LUND UNIVERSITY

On Trajectory Generation for Robots

Ghazaei, Mahdi

2016

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

On Trajectory Generation for Robots

M. Mahdi Ghazaei Ardakani

LUND
UNIVERSITY

Department of Automatic Control

Ph.D. Thesis TFRT-1116
ISBN 978-91-7753-048-0 (print)
ISBN 978-91-7753-049-7 (web)
ISSN 0280–5316

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

Printed in Sweden by Media-Tryck.
Lund 2016

To my parents

Abstract

A fundamental problem in robotics is the generation of motion for a task. How to translate a task to a set of movements is a non-trivial problem. The complexity of the task, the capabilities of the robot, and the desired performance, affect all aspects of the trajectory; the sequence of movements, the path, and the course of motion as a function of time.

This thesis is about trajectory generation and advances the state of the art in several directions; special attention to trajectories in constrained situations when interaction forces are involved is paid; we bring a control perspective to trajectory generation and propose novel solutions for online trajectory generation with a quick response to sensor inputs; we formulate and find optimal trajectories for various problems, closing the gap between path planning and trajectory generation; the inverse problem of finding the control signal corresponding to a desired trajectory is investigated and we extend the applicability of an existing algorithm to a wider range.

Designing trajectories for tasks involving force interaction is difficult, since both the knowledge of the task and the dynamics of the robot are necessary. Alternatively, we can acquire human-generated trajectory. In this thesis, an immersive interface for task demonstration is proposed, where the operator can sense and act through the robot. This is achieved by coupling two robotic systems on a dynamical level. Limitations caused by the singular configurations or the reach of either of the robots are naturally reflected to the other as haptic feedback.

We consider a closed-loop approach to trajectory generation for fixed-time problems, where a desired target state (possibly time varying) is achieved by acting upon the feedback from the actual state of a robot. Using the Hamilton-Jacobi-Bellman equation, we derive an optimal controller for the fixed-time trajectory-generation problem with a minimum-jerk cost functional. The controller instantaneously updates the trajectory as a result of changes in the reference signal and/or the robot states. Moreover, a smooth transition between the finite-horizon and an infinite-horizon problem is de-

veloped. This enables switching smoothly to a tracking mode when a moving target is reached.

An analytic solution to the problem of fixed-time trajectory generation with a quadratic cost function under velocity and acceleration constraints is derived. This problem has a wide range of applications in motion planning. The advantage of the analytic solution compared to numerical optimization of the discretized problem is the unlimited resolution of the solution and the efficiency of the calculation, allowing sensor-based replanning and on-line trajectory generation.

To extend the idea of closed-loop trajectory generation, by accommodating a more generic form of system dynamics and constraints, we adopt the Model Predictive Control (MPC) framework. We give the interpretation that in the tracking problem the desired output at every sample is specified, while in point-to-point trajectory planning, it is limited to certain samples. This view unifies tracking and point-to-point trajectory generation problems, hence eliminating the need of a separate layer for reference generation. We discuss various choices of models, objective functions, and constraints for generating trajectories to transfer the state of the robot while respecting physical limitations on the motion as well as fulfilling computational real-time requirements. Experiments on an industrial robot in a ball-catching task show the effectiveness of the approach also in demanding scenarios.

A rigid-body model of a finger interacting with a trackball is considered. The proposed system and its extension with more fingers are suitable candidates for studying trajectory generation where interaction plays an important role, such as in assembly and manipulation tasks. The trajectory generation algorithm has to handle a number of important features such as unilateral and non-holonomic constraints. Additionally, in this problem task planning, path planning, and trajectory generation are strongly interrelated, which makes using an integrated approach to trajectory generation inevitable. We derive a hybrid, high-index differential-algebraic equation for modeling the system dynamics, which is used for both simulation and finding optimal trajectories.

Iterative Learning Control (ILC) fits into the picture of trajectory generation considering the fact that it finds a proper control signal for obtaining a desired trajectory. We focus on a specific regime of convergence of an ILC algorithm, which is traditionally ignored. We derive frequency-domain criteria for the convergence of linear iterative learning control (ILC) on finite-time intervals that are less restrictive than existing ones in the literature. Additionally, using an example we put forward an idea of how ILC can be adapted for point-to-point trajectory planning.

Acknowledgments

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 students Magnus Linderoth, Andreas Stolt, Björn Olofsson, and Olof Sörnmo 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 none of the results in this thesis were possible. I am equally thankful to the administrative staff, Eva Westin and Mika Nishimura, Ingrid Nilsson, and Monika Rasmusson for maintaining a smooth workflow in the department.

It was a great opportunity to share office with a former PhD student, Daria Majidian and former guest PhD student, Mariette Annergren. Specially, our diverse non-technical discussions were fun and I could also learn a great deal about Swedish culture and Sweden. The extended and extra coffee breaks with former PhD students and friends Meike Rönn and Alina Andersson were indeed the best recovery from the chores for doing a better research. Also, I wish to express my gratitude to the colleagues and friends who took their time and helped me with the proofreading of this thesis.

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. This is commonly considered as a sub-problem in motion planning, wherein the ultimate goal is to find fully automated procedures to generate a sequence of movements to perform a task [Nottensteiner et al., 2016; Thomas et al., 2015]. Typically, a task requires to bring about a certain state in the world. Hence, the aim of the motion is 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 approach is such 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, 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 consideration is whether to use a pure kinematic model or to model the complete linear or nonlinear dynamics of the system. Obviously, the latter is more complex and the computations can be more time consuming. In general, a dynamic model is required to guarantee satisfaction of the physical constraints such as limitations in actuators. 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 is desired to either minimize the time for executing a task or the energy consumed during the motion. Hence, motion-planning problems are often formulated as optimal-control problems. The cost functional can couple the different degrees-of-freedom (DOF) of the system and result in a coordination between them.

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 computing an optimal motion plan [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.

In the context of industrial manipulators, algorithms and methods to perform offline trajectory generation for time-optimal path tracking 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 in 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 time allocated to a job. Hence, the solution to fixed-time problems can prove valuable by naturally leading to coordination while reducing wear and tear of the robotic system. A sub-optimal solution to fixed-time trajectory planning for robot manipulators was proposed by [Dulęba, 1997]. In an-

other approach, the fixed-time optimal solution in the space of parametrized trajectories were found [Yang et al., 2012].

In this thesis, some new ideas for trajectory generation are explored. Initially, we investigate a robotic setup for acquiring trajectories from human demonstration as faithfully as possible. Fast algorithms for obtaining solutions to fixed-time point-to-point trajectory generation is a major theme in this work. The motivation is the necessity to regenerate or adjust trajectories in dynamic environments as soon as new sensor information becomes available. We propose a closed-loop approach as well as various optimal strategies to this problem. For tracking a trajectory, obtaining the required control signal is important. This leads us to iterative learning control and its convergence issues.

1.2 Contributions

The major contributions of this thesis are:

- A control law for dual-arm lead-through programming system;
- A closed-loop minimum-jerk trajectory generation algorithm;
- An analytic solution for point-to-point fixed-time trajectory planning under maximum velocity and maximum acceleration constraints;
- Adapting the Model Predictive Control framework to fixed-time point-to-point trajectory generation;
- Modeling of a ball-and-finger system for studying optimal trajectories under varying contacts and dynamics;
- A less conservative stability criterion for Iterative Learning Control.

1.3 Publications

Preliminary versions of parts of the research presented in this thesis have been published in the licentiate thesis by the author [Ghazaei Ardakani, 2015]. The current thesis is mainly based on the papers and submitted manuscripts as detailed below:

Ghazaei Ardakani, M. M. and K. Nilsson (2016). “Haptic interface for task demonstration”. Unpublished Manuscript.

In the above publication, the idea of using a second arm of a dual-arm robot as a haptic interface is attributed to both authors. The control design

and the results are entirely due to the first author. The second author has also contributed in structuring of the manuscript and improving the text. Chapter 2 is derived, in part, from this manuscript.

Ghazaei Ardakani, M. M., A. Robertsson, and R. Johansson (2015). "Online minimum-jerk trajectory generation". In: *Proc. IMA Conf. Mathematics of Robotics*. Oxford, UK.

The above publication is based on the results of the first author presented earlier in the licentiate thesis by the first author, with proof-reading and improvements suggested by the second and the third co-authors. Chapter 3 is based, in part, on this article.

Ghazaei Ardakani, M. M., M. Stemmann, A. Robertsson, and R. Johansson (2015). "An analytic solution to fixed-time point-to-point trajectory planning". In: *Proc. IEEE Conf. Control Applications (CCA)*. Sydney, Australia, pp. 306–311.

The derivation of the analytic solution to the fixed-time trajectory planning was done together with Meike Stemmann. The first author has completed the solution and provided numerical results. The third and the fourth co-authors provided constructive feedback on the manuscript. Chapter 5 is based, in part, on this article.

Ghazaei Ardakani, M. M., B. Olofsson, A. Robertsson, and R. Johansson (2015). "Real-time trajectory generation using model predictive control". In: *Proc. IEEE Int. Conf. Automation Science and Engineering (CASE)*. Gothenburg, Sweden, pp. 942–948.

Ghazaei Ardakani, M. M., B. Olofsson, A. Robertsson, and R. Johansson (2016). "Real-time trajectory generation using model predictive control". *IEEE Trans. Automation Science and Engineering (T-ASE)*. Submitted Manuscript.

The problem formulation in the above publications is due to the first author. The verification of the ideas and the implementation of algorithms were done together with the second author. The first two authors were responsible for extending the first paper into the second manuscript, while the ideas and some preliminary results had already been reported in the licentiate thesis by the first author. The manuscripts were prepared by the first and the second author, while proof-reading and improvements of the texts were suggested by the third and the fourth co-authors. Chapter 6 is based on these articles.

Ghazaei Ardakani, M. M. and F. Magnusson (2016). "Ball and finger system: Modeling and optimal trajectories". *Autonomous Robots*. In Submission.

The research idea of the publication above is due to the first author of the paper. The mathematical modeling and simulations were done by the first author, while an equal division of the work for the optimization part of the research between both authors is asserted. The contribution of the second author in discussing all aspects of the research is also acknowledged. The manuscript was collaboratively prepared. Chapter 7 is derived, in parts, on this manuscript.

Ghazaei Ardakani, M. M., S. Z. Kong, and B. Bernhardsson (2016). "On the convergence of iterative learning control". *Automatica*. Accepted for publication.

The non-monotone convergence phenomena in ILC were observed by the third author and proposed as a research problem. The analysis and further examples are due to the author of this thesis. The second and the third authors contributed by improving the rigorousness of the arguments and proofs. Chapter 8 is based, in part, on this article.

Other publications:

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

Capurso, M., M. M. Ghazaei Ardakani, R. Johansson, A. Robertsson, and P. Rocco (2017). "Sensorless kinesthetic teaching of robotic manipulators assisted by an observer-based force control". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Singapore. Submitted Manuscript.

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 (BioRob)*. São Paulo, Brazil, pp. 345–351.

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

Ghazaei Ardakani, M. M. and R. Johansson (2016). *Recovery of Uniform Samples and Spectrum of Band-limited Irregularly Sampled Signals*. Technical Report TFRT--7647--SE. Dept. Automatic Control, Lund University, Sweden.

Ghazaei Ardakani, M. 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)*. Phucket, Thailand, pp. 1273–1278.

Stolt, A., F. B. Carlson, M. M. Ghazaei Ardakani, I. Lundberg, A. Robertsson, and R. Johansson (2015). "Sensorless friction-compensated passive lead-through programming for industrial robots". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. Hamburg, Germany, pp. 3530–3537.

1.4 Thesis Outline

The thesis is organized as follows. In Chapter 2, we present methods to acquire human-generated trajectories as faithfully as possible. The proposed setup is especially useful when force interaction with the environment is required. Chapter 3 presents methods for extending a trajectory to a larger workspace and a closed-loop perspective on trajectory generation. A number of different optimization-based approaches to trajectory generation are presented in Chapter 4. We study the applicability of simple kinematic models compared to dynamic models. In Chapter 5, an analytic solution to fixed-time trajectory planning with state constraints is derived. We adapt Model Predictive Control (MPC) for point-to-point trajectory generation in Chapter 6. Chapter 7 deals with modeling and optimal control of a hybrid system. We propose a ball-and-finger system for studying optimal trajectories where interaction with the environment is required. Iterative Learning Control (ILC) provides a method to obtain the required control signal in order to achieve a desired trajectory. In Chapter 8, we look into the convergence of ILC. We also demonstrate how ILC can be applied to point-to-point trajectory planning, where the desired trajectory is not fully specified. Finally, conclusions are drawn in Chapter 9, where also ideas for future research are presented.

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 assist human workers with difficult and monotonous tasks. Numerous industrial environments have utilized robotic systems to increase the production, improve the efficiency of processes, and to improve the work environment for human workers by delegating the monotonous tasks to robots. A new generation of robots, such as the 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. The programming of robot systems for complex tasks involving force interaction is still a difficult problem. There are numerous suggested solutions in the literature and many of them rely on trajectory programming prior to operation.

Robotic motions can be generated based on teaching via some human–robot interface (HRI) or 3D simulation programming environments. The manipulation of the robotic systems is possible not only with the use of a teach pendant but also with direct physical 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.

Many tasks are still carried out manually, since it is overly difficult to program a robot to achieve a similar performance. Typically, these tasks involve interaction with an object or the environment, where the success of the task largely relies on the skills of the human. To program a robot, these skills need to be transferred to the robot. The most natural way for

Figure 2.1 A prototype of ABB YuMi Robot [ABB Robotics, 2015; Kock et al., 2011]. Early experiments for lead-through programming were carried out with this robot.

a human to do this is via demonstration. For teaching by demonstration, if the robot does not have a similar kinematic structure to the operator doing the demonstration, 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 [Billard et al., 2008]. Therefore, this approach has been widely used in human skill acquisitions, [Argall et al., 2009; Lee et al., 2015]. Despite this, the interface between humans and robots can be inconvenient and difficult for accurately transferring motions because of mechanical properties of robots such as inertia and friction. Although compliant motion control could be employed to reduce inertial/friction forces, direct teaching of industrial robots is still limited. Moreover, the mechanical coupling between the operator, the robot, and the workpiece makes it impossible to record faithfully the required force values for a task. Hence, an interface for demonstrating a task can contribute by allowing the operator to perceive the differences and limitations of the robotic system.

Using haptic feedback in teleoperation of robots [Hokayem and Spong, 2006] and in virtual reality [Constantinescu, 2008] has been the subject of research for many years. Teleoperation with haptic feedback provides a suitable framework for immersive demonstration of a task. Utilizing both visual and haptic feedback from a robot or a model of it, an operator can ideally feel and perceive a task from the robot's perspective, hence enabling accurate demonstration including force specification.

Four-channel haptic systems are the most general form of haptic devices

where position and interaction forces are measured from both sides [Aliaga et al., 2004]. With the improved techniques to estimate joint forces and TCP wrenches [Wahrburg et al., 2014; Stolt et al., 2015b; Linderoth et al., 2013], many of the existing robot manipulators can be used as haptic devices without additional cost. Moreover, as dual-arm robots become popular, the idea in this chapter is that one of the arms can be utilized as a convenient interface for demonstrating poses, trajectories, and forces.

Several factors limit the transparency of a teleoperation system. In the one-dimensional linearized case, it is shown that the transparency and stability are conflicting requirements [Lawrence, 1993]. Delays are detrimental to both transparency and stability and have been studied widely [Anderson and Spong, 1989; Lee and Spong, 2006; Hatanaka et al., 2015]. However, in teaching or virtual reality scenarios, delays can be avoided since a centralized implementation is possible. The transparency is further limited by the structures of the master or the slave devices. When two robotic systems are employed in a master-slave configuration, their workspace is limited to the points reachable by both systems simultaneously. This defines a common workspace for the robots. At the boundary of this common workspace, one or both of the systems are typically in a singular configuration with reduced manipulability.

Allowing the arms to have different configurations, substantially increases the flexibility in demonstration. For example, the demonstration of a task can be performed in a part of the workspace that is more convenient for the operator. Additionally, since we considered using robots that are not specifically designed as haptic devices, further theoretical developments to build a generic control approach toward an immersive user experience were demanded.

In this chapter, three approaches to dual arm lead-through demonstration are presented. Firstly in Sec. 2.2, the required constraint between two arms is formalized. Thereafter, in order to achieve the motion constraint we explain Cartesian-space and joint-space approaches as well as a more sophisticated approach using virtual constraints. For the Cartesian-space approach in Sec. 2.3, wrist-mounted force/torque sensors are ideally required. For the joint-space approach in Sec. 2.4, we make use of the estimated torques at each joint. Section 2.5 treats the complete dynamics of manipulators and the method can be used with or without force/torque sensors. The first two approaches have been implemented and tested on the YuMi prototype robot Frida[1], whereas the third one is investigated in simulations.

[1] The results are not included in the thesis.

Previous Research

Currently, several approaches to mapping between a slave and a master device exist. A point-to-point kinematic mapping to maximize the common workspace and to reduce the effect of the deficient subspace has been proposed in [Chen et al., 2007]. The problem of geometric correspondence has been solved by adding proper offsets and scaling [Rebelo and Schiele, 2012], wherein the placement of the slave has been optimized to maximize the manipulability and redundant degrees of freedom (DOF) were mapped using arm angles. In another approach, the mapping has been done using closed-form inverse kinematics and a measure of anthropomorphism based on the distance between the elbows [Liarokapis et al., 2012]. Redundancy has been utilized to avoid singularity. In [Salvietti et al., 2013] the kinematics have been abstracted away by considering the effect of interacting with a virtual object. They used forward mapping for the geometry and backward mapping concerning the forces (impedance type haptic device). To map a small workspace to a large one, scaling and drifting methods have been used [Chotiprayanakul and Liu, 2009].

2.2 Motion Constraint

Assume that the system has in total $n = n_1 + n_2$ degrees of freedom, where n_1 is the DOF of the first arm and n_2 the second arm. Let us denote the generalized coordinates for both arms by $q \in \mathbb{R}^n$ and split it into the coordinates related to the first and the second arm according to $q^T := (q_1^T, q_2^T)$. The geometrical constraints between the two end-effectors can be expressed as

$$p_2 - p_1 = \Delta p, \tag{2.1a}$$

$$R_1^T R_2 = \Delta R. \tag{2.1b}$$

Here, the variables with subscripts 1 and 2 concern the first and the second arm, respectively. The position of the end-effectors are denoted by $p \in \mathbb{R}^3$, and $R \in SO(3)$ denotes the orientation, Δp and ΔR are offsets in the position and the orientation, respectively.

Multiplying (2.1b) by R_1 from the left results in $R_2 = R_1 \Delta R$. Now, by differentiating both sides w.r.t. time, we find

$$S(\omega_2) R_2 = S(\omega_1) R_1 \Delta R + 0 = S(\omega_1) R_2. \tag{2.2}$$

Hence,

$$S(\omega_2 - \omega_1) R_2 = 0. \tag{2.3}$$

Here $S(\omega)$ is the skew-symmetric matrix corresponding to the vector product by the angular velocity ω. Therefore, fixed relative positions and orientations imply the following kinematical constraints

$$\begin{aligned} v_2 - v_1 &= 0, \\ \omega_2 - \omega_1 &= 0, \end{aligned} \tag{2.4}$$

where $v_i = \mathrm{d}p_i/\,\mathrm{d}t$. By expressing these equations in the generalized coordinates, we find the differential kinematics relationships

$$\begin{aligned} J_{2P}(q_2)\dot{q}_2 - J_{1P}(q_1)\dot{q}_1 &= 0, \\ J_{2O}(q_2)\dot{q}_2 - J_{1O}(q_1)\dot{q}_1 &= 0. \end{aligned} \tag{2.5}$$

Here $J_{iP}(.)$ and $J_{iO}(.)$ are the translational and rotational geometrical Jacobians w.r.t. the end-effectors.

Hence, by differentiating the geometrical constraints with respect to time, we have found a kinematical constraint such that

$$G\dot{q} + g_0 = 0, \tag{2.6}$$

where $G \in \mathbb{R}^{6\times n}$ is

$$G = \begin{bmatrix} -J_{1P}(q_1) & J_{2P}(q_2) \\ -J_{1O}(q_1) & J_{2O}(q_2) \end{bmatrix} = [-J_1, J_2], \tag{2.7}$$

and $g_0 = 0 \in \mathbb{R}^{n\times 1}$. Here we have defined J_1 and J_2 by stacking together the translational and rotational Jacobians.

2.3 Cartesian-Space Approach

In this approach, the end-effectors of both arms represent the same entity, although there might exist an arbitrary, yet fixed, amount of offset in their positions and orientations. In other words, if the initial offset is removed, we can assume that the manipulators interact with the same object at the same point simultaneously. Accordingly, the forces F and torques τ on this virtual object is the sum of the interaction forces from each arm

$$\begin{aligned} F &= F_1 + F_2, \\ \tau &= \tau_1 + \tau_2, \end{aligned} \tag{2.8}$$

where indices refer to the arms.

Assuming the virtual object is solid, the motion of the object is governed by

$$\begin{aligned} F &= m\dot{v} - Dv, \\ \tau &= I\dot{\omega} - D_o\omega. \end{aligned} \tag{2.9}$$

Here m is the mass, D is the viscous friction, I is the inertia tensor, D_o is the rotational damping, v is velocity, and ω is angular velocity.

For each arm, the resulting joint motion in the joint-space is obtained by

$$\dot{q}_i = J_i^{\dagger}(q_i)\begin{pmatrix} v \\ \omega \end{pmatrix}, \tag{2.10}$$

$$q_i = \int_0^t \dot{q}_i \, dt + c_i, \tag{2.11}$$

where c_i is the initial joint value, $J^{\dagger}(q_i)$ denotes the pseudo-inverse of the Jacobian at given joint angles. The joint angles and velocities are fed to the internal robot controller.

2.4 Joint-Space Approach

In the Cartesian-space approach, no attention to internal dynamics has been paid. For example, for redundant robots, it may not be possible to achieve a desired configuration for an arm just by interacting with the end-effector. Thus, in this section we look into a joint-space approach for fulfilling the motion constraint. Additionally, in this approach, estimation of the torques in each joint can be directly used when there is no wrist-mounted force/torque sensor.

Denoting the estimated torque by $\hat{\tau}$, an admittance control law for the robot is described as below

$$M\ddot{q}^{\mathrm{a}} = \hat{\tau} - D\dot{q}^{\mathrm{a}}. \tag{2.12}$$

Here q^{a} is the reference joint angle due to the admittance law, M is the desired mass matrix and D corresponds to the desired damping.

The motion of the arms in Cartesian space must be equal. According to Sec. 2.2, it means that

$$J_1\dot{q}_1 = J_2\dot{q}_2. \tag{2.13}$$

We can assume the motion of each arm consists of a contribution from the local admittance regulator $\dot{q}_i^{\mathrm{a}}$ and the motion induced by the other arm $\dot{q}_{j\to i}$. Therefore, we can write,

$$\begin{aligned} \dot{q}_1 &= \dot{q}_{2\to 1} + \dot{q}_1^{\mathrm{a}} \\ \dot{q}_2 &= \dot{q}_{1\to 2} + \dot{q}_2^{\mathrm{a}} \end{aligned} \tag{2.14}$$

Substituting (2.14) into (2.13), we find

$$J_1\dot{q}_{2\to 1} + J_1\dot{q}_1^{\mathrm{a}} = J_2\dot{q}_{1\to 2} + J_2\dot{q}_2^{\mathrm{a}} \tag{2.15}$$

A choice which makes this equality true is

$$\begin{aligned} \dot{q}_{1\to 2} &= J_2^{\dagger} J_1 \dot{q}_1^{\mathrm{a}} \\ \dot{q}_{2\to 1} &= J_1^{\dagger} J_2 \dot{q}_2^{\mathrm{a}}. \end{aligned} \tag{2.16}$$

Another possible choice is to find $\dot{q}_{1\to 2}$ and $\dot{q}_{2\to 1}$ that minimize $\|\dot{q}_{1\to 2}\| + \|\dot{q}_{2\to 1}\|$ for some norm.

Since these relations are on the velocity level, a drift in the relative position and orientation might occur. Assuming one robot is the master and the other one is the slave, this can be corrected by closing the loop for the position and orientation of the slave.

Another possibility is to mirror the motion of the arms. This implies $v_{1,y} = -v_{2,y}$, $\omega_{1,y} = -\omega_{2,y}$, $\omega_{1,z} = -\omega_{2,z}$ where indices refer to arms and the components of translational velocities, v, and angular velocities, ω. Applying these changes to (2.13), we obtain

$$J_1 \dot{q}_1 = S J_2 \dot{q}_2, \tag{2.17}$$

where the matrix S is defined as

$$S := \mathrm{diag}(1, -1, 1, 1, -1, -1). \tag{2.18}$$

A possible solution for the induced velocities in this case is

$$\begin{aligned} \dot{q}_{1\to 2} &= J_2^{\dagger} S J_1 \dot{q}_1^{\mathrm{a}} \\ \dot{q}_{2\to 1} &= J_1^{\dagger} S J_2 \dot{q}_2^{\mathrm{a}}. \end{aligned} \tag{2.19}$$

2.5 Dynamic Coupling

In many approaches, no attempt has been made to couple dissimilar robotic manipulators with arbitrary configurations in a master-slave setup. Instead, the placement or configurations of master and slave manipulators are optimized in order to mitigate the problems due to mismatch between the workspace of the robots. In virtual reality applications, the direct approach to tackle issues related to dissimilar kinematics of the master and slave devices has been to employ virtual spring-damper elements to penalize user motion along the direction resisted by the virtual mechanism [Constantinescu et al., 2006; Constantinescu, 2008]. However, this approach cannot guarantee perfect tracking and is not easily extendable to arbitrary structures.

The methods discussed in the previous sections couple the manipulators on the kinematic level. These methods are also prone to issues with singularity since the effect of the coupling on the overall motion has been

ignored. In this section, we consider a generic approach to mapping between a master and a slave device, which allows integration of the multibody dynamics of the manipulators into the control design. The key idea is to derive forces and torques that are required to enforce desired virtual kinematic constraints. Controller synthesis via virtual constraints offers a great tool for control of various mechanical structures; see, e.g., [Freidovich et al., 2008; Shiriaev et al., 2007; Westervelt et al., 2007].

The approach in this section is well-suited for kinesthetic teaching. More specifically, it relies on coupling between two serial arms on a dynamical level. This approach results in a mapping between two, not necessarily similar, arms, yet allowing a theoretically perfect position tracking during free motion and perfect tracking of forces in hard contact tasks of the end-effectors, when there is no kienematic singularity. Especially, it allows to use any 6-DOF or redundant robotic arm, mounted on any surface, and with any initial position in a dual-arm haptic setup. Either of the arms can pass through singular configurations while the haptic feedback is being adjusted, complying with restricted directions. Additionally, if there is a possibility to estimate/measure external forces at the joint level, any point of the mechanisms can behave as a haptic interface.

Tool Analogy

Transparency with respect to the environment is at odds with the desire to provide feedback concerning the limitations of a teleoperation system to the operator. The operator often expects haptic feedback associated with the presence of the robotic system as long as it is not disrupting the task. The need for reflecting the physical limitations of the teleoperation system to the operator becomes of major importance in several cases, e.g., when arbitrary robots are employed as haptic devices, the kinematics of master and slave robots are dissimilar, or similar master and slave devices are employed in different configurations.

In this section we suggest a tool analogy, in order to find a principled way to map between a master and a slave device. This means that the teleoperation system behaves as a physical tool utilized by the operator to interact with the environment. Figure 2.2 illustrates this idea. The motion of the tool is uniquely determined by knowing the physical properties of the tool, interaction forces by the environment F_e and the force exerted by the human F_h. In this model, the transparency of the teleoperation system is determined by the extent that we can reduce the mass and the friction of the tool. There is also no distinction between the functionality of the master and slave devices, e.g., the arms of a dual-arm robot, as both sides can be regarded as either master or slave. However, for the sake of naming we refer to the side arranged to interact with a workpiece as slave and the side interacting with the operator as master.

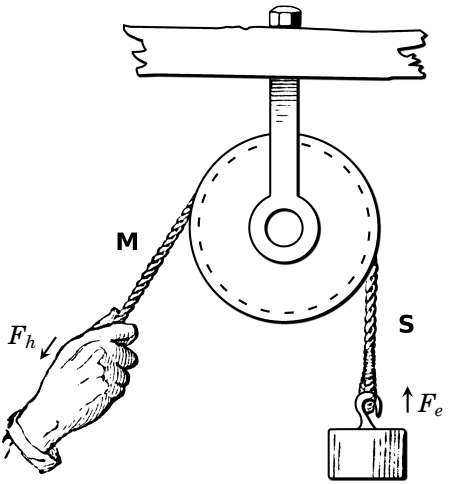


Figure 2.2 Tool Analogy: the pulley is used as a tool for interacting with the mass. The geometrical and physical properties of the string and pulley affect how the user perceives the environment. The force exerted by the human F_h is affected by the mass of the rope and the friction in the pulley, hence in general differs from the force interacting with the environment F_e. "M" and "S" denote the master and slave sides, respectively. The illustration is adapted from [Privat Deschanel, 1884].

Control Principle

Let us consider the nonlinear system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u(x), \\ y &= h(x), \end{aligned} \tag{2.20}$$

where x, u, y denote the state vector, control signals, and outputs, respectively. We would like to find $u(x)$ that results in y identically equal to zero and to find the zero dynamics of the system [Isidori, 1995]. We define y as the deviation of the relative translation and orientation between a frame on the slave arm and the master arm from a desired offset. Without loss of generality, we choose these frames to be located at the end-effectors of the arms. By zeroing the output, i.e., imposing the virtual constraint, we make sure that the end-effectors maintain the fixed offset. We use the motion constraint introduced in Sec. 2.2 as a virtual constraint.

