Topics in Trajectory Generation for Robots

Ghazaei, Mahdi

2015

Document Version:
Publisher's PDF, also known as Version of record

Topics in Trajectory Generation for Robots

M. Mahdi Ghazaei Ardakani

LUND
UNIVERSITY

Department of Automatic Control

Lic. Tech. Thesis
ISRN LUTFD2/TFRT--3265--SE
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2015

Abstract

A fundamental problem in robotics is generating the motion for a task. How to translate a task to motion or a series of movements is a non-trivial problem. The complexity of the task, the structure of the robot, and the desired performance determine the sequence of movements, the path, and the course of motion as a function of time, namely the *trajectory*. As we discuss in this thesis, a trajectory can be acquired from a human demonstration or generated by carefully designing an objective function. In the first approach, we examine a number of robotic setups that are suitable for human demonstration. More notably, admittance control as a new dimension to the robot-assisted teleoperation is investigated. We also describe a free-floating behavior which makes robust lead-through programming possible, especially in scenarios where a robot comes into contact with stiff materials. As a way to utilize these setups, we present some ideas for developing a high-level language for an event-based programming common to assembly tasks.

For an acquired or a mathematically designed trajectory, so to say a motion template, we show how time and coordinate scaling as well as a time-dependent coordinate transformation can be used to extend the template to a larger workspace. The time-dependent coordinate transformation lays the ground for a closed-loop approach to trajectory generation for fixed-time problems. In the closed-loop approach, a desired reference signal (possibly time varying) is tracked by acting upon the feedback from the actual state of a robot. Using the Hamilton-Jacobi-Bellman equation we derive a closed-loop solution to the fixed-time trajectory-generation problem with a minimum-jerk cost functional. In this case, it turns out that the time-dependent coordinate transformation does not affect the optimality of the solution. Moreover, we show the resulting trajectory coincides with a fifth-order polynomial function of time that instantaneously updates due to the changes in the reference signal and/or the robot states.

We make a short comparison between kinematic and dynamic models for generating trajectories. The conclusion is that given conservative kinematic constraints, both models behave in a similar way. Having this in mind, we derive an analytic solution to the problem of fixed-time trajectory generation with a quadratic cost function under velocity and acceleration constraints. Despite the simplicity of this

problem, it has a wide range of applications within motion planning. The advantage of the analytic solution compared to an on-line optimization approach lies in the efficiency of the calculation and the accuracy of the solution.

To extend the idea of closed-loop trajectory generation, by accommodating a more generic form of system dynamics and constraints, we adapt the Model Predictive Control (MPC) framework. MPC is traditionally applied to tracking problems, i.e., when there is an explicit reference signal. Thus, it is a common practice to have a separate layer that generates the reference signal. We have proposed an integrated approach by introducing a final state constraint in the formulation. Additionally, we give the interpretation that the difference between tracking and point-to-point trajectory-planning problems is in the density of the specified desired reference signal. In other words, in the tracking problem the desired output at every sample is specified while in the point-to-point trajectory planning, it is limited to certain samples only. We utilize a strategy to reduce the discretization time successively. This way, we respect the real-time constraints for computation time while the accuracy of the solution is gradually improved as the deadline approaches. We have verified our proposed MPC approach to trajectory generation in a ball-catching experiment.

Acknowledgements

I have been fortunate to work in a vibrant research environment at the Department of Automatic Control at Lund University. I would like to thank my supervisor Prof. Rolf Johansson, and co-supervisors Prof. Anders Robertsson and Prof. Jacek Malec for their constructive comments on my research and this thesis. I would like to give special thanks to Assoc. Prof. Klas Nilsson for his thoughtful discussions and ideas during various project meetings.

My thanks extend to the people in the robotic laboratory, especially former PhD student Dr. Magnus Linderoth and my colleagues Andreas Stolt and Björn Olofsson for helping and sharing their experiences. Without constant support on hardware and software by our research engineers, Anders Nilsson, Anders Blomdell, Leif Andersson, Pontus Andersson, and Martin Holmstrand none of the results in this thesis were possible. I am equally thankful to the administrative staff for maintaining a smooth workflow in the department.

It was a great opportunity to share office with a former PhD student, Dr. Daria Majidian. As we believed, we had many insightful discussions about technical and less technical topics. I wish to express my gratitude to Asst. Prof. Martina Maggio and Dr. Alessandro V. Papadopoulos for encouragements along the way.

Financial support is acknowledged from the European Union's seventh framework program (FP7/2007-2013) under grant agreement PRACE (Ref. #285380). The author is a member of the LCCC Linnaeus Center and the eLLIIT Excellence Center at Lund University.

Finally, I would like to make an excuse to readers for insufficiency and inevitable errors in this work. Given a finite time one must choose between a submitted thesis and a perfect one!

Contents

1

Introduction

1.1 Background and Motivation

Trajectory generation is an inherent problem in motion control for robotic systems, such as industrial manipulators and mobile platforms, in order to transfer the system from one initial state to another desired state [Kröger and Wahl, 2010].

A common strategy to motion planning is the decoupled approach [LaValle, 2006; Verscheure et al., 2009] since it reduces the complexity of the complete motion-planning problem. This strategy means that a path is first determined taking into account the geometry of the task and the environment, e.g., obstacles or other robots in a shared workspace. Subsequently, a trajectory planning is performed in order to achieve tracking of the *a priori* planned geometric path.

Various modeling assumptions have to be made. A major difference is due to whether a pure kinematic model is considered, or if the complete linear or nonlinear dynamics of the system are modeled and considered. Obviously, the latter is more complex and the computations are significantly more time consuming. In general, a dynamic model is required to guarantee satisfaction of the physical constraints such as actuator limitations. This can limit the performance of the decoupled approach or even give rise to infeasible trajectories for a certain path.

A prerequisite for modern robot control is the sensor measurements of both internal robot quantities (typically by joint encoders/resolvers in manipulators and wheel encoders on mobile platforms) and external sensors providing information about the state of the workspace (such as vision and force/torque sensors) [Kröger and Wahl, 2010]. Consequently, another desired characteristic of the motion planning in uncertain or unstructured environments is the ability to react to sensor inputs. In practice, this means that the path as well as the corresponding trajectory need to be recomputed online with real-time constraints.

In many applications, the trajectory generation is desired to be performed such that the time for executing a task, or the energy consumed, during the motion is minimized. Hence, motion-planning problems are often formulated as optimal-control problems. The cost functional implicitly results in a coordination between the different degrees-of-freedom (DOF) of the system.

As an alternative to the decoupled approach in cases where only the initial and final states are of interest, a direct integrated approach to motion planning can be pursued. Ultimately, this leads to a motion planning where the full potential of the system is utilized, since the geometric, kinematic, and dynamic constraints can be considered directly when performing an optimal motion planning [Choset, 2005]. The usage of this approach, though, has been limited because of the time required for computing solutions online for complex dynamic systems such as industrial robots with multiple DOF.

Algorithms and methods for offline trajectory generation for time-optimal path tracking for industrial manipulators were already developed in the 1980s [Bobrow et al., 1985; Shin and McKay, 1985]. An overview of trajectory-generation methods for robots is provided in [Kröger, 2010]. Most of the previously suggested methods for online trajectory generation for mechanical systems were based on a library of analytic expressions, parametrized in the initial and final states of the desired motion and certain constraints [Kröger and Wahl, 2010; Kröger, 2011a]. Efficient algorithms and data structures for online trajectory generation based on this method were implemented and distributed as part of the Reflexxes motion library [Kröger, 2011b]. Other approaches to online trajectory generation based on parametrized motion patterns were considered in [Castain and Paul, 1984], [Macfarlane and Croft, 2003], and [Haschke et al., 2008].

The major advantage of the existing analytic solutions is that they can be computed extremely fast. Nevertheless, they typically address time-optimal problems and are limited in the constraints that can be handled in the motion planning. Alternatively, in [Verscheure et al., 2009] it was shown how the time-optimal path-tracking problem can be solved using convex optimization techniques under certain assumptions on the robot model.

While the minimum-time trajectories are of interest for defining an upper bound for productivity of a robotic system, they put the system under maximal stress. In practice, (e.g., in a production line) several components are involved and there is a sequence of dependent operations that determines the required time. Hence, the solution to fixed-time problems can prove valuable by reducing wear and tear of the robotic system. It appears that the fixed-time optimal trajectory-planning problems are relatively underexplored. Sub-optimal solution to fixed-time trajectory planning for robot manipulators was proposed by [Duleba, 1997]. In another approach, the fixed-time optimal solution in the space of parametrized trajectories were found [Yang et al., 2012].

The motivation for this thesis is to examine some new ideas for trajectory generation to address some of the existing issues or to extend the previous methods. Initially, we investigate robotic setups for acquiring trajectories from human demonstration as faithfully as possible. Fast algorithms to obtain solutions to fixed-time point-to-point trajectory generation is a major theme in this work. We propose a closed-loop approach as well as various optimal strategies to this problem.

1.2 Contributions

The major contributions of this thesis are:

- Derivation of a closed-loop minimum-jerk trajectory generation,
- Derivation of an analytic solution for point-to-point fixed-time trajectory planning under maximum velocity and maximum acceleration constraints,
- Extension of Model Predictive Control framework to point-to-point trajectory generation,
- Robot-assisted teleoperation by means of compliance,
- Passive lead-through using free-floating motion,
- Extending guarded motion with fault management and priority notions.

1.3 Publications

The thesis is, by and large, a standalone piece of work, except the contents of Sections 2.3 and 4.4 which are based on the papers below.

Ghazaei Ardakani, M. M., J. H. Cho, R. Johansson, and A. Robertsson (2014). "Trajectory generation for assembly tasks via bilateral teleoperation". In: *Proc. of the 19th IFAC World Congress*. Vol. 19. 1. Cape Town, South Africa, pp. 10230–10235. DOI: 10.3182/20140824-6-ZA-1003.02559.

Ghazaei Ardakani, M. M., B. Olofsson, A. Robertsson, and R. Johansson (2015). "Real-time trajectory generation using model predictive control". Unpublished manuscript.

The idea of the first paper is due to the first author, by whom the major part of the implementation was carried out. The second author was responsible for the implementation of the interface to the master device and contributed in the preparation of the manuscript. The third and fourth authors were involved in proof-reading and improving the manuscript.

The problem formulation of the second paper is due to the first author. The verification of the ideas and the implementation of algorithms were done together with the second author. The manuscript was prepared by the first and the second author, while proof-reading and improvements of the text were suggested by the third and the fourth co-authors.

Code generation and the interface for programming a guarded motion in ABB RobotStudio was developed by Maj Stenmark [Stenmark et al., 2014]. The programming concept and the improvements in the robotic architecture and their related implementations are due to the author of this thesis.

The author has used some of the results from the EU/FP7 ROSETTA project[1]. Specifically, the iTaSC [De Schutter et al., 2007] implementation in Simulink was done as part of this project. Additionally, the gravity and friction models are due to Andreas Stolt [Stolt, 2012].

The derivation of the analytic solution to the fixed-time trajectory planning (Section 4.3) was done together with Meike Stemmann. The author has completed the solution and provided numerical results.

Other publications:

The following list of papers, in which the author has contributed, were decided not to be a part of this thesis.

From, P. J., J. H. Cho, A. Robertsson, T. Nakano, M. Ghazaei, and R. Johansson (2014). "Hybrid stiff/compliant workspace control for robotized minimally invasive surgery". In: *Proc. 5th IEEE RAS EMBS Int. Conf. Biomedical Robotics and Biomechatronics, 2014*, pp. 345–351.

Ghazaei Ardakani, M. M. and R. Johansson (2012). "Interpolation and spectral estimation of non-uniformly sampled signals". Unpublished manuscript.

Ghazaei Ardakani, M. and B. Bernhardsson (2014). *Spurious convergence of iterative learning control*. Poster presented at Reglermöte 2014, June 3–4, Linköping University, Linköping, Sweden.

Ghazaei Ardakani, M., H. Jörntell, and R. Johansson (2011). "ORF-MOSAIC for adaptive control of a biomimetic arm". In: *Proc. IEEE Int. Conf. Robotics and Biomimetics (ROBIO), 2011*, pp. 1273–1278.

1.4 Thesis Outline

In Chapter 2, we present methods to acquire human-generated trajectories as faithfully as possible. Additionally, a software architecture for programming robots using these methods is proposed. Chapter 3 presents methods for extending a trajectory to a larger workspace and a closed-loop perspective on trajectory generation. Some of the optimization-based approaches to trajectory generation are presented in Chapter 4. First, we study the applicability of simple kinematic models compared to dynamic models. Second, an analytic solution to fixed-time trajectory planning with state constraints is derived. Third, we adapt Model Predictive Control for point-to-point trajectory generation. Finally, conclusions are drawn in Chapter 5 and ideas for future research are presented.

[1] European Community's Seventh Framework Programme FP7/2007-2013 – Challenge 2 — Cognitive Systems, Interaction, Robotics – under grant agreement No. 230902 – ROSETTA

2

Human-Generated Trajectories via Haptic Interface

2.1 Introduction

Robots have been used for decades to conduct repetitive tasks, e.g., assembly and pick-and-place operations, to replace human workers. Numerous industrial environments have utilized robotic systems to increase the amount of outcomes, improve the efficiency of processes, and to improve the work environment for human workers by delegating the monotonous tasks to robots. New generation of robots, such as ABB YuMi [ABB Robotics, 2015; Kock et al., 2011] shown in Fig. 2.1, has been built to work side-by-side with human workers, hence requiring a more intuitive way of programming and interaction with humans. However, programming of robot systems for complex and flexible tasks has still remained a challenging problem. There are numerous suggested solutions in the literature, many of which rely on trajectory programming prior to operation. In general, those methods requiring state-event modeling are not easily implementable or compatible for different tasks.

Generating robotic motions is usually based on teaching via some human–robot interface (HRI) or 3D simulation programming environments. The manipulation of the robotic systems is possible not only using a teach pendant but also with direct manipulation, so called lead-through programming. The direct manipulation of robots is much more intuitive than the conventional text-based programming or even graphical simulation since trajectories are generated via human demonstration. For teaching by demonstration, if the robot does not have a similar kinematic structure as human, the mapping between robot motion and human motion will not be trivial. By using the robot as part of the demonstration of a task, this problem could be entirely avoided. Therefore, this approach has been widely used in human skill acquisitions, [Argall et al., 2009; Lee et al., 2012].

Figure 2.1 ABB YuMi Robot [ABB Robotics, 2015; Kock et al., 2011] used mainly for experiments in Sections 2.2 and 2.4.

Despite this, the interface between humans and robots can be inconvenient and difficult for accurately transferring motions due to mechanical properties of robots such as inertia and friction. Hence, this approach is mainly used for small and lightweight robots. Although compliant motion control could be employed to reduce inertial/friction forces, direct teaching of industrial robots is still limited.

In this chapter we present two setups suitable for teaching robots. In Section 2.2, we describe a simple, yet quite robust approach for direct teaching. Using admittance control and motion constraints, we explain how the robot system could help an operator to accomplish a task in Section 2.3. An event-based programming approach is sketched in 2.4 which can utilize the setups described in this chapter. The conclusions are drawn in Section 2.5.

2.2 Free-Floating Motion

Typical implementation of lead-through requires knowledge of forces and torques at the end-effector. To implement a behavior for manipulators to react to the forces applied to an arbitrary point, force sensing at each joint might be required. While the cost of sensors and practical problems have made the implementation of robots with joint sensors quite limited, the estimation of forces at each joint provides a reasonable alternative. Nevertheless, the accuracy of the estimation and the inherent uncertainty due to Coulomb friction creates limitations for in-contact operation.

If a robot has low mass and the motors are back-drivable (low gear ratio is required), we do not have to fully cancel out the dynamics of the robot. For a kinesthetic teaching, the dominating undesirable dynamics is the effect of the gravity and

friction. Therefore, gravity and friction compensation can drastically improve the interaction with the robot.

In our proposed approach, the friction will partly be on our side since it stabilizes the robot. The natural deadzone due to the static friction adds robustness against model uncertainties too. When switching to free-floating motion, we disable the internal feedback controller and use only a feedforward signal calculated by

$$\tau_{ff} = g(q) + F_C(q,\dot{q}) + L(q) + kJ(q)h_{ex}. \tag{2.1}$$

Here, q represents joint angles, $\dot{q}$ is the vector of joint velocities, the geometric Jacobian is denoted by $J(q)$, h_{ex} denotes the force/torque measurements from a wrist-mounted sensor, k is a static gain, $F_C(q,\dot{q})$ denotes the friction model used for friction compensation, $L(q)$ is a model for increasing the stiffness close to the joint limits, and $g(q)$ is the gravity compensation term.

Note that in our implementation, it was not possible to reset the integrators of the internal controller. Therefore, special care was taken to make a bumpless transition between the free-floating and the closed-loop modes.

Partially Closing the loop

In robot programming by jogging or by lead-through demonstration, it is often desired to lock the orientation and/or the translation along certain axes in Cartesian space. This is possible by implementing a PID controller [Siciliano and Villani, 1999]

$$F_{imp} = K_t \Delta p - Dv + \frac{1}{s} K_i \Delta p \tag{2.2}$$

$$\tau_{imp} = K_o \eta\varepsilon - D_o \omega + \frac{1}{s} K_{io} \eta\varepsilon. \tag{2.3}$$

Here, $\Delta p \in \mathbb{R}^3$ is the error in the Cartesian space and η and ε are defined such that $q = (\eta, \varepsilon)$ is a unit quaternion [Chou, 1992] representing the rotational error, and v and ω are the translational and the rotational velocities, respectively. We use the diagonal matrices $K_t, D, K_i \in \mathbb{R}^{3\times3}$ for the translation part and $K_o, D_o, K_{io} \in \mathbb{R}^{3\times3}$ for the rotational part. The elements of K_t, K_i, and D corresponding to free translational axes (not locked) are set to zero. In a similar way, the elements of K_o, D_o, and K_{io} corresponding to free rotational axes are set to zero.

After disabling the internal controller, the torque delivered to the robot will be

$$\tau = \tau_{ff} + \tau_{cl}, \tag{2.4}$$

where

$$\tau_{cl} = J^T(q) \begin{pmatrix} F_{imp} \\ \tau_{imp} \end{pmatrix}. \tag{2.5}$$

Results and Discussion

In this section the results of the free-floating motion is presented. In the first three experiments, we investigate the effect of friction compensation term and the term related to the force sensor $kJ(q)h_{ex}$ according to (2.1). The objective is to lead-through the robot along each Cartesian axis while keeping the orientation constant. This task is more difficult than moving a single joint, since the friction of several joints add up together. Figure 2.2 shows the result for the case when only the gravity compensation is active. The operator follows a predefined path in each experiment, starting from the gripper in touch with a table and finishing at the same point. The gripper is moved in the positive z-direction defined upwards with respect to the table, then followed by a motion forward and then backward in y-direction, defined perpendicular to the robot base and parallel to the table, and then in the positive x-direction away from the robot base and back again. Finally the gripper is brought to its initial position by moving in the negative z-direction.

Figure 2.3 shows the result of a similar experiment. This time, the friction compensation was activated as soon as *motion* was detected. The method adds an additional feedforward torque to each joint equal to 40% of its estimated static friction band. In Fig. 2.4 corresponding to the third experiment, the friction compensation was lowered to 30%. However, it was activated as soon as an *external force* was detected. Additionally, the external force was amplified with a factor of two and added to the feedforward torque signal according to 2.1. The external torques were ignored since they contribute by reducing the rotational resistance, which contradicted the point of this experiment.

Although Figs. 2.2, 2.3, and 2.4 look qualitatively similar, the force values in x and y directions are almost halved with the introduction of every new feature. This indicates that the operator could achieve the same task by a reduced amount of effort, i.e., in the last experiment by a quarter of the forces required in the first experiment. This advocates integrating both friction compensation and a force sensor. Free-floating motion proved to be extremely robust. It is because of the passivity of the mechanical system and the lack of any feedback loop. However, the gravity and the friction compensation methods require special attention, since in an entirely open-loop strategy there is no way to compensate for model errors. Using adaptive control for re-tuning models can be a possible solution in this case.

The fourth experiment shows the result of partially closing the loop. In this experiment, we have locked the position while letting the orientation of the gripper to be adjusted. Figures 2.5 and 2.6 show the orientation of the gripper represented by a unit quaternion and joint angles, respectively. As seen, both the orientation and the joint angles are varied over a large domain. Nevertheless, the displacement in the flange position shown in Fig. 2.7 is less than 3 [mm]. This value can probably be improved. However, due to the non-linearities, especially the backlash and the unmodeled friction, in addition to the inherent limitations in the mechanical stiffness of the robot and its actuation, the stiffness cannot be arbitrarily large.

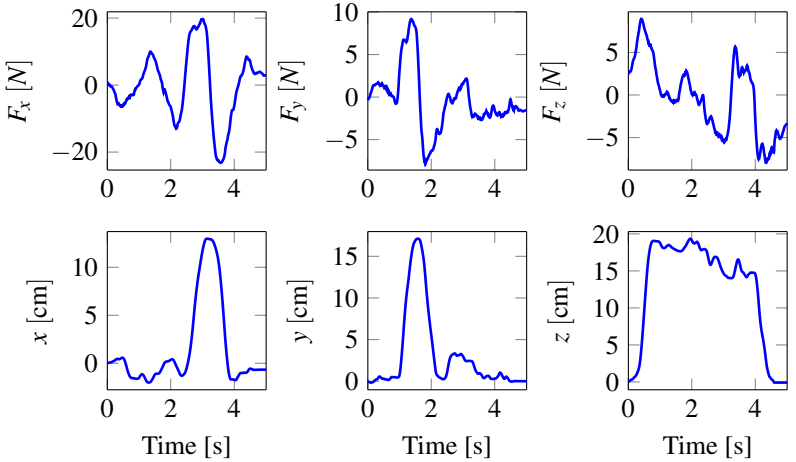

Figure 2.2 Experiment 1 (only gravity compensation): lead-through of the gripper following the sequence $(+z, +y, -y, +x, -x, -z)$ while trying to keep the orientation constant. The components of the force applied to the gripper are depicted in the upper part and the position coordinates in the lower part.

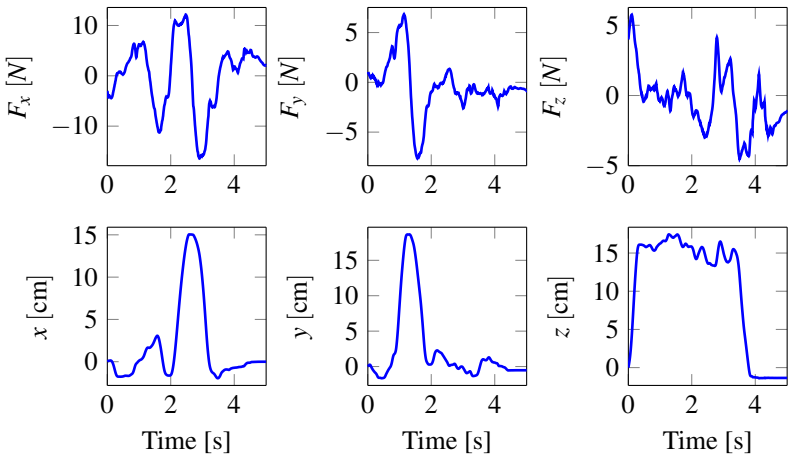

Figure 2.3 Experiment 2 (friction compensation activated): lead-through of the gripper following the sequence $(+z, +y, -y, +x, -x, -z)$ while trying to keep the orientation constant. The components of the force applied to the gripper are depicted in the upper part and the position coordinates in the lower part. Compare the forces with the baseline in Fig. 2.2.

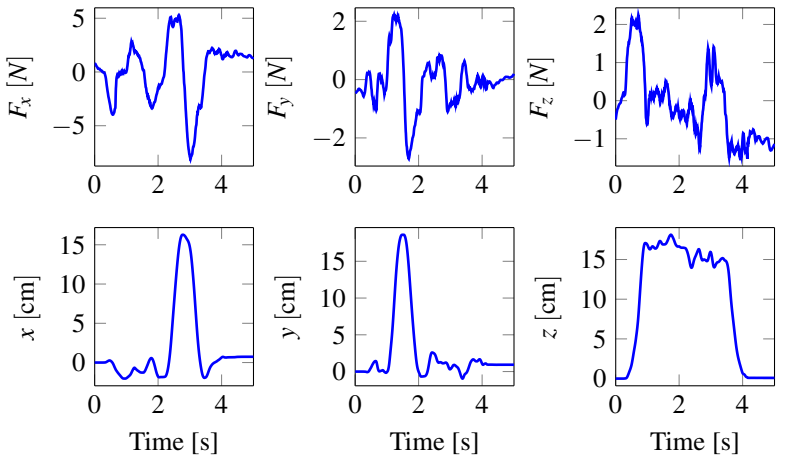

Figure 2.4 Experiment 3 (friction compensation and force sensor activated): lead-through of the gripper following the sequence of $(+z, +y, -y, +x, -x, -z)$ while trying to keep the orientation constant. The components of the force applied to the gripper are depicted in the upper part and the position coordinates in the lower part. Compare the forces with the baseline in Fig. 2.2.

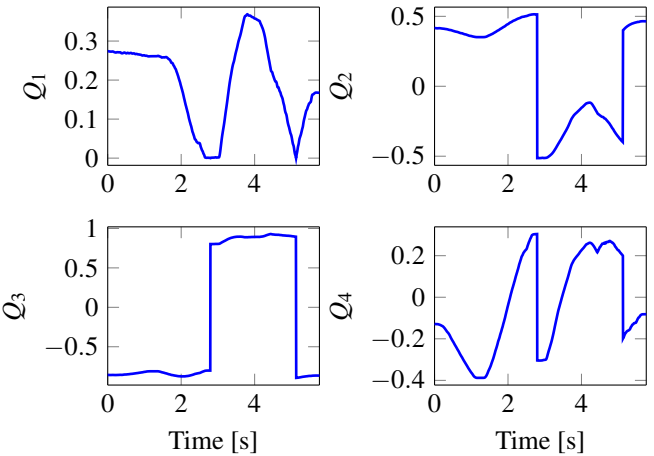

Figure 2.5 Experiment 4: reorientation of the gripper by interacting with an arbitrary point on the arm while the flange position is locked; the curves show the orientation of the gripper represented by a unit quaternion $Q = (Q_1, Q_2, Q_3, Q_4)$.

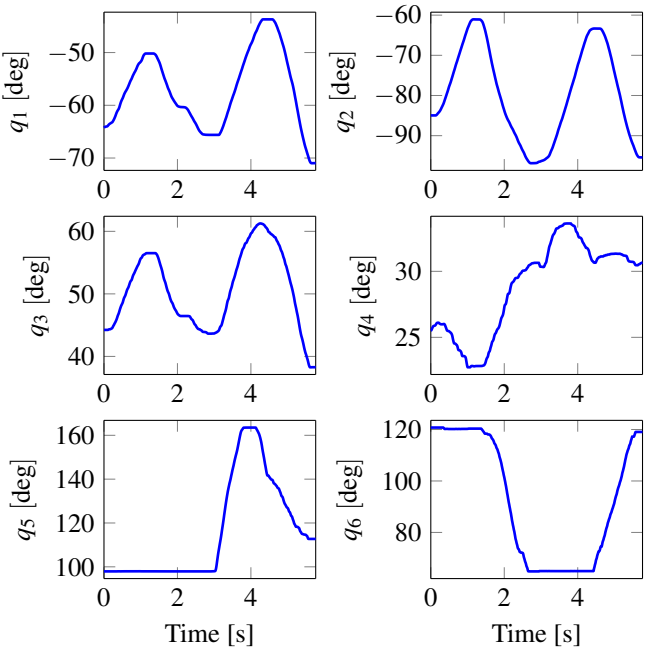

Figure 2.6 Experiment 4: reorientation of the gripper by interacting with an arbitrary point on the arm while the flange position is locked; the curves show the joint angels. The joint values for wrist (q_7) have been omitted.

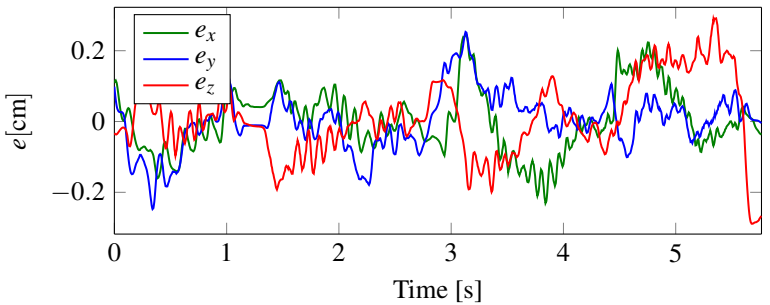

Figure 2.7 Experiment 4: reorientation of the gripper by interacting with an arbitrary point on the arm while the flange position is locked; the curves show the displacement of the locked position in different directions due to the limited stiffness.

2.3 Robot-Assisted Teleoperation

An alternative approach to the kinesthetic teaching for generating trajectories is teleoperation of a robot. In the teleoperation setup, a human operator manipulates a “master device" to control a “slave robot" in a remote environment. The master device is to decode the commands issued by an operator and forward them to the slave robot. For precise remote control of the slave robot, it is required to provide sensory feedback such as visual, haptic, or auditory feedback to the operator. Haptic feedback provides the operator with the sense of touch of what the robot may perceive in contact operation with the work space or the environment. Thus, receiving haptic feedback is essential when there is a contact between the slave robot and the environment. Besides the visual feedback, the haptic feedback establishes a bilateral communication channel between the operator and the robot.

The control of bilateral teleoperation systems has been extensively studied for decades, see [Hokayem and Spong, 2006] and the references therein. A schematic diagram of a bilateral teleoperation system is illustrated in Fig. 2.8. The control architecture represents which information has to be exchanged between the two sides [Aliaga et al., 2004]. The ultimate goal of bilateral teleoperation is to achieve transparency by means of two-directional position and force tracking [Lawrence, 1993; Yokokohji and Yoshikawa, 1994]. Despite recent advances in bilateral teleoperation in the task space [Liu and Chopra, 2012; Wang, 2013], experiments are still mainly limited to robots with a few degrees of freedom (DOF). In general, the design of a bilateral controller for industrial robots is challenging due to practical problems including inertia, friction, control bandwidth, and time delay.

In this section, we propose a convenient framework to generate robotic trajectories within the teleoperation setup. Since delay makes it very difficult to handle stiff contacts in bilateral teleoperation [Colgate and Hogan, 1989], a compliant motion control has been employed. Moreover, the targeted compliance of the slave robot can be adjusted by the operator in real-time. This tele-admittance idea increases the flexibility for various tasks. We implemented the proposed approach in a 6-DOF setting and verified it with a couple of assembly tasks.

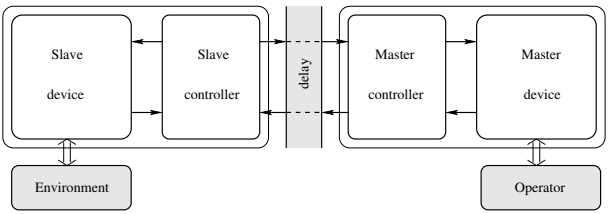

Figure 2.8 Bilateral teleoperation system: information is exchanged between the master and the slave in a bilateral manner.

The remainder of this section is organized as follows. Firstly, a mapping between a master device and a slave robot is defined. Secondly, the control architecture of the system is described in terms of tele-admittance and haptic feedback to the operator. Thirdly, the experimental results are illustrated by the interaction with a solid block, a peg-in-hole experiment, and a snap-fit task, respectively. Finally, the issues raised in the experiments and some ideas for further improvements are discussed.

Kinematic Chain

A mapping between the degrees of freedom of a master device and a slave robot must be defined. To have a natural and generic approach we make use of the following feature frames. Each feature frame defines a coordinate frame and is attached to an object according to the iTaSC formalism [De Schutter et al., 2007].

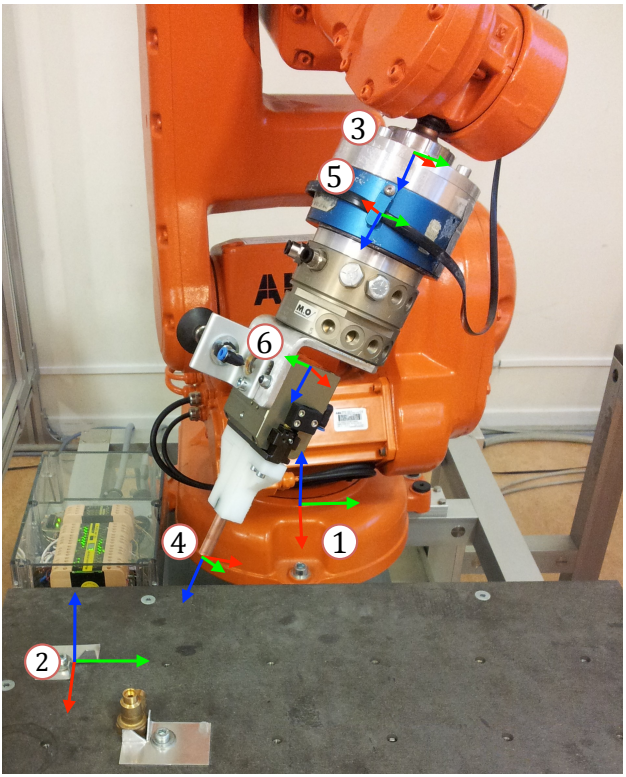

Figure 2.9 Visualization of slave (robot) frames: 1) Slave Base Frame, 2) Slave Task Frame, 3) Flange Frame, 4) Slave Frame, 5) Sensor Frame, 6) Wrist Frame.

For the master device, we make use of:

1. *Master Base frame*: this is the basic coordinate system provided by the haptic device. The position and orientation values are relative to this frame.
2. *Master Task frame*: this defines a natural frame for the manipulation tasks, which might be rotated, e.g., with respect to the orientation of the haptic device.
3. *Handle frame*: this is a frame firmly attached to the handle of the haptic device and is usually predefined.
4. *Master frame*: this allows for an offset between the handle frame and the desired one, e.g., when the handle is augmented with a tool.

Similarly, for the slave (robot) we define:

1. *Slave Base frame*: this is the default coordinate frame for a robot, which normally has its origin at the base of a robot.
2. *Slave Task frame*: this frame defines a coordinate system which is more convenient for specifying a task.
3. *Flange frame*: this is a frame firmly attached to the flange of the robot.
4. *Slave frame (Tool frame)*: the origin of this frame is located at the tool center point (TCP). The frame defines the tool coordinate system.
5. *Sensor frame*: this frame is attached to the force/torque sensor and is aligned with its coordinate system.
6. *Wrist frame*: this is a frame for active compliance of the robot. The origin of this frame acts as a virtual joint with respect to external forces. The designated principal moments of inertia are aligned with this frame.

Various slave coordinate frames are illustrated in Fig. 2.9. The relationships between these frames are represented in Fig. 2.10. Each transformation corresponds to a translation and a rotation. The solid arrows represent the given transformations (either fixed or measurable). The dashed arrows correspond to the inferred transformations.

We define the mapping between the haptic interface and the robot as the translational and the rotational velocity coupling between the master frame with respect to the master task frame and the slave frame with respect to the slave task frame. In case of no scaling of motion and no interaction forces, this is equivalent to the matching of the slave frame and the master frame relative to their corresponding task frames

$$T_{211} = T_{111}. \tag{2.6}$$

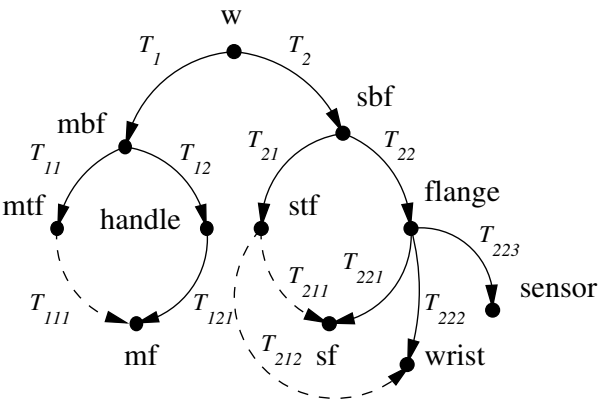

Figure 2.10 Scene graph: world (w), master base frame (mbf), master task frame (mtf), master frame (mf), slave base frame(sbf), slave task frame (stf), slave frame (sf), handle, flange, sensor and wrist frames.

This can equivalently be described in terms of typical master and slave transformations

$$T_{211} = T_{221}T_{22}T_{21}^{-1} \tag{2.7}$$

$$T_{111} = T_{121}T_{12}T_{11}^{-1} \tag{2.8}$$

$$T_{22} = T_{221}^{-1}T_{121}T_{12}T_{11}^{-1}T_{21}. \tag{2.9}$$

In order to allow for position references by other devices, such as a teach-pendant, an offset between the master frame and the slave frame needs to be calculated. The offset could be used also to compensate for an initial mismatch between the two frames. Adding an offset to T_{111} is equivalent to updating the translational part of T_{11} and the rotational part of T_{121}.

Compliant frame

In case of external forces, we superimpose a displacement ΔT to the equation of the motion at the wrist point

$$T_{212} = T_{222}T_{221}^{-1}T_{211} = \Delta T T_{222}T_{221}^{-1}T_{111}. \tag{2.10}$$

Taking into account the compliant frame, we get a new transformation between the flange frame and the robot base frame

$$\tilde{T}_{22} = T_{222}^{-1}\Delta T T_{222}T_{22}. \tag{2.11}$$

For the representation purpose, instead of transformation matrices, we use a vector of $\mathbf{p} \in \mathbb{R}^3$ and a quaternion $\mathbf{q} \in \mathbb{R}^4$ with the unity constraint $||\mathbf{q}|| = 1$. If

$\mathbf{n}$ corresponds to the axis of rotation and θ the amount of rotation, this can be represented by $\mathbf{q} = (\eta, \boldsymbol{\varepsilon})$. The scalar part is $\eta = \cos(\theta/2)$ and the vector part is $\boldsymbol{\varepsilon} = \sin(\theta/2) \cdot \mathbf{n}$ [Chou, 1992; Hamilton, 1847].

The immediate benefit of such a representation is that it is singularity-free. All computation of rotations can be carried out by making use of quaternion algebra, specifically quaternion multiplication and inversion. Additionally, we can describe the equations of the controller corresponding to the ΔT transformation in terms of $\mathbf{q}$ and $\mathbf{p}$.

Control Architecture

We employ a modified force-velocity control architecture, which requires position information of the master device and force information of the slave robot. Let us denote $\mathbf{v}_m$ as the translation velocity of the master frame with respect to the master task frame. Then, $\mathbf{v}_m$ is transmitted to the slave side with a motion scaling factor γ and used as a reference velocity for the slave robot. To enable micro motion at the slave side, γ is generally set to less than one.

Tele-admittance Often due to limitations such as bandwidth and delay, it is not possible to have a fully transparent haptic teleoperation. Therefore, it is desirable to give the robot some extent of autonomy. Specially, when dealing with contact forces, the delay in the feedback might impede the operator to react in time. Consequently, a stiff industrial robot may cause workpiece or tool damages. A human-like strategy to mitigate this problem is to adjust the arm impedance. In case of high uncertainties, decreasing the impedance results in reducing the interaction forces. It was shown by [Ajoudani et al., 2012] that transferring the desired impedance is an effective strategy for teleoperation. Thus, we allow the operator to adjust it remotely.

A 6-DOF admittance controller was implemented to achieve active compliance at the tool center point (TCP), [Siciliano and Villani, 1999; Caccavale et al., 1999; Hogan, 1985]. The controller provides an isotropic translational and rotational admittance behavior with tunable parameters. The robot acts as a mass-spring-damper system. The rotational part allows for different moments of inertia along each axis.

Similar to [Villani and De Schutter, 2008], we use the notation $m, k_t, D \in \mathbb{R}$ for the translation part and $M \in \mathbb{R}^{3\times3}$, and $k_o, D_o \in \mathbb{R}$ for the rotational part. Using the admittance controller, we enforce the following dynamics for the wrist frame due to the vector of the external force $\mathbf{F}_i$ and the vector of torque $\boldsymbol{\tau}_i$

$$\dot{\mathbf{p}} = \mathbf{v} \tag{2.12}$$

$$m\dot{\mathbf{v}} = -k_t\mathbf{p} - D\mathbf{v} + \mathbf{F}_i \tag{2.13}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\tilde{\boldsymbol{\omega}} \otimes \mathbf{q} \tag{2.14}$$

$$M\dot{\boldsymbol{\omega}} = -2k_o\eta\boldsymbol{\varepsilon} - D_o\boldsymbol{\omega} - \boldsymbol{\omega} \times M\boldsymbol{\omega} + \boldsymbol{\tau}_i, \tag{2.15}$$

where $\mathbf{p}$ is the displacement vector of the wrist from the commanded position, $\mathbf{v}$ is the velocity vector, $\tilde{\boldsymbol{\omega}} = (0, \boldsymbol{\omega})$, and "$\otimes$" and "$\times$" indicate quaternion multiplication and cross product, respectively.

We choose the wrist frame for numerical integration of Eqs. (2.12)–(2.15). Since we have assumed that the principal moments of inertia are aligned with this frame, the matrix M becomes diagonal in (2.15). To ensure the unity constraint on the quaternion, it is normalized after the numerical integration.

Assuming we have an ideal reference tracking, we equate the reference signals with the position of the calculated compliant frame above.

Haptic feedback On the master side, the haptic feedback to the operator given in the master task frame, $\mathbf{F}_{fb}^{MTF}$, can be designed as follows. We denote the vector of forces at the TCP point given in the slave task frame by $\mathbf{F}_i^{STF}$ and the translational velocity of the slave frame with respect to the slave task frame by $\mathbf{v}_s$. Note that these variables appear delayed.

$$\mathbf{F}_{fb}^{MTF} = \gamma(D_2 - D_1)\mathbf{v}_s - D_2\mathbf{v}_m + \alpha\mathbf{F}_i^{STF}, \tag{2.16}$$

where $\mathbf{v}_m$ is the translation velocity of the master frame with respect to the master task frame, α represents a force scaling factor to adjust direct haptic feedback to the operator, and D_1 and D_2 correspond to the free-motion damping and in-contact damping, respectively. When there is no contact between the slave and the environment and the delay is not significant, $\mathbf{F}_i^{STF}$ and $\mathbf{v}_m - \gamma\mathbf{v}_s$ are almost equal to zero so that the operator perceives only some damping force corresponding to D_1. On the other hand, in contact $\mathbf{v}_s \approx 0$ and the operator perceives a scaled force with the other damping coefficient, D_2. Two damping factors were deployed to enhance the contact detection and stability.

Let R_{11} be the rotation of T_{11} transformation. The force feedback in the master base frame then can be calculated as

$$\mathbf{F}_{fb} = R_{11}\mathbf{F}^{MTF}. \tag{2.17}$$

Experimental Setup

Our setup consisted of a Force Dimension Omega-7 device which is a 7-DOF haptic interface with closed loop stiffness of 14.5 (N/mm), see [*Omega-7 Overview*]. We used the extra DOF of the device (the extent of the openness of the gripper) to adjust admittance remotely. Moreover, an industrial robot, ABB IRB 140 [ABB Robotics, 2014], a wrist mounted six axis force/torque transducer with internal electronics, JR3 100M40A were used. The controller was implemented in Simulink® and using the Real-time Workshop, the code was compiled for ExtCtrl, [Blomdell et al., 2010]. The final code runs at 250 Hz on a Fedora Xenomai machine connected via the ExtCtrl interface to the robot.

See Fig. 2.11 for the setup. Table 2.1 represents the nominal parameters of the slave controller.

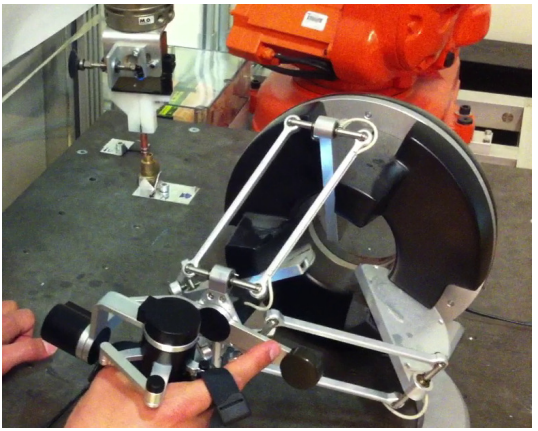

Figure 2.11 Peg-in-hole setup: (Front) Master side with Omega-7 haptic interface. (Back) IRB140 robot with metal pipe held in the gripper.

Results

Three different scenarios were tested. The operator sequentially touched three sides of a solid block in the first experiment. The second experiment was a peg-in-hole operation. The third one involved a more challenging task of snap-fit assembly.

The pushing task was meant to demonstrate the compliant behavior of the slave robot. The operator starts from the right hand side of the block and pushes the block. Then the end-effector is moved behind the block to push it along the x-axis. Finally, the end-effector is moved to the top of the block and to push it downward; see Figs. 2.12, 2.13, 2.14, and 2.15 for details. Note that the last interaction is almost perpendicular to the surface and does not cause any torque.

As expected, the robot gives way to the external forces according to the demanded admittance. In general, the slave robot lags behind the master device. This is due to imperfect tracking and delay.

The peg-in-hole problem is a classical assembly operation. In this experiment, the operator inserts a small metal pipe inside a valve by using teleoperation. Figure 2.11 shows the setup for this task. By opening/closing the gripper of the master

Table 2.1 Controller parameters

m	k_t	D_t
1.5	10	0.4

M_{xx}	M_{yy}	M_{zz}	k_o	D_o
0.2669	0.2669	0.0338	18	5

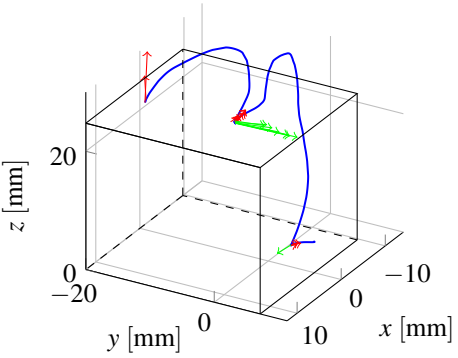

Figure 2.12 Pushing a solid block. The trajectory is in blue, force vectors are in red, and torque vectors are in green.

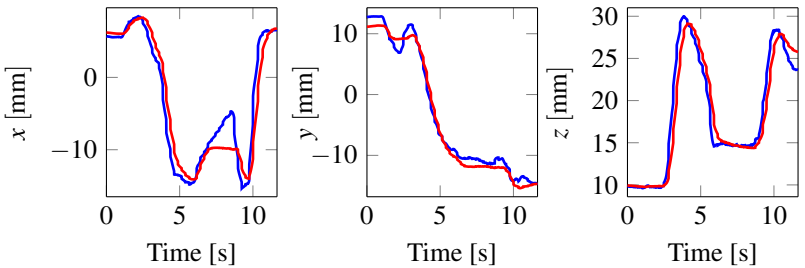

Figure 2.13 Pushing a solid block. Comparison of master (blue) and slave (red) trajectories.

device—*i.e.*, by changing the relative distance between the index finger and the thumb—the operator can adjust the stiffness in the algorithm. The slave frame superimposed on the trajectory is illustrated in Fig. 2.16. Figure 2.17 visualizes the interaction forces and torques.

The third task involves installing a switch in an emergency button box, see Fig. 2.18. The operator must approach the box with a specific orientation of the switch. After alignment, the switch should be rotated and pressed until it snaps. Figure 2.19 illustrates the trajectory and the slave frame. In Fig. 2.20, the interaction forces and torques are visualized. This task will be called the snap-fit task.

Discussion

Stability Since the bilateral teleoperation system is coupling two dynamical systems (master and slave), it leads potentially to instability. The analysis and synthesis are made difficult by several factors including transmission time delays between

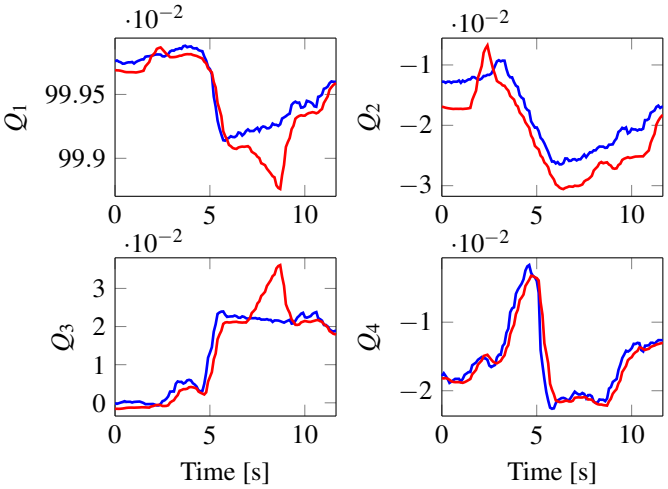

Figure 2.14 Pushing a solid block. Comparison of master (blue) and slave (red) orientations described by the quaternion $Q = (Q_1, Q_2, Q_3, Q_4)$.

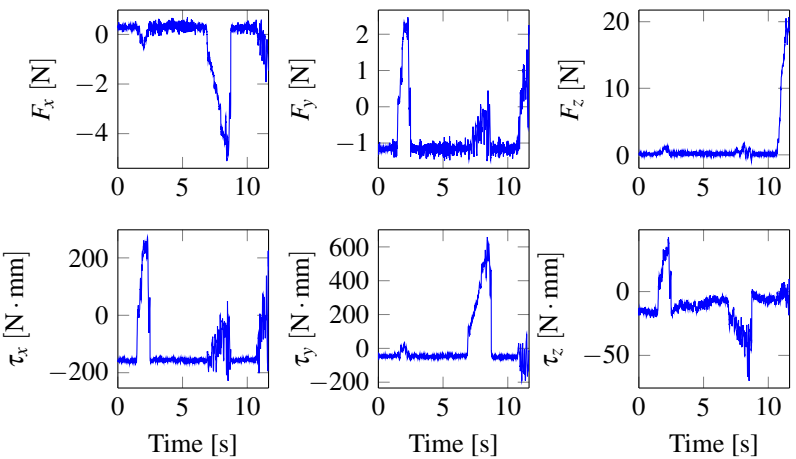

Figure 2.15 Pushing a solid block. Interaction forces and torques.

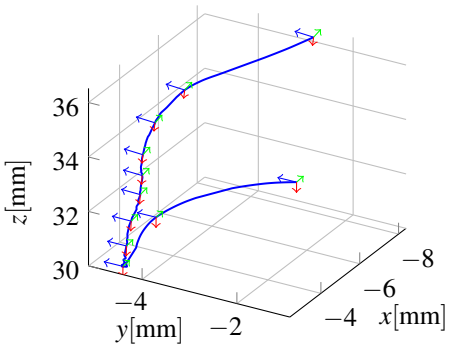

Figure 2.16 Peg-in-hole operation. The trajectory and the tool frame are illustrated.

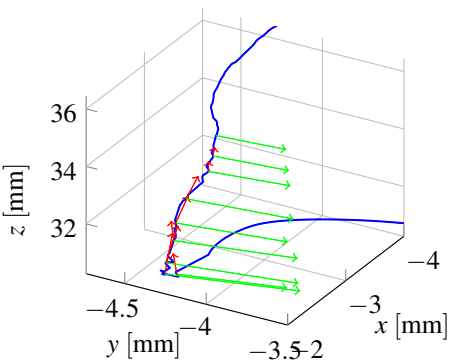

Figure 2.17 Peg-in-hole operation zoomed in. The trajectory is in blue. The red arrows are proportional to the contact forces and green arrows to the torques (the right-hand convention).

master and slave, and uncertain dynamics of operator and environment [Hokayem and Spong, 2006]. The use of an industrial robot as a slave device may increase these factors in terms of control bandwidth and nonlinear dynamics.

Although the given experimental setup provides relatively high update rate for an industrial robot, it is less than the recommended update rate of 1 kHz for haptic interfaces [Burdea and Brooks, 1996]. Therefore, it may cause an additional delay in tracking reference trajectories on the slave. There is also a time delay over the Ethernet network between the master and slave. There are some solutions to deal with the network delays; see [Kristalny and Cho, 2012] and references therein. However, they are not practical for the internal delays.

As a remedy, we employed a force scaling factor α on the haptic feedback to the operator. The scaling factor contributes to the closed-loop stability by de-

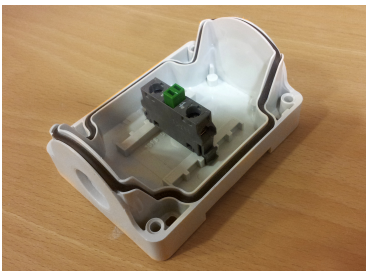

Figure 2.18 Snap-fit experiment: a part of an emergency switch is to be assembled; the dark-gray electric switch is to be fastened by snapping it to the rail of the light-gray box.

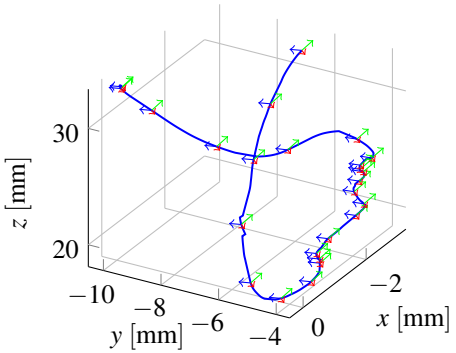

Figure 2.19 Snap-fit operation. The trajectory and the slave frame are illustrated.

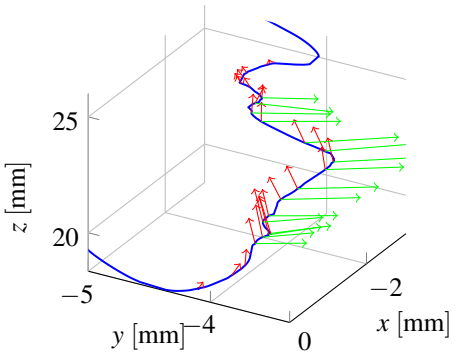

Figure 2.20 Snap-fit operation zoomed in. The trajectory is in blue. The red arrows are proportional to the contact forces and green arrows to the torques.

creasing the overall loop gain (motion of master→motion of slave→environment's reaction→force feedback). Although a lower α may disturb the perception of the environment, it provides closed-loop stability under several uncertainties discussed above. To achieve higher α, the control bandwidth and delays can be addressed by time-domain passivity [Ryu and Preusche, 2007] or other approaches.

Autonomy in operation As discussed earlier, it is not always possible to build a transparent teleoperation system. This problem is more prominent if we do not have a 4-channel teleoperation system [Tavakoli et al., 2007]. The master device used in our experiment provides only the measurement of the position and not the forces. Additionally, it is not possible to feedback the torques to the operator. To mitigate these problems, we have implemented an admittance control algorithm. By choosing a suitable wrist frame and adjusting the admittance parameters, ΔT in (2.11) can in practice fill the gaps. For example, in the snap-fit task the indirect force control strategy helped the switch glide into its final position without demanding an accurate operation from the operator. Hence, the admittance control provides a loose coupling between the master and the slave. From this perspective, we have equipped the robot with a certain degree of autonomy.

This setup could be viewed as a *cooperation* between the operator and the slave robot. This can be complemented with enforcing pure kinematic constraints on the motion of the slave robot, which can relax the operator from taking them directly into account.

For comparison, we performed the same tasks while either turning off the force controller in the robot or the force feedback to the operator. Through a series of experiments, we realized that tasks could be performed in a shorter time and with lower values of interaction forces in the cooperative scenario when using the suggested force control as compared to the other scenarios.

Extension: Imposing Kinematic Constraints

Suppose the orientation is constrained while the position in the Cartesian space could vary. We can formulate this as a minimization problem between the original desired motion and the constrained one as below

$$\begin{aligned} \underset{\bar{\dot{q}}}{\text{minimize}} \quad & \|J_t \bar{\dot{q}} - J_t \dot{q}\| \\ \text{s. t.} \quad & J_o \bar{\dot{q}} = 0. \end{aligned} \tag{2.18}$$

Here $\dot{q}$ denotes the desired motion, $\bar{\dot{q}}$ is the projected motion to the null-space of J_o, J_t is the translational part of the Jacobian and J_o is the Jacobian for the orientation.

From the constraint equation we conclude, that $\bar{\dot{q}}$ lies in the null space of J_o.

Therefore,

$$\bar{\dot{q}} = Nu \tag{2.19}$$

$$N := I - (J_o^{\dagger} J_o) \tag{2.20}$$

where u is an arbitrary vector and $J_o^{\dagger}$ is the Moore-Penrose pseudoinverse of J_o.

Now, we can remove the constraint by substituting (2.19) in the objective functional of (2.18) to get

$$\underset{u}{\text{minimize}} \quad \|J_t N u - J_t \dot{q}\| \tag{2.21}$$

The solution to (2.21) or the least-squares solution in case of additional redundancy can readily be found to be

$$u = (J_t N)^{\dagger} J_t \dot{q}. \tag{2.22}$$

Now, we substitute (2.22) back into (2.19) to get

$$\bar{\dot{q}} = M\dot{q} \tag{2.23}$$

$$M := N(J_t N)^{\dagger} J_t. \tag{2.24}$$

In the case of assisted teleoperation, we can imagine that using the projected motion, the operator can focus on controlling the position after fixing the orientation. A straightforward generalization of this approach is to partition the Jacobian matrix into rows corresponding to hard constraints (in this example J_o) and rows corresponding to soft constraints (in this example J_t). For a robot, such as YuMi [ABB Robotics, 2015; Kock et al., 2011] with 7 degrees of freedom for each arm, we may augment the Jacobian with partial or full Jacobians for other points than the end-effector (e.g., the elbow) and proceed as in the example. The approach presented here is similar to the task priority concept proposed for controlling redundant robots [Nakamura et al., 1987].

2.4 Programming Using Guarded Motion

Guarded motions provide a rich vocabulary for event-based programming common in assembly tasks. In such scenarios, a task requires the fulfillment of certain geometric or force-related constraints. Earlier works utilizing guarded motions, show the flexibility of this approach [Stolt et al., 2011; Stolt et al., 2013; Stenmark et al., 2014]. Guarded motions can be utilized as a step in a Sequential Function Chart (SFC). The SFC allows for sequencing, branching, and parallelism between different steps. In this section, we present some enhancements to a guarded motion compared to earlier works. The main contributions of this section are as follows

- Formalizing guarded motions

- Extending guarded motions with fault management and the notion of priority
- An infrastructure to facilitate programming of guarded motion parameters by demonstration

To encapsulate information required for specifying guarded motions typical in assembly scenarios, we define a guarded motion by the following elements:

- $frame_1$ and $frame_2$
- Features
- Motion Constraint
- Maintenance Constraint
- Terminal Condition
- Terminal Action
- Failure Condition
- Failure Recovery Action
- Priority

These elements are described in detail in the rest of this section.

Relations between surfaces can be described by attaching a local frame to each surface. Each frame is associated with a kinematic chain defined from a fixed reference coordinate system to the frame. Therefore, frames contain information about fixed and actuated elements of their transformation. The relation between frames can in turn be defined by the resulting feature coordinates. Basically, each feature coordinate corresponds to a degree of freedom between two frames. If we denote the transformation of $frame_1$ from a fixed coordinate system by T_1^w and the transformation of $frame_2$ by T_2^w, the feature coordinates are defined such that the resulting transformation T_f fulfills $T_2^w = T_1^w T_f$; see Fig. 2.21.

The frames in a guarded motion, $frame_1$ and $frame_2$, are specified by their kinematic chains. A kinematic chain is an ordered list of either fixed or actuated transformations. The kinematics of robots are predefined and are given a name to avoid unnecessary redefinition. Multiple consecutive fixed transformations can be combined to form a single transformation too. Thus, for each frame we define a robot or "`none`" and at most two fixed transformations such that

$$T_i^w = T_{i1} T_{ir} T_{i2}, \qquad i \in \{1,2\} \tag{2.25}$$

Here T_{i1} and T_{i2} denote the fixed transformations and T_{ir} is determined by the name of the robot. If no robot is selected, then the only required fixed transformation is the

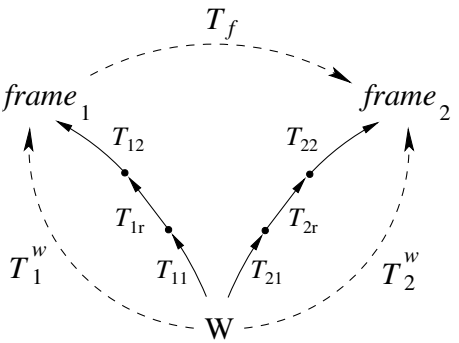

Figure 2.21 The kinematic loop concerning feature coordinates. Each transformation from world to a frame is subdivided into three transformations such that $T_i^w = T_{i1}T_{ir}T_{i2}$, where T_{ir} corresponds to an actuated chain, and T_{i1} and T_{i2} to fixed transformations.

world-to-frame transformation, which could be defined by setting either T_{i1} and T_{i2}. Otherwise, T_{i1} and T_{i2} represent world-to-base and robot-to-frame transformations, respectively.

Typically, the world-to-frame transformation corresponds to the frame attached to a workpiece and robot-to-frame to the transformation between flange and tool. Hence, the name of a robot often specifies the transformation from its base to its flange. For *Features*, a chain of named elementary transformations completing the kinematics loop is specified. The elementary transformations belong to the set of $\{x, y, z, Rot\,x, Rot\,y, Rot\,z\}$ and has the interpretation of translation and rotation about the axes of the current frame. The features are often chosen to be

$$(x, y, z, Rot\,z, Rot\,y, Rot\,x), \tag{2.26}$$

equivalent to specifying the origin of $frame_2$ by Cartesian coordinates in $frame_1$ and its orientation by Roll-Pitch-Yaw (RPY) angles. Other orders or combinations are valid as long as they completely specify the number of degrees of freedom between the two frames.

Moreover, we define feature forces in a consistent way with the feature coordinates. A feature force is a component of the forces and torques applied to $frame_1$ along a feature axis. Based on this definition, if a feature coordinate is equal to zero or the frames are in rigid contact with an object which is not accelerated, it does not matter if the sensor measurements are taken from the side of $frame_1$ or $frame_2$. Otherwise, in order to use the sensor attached to the side of $frame_2$, some assumptions about the object being manipulated need to be made.

Motion Constraint specifies a set of constraints regarding the variation of one or several feature coordinates as a function of time. Constant velocity is the simplest

type of motion with two parameters, a feature axis and a velocity. Depending on the axes used and the specification of the features, constant velocity motion can result in a variety of motions such as linear, circular, spiral, etc. A circular motion around an axis can be achieved by adding an offset to the robot-frame transformation. For example, if we use the typical specification of the features in (2.26), a circular motion around the z-axis is obtained by the displacement of the robot-frame transformation with the desired radius in a plane perpendicular to this axis. Setting a motion constraint for the axis corresponding to $Rot\,z$ leads to the desired motion. A spiral motion can be achieved by combination of two constant velocity motions.

In this approach, it is quite important to explicitly define the constraints. For example, if we specify a planar motion by setting motion constraints for the x- and y-axes, the motion in z-direction and the orientation are still free. If this is not desired, a *Maintenance Constraint* can be defined for the constraint that is supposed to be maintained under the motion. Maintenance constraints include position, orientation, impedance, or force constraints. A maintenance constraint as well as a motion constraint specifies a value for a controlled quantity. Each controlled quantity in turn depends on a controller.

Terminal Condition defines a set of conditions that determines a successful completion of a guarded motion and results in a state transition to the *Terminal Action*. Since there is no guarantee that a guarded motion successfully terminates, the condition for detecting a fault scenario is specified by the *Failure Condition* and the *Failure Recovery Action* is taken if the condition is satisfied. The conditions are expressed with relational operators, e.g., $==$, $>$, and $<$ and a mathematical expression based on the measured or estimated quantities, e.g., feature force, feature coordinate, sensor readings, and time. Note that both terminal and failure actions are steps outside of a guarded motion in an SFC and can be other guarded motions or generic commands such as gripper open/close, turn on/off spindle, or commands to the robot's native controller (e.g., RAPID program).

We have adopted JGrafchart for the implementation of SFC in our work [Årzén et al., 2002; Theorin, 2013]. Hence, a guarded motion is a macro step in JGrafchart. Following the execution model of JGrafchart, the terminal action is activated as soon as the terminal condition is satisfied. The guarded motions which run in parallel share the same terminal action, which is activated when all the terminal conditions are satisfied. Any sequence of guarded motions could be united to form a composite guarded motion. This allows for hierarchical design and hiding details. For example, if for a special system after each motion a small amount of back-off is required, both the main guarded motion and the terminal correction could be combined to form a new composite guarded motion. The frames are by default inherited from the parent to avoid redefinition.

There can be several motion and maintenance constraints in one guarded motion. Additional constraints can be imposed by parallel guarded motions. This is useful when a single guarded motion only partially constrains the motion and extra degrees of freedom need to be constrained using other frames, e.g., in a dual-arm

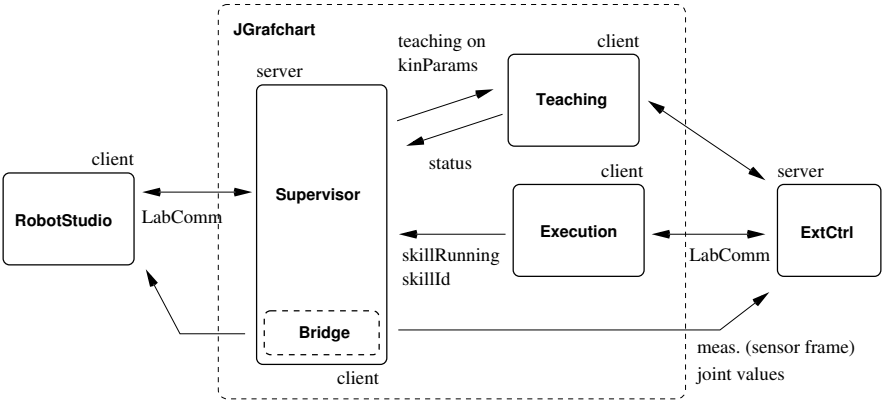

Figure 2.22 Architecture of the system for programming using guarded motion

assembly task. *Priorities* affect the way the constraints are evaluated in parallel guarded motions. Motion constraints with lower priority are evaluated subject to higher priority constraints. For example, if a peg-in-hole operation needs to be done and there is redundancy in the task, a parallel guarded motion with lower priority can move the elbow away from a potential danger zone without affecting the task. After taking into account priorities, over- and under-constrained motions are managed automatically by constraint handling according to the iTaSC framework [De Schutter et al., 2007]. For more advanced constraint handling, we consider using a higher level scheduler.

The guarded motions presented in this section are partially implemented [Stolt and Linderoth, 2011; Ghazaei and Hofele, 2014]. Additionally, using the setups described in Sections 2.2 and 2.3, we have developed a robotic architecture that facilitates programming using guarded motions. In this system, there are two modes of operation, teaching and execution. In the teaching mode, the architecture allows the flow of information from sensors and controller to the engineering tool ABB RobotStudio, where the operator creates a new task. Each task is a composition of guarded motions or other tasks. An SFC is generated for the task which is loaded into JGrafchart [Theorin, 2013] and executed on the robot. The supervisor block depicted in Fig. 2.22 is responsible for switching between teaching and execution modes and receives feedback from measurements and the status of the robot. The teaching block is another SFC which implements the logic of various teaching modules. The execution block is loaded dynamically as soon as the code generation by RobotStudio is finished.

An example of an SFC for a task is shown in Fig. 2.23. The task is to move downwards until a pin is detected and then pick it up. The representation of the pick-up task in RobotStudio is on the left side. “Search -z” is a guarded motion

with constraint on the motion in negative z direction and two terminal conditions; either a force threshold or a guard distance. A failure situation can be detected by the fault condition. In this case, the fault condition is the same as the guard distance (not shown in the figure).

In this section, we took the first step to formalize a domain specific language based on guarded motions for robots. We put forward ideas that enhance the guarded motions for a broader application in assembly tasks. We proposed a combination of sequential programming and some behavioral elements such as task priorities and handling of simultaneous constraints. The parallel guarded motions and consistent feature forces make scenarios with multiple robots easier to define and manage. Integration of the fault detection directly into the guarded motion together with hierarchical structure allows easier development and maintenance of robust skills.

