to switch from ante-impact to interim to post-impact phase.

1) Ante-Impact Phase: During the ante-impact phase, the robot is moving freely, while tracking the learned position trajectory. The commanded force (5) is used to track the position reference trajectory, with $\boldsymbol{F}_d = \boldsymbol{0}$, since the robot does not have to exert a force on the environment.

2) Post-Impact Phase: The robot is in full contact with its environment during the post-impact phase, meaning that it is exerting a force on the environment. The commanded force formulation (5) is used, where $\boldsymbol{F}_d$ is a learned desired force the robot has to exert, in addition to the attractor force to the reference position. This desired force is learned based on estimated external force data.

3) Interim Phase: As mentioned in Sec. I, a tracking error cannot be defined during the interim phase, since it is impossible to predict the order in which impacts take place. Assuming that when an impact takes place, the system is damped by the physical structure of the robot and the environment, an interim control strategy that uses solely position feedback is proposed in [19], where the extended ante-impact position reference is tracked during the interim phase. This means that $\boldsymbol{D} = \boldsymbol{0}$ in (5), and $\boldsymbol{F}_d = \boldsymbol{0}$ as in the ante-impact phase. However, when the physical structures of the robot and the environment lack damping, it might be a good idea to actively provide damping in the control equation to reduce oscillations. Therefore we propose to continue to track the ante-impact trajectory during the interim phase, while damping is actively provided in the control equation.

4) Switching Strategy: Whereas in a numerical simulation, such as [7] and [19], there can be switched from ante-impact to interim phase and from interim to post-impact phase by using knowledge of the simulation, in practical experiments we can only rely on the detection of impacts. Similar to the numerical simulations, the switch from the ante-impact to interim phase can be done at the detection of the first impact. However, switching from interim to post-impact phase is not so straightforward. Besides that the order in which the impacts takes is impossible to predict, multiple impacts can take place around the same time, so that they will be detected as one impact. In addition, the total number of impacts is also configuration and shape dependent, making it impossible to determine which impact detection corresponds to the end of the interim phase. Therefore we propose to use a switching strategy, in which the switching from interim to post-impact phase takes place at time $t_1 + T_{int}$, where t_1 is the time instant of the first detected impact, and T_{int} is a constant duration of the interim phase, of which the value can be determined based on demonstrations.

IV. Experimental Study

To validate the impact detection method that is described in Sec. III-C and the control approach from Sec. III-D, two different experiments have been conducted with a Franka Emika Panda 7-DoF robotic arm. In both the experiments, the robot had to learn ideal simultaneous impact tasks. An HTC VIVE Pro Controller 2.0 is used with two base stations to provide reference poses for the demonstrations.

A. Tasks and Setup

The two experiments are shown in Fig. 5. A wooden plank with a length of 70cm is used as end-effector. In the first experiment, the robot has the task to stamp on the table. After establishing contact with the table, the robot has to keep pushing down, so that contact with the table is maintained. In the second task, the robot has to wipe out a whiteboard. The vertical movement is similar to the motion of the first task. The addition of this experiment w.r.t. the first experiment, is the horizontal motion. The robot moves horizontally with an approximate constant velocity, but when contact with the

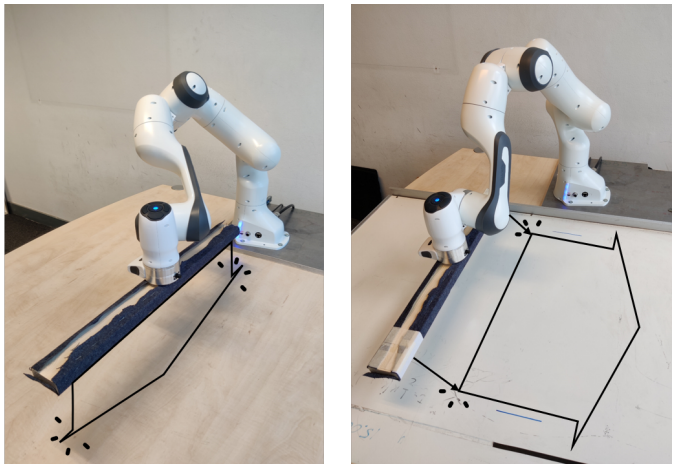


Fig. 5. From left to right experiments 1-2. The arrow shows the movement direction.

whiteboard is established, friction forces act on the robot in opposite direction of the horizontal movement. Part of the impact energy will be absorbed by the frictional forces, providing physical damping during the interim phase.

In the experiments, it is tried to create inelastic practical-simultaneous impacts, such that after initial contact between the robot and the environment, the end-effector does not bounce off to make contact again, but stays in contact. Soft covering reduces the impact [27], so the wooden plank is covered in cloth. In addition, the end-effector is moved with maximum feasible velocities, as impacting the environment with larger velocities can result in bouncing. By covering the plank in cloth, a sort of bone-flesh structure is created, which provides some damping.

For performing demonstrations, the impedance controller (5) with $\boldsymbol{F}_d = \mathbf{0}$ is used, where each of the components of the commanded force $\boldsymbol{F}_{cmd}$ is saturated to not exceed the value of 50N. The value of 50N is determined experimentally, such that the robot does not violate its joint torque limits when impacting, but can still exert a large force. This safety measurement prevents the robot from breaking itself or the environment by accident and it helps prevent the robot from shutting itself in safety mode for exceeding its joint torque limits.

Both during the demonstrations and the executions of the learned tasks the robot is controlled with a frequency of 500Hz. As mentioned in Sec. II-E, data acquisition has to happen with a high sampling frequency in order to detect impacts, since impacts take only a couple of milliseconds. This high frequency provides the JA filter with enough data to detect impacts, while the 2ms between two consecutive datapoints are sufficient to complete its calculations. In addition it allows the robot to respond with a delay of only 2ms to a detected impact.

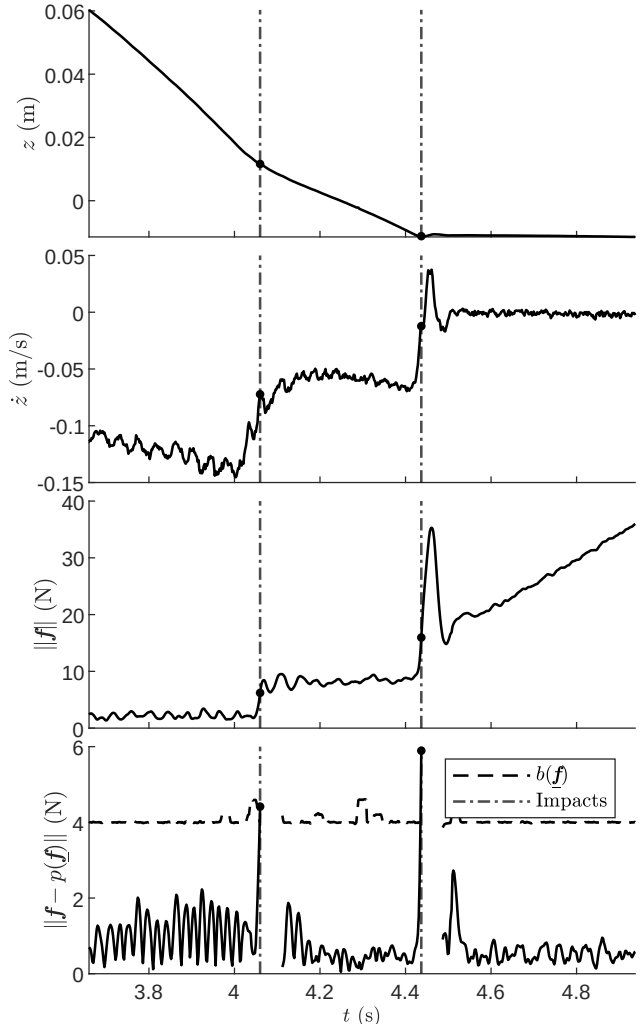


Fig. 6. Applying a JA filter with a window length of $M = 10$ on demonstration data of the stamping experiment results in the detection of 2 impacts. After the detection of an impact, there is a time window in which there are no prediction and bound available, since the JA filter is turned off after a detection.

B. Impact Detection

In order to use the switching strategy described in Sec. III-D, a method that detects the moment an impact has taken place is required. Impacts are detected under the conditions (11) and (15), using prediction (13) and bound (14). The software from Franka Emika is providing external force estimation by using the momentum observer described in [24]. In these estimations, the time from the start of an impact until a peak in the estimated force magnitude, easily takes more than 10 datapoints. With this knowledge, a JA filter with a maximum window length of $M = 10$ datapoints, and an impact duration of $T_{imp} = 25 \cdot 0.002\text{s} = 0.05\text{s}$ is applied. The bounding constant $\epsilon = 4.0/0.002 = 2000\text{N}\,\text{s}^{-1}$ is chosen such that the peaks in the difference between measurements and predictions $\|\boldsymbol{f} - p(\underline{\boldsymbol{f}})\|$ in Fig. 6 exceed the bounding value.

Fig. 6 shows that the detected impact times are close to the times of jumps in the velocity and the external force. Two clear peaks in the difference between measurements and predictions are visible around the jump times, indicating that with the right choice for ϵ, impacts will be detected at these points in time. Note that impacts are detected with a few milliseconds delay. The points of detection in Fig. 6 are at points in time when jumps in external force and velocity already have been (partially) made. The reason of this delay is that an impact is only detected when the difference between measurement and prediction exceeds the bounding value ϵ, which takes a few milliseconds after the jump has started.

Something else that can be seen in Fig. 6 at the bottom, is that at some points in time, the bounding value is a bit larger than the nominal value of $4.0\mathrm{N}$. This is due to a datastream that is not entirely stable, and some datapoints arrive later than the planned $2\mathrm{ms}$ sampling period after the previous datapoint. Due to the formulation of the bounding function, the bounding value is larger at these points, and the method is robust against this instability in the datastream. This robustness can be increased by using a larger window length, at the price of a larger delay.

Figure 7 shows the difference between measurements and predictions when the JA filter is applied without bounding function, i.e. no impacts are detected, so m_k in (17) is never reset to 0. In such a plot, the influences of tuning the JA filter and disturbances in the signal clearly can be seen. In Fig. 7, at the time of the second detected impact, at $t \approx 4.4\mathrm{s}$, multiple peaks show up, where in each peak the difference between measurement and prediction exceeds the bounding value of $4.0\mathrm{N}$ for more than one datapoint, indicating that if the JA filter was not turned off for some time after the detection of an impact according to (17), multiple detections of the same impact would have taken place.

Furthermore, some oscillatory behaviour in the difference between measurements and predictions can be seen in Fig. 6 at the bottom (especially until the first impact at $t \approx 4.1\mathrm{s}$). These oscillations are caused by noise in the external force estimation, and are undesired, since the bounding value has to be adjusted in order to not detect false positives. Whereas the robot starts in free motion, with no external forces applied, it can still be seen in Fig. 6 that the external force magnitude is not equal to $0\mathrm{N}$, and is oscillating. This inaccuracy in the external force estimation of the Panda robotic arm is addressed by [28] and can be caused by noise in sensors, inaccurate modeling of dynamics, and friction. Since we are looking at the external force rate, the offset is not a problem, but the oscillations are. In Fig. 7, the influence of the oscillations in the external force estimation to the difference between measurements and predictions can clearly be seen. Around the time of the first impact, at $t \approx 4.1\mathrm{s}$, the lower peaks, that are caused by these oscillations, reach values up to half the value of the first impact peak, making it difficult to choose an appropriate threshold that indicates whether a peak is an impact or not. Thus, with the provided external force estimations, impacts can only be detected with a suitable bounding value, and the jump in the external force estimation must be large enough to be distinguished from the estimation noise.

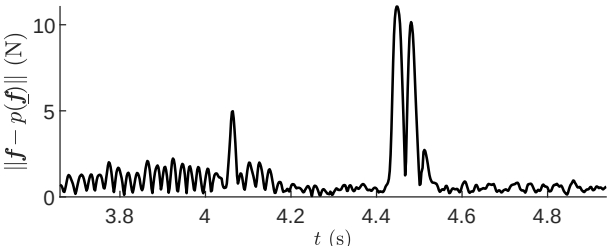


Fig. 7. The difference between measurements and predictions when using the JA filter of Fig. 6 without a bounding function.

C. Executing the Learned Task

To validate the use of tracking the ante-impact position reference in the interim phase as originally proposed in [19] on a real robot setup, and to show the effect of using active damping, the stamping and sliding tasks, illustrated in Fig. 5, will be executed using the learned and post-processed ante- and post-impact reference trajectories created as explained in Sec. III-A and III-B. We will compare the two controllers with an interim strategy, i.e. the two controllers described in Sec. III-D, of which one uses damping during the interim phase and one uses purely feedforward control, against a) an impact-unaware controller, which switches to the post-impact strategy at the nominal impact time, and is considered as baseline, and b) the classical RS controller [17], which switches directly to the post-impact phase at the detection of an impact, and has no intermediate phase. In both the stamping and the sliding task, the duration of the interim phase, as described in Sec. III-D, $T_{int} = 300\mathrm{ms}$.

To simplify the movement and focus only on the linear movement direction, in both of the experiments it is tried to maintain a constant orientation of the end-effector. In order to prevent undesired rotations from still happening due to recorded post-impact motions, we have hardcoded a rotational velocity of 0, and a desired torque trajectory of 0. Further research can be about investigating more complex movements, with a nonzero rotational velocity.