Dynamics

To derive the dynamics of the system, we start off from the kinetic energy

$$\mathcal{T} = \sum_i \frac{1}{2} m_i \dot{q}^T (J_P^i)^T J_P^i \dot{q} + \sum_i \frac{1}{2} \dot{q}^T (J_O^j)^T R_{\ell_i} I_i R_{\ell_i}^T J_O^i \dot{q}, \tag{2.21}$$

where m_i and I_i denote the mass and inertia matrix of link i, respectively, J_P^i and J_O^i denote partial Jacobians up to link i and R_{ℓ_i} is the rotation matrix from the world coordinate to link i. Since we aim to compensate the gravitational force, we do not include the potential energy in the Lagrangian $\mathcal{L}$. Hence, we have

$$\mathcal{L} = \mathcal{T}. \tag{2.22}$$

Assuming only viscous friction with coefficient μ_v, the equations of motion for the constrained system according to the Lagrange–d'Alembert theorem [Bloch, 2003] can be derived as

$$\begin{aligned} M(q)\ddot{q} + C(q,\dot{q})\dot{q} &= Q^e + Q^{kc} - \mu_v \dot{q}, \\ G\dot{q} + g_0 &= 0, \end{aligned} \tag{2.23}$$

where $M(q)$ is the mass matrix, $C(q,\dot{q})\dot{q}$ denotes the total effect of centripetal and Coriolis forces. The generalized external forces and the generalized forces due to the kinematical constraints are denoted by Q^e and Q^{kc}, respectively and fulfill the relations

$$Q^e = \tau + J^T h^e, \tag{2.24}$$

$$Q^{kc} = G^T \lambda, \tag{2.25}$$

where τ is the vector of external torques applied at the joints, h^e is the vector of forces and torques exerted on the end-effector, and $\lambda(t,q,\dot{q}) \in \mathbb{R}^6$ are Lagrange multipliers.

By introducing subscripts 1 and 2 for the parameters and variables, we find

$$M(q) := \text{blkdiag}\left(B_1(q_1), B_2(q_2)\right), \tag{2.26}$$

$$J(q) := \text{blkdiag}(J_1(q_1), J_2(q_2)), \tag{2.27}$$

$$C(q,\dot{q}) := \text{blkdiag}\left(C_1(q_1,\dot{q}_1), C_2(q_2,\dot{q}_2)\right), \tag{2.28}$$

$$h^e := \begin{bmatrix} h_1^e \\ h_2^e \end{bmatrix}, \quad Q^{kc} := \begin{bmatrix} Q_1^{kc} \\ Q_2^{kc} \end{bmatrix}, \tag{2.29}$$

where blkdiag$(\cdot)$ denotes the block diagonal concatenation operator and the equation of motion for each individual arm is

$$B_i(q_i)\ddot{q}_i + C_i(q_i,\dot{q}_i)\dot{q}_i = \tau_i + J_i^T(q_i) h_i^e + Q_i^{kc} - \mu_v \dot{q}_i. \tag{2.30}$$

Therefore, the control law for each arm to maintain the constraint is equal to the constraint force and is derived as

$$\begin{aligned} Q_1^{kc} &= -J_1^T(q_1)\lambda, \\ Q_2^{kc} &= J_2^T(q_2)\lambda. \end{aligned} \tag{2.31}$$

By introducing $x^T = \left(q^T, \dot{q}^T\right)$ and substituting λ with u, we can write (2.23) as follows:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} q \\ M^{-1}(q)\left(-C(q,\dot{q})\dot{q} - \mu_v\dot{q} + Q^e + G^T(q)u\right) \end{bmatrix} =: f(x) + g(x)u, \\ \dot{y} &= G(q)\dot{q} =: \frac{\partial h}{\partial x}\dot{x}, \end{aligned} \tag{2.32}$$

where $M^{-1} = \text{blkdiag}\left(B_1^{-1}(q_1), B_2^{-1}(q_2)\right)$. The system 2.32 is in the standard form (2.20), except that the output equation has been replaced with its derivative. Equations (2.23)–(2.25) define a system of index-2 Differential Algebraic Equations (DAEs), which can be solved numerically. From a control-design perspective, the solution is the zero dynamics of system (2.32) with relative degree two.

Let us define $\Gamma := GM^{-1}G^T$. The Lagrange multipliers, λ, can be calculated by considering the general results given in [Isidori, 1995]

$$\begin{aligned} \lambda = u^*(x) &= -\left(L_g L_f h(x)\right)^{-1} L_f^2 h(x) \\ &= \Gamma^{-1}\left(GM^{-1}\left(C\dot{q} - \tau - J^T h^e + \mu_v\dot{q}\right) - \sum_{k=1}^{n}\left(\frac{\partial G}{\partial q^k}\dot{q}^k\right)\dot{q}\right), \end{aligned} \tag{2.33}$$

where $L_f h(x)$ denotes the Lie derivative of $h(x)$ with respect to f and the superscript in q^k denotes the kth DOF. Substituting λ given in (2.33) back into (2.32) results in $\ddot{y} = 0$ and the zero dynamics are given by

$$M(q)\ddot{q} + R\left(C(q,\dot{q})\dot{q} - \tau - J^T h^e + \mu_v\dot{q}\right) + G^T\Gamma^{-1}\sum_{k=1}^{2n}\left(\frac{\partial G}{\partial q^k}\dot{q}^k\right)\dot{q} = 0, \tag{2.34}$$

where $R = I_{n\times n} - P$, and $P = G^T\Gamma^{-1}GM^{-1}$.

By substituting (2.33) into (2.31), we observe how the nonlinear feedback from both arms contributes to the control law. Assuming an accurate model of the arm and after gravity compensation of the arms, these control laws are applicable. However, in order to achieve a different behavior for a robot than the stipulated one by its mechanical properties, we can solve (2.34) numerically and set the reference of each joint controller to the solution obtained in a cascaded control architecture.

Note that it is possible to calculate the motion, even when Γ is ill-conditioned, which is a consequence of G being rank deficient. We take the derivative of the constraint conditions (2.6) to obtain

$$G\ddot{q} + \sum_{k=1}^{n} \left(\frac{\partial G}{\partial q^k}\dot{q}^k\right)\dot{q} = 0. \tag{2.35}$$

The relation (2.35) defines an underdetermined system of equations. In general, it has either no solution or infinitely many solutions. By using a pseudo-inverse when G loses rank, we can find a solution of the form

$$\ddot{q} = Ks + \ddot{q}_0, \tag{2.36}$$

where K is an $n \times (n - \text{rank}(G))$ matrix expanding the nullspace of G, i.e., $GK = 0$, s is a vector of $n - \text{rank}(G)$ unknowns, and $\ddot{q}_0$ is any regularized least-squares solution to (2.35). Substituting this expression for $\ddot{q}$ into (2.23), and premultiplying both sides of the first equation by K^T to eliminate λ, produces

$$K^T M K s = K^T(\tau + J^T h^e - C(q,\dot{q})\dot{q} - \mu_v \dot{q} - M\ddot{q}_0). \tag{2.37}$$

In (2.37), $K^T M K$ is full rank. Thus, we can solve for s. Substituting s back into (2.36) gives the value of $\ddot{q}$.

By substituting the constraint forces (2.31) into the equation of the motion of the corresponding arm (2.30), it becomes clear that λ plays the same role as the external wrenches h_i^e. Moreover, we conclude that λ can be interpreted as the cut forces as if the two arms were attached at the end-effectors. This allows an alternative implementation of these virtual constraints using the Newton-Euler formulation, where dynamics of the resulting closed chain is derived by considering interaction forces between links.

If the arms are standing still, i.e., all the time derivatives are equal to zero, and either of the arms is impacted by an external wrench opposite to the other one ($h_2^e = -h_1^e$), it is easy to verify that $\lambda = h_1^e$ is a solution to (2.23). This guarantees a perfect transfer of forces at the end-effector to the other arm in the steady state when there is a hard contact and no singular configuration. When J_i loses rank, the equilibrium can be reached not only by equal and opposite forces but also by

$$\begin{aligned} h_2^e &\in \{-h_e + n \mid n \in \mathcal{N}(J_2^T)\}, \\ h_1^e &\in \{h_e + n \mid n \in \mathcal{N}(J_1^T)\}, \end{aligned} \tag{2.38}$$

where $h_e \in \mathbb{R}^6$ is an arbitrary wrench and $\mathcal{N}(\cdot)$ denotes the null space.

2.6 Extensions

In this section, we consider three typical scenarios encountered in the applications, which can be handled by the method developed in Sec. 2.5.

Similar Arms

As a special case, assume that the arms are similar and that the only difference is the mounting plane and the joint values. Let $\mathfrak{q} \in \mathbb{R}^{n/2}$ represent the generalized coordinates of an arm and R_0 denote the rotation of the mounting plane of arm 2 with respect to arm 1. By expressing Jacobians and moments of inertia in the rotated frame, we find

$$\begin{aligned} B_2(\mathfrak{q}) &= \sum_{i=1}^{n_2} \left(m_i J_{2P}^{(i)T} J_{2P}^{(i)} + J_{2O}^{(i)T} R_{\ell_i} I_i R_{\ell_i}^T J_{2O}^{(i)} \right) \\ &= \sum_{i=1}^{n_2} \left(m_i J_{1P}^{(i)T} R_0^T R_0 J_{1P}^{(i)} + J_{1O}^{(i)T} R_0^T \left(R_0 R_{\ell_i} I_i R_{\ell_i}^T R_0^T \right) R_0 J_{1O}^{(i)} \right) \\ &= \sum_{i=1}^{n_1} \left(m_i J_{1P}^{(i)T} J_{1P} + J_{1O}^{(i)T} R_{\ell_i} I_i R_{\ell_i}^T J_{1O}^{(i)} \right), \end{aligned} \tag{2.39}$$

This proves that $\tilde{B}(\mathfrak{q}) := B_1(\mathfrak{q}) = B_2(\mathfrak{q})$, hence $\tilde{C}(\mathfrak{q}, \dot{\mathfrak{q}}) := C_1(\mathfrak{q}, \dot{\mathfrak{q}}) = C_2(\mathfrak{q}, \dot{\mathfrak{q}})$. Since no conservative forces (e.g., gravity) is included, we arrive at the conclusion that the dynamics of the arms are identical in $\mathfrak{q}$ coordinates.

Redundant Robots

To extend the approach to redundant robots, additional virtual constraints may be added. This allows impacting all the DOF from either side, not only those required for maintaining the offset between the end-effectors. In this subsection, we show specifically how constraints in the joint space as well as on relative distances can be introduced.

Assume H_i is a matrix, where each row has exactly one non-zero element corresponding to a redundant joint. Consequently, we write a constraint in the joint space as

$$H_2 q_2 - H_1 q_1 = \Delta q, \tag{2.40}$$

where Δq is a constant vector. By taking the derivative of (2.40) w.r.t. time, we conclude

$$H_2 \dot{q}_2 - H_1 \dot{q}_1 = 0. \tag{2.41}$$

Now, augmenting this constraint to (2.7) results in

$$G = \begin{bmatrix} -J_1 & J_2 \\ -H_1 & H_2 \end{bmatrix}, \tag{2.42}$$

and the rest of the calculations remain the same as before.

In the case of human-like robots, another approach can be to maintain the distance between the elbows. Assume $\Delta r := r_2 - r_1$ where r_i is the position vector of the elbow of the ith arm. Therefore, the constraint can be expressed using the 2-norm as $\|\Delta r\| = d \in \mathbb{R}$. By differentiation with respect to time, we find

$$\|\Delta r\|^2 = d^2 \Rightarrow \Delta r \cdot \Delta v = 0, \tag{2.43}$$

where Δv is the relative velocity of the elbows. Rewriting (2.43) in terms of generalized coordinates, results in

$$\Delta r \cdot \Delta v = \Delta r^T \left(J_{2P}^{(e)} \dot{q}_2 - J_{1P}^{(e)} \dot{q}_1\right) = \Delta r^T \begin{bmatrix} -J_{1P}^{(e)} & J_{2P}^{(e)} \end{bmatrix} \dot{q} = 0, \tag{2.44}$$

where $J_{iP}^{(e)}$ denotes the translational Jacobian w.r.t. the ith elbow. Therefore, the matrix G should be augmented according to

$$G = \begin{bmatrix} -J_1 & J_2 \\ -\Delta r^T J_{1P}^{(e)} & \Delta r^T J_{2P}^{(e)}, \end{bmatrix}. \tag{2.45}$$

Joint Limits

Joint limits can be handled by adding repulsive forces close to the limits. A physically motivated model for this purpose is the Hunt and Crossley model [Hunt and Crossley, 1975]

$$L(q, \dot{q}) = K\delta^{\alpha}(q)\left(1 + \frac{3}{2}c\dot{\delta}(q, \dot{q})\right), \tag{2.46}$$

where δ is the amount of compression, K is the force stiffness, the exponent α depends on the local geometry around the contact area, and c denotes the force damping weight. However, such models may lead to very stiff problems. Consequently, if the joint is pushed toward the direction of the joint limit despite the repulsive force, numerical issues might arise. Therefore, as soon as a joint reaches zero velocity by the repulsive forces, if $\delta > 0$, we can add a zero velocity constraint to the joint. This constraint is relaxed when the constraint force is in the direction toward the joint limit.

For each active constraint, a row with one in the column where the joint has reached its limit and zeros elsewhere is augmented to G, i.e.,

$$G_{c_i,j} = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise,} \end{cases} \tag{2.47}$$

where i is the joint that has hit its limit and c_i denotes the row number associated with this constraint.

Using reinitialization of the states, it is possible to bring the reference velocities of the joints to zero immediately. Therefore, repulsive forces are not required at all, if the states are reinitialized as soon as a joint limit is reached.

2.7 Simulations and Results

Modern simulation environments are able to perform automatic differentiation [Griewank and Walther, 2008] and to solve differential algebraic equations. Specifically, the solver of Dymola [Dymola, 2016] is able to derive an index-1 DAE by the dummy variable method [Mattsson and Söderlind, 1993; Mattsson et al., 2000], such that the original geometrical constraints are retained and the required derivatives are calculated automatically. Although the closed-loop system using (2.31) without a stabilizing term is prone to numerical drift in the geometrical constraints, the index-1 DAE obtained by the dummy variable method does not suffer from accumulating errors.

To implement the algorithms, we have developed a software in Maple to calculate the equations for an arbitrary serial manipulator specified by DH parameters [Denavit and Hartenberg, 1955], mass, inertia tensor and center of gravity of each link. The resulting equations were manually imported to Dymola for simulation. We have also used the Newton-Euler formulation based on the Modelical Multibody Library [Otter et al., 2003], which allows us to build a robot from its components. The advantage of this method is that no explicit calculation of the Jacobians for the end-effectors is required.

In the following, we report the simulation results of two types of manipulators; two 6-DOF Robots and ABB YuMi [ABB Robotics, 2015].

6-DOF Robot

For the simulated example, we have used the same DH parameters specified in Table 2.1 for both arms. The center of mass of each link was considered to be in its middle. The inertia tensors were assumed diagonal $I_{xx} = I_{yy} = 0.005$, $I_{zz} = 0.001\,\mathrm{kg\,m^2}$. The masses were equal to 1 kg. The viscous friction coefficient was $\mu_v = 5\,\mathrm{N\,m\,s\,rad^{-1}}$.

The end-effector of arm 1 was pushed in the x direction with 10 N at $t = 1\,\mathrm{s}$. After one second, arm 2 was pushed in the opposite direction with the same amount of force. At the same time, arm 1 was pushed downwards and again after a second, arm 2 was pushed in the opposite direction with the same amount of force. Figure 2.3 shows the external forces at the end-effectors. The resulting movements of the end-effectors and joint angles are shown in Figs. 2.4 and 2.5, respectively. Note that since after a while the forces on arm 1 and arm 2 are opposite each other, the motion is stopped.

Table 2.1 DH parameters for an anthropomorphic arm with a spherical wrist.

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	θ_1
2	0.6	0	0	θ_2
3	0	$\pi/2$	0	θ_3
4	0	$-\pi/2$	0.5	θ_4
5	0	$\pi/2$	0	θ_5
6	0	0	0.5	θ_6

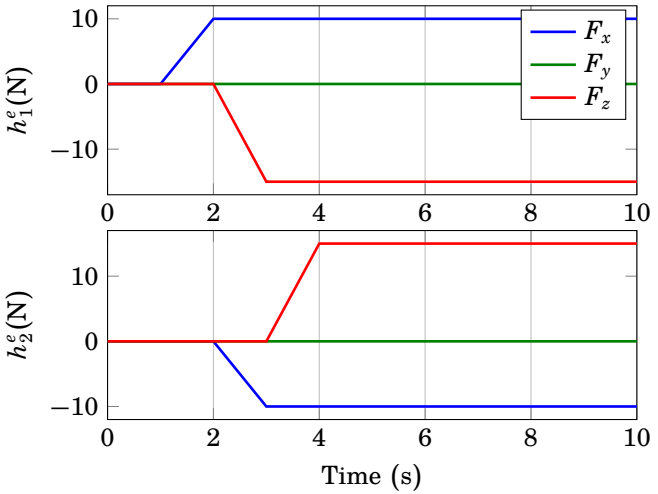


Figure 2.3 External forces applied at the end-effectors of the arms.

Figure 2.6 illustrates the constraint forces at the joints required to maintain the relative position. Moreover, we see in Fig. 2.7 that the relative position and orientation are constant.

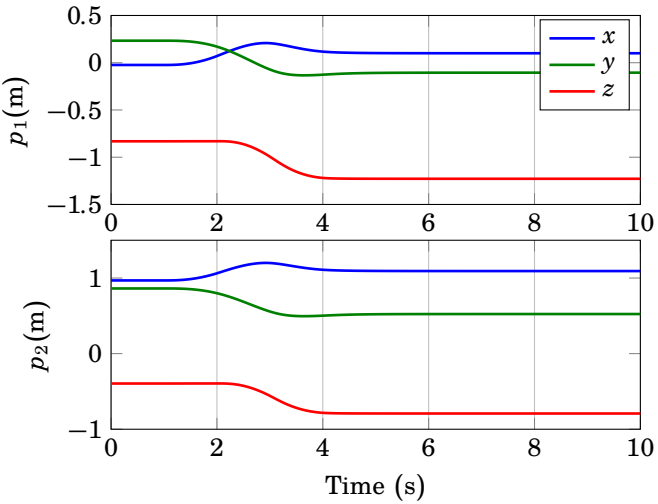


Figure 2.4 Positions of the end-effectors. The movement stops quickly as the forces at the end-effectors become opposite to each other.

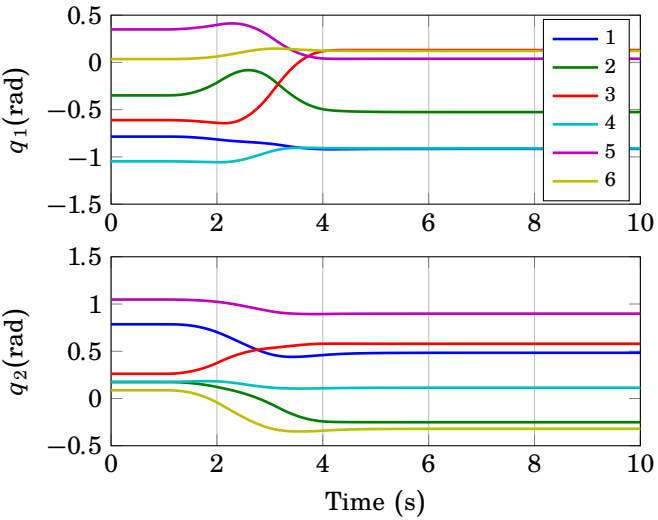


Figure 2.5 Joint angles of each arm as a result of the external forces shown in Fig. 2.3.

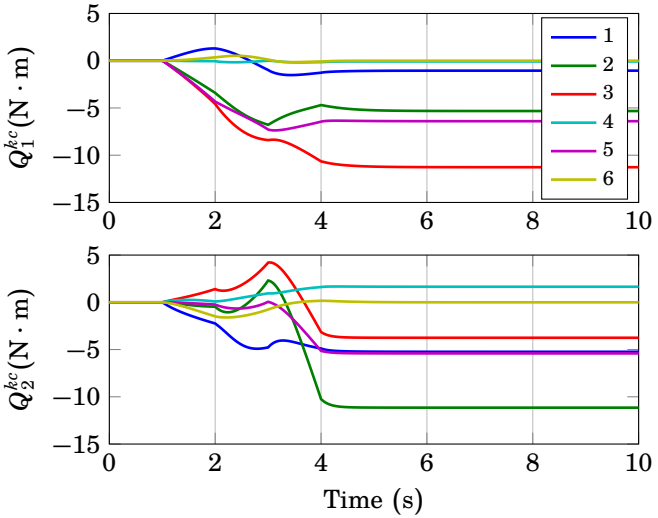


Figure 2.6 Constraint torques required to maintain the fixed offset between the end-effectors.

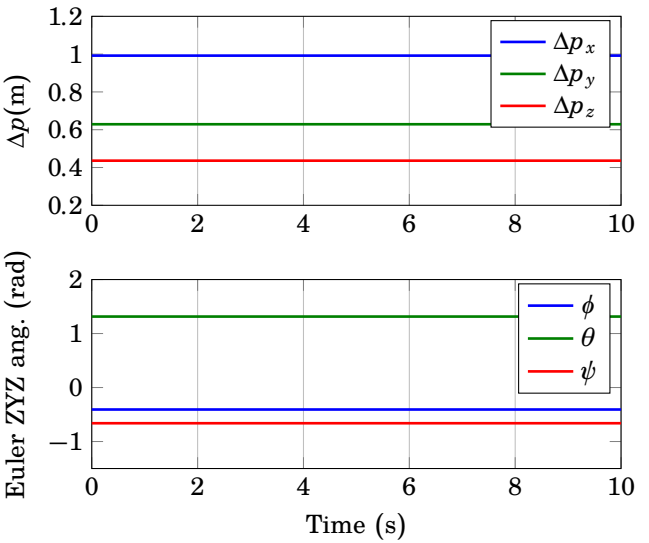


Figure 2.7 Position and orientation offsets are kept constant.

ABB YuMi

A rigid body model of the mechanical structure of the YuMi was built based on publicly available CAD files of the robot. Information about joint limits was extracted from the datasheet of the robot [ABB Robotics, 2015]. Additionally, viscous friction with $\mu_v = 0.1\,\mathrm{N\,m\,s\,rad^{-1}}$, representing the desired damping, was included in all the joints.

In the first simulation experiment, we considered using the right arm to interact with a spring in parallel to a damper attached to the left arm. The spring and damping constants were $c = 50\,\mathrm{N\,m^{-1}}$ and $d = 30\,\mathrm{N\,s\,m^{-1}}$, respectively. A constraint was also included to keep the initial distance between the elbows constant. The spring was initially compressed and as a result it exerted some force on the right arm. After being decompressed, an external force was applied to the left arm in the x-direction. After 6 s, the force was increased to 20 N. Hence, the spring was further stretched in the same direction. Figure 2.8 illustrates the robot in the initial, right before increasing the force, and the final configurations. The paths obtained by the end-effectors are shown in blue. In Fig. 2.9, the offsets in the position and the distance between the elbows are displayed. As it can be seen, the position tracking is perfect within the accuracy of the solver which was 10^{-6}. The same holds for the orientation of the end-effectors, which had the relative rotation of 180° around the local x-axis.

Figure 2.10 shows the force tracking of the end-effectors. After approximately 2 s and 5 s, respectively, when the system approached an equilibrium, we observe good force tracking. However, by increasing the force on the master side the robot reaches a singular configuration, where higher forces than

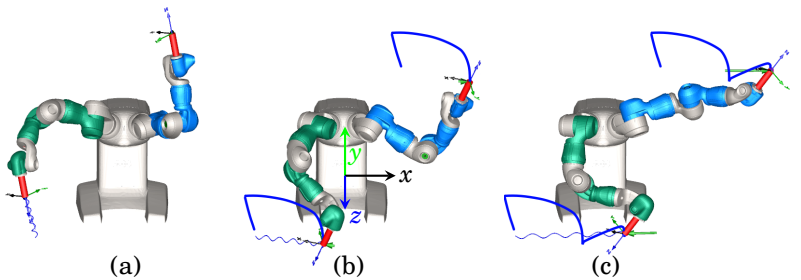


Figure 2.8 The graphical representation of the ABB YuMi robot in the first simulation experiment; (a) shows the initial condition, (b) right before increasing the force, and (c) the final configuration. The paths in blue correspond to the synchronized movement of the end-effectors where the end-effectors have the same position and orientation disregarding the fixed initial offset. The spring exemplifies the interaction with the environment. The green arrows illustrate the amount of forces at the interaction points.

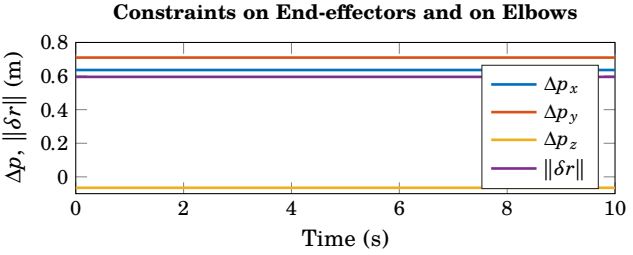


Figure 2.9 The position offset between end-effectors and the distance between the elbows are constant with the resolution of 10^{-8}.

the forces applied to the slave side can be resisted. This is the reason for poor force tracking toward the end of the trajectory. However, at the same time this implies that the operator is able to perceive a haptic feedback because of the limitations of the mechanism.

In the second simulation experiment, free motion was considered and force interaction with a point different from the typical end-effectors. In contrast to the previous simulation, no constraint between the elbows was included. From 0.5 s on, a constant force of 3 N, always perpendicular to the elbow of the left arm was applied to the elbow. This made the elbow rotate clockwise until almost 3 s when joint 3 reached its lower limit and afterwards the direction of rotation reversed. The joint angles are shown in Fig. 2.11. We see that the joint constraint was released after almost a second as soon as the required constraint torque (given in the same direction as the axis of the joint) became negative.

2.8 Discussion

The proposed model for task demonstration solves the problem of capturing the interaction forces in addition to solving the problem of correspondence between a human and a robot simultaneously. Thanks to the separation of the operator–robot interface from the robot–workpiece interface, the dual-arm lead-through programming (LTP) interface is more operator-friendly and less prone to unwanted demonstration side-effects compared to single-arm setups [Ghazaei Ardakani, 2015; Stolt et al., 2015a]. Since the initial orientation and positions of the arms are free to be chosen, the operator has large flexibility for demonstration of a task from a convenient location with respect to the workpiece.

A major difficulty in using dissimilar robots for teleoperation is singularity avoidance. When a robot reaches a singular configuration, its controllability becomes limited. Consequently, one of or both the arms fail to follow a

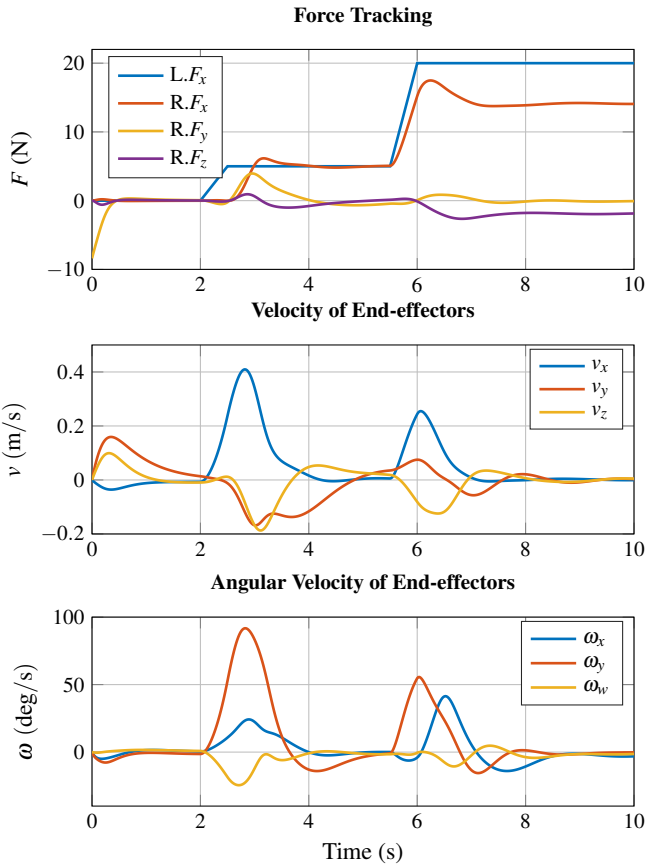


Figure 2.10 Comparison of the forces at the slave (right R) and the master (left L) arm of YuMi. The forces at the end-effectors of the master and slave robots track each other. When one of or both the arms become singular, additional haptic forces can be perceived by the operator.

desired motion in Cartesian space. Using approaches such as iTaSC [Borghesan et al., 2012a; Borghesan et al., 2012b] for defining purely kinematic constraints might lead to infeasible Cartesian motion due to the limitations of one or both robots. A standard approach to deal with the calculation of velocities at singular configurations is to use a damped pseudo-inverse. However, that will eventually result in losing the constant relative orientation and position. In unfortunate scenarios, this can even lead to getting stuck in a singular configuration. This is the case for the approaches in Secs. 2.3 and 2.4.

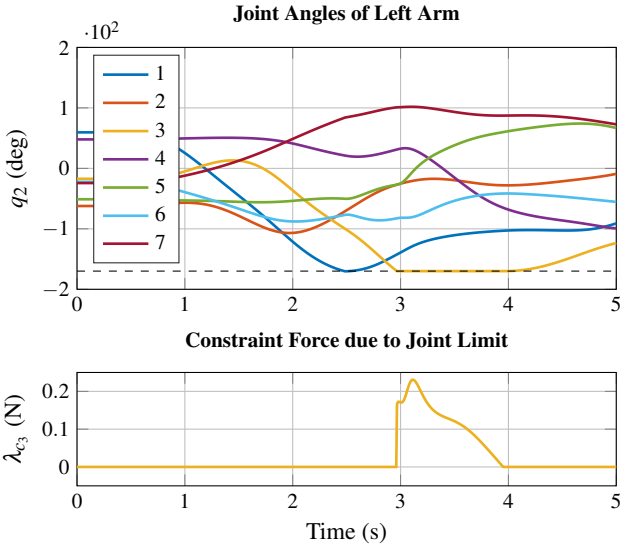


Figure 2.11 Above the joint angles of the left arm as the left elbow was pushed with a force of 3 N perpendicular to it. The dashed line represents the lower bound on the angle of the 3rd joint. Below, we see the constraint torque due to the joint limit of the 3rd joint.

While the detection of a singularity and the axes affected are straightforward, e.g., based on manipulability ellipsoids [Spong et al., 2006], these methods cannot be used to distinguish between the direction towards the singularity or away from it. Thus, it might be required to determine all singular configurations in advance in order to be able to provide a meaningful haptic feedback to the operator. On the other hand, using the virtual constraint approach in Sec. 2.5, kinematic constraints in singular configurations influence both arms via a dynamic coupling. As long as G in (2.7) does not lose rank, Γ in (2.33) is non-singular since the mass matrix B is positive definite, and a solution can be obtained. Even when G loses rank, it is still possible to find a unique solution to the motion according to (2.37) and (2.36), although it may lead to a temporary relaxation of the constraint.

