Impact-Aware Manipulation by Dexterous Robot Control and Learning in Dynamic Semi-Structured Logistic Environments

I.Control report

Dissemination level	Public (PU)
Work package	WP4: I.Control
Periodic report	D4.1
Version	F - 1.0
Submission date	06/04/2024
Due date	30/09/2023

www.i-am-project.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871899

Authors

Authors in alphabetical order		
Name	**Organisation**	**Email**
Pierre GERGONDET	CNRS	pierre.gergondet@gmail.com
Abderrahmane KHEDDAR	CNRS	kheddar@lirmm.fr
Harshit KHURANA	EPFL	harshit.khurana@epfl.ch
Alessandro SACCON	TUE	a.saccon@tue.nl
Jari J. van STEEN	TUE	j.j.v.steen@tue.nl
Ahmed ZERMANE	CNRS	ahmed.zermane@lirmm.fr

Control sheet

Version history			
Version	**Date**	**Modified by**	**Summary of changes**
0.1	23/12/2022	Abderrahmane KHEDDAR	TOC & first contents
0.11	21/09/2023	Jari VAN STEEN	Addition TU/e results on T4.3
0.12	02/10/2023	Jari VAN STEEN	Update T4.3 results based on peer-review comments
0.2	05/10/2023	Pierre GERGONDET	Update Benchmarking
0.21	17/10/2023	Harshit KHURANA	Update EPFL results on Control for Hitting
0.22	20/10/2023	Michael BOMBILE	Update EPFL results on Self-Correcting QP
0.5	20/12/2023	Ahmed ZERMANE	Pre-final version ready for peer-review
0.9	27/03/2024	Ahmed ZERMANE	Peer-review comments addressed
0.91	04/04/2024	Harshit KHURANA, Ahmed ZERMANE	Additional comments addressed
1.0	06/04/2024	Jos DEN OUDEN	Revised version ready for submission, quality check

Peer reviewers		
	Reviewer name	**Date**
Reviewer 1	Alexander OLIVA	14/03/2024
Reviewer 2	Jos DEN OUDEN	29/03/2024

Legal disclaimer

TABLE OF CONTENTS

ABBREVIATIONS

Abbreviation	Definition
EC	European Commission
IDs	Inverse Dynamics
MPC	Model Predictive Control
PU	Public
QP	Quadratic Programming
WBQP	Whole body task and joint space QP with constraints

1 Introduction

1.1 I.AM. project background

Europe is leading the market of torque-controlled robots. These robots can withstand physical interaction with the environment, including impacts, while providing accurate sensing and actuation capabilities. I.AM leverages this technology and strengthens European leadership by endowing robots to exploit intentional impacts for manipulation. I.AM focuses on impact aware manipulation in logistics, a new area of application for robotics which will grow exponentially in the coming years, due to socio-economical drivers such as booming of e-commerce and scarcity of labour. I.AM relies on four scientific and technological research lines that will lead to breakthroughs in modeling, sensing, learning and control of fast impacts:

1. I.Model offers experimentally validated accurate impact models, embedded in a highly realistic simulator to predict post-impact robot states based on pre-impact conditions;
2. I.Learn provides advances in planning and learning for generating desired control parameters based on models of uncertainties inherent to impacts;
3. I.Sense develops an impact-aware sensing technology to robustly assess velocity, force, and robot contact state in close proximity of impact times, allowing to distinguish between expected and unexpected events;
4. I.Control generates a framework that, in conjunction with the realistic models, advanced planning, and sensing components, allows for robust execution of dynamic manipulation tasks.

This integrated paradigm, I.AM, brings robots to an unprecedented level of manipulation abilities. By incorporating this new technology in existing robots, I.AM enables shorter cycle time (10%) for applications requiring dynamic manipulation in logistics. I.AM will speed up the take-up and deployment in this domain by validating its progress in three realistic scenarios: a bin-to-belt application demonstrating object tossing, a bin-to-bin application object fast boxing, and a case depalletizing scenario demonstrating object grabbing.

1.2 I.Control background

This deliverable sums up the work that have been conducted all along the project in WP4: I.AM. I-Control. We recall the WP4 I-Control objectives are:

- *Extending existing whole-body, constrained, optimization-based, QP robot control architecture to explicitly account for the execution of intentional impacts tasks*: the main ideas we came with in I.AM., is (i) to integrate contact awareness in the constraints of the QP to account for state jumps inherent to impact in robot state variables that appear in the QP tasks and constraints, (ii) to enhance the constraints with impact limitations, and (iii) being robust to impact location and some modeling parameters.
- *Enhancing the QP control with a short-time horizon model preview control to handle the fast dynamics in meeting impacts and releasing abrupt contacts*: this objective is still on-going for the time being. As we only consider the QP controller with a one-step ahead view of the impact,

hence assuming the impact will occur in the next iteration at each cycle (which is indeed very conservative), it is important to see how far one can achieve in a full MPC implementation of the QP control.

- *Use of Control theory to assess the stability and robustness of QP based controller with adaptive stiffness, also as coupled with dynamical systems developed in I-Learn*: novel ideas and results that have been achieved are reported.

1.3 Purpose of the deliverable

This deliverable D4.1 aims at providing a synthesis of the main achievements made in the control of the robots and the demonstrators of I.AM.

1.4 Intended audience

The dissemination level of D4.1 is 'public' (PU) – meant for members of the Consortium (including Commission Services) and the general public. This document is intended to serve as an internal guideline for the entire I.AM. Consortium and provide the consortium's implementation plans regarding data management.

2 Impact-Aware QP-Control

When a robot collides – intentionally or inadvertently – with a rigid surface with a fairly high relative speed, the induced forces are impulsive and the contact state is uncertain. The shock propagates through the robot linkage into the joints and can severely damage some parts of the hardware, e.g., the harmonic gears and torque sensors (if any).
A common remedy is to carefully plan contact transitions with near-to-zero relative speed. However, this strategy can not achieve specific tasks that require relatively high impact such as swift grabbing. For such tasks, the following robotic issues shall be improved concurrently: (i) design impact-resilient hardware (this is not tackled in I.AM.); and (ii) devise robust control strategies that switch the robot equations of motions and subsequent controllers following a transition policy called reset maps. The switching often requires a precise impact model and knowledge of additional parameters that depend on the environment and the robot, e.g., the impact localization on the robot (and on the environment surface), the contact normal, and the exact impact time. Acquiring these parameters *in-situ*, instantaneously, and reliably is not always possible in practice.

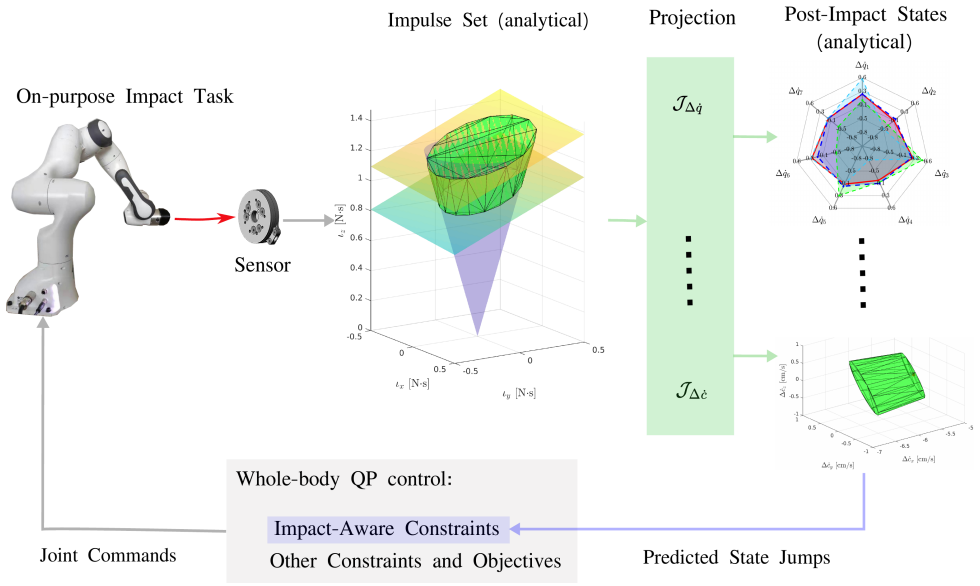


Figure 1: The impact-aware QP regulates the contact velocity in a modified search space to ensure that the post-impact state jumps are hardware-affordable.

An impact event is instantaneous and too short for a robot to react effectively. The impact duration (i.e., time interval) depends on the particular contact properties and the robot controller. In our recent studies [1], an impact lasted about 20 msec. This is the reason why, any impact-friendly control strategy shall act *a priori* and *a posteriori*.
To tackle on-purpose impact tasks safely, CNRS challenged the possibility of building impact-awareness on the existing task-space control framework[1] instead of devising an entirely new control scheme. There are many reasons for this choice:

- the framework has been proved to be efficient in handling complex industrial scenarios and multi-robot control [2, 3];

[1] mc_rtc `https://github.com/jrl-umi3218/mc_rtc`

- to our best knowledge, on-purpose impact objectives using the task-space quadratic programming (QP) formalism was not explicitly addressed;
- there is a relatively large community using it beyond the CNRS' circle. Our guiding quest is to try as much as possible to enhance task-space QP control to deal with impact without considerably changing its structure. In other words: can we envision handling impact tasks simply by adding or reformulating task objectives and constraints without introducing new decision variables? Our on-going research in I.AM. suggests that the answer to this latter question is *yes, to some extent*.

Formulating hardware impact limits[2] as additional QP constraints, is straightforward but insufficient. The main problem is dealing with unexpected state jumps that may damage the robot. Unexpected because, in practice, even if impact is planned to occur, there will be uncertainties on both contact time and location. The feedback velocity and force jump compromise the constraints' feasibility. Consequently, we reformulate the usual constraints to account for such discontinuities. Then the QP is impact-aware and robust to state jumps by modifying the search space according to a one-step-ahead prediction of nearby intended (i.e., expected) impacts. As a result, the controller updates the optimized – and hence feasible – impact velocity reference in every control cycle. If the impact happened, the robot would fulfill both the hardware resilience and task-dictated constraints bounds.
We summarize our main adding as follows:

- bound the *post-impact states* with analytically-constructed convex sets (half-space represented polyhedra) assuming the impact is frictional in three dimensions;
- bound a generic post-impact robot state with a closed-form *impact-aware template constraint*;
- assess our *impact-aware control design* through experiments on the Franka Emika robot manipulator but also with a humanoid robot.

We build our QP improvement, see Figure 1, on top of our initial concept proposed in simulation for fixed-base robots in [4] and on our recent experimental modeling work done in I.AM. and published in [5, 1]. The main results in I.AM. for this task are gathered in a journal paper (IJRR) [6].
Currently, the QP control framework is made available (open source and open access) and used by all the I.AM. partners that required it. See also WP5 for its integration with AGX impact simulation and the I.AM. scenarios. It is intensively used by TU/e to integrate all the demonstrators and that contributed in many fruitful discussions. It has also been installed at TUM and EPFL.

[2] Very few robot providers disclose the hardware's impact-resilience bounds.

3 Model Preview and Adaptive Control

Prioritized task-space multi-objective QP approaches enable online tasks insertion, removal or priority swapping to lower subsequent discontinuity that may occur at the joint-acceleration and torque. On the other hand, the constraints have a higher priority over the tasks. The latter are generally achieved 'at best' while fulfilling all the constraints. Unfortunately, introducing constraints online, namely those for impact-awareness, could make the QP infeasible. In fact, all the constraints have the same level of priority. Consequently, if at least two constraints conflict, the constraint-set becomes empty. We say then that these constraints are incompatible.
The main reason for this infeasibility is the QP controller being purely reactive, it solves the optimization problem based on the current robot state. However, dealing with potential incompatible constraints requires a prediction of the future robot states to know what are the current actions to do to ensure the viability of the control problem in the next iterations.

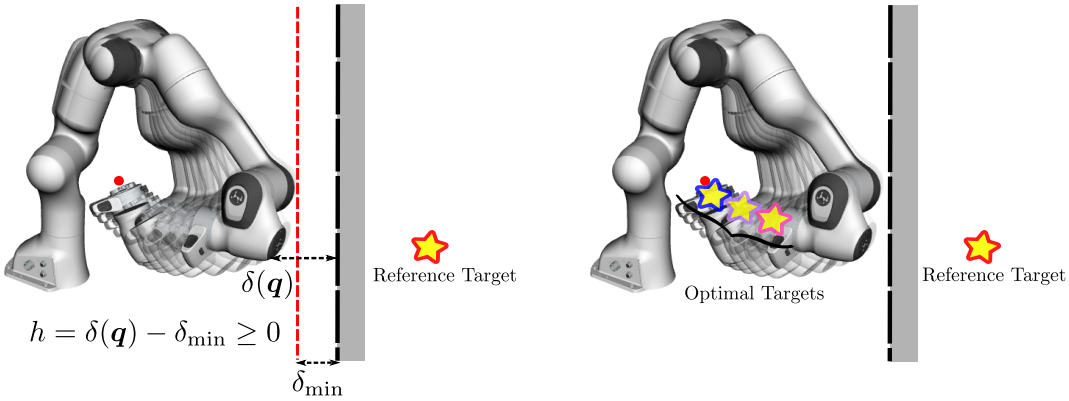


Figure 2: Classical approach (left): reference target beyond the obstacle, QP collision constraint takes care of avoiding the collision with the wall. Proposed approach (right): sequence of optimal targets (edge color gradually shifts from blue to pink according to the time evolution) are tracked by QP such that the collision with the wall is avoided. The red point denotes the initial task-space state.

Since the robot motion is essentially driven by the tasks' dynamics defined in QP, our idea is to ensure the constraints compatibility at the task level by modifying task targets to account for the limitations and constraints. For instance, instead of arbitrarily defining set-point or trajectory references and let the QP constraints take care of avoiding collisions at the risk of running into infeasibility, we compute a sequence of *optimal* targets that converge to the reference targets while satisfying the hardware limits and avoiding collision if any (2). These optimal targets are computed by the so-called reference governor (see [7], and also a recent survey in [8]).
In I.AM. we are working at formulating such a reference governor using a linear MPC layer on top of the closed-loop system constituted by the QP controller and the robot. Based on the system's closed-loop dynamics, the MPC predicts the robot state over a finite horizon and enforces the different constraints on these predicted states. Hence, MPC outputs *constraints-compatible* optimal targets to be tracked by the QP tasks.
In what follows, we describe in details the implementation we are undergoing for the time being.
We here consider a fixed-base manipulator moving in free space without contacts. The joint position, velocity and acceleration are denoted as $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$. The multi-body robot dynamics writes $\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} +$

$\mathbf{N}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{\tau}_g=\boldsymbol{\tau}$. Furthermore, mechatronic hardware limits are encoded as lower and upper bounds on joint accelerations $\ddot{\boldsymbol{q}}_{\min}\leq\ddot{\boldsymbol{q}}\leq\ddot{\boldsymbol{q}}_{\max}$ and joint torques $\boldsymbol{\tau}_{\min}\leq\boldsymbol{\tau}\leq\boldsymbol{\tau}_{\max}$. The kinematic mapping between the robot end-effector position and the joint-space is formulated as:

$$\boldsymbol{p}=f_{\boldsymbol{p}}(\boldsymbol{q})\in\mathbb{R}^m, \tag{1}$$

$$\dot{\boldsymbol{p}}=\mathbf{J}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\in\mathbb{R}^m, \tag{2}$$

$$\ddot{\boldsymbol{p}}=\mathbf{J}^{\boldsymbol{p}}\ddot{\boldsymbol{q}}+\dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\in\mathbb{R}^m. \tag{3}$$

Let the primary task-space objective given by reference end-effector position, velocity and acceleration targets $\boldsymbol{p}_{\text{ref}}(t),\dot{\boldsymbol{p}}_{\text{ref}}(t),\ddot{\boldsymbol{p}}_{\text{ref}}(t)\in\mathbb{R}^m$, and reference joint-space positions, velocities and accelerations $\boldsymbol{q}_{\text{ref}}(t),\dot{\boldsymbol{q}}_{\text{ref}}(t),\ddot{\boldsymbol{q}}_{\text{ref}}(t)\in\mathbb{R}^n$. And $f_{\boldsymbol{p}}$, $\mathbf{J}^{\boldsymbol{p}}$ are the forward kinematics and the corresponding jacobian. Furthermore, the robot safety is handled by keeping its configuration inside a set $\mathcal{C}$ defined as $\mathcal{C}=\{\boldsymbol{q}\in\mathbb{R}^n:h(\boldsymbol{q})\geq 0\}$ where $h(\boldsymbol{q})$ is a barrier function denoting the distance to the boundary $h(\boldsymbol{q})=0$ and which has to remain positive. Hereafter, we drop the potential dependencies on time and $\boldsymbol{q}$ for ease of reading.

The first step in formulating MPC is to have the dynamic model of the inner-loop to be controlled by MPC. Since the MPC computes optimal targets for the tasks, the inner-loop denotes the closed-loop formulation of the tasks combined in WBQP (whole body task and joint space QP with constraints). To do so, we need first to compute the closed-form solution of the WBQP in terms of $\ddot{\boldsymbol{q}}$, which is then mapped to the task-space to have the corresponding closed-loop task dynamics. Nevertheless, the existence of inequality constraints in WBQP makes it impossible to have an exploitable closed-form solution $\ddot{\boldsymbol{q}}$. In what follows, we consider the unconstrained WBQP. Then, we show how the constraints can be implemented in MPC.

Let us consider the WBQP without constraints

$$\min_{\ddot{\boldsymbol{q}}}\ \frac{w_{\ddot{\boldsymbol{p}}}}{2}\left\|\mathbf{J}^{\boldsymbol{p}}\ddot{\boldsymbol{q}}-\left(\ddot{\boldsymbol{p}}_{\text{pd}}-\dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\right)\right\|^2+\frac{w_{\ddot{\boldsymbol{q}}}}{2}\left\|\ddot{\boldsymbol{q}}-\ddot{\boldsymbol{q}}_{\text{pd}}\right\|^2, \tag{4}$$

where $w_{\ddot{\boldsymbol{p}}}$ and $w_{\ddot{\boldsymbol{q}}}$ are the optimization weights, which can be written in a compact form

$$\min_{\ddot{\boldsymbol{q}}}\ \frac{1}{2}\left\|\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}}\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}\ddot{\boldsymbol{q}}-\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\left(\ddot{\boldsymbol{p}}_{\text{pd}}-\dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\right)\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\ddot{\boldsymbol{q}}_{\text{pd}}\end{bmatrix}\right\|^2. \tag{5}$$

The closed-form solution of the compact WBQP (5) is given as

$$\ddot{\boldsymbol{q}}=\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}}\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}^{+}\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\left(\ddot{\boldsymbol{p}}_{\text{pd}}-\dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\right)\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\ddot{\boldsymbol{q}}_{\text{pd}}\end{bmatrix}, \tag{6}$$

where the superscript $(.)^+$ denotes the Moore-Penrose inverse

$$\begin{aligned}\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}}\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}^{+}&=\left(\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}\top} & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}}\\ \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}\right)^{-1}\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}\top} & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}\\ &=\underbrace{\left(w_{\ddot{\boldsymbol{p}}}\mathbf{J}^{\boldsymbol{p}\top}\mathbf{J}^{\boldsymbol{p}}+w_{\ddot{\boldsymbol{q}}}\mathbf{I}\right)^{-1}}_{\mathbf{G}}\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{J}^{\boldsymbol{p}\top} & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I}\end{bmatrix}\\ &=\begin{bmatrix}\sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{G}\mathbf{J}^{\boldsymbol{p}\top} & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{G}\end{bmatrix}\in\mathbb{R}^{n\times(m+n)}\end{aligned} \tag{7}$$

By replacing (7) into (6), we get

$$\begin{aligned}
\ddot{\boldsymbol{q}} &= \begin{bmatrix} \sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{G}\mathbf{J}^{\boldsymbol{p}\mathsf{T}} & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{G} \end{bmatrix} \begin{bmatrix} \sqrt{w_{\ddot{\boldsymbol{p}}}}\mathbf{I} & 0 \\ 0 & \sqrt{w_{\ddot{\boldsymbol{q}}}}\mathbf{I} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{p}}_{\mathrm{pd}} - \dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}}_{\mathrm{pd}} \end{bmatrix}, \\
&= \begin{bmatrix} w_{\ddot{\boldsymbol{p}}}\mathbf{G}\mathbf{J}^{\boldsymbol{p}\mathsf{T}} & w_{\ddot{\boldsymbol{q}}}\mathbf{G} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{p}}_{\mathrm{pd}} - \dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}}_{\mathrm{pd}} \end{bmatrix}, \\
&= \begin{bmatrix} \boldsymbol{\mathcal{J}}_{\ddot{\boldsymbol{p}}} & \boldsymbol{\mathcal{J}}_{\ddot{\boldsymbol{q}}} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{p}}_{\mathrm{pd}} - \dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}}_{\mathrm{pd}} \end{bmatrix}, \boldsymbol{\mathcal{J}}_{\ddot{\boldsymbol{p}}} \in \mathbb{R}^{n\times m},\ \boldsymbol{\mathcal{J}}_{\ddot{\boldsymbol{q}}} \in \mathbb{R}^{n\times n}, \\
&= \left(\ddot{\boldsymbol{p}}_{\mathrm{pd}} - \dot{\mathbf{J}}^{\boldsymbol{p}}\dot{\boldsymbol{q}}\right) + \ddot{\boldsymbol{q}}_{\mathrm{pd}}.
\end{aligned} \tag{8}$$

In order to predict the trajectories of the continuous domain dynamic system over a preview horizon T_{preview}, $\mathbf{A}_c$ and $\mathbf{B}_c$ need to remain constant. This implies subsequently constant Jacobians $\mathbf{J}^{\boldsymbol{p}}$ and $\mathbf{J}^h$, and their derivatives $\dot{\mathbf{J}}^{\boldsymbol{p}}$ and $\dot{\mathbf{J}}^h$. This is a valid assumption since the robot configuration is considered to undergo small changes during T_{preview}. In addition, the prediction requires the discretization of the model using the time step Δt during which $\boldsymbol{s}$ and $\boldsymbol{u}$ are constant, and such that $N\Delta t = T_{\mathrm{preview}}$ with $N \in \mathbb{N}$ is the preview-horizon length, which yields to

$$\boldsymbol{s}_{i+1} = \mathbf{A}_d\boldsymbol{s}_i + \mathbf{B}_d\boldsymbol{u}_i, \tag{9}$$

where the subscript $(.)_i$ denotes the state or command at $t = i\Delta t$, $i = 0, \ldots, N$, and $\mathbf{A}_d, \mathbf{B}_d$ are the discrete-time formulation of $\mathbf{A}_c, \mathbf{B}_c$, respectively such that

$$\begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \exp\left(\begin{bmatrix} \mathbf{A}_c & \mathbf{B}_c \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \Delta t\right) \tag{10}$$

Based on the discrete-time model (9), the preview model is constructed such that

$$\underline{\boldsymbol{s}} = \underline{\mathbf{A}}\boldsymbol{s}_0 + \underline{\mathbf{B}}\,\underline{\boldsymbol{u}}, \tag{11}$$

$$\underline{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_{N+1} \end{bmatrix} \in \mathbb{R}^{(2m+2n+2)(N+1)},\ \underline{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_0 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} \in \mathbb{R}^{(3m+3n)(N+1)}, \tag{12}$$

$$\underline{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_d \\ \mathbf{A}_d^2 \\ \vdots \\ \mathbf{A}_d^{N+1} \end{bmatrix} \in \mathbb{R}^{(2m+2n+2)(N+1)\times(2m+2n+2)}, \tag{13}$$

$$\underline{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_d & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^N\mathbf{B}_d & \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d \end{bmatrix} \in \mathbb{R}^{(2m+2n+2)(N+1)\times(3m+3n)(N+1)}, \tag{14}$$

where $\boldsymbol{s}_0$ is the initial state, $\underline{\boldsymbol{s}}$ denotes the stack of predicted states and $\underline{\boldsymbol{u}}$ is the stack of control input sequence along the preview horizon T_{preview}. The MPC computes $\underline{\boldsymbol{u}}$ that minimizes a sum of weighted quadratic cost-functions and a set of constraints on $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{s}}$. In the next subsection, we detail the MPC cost-functions and constraints formulation.

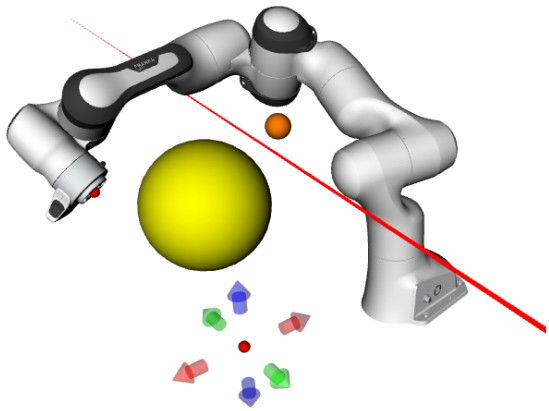


Figure 3: Panda end-effector moving towards the Cartesian target while avoiding collision with the yellow ball and keeping its CoM (orange point) behind the red line.

Generally, the criteria to be minimized in the cost-functions denote the desired performances that the closed-loop system should enjoy. Often, not all the performance criteria can be simultaneously met. Hence, depending on the context and application, to the criteria describing the desired behavior of the robot should be assigned a high weight compared to other criteria that are intended primarily for optimization purposes. Conversely, the constraints describe the limitations and bounds on both the control input and the state and have a higher priority over the performance criteria. These constraints are qualified as “soft” (small violation is tolerated) or “strict” (cannot be violated).
For instance, we can define the following performance criteria:

- steer the control input toward the reference targets:

$$\min \frac{w_1}{2} \left\| \boldsymbol{u}_i - \boldsymbol{u}_{\text{ref}} \right\|^2 = \min \frac{w_1}{2} \left\| \boldsymbol{u}_i - \begin{bmatrix} \boldsymbol{p}_{\text{ref}} \\ \dot{\boldsymbol{p}}_{\text{ref}} \\ \ddot{\boldsymbol{p}}_{\text{ref}} \\ \boldsymbol{q}_{\text{ref}} \\ \dot{\boldsymbol{q}}_{\text{ref}} \\ \ddot{\boldsymbol{q}}_{\text{ref}} \end{bmatrix} \right\|^2, \tag{15}$$

- minimize the tracking error between the state and reference targets:

$$\min \frac{w_2}{2} \left\| \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{s}_i - \begin{bmatrix} \boldsymbol{p}_{\text{ref}} \\ \dot{\boldsymbol{p}}_{\text{ref}} \\ \boldsymbol{q}_{\text{ref}} \\ \dot{\boldsymbol{q}}_{\text{ref}} \end{bmatrix} \right\|^2 = \min \frac{w_2}{2} \left\| \mathbf{C}\boldsymbol{s}_i + \boldsymbol{d} \right\|^2, \tag{16}$$

- minimize the jump on joint velocity:

$$\begin{aligned} \min \frac{w_3}{2} \left\| \dot{\boldsymbol{q}}_{i+1} - \dot{\boldsymbol{q}}_i \right\|^2 &= \min \frac{w_3}{2} \left\| \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \left((\mathbf{A}_d - \mathbf{I})\, \boldsymbol{s}_i + \mathbf{B}_d \boldsymbol{u}_i \right) \right\|^2 \\ &= \min \frac{w_3}{2} \left\| \mathbf{E}\boldsymbol{s}_i + \mathbf{F}\boldsymbol{u}_i \right\|^2, \end{aligned} \tag{17}$$

- cost on the terminal trajectory point

$$\min \frac{w_4}{2} \left\| \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{s}_{N+1} - \begin{bmatrix} \boldsymbol{p}_{\text{ref}} \\ \dot{\boldsymbol{p}}_{\text{ref}} \end{bmatrix} \right\|^2 = \min \frac{w_4}{2} \left\| \mathbf{G}\boldsymbol{s}_{N+1} + \boldsymbol{g} \right\|^2, \tag{18}$$

along with the following the constraints:

- joint position constraint:

$$\dot{\boldsymbol{q}} + \alpha \left(\boldsymbol{q} - \boldsymbol{q}_{\max} \right) \leq \mathbf{0} \tag{19}$$

$$\dot{\boldsymbol{q}} + \alpha \left(\boldsymbol{q} - \boldsymbol{q}_{\min} \right) \geq \mathbf{0} \tag{20}$$

$$\alpha \boldsymbol{q}_{\min} \leq \begin{bmatrix} \mathbf{0} & \mathbf{0} & \alpha\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{s}_i \leq \alpha \boldsymbol{q}_{\max}, \tag{21}$$

- joint velocity constraint:

$$\dot{\boldsymbol{q}}_{\min} \leq \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{s}_i \leq \dot{\boldsymbol{q}}_{\max}, \tag{22}$$

- joint acceleration constraint:

$$\ddot{\boldsymbol{q}}_{\min} \leq \frac{1}{\Delta t} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \left((\mathbf{A}_d - \mathbf{I})\, \boldsymbol{s}_i + \mathbf{B}_d \boldsymbol{u}_i \right) \leq \ddot{\boldsymbol{q}}_{\max}, \tag{23}$$

- dynamic constraint

$$\begin{aligned} \boldsymbol{\tau}_{\min} &\leq \mathbf{M}\ddot{\boldsymbol{q}}_i + \mathbf{N}\dot{\boldsymbol{q}}_i + \boldsymbol{\tau}_g \leq \boldsymbol{\tau}_{\max}, \\ \boldsymbol{\tau}_{\min} - \boldsymbol{\tau}_g &\leq \left(\frac{\mathbf{M}}{\Delta t} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} (\mathbf{A}_d - \mathbf{I}) + \mathbf{N} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \right) \boldsymbol{s}_i \\ &\quad + \frac{\mathbf{M}}{\Delta t} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{B}_d \boldsymbol{u}_i \leq \boldsymbol{\tau}_{\max} - \boldsymbol{\tau}_g, \end{aligned} \tag{24}$$

- distance constraint formulation:

$$\begin{aligned} \dot{h}_i + \alpha h_i \geq 0,\ \alpha > 0, \\ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \alpha \end{bmatrix} \boldsymbol{s}_i \geq 0. \end{aligned} \tag{25}$$

  with α denoting the convergence profile to the boundary. The lower α, the smoother the convergence, and the earlier the deceleration.

These control performances and constraints are not exhaustive. In fact, we can further consider other control objectives like minimizing the joint jerk $\left\| \dddot{\boldsymbol{q}} \right\|^2$ and Cartesian velocity of the end-effector position $\dot{\boldsymbol{p}}$.
Note that the joint-position and distance constraints could well have been simply formalized as (in a generic form)

$$h_i \geq 0. \tag{26}$$

Nevertheless, this formulation is not convenient. The constraint formulation needs to ensure two properties: the *forward invariance* and *asymptotic stability* of the set $\mathcal{C} = \{\boldsymbol{q} \in \mathbb{R}^n : h(\boldsymbol{q}) \geq 0\}$. The former property ensures that the constraint will be satisfied forward in time if $h_0 \geq 0$, whereas the latter guarantees the converges to $\mathcal{C}$ if $h_0 < 0$. Indeed, 26 does enable the first property but not the second. On the other hand, the formulation 25 exploits the Control Barrier Function (CBF) formalism and thereby does ensure the two above properties: $\dot{h} = 0$ if $h = 0$ (stop at the constraint boundary) and $\dot{h} > 0$ if $h < 0$ (increase the distance to converge back to $h = 0$).
To ensure the MPC feasibility, slack variables $\boldsymbol{\delta} \in \mathbb{R}^d$ can be considered to relax the soft constraints that are potentially in conflict with other hard constraints. $d \in \mathbb{N}$ is then the number of relaxed constraints. For example, the relaxed distance constraint is

$$\dot{h}_i + \alpha h_i + \delta_i \geq 0,\ \delta_i \in \mathbb{R} \tag{27}$$

The command vector is then extended to include the slack variables $\tilde{\boldsymbol{u}}^\mathsf{T} = \begin{bmatrix} \boldsymbol{\delta}^\mathsf{T} & \boldsymbol{u}^\mathsf{T} \end{bmatrix}$, and accordingly the matrix $\tilde{\mathbf{B}}_d = \begin{bmatrix} \mathbf{0} & \mathbf{B}_d \end{bmatrix}$. Hereafter, the tilde sign is dropped for ease of notations. In addition, the cost-function $w_5 \left\| \boldsymbol{\delta} \right\|^2$ is then added to penalize the slack variables amplitude. Another key reason for formulating the joint-position constraint as a joint-velocity constraint in 21 is to reduce the size of the slack variables dimension d, and optimize the computation time. In fact, instead of using $\boldsymbol{\delta} \in \mathbb{R}^{d=2n}$ to relax separately the joint-position (expressed as in 26) and velocity constraints 22, only $\boldsymbol{\delta} \in \mathbb{R}^{d=n}$ is sufficient to relax both constraints since they are expressed in terms of velocity in 21 and 22.
Slack variables could have also been considered to ensure the feasibility of the reactive WBQP by relaxing (softening) some or all the constraints. Yet, this will only ensure point-wise-in-time feasibility at the expense of constraints satisfaction. In contrast, relaxed MPC ensures the feasibility along the preview horizon with the predicted states. Furthermore, we show in the experiment section that the relaxation of the constraint 25 does not necessarily imply the violation of the distance constraint.
Using the compact system model 11, all the weighted cost-functions from 15 to 18 and constraints from 21 to 25 can be expressed in a compact form that depends only on $\underline{\boldsymbol{u}}$, and which yields to the following linear MPC formulated as a weighted-prioritized QP

$$\min_{\underline{\boldsymbol{u}}} \left\| \mathbf{S}\boldsymbol{s}_0 + \mathbf{U}\underline{\boldsymbol{u}} + \boldsymbol{v} \right\|^2 \tag{28a}$$

$$\text{s.t: } \mathbf{W}\boldsymbol{s}_0 + \mathbf{Y}\underline{\boldsymbol{u}} + \boldsymbol{z} \leq \mathbf{0} \tag{28b}$$

28a encompasses the weighted sum of cost-functions on control (15), state (16), mixed (17) and terminal trajectory point (18), and similarly for the constraint 28b.
At each MPC instantiation, the initial state $\boldsymbol{s}_0$ is computed based on the actual robot state using the sensory feedback. Once the MPC computation is performed, only the first control component $\boldsymbol{u}_0$ is sent as optimal targets for the end-effector and posture tasks defined in the WBQP such that

$$\ddot{\boldsymbol{p}}_{\mathrm{pd}} = \ddot{\boldsymbol{p}}^* - \mathbf{D}_{\dot{\boldsymbol{p}}} \left(\dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}^*\right) - \mathbf{P}_{\boldsymbol{p}} \left(\boldsymbol{p} - \boldsymbol{p}^*\right) \tag{29}$$

$$\ddot{\boldsymbol{q}}_{\mathrm{pd}} = \ddot{\boldsymbol{q}}^* - \mathbf{D}_{\dot{\boldsymbol{q}}} \left(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^*\right) - \mathbf{P}_{\boldsymbol{q}} \left(\boldsymbol{q} - \boldsymbol{q}^*\right) \tag{30}$$

Then, WBQP computes the joint commands (joint-torque $\boldsymbol{\tau}$ for torque-controlled robots, and joint-acceleration $\ddot{\boldsymbol{q}}$ double integrated to have $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ for kinematic-controlled robots). Note that, unlike the reactive WBQP that is based on the task gains, the robot motion dynamics is tuned by the MPC weights especially w_1 and w_2 which denotes how fast the optimal target and predicted state converge to the reference target, respectively.

n	m	d	N	Δt	α	$w_{\ddot{\boldsymbol{p}}}$	$w_{\ddot{\boldsymbol{q}}}$	$\mathbf{P}_{\boldsymbol{p}}$	$\mathbf{D}_{\dot{\boldsymbol{p}}}$	$\mathbf{P}_{\boldsymbol{q}}$	$\mathbf{D}_{\dot{\boldsymbol{q}}}$
7	3	9	10	0.12	$\frac{1}{4\Delta t}$	500	10	$10\mathbf{I}$	$2\sqrt{10}\mathbf{I}$	$1\mathbf{I}$	$2\mathbf{I}$

w_1	w_2	w_3	w_4	w_5
2000	500	30	200	$20 \cdot 10^6$

Table 1: Meta-parameters used in simulation.

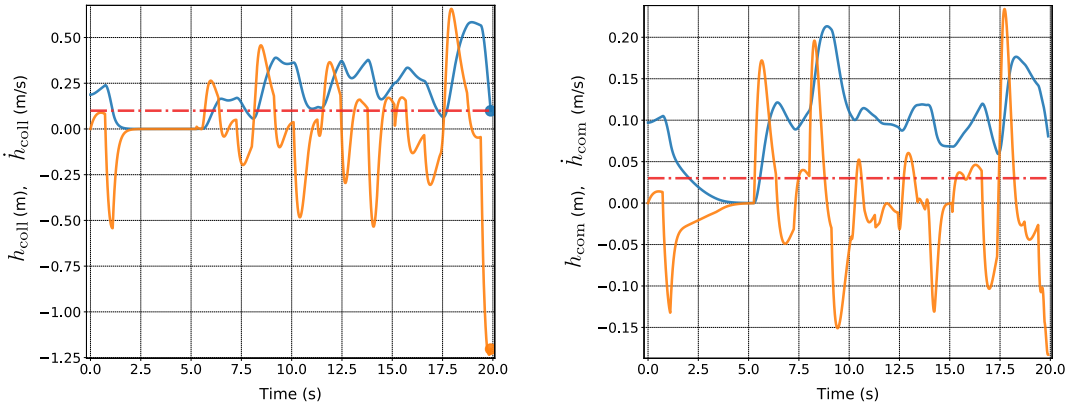


Figure 4: Time evolution of h_* (blue) and $\dot{h}_*$ (orange) in the case of reactive WBQP control scheme (baseline approach): (a) h_{coll} and $\dot{h}_{\mathrm{coll}}$, (b) h_{com} and $\dot{h}_{\mathrm{com}}$. The collision and CoM constraints are inserted in WBQP if h_{coll} and h_{com} are less than their respective margins shown in red dash-dotted line.

Preliminary experiments are being conducted on the Panda robot.

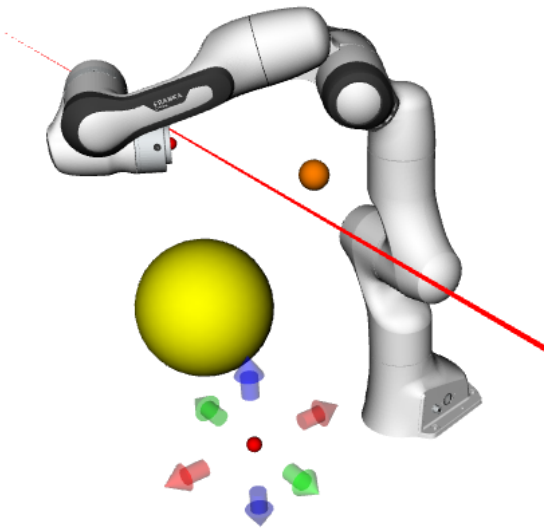


Figure 5: Robot configuration when reactive WBQP controller (baseline approach) fails to find a solutions.

4 Self-Correcting QP-Based Robot Control

Quadratic programming (QP)-based controllers allow many robotic systems, such as humanoids, to successfully undertake complex motions and interactions. However, these approaches rely heavily on adequately capturing the underlying model of the environment and the robot's dynamics. This assumption, nevertheless, is rarely satisfied, and we usually turn to well-tuned end-effector PD controllers to compensate for model mismatches.

In this work, EPFL proposed to augment traditional QP-based controllers with a learned residual inverse dynamics (IDs) model and an adaptive control law that adjusts the QP online to account for model uncertainties and unforeseen disturbances. In particular, we propose:

1. learning a residual IDs model using the Gaussian Process and linearizing it so that it can be incorporated inside the QP-control optimization procedure and
2. a novel combination of adaptive control and QP-based methods to avoid the manual tuning of end-effector PID controllers and faster convergence in learning the residual dynamics model.

In simulation, we extensively evaluate our method in several robotic scenarios ranging from a 7-degrees of freedom (DoFs) manipulator tracking a trajectory to a humanoid robot performing a waving motion for which the model used by the controller and the one used in the simulated world do not match (unmodeled dynamics). Finally, we also validate our approach in physical robotic scenarios where a 7-DoFs robotic arm performs tasks where the model of the environment (mass, friction coefficients, etc.) is not fully known.

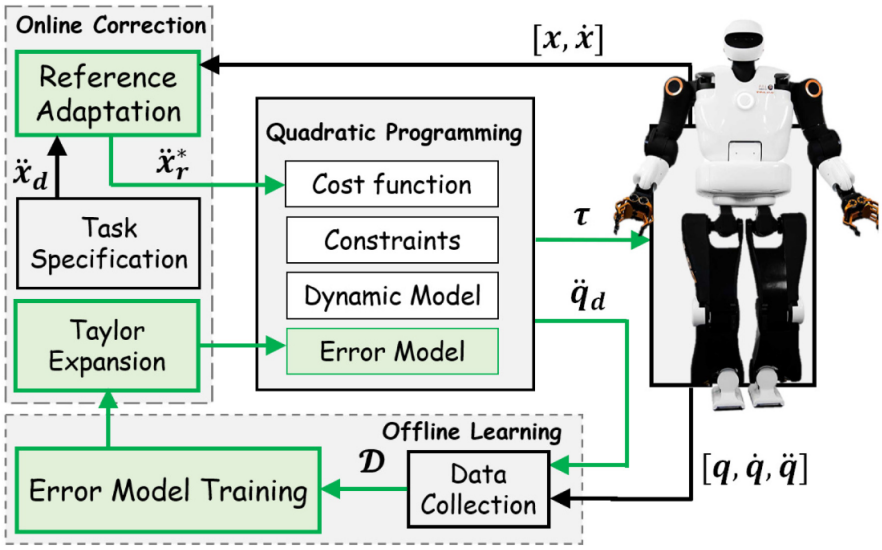


Figure 6: The proposed framework for robot torque control uses a QP scheme. The green components indicate our contributions developed in this study. This framework computes the joint torques τ to realize a desired end-effector acceleration $\ddot{x}_d$. Through online reference adaptation, given the feedback from robot end-effectors, and error model exploitation, using Taylor expansion, our approach enables the QP to correct itself such that the expected joint accelerations, $\ddot{q}_d$, converge to the actual values, $\ddot{q}$.

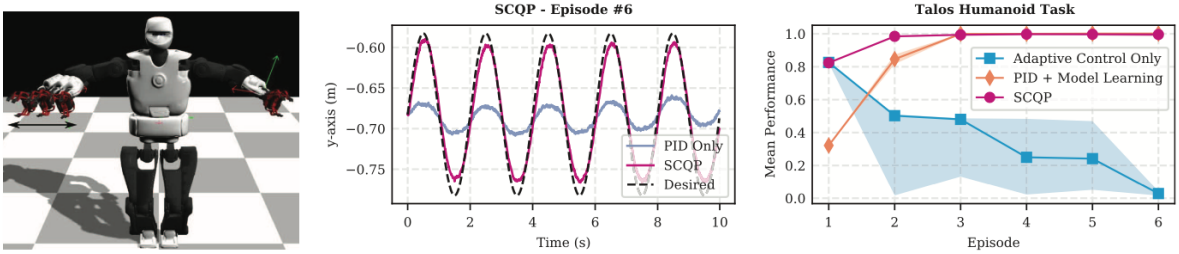


Figure 7: Successful trial of the Talos task. Through ID learning and online adaptation, SCQP learns to perform the waving task (left) without falling despite relatively big mismatches in the mass of the arms and the friction coefficient. The phase tracking accuracy during the waving task (middle) indicates the effectiveness of model learning in SCQP after only a few trials compared to the other approaches (right). Solid lines are the median over ten replicates, and the shaded regions are the areas between the 25th and 75th percentiles.

This work was recently published in Transactions of Systems, Man, and Cybernetics [9].

5 Stability, Robustness and Performance

This task has started in M6 and will end in M42. The goal of this task is to analyze the impact-aware QP control approach from a control theory perspective. There are two concurrent studies: the first is conducted by CNRS and the second conducted by TU/e.

5.1 General stability of QP control

The first study concerns the general stability of the QP w.r.t to the task's gains and the forward invariance of the QP constraints. Task-space quadratic programming (QP) is an elegant approach for controlling robots subject to constraints. Yet, in the case of kinematic-controlled (i.e., high-gains position or velocity) robots, closed-loop QP control scheme can be prone to instability depending on how the gains related to the tasks or to the distance-constraints are chosen. In I.AM. we address such instability shortcomings considering that T4.1 will result in a QP with impact-aware enhanced constraints. First, we demonstrated by simulation and theoretically the non-robustness of the closed-loop system against non-modeled dynamics, such as those relative to joint-dynamics, flexibilities, external perturbations, etc. Then, we proposed in [10] a robust QP control formulation based on high-level integral feedback terms in the task-space (including the distance-constraints). The proposed method is formally proved to ensure the closed-loop robust stability. Our approach is intended to be applied to any kinematic-controlled robot under practical assumptions and assessed through experiments on a fixed-base robot performing stable fast motions, and a floating-base humanoid robot robustly reacting to perturbations to keep its balance.
The stability analysis is based on Lyapunov theory. Our solution is elegant as it applies directly to the existing QP templated tasks (including the impact-aware ones). In contrast to [11] where the joint controller model is required, our approach is straightforwardly used with any kinematic-controlled robot as it does not require the exact knowledge of the joint-dynamics model which we only assume to be Input-to-State-Stable (ISS). We also further advanced [2] in two ways: (i) we account for the lack of joint-dynamics in closed-loop; (ii) we include the distance-constraints in the stability analysis. Although we define the set robust stability similarly to [12], our approach is different in three main points: (i) the factors against which the robustness is enforced, (ii) the formalism adopted to prove robustness, and (iii) the nature of the term added to achieve robustness.
To sum-up, our contributions in this task are as follows:

- Task-space QP formulation that guarantees global robust task-stability;
- Robust distance-constraint formulation;
- Task stability investigations in the case of multi-objective weighted-prioritized robust constrained QP;
- Integration into our existing framework and validation on a robotic manipulator and a humanoid robot.

This work is published in IEEE TRO [10].

5.2 Robustness to velocity jumps in impact-aware control

The second study, conducted by the TU/e, attempts to robustify the QP controller to ensure that velocity jumps triggered by impacts do not lead to unstable behavior. Such unstable behavior could be triggered by peaks in the velocity tracking error, if the contact state of the reference used for control does not match that of the actual system as a result of a mismatch between the expected and actual impact times. This can in turn trigger unexpected jumps in the input signals, potentially destabilizing the system.
To deal with this phenomenon within the context of QP control, the existing framework of reference spreading is extended. Reference spreading, initially defined in [13], removes input peaks in tracking control by defining an extended ante-impact and post-impact reference that are connected by an analytical impact map, and overlap around the nominal impact time. By switching to the post-impact reference upon impact detection, the reference corresponding to the actual contact state is always used, making the system robust to uncertainties in the impact timing.
In [14], this framework of reference spreading has been extended to blend with task-space QP control rather than the previously used state feedback control, allowing the reference spreading framework to be used with the QP controllers developed in this project. Furthermore, this approach extended the framework to avoid input peaks for control of motions with nominally simultaneous impacts by introducing a novel interim-impact phase aside from the ante- and post-impact phases. This is highly relevant for the GRAB scenario, as this makes the system robust to non-simultaneity of the impacting robots and points on the end effector surfaces. [15] further extends the reference spreading framework by defining a similar impact-aware QP control approach for time-invariant references, much like the Dynamical Systems-based control approach developed by EPFL, adding robustness to changing initial conditions or unexpected disturbances that would cause a large tracking error for time-based tracking approach. In [16], this time-invariant impact-aware QP control approach is further extended to suit for a dual-arm system, pursuing simultaneity of impacts between both robots, even when the robots start in asymmetric initial conditions with respect to the expected impact points.

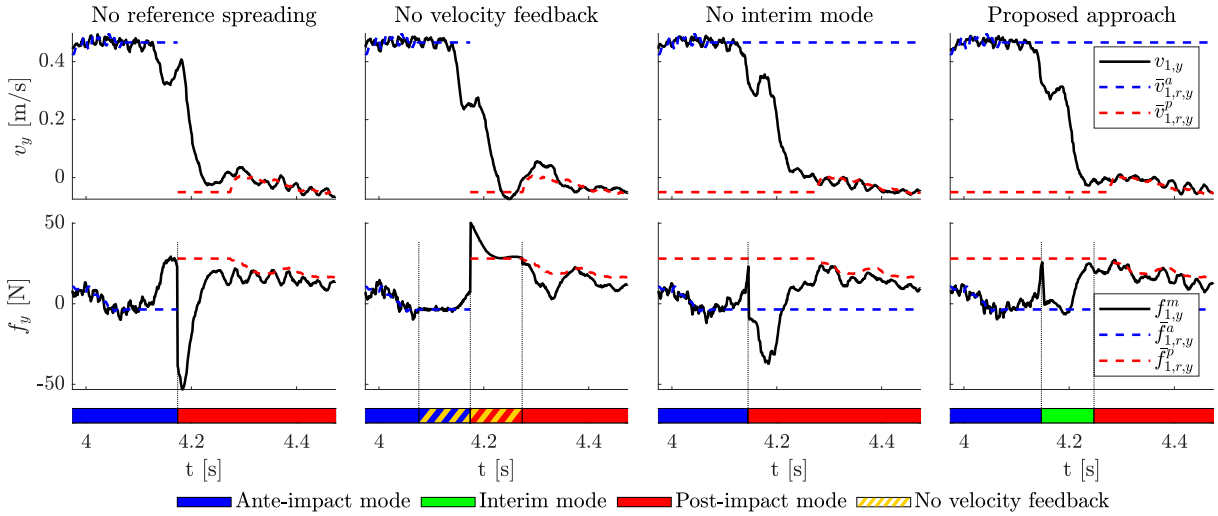


Figure 8: Normal velocity and input force of one of the robots during GRAB experiments with reference spreading (RS), compared against three baseline control approaches. Red and blue dashed lines indicate the (extended) ante- and post-impact references, and vertical dashed lines indicate mode switching times, triggered by the nominal impact time (left two plots) or impact detection (right two plots).

While the approaches in [14, 15, 16] are validated by means of simulations with a planar robot, experimental validation has been performed in [17] with two Franka Emika robots, showing the effectiveness of the time-based reference spreading approach in the context of the GRAB scenario. Figure 8, taken from [17], shows these results for the proposed reference spreading-based control approach and three baseline control approaches. For the baseline control approach that does not use reference spreading at all, and the approach that uses reference spreading without the aforementioned interim-impact mode, large peaks in the desired force signal f_y can be observed. For the baseline approach that removes velocity feedback around the impact time to prevent peaking, this force peak is indeed less severe. However, a large vibration in the velocity v_y can be observed, causing poor performance in the execution of the desired grasping task. The proposed approach, which uses reference spreading with an interim-impact mode, is the only approach that shows a reduction of the peak forces without losing control performance.
For the future, also the time-invariant reference spreading approach of [15, 16] will be experimentally validated. Finally, the aim is to show a formal proof of stability for the proposed reference spreading control approaches.

6 Non-prehensile manipulation control via hitting

The hitting dynamical system is a function that provides the desired end effector velocity of the robot given its position and the directional inertia (described in Task 2.3.1a). This direction is the direction of hitting. If we are aware of a desired directional inertia value or a property of the desired directional inertia such as it should be locally maximum, then we can add these desired tasks to the Null space of the controller. This allows for the robot to move along the path and allow for the inertia to possess the desired property, if feasible. Let, $\lambda_h(q) = \hat{h}^T \Lambda_t(q) \hat{h}$ be the directional inertia, where $\hat{h}$ is the unit vector in the hitting direction, and $\Lambda_t(q)$ is the translational task space inertia of the robot at its end effector. As the robot end effector moves along $f(\chi)$ (the dynamical system), the following controller allows for increasing the inertia perceived at the end effector along the direction of the vector field $f(\chi)$

$$\dot{q} = J^{\dagger}(q) f(\chi) + N[\beta_1(q_m - q) + \beta_2 \nabla_q \lambda_h(q)], \quad \beta_1, \beta_2 \in \mathbb{R}_+$$

where, $N = I - J(q)^{\dagger} J(q)$ is the dynamically consistent Null space for joint velocities, q_m is a maximum velocity manipulability confihuration at the desired hitting cartesian position and β_1, β_2 are hyperparameters, which control the relative weights of achieving q_m and moving in the direction of increasing directional inertia.
Instead of maximizing the directional inertia, if we want to achieve a desired directional inertia λ^* while tracking the dynamical system, with priority to the latter, in the null space, $\nabla_q \lambda_h(q)$ is multiplied with $(\lambda_h(q) - \lambda^*)$. This changes the joint velocities towards the configuration with the desired directional inertia if the redundancy allows for it.

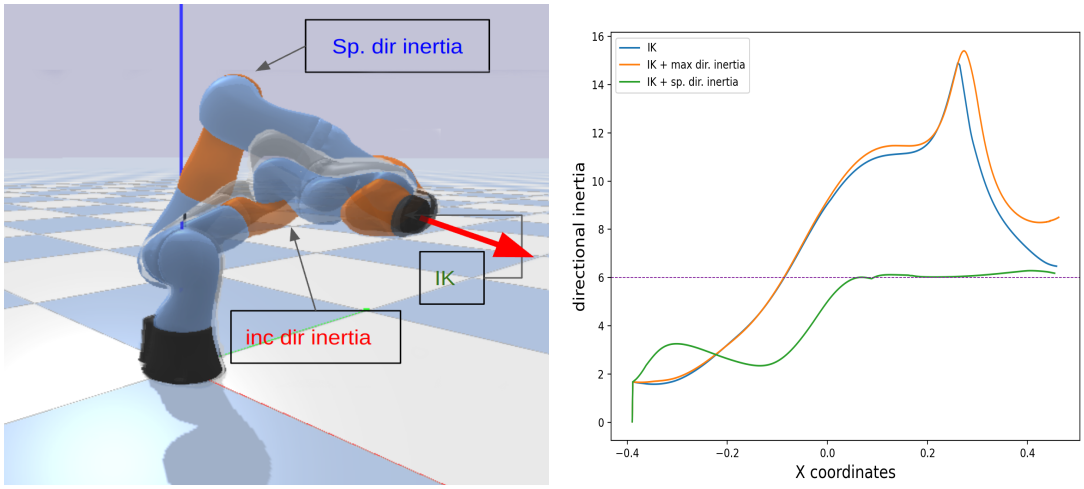
$$\dot{q} = J^{\dagger}(q) f(\chi) + N[\beta_1(q_m - q) + \beta_2(-\nabla_q \lambda_h(q)(\lambda_h(q) - \lambda^*))], \quad \beta_1, \beta_2 \in \mathbb{R}_+$$

The effect of the above controllers can be seen in the figure 9. The increasing directional inertia controller tries to align the entire robot in the direction of hitting, while the specific directional inertia controller leads to a a part of the robot aligned in the hitting direction.
One step further from the directional inertia, we also move in the direction of achieving the desired translational inertia matrix (Λ^*). Stein distance is used as a distance metric between the current and desired inertia matrix, represented as $g(q)$. We minimize the distance between the current inertia matrix and the desired inertia matrix, while following a DS under joint position and velocity constraints.

$$\begin{aligned} \dot{q} = & \underset{\dot{q}}{\text{argmin}} \quad \frac{1}{2} \left\| f(\chi) - J\dot{q} \right\|_2^2 + k_1(\nabla_q g(q))\dot{q} + k_2 \left\| \dot{q} \right\|_2^2 \\ \text{s.t.} \quad & \dot{q}_{min} \leq \dot{q} \leq \dot{q}_{max}, \quad q_{min} \leq q \leq q_{max} \\ & k_1, k_2 \in \mathbb{R}_+ \end{aligned} \tag{31}$$

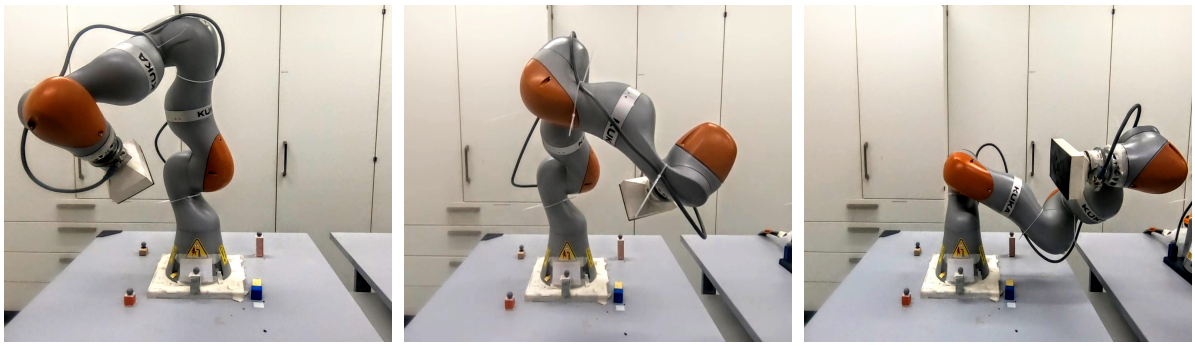
where, k_1 and k_2 are the weights for the penalties on derivative of the stein distance and the joint velocity norm. The figure 10 shows that following the inertia based controller leads to different configuration, the inertia of which is closer to the desired inertia matrix. The code for the controllers with the robot simulation is publicly available here [3]
This work has been accepted for publication in IEEE T-RO.

[3] https://github.com/epfl-lasa/hitting$_s im.git$



(a) (b)

Figure 9: Figure (a): The three different control systems lead to different joint configurations at the desired final position. The IK controller leads to the joint configuration closest to the initial configuration. With the increasing directional inertia controller, the robot's configuration is aligned with the desired direction. Since the specific directional inertia is lower than the maximum achievable in the desired direction and the directional inertia achieved by the inverse kinematics controller, the configuration achieved with the specific inertia controller is least aligned with the hitting direction; Figure (b): Quantitative comparison of the controllers - The three curves show the different directional inertia of the robot while following a trajectory with three different controllers. The green curve shows the controller trying to maintain the directional inertia along the trajectory equal to 6 units while following the trajectory.



(a) (b) (c)

Figure 10: Figure (a): This is the starting configuration of the robot; Figure (b): Final configuration of the robot under the IK Controller. This configuration is moving away from the desired inertia matrix; Figure (c): Final configuration of the robot under the Inertia QP Controller. This configuration is closer to the desired inertia configuration while achieving the final position

7 Benchmarking

Benchmarking impact-aware QP control was made through the APIs developed for all the robots used in I.AM.
In mc_rtc, an interface is used to enable communication between the control framework and a robot's hardware or simulation. The role of an interface is to provide inputs to the control framework – such as force sensors, joint sensors' or inertial measurement units' readings – and forward the output of the controller to the real or simulated robot.
The benefit of this approach with regard to benchmarking is that the same controller can run in simulation and on real hardware. The only change is related to the interface being used. Moreover, the same controller can even be used with different robots provided it accounts for minor differences – for example, frame names.
Within the scope of I.AM., four APIs were developed to allow us to deploy impact-aware QP control on three robotic platforms used within the project – Franka Emika Panda robots, Univeral Robots UR 10 robot, and KUKA IIWA7 and IIWA14 robots – as well as one interface for enabling the usage of the control framework within the Algoryx Dynamics simulator.
The four interfaces that were developed for benchmarking I-Control are as follows:

1. mc_click [4] integrates mc_rtc and the Algoryx Dynamics simulation by using the Click protocol jointly developed by CNRS and Algoryx. This protocol is described in Deliverable 1.2;
2. mc_rtc_ros_control [5] provides an interface between mc_rtc and the ros_control framework [6]; this is used to control Universal Robots robots;
3. mc_franka [7] provides an interface between mc_rtc and libfranka [8]; this is used to control Franka Emika Panda robots;
4. mc_kuka_fri [9] is an interface between mc_rtc and KUKA FRI [10]; this is used to control KUKA robots;

Furthermore, the mc_iam [11] plugin was developed to provide a hardware-agnostic interface to the vacuum gripper used within the project.

[4] https://gitlab.tue.nl/h2020-i-am-project/mc_click
[5] https://github.com/mc-rtc/mc_rtc_ros_control
[6] http://wiki.ros.org/ros_control
[7] https://github.com/jrl-umi3218/mc_franka
[8] https://frankaemika.github.io/docs/libfranka.html
[9] https://github.com/gergondet/mc_kuka_fri
[10] https://www.kuka.com/en-us/products/robotics-systems/software/system-software/sunriseos
[11] https://gitlab.tue.nl/h2020-i-am-project/mc_iam

8 Conclusion

In this Deliverable D4.1, we provided an overview of the publications associated with the I.Control framework. The report focused essentially on tasks T4.1 to T4.3 and provided approach summaries and main results related to each publication pertaining to each task.
More specifically, the report presented four main methods, where one is used to formulate a QP-aware controllers, the second one is on model preview and adaptive control using mpc-governor, and the last two concurrent methods are formulated around the general stability of the QP .
These solutions are tested on real robots, using the QP control framework that is already made available to all I.AM. partners that required it. Moreover, in Section 7, all the APIs that were developed within I.AM. are briefly described.
The final solutions of I.Control will be integrated with those of I.Model, I.Sense and I.Learn and then validated in the GRAB scenario, which will be reported in Deliverable 5.5.

9 REFERENCES

References

[1] Y. Wang, N. Dehio, and A. Kheddar, "Predicting post-impact joint velocity jumps on kinematics controlled manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6226 – 6233, 2022.

[2] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE-Transactions on Automatic Control*, vol. 63, no. 6, pp. 1632–1647, Jun. 2018.

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

[4] Y. Wang and A. Kheddar, "Impact-friendly robust control design with task-space quadratic optimization," in *Proceedings of Robotics: Science and Systems*, vol. 15, Freiburg, Germany, 24-26 June 2019, p. 32.

[5] Y. Wang, N. Dehio, and A. Kheddar, "On inverse inertia matrix and contact-force model for robotic manipulators at normal impacts," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3648–3655, 2022.

[6] Y. Wang, D. Niels, A. Tanguy, and A. Kheddar, "Impact-aware task-space quadratic-programming control," *International Journal of Robotic Research*, 2023. [Online]. Available: https://arxiv.org/pdf/2006.01987.pdf

[7] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415–419, 1998.

[8] E. Garone, S. Di Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306–328, 2017.

[9] F. Khadivar, K. Chatzilygeroudis, and A. Billard, "Self-correcting quadratic programming-based robot control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 8, pp. 5236–5247, 2023.

[10] M. Djeha, P. Gergondet, and A. Kheddar, "Robust task-space quadratic programming for kinematic-controlled robots," *IEEE Transactions on Robotics*, p. In press, 2023.

[11] A. Singletary, S. Kolathaya, and A. D. Ames, "Safety-critical kinematic control of robotic systems," *IEEE Control Systems Letters*, vol. 6, pp. 139–144, 2022.

[12] S. Kolathaya and A. D. Ames, "Input-to-state safety with control barrier functions," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 108–113, 2019.

[13] A. Saccon, N. Van De Wouw, and H. Nijmeijer, "Sensitivity analysis of hybrid systems with state jumps with application to trajectory tracking," in *53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 3065–3070.

[14] J. J. Van Steen, N. Van de Wouw, and A. Saccon, "Robot Control for Simultaneous Impact Tasks via Quadratic Programming Based Reference Spreading," in *American Control Conference*. IEEE, 2022.

[15] ——, “Robot Control for Simultaneous Impact Tasks through Time-Invariant Reference Spreading,” in *American Control Conference*. IEEE, 2023.

[16] J. J. Van Steen, A. Coşgun, N. Van de Wouw, and A. Saccon, “Dual Arm Impact-Aware Grasping through Time-Invariant Reference Spreading Control,” in *arXiv:2212.00877*, 2022.

[17] J. Van Steen, G. van den Brandt, N. Van de Wouw, J. Kober, and A. Saccon, “Quadratic Programming-based Reference Spreading Control for Dual-Arm Robotic Manipulation with Planned Simultaneous Impacts,” in *arXiv:2305.08643*, 2023.