1) Stamping Task: As explained in Sec. IV-A, in the stamping task, the robot executes a vertical stroke movement to impact the table, after which it keeps pushing down. The associated plots can be seen in Fig. 8. The dashed-dotted lines are the references for the different strategies. It can be seen in Fig. 8 that the references switch to the post-impact phase at different points in time. The baseline (blue) reference switches first, because the nominal impact time is at $t \approx 3.8\mathrm{s}$. Second, the classic RS controller (red) reference switches to the post-impact phase, because the first impact for this execution takes place at $t \approx 3.9\mathrm{s}$. The controllers with an interim phase (yellow and purple) switch both at $t \approx 4.2\mathrm{s}$, which is $0.3\mathrm{s}$ after their first detected impact. The slight difference in switching times is due to different impact detection times of the first

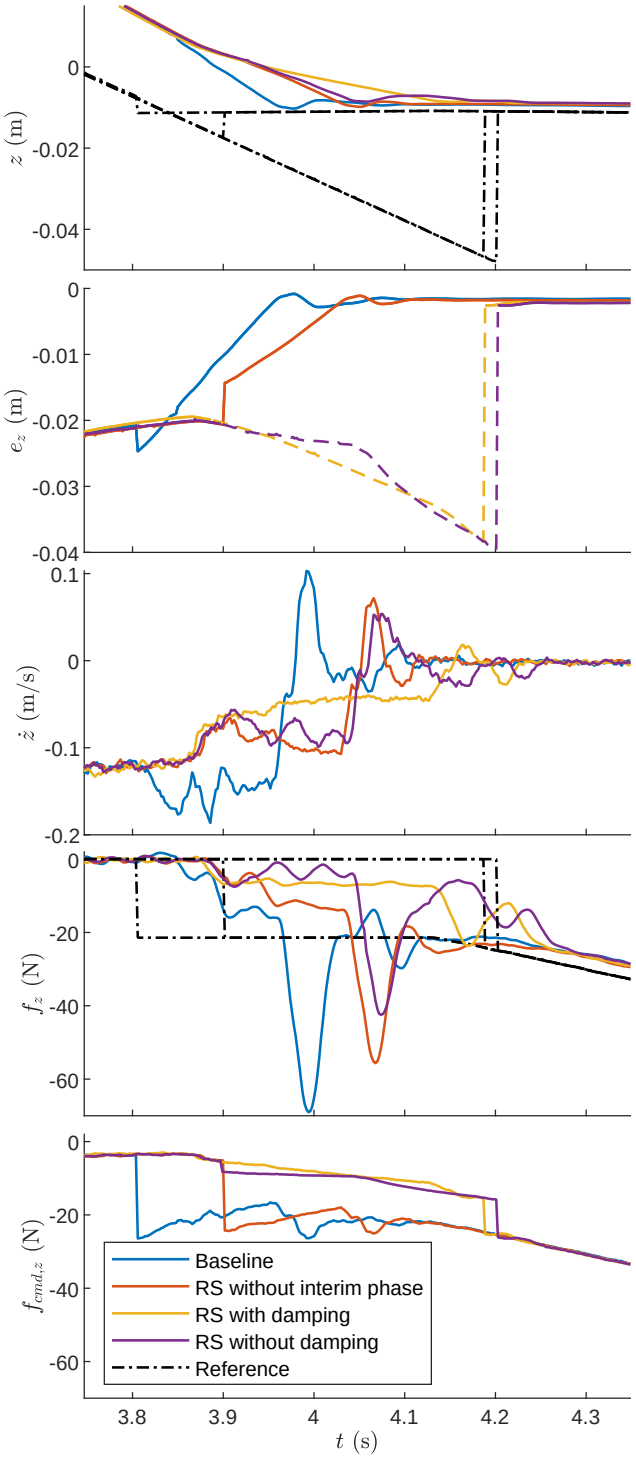


Fig. 8. Position, position error, velocity, contact force, and commanded force in the Z direction for the stamping task. The references for the position and force are shown in dash-dotted lines. The position error is the difference between reference and position. During the interim phase, the error is shown as a dashed line, because strictly speaking, this is not the actual error since the reference is not informative, but only helps to establish contact. The lag in the position with respect to the position reference is due to the fact that no velocity reference is used.

impact. This is inevitable, and can be caused by slightly different conditions, like slight difference in initial conditions, friction in the joints, and noise in the power supply.

The beneficial use of RS (red) in comparison with the baseline (blue) can clearly be seen in Fig. 8. The position error of the baseline controller converges quickly to 0. However, the post-impact force is applied too early, causing a large impact force and a large velocity overshoot at $t \approx 4.0$s, in comparison with the impact-aware strategies at $t \approx 4.1$s. It can also be seen that the end-effector bounces off the impacted environment when using the baseline controller, after the position error has reached a value of 0 at $t \approx 4.0$s. The same can be seen in the velocity plot, which shows a positive value at this time, indicating that the robot is moving away from the table. The same thing happens with the classical RS controller (red) at $t \approx 4.1$s, but to a lesser extent.

For the controllers that make use of an interim strategy, the idea behind removing the active damping in (5), is that the system is damped by the impacts. Under this assumption, applying only active stiffness, while tracking the position reference, should result in a smooth establishment of contact with the impacted environment. However, Fig. 8 shows oscillations in the velocity (third plot), and a peak in the contact force (fourth plot) at the moment of impact at $t \approx 4.1$s in the RS controller where damping is removed during the interim phase (purple). The velocity even reaches a positive value, indicating that the robot bounces off the impacted environment. This indicates that the robot's physical structure and the environment are not providing sufficient damping for a smooth establishment of contact. To compensate for this, active damping can provided by keeping a damping term in the control equation. It can be seen in the yellow plot that having an active damping term results in a much lower impact force and much smaller and less oscillations in the velocity signal when compared to the controller that uses no active damping in the interim phase.

Another interesting aspect of adding an interim control strategy can be seen when looking at the plot of the commanded force, that is $\boldsymbol{f}_{cmd}$ in $\boldsymbol{F}_{cmd} = [\boldsymbol{f}_{cmd}^\top \ \boldsymbol{m}_{cmd}^\top]^\top$ in (5). In Fig. 8 at the bottom, the Z component of $\boldsymbol{f}_{cmd}$ for the stamping task is plotted. It can be seen that applying the post-impact force reference before fully establishing contact, results in large jumps in $\boldsymbol{f}_{cmd}$. This is the case for the baseline controller (blue) at $t \approx 3.8$s and the classical RS controller (red) at $t \approx 3.9$s. Adding an interim phase where the extended ante-impact trajectory is tracked, helps to establish contact and to already exert a force on the environment, before switching to a post-impact strategy. This makes the jump in $\boldsymbol{f}_{cmd}$ significantly smaller when there is switched to the post-impact strategy at $t \approx 4.2$s. It can be seen in Fig. 8, that for the controller that does not use damping in the interim phase (purple), another jump in $\boldsymbol{f}_{cmd}$ takes place at $t \approx 3.9$s. This jump occurs due the fact that damping is removed at the beginning of the interim phase.

2) Sliding Task: As explained in Sec. IV-A, the sliding movement absorbs part of the impact energy, and this (par-

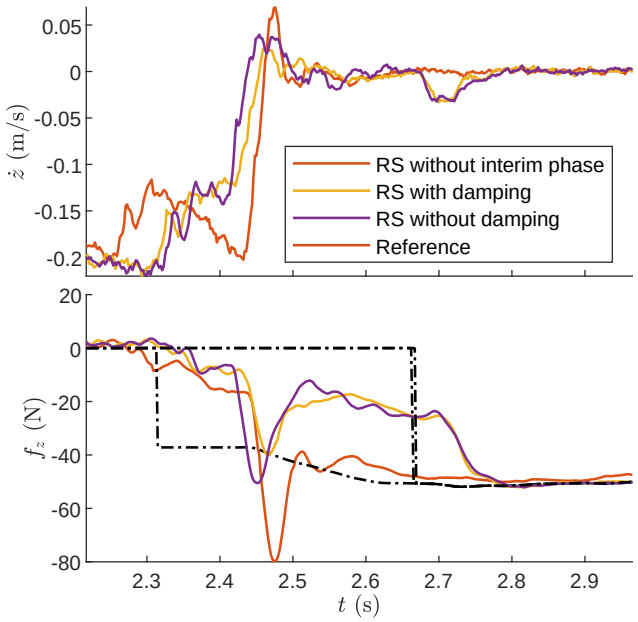


Fig. 9. Velocity and contact force in the Z direction for the sliding task.

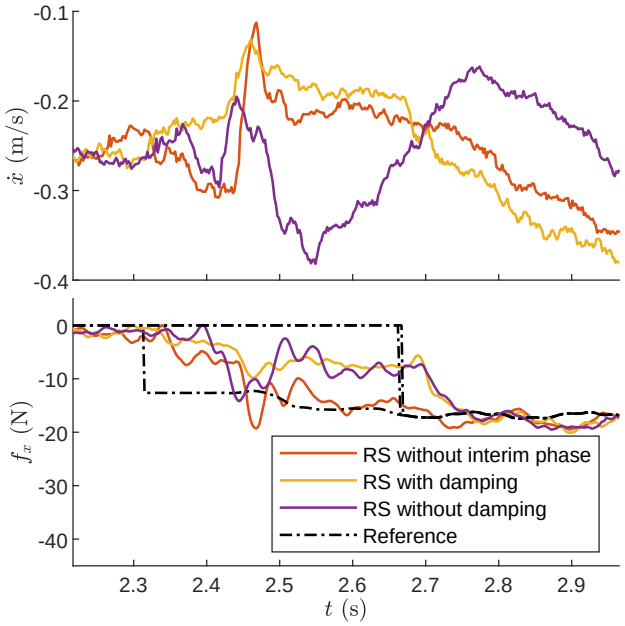


Fig. 10. Velocity and contact force in the X direction for the sliding task.

tially) damps the robot. The resulting velocities and forces in Z direction can be seen in Fig. 9. Since the ante-impact velocity is larger for this task than for the pushing task, the recorded contact forces are larger at their maximum values as well, especially at the baseline (not shown as it exceeds robot limits and experiment had to be stopped) and RS without interim phase (red). This large step in force reference makes the robot exceed its feasible joint torque limits when using the baseline controller, and therefore this could not be recorded. This does however show the usefulness of RS.

In Fig. 9, it can be seen that the physical damping has effect, because the velocity profiles of the two controllers with an interim phase strategy (yellow and purple) are almost identical. The force plots of the two controllers is also very similar, with a slightly larger impact force at $t \approx 2.45$s for the controller that does not actively use damping in the interim phase (purple). The trajectories in the X direction however are different. Figure 10 shows that the lack of active damping causes the velocity to reach larger values, and the force to oscillate, which might not be the desired behaviour. An idea could be to actively dampen the movement, except for the direction in which the impact takes place, under the condition that damping in that direction is provided sufficiently by the physical structures of the robot and the environment.

V. Conclusion and Future Work

In this paper, a validation of RS control is performed on a real robot setup, building up from the IA-LfD framework originally detailed in [7], and by filling in the missing components needed to translate this framework from numerical simulation to physical experiments. We have developed an impact detection method that can be used within the framework of RS. Furthermore, we have shown the effectiveness of using an interim phase in case of planned simultaneous impacts. We have also shown that when damping is lacking in the physical structures of the robot and the environment, the addition of active damping during the interim phase can help to smoothly establish contact between the robot and the environment, by reducing the peak force.

We think however, that the physical structure of the robot should provide damping when establishing contact. Future work can involve the design of end-effectors and robots that provide enough damping. In addition, since current provided external force signal is somewhat noisy, an improvement for the detection of impacts would be to have better external force estimations, to be capable of detecting impacts involving lower forces. Better results from the momentum observer can be achieved by modeling disturbances such as friction, e.g. with friction observer such as the one described in [29]. Also, it can be investigated how to use the original formulation of the JA filter [22] to detect impacts. In our research, this implementation gave issues with the instable datastream of the robot, where sometimes one or two datapoints were missing. If the original formulation of the JA filter is used, the missing datapoints have to be taken into account, or a more stable datastream has to be used. Further research in combining LfD with RS can be done. As mentioned in Sec. III-A, the time-alignment of different demonstrations could be further investigated. Other possible research can be about investigating more complex movements, involving a nonzero rotational velocity, or about making small changes in the trajectories, for example by using different starting and ending conditions, or about removing the time-dependence of reference trajectories,

e.g., by teaching the impact map to the robot based on demonstrations to predict the post-impact velocity.

REFERENCES

[1] S. Calinon, *Learning from Demonstration (Programming by Demonstration)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1007/978-3-642-41610-1_27-1

[2] J. J. B. Biemond, N. van de Wouw, W. P. M. H. Heemels, and H. Nijmeijer, "Tracking control of mechanical systems with impacts," in *2012 American Control Conference (ACC)*, 2012, pp. 258–263.

[3] Y. Wang, N. Dehio, A. Tanguy, and A. Kheddar, "Impact-aware task-space quadratic-programming control," 2020. [Online]. Available: https://arxiv.org/abs/2006.01987

[4] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.

[5] I. Aouaj, V. Padois, and A. Saccon, "Predicting the post-impact velocity of a robotic arm via rigid multibody models: an experimental study," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2264–2271.

[6] M. Rijnen, H. L. Chen, N. van de Wouw, A. Saccon, and H. Nijmeijer, "Sensitivity analysis for trajectories of nonsmooth mechanical systems with simultaneous impacts: a hybrid systems perspective," in *2019 American Control Conference (ACC)*, 2019, pp. 3623–3629.

[7] S. de Zwart, "Impact-aware learning from demonstration," MSc thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering (3mE), Delft Center for Systems and Control (DCSC), 2019. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Ac6f91fb2-2544-4802-bcda-4ee70ab0e2be

[8] N. Hogan, "Impedance control: An approach to manipulation: Part i - theory, part ii - implementation, part iii - applications," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–24, 03 1985.

[9] T. Senoo, M. Koike, K. Murakami, and M. Ishikawa, "Impedance control design based on plastic deformation for a robotic arm," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 209–216, 2017.

[10] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning—a review," *Frontiers in Robotics and AI*, vol. 7, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.590681

[11] S. Sidhik, M. Sridharan, and D. Ruiken, "Towards a framework for changing-contact robot manipulation," 2021. [Online]. Available: https://arxiv.org/abs/2106.10969

[12] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "Ilosa: Interactive learning of stiffness and attractors," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7778–7785, 2021.

[13] Y. Karayiannidis, L. Droukas, D. Papageorgiou, and Z. Doulgeri, "Robot control for task performance and enhanced safety under impact," *Frontiers in Robotics and AI*, vol. 2, p. 34, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2015.00034

[14] J. Kim, H. Cho, A. Shamsuarov, H. Shim, and J. H. Seo, "State estimation strategy without jump detection for hybrid systems using gluing function," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 139–144.

[15] L. Menini and A. Tornambe, "Asymptotic tracking of periodic trajectories for a simple mechanical system subject to nonsmooth impacts," *IEEE Transactions on Automatic Control*, vol. 46, no. 7, pp. 1122–1126, 2001.

[16] F. Forni, A. R. Teel, and L. Zaccarian, "Follow the bouncing ball: Global results on tracking and state estimation with impacts," *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1470–1485, 2013.

[17] M. Rijnen, A. Saccon, and H. Nijmeijer, "On optimal trajectory tracking for mechanical systems with unilateral constraints," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2561–2566.

[18] ——, "Reference spreading: Tracking performance for impact trajectories of a 1dof setup," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 1124–1131, 2020.

[19] J. J. van Steen, N. van de Wouw, and A. Saccon, "Robot control for simultaneous impact tasks via QP based reference spreading," 2022, accepted for IEEE international conference on Decision and Control (CDC). Preprint available: https://arxiv.org/abs/2111.05211.

[20] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf

[21] P. J. Cuijpers and M. A. Reniers, "Modeling an impact control strategy using hypa," in *Analysis and Design of Hybrid Systems 2006*, ser. IPV–IFAC Proceedings Volume, C. Cassandras, A. Giua, C. Seatzu, and J. Zaytoon, Eds. Amsterdam: Elsevier, 2006, pp. 56–63. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080446134500136

[22] M. Rijnen, A. Saccon, and H. Nijmeijer, "Motion signals with velocity jumps: Velocity estimation employing only quantized position data," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1498–1505, 2018.

[23] H. Park, Sangin Park, and S. Kim, "Variable-speed quadrupedal bounding using impulse planning: Untethered high-speed 3d running of mit cheetah 2," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5163–5170.