By including the dynamics according to Sec. 2.5, the motion will comply with the kinematic constraints as long as the constraint forces remain bounded. This way the kinematic constraints are naturally integrated into the calculation of the motion. Moreover, in addition to detecting those states where constraints cannot be satisfied, we can quantify the feasibility of constraints based on the amount of force required to maintain them. It is important to note that any dynamical properties within the limitations of

the actuators may be assigned to the robots. Thus, the values of the actual physical properties are not required.

At the geometrical level, several choices for constraining the orientation of the end-effectors are possible, including the three outer diagonal elements of the rotation matrix of the residues, Euler angles, and three out of four elements of a unit quaternion corresponding to (2.1b). It is required to investigate which geometrical constraint formulation for rotation offers a numerically more stable solution. On the other hand, if no such constraint is used, there is no need for solving explicitly the inverse kinematics problem, which is typically a difficult nonlinear problem. This comes certainly at the cost of possible drifts.

Note that the control law (2.31) results in $\ddot{y} = 0$, i.e., the 2nd derivative of the constraint equals to zero. A control approach to ensure no drifts is to add a stabilizing term. This can for instance be accomplished by summing up a control signal to the slave side using Cartesian-space position control to converge exponentially to the position and orientation of the master plus the desired offset.

The aim of this chapter was to formulate a proper model for dual-arm LTP interface rather than suggesting a new approach for solving constrained dynamics. Therefore, to solve the resulting DAE the general approaches for constrained [Laulusa and Bauchau, 2008; Bauchau and Laulusa, 2008; Cuadrado et al., 1997; Arabyan and Wu, 1998] and over-constrained systems, e.g., by introducing elasticity [Zahariev and Cuadrado, 2011] and elimination of redundant constraints [Müller, 2014] may be applied.

If we consider two 6-DOF serial robots and G being rank deficient, the resulting mechanism is most likely unmovable provided that the configurations of the robots are not identical. The reachability of such states has to be investigated. However, assuming those states are reachable, reinitialization of the velocities will not be sufficient and the constraint must be relaxed temporarily to resume the motion as suggested earlier.

For dual-arm humanoid robots, the redundant degrees of freedom between master and slave arms can be mapped by coupling the swivel angles of the arms [Tolani et al., 2000]. Constraints of such type can be directly implemented into a DAE solver. Also, including a small gravity load at the end-effectors can be useful, since it encourages the operator to grasp at least one of the end-effectors when the system is activated, similarly to holding a tool.

The same technique as described in Sec. 2.5 can be used to derive task related forces such as those required to limit the motion to a certain plane or along a certain line. Coupling between other points than the actual end-effectors, locking certain joint angles or Cartesian directions, and scaling can also be introduced to create a variety of motion.

The first two approaches, described in Secs. 2.3 and 2.4, were imple-

mented and tested in the lab on the Frida robot. As a future work, we consider the implementation of the approach introduced in Sec. 2.5 on an ABB YuMi robot. This requires stepping forward the DAE solver in each sampling time based on the current state of the robot and the new estimates of external forces and consequently updating the reference trajectories to the robot.

2.9 Conclusion

In this chapter, the problem of a suitable interface for task demonstration involving force interaction has been addressed. The dual-arm model was proposed for instructing robots in an immersive way in order to collect human-generated trajectories. This model can solve the problem of capturing the interaction forces in addition to the problem of mapping between a human demonstration and a robot simultaneously. Thanks to separation of the operator–robot interface from the robot–workpiece interface, dual-arm demonstration is more operator-friendly and offers less unwanted demonstration side-effects compared to single-arm lead-through. Since the initial orientation and positions of the arms are free to be chosen, the operator has large flexibility for demonstration of a task from a convenient location with respect to the workpiece.

We presented Cartesian-space and joint-space strategies as well as an approach based on virtual constraints. A tool analogy for designing a haptic interface was proposed, where restrictions in the mechanism are naturally reflected to the operator as haptic feedback. Specifically, we derived a control law coupling the motion of two robotic arms based on introducing virtual constraints between the multi-body models of the arms. Using this formalism, robots with different kinematics can be employed while they can pass through singular configurations. Additionally, any point on the the master or slave arms behaves as a haptic interface, given that force measurement/estimation at joints is available. Our approach leads to an immersive haptic interface, which allows the operator to generate trajectories reproducible by the robot in an intuitive way.

The developed approaches in this chapter aimed for collecting both position and force data. The data can be used for example for providing the parameters of a guarded motion. 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 various 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 at a proper pick-up point 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. As another example, consider a quadcopter task which is to follow/catch flying objects as soon as one crosses a border. A camera system detects the object and provides estimates of its current position, velocity, and acceleration as well as an estimate of its arrival time at the border. We choose a minimum-jerk trajectory profile to generate smooth motion. These applications fall under the fixed-time trajectory planning problems. Moreover, the trajectory has to be re-planned online, i.e., as soon as new estimates are obtained.

The main motivation for this chapter is to provide answers to the following questions: how to update trajectories immediately as a result of changes in the (moving) target while ensuring the continuity of the position, velocity and acceleration of the trajectory and how to ensure that there is a smooth transition between trajectory planning and tracking modes.

A generic way to deal with online trajectory generation is to buffer each segment of the trajectory (or its parameters) and implement switching between the pieces. However, this approach becomes inefficient if the update rate of the trajectory is high. The questions lead us to the reformulation of

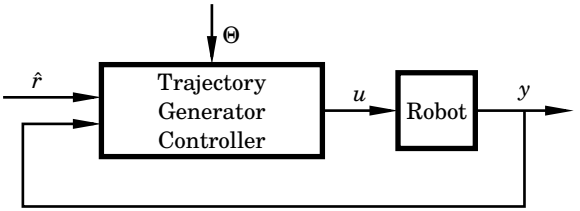


Figure 3.1 Block diagram of trajectory generation.

the trajectory generation as a dynamical system with a trajectory-generation controller. This allows for a fully reactive trajectory-generation method with continuous reactions to the changes in the target. In contrast to a mathematically designed or an optimal trajectory as solely a function of time, we regard a trajectory as an output of a dynamical system. The exogenous input signal defines the set-point for the trajectory generator.

For time-optimal problems, there exist fast open-loop control algorithms [Kröger, 2011b], which might even be used for closing the loop in the trajectory generation. In this chapter, we propose a complete closed-loop 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 time coordination between several degrees of freedom or entities.

Assume that the motion of a robot is described by

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

where y represents the position of the robot, x denotes the states of the system, and t is the time. Given the structure depicted in Fig. 3.1, we wish to generate a control signal

$$u = k(\hat{r}, x, t, \Theta), \tag{3.2}$$

which results in the desired motion. In the control law, Θ denotes a set of parameters and $\hat{r}$, the exogenous input signal, corresponds 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

$$\hat{r}(t) = r(t + d). \tag{3.3}$$

In general, the motion patterns can be acquired via human demonstration [Khansari-Zadeh and Billard, 2011] or designed by a mathematical

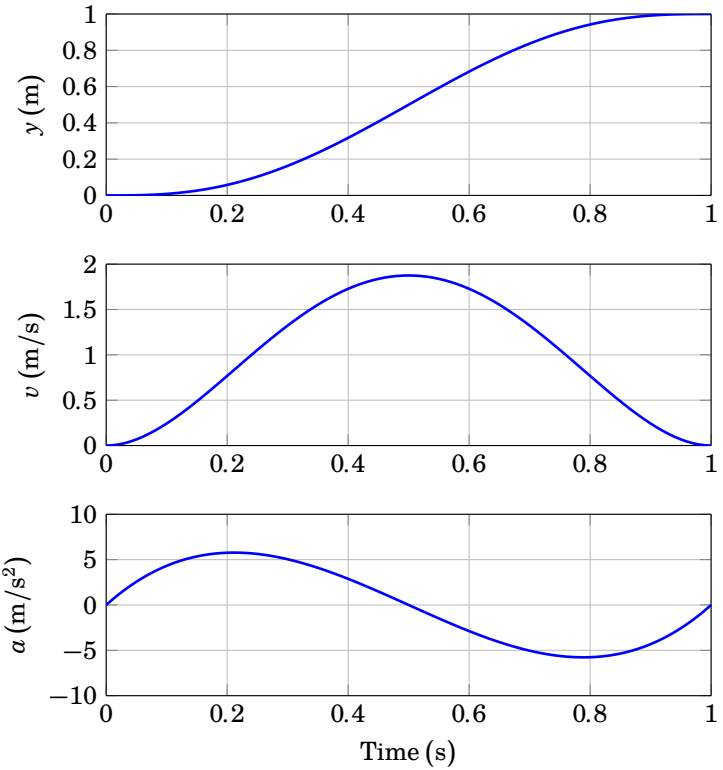


Figure 3.2 Motion template (3.4): The position, velocity, and acceleration vs. time are illustrated. The initial time, t_0, and the initial position, y_0, are set to zero.

formula [Paul, 1979; Taylor, 1979]. We call these patterns motion templates. For trajectory generation, polynomials play an important role [Paul, 1972; Paul, 1979; Lin et al., 1983; Taylor, 1979]. They show up for example as partial solutions to the minimum-time problems or they are used as basis functions. 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

$$y = y_0 + (y_0 - 1)(-10(t - t_0)^3 + 15(t - t_0)^4 - 6(t - t_0)^5), \qquad (3.4)$$

where y_0 and t_0 denote the initial position and initial time, respectively. Figure 3.2 illustrates the motion template.

The main focus in this chapter is to present a closed-loop solution to

trajectory generation. For this purpose, the template has to be generalized in order to give a complete description of the desired movements in all possible situations. Simple operations such as time and coordinate scaling are introduced. We use these techniques later to extend our template to the entire workspace while respecting certain constraints. As a generalization of the template (3.4), we consider a 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 after reaching a certain 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 Sec. 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 three operations, time scaling, coordinate scaling, and a coordinate transformation.

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.

The scaling operations allow us to generalize the motion trajectories generated by a dynamical system to different distance and timing requirements. We will take advantage of this in Sec. 3.7 to provide a new parametrization for the trajectory generation. The coordinate transformation makes it possible to use the current value of the reference signal instead of its predicted value.

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

$$\tilde{t} := \alpha(t) \Rightarrow t = \alpha^{-1}(\tilde{t}). \tag{3.5}$$

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

$$\begin{aligned}
\tilde{y}(\tilde{t}) &= y(t) \\
\tilde{y}'(\tilde{t}) &= \frac{\dot{y}(t)}{\dot{\alpha}(t)} \\
\tilde{y}''(\tilde{t}) &= \frac{\ddot{y}(t)}{\dot{\alpha}^2(t)} - \frac{\ddot{\alpha}(t)}{\dot{\alpha}^3(t)}\dot{y}(t) \\
\tilde{y}'''(\tilde{t}) &= \frac{\dddot{y}(t)}{\dot{\alpha}^3(t)} - 3\frac{\ddot{\alpha}(t)}{\dot{\alpha}^4(t)}\ddot{y}(t) + \left(3\frac{\ddot{\alpha}^2(t)}{\dot{\alpha}^5(t)} - \frac{\dddot{\alpha}(t)}{\dot{\alpha}^4(t)}\right)\dot{y}(t)
\end{aligned} \tag{3.6}$$

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

As an example, we can choose $\alpha(t) = \alpha t$ to be a linear function, where α is constant. Then, $\tilde{t} = \alpha t$ and

$$\begin{aligned}
\tilde{y}(\tilde{t}) &= y\left(\frac{\tilde{t}}{\alpha}\right) \\
\tilde{y}'(\tilde{t}) &= \alpha^{-1}\dot{y}\left(\frac{\tilde{t}}{\alpha}\right) \\
\tilde{y}''(\tilde{t}) &= \alpha^{-2}\ddot{y}\left(\frac{\tilde{t}}{\alpha}\right) \\
\tilde{y}'''(\tilde{t}) &= \alpha^{-3}\dddot{y}\left(\frac{\tilde{t}}{\alpha}\right),
\end{aligned} \tag{3.7}$$

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

$$\begin{aligned}
\tilde{y} &= \beta(y) \\
\dot{\tilde{y}} &= \beta'(y)\dot{y} \\
\ddot{\tilde{y}} &= \beta''(y)\dot{y}^2 + \beta'(y)\ddot{y} \\
\dddot{\tilde{y}} &= \beta'''(y)\dot{y}^3 + 3\beta''(y)\ddot{y}\dot{y} + \beta'(y)\dddot{y}.
\end{aligned} \tag{3.8}$$

Similar to the time scaling, we can choose $\beta(y) = \beta y$ to be a linear function, where β is constant. Then,

$$\begin{aligned}
\tilde{y}(t) &= \beta y(t) \\
\dot{\tilde{y}}(t) &= \beta\dot{y}(t) \\
\ddot{\tilde{y}}(t) &= \beta\ddot{y}(t) \\
\dddot{\tilde{y}}(t) &= \beta\dddot{y}(t).
\end{aligned} \tag{3.9}$$

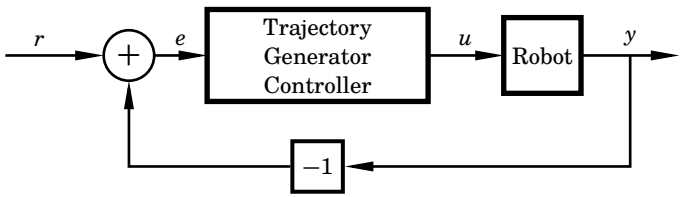


Figure 3.3 Modified block diagram of trajectory generation.

By combining the coordinate scaling and time scaling operations, it is possible to parametrize the classes of position-, velocity-, and acceleration-preserving transformations. Considering (3.9) and (3.7), which correspond 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 one-dimensional 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, 2000], it is possible to use the transformation $e(t) = r(t) - y(t)$ where $r(t)$ denotes the target position. Now, if e evolves according to the dynamics of the autonomous system, y will eventually track r. This can alternatively be interpreted as a movement composed of a tracking term $r(t)$ plus a non-linear shaping function $e(t)$. To clarify this point, rewrite the transformation as

$$y(t) = r(t) - e(t), \tag{3.10}$$

The shaping function satisfies

$$\begin{aligned} e(t_0) &= r(t_0) - y(t_0) \\ e(t_f) &= 0. \end{aligned} \tag{3.11}$$

Note that (3.10) 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. 3.3. 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. 3.3 compared to Fig. 3.1 is that the generated trajectory depends only on $r(t) - y(t)$.

As an example, assume that the robot is supposed to reach position y_f, velocity v_f, and acceleration a_f from y_0, v_0, and a_0 within 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 previous discussion, 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

$$\begin{aligned} r_1(t) &= y_f + v_f(t-d) + \frac{1}{2}a_f(t-d)^2 \\ r_2(t) &= v_f + a_f(t-d) \\ r_3(t) &= a_f. \end{aligned} \tag{3.12}$$

On the other hand, the motion of the robot can be derived from (3.10) to be

$$\begin{aligned} y(t) &= y_f + v_f(t-d) + \frac{1}{2}a_f(t-d)^2 - e_1(t) \\ v(t) &= v_f + a_f(t-d) - e_2(t) \\ a(t) &= a_f - e_3(t). \end{aligned} \tag{3.13}$$

By setting $t = 0$ and $t = d$ in (3.13), we obtain the following relationships

$$\begin{aligned} e_1(0) &= y_f - v_f d + \frac{1}{2}a_f d^2 - y_0 = y_0^{\text{target}} - y_0^{\text{robot}} \\ e_2(0) &= v_f - a_f d - v_0 = v_0^{\text{target}} - v_0^{\text{robot}} \\ e_3(0) &= a_f - a_0 = a_0^{\text{target}} - a_0^{\text{robot}} \\ e_1(d) &= 0 \\ e_2(d) &= 0 \\ e_3(d) &= 0. \end{aligned} \tag{3.14}$$

3.3 Minimum-Jerk Model

In [Flash and Hogan, 1985], a minimum-jerk model has been proposed, which provides a kinematic description of the voluntary motion of the human hand in planar scenarios. The model is able to predict the bell-shaped velocity profiles of the hand in point-to-point movements as well as the characteristics of the curvature in via-point movements. According to this model, the cost functional to be minimized is:

$$C = \frac{1}{2}\int_0^{d_0} \left(\dddot{X}^2 + \dddot{Y}^2\right) dt, \tag{3.15}$$

where X and Y represent the coordinates and d_0 the duration of the movement. Assuming that the X and the Y coordinates are decoupled, it is

possible to divide (3.15) into two one-dimensional optimization problems. Using the variational principle, the solution was shown to be a fifth-order polynomial [Flash and Hogan, 1985; Shadmehr, 2005], hence being able to reproduce our template (3.4). Here, we adopt the control-problem formulation in order to derive the result. This formulation gives us new insights for trajectory generation.

To minimize the jerk, each decoupled degree of freedom can be represented by a triple integrator. Let u denote the jerk and $y(t)$ the trajectory, then

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u \\ y &= x_1 \end{aligned} \tag{3.16}$$

or equivalently,

$$\dot{x} = \begin{pmatrix} x_2 & x_3 & u \end{pmatrix}^T =: f(t, x, u). \tag{3.17}$$

We define the reference signal $r(t) \in \mathbb{R}^3$ such that $r_1(t)$, $r_2(t)$, $r_3(t)$ denote the current position, velocity and acceleration of the target, respectively. Given the reference signal $r(t)$ and the desired time t_f, we wish to design $u(x, r, t)$ that produces the solution to

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_{t_0}^{t_f} u^2 \, dt \\ \text{subject to} \quad & (3.16) \\ & x(t_0) = x_0, \\ & x(t) = r(t) \text{ for } t \geq t_f. \end{aligned} \tag{3.18}$$

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

$$\begin{aligned} \dot{x}^* &= H_p(x^*, u^*, p, p_0) \\ \dot{p} &= -H_x(x^*, u^*, p, p_0) \\ x^*(t_0) &= x_0, \quad x^*(t_f) = r(t_f) \end{aligned} \tag{3.19}$$

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 correspond to the optimal solution. The optimal control maximizes the Hamiltonian, that is

$$H(x^*(t), u^*(t), p(t), p_0) \geq H(x^*(t), u, p(t), p_0) \tag{3.20}$$

Denoting the running cost for problem (3.18) by $L(x,u)$, the Hamiltonian is

$$\begin{aligned} H(x,u,p,p_0) &= \langle p, f(x,u)\rangle + p_0 L(x,u) \\ &= \begin{pmatrix} p_1 & p_2 & p_3 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ u \end{pmatrix} + p_0 u^2. \end{aligned} \tag{3.21}$$

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

$$\frac{\partial H}{\partial u} = 2p_0 u + p_3 = 0 \Rightarrow u^* = -\frac{p_3}{2p_0}. \tag{3.22}$$

Consequently, the Hamiltonian along the optimal trajectory is

$$H(x^*,u^*,p,p_0) = p_1 x_2^* + p_2 x_3^* - \frac{p_3^2}{4p_0}, \tag{3.23}$$

where

$$\begin{aligned} \dot{p}_1 &= -H_{x_1} = 0 \\ \dot{p}_2 &= -H_{x_2} = -p_1 \\ \dot{p}_3 &= -H_{x_3} = -p_2. \end{aligned} \tag{3.24}$$

These equations combined with (3.22) give us

$$u^* = k_1 t^2 + k_2 t + k_3, \tag{3.25}$$

with coefficients k_1, k_2, and k_3 to be determined. Integrating the control signal three times results in x_1, which is apparently a fifth-order polynomial. For the sake of simplicity, we assume $t_0 = 0$ and $x(t_f) = 0$. By matching the initial and final conditions, we obtain

$$\begin{aligned} y = x_1^* &= y_0(1 - 10t_n^3 + 15t_n^4 - 6t_n^5) + v_0 d_0 t_n(1 - 6t_n^2 + 8t_n^3 - 3t_n^4) \\ &\quad + \frac{a_0}{2} d_0^2 t_n^2 (1 - 3t_n + 3t_n^2 - t_n^3) \\ \dot{y} = x_2^* &= \frac{y_0}{d_0}(-30t_n^2 + 60t_n^3 - 30t_n^4) + v_0(1 - 18t_n^2 + 32t_n^3 - 15t_n^4) \\ &\quad + a_0 d_0 t_n (1 - \frac{9}{2}t_n + 6t_n^2 - \frac{5}{2}t_n^3) \\ \ddot{y} = x_3^* &= \frac{y_0}{d_0^2}(-60t_n + 180t_n^2 - 120t_n^3) + \frac{v_0}{d_0}(-36t_n + 96t_n^2 - 60t_n^3) \\ &\quad + a_0(1 - 9t_n + 18t_n^2 - 10t_n^3), \end{aligned} \tag{3.26}$$

and the optimal control signal is

$$\begin{aligned} \dddot{y} = u^* &= \frac{y_0}{d_0^3}(-60 + 360t_n - 360t_n^2) + \frac{v_0}{d_0^2}(-36 + 192t_n - 180t_n^2) \\ &\quad + \frac{a_0}{d_0}(-9 + 36t_n - 30t_n^2) \end{aligned} \tag{3.27}$$

where $t_n := (t - t_0)/d_0$ is the normalized elapsed time with respect to the duration $d_0 = t_f - t_0$ and y_0, v_0, and a_0 are initial position, velocity, and acceleration, respectively.

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

$$-V_t(t, x) = \inf_{u \in U} \{L(t, x, u) + \langle V_x(t, x), f(t, x, u) \rangle\}, \tag{3.28}$$

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

$$J(t, x, u) := \int_t^{t_f} L(s, x(s), u(s))\, ds + K(x(t_f)) \tag{3.29}$$

$$V(t, x) := \inf_{u[t, t_f]} J(t, x, u). \tag{3.30}$$

Here, $U \subseteq \mathbb{R}$ defines the control set, $L(\cdot)$ and $K(\cdot)$ denote the running cost and the terminal cost, respectively. For the minimum-jerk problem, we have

$$L(t, x, u) = u^2, \tag{3.31}$$

$$K(x(t_f)) = 0, \tag{3.32}$$

$$\begin{aligned} -V_t(t, x) &= \inf_{u \in U} \{u^2 + \langle V_x(t, x), f(t, x, u) \rangle\} \\ &= \min_{u \in \mathbb{R}} \{u^2 + V_{x_1} x_2 + V_{x_2} x_3 + V_{x_3} u\}. \end{aligned} \tag{3.33}$$

The optimum is achieved for

$$\frac{\partial (u^2 + V_{x_1} x_2 + V_{x_2} x_3 + V_{x_3} u)}{\partial u} = 0 \Rightarrow u^* = -\frac{V_{x_3}}{2}. \tag{3.34}$$

Therefore,

$$-V_t(t, x) = -\frac{V_{x_3}^2}{4} + V_{x_1} x_2 + V_{x_2} x_3. \tag{3.35}$$

As a result of the application of the maximum principle, (3.27) gives us an expression for $\dddot{y}$ along the optimal path. Now, considering the value function in (3.30), i.e., the cost to go, and using the definition of the cost function in (3.29), we conclude

$$V(t, x) = \int_t^{t_f} \dddot{y}^2(s)\, ds. \tag{3.36}$$

Note that to calculate (3.36), the initial state is the current state, the duration d_0 is equal to the remaining time $t_f - t$, and the normalized elapsed time t_n is $(s - t)/(t_f - t)$. It is straightforward to verify that the

resulting value function satisfies the HJB equation. Accordingly, from (3.34) we derive

$$u^* = -\frac{V_{x_3}}{2} = -\left(60\frac{x_1}{(t_f-t)^3} + 36\frac{x_2}{(t_f-t)^2} + 9\frac{x_3}{t_f-t}\right). \tag{3.37}$$

This reformulation gives us a control law for generating the minimum-jerk trajectory. The feedback signal is linear in the states, but nonlinear with respect to the time. Note that the same result can be obtained from (3.27). Interpreting the current point as the new initial point, we can set $t_n = 0$ to obtain

$$u^* = \dddot{y} = -\left(60\frac{x_1 - y_f}{(t_f-t)^3} + \frac{36x_2 + 24v_f}{(t_f-t)^2} + \frac{9x_3 - 3a_f}{t_f-t}\right). \tag{3.38}$$

Equation (3.38) gives us a solution for arbitrary initial and final points and the final time t_f.

3.4 Switching to Tracking

An obvious issue with (3.37) is that it is sensitive to the errors in the states when the time approaches t_f. Without loss of generality, assume there is some noise $\epsilon(t)$ in the velocity measurement. Thus, the control signal (3.37) is modified to

$$u = -\left(60\frac{x_1}{(t_f-t)^3} + 36\frac{x_2 + \epsilon(t)}{(t_f-t)^2} + 9\frac{x_3}{t_f-t}\right). \tag{3.39}$$

Substituting (3.39) into (3.16), we find that the closed-loop system obeys the differential equation

$$\dddot{y} = -\left(60\frac{y}{(t_f-t)^3} + 36\frac{\dot{y}}{(t_f-t)^2} + 9\frac{\ddot{y}}{t_f-t}\right) - 36\frac{\epsilon(t)}{(t_f-t)^2}. \tag{3.40}$$

Thus, as t approaches t_f, a very small noise level can blow up the control signal. A remedy to this problem is to switch to an infinite-horizon problem when the remaining time, $d = t_f - t$, becomes small. In the following, we show that this transition can be done smoothly by keeping d constant after reaching a certain remaining time.

The plot of the closed-loop poles of the system for fixed values of d is shown in Fig. 3.4. Considering (3.40), the characteristic equation of the closed-loop system for a fixed remaining time d is

$$d^3s^3 + 9d^2s^2 + 36ds + 60 = 0. \tag{3.41}$$

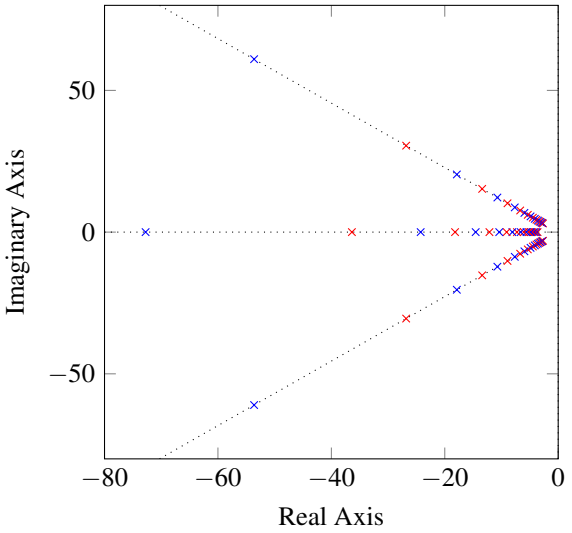


Figure 3.4 Closed-loop poles for various values of the remaining time $d = [1 : -0.05 : 0.05]$.

If p is a solution of (3.41) for $d = 1$, then p/d is a solution for any given d. Thus, $\arg(p)$ is independent of the remaining time while the poles move toward infinity as the remaining time approaches zero. Furthermore, we can rewrite the characteristic equation as

$$(s + \gamma\omega)(s^2 + 2\zeta\omega s + \omega^2) = 0, \tag{3.42}$$

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

3.5 Generalization of Template

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

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\left(60\frac{x_1}{(1-t)^3} + 36\frac{x_2}{(1-t)^2} + 9\frac{x_3}{1-t}\right). \end{aligned} \tag{3.43}$$

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 (3.38) is already in its general form, we consider using the coordinate transformation introduced in Section 3.2. In doing so 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 Sec. 3.7 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 (3.38) 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

$$\begin{aligned} \tilde{x}_1(\tilde{t}) &= \alpha^0 x_1(t) \\ \tilde{x}_2(\tilde{t}) &= \alpha^{-1} x_2(t) \\ \tilde{x}_3(\tilde{t}) &= \alpha^{-2} x_3(t). \end{aligned} \tag{3.44}$$

This results in

$$\begin{aligned} \tilde{x}_1' &= \tilde{x}_2 \\ \tilde{x}_2' &= \tilde{x}_3 \\ \tilde{x}_3' &= -\alpha^{-3}\left(60\frac{\tilde{x}_1}{\left(1-\frac{\tilde{t}}{\alpha}\right)^3} + 36\frac{\alpha\tilde{x}_2}{\left(1-\frac{\tilde{t}}{\alpha}\right)^2} + 9\frac{\alpha^2\tilde{x}_3}{1-\frac{\tilde{t}}{\alpha}} \right). \end{aligned} \tag{3.45}$$

The choice of new variables is done such that the first two differential equations remain the same. Since the variable names do not matter, we can rewrite the equations as below

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\left(60\frac{x_1}{(\alpha-t)^3} + 36\frac{x_2}{(\alpha-t)^2} + 9\frac{x_3}{\alpha-t} \right). \end{aligned} \tag{3.46}$$

In this case, it is obvious that α amounts to the total time. Note that the new initial conditions are calculated according to (3.44).

Now, we consider coordinate scaling. We define

$$\begin{aligned} \tilde{x}_1(t) &= \beta x_1(t) \\ \tilde{x}_2(t) &= \beta x_2(t) \\ \tilde{x}_3(t) &= \beta x_3(t). \end{aligned} \tag{3.47}$$

With these changes of variables,

$$\begin{aligned}
\dot{\tilde{x}}_1 &= \tilde{x}_2 \\
\dot{\tilde{x}}_2 &= \tilde{x}_3 \\
\dot{\tilde{x}}_3 &= -\left(60\frac{\tilde{x}_1}{(\alpha-t)^3} + 36\frac{\tilde{x}_2}{(\alpha-t)^2} + 9\frac{\tilde{x}_3}{\alpha-t}\right).
\end{aligned} \tag{3.48}$$

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

$$u = -\left(60\frac{\tilde{x}_1}{(\alpha-t)^3} + 36\frac{\tilde{x}_2}{(\alpha-t)^2} + 9\frac{\tilde{x}_3}{\alpha-t}\right). \tag{3.49}$$

Now, we show that (3.38) can be derived from (3.46). Note that the coordinate transformation (3.13) does not 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. 3.3, the inputs to trajectory generator controller is $e(t)$. Therefore, from (3.46) we have

$$\begin{aligned}
\dot{e}_1 &= e_2 \\
\dot{e}_2 &= e_3 \\
\dot{e}_3 &= -\left(60\frac{e_1}{(\alpha-t)^3} + 36\frac{e_2}{(\alpha-t)^2} + 9\frac{e_3}{\alpha-t}\right).
\end{aligned} \tag{3.50}$$

By renaming $y(t)$, $v(t)$, and $a(t)$ in (3.13) to x_1, x_2, and x_3, respectively, we obtain

$$\begin{aligned}
x_1(t) &= y_f + v_f(t-\alpha) + \frac{1}{2}a_f(t-\alpha)^2 - e_1(t) \\
x_2(t) &= v_f + a_f(t-\alpha) - e_2(t) \\
x_3(t) &= a_f - e_3(t).
\end{aligned} \tag{3.51}$$

The time derivatives of (3.51) are

$$\begin{aligned}
\dot{x}_1 &= v_f + a_f(t-\alpha) - \dot{e}_1 \\
\dot{x}_2 &= a_f - \dot{e}_2 \\
\dot{x}_3 &= -\dot{e}_3,
\end{aligned} \tag{3.52}$$

respectively. These equations can be rewritten using (3.50) to get

$$\begin{aligned}
\dot{x}_1 &= v_f + a_f(t-\alpha) - e_2 = x_2 \\
\dot{x}_2 &= a_f - e_3 = x_3 \\
\dot{x}_3 &= -\left(60\frac{e_1}{(\alpha-t)^3} + 36\frac{e_2}{(\alpha-t)^2} + 9\frac{e_3}{\alpha-t}\right) \\
&= -\left(60\frac{x_1-y_f}{(\alpha-t)^3} + \frac{36x_2+24v_f}{(\alpha-t)^2} + \frac{9x_3-3a_f}{\alpha-t}\right).
\end{aligned} \tag{3.53}$$

The rightmost sides of the equations are derived by substituting $e(t)$ from (3.51). By changing $\alpha - t$ to $t_f - t$, the proof of the validity of (3.38) is completed.