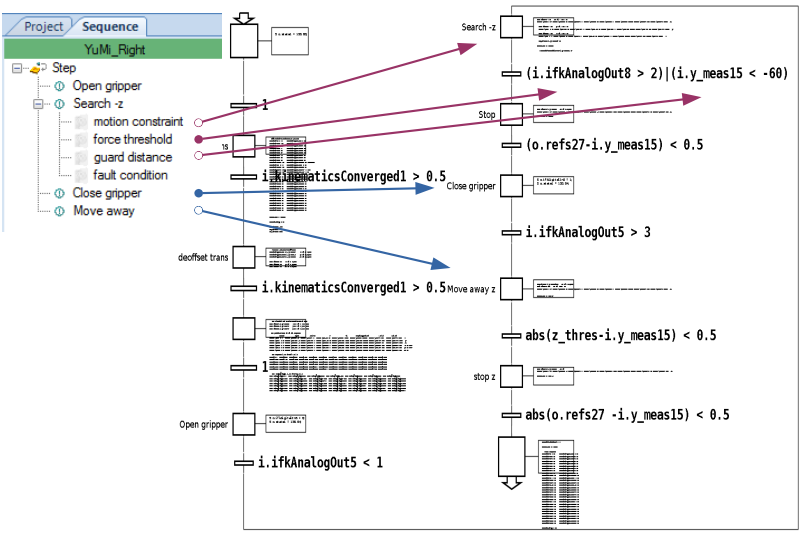

Figure 2.23 An example of a task implemented in JGrafchart [Theorin, 2013] on the right and the representation in RobotStudio on the left: picking up a pin by searching in negative z direction with respect to the feature coordinate system.

2.5 Conclusions

The free-floating mode provides a robust lead-through teaching interface to a robot. In this mode, the internal feedback loop of the robot is disabled. The robot is free-floating by compensating for the gravity and friction. As a result of an open-loop control the stability of this approach, when the robot is in contact with stiff materials, is unparalleled. This approach was successfully implemented on the ABB YuMi robot [ABB Robotics, 2015].

To generate trajectories for assembly tasks, the detection of contacts by the operator plays an important role. The task can be simplified by providing force feedback to the operator. We have introduced the notion of slave-assisted teleoperation by means of force control. The active compliance on the slave side together with a tele-admittance strategy gives the operator more freedom to manipulate objects with lower risk of damaging a workpiece. Moreover, we have introduced a structured way to define required coordinate frames for teleoperation. This facilitates a quick setup of multi-degrees of freedom teleoperation and customizing it according to the preference of an operator.

The developed setups in this chapter aimed for collecting both position and force data for assembly tasks. The data is used for providing the parameters of guarded motions. We have extended guarded motions with the notion of priority and fault management. Furthermore, we can imagine processing the data for learning purposes. The force/torque measurement provides valuable information for triggering segmentation and defining tolerances.

3

Instantaneous Trajectory Generation Based on a Motion Template

3.1 Introduction

Consider a robot assignment to pick different objects from a conveyor belt. The best gripping is achieved when the gripper has a relative velocity of zero with respect to the object. However, the speed of the conveyor belt can vary due to varying loads. The current estimates of the position and velocity as well as an estimate of the arrival of the object are provided by a camera system. Moreover, to increase the service life of the robot and not to excite its vibration modes, the motion has to follow a predefined trajectory profile.

The main motivation for this chapter is to provide answers to the following questions: how to continuously update the trajectories for moving targets and how to ensure that there is a smooth transition between trajectory generation and tracking. This leads us to the formulation of the trajectory generation as a dynamical system with a trajectory-generation controller. In contrast to mathematically designed or optimal trajectories purely as a function of time, we regard a trajectory as an output of a dynamical system.

For time-optimal problems, methods for closed-loop trajectory generation have already been considered [Kröger and Wahl, 2010]. In this chapter, we propose a solution for a fixed-time problem. The fixed-time problems are of importance when a less aggressive strategy than a minimum-time solution is sufficient. Moreover, fixed-time motions lend themselves to the coordination between several entities and scheduling of tasks.

Assume that the motion of a robot is described by

$$\begin{aligned} \dot{x} &= f(t,x,\hat{r}),\ x(0) = x_0 \\ y &= g(t,x) \end{aligned} \tag{3.1}$$

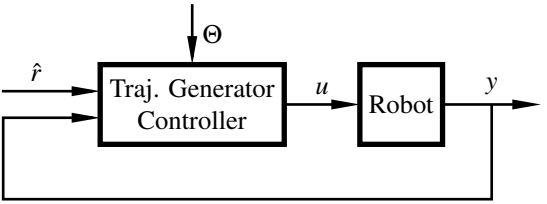

Figure 3.1 Block diagram of trajectory generation.

where y represents the position of the robot and $\hat{r}(t)$ the effect of exogenous inputs. Given the structure depicted in Fig. 3.1, we wish to generate a control signal $u = k(\hat{r}, x, t, \Theta)$, which results in the desired motion. In the control law, Θ denotes a set of parameters and $\hat{r}$ correspond to a high-level reference signal. The high-level reference signal is typically a set-point related to the future value of the target. For example, the current desired reference can be the position of an object on the conveyor belt in d seconds ahead in time. If $r(t)$ denotes the current position, we have

$$\hat{r}(t) = r(t+d). \tag{3.2}$$

In general, the motion patterns can be acquired via human demonstration [Khansari-Zadeh and Billard, 2011] or designed by a mathematical formula [Paul, 1979; Taylor, 1979]. We call these patterns motion templates. Specifically, we focus on the one-dimensional problem and study a fifth-order polynomial template, i.e., a trajectory generated by a fifth-order polynomial traveling one unit of distance in one unit of time. The template can mathematically be expressed as

$$y = p_0 + (p_0 - 1)(-10(t-t_0)^3 + 15(t-t_0)^4 - 6(t-t_0)^5), \tag{3.3}$$

where p_0 and t_0 denote the initial position and initial time, respectively. All the methods described in this chapter concern the generalization of this template. Nevertheless, the main focus is to present a closed-loop solution to trajectory generation. Simple operations such as time and coordinate scaling are introduced. We use these techniques later to extend our template to the whole workspace while respecting certain constraints. As a generalization of this template, we consider a larger class of trajectories that are obtained by a minimum-jerk model.

Since having hard constraints on the final time poses certain robustness issues when there is a disturbance, we propose three methods to relax this constraint. In the first method, a smooth transition between a finite-horizon problem and an infinite horizon problem is suggested by limiting the minimum remaining time. In the second method, the remaining time is reset whenever it is impossible to meet the deadline given the constraints. In the third method, the explicit dependency on time and the duration is removed. Instead, a new set of parameters is derived based on the

scaling techniques introduced in Section 3.2. Irrespective of the method used for relaxing the final time constraint, all of the methods can reproduce an approximation of the template.

3.2 Basic Operations on Trajectories

Trajectories can be generalized by applying a set of operations. The simplest one is translation, which implies adding an offset to each coordinate. In the planar or three-dimensional case, it is possible to consider rotation [Paul, 1979]. In this section, we consider a few operations that allow us to extend a trajectory to a larger workspace while preserving some desired properties of the original trajectory, such as the average velocity.

Time Scaling

Here, we consider the change of the system (3.1) under scaling of time. Dynamic scaling [Hollerbach, 1984; Dahl and Nielsen, 1990] can be considered as a special case, when time scaling is applied to the dynamic equations of a robot. Let us introduce a new variable for time, denoted by $\tilde{t}$, which is an increasing and differentiable function of time

$$\tilde{t} := \alpha(t) \Rightarrow t = \alpha^{-1}(\tilde{t}). \tag{3.4}$$

The impact of this change of variable on the output and its derivatives is as follows,

$$\tilde{y}(\tilde{t}) = y(t) \tag{3.5}$$

$$\tilde{y}'(\tilde{t}) = \frac{\dot{y}(t)}{\dot{\alpha}(t)} \tag{3.6}$$

$$\tilde{y}''(\tilde{t}) = \frac{\ddot{y}(t)}{\dot{\alpha}^2(t)} - \frac{\ddot{\alpha}(t)}{\dot{\alpha}^3(t)}\dot{y}(t) \tag{3.7}$$

$$\tilde{y}'''(\tilde{t}) = \frac{\dddot{y}(t)}{\dot{\alpha}^3(t)} - 3\frac{\ddot{\alpha}(t)}{\dot{\alpha}^4(t)}\ddot{y}(t) + \left(3\frac{\ddot{\alpha}^2(t)}{\dot{\alpha}^5(t)} - \frac{\dddot{\alpha}(t)}{\dot{\alpha}^4(t)}\right)\dot{y}(t) \tag{3.8}$$

where $\{\cdot\}' := d\{\cdot\}/d\tilde{t}$ and $\{\dot{\cdot}\} := d\{\cdot\}/dt$.

As an example, we can choose $\alpha(t) = \alpha$ to be constant. Then, $\tilde{t} = \alpha t$ and

$$\tilde{y}(\tilde{t}) = y\left(\frac{\tilde{t}}{\alpha}\right) \tag{3.9}$$

$$\tilde{y}'(\tilde{t}) = \alpha^{-1}\dot{y}\left(\frac{\tilde{t}}{\alpha}\right) \tag{3.10}$$

$$\tilde{y}''(\tilde{t}) = \alpha^{-2}\ddot{y}\left(\frac{\tilde{t}}{\alpha}\right) \tag{3.11}$$

$$\tilde{y}'''(\tilde{t}) = \alpha^{-3}\dddot{y}\left(\frac{\tilde{t}}{\alpha}\right) \tag{3.12}$$

Coordinate Scaling

Now consider the scaled output $\tilde{y}$. The impact of this change of variable on the output $\tilde{y} := \beta(y)$ is as follows

$$\tilde{y} = \beta(y) \tag{3.13}$$
$$\dot{\tilde{y}} = \beta'(y)\dot{y} \tag{3.14}$$
$$\ddot{\tilde{y}} = \beta''(y)\dot{y}^2 + \beta'(y)\ddot{y} \tag{3.15}$$
$$\dddot{\tilde{y}} = \beta'''(y)\dot{y}^3 + 3\beta''(y)\ddot{y}\dot{y} + \beta'(y)\dddot{y} \tag{3.16}$$

Similar to the time scaling, we can choose $\beta(y) = \beta$ to be constant. Then, $\tilde{y} = \beta y$ and

$$\tilde{y}(t) = \beta y(t) \tag{3.17}$$
$$\dot{\tilde{y}}(t) = \beta \dot{y}(t) \tag{3.18}$$
$$\ddot{\tilde{y}}(t) = \beta \ddot{y}(t) \tag{3.19}$$
$$\dddot{\tilde{y}}(t) = \beta \dddot{y}(t) \tag{3.20}$$

By combining the coordinate scaling and time scaling operations, it is possible to parametrize the classes of position-, velocity-, and acceleration-preserving transformations. Considering (3.17)–(3.20) and (3.9)–(3.12) corresponding to the constant scaling factors, we can choose $\beta = \alpha$, $\beta = \alpha^2$, or $\beta = \alpha^3$ to achieve average velocity-, average acceleration-, or average jerk-preserving transformations, respectively. Note that in these cases, it is possible to reach all the positions in the workspace by changing only one parameter.

Coordinate Transformation

Assume that we are given an autonomous dynamical system whose states go from any initial condition to zero. Considering the servo problem to follow an arbitrary reference [Glad and Ljung, 2011], it is possible to use the transformation $e(t) = r(t) - y(t)$. This implies that the trajectory generation is composed of a tracking term $r(t)$ plus a non-linear shaping function $e(t)$. This strategy is advantageous as long as the current value of the reference (target) provides some useful information for the robot (tracker). To clarify this point, rewrite the transformation as

$$y(t) = r(t) - e(t) \tag{3.21}$$

where $r(t)$ denotes the target position and $e(t)$ the shaping function. The shaping function satisfies

$$e(t_0) = r(t_0) - y(t_0) \tag{3.22}$$
$$e(t_f) = 0. \tag{3.23}$$

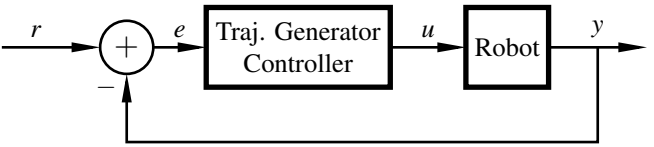

Figure 3.2 Modified block diagram of trajectory generation.

Note that (3.21) does not necessarily specify a causal relation between $e(t)$ and $y(t)$. Thus, both of these signals could be caused by $r(t)$ as in Fig. 3.2. Additionally, in this setup we make use of the current value of $r(t)$ instead of $\hat{r}(t) = r(t+d)$, which is an estimated value in the future. The underlying assumption for the modified model in Fig. 3.2 compared to Fig. 3.1 is that the generated trajectory depends only on $r(t) - y(t)$.

As an example, consider the robot is supposed to reach position y_f, velocity v_f, and acceleration a_f from y_0, v_0, and a_0 with duration d. Firstly, note that the continuity of higher derivatives is of no concern since no explicit constraint is given for them. Secondly, from the discussion above, we can equivalently describe the problem as reaching a target that has a constant acceleration a_f and reaches y_f with velocity v_f in d seconds. The target motion can be described by the following equations

$$r_1(t) = y_f + v_f(t-d) + \frac{1}{2}a_f(t-d)^2 \tag{3.24}$$

$$r_2(t) = v_f + a_f(t-d) \tag{3.25}$$

$$r_3(t) = a_f. \tag{3.26}$$

On the other hand, the motion of the robot can be derived by Eq. (3.21) to be

$$y(t) = y_f + v_f(t-d) + \frac{1}{2}a_f(t-d)^2 - e_1(t) \tag{3.27}$$

$$v(t) = v_f + a_f(t-d) - e_2(t) \tag{3.28}$$

$$a(t) = a_f - e_3(t). \tag{3.29}$$

By setting $t = 0$ and $t = d$ in Eq. (3.27)–(3.29), we obtain the following relationships

$$e_1(0) = y_f - v_f d + \frac{1}{2}a_f d^2 - y_0 \tag{3.30}$$

$$e_2(0) = v_f - a_f d - v_0 \tag{3.31}$$

$$e_3(0) = a_f - a_0 \tag{3.32}$$

$$e_1(d) = 0 \tag{3.33}$$

$$e_2(d) = 0 \tag{3.34}$$

$$e_3(d) = 0. \tag{3.35}$$

The conditions can equivalently be written as

$$e_1(0) = y_0^{\text{target}} - y_0^{\text{robot}} \tag{3.36}$$
$$e_2(0) = v_0^{\text{target}} - v_0^{\text{robot}} \tag{3.37}$$
$$e_3(0) = a_0^{\text{target}} - a_0^{\text{robot}} \tag{3.38}$$
$$e_1(d) = 0 \tag{3.39}$$
$$e_2(d) = 0 \tag{3.40}$$
$$e_3(d) = 0. \tag{3.41}$$

3.3 Minimum-Jerk Model

Using piece-wise polynomials is a common approach to trajectory generation [Paul, 1972; Paul, 1979; Lin et al., 1983; Taylor, 1979]. Specially, a fifth-order polynomial allows for specifying both initial and final values of position, velocities, and accelerations. Fifth-order polynomials have attracted researchers from a different perspective too. Namely, they show up as the solution to a simplistic kinematic description of human-generated trajectories by a *minimum-jerk* model [Flash, 1987]. An important feature of this model is the ability to predict the bell-shaped velocity profile of the arm movement. In a planar case, the cost functional to be minimized is [Flash, 1987]:

$$C = \frac{1}{2} \int_0^{t_f} \left(\dddot{X}^2 + \dddot{Y}^2 \right) dt, \tag{3.42}$$

where X and Y represent the coordinates, dots denote time derivative, and t_f the duration of the movement. First, note that since the X and Y coordinates in (3.42) are independent, it is possible to solve two one-dimensional optimization problems instead. Using the variational principle, the solution is easily found to be a fifth-order polynomial [Shadmehr, 2005]. Here, we adopt the control-problem formulation in order to derive the result. This formulation gives us new insights for trajectory generation.

For minimizing the jerk, a triple integrator can represent the dynamical system for each direction as

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u \tag{3.43}$$
$$y = x_1 \tag{3.44}$$

or equivalently,

$$\dot{x} = f(t,x,u) = \begin{pmatrix} x_2 \\ x_3 \\ u \end{pmatrix}. \tag{3.45}$$

According to the Pontryagin maximum principle [Pontryagin et al., 1962; Liberzon, 2011],

$$\dot{x}^* = H_p(x^*, u^*, p^*, p_0^*) \tag{3.46}$$
$$\dot{p}^* = -H_x(x^*, u^*, p^*, p_0^*) \tag{3.47}$$
$$x^*(t_0) = x_0, \qquad x^*(t_f) = x_f \tag{3.48}$$

Here, H denotes the Hamiltonian, the subscripts denote partial derivatives with respect to the given variable, x and p are the states and the costates, respectively, t_0 and t_f denote the initial and final time, respectively, and variables with star denote the optimal solution. For this optimization problem, we have the Hamiltonian

$$H(x,u,p,p_0) = \langle p, f(x,u) \rangle + p_0 L(x,u) \tag{3.49}$$
$$= \begin{pmatrix} p_1 & p_2 & p_3 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ u \end{pmatrix} + p_0 u^2 \tag{3.50}$$

By the partial differentiation of H with respect to u, we find the extremum as below,

$$\frac{\partial H}{\partial u} = 2p_0 u + p_3 = 0 \tag{3.51}$$
$$\Rightarrow u^* = -\frac{p_3}{2p_0}. \tag{3.52}$$

Consequently, the Hamiltonian is

$$H(x^*, u^*, p^*, p_0^*) = p_1 x_2 + p_2 x_3 - \frac{p_3^2}{4p_0}, \tag{3.53}$$

where

$$\dot{p}_1 = -H_{x_1} = 0 \tag{3.54}$$
$$\dot{p}_2 = -H_{x_2} = -p_1 \tag{3.55}$$
$$\dot{p}_3 = -H_{x_3} = -p_2. \tag{3.56}$$

These equations combined with (3.52) result in

$$u = k_1 t^2 + k_2 t + k_3 \tag{3.57}$$

Integrating the control signal three times results in x_1 which is apparently a fifth-order polynomial. By matching the initial and final conditions, we obtain

$$\begin{aligned} y = x_1 &= p_0 + (p_0 - p_f)(-10t_n^3 + 15t_n^4 - 6t_n^5) \\ &+ v_0 dt_n + v_0 dt_n(-6t_n^2 + 8t_n^3 - 3t_n^4) - v_f dt_n(4t_n^2 - 7t_n^3 + 3t_n^4) \\ &+ \frac{a_0}{2} d^2 t_n^2 + \frac{a_0}{2} d^2 t_n^2(-3t_n + 3t_n^2 - t_n^3) - \frac{a_f}{2} d^2 t_n^2(-t_n + 2t_n^2 - t_n^3) \end{aligned} \tag{3.58}$$

$$\begin{aligned} \dot{y} = x_2 &= \frac{p_0 - p_f}{d}(-30t_n^2 + 60t_n^3 - 30t_n^4) \\ &+ v_0 + v_0(-18t_n^2 + 32t_n^3 - 15t_n^4) - v_f(12t_n^2 - 28t_n^3 + 15t_n^4) \\ &+ a_0 dt_n + \frac{a_0}{2} dt_n(-9t_n + 12t_n^2 - 5t_n^3) - \frac{a_f}{2} d^2 t_n(-3t_n + 8t_n^2 - 5t_n^3) \end{aligned} \tag{3.59}$$

$$\begin{aligned} \ddot{y} = x_3 &= \frac{p_0 - p_f}{d^2}(-60t_n + 180t_n^2 - 120t_n^3) \\ &+ \frac{v_0}{d}(-36t_n + 96t_n^2 - 60t_n^3) - \frac{v_f}{d}(24t_n - 84t_n^2 + 60t_n^3) \\ &+ a_0 + a_0(-9t_n + 18t_n^2 - 10t_n^3) - a_f(-3t_n + 12t_n^2 - 10t_n^3), \end{aligned} \tag{3.60}$$

and the control signal is

$$\begin{aligned} \dddot{y} = u &= \frac{p_0 - p_f}{d^3}(-60 + 360t_n - 360t_n^2) \\ &+ \frac{v_0}{d^2}(-36 + 192t_n - 180t_n^2) - \frac{v_f}{d^2}(24 - 168t_n + 180t_n^2) \\ &+ \frac{a_0}{d}(-9 + 36t_n - 30t_n^2) - \frac{a_f}{d}(-3 + 24t_n - 30t_n^2) \end{aligned} \tag{3.61}$$

where $t_n := t/d$ is the normalized time with respect to the duration.

Let us now consider the Hamilton-Jacobi-Bellman (HJB) equation [Bellman and Kalaba, 1965; Liberzon, 2011]

$$-V_t(t,x) = \inf_{u \in U} \{L(t,x,u) + \langle V_x(t,x), f(t,x,u) \rangle\}, \tag{3.62}$$

with the cost function J and the value function V defined as

$$J(t,x,u) = \int_t^{t_f} L(s,x(s),u(s))\,ds + K(x(t_f)) \tag{3.63}$$

$$V(t,x) := \inf_{u[t,t_f]} J(t,x,u). \tag{3.64}$$

Here, $U \subset \mathbb{R}$ defines the control set and $K(\cdot)$ denotes the terminal cost.

For the minimum-jerk problem, we have

$$L(t,x,u) = u^2, \qquad K(x(t_f)) = 0 \tag{3.65}$$

$$-V_t(t,x) = \inf_{u \in U} \{u^2 + \langle V_x(t,x), f(t,x,u) \rangle\} \tag{3.66}$$

$$= \min_{u \in \mathbb{R}} \{u^2 + V_{x_1} x_2 + V_{x_2} x_3 + V_{x_3} u\} \tag{3.67}$$

where we use the notation

$$V_{x_k} := \frac{\partial V}{\partial x_k}. \tag{3.68}$$

The optimum is achieved where

$$\frac{\partial (u^2 + V_{x_1} x_2 + V_{x_2} x_3 + V_{x_3} u)}{\partial u} = 0 \Rightarrow \tag{3.69}$$

$$u = -\frac{V_{x_3}}{2}. \tag{3.70}$$

Therefore,

$$-V_t(t,x) = -\frac{V_{x_3}^2}{4} + V_{x_1} x_2 + V_{x_2} x_3. \tag{3.71}$$

As a result of the application of the maximum principle, we obtained an expression for $\dddot{y}$ along the optimal path. For the sake of simplicity, we consider the final state to be the origin. Therefore, Eq. (3.61) can be written as

$$\begin{aligned} \dddot{y} =& \frac{y_0}{d^3}(-60 + 360 t_n - 360 t_n^2) \\ &+ \frac{v_0}{d^2}(-36 + 192 t_n - 180 t_n^2) \\ &+ \frac{a_0}{d}(-9 + 36 t_n - 30 t_n^2). \end{aligned} \tag{3.72}$$

Additionally, we consider that the total duration is 1. Considering the value function in (3.64), i.e., the cost to go, and using the definition of the cost functional in (3.63), d has to be substituted with the remaining time $1-t$ and t_n with the normalized elapsed time $(s-t)/(1-t)$. This results in

$$V(t,x) = \int_t^1 \dddot{y}^2(s)\, ds. \tag{3.73}$$

It is straightforward to verify that this value function satisfies the HJB equation. Accordingly, from (3.70) we derive

$$u = -\frac{V_{x_3}^2}{2} = -\left(60\frac{x_1}{(1-t)^3} + 36\frac{x_2}{(1-t)^2} + 9\frac{x_3}{1-t}\right). \tag{3.74}$$

This reformulation gives us a feedback law for generating the trajectory. The feedback signal is linear in the states, but non-linear with respect to the time. Note that the same result can be obtained from (3.61). Interpreting the current point as the new initial point, we can set $t_n = 0$ to obtain

$$u = \dddot{y} = -\left(60\frac{x_1 - y_f}{(d-t)^3} + \frac{36x_2 + 24v_f}{(d-t)^2} + \frac{9x_3 - 3a_f}{d-t}\right). \tag{3.75}$$

Equation (3.75) gives us a solution for arbitrary initial and final points and duration d.

An obvious issue with this model is that it is quite sensitive to the errors in the states when the time approaches d. Without loss of generality, assume there is some noise $\varepsilon(t)$ in the velocity measurement. The closed-loop system in this case is

$$\dddot{y} = -\left(60\frac{y}{(d-t)^3} + 36\frac{\dot{y}}{(d-t)^2} + 9\frac{\ddot{y}}{d-t}\right) + 36\frac{\varepsilon(t)}{(d-t)^2}. \tag{3.76}$$

As t approaches d, a very small noise can blow up the control signal. A remedy to this problem would be switching to an infinite-horizon problem when $\tilde{d} = d - t$ becomes small. In the following, we show that this transition can be done smoothly.

Consider the diagram of the closed-loop poles of the system for a fixed t in Fig. 3.3. The characteristic equation of the closed-loop system for a fixed remaining time $\tilde{d}$ is

$$\tilde{d}^3 s^3 + 9\tilde{d}^2 s^2 + 36\tilde{d}s + 60 = 0. \tag{3.77}$$

If p is a solution for $\tilde{d} = 1$ in (3.77), then $p/\tilde{d}$ is a solution for a given $\tilde{d}$. Therefore, $\arg(p)$ is independent of the remaining time while the poles move toward infinity as the remaining time approaches zero.

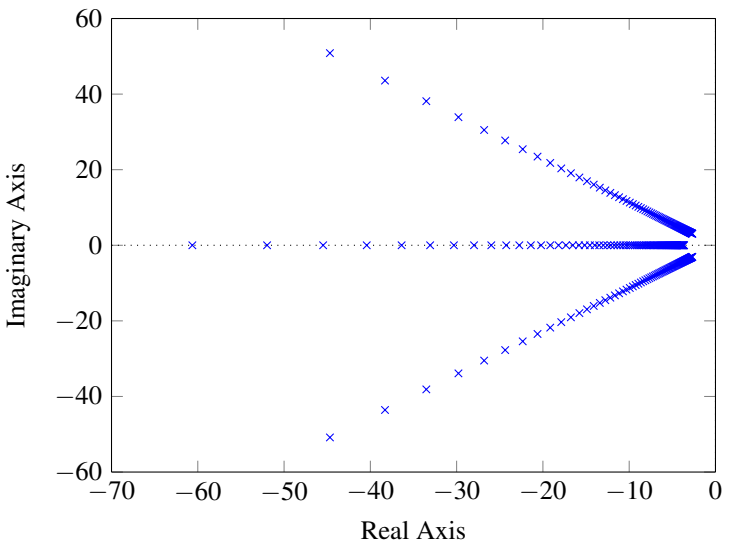

Figure 3.3 Eigenvalues for varying values of the remaining time $\tilde{d} = [1 : -0.01 : 0.06]$.

We can further rewrite the characteristic equation as

$$(s+a\omega)(s^2+2\zeta\omega s+\omega^2)=0 \tag{3.78}$$

where $(\omega\tilde{d})^2 = 12-2\sqrt[3]{32}+6\sqrt[3]{3}\approx 16.493$, $\zeta=(6-\sqrt[3]{32}+\sqrt[3]{3})/2\omega\tilde{d}\approx 0.66$, and $a=(3+\sqrt[3]{32}-\sqrt[3]{3})/\omega\approx 0.896$. Given the above values and the knowledge of the noise in the system, it is possible to find a minimum acceptable value for $\tilde{d}$. Therefore, a smooth transition is obtained by saturating $\tilde{d}$ from below.

3.4 Generalization of Template

Consider the system obtained by normalizing time to 1 and the final state to the origin

$$\dot{x}_1 = x_2 \tag{3.79}$$
$$\dot{x}_2 = x_3 \tag{3.80}$$
$$\dot{x}_3 = -\left(60\frac{x_1}{(1-t)^3}+36\frac{x_2}{(1-t)^2}+9\frac{x_3}{1-t}\right). \tag{3.81}$$

The aim of this section is to generalize the motion trajectories generated by this system to the whole workspace and to different timing requirements. Although (3.75) is already in its general form, we consider using the coordinate transformation introduced in Section 3.2 instead. In doing this we have several purposes. In the first place, the coordinate transformation allows us to use the current value of the reference signal instead of its predicted value. Secondly, we can show how the scaling methods can be applied to a dynamical system. We will take advantage of this in Section 3.6 where the scaling provides a new parametrization for the trajectory generation. Thirdly, we apply the coordinate transformation to our solution of the HJB equation to show that Eq. (3.75) indeed gives us the correct feedback law if $\hat{r}(t)$ is preferred.

First, we consider time scaling $\tilde{t}=\alpha t$. Let us define

$$\tilde{x}_1(\tilde{t}) = \alpha^0 x_1(t) \tag{3.82}$$
$$\tilde{x}_2(\tilde{t}) = \alpha^{-1} x_2(t) \tag{3.83}$$
$$\tilde{x}_3(\tilde{t}) = \alpha^{-2} x_3(t). \tag{3.84}$$

This results in

$$\tilde{x}_1' = \tilde{x}_2 \tag{3.85}$$
$$\tilde{x}_2' = \tilde{x}_3 \tag{3.86}$$
$$\tilde{x}_3' = -\alpha^{-3}\left(60\frac{\tilde{x}_1}{\left(1-\frac{\tilde{t}}{\alpha}\right)^3}+36\frac{\alpha\tilde{x}_2}{\left(1-\frac{\tilde{t}}{\alpha}\right)^2}+9\frac{\alpha^2\tilde{x}_3}{1-\frac{\tilde{t}}{\alpha}}\right). \tag{3.87}$$

The choice of new variables is done so that the first two differential equations remain the same.

Since the variable names do not matter, we can rewrite the equations as below

$$\dot{x}_1 = x_2 \tag{3.88}$$
$$\dot{x}_2 = x_3 \tag{3.89}$$
$$\dot{x}_3 = -\left(60\frac{x_1}{(\alpha-t)^3} + 36\frac{x_2}{(\alpha-t)^2} + 9\frac{x_3}{\alpha-t}\right). \tag{3.90}$$

In this case it is not a surprise that α amounts to the total time. Note that the new initial conditions are calculated according to (3.82)-(3.84).

Now, we consider coordinate scaling. We define

$$\tilde{x}_1(t) = \beta x_1(t) \tag{3.91}$$
$$\tilde{x}_2(t) = \beta x_2(t) \tag{3.92}$$
$$\tilde{x}_3(t) = \beta x_3(t). \tag{3.93}$$

With these changes of variables,

$$\dot{\tilde{x}}_1 = \tilde{x}_2 \tag{3.94}$$
$$\dot{\tilde{x}}_2 = \tilde{x}_3 \tag{3.95}$$
$$\dot{\tilde{x}}_3 = -\left(60\frac{\tilde{x}_1}{(\alpha-t)^3} + 36\frac{\tilde{x}_2}{(\alpha-t)^2} + 9\frac{\tilde{x}_3}{\alpha-t}\right). \tag{3.96}$$

Note that all the equations remain unchanged and the only change is in the initial conditions. This is not a surprise since the control law derived is valid for any initial condition, i.e., any coordinate scaling. Thus, the control signal considering the effect of both time and coordinate scaling is

$$u = -\left(60\frac{\tilde{x}_1}{(\alpha-t)^3} + 36\frac{\tilde{x}_2}{(\alpha-t)^2} + 9\frac{\tilde{x}_3}{\alpha-t}\right). \tag{3.97}$$

Now, we show that Eq. (3.75) can be derived from Eqs. (3.85)–(3.87). Note that the coordinate transformation introduced in Section 3.2 does affect jerk (its third time derivative is equal to zero) and hence the cost functional in the special case of minimum-jerk is invariant under this transformation. Therefore, the solution after the transformation is optimal for the new initial and final states.

According to Fig. 3.2, the inputs to trajectory generator controller is $e(t)$. Therefore, by changing α to d from (3.85)–(3.87) we have

$$\dot{e}_1 = e_2 \tag{3.98}$$
$$\dot{e}_2 = e_3 \tag{3.99}$$
$$\dot{e}_3 = -\left(60\frac{e_1}{(d-t)^3} + 36\frac{e_2}{(d-t)^2} + 9\frac{e_3}{d-t}\right). \tag{3.100}$$

By renaming $y(t)$, $v(t)$, and $a(t)$ in Eqs. (3.27)–(3.29) to x_1, x_2, and x_3, respectively, we obtain

$$x_1(t) = y_f + v_f(t-d) + \frac{1}{2}a_f(t-d)^2 - e_1(t) \tag{3.101}$$
$$x_2(t) = v_f + a_f(t-d) - e_2(t) \tag{3.102}$$
$$x_3(t) = a_f - e_3(t). \tag{3.103}$$

The time derivatives of (3.101)–(3.103) are

$$\dot{x}_1 = v_f + a_f(t-d) - \dot{e}_1 \tag{3.104}$$
$$\dot{x}_2 = a_f - \dot{e}_2 \tag{3.105}$$
$$\dot{x}_3 = -\dot{e}_3, \tag{3.106}$$

respectively. These equations can be rewritten using (3.98)–(3.100) to get

$$\dot{x}_1 = v_f + a_f(t-d) - e_2 = x_2 \tag{3.107}$$
$$\dot{x}_2 = a_f - e_3 = x_3 \tag{3.108}$$
$$\dot{x}_3 = -\left(60\frac{e_1}{(d-t)^3} + 36\frac{e_2}{(1-t)^2} + 9\frac{e_3}{d-t}\right) \tag{3.109}$$
$$-\left(60\frac{x_1 - y_f}{(d-t)^3} + \frac{36x_2 + 24v_f}{(d-t)^2} + \frac{9x_3 - 3a_f}{d-t}\right). \tag{3.110}$$

The rightmost sides of the equations are derived by substituting $e(t)$ from (3.101)–(3.103). This completes the proof of the validity of (3.75).

3.5 Resetting Time

If a new target appears or there is a large variation in the previous target, we might wish to reset the remaining time. A new $\tilde{d}$ can be calculated to satisfy a set of constraints. For instance, to find the minimum d such that the average velocity will be unchanged. This is readily possible if the maximum of each state for $d = 1$ from any initial state is known. Using the time scaling introduced in Section 3.2, we can make sure the constraints are respected.

Generally speaking, resetting time can be triggered by detecting a large change in the target point. In case of minimum-jerk trajectories, we can make use of the fact that d^5y/dt^5 is a constant of motion, i.e., as long as we the curve is traversed,

this value is constant. By differentiating (3.61) we obtain,

$$\begin{aligned}\frac{d^4y}{dt^4} =& \frac{y_0 - y_f}{d^4}(360 - 720t_n) \\ &+ \frac{v_0}{d^3}(192 - 360t_n) - \frac{v_f}{d^3}(-168 + 360t_n) \\ &+ \frac{a_0}{d^2}(36 - 60t_n) - \frac{a_f}{d^2}(24 - 60t_n) \end{aligned} \tag{3.111}$$

$$\frac{d^5y}{dt^5} = \frac{y_0 - y_f}{d^5}(-720) + \frac{v_0 + v_f}{d^4}(-360) + \frac{a_0 - a_f}{d^3}(-60) \tag{3.112}$$

Define c to be the constant value of a desired trajectory, such that $d^5y/dt^5 = c$. We can use this value to detect deviations in the trajectory. Consequently, a new d can be calculated to satisfy a desired property as described earlier. For example, by solving the following fifth-order equation we find a new d such that the value of d^5y/dt^5 is unchanged. Rewriting Eq. (3.112) gives us a fifth-order equation to calculate d,

$$cd^5 + 60(a_0 - a_f)d^2 + 360(v0 + v_f)d + 720(x_0 - x_f) = 0.$$

Figure 3.4 illustrates curves with $d^5y/dt^5 = 720$. Note that there is a drift in the final time for all the curves except the blue one, which starts in $x(0) = (-1, 0, 0)$.

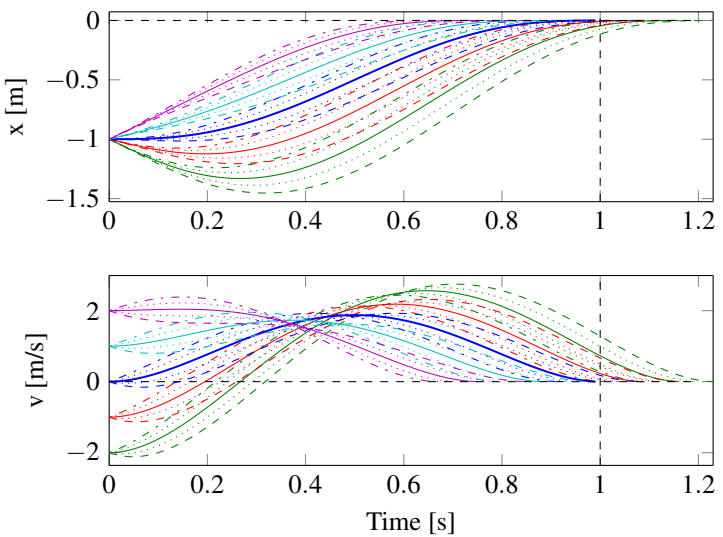

Figure 3.4 Curves starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ with $d^5y/dt^5 = 720$. For the legends, see Fig. 3.5

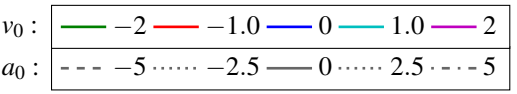

Figure 3.5 Legends for Figs. 3.4, 3.8, 3.9, 3.10, and 3.11.

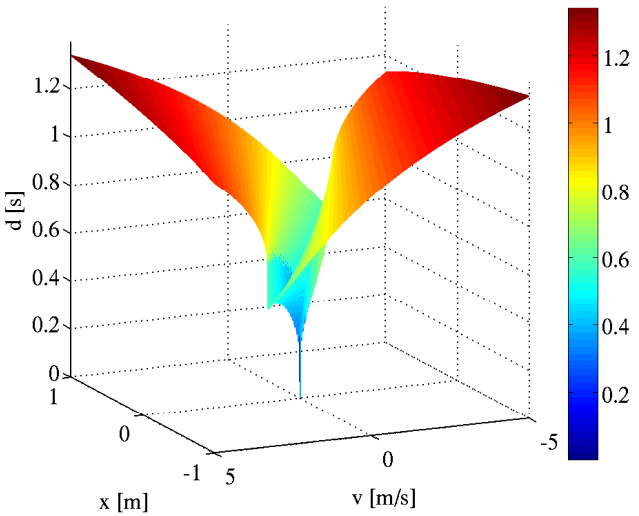

Figure 3.6 The remaining time as a function of the state for a fixed $d^5y/dt^5 = 720$ and $a = 0$.

Figure 3.6 visualizes the remaining time d as a function of the initial state given the assumption of $d^5y/dt^5 = 720$.

3.6 Time-Invariant Model

Time-invariant models, which typically arise in time-optimal or infinite time solutions, do not suffer from some of the limitations imposed by the fixed-time problems. This implies that the variations in the target automatically reset the time. Hence, no signal indicating a new target needs to be calculated. In this section, we examine an approximate time-invariant model. The idea is to build an approximator of the remaining time from the states. Considering the following model,

$$1 - t \approx \left| k_0 \operatorname{sgn}(y) \sqrt[3]{|y|} + k_1 \operatorname{sgn}(\dot{y}) \sqrt{|\dot{y}|} + k_2 \ddot{y} + k_3 \right|, \tag{3.113}$$

where y denotes the position. The model is heuristically designed such that it gives a low error for the estimation of the remaining time along the motion template in (3.3).

We use the least-squares method to estimate the coefficients of the model for the template. The mean-square-error for the motion template is 0.0365; see Fig. 3.7 for the impact of the initial value on the quality of the approximation. According to Fig. 3.7, the remaining time is overestimated for negative initial velocities (moving away from the target) and underestimated for positive velocities (moving toward the target).

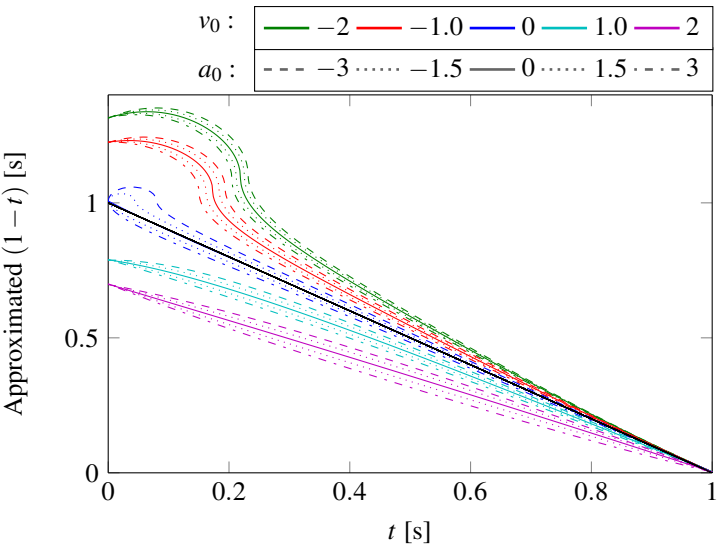

Figure 3.7 The approximated remaining time versus the actual time for $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-3 : 1.5 : 3]$. The black line shows the ideal case. The approximation when $v_0 = 0$ and $a_0 = 0$ coincides well with the ideal case, MSE ≈ 0.0365.

Using the generalization method discussed in Section 3.4, we can easily extend the time-invariant model to different time and coordinate scales.

3.7 Simulations

In this section, we show the simulation results of the closed-loop trajectory generation. We assume that the states are measurable. Therefor, instead of the output y in Fig. 3.2, the states x are used. To visualize the evolution of the states, we use phase portraits.

The first experiment shows the result of the control law in Eq. (3.97) with the change of $\alpha - t$ to $\min(\alpha - t, 0.06)$. The reference is set to zero and the initial position is set to -1. The initial velocity and the initial acceleration are varied. With $\alpha = 1$, we are supposed to obtain the template, i.e., a fifth order polynomial

moving one unit of distance in one unit of time. In Fig. 3.8, the solid blue line in the middle corresponds to this trajectory. Pay special attention to the final state and the final time. For this control law, for a fixed α irrespective of the initial state, the final state is equal to the reference signal and the final time is equal to $\alpha = 1$. Figure 3.9 shows the corresponding phase portrait. Note that the trajectories in the phase plane cross each other since the state-space has a higher dimension than two and the system is time-variant.

The second experiment deals with the time-invariant model obtained by approximating the remaining time along the template according to Eq. (3.113), see Figs. 3.10 and 3.11. Although similar to the previous experiment the solid blue curve matches the template, the fixed-time is not respected for other initial conditions. The final position is reached later than 1 [s] for the negative initial velocities and earlier for the positive initial velocities. This is due to the fact that the approximation depicted in Fig. 3.7 overestimates the remaining time for negative initial velocities and underestimates for positive initial velocities.

The third experiment shows the result of the closed-loop trajectory generation for a moving target. The blue and green curves correspond to the current position of a target and the robot, respectively. Every second, a new target is activated. The objective in Figs. 3.12 and 3.13 is to hit the target in 0.8 [s] and to continue tracking it until a new target is detected. The trajectory generator is the same as the one in the first experiment. As expected, there is a smooth transition to tracking mode after reaching the target. In Figs. 3.14 and 3.15, the simulation results of reaching the same target with a shorter time interval of 0.4 [s] are shown. Changing the duration is done by setting the parameter α. Note how the velocity, the acceleration and the jerk are scaled in these figures compared to Figs. 3.12 and 3.13.

The fourth experiment investigates the effect of changing the parameters α and β for the time-invariant model described in Section 3.6. We consider the unit step response (i.e., a target 1 [m] away) and a step response with amplitude 2 in Fig. 3.16. Note the following interpretation of the parameters: starting from rest, a target which is located β [m] away is reached in α [s]. If the target is closer than β, it is reached earlier than α and if it is further away, it is reached later. Note also the effect of the scaling on different quantities described by Eqs. (3.9)–(3.12) and (3.17)–(3.20).

In the final experiment, we employ the resetting-time strategy according to Section 3.5. Deviations in the trajectory due to disturbances are detected by evaluating d^5y/dt^5. Every time d^5y/dt^5 changes more than a threshold of 0.01, a new remaining time is calculated such that the constraint on $|d^5y/dt^5| < c$ is satisfied. In this experiment, the measured states are affected by an additive white Gaussian noise with a variance of 0.02. The results are shown in Figs. 3.17 and 3.18.

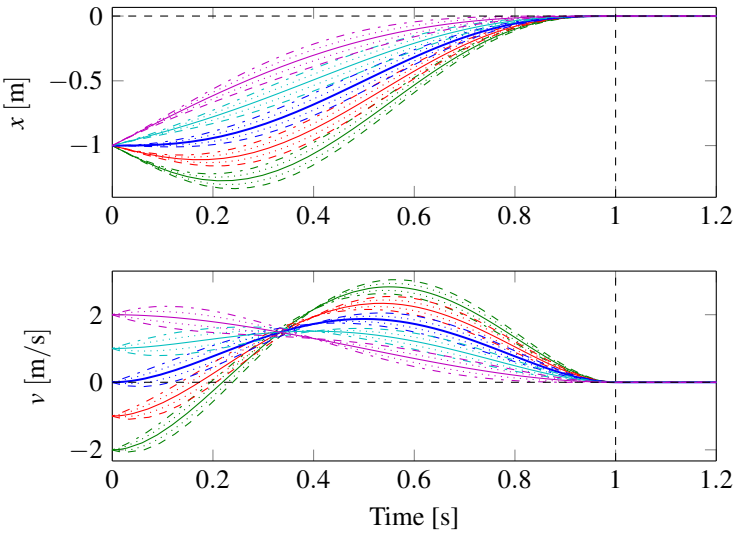

Figure 3.8 Experiment 1: curves starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ with $\alpha = 1$ for the control law of (3.97). For the legends, see Fig. 3.5

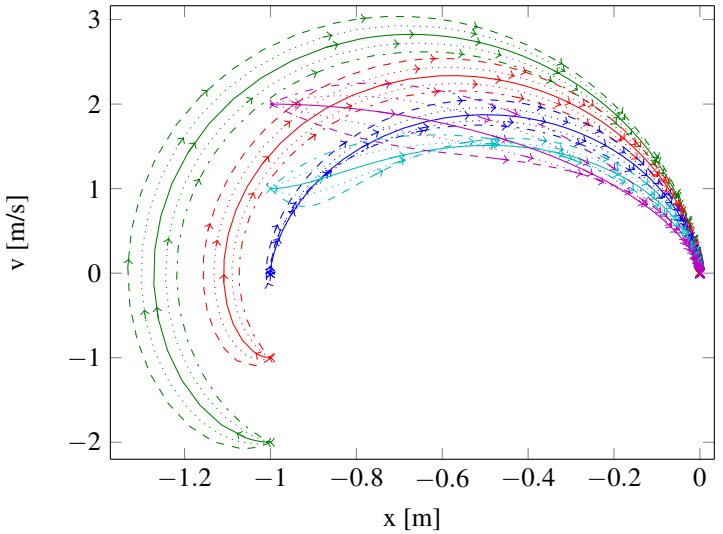

Figure 3.9 Experiment 1: phase portrait for trajectories of Fig. 3.8 starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ with $\alpha = 1$. For the legends, see Fig. 3.5

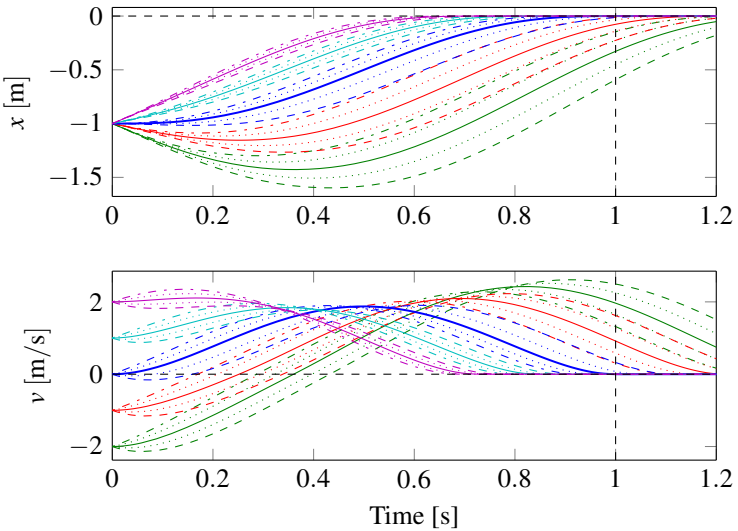

Figure 3.10 Experiment 2: curves starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ for the time-invariant model using (3.113) and $\alpha = 1$. For the legends, see Fig. 3.5

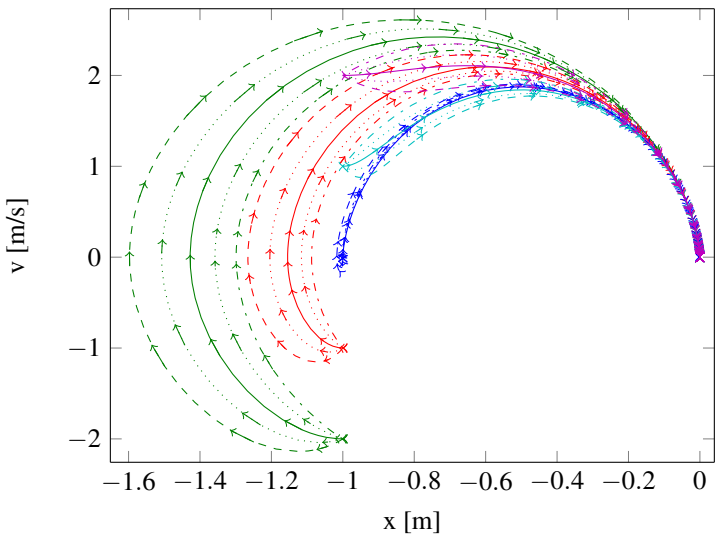

Figure 3.11 Experiment 2: phase portrait for trajectories of Fig. 3.10 starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ with an approximation of d. For the legends, see Fig. 3.5

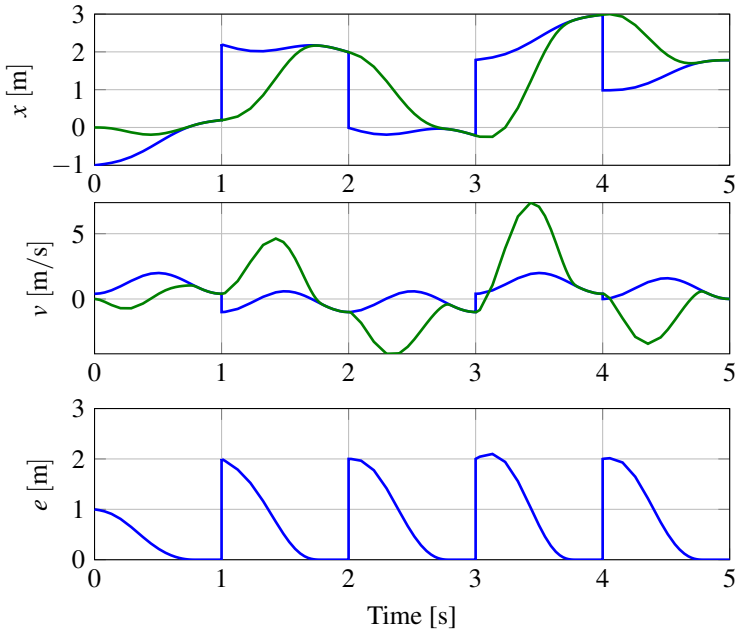

Figure 3.12 Experiment 3: intercepting a moving target in 0.8 [s]: positions, velocities and the error as a function of time. The blue and green curves correspond to the target and the robot, respectively. Every second, a new target is activated.

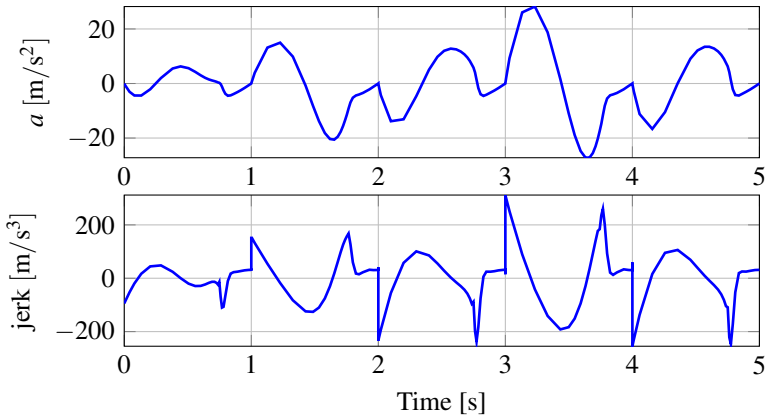

Figure 3.13 Experiment 3: intercepting a moving target in 0.8 [s]: acceleration and jerk as a function of time.

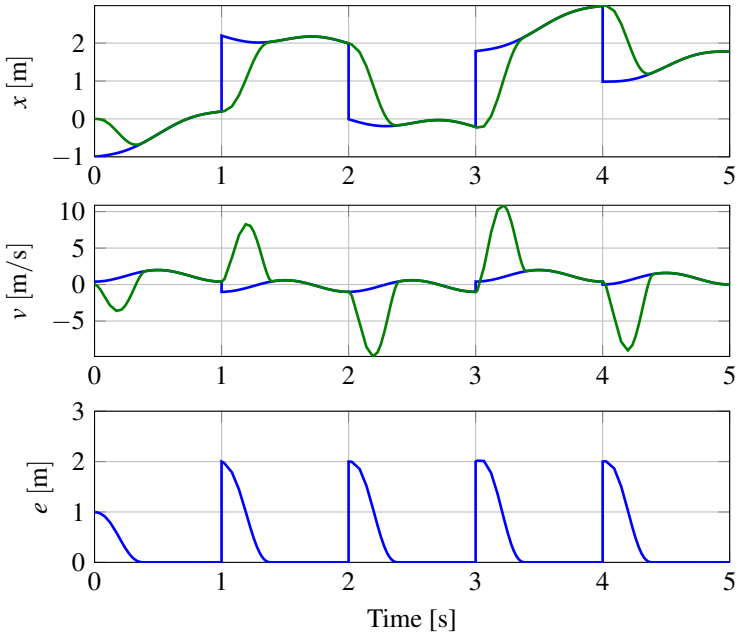

Figure 3.14 Experiment 3: intercepting a moving target in 0.4 [s]: positions, velocities and the position error as a function of time. The blue and green curves correspond to the target and the robot, respectively. Every second, a new target is activated.

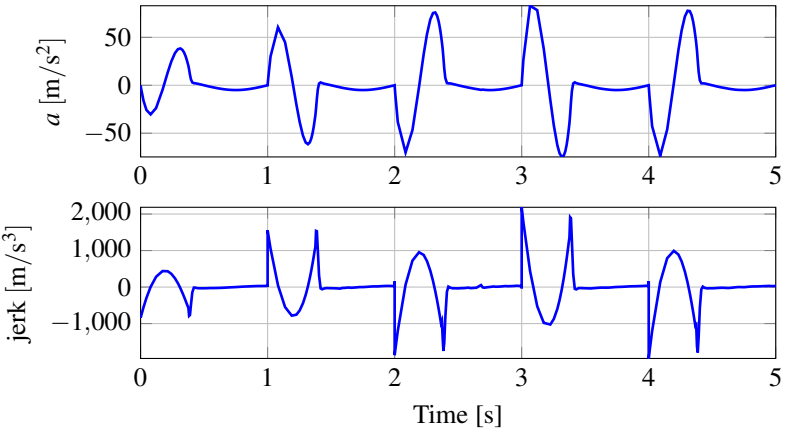

Figure 3.15 Experiment 3: intercepting a moving target in 0.4 [s]: acceleration and jerk as a function of time.

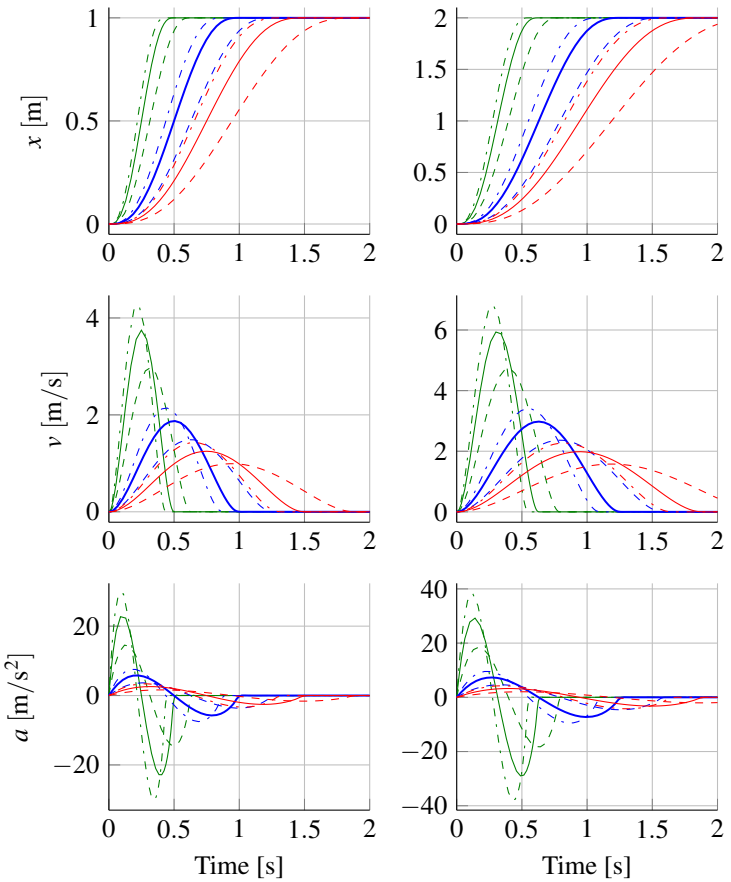

Figure 3.16 Experiment 4: effect of changing parameters for the time-invariant model: $\alpha \in \{0.5, 1, 1.5\}$ corresponds to green, blue, and red, respectively, and $\beta \in \{0.5, 1, 1.5\}$ corresponds to dashed, solid, and dash-dot line, respectively. On the left the step response with magnitude 1 and on the right the step response with magnitude 2 are illustrated.

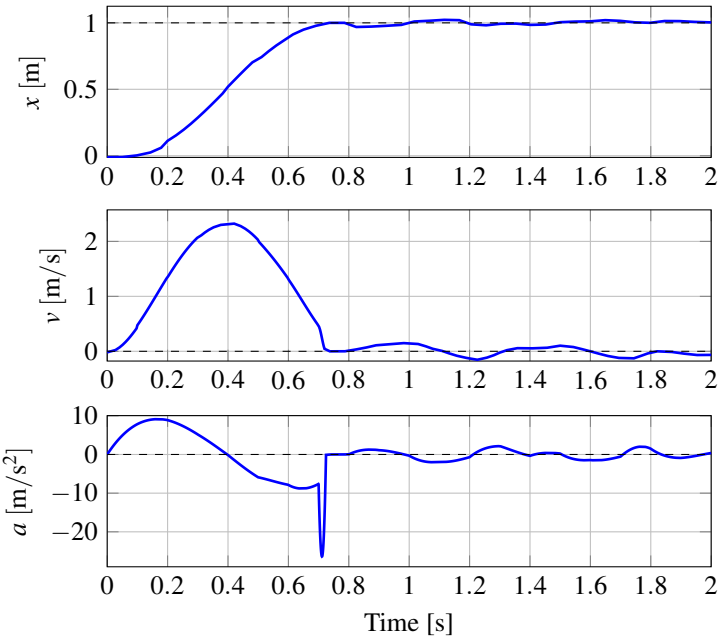

Figure 3.17 Resetting time: an additive Gaussian noise with variance 0.02 affects the states. The remaining time is reset so that the condition $d^5y/dt^5 \leq 4000$ is maintained.

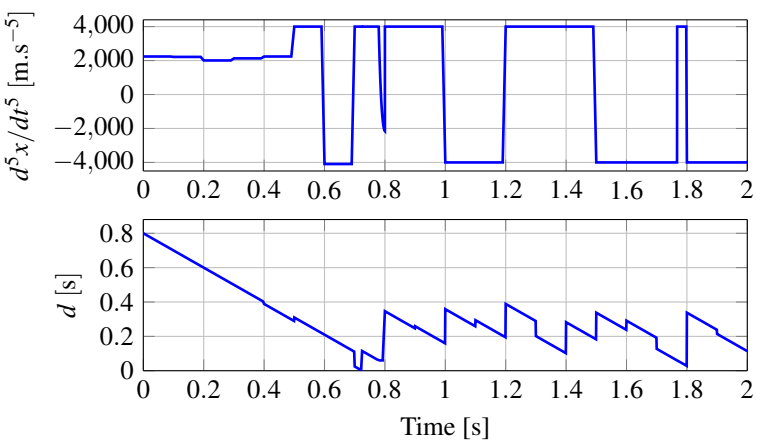

Figure 3.18 Resetting time: an additive Gaussian noise with variance 0.02 affects the states. The remaining time d is reset so that the condition $d^5y/dt^5 \leq 4000$ is maintained. The upper plot shows d^5y/dt^5 and the lower plot illustrates d.

3.8 Discussion

There are at least two strategies for the planning problems when the target is moving. One can estimate a time $\hat{t}_f$ and a desired *future value* for the states, such that the generated trajectory meets this point, i.e., $y(\hat{t}_f) = \hat{r}_f$. The procedure is repeated as soon as a better estimate is obtained. The other strategy is that the generated trajectory should track the *current value* of the target but additionally superimpose a motion which eliminates the initial offset. These strategies do not necessarily lead to the same solution. The first strategy works better than the second one if a good model of the target's motion for estimation is available. However, the second strategy leads to a smooth transition between trajectory planning and tracking. Since the states in many physical systems cannot change discontinuously, the second strategy is advantageous when the time horizon is short. This justifies the modified model of the trajectory generation in Fig. 3.2, which depends only on the error $r(t) - y(t)$ in comparison to Fig. 3.1.

It is required to pay special attention to the source of changes when designing trajectory generation in a closed-loop setting. Variations in the target point do not necessarily have to impact the trajectory in the same way as the disturbances on the robot. This requires a controller with two degrees of freedom where, for instance, different filters are used for the state feedback and the reference signal.

In its current form, the controller assumes the full knowledge of the states of the robot. Therefore, an observer will be required if the states are not measurable. The benefit of an observer is that the trajectory generation will be less sensitive to the measurement noise. With regard to this, we can for example use an internal model together with the trajectory-generator controller such that the actual feedback from the robot is only limited to certain instances. The generated trajectory is then tracked by a common feedback controller. The update of the state of the internal model is only triggered when the deviation between the internal model and the robot is too large. This way, we have obtained a two-degrees-of-freedom controller with different responses to measurement noise and to changes in the reference signal.

In general, both finite-time and minimum-time controller models for trajectory generation perform poorly for disturbance rejection close to the target. The former eliminates the disturbances in the same fixed-time period as it generates the trajectory. Thus, as the time runs out, the controller needs to increase its effort for compensation. On the other hand, the optimal-time controller uses the maximum effort all the time. So, when the current state is in the vicinity of the desired state, the noise can lead to an aggressive control signal. One suggestion to mitigate this problem is to relax the constraint at the final time and instead introduce a cost for it. This can for example be realized by introducing a time-dependent or state-dependent cost for a Linear Quadratic Regulator (LQR) controller.

If in a real setup the state feedback is going to be used, the robustness of the closed-loop for a non-ideal robot model needs to be studied. Also, in the approximation methods introduced in Section 3.6, an important aspect is ensuring a global

attractor for the approximated system, i.e., any initial error is reduced to zero.

3.9 Conclusions

Time and coordinate scaling are powerful operators for extending trajectories to a wider workspace. The coordinate transformation in Section 3.2 made it possible to generalize trajectories to follow an arbitrary reference signal. This transformation did not affect the optimality of the minimum-jerk trajectories.

We proposed a controller model for trajectory generation with continuous reactions to the changes in the target. We solved the Hamilton-Jacobi-Bellman equation in order to find the optimal minimum-jerk controller. The result was a time-varying linear feedback law which produces fifth-order polynomials. For this controller, we showed that saturating the remaining time from below naturally leads to a smooth transition between trajectory generation and tracking modes.

The closed-loop system in our reformulation of the minimum-jerk model generates trajectories which have a bell-shaped velocity profile. Therefore, our results offer an alternative solution to some of the bio-inspired approaches; see for example [Degallier et al., 2011]. An extension of our work can be the derivation of a controller for trajectory generation for more complex dynamical models than a triple integrator.

4

Optimization-based Trajectory Generation

4.1 Introduction

Assume a control system

$$\dot{x} = f(t,x,u), \qquad x(t_0) = x_0 \tag{4.1}$$

where $x \in X \subset \mathbb{R}^n$ is the state, $u \in U \subset \mathbb{R}^m$ is the control signal, $t \in \mathbb{R}$ is the time, t_0 is the initial time, and x_0 is the initial state.

An optimal control problem is usually characterized by a cost functional to be minimized. A generic cost functional for a deterministic setting, can be formulated as below [Liberzon, 2011]

$$J(u) := \int_{t_0}^{t_f} L(t,x(t),u(t))\,dt + K(t_f,x_f) \tag{4.2}$$

Here t_f and $x_f := x(t_f)$ are the final (or terminal) time and state, $L : \mathbb{R} \times \mathbb{R}^n \times U \to \mathbb{R}$ is the running cost, and $K : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ is the terminal cost. In addition to constraints imposed on the input and output signals by the dynamical system, other constraints corresponding to the control set U, possible states X, initial and target sets can be considered.

Several interesting trajectory planning problems can be cast in this framework. For solving a trajectory planning problem, it is usually beneficial to solve the path-planning problem first and then find a trajectory by assigning time to each point along the path. Nevertheless, for the specification of the problem, such a separation seems unnecessary since both problems could be specified in a unified way in the optimization framework.

We enumerate a few trajectory planning problems in the following with no intention to be comprehensive. In the list, Q and R are appropriate weighting matrices,

F and G are constraint matrices for the states and the inputs, correspondingly, $g(\cdot)$ and $h(\cdot)$ are two possibly non-linear functions defining terminal constraints and path constraints, respectively, r is the reference signal, and y is a function of the state.

1. Tracking problem (variable end-point, fixed-time)

$$J(u) = \int_0^{t_f} \left(e^T Q e + u^T R u\right) dt \tag{4.3}$$

   subject to $Fx \leq b$, $Gu \leq a$, where $e = x_r - x$

2. Point-to-point planning (fixed end-point, fixed-time)

$$J(u) = \int_0^{t_f} (x^T Q x + u^T R u)\, dt + K(x(t_f)) \tag{4.4}$$

   subject to $Fx \leq b$, $Gu \leq a$, and $y(t_f) = y_f$

3. Hitting a (moving) target (fixed/variable end-point, free-time)

$$J(u, t_f) = \int_0^{t_f} (x^T Q x + u^T R u)\, dt + K(x(t_f)) \tag{4.5}$$

   subject to $Fx \leq b$, $Gu \leq a$, $t_f < T_f$, and $y(t_f) = g(t_f)$

4. Time-optimal point-to-point planning (fixed end-point, free-time), subject to $Fx \leq b$, $Gu \leq a$, and $y(t_f) = y_f$

5. Time-optimal given a path (variable end-point, free-time), subject to $Fx \leq b$, $Gu \leq a$ and $y(t_f) = y_f$, and $h(x(t)) = 0$

6. Time-optimal given a trajectory profile (variable end-point, free-time), subject to $Fx \leq b$, $Gu \leq a$, $y(t_f) = y_f$ and $J(u, t_f) \leq V(t_f)$, where

$$J(u, t_1) = \int_0^{t_1} (x^T Q x + u^T R u)\, dt, \tag{4.6}$$

   and

$$V(t) = \min_u J(u, t) \tag{4.7}$$

   with neither state nor input constraints.

Note that to be strictly correct if $y(t_f)$ does not fully constrain the states, the problem is usually not called fixed end-point.

Trajectory generation based on nonlinear optimization has been proposed in [Bauml et al., 2010] and [Lampariello et al., 2011] for ball-catching robot systems. Model Predictive Control (MPC) has been proposed for control and path tracking

for mobile robots; see, for example [Klančar and Škrjanc, 2007], [Howard et al., 2010], and [Kanjanawanishkul and Zell, 2009]. In [Shim and Sastry, 2007], MPC is used for tracking of linearly interpolated way-points. A two-layer architecture, with a generic optimal control formulation at the high-level and MPC at the low-level to track the high-level solution, is proposed in [Norén, 2013].

The motivation for this chapter is to improve the efficiency of trajectory planning expressed in the optimization framework. The impact of model selection for optimal control problems is briefly studied. Specifically, we compare a kinematic model with a dynamic model in Section 4.2. In the following sections, the problem of planning a trajectory between an initial and a final state at a given time for mechanical systems is tackled. In Section 4.3, we find an analytic solution to a kinematic model with kinematic constraints. In Section 4.4, as opposed to previous research we propose MPC [Garcia et al., 1989; Rawlings and Mayne, 2009] as a way to simultaneously solve both the path planning and the trajectory generation problems. To fulfill real-time constraints in modern motion controllers for robotic systems, we develop a method relying on convex optimization. The results from the experiments on an industrial robot in a demanding task of ball-catching are provided in the same section. Section 4.5 provides an example in which the full power of MPC for solving the trajectory and path planning can be utilized. Conclusions are drawn in Section 4.6.

4.2 Comparison of Kinematic and Dynamic Models

In this section, we study the relevance of kinematic models for optimal control of robots.

Kinematic Model

Let us assume that we have a robot with two degrees of freedom. The kinematics of the robot can be described by a two-dimensional double-integrator as

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= u_x \\ \dot{v}_y &= u_y \end{aligned} \tag{4.8}$$

The coordinates x and y can be interpreted as generalized coordinates, i.e., depending on the structure of a robot, as either Cartesian coordinates or joint values.

Dynamic Model

The equations of motion for a serial robot are usually derived by the Lagrange equations. These equations have a generic form of

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + E(q,\dot{q}) + G(q) = \tau, \tag{4.9}$$

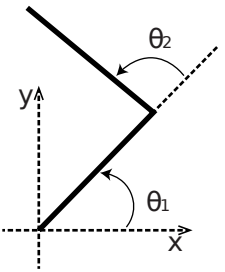

Figure 4.1 Schematic of two-link robot

where $H(q)$ is a symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q}$ the vector of centripetal and Coriolis torques, $E(q,\dot{q})$ the effect of an external force field and $G(q)$ the term due to gravity, $q \in \mathbb{R}^n$ denotes the vector of joint angles, and its time derivative is denoted by the dot operator. The system is driven by the torque vector $\tau \in \mathbb{R}^n$.

For the sake of simplicity, a two-link planar robot is considered. If we limit the motion in a horizontal plane, the effect of the gravity can be ignored. In this case, the matrices corresponding to (4.9) are

$$H(q) = \begin{pmatrix} a_1 + 2a_3\cos(q_2) + a_2 & a_3\cos(q_2) + a_2 \\ a_3\cos(q_2) + a_2 & a_2 \end{pmatrix} \tag{4.10}$$

$$C(q,\dot{q}) = \begin{pmatrix} -a_3\sin(q_2)\dot{q}_2 & -a_3\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ a_3\sin(q_2)\dot{q}_1 & 0 \end{pmatrix} \tag{4.11}$$

$$a_1 = m_2 L_1^2 + m_1 r_1^2 + I_1 \tag{4.12}$$

$$a_2 = m_2 r_2^2 + I_2 \tag{4.13}$$

$$a_3 = m_2 L_1 r_2 \tag{4.14}$$

where the subscripts of q indicate its elements, with indices denoting the link number, m, L, r, and I referring to link mass, link length, the distance between the previous joint to the center of gravity of the link, and moment of inertia around the center of mass, respectively. For a numerical experiment, values are chosen according to Table 4.1.

Table 4.1 Geometrical and dynamical parameters for the numerical simulation

	m [kg]	L [m]	r [m]	I [kg·m^2]
Link 1	1.59	0.3	0.13	0.0216
Link 2	1.44	0.35	0.14	0.0089

Problem Formulation

Minimum Time Problem A few different problems are considered.

P1: With state constraints purely on kinematic variables:

$$\begin{aligned} \underset{u,t_f}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & (4.8) \\ & |u_x| \leq 1,\ |u_y| \leq 1 \\ & x(0) = 2,\ y(0) = 1 \\ & v_x(0) = 0,\ v_y(0) = 0 \\ & x(t_f) = y(t_f) = 0 \\ & v_x(t_f) = v_y(t_f) = 0 \end{aligned} \tag{4.15}$$

P2: With state constraints purely on dynamic variables:

$$\begin{aligned} \underset{\tau,t_f}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & (4.9) \\ & |\tau_1| \leq 1,\ |\tau_2| \leq 1 \\ & q_1(0) = \pi/4,\ q_2(0) = \pi/10, \\ & \omega_1(0) = \omega_2(0) = 0 \\ & q_1(t_f) = q_2(t_f) = 0 \\ & \omega_1(t_f) = \omega_2(t_f) = 0 \end{aligned} \tag{4.16}$$

where $\omega_i(\cdot) := \dot{q}_i(\cdot)$, $i \in \{1,2\}$.

P3: With state constraints purely on kinematic variables:

$$\begin{aligned} \underset{\tau,t_f}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & (4.9) \\ & |\dot{\omega}_1| \leq 2,\ |\dot{\omega}_2| \leq 2 \\ & q_1(0) = \pi/4,\ q_2(0) = \pi/10 \\ & \omega_1(0) = \omega_2(0) = 0 \\ & q_1(t_f) = q_2(t_f) = 0 \\ & \omega_1(t_f) = \omega_2(t_f) = 0 \end{aligned} \tag{4.17}$$

P4: With constraints on both kinematic and dynamic variables:

$$\begin{aligned} \underset{\tau,t_f}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & (4.9) \\ & |\dot{\omega}_1| \leq 10,\ |\dot{\omega}_2| \leq 10 \\ & |\tau_1| \leq 1,\ |\tau_2| \leq 1 \\ & q_1(0) = \pi/4,\ q_2(0) = \pi/10 \\ & \omega_1(0) = \omega_2(0) = 0 \\ & q_1(t_f) = q_2(t_f) = 0 \\ & \omega_1(t_f) = \omega_2(t_f) = 0 \end{aligned} \tag{4.18}$$

Additional Constraints

The time-optimal control problem for systems of more than one degree of freedom can be under-constrained. This can easily be understood when the constraints in one dimension (in the first example x) determine the final time. In such cases, various motions in other dimensions can meet the final time without violating the constraints. In order to get a unique solution, it is necessary to add penalties for the states and/or the input signals. We can also without compromising the optimal time, solve a second optimization problem. The new problem is fixed-time and a combination of states and input signals can be penalized in the cost function according to

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_0^{t_f} v_x^2 + v_y^2 + 0.001(u_x^2 + u_y^2)\,dt \\ \text{subject to} \quad & (4.8) \\ & x(0) = 2,\ y(0) = 1 \\ & |u_x| \leq 1,\ |u_y| \leq 1 \\ & x(t_f) = y(t_f) = 0 \\ & v_x(t_f) = v_y(t_f) = 0 \end{aligned} \tag{4.19}$$

$$\begin{aligned} \underset{\tau}{\text{minimize}} \quad & \int_0^{t_f} \omega_1^2 + \omega_2^2 + 0.01(\tau_1^2 + \tau_2^2)\,dt \\ \text{subject to} \quad & (4.9) \\ & |\tau_1| \leq 1,\ |\tau_2| \leq 1 \\ & q_1(0) = \pi/4,\ q_2(0) = \pi/10 \\ & \omega_1(0) = \omega_2(0) = 0 \\ & q_1(t_f) = q_2(t_f) = 0 \\ & \omega_1(t_f) = \omega_2(t_f) = 0. \end{aligned} \tag{4.20}$$

Adding penalties on the derivatives of the input signals can also be beneficial for smoothing out the signals. Another interesting cost function is the length of the curve, i.e., $\int_0^{t_f} \sqrt{v_x^2 + v_y^2}\, dt$

Obstacle Avoidance It is also interesting to include some non-convex constraints. Such constraints usually appear in obstacle-avoidance problems. The effect of a constraint of the following form has been studied

$$(y - y_0)^2 + (x - x_0)^2 \geq c^2. \tag{4.21}$$

Results

The solutions to the optimization problems (4.15)–(4.18) are presented in this section. The optimization problems were solved by numerical methods based on direct collocation [Biegler et al., 2002]. The implementation relied on the open-source software platform JModelica.org [Åkesson et al., 2010; *JModelica.org* 2015].

As we see in Figs. 4.2 and 4.6, one of the control signals switches rapidly. This is due to the fact that there is no unique solution to those time-optimal problems, which results in the strange behavior of the solver.

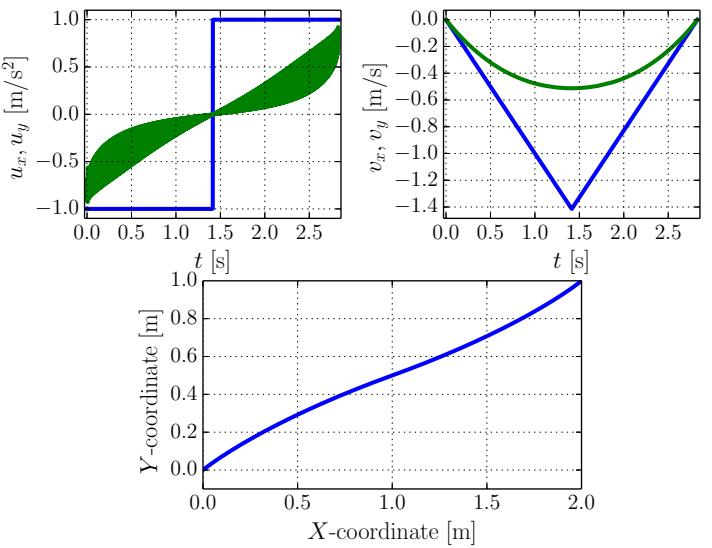

Figure 4.2 Solution of (4.15): the blue and green curves correspond to motion along the x- and y-axis, respectively. In the X–Y plot, the solid line corresponds to the position in 2D.

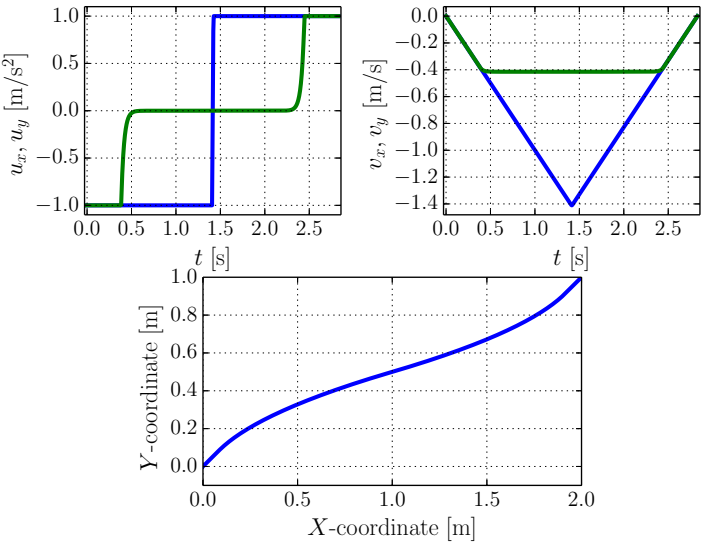

Figure 4.3 Solution of (4.15) after the second optimization. The blue and green curves correspond to motion along the x-axis and y-axis, respectively. In the X–Y plot, the solid line corresponds to the position in 2D.

Conclusion

The time-optimal solution for a problem with multiple degrees of freedom is typically non-unique. Given the minimum time, we can solve a second optimization problem with penalties on the states and/or the inputs.

In the case of no constraints on dynamic variables or tight constraints on kinematic variables, the dynamic model can be ignored and the whole dynamics be approximated by a double integrator. This is due to the fact that the matrix $H(q)$ is positive definite and thus invertible [Siciliano et al., 2009]. Consequently, it is always possible to calculate the required τ within the constraints to achieve the same bang-bang solution as in (4.15).

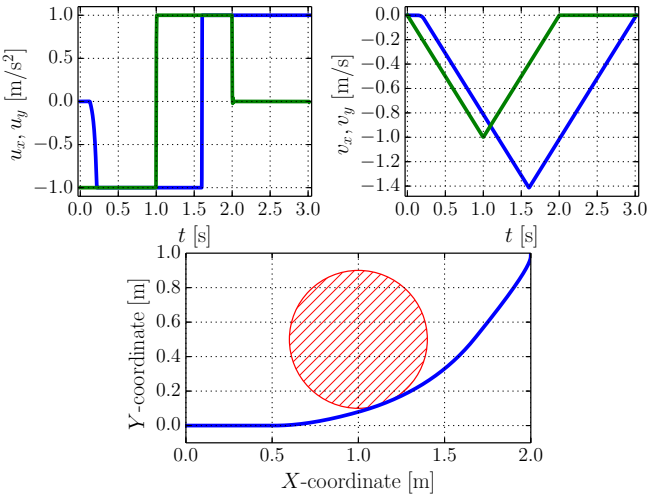

Figure 4.4 Solution of (4.15) with obstacle avoidance. The blue and green curves correspond to motion along the x-axis and y-axis, respectively.

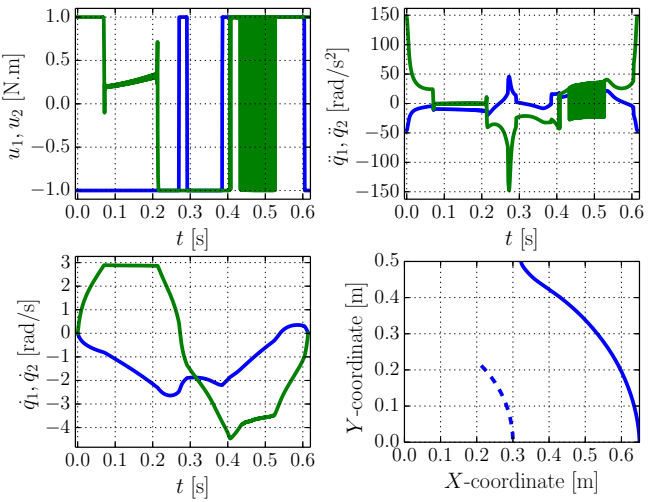

Figure 4.5 Solution of (4.16). The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the second joint, respectively.

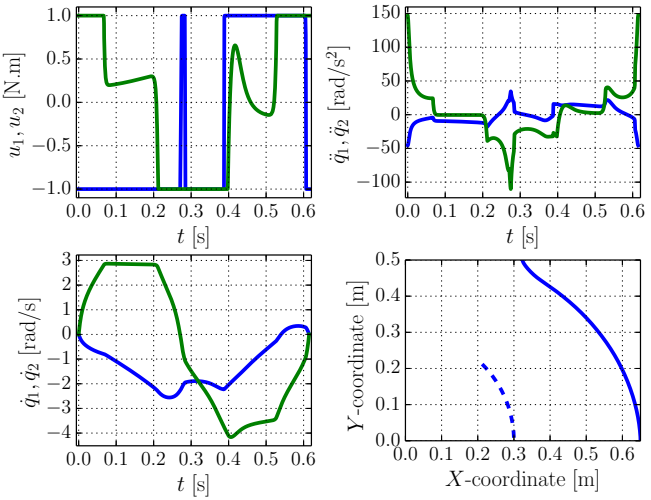

Figure 4.6 Solution of (4.16) after the second optimization. The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the second joint, respectively.

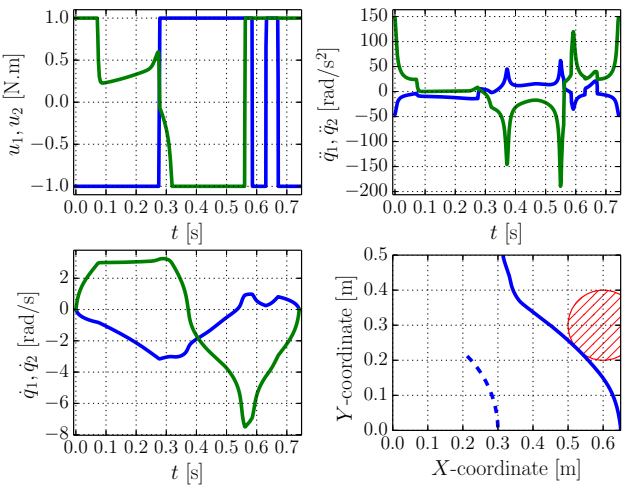

Figure 4.7 Solution of (4.16) with obstacle avoidance. The blue and green curves correspond to joint one and two, respectively.

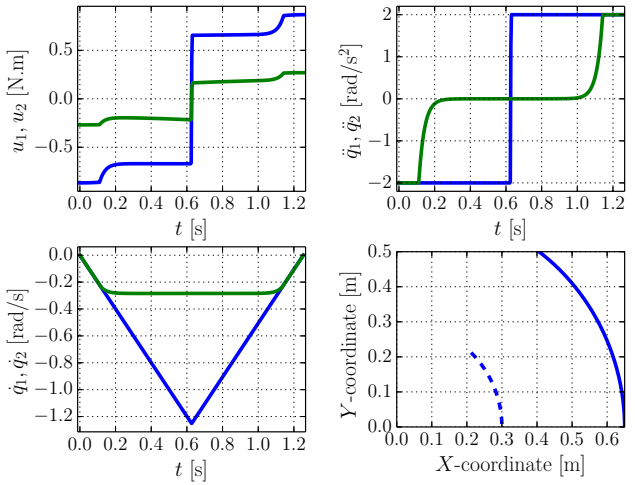

Figure 4.8 Solution of (4.17) with only active kinematic constraints after the second optimization.

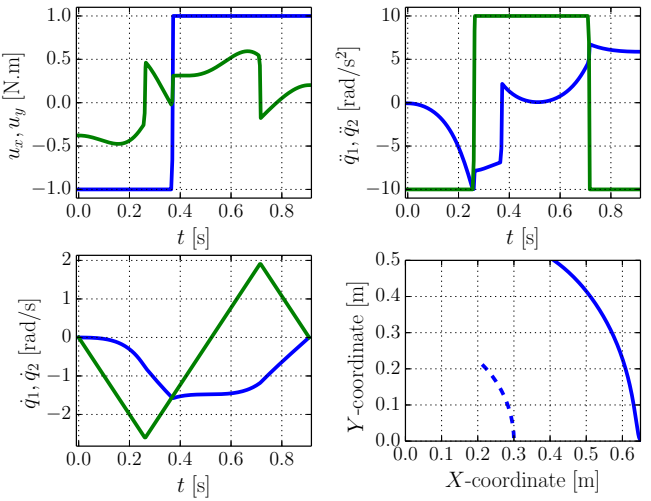

Figure 4.9 Solution of (4.18) with both kinematic and dynamic constraints after the second optimization. The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the second joint, respectively.

4.3 An Analytic Solution to Fixed Time Trajectory Planning

In this section, we present two main theorems in order to find a solution to an optimal control problem with state variable inequality constraints (SVIC) based on [Seierstad and Sydsæter, 1987; Hartl et al., 1995]. These theorems are derived from the maximum principle [Pontryagin et al., 1962] and concern direct and indirect adjoining approaches. Furthermore, we apply the direct adjoining approach to solve a trajectory planning problem for a double integrator model. The results are compared with the numerical solution obtained by an interior-point method [Boyd and Vandenberghe, 2004].

Problem Formulation

The aim is to find the maximum of the following objective functional J subject to the constraints on states and control signals [Seierstad and Sydsæter, 1987; Hartl et al., 1995]:

$$\begin{aligned} J(u) &= \int_0^{t_f} F(x(t),u(t),t)dt + S(x(t_f),t_f) \\ \dot{x}(t) &= f(x(t),t), \qquad x(0) = x_0 \end{aligned} \tag{4.22}$$

$$\begin{aligned} g(x(t),u(t),t) &\geq 0 \\ h(x(t),t) &\geq 0 \\ a(x(t_f),t_f) &\geq 0 \\ b(x(t_f),t_f) &= 0, \end{aligned} \tag{4.23}$$

where

$$h : E^n \times E \to E^q, \qquad g : E^n \times E^m \times E \to E^s \tag{4.24}$$

$$a : E^n \times E \to E^l, \qquad b : E^n \times E \to E^{l'}. \tag{4.25}$$

General Definition and Condition

The order of pure-state constraints as well as junction times are defined here. Additionally, a constraint qualification condition is presented. The definition and the condition are used when the theories are presented. Note that when two symbols appear after each other, depending on the dimensions, dot product or matrix multiplication is intended.

Order of Pure State Constraints

$$\begin{aligned} h^0(x,u,t) &= h = h(x,t) \\ h^1(x,u,t) &= \dot{h} = h_x(x,t)f(x,u,t) + h_t(x,t) \\ h^2(x,u,t) &= \dot{h}^1 = h_x^1(x,t)f(x,u,t) + h_t^1(x,t) \\ &\vdots \\ h^p(x,u,t) &= \dot{h}^{p-1} = h_x^{p-1}(x,t)f(x,u,t) + h_t^{p-1}(x,t) \end{aligned} \tag{4.26}$$

The state constraint is of order p iff

$$h_u^i(x,u,t) = 0, \text{ for } 0 \le i \le p-1, \quad h_u^p(x,u,t) \neq 0, \tag{4.27}$$

Junction Times With respect to the ith constraint, an interval $[\tau_1, \tau_2]$ is called a *boundary interval* if $h_i(x(t),t) = 0$ for all $t \in [\tau_1, \tau_2]$. A subinterval $(\tau_1, \tau_2) \subset [0, t_f]$ is called an interior interval of a trajectory $x(\cdot)$ if $h_i(x(t),t) > 0$ for all $t \in (\tau_1, \tau_2)$. If an interior interval ends at τ_1 and a boundary interval starts at τ_1, the instant τ_1 is called an *entry time*. Correspondingly, τ_2 is called an *exit time* if there is a boundary interval ending at $t = \tau_2$ and an interior interval starting at τ_2. A *contact time* is the instant that the trajectory just touches the boundary, i.e., $h(x(\tau),\tau) = 0$ and the trajectory is in the interior just before and after τ. Entry, exit, and contact times are called *junction times*.

Constraint Qualification The constraint qualification must hold for terminal constraints:

$$\operatorname{rank} \begin{bmatrix} \partial a/\partial x & \operatorname{diag}(a) \\ \partial b/\partial x & 0 \end{bmatrix} = l + l' \tag{4.28}$$

Additionally, for mixed constraints, i.e., the constraints involving both states and input signals, we have

$$\operatorname{rank}[\partial g/\partial u \quad \operatorname{diag}(g)] = s \tag{4.29}$$

Direct Adjoining approach

In this approach the mixed constraints as well as the pure constraints are directly adjoined to the Hamiltonian denoted by H to form the Lagrangian denoted by L.

Hamiltonian and Lagrangian (D-form):

$$H(x,u,\lambda_0,\lambda,t) = \lambda_0 F(x,u,t) + \lambda f(x,u,t) \tag{4.30}$$

$$L(x,u,\lambda_0,\lambda,\mu,\nu,t) = H(x,u,\lambda_0,\lambda,t) + \mu g(x,u,t) + \nu h(x,u,t) \tag{4.31}$$

The costate is a mapping $\lambda(\cdot) : [0,t_f] \to E^n$ and multiplier functions $\mu(\cdot)$ and $\nu(\cdot)$ are mappings $[0,t_f]$ into E^s and E^q, respectively.

Control region:

$$\Omega(x,t) = \{u \in E^m \,|\, g(x,u,t) \geq 0\} \tag{4.32}$$

THEOREM 1—DIRECT ADJOINING APPROACH
[Hartl et al., 1995] Let $\{x^*(\cdot), u^*(\cdot)\}$ be an optimal pair for problem (4.22)–(4.25) over $[0,t_f]$ such that

- $u^*(\cdot)$ is right-continuous with left-hand limits
- constraint qualification holds for every tripple $\{t, x^*(t), u\}, t \in [0,t_f], u \in \Omega(x^*(t),u)$
- Assume $x^*(t)$ has only finitely many junction times,

then there exists

- a constant $\lambda_0 \geq 0$
- a piecewise absolutely continuous costate trajectory $\lambda(\cdot)$ mapping $[0,t_f]$ into E^n
- piecewise continuous multiplyer functions $\mu(\cdot)$ and $\nu(\cdot)$ mapping $[0,t_f]$ into E^s and E^q, respectively
- a vector $\eta(\tau_i) \in E^q$ for each point τ_i of discontinuity of $\lambda(\cdot)$
- $\alpha \in E^l$ and $\beta \in E^{l'}$, $\gamma \in E^q$, not all zero,

such that the following conditions hold almost everywhere:
Hamiltonian maximization

$$u^*(t) = \underset{u \in \Omega(x^*(t),t)}{\arg\max}\ H(x^*(t),u,\lambda_0,\lambda(t),t) \tag{4.33}$$

and conditions on the optimal Hamiltonian and Lagrangian, costates and multipliers

$$L_u^*[t] = H_u^*[t] + \mu g_u^*[t] = 0 \tag{4.34}$$

$$\dot{\lambda}(t) = -L_x^*[t] \tag{4.35}$$

$$\mu(t) \geq 0, \qquad \mu(t) g^*[t] = 0 \tag{4.36}$$

$$\nu(t) \geq 0, \qquad \nu(t) h^*[t] = 0 \tag{4.37}$$

$$\text{and} \quad dH^*[t]/dt = dL^*[t]/dt = L_t^*[t] \triangleq \partial L^*[t]/\partial t. \tag{4.38}$$

At the terminal time t_f, the transversality conditions hold:

$$\lambda(t_f^-) = \lambda_0 S_x^*[t_f] + \alpha a_x^*[t_f] + \beta b_x^*[t_f] + \gamma h_x^*[t_f] \tag{4.39}$$

$$\alpha \geq 0,\ \gamma \geq 0, \quad \alpha a^*[t_f] = \gamma h^*[t_f] = 0 \tag{4.40}$$

For any time τ in the boundary interval and for any contact time τ, the costate trajectory may have a discontinuity given by the following conditions

$$\lambda(\tau^-) = \lambda(\tau^+) + \eta(\tau)h_x^*[\tau] \tag{4.41}$$
$$H^*\left[\tau^-\right] = H^*\left[\tau^+\right] - \eta(\tau)h_t^*[\tau] \tag{4.42}$$
$$\eta(\tau) \geq 0, \quad \eta(\tau)h^*[\tau] = 0. \tag{4.43}$$

□

Indirect Adjoining approach

To form the Lagrangian, first or higher-degrees derivatives of the pure-state constraints $h(x(t),t)$ in (4.23), are adjoined to the Hamiltonian. Here we present the theorem for first-order constraints.

Hamiltonian and Lagrangian (P-form):

$$H(x,u,\lambda_0,\lambda,t) = \lambda_0 F(x,u,t) + \lambda f(x,u,t) \tag{4.44}$$
$$L(x,u,\lambda_0,\lambda,\mu,\nu,t) = H(x,u,\lambda_0,\lambda,t) + \mu g(x,u,t) + \nu h^1(x,u,t) \tag{4.45}$$

Control region:

$$\Omega(x,t) = \left\{u \in E^m \,|\, g(x,u,t) \geq 0,\ h^1(x,u,t) \geq 0 \text{ if } h(x,t) = 0\right\} \tag{4.46}$$

THEOREM 2—INDIRECT ADJOINING APPROACH
[Hartl et al., 1995] Let $\{x^*(\cdot), u^*(\cdot)\}$ be an optimal pair for problem (4.22)–(4.25) such that

- $x^*(\cdot)$ has only finitely many junction times,
- strong constraint qualification (4.28) and (4.29) hold,

then there exist

- a constant $\lambda_0 \geq 0$,
- a piecewise absolutely continuous costate trajectory λ mapping $\left[0,t_f\right]$ into E^n,
- piecewise continuous multiplyer functions $\mu(\cdot)$ and $\nu(\cdot)$ mapping $\left[0,t_f\right]$ into E^s and E^q, respectively,
- a vector $\eta(\tau_i) \in E^q$ for each point τ_i of discontinuity of $\lambda(\cdot)$,
- $\alpha \in E^l$ and $\beta \in E^{l'}$, not all zero,

such that the following conditions hold almost everywhere:
Hamiltonian maximization

$$u^*(t) = \underset{u\in\Omega(x^*(t),t)}{\arg\max}\ H(x^*(t),u,\lambda_0,\lambda(t),t) \tag{4.47}$$

and conditions on the optimal Hamiltonian and Lagrangian, costates and multipliers

$$L_u^*[t] = 0 \tag{4.48}$$

$$\dot{\lambda}(t) = -L_x^*[t] \tag{4.49}$$

$$\mu(t) \geq 0, \qquad \mu(t)g^*[t] = 0. \tag{4.50}$$

Here, ν_i is nondecreasing on boundary intervals of h_i, $i = 1,2,\cdots,q$, with

$$\nu(t) \geq 0,\ \dot{\nu}(t) \leq 0,\ \nu(t)h^*[t] = 0 \tag{4.51}$$

$$\text{and} \quad dH^*[t]/dt = dL^*[t]/dt = L_t^*[t], \tag{4.52}$$

whenever these derivatives exist. At the terminal time t_f, the transversality conditions

$$\lambda(t_f^-) = \lambda_0 S_x^*\left[t_f\right] + \alpha a_x^*\left[t_f\right] + \beta b_x^*\left[t_f\right] + \gamma h_x^*\left[t_f\right] \tag{4.53}$$

$$\alpha \geq 0,\ \gamma \geq 0, \quad \alpha a^*\left[t_f\right] = \gamma h^*\left[t_f\right] = 0 \tag{4.54}$$

hold. At each entry or contact time, the costate trajectory λ may have a discontinuity of the form:

$$\lambda(\tau^-) = \lambda(\tau^+) + \eta(\tau)h_x^*[\tau] \tag{4.55}$$

$$H^*\left[\tau^-\right] = H^*\left[\tau^+\right] - \eta(\tau)h_t^*[\tau] \tag{4.56}$$

$$\eta(\tau) \geq 0, \quad \eta(\tau)h^*[\tau] = 0 \tag{4.57}$$

$$\text{and } \eta(\tau_1) \geq \nu(\tau_1^+) \text{ for entry time } \tau_1. \tag{4.58}$$

□

Example

We consider a double integrator as the model of our system and consider a quadratic cost function. The system is supposed to be at rest initially and to finally arrive at the origin with zero velocity. There are constraints on both input and on the velocity, i.e., u and x_2, correspondingly. Q is assumed to be diagonal.

$$\text{minimize} \qquad \int_0^{t_f} x^T Q x + u^T R u\, dt \tag{4.59}$$

$$\text{subject to} \qquad \dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) \tag{4.60}$$

$$|u(t)| \leq 1 \tag{4.61}$$

$$|x_2(t)| \leq c \tag{4.62}$$

$$x(0) = \begin{pmatrix} x_i, & 0 \end{pmatrix}^T \quad x(1) = 0_{2\times 1} \tag{4.63}$$

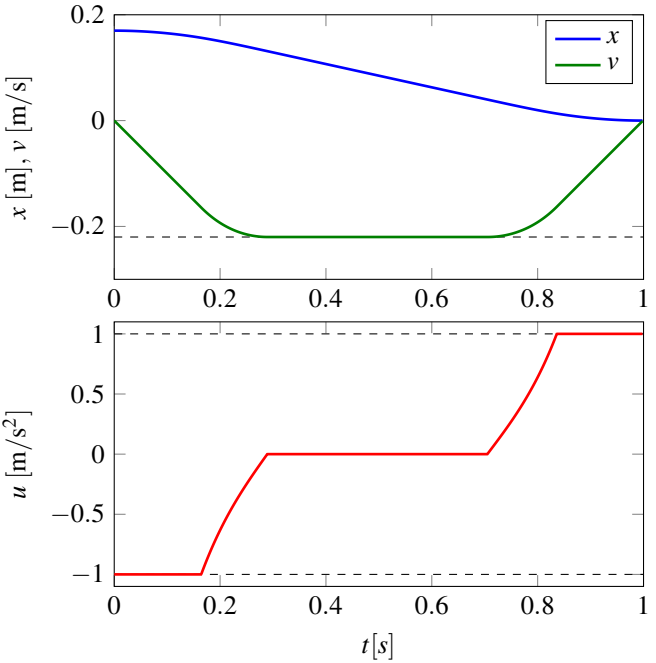

Figure 4.10 Numerical solution to the example in (4.59)–(4.65) with 1000 discretization points.

For the numerical solution shown in Fig. 4.10, the following values are used

$$R = r = 0.1, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix} \tag{4.64}$$

$$c = 0.22, \quad x_i = 0.17, \quad t_f = 1. \tag{4.65}$$

Now we customize the conditions and the result of the direct adjoining approach to our example.

Parameters, Hamiltonian and Lagrangian for Direct Approach By comparing (4.59)–(4.63) with (4.22), we identify

$$F(x(t),u(t),t) = -(x^T Q x + u^T R u) \tag{4.66}$$

$$S(t_f) = 0 \tag{4.67}$$

$$f_1(x(t),u(t),t) = x_2(t) \tag{4.68}$$

$$f_2(x(t),u(t),t) = u(t) \tag{4.69}$$

Similarly, from (4.30) and (4.31)

$$H = -\lambda_0(q_1 x_1(t)^2 + q_2 x_2(t)^2 + r u(t)^2) + \lambda_1(t) x_2(t) + \lambda_2(t) u(t) \tag{4.70}$$
$$L = H + \mu_1(t)(1-u(t)) + \mu_2(t)(1+u(t)) \tag{4.71}$$
$$+ \nu_1(t)(c - x_2(t)) + \nu_2(t)(c + x_2(t)) \tag{4.72}$$

Constraints Identification of the constraints with (4.23) results in

$$h(x(t)) = \begin{pmatrix} c - x_2(t) \\ c + x_2(t) \end{pmatrix} \tag{4.73}$$
$$g(u(t)) = \begin{pmatrix} 1 - u(t) \\ 1 + u(t) \end{pmatrix} \tag{4.74}$$
$$b(x(t)) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \tag{4.75}$$

We find the order of the constraints as below

$$h^0 = \begin{pmatrix} c - x_2 \\ c + x_2 \end{pmatrix} \Rightarrow h_u^0 = 0 \tag{4.76}$$
$$h^1 = \begin{pmatrix} -\dot{x}_2(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} -u(t) \\ u(t) \end{pmatrix} \Rightarrow h_u^1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \tag{4.77}$$

Thus, the order of the constraints is $p = 1$. The constraint qualification condition holds too, since

$$\operatorname{rank} \begin{pmatrix} -1 & 1-u & 0 \\ 1 & 0 & 1+u \end{pmatrix} = 2 = s \quad \forall u \tag{4.78}$$

Hamiltonian Maximization Without constraints, the *Hamiltonian is maximized* for

$$\hat{u}(t) = \frac{1}{2}\frac{1}{r}\frac{\lambda_2(t)}{\lambda_0} \tag{4.79}$$

Considering the constraints, the *optimal solution* is

$$u^*(t) = \begin{cases} -1 & \text{if } \lambda_2(t) < -2r \\ \frac{1}{2}\frac{1}{r}\frac{\lambda_2(t)}{\lambda_0} & \text{if } -2r \le \lambda_2(t) \le 2r \\ 1 & \text{if } \lambda_2(t) > 2r \end{cases} \tag{4.80}$$

By comparing with the numerical solution in Fig. 4.10, we get the following time-dependent optimal solution u^*:

$$u^*(t) = \begin{cases} -1 & \text{for } t < \tau_1 \\ \frac{1}{2}\frac{1}{r}\frac{\lambda_2(t)}{\lambda_0} & \text{for } \tau_1 \le t \le \tau_2 \\ 0 & \text{for } \tau_2 < t < \tau_3 \\ \frac{1}{2}\frac{1}{r}\frac{\lambda_2(t)}{\lambda_0} & \text{for } \tau_3 \le t \le \tau_4 \\ 1 & \text{for } \tau_4 < t \end{cases} \tag{4.81}$$

The Conditions We evaluate conditions (4.34)–(4.37) here:

$$\begin{aligned} L_u^*[t] &= H_u^*[t] + \mu g_u^*[t] = 0 \\ &\Rightarrow -2ru(t) + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0 \end{aligned} \tag{4.82}$$

$$\begin{aligned} \dot{\lambda}(t) &= -L_x^*[t] \\ &\Rightarrow \begin{cases} \dot{\lambda}_1(t) = 2\lambda_0 q_1 x_1(t) \\ \dot{\lambda}_2(t) = 2\lambda_0 q_2 x_2(t) - \lambda_1(t) + \nu_1(t) - \nu_2(t) \end{cases} \end{aligned} \tag{4.83}$$

$$\begin{aligned} \mu(t) &= \begin{pmatrix} \mu_1(t) & \mu_2(t) \end{pmatrix}^T \ge 0 \\ \mu(t)g^*[t] = 0 &\Rightarrow (\mu_1(t) + \mu_2(t)) - (\mu_1(t) - \mu_2(t))u^*(t) = 0 \end{aligned} \tag{4.84}$$

$$\begin{aligned} \nu(t) &= \begin{pmatrix} \nu_1(t) & \nu_2(t) \end{pmatrix}^T \ge 0, \dot{\nu}(t) \le 0, \\ \nu(t)h^*(t) = 0 &\Rightarrow (\nu_1(t) + \nu_2(t))c - (\nu_1(t) - \nu_2(t))x_2(t) = 0 \end{aligned} \tag{4.85}$$

Similarly, (4.38) results in

$$\begin{aligned} &\frac{dH^*[t]}{dt} = \frac{dL^*}{dt} = L_t^*[t] \Rightarrow \\ &-2q_1x_1(t)x_2(t) - 2q_2x_2(t)u(t) - 2ru(t)\dot{u}(t) + \lambda_1(t)u(t) \\ &\quad + \dot{\lambda}_1(t)x_2(t) + \lambda_2(t)\dot{u}(t) + \dot{\lambda}_2(t)u(t) = 0 \end{aligned} \tag{4.86}$$

$$\begin{aligned} &\dot{u}(t)(\mu_1(t) - \mu_2(t)) + (\dot{\mu}_1(t) - \dot{\mu}_2(t))u(t) \\ &\quad - (\dot{\mu}_1(t) + \dot{\mu}_2(t)) + (\nu_1(t) - \nu_2(t))u(t) \\ &\quad + (\dot{\nu}_1(t) - \dot{\nu}_2(t))x_2(t) - c(\dot{\nu}_1(t) + \dot{\nu}_2(t)) = 0 \end{aligned} \tag{4.87}$$

Transversality conditions

$$\begin{aligned} \lambda(t_f^-) &= \beta b_x^*[t_f] + \gamma h_x^*[t_f] \\ \begin{pmatrix} \lambda_1[1^-] \\ \lambda_2[1^-] \end{pmatrix} &= I_{2\times 2}\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \\ \gamma \ge 0, \gamma h[t_f] = 0 &\Rightarrow (\gamma_1 + \gamma_2)c - (\gamma_1 - \gamma_2)x_2(t_f) = 0 \end{aligned} \tag{4.88}$$

Since $x_2(t_f) = 0$, we conclude

$$\gamma_1 = \gamma_2 = 0. \tag{4.89}$$

In view of (4.81), the solution is divided into different regions. We consider each case separately and calculate expressions for the states, the costates and the the multiplier functions using the knowledge of $u^*(t)$ in each specific region and considering conditions (4.82)–(4.87).

Case 1, $u^*(t) = -1,\ t < \tau_1$**:** Assuming that $x_i > 0$

$$\begin{aligned} x_1(t) &= -\frac{1}{2}t^2 + x_i \\ x_2(t) &= -t \end{aligned} \tag{4.90}$$

From (4.82) follows

$$2r + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0 \tag{4.91}$$

From (4.84) follows

$$\mu_1(t) = 0,\ \mu_2(t) \text{ free} \tag{4.92}$$

From (4.85) follows

$$\nu_1(t) = \frac{t-c}{t+c}\nu_2(t) \tag{4.93}$$

From (4.87) follows

$$2\dot{\mu}_1 + (t+c)\dot{\nu}_1 - (t-c)\dot{\nu}_2 + (\nu_1 - \nu_2) = 0 \tag{4.94}$$

From (4.83) follows

$$\begin{cases} \dot{\lambda}_1(t) = -q_1 t^2 + 2q_1 x_i \\ \dot{\lambda}_2(t) = -2q_2 t - \lambda_1(t) + \nu_1(t) - \nu_2(t) \end{cases} \tag{4.95}$$

Combined with (4.86) follows

$$\nu_1(t) = \nu_2(t) \tag{4.96}$$

From (4.93) and (4.96) follows

$$\nu_1(t) = \nu_2(t) = 0$$

From (4.95) follows

$$\begin{cases} \lambda_1(t) = -\frac{1}{3}q_1 t^3 - q_1 x_i t + K_1 \\ \lambda_2(t) = \frac{1}{12}q_1 t^4 - (q_1 x_i + q_2)t^2 + tK_1 + K_2, \end{cases} \tag{4.97}$$

where K_1 and K_2 are appropriate constants.

Case 2, $u^*(t) = 0,\ \tau_2 < t < \tau_3$**:**

$$\begin{aligned} x_1(t) &= -ct + K_6 \\ x_2(t) &= -c \end{aligned} \tag{4.98}$$

From (4.82) follows

$$\lambda_2(t) - \mu_1(t) + \mu_2(t) = 0 \tag{4.99}$$

From (4.84) follows

$$\mu_1(t) = \mu_2(t) = 0 \tag{4.100}$$

From (4.85) follows

$$\nu_1(t) = 0,\ \nu_2(t) \text{ free} \tag{4.101}$$

Equation (4.87) provides no new information. From (4.83) follows

$$\begin{aligned} \dot{\lambda}_1(t) &= -2q_1ct + 2q_1K_6 \\ \dot{\lambda}_2(t) &= -\lambda_1(t) - 2q_2c - \nu_2(t) \end{aligned} \tag{4.102}$$

Equation (4.86) gives the same equation for $\dot{\lambda}_1(t)$ as (4.83). From (4.99) and (4.100) we conclude $\lambda_2(t) = 0$. Considering this, (4.102) simplifies to

$$\nu_2(t) = -\lambda_1(t) - 2q_2c \tag{4.103}$$

$$\lambda_1(t) = -q_1ct^2 + K_6t + K_7 \tag{4.104}$$

Case 3, $u^*(t) = 1,\ t > \tau_4$**:**

$$\begin{aligned} x_1(t) &= \frac{1}{2}t^2 + K_9t + K_8 \\ x_2(t) &= t + K_9 \end{aligned} \tag{4.105}$$

From (4.82) follows

$$-2r + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0 \tag{4.106}$$

From (4.84) follows

$$\mu_1(t) \text{ free},\ \mu_2(t) = 0 \tag{4.107}$$

From (4.85) follows

$$(c-t)\nu_1(t) + (c+t)\nu_2(t) - (\nu_1(t) - \nu_2(t))K_9 = 0 \tag{4.108}$$

From (4.87) follows

$$\nu_1(t) - \nu_2(t) + (\dot{\nu}_1(t) - \dot{\nu}_2(t))(t - c + K_9) = 0 \tag{4.109}$$

From (4.83) follows

$$\begin{cases} \dot{\lambda}_1(t) = 2q_1(\frac{1}{2}t^2 + K_9 t + K_8) \\ \dot{\lambda}_2(t) = 2q_2(t + K_9) - \lambda_1(t) + \nu_1(t) - \nu_2(t) \end{cases} \tag{4.110}$$

Combined with (4.86) follows

$$\nu_1(t) = \nu_2(t) \tag{4.111}$$

From (4.108) and (4.111) follows $\nu_1(t) = \nu_2(t) = 0$. From (4.110) follows

$$\begin{cases} \lambda_1(t) = -\frac{1}{3}q_1 t^3 + q_1 K_9 t^2 + 2q_1 K_8 t + K_{10} \\ \lambda_2(t) = \frac{1}{12}q_1 t^4 - \frac{1}{3}q_1 K_9 t^3 + (q_2 - q_1 K_8)t^2 \\ \qquad\qquad +(2q_2 K_9 - K_{10})t + K_{11} \end{cases} \tag{4.112}$$

Case 4, $u^*(t) = \frac{1}{2r}\lambda_2(t),\ \tau_1 < t < \tau_2,\ \tau_3 < t < \tau_4$**:** From (4.82) follows

$$u(t) = \frac{\lambda_2(t) - \mu_1(t) + \mu_2(t)}{2r} \tag{4.113}$$

$$\Rightarrow \mu_1(t) = \mu_2(t) \tag{4.114}$$

From (4.84) follows

$$\mu_1(t) + \mu_2(t) = 0 \tag{4.115}$$

Therefore, $\mu_1(t) = \mu_2(t) = 0$.
From (4.85) follows

$$c(\nu_1(t) + \nu_2(t)) - (\nu_1(t) - \nu_2(t))x_2(t) = 0 \tag{4.116}$$

From (4.87) follows

$$(\nu_1(t) - \nu_2(t))\frac{\lambda_2(t)}{2r} + (\dot{\nu}_1(t) - \dot{\nu}_2(t))x_2(t) - c(\dot{\nu}_1(t) + \dot{\nu}_2(t)) = 0 \tag{4.117}$$

From (4.86) follows

$$\left(-2q_2 x_2(t) + \lambda_1(t) + \dot{\lambda}_2(t)\right)\lambda_2(t) = 0 \tag{4.118}$$

If $\lambda_2(t) \neq 0$,

$$-2q_2 x_2(t) + \lambda_1(t) + \dot{\lambda}_2(t) = 0. \tag{4.119}$$

Comparing with (4.83) and using (4.85), we conclude that $\nu_1(t) = \nu_2(t) = 0$. Using (4.119), system dynamics, and $u^*(t) = \frac{1}{2r}\lambda_2(t)$, we can conclude

$$\lambda_2^{(4)}(t) = \frac{1}{r}(q_2\ddot{\lambda}_2(t) - q_1\lambda_2(t)) \tag{4.120}$$

From this follows that

$$\lambda_2(t) = C_1 e^{-\sigma_1 t} + C_2 e^{\sigma_1 t} + C_3 e^{-\sigma_2 t} + C_4 e^{\sigma_2 t}, \tag{4.121}$$

with

$$\sigma_1 = \sqrt{\frac{q_2 + \sqrt{q_2^2 - 4rq_1}}{2r}}$$
$$\sigma_2 = \sqrt{\frac{q_2 - \sqrt{q_2^2 - 4rq_1}}{2r}},$$

and C_i are appropriate constants of integration. From the system equations (4.60)

$$\begin{aligned}
\dot{x}_2(t) =& u(t) \Rightarrow \\
x_2(t) =& \frac{1}{2r}\left(-\frac{C_1}{\sigma_1}e^{-\sigma_1 t} + \frac{C_2}{\sigma_1}e^{\sigma_1 t} - \frac{C_3}{\sigma_2}e^{-\sigma_2 t} + \frac{C_4}{\sigma_2}e^{\sigma_2 t}\right) + \kappa_1 \qquad (4.122) \\
\dot{x}_1(t) =& x_2(t) \Rightarrow \\
x_1(t) =& \frac{1}{2r}\left(\frac{C_1}{\sigma_1^2}e^{-\sigma_1 t} + \frac{C_2}{\sigma_1^2}e^{\sigma_1 t} + \frac{C_3}{\sigma_2^2}e^{-\sigma_2 t} + \frac{C_4}{\sigma_2^2}e^{\sigma_2 t}\right) \\
& + \kappa_1 t + \kappa_2 \qquad (4.123)
\end{aligned}$$

With (4.119) it follows

$$\begin{aligned}
\lambda_1(t) =& \left(\sigma_1 - \frac{q_2}{\sigma_1 r}\right)\left(C_1 e^{-\sigma_1 t} - C_2 e^{\sigma_1 t}\right) \\
& + \left(\sigma_2 - \frac{q_2}{\sigma_2 r}\right)\left(C_3 e^{-\sigma_2 t} - C_4 e^{\sigma_2 t}\right) + 2q_2\kappa_1 \qquad (4.124)
\end{aligned}$$

Additionally, substituting (4.123) and (4.124) into (4.83) results in

$$\begin{aligned}
& \left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 t} + C_2 e^{\sigma_1 t}\right) \\
& + \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 t} + C_4 e^{\sigma_2 t}\right) \\
& = 2rq_1(\kappa_1 t + \kappa_2) \qquad (4.125)
\end{aligned}$$

The equations derived in this part apply to two regions though with different constants of integration. For the first interval, $\tau_1 < t < \tau_2$, we will keep constants $C_i, i \in \{1..4\}$ unchanged but κ_1 and κ_2 will be substituted with K_{12} and K_{13}, respectively. For the second interval, $\tau_3 < t < \tau_4$, constants D_i, K_{14}, and K_{15} will be used.

Initial and Final Conditions: Initial condition on x are already used in (4.90). Final conditions for x:

$$x_1(1) = 0,\ x_2(1) = 0 \tag{4.126}$$

Therefore, from (4.105), when $u = 1$, follows

$$K_8 = \frac{1}{2},\ K_9 = -1 \tag{4.127}$$

Continuity of u: We evaluate the control signal at junction times from below and above and equate the expressions:

$$\begin{aligned} &u(\tau_1^+) = -1 \Rightarrow \lambda_2(\tau_1^+) = -2r \Rightarrow \\ &\quad C_1 e^{-\sigma_1 \tau_1} + C_2 e^{\sigma_1 \tau_1} + C_3 e^{-\sigma_2 \tau_1} + C_4 e^{\sigma_2 \tau_1} = -2r \end{aligned} \tag{4.128}$$

$$\begin{aligned} &u(\tau_2^-) = 0 \Rightarrow \lambda_2(\tau_2^-) = 0 \Rightarrow \\ &\quad C_1 e^{-\sigma_1 \tau_2} + C_2 e^{\sigma_1 \tau_2} + C_3 e^{-\sigma_2 \tau_2} + C_4 e^{\sigma_2 \tau_2} = 0 \end{aligned} \tag{4.129}$$

$$\begin{aligned} &u(\tau_3^+) = 0 \Rightarrow \lambda_2(\tau_3^+) = 0 \Rightarrow \\ &\quad D_1 e^{-\sigma_1 \tau_3} + D_2 e^{\sigma_1 \tau_3} + D_3 e^{-\sigma_2 \tau_3} + D_4 e^{\sigma_2 \tau_3} = 0 \end{aligned} \tag{4.130}$$

$$\begin{aligned} &u(\tau_4^-) = 1 \Rightarrow \lambda_2(\tau_4^-) = 2r \Rightarrow \\ &\quad D_1 e^{-\sigma_1 \tau_4} + D_2 e^{\sigma_1 \tau_4} + D_3 e^{-\sigma_2 \tau_4} + D_4 e^{\sigma_2 \tau_4} = 2r \end{aligned} \tag{4.131}$$

Continuity of x: We evaluate the states at junction times from below and above and equate the expressions:

$$\begin{aligned} &x_1(\tau_1^-) = x_1(\tau_1^+) \Rightarrow \\ &-\frac{1}{2}\tau_1^2 + x_i = \frac{1}{2r}\left(\frac{C_1}{\sigma_1^2} e^{-\sigma_1 \tau_1} + \frac{C_2}{\sigma_1^2} e^{\sigma_1 \tau_1} + \frac{C_3}{\sigma_2^2} e^{-\sigma_2 \tau_1} + \frac{C_4}{\sigma_2^2} e^{\sigma_2 \tau_1}\right) \\ &\qquad + K_{12}\tau_1 + K_{13} \end{aligned} \tag{4.132}$$

$$\begin{aligned} &x_2(\tau_1^-) = x_2(\tau_1^+) \Rightarrow \\ &-\tau_1 = \frac{1}{2r}\left(-\frac{C_1}{\sigma_1} e^{-\sigma_1 \tau_1} + \frac{C_2}{\sigma_1} e^{\sigma_1 \tau_1} - \frac{C_3}{\sigma_2} e^{-\sigma_2 \tau_1} + \frac{C_4}{\sigma_2} e^{\sigma_2 \tau_1}\right) + K_{12} \end{aligned} \tag{4.133}$$

$$\begin{aligned} &x_1(\tau_2^-) = x_1(\tau_2^+) \Rightarrow \\ &\frac{1}{2r}\left(\frac{C_1}{\sigma_1^2} e^{-\sigma_1 \tau_2} + \frac{C_2}{\sigma_1^2} e^{\sigma_1 \tau_2} + \frac{C_3}{\sigma_2^2} e^{-\sigma_2 \tau_2} + \frac{C_4}{\sigma_2^2} e^{\sigma_2 \tau_2}\right) + K_{12}\tau_1 + K_{13} \\ &= -c\tau_2 + K_6 \end{aligned} \tag{4.134}$$

$$x_2(\tau_2^-) = x_2(\tau_2^+) \Rightarrow$$
$$\frac{1}{2r}\left(-\frac{C_1}{\sigma_1}e^{-\sigma_1\tau_2} + \frac{C_2}{\sigma_1}e^{\sigma_1\tau_2} - \frac{C_3}{\sigma_2}e^{-\sigma_2\tau_2} + \frac{C_4}{\sigma_2}e^{\sigma_2\tau_2}\right) + K_{12} = -c \tag{4.135}$$

$$x_1(\tau_3^-) = x_1(\tau_3^+) \Rightarrow$$
$$-c\tau_3 + K_6 = \frac{1}{2r}\left(\frac{D_1}{\sigma_1^2}e^{-\sigma_1\tau_3} + \frac{D_2}{\sigma_1^2}e^{\sigma_1\tau_3} + \frac{D_3}{\sigma_2^2}e^{-\sigma_2\tau_3} + \frac{D_4}{\sigma_2^2}e^{\sigma_2\tau_3}\right)$$
$$+ K_{14}\tau_3 + K_{15} \tag{4.136}$$

$$x_2(\tau_3^-) = x_2(\tau_3^+) \Rightarrow$$
$$-c = \frac{1}{2r}\left(-\frac{D_1}{\sigma_1}e^{-\sigma_1\tau_3} + \frac{D_2}{\sigma_1}e^{\sigma_1\tau_3} - \frac{D_3}{\sigma_2}e^{-\sigma_2\tau_3} + \frac{D_4}{\sigma_2}e^{\sigma_2\tau_3}\right) + K_{14} \tag{4.137}$$

$$x_1(\tau_4^-) = x_1(\tau_4^+) \Rightarrow$$
$$\frac{1}{2r}\left(\frac{D_1}{\sigma_1^2}e^{-\sigma_1\tau_4} + \frac{D_2}{\sigma_1^2}e^{\sigma_1\tau_4} + \frac{D_3}{\sigma_2^2}e^{-\sigma_2\tau_4} + \frac{D_4}{\sigma_2^2}e^{\sigma_2\tau_4}\right) + K_{14}\tau_4 + K_{15}$$
$$= \frac{1}{2}\tau_4^2 + \frac{1}{2} - \tau_4 \tag{4.138}$$

$$x_2(\tau_4^-) = x_2(\tau_4^+) \Rightarrow$$
$$\frac{1}{2r}\left(-\frac{D_1}{\sigma_1}e^{-\sigma_1\tau_4} + \frac{D_2}{\sigma_1}e^{\sigma_1\tau_4} - \frac{D_3}{\sigma_2}e^{-\sigma_2\tau_4} + \frac{D_4}{\sigma_2}e^{\sigma_2\tau_4}\right) + K_{14} = \tau_4 - 1 \tag{4.139}$$

Continuity of λ: The adjoint function λ is not in general continuous. However, this example fulfills the condition of Proposition 4.2 in [Hartl et al., 1995], thus the continuity at junction times is guaranteed. From the dimensions of constraint functions $h(\cdot)$ and $g(\cdot)$, we have $s = q = 2$. The control signal $u^\star$ is assumed to be continuous and

$$\operatorname{rank}\begin{pmatrix} \partial g^*[\tau]/\partial u & \operatorname{diag}(g^*[\tau]) & 0 & 0 \\ \partial h^{1*}[\tau]/\partial & 0 & 0 & \operatorname{diag}(h^*[\tau]) \end{pmatrix} \tag{4.140}$$
$$= \operatorname{rank}\begin{pmatrix} -1 & 1-u(\tau) & 0 & 0 & 0 \\ 1 & 0 & 1+u(\tau) & 0 & 0 \\ -1 & 0 & 0 & c-x_2(\tau) & 0 \\ 1 & 0 & 0 & 0 & c+x_2(\tau) \end{pmatrix} = 4 = s+q \tag{4.141}$$

We evaluate the costates at junction times from below and above and equate the expressions:

$$\begin{aligned}
&\lambda_1(\tau_1^-) = \lambda_1(\tau_1^+) \Rightarrow \\
&-\frac{1}{3}q_1\tau_1^3 - q_1 x_i \tau_1 - K_1 = \left(\sigma_1 - \frac{q_2}{\sigma_1 r}\right)\left(C_1 e^{-\sigma_1\tau_1} - C_2 e^{\sigma_1\tau_1}\right) \\
&+\left(\sigma_2 - \frac{q_2}{\sigma_2 r}\right)\left(C_3 e^{-\sigma_2\tau_1} - C_4 e^{\sigma_2\tau_1}\right) + 2q_2 K_{12} \qquad (4.142) \\
&\lambda_2(\tau_1^-) = \lambda_2(\tau_1^+) \Rightarrow \\
&\frac{1}{12}q_1\tau_1^4 - (q_1 x_i + q_2)\tau_1^2 + \tau_1 K_1 + K_2 \\
&= C_1 e^{-\sigma_1\tau_1} + C_2 e^{\sigma_1\tau_1} + C_3 e^{-\sigma_2\tau_1} + C_4 e^{\sigma_2\tau_1} \qquad (4.143)
\end{aligned}$$

$$\begin{aligned}
&\lambda_1(\tau_2^-) = \lambda_1(\tau_2^+) \Rightarrow \\
&\left(\sigma_1 - \frac{q_2}{\sigma_1 r}\right)\left(C_1 e^{-\sigma_1\tau_2} - C_2 e^{\sigma_1\tau_2}\right) \\
&+\left(\sigma_2 - \frac{q_2}{\sigma_2 r}\right)\left(C_3 e^{-\sigma_2\tau_2} - C_4 e^{\sigma_2\tau_2}\right) + 2q_2 K_{12} \\
&= -q_1 c\tau_2^2 + K_6\tau_2 + K_7 \qquad (4.144) \\
&\lambda_2(\tau_2^-) = \lambda_2(\tau_2^+) \Rightarrow \\
&C_1 e^{-\sigma_1\tau_2} + C_2 e^{\sigma_1\tau_2} + C_3 e^{-\sigma_2\tau_2} + C_4 e^{\sigma_2\tau_2} = 0 \qquad (4.145)
\end{aligned}$$

$$\begin{aligned}
&\lambda_1(\tau_3^-) = \lambda_1(\tau_3^+) \Rightarrow \\
&-q_1 c\tau_3^2 + K_6\tau_3 + K_7 = \left(\sigma_1 - \frac{q_2}{\sigma_1 r}\right)\left(D_1 e^{-\sigma_1\tau_3} - D_2 e^{\sigma_1\tau_3}\right) \\
&+\left(\sigma_2 - \frac{q_2}{\sigma_2 r}\right)\left(D_3 e^{-\sigma_2\tau_3} - D_4 e^{\sigma_2\tau_3}\right) + 2q_2 K_{14} \qquad (4.146) \\
&\lambda_2(\tau_3^-) = \lambda_2(\tau_3^+) \Rightarrow \\
&0 = D_1 e^{-\sigma_1\tau_2} + D_2 e^{\sigma_1\tau_3} + D_3 e^{-\sigma_2\tau_3} + D_4 e^{\sigma_2\tau_3} \qquad (4.147)
\end{aligned}$$

$$\begin{aligned}
&\lambda_1(\tau_4^-) = \lambda_1(\tau_4^+) \Rightarrow \\
&\left(\sigma_1 - \frac{q_2}{\sigma_1 r}\right)\left(D_1 e^{-\sigma_1\tau_4} - D_2 e^{\sigma_1\tau_4}\right) + \left(\sigma_2 - \frac{q_2}{\sigma_2 r}\right)\left(D_3 e^{-\sigma_2\tau_4} - D_4 e^{\sigma_2\tau_4}\right) \\
&+2q_2 K_{14} = -\frac{1}{3}q_1\tau_4^3 - q_1\tau_4^2 + K_{10} + q_1\tau_4 \qquad (4.148)
\end{aligned}$$

$$\lambda_2(\tau_4^-) = \lambda_2(\tau_4^+) \Rightarrow$$
$$D_1 e^{-\sigma_1 \tau_2} + D_2 e^{\sigma_1 \tau_4} + D_3 e^{-\sigma_2 \tau_4} + D_4 e^{\sigma_2 \tau_4}$$
$$= \frac{1}{12} q_1 \tau_4^4 + \frac{1}{3} q_1 \tau_4^3 + (q_2 - \frac{1}{2} q_1) \tau_4^2 + (-2q_2 - K_{10}) \tau_4 + K_{11} \tag{4.149}$$

Evaluation of Equation (4.125)**:** Note that (4.125) must hold in Case 4. Therefore, it must hold even when the time approaches the junction times. Evaluation of this equation at junction times gives us four additional equations.

$$\left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 \tau_1} + C_2 e^{\sigma_1 \tau_1}\right)$$
$$+ \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 \tau_1} + C_4 e^{\sigma_2 \tau_1}\right)$$
$$= 2rq_1 \left(K_{12}\tau_1 + K_{13}\right) \tag{4.150}$$

$$\left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 \tau_2} + C_2 e^{\sigma_1 \tau_2}\right)$$
$$+ \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 \tau_2} + C_4 e^{\sigma_2 \tau_2}\right)$$
$$= 2rq_1 \left(K_{12}\tau_2 + K_{13}\right) \tag{4.151}$$

$$\left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 \tau_3} + C_2 e^{\sigma_1 \tau_3}\right)$$
$$+ \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 \tau_3} + C_4 e^{\sigma_2 \tau_3}\right)$$
$$= 2rq_1 \left(K_{14}\tau_3 + K_{15}\right) \tag{4.152}$$

$$\left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 \tau_4} + C_2 e^{\sigma_1 \tau_4}\right)$$
$$+ \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 \tau_4} + C_4 e^{\sigma_2 \tau_4}\right)$$
$$= 2rq_1 \left(K_{14}\tau_4 + K_{15}\right) \tag{4.153}$$

These conditions result in 24 unknowns and 24 equations, most of them nonlinear. By solving these equations, the junction times and the integration constants are determined and hence the solution to the optimal control problem.

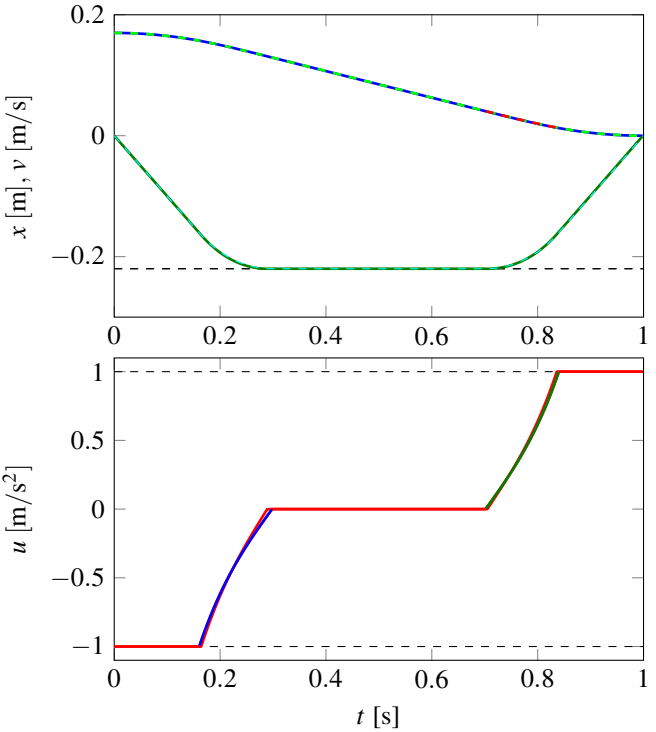

Figure 4.11 Comparison of the numerical and analytic solutions. The blue and green pieces in the lower plot belong to the analytic solution. The velocities and positions are almost indistinguishable.

Results

Except for two trivial equations which determine K_8 and K_9, the rest of the equations are nonlinear. Note that if there is no solution that satisfies $0 \leq \tau_1 \leq \tau_2 \leq \tau_3 \leq \tau_4 \leq 1$, other scenarios where there is no state/input saturation, must be considered. The problem is infeasible if there is no solution to either of these scenarios.

We found a numerical solution to these equations. The numerical solution obtained by the interior method and the analytic solution are compared in Fig. 4.11. To further compare the algorithms, we ran 10 experiments starting from a random initial guess for the solution of the nonlinear equations. The `fsolve` function in Matlab could find a solution on average in 1.27 [s] on an Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz running Fedora 20. The same problem was solved by using CVX package [Grant and Boyd, 2014; Grant and Boyd, 2008] in Matlab, with the sampling time of $h = 1$ [ms]. The interior point method approach took on average 14.232 [s], and including the overheads 53.753 [s]. The average cost obtained by CVX was

386.08h while the analytical approach resulted in 385.35h.

Conclusion

Since the Hamiltonian H to the fixed-time trajectory planning problem is concave, we can conclude that the solution to our example, found by the direct approach, is optimal, according to the Mangasarian-type sufficient condition [Mangasarian, 1966; Hartl et al., 1995]. Note that, although we used a numerical approach to find the constants, this approach is independent of the number of discretization points compared to the interior point method. In other words, there is an analytic expression for the solution that is parametrized by the unknowns. However, the complexity of this approach, without developing special software, makes it impractical for larger systems.

4.4 Model Predictive Control Approach

In this section, we study a scenario with a ball-catching robot [Linderoth et al., 2010; Linderoth, 2013], where the estimates of the contact position of the ball are delivered online using a vision system. The more vision data is collected, the lower is the uncertainty in the new estimates. Thus, a new trajectory needs to be computed when new estimates become available.

The focus of this section is trajectory generation for mechanical systems using a direct approach. More specifically, we propose an approach to trajectory generation based on Model Predictive Control (MPC) [Mayne et al., 2000; Maciejowski, 1999] and the receding-horizon principle. The MPC principle is used in our experiments to compute reference values (joint position and velocity reference values) for the underlying motion-control system. For a long time, MPC has been hampered by comparably long computation times, thus implying a lower bound on the sampling times that can be achieved. However, MPC problems with quadratic cost functions and linear constraints have been efficiently solved since more than ten years ago [Bemporad, 2004] and with recent advances in algorithms and computing power within milliseconds [Mattingley and Boyd, 2012; Boyd and Vandenberghe, 2004]. Therefore, solutions to motion-planning problems can be computed quickly under certain assumptions on the model and on the constraints, which enables real-time motion planning under high sampling frequencies. With this approach, it is also possible to find solutions to motion planning problems where analytic solutions are not available.

Problem Formulation

In many robotic applications, it is desired to achieve a certain state of the robot at a given time. This will result in a reference tracking problem, if the desired state is specified as a function of time during the whole execution. On the other hand, if the desired state is discontinuous in time, we need to do planning between the points,

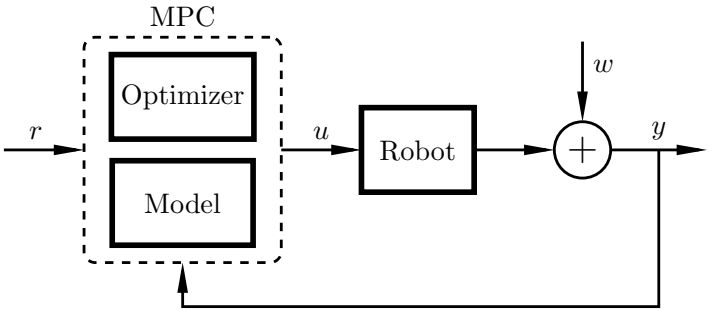

Figure 4.12 Trajectory generation using Model Predictive Control, r is the reference signal, u is the control signal, and w is the measurement noise.

i.e., transferring the robot from an initial state to the next state in a given time. In either case, the motion of the robot must fulfill certain constraints. Moreover, we wish to define a desired behavior, which can be specified by an objective function.

We propose using the model predictive approach to solve this type of problems. A central notion in MPC is using a model to predict the behavior of a system. Accordingly, it is possible to optimize the objective function over a receding horizon considering the predicted outputs. The optimization is usually carried out in a feedback loop and only the first control action of a whole sequence is applied [Maciejowski, 1999; Mayne et al., 2000]. Figure 4.12 shows a schematic of trajectory planning using MPC.

In this framework, we require an objective function, a model of the system, state constraints, and a time horizon. Following the presentation in [Mayne et al., 2000], the cost for state x at time k is defined as

$$V(x,k,\mathscr{U}) = \sum_{i=k}^{k+N-1} \ell(\boldsymbol{x}(i),u(i)) + F\left(x(k+N)\right), \tag{4.154}$$

where $\mathscr{U} = [u(k),u(k+1),\cdots,u(k+N-1)]^T$, N is the time horizon, and $\boldsymbol{x}(i)$ is the state trajectory resulting from an initial state $x(k)$ and a control sequence $\mathscr{U}$. The terminal time $k+N$ increases with time k which is referred to as a receding horizon. The system to be controlled is described by the following equations

$$x(k+1) = f(x(k),u(k)), \tag{4.155}$$
$$y(k) = h(x(k)). \tag{4.156}$$

The control and state sequence must satisfy

$$u(k) \in \mathbb{U} \tag{4.157}$$
$$x(k) \in \mathbb{X} \tag{4.158}$$
$$x(k+N) \in \boldsymbol{X}_f \subset \mathbb{X}, \tag{4.159}$$

where $\mathbb{U}$ is a compact subset of $\mathbb{R}^m$ and $\mathbb{X}$ is a closed subset of $\mathbb{R}^n$.

Given the initial state $x(k)$, the following optimization problem is solved in each iteration:

$$\begin{aligned} \underset{\mathscr{U}}{\text{minimize}} \quad & V(x,k,\mathscr{U}) \\ \text{subject to} \quad & (4.155),(4.156) \\ & u(i) \in \mathbb{U} \\ & x(i) \in \mathbb{X} \\ & x(k+N) \in \boldsymbol{X}_f \subset \mathbb{X} \end{aligned} \tag{4.160}$$

It is desired to investigate how the MPC framework is applicable to the trajectory generation for point-to-point and reference tracking problems. Furthermore, we wish to find a set of assumptions and methods which allow for real-time implementation.

Methods

In this section, we introduce instances of the MPC problem which are suitable for fast trajectory generation. In order to establish a concrete example at the end of this section, various aspects of the MPC problem are explained.

Linear Models with Quadratic Cost Functions

In this case, the system dynamics are assumed to be linear and given by

$$x(k+1) = A_d x(k) + B_d u(k) \tag{4.161}$$

$$y(k) = C_d x(k) + D_d u(k) \tag{4.162}$$

$$z(k) = \tilde{C} x(k) + \tilde{D} u(k), \tag{4.163}$$

where $z(k)$ denotes the controlled states [Maciejowski, 1999].

Given the system equations, we can formulate the following cost function for the trajectory generation problem:

$$V(x,k,\mathscr{U}) = \sum_{i=k}^{k+N} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)}, \tag{4.164}$$

where the norm is defined as $\|a\|^2_W = a^T W a$ and Q and R are time-dependent weight matrices. The reference signal is denoted by $r(i)$.

We also consider linear constraints

$$F\mathscr{U}(k) \leq f, \tag{4.165}$$

$$G\mathscr{Z}(k) \leq g, \tag{4.166}$$

where $\mathscr{Z}(k) = [z(k+1), z(k+2), \cdots, z(k+N)]^T$, F, and G are matrices whose dimensions are determined by the number of constraints and the time horizon.

Note that in (4.164), r does not need to define the desired controlled states at every sample. To make this clear, assume that $\Psi(k) \subset \{k, \cdots, k+N\}$ denotes the set of indices for which there are desired values for the control states. Accordingly, we can rewrite (4.164) as

$$V(x,k,\mathscr{U}) = \sum_{i\in\Psi(k)} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k}^{k+N} \|z(i)\|^2_{\bar{Q}(i)} + \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)}, \tag{4.167}$$

where, $\bar{Q}(i) = Q(i)$ if $i \notin \Psi(k)$ and $r^T(i)\bar{Q}(i)r(i) = 0$, if $i \in \Psi(k)$.

For reference tracking problems, usually the first and the last terms are important. Provided that constraints for the desired controlled states are defined, the first term can be ignored in the point-to-point trajectory generation. Nevertheless, if soft constraints are preferred, all of the terms might be used. The benefit is immediate if the optimization problem with explicit constraints on $z(i)$, $i \in \Psi$, is infeasible.

Since the optimization runs in the control loop, it is possible to account for changes in the target or deviation of the robot. If perfect tracking for the robot is assumed, the loop can be closed around the model too, providing an open-loop control strategy.

Although it is possible to introduce constraints on the states at any discrete time, we might limit ourselves to the value at the final time, *i.e.*, $z(k+N)$. This strategy is advantageous if no information about the next target is available ahead in time or we do not want this information to impact the current decision. In such cases, we can successively reduce the sampling time while keeping the number of discretization points constant—*i.e.*, the time horizon in discrete time remains constant while the time horizon in continuous time gradually decreases. This allows for improving the resolution of the solution as the system follows the trajectory towards the target state.

General Considerations

Convexity and Optimality In general, there might exist several local optima to the underlying optimization problem. The global optimum may not be easy to find, considering a solution exists. In case of multiple local optima, it is also required to use approaches such as a warm start of the optimization to avoid jumping between different solutions. On the other hand, at the cost of limiting the scope of the problems, we can use a convex cost function and convex and compact constraint sets, as were used here. Under these conditions, we are assured that if a solution exists, it is globally optimal [Boyd and Vandenberghe, 2004].

Models The models for trajectory planning must specify the relation between the actuation and the motion. The motion can be specified in any set of generalized

coordinates. However, choosing a certain coordinate system can significantly simplify the equations. Also, depending on the application it is sometimes more natural to use for example Cartesian coordinates rather than joint values. Furthermore, the motion can be specified in terms of position, velocity, or higher order time derivatives.

A control variable can be a dynamic variable such as force and torque. Typically, the dynamic equations are highly non-linear. Without using any form of approximation (*e.g.*, linearization), this fact is usually a limiting factor for many optimization algorithms.

Cost Function In general, the cost does not need to have a physical interpretation. For trajectory generation, we might consider punishing high values of the acceleration or velocities. However, we might choose the cost $V(x,k,\mathscr{U})$ to model, for instance, the distance traveled by a robot or the mechanical energy.

Let ΔX denote the changes in the vector of position, then the length is approximately proportional to $(\sum \Delta X^2)^{1/2}$. Also, introducing cost in the form of $\sum \dot{X}^2$ can lead to keeping the kinetic energy low.

Constraints In addition to the constraints related to the final state or the states of the via points, physical constraints can be included. Bounds on the actuation, velocity, and joint angles can be expressed directly as state or control constraints. Workspace and obstacles can also be considered. Specially, a typical workspace can be expressed as a set of linear equations

$$G \begin{bmatrix} X(k) \\ 1 \end{bmatrix} \leq 0 \tag{4.168}$$

where X is a subset of coordinates and G is a matrix. By including velocities in X, it is possible to define velocity dependent boundaries too. Obstacles are more conveniently approximated by ellipsoids as below

$$X^T G X \geq 1. \tag{4.169}$$

These type of constraints are not convex.

Interpolation of Trajectories Due to time constraints of real-time systems, it is always required to have a valid trajectory. This is true even if the optimization algorithm fails. This can happen if the problem is infeasible or the maximum number of iterations has been reached. Therefore, we consider piece-wise trajectories. In other words, the previous trajectory is valid until the new one is ready to be switched to. If there is no new point, the robot stays still in its final position.

Furthermore, it is required to interpolate between calculated trajectory points. Interpolation makes it possible to have a lower sampling rate for the optimization than for the controlled system. Assuming T is a vector of increasing time instants

and $\mathscr{X}$ the corresponding states,

$$T = \begin{pmatrix} t_1, & t_2, & \cdots, t_n \end{pmatrix} \tag{4.170}$$

$$\mathscr{X} = \begin{pmatrix} x_1, & x_2, & \cdots, x_n \end{pmatrix}, \tag{4.171}$$

we can consider a linear interpolation such that

$$x(t) = x_n + \frac{x_{n+1} - x_n}{t_{n+1} - t_n} t, \qquad t_n \leq t < t_{n+1} \tag{4.172}$$

Discretization In the MPC framework, the dynamics of a system are described by difference equations. Assuming the following linear model and h as the sampling time,

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.173}$$

$$y(t) = Cx(t), \tag{4.174}$$

the discretized equations of the system by the predictive first-order-hold sampling method [Åström and Wittenmark, 2011] can be obtained as follows

$$x(k+1) = \Phi x(k) + \frac{1}{h}\Gamma_1 u(k+1) + (\Gamma - \frac{1}{h}\Gamma_1)u(k), \tag{4.175}$$

$$y(k) = Cx(k), \tag{4.176}$$

where

$$\begin{pmatrix} \Phi & \Gamma & \Gamma_1 \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \end{pmatrix} \exp\left(\begin{pmatrix} A & B & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{pmatrix} h \right). \tag{4.177}$$

Note that (4.175) can be rewritten in the standard form by a change of variables

$$\zeta(k+1) = \Phi\zeta(k) + \left(\Gamma + \Phi(I - \frac{1}{h}\Gamma_1) \right) u(k) \tag{4.178}$$

$$y(k) = C\zeta(k) + (\frac{1}{h}C\Gamma_1)u(k). \tag{4.179}$$

This choice of discretization is motivated by the linear interpolation of the trajectories as described earlier.

Example

To illustrate the approach, we consider a robotic model concerning only the kinematic variables. Accordingly, a multi-dimensional multi-stage integrator can be used as a model. Since robots have various structures, we choose generalized coordinates q for the representation of the states. In this way, q might represent either the

joint values or the Cartesian coordinates depending on the application. Specifically, we choose the state vector as

$$x = \left[q_1, \dot{q}_1, \ddot{q}_1, \cdots, q_d, \dot{q}_d, \ddot{q}_d\right]^T, \tag{4.180}$$

where d is the number of DOF.

The continuous-time model according to (4.174) for triple integrators for d DOF is specified by the matrices,

$$A = \text{diag}\left([\tilde{A}, \cdots, \tilde{A}]\right), \quad B = [\tilde{B}^T, \cdots, \tilde{B}^T]^T, \tag{4.181}$$

$$C = I_{3d} \tag{4.182}$$

where $A \in \mathbb{R}^{3d\times 3d}$, $B \in \mathbb{R}^{3d\times 1}$, and

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \tilde{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.183}$$

Assuming $\tilde{C} = I_{3d}$, the matrices for the discretized model concerning the controlled states can be calculated as

$$\Phi = \text{diag}\left([\tilde{\Phi}, \cdots, \tilde{\Phi}]\right), \quad \Gamma_1 = [\tilde{\Gamma_1}^T, \cdots, \tilde{\Gamma_1}^T]^T, \tag{4.184}$$

$$\Gamma = [\tilde{\Gamma}^T, \cdots, \tilde{\Gamma}^T]^T, \tag{4.185}$$

where $\Phi \in \mathbb{R}^{3d\times 3d}$, $\Gamma_1, \Gamma \in \mathbb{R}^{3d\times 1}$, and

$$\tilde{\Phi} = \begin{bmatrix} 1 & h & \frac{h^2}{2} \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \frac{1}{h}\tilde{\Gamma}_1 = \begin{bmatrix} \frac{h^3}{24} \\ \frac{h^2}{6} \\ \frac{h}{2} \end{bmatrix}, \tilde{\Gamma} = \begin{bmatrix} \frac{h^3}{6} \\ \frac{h^2}{2} \\ h \end{bmatrix}. \tag{4.186}$$

In this problem, it is straightforward to include constraints on linear combinations of the kinematic variables. From a practical perspective, we consider limiting joint values, velocities, accelerations, and jerks. We form matrices F and G in (4.165) and (4.166), respectively, such that

$$|u(i)| \leq \left[u^1_{\max}, \quad \cdots, \quad u^n_{\max}\right]^T, \qquad k \leq i < k+N \tag{4.187}$$

$$z^j_{\max} = \left[q^j_{\max}, \quad v^j_{\max}, \quad a^j_{\max}\right], \qquad 1 \leq j < d \tag{4.188}$$

$$|z(i)| \leq \left[z^1_{\max}, \quad \cdots, \quad z^d_{\max}\right]^T, \qquad k \leq i < k+N \tag{4.189}$$

$$z(k+N) = r_f, \tag{4.190}$$

where r_f is the desired final state.

Now, assuming (4.167) with $\Psi(k) = \emptyset$ and $\bar{Q}(k+N) = 0$, *i.e.*, ignoring the first term and the cost on the final state, we have a full specification of the MPC problem for point-to-point trajectory generation. Note that the desired target state is introduced as a constraint on the final state. This means that we can successively reduce the sampling time, as described in the last paragraph in Section 4.4.

Implementation

The method proposed in Section 4.4 was implemented and experimentally evaluated on an industrial robot system. The system consisted of an IRB140 industrial manipulator [ABB Robotics, 2014], combined with an IRC5 control cabinet. The control system was further equipped with a research interface, ExtCtrl [Blomdell et al., 2010], in order to enable implementation of the proposed trajectory-generation method as an external controller. The research interface permits low-level access to the joint-position and velocity controllers in the axis computer of the control cabinet at a sample rate of 4 [ms]. More specifically, reference values for the joint positions and velocities can be specified and sent to the joint controllers, while the measured joint positions and velocities and the corresponding reference joint torques were sent back to the external controller from the main control cabinet with the same sample rate. The implementation of the trajectory generation was made in the programming language Java and the communication with the robot controller was handled using the LabComm communication protocol [*LabComm Protocol* 2014].

The inverse kinematics of the industrial manipulator is required in order to transform the desired final point in Cartesian space to joint space that can be used in the motion planning. The required kinematics were also available in Java from a previous implementation [Linderoth, 2013].

For implementation of the solution of the MPC optimization problem in Section 4.4, the CVXGEN code generator [Mattingley and Boyd, 2012] was used. The generator produced C code, which subsequently was interfaced with the Java program using the Java Native interface. The generated C code was optimized for performance in terms of time complexity. It enabled solution of the required quadratic programs in the MPC within 0.5 [ms]—*i.e.*, each optimization cycle including overhead required approximately 1–4 [ms] (for four DOF used for the considered task and a time horizon of 20 samples) on a standard personal computer with an Intel i7 processor with four cores. In order to preserve the numerical robustness of the solver also in the case of short sampling periods, a scaling of the equations, the inputs, and the state constraints was introduced in the optimization. This was important considering the fact that the matrices might be poorly conditioned close to the final time.

For evaluating the performance of the proposed trajectory-generation approach in a challenging scenario, we considered the task of catching balls with the robot. We employed the computer-vision algorithms and infrastructure developed in [Linderoth, 2013] for detection of the balls thrown and prediction of the target point. Two cameras detected the ball thrown towards the robot, and the image analysis algorithm estimated the position and velocity which provided the basis for the prediction of the target state. A photo of the experimental setup is shown in Fig. 4.13.

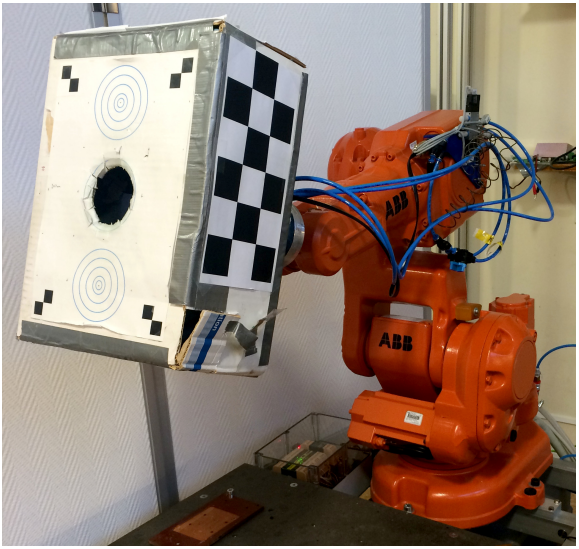

Figure 4.13 The experimental setup used for evaluation of the trajectory generation method. The task is to catch a ball in the box on the end-effector; two cameras (not visible in the figure) were used for detection of the ball.

Experiments and Results

The proposed approach to trajectory generation was evaluated in three different experiments in order to validate the correctness and to quantify the performance. The experiments concerned trajectory generation on joint level. The first experiment was performed in simulation, whereas the second and the third experiments were executed on the robot setup described in Section 4.4. In the experiments on the robot, the open-loop strategy was employed—*i.e.*, no feedback from the robot measurements was used. The time horizon was set to 20 samples. We assumed no coupling between the degrees of freedom. Therefore, for each degree of freedom we employed the constant weighting matrices $Q = \text{diag}([0, 1, 1])$ and $R = 0.001$.

Single Target Point In the first experiment, a target state was provided to the trajectory generator. Starting at rest at an angle of $q = 0$ [rad], the desired final state was to reach $q = 1$ [rad] at $t = 1$ [s] with $v = 0.5$ [rad/s] and $a = 0$ [rad/s^2]. The constraints in the optimization were chosen as $q_{\max} = 2$ [rad], $v_{\max} = 1.2$ [rad/s], $a_{\max} = 100$ [rad/s^2], and $u_{\max} = 250$ [rad/s^3]. The results of the experiment are provided in Fig. 4.14. Initially, a trajectory with comparably low time resolution was computed as an approximation of the true trajectory for transferring the system from the initial state to the final state. With a period of 200 [ms], the same target state was sent to the trajectory generator. A new optimal trajectory was computed with the initial state being the state at the current time from the previously computed trajectory.

This resulted in a new trajectory with increased time resolution. Considering that this procedure was repeated during the execution, a sequential refinement of the trajectory was achieved when moving closer to the target state where accuracy was required, see Fig. 4.14.

Sequence of Target Points In the second experiment, a sequence of different target states was provided to the trajectory generator, with 2 [s] between each new target state. The different target points were located at the perimeter of a rectangle centered at the home position of the robot end-effector in the workspace. Thus, these targets required motion along different Cartesian directions of the robot. Each target state was desired to be reached after 200 [ms], with the robot being at rest. The constraints in the optimization were chosen as $q_{\max} = 2$ [rad], $v_{\max} = \pi$ [rad/s], $a_{\max} = 45$ [rad/s^2], and $u_{\max} = 1500$ [rad/s^3]. Each target state was sent with a sample period of 20 [ms], resulting in the successive refinement of the trajectory as described in the previous subsection. Once the robot reached the target state and paused for 50 [ms], a new trajectory for returning to the home position was executed. The resulting trajectories obtained for the angle, velocity, and acceleration of joint 1 and 2 are shown in Fig. 4.15.

In order to further evaluate the performance of the trajectory generator, a detailed view of the position reference given by the optimal trajectory and the corresponding measured joint position for joint 2 is depicted in Fig. 4.16. The figure indicates the time instants at which the target state was sent and consequently when a new trajectory generation was performed. The computation time for each trajectory generation was below the sample rate of the robot, which meant that the optimal trajectory could be executed immediately.

Ball-Catching Experiments The most demanding evaluation performed was the computation of optimal trajectories for the robot to catch balls thrown towards it. The time period from the detection of the ball until it reached the robot was distributed in the interval $[200, 800]$ [ms]. Hence, the quality of the task depended on the satisfaction of the real-time constraints in order to transfer the robot from the home position to the desired final state at the desired, predicted arrival time. During the throws, as soon as new data from the vision sensors were available, the estimated contact positions (and the corresponding arrival times) of the ball were updated, leading to sequential recomputations of the optimal trajectory. Since not all DOF of the robot were required for the task, joints 4 and 6 were not used during the experiment. The trajectories were generated such that the robot should be at rest at the target point at the predicted arrival time with some margin. After reaching the final target, the robot paused there shortly in order to catch the ball, and finally returned to the home position. In the case that the estimated arrival time was already passed, no optimal trajectory was computed. If the robot given the velocity and acceleration constraints could not meet the arrival time—*i.e.*, the optimization problem was infeasible—we still commanded the robot to the target position at an estimate of the minimum required time. This improved the ball-catching performance since the

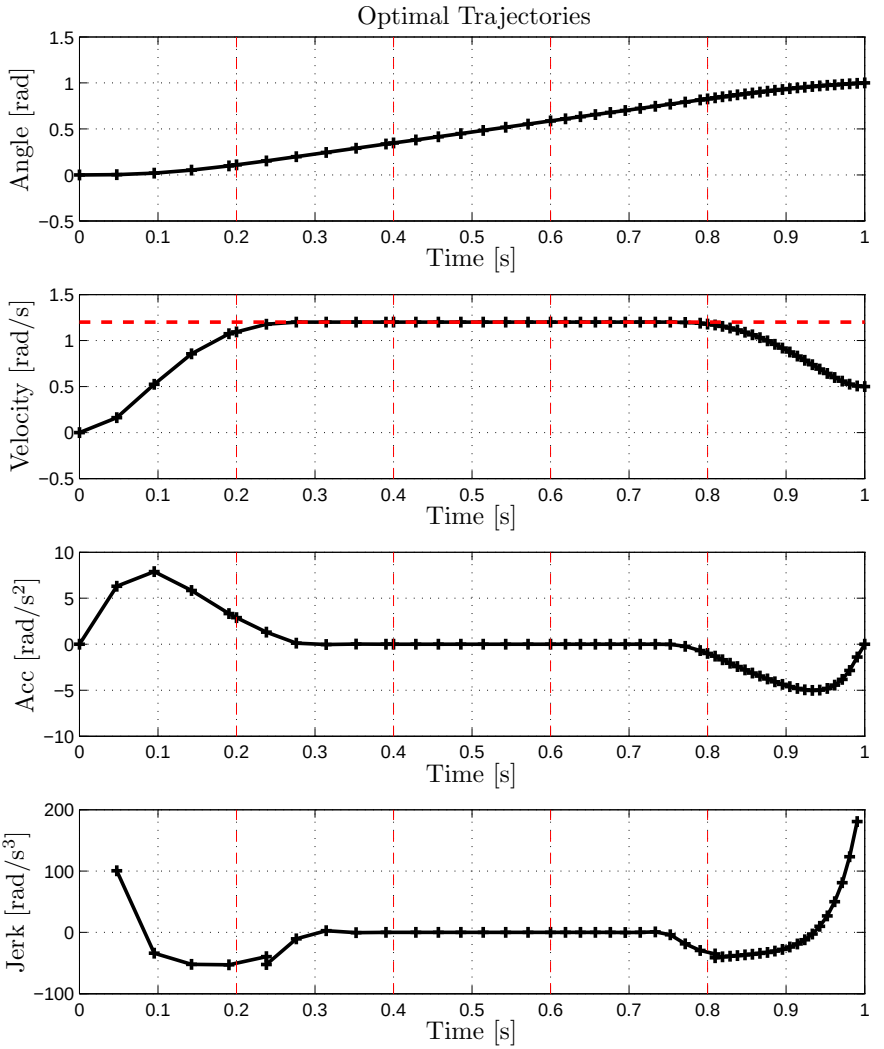

Figure 4.14 Results from a simulation where the same target point was sent to the trajectory generator at the time instants indicated by the vertical, dashed red lines. A clear increase in the time resolution of the trajectory is observed when moving closer to the target. Also, it is clear that the constraint on the velocity is active during a major part of the motion.

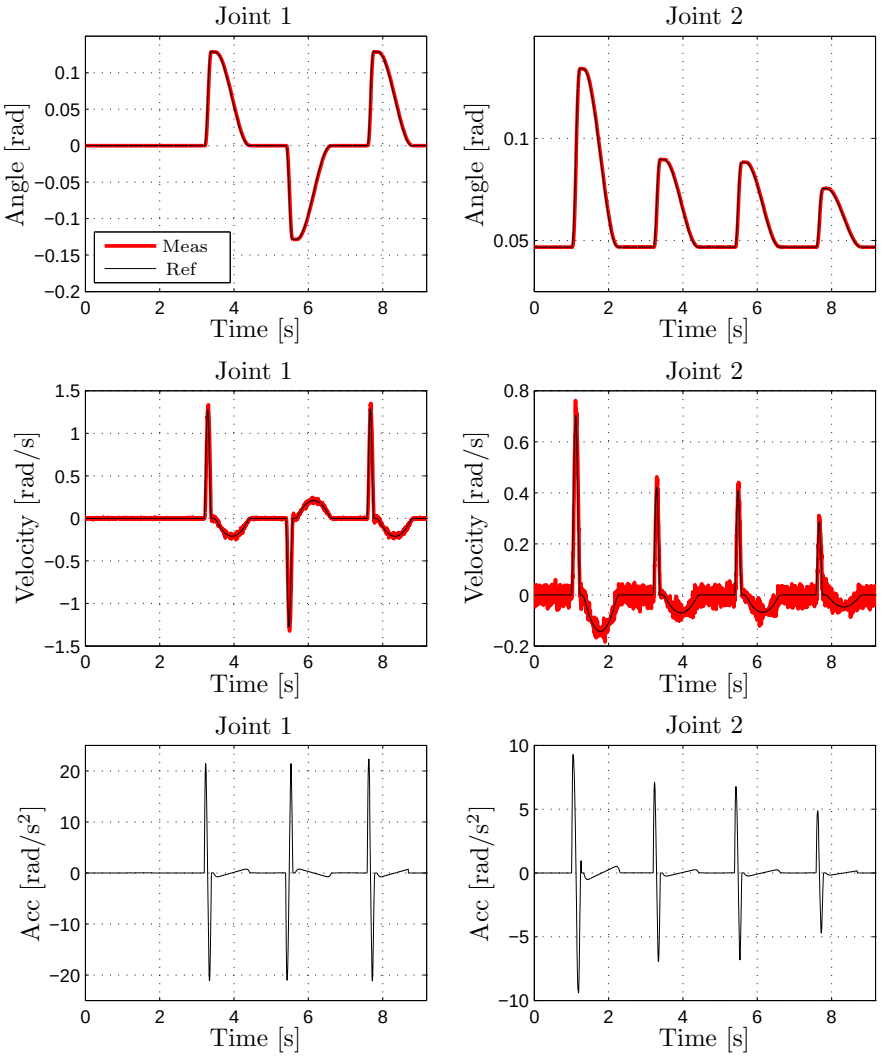

Figure 4.15 Experimental results from joint 1 and 2 of the robot, where a sequence of four target states were sent to the trajectory generator, each followed by a new target state coinciding with the home position of the robot. The good tracking of the computed optimal trajectories is clear from the experiments with different target states, corresponding to different points in the robot workspace.

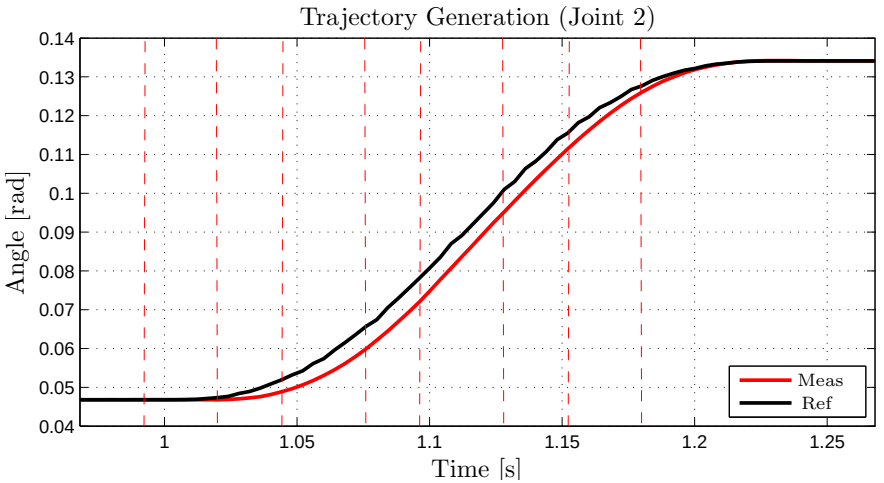

Figure 4.16 A detailed view of one of the motion segments in Fig. 4.15. The time instants at which the target state was sent are indicated by the vertical, dashed red lines in the figure. The delay between the reference and the performed trajectory is ≈ 1 [ms].

initial target point estimates had larger uncertainty. By moving towards the target point, there was a higher chance of catching the ball as more accurate estimates were obtained. Moreover, the constraints in the optimization were chosen based on physical considerations of the joint properties as $q_{\max} = 2$ [rad], $v_{\max} = \pi$ [rad/s], $a_{\max} = 45$ [rad/s^2], and $u_{\max} = 1500$ [rad/s^3].

Several experiments were performed with balls thrown with random initial velocities and along different directions. The results with regard to trajectory generation from one representative experiment are presented in Fig. 4.18 for the joints that were active in the robot motion. It is clear that the robot tracks the position and velocity references computed by the trajectory generator closely. In the figure, the time instants at which new sensor data arrived are also indicated. During the motion of the ball towards the robot, the computer-vision algorithm sent the current estimate of the contact point at an approximate sampling period of 4 [ms], initiating a trajectory generation with a possibly updated target state. The online replanning is also visible in the trajectory data shown in Fig. 4.18. The figure also shows the time margin—*i.e.*, the difference between the arrival time and the earliest possible time for reaching the ball, given the constraints.

In order to verify that the real-time constraints of the trajectory generation method were satisfied in this task, the computation times were measured for 100 cycles of optimization during a sequence of ball throwing. The results are visualized in Fig. 4.19. It can be observed that all computations are within the sample period of the robot at 4 [ms] with the average of 2.7 [ms].

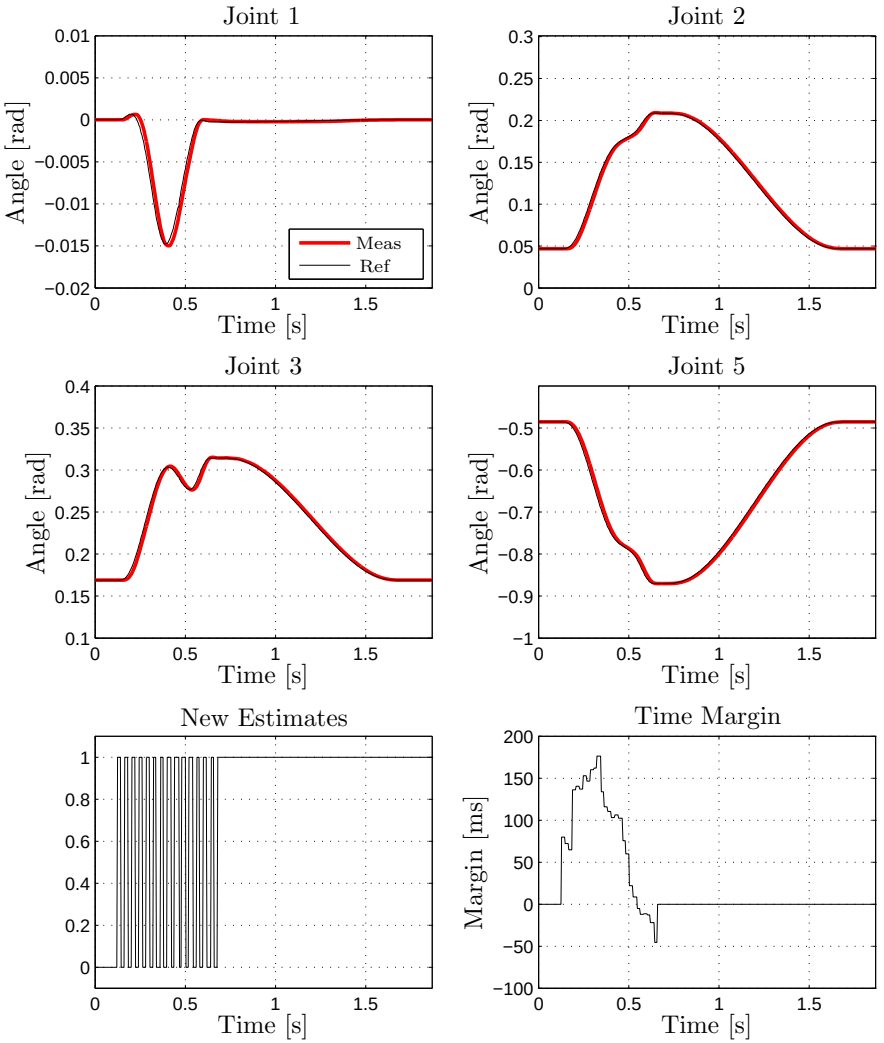

Figure 4.17 Results from the ball-catching experiment. As the ball moved towards the robot, new estimates of the contact position and the arrival time were obtained, thus the trajectory was recalculated. A new estimate was obtained on the rising or falling edge of the signal shown in the lower left plot. The lower right plot shows the time margin as defined in the text.

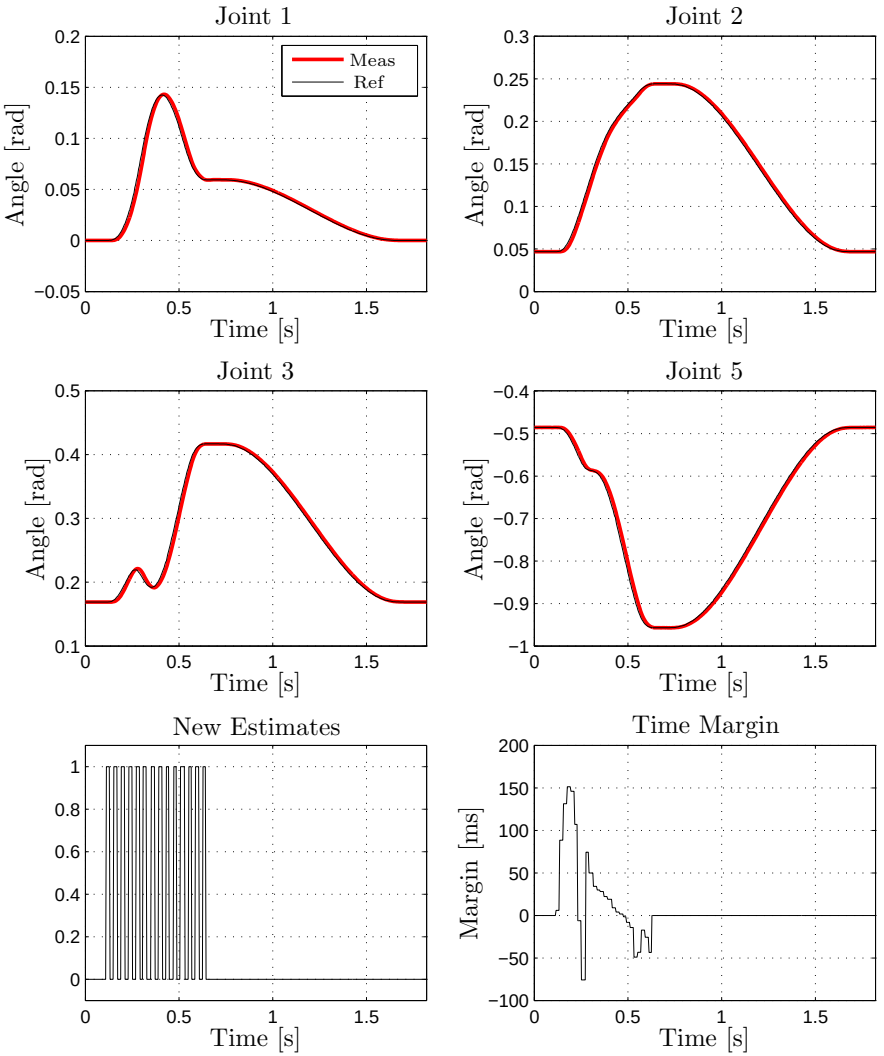

Figure 4.18 Results from the ball-catching experiment. As the ball moved towards the robot, new estimates of the contact position and the arrival time were obtained, thus the trajectory was recalculated. A new estimate was obtained on the rising or falling edge of the signal shown in the lower left plot. The lower right plot shows the time margin as defined in the text.

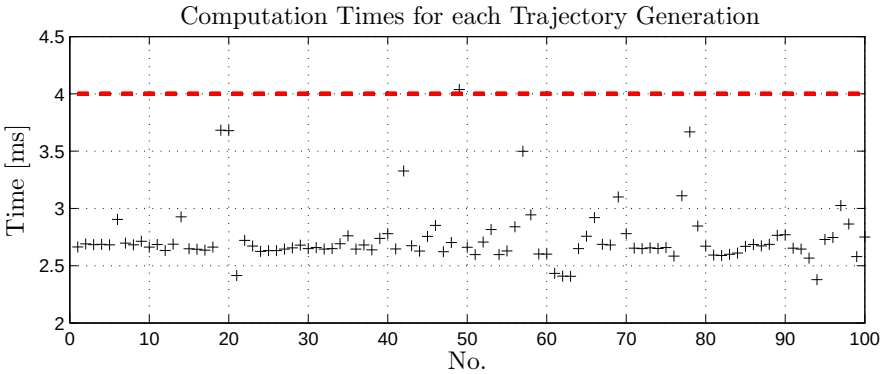

Figure 4.19 The computation times for the trajectory generation during ball-catching experiments for 100 different executions (four joints were employed). The horizontal, dashed red line represents the sample period of the robot system, i.e., the deadline for the trajectory generation in real-time.

Discussion

As an alternative to previously suggested methods for online trajectory generation with real-time constraints for mechanical systems, we have proposed an approach based on the idea of Model Predictive Control. The main characteristic of the proposed method is that it allows generation of trajectories that are optimal with respect to a quadratic cost function, and satisfying linear constraints on the input and state variables. In contrast to the typical application of MPC for tracking purposes, we have considered a trajectory generation perspective. Experimental evaluations were carried out in a demanding and time-critical ball-catching task with online updates of the estimated target state from a vision system as the ball approached the robot. The computation times were below the sample rate of the robot system used for the evaluation, enabling real-time trajectory generation.

Another characteristic feature of our algorithm is that it improves the accuracy of the optimal trajectory when approaching the desired final state, by increasing the time resolution. Hence, even though a small number of discretization points are used in the initial trajectory generation, a successive refinement of the trajectory is achieved. In the case of modeling uncertainty and disturbances in the online robot execution—*i.e.*, discrepancies between the computed optimal trajectories and the robot trajectories—feedback from the measurements of the actual trajectory tracking would be beneficial. This would be possible based on our algorithm as well; however, it requires a theoretical analysis of the dynamics of the closed-loop trajectory generation in order to guarantee stability under given assumptions on the model uncertainty and the disturbance characteristics. This would require using a robust MPC approach [Maciejowski, 1999].

Compared to previously suggested methods for real-time generation of trajecto-

ries, in particular [Kröger and Wahl, 2010; Haschke et al., 2008], and [Macfarlane and Croft, 2003], our method gives the solution of the fixed-time problem and adds more freedom in the formulation of the motion-planning problem since it allows quadratic cost functions and arbitrary linear constraints. The computation times for our method is longer than the times reported for the algorithms in the mentioned references. In our implementation, no effort was made to optimize the code with respect to the overhead in the implementation. Despite this, it satisfied the real-time requirements of 4 [ms] [Blomdell et al., 2010]. In cases where the DOF can be decoupled, such as for the implementation presented in this section, an efficient way to reduce the time-complexity is to distribute the computation for each DOF on the different cores of the CPU. This would allow to scale the proposed algorithm to a high number of DOF, since the major part of the computation time is spent on solving convex optimization problems. Moreover, it allows for an increased resolution of the discretization grid, if prompted by the accuracy requirements of the task.

Finally, since the estimate of the ball positions are inaccurate in the beginning, we can imagine improving the performance of ball-catching by a two-step approach. First a rough planning with a punishment on the deviation from the target state is performed. As soon as we have a low enough variance in the target point estimate, it is possible to perform an accurate trajectory planning with an equality constraint on the target state. By this strategy, the robot starts moving already from the first estimate toward the target, so increasing the chance of catching it later.

4.5 Cartesian Space Planning

Constraints related to each joint, such as maximum angles or angular velocities, are naturally expressed in the joint space. On the other hand, constraints on the motion of the end-effector are more conveniently expressed in the Cartesian space. Ideally, both joint-space and Cartesian-space constraints could exist and the relation between the variables could be established by forward or inverse kinematics. However, due to the complexity of the forward and inverse kinematics, this might lead to difficult nonlinear equations or in general irregular and non-convex constraint sets in one or the other variable sets.

In this section, we present an example to show that certain geometrical constraints in the task space could be approximated by a convex set in the joint space. In such case, numerical methods can effectively find a solution for the planning problem. For example, if the workspace limits could be approximated by a convex set in the joint space, the full power of the MPC approach for planning a trajectory to ball-catching scenario could be utilized.

Assume a robotic arm with three degrees of freedom with DH parameters given in Table 4.2. The inverse kinematics of this arm with all 8 solutions are given by [Siciliano et al., 2009]

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	q_1
2	a_2	0	0	q_2
3	a_3	0	0	q_3

Table 4.2 DH parameters for an anthropomorphic arm

$$c_3 = \frac{x^2+y^2+z^2-a_2^2-a_3^2}{2a_2a_3} \tag{4.191}$$

$$s_3 = \sqrt{1-c_3^2} \tag{4.192}$$

$$b = \sqrt{x^2+y^2} \tag{4.193}$$

$$q_{3,I} = \text{Atan2}\,(s_3, c_3) \tag{4.194}$$

$$q_{3,II} = \text{Atan2}\,(-s_3, c_3) \tag{4.195}$$

$$q_{2,I} = \text{Atan2}\,((a_2+a_3c_3)z - a_3s_3b, (a_2+a_3c_3)b + a_3s_3z) \tag{4.196}$$

$$q_{2,II} = \text{Atan2}\,((a_2+a_3c_3)z + a_3s_3b, -(a_2+a_3c_3)b + a_3s_3z) \tag{4.197}$$

$$q_{2,III} = \text{Atan2}\,((a_2+a_3c_3)z - a_3(-s_3)b, (a_2+a_3c_3)b + a_3(-s_3)z) \tag{4.198}$$

$$q_{2,IV} = \text{Atan2}\,((a_2+a_3c_3)z + a_3(-s_3)b, -(a_2+a_3c_3)b + a_3(-s_3)z) \tag{4.199}$$

$$q_{1,I} = \text{Atan2}\,(y, x) \tag{4.200}$$

$$q_{1,II} = \text{Atan2}\,(-y, -x)\,. \tag{4.201}$$

Limiting ourselves to the $(q_{1,I}, q_{2,I}, q_{3,I})$-configuration, Fig. 4.20 shows an approximation of a thin spherical cap (a portion of a sphere cut off by a plane) in the Cartesian space by another spherical cap in the joint space. In this example, the parameters are $a_2 = 1.0$ [m] and $a_3 = 1.4$ [m].

4.6 Conclusions

Various trajectory generation problems including tracking, point-to-point, and time-optimal given a path or a trajectory profile problems can be formulated in the optimal control framework. We showed, given conservative constraints on kinematic variables—*i.e.*, on the position, velocity, and the acceleration—a kinematic model corresponding to a multiple degree-of-freedom double integrator could suffice in many applications.

An analytic solution to the fixed-time optimal point-to-point trajectory planning problem with velocity and acceleration constraints was derived. The benefit of the

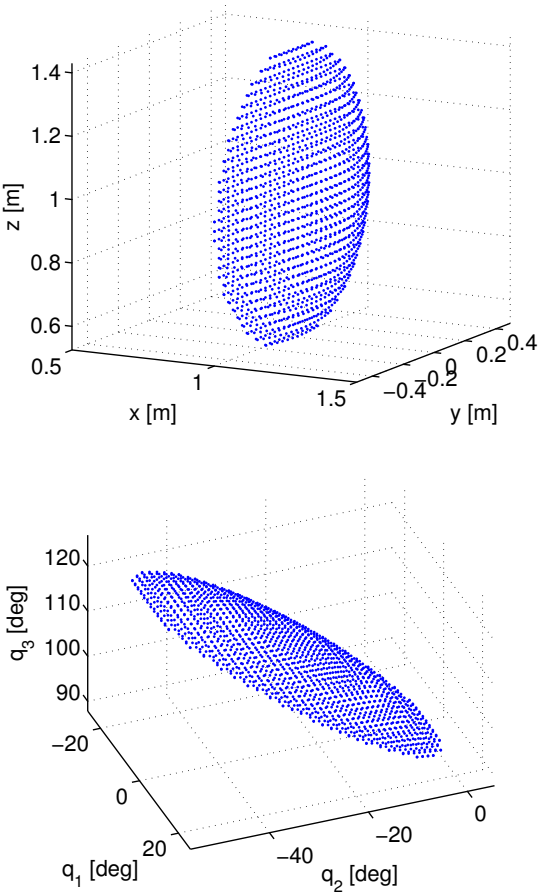

Figure 4.20 A thin spherical cap in the joint space and its corresponding image in the Cartesian space.

analytic solution is that its computation time is independent of the number of discretization points. Compared to time-optimal solutions, in the fixed-time problem the synchronization between degrees of freedom is not a problem since all motions must follow the same given fixed-time.

We showed that model predictive control can offer a framework for generating trajectories, which goes beyond tracking problems. In this framework, the path planning can be integrated as part of the trajectory generation, as long as a desired time is given. We highlighted similarities between trajectory planning, i.e., the problem of finding a proper control signal which transfers the system from the current states

to a desired final state at a given time, and the tracking problem.

A subset of MPC problems—*i.e.*, with quadratic cost functions and linear constraints—has the potential of being efficiently implemented to satisfy the requirements of modern robotic systems. This fact allowed us to make a real-time implementation of point-to-point trajectory planning for the task of ball-catching with the feature of successive refinement of the planned trajectory.

5

Conclusions and Future Research

In this thesis, two robotic setups were proposed for collecting human-generated trajectories and interaction forces in a natural way. In the first approach, the operator can directly interact with a robot in a so-called free-floating mode. The resulting passive lead-through programming is quite robust and does not suffer from common instability issues for interacting with stiff objects. In the robot-assisted teleoperation, a systematic way to define a 6-DOF teleoperation setup was described. We utilized an admittance controller on the slave side in order to enhance operator experience. Commanding a robot with admittance parameters instead of purely positions or forces has the benefit of allowing the robot to naturally follow the direction of external forces, hence assisting the operator. While in the single-arm lead-through programming, there is an unavoidable mechanical coupling between the operator, the manipulator, and the workpiece, using teleoperation decouples the operator–robot interface from the robot–workpiece interface. Thanks to this separation, it is possible to interact with the workpiece with a higher degree of transparency, which results in less unwanted demonstration side-effects compared to typical lead-through approaches.

A programming concept based on guarded motions was briefly discussed. We proposed enhancements to the existing models of the guarded motion by introducing a fault recovery mechanism and a more advanced constraint handling. With the help of the robotic setups for collecting human-generated trajectories and the designed infrastructure, an operator can interactively provide the required parameters of the guarded motions to build a task.

There are many applications in which only initial and final states matter for defining a task. To motivate the use of simplistic models for robots, we compared kinematic and dynamic models. Provided that kinematic constraints are conservative enough, our examples showed that kinematic models would be sufficient for trajectory planning. Having this in mind, we considered point-to-point trajectory planning and proposed a few solutions. We derived an analytic solution to the prob-

lem of fixed-time optimal trajectory planning with maximum velocity and maximum acceleration constraints using the maximum principle. In another approach, rather than finding an optimal trajectory, we developed an instantaneous trajectory generator in the form of an optimal controller using the Hamilton-Jacobi-Bellman equation. The controller updates the trajectory in a closed-loop fashion as a result of the changes in the state of the target and/or the state of the robot. Along the same line of thought, we proposed Model Predictive Control (MPC) for point-to-point trajectory planning. MPC enabled us to consider a wider range of constraints and models. The MPC approach was evaluated in a ball-catching experiment with real-time constraints.

The ultimate goal in trajectory generation can be viewed as developing methods to arrive at a low-level representation of motion (and its corresponding input to a system) from a high-level specification and available inputs. In this sense, a trajectory planner is no different than a controller which maps set-points and current measurements to a control signal.

As there are many ways to synthesize a controller, there exist many approaches to design a trajectory and a trajectory planner. In any case, it is of great importance how we specify tasks and how we measure performance. For instance, in guarded motion-based programming, we proposed a way to specify sensor-based piece-wise trajectories to accomplish a task. This is in essence similar to what is called procedural programming. On the other hand, we could have specified relationships between objects and tolerances in a declarative way to come up with such a procedure. Using the optimal control framework, i.e., defining a cost functional, constraints, and models was a step toward this goal.

This thesis can naturally be extended in different directions. Firstly, the models can be improved to better capture the dynamics of robots, e.g., by introducing the element of uncertainty in both models and measurements. Secondly, the constraints and the cost functional could be revised to be more representative of the tasks. Thirdly, the representation and reuse of human-generated trajectories can be investigated thoroughly. Trajectories augmented with sensory inputs, such as force/torque measurements, provide information-rich data for programming robots. A possible use of this information in the context of the optimal control is to warm start optimization algorithms from a demonstrated feasible solution.

Bibliography

ABB Robotics (2014). *ABB IRB140 Industrial Robot Data sheet*. Data sheet nr. PR10031 EN_R1.

ABB Robotics (2015). *YuMi product page*. `http://new.abb.com/products/robotics/yumi`. Accessed: 2015-01-15.

Ajoudani, A, N. Tsagarakis, and A Bicchi (2012). "Tele-impedance: towards transferring human impedance regulation skills to robots". In: *Proc. ICRA 2012 IEEE Int. J. on Rob. and Autom.* St Paul, MN, USA, pp. 382–388.

Åkesson, J., K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit (2010). "Modeling and optimization with Optimica and JModelica.org–languages and tools for solving large-scale dynamic optimization problems". *Computers and Chemical Engineering* **34**:11, pp. 1737–1749.

Aliaga, I., A. Rubio, and E. Sanchez (2004). "Experimental quantitative comparison of different control architectures for master-slave teleoperation". *IEEE Trans. Control Systems Technology* **12**:1, pp. 2–11.

Argall, B. D., S. Chernova, M. Veloso, and B. Browning (2009). "A survey of robot learning from demonstration". *Robotics and Autonomous Systems* **57**:5, pp. 469–483.

Årzén, K.-E., R. Olsson, and J. Åkesson (2002). "Grafchart for procedural operator support tasks". In: Camacho, E. et al. (Eds.). *Proc. 15th IFAC World Congress on Computers for Control*. Vol. L. Elsevier, pp. 85–90. ISBN: 0080442277.

Åström, K. J. and B. Wittenmark (2011). *Computer-controlled systems: theory and design*. Courier Dover Publications, Mineola, New York.

Bauml, B., T Wimbock, and G. Hirzinger (2010). "Kinematically optimal catching a flying ball with a hand-arm-system". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2010*. Taipei, Tawian, pp. 2592–2599.

Bellman, R. and R. E. Kalaba (1965). *Dynamic programming and modern control theory*. Academic Press, New York.

Bemporad, A. (2004). *Hybrid Toolbox - User's Guide*. `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`. Accessed: 2015-02-15.

Biegler, L. T., A. M. Cervantes, and A. Wächter (2002). "Advances in simultaneous strategies for dynamic process optimization". *Chemical Engineering Science* **57**:4, pp. 575–593.

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010). "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers". In: *Proc. Workshop "Innovative Robot Control Architectures for Demanding (Research) Applications—How to Modify and Enhance Commercial Controllers", IEEE Int. Conf. Rob. and Autom. (ICRA)*. Anchorage, AK, pp. 62–66.

Bobrow, J. E., S Dubowsky, and J. S. Gibson (1985). "Time-optimal control of robotic manipulators along specified paths". *Int. J. Robotics Research* **4**:3, pp. 3–17.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. 6th. Cambridge university press, Cambridge, UK.

Burdea, G. and F. Brooks (1996). *Force and touch feedback for virtual reality*. Wiley New York.

Caccavale, F., C. Natale, B. Siciliano, and L. Villani (1999). "Six-DOF impedance control based on angle/axis representations". *IEEE Trans. Rob. and Autom.* **15**:2, pp. 289–300.

Castain, R. H. and R. P. Paul (1984). "An on-line dynamic trajectory generator". *Int. J. Robotics Research* **3**:1, pp. 68–72.

Choset, H. M. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press, Cambridge, MA.

Chou, J. (1992). "Quaternion kinematic and dynamic differential equations". *IEEE Trans. Rob. and Autom.,* **8**:1, pp. 53–64. ISSN: 1042-296X.

Colgate, E. and N. Hogan (1989). "An analysis of contact instability in terms of passive physical equivalents". In: *Proc. IEEE Int. Conf. Robotics and Automation, 1989*, pp. 404–409.

Dahl, O. and L. Nielsen (1990). "Torque limited path following by on-line trajectory time scaling". *IEEE Trans. Rob. and Autom.* **6**, pp. 554–561.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007). "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty". *Int. J. Robotics Research* **26**:5, pp. 433–455.

Degallier, S., L. Righetti, S. Gay, and A. Ijspeert (2011). "Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives". *Auton. Robots* **31**:2-3, pp. 155–181. ISSN: 0929-5593.

Duleba, I. (1997). "Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution". *Robotica* **15**:05, pp. 555–562.

Flash, T. (1987). "The control of hand equilibrium trajectories in multi-joint arm movements". *Biological Cybernetics* **57** (4), pp. 257–274. ISSN: 0340-1200.

Force Dimension. *Omega-7 overview*. `http://www.forcedimension.com/products/omega-7/overview`. Accessed: 2015-02-15.

Garcia, C. E., D. M. Prett, and M. Morari (1989). "Model predictive control: theory and practice–a survey". *Automatica* **25**:3, pp. 335–348.

Ghazaei, M. and J. Hofele (2014). *D3.4: Report on fast tactile feeding strategies*. Confidential Deliverable. PRACE Project (EU/FP7/2007-2013 grant agreement no. 285380).

Glad, T. and L. Ljung (2011). *Reglerteori: flervariabla och olinjära metoder*. Studentlitteratur, Lund. ISBN: 978-91-44-03003-6.

Grant, M. and S. Boyd (2014). *CVX: Matlab software for disciplined convex programming, version 2.1*. `http://cvxr.com/cvx`. Accessed: 2015-02-15.

Grant, M. C. and S. P. Boyd (2008). "Graph implementations for nonsmooth convex programs". In: *Recent advances in learning and control*. Springer, pp. 95–110.

Hamilton, W. R. (1847). "On quaternions". In: *Proc. the Royal Irish Academy*. Vol. 3. 1847, pp. 1–16.

Hartl, R. F., S. P. Sethi, and R. G. Vickson (1995). "A survey of the maximum principles for optimal control problems with state constraints". *SIAM Review* **37**:2, pp. 181–218. ISSN: 00361445.

Haschke, R., E. Weitnauer, and H. Ritter (2008). "On-line planning of time-optimal, jerk-limited trajectories". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2008*. Nice, France, pp. 3248–3253.

Hogan, N. (1985). "Impedance control: an approach to manipulation: part III applications". *J. Dynamic Systems, Measurement, and Control* **107**:2, pp. 17–24.

Hokayem, P. F. and M. W. Spong (2006). "Bilateral teleoperation: an historical survey". *Automatica* **42**:12, pp. 2035–2057.

Hollerbach, J. M. (1984). "Dynamic scaling of manipulator trajectories". *J. Dynamic Systems, Measurement, and Control* **106**:1, pp. 102–106.

Howard, T. M., C. J. Green, and A. Kelly (2010). "Receding horizon model-predictive control for mobile robot navigation of intricate paths". In: *Field and Service Robotics*. Springer, pp. 69–78.

JModelica.org (2015). `http://www.jmodelica.org`. Accessed: 2015-02-15.

Kanjanawanishkul, K. and A. Zell (2009). "Path following for an omnidirectional mobile robot based on model predictive control". In: *Proc. IEEE Int. Conf. Rob. and Autom., 2009*. Kobe, Japan, pp. 3341–3346.

Khansari-Zadeh, S. and A. Billard (2011). "Learning stable nonlinear dynamical systems with Gaussian mixture models". *IEEE Trans. Robotics* **27**:5, pp. 943–957.

Klančar, G. and I. Škrjanc (2007). "Tracking-error model-based predictive control for mobile robots in real time". *Robotics and Autonomous Systems* **55**:6, pp. 460–469.

Kock, S., T. Vittor, B. Matthias, H. Jerregård, M. Källman, I. Lundberg, R. Mellander, and M. Hedelind (2011). "Robot concept for scalable, flexible assembly automation: a technology study on a harmless dual-armed robot". In: *IEEE Int. Symposium on Assembly and Manufacturing (ISAM), 2011*. Tampere, Finland, pp. 1–5.

Kristalny, M. and J. H. Cho (2012). "On the decentralized H_2 optimal control of bilateral teleoperation systems with time delays". In: *Proc. IEEE 51st Annual Conf. Decision and Control (CDC), 2012*. Maui, Huwaii, USA, pp. 6908–6914.

Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems*. Vol. 58. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, Germany.

Kröger, T. (2011a). "On-line trajectory generation: straight-line trajectories". *IEEE Trans. Robot.* **27**:5, pp. 1010–1016.

Kröger, T. (2011b). "Opening the door to new sensor-based robot applications — the Reflexxes motion libraries". In: *Proc. IEEE Int. Conf. Rob. and Autom. (ICRA), 2011*. Shanghai, China, pp. 1–4.

Kröger, T. and F. M. Wahl (2010). "On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events". *IEEE Trans. Robot.* **26**:1, pp. 94–111.

LabComm Protocol (2014). `http://wiki.cs.lth.se/moin/LabComm`. Accessed: 2015-02-15.

Lampariello, R., D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters (2011). "Trajectory planning for optimal robot catching in real-time". In: *Proc. IEEE Int. Conf. Rob. and Autom. (ICRA), 2011*. Shanghai, China, pp. 3719–3726.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge Univ. Press, Cambridge, UK. URL: `http://planning.cs.uiuc.edu`.

Lawrence, D. (1993). "Stability and transparency in bilateral teleoperation". *IEEE Trans. Rob. and Autom.* **9**:5, pp. 624–637.

Lee, S. H., I. H. Suh, S. Calinon, and R. Johansson (2012). "Learning basis skills by autonomous segmentation of humanoid motion trajectories". In: *Proc. 12th IEEE-RAS Int. Conf. Humanoid Robots, 2012*. Osaka, Japan, pp. 112–119.

Liberzon, D. (2011). *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, New Jersey. ISBN: 9780691151878.

Lin, C.-S., P.-R. Chang, and J. Luh (1983). "Formulation and optimization of cubic polynomial joint trajectories for industrial robots". *IEEE Trans. Automatic Control* **28**:12, pp. 1066–1074. ISSN: 0018-9286.

Linderoth, M. (2013). *On Robotic Work-Space Sensing and Control*. Thesis No. TFRT–1098–SE. PhD thesis. Department of Automatic Control, Lund University, Sweden.

Linderoth, M., A. Robertsson, K. Åström, and R. Johansson (2010). "Object tracking with measurements from single or multiple cameras". In: *Proc. IEEE Int. Conf. Rob. and Autom. (ICRA), 2010*. Anchorage, AK, pp. 4525–4530.

Liu, Y.-C. and N. Chopra (2012). "Controlled synchronization of heterogeneous robotic manipulators in the task space". *IEEE Trans. Robotics* **28**:1, pp. 268–275. ISSN: 1552-3098.

Macfarlane, S. and E. A. Croft (2003). "Jerk-bounded manipulator trajectory planning: design for real-time applications". *IEEE Trans. Rob. Autom.* **19**:1, pp. 42–52.

Maciejowski, J. M. (1999). *Predictive Control with Constraints*. Addison-Wesley, Boston, MA.

Mangasarian, O. L. (1966). "Sufficient conditions for the optimal control of nonlinear systems". *SIAM J. on Control* **4**:1, pp. 139–152.

Mattingley, J. and S. Boyd (2012). "CVXGEN: A Code Generator for Embedded Convex Optimization". *Optimization and Engineering* **13**:1, pp. 1–27.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. Scokaert (2000). "Constrained model predictive control: stability and optimality". *Automatica* **36**:6, pp. 789–814.

Nakamura, Y., H. Hanafusa, and T. Yoshikawa (1987). "Task-priority based redundancy control of robot manipulators". *Int. J. Robotics Research* **6**:2, pp. 3–15.

Norén, C. (2013). *Path Planning for Autonomous Heavy Duty Vehicles using Nonlinear Model Predictive Control*. Tech. rep. LiTH-ISY-EX–13/4707–SE. Dep. of E.E. Linköpings universitet.

Paul, R. (1979). "Manipulator Cartesian path control". *IEEE Trans. Systems, Man and Cybernetics* **9**:11, pp. 702–711. ISSN: 0018-9472.

Paul, R. (1972). *Modelling, trajectory calculation and servoing of a computer controlled arm*. Tech. rep. memo. AIM 177. Stanford Artificial Intelligence Laboratory. URL: `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0785071`.

Pontryagin, L. S., V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko (1962). *The Mathematical Theory of Optimal Processes*. Interscience, New York. ISBN: ISBN 2-88124-077-1.

Rawlings, J. and D. Mayne (2009). *Model Predictive Control: Theory and Design*. Nob Hill Publishing, LLC.

Ryu, J.-H. and C. Preusche (2007). "Stable bilateral control of teleoperators under time-varying communication delay: time domain passivity approach". In: *Proc. IEEE Int. Conf. Rob. and Automation*, pp. 3508–3513.

Seierstad, A. and K. Sydsæter (1987). *Optimal Control Theory with Economic Applications*. Elsevier Science B. B. Academic, Amsterdam, The Netherlands.

Shadmehr, R. (2005). *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT press, Cambridge, MA.

Shim, D. H. and S. Sastry (2007). "An evasive maneuvering algorithm for UAVs in see-and-avoid situations". In: *American Control Conf. ACC'07, 2007*. IEEE. New York, USA, pp. 3886–3891.

Shin, K. G. and N. D. McKay (1985). "Minimum-time control of robotic manipulators with geometric path constraints". *IEEE Trans. Autom. Control* **30**:6, pp. 531–541.

Siciliano, B. and L. Villani (1999). *Robot Force Control*. Kluwer Int. series in engineering and computer science. Kluwer Academic, Dordrecht, The Netherlands. ISBN: 9780792377337.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009). *Robotics: modelling, planning and control*. Springer Verlag, London.

Stenmark, M., J. Malec, and A. Stolt (2014). "From High-Level Task Descriptions to Executable Robot Code".

Stolt, A. (2012). *Robotic Assembly and Contact Force Control*. Licentiate Thesis ISRN LUTFD2/TFRT--3256--SE. Department of Automatic Control, Lund University, Sweden.

Stolt, A. and M. Linderoth (2011). *D3a: Delivery Report on basic assembly skill with force control demonstrated*. Confidential Deliverable. ROSETTA Project (EU/FP7/2007-2013 grant agreement no. 230902).

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2011). "Force controlled assembly of emergency stop button". In: *Proc. IEEE Int. Conf. Rob. and Autom. (ICRA), 2011*. IEEE. Shanghai, China, pp. 3751–3756.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2013). "Robotic assembly of emergency stop buttons". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2013*. IEEE. Tokyo, Japan, pp. 2081–2081.

Tavakoli, M., A. Aziminejad, R. Patel, and M. Moallem (2007). "Enhanced transparency in haptics-based master-slave systems". In: *American Control Conf., ACC '07, 2007*, pp. 1455–1460.

Taylor, R. H. (1979). "Planning and execution of straight line manipulator trajectories". *IBM J. Research and Development* **23**:4, pp. 424–436. ISSN: 0018-8646.

Theorin, A. (2013). *Adapting Grafchart for Industrial Automation*. Licentiate Thesis ISRN LUTFD2/TFRT--3260--SE. Department of Automatic Control, Lund University, Sweden.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2009). "Time-optimal path tracking for robots: a convex optimization approach". *IEEE Trans. Autom. Control* **54**:10, pp. 2318–2327.

Villani, L. and J. De Schutter (2008). "The handbook of robotics". In: Siciliano, B. et al. (Eds.). Springer, Berlin Heidelberg. Chap. 7, pp. 164–170.

Wang, H. (2013). "Task-space synchronization of networked robotic systems with uncertain kinematics and dynamics". *IEEE Trans. Automatic Control* **58**:12, pp. 3169–3174. ISSN: 0018-9286.

Yang, J., M. Dong, and Y. Tang (2012). "A real-time optimized trajectory planning for a fixed wing UAV". In: *Advances in Mechanical and Electronic Engineering*. Springer, Berlin Heidelberg, pp. 277–282.

Yokokohji, Y. and T. Yoshikawa (1994). "Bilateral control of master-slave manipulators for ideal kinesthetic coupling-formulation and experiment". *IEEE Trans. Rob. and Autom.* **10**:5, pp. 605–620.

Department of Automatic Control
Lund University
Box 118
SE-221 00 Lund Sweden

Document name
LICENTIATE THESIS

Date of issue
March 2015

Document Number
ISRN LUTFD2/TFRT--3265--SE

Author(s)
M. Mahdi Ghazaei Ardakani

Supervisor
Prof. Rolf Johansson, Prof. Anders Robertsson, Prof. Jacek Malec

Sponsoring organisation
EU FP7, PRACE (Ref. #285380), LCCC

Title and subtitle
Topics in Trajectory Generation for Robots

Abstract

A fundamental problem in robotics is generating the motion for a task. How to translate a task to motion or a series of movements is a non-trivial problem. The complexity of the task, the structure of the robot, and the desired performance determine the sequence of movements, the path, and the course of motion as a function of time, namely the *trajectory*. As we discuss in this thesis, a trajectory can be acquired from a human demonstration or generated by carefully designing an objective function. In the first approach, we examine a number of robotic setups which are suitable for human demonstration. More notably, admittance control as a new dimension to the robot-assisted teleoperation is investigated. We also describe a free-floating behavior which makes robust lead-through programming possible. As a way to utilize these setups, we present some ideas for developing a high-level language for an event-based programming common to assembly tasks.

Since immediate reaction to variations in the target state and/or robot state is desirable, we reformulate the trajectory generation problem as a controller design problem. Using the Hamilton-Jacobi-Bellman equation, we derive a closed-loop solution to the fixed-time trajectory-generation problem with a minimum-jerk cost functional. We show that the resulting trajectory coincides with a fifth-order polynomial function of time that instantaneously updates due to changes in the reference signal and/or the robot states.

A short comparison is made between kinematic and dynamic models for generating optimal trajectories. The conclusion is that given conservative kinematic constraints, both models behave in a similar way. Having this in mind, we derive an analytic solution to the problem of fixed-time trajectory generation with a quadratic cost function under velocity and acceleration constraints. The advantage of the analytic solution compared to an on-line optimization approach lies in the efficiency of the computation.

To extend the idea of closed-loop trajectory generation, we adapt the Model Predictive Control (MPC) framework. MPC is traditionally applied to tracking problems, i.e., when there is an explicit reference signal. Thus, it is a common practice to have a separate layer that generates the reference signal. We propose an integrated approach by introducing a final state constraint in the formulation. Additionally, we give the interpretation that the difference between tracking and point-to-point trajectory-planning problems is in the density of the specified desired reference signal. We utilize a strategy to reduce the discretization time successively. This way, we respect the real-time constraints for computation time while the accuracy of the solution is gradually improved as the deadline approaches. We have verified our proposed MPC approach to trajectory generation in a ball-catching experiment.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
120

Recipient's notes

Security classification