[24] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[25] B. Proper, "Aim-aware collision monitoring for robot-environment edge impacts," MSc thesis, Eindhoven University of Technology, Department of Mechanical Engineering, Dynamics and Control section, 2021, report number: DC 2021.081. [Online]. Available: https://research.tue.nl/en/studentTheses/aim-aware-collision-monitoring-for-robot-environment-edge-impacts

[26] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Autonomous Robots*, vol. 41, 03 2017.

[27] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[28] R. A. B. Petrea, M. Bertoni, and R. Oboe, "On the interaction force sensing accuracy of franka emika panda robot," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[29] L. Le Tien, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Friction observer and compensation for control of robots with joint torque measurement," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3789–3795.

A

From Demonstrations to Reference Trajectories

As explained in Sec. III-A, when demonstrations are recorded, the data of each demonstration is first segmented into an ante-impact, a post-impact, and an interim phase. The datasets are aligned based on the end of the ante-impact phase and the start of the post-impact phase, and then trimmed to cover the same timespan. After extending the trajectories for using Reference Spreading, Probabilistic Movement Primitives (ProMPs) are created to represent the trajectories.

To formulate the segmentation, the datapoints will be labeled according to their phase. Datapoints belonging to the ante-impact phase get the superscript ^{a}, whereas datapoints belonging to the post-impact phase get the superscript ^{p}. Datapoints in the interim phase get no superscript, as they are not used in creating reference trajectories.

Data of demonstration d consist of the Cartesian position $\boldsymbol{x}^d$ and velocity $\dot{\boldsymbol{x}}^d$, Cartesian orientation $\boldsymbol{\theta}^d$, and, only in the post-impact phase, contact force $\boldsymbol{f}^d$.

A.1. Segmentation and Alignment

As explained in Sec. III-A, the segmentation of data is done based on the time that impacts take place. The ante-impact phase starts at the beginning of the trajectory and ends at the start of the interim phase. The interim phase starts at the time of the first impact and ends one datapoint before the last impact. The last impact marks the start of the post-impact phase in the segmentation, so that enough data are available to estimate the post-impact velocity, which will be explained in Sec. A.2. The interim phase is not used to create reference trajectories.

A.1.1. Aligning Demonstrations

The datasets of demonstrations are aligned with the time when contact is partially/fully established as reference point, since this is a common point in time for all the data of the ante-impact phase and the post-impact phase.

Ante-Impact Phase

In the ante-impact phase contact is (partially) established at the end of the trajectory. The timestamp of each datapoint is amended such that the impact takes place at $t_i = 0$. This results in

$$\hat{t}_k^{a|d} = \tilde{t}_k^{a|d} - \tilde{t}_{\hat{L}_{a|d}-1}^{a|d} \quad k \in \{0, \hat{L}_{a|d} - 1\}, \tag{A.1}$$

with $\tilde{t}_k^{a|d}$ denoting the recorded timestamp of the datapoint at timestep k in the ante-impact phase of demonstration d, $\hat{L}_{a|d}$ is the number of datapoints in the ante-impact phase before trimming, and $\hat{t}_k^{a|d}$ is the time-aligned timestamp before trimming.

Post-Impact Phase
A similar approach is used for the post-impact phase, but contact is here fully established at the start of the trajectory. The time-alignment is now formulated as

$$\hat{t}_k^{p|d} = \tilde{t}_k^{p|d} - \tilde{t}_0^{p|d} \quad k \in \{0, \hat{L}_{p|d} - 1\}, \tag{A.2}$$

where $\tilde{t}_k^{p|d}$ denotes the recorded timestamp of the datapoint at timestep k in the post-impact phase of demonstration d, $\hat{L}_{p|d}$ is the number of datapoints in the post-impact phase before trimming, and $\hat{t}_k^{p|d}$ is the time-aligned timestamp before trimming.

A.1.2. Trimming Data
In order to create ProMPs, the datasets are trimmed to cover the same time domain. Trimming is done based on the dataset with the shortest time interval, that is the time interval from the start until the end of the dataset.

Ante-Impact phase
For the ante-impact phase, the dataset with the shortest time interval is the dataset with the last starting time. The trimmed ante-impact phase of demonstration d consist of datapoints that fulfill the condition

$$\hat{t}_k^{a|d} \geq \max_d \hat{t}_0^{a|d} \quad k \in \{0, \hat{L}_{a|d} - 1\}. \tag{A.3}$$

The resulting ante-impact dataset now has $L_{a|d}$ datapoints.

Post-Impact phase
A similar approach is used for the post-impact phase, but now the dataset with the shortest time interval is the dataset with the first ending time. The trimmed post-impact phase of demonstration d consists of datapoints that fulfill the condition

$$\hat{t}_k^{p|d} \leq \min_d \hat{t}_{\hat{L}_{p|d}-1}^{p|d} \quad k \in \{0, \hat{L}_{p|d} - 1\}. \tag{A.4}$$

The resulting post-impact dataset now has $L_{p|d}$ datapoints.

A.2. Extending Trajectories
In order to apply RS, the ante-impact and post-impact trajectories need to be extended, so that they overlap during the time interval where impacts can be expected. As explained in Sec. III-B, for extending the trajectories, mainly the ideas of [10] have been used. In that approach, the position and velocity trajectories are extended by using a constant velocity, whereas the force trajectories are extended by keeping a constant value. Extending the velocity trajectories with a constant value will keep the ante-impact and post-impact velocities the same as in the demonstrations, even though there is a mismatch in nominal impact time and actual impact time. Contact establishment is ensured by extending the position with a constant slope, corresponding to the extended velocity. The constant extension of force trajectories is done due to similar reasoning. The post-impact force will remain the same, even when there is a mismatch between nominal impact time and actual impact time.

In the demonstration has been tried to maintain a constant orientation. However, around the time of impacts the end-effector of the robot is likely to rotate a bit, due to the impacts. If these orientations are extended using a constant slope, then over time the difference between the extended orientation and our desired, constant orientation can become large. Therefore the orientation is extended using a constant value.

Furthermore, we noticed that the velocity trajectories show oscillations at the start of the post-impact phase. The velocity is not yet converged to its steady-state value, which makes extensions using the values of the first datapoint not redundant. A method that estimates the post-impact velocity [2], that uses roughly three periods of oscillations, is used to estimate the post-impact velocity, which will be used to extend the position and velocity data. After these three oscillation periods the data is converged better to its steady state value. Therefore the position and velocity data in this time interval are replaced by their extended versions.

A.2.1. Extended Datapoints

The number of extended datapoints is

$$N_{ext} = \lceil T_{ext} \cdot f_s \rceil, \tag{A.5}$$

with T_{ext} the extension time interval, f_s the sampling frequency of the recorded data, and $\lceil \rceil$ that stands for ceiling the result. The timestamps of these datapoints are divided evenly, with a time difference of $1/f_s$ between them. In the ante-impact phase the timestamps of the extended trajectory are

$$\bar{t}_k^{a|d} = \begin{cases} t_k^{a|d} & k \in \{0, L_{a|d} - 1\}, \\ t_{L_{a|d}-1}^{a|d} + [k - L_{a|d} + 1] \cdot T_s & k \in \{L_{a|d}, L_{a|d} + N_{ext} - 1\}, \end{cases} \tag{A.6}$$

where $T_s = 1/f_s$ is the sampling period. The timestamps of the extended post-impact trajectory are

$$\bar{t}_k^{p|d} = \begin{cases} t_0^{p|d} + k \cdot T_s & k \in \{-N_{ext}, -1\}, \\ t_k^{p|d} & k \in \{0, L_{p|d} - 1\}. \end{cases} \tag{A.7}$$

In the post-impact position and velocity trajectories, the datapoints in the first three oscillation periods are replaced by their extended version. With the time interval of three oscillations T_{fit}, the selection of datapoints in this interval fulfill the condition

$$t_k^{p|d} \leq T_{fit} \quad k \in \{0, L_{p|d}\}. \tag{A.8}$$

If the number of datapoints in this selection is N_{fit}, then the post-impact phase has $N_{ext}+N_{fit}$ extended datapoints in the position and velocity trajectories.

A.2.2. Extending Data with a Constant Value

The trajectories of the velocity, orientation, and contact force are all extended using a constant value. Dataset $y^{a|d}$ of the ante-impact trajectory of demonstration d can be extended using a constant value, which is the datapoint just before the interim phase $y_{ext}^{a|d} = y_{L_{a|d}-1}^{a|d}$. Using this constant value, the extended ante-impact trajectory is formulated as

$$\bar{y}_k^{a|d} = \begin{cases} y_k^{p|d} & k \in \{0, L_{a|d} - 1\}, \\ y_{ext}^{a|d} & k \in \{L_{a|d}, L_{a|d} + N_{ext} - 1\}. \end{cases} \tag{A.9}$$

A similar approach is used for the post-impact trajectory. Using post-impact value $y_{ext}^{p|d}$, the extended post-impact trajectory for the orientation and force is formulated as

$$\bar{y}_k^{p|d} = \begin{cases} y_{ext}^{p|d} & k \in \{-N_{ext}, -1\}, \\ y_k^{p|d} & k \in \{0, L_{p|d} - 1\}. \end{cases} \tag{A.10}$$

The first datapoint of the post-impact phase can be taken as constant extension value, $y_{ext}^{p|d} = y_0^{p|d}$.

As mentioned, in the post-impact velocity trajectory the datapoints of the first three oscillations are replaced by its extended version. The extension of the post-impact velocity trajectory is formulated as

$$\bar{\boldsymbol{v}}_k^{p|d} = \begin{cases} \boldsymbol{v}_{ext}^{p|d} & k \in \{-N_{ext}, N_{fit} - 1\}, \\ \boldsymbol{v}_k^{p|d} & k \in \{N_{fit}, L_{p|d} - 1\}, \end{cases} \tag{A.11}$$

with $\boldsymbol{v}_{ext}^{p|d}$ an approximation of the post-impact velocity. The method to get this approximation is explained in Sec. A.2.4.

A.2.3. Extending Position Data

The position trajectories can be extended using a constant velocity. For the ante-impact trajectory of demonstration d, this can be done by using the velocity just before the start of the interim phase $\boldsymbol{v}^{a|d} = \dot{\boldsymbol{x}}_{L_{a|d}-1}^{a|d}$. The extended ante-impact position trajectory of demonstration d can now be formulated as

$$\bar{\boldsymbol{x}}_k^{a|d} = \begin{cases} \boldsymbol{x}_k^{a|d} & k \in \{0, L_{a|d} - 1\}, \\ \boldsymbol{x}_{L_{a|d}-1}^{a|d} + \boldsymbol{v}^{a|d} \cdot \bar{t}_k^{a|d} & k \in \{L_{a|d}, L_{a|d} + N_{ext} - 1\}. \end{cases} \tag{A.12}$$

The post-impact trajectory can be extended by using the post-impact velocity $\boldsymbol{v}^{p|d}$. The method to give an approximation of this is explained in Sec. A.2.4. Using this post-impact velocity, the extended post-impact position trajectory can be formulated as

$$\bar{\boldsymbol{x}}_k^{p|d} = \begin{cases} \boldsymbol{x}_{N_{fit}}^{p|d} - \boldsymbol{v}^{p|d} \cdot [\bar{t}_{N_{fit}}^{p|d} - \bar{t}_k^{p|d}] & k \in \{-N_{ext}, N_{fit} - 1\}, \\ \boldsymbol{x}_k^{p|d} & k \in \{N_{fit}, L_{p|d} - 1\}. \end{cases} \tag{A.13}$$

A.2.4. Approximation of the Post-Impact Velocity

Due to the oscillations in the velocity signal after an impact has taken place, taking the velocity value at a time directly after the impact for extending the trajectories is not very reliable. Taking the velocity value of a slightly different timestamp can result in a totally different value. A good alternative would be using the velocity value of the rigid-body response.

An approximation of the post-impact velocity of the rigid-body response can be retrieved using the method of [2]. On each of the components of the Cartesian velocity of $\dot{\boldsymbol{x}}_k^{p|d}$, with $k \in \{0, N_{fit}\}$, the fitting function

$$f_{fit}(t, \boldsymbol{p}) = v^- + at + A(e^{\gamma t} \cos(\omega t + \phi) - \cos(\phi)) \tag{A.14}$$

is fitted, where t denotes the time after the impact, v^- the ante-impact velocity, and $\boldsymbol{p} := (a, A, \gamma, \omega, \phi)$ is the set of optimization parameters. Because the post-impact trajectory starts at the time of the last impact of the interim phase, the value of the first datapoint is used as ante-impact velocity v^-.

After optimizing the parameters $\boldsymbol{p}$ via SciPy's nonlinear optimization function `curve_fit` [9] the virtual rigid-body response is retrieved by

$$v_{rb}(t) = v^- - A\cos(\phi) + at, \tag{A.15}$$

and the value at impact time $v_{rb}(0) = v^- - A\cos(\phi)$ is considered as the post-impact velocity.

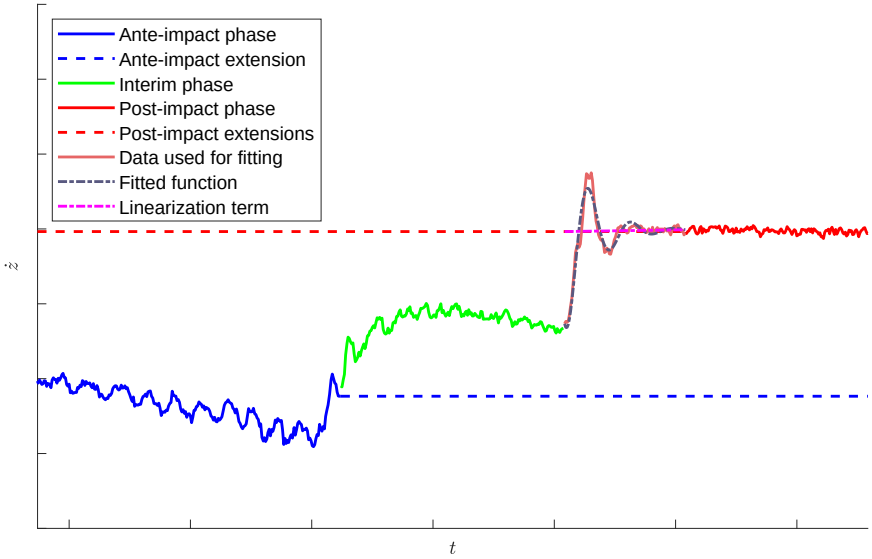


Figure A.1: The segmentation of velocity data into ante-impact, interim, and post-impact phase. The ante-impact and post-impact trajectories are extended around their nominal impact time. A part of the post-impact data is used to estimate the post-impact velocity. This part is replaced by an extended version, due to its oscillatory behaviour.

A.3. Creating ProMPs

ProMPs are introduced by [5] as a general probabilistic framework for representing and learning movements. A ProMP represents the movement as a distribution of demonstrations. In addition to that, it is possible to define different start and goal positions, as well as via-points. Using the ProMP, a different trajectory can be calculated with these via-points, while taking into account the variability over the demonstration. For fitting a ProMP, it is important for the demonstrations to be time-aligned.

ProMPs are used to represent the reference trajectories of the position, orientation and contact force of the robot's end-effector. The velocity data from demonstrations is used to improve the accuracy of the position ProMPs. Separate ProMPs are created for both the ante-impact and the post-impact phase.