3.6 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, it is possible to find the minimum d such that the average velocity will be unchanged. This is readily possible if the maximum of each state for $d_0 = 1$ 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 the optimal curve is traversed, this value is constant. By differentiating (3.27) we obtain,

$$\begin{aligned}
\frac{d^4y}{dt^4} &= \frac{y_0-y_f}{d_0^4}(360-720t_n) + \frac{v_0}{d_0^3}(192-360t_n) - \frac{v_f}{d_0^3}(-168+360t_n) \\
&\quad + \frac{a_0}{d_0^2}(36-60t_n) - \frac{a_f}{d_0^2}(24-60t_n)
\end{aligned} \tag{3.54}$$

$$\frac{d^5y}{dt^5} = \frac{y_0-y_f}{d_0^5}(-720) + \frac{v_0+v_f}{d_0^4}(-360) + \frac{a_0-a_f}{d_0^3}(-60). \tag{3.55}$$

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 (3.55) gives us a fifth-order equation to calculate d_0,

$$cd^5 + 60(a_0-a_f)d^2 + 360(v_0+v_f)d + 720(x_0-x_f) = 0.$$

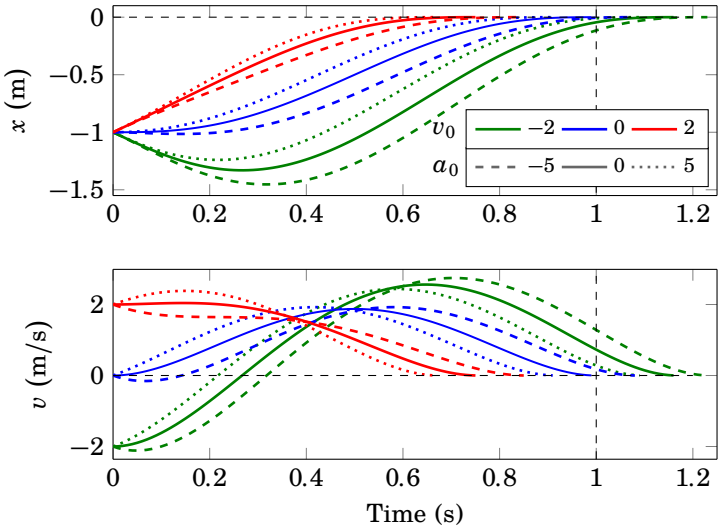


Figure 3.5 Curves starting from $x_0 = -1$, $v_0 = \{-2, 0, 2\}$, $a_0 = \{-5, 0, 5\}$ with $d^5y/dt^5 = 720$. The final time is not fixed as expected.

Figure 3.5 illustrates curves with $d^5y/dt^5 = 720$. Note that there is an intended drift in the final time for all the curves except the blue one, which starts in $x(0) = (-1, 0, 0)^T$. Figure 3.6 visualizes the remaining time d as a function of the initial state given the assumption of $d^5y/dt^5 = 720$.

3.7 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. The approximator is not unique and its choice affects the generalization of the template. Consider the following model

$$1 - t \approx \left| k_0 \operatorname{sgn}(y) \sqrt[3]{|y|} + k_1 \operatorname{sgn}(\dot{y}) \sqrt{|\dot{y}|} + k_2 \ddot{y} + k_3 \right|, \tag{3.56}$$

where y denotes the position. In this example, we have taken a feature selection approach. In other words, a linear combination of the powers of the state variables are considered, such that it gives a low error for the

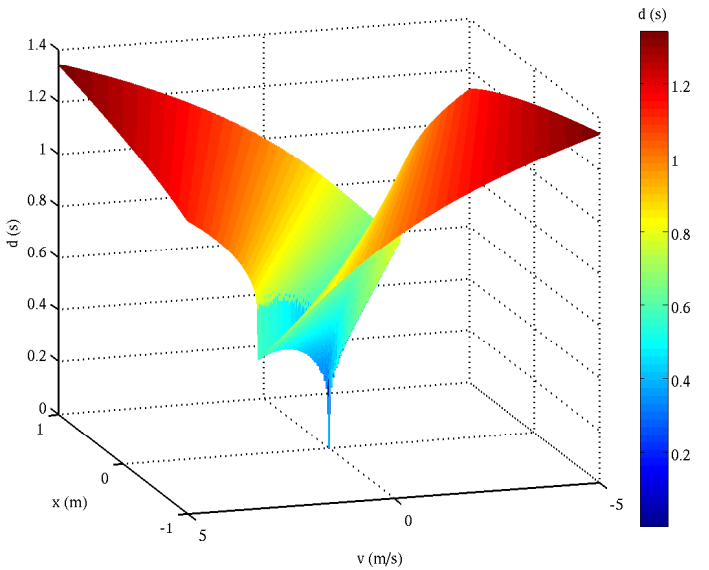


Figure 3.6 The remaining time as a function of the state for a fixed $d^5y/dt^5 = 720$. In this example, the plot is given for $a = 0$.

estimation of the remaining time along the motion template in (3.4). We use the least-squares method [Lawson and Hanson, 1995] to estimate the coefficients of the model for the template. The mean-square-error (MSE) 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).

Using the generalization method discussed in Sec. 3.5, we can easily extend the time-invariant model to different time and coordinate scales.

3.8 Simulations

In this section, we show simulation results of the closed-loop trajectory generation. We assume that the states are measurable. Therefore, instead of the output y in Fig. 3.3, the states x are used. To visualize the evolution of the states, we use phase portraits.

The first simulation shows the result of the control law in (3.49) with

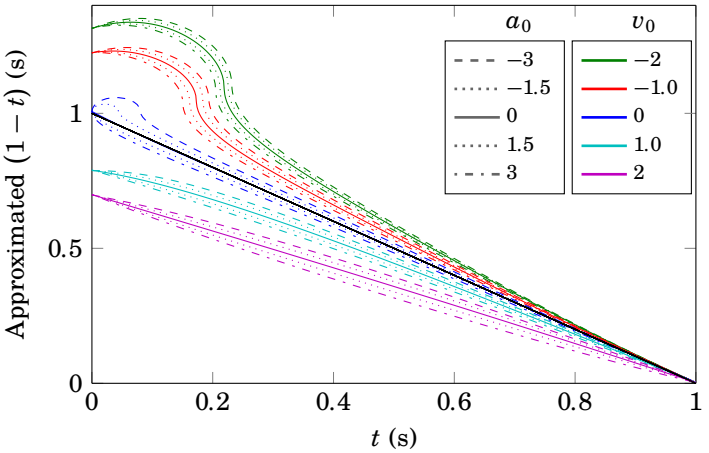


Figure 3.7 The approximated remaining time versus the actual time for $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-3 : 1.5 : 3]$. The black line shows the ideal case. The approximation when $v_0 = 0$ and $a_0 = 0$ coincides well with the ideal case, MSE ≈ 0.0365.

the change of $\alpha - t$ to $\max(\alpha - t, 0.06)$, i.e.,

$$u = -\left(60\frac{e_1}{d^3} + 36\frac{e_2}{d^2} + 9\frac{e_3}{d}\right), \quad d = \max(\alpha - t, 0.06). \tag{3.57}$$

The reference is set to zero and the initial position is set to -1. The initial velocity and the initial acceleration are varied. With $\alpha = 1$ s, the control law is expected to result in fifth-order polynomials with zero velocity and acceleration at the origin in one unit of time. In Fig. 3.8, the solid blue line in the middle corresponds to the template. 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 α. Figure 3.9 visualizes the evolution of the states in a 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-varying.

The second simulation deals with the time-invariant model obtained by approximating the remaining time along the template according to (3.56), see Figs. 3.10 and 3.11. Similarly to the previous simulation, the solid blue curve matches the template. Nevertheless, 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 a result of the fact that the approximation depicted in Fig. 3.7 overestimates the remaining time for negative initial velocities and underestimates it for

positive initial velocities.

The third simulation illustrates the result of the closed-loop trajectory generation for a moving target. Every second, a new target is activated. The objective is to intercept the target in 0.8 s and to continue tracking it until a new target is detected. In Fig. 3.12, the solid blue and dashed green curves correspond to the robot and the active target, respectively. The trajectory generator is the same as the one in the first simulation. As seen in the figure, there is a smooth transition to a tracking mode when the target is almost reached and the position, velocity, and acceleration of the generated trajectory are continuous.

In Figs. 3.13 and 3.14, 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 as compared to Fig. 3.12.

The fourth simulation investigates the effect of changing the parameters α and β for the time-invariant model described in Sec. 3.7. We consider the unit step response (i.e., a target 1 m away) and a step response with amplitude 2 in Fig. 3.15. 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 (3.7) and (3.9).

In the final simulation, we employ the resetting-time strategy according to Sec. 3.6. Deviations in the trajectory due to disturbances are detected by evaluating d^5y/dt^5 according to (3.55). The constraint $|d^5y/dt^5| < 2500$ is maintained by updating the remaining time, whenever the fifth time derivative of y hits the threshold. In this simulation, the measured states are affected by additive white Gaussian noise with a variance of 0.02. The results are shown in Fig. 3.16.

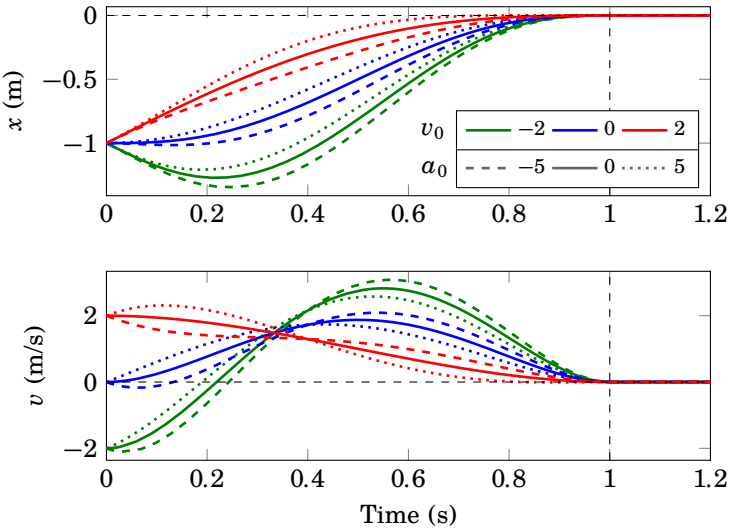


Figure 3.8 Simulation 1: Curves resulting from the control law (3.57), starting from $x_0 = -1$, $v_0 = \{-2, 0, 2\}$, and $a_0 = \{-5, 0, 5\}$ with $\alpha = 1$ s and $r(t) = 0$.

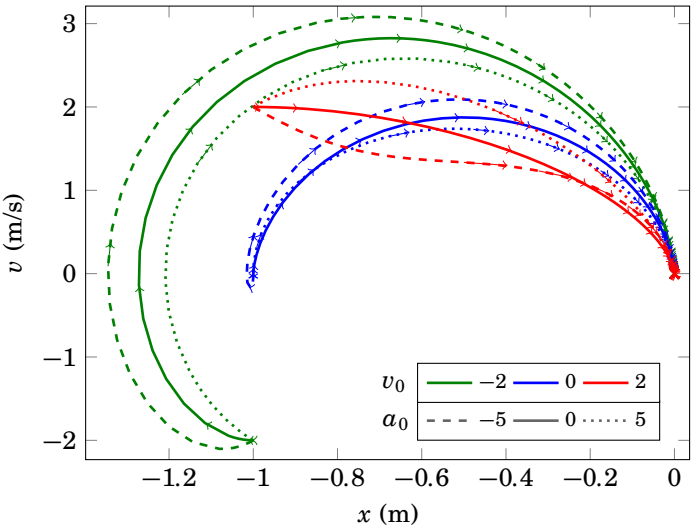


Figure 3.9 Simulation 1: Phase portrait for the trajectories in Fig. 3.8 starting from $x_0 = -1$, $v_0 = \{-2, 0, 2\}$, and $a_0 = \{-5, 0, 5\}$ with $\alpha = 1$ s and $r(t) = 0$.

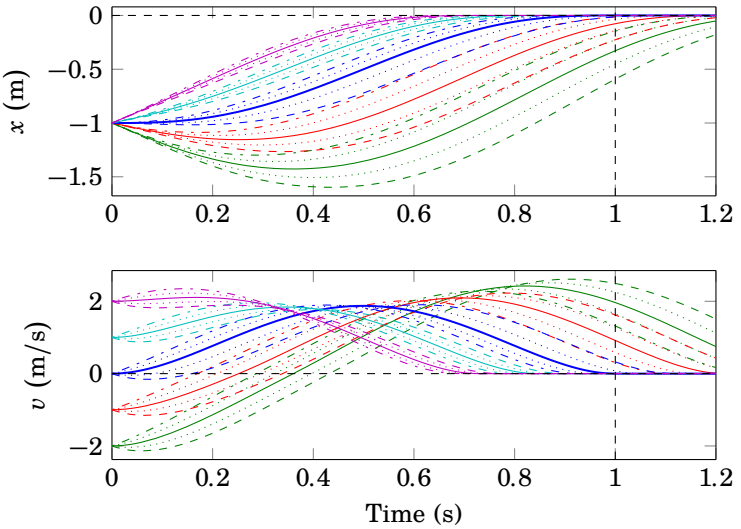


Figure 3.10 Simulation 2: Curves starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ for the time-invariant model using (3.56) and $\alpha = 1$. For the legends, see Fig. 3.11

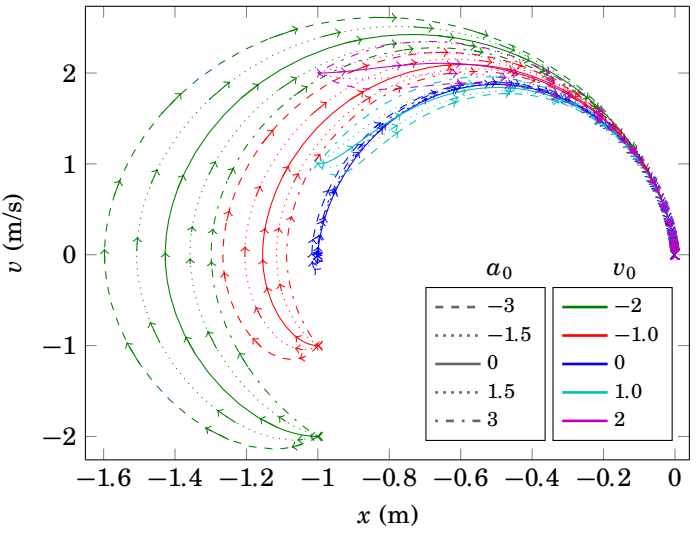


Figure 3.11 Simulation 2: Phase portrait for the trajectories in Fig. 3.10 starting from $x_0 = -1$, $v_0 = [-2 : 1 : 2]$, $a_0 = [-5 : 2.5 : 5]$ with an approximation of d.

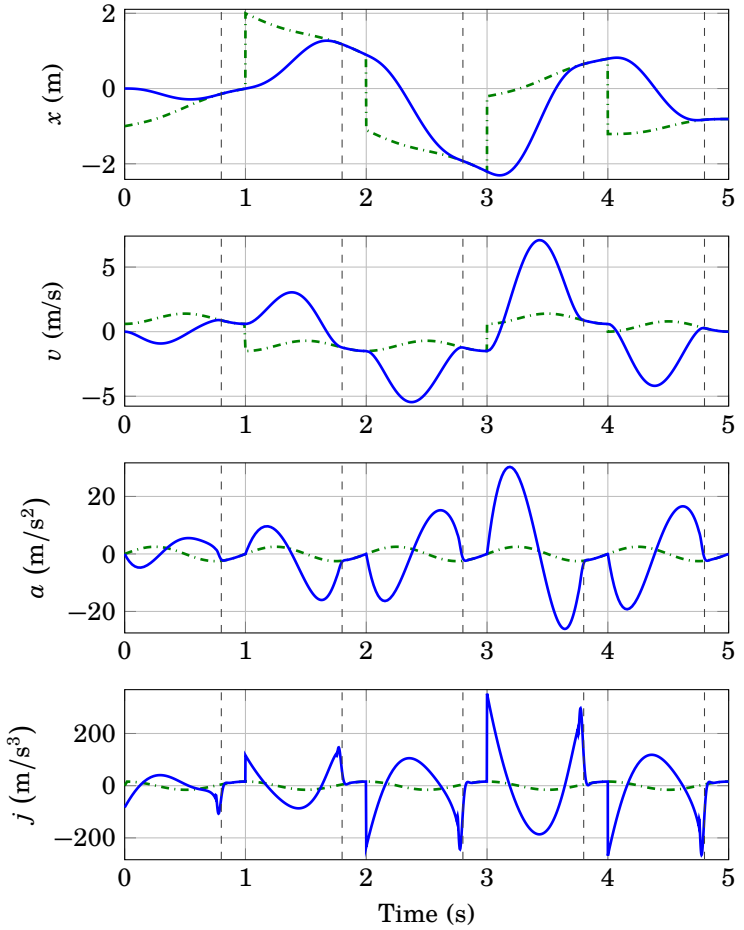


Figure 3.12 Simulation 3: Intercepting a moving target in 0.8 s; positions, velocities, acceleration, and jerk vs. time. The solid blue and the dashed green curves correspond to the robot and the target, respectively. Every second, a new target is activated. The trajectory generation softly switches to a tracking mode before the deadline. As expected, x, v, and a are continuous.

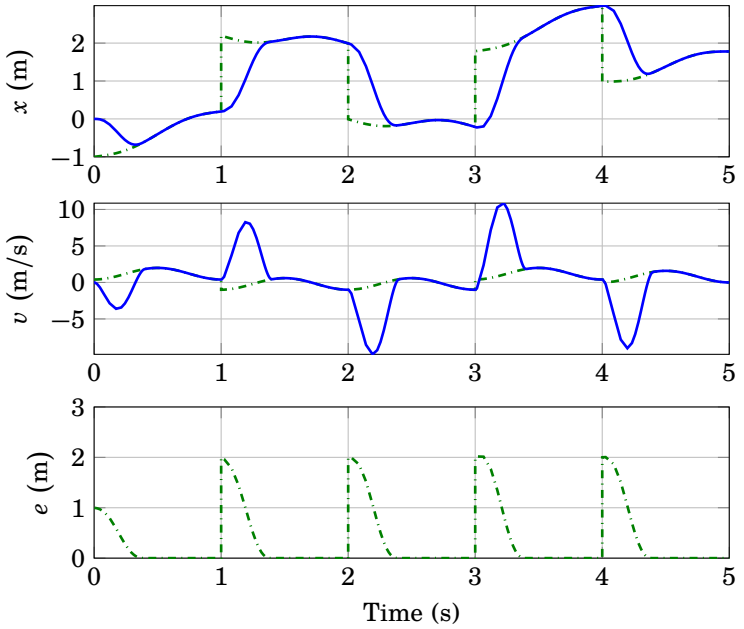


Figure 3.13 Simulation 3: Intercepting a moving target in 0.4 s; positions, velocities, and the position error as a function of time. The solid blue and the dashed green curves correspond to the robot and the target, respectively. Every second, a new target is activated.

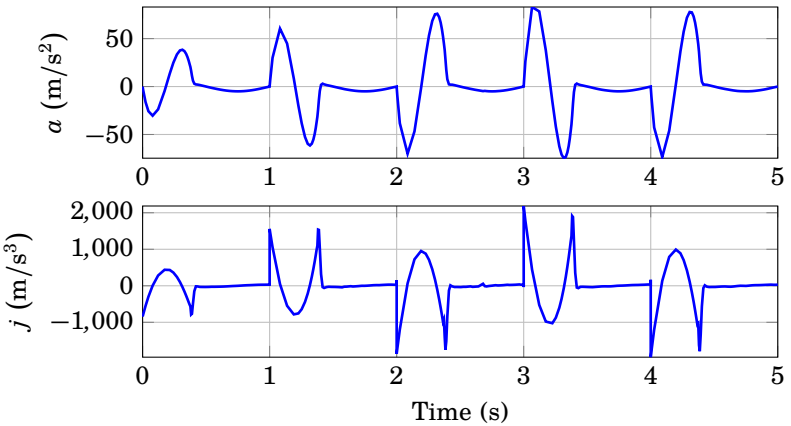


Figure 3.14 Simulation 3: Intercepting a moving target in 0.4 s; acceleration and jerk as a function of time.

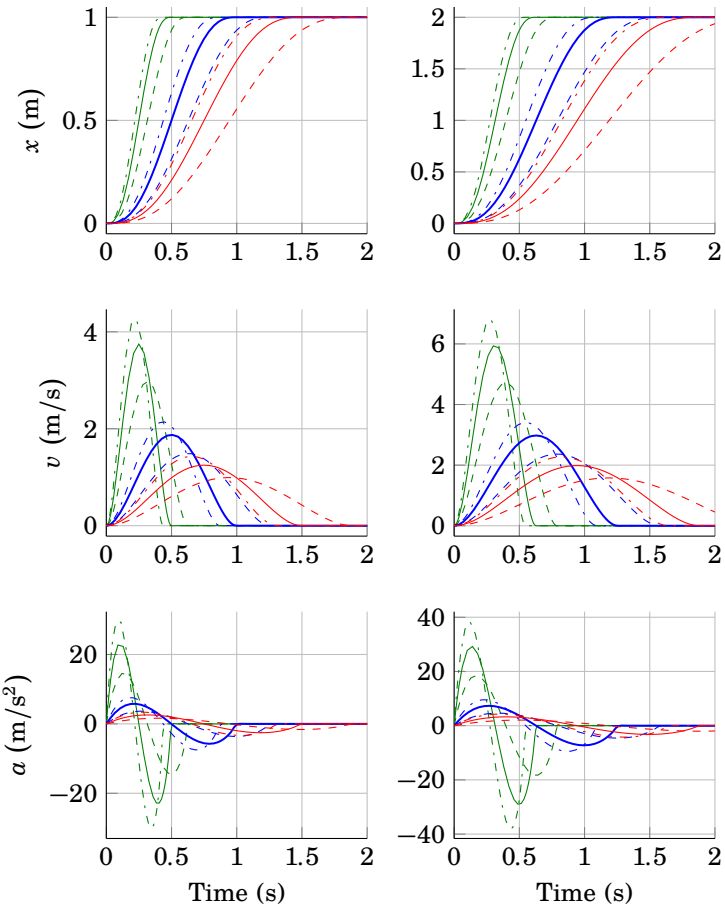


Figure 3.15 Simulation 4: Effects of changing parameters for the time-invariant model; $\alpha \in \{0.5, 1, 1.5\}$ corresponds to green, blue, and red, respectively, and $\beta \in \{0.5, 1, 1.5\}$ corresponds to dashed, solid, and dash-dot line, respectively. On the left, the step response with magnitude 1 and on the right, the step response with magnitude 2 are illustrated.

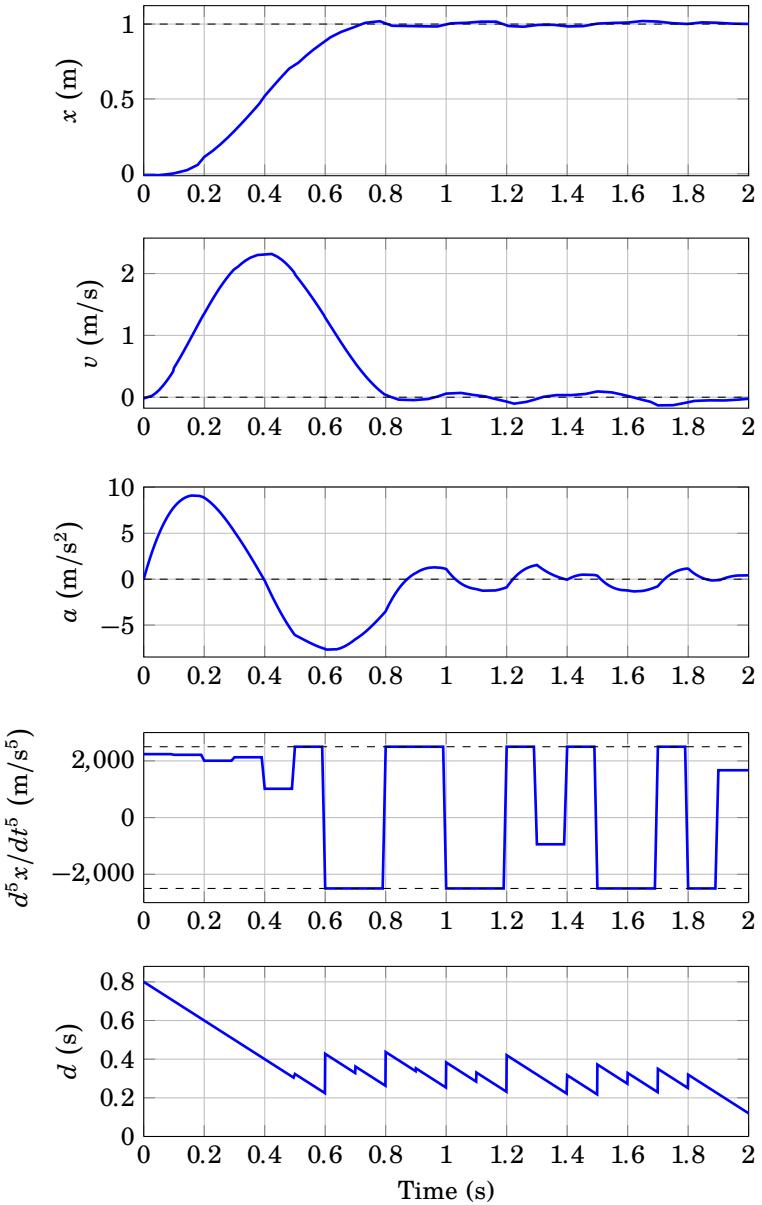


Figure 3.16 Resetting-time strategy: Additive Gaussian noise with variance 0.02 affects the states. The remaining time d is reset such that the condition $|d^5y/dt^5| \leq 2500$ is maintained.

3.9 Discussion

Considering the online trajectory generation for a moving target, there are at least two strategies. One can estimate a time $\hat{t}_f$ and a desired *future value* for the states $\hat{r}_f$, such that the generated trajectory meets this target state, i.e., $x(\hat{t}_f) = \hat{r}_f$. The procedure is repeated as soon as a better estimate is obtained. The other strategy is that the generated trajectory tracks the *current value* of the target but additionally superimposes a motion that eliminates the initial offset. These strategies do not necessarily lead to the same solution. The first strategy works better if an accurate estimation of the target's final state is possible, while the second strategy naturally leads to a smooth transition between trajectory planning and tracking modes. 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. 3.3, which depends only on the error $r(t) - y(t)$ in comparison to Fig. 3.1. More importantly, in case of the minimum-jerk trajectories the second strategy does not affect the optimality of the solution if at any time instant, $r(t)$ and t_f provide the best estimate of the target state.

In its current form, the controller assumes 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 [Thelander Andrén and Cervin, 2016]. This way, we have obtained a two-degrees-of-freedom controller with different responses to measurement noise and to changes in the reference signal.

Minimum-jerk trajectories can be time scaled to accommodate limitations on the kinematic variables similarly to the approach by [Dahl and Nielsen, 1990, Dahl and Nielsen (1990)]. If the constraints are constant, the solution to the minimum-time problem provides the lowest allowable t_f [Hehn and D'Andrea, 2011], i.e., any trajectory generated with a larger time interval than the minimum time satisfies the constraints.

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 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. Therefore, 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 final constraint 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 [Glad and Ljung, 2000].

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 Sec. 3.7, an important aspect is ensuring a global attractor for the approximated system, i.e., any initial error should eventually be reduced to zero.

3.10 Conclusions

Time and coordinate scaling are powerful operators for extending trajectories to a wider workspace. The coordinate transformation in Sec. 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 and laid the ground for closed-loop trajectory generation.

We proposed a controller model for trajectory generation with continuous reactions to the changes in the target. The Hamilton-Jacobi-Bellman equation was solved in order to find the optimal minimum-jerk controller. The result is a time-varying linear feedback law, which produces fifth-order polynomials for piece-wise constant target states. For this controller, we showed that limiting the remaining time from below naturally leads to a smooth transition between trajectory planning and tracking modes. Thus, we have obtained a fully reactive trajectory-generation method for possibly moving targets with the desirable properties of minimum-jerk trajectories.

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

Trajectory generation is an inherent problem in motion control for robotic systems, such as industrial manipulators and mobile platforms. The motion-planning problem is to define the path, and the course of motion as a function of time, namely the trajectory. 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 [LaValle, 2006]. For solving a trajectory-planning problem, it is in many cases beneficial to solve the path-planning problem first and then find a trajectory by assigning time to each point along the path [LaValle, 2006; Verscheure et al., 2009]. Nevertheless, for the specification of the problem, such separation is unnecessary in the optimization framework. Therefore, as long as there is no computational constraint, both path planning and trajectory planning can be solved in an integrated approach. Ultimately, this leads to a motion planning where the full potential of the system is utilized since the inherent interrelation between these problems is taken into account.

In this chapter, we summarize the most common trajectory-generation problems using the optimization framework. A number of examples and numerical results are provided. The impact of model selection for optimal control problems is briefly studied. Specifically, we compare a kinematic model with a dynamic model.

4.2 Optimal Trajectory Problems

Assume a system according to

$$\begin{aligned} \dot{x} &= f(t, x, u), \qquad x(t_0) = x_0, \\ y &= g(x, u), \end{aligned} \tag{4.1}$$

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

An optimal control problem is usually characterized by a cost functional to be minimized. A generic cost functional for a deterministic setting in the Bolza form can be formulated as [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, respectively,

$$L : \mathbb{R} \times X \times U \to \mathbb{R}$$

is the running cost, and

$$K : \mathbb{R} \times X \to \mathbb{R}$$

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.

We enumerate a few trajectory-planning problems in the following with no intention to be comprehensive. For all the problems, a dynamical system such as (4.1) is assumed. In the list, Q and R are appropriate weighting matrices, F and G are constraint matrices for the states and the inputs, respectively, $g(\cdot)$ and $h(\cdot)$ are two, possibly non-linear, functions defining terminal constraints and path constraints, respectively, and r is the reference signal. The trajectory-planning problems are:

I. 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 $e = r - y$.

II. Fixed-time 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 $y(t_f) = y_f$.

III. 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 $y(t_f) = g(t_f)$.

IV. Time-optimal point-to-point planning (fixed end-point, free-time)

$$J(u) = \int_0^{t_f} dt, \tag{4.6}$$

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

V. Time-optimal point-to-point planning given a path (fixed end-point, free-time)

$$J(u) = \int_0^{t_f} dt, \tag{4.7}$$

subject to $Fx \leq b$, $Gu \leq a$ and $y(t_f) = y_f$, and $h(x(t)) = 0$.

VI. Time-optimal point-to-point planning given an integral constraint (fixed end-point, free-time),
subject to $Fx \leq b$, $Gu \leq a$, $y(t_f) = y_f$ and $\tilde{J}(u, t_f) \leq V(t_f)$, where

$$\tilde{J}(u, t) = \int_0^{t} (x^T Q x \ + u^T R u)\, dt. \tag{4.8}$$

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

Problem I requires a known reference, which is typically the result of another optimization problem. This type of problem is mainly used for the online application of trajectory generation, e.g., in model predictive control (MPC). Problem II is a point-to-point planning type with an explicit constraint on time. If we do not limit ourselves to a quadratic cost functional, Problem II can also be used to define a shortest traveled-distance problem. For example, in two dimensions, the length of a curve is calculated according to

$$\text{length} = \int_0^{t_f} \sqrt{v_x^2 + v_y^2}\, dt \tag{4.9}$$

where v_x and v_y denote the velocities in x and y directions, respectively. Problem III can be used for the tasks requiring coordination between two objects, for example catching a flying object.

The minimum-time problems are typically associated with Problems IV and V. When the degrees of freedom are decoupled, Problem IV can be solved

analytically. Efficient implementation of the solution is readily available for CNC machines and industrial robots [Kröger, 2011b]. For Problem V, the path is typically obtained from a path planner taking into account the geometrical constraints. Thereafter, a trajectory is assigned to the path. The advantage of this approach is that there exists a convex reformulation of the trajectory generation [Verscheure et al., 2009].

In Problem VI, $V(\cdot)$ is a function defining the budget for the integrated cost for a given time. An interesting feature of this problem is that although it is time optimal, it does not necessarily lead to a bang-bang solution. The reason is that whenever the integral constraint is active, the solution can be obtained by minimizing (4.8) subject to the rest of the constraints, but given a fixed final time. In other words, the problem is equivalent to minimizing an integral cost while an optimal final time has to be simultaneously calculated. Therefore, in many scenarios, the solution to this problem can be of more relevance than time-optimal, fixed-time or time-energy optimal [Shiller, 1994b; Verscheure et al., 2008] problems. For example, defining

$$V(t) := \min_u \tilde{J}(u, t) \tag{4.10}$$

results in minimum-time trajectories with predefined trajectory profiles. As another example consider

$$V(t) := Ct. \tag{4.11}$$

In this case, C has the interpretation of the maximum average rate of expenditure. For mechanical systems, C can for example be used to introduce a power limit. Let us consider a double-integrator system according to

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u, \end{aligned} \tag{4.12}$$

where $u \in \mathbb{R}$ is the input signal interpreted as acceleration, $x_2 \in \mathbb{R}$ and $x_1 \in \mathbb{R}$ denote the velocity and position, respectively. We define $V(t)$ according to (4.11) and solve

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & (4.12), \\ & \dot{\tilde{J}} = x_2^2 + wu^2, \\ & \tilde{J} \leq Ct_f, \\ & x(0) = [0,\ 0]^{\mathrm{T}}, \\ & x(t_f) = [y_f,\ 0]^{\mathrm{T}}. \end{aligned} \tag{4.13}$$

Figure 4.1 illustrates the numerical solution to (4.13) for $C = 0.2$, $w = 0.5$, and $y_f \in \{1/3, 1, 5/3\}$.

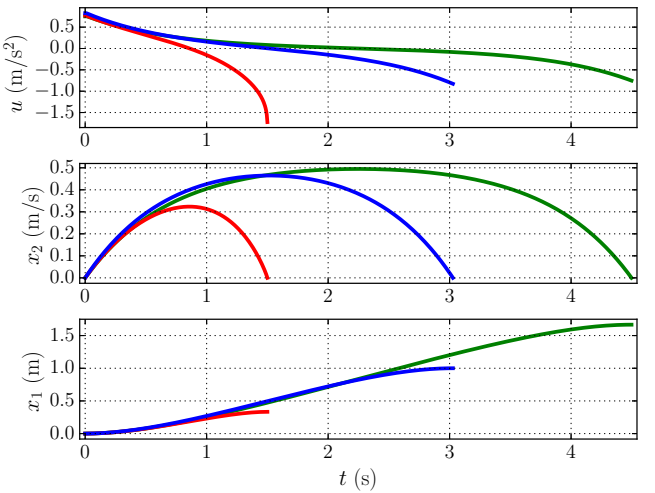


Figure 4.1 Solution of (4.13) for various final positions y_f; the positions, velocities, and accelerations are shown. The red, blue, and green curves correspond to $y_f = 1/3$, $y_f = 1$, and $y_f = 5/3$, respectively. The time-optimal solution is not bang-bang. The final time varies as the final target is changed.

4.3 Simple Models

In this section, we present two simple models corresponding to a kinematic model and a dynamic model, which are used later to find optimal trajectories for robots. By kinematic models, we mean models involving only kinematic variables, i.e., position, and its derivatives, while dynamic models include the relation between kinematic variables and dynamic variables, e.g., forces.

Kinematic Model

Assume that we have a robot with two degrees of freedom. The motion of the robot can be described by a two-dimensional double-integrator according to

$$\begin{aligned} \dot{x} &= v_x\,, \\ \dot{y} &= v_y\,, \\ \dot{v}_x &= u_x\,, \\ \dot{v}_y &= u_y\,. \end{aligned} \tag{4.14}$$

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.

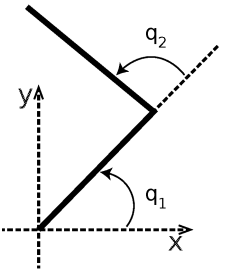


Figure 4.2 Schematic of a two-link robot.

Dynamic Model

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

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{4.15}$$

where $M(q)$ is a symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q}$ the vector of centripetal and Coriolis torques, and $G(q)$ the term due to gravity, $q \in \mathbb{R}^n$ denotes the vector of joint angles, and n denotes the number of degrees of freedom. 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 (Fig. 4.2). If we limit the motion to a horizontal plane, the effect of the gravity can be ignored. In this case, the matrices corresponding to (4.15) are

$$M(q) = \begin{pmatrix} a_1 + 2a_3\cos(q_2) + a_2 & a_3\cos(q_2) + a_2 \\ a_3\cos(q_2) + a_2 & a_2 \end{pmatrix} \tag{4.16}$$

$$C(q,\dot{q}) = \begin{pmatrix} -a_3\sin(q_2)\dot{q}_2 & -a_3\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ a_3\sin(q_2)\dot{q}_1 & 0 \end{pmatrix} \tag{4.17}$$

where

$$a_1 = m_2 L_1^2 + m_1 r_1^2 + I_1 \tag{4.18}$$

$$a_2 = m_2 r_2^2 + I_2 \tag{4.19}$$

$$a_3 = m_2 L_1 r_2. \tag{4.20}$$

The subscripts of q indicate the elements of the vector. With indices corresponding to the link number, m, L, r, and I denote 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 our numerical simulations, 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

4.4 Minimum-Time Point-to-Point Problems

In this section, we consider a number of different scenarios to evaluate the choice of model and the constraints. The focus is the comparison of dynamic versus kinematic models. The problems are instances of time-optimal point-to-point trajectory planning, Problem IV and VI in Sec. 4.2, with various constraints on the states or the input signals.

We choose z as the state variable for both models presented in Sec. 4.3. The inputs are also denoted by u. Hence, we can rewrite the system (4.14) as

$$\dot{z} = f_k(z, u), \tag{4.21}$$

where $z = [x,\ y, v_x,\ v_y]$ and $u = [u_x,\ u_y]$. Similarly, the system (4.15) can be rewritten as

$$\dot{z} = f_d(z, u) \tag{4.22}$$

where $z = [q_1,\ q_2, \dot{q}_1,\ \dot{q}_2]$ and $u = [\tau_1,\ \tau_2]$. In both models, z is a purely kinematic variable. For the kinematic model, u is, however, a kinematic variable, while for the dynamic model u is a dynamic variable.

The time-optimal point-to-point problems considered have this common structure

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & t_f \\ \text{subject to} \quad & \dot{z} = f(z, u), \\ & z(0) = z_0, \\ & z(t_f) = 0. \end{aligned} \tag{4.23}$$

For the kinematic model, we use

$$z_0 = [2,\ 1,\ 0,\ 0]^T, \tag{4.24}$$

and for the dynamic model, we use

$$z_0 = [\frac{\pi}{4},\ \frac{\pi}{10},\ 0,\ 0]^T. \tag{4.25}$$

For problem (4.23), additional constraints are considered according to one of the following scenarios:

1. Kinematic model $f = f_k$, with inequality constraints purely on kinematic variables

$$|u_1| \leq 1, \ |u_2| \leq 1. \tag{4.26}$$

2. Dynamic model $f = f_d$, with inequality constraints purely on dynamic variables

$$|u_1| \leq 1, \ |u_2| \leq 1. \tag{4.27}$$

3. Dynamic model $f = f_d$, with inequality constraints purely on kinematic variables

$$|\dot{z}_3| \leq 2, \ |\dot{z}_4| \leq 2. \tag{4.28}$$

4. Dynamic model $f = f_d$, with inequality constraints on both kinematic and dynamic variables

$$\begin{aligned} &|u_1| \leq 1, \ |u_2| \leq 1, \\ &|\dot{z}_3| \leq 10, \ |\dot{z}_4| \leq 10. \end{aligned} \tag{4.29}$$

5. Kinematic model $f = f_k$, with inequality constraints on an integral cost

$$\begin{aligned} \dot{\tilde{J}} &= z_2^2 + z_3^2 + \gamma(u_1^2 + u_2^2), \\ \tilde{J} &\leq C t_f, \end{aligned} \tag{4.30}$$

   where $\gamma = 0.5$ and $C = 0.2$.

6. Dynamic model $f = f_d$, with inequality constraints on an integral cost

$$\begin{aligned} \dot{\tilde{J}} &= z_2^2 + z_3^2 + \gamma_2(\dot{z}_2^2 + \dot{z}_3^2) + \gamma_1(u_1^2 + u_2^2), \\ \tilde{J} &\leq C t_f, \end{aligned} \tag{4.31}$$

   where $\gamma_1 = 0.5$, $\gamma_2 = 0.01$, and $C = 0.2$.

4.5 Additional Constraints

The time-optimal control problem for systems with 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, e.g., in

the same manner as for the time-energy optimal problems [Shiller, 1994a]. If the DOF can be decoupled, fixed trajectory profiles can also be assigned to those DOF that have to be synchronized [Kröger, 2011a]. Furthermore, without compromising the optimal time, we can solve a second optimization problem. The new problem is fixed time. Hence, a combination of states and input signals can be penalized. Given t_f from the solution of the minimum-time problem, we solve

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_0^{t_f} z_2^3 + z_4^2 + w(u_1^2 + u_2^2)\, dt \\ \text{subject to} \quad & \dot{z} = f(z, u), \\ & z(0) = z_0, \\ & z(t_f) = 0. \end{aligned} \tag{4.32}$$

where w is a small weight. In the numerical examples, we set $w = 0.001$ for the kinematic model and to $w = 0.02$ for the dynamic model. Adding penalties on the higher derivatives can also be beneficial for smoothing out the signals.

Obstacle Avoidance

It is relevant to include some non-convex constraints. Such constraints usually appear in obstacle-avoidance problems. We study the effect of a constraint of the following form

$$(y - y_0)^2 + (x - x_0)^2 \geq c^2. \tag{4.33}$$

This defines the area outside of a circle with the radius of c centered at (x_0, y_0).

4.6 Simulation Results

The results concerning Scenarios 1–6 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].

The resulting trajectory for Scenario 1 after the 2nd optimization is depicted in Fig. 4.3. Figure 4.4 illustrates the solution of Scenario 1, including obstacle avoidance. In this case the time-optimal problem does not have extra degrees of freedom. Therefore, a second optimization was not required.

Scenario 2 is addressed next. As we see in Fig. 4.5, one of the control signals switches rapidly. This is because of the fact that there is no unique

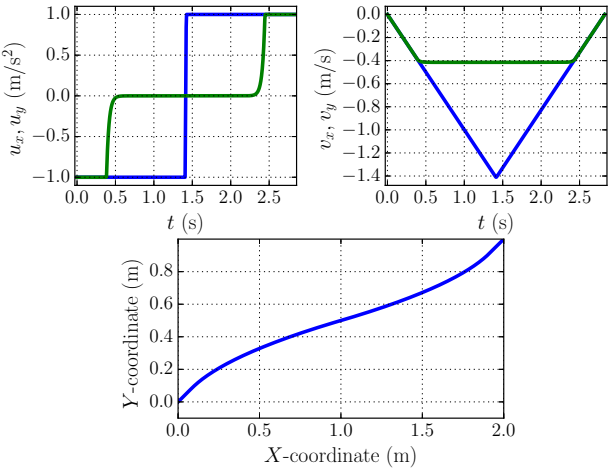


Figure 4.3 The trajectory for the kinematic model with only kinematic inequality constraints (Scenario 1) after the 2nd optimization. The blue and green curves correspond to motion along the x-axis and y-axis, respectively. In the X–Y plot, the curve corresponds to the position in 2D.

solution to this time-optimal problem, which results in the strange behavior of the solver. As shown in Fig. 4.6, this issue can be resolved by solving a second optimization problem according to Sec. 4.5. In Fig. 4.7, the result for the time-optimal point-to-point motion planning with the obstacle avoidance using the dynamic model is given.

Figure 4.8 illustrates the result of Scenario 3. Comparing with Fig. 4.3, it is evident that the complete dynamics of the system do not play a role in the obtained trajectory because of the tight bounds on the kinematic variables. This is, however, not the case when a mixed set of constraints are active as seen in Fig. 4.9.

For the last two scenarios, where an integral inequality constraint has been considered, there is no need to consider a second optimization.The results for the kinematic and the dynamic models are shown in Figs. 4.10 and 4.11, respectively, which are qualitatively similar.

4.7 Conclusion

A number of optimization problems applicable to trajectory generation were discussed. We also presented numerical examples for illustration. An important observation is that the time-optimal solution for a problem with

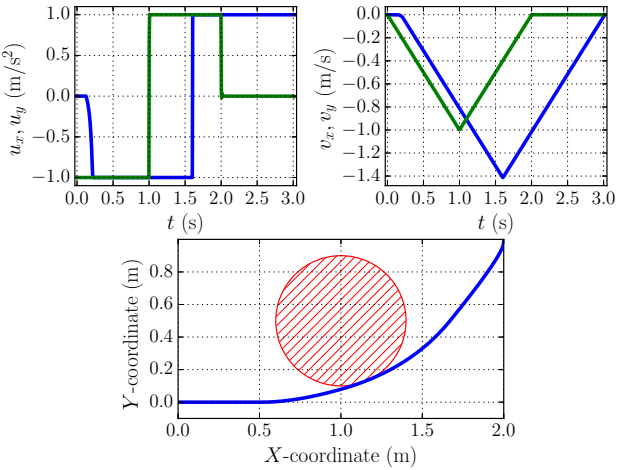


Figure 4.4 The trajectory for Scenario 1 with obstacle avoidance (red circle in the lower plot). The blue and green curves correspond to motion along the x-axis and y-axis, respectively. In the X–Y plot, the curve corresponds to the position in 2D.

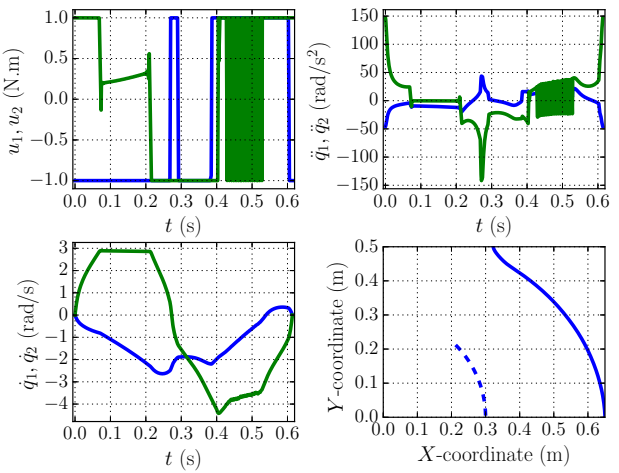


Figure 4.5 The trajectory of Scenario 2 without a 2nd optimization. The blue and green curves correspond to joint one and two, respectively. For joint two, the control signal and the acceleration switch rapidly without much effect on the outputs. Compare with 4.6.

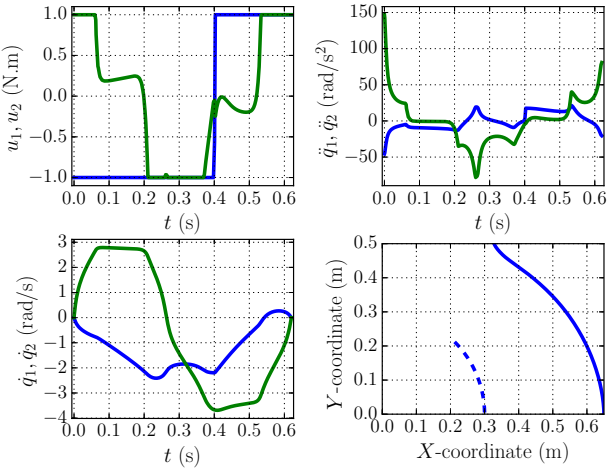


Figure 4.6 The trajectory for the dynamic model with only dynamic inequality constraints (Scenario 2) after the 2nd optimization. The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the 2nd joint, respectively.

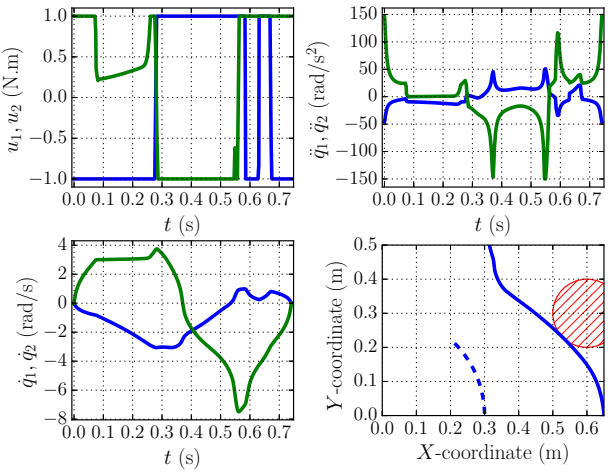


Figure 4.7 The trajectory of Scenario 2 with obstacle avoidance (red circle in the lower plot). The blue and green curves correspond to joint one and two, respectively.

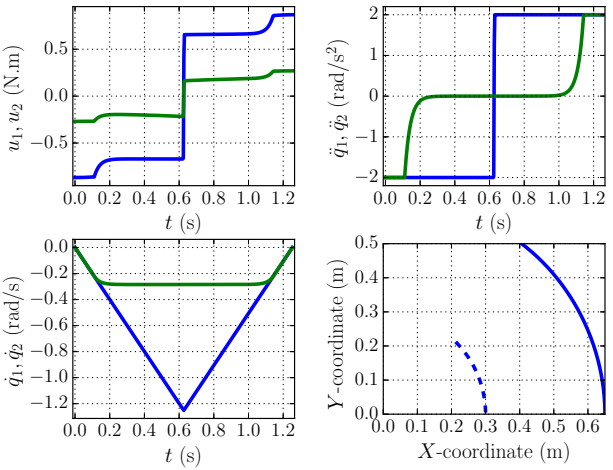


Figure 4.8 The dynamic model with only active kinematic constraints (Scenario 3) after the 2nd optimization. Compare with Fig. 4.3.

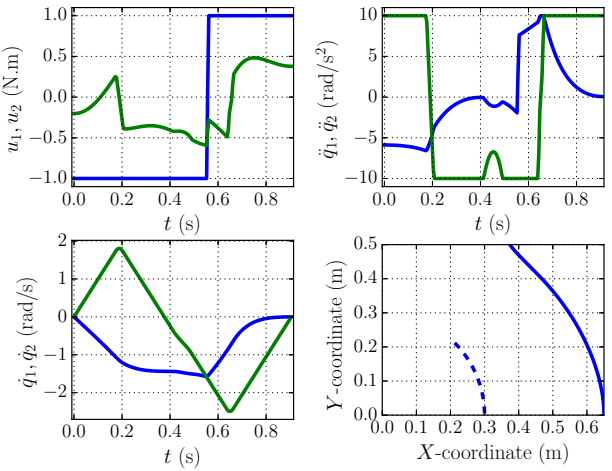


Figure 4.9 The dynamic model with both kinematic and dynamic constraints (Scenario 4) after the 2nd optimization. The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the 2nd joint, respectively.

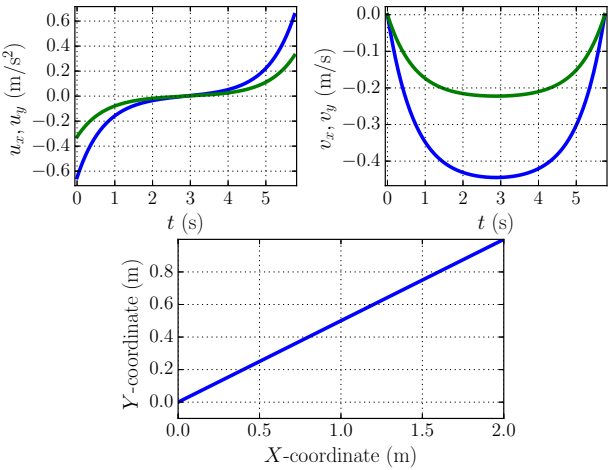


Figure 4.10 The trajectory for the kinematic model with the integral constraint (Scenario 5). The blue and green curves correspond to motion along the x-axis and y-axis, respectively. In the X–Y plot, the curve corresponds to the position in 2D.

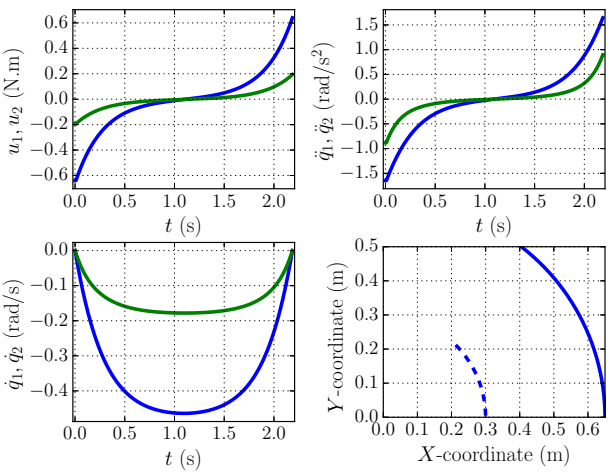


Figure 4.11 The trajectory for the dynamic model with the integral constraint (Scenario 6). The blue and green curves correspond to joint one and two, respectively. In the X–Y plot, the solid and the dashed line correspond to the position of the end-effector and the 2nd joint, respectively.

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. Moreover, we introduced a new time-optimal point-to-point trajectory generation problem, by including an integral constraint. We showed that as long as the constraint is tight, the solution behaves as the fixed-time point-to-point problem with the same cost function as $\tilde{J}$.

In the case of holonomic dynamics, if no constraints on dynamic variables are given or the constraints on kinematic variables are tight, the dynamic model can be ignored and the system can be approximated by a double integrator. This is because of the fact that the matrix $M(q)$ is positive definite and thus invertible [Siciliano et al., 2009]. Consequently, using the inverse dynamics, it is always possible to compute the required τ within the constraints to achieve the same solution as with the kinematic model.

In the motion-planning procedure, various modeling assumptions have to be made. The major difference with respect to modeling is whether a kinematic or a dynamic model is considered. Although dynamic models are required for achieving the highest possible performance, a kinematic model suffices in many applications, given conservative constraints on the kinematic variables—i.e., on the position, the velocity, and the acceleration.

5

An Analytic Solution to Fixed-Time Trajectory Planning

5.1 Introduction

A desired characteristic of motion planning in uncertain environments is the ability to react to sensor inputs [Kröger and Wahl, 2010]. Motion replanning based on sensor information requires algorithms that can quickly generate a motion trajectory. The concept of online trajectory generation, i.e., generating trajectories within each control cycle, is discussed in [Kröger, 2010]. There are many methods for online trajectory generation with the objective of time-optimality based on analytic expressions [Castain and Paul, 1984; Macfarlane and Croft, 2003; Haschke et al., 2008; Kröger and Wahl, 2010]. The difference between these methods lies in their generality with regard to the constraints and the initial and final states of the desired motion.

Using the existing analytic solutions, time-optimal or nearly time-optimal trajectories can be computed extremely fast. While the minimum-time trajectories are of interest for defining an upper bound for the productivity of a robotic system, they can maximize the wear of the system. In practice, other factors such as coordination between different units often determine the required time. Hence, the solution to fixed-time problems with a cost function motivated by the application can prove valuable by reducing the wear of the robotic system. A common approach to fixed-time problems is fitting a piece-wise polynomial between the starting point and a final point [Paul, 1972; Lin et al., 1983; Taylor, 1979] without considering an explicit cost function. Optimizing the energy or power consumption [Stryk and Schlemmer, 1994] or the effort [Martin and Bobrow, 1997] was suggested in other approaches. The solutions were obtained by numerical methods either by discretization or optimizing parameters over a set of basis functions.

A sub-optimal solution to the fixed-time trajectory planning considering a more generic cost function was proposed in [Dulęba, 1997].

In this chapter, we consider the fixed-time trajectory-generation problem with a quadratic cost under velocity and acceleration constraints. The resulting trajectory can for example be used in pick-and-place tasks to transfer the current state (position and velocity) of a manipulator to a new one in a given time. The purpose is to find a computationally efficient solution, such that a new trajectory can be computed quickly if it is called upon by new sensor measurements. We cast this problem into the framework of optimal control with state variable inequality constraints (SVIC). In order to find a solution to the optimal control problem, we present two main theorems 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. Specifically, we use the direct adjoining approach to find a solution to a fixed-time trajectory planning problem. This leads to a system of equations that determines a set of parameters for the analytic solution. The system of equations is solved numerically and the resulting trajectories are compared with the numerical solution obtained by discretizing the model and using an interior-point method [Boyd and Vandenberghe, 2004].

5.2 Problem Formulation

Since we are concerned with kinematic variables, specifically constraining the velocity and acceleration, a double integrator is a sufficient model for each degree of freedom (DOF). This implies that the acceleration of each DOF can be independently controlled. The robotic system has an initial error with respect to a desired final state (a certain position at rest), which is supposed to become zero at a given fixed time. The error in the position denoted by x_1 is unconstrained, but there are constraints on both the acceleration u and on the velocity x_2. We assume a quadratic cost function specified by $R > 0$ and a diagonal matrix $Q := \mathrm{diag}(q_1, q_2) > 0$. The problem can be compactly written as:

$$\underset{u}{\text{minimize}} \quad \int_0^{t_f} x^T Q x + u^T R u \, dt \tag{5.1a}$$

$$\text{subject to} \quad \dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) \tag{5.1b}$$

$$|u(t)| \leq 1 \tag{5.1c}$$

$$|x_2(t)| \leq c \tag{5.1d}$$

$$x(0) = \begin{pmatrix} x_0, & v_0 \end{pmatrix}^T, \quad x(t_f) = 0_{2\times 1} \tag{5.1e}$$

5.3 Preliminaries

Following the presentation of [Seierstad and Sydsæter, 1987; Hartl et al., 1995], the control problem with SVIC is specified by an objective functional J to be maximized subject to constraints on the states and the control signal:

$$\begin{aligned}
&\underset{u}{\text{maximize}} && J(u) = \int_0^{t_f} F(x(t), u(t), t)dt + S(x(t_f), t_f) && \text{(5.2a)}\\
&\text{subject to} && \dot{x}(t) = f(x(t), t), \quad x(0) = x_i && \text{(5.2b)}\\
&&& g(x(t), u(t), t) \geq 0 && \text{(5.2c)}\\
&&& h(x(t), t) \geq 0 && \text{(5.2d)}\\
&&& a(x(t_f), t_f) \geq 0 && \text{(5.2e)}\\
&&& b(x(t_f), t_f) = 0, && \text{(5.2f)}
\end{aligned}$$

where

$$h : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^q, \qquad g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^s$$
$$a : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^\ell, \qquad b : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{\ell'}.$$

General Definitions and Conditions

Following [Hartl et al., 1995], the order of pure state constraints as well as junction times are defined here. Moreover, a constraint qualification condition is 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 Consider the derivatives of the function h with respect to time

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

The state constraint is of order p iff

$$h_u^i(x, u, t) = 0, \text{ for } 0 \leq i \leq p-1, \quad h_u^p(x, u, t) \neq 0. \tag{5.3}$$

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 for terminal constraints requires

$$\operatorname{rank} \begin{bmatrix} \partial a/\partial x & \operatorname{diag}(a) \\ \partial b/\partial x & 0 \end{bmatrix} = \ell + \ell', \tag{5.4}$$

where ℓ and ℓ' are the dimensions of the range of the functions a and b, respectively. Additionally, for mixed constraints, i.e., the constraints involving both states and input signals, we require

$$\operatorname{rank}[\partial g/\partial u \quad \operatorname{diag}(g)] = s, \tag{5.5}$$

where s is the dimension of the range of the function g.

Direct Adjoining Approach

In this approach, the mixed constraints as well as the pure constraints are directly adjoined to the Hamiltonian H to form the Lagrangian L. The Hamiltonian and the so called D-form Lagrangian are defined as [Hartl et al., 1995]

$$H(x, u, \lambda_0, \lambda, t) = \lambda_0 F(x, u, t) + \lambda f(x, u, t) \tag{5.6}$$

$$L(x, u, \lambda_0, \lambda, \mu, \nu, t) = H + \mu g(x, u, t) + \nu h(x, u, t). \tag{5.7}$$

The costate is a mapping $\lambda(\cdot) : [0, t_f] \to \mathbb{R}^n$ and multiplier functions $\mu(\cdot)$ and $\nu(\cdot)$ are mappings from $[0, t_f]$ into $\mathbb{R}^s$ and $\mathbb{R}^q$, respectively. The control region is defined as:

$$\Omega(x, t) = \{u \in \mathbb{R}^m \mid g(x, u, t) \geq 0\} \tag{5.8}$$

Theorem 1—Informal Theorem 4.1 from [Hartl et al., 1995]
Let $\{x^*(\cdot), u^*(\cdot)\}$ be an optimal pair for problem (5.2) over $[0, t_f]$ such that

- $u^*(\cdot)$ is right-continuous with left-hand limits,

- the constraint qualification holds for every tripple $\{t, x^*(t), u\}$, $t \in [0, t_f]$, $u \in \Omega(x^*(t), t)$;
- 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 $\mathbb{R}^n$,
- piecewise continuous multiplier functions $\mu(\cdot)$ and $\nu(\cdot)$ mapping $[0, t_f]$ into $\mathbb{R}^s$ and $\mathbb{R}^q$, respectively,
- a vector $\eta(\tau_i) \in \mathbb{R}^q$ for each point τ_i that is a discontinuity of $\lambda(\cdot)$,
- $\alpha \in \mathbb{R}^\ell$ and $\beta \in \mathbb{R}^{\ell'}$, $\gamma \in \mathbb{R}^q$, not all zero,

such that the following conditions hold almost everywhere:
Hamiltonian maximization

$$u^*(t) = \underset{u \in \Omega(x^*(t),t)}{\arg\max}\, H(x^*(t), u, \lambda_0, \lambda(t), t) \tag{5.9}$$

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

$$L_u^*[t] = H_u^*[t] + \mu g_u^*[t] = 0 \tag{5.10}$$
$$\dot{\lambda}(t) = -L_x^*[t] \tag{5.11}$$
$$\mu(t) g^*[t] = 0, \quad \mu(t) \geq 0 \tag{5.12}$$
$$\nu(t) h^*[t] = 0, \quad \nu(t) \geq 0 \tag{5.13}$$

and

$$dH^*[t]/dt = dL^*[t]/dt = L_t^*[t] := \partial L^*[t]/\partial t; \tag{5.14}$$

At the terminal time t_f, the transversality conditions hold:

$$\lambda(t_f^-) = \lambda_0 S_x^*[t_f] + \alpha a_x^*[t_f] + \beta b_x^*[t_f] + \gamma h_x^*[t_f] \tag{5.15}$$
$$\alpha a^*[t_f] = \gamma h^*[t_f] = 0, \quad \alpha \geq 0,\ \gamma \geq 0; \tag{5.16}$$

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

$$\lambda(\tau^-) = \lambda(\tau^+) + \eta(\tau) h_x^*[\tau] \tag{5.17}$$
$$H^*[\tau^-] = H^*\left[\tau^+\right] - \eta(\tau) h_t^*[\tau] \tag{5.18}$$
$$\eta(\tau) h^*[\tau] = 0, \quad \eta(\tau) \geq 0. \tag{5.19}$$

□

Remark 1
If the control signal appears linearly, formal proofs for the necessity of the conditions are available [Maurer, 1977]. □

Indirect Adjoining Approach

To form the Lagrangian, first or higher-degree derivatives of the pure-state constraints $h(x(t), t)$ in (5.2), are adjoined to the Hamiltonian. Here we present the theorem for first-order constraints. The Hamiltonian and the so called P-form Lagrangian are defined as [Hartl et al., 1995]

$$H(x, u, \lambda_0, \lambda, t) = \lambda_0 F(x, u, t) + \lambda f(x, u, t) \tag{5.20}$$

$$L(x, u, \lambda_0, \lambda, \mu, \nu, t) = H(x, u, \lambda_0, \lambda, t) + \mu g(x, u, t) + \nu h^1(x, u, t). \tag{5.21}$$

The control region is defined as:

$$\Omega(x, t) = \left\{u \in \mathbb{R}^m \mid g(x, u, t) \geq 0,\ h^1(x, u, t) \geq 0 \text{ if } h(x, t) = 0\right\} \tag{5.22}$$

Theorem 2—Informal Theorem 5.1 from [Hartl et al., 1995]
Let $\{x^*(\cdot), u^*(\cdot)\}$ be an optimal pair for problem (5.2) such that

- $x^*(\cdot)$ has only finitely many junction times,
- strong constraint qualification (5.4) and (5.5) hold,

then there exist

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

such that the following conditions hold almost everywhere:
Hamiltonian maximization

$$u^*(t) = \underset{u \in \Omega(x^*(t), t)}{\arg\max}\ H(x^*(t), u, \lambda_0, \lambda(t), t) \tag{5.23}$$

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

$$L_u^*[t] = 0 \tag{5.24}$$

$$\dot{\lambda}(t) = -L_x^*[t] \tag{5.25}$$

$$\mu(t) g^*[t] = 0, \quad \mu(t) \geq 0. \tag{5.26}$$

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

$$\nu(t)h^*[t] = 0,\ \nu(t) \geq 0,\ \dot{\nu}(t) \leq 0 \tag{5.27}$$

and

$$dH^*[t]/dt = dL^*[t]/dt = L_t^*[t], \tag{5.28}$$

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

$$\lambda(t_f^-) = \lambda_0 S_x^*[t_f] + \alpha a_x^*[t_f] + \beta b_x^*[t_f] + \gamma h_x^*[t_f] \tag{5.29}$$

$$\alpha a^*[t_f] = \gamma h^*[t_f] = 0, \quad \alpha \geq 0,\ \gamma \geq 0 \tag{5.30}$$

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

$$\lambda(\tau^-) = \lambda(\tau^+) + \eta(\tau)h_x^*[\tau] \tag{5.31}$$

$$H^*[\tau^-] = H^*\left[\tau^+\right] - \eta(\tau)h_t^*[\tau] \tag{5.32}$$

$$\eta(\tau)h^*[\tau] = 0, \quad \eta(\tau) \geq 0 \tag{5.33}$$

and for entry time τ_1

$$\eta(\tau_1) \geq \nu(\tau_1^+). \tag{5.34}$$

□

5.4 Solution Based on Direct Adjoining Approach

In this section, we apply the direct adjoining approach to our problem. Theorem 1 and 2 are under some technical assumptions equivalent and can either be applied to our problem. In general, the Hamiltonian maximization in the direct adjoining approach provides more information [Hartl et al., 1995], while the indirect approach might lead to simpler equations because of using h^1 instead of h.

First, the variables and parameters are identified with the general problem (5.2). Then, we evaluate the different conditions imposed on the solution by Theorem 1 one by one. From the Hamiltonian maximization, we find that the solution is divided into several regions. The regions are determined by active constraints. In each region, the conditions are evaluated and expressions for the control signal, states, costates, and multipliers are derived. Furthermore, the conditions at the boundaries and the continuity of the control signal and states are considered.

Depending on the problem, various scenarios may arise in which no constraint, one of the constraints or all become active. In this thesis, we present the scenario where both the constraints on the control signal and the state become active. The solution to other scenarios can be derived similarly.

Parameters, Hamiltonian and Lagrangian

By comparing (5.1) with (5.2), we identify

$$F\left(x(t), u(t), t\right) = -(x^T Q x + u^T R u) \tag{5.35}$$
$$S(t_f) = 0 \tag{5.36}$$
$$f_1\left(x(t), u(t), t\right) = x_2(t) \tag{5.37}$$
$$f_2\left(x(t), u(t), t\right) = u(t). \tag{5.38}$$

In addition, from (5.6) and (5.7)

$$H = -\lambda_0\left(q_1 x_1^2(t) + q_2 x_2^2(t) + r u^2(t)\right) + \lambda_1(t) x_2(t) + \lambda_2(t) u(t) \tag{5.39}$$
$$\begin{aligned} L = H &+ \mu_1(t)(1 - u(t)) + \mu_2(t)(1 + u(t)) + \nu_1(t)(c - x_2(t)) \\ &+ \nu_2(t)(c + x_2(t)). \end{aligned} \tag{5.40}$$

Constraints

Identifying the constraints with (5.2) results in

$$h(x(t)) = \begin{pmatrix} c - x_2(t) \\ c + x_2(t) \end{pmatrix} \tag{5.41}$$
$$g(u(t)) = \begin{pmatrix} 1 - u(t) \\ 1 + u(t) \end{pmatrix} \tag{5.42}$$
$$b(x(t)) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}. \tag{5.43}$$

The order of the constraints is found to be $p = 1$ by the following calculation

$$h^0 = \begin{pmatrix} c - x_2 \\ c + x_2 \end{pmatrix} \Rightarrow h_u^0 = 0 \tag{5.44}$$
$$h^1 = \begin{pmatrix} -\dot{x}_2(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} -u(t) \\ u(t) \end{pmatrix} \Rightarrow h_u^1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}. \tag{5.45}$$

The constraint qualification conditions (5.4)–(5.5) hold, since

$$\operatorname{rank} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = 2 = \ell' \tag{5.46}$$
$$\operatorname{rank} \begin{pmatrix} -1 & 1 - u & 0 \\ 1 & 0 & 1 + u \end{pmatrix} = 2 = s \quad \forall u \tag{5.47}$$

Hamiltonian Maximization

Without constraints, the Hamiltonian is maximized for

$$\hat{u}(t) = \frac{1}{2}\frac{1}{r}\frac{\lambda_2(t)}{\lambda_0}. \tag{5.48}$$

Considering the constraints, the *optimal solution* is

$$u^*(t) = \begin{cases} -1 & \text{if } \lambda_2(t)/\lambda_0 < -2r \\ \dfrac{1}{2r}\dfrac{\lambda_2(t)}{\lambda_0} & \text{if } -2r \leq \lambda_2(t)/\lambda_0 \leq 2r \\ 1 & \text{if } \lambda_2(t)/\lambda_0 > 2r \end{cases} \tag{5.49}$$

Given the scenario that both the constraints on the control signal and the state become active, we conclude the following time-dependent optimal solution u^* (see Fig. 5.1 for an illustration)

$$u^*(t) = \begin{cases} -1 & \text{for } t < \tau_1 \\ \dfrac{1}{2r}\dfrac{\lambda_2(t)}{\lambda_0} & \text{for } \tau_1 \leq t \leq \tau_2 \\ 0 & \text{for } \tau_2 < t < \tau_3 \\ \dfrac{1}{2r}\dfrac{\lambda_2(t)}{\lambda_0} & \text{for } \tau_3 \leq t \leq \tau_4 \\ 1 & \text{for } \tau_4 < t \end{cases} \tag{5.50}$$

where τ_i, $i \in \{1, \ldots, 4\}$ are junction times.

The Conditions

We evaluate conditions (5.10)–(5.13) here:

$$\begin{aligned} L_u^*[t] &= H_u^*[t] + \mu g_u^*[t] = 0 \\ &\Rightarrow -2ru(t) + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0 \end{aligned} \tag{5.51}$$

$$\begin{aligned} \dot{\lambda}(t) &= -L_x^*[t] \\ &\Rightarrow \begin{cases} \dot{\lambda}_1(t) = 2\lambda_0 q_1 x_1(t) \\ \dot{\lambda}_2(t) = 2\lambda_0 q_2 x_2(t) - \lambda_1(t) + \nu_1(t) - \nu_2(t) \end{cases} \end{aligned} \tag{5.52}$$

$$\begin{aligned} \mu(t) &= \begin{pmatrix} \mu_1(t) & \mu_2(t) \end{pmatrix}^T \geq 0,\ \mu(t)g^*[t] = 0 \\ &\Rightarrow (\mu_1(t) + \mu_2(t)) - (\mu_1(t) - \mu_2(t))u^*(t) = 0 \end{aligned} \tag{5.53}$$

$$\begin{aligned} \nu(t) &= \begin{pmatrix} \nu_1(t) & \nu_2(t) \end{pmatrix}^T \geq 0,\ \dot{\nu}(t) \leq 0,\ \nu(t)h^*(t) = 0 \\ &\Rightarrow (\nu_1(t) + \nu_2(t))c - (\nu_1(t) - \nu_2(t))x_2(t) = 0 \end{aligned} \tag{5.54}$$

Additionally, (5.14) results in

$$\begin{aligned}
&\frac{dH^*[t]}{dt} = \frac{dL^*}{dt} = L_t^*[t] \Rightarrow \\
&-2q_1x_1(t)x_2(t) - 2q_2x_2(t)u(t) - 2ru(t)\dot{u}(t) \\
&+\lambda_1(t)u(t) + \dot{\lambda}_1(t)x_2(t) + \lambda_2(t)\dot{u}(t) + \dot{\lambda}_2(t)u(t) = 0 \qquad (5.55)\\
&\dot{u}(t)(\mu_1(t) - \mu_2(t)) + (\dot{\mu}_1(t) - \dot{\mu}_2(t))u(t) \\
&\quad -(\dot{\mu}_1(t) + \dot{\mu}_2(t)) + (\nu_1(t) - \nu_2(t))u(t) \\
&\quad +(\dot{\nu}_1(t) - \dot{\nu}_2(t))x_2(t) - c(\dot{\nu}_1(t) + \dot{\nu}_2(t)) = 0 \qquad (5.56)
\end{aligned}$$

Transversality Conditions

The relations (5.15)–(5.16) are evaluated as follows:

$$\begin{aligned}
&\lambda(t_f^-) = \beta b_x^*[t_f] + \gamma h_x^*[t_f] \Rightarrow \\
&\begin{pmatrix}\lambda_1[1^-]\\ \lambda_2[1^-]\end{pmatrix} = I_{2\times 2}\begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix} + \begin{pmatrix}0 & -1\\ 0 & 1\end{pmatrix}^T\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \qquad (5.57)\\
&\gamma \geq 0, \gamma h[t_f] = 0 \Rightarrow (\gamma_1 + \gamma_2)c - (\gamma_1 - \gamma_2)x_2(t_f) = 0 \qquad (5.58)
\end{aligned}$$

Since $x_2(t_f) = 0$, we conclude that $\gamma_1 = \gamma_2 = 0$.

Regions

In view of (5.50), the solution is divided into different regions. We consider each case separately and using the knowledge of $u^*(t)$ in each specific region and (5.51)–(5.56), expressions for the states, the costates, and the multiplier functions are calculated. Hereafter, we set $\lambda_0 = 1$.

Case 1, $u^*(t) = -1,\ t < \tau_1$ Assuming that $x_0 > 0$

$$\begin{aligned}
x_1(t) &= -\frac{1}{2}t^2 + v_0t + x_0 \\
x_2(t) &= -t + v_0
\end{aligned} \qquad (5.59)$$

From (5.51), it follows that

$$2r + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0. \qquad (5.60)$$

From (5.53), it follows that

$$\mu_1(t) = 0,\ \mu_2(t) \geq 0. \qquad (5.61)$$

From (5.54), it follows that

$$\nu_1(t) = \frac{t-c}{t+c}\nu_2(t). \qquad (5.62)$$

From (5.56), it follows that

$$2\dot{\mu}_1 + (t+c)\dot{\nu}_1 - (t-c)\dot{\nu}_2 + (\nu_1 - \nu_2) = 0. \tag{5.63}$$

From (5.52), it follows that

$$\begin{aligned} \dot{\lambda}_1(t) &= -q_1 t^2 + +2q_1 v_0 t + 2q_1 x_0 \\ \dot{\lambda}_2(t) &= -2q_2 t + 2q_2 v_0 - \lambda_1(t) + \nu_1(t) - \nu_2(t), \end{aligned} \tag{5.64}$$

and combined with (5.55), we conclude that

$$\nu_1(t) = \nu_2(t). \tag{5.65}$$

From (5.62) and (5.65), it follows that

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

From (5.64), it follows that

$$\begin{aligned} \lambda_1(t) &= -\frac{1}{3}q_1 t^3 + q_1 v_0 t^2 + 2q_1 x_0 t + K_1 \\ \lambda_2(t) &= \frac{1}{12}q_1 t^4 - \frac{1}{3}q_1 v_0 t^3 - (q_1 x_0 + q_2)t^2 + (2q_2 v0 - K_1)t + K_2, \end{aligned} \tag{5.66}$$

where K_1 and K_2 are appropriate constants of integration.

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

$$\begin{aligned} x_1(t) &= -ct + K_3 \\ x_2(t) &= -c \end{aligned} \tag{5.67}$$

From (5.51), it follows that

$$\lambda_2(t) - \mu_1(t) + \mu_2(t) = 0. \tag{5.68}$$

From (5.53), it follows that

$$\mu_1(t) = \mu_2(t) = 0. \tag{5.69}$$

From (5.54), it follows that

$$\nu_1(t) = 0,\ \nu_2(t) \geq 0. \tag{5.70}$$

From (5.52), it follows that

$$\begin{aligned} \dot{\lambda}_1(t) &= -2q_1 ct + 2q_1 K_3 \\ \dot{\lambda}_2(t) &= -\lambda_1(t) - 2q_2 c - \nu_2(t). \end{aligned} \tag{5.71}$$

From (5.68) and (5.69), we conclude that $\lambda_2(t) = 0$. Considering this, (5.71) simplifies to

$$\nu_2(t) = -\lambda_1(t) - 2q_2 c \tag{5.72}$$

$$\lambda_1(t) = -q_1 ct^2 + 2q_1 K_3 t + K_4. \tag{5.73}$$

Case 3, $u^*(t) = 1,\ t > \tau_4$ By integrating the control signal, we obtain

$$\begin{aligned} x_1(t) &= \frac{1}{2}t^2 + K_5 t + K_6 \\ x_2(t) &= t + K_5. \end{aligned} \tag{5.74}$$

From (5.51), it follows that

$$-2r + \lambda_2(t) - \mu_1(t) + \mu_2(t) = 0. \tag{5.75}$$

From (5.53), it follows that

$$\mu_1(t) \geq 0,\ \mu_2(t) = 0. \tag{5.76}$$

From (5.54), it follows that

$$(c - t)\nu_1(t) + (c + t)\nu_2(t) - (\nu_1(t) - \nu_2(t))K_5 = 0. \tag{5.77}$$

From (5.56), it follows that

$$\nu_1(t) - \nu_2(t) + (\dot{\nu}_1(t) - \dot{\nu}_2(t))(t - c + K_5) = 0. \tag{5.78}$$

From (5.52), it follows that

$$\begin{aligned} \dot{\lambda}_1(t) &= 2q_1(\frac{1}{2}t^2 + K_5 t + K_6) \\ \dot{\lambda}_2(t) &= 2q_2(t + K_5) - \lambda_1(t) + \nu_1(t) - \nu_2(t), \end{aligned} \tag{5.79}$$

and combined with (5.55), we conclude that

$$\nu_1(t) = \nu_2(t). \tag{5.80}$$

From (5.77) and (5.80), it follows that $\nu_1(t) = \nu_2(t) = 0$. From (5.79), it follows that

$$\begin{aligned} \lambda_1(t) &= \frac{1}{3}q_1 t^3 + q_1 K_5 t^2 + 2q_1 K_6 t + K_7 \\ \lambda_2(t) &= -\frac{1}{12}q_1 t^4 - \frac{1}{3}q_1 K_5 t^3 + (q_2 - q_1 K_6)t^2 \\ &\quad + (2q_2 K_5 - K_7)t + K_8 \end{aligned} \tag{5.81}$$

Case 4, $u^*(t) = \frac{1}{2r}\lambda_2(t),\ \tau_1 < t < \tau_2$ ***or*** $\tau_3 < t < \tau_4$ From (5.51), it follows that

$$u(t) = \frac{\lambda_2(t) - \mu_1(t) + \mu_2(t)}{2r} \Rightarrow \mu_1(t) = \mu_2(t). \tag{5.82}$$

From (5.53), it follows that

$$\mu_1(t) + \mu_2(t) = 0. \tag{5.83}$$

Therefore, $\mu_1(t) = \mu_2(t) = 0$.
From (5.55), it follows that

$$\left(-2q_2x_2(t) + \lambda_1(t) + \dot{\lambda}_2(t)\right)\lambda_2(t) = 0. \tag{5.84}$$

In this region, $\lambda_2(t)$ cannot be identically equal to zero. Thus,

$$-2q_2x_2(t) + \lambda_1(t) + \dot{\lambda}_2(t) = 0. \tag{5.85}$$

Comparing with (5.52) and using (5.54), we conclude that $\nu_1(t) = \nu_2(t) = 0$. Using (5.85), the system equations (5.1b), and $u^*(t) = \frac{1}{2r}\lambda_2(t)$, we conclude that

$$\lambda_2^{(4)}(t) = \frac{1}{r}(q_2\ddot{\lambda}_2(t) - q_1\lambda_2(t)). \tag{5.86}$$

From (5.86), it follows that

$$\lambda_2(t) = C_1e^{-\sigma_1 t} + C_2e^{\sigma_1 t} + C_3e^{-\sigma_2 t} + C_4e^{\sigma_2 t}, \tag{5.87}$$

where

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

and C_i are appropriate constants of integration.

Let us define

$$\begin{aligned}
A(C,t) &:= \frac{C_1}{\sigma_1^2}e^{-\sigma_1 t} + \frac{C_2}{\sigma_1^2}e^{\sigma_1 t} + \frac{C_3}{\sigma_2^2}e^{-\sigma_2 t} + \frac{C_4}{\sigma_2^2}e^{\sigma_2 t}\\
A'(C,t) &= -\frac{C_1}{\sigma_1}e^{-\sigma_1 t} + \frac{C_2}{\sigma_1}e^{\sigma_1 t} - \frac{C_3}{\sigma_2}e^{-\sigma_2 t} + \frac{C_4}{\sigma_2}e^{\sigma_2 t}\\
A''(C,t) &= C_1e^{-\sigma_1 t} + C_2e^{\sigma_1 t} + C_3e^{-\sigma_2 t} + C_4e^{\sigma_2 t}\\
B(C,t) &:= \frac{\sigma_2^2}{\sigma_1}\left(C_1e^{-\sigma_1 t} - C_2e^{\sigma_1 t}\right) + \frac{\sigma_1^2}{\sigma_2}\left(C_3e^{-\sigma_2 t} - C_4e^{\sigma_2 t}\right).
\end{aligned}$$

Thus, we can write

$$u(t) = \frac{1}{2r}A''(C,t). \tag{5.88}$$

From the system equations (5.1b), it follows that

$$\dot{x}_2(t) = u(t) \qquad \Rightarrow x_2(t) = \frac{1}{2r}A'(C,t) + \kappa_1 \tag{5.89}$$

$$\dot{x}_1(t) = x_2(t) \qquad \Rightarrow x_1(t) = \frac{1}{2r}A(C,t) + \kappa_1 t + \kappa_2, \tag{5.90}$$

where κ_1 and κ_2 are appropriate constants of integration. Considering (5.85), it follows that

$$\lambda_1(t) = -B(C,t) + 2q_2\kappa_1, \tag{5.91}$$

where we have also used the fact

$$\sigma_1 - \frac{q_2}{\sigma_1 r} = -\frac{\sigma_2^2}{\sigma_1},$$
$$\sigma_2 - \frac{q_2}{\sigma_2 r} = -\frac{\sigma_1^2}{\sigma_2}.$$

Moreover, substituting (5.90) and (5.91) into (5.52) results in

$$\left(q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2\right)\left(C_1 e^{-\sigma_1 t} + C_2 e^{\sigma_1 t}\right) + \left(q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2\right)\left(C_3 e^{-\sigma_2 t} + C_4 e^{\sigma_2 t}\right) = 2rq_1\left(\kappa_1 t + \kappa_2\right). \tag{5.92}$$

From the definition, we have

$$q_2 - \frac{q_1}{\sigma_2^2} - r\sigma_2^2 = 0,$$
$$q_2 - \frac{q_1}{\sigma_1^2} - r\sigma_1^2 = 0.$$

Now, using the assumption $q_1 \neq 0$ and $r \neq 0$ we conclude

$$\kappa_1 t + \kappa_2 = 0. \tag{5.93}$$

The equations derived in this part apply to two regions. For the first interval, $\tau_1 < t < \tau_2$, we will use constants C_i, $i \in \{1, \ldots, 4\}$ and for the second interval, $\tau_3 < t < \tau_4$, constants D_i. Note that since it is assumed that the junction times τ_i are distinct, (5.93) must hold in more than one point. Accordingly,

$$\kappa_1 = \kappa_2 = 0. \tag{5.94}$$

Initial and Final Conditions:

The initial condition on x has already been used in (5.59). Without loss of generality, we normalize the final time to 1. Thus, the final conditions for x are

$$x_1(1) = 0, \;\; x_2(1) = 0. \tag{5.95}$$

Therefore, from (5.74), when $u^* = 1$ it follows that

$$K_5 = -1, \;\; K_6 = \frac{1}{2}. \tag{5.96}$$

Continuity of u^*:

The Hamiltonian (5.39) is regular, i.e., the maximization of H w.r.t. u is unique. Therefore, u^* is continuous everywhere including the points on the boundary according to Proposition 4.3 in [Hartl et al., 1995]. We evaluate the control signal at junction times from below and above and equate the expressions:

$$u^*(\tau_1^+) = -1,\ u^*(\tau_2^-) = 0,\ u^*(\tau_3^+) = 0,\ u^*(\tau_4^-) = 1 \Rightarrow$$

$$A''(C, \tau_1) + 2r = 0 \tag{5.97}$$
$$A''(C, \tau_2) = 0 \tag{5.98}$$
$$A''(D, \tau_3) = 0 \tag{5.99}$$
$$A''(D, \tau_4) - 2r = 0 \tag{5.100}$$

Continuity of x^*:

By integrating u^*, we arrive at the states. Hence, x^* is continuous too. The values of the states at junction times from below and above are equated:

$$x_1^*(\tau_i^-) = x_1^*(\tau_i^+),\ x_2^*(\tau_i^-) = x_2^*(\tau_i^+),\ i \in \{1, \ldots, 4\} \Rightarrow$$

$$A(C, \tau_1) + r\left(\tau_1^2 - v_0\tau_1 - 2x_0\right) = 0 \tag{5.101}$$
$$A'(C, \tau_1) + 2r(\tau_1 - v_0) = 0 \tag{5.102}$$
$$A(C, \tau_2) + 2r\left(c\tau_2 - K_3\right) = 0 \tag{5.103}$$
$$A'(C, \tau_2) + 2rc = 0 \tag{5.104}$$
$$A(D, \tau_3) + 2r\left(c\tau_3 - K_3\right) = 0 \tag{5.105}$$
$$A'(D, \tau_3) + 2rc = 0 \tag{5.106}$$
$$A(D, \tau_4) - r\left(\tau_4 - 1\right)^2 = 0 \tag{5.107}$$
$$A'(D, \tau_4) - 2r\left(\tau_4 - 1\right) = 0 \tag{5.108}$$

Continuity of λ:

The continuity of the adjoint function λ at junction times is guaranteed since our problem fulfills the conditions of Proposition 4.2 in [Hartl et al., 1995]. To show this, from the dimensions of the constraint functions $h(\cdot)$

and $g(\cdot)$, we have $s = q = 2$. The control signal u^* is continuous and

$$\begin{aligned}
&\operatorname{rank}\begin{pmatrix} \partial g^*[\tau]/\partial u & \operatorname{diag}(g^*[\tau]) & 0 \\ \partial h^{1*}[\tau]/\partial u & 0 & \operatorname{diag}(h^*[\tau]) \end{pmatrix} \\
&= \operatorname{rank}\begin{pmatrix} -1 & 1-u(\tau) & 0 & 0 & 0 \\ 1 & 0 & 1+u(\tau) & 0 & 0 \\ -1 & 0 & 0 & c-x_2(\tau) & 0 \\ 1 & 0 & 0 & 0 & c+x_2(\tau) \end{pmatrix} \\
&= 4 = s+q. \qquad (5.109)
\end{aligned}$$

Now, we evaluate the costates at junction times from below and above and equate the expressions:

$$\lambda_1(\tau_i^-) = \lambda_1(\tau_i^+),\ \lambda_2(\tau_i^-) = \lambda_2(\tau_i^+),\ i \in \{1,\dots,4\} \Rightarrow$$

$$B(C,\tau_1) - \frac{1}{3}q_1\tau_1^3 + q_1 v_0\tau_1^2 + 2q_1x_0\tau_1 + K_1 = 0 \qquad (5.110)$$

$$\frac{1}{12}q_1\tau_1^4 - \frac{1}{3}q_1v_0\tau_1^3 - (q_1x_0+q_2)\tau_1^2 + (2q_2v0 - K_1)\tau_1 + K_2 + 2r = 0 \qquad (5.111)$$

$$B(C,\tau_2) - q_1c\tau_2^2 + 2q_1K_3\tau_2 + K_4 = 0 \qquad (5.112)$$

$$B(D,\tau_3) - q_1c\tau_3^2 + 2q_1K_3\tau_3 + K_4 = 0 \qquad (5.113)$$

$$B(D,\tau_4) + \frac{1}{3}q_1\tau_4^3 - q_1\tau_4^2 + q_1\tau_4 + K_7 = 0 \qquad (5.114)$$

$$\frac{1}{12}q_1\tau_4^4 + \frac{1}{3}q_1\tau_4^3 + (\frac{1}{2}q_1 - q_2)\tau_4^2 + (2q_2+K_7)\tau_4 - K_8 + 2r = 0 \qquad (5.115)$$

5.5 Results

The conditions of Theorem 1 lead to 24 unknowns (K_i, $i \in \{1,\dots,8\}$, κ_i, $i \in \{1,\dots,2\}$ for two regions, C_i, D_i, $i \in \{1,\dots,4\}$, and τ_i, $i \in \{1,\dots,4\}$) and 24 equations, (5.94) and (5.96)–(5.115). By solving these equations, the junction times and the integration constants are determined, and hence the solution to the optimal control problem. There are six trivial equations corresponding to (5.94) and (5.96). The rest are nonlinear in τ_i, but linear in the other parameters. Note that we have only presented the result of the scenario where $0 \le \tau_1 < \tau_2 < \tau_3 < \tau_4 \le 1$. If there is no solution to the equations, other scenarios where no or only some of the state/input constraints become active, must be considered. The problem is infeasible if there is no solution to any of these scenarios.

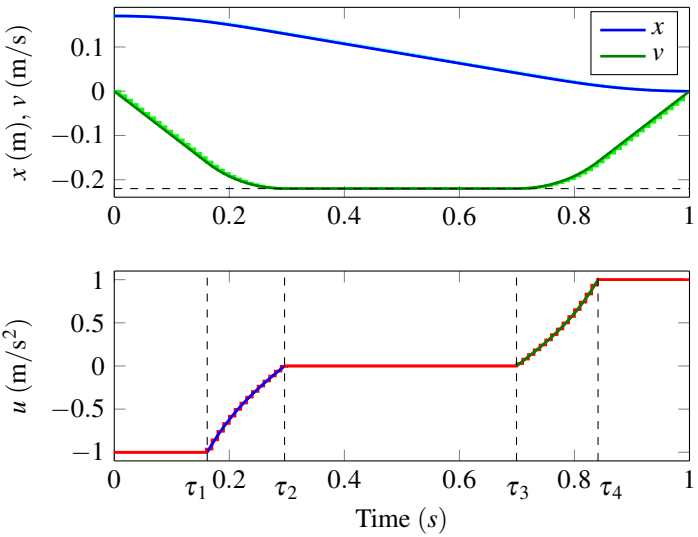


Figure 5.1 Comparison of the result of the interior-point method with 100 discretization points and the analytic solution to (5.1) with the parameters given in (5.116). The smooth curves in the upper plot as well as the blue and green pieces in the lower plot belong to the analytic solution.

Example

We report the results for the following example:

$$\begin{aligned} R = r = 0.1, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix} \\ c = 0.22, \quad x_0 = 0.17, \quad v_0 = 0, \quad t_f = 1. \end{aligned} \tag{5.116}$$

The numerical solution obtained by the interior-point method and the analytic solution are compared in Fig. 5.1. We ran ten experiments starting from a random initial guess for the solution of the nonlinear equations. The `fsolve` function in Matlab was able to find a solution on average in 0.0825 s on an Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz with 32GB RAM running Fedora 20. The same problem was solved using the 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 with the overheads included 53.753 s). The cost obtained by the analytical approach was $385.352h$, which was confirmed by the numerical results. The effect on the control signal by varying Q and R is shown in Fig. 5.2.

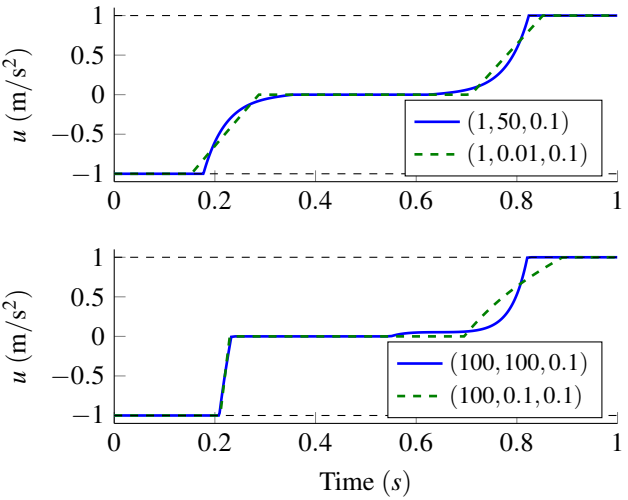


Figure 5.2 An illustrative example for the effect of various cost functions (q_1, q_2, r). The other parameters for the problem are according to (5.116).

5.6 Discussion

Since the Hamiltonian H in (5.39) is concave, we 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 use a numerical approach to find the constants, this approach is independent of the number of discretization points. In other words, there is an analytic expression for the solution that is parametrized by the unknowns.

The minimum-time solutions proposed in [Macfarlane and Croft, 2003; Haschke et al., 2008; Kröger and Wahl, 2010] result in "bang-bang" solutions. By including constraints on jerk or higher order derivatives of the position, it is possible to make the transitions smoother. However, in contrast to our approach no explicit cost on the states and control signal is taken into account. Moreover, the time synchronization between different degrees of freedom requires a special solution when the minimum-time problem is individually solved for each degree of freedom [Kröger and Wahl, 2010] while this synchronization comes for free in fixed-time problems. Compared to the piece-wise polynomial approaches [Paul, 1972; Lin et al., 1983; Taylor, 1979], our solution takes into account a quadratic cost function as well as the constraints on velocity and acceleration.

In many industrial manipulators, individual joint velocities and positions are used as reference inputs for each DOF. The required torque values

can also be calculated if the inverse dynamics is available. Thus, our approach is applicable for a large class of robots. However, the assumption of decoupled DOF does not hold in general when there are obstacles. In such cases, via-points can be included or a multiple DOF problem with the obstacles represented as state constraints needs to be solved. Finding an analytic solution in the latter approach is challenging.

5.7 Conclusion

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 analytic solution is that its computation time is independent of its resolution, while a numerical approach based on discretization might fail for a high sampling rate. Since the solution can be computed efficiently, there is an opportunity for online trajectory generation. Compared to time-optimal solutions, the fixed-time problem solved here allows for an application-specific cost function. Additionally, the synchronization between several degrees of freedom is for free since all movements must follow the same given fixed-time. For more complex dynamics, using the maximum principle to derive analytic equations seems impractical. However, given the benefits of an analytic solution, special software can be developed to assist with this approach.

6

Real-Time Trajectory Generation using Model Predictive Control

6.1 Introduction

A desired characteristic of the motion planning in uncertain environments is the ability to react to sensor inputs [Kröger, 2010]. In this chapter, we provide experimental results for a ball-catching robot [Linderoth, 2013], where the estimates of the contact position of the ball are delivered online from a vision system. As more vision data become available, the position can be estimated with lower uncertainty. Thus, a new trajectory needs to be computed when new estimates are delivered. Our approach to trajectory generation is based on the receding-horizon principle offered by Model Predictive Control (MPC) [Mayne et al., 2000; Maciejowski, 1999]. By employing a final-state constraint in the MPC formulation, it is possible to solve fixed-time motion-planning problems where analytic solutions are not available. In contrast to the typical application of MPC for reference-tracking problems, we adopt a trajectory-generation perspective. In our experiments, we use the aforementioned strategy to compute reference values (joint position and velocity reference values) for an underlying motion-control system.

Previous Research

Methods for trajectory generation based on polynomial functions of time were already developed in the 1970s [Paul, 1979; Taylor, 1979]. Time-optimal path tracking for industrial manipulators was suggested in the 1980s [Bobrow et al., 1985; Shin and McKay, 1985]. An overview of trajectory-generation methods for robots is provided in [Kröger, 2010]. There are many methods for online trajectory generation with the objective of time-optimality based on analytic expressions, parametrized in the initial and

final states of the desired motion and certain constraints [Castain and Paul, 1984; Macfarlane and Croft, 2003; Haschke et al., 2008; Kröger and Wahl, 2010]. The difference between these methods is in their generality with respect to the initial and final states and constraints. A sub-optimal solution to the fixed-time trajectory planning for robot manipulators was proposed in [Dulęba, 1997]. In another approach, the fixed-time optimal solution in the space of parametrized trajectories was found [Yang et al., 2012].

Trajectory generation based on optimization using a kinematic model has been proposed in [Bauml et al., 2010] for ball-catching robotic systems. Methods for catching flying objects with robots were also considered in [Linderoth et al., 2010; Lampariello et al., 2011]. In the latter, a dynamic model was employed and machine-learning algorithms were used in the online execution. MPC has been proposed for control and path tracking for mobile robots; see, e.g., [Howard et al., 2010; Kanjanawanishkul and Zell, 2009; Klančar and Škrjanc, 2007]. A two-layer architecture was proposed in [Norén, 2013] with a generic optimal-control formulation at the high-level and MPC at the low-level for tracking the solution generated by the high-level planner. Nonlinear optimization techniques to solve finite-horizon MPC problems with complex dynamical models have been proposed and used for controlling humanoids [Erez et al., 2013] and for hand manipulation [Kumar et al., 2014].

Contributions

The major contribution of this chapter is a method for fast online motion planning, given an initial state of a robot and a desired final state at a given time. The approach does not rely on any predefined trajectory or path. To this purpose, we have applied the MPC framework to fixed-time point-to-point trajectory generation. In our approach, it is possible to solve the trajectory-planning problem online with application-specific cost functions and a broad class of constraints on inputs and states. The algorithm relies on convex optimization, such that real-time computational constraints for modern robotic control systems can be satisfied. Various choices of models and constraints are illustrated with simulation results. Further contributions of this chapter are a complete implementation of one instance of the approach to motion planning and a subsequent experimental evaluation in a challenging scenario of ball catching.

Outline

The structure of this chapter is as follows. A problem formulation is provided in Sec. 6.2, where a generic motion-planning problem using MPC is defined. Following this section, the methods that lead to a concrete implementation of a real-time trajectory generator are presented in Sec. 6.3.

Extensive simulation results illustrating different aspects of the approach are presented in Sec. 6.4. In Sec. 6.5, the experimental setup is detailed and the implementation aspects are discussed. The results from evaluation experiments as well as the execution of the proposed motion-planning method on a robot in a demanding task of ball-catching are provided in Sec. 6.6. Finally, the results of this chapter and the methods are discussed in Sec. 6.7 and conclusions are drawn in Sec. 6.8.

6.2 Problem Formulation

In many robotic applications, it is desirable to attain 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 unspecified during certain intervals in time, we need to do planning between the points, i.e., transferring the robot from an initial state to the next state in the given time. In either case, the motion of the robot must fulfill predefined constraints. Moreover, we wish to specify a desired behavior, which in this chapter is implicitly characterized by the optimum of an objective function.

We develop an approach based on model prediction and receding horizon to solve this type of problems. A central notion in MPC is the use of 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 states and outputs. The optimization is usually carried out at each sample. Then, only the first sample in the computed control inputs over the prediction horizon is applied [Maciejowski, 1999; Mayne et al., 2000]. Figure 6.1 shows a schematic of trajectory planning using MPC.

In the MPC framework, it is required to define a model of the system, an objective function, state and input constraints, and a prediction horizon. We consider linear models on the form

$$x(k+1) = A_d x(k) + B_d u(k), \tag{6.1}$$

$$y(k) = C_d x(k) + D_d u(k), \tag{6.2}$$

$$z(k) = \tilde{C} x(k) + \tilde{D} u(k). \tag{6.3}$$

Here, $u(k)$, $x(k)$, $y(k)$, and $z(k)$ denote the control signal, the states, the measured outputs and the controlled variables, respectively [Maciejowski, 1999]. Denoting the the prediction horizon by N, let us define

$$\mathcal{U}^T = \left[u^T(k), u^T(k+1), \ldots, u^T(k+N-1)\right]. \tag{6.4}$$

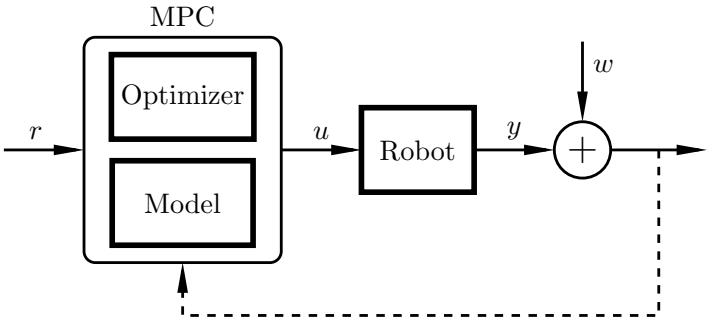


Figure 6.1 Schematic description of trajectory generation using MPC; r is the reference signal, u is the control signal, y is the output signal, and w is the output disturbance. The feedback from the output to the MPC is not present if an open-loop strategy is used.

We consider a quadratic cost at time k as

$$V_k(\mathcal{U}) = \sum_{i=k+1}^{k+N} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)}, \tag{6.5}$$

where the norm is defined as $\|a\|^2_W = a^T W a$ for a positive semidefinite weight matrix W and the reference signal is denoted by $r(i)$. Additionally, Q and R are time-dependent weight matrices. Linear constraints on the control and the states are assumed as follows

$$\begin{aligned} F\mathcal{U}(k) &\leq f, \\ G\mathcal{Z}(k) &\leq g, \end{aligned} \tag{6.6}$$

where F and G are matrices whose dimensions are determined by the system dynamics (6.1)–(6.3), the number of constraints, and the prediction horizon, and

$$\mathcal{Z}^T = \left[z^T(k+1), z^T(k+2), \ldots, z^T(k+N)\right]. \tag{6.7}$$

Given the initial state $x(k)$ of the system, the following optimization problem is solved at each sample k [Mayne et al., 2000]:

$$\begin{aligned} \underset{\mathcal{U}}{\text{minimize}} \quad & V_k(\mathcal{U}) \\ \text{subject to} \quad & (6.1),\ (6.3),\ (6.6) \\ & z(k+N) = r_f(k), \quad \text{(optional)} \end{aligned} \tag{6.8}$$

where $r_f(k)$ is the constraint on the controlled variable at the final time in the prediction horizon.

It is desirable to investigate how the MPC framework is applicable to trajectory generation for point-to-point problems with a fixed final time. Further, we wish to find a set of assumptions and methods that allow for real-time solutions.

6.3 Methods

In this section, we specialize the general linear MPC framework in Sec. 6.2 to point-to-point trajectory generation. We also provide examples of models and constraints, which are later used in our simulations and experiments on a robotic setup.

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, even if the problem is feasible. In case of multiple local optima, special care must be taken to avoid jumping between different solutions, which may lead to divergence. On the other hand, at the cost of limiting the scope of the problems, choosing convex cost functions and convex and compact constraint sets, we are assured that if a solution exists, it has to be globally optimal [Boyd and Vandenberghe, 2004].

Point-to-Point Planning

In the general objective function (6.5), the desired controlled states do not need to be specified at every sample. This is an important feature for point-to-point trajectory generation. To clarify this point, assume that

$$\Psi(k) \subset \{k+1, \ldots, k+N\}, \tag{6.9}$$

denotes the set of indices where there are desired values for the controlled states z. By assuming that $r(i) = 0$ when $i \notin \Psi(k)$, we can rewrite (6.5) as

$$V_k(\mathcal{U}) = \sum_{i \in \Psi(k)} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k+1}^{k+N} \|z(i)\|^2_{\bar{Q}(i)} + \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)}, \tag{6.10}$$

where $\bar{Q}(i) = Q(i)$ if $i \notin \Psi(k)$ and $\bar{Q}(i) = 0$ if $i \in \Psi(k)$. For reference-tracking problems, the first and the last terms are usually important. In the point-to-point trajectory generation, provided that explicit constraints for the desired controlled states are defined, the first term can be ignored. Nevertheless, if soft constraints are preferred, then all of the terms might be used. In this case, the benefit is immediate if the optimization problem with explicit constraints on the controlled variables $z(i)$, $i \in \Psi(k)$, is infeasible.

Since the optimization in general runs at every sampling instant, it is possible to account for the updates in the target or deviations from the planned trajectories of the robot. If good trajectory-tracking control of the robot could be assumed, the feedback loop can be closed around the model too, providing an open-loop control strategy (*cf.* Fig. 6.1).

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

Interpolation of Trajectories

It is required to have a valid trajectory in every moment because of the real-time execution of the motion planning. This is obviously the case also when the optimization algorithm fails, for example as a result of an infeasible problem specification. Thus, we consider piece-wise trajectories. The previous trajectory is valid until the new one is ready to be switched to. The robot is stopped in its final position if no other piece of trajectory for the current time exists.

Since MPC is formulated for discrete-time systems, we need to interpolate between the computed trajectory points. The interpolation makes it possible to have a lower sampling rate for the optimization in the trajectory generation than for the controlled system. Assuming that T is a vector of increasing time instants and $\mathcal{U}$ is a matrix of the corresponding inputs,

$$T = \begin{bmatrix} t_1, & t_2, & \dots, & t_n \end{bmatrix}, \tag{6.11}$$

$$\mathcal{U}^T = \begin{bmatrix} u^T(1), & u^T(2), & \dots, & u^T(n) \end{bmatrix}, \tag{6.12}$$

we consider a linear interpolation such that

$$u(t) = u(k) + \frac{u(k+1) - u(k)}{t_{k+1} - t_k}(t - t_k), \qquad t_k \leq t < t_{k+1}. \tag{6.13}$$

Discretization

In the MPC framework, the dynamics of a system is described by difference equations. Assuming the following continuous-time linear model

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{6.14}$$

the discrete-time system obtained using the predictive first-order-hold sampling method [Åström and Wittenmark, 2011] with a sampling period h is

$$\begin{aligned} x(k+1) &= \Phi x(k) + \frac{1}{h}\Gamma_1 u(k+1) + (\Gamma - \frac{1}{h}\Gamma_1)u(k), \\ y(k) &= Cx(k), \end{aligned} \tag{6.15}$$

where

$$\begin{bmatrix}\Phi & \Gamma & \Gamma_1\end{bmatrix} = \begin{bmatrix}I & 0 & 0\end{bmatrix} \exp\left(\begin{bmatrix}A & B & 0\\ 0 & 0 & I\\ 0 & 0 & 0\end{bmatrix} h\right). \tag{6.16}$$

Note that (6.15) can be rewritten in the standard form by changing to a new state variable ζ according to

$$\zeta(k+1) = \Phi\zeta(k) + \left(\Gamma + \frac{1}{h}(\Phi - I)\Gamma_1\right)u(k), \tag{6.17}$$

$$y(k) = C\zeta(k) + \frac{1}{h}C\Gamma_1 u(k). \tag{6.18}$$

This choice of discretization method is motivated by the linear interpolation of the trajectories as defined in (6.13).

Models

The models for the trajectory planning are developed to approximate the relation between the motion and the actuation. The motion can in principle 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 Cartesian coordinates in operational space rather than joint coordinates. Furthermore, the motion can be specified in terms of position, velocity, or higher-order time derivatives of the position. A control signal may be a dynamic quantity such as force or torque. Typically, the dynamic equations of robotic systems are highly non-linear. Therefore, without using any form of approximation, this fact limits the usage of a linear MPC framework.

Here, we provide two examples of models for which the linear MPC approach is applicable:

Chains of Integrators Assuming a good tracking performance for a robot, a simple kinematic robot model using only the kinematic variables can be constructed by multiple decoupled chains of integrators, see, e.g., [LaValle, 2006]. Since robots have various structures, we choose the generalized coordinates q for the representation of the states. In this way, q may

represent either the joint values or the Cartesian coordinates depending on the application. We choose the state vector as

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

where d is the number of degrees of freedom.

The continuous-time model (6.14), where each DOF is assumed to be a triple integrator, is specified by the matrices

$$A = \text{blkdiag}\left([\tilde{A}, \ldots, \tilde{A}]\right),\ B = \text{blkdiag}\left([\tilde{B}, \ldots, \tilde{B}]\right),$$

and $C = I_{3d}$, where I_{3d} denotes an identity matrix in $\mathbb{R}^{3d\times 3d}$, blkdiag$(\cdot)$ forms a block diagonal matrix from the given list of matrices, $A \in \mathbb{R}^{3d\times 3d}$, $B \in \mathbb{R}^{3d\times d}$, and

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

The matrices for the discretized model concerning the controlled states can be calculated as below

$$\Phi = \text{blkdiag}\left([\tilde{\Phi}, \ldots, \tilde{\Phi}]\right),\ \Gamma_1 = \text{blkdiag}\left([\tilde{\Gamma}_1, \ldots, \tilde{\Gamma}_1]\right),$$
$$\Gamma = \text{blkdiag}\left([\tilde{\Gamma}, \ldots, \tilde{\Gamma}]\right),$$

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

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

Linearized Model As another example, let us assume that we have identified a linear model from the joint reference velocities to the joint velocities around a certain working point in the task space of the robot. As long as the range of the movements is small, this model can be employed for the purpose of trajectory generation. The benefits from such models are immediate when the mechanism has certain resonant frequencies, resulting from inherent flexibilities in the mechanical parts or coupling between the DOF. Assume that for each pair of an input and a DOF, we have identified a linear model such that

$$v_i = H_{i,j} v_j^r,$$

where v^r and v denote the velocity reference and the velocity along the DOF, respectively. We can construct a complete model from the reference acceleration a_r to actual positions and velocities such that

$$x = \tilde{H} a^r,$$

where $x = \left[q_1, \dot{q}_1, \ldots, q_d, \dot{q}_d\right]^T$, $a_r = \left[a_1^r, a_2^r, \ldots, a_d^r\right]^T$, and

$$\tilde{H} = \begin{pmatrix} \frac{1}{s^2}H_{1,1} & \frac{1}{s^2}H_{1,2} & \cdots & \frac{1}{s^2}H_{1,d} \\ \frac{1}{s}H_{1,1} & \frac{1}{s}H_{1,2} & \cdots & \frac{1}{s}H_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{s^2}H_{d,1} & \frac{1}{s^2}H_{d,2} & \cdots & \frac{1}{s^2}H_{d,d} \\ \frac{1}{s}H_{d,1} & \frac{1}{s}H_{d,2} & \cdots & \frac{1}{s}H_{d,d} \end{pmatrix}. \tag{6.19}$$

Finally, we convert $\tilde{H}$ to a state-space model and compute its discrete-time counterpart according to Sec. 6.3.

Cost Function

When choosing a cost function in the optimization, physical interpretations are often beneficial but in general the cost does not need to have a physical meaning. For trajectory generation, we might consider punishing high values of the acceleration or the velocity. We might also choose the cost $V_k(\mathcal{U})$ to model for instance the distance traveled by a robot or the mechanical energy. Let ΔX denote the changes in the vector of positions, then the length is approximately proportional to $\sum \|\Delta X\|$. Also, introducing a cost on the form $\sum \dot{X}^2$ can lead to keeping the kinetic energy low. The MPC problem for point-to-point trajectory generation can be formulated by assuming the objective function (6.10) with $\Psi(k) = \emptyset$ and $\bar{Q}(k+N) = 0$, i.e., ignoring the first term and the cost on the final state.

For the purpose of illustrating certain additional features of possible cost functions, the remaining part of this subsection elaborates on the basic formulation outlined in the previous paragraph. An abrupt brake of the robot motion often leads to the excitation of vibration modes. To avoid this problem, we can shape the control signal such that it smoothly goes to zero toward the end of the motion. For example, an exponential weighting function for the control signal can be applied from sample k'

onward according to

$$R(i) = \begin{cases} c^{\left(\frac{i-k'+1}{N+k-k'}\right)} R, & \text{if } k' \leq i < N+k \\ R, & \text{if } k \leq i < k' \end{cases} \tag{6.20}$$

where c denotes the maximum weight at the end of the prediction horizon, and R is a constant matrix. Similarly, the deviation of the states from r_f can be penalized more aggressively close to the end of the trajectory to guarantee that the system remains in the vicinity of the desired final state.

Another possibility is to filter out undesired frequencies by introducing a cost term in the frequency domain. This requires expressing the time signals in the frequency domain using the Discrete Fourier Transform (DFT) kernel [Sundararajan, 2001]. Let K denote the DFT kernel for a given length and F be a discrete-time filter specifying the penalties on different frequency components, then a cost in the frequency domain for signal a can be formulated as $\|a\|_W^2$. The weight matrix is defined as

$$W = \text{real}\left(K^* \text{diag}\left(F^* F\right) K\right), \tag{6.21}$$

where * denotes the complex conjugate transpose and $\text{diag}(A)$ is a diagonal matrix composed of the diagonal elements in the matrix A.

Constraints

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

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

where X is a subset of the Cartesian coordinates and G is a matrix. By including velocities in X, it is possible to define velocity-dependent boundaries as well.

It is straightforward to include constraints on linear combinations of the kinematic variables. From a practical perspective, we may consider limiting the coordinate positions, the velocities, the accelerations, and the jerks.

Therefore, matrices F, G and vectors f, g in (6.6) can be formed such that

$$\begin{aligned} |u(i)| &\leq \left[u_1^{\max}, \quad \ldots, \quad u_d^{\max}\right]^T, \quad k \leq i \leq k+N-1 \\ z_j^{\max} &= \left[q_j^{\max}, \quad v_j^{\max}, \quad a_j^{\max}\right], \quad 1 \leq j \leq d \\ |z(i)| &\leq \left[z_1^{\max}, \quad \ldots, \quad z_d^{\max}\right]^T, \quad k+1 \leq i \leq k+N \\ z(k+N) &= r_f(k), \end{aligned}$$

where $r_f(k)$ is the desired final state according to (6.8). Note that if the desired target state is a constraint on the final state, we can successively reduce the sampling period, as described in the last paragraph in Sec. 6.3.

Constraints related to each joint of the robot, 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, because of the complexity of the forward and inverse kinematics, this might lead to difficult nonlinear equations, or in general non-convex constraint sets, in one or the other variable sets.

The full power of the linear MPC approach for planning a trajectory is utilized when the workspace limits can be approximated by a convex set in the joint space. Here, we consider an example to show that certain geometrical constraints in the task space could be approximated by a convex set in the joint space. However, such an approximation implies that we limit the solution to one of several possible configurations for robots that are redundant with respect to the task.

In this illustration of the method, we consider the kinematics of an ABB IRB140 robot [ABB Robotics, 2014]. The end-effector is required to move in a region defined by a thin volume located in front of the robot. We fix joints 4, 5, and 6 to 0, 40, and 0 degs., respectively, and find the range of values for the first three joints of the robot that projects the position of the end-effector to the desired volume in the Cartesian space. In Fig. 6.2, an ellipsoid is fitted to the allowed points in the joint space. In general, the convex hull [Boyd and Vandenberghe, 2004] or the largest convex subset of the acceptable points can be computed.

6.4 Simulation Results

Prior to executing experiments on the robotic setup with the developed approach to trajectory generation, simulations were performed to validate the method and investigate its characteristic features for different choices

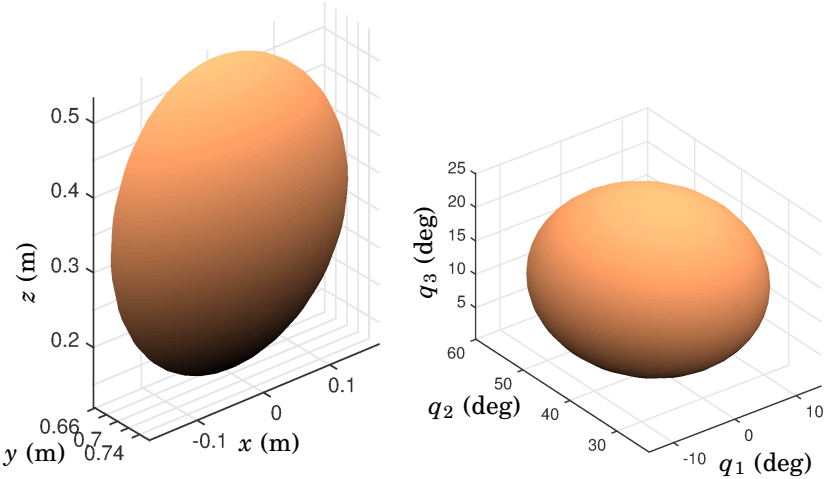


Figure 6.2 A Cartesian space volume defined in the task space (a) approximated by a convex geometrical constraint in the joint space (b). Note that a thin sphere cap in the Cartesian space is approximated by an ellipsoid in the joint space.

of the cost function and the constraints, combined with the modeling approaches outlined in Sec. 6.3. Two main simulation results are presented and evaluated in this section. The first simulation shows the computed optimal trajectories for a single target point with the approach outlined in Sec. 6.3, employing a model based on chains of integrators. The second simulation concerns the approach to trajectory generation where a linearized local dynamic model of the robot is employed and geometrical constraints are enforced on the robot position as described in Sec. 6.3.

Single Target Point

In the first simulation, a single desired target state was provided to the trajectory generator. The model with a chain of integrators defined in Sec. 6.3 was employed. For the cost function (6.10), we used the constant weighting matrices $Q = \text{blkdiag}([0,\ 1,\ 1])$ and $R = 0.001$. The prediction horizon was set to 20 samples. Starting at rest at $q = 0\,\text{rad}$, the desired state to reach was $q = 1\,\text{rad}$ at time $t = 1\,\text{s}$ with velocity $v = 0.5\,\text{rad/s}$ and acceleration $a = 0\,\text{rad/s}^2$. The constraints were chosen as $q_{\max} = 2\,\text{rad}$, $v_{\max} = 1.2\,\text{rad/s}$, $a_{\max} = 100\,\text{rad/s}^2$, and $u_{\max} = 250\,\text{rad/s}^3$. The results of the simulation are provided in Fig. 6.3, where the trajectories are visualized with black lines and the sampling instants are indicated with $+$. 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 obtained 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 as the system moved closer to the target state where accuracy was required, see Fig. 6.3.

Dynamic Model and Geometrical Constraints

Additional simulation experiments were performed in order to evaluate the performance and behavior of the approach. More specifically, the algorithms developed were evaluated with a more complex model comprising consideration of the inherently flexible dynamics of a robot in place of the model with decoupled chains of integrators. We also investigated the effect of geometric constraints in task space in this simulation. Considering that the main objectives of these simulations are to visualize the effects of different constraints and model choices, the trajectory generation is not applied in receding horizon at each sample, but rather computed over the complete time horizon directly. This approach does not imply any limitations in the evaluation, since the methods could also be employed with the iterative approach to trajectory generation in a receding-horizon fashion.

Dynamic Model A simulation of trajectory generation in joint space was performed, where the approach defined in Sec. 6.3 with a linearized model describing the flexible local dynamics of a robot with three joints was employed. The coupling dynamics—i.e., the transfer functions $H_{i,j}$ in (6.19)—were chosen in the form of

$$K \frac{w_0^2}{s^2 + 2\zeta_0 w_0 s + w_0^2}, \tag{6.22}$$

where K is a coupling factor (which is equal to one for the diagonal terms $H_{i,i}$), w_0 is the natural frequency, and ζ_0 is the damping. In the simulation, the model parameters were chosen to match the typical dynamics exhibited by a conventional industrial manipulator. The coupling dynamics between the different actuation directions were symmetrically introduced for physical reasons. A shaping function with exponential growth toward the end of the motion was introduced in the cost function, see the relation in (6.20) in Sec. 6.3. This function acted on the joint angles, the joint velocities, and the inputs in the optimization. The intention, with such a choice, is to obtain trajectories that smoothly approach the desired final state.

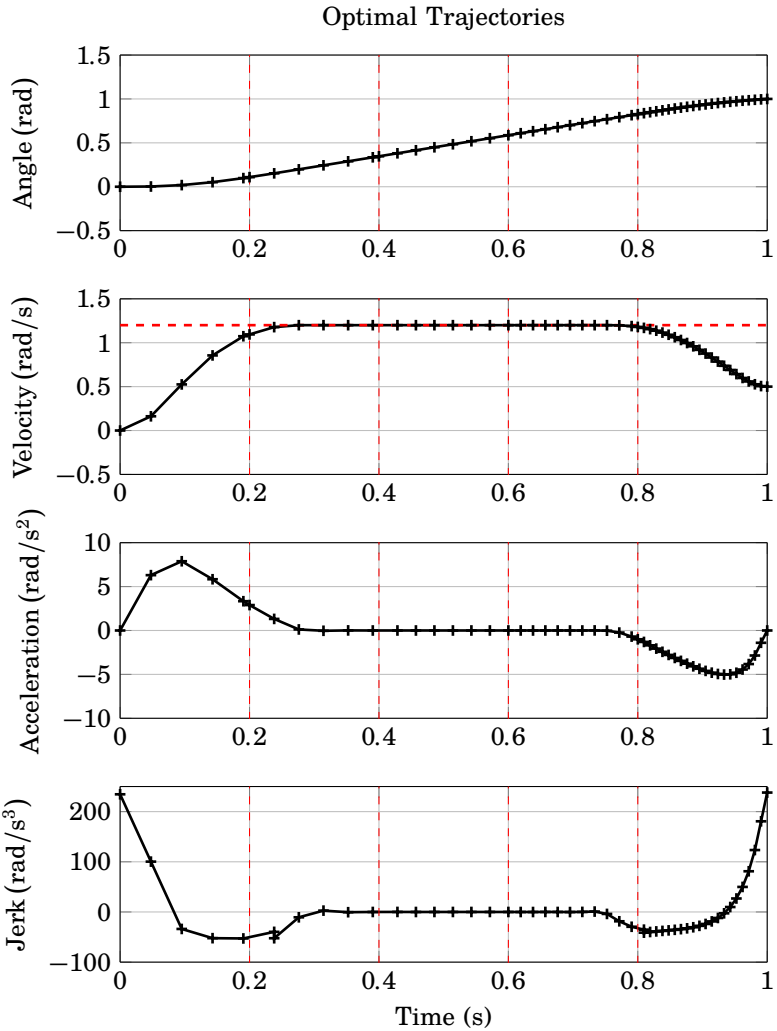


Figure 6.3 Results from a simulation where the same target point was sent repeatedly to the trajectory generator at the time instants indicated by the vertical, dashed red lines. A clear increase in the time-resolution of the trajectory is observed when moving closer to the target. Also, it is obvious that the constraint on the velocity is active during a major part of the motion.

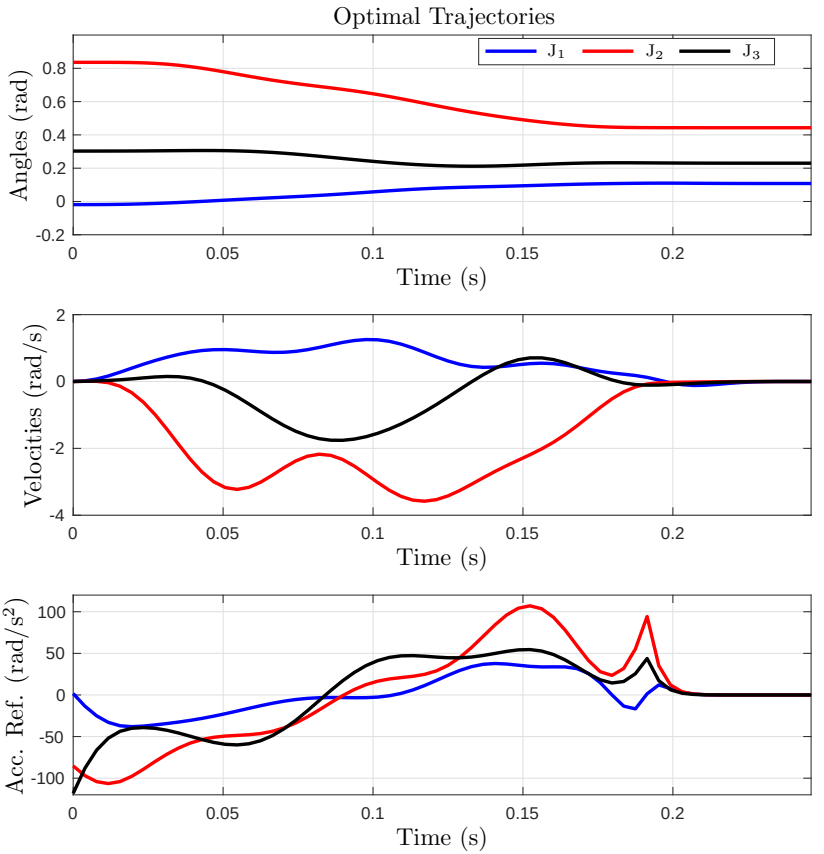


Figure 6.4 Optimal trajectories obtained with the linearized model describing the local dynamics of a flexible 3-DoF robot in an area close to the desired final point in the generated trajectories. The joints are denoted by J_i, $i \in \{1, 2, 3\}$. Compare with Fig. 6.5, where the same point-to-point motion planning task was considered but with an oversimplified model.

Figure 6.4 shows the trajectories obtained using the linearized model of the robot dynamics in a point-to-point motion. The effects of the oscillatory dynamics in the model are clearly visible. Geometrical constraints in task space were also enforced so as to show that this kind of constraints could be combined with a more complex model of the system to be controlled.

In order to highlight some of the differences between using the model based on the chains of integrators and the more complex model comprising oscillatory dynamics and coupling between the DOF, we performed another

experiment, where we applied the input trajectories (i.e., the acceleration references) computed in a trajectory generation with the simpler model to the more complex linearized model. The resulting trajectories as well as those computed based on a pure kinematic model with decoupled integrators are compared in Fig. 6.5. There is a clear difference between the trajectories in the two cases, which indicate that there is a possible major benefit of using the more complex model when such model is available. It could also be observed that using the inputs computed based on the double-integrator model on a system described by the linearized model does not guarantee that the final position constraints are fulfilled. This is clear from the simulation results in Fig. 6.5. This effect, though, will be suppressed if feedback controllers for the joint positions and velocities are used, which is standard in conventional industrial robot control systems. However, compared to using a more accurate model this strategy implies utilizing more control authority.

Geometrical Constraints The optimal trajectories for a point-to-point motion were computed based on the same model as used in the previous subsection. The trajectory generation was performed on joint level employing the kinematics of the first three joints of a robot manipulator ABB IRB140 [ABB Robotics, 2014]. Geometrical constraints were enforced in the trajectory generation using the approach outlined in Sec. 6.3, with a convex approximation of the allowed area in the task space (*cf.* Fig. 6.2). The remaining constraints on the other kinematic joint variables were chosen in the same way as in the previous simulation in this subsection. The simulation was performed both with and without the geometrical constraints and the results are shown in Fig. 6.6. In the figure, the part where the original path violates the geometric constraint is indicated in red and it can be observed that the other path, resulting from the trajectory generation with the additional geometrical constraint, is within the allowed area. It is thus clear that such geometrical constraints could be handled efficiently in the trajectory generation, if a suitable convex description of the area of interest could be found.

6.5 Experimental Setup

The method based on a decoupled kinematic robot model using a chain of integrators as described in Sec. 6.3 was implemented and experimentally evaluated on an industrial robot system. The system consisted of an ABB IRB140 industrial manipulator [ABB Robotics, 2014], combined with an ABB IRC5 control cabinet. The control system was further equipped with a research interface, ExtCtrl [Blomdell et al., 2010], in order to implement the developed trajectory-generation method as an external controller. The

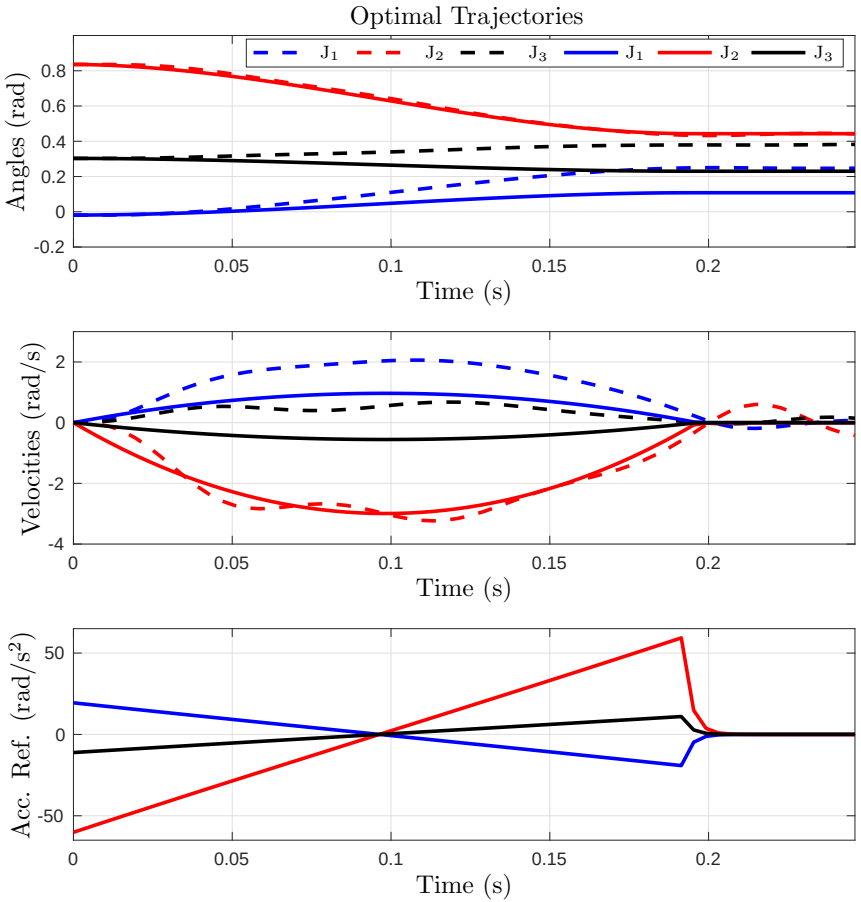


Figure 6.5 Comparison of the optimal trajectories obtained with the double-integrator model of the 3-DoF robot and the corresponding trajectories obtained when the same input is applied to the linearized model. The trajectories computed with the model based on decoupled double-integrators are shown (solid) and the simulated trajectories when the same inputs are applied to the linearized model are shown (dashed). The joints are denoted by J_i, $i \in \{1, 2, 3\}$. Note that the target point is not achieved with the oversimplified model in an open-loop strategy.

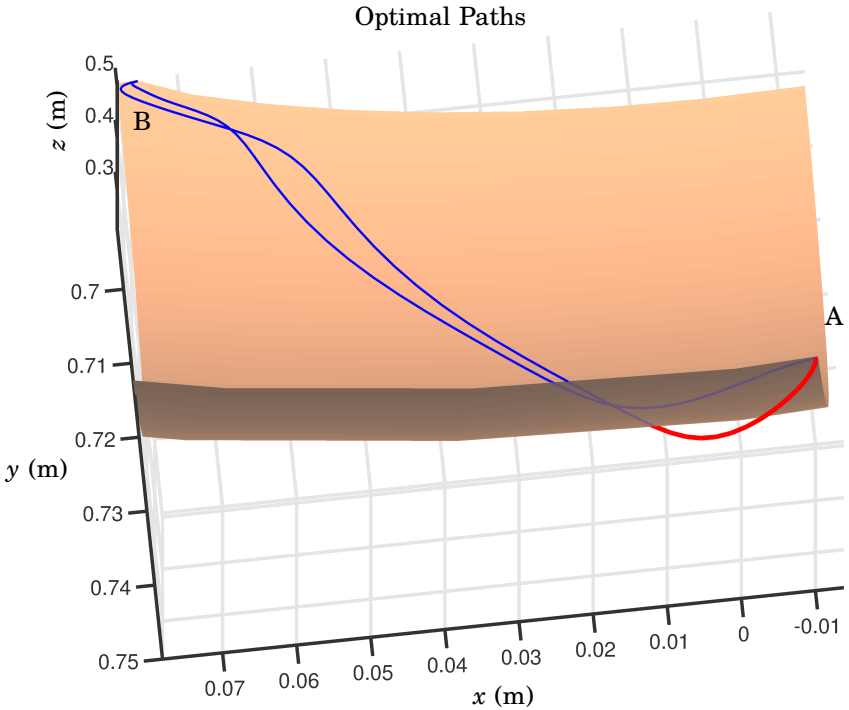


Figure 6.6 Comparison of the paths from point A to point B obtained with and without the geometrical constraint corresponding to an allowed area in the task space in a point-to-point movement. The surface shows the boundary of the geometrical constraint and the part of the original path that violates the constraint is shown in red.

research interface permits low-level access to the joint position and velocity controllers in the control cabinet at a sample period 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 period. The implementation of the trajectory generation was made in the programming language Java and the communication with the external robot controller was handled using the LabComm protocol [LabComm, 2015].

The inverse kinematics of the robot were required in order to transform the desired target point in Cartesian space to joint-space coordinates that can be used in the motion planning. The kinematics required were also available in Java from a previous implementation [Linderoth, 2013]. For

implementation of the solution of the MPC optimization problem, the CVXGEN code generator [Mattingley and Boyd, 2012; Boyd and Vandenberghe, 2004] was used. The generator produced C code, which subsequently was integrated with the Java program using the Java Native interface. The generated C code was by default optimized for performance in terms of time complexity. It enabled solution of the required quadratic program in the MPC within 0.5 ms—i.e., each optimization cycle including overhead required approximately 1–4 ms (for the four DOF used in the considered task and a prediction 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, scaling of the equations, the inputs, and the state constraints was introduced in the optimization. This is important, considering the fact that the matrices might be poorly conditioned in the case of short sampling periods when close to the final time.

For evaluating the performance of the trajectory-generation method 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 et al., 2010; Linderoth, 2013] for detection of the balls thrown and prediction of the target point. Two cameras detected the ball thrown toward the robot, and the image-analysis algorithm estimated the position and velocity, which provided the basis for the model-based prediction of the target state. A photo of the experimental setup is shown in Fig. 6.7. A movie showing the ball-catching experiment is available online [Ghazaei Ardakani et al., 2015].

6.6 Experiments and Results

The approach to trajectory generation using chains of integrators in Sec. 6.3 was evaluated in two different experiments on the robot. The experiments concerned trajectory generation in joint space. Both experiments were executed on the robot setup described in Sec. 6.5. In the experiments on the robot, the open-loop strategy to trajectory generation was employed—i.e., no feedback from the robot measurements was used. The prediction horizon was set to 20 samples. We assumed that the DOF can be independently controlled. For each DOF, we employed the constant weighting matrices $Q = \text{blkdiag}([0,\ 1,\ 1])$ and $R = 0.001$. This means that we penalize a quadratic function of velocity, acceleration, and jerk according to (6.10). In contrast to the jerk, which is the input to the adopted model, we used the computed joint velocity and position values as the reference inputs to the robot.

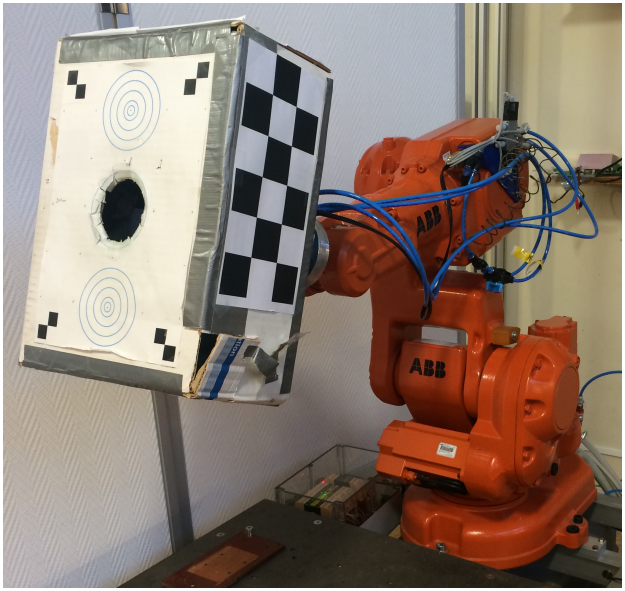


Figure 6.7 The experimental setup used for evaluation. The task is to catch a ball in the box attached to the end-effector; two cameras (not visible in the figure) were used for detection of the ball.

Sequence of Target Points

In the first experiment on the robot system, a sequence of different target states 2 s apart from each other was provided to the trajectory generator. The target points were located at the perimeter of a rectangle centered at the home position of the robot end-effector in the task space. Thus, these targets required motion along different Cartesian directions of the robot. Each target state was desired to be reached after 200 ms, with zero velocity and acceleration. The constraints in the optimization were chosen based on physical properties of the joints [ABB Robotics, 2014], with some margin, according to $q_{\max} = 2\,\text{rad}$, $v_{\max} = \pi\,\text{rad/s}$, $a_{\max} = 45\,\text{rad/s}^2$, and $u_{\max} = 1500\,\text{rad/s}^3$ for all joints. Each target state was sent with a sample period of 20 ms, resulting in the successive refinement of the trajectory. Once the robot reached the target state and paused there for 50 ms, a new trajectory for returning to the home position was computed and executed. Figure 6.8 shows the trajectories for the angle, velocity, and acceleration of joints 1 and 2.

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 position for joint 2 is illustrated in Fig. 6.9.

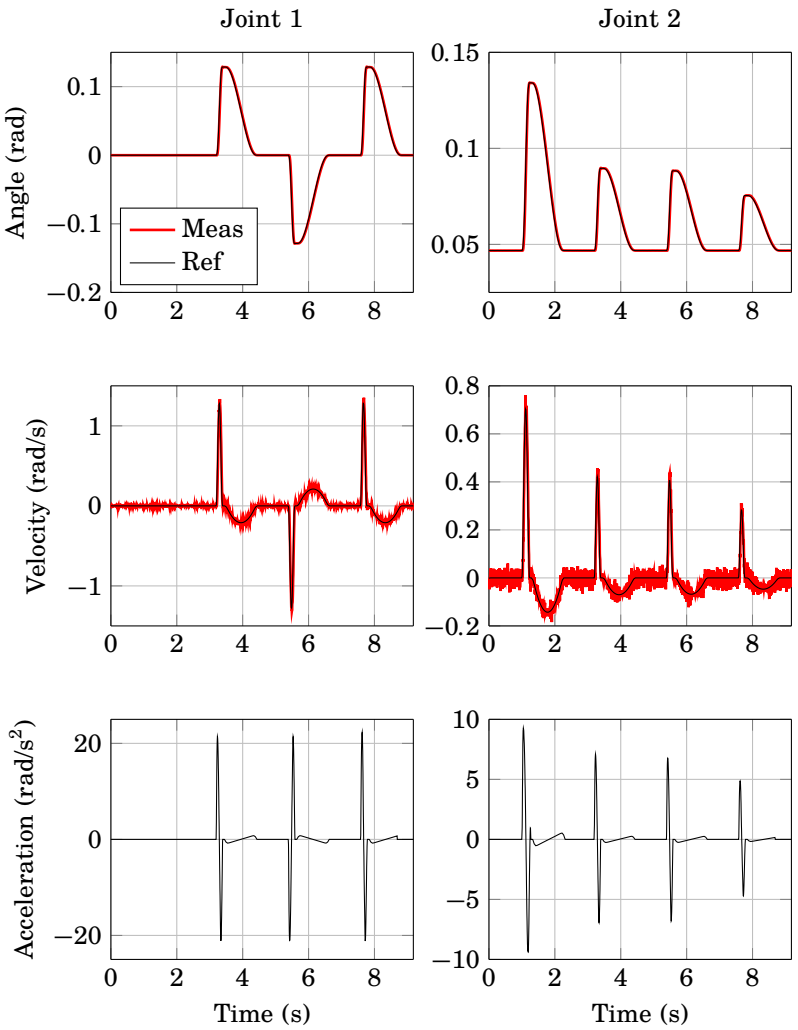


Figure 6.8 Experimental results from joint 1 and 2 of the robot where a sequence of four target states was sent to the trajectory generator, each followed by a new target state coinciding with the home position of the robot. The good tracking of the computed optimal trajectories is clear from the experiments with a sequence of target states, corresponding to different points in the robot task space.

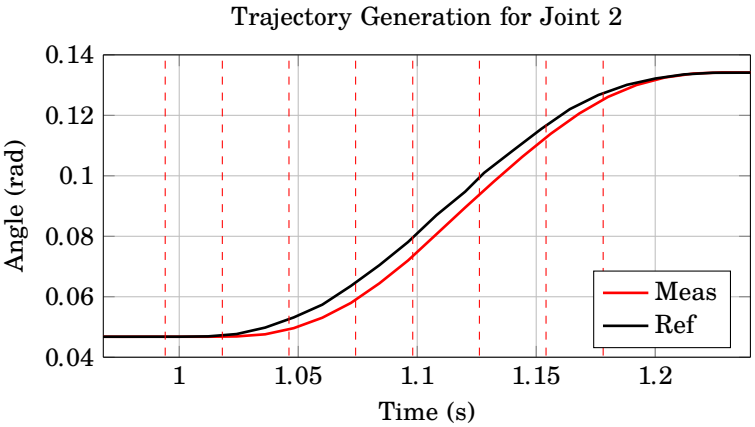


Figure 6.9 A detailed view of one of the motion segments for joint 2 in Fig. 6.8. The time instants when the target state was sent to the trajectory generator are indicated by the vertical, dashed red lines in the figure. The delay between the reference and the actual trajectory is approximately 8 ms.

The figure indicates the time instants at which the target state was sent and consequently a new trajectory generation was performed. The computation time for each trajectory generation was below the sample period of the robot controller, which meant that the optimal trajectory could be executed immediately. Further analysis of the computation time is presented in Sec. 6.6.

Ball-Catching Experiments

The most demanding evaluation of the method was the computations of optimal trajectories for the robot to catch balls thrown toward it. The time period from the release of the ball until it reached the robot was distributed in the interval [200, 800] ms. Hence, the success rate of the task depended on the satisfaction of the real-time constraints for computing a trajectory for transferring the robot from the home position to the desired final state at the predicted arrival time. As soon as new data from the vision sensors and image-analysis algorithms were available, the estimated contact point (and the corresponding arrival time) of the ball was updated, leading to recomputations of the optimal trajectory. Since not all DOF of the robot were required for the task, joints 4 and 6 were set to fixed values during the experiments. The trajectories were generated such that the robot should be at the target point with zero velocity and acceleration at the predicted arrival time with a predefined 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, acceleration, and jerk constraints, could not meet the arrival time—i.e., the optimization problem was infeasible—we still computed a trajectory for the robot to reach the target position at an estimate of the minimum required time. This improved the ball-catching performance since the initial target-point estimates exhibited large uncertainty. By moving toward the target point, there was a higher chance of catching the ball later when more accurate estimates were obtained. The constraints were chosen in the same way as for the experiment in Sec. 6.6.

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. 6.10 for the joints that were active in the robot motion. It is clear that the robot tracks the position references (computed by the trajectory generator) closely. In the lower left plot in the figure, the time instants at which new sensor data arrived are also indicated. During the motion of the ball toward the robot, the computer-vision algorithm sent the current estimate of the contact point and arrival time 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. 6.10. The figure also shows the time margin—i.e., the difference between the estimated arrival time and the earliest possible time for reaching the ball, given the constraints.

In order to verify that the real-time requirement of the trajectory-generation method was satisfied in this task, the computation times for all four DOF were measured for 100 cycles of optimization during a sequence of ball throwing. The histogram of the computation times is shown in Fig. 6.11. It can be observed that all the computation times except one are within the sampling period of the robot at 4 ms, with the average and standard deviation being 2.74 ± 0.26 ms. According to the description of Sec. 6.3, if the real-time deadline is missed, the robot continues using the latest computed trajectory.

6.7 Discussion

We have proposed an approach based on Model Predictive Control for online trajectory generation with real-time constraints for robots. The main characteristic of this method is that it allows generation of trajectories that are optimal with respect to a quadratic cost function (possibly extended with additional terms), while satisfying a final-time constraint as well as linear constraints on the input and state variables. Possible extensions

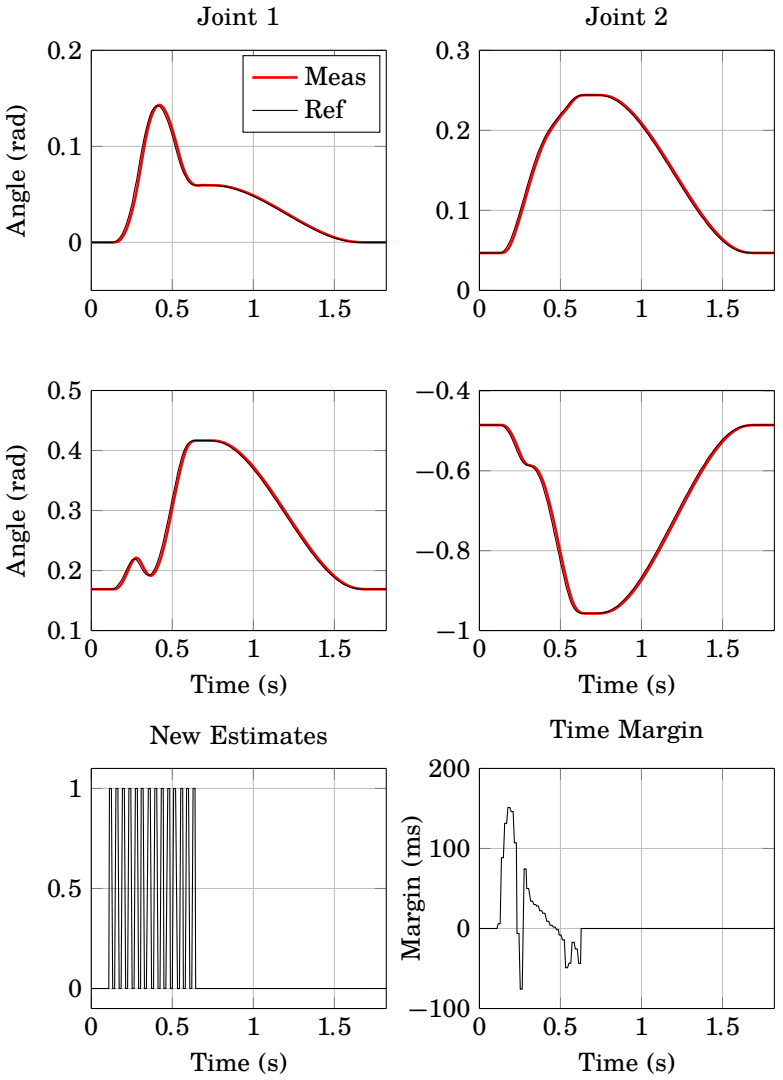


Figure 6.10 Results from a ball-catching experiment. As the ball moved toward the robot, new estimates of the contact position and the arrival time were obtained and thus the trajectory was recomputed. A new estimate was obtained on the edge of the signal shown in the lower left plot. The lower right plot shows the margin of the current estimate of the arrival time with respect to the minimum time required to arrive at the target (see the definition in the text). The negative values imply that given the latest estimate, the target cannot be met in time.

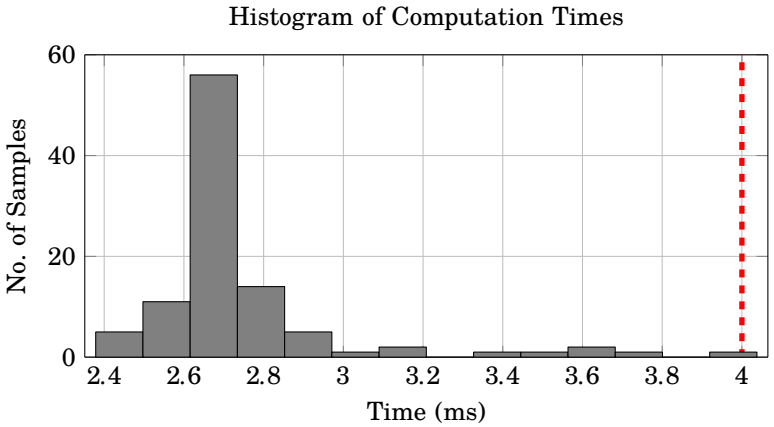


Figure 6.11 The histogram of computation times for the trajectory generation during ball-catching experiments based on 100 different executions (four joints were employed). The vertical, dashed red line represents the sampling period of the robot system.

with respect to the cost function and convex constraints were discussed and evaluated in simulations. The MPC solution considered is based on linear systems and convex constraints. This implies that dynamical variables of many manipulators such as forces and torques cannot be directly used in the optimization, neither as part of the cost function nor constraints. However, no severe limitation is introduced unless extremely high performance, *i.e.*, working close to the physical limits, is demanded. On the other hand, this restriction allows to obtain a convex problem, which does not suffer from local optima and can be solved efficiently. This aspect was illustrated in robot experiments in this paper using decoupled chains of integrators as defined in Sec. 6.3 and linear constraints on the kinematic variables. In addition, a more complex linear model, comprising oscillatory dynamics, was considered and evaluated in simulations. Another interesting feature for trajectory generation that was investigated in simulation was to enforce explicit position constraints in task space by corresponding approximate constraints in joint space.

In contrast to [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. Note that there is an important difference between our method and methods based on a fixed trajectory profile [Paul, 1979; Taylor, 1979], such as a fifth-order polynomial, or time scaling [Hollerbach, 1984] of minimum-time solutions. With neither of these other methods, it is possible to guaran-

tee that the solution will both fulfill the state, the input, and the final-time constraints and result in the lowest possible value of an objective function that is a function of the states and the control signals. To highlight the characteristics of the solution computed with the developed trajectory generator, the trajectories shown in Fig. 6.3 in Sec. 6.4 for a single target point were compared to the solutions obtained by minimizing the integral of the jerk (blue curves) and by scaling the minimum-time solution (green curves) in Fig. 6.12. These trajectories were computed based on the same kinematic model using a chain of integrators as employed in the MPC approach. The first alternative approach considered was to minimize the integral of the squared jerk, while fulfilling the initial conditions and the end constraints at the final time t_f, see Sec. 6.4. Consequently, the cost function in continