

Time-optimal Motion Planning for Physical Human-robot Interaction

by

Aruzhan Karekeshova

Submitted to the Department of Robotics and Mechatronics
in partial fulfillment of the requirements for the degree of

Master of Science in Robotics

at the

NAZARBAYEV UNIVERSITY

April 2026

© Nazarbayev University 2026. All rights reserved.

Author
Department of Robotics and Mechatronics
April 17, 2026

Certified by
Matteo Rubagotti
Professor
Thesis Supervisor

Accepted by
Elizabeth Arkhangelsky
Dean, School of Engineering and Digital Sciences

Time-optimal Motion Planning for Physical Human-robot Interaction

by

Aruzhan Karekeshova

Submitted to the Department of Robotics and Mechatronics
on April 17, 2026, in partial fulfillment of the
requirements for the degree of
Master of Science in Robotics

Abstract

Collaborative robot manipulators that share their workspace with humans must complete the assigned tasks efficiently and guarantee safety by controlling speed within the determined limits. Existing safety-aware controllers enforce distance-proportional speed limits derived from ISO/TS 15066 to prevent injury, but the cost functions were not formulated to reach the goal quickly. This research proposes a Time-optimal Non-linear Model Predictive Control method that addresses this gap. The approach integrates an exponential L1 weighting cost with Speed and Separation Monitoring safety constraints within a single receding-horizon optimal control problem. The designed system drives the robot to reach the predefined goal configurations in the minimum number of prediction steps allowed by the human proximity. The controller is implemented using CasADi and acados on a UR5 robot manipulator and tested in a PyBullet simulation.

Thesis Supervisor: Matteo Rubagotti

Title: Professor

Contents

1	Introduction	11
1.1	Physical Human-Robot Interaction and Safety Standards	12
1.2	Nonlinear Model Predictive Control for Robot Manipulators	13
1.3	Safety-aware NMPC for Physical Human-Robot Interaction	14
1.4	Time-optimal Motion Planning	16
2	System Modeling	19
2.1	UR5 Robot Model	19
2.1.1	Overview	19
2.1.2	Forward Kinematics	19
2.1.3	Cartesian Velocities of Sphere Centers	21
2.1.4	Prediction Model	21
2.1.5	Robot Bounding Sphere Radii	22
2.2	Human Body Representation	22
2.2.1	Bounding Sphere Model	22
2.2.2	Motion-capture Data Pipeline	24
2.3	Simulation Environment and Control Architecture	24
2.3.1	PyBullet Simulation	24
2.3.2	Two-level Control Architecture	25
2.4	Human Safety Constraint	26
3	Problem Formulation	29
3.1	Time-optimal NMPC with Human Safety Constraints	29

3.1.1	Motivation	29
3.1.2	OCP Formulation	29
3.1.3	Interaction Between Time-optimality and Safety	30
4	Implementation	33
4.1	Overview	33
4.2	Software Stack	34
4.3	NMPC Solver Generation	34
4.3.1	Common Elements	34
4.3.2	Discrete-time Prediction Model	35
4.3.3	TO-NMPC Solver	36
5	Results and Discussion	39
5.1	Experimental Setup	39
5.2	Solver Timing	40
5.3	Baseline: Quadratic-Cost Safety-Aware NMPC	41
5.3.1	Robot-Only Condition	41
5.3.2	Human-Present Condition	43
5.4	Proposed: Time-Optimal Safety-Aware NMPC	46
5.4.1	Robot-Only Condition	46
5.4.2	Human-Present Condition	47
5.5	Discussion	50
5.5.1	Confirmation of the Central Claim	50
5.5.2	The Human-Present Speedup is Larger than the Robot-Only Speedup	50
5.5.3	Solver Timing Interpretation	50
6	Conclusion	53
A	Tables	55
B	Figures	57

List of Figures

2-1	Simulation environment.	25
5-1	QC-NMPC joint trajectories - robot-only condition (no human present). From top: joint positions, actual joint velocities, reference joint velocities from the high-level NMPC, and joint accelerations over a 60-second run. The robot performs repeated point-to-point motions between goal configurations.	42
5-2	QC-NMPC joint trajectories - human-present condition. Layout identical to Figure 5-1.	44
5-3	SSM constraint satisfaction for the QC-NMPC (human-present condition). Each subplot corresponds to one of the seven robot bounding sphere test points. Red: SSM speed limit $\tilde{v}_i$ (distance-proportional). Black: actual Cartesian sphere speed v_i . Blue dashed lines: instants of closest human approach.	45
5-4	TO-NMPC joint trajectories - robot-only condition (no human present). Layout identical to Figure 5-1. $\gamma = 3.2$	46
5-5	TO-NMPC joint trajectories - human-present condition. Layout identical to Figure 5-1. $\gamma = 2.6$	48
5-6	SSM constraint satisfaction for the TO-NMPC (human-present condition). Layout identical to Figure 5-3. Red: SSM speed limit $\tilde{v}_i$. Black: actual sphere speed v_i . Blue dashed lines: closest human approach instants.	49

List of Tables

2.1	Robot bounding sphere offset translations.	20
2.2	Robot bounding sphere radii.	23
2.3	Human bounding sphere radii (representative values).	24
4.1	TO-NMPC solver configuration (<code>to_disc_high_mpc_accel</code>).	36
5.1	High-level solver solve times (ms) over 240 steps.	40

Chapter 1

Introduction

The development of sensing and computation increased the deployment of robotic systems in environments shared with humans. Modern industrial robots are designed to work alongside human operators in a shared environment, augmenting human capabilities rather than replacing them entirely. The shift from isolated automation to human-robot collaboration is a fundamental conflict that motivates the present research. A robot in a shared workspace must be effective in completing assigned tasks in the required time to justify its presence and add value to productivity. A robot has to operate safely at all times, so that its movements do not pose any threat to the human co-workers in the same workspace. These two objectives are not naturally aligned. If the robot has been programmed to optimize speed, it can generate unsafe motions in the presence of a human, and if it was designed for safety, it can move very slowly so that its operational efficiency is strictly reduced.

Time-optimal motion planning is the problem of finding the fastest feasible trajectory between two configurations and in the absence of humans the robot operates at the limits of its kinematics and dynamics. In the shared workspace where the environment is no longer static, safety constraints become dynamic and dependent on human proximity, and the violations of those constraints are potentially injurious to human beings.

1.1 Physical Human-Robot Interaction and Safety Standards

Physical human-robot interaction became one of the main challenges in modern robotics with the introduction of collaborative robots in shared workspaces. The international standard ISO/TS 15066 [8] provides a regulatory framework for collaborative robot operation and defines the Speed and Separation Monitoring (SSM) mode as a primary safety mechanism. According to SSM constraints, the robot’s speed need to be reduced as the distance between the robot and human decreases, to make sure the robot can come to a complete stop before physical contact occurs.

Zanchettin et al. [19] designed a real-time implementation of the SSM condition for industrial shared environments and verified by experiments that speed-dependent safety monitoring can be integrated into robot control without overly restrictive safety constraints.

Farajtabar et al. [3] provided an extensive survey of contact-based physical human-robot interaction, including perception, planning, and control, and identified the balance between safety and task efficiency as a recurring problem. Their work offers a structured classification of methods that extends to reactive collision avoidance, impedance control, and optimisation-based planning, and concludes that no existing method resolves the problem of achieving high productivity while still guaranteeing human safety. Another parallel area of research focuses on predicting human motion to improve the quality of safety constraints.

Zheng et al. [22] consider the issue of human arm movement prediction to avoid the collisions in a shared environment, demonstrating that the integration of predictive data of the operator’s reaching trajectory into an SSM-based controller minimizes unnecessary stops. This inspires the adoption of prerecorded motion-capture datasets as an approximation of online human motion estimates in the simulation experiments of the current study.

1.2 Nonlinear Model Predictive Control for Robot Manipulators

Model Predictive Control is an optimization-based control strategy that predicts the behavior of the controlled system over a finite prediction horizon and adapts the robot responses considering system constraints and limitations.

At the numerical layer, the BLASFEO library [6] provides a high-performance set of linear algebra functions adapted to the small to medium dense matrix problems that occur in embedded optimal control.

The HPIPM framework [5], which builds on BLASFEO, is a high-performance structured programming solver intended for MPC loops. It offers interior-point methods with full and partial condensing options that are exploited in this work.

Both libraries are key components of the acados [16] software stack, a modular open-source framework that generates optimised C code to solve SQP-RTI models and offers Python and MATLAB interfaces.

CasADi [1] solves the symbolic layer of all optimal control problem expressions - forward kinematics, sphere velocities, and constraint functions, automatically differentiating first and second-order derivatives for arbitrary nonlinear expressions, and making the resulting solvers fast and analytically correct.

Astudillo et al. [2] demonstrated the practical feasibility of real-time NMPC for robot manipulators by developing a position and orientation tunnel-following NMPC scheme for 7-DOF robots that achieves solution times below 5 ms using acados and CasADi with sequential convex quadratic programming.

More recently, Yaren and Kizir [18] suggested an NMPC using Spatial Operator Algebra (SOA) to model dynamics with 60-70% lower computation costs than Newton-Euler-based implementations, and showed real-time closed-loop experiments on a 5-DOF arm with external disturbances.

The related issue is discussed by Faroni et al. [4] in terms of trajectory-scaling, which suggests a predictive methodology to scale up an online trajectory with joint velocity, acceleration, and torque constraints. They minimise the online calculation by

parametrising inputs and iteratively linearising the OCP around the previous prediction, making it a quadratic programme that can be solved at 1 ms sampling periods. Their formulation shows that receding-horizon approaches can be used to solve kinodynamically constrained robot motion at industrial control rates, and motivates the adoption of a discrete-time second-order integrator prediction model in the current work.

1.3 Safety-aware NMPC for Physical Human-Robot Interaction

One of the main challenges in collaborative robotics is to maintain productivity while guaranteeing human safety in shared workspaces. The authors of [13] present a Non-linear Model Predictive Control framework for safe physical human-robot interaction. Their work relies on the Speed and Separation Monitoring principle, which assigns velocity limits dependent on the distance between the human and the robot. Integrating SSM-based safety constraints into a receding-horizon optimization problem permits the controller to actively replan the robot’s motion to avoid unnecessary stops. The control structure consists of a long-horizon NMPC responsible for global trajectory planning and a short-horizon NMPC used for fast local adjustment to human motion. Experiments demonstrate that the robot approaches its goal configuration while keeping safe distances from the human, and that the presented method achieves higher productivity than other SSM schemes.

The work described in [12] extends the approach from [13] by applying deep neural networks to reduce the computational burden. The authors train different DNN architectures to imitate the NMPC scheme, including feedforward networks using current and past human pose data and versions using a reduced-dimensional encoding of human pose with an autoencoder. DNN-based controllers outperformed NMPC in task completion time and cost function evaluation, while maintaining safety constraints.

Rubagotti et al. [14] extended this framework with a Scenario-based NMPC (S-NMPC) that constructs a scenario tree of plausible human motion futures via Higher-Order Markov Chains and enforces safety constraints across all scenario branches simultaneously. The method was validated experimentally on a Kinova Gen3 robot in a pick-and-place task alongside an actual human operator, demonstrating superior performance over both a single-scenario NMPC and a fixed-path continuous SSM baseline.

Oleinikov et al. [11] proposed an ellipsoidal terminal constraint extension of the SA-NMPC that enlarges the domain of attraction compared to the original point terminal constraint. Theoretical results on recursive feasibility and closed-loop stability are proved for both cases, and the effectiveness is verified with experiments on a manipulator.

Another method is adopted by Merckaert et al. [9] who present a computationally efficient control scheme of safe human-robot coexistence using the Explicit Reference Governor (ERG) formalism to impose state and input constraints in real time without solving a full OCP at each time step. The method has been tested on the Franka Emika Panda at 1 kHz and ensures that the robot is able to navigate to the required end-effector position without violating speed constraints or colliding with stationary objects or people. Another extension [10] integrates the ERG with an RRT path planner to support more complex environments, and has been shown to operate safely in highly dynamic human-robot coexistence conditions.

A common limitation shared by all the formulations reviewed above is that none explicitly optimize for time efficiency. The quadratic tracking or regulation cost in each case guides the robot to the goal but does not minimize the time required to get there. The robot arrives safely, but not as fast as its constraints would allow. This leaves recoverable performance on the table and is the central limitation that the present thesis addresses.

1.4 Time-optimal Motion Planning

Time optimal motion planning solves the problem of determining the fastest feasible trajectory between two configurations. Time-Optimal Path Parameterization (TOPP) is the classical minimum-time path following method, parameterising a given geometric path by a scalar variable and optimizing the time-scaling profile under dynamic constraints. Zhang et al. [21] use dynamics-based TOPP algorithm to control manipulators and show smooth, time-optimal motion on a set of practical paths. Wang et al. [17] fill a significant gap of classical second-order TOPP problem by extending it to include third-order (jerk) constraints and modeling it as a sequence of linear programmes that can be solved in real time.

A key connection between TOPP and SSM safety is offered by Fujii and Pham [7], who introduce a time-optimal control policy on TOPP that directly ensures the SSM condition of ISO/TS 15066. They show that any robot motion that is strictly faster than their policy is a collision with a human motion in a non-stationary state, and they show by simulation on a 6-DOF industrial robot that their approach is strictly less conservative than state-of-the-art safe control approaches. This is the nearest available work to the aim of the current thesis: to integrate time-optimality and ISO-conformant SSM safety. The most notable difference is that Fujii and Pham consider a fixed geometric path and act in an open-loop path-parameterisation framework, whilst the current thesis uses a receding-horizon NMPC that re-plans the path and implements SSM constraints online as the human moves.

In the MPC literature, Verschueren et al. [15] directly deal with time-optimality by providing an NMPC scheme that exponentially increases the cost of each stage to steer the robot to a goal position in the fewest prediction steps possible. The exponential weighting instead of minimising physical time horizon gives a strong incentive to arrive early: the solver is penalised by the weight of position errors, which increases exponentially with the power of gamma, so the solver has an incentive to bring the system to rest in as few steps as possible. The stability of this scheme is set without time scaling, which is practically appealing to embedded implementation.

This expression is embraced as the time-optimal cost structure of the current thesis.

Time-optimal point-to-point motion planning and control under stochastic uncertainty can be formulated with a robustified two-stage optimal control problem, as it was presented in [20]. Time-optimal trajectories lie close to state and input constraints and become unreliable when disturbances are present. The authors implement a two-stage optimal control problem to tackle this issue: it includes uncertainty propagation and constraint satisfaction along with preserving time-optimal behavior. In the first stage, a nominal trajectory with stabilizing feedback gains is computed by solving a fixed-time-grid optimal problem. Uncertainty - modeled as Gaussian process noise - is propagated along the trajectory using a linearized model. The resulting ellipsoidal uncertainty tubes were used to derive smooth chance-constraint safety margins that tighten the original state and input constraints. Consequently, uncertainty growth is reduced and the nominal path stays feasible under disturbances. The horizon length varies through time-scaling in the second stage, where the optimal control problem is solved to minimize total traversal time using the safety margins from the first stage computations. The outcome is a time-optimal trajectory that is strengthened to be feasible in the presence of limited disturbances. This article inspires further generalization of the current thesis to unpredictable human movement.

Chapter 2

System Modeling

2.1 UR5 Robot Model

2.1.1 Overview

The robotic platform used in this thesis is the Universal Robots UR5, a 6-degree-of-freedom (DOF) serial-link manipulator widely used in collaborative robotics research. The UR5 has a maximum reach of 850 mm and a payload capacity of 5 kg. Its six revolute joints are denoted

$$q = [q_1, q_2, q_3, q_4, q_5, q_6]^T \in \mathbb{R}^6, \quad (2.1)$$

with joint velocities $\dot{q} \in \mathbb{R}^6$ and joint accelerations $\ddot{q} \in \mathbb{R}^6$.

2.1.2 Forward Kinematics

The forward kinematics of the UR5 are implemented as a sequence of homogeneous transformation matrices following the modified Denavit-Hartenberg (DH) convention. The transformation from the robot base frame to each successive link frame is com-

puted as a chain of matrix products:

$$T_1 = T_{B1} T_{12}, \quad T_2 = T_1 T_{23}, \quad T_3 = T_2 T_{34}, \quad T_4 = T_3 T_{45}, \quad T_5 = T_4 T_{56}, \quad T_6 = T_5 T_{6,\text{tool}}, \quad (2.2)$$

where each elementary transformation $T_{i-1,i}(q_i)$ is a function of the corresponding joint angle q_i and the fixed DH parameters of that joint.

Beyond the standard joint frames, seven bounding sphere center positions are defined along the robot body by applying fixed offset transformations $T_{P_j,\text{shift}}$ to the relevant link frames:

$$c_1(q) = (T_{B1} T_{P_1,\text{shift}})_{1:3,4}, \quad c_2(q) = (T_1 T_{P_2,\text{shift}})_{1:3,4}, \quad (2.3)$$

$$c_3(q) = (T_2 T_{P_3,\text{shift}})_{1:3,4}, \quad c_4(q) = (T_2 T_{P_4,\text{shift}})_{1:3,4}, \quad (2.4)$$

$$c_5(q) = (T_2 T_{P_5,\text{shift}})_{1:3,4}, \quad c_6(q) = (T_3 T_{P_6,\text{shift}})_{1:3,4}, \quad (2.5)$$

$$c_7(q) = (T_5 T_{P_7,\text{shift}})_{1:3,4}, \quad (2.6)$$

where the notation $(\cdot)_{1:3,4}$ extracts the translational component (fourth column, first three rows) of the homogeneous transform. The offset values for each shift transformation are listed in Table 2.1. These offsets place the sphere centers at representative points along each link segment, ensuring that the robot geometry is adequately covered by the bounding sphere model.

Table 2.1: Robot bounding sphere offset translations.

Sphere	Applied to frame	Offset axis	Offset value (m)
c_1	T_{B1}	y	0.0679
c_2	T_1	z	0.2125
c_3	T_2	y	0.0679
c_4	T_2	z	0.1294
c_5	T_2	z	0.2589
c_6	T_3	z	0.0465
c_7	T_5	y	0.0411

All forward kinematics expressions are implemented symbolically in CasADi [1], enabling automatic differentiation of sphere center positions with respect to joint con-

figurations. This is required by the acados SQP solver [16] for computing constraint Jacobians at each NMPC iteration.

2.1.3 Cartesian Velocities of Sphere Centers

The Cartesian velocity of each robot bounding sphere center $c_j(q)$ under joint velocity $\dot{q}$ is:

$$\dot{c}_j = J_{c_j}(q) \dot{q}, \quad J_{c_j}(q) = \frac{\partial c_j(q)}{\partial q} \in \mathbb{R}^{3 \times 6}. \quad (2.7)$$

The squared Cartesian speed of sphere j is then:

$$v_j^2(q, \dot{q}) = \|\dot{c}_j\|^2 = \dot{c}_{j,x}^2 + \dot{c}_{j,y}^2 + \dot{c}_{j,z}^2. \quad (2.8)$$

This quantity is computed for all seven sphere centers using the CasADi `jtimes` function, which evaluates the Jacobian-vector product $J_{c_j}(q)\dot{q}$ without forming the full Jacobian matrix explicitly. The resulting squared speed vector

$$\mathbf{v}^2(q, \dot{q}) = [v_1^2, v_2^2, v_3^2, v_4^2, v_5^2, v_6^2, v_7^2]^\top \in \mathbb{R}^7 \quad (2.9)$$

is used directly in the human safety constraint formulation (Section 2.4).

2.1.4 Prediction Model

The NMPC prediction model is a second-order kinematic integrator. The state vector is:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^{12}. \quad (2.10)$$

The two formulations differ in their control input definition and discretization strategy, as described below.

Discrete-time Integrator

In the time-optimal NMPC (TO-NMPC), the control input is augmented with twelve L1 slack variables:

$$u = \begin{bmatrix} \ddot{q} \\ s \end{bmatrix} \in \mathbb{R}^{18}, \quad s = [s_1, \dots, s_{12}]^\top \geq 0. \quad (2.11)$$

The discrete-time dynamics are formulated directly as a second-order integrator with sampling time $T_s = 0.25$ s:

$$q_{k+1} = q_k + T_s \dot{q}_k + \frac{T_s^2}{2} \ddot{q}_k, \quad (2.12)$$

$$\dot{q}_{k+1} = \dot{q}_k + T_s \ddot{q}_k. \quad (2.13)$$

The horizon is $N = 10$ steps, giving a total horizon of $T_f = 2.5$ s. The coarser time grid is a deliberate design choice consistent with the exponential weighting formulation of Verschueren et al. [15]: because the exponential weights grow as γ^k with $\gamma = 3.0$, the solver is strongly incentivized to reach the goal within as few steps as possible, making the exact sampling time less critical for time-optimality than in a tracking formulation.

2.1.5 Robot Bounding Sphere Radii

The seven bounding spheres enclosing the UR5 links are assigned the radii listed in Table 2.2, chosen to conservatively overapproximate the physical cross-section of each link segment.

2.2 Human Body Representation

2.2.1 Bounding Sphere Model

The human operator is modeled as a set of $N_h = 14$ bounding spheres enclosing the upper part of the body. This representation follows the approach of Oleinikov et

Table 2.2: Robot bounding sphere radii.

Sphere	Associated link	Radius R_j^r (m)
c_1	Shoulder	0.15
c_2	Upper arm	0.15
c_3	Upper arm (mid)	0.15
c_4	Upper arm (lower)	0.08
c_5	Upper arm (end)	0.08
c_6	Forearm	0.12
c_7	Wrist	0.10

al. [13] and is standard in the safety-aware NMPC literature for pHRI. Each sphere $i \in \{1, \dots, 14\}$ is characterized by four parameters:

$$p_i^h = \begin{bmatrix} x_i \\ y_i \\ z_i \\ r_i \end{bmatrix} \in \mathbb{R}^4, \quad (2.14)$$

where $(x_i, y_i, z_i)^\top$ is the sphere center position expressed in the robot base frame, and $r_i > 0$ is the sphere radius. The full human state at a given time step is assembled into a 4×14 parameter matrix:

$$P^h = \begin{bmatrix} p_1^h & p_2^h & \dots & p_{14}^h \end{bmatrix} \in \mathbb{R}^{4 \times 14}. \quad (2.15)$$

This matrix is passed to the SA-NMPC as a runtime parameter at each solve step, together with the 3D Cartesian goal position, yielding a total parameter vector of dimension $4 \times 14 + 3 = 59$.

The 14 spheres correspond to the major segments of the upper human body. The sphere radii vary by body segment, with larger radii assigned to the torso and shoulders and smaller radii to the forearms and wrists, as listed in Table 2.3.

Table 2.3: Human bounding sphere radii (representative values).

Sphere index	Body segment	Radius r_i^h (m)
1, 2	Torso (upper, mid)	0.25
3-8	Shoulders, upper arms	0.15
9, 10	Forearms	0.10
11, 12	Wrists	0.13
13	Head / neck	0.20
14	Torso (lower)	0.25

2.2.2 Motion-capture Data Pipeline

Human motion is provided to the NMPC controller via prerecorded motion-capture data. The data is stored in CSV format, with each row corresponding to one time step and containing the (x, y, z) coordinates of each of the 14 sphere centers expressed in the robot base frame. Sphere radii are fixed across all time steps and are embedded in the solver parameter vector.

At runtime, the `human_motion_player` ROS 2 node reads the CSV file, indexes it by simulation time step, and publishes the current 4×14 human sphere matrix as a ROS 2 topic at a rate synchronized with the NMPC control loop. The high-level solver subscribes to this topic and uses the published sphere positions as runtime parameters when evaluating the safety constraints at each solve step.

2.3 Simulation Environment and Control Architecture

2.3.1 PyBullet Simulation

The simulation environment is implemented using PyBullet, an open-source rigid-body physics simulator, wrapped in a custom ROS 2 node (`pb_ur5_sim`). Two simulation modes are available: `my_node` for robot-only experiments and `my_node_with_human` for experiments including the human sphere model. The simulated UR5 is loaded from its URDF description stored in the `ur5_urdf` package. The PyBullet node subscribes

to joint reference commands published by the low-level solver and applies them to the simulated robot at each simulation timestep, publishing back the resulting joint states as feedback to the control system.

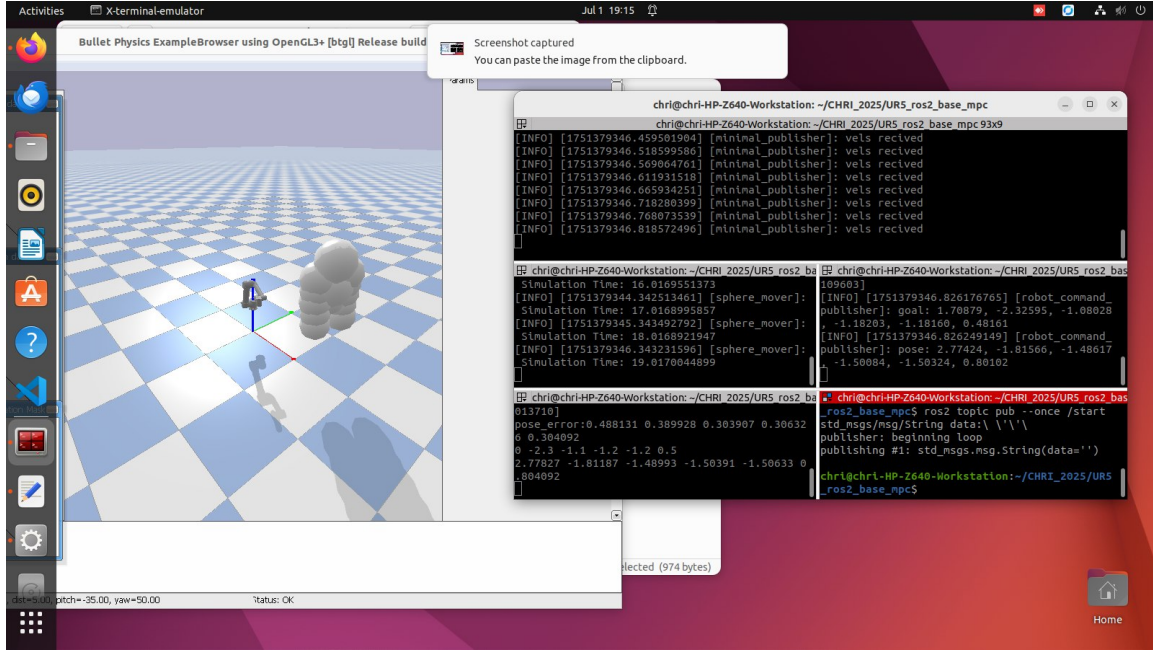


Figure 2-1: Simulation environment.

2.3.2 Two-level Control Architecture

The high-level NMPC runs at the NMPC sampling rate and solves the OCP at each step to compute a reference joint trajectory over the prediction horizon. The first element of the optimal state sequence is extracted and published as the joint reference for the current step. The low-level solver runs at a higher frequency and tracks the joint reference provided by the high-level NMPC, compensating for tracking errors between the planned and actual joint states and providing smooth joint commands to the PyBullet simulation. This separation of timescales is consistent with the approach of Oleinikov et al. [13].

2.4 Human Safety Constraint

The safety constraint in the NMPC enforces a distance-proportional speed limit inspired by the Speed and Separation Monitoring (SSM) concept of ISO/TS 15066 [8]. For each combination of robot sphere $j \in \{2, \dots, 7\}$ and human sphere $i \in \{1, \dots, 14\}$, the squared Cartesian speed of robot sphere j must not exceed a fraction α^2 of the squared surface-to-surface separation between the two spheres:

$$v_j^2(q_k, \dot{q}_k) \leq \alpha^2 \cdot d_{ij}^2(q_k), \quad \forall i \in \{1, \dots, 14\}, j \in \{2, \dots, 7\}, \quad (2.16)$$

where the squared surface-to-surface distance is:

$$d_{ij}^2(q_k) = \|c_j(q_k) - p_i^h(1:3)\|^2 - (R_j^r + r_i^h)^2, \quad (2.17)$$

and $\alpha = 0.75$ is a tuning parameter. This constraint has a clear physical interpretation: when the robot sphere is far from the human sphere, d_{ij} is large and the speed limit is permissive; as the separation decreases, the allowable speed decreases quadratically; when the bounding geometries are in contact ($d_{ij} = 0$), the constraint forces the robot speed to zero. This generates $6 \times 14 = 84$ nonlinear inequality constraints per prediction step, for a total of $84 \times 20 = 1,680$ constraints across the horizon.

These constraints are implemented as soft constraints in acados using slack variables $\xi_{k,ij} \geq 0$, relaxing (2.16) to:

$$v_j^2(q_k, \dot{q}_k) \leq \alpha^2 \cdot d_{ij}^2(q_k) + \xi_{k,ij}, \quad (2.18)$$

with a quadratic slack penalty $Z_u = 10,000$ and zero linear penalty. This large penalty ensures that violations occur only when the OCP would otherwise be infeasible, while keeping violations small when they do occur.

In addition to the human sphere velocity constraints, the NMPC enforces three further geometric constraints:

Table avoidance. The vertical positions of robot sphere centers c_3 , c_6 , and c_7 must remain above the workspace table surface:

$$c_{j,z}(q_k) \geq R_j^r, \quad j \in \{3, 6\}; \quad c_{7,z}(q_k) \geq 0.05 + R_7^r. \quad (2.19)$$

Stick constraint. A smooth exponential repulsive potential steers the end-effector away from the human torso center $\bar{p}^h = \frac{1}{3}(p_1^h + p_2^h + p_{14}^h)$:

$$h_{\text{stick}}(x_k, P_k^h) = \exp\left(-\gamma_s \cdot \frac{\|c_7(q_k) - \bar{p}^h\|^2 - r_{\text{off}}^2}{\|c_7(q_k) - p_f\|^2}\right) \leq 0, \quad (2.20)$$

where $p_f \in \mathbb{R}^3$ is the Cartesian goal position, $\gamma_s = 3.0$, and $r_{\text{off}}^2 = 0.50 \text{ m}^2$. This function is negative when the end-effector is far from the human or close to the goal, and becomes positive - violating the constraint - when the end-effector approaches the human torso.

Self-collision avoidance. The wrist sphere c_7 must not collide with the shoulder sphere c_1 :

$$\|c_7(q_k) - c_1(q_k)\|^2 \geq (R_1^r + R_7^r)^2. \quad (2.21)$$

Chapter 3

Problem Formulation

3.1 Time-optimal NMPC with Human Safety Constraints

3.1.1 Motivation

The central claim of this thesis is that the exponential weighting objective retains its time-optimality property after the addition of human safety constraints - that is, the robot still reaches the goal configuration in the minimum number of steps permitted by the current human proximity, rather than following a conservatively slow trajectory.

3.1.2 OCP Formulation

The TO-NMPC extends OCP by adding the human safety constraints, table avoidance, and stick constraint, while keeping the exponential weighting cost and the discrete-time prediction model unchanged. The resulting OCP at time step t_k is:

$$\begin{aligned}
& \underset{\ddot{q}_0, \dots, \ddot{q}_{N-1}, s_0, \dots, s_{N-1}}{\text{minimize}} & \sum_{k=0}^{N-1} \gamma^k \sum_{i=1}^{12} s_{k,i} & (3.1a) \\
& \text{subject to} & q_{k+1} = q_k + T_s \dot{q}_k + \frac{T_s^2}{2} \ddot{q}_k & , (3.1b) \\
& & \dot{q}_{k+1} = \dot{q}_k + T_s \ddot{q}_k, \quad k = 0, \dots, N-1 & , (3.1c) \\
& & x_0 = x(t_k) & , (3.1d) \\
& & \ddot{q}_{\min} \leq \ddot{q}_k \leq \ddot{q}_{\max} & , (3.1e) \\
& & q_{\min} \leq q_k \leq q_{\max}, \quad \dot{q}_{\min} \leq \dot{q}_k \leq \dot{q}_{\max} & , (3.1f) \\
& & q_{k,i} - q_{f,i} \leq s_{k,i}, \quad -(q_{k,i} - q_{f,i}) \leq s_{k,i}, \quad s_{k,i} \geq 0, & (3.1g) \\
& & v_j^2(q_k, \dot{q}_k) \leq \alpha^2 d_{ij}^2(q_k, p_i^h(t_k)) + \xi_{k,ij}, \quad \forall i, j & , (3.1h) \\
& & h_{\text{table}}(x_k) \leq 0 & , (3.1i) \\
& & h_{\text{stick}}(x_k, P_k^h) \leq 0 & , (3.1j) \\
& & h_{\text{self}}(x_k) \leq 0 & , (3.1k) \\
& & \xi_{k,ij} \geq 0, \quad \frac{1}{2} \xi_{k,ij}^2 Z_u \leq \epsilon & , (3.1l) \\
& & q_N = q_f & (3.1m)
\end{aligned}$$

The runtime parameter vector of the TO-NMPC combines the parameters of both constituent formulations:

$$p_k = [\gamma^k, q_f^\top, \text{vec}(P_k^h)^\top, p_{f,\text{cart}}^\top]^\top \in \mathbb{R}^{1+6+56+3} = \mathbb{R}^{66}, \quad (3.2)$$

where $\text{vec}(P_k^h) \in \mathbb{R}^{56}$ stacks the (x, y, z) coordinates of all 14 human sphere centers (radii are fixed), and $p_{f,\text{cart}} \in \mathbb{R}^3$ is the Cartesian goal position used in the stick constraint.

3.1.3 Interaction Between Time-optimality and Safety

The exponential weighting cost and the human safety constraint have opposing effects on the robot's velocity: the cost pushes the robot to move as fast as possible to

minimize the weighted sum of position errors, while the safety constraint imposes an upper bound on speed that decreases as the human approaches. The balance between these two forces produces the following qualitative behavior, which distinguishes the TO-NMPC from both its constituent formulations:

- **Human far from robot path:** Safety constraints are inactive. The robot moves at maximum allowable joint velocity, reaching the goal in minimum time. Behavior is identical to the TO-NMPC without human.
- **Human approaching the robot:** Safety constraints become active as d_{ij} decreases. The exponential cost still pushes for speed, but the safety constraint limits it. The robot decelerates proportionally to proximity, consistent with the SSM spirit of ISO/TS 15066 [8].
- **Human blocking the robot path:** If the human occupies a position on or near the planned trajectory, the safety constraints may force the robot to stop or significantly deviate. The soft constraint formulation ensures the solver remains feasible, penalizing any violation with $Z_u = 10,000$ rather than declaring infeasibility.
- **Human moving away:** As d_{ij} increases, the safety constraint relaxes and the exponential cost drives the robot back toward maximum speed. Recovery is immediate in the next NMPC solve, unlike conservative fixed-speed strategies that maintain a slow speed regardless of human distance.

Chapter 4

Implementation

4.1 Overview

This chapter describes how the theoretical formulations presented in Chapter 3 are realized in software and connected into a working experimental system. The implementation spans three layers: the *solver layer*, where the NMPC problems are formulated symbolically and compiled into real-time solvers using CasADi [1] and acados [16]; the *middleware layer*, where ROS 2 nodes handle communication between components; and the *simulation layer*, where PyBullet provides a physics-based environment for validating the controllers before deployment on the real robot.

The overall system is organized as a ROS 2 workspace containing four packages - `pb_ur5_sim`, `sim_utils`, `mpc_low`, and `mpc_high/mpc_to` - each responsible for a distinct functional role. Three acados solver configurations are implemented, generated from their respective Jupyter notebooks:

- `disc_high_mpc_accel.ipynb` - Quadratic-cost NMPC with discrete dynamics, quadratic cost, no human constraints.
- `to_disc_high_mpc_accel.ipynb` - Time-optimal NMPC with discrete dynamics, exponential L1 cost, no human constraints.

4.2 Software Stack

The full software stack is built on the following open-source frameworks:

- **Ubuntu 22.04 LTS** as the operating system.
- **ROS 2 Humble** as the robotics middleware, providing inter-process communication via topics and the `colcon` build system.
- **CasADi** [1] for symbolic formulation of all OCP expressions - forward kinematics, bounding sphere velocities, constraint functions - with automatic differentiation of first- and second-order derivatives.
- **acados** [16] for generating optimized C code implementing the SQP-RTI solver, integrator, and QP subproblem. The generated code is compiled into a shared library and accessed from Python via a Cython wrapper.
- **PyBullet** as the physics simulation environment for the UR5 robot and the human sphere model.
- **HPIPM** as the interior-point QP solver within acados, used with full condensing for all three solver configurations.

The Cython interface is used rather than the standard acados ctypes interface for its lower per-call overhead, which is important at the control frequencies involved.

4.3 NMPC Solver Generation

4.3.1 Common Elements

All three solver configurations share the same forward kinematics implementation. The UR5 kinematics are defined as a sequence of CasADi symbolic matrix multiplications corresponding to the homogeneous transformation matrices described in Section 2.1.2. The seven bounding sphere center positions $c_1(q), \dots, c_7(q)$ are extracted

as the translational components of fixed offset transforms applied to the relevant link frames, as listed in Table 2.1.

The squared Cartesian velocities of all sphere centers are computed using the CasADi `jtimes` function, which evaluates the Jacobian-vector product $J_{c_j}(q)\dot{q}$ without forming the full Jacobian matrix explicitly:

$$v_j^2(q, \dot{q}) = \| \text{jtimes}(c_j, q, \dot{q}) \|^2, \quad j = 1, \dots, 7. \quad (4.1)$$

This avoids the memory and computation cost of forming and storing the full 3×6 Jacobian matrix at each evaluation, which is beneficial given that this expression appears in up to $6 \times 14 = 84$ constraint evaluations per prediction step in the human-constrained variants.

The self-collision constraint is defined identically across all solvers:

$$h_{\text{self}}(x_k) = (R_1^r + R_7^r)^2 - \|c_7(q_k) - c_1(q_k)\|^2 \leq 0, \quad (4.2)$$

and registered as a nonlinear hard inequality constraint via `con_h_expr` in `acados`.

4.3.2 Discrete-time Prediction Model

Both the QC-NMPC and the TO-NMPC use the same discrete-time second-order integrator with sampling time $T_s = 0.25$ s. The state vector is $x = [q^\top, \dot{q}^\top]^\top \in \mathbb{R}^{12}$ and the discrete-time dynamics are defined symbolically in CasADi as:

$$q_{k+1} = q_k + T_s \dot{q}_k + \frac{T_s^2}{2} \ddot{q}_k, \quad (4.3)$$

$$\dot{q}_{k+1} = \dot{q}_k + T_s \ddot{q}_k, \quad (4.4)$$

passed directly to `acados` via `disc_dyn_expr` with `integrator_type = 'DISCRETE'`. The horizon for both solvers is $N = 10$ steps, giving a total prediction time of $T_f = 2.5$ s. This bypasses the `acados` numerical integrator entirely, giving the solver analytically exact dynamics at no approximation cost.

4.3.3 TO-NMPC Solver

The TO-NMPC solver uses the same discrete-time dynamics as the QC-NMPC but replaces the quadratic cost with an exponential L1 weighting cost, augments the control vector with slack variables, and adds a hard terminal equality constraint. The complete configuration is listed in Table 4.1.

Table 4.1: TO-NMPC solver configuration (`to_disc_high_mpc_accel`).

Parameter	Value
State dimension n_x	12
Control dimension n_u	12
Horizon N	10
Sampling time T_s	0.25 s
Total horizon T_f	2.5 s
Cost type	EXTERNAL
Exponential weighting γ	3.2 and 2.6
Joint velocity bounds	$[-\pi, \pi]$ rad/s
Joint acceleration bounds	$[-2.0, 2.0]$ rad/s ²
Integrator	DISCRETE
NLP solver	SQP_RTI
QP solver	FULL_CONDENSING_HPIPM
Hessian approximation	EXACT
Levenberg-Marquardt	10^{-5}
Terminal constraint	Hard: $q_N = q_f$
Human constraints	Yes
Runtime parameters	$p_k \in \mathbb{R}^7$

Augmented control vector. The control input is augmented with six L1 slack variables:

$$u_k = \begin{bmatrix} \ddot{q}_k \\ s_k \end{bmatrix} \in \mathbb{R}^{18}, \quad s_k = [s_{k,1}, \dots, s_{k,12}]^\top \geq 0. \quad (4.5)$$

The slack variables occupy positions 6-17 of the control vector and are subject to box constraints $s_{k,i} \in [0, 10^6]$.

External cost function. The stage cost is configured as `EXTERNAL`:

$$\ell_k = w_k \cdot \sum_{i=1}^{12} s_{k,i}, \quad w_k = \gamma^k, \quad (4.6)$$

expressed in the acados Python interface as:

```
ocp.model.cost_expr_ext_cost = w * sum1(s)
```

where `w = p[0]` is the first element of the runtime parameter vector and `s = sym_u[6:18]`.

Per-stage parameter update. The exponential weight $w_k = \gamma^k$ and the goal configuration q_f are passed as a per-stage runtime parameter vector $p_k = [\gamma^k, q_f^\top]^\top \in \mathbb{R}^7$, updated before each solve:

```
for k in range(N):
    ocp_solver.set(k, "p",
        np.concatenate([gamma**k, goal_for_cost]))
```

This assigns weight 1.0 at stage 0, growing to $3.0^9 \approx 19,683$ at stage 9, creating a strong incentive for early arrival at the goal.

L1 slack constraints. The L1 norm reformulation is implemented as $3 \times 12 = 36$ nonlinear inequality constraints per step. Defining $\theta_{\text{err},i} = q_{k,i} - q_{f,i}$, the constraint expression is:

$$h_{\text{L1}}(x_k, u_k) = \begin{bmatrix} \theta_{\text{err}} - s_k \\ -\theta_{\text{err}} - s_k \\ -s_k \end{bmatrix} \leq 0, \quad (4.7)$$

with upper bounds set to zero and lower bounds to -10^9 . This constraint is registered via `con_h_expr`, applied identically at all stages.

Terminal constraint. The terminal equality constraint $q_N = q_f$ is implemented via tight terminal box constraints:

```

ocp.constraints.idxbx_e = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
ocp.constraints.lbx_e = np.concatenate([xf, np.zeros(6)])
ocp.constraints.ubx_e = np.concatenate([xf, np.zeros(6)])

```

applied only to joint positions (indices 0-5). Terminal joint velocities are free within the box bounds.

Exact Hessian. An exact Hessian is used (**EXACT**) with all three exact Hessian flags enabled. This choice is motivated by the external cost structure: the Gauss-Newton approximation is well-suited to least-squares costs but may lose curvature information for the linear-in-slacks external cost. A Levenberg-Marquardt regularization of 10^{-5} ensures positive definiteness.

Chapter 5

Results and Discussion

5.1 Experimental Setup

All experiments are conducted in the PyBullet rigid-body physics simulator using the UR5 six-degree-of-freedom manipulator loaded from its URDF description. The simulation environment runs inside a custom ROS 2 node at a fixed time step synchronised with the NMPC control loop. The ROS 2 workspace contains four packages: PyBullet simulation node, human motion player, low-level joint tracker, and high-level TO-NMPC and QC-NMPC solvers.

Two experimental conditions are evaluated:

- **Human-present condition:** the simulation node is used together with the human motion player node, which reads prerecorded motion-capture data from a CSV file and publishes the 4×14 human sphere matrix to the NMPC at each control step.
- **Robot-only condition:** the simulation node is used with no human present in the workspace.

In both conditions the robot executes repeated point-to-point motions between a fixed home configuration and a sequence of goal configurations for a total experiment duration of approximately 60 seconds (240 logged steps at $T_s = 0.25$ s). Performance

is logged to CSV files and post-processed by the script for comparing the solvers, which computes per-step solve times, goal-switch times, and full-cycle times for both controllers.

The two controllers under comparison are:

- **QC-NMPC**: the baseline safety-aware NMPC with a quadratic regulation cost.
- **TO-NMPC**: the proposed time-optimal NMPC with an exponential L1 weighting cost, slack-augmented control vector, and hard terminal equality constraint.

Both controllers share identical prediction horizon ($N = 10$, $T_f = 2.5$ s), sampling time ($T_s = 0.25$ s), joint bounds, and - in the human-present condition - identical SSM safety constraints with $\alpha = 0.75$.

5.2 Solver Timing

Before examining trajectory quality, it is instructive to compare the per-step computational cost of the two solvers, since real-time feasibility is a prerequisite for deployment. Table 5.1 summarises the solve time statistics extracted from the performance logs.

Table 5.1: High-level solver solve times (ms) over 240 steps.

Condition	Controller	Mean (ms)	Median (ms)	Max (ms)
Human present	QC-NMPC	47.842	49.669	78.257
Human present	TO-NMPC	27.763	28.891	37.778
No human	QC-NMPC	14.545	14.021	49.653
No human	TO-NMPC	24.456	24.050	71.837

In the human-present condition the TO-NMPC is substantially faster than the QC-NMPC: mean solve time is 27.8 ms versus 47.8 ms, a reduction of 42%, and the maximum solve time drops from 78.3 ms to 37.8 ms. This is counterintuitive

at first glance, since the TO-NMPC adds the exponential cost, slack variables, and 1,680 additional nonlinear safety constraints. The explanation lies in the structure of the SQP-RTI solver: the exponential weighting cost creates a strong, consistent gradient toward the terminal constraint, which helps the SQP warm-start converge in fewer iterations per step. The QC-NMPC’s quadratic cost produces a flatter cost landscape near the goal, requiring more SQP iterations to certify optimality at each step, especially when the human is nearby and the safety constraints are active.

In the robot-only condition the situation reverses: the QC-NMPC solves faster (14.5 ms mean) than the TO-NMPC (24.5 ms mean). Without the 1,680 safety constraints, the QC-NMPC’s simpler quadratic cost results in a well-conditioned QP subproblem. The TO-NMPC’s external cost with exact Hessian is more expensive to evaluate and to factor, and this overhead is not offset by faster SQP convergence in the absence of the active safety constraints. Nevertheless, at 24.5 ms mean the TO-NMPC remains well within the 250 ms sampling period, confirming real-time feasibility in both conditions.

The maximum solve time of 71.8 ms for the TO-NMPC in the robot-only condition warrants attention. This outlier likely corresponds to the first solve step, where no warm-start is available and the SQP must converge from a cold initialisation. Subsequent steps benefit from the previous solution as a warm-start, which is why the median (24.0 ms) is close to the mean.

5.3 Baseline: Quadratic-Cost Safety-Aware NMPC

5.3.1 Robot-Only Condition

Figure 5-1 shows the joint-space trajectories produced by the QC-NMPC over 59.8 seconds in the absence of a human operator. The four panels show, from top to bottom: joint positions (rad), actual joint velocities (rad/s), reference joint velocities generated by the high-level NMPC (rad/s), and commanded joint accelerations (rad/s²).

The position panel reveals smooth, periodic motion between goal configurations,

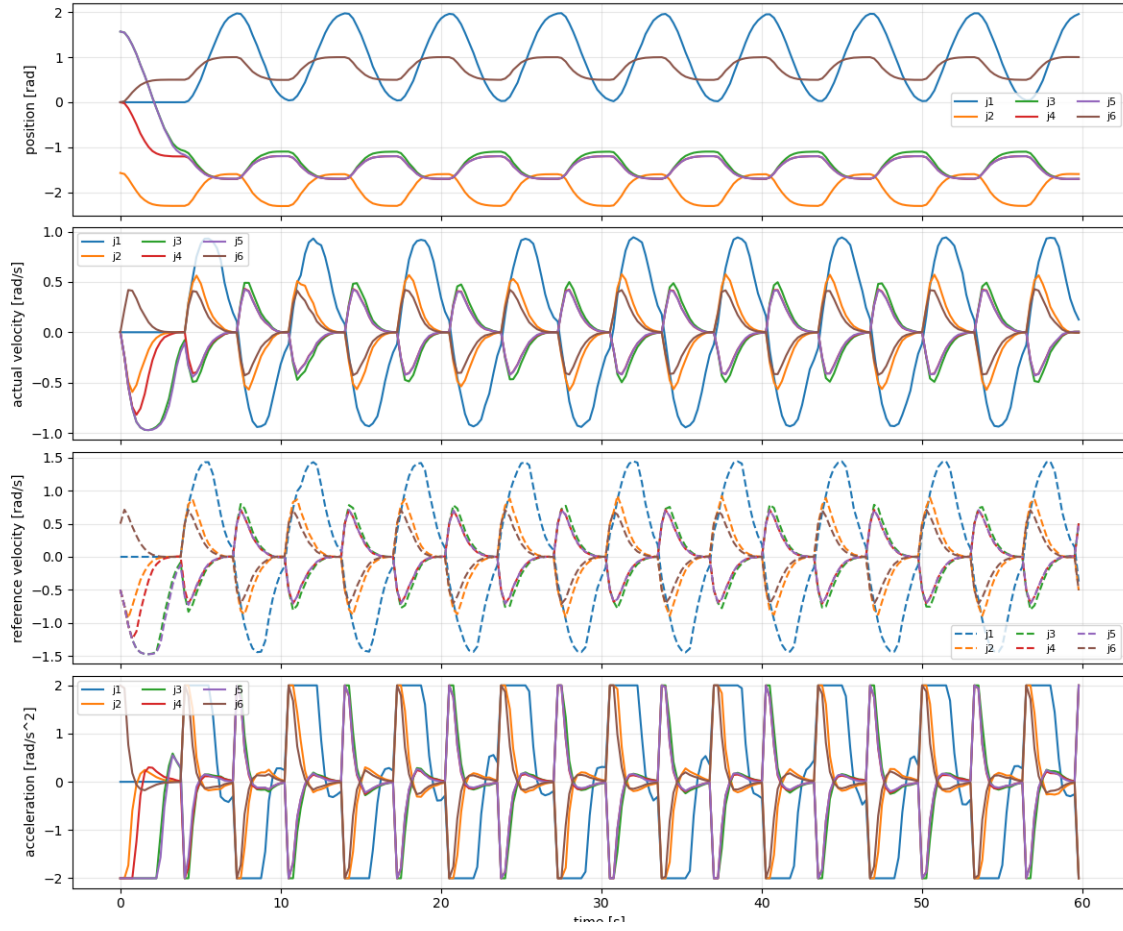


Figure 5-1: QC-NMPC joint trajectories - robot-only condition (no human present). From top: joint positions, actual joint velocities, reference joint velocities from the high-level NMPC, and joint accelerations over a 60-second run. The robot performs repeated point-to-point motions between goal configurations.

with joints j_1 and j_2 exhibiting the largest excursions (± 2 rad) and joints j_3 - j_6 remaining in a tighter range consistent with the task geometry. Actual velocities peak at approximately ± 1 rad/s for j_1 and remain below ± 0.75 rad/s for the remaining joints, well within the ± 10 rad/s bounds. The reference velocity panel shows that the high-level NMPC generates smooth references that the low-level tracker follows faithfully with no visible inter-step discontinuities. The acceleration panel exhibits bang-bang-like saturation at ± 2 rad/s² for j_1 and j_2 during approach and braking phases, while the remaining joints show moderate excursions.

Over the 59.8-second run the QC-NMPC completed **16 full cycles** with a mean full-cycle time of **6.562 s** (median 6.500 s, max 6.785 s) and **18 goal switches** with a mean switch time of **3.321 s**. The controller reaches each goal reliably and velocities are smooth. However, the quadratic cost does not drive the robot to exploit the full speed envelope: actual velocities are consistently below the kinematic limits, indicating that task completion time is not being minimised.

5.3.2 Human-Present Condition

Figure 5-2 shows the effect of introducing the moving human operator. The velocity panel shows significantly more irregular profiles than the robot-only case, with frequent transient drops to near zero as the human approaches robot spheres and the SSM constraints become active. This stop-and-go behaviour is especially pronounced for joints j_1 and j_5 . The position trajectories retain their overall periodic structure but show visible plateaus and delayed arrivals at time instants corresponding to human proximity events. The acceleration panel is considerably more erratic than in the robot-only case. Over the 59.8-second run the QC-NMPC completed **13 full cycles** with a mean full-cycle time of **7.057 s** (median 5.740 s, max 13.509 s) and **15 goal switches** with a mean switch time of **3.702 s**. Compared to the robot-only condition, the mean full-cycle time increases by 7.5%, reflecting the additional time spent slowing for the human.

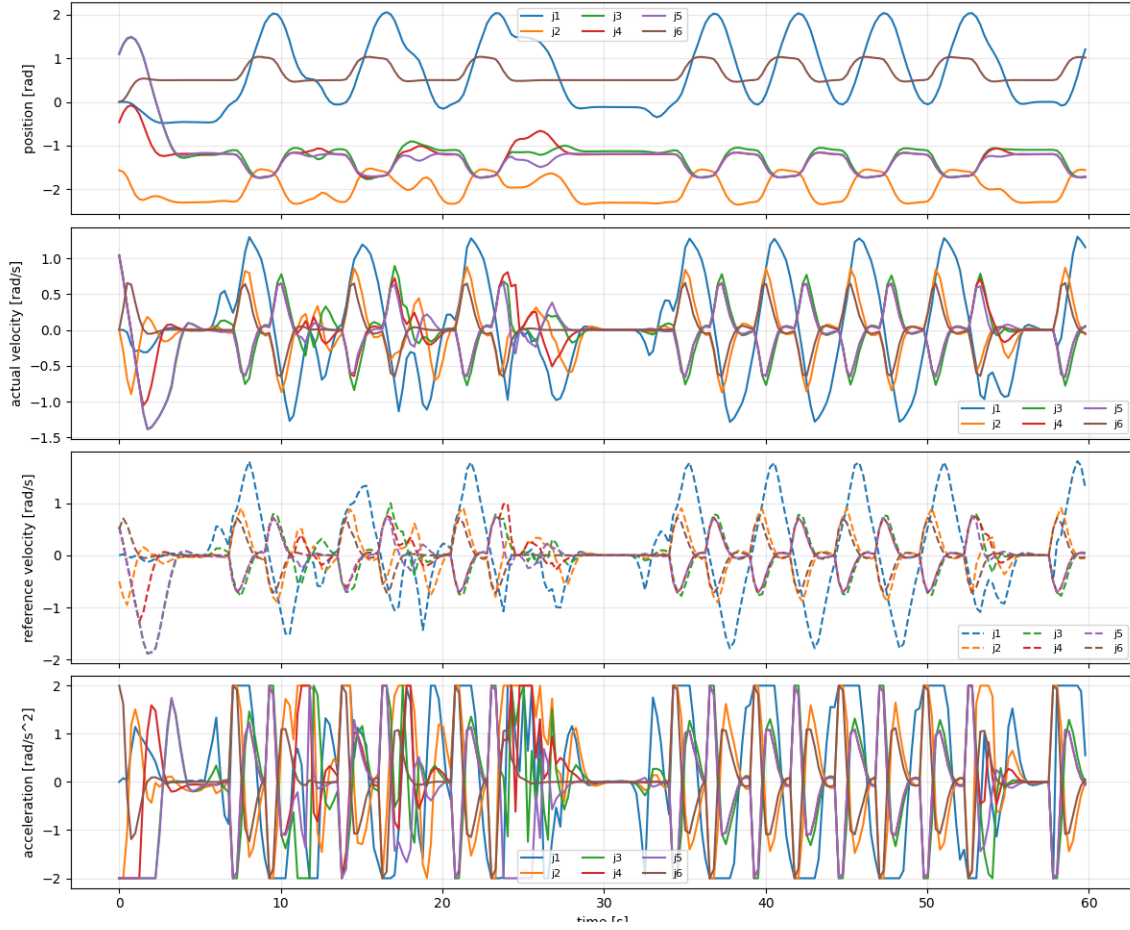


Figure 5-2: QC-NMPC joint trajectories - human-present condition. Layout identical to Figure 5-1.

The SSM constraint satisfaction is shown in Figure 5-3. For all seven bounding sphere test points, the actual speed (black) remains strictly below the SSM limit speed (red) at every time instant, confirming zero violations over the entire 240-step run. Spheres closer to the end-effector (test points 5-7) exhibit the largest speed excursions but remain within limits even during close human proximity, as indicated by the sharp reductions in the limit curve at the dashed blue event markers. The shoulder-region spheres (test points 1-2) remain near zero throughout, reflecting the tight speed cap imposed by their persistent proximity to the human torso. A key observation is that during human-distant periods, the actual speed for test points 3-7 remains well below the limit - often by 1-1.5 m/s. This gap is the signature of the quadratic cost's conservatism: the controller does not exploit the available speed

headroom even when the safety constraints are inactive.

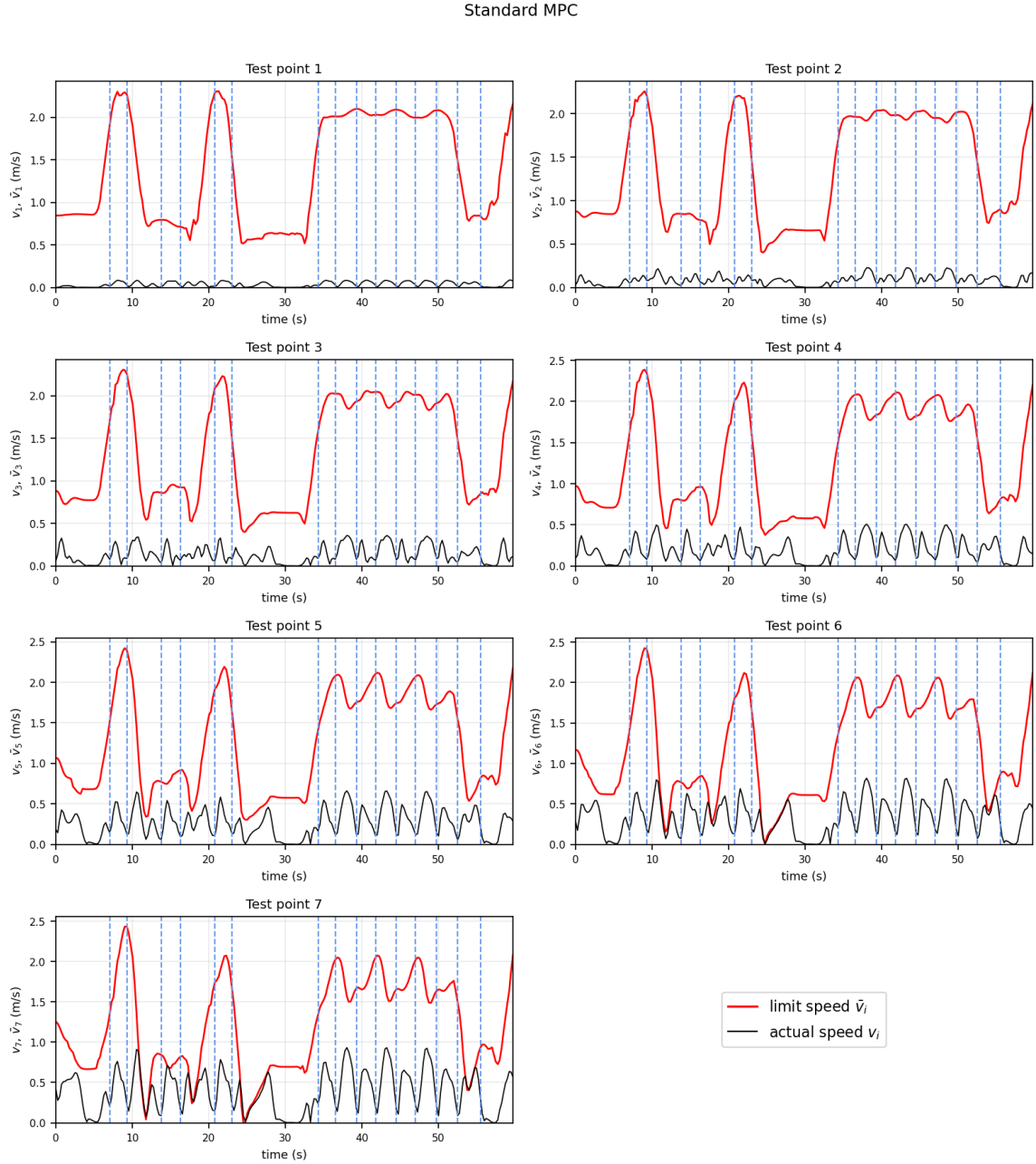


Figure 5-3: SSM constraint satisfaction for the QC-NMPC (human-present condition). Each subplot corresponds to one of the seven robot bounding sphere test points. **Red:** SSM speed limit $\tilde{v}_i$ (distance-proportional). **Black:** actual Cartesian sphere speed v_i . **Blue dashed lines:** instants of closest human approach.

5.4 Proposed: Time-Optimal Safety-Aware NMPC

5.4.1 Robot-Only Condition

Figure 5-4 shows the TO-NMPC joint trajectories in the absence of a human operator.

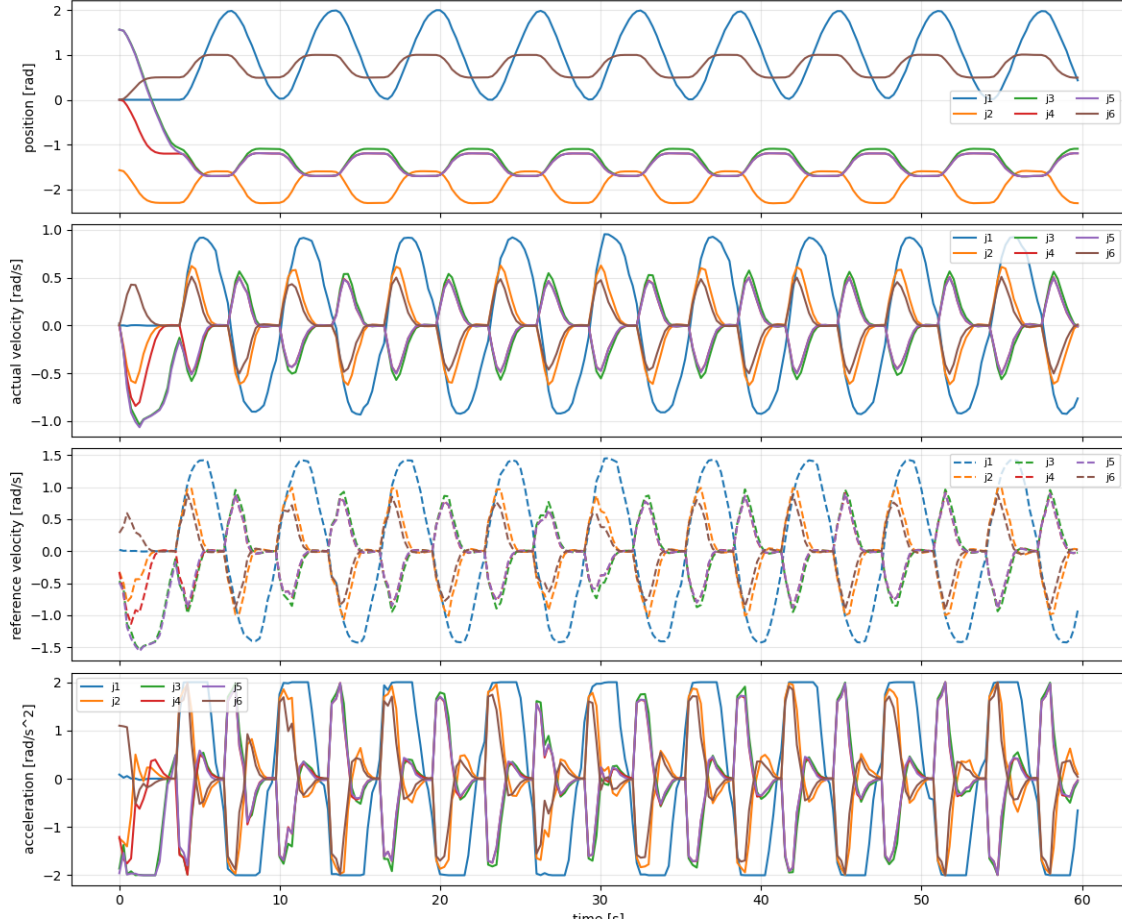


Figure 5-4: TO-NMPC joint trajectories - robot-only condition (no human present). Layout identical to Figure 5-1. $\gamma = 3.2$.

The position trajectories are qualitatively similar to the QC-NMPC case, confirming that both controllers reach the same sequence of goals. The key differences appear in the velocity and acceleration panels. Actual joint velocities are noticeably larger in magnitude: j_1 peaks approach ± 1 rad/s and j_2, j_4, j_5 reach ± 0.75 -1 rad/s, compared to ± 0.5 rad/s for the same joints under QC-NMPC. The reference velocity panel shows higher-amplitude signals that the low-level controller tracks closely. The acceleration panel is the most diagnostic: multiple joints saturate simultaneously

at $\pm 2 \text{ rad/s}^2$ for sustained intervals. This simultaneous multi-joint saturation is the classical bang-bang signature of time-optimal control.

Over the 59.7-second run the TO-NMPC completed **16 full cycles** with a mean full-cycle time of **6.325 s** (median 6.262 s, max 6.523 s) and **18 goal switches** with a mean switch time of **3.193 s**. This represents a **3.7% reduction** in mean full-cycle time relative to the QC-NMPC baseline (6.562 s), yielding a **speedup of $\times 1.037$** .

5.4.2 Human-Present Condition

Figure 5-5 shows the TO-NMPC trajectories with a moving human operator present. This is the most complex behaviour of all four conditions. When the human is distant from the robot path, actual velocities reach their largest values of any condition: j_1 peaks approach $\pm 1.5 \text{ rad/s}$ and j_2 reaches $\pm 1 \text{ rad/s}$. When the human approaches, velocities drop sharply to near zero within a single control period. Recovery is equally rapid: as the human retreats, the controller immediately resumes high-speed motion without the prolonged slow phase observed under QC-NMPC. The acceleration panel is correspondingly more aggressive than in all other conditions, with multiple joints simultaneously saturating at $\pm 2 \text{ rad/s}^2$ in rapid alternation throughout the 60-second run. Over the 59.7-second run the TO-NMPC completed **16 full cycles** with a mean full-cycle time of **6.282 s** (median 5.371 s, max 13.502 s) and **18 goal switches** with a mean switch time of **3.305 s**. This represents an **11.0% reduction** in mean full-cycle time relative to the QC-NMPC baseline (7.057 s), corresponding to a **speedup factor of $\times 1.123$** . The improvement is substantially larger than in the robot-only condition because the QC-NMPC is additionally penalised by conservative behaviour during human-distant periods, whereas the TO-NMPC exploits these periods to move at near-maximum speed.

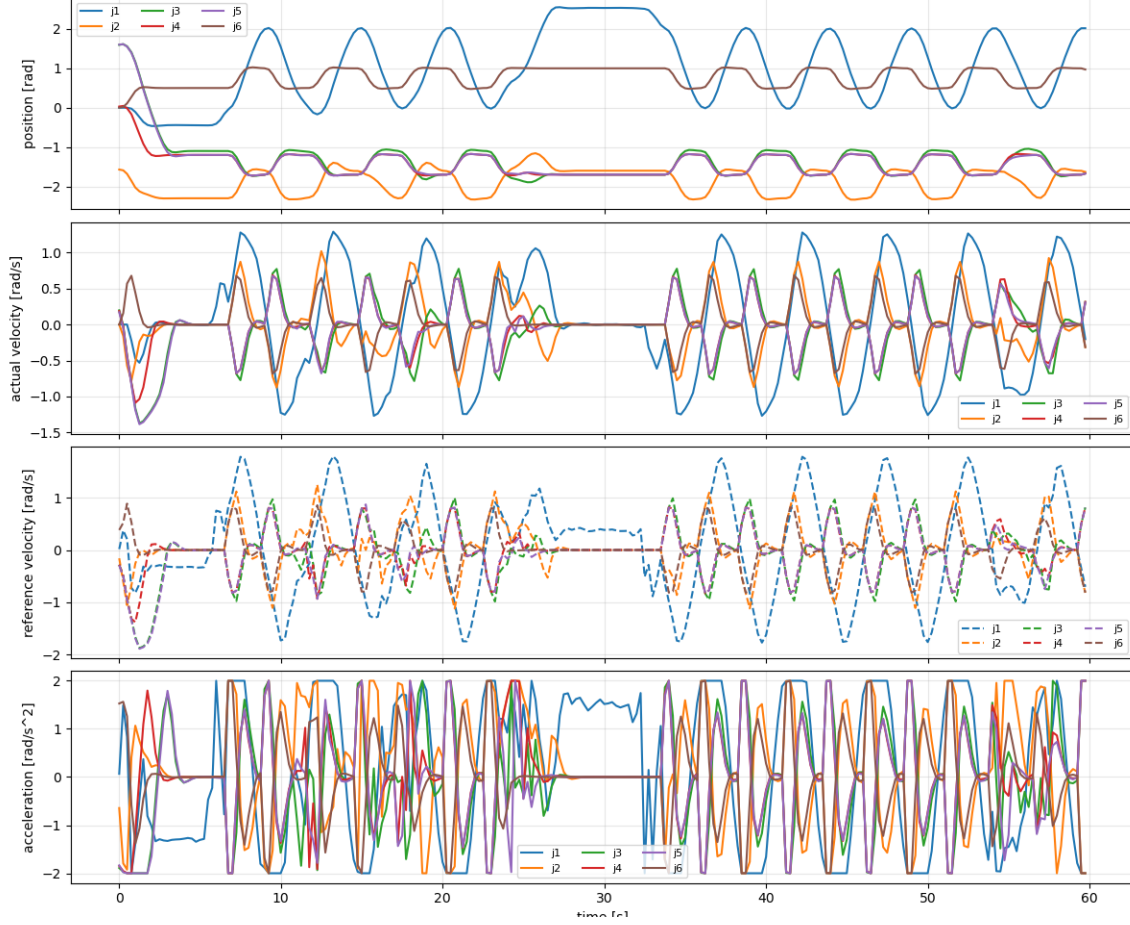


Figure 5-5: TO-NMPC joint trajectories - human-present condition. Layout identical to Figure 5-1. $\gamma = 2.6$.

The SSM constraint satisfaction is shown in Figure 5-6. For all seven bounding sphere pairs the actual speed remains below the SSM limit with no violations observed at any of the event markers.

The most important comparison between Figures 5-3 and 5-6 is the behaviour *between* the dashed blue lines, when the human is distant and the SSM limit is permissive. Under QC-NMPC the actual speed remains well below the limit. Under TO-NMPC the actual speed rises toward the limit during these periods: the black curve tracks the red curve with a consistently smaller gap. At the closest approach instants, the actual speed descends rapidly to near zero in both controllers, confirming that the safety constraint is honoured in both cases. The behaviour at test points 5-7 is particularly informative: the limit speed drops sharply and repeatedly as the

human approaches, and the actual speed tracks these reductions closely, descending to near zero when the limit approaches zero.

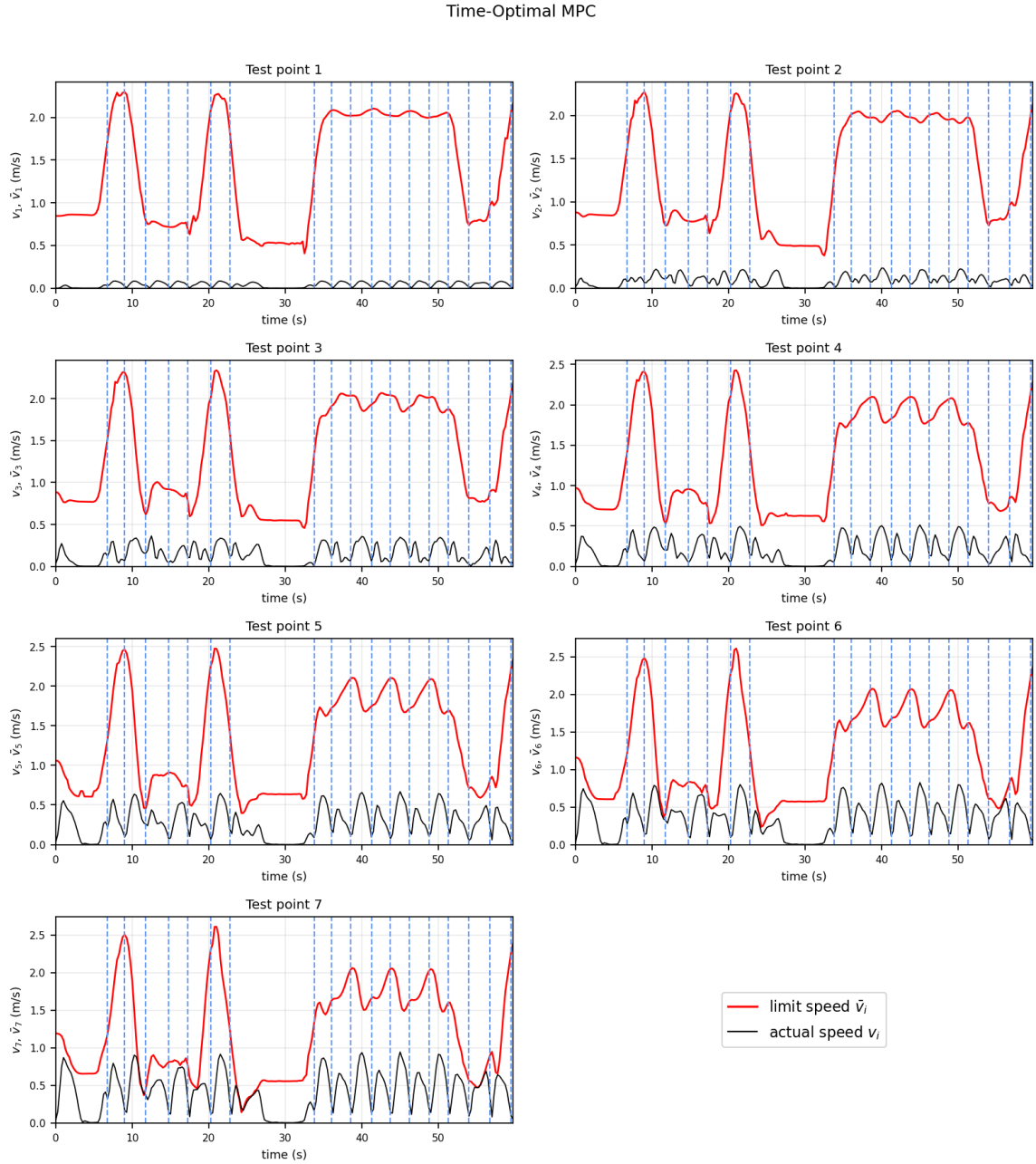


Figure 5-6: SSM constraint satisfaction for the TO-NMPC (human-present condition). Layout identical to Figure 5-3. **Red:** SSM speed limit $\tilde{v}_i$. **Black:** actual sphere speed v_i . **Blue dashed lines:** closest human approach instants.

5.5 Discussion

5.5.1 Confirmation of the Central Claim

The experimental results directly support the central claim of this thesis: the exponential weighting objective retains its time-optimality property after the addition of SSM human safety constraints. The TO-NMPC achieves a $\times 1.123$ speedup over the QC-NMPC baseline in the human-present condition while maintaining zero constraint violations. The qualitative behaviour visible in Figures 5-5 and 5-6 - operating near the speed constraint boundary when the human is far, decelerating sharply as the human approaches, and recovering immediately as the human retreats - is consistent with the previous theoretical analysis.

5.5.2 The Human-Present Speedup is Larger than the Robot-Only Speedup

The $\times 1.123$ speedup with a human present versus $\times 1.037$ without deserves careful interpretation. In the robot-only condition the QC-NMPC is already relatively efficient: without safety constraints, the quadratic cost provides a reasonable gradient toward the goal. In the human-present condition the QC-NMPC exhibits an additional form of conservatism: it slows unnecessarily during human-distant periods because its cost function provides no incentive to recover speed. Comparing Figures 5-3 and 5-6, the gap between the actual and limit speed curves is consistently larger for the QC-NMPC during these intervals. The TO-NMPC, driven by the exponential weights $w_k = 3^k$, closes this gap and recovers the lost productivity.

5.5.3 Solver Timing Interpretation

The counterintuitive result that the TO-NMPC solves faster than the QC-NMPC in the human-present condition (27.8 ms vs 47.8 ms mean) warrants further discussion. The TO-NMPC's exponential cost with $\gamma = 3.0$ produces stage weights growing from 1.0 at stage 0 to $3^9 \approx 19,683$ at stage 9. This steep gradient dominates the Hessian

structure at later stages and creates a strongly convex QP subproblem that HPIPM can solve efficiently. The QC-NMPC’s quadratic cost produces a nearly flat cost landscape near the goal, which increases the number of SQP-RTI iterations required for convergence, especially when the active-set structure changes at each step due to the 1,680 safety constraints. In the robot-only condition this effect disappears because the safety constraints are absent, and the QC-NMPC’s simpler cost structure leads to faster QP solves.

Chapter 6

Conclusion

Collaborative robots are most useful when they can move quickly in the presence of humans, but safety standards and the control methods designed to satisfy them suppress the speed rather than optimise. The standard methods use distance-proportional speed limit as a constraint and choose a cost function that merely guides the robot to the goal, and neglect the issue of time spent to complete the task.

In this thesis, the cost function is designed to reach the goal as fast as the safety constraints allow by integrating the exponential L1 weighting cost, originally proposed for the time-optimal NMPC setting by Verschueren et al.[15] to the SSM-constrained pHRI framework of Oleinikov et al. The solver eliminates position error as early as possible by assigning stage weights that grow as γ^k over the prediction horizon. The system pushes the robot toward its kinematic boundary when the human is far away. When the human gets closer and the SSM limit tightens, the same cost accepts the enforced deceleration and then immediately re-accelerates as soon as the constraint relaxes.

In a 60-second simulation with a moving human operator, the TO-NMPC completed 16 full task cycles compared to 13 for the QC-NMPC baseline and recorded zero SSM violations across all steps. The mean full-cycle time decreased from 7.057 s to 6.282 s, a speedup of $\times 1.123$. In the robot-only condition the speedup was smaller 1.037. The exponential cost provides its greatest advantage specifically when a human is present. The TO-NMPC has complex formulation with external cost, slack-

augmented control vector, additional safety constraint evaluations per horizon, exact Hessian and solved 42% faster than the QC-NMPC in the human-present condition (mean 27.8 ms versus 47.8 ms) and with substantially lower variance. The explanation is structural: the steep exponential cost gradient creates a strongly convex QP subproblem that the HPIPM solver resolves in fewer SQP-RTI iterations, even when the safety constraints are actively changing the problem geometry at each step. The QC-NMPC’s flat quadratic cost landscape, by contrast, requires more iterations to certify optimality when 1,680 constraints are active. This finding suggests that cost-function design is not merely a matter of shaping the closed-loop trajectory - it also shapes the difficulty of the online optimisation problem, and a well-chosen cost can simultaneously improve both task performance and computational efficiency.

The choice of cost function in the safety-aware robot controllers is not a secondary tuning decision made after the safety constraints have been set. It is a primary design variable that determines how productively the robot moves and how fast the controller solves its own optimisation problem. Safety and efficiency are not opposing objectives. With the right cost structure, they are compatible.

Appendix A

Tables

Appendix B

Figures

Bibliography

- [1] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2018.
- [2] Alejandro Astudillo, Joris Gillis, Moritz Diehl, Wilm Decré, Goele Pipeleers, and Jan Swevers. Position and orientation tunnel-following nmpe of robot manipulators based on symbolic linearization in sequential convex quadratic programming. *IEEE Robotics and Automation Letters*, 7(2):2867–2874, 2022.
- [3] Mohammad Farajtabar and Marie Charbonneau. The path towards contact-based physical human-robot interaction, 2024.
- [4] Marco Faroni, Manuel Beschi, Corrado Guarino Lo Bianco, and Antonio Visioli. Predictive joint trajectory scaling for manipulators with kinodynamic constraints. *Control Engineering Practice*, 95:104264, 2020.
- [5] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control, 2020.
- [6] Gianluca Frison, Dimitris Kouzoupis, Tommaso Sartor, Andrea Zanelli, and Moritz Diehl. Blasfeo. *ACM Transactions on Mathematical Software*, 44:1–30, 2018.
- [7] Shohei Fujii and Quang-Cuong Pham. Time-optimal path tracking with iso safety guarantees. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5926–5933, 2023.
- [8] International Organization for Standardization. Iso/ts 15066: Robots and robotic devices - collaborative robots. Technical report, ISO, 2016.
- [9] Kelly Merckaert, Bryan Convens, Chi ju Wu, Alessandro Roncone, Marco M. Nicotra, and Bram Vanderborght. Real-time motion control of robotic manipulators for safe human-robot coexistence. *Robotics and Computer-Integrated Manufacturing*, 73:102223, 2022.
- [10] Kelly Merckaert, Bryan Convens, Marco M. Nicotra, and Bram Vanderborght. Real-time constraint-based planning and control of robotic manipulators for safe human-robot collaboration. *Robotics and Computer-Integrated Manufacturing*, 87:102711, 2024.

- [11] Aigerim Nurbayeva and Matteo Rubagotti. Nonlinear model predictive control with set terminal constraint for safe robot motion planning via speed and separation monitoring. *Control Engineering Practice*, 154:106155, 2025.
- [12] Aigerim Nurbayeva, Almas Shintemirov, and Matteo Rubagotti. Deep imitation learning of nonlinear model predictive control laws for a safe physical human–robot interaction. *IEEE Transactions on Industrial Informatics*, 19(7):8384–8395, 2023.
- [13] Artemiy Oleinikov, Sanzhar Kusdavletov, Almas Shintemirov, and Matteo Rubagotti. Safety-aware nonlinear model predictive control for physical human–robot interaction. *IEEE Robotics and Automation Letters*, 6(3):5665–5672, 2021.
- [14] Artemiy Oleinikov, Sergey Soltan, Zarema Balgabekova, Alberto Bemporad, and Matteo Rubagotti. Scenario-based model predictive control with probabilistic human predictions for human–robot coexistence. *Control Engineering Practice*, 142:105769, 2024.
- [15] Robin Verschueren, Hans Joachim Ferreau, Alessandro Zananini, Mehmet Mercangöz, and Moritz Diehl. A stabilizing nonlinear model predictive control scheme for time-optimal point-to-point motions. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2525–2530, 2017.
- [16] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 2021.
- [17] Ruobing Wang, Yanlin Xie, Xigang Chen, and Yangmin Li. Path-constrained time-optimal motion planning for robot manipulators with third-order constraints. *IEEE/ASME Transactions on Mechatronics*, 28(6):3005–3016, 2023.
- [18] Tuğçe Yaren and Selçuk Kizir. Real-time nonlinear model predictive control of a robotic arm using spatial operator algebra theory. *Journal of Field Robotics*, 42:2337–2354, 2025.
- [19] Andrea Maria Zanchettin, Nicola Maria Ceriani, Paolo Rocco, Hao Ding, and Björn Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2):882–893, 2016.
- [20] Shuhao Zhang and Jan Swevers. Robustified time-optimal point-to-point motion planning and control under uncertainty, 2025.
- [21] Tie Zhang, Meihui Zhang, and Yanbiao Zou. Time-optimal and smooth trajectory planning for robot manipulators. *International Journal of Control, Automation and Systems*, 19:521–531, 2020.

- [22] Pu Zheng, Pierre-Brice Wieber, Junaid Baber, and Olivier Aycard. Human arm motion prediction for collision avoidance in a shared workspace. *Sensors*, 22:6951, 2022.