A.3.1. Formulation of ProMPs

In order to create a ProMP, a function that consist of weighted basis function is fitted on the data of each demonstration. The Z weights $\boldsymbol{w}^d$ are optimized such that demonstration dataset y^d can be approximated by

$$\tilde{\boldsymbol{y}}^d(t) = \boldsymbol{\Phi}^\top(t)\boldsymbol{w}^d, \tag{A.16}$$

with Z basis functions $\boldsymbol{\Phi}(t) = [\boldsymbol{\phi}_1(t)^\top, ..., \boldsymbol{\phi}_Z(t)^\top]^\top$. The set of basis functions $\boldsymbol{\phi}_z(t) = [\phi_z(t)]$ when only the signal $\boldsymbol{y} = [y]^\top$ itself is known, but can also be extended with derivatives e.g. $\boldsymbol{\phi}_z(t) = [\phi_z(t), \dot{\phi}_z(t)]$ when the first order derivative $\boldsymbol{y} = [y, \dot{y}]^\top$ is also known.

The optimal values for the weights of demonstration d can be calculated via

$$\boldsymbol{w}^d = \boldsymbol{\Phi}_t^\dagger \boldsymbol{y}_t^d, \tag{A.17}$$

where $\boldsymbol{y}_t^d = [y^d(0), ..., y^d(T)]^\top$ and $\boldsymbol{\Phi}_t^\dagger$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{\Phi}_t$. With the weights for each demonstration calculated, the ProMP can approximate the mean trajectory with

$$\tilde{y}(t) = \boldsymbol{\Phi}^\top(t)\boldsymbol{\mu}_w, \tag{A.18}$$

with $\boldsymbol{\mu}_w$ denoting the mean set of weights.

A.3.2. Radial Basis Functions

Radial Basis Functions (RBFs) are basis functions that are often used for stroke movements in ProMPs [5]. The zth RBF is formulated as a Gaussian

$$\phi_z(t) = e^{-\frac{(t-c_z)^2}{2h}}, \tag{A.19}$$

with center c_z and width h. The centers of the different RBFs can be evenly distributed over the time interval of the ProMP.

A.3.3. The derivative of the ProMP

The use of a combination of different orders of derivatives results in a better approximation of the original signal [4]. For using the first order derivative of the data, the set of basis functions $\boldsymbol{\phi}_z(t) = [\phi_z(t), \dot{\phi}_z(t)]$. The derivative of RBF $\phi_z(t)$ is

$$\dot{\phi}_z(t) = -\frac{t - c_z}{h}\phi_z(t). \tag{A.20}$$

When the first order derivative is used to create the ProMP, the ProMP can also be used to give an accurate approximation of the first order derivative.

A.4. Representation of Orientation

For the creation of reference trajectories, the orientation of the end-effector is represented in Euler angles. Even though the orientation is almost constant throughout the entire trajectory, small changes of the robot's configuration can result in discontinuities in the Euler angles, where they switch from π to $-\pi$ or vice versa. Such a discontinuity is problematic for the creation of ProMPs, and a continuous signal is required.

A solution is offered by rotation matrices. Whereas a rotation matrix is made up of 9 different values, only 6 of them are needed to represent the rotation. These 6 values are the sin and cos of the Euler angles, which are continuous signals. Using these values, the rotation matrix can be calculated according to

$$\boldsymbol{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{A.21}$$

In stead of using 3 Euler angles, a reference trajectory now has to be created for 6 variables. A better approach would be to normalize the orientation, such that the Euler angles stay close to a value of 0. This can be accomplished by using the initial orientation as frame of reference. The normalized orientation can be formulated as

$$\hat{\boldsymbol{R}} = \bar{\boldsymbol{R}}^{-1}\boldsymbol{R}_k, \tag{A.22}$$

with $\hat{\boldsymbol{R}}$ the rotation matrix representing the normalized orientation, $\bar{\boldsymbol{R}}$ a normalization rotation matrix representing the frame of reference, and $\boldsymbol{R}_k$ the rotation matrix at timestep k. As normalization matrix, the initial rotation matrix $\boldsymbol{R}_0$ can be chosen, such that $\bar{\boldsymbol{R}}^{-}1\boldsymbol{R}_0 = I$, I denoting the identity matrix. If during the trajectory the end-effector stays close to its initial orientation, the Euler angles of the normalized orientation can be used to create reference trajectories, as no discontinuities take place anymore. Besides the normalized Euler angles, the normalization matrix needs to be remembered to convert normalized orientation back to the robot's operational space.

A.5. ProMPs from Demonstration Data

The extension of recorded demonstration data is done via the method described in Sec. A.2, with an extension time interval of $T_{ext} = 2.0s$, which is sufficiently large. For the approximation of the post-impact velocity, a time interval of $T_{fit} = 200ms$ is used, because $200ms$ roughly corresponds to three oscillation periods of the velocity data. For each ProMP there are 70 RBFs used per second, with evenly divided centers c_z over the time interval of the ProMP. They all have a width $h = 0.25ms$. A number of resulting ProMPs can be seen in Fig. A.2 for the stamping task, and Fig. A.3 and A.4 for the sliding task. The orientation is represented as normalized orientation, as described in Sec. A.4. The normalization rotation matrix is chosen to be the initial rotation matrix of the first demonstration.

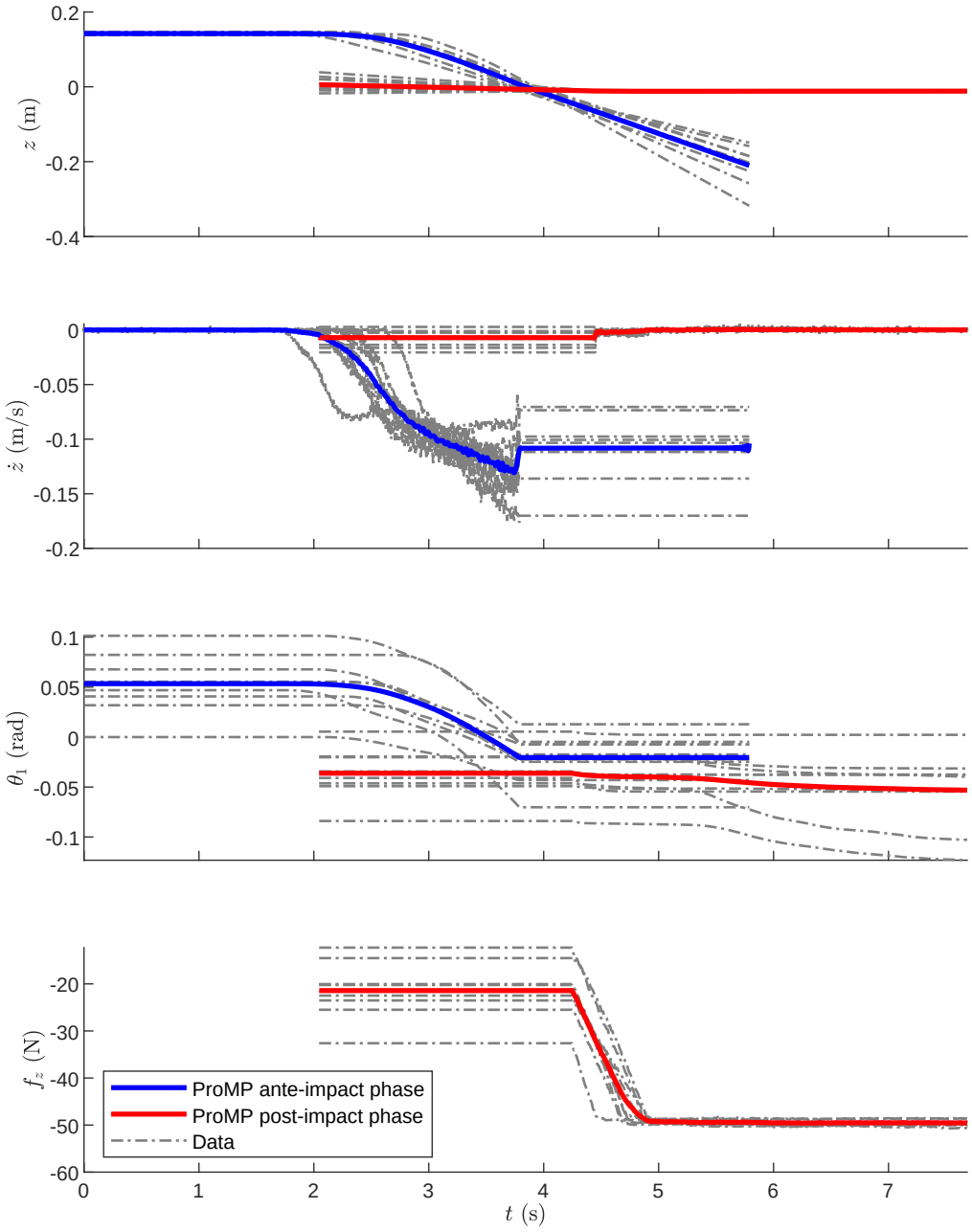


Figure A.2: ProMPs of the stamping task for the position, velocity and force in the Z direction, and for the first component of the normalized orientation.

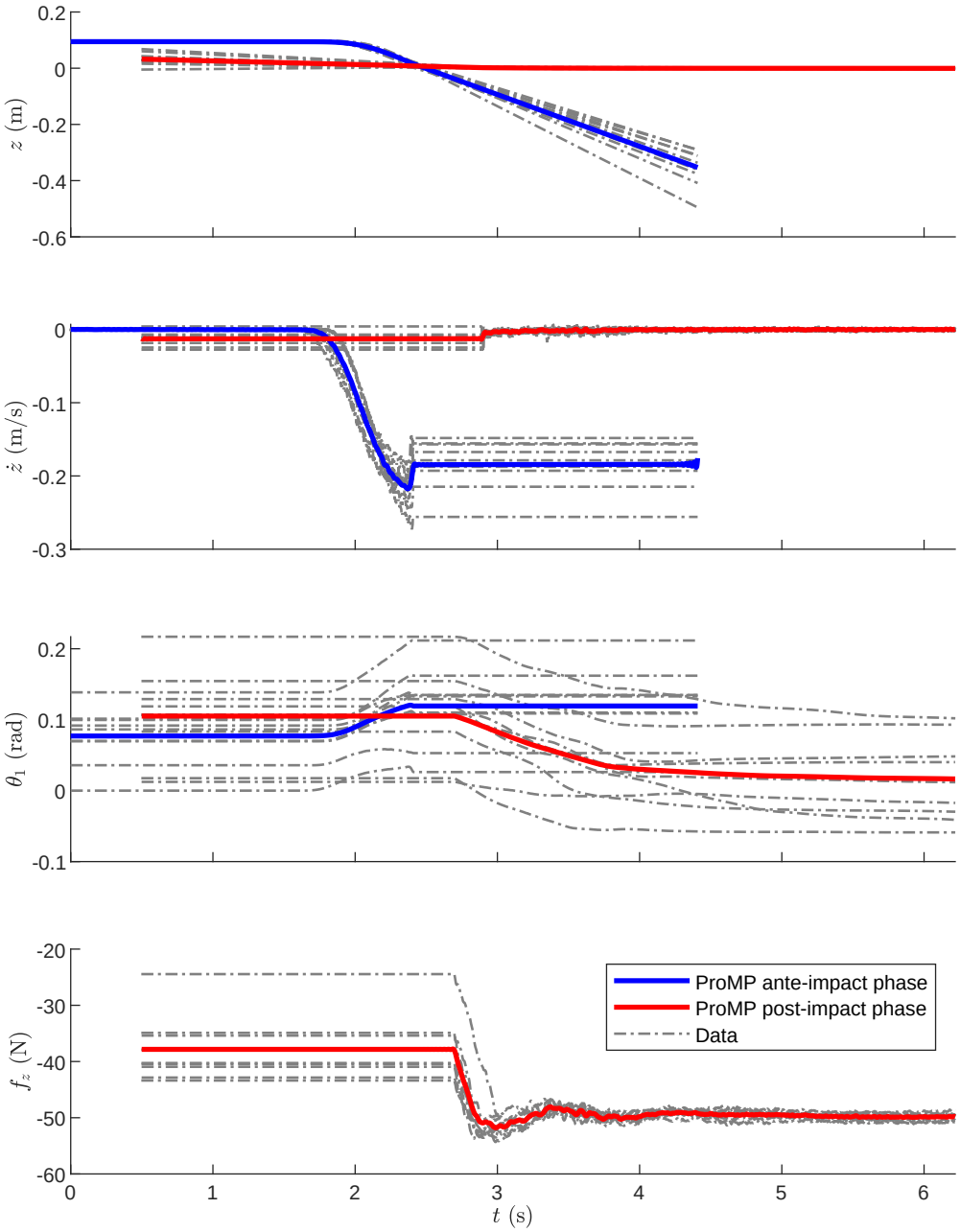


Figure A.3: ProMPs of the sliding task for the position, velocity and force in the Z direction, and for the first component of the normalized orientation.

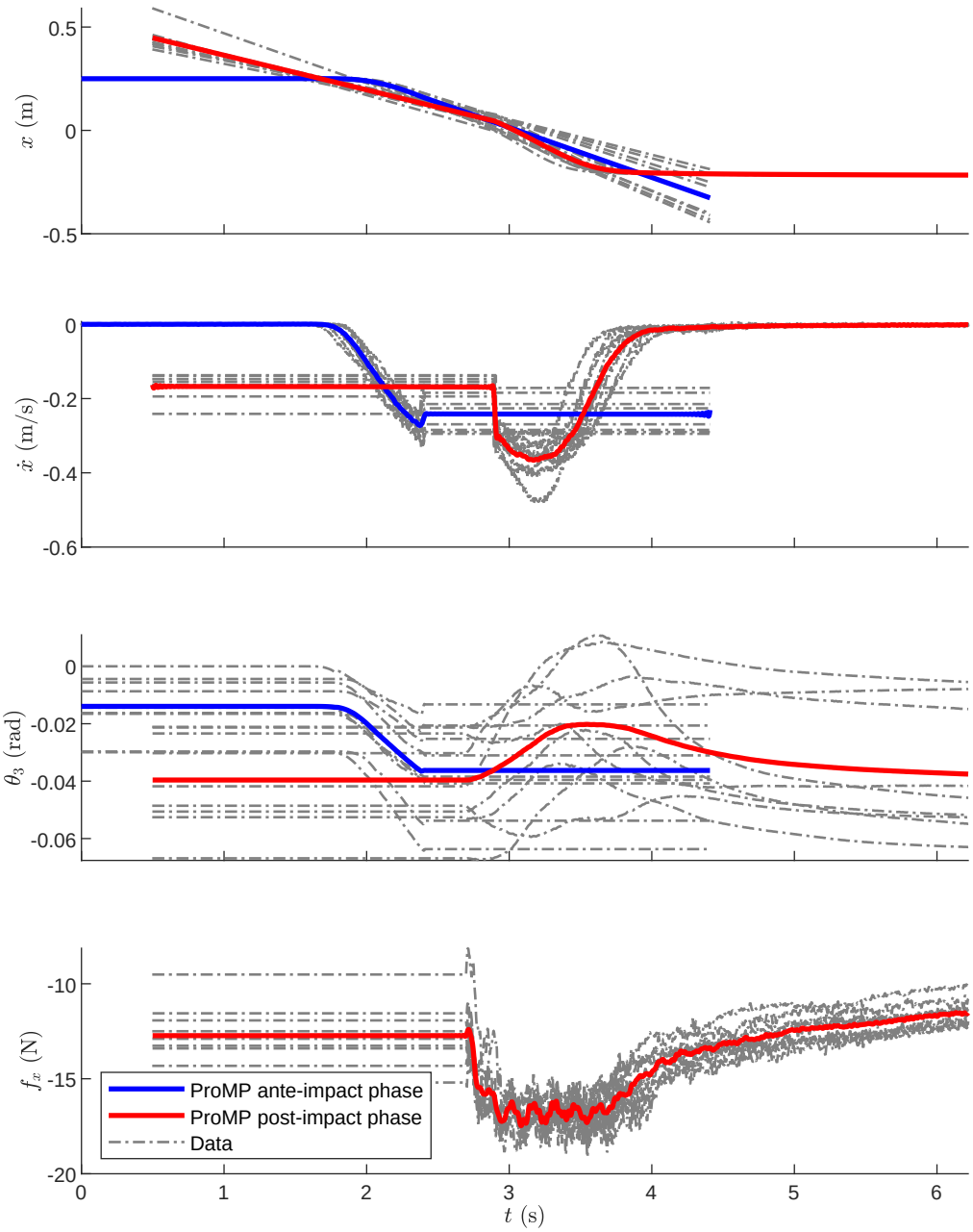


Figure A.4: ProMPs of the stamping task for the position, velocity and force in the X direction, and for the third component of the normalized orientation.

B

Impact Detection

The method that is used to detect impacts is explained in Sec. III-C. Before this method was developed, other methods have been tried to detect impacts. These methods and their results are presented in this chapter. Finally some additional plots of the current method are shown.

The plots in this chapter are based on the data of one of the demonstrations of the stamping experiment. As explained in Sec. IV-A, in this task, the end-effector moves vertically, until it stamps on a table. Velocities and forces act especially in the Z direction. The position, velocity and force plot of this dataset are shown in Fig. B.1. It can be seen that jumps in the velocity and force take place at $t \approx 4$s and $t \approx 4.5$s. These times correspond to the changes in the slope of the position plot.

B.1. Early Attempts

Early attempts of detecting impacts using a Jump-Aware filter, have been made by finding jumps in the velocity data using Cartesian position data, and by using external force data. The attempts make either use of the JA filter [7] or the force rate [6], that are both explained in Sec. II-B. All cases in which the JA filter is used, make use of a constant bound $b(\underline{q}_k) = c_b$, and a certain prediction function $p(\underline{q}_k)$.

B.1.1. Jump-Aware Filter with Position Data

Since impacts result in velocity jumps, impacts can be detected by identifying jumps in the velocity. In [7] this is done based on position encoder data, whereas in our method, we use the data in the robot's operational space. A prediction for the current timestep t_k can be made by fitting a second order polynomial to the m_k previous measurements and extrapolate at $t = t_k$. In the second order polynomial function $y(s) = \xi_0 + \xi_1 s + \xi_2 s^2$, with $s = t - t_k$, $\boldsymbol{\xi} = [\xi_0, \xi_1, \xi_2]^\top$ are the fitting parameters. The parameters can be retrieved via linear least squares fitting methods. With this function fitted on previous measurements, the prediction for the next timestep t_k is $y(t_k - t_k) = y(0) = \xi_0$. The prediction for the current position $\boldsymbol{x}_k$ can be made by predicting each of the individual components of $\boldsymbol{x}_k$ using a polynomial function.

Applying a JA filter to the position data, with a second order polynomial as prediction function, and a maximum window length of $M = 10$, results in the difference between measurements and predictions that is shown in Fig. B.2. It can be seen that the peaks at the jumping times in the velocity ($t \approx 4.0$s and $t \approx 4.5$s) are present, although there are several thin larger peaks. Zooming in on one of those small peaks in Fig. B.3 shows that these peaks exist due to missing datapoints. A datapoint at $t \approx 3.366$s is missing, and its value is shifted to the datapoint at $t \approx 3.368$s, causing an inaccurate prediction.

The redundancy of the JA filter can be improved by using a larger window length. The resulting plot of the difference between measurements and predictions where a maximum window length of $M = 35$ is used, is shown in Fig. B.4. Although the peaks due to missing datapoints still exist, the peaks corresponding to the impacts are larger now, meaning that with the right choice for a bounding value, the impacts will be detected. Tuning can still be problematic, as a bounding value that is slightly too low can result in false positive detections due to those peaks of missing datapoints, and a value that is

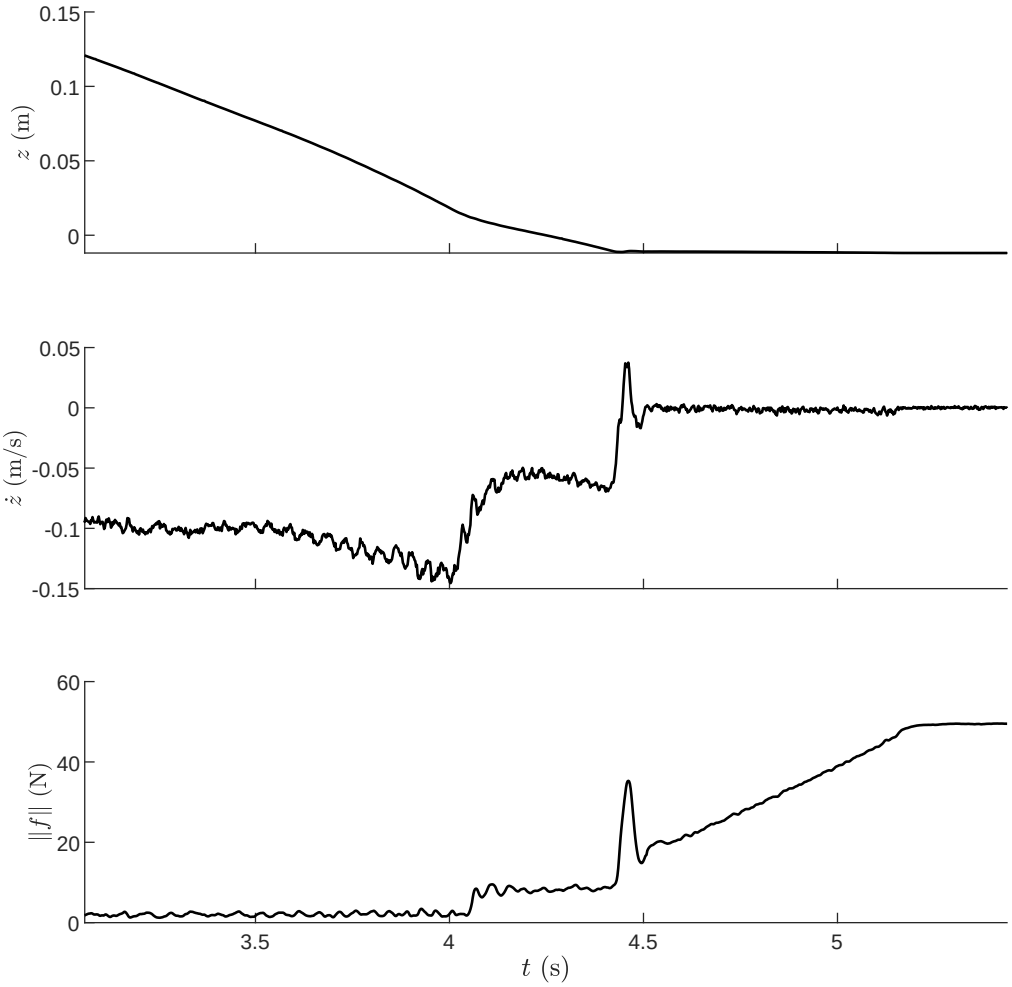


Figure B.1: Position, velocity and estimated external force in the Z direction.

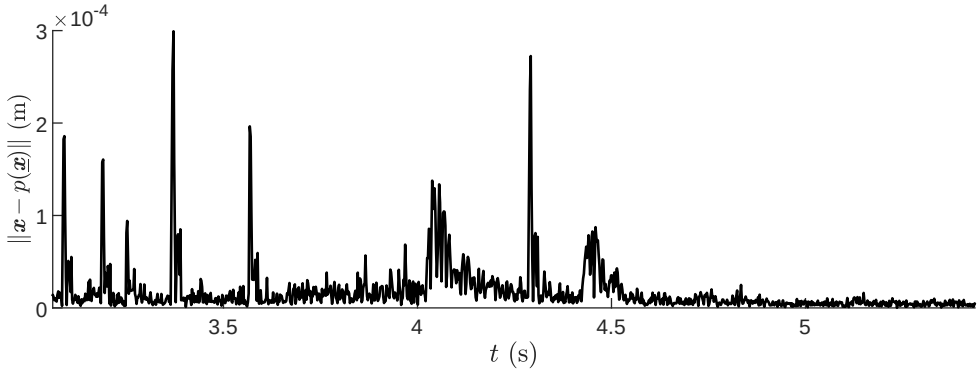


Figure B.2: Difference between position measurements and predictions. Predictions are made by fitting a second order polynomial. The JA filter uses a maximum window length of $M = 10$.

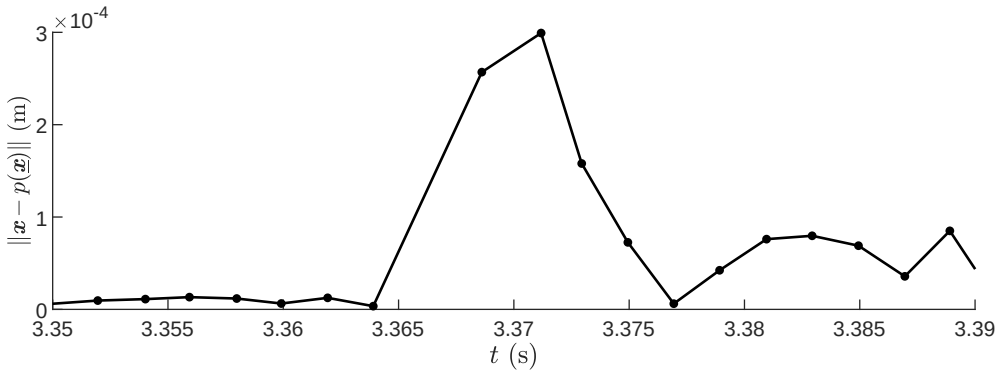


Figure B.3: The plot of Fig. B.2 zoomed in around the peak at $t \approx 3.372$. This peak is caused by a missing datapoint at $t \approx 3.366$.

too large results in missed detections. In addition, the use of a larger window length results in a larger delay between the actual impact and the detection.

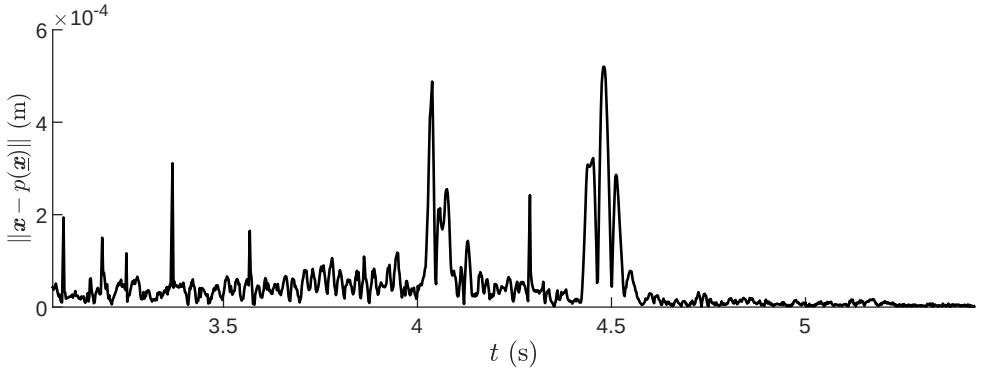


Figure B.4: Difference between position measurements and predictions. Predictions are made by fitting a second order polynomial. The JA filter uses a maximum window length of $M = 35$.

B.1.2. Detecting Impacts with Force Rate

As mentioned in Sec. I, impacts take place when the robot collides with the environment, resulting in large external forces that suddenly act on the robot. As explained in Sec. II-C, a momentum observer that gives a first order filtered estimation of the external forces, can be used to identify peaks in the external forces. The author of [6] uses this to detect impacts based on the force rate. According to this method, an impact is detected when $\|\boldsymbol{f}_k - \boldsymbol{f}_{k-1}\|/(t_k - t_{k-1}) > \epsilon$, with $\boldsymbol{f}$ the estimated external force, and ϵ a constant. The magnitude of the Euler backward derivative of the external force estimation is shown in Fig. B.5. Two regions with larger values can be seen, corresponding to the times of impact. Because this method detects impacts based on only 2 datapoints, it is not robust to missing datapoints and noise.

B.1.3. Jump-Aware Filter with Force Rate

As explained in Sec. III-C, the method of Sec. B.1.2 can be rewritten as a JA filter. This gives the advantage that an impact detection is based on more than two datapoints, making it more robust to noise and missing datapoints.

A more detailed version of Fig. 6 can be seen in Fig. B.6.

Based on Fig. B.6, the bounding constant in (14) can be set to $\epsilon = 4.0/0.002\mathrm{N\,s^{-1}}$, with average sampling period $T_s = 0.002$s, so that both the impacts are detected. The detected impacts in the position, velocity and external force plot can be seen in Fig. B.7, which is a more detailed version of Fig. 5.

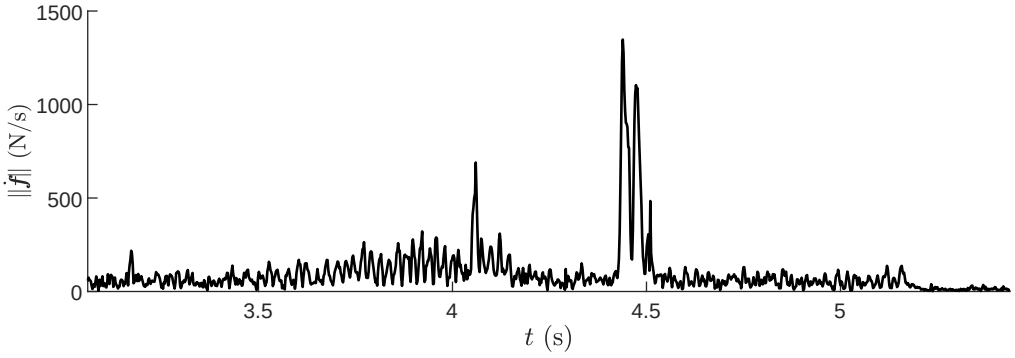


Figure B.5: Magnitude of the Euler backward derivative of the external force.

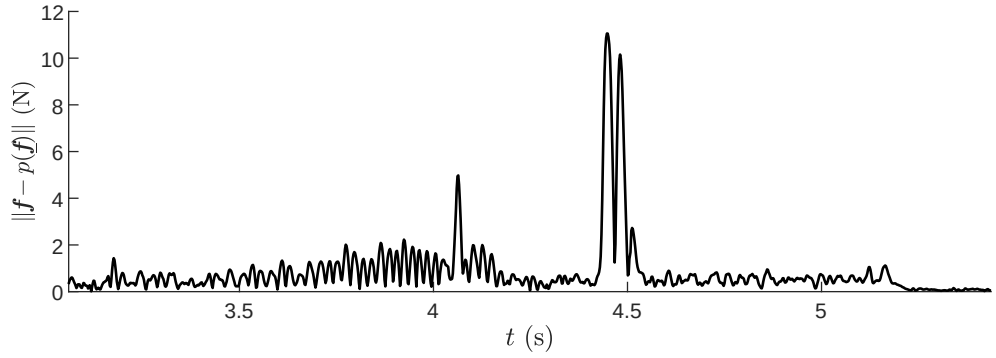


Figure B.6: Difference between external force measurements and predictions. Predictions are made by fitting a zero order polynomial. The JA filter uses a maximum window length of $M = 10$.

B.2. Detection Delay

A problem of this method is that a delay between the actual impact and the moment of detection is inevitable. There are not only delays in data acquisition, momentum observers, and algorithm calculations, but there is also the threshold of the bounding function $b(\underline{\boldsymbol{f}}_k)$ (11) that needs to be exceeded before an impact is detected. When this is finally the case, contact between the robot and the environment has already been established. Even though all these delays added up can be in the order of a few milliseconds, this still has significance, because an impact also takes just a couple of milliseconds. For the online impact detection this delay is not problematic, because it will result in a mismatch of only a few milliseconds between impact and reaction time. It is however problematic for the offline learning process. The author of [1] points out that a delay of only 5 milliseconds can already be enough for unreliable fitting of the post-impact velocity of [2], which is described in Sec. A.2.4.

An attempt to determine the detection delay and estimate the real time of impact has been made under the assumption that without external disturbances the prediction function $p(\underline{\boldsymbol{f}}_k)$ should provide an accurate prediction for the next datapoint $\boldsymbol{f}_k$, or $p(\underline{\boldsymbol{f}}_k) \approx \boldsymbol{f}_k$. When this is the case, then the difference between measurement and prediction $\|\Delta_k\| = \|\boldsymbol{f}_k - p(\underline{\boldsymbol{f}}_k)\|$ should be very small. An estimation of the actual time of detection can now be done by finding the first local minimum of $\|\Delta_k\|$, with $k \in \{i, i-1, ..., i-M_i\}$, where i stands for the timestep the impact is detected at, and M_i stands for the maximum window length to find this local minimum.

This analysis is performed on the results of Fig. B.7 to find the detection delays of the impacts. The result can be seen in Fig. B.8. It can be seen that when taking the datapoint with a local minimum value in the difference between measurements and predictions, the impact is detected at the start of the jump in force and velocity, instead of halfway. Although this noncausal method cannot be used online, it can be used for better segmentation in the offline algorithm.

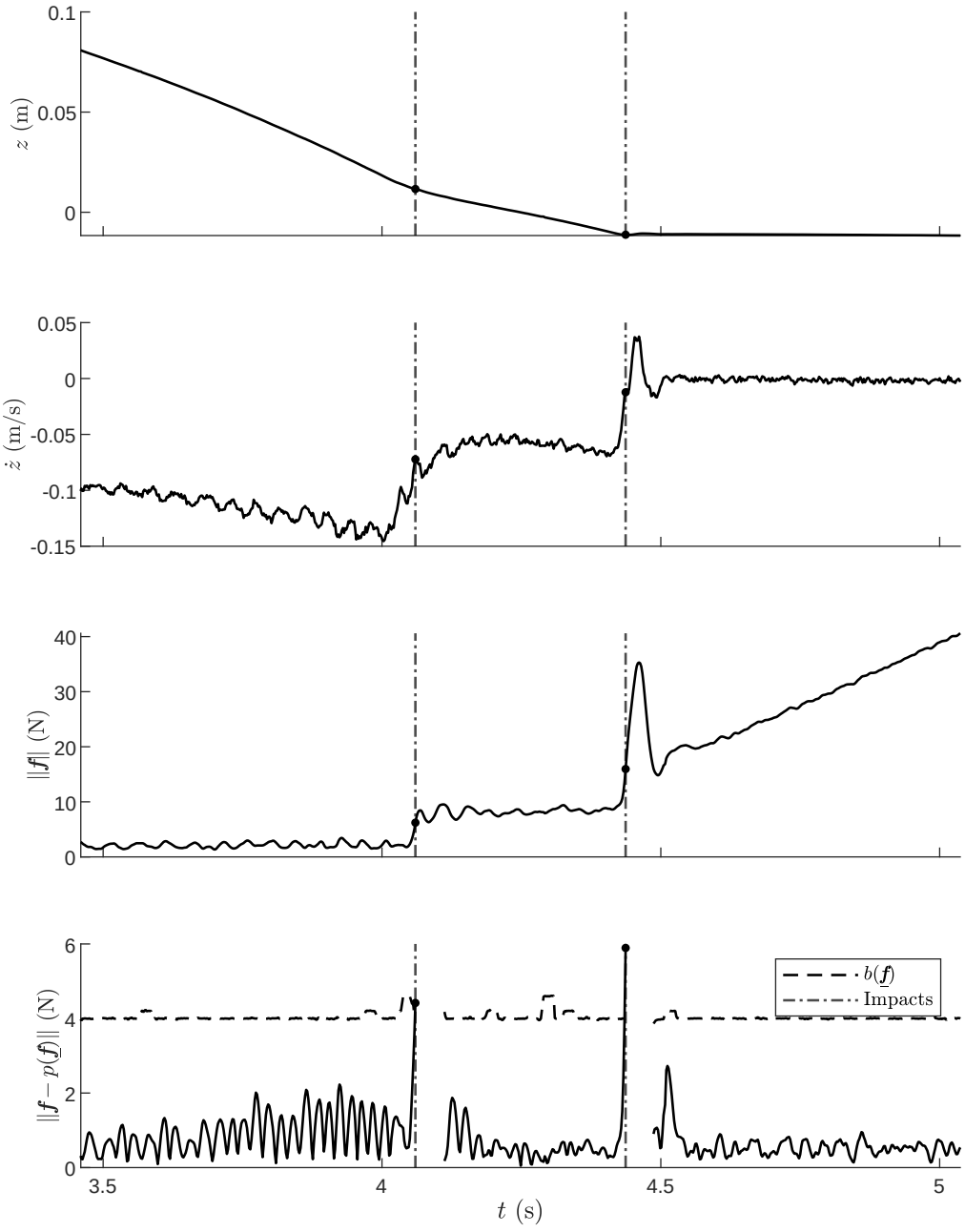


Figure B.7: Detected impacts shown in the position, velocity, and force plot, and the plot that shows the difference between measurements and predictions.

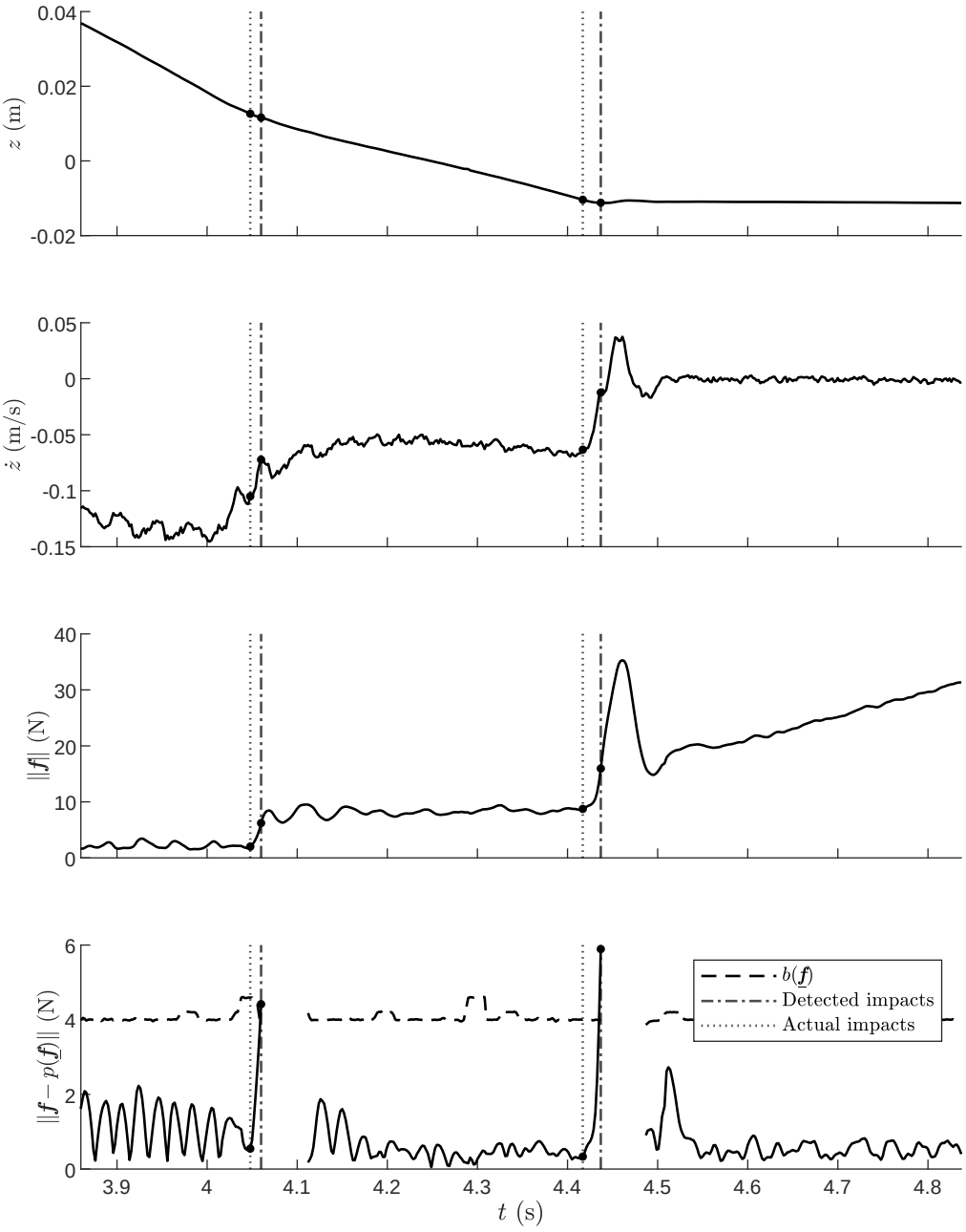


Figure B.8: Analyzing the detection delay results in timestamps that are more likely to be the start of the impact.

C

Experimental Results

In addition to the plots in Sec. IV-C, this appendix contains plots of the position, position error, velocity, contact force, and commanded force for both the stamping task and the sliding task.

C.1. Stamping task

For the stamping task, the plots of the Z direction are shown in Fig. C.1. These are the same plots that are already shown in Sec. IV-C. The corresponding plots for the X and Y direction have values that stay approximately constant during the entire trajectory.

Figure C.2 shows the corresponding plots for the rotation around the X axis. The rotational position reference remains almost constant through the entire trajectory, and there is no torque reference. Deviations in the rotational velocity and the torque are mainly caused by a slightly tilted configuration when the robot establishing contact. The baseline controller has the largest overshoots in rotational velocity and torque, corresponding to the large impact forces at $t \approx 3.9$s and $t \approx 4.0$s. The controller that uses damping in the interim phase has the smoothest transition from ante-impact to post-impact phase, since the impact forces are the lowest when using this controller.

C.2. Sliding task

Figures C.4 and C.3 show the plots in the Z and X directions, which are also partly represented in Sec. IV-C. For the Z direction plots of the controller that use an interim phase strategy, besides the similar velocity and force trajectories, the position, position error, and commanded force plots are also very identical.

Figures C.6 and C.5 show the plots of the rotation around the Z and X axis. Deviations in the rotations around the X axis are mainly caused by the trajectory in the Z direction. It can be seen in C.5 that the trajectories for both the controllers with an interim strategy are similar, although the controller that does not use active damping in the interim phase has larger overshoots in the contact torque and the rotational velocity at the moment of impact at $t \approx 2.45$s.

The rotations around the Z axis are mainly caused by the frictional forces in the X direction. Similar to the translational Z direction, the lack of damping in the controller that uses no damping in the interim phase, causes large rotational velocities and large deviations of the rotation. When this is undesired, it can be considered to only turn off the damping in the translational Z and the rotational X directions, since the impact is only acting in those directions.

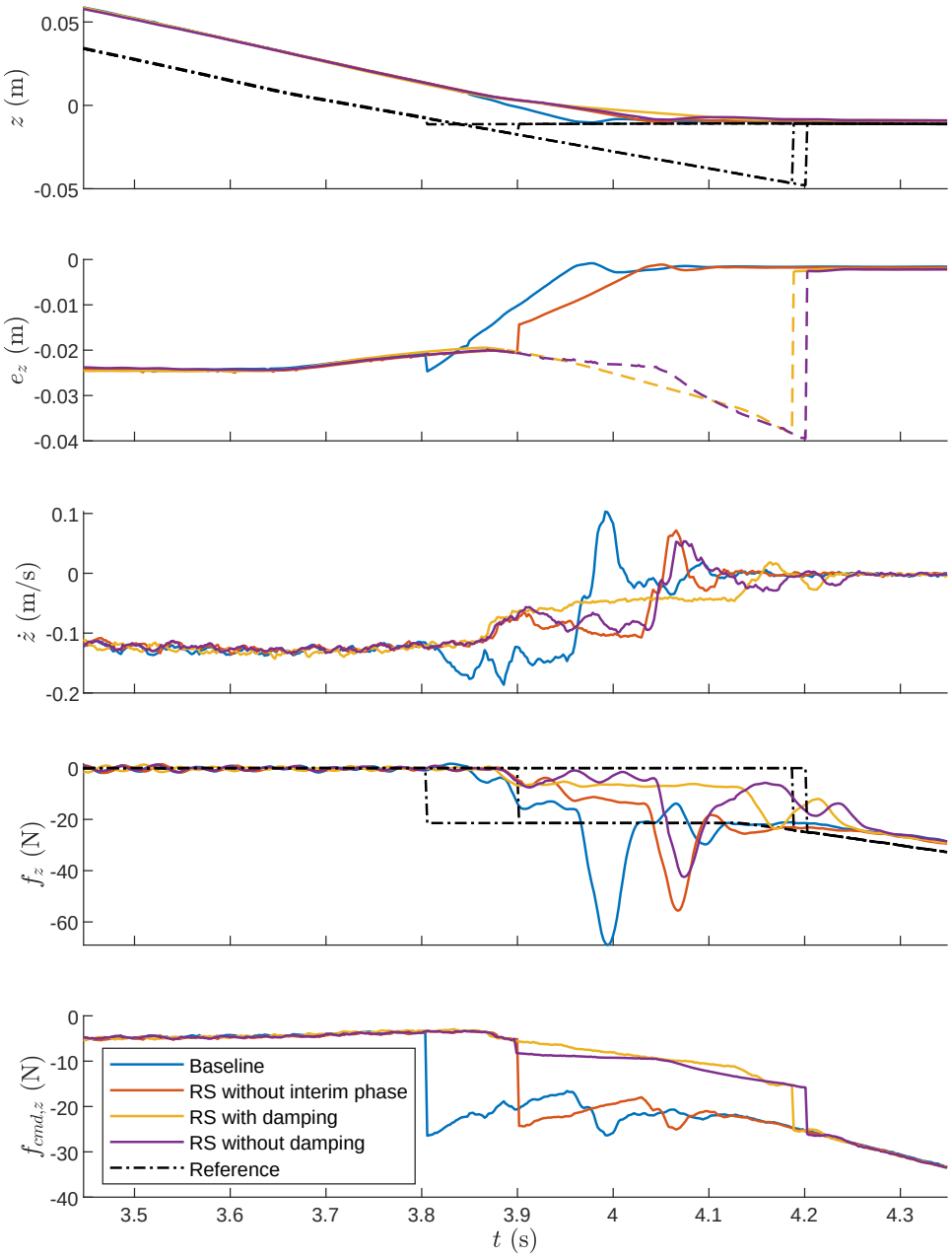


Figure C.1: Position, position error, velocity, contact force, and commanded force in the Z direction of the stamping task. During the interim phase, the error is shown as a dashed line.

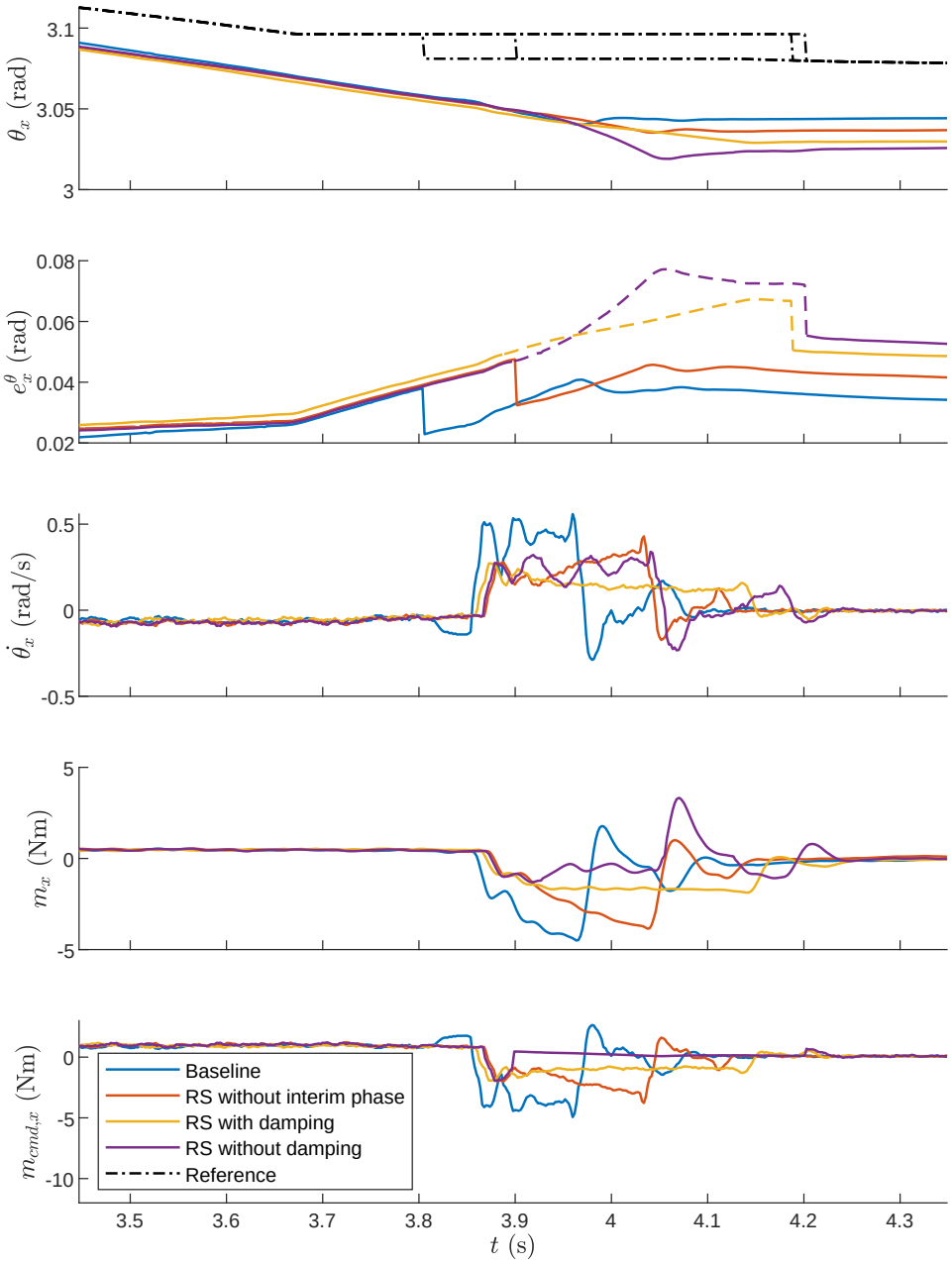


Figure C.2: Rotation, rotation error, rotational velocity, contact torque, and commanded torque around the X axis in the stamping task. During the interim phase, the error is shown as a dashed line.

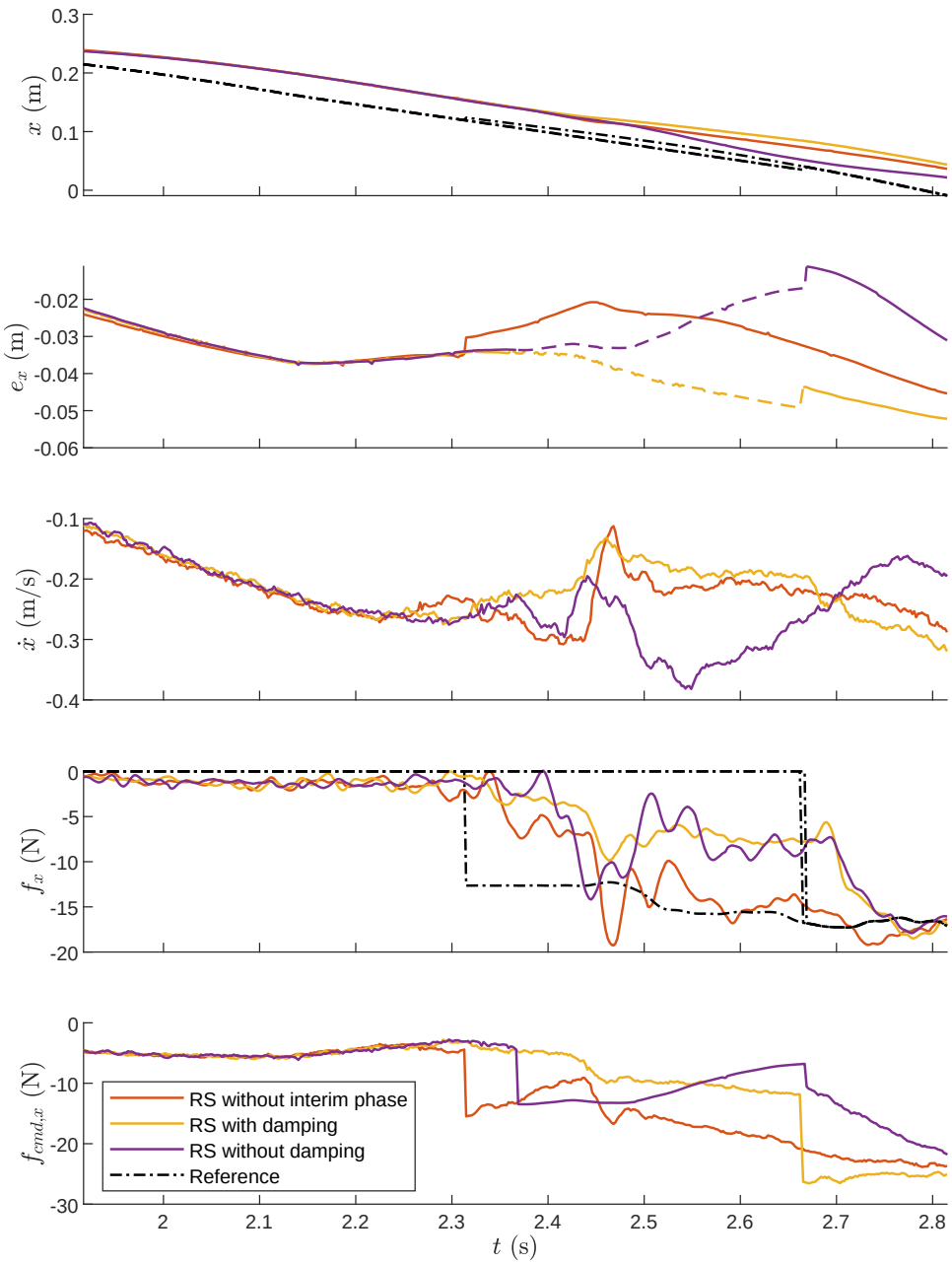


Figure C.3: Position, position error, velocity, contact force, and commanded force in the X direction of the sliding task. During the interim phase, the error is shown as a dashed line.

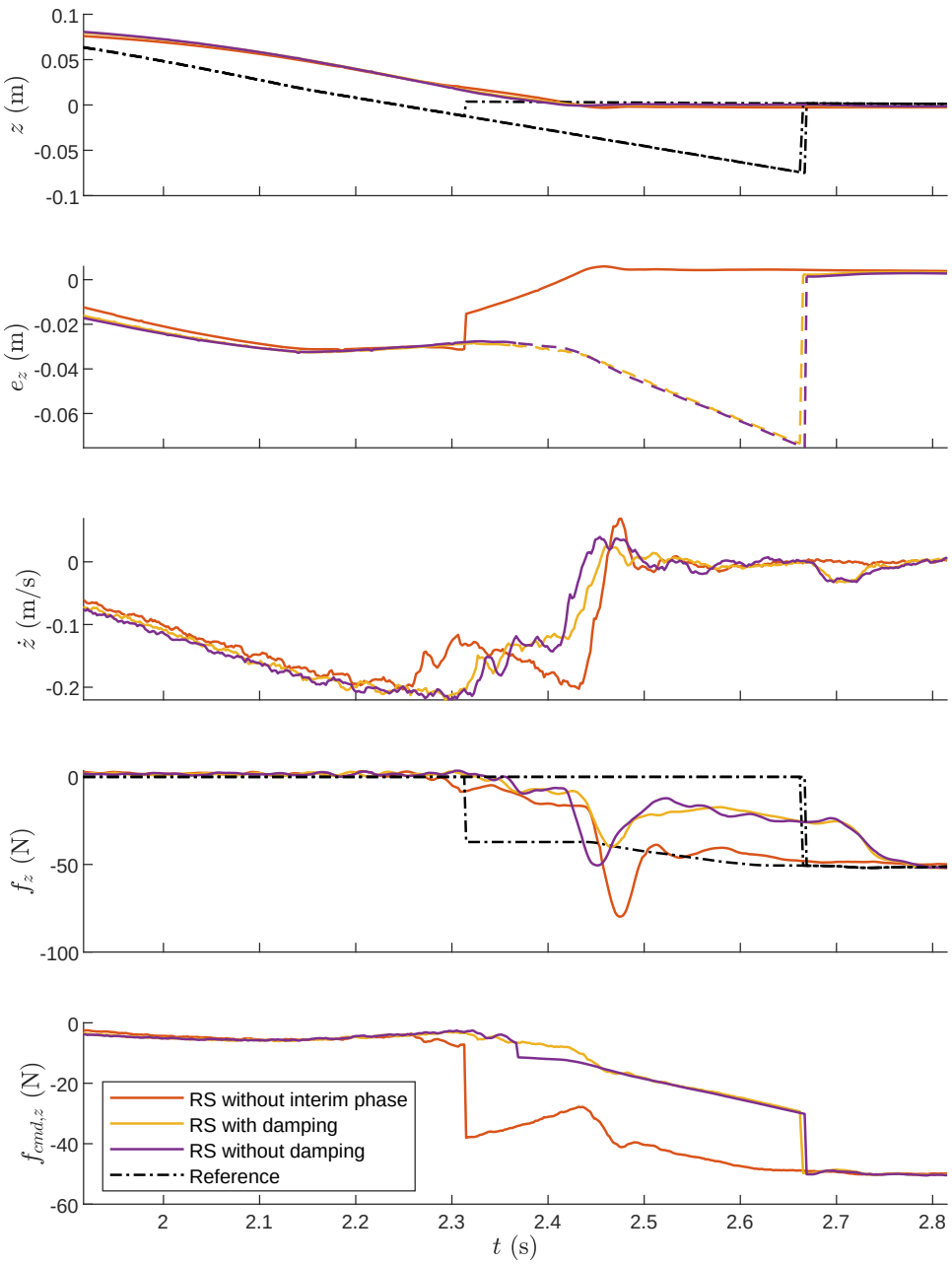


Figure C.4: Position, position error, velocity, contact force, and commanded force in the Z direction of the sliding task. During the interim phase, the error is shown as a dashed line.

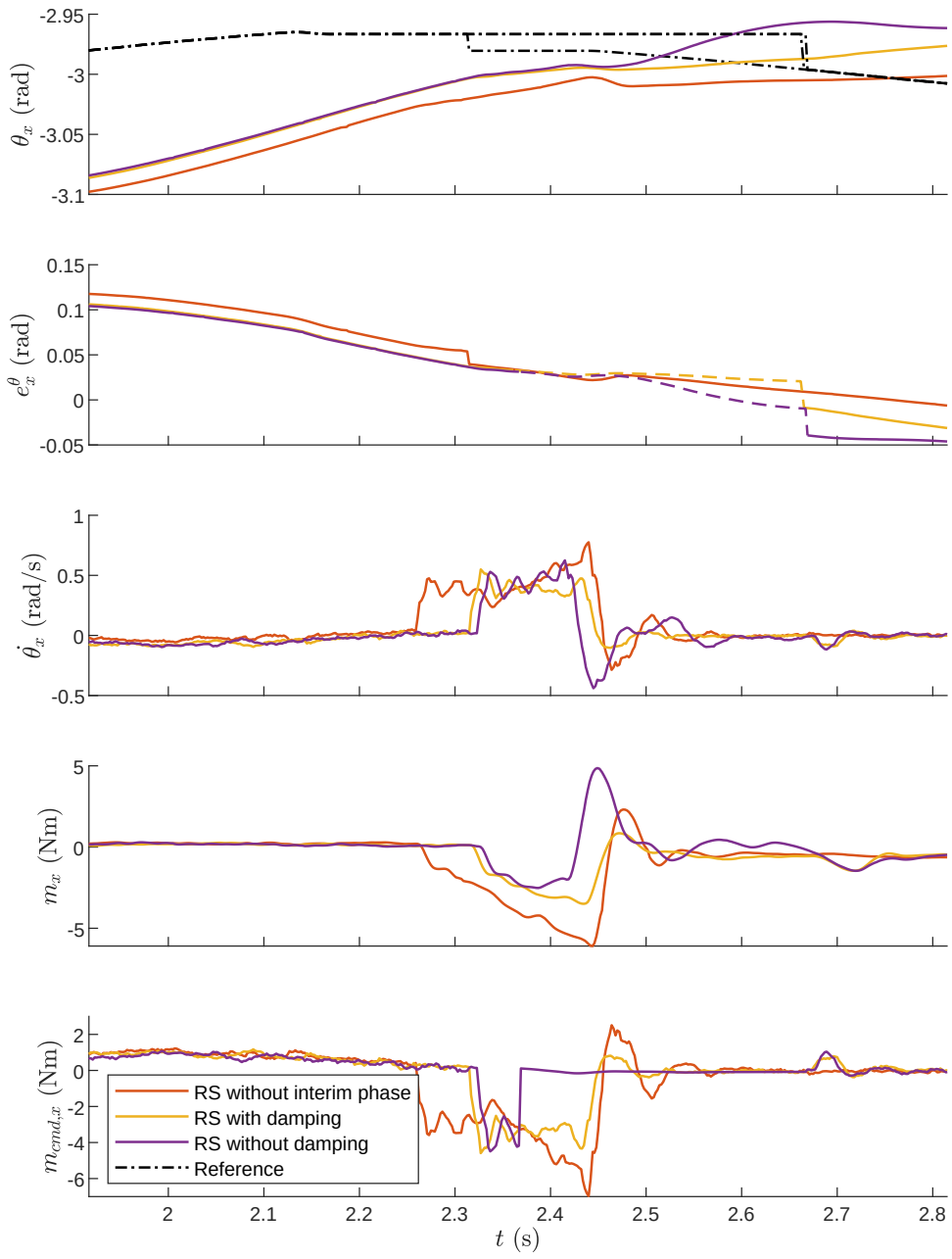


Figure C.5: Rotation, rotation error, rotational velocity, contact torque, and commanded torque around the X axis in the sliding task. During the interim phase, the error is shown as a dashed line.

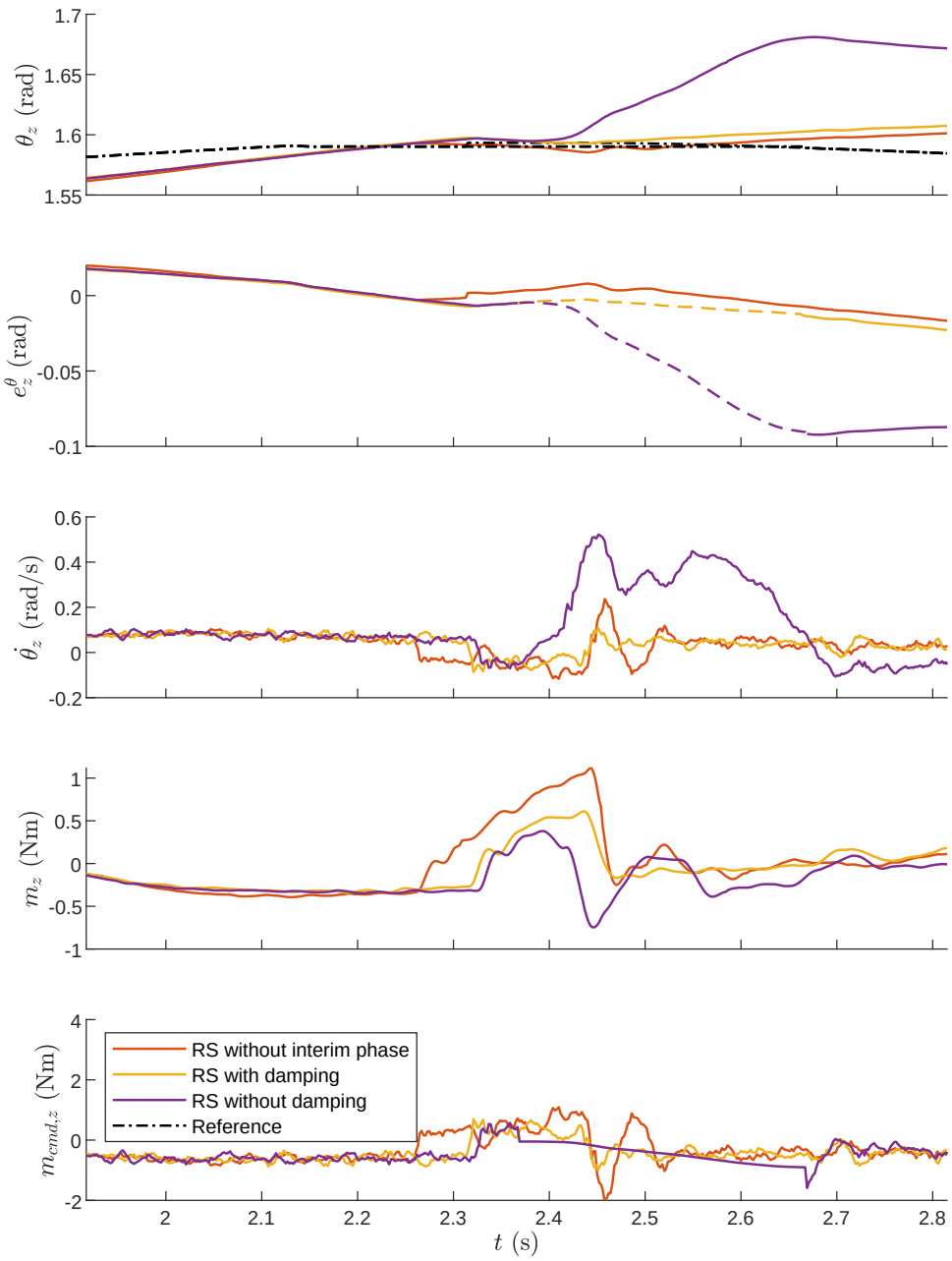


Figure C.6: Rotation, rotation error, rotational velocity, contact torque, and commanded torque around the Z axis in the sliding task. During the interim phase, the error is shown as a dashed line.

D

Teleoperation Device

An HTC VIVE Pro Controller 2.0, depicted in Fig. D.1, is used as device to demonstrate tasks to the robot. The whole teleoperation setup consists of the controller and two base stations. The device has several tracking points ((6)) that the base stations use to determine its location and orientation, in its own Cartesian reference frame. The pose of the device is used to create reference poses for the robot's end-effector. To read out the pose of the controller, code from [8] is used. In between the process of reading the Cartesian pose of the device and the reference pose that was sent to the robot's controller, a converter is implemented. This converter takes care of converting the reference frame of the device to the reference frame of the robot, and has some safety measures implemented. The workflow can be seen in Fig. D.2.

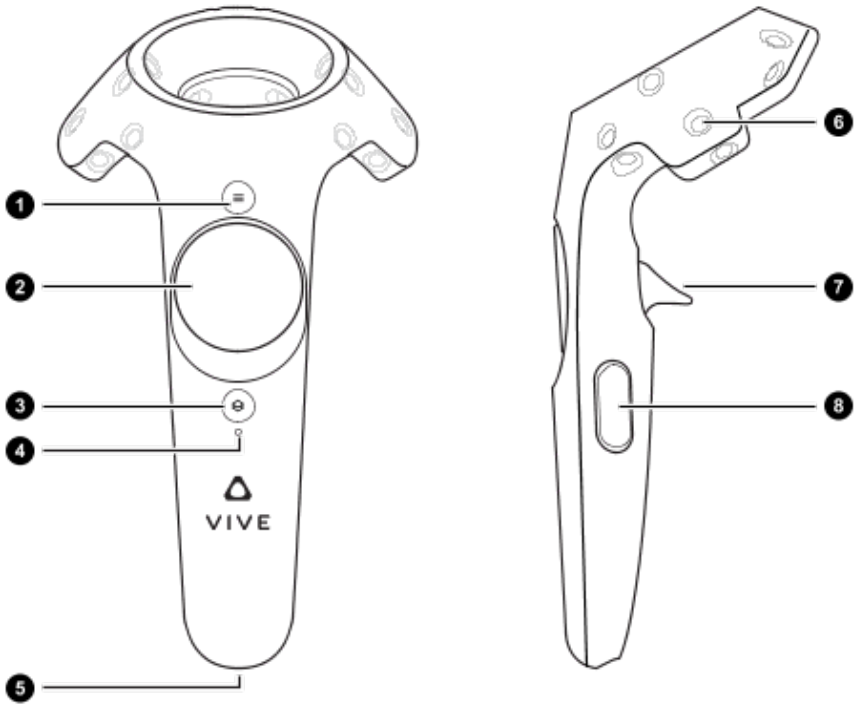


Figure D.1: Schematic visualisation of the HTC VIVE controller. Image retrieved from [3].

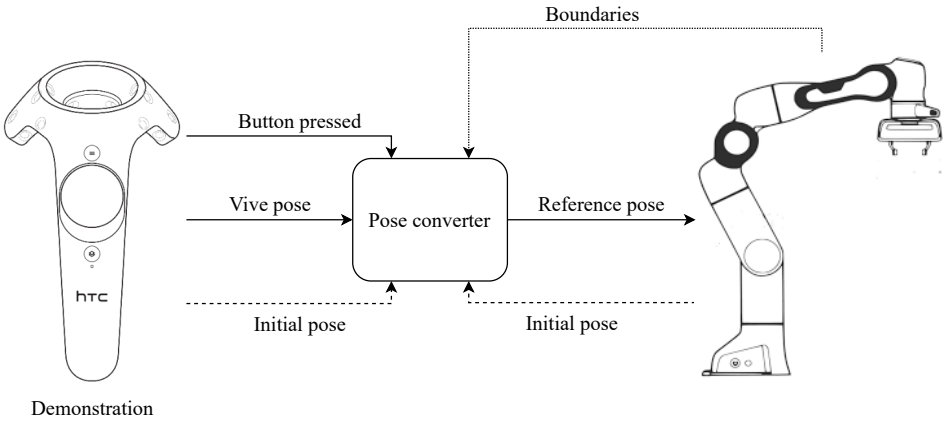


Figure D.2: Teleoperation demonstration workflow. After the safety button is released, the initial poses of the VIVE controller and the robot are stored. New poses of the VIVE controller are used to calculate reference points for the robot, while keeping the robot's boundaries into account.

D.1. Giving Commands

Before starting the VIVE controller, the reference frames of the robot and the device are aligned to be parallel. This is achieved by finding the yaw angle γ by which the device's reference frame is rotated around the vertical Z axis, with respect to the robot's reference frame. At the start of the execution, the initial position $\boldsymbol{x}_0$ and orientation $\boldsymbol{\theta}_0$ of the robot, and the initial position of the device $\boldsymbol{x}_0^{vive}$ and orientation $\boldsymbol{\theta}_0^{vive}$ are stored. Using the current position of the VIVE controller $\boldsymbol{x}^{vive}$, the robot's reference position is calculated using the distance the device has traveled, according to

$$\boldsymbol{x}_d = \boldsymbol{x}_0 + \boldsymbol{x}^{vive} - \boldsymbol{x}_0^{vive}. \tag{D.1}$$

The reference orientation is calculated using the rotation matrices of the stored orientations and the current orientation of the VIVE controller.

$$\boldsymbol{R}_d = (\boldsymbol{R}_0^{vive^{-1}} \boldsymbol{R}^{vive}) \boldsymbol{R}_0, \tag{D.2}$$

where rotation matrix $\boldsymbol{R}$ can be constructed using the Euler angles of orientation $\boldsymbol{\theta}$ and vice versa.

D.2. Safety Measures

The VIVE controller has several buttons, of which the one in the back ((7)) is used to stop the device. With this button pressed, the converter does not send new reference poses to the robot. The initial position $\boldsymbol{x}_0^{vive}$ and orientation $\boldsymbol{\theta}_0^{vive}$ are reset when the button is released.

Furthermore there is a saturation on the reference position. The user can set limitations for $\boldsymbol{x}_{min} <= \boldsymbol{x}_d <= \boldsymbol{x}_{max}$, resulting in the converter to change the desired position to the closest value within range if the calculated value falls outside the boundaries. The saturation is used for safety, because it can prevent the robot to collide with the environment in an undesired way, as well as for decreasing the difficulty of the demonstrations. It is easier for the human demonstrator to focus on one or two translational directions than on three.

References

[1] Ilias Aouaj. "Prediction accuracy of nonsmooth impact laws in robot-environment collision experiments". Report number: DC 2019.070. MSc thesis. Eindhoven University of Technology, Department of Mechanical Engineering, Dynamics and Control section, 2019. URL: https://research.tue.nl/en/studentTheses/26cbf58d-905b-499d-9651-4f45b1350fc3.

[2] Ilias Aouaj, Vincent Padois, and Alessandro Saccon. "Predicting the Post-Impact Velocity of a Robotic Arm via Rigid Multibody Models: an Experimental Study". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2264–2271. DOI: 10.1109/ICRA48506.2021.9561768.

[3] *HTC Vive Controller*. accessed Februari 1, 2022. URL: https://business.vive.com/us/support/vive-pro-secure/category_howto/about-the-controllers---2018.html.

[4] Nam Mai-Duy and Thanh Tran-Cong. "Approximation of function and its derivatives using radial basis function networks". In: *Applied Mathematical Modelling* 27.3 (2003), pp. 197–220. ISSN: 0307-904X. DOI: https://doi.org/10.1016/S0307-904X(02)00101-4.

[5] Alexandros Paraschos et al. "Probabilistic Movement Primitives". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

[6] B.W.B. Proper. "Aim-Aware Collision Monitoring for Robot-Environment Edge Impacts". Report number: DC 2021.081. MSc thesis. Eindhoven University of Technology, Department of Mechanical Engineering, Dynamics and Control section, 2021. URL: https://research.tue.nl/en/studentTheses/aim-aware-collision-monitoring-for-robot-environment-edge-impacts.

[7] M. Rijnen, A. Saccon, and H. Nijmeijer. "Motion Signals With Velocity Jumps: Velocity Estimation Employing Only Quantized Position Data". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1498–1505. DOI: 10.1109/LRA.2018.2800097.

[8] Robosavvy. *vive_ros*. https://github.com/robosavvy/vive_ros. 2019.

[9] SciPy. *optimization.curve_fit*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html. accessed Januari 26, 2022.

[10] Sjouke de Zwart. "Impact-Aware Learning from Demonstration". MSc thesis. Delft University of Technology, Faculty of Mechanical, Maritime, Materials Engineering (3mE), Delft Center for Systems, and Control (DCSC), 2019. URL: https://repository.tudelft.nl/islandora/object/uuid%3Ac6f91fb2-2544-4802-bcda-4ee70ab0e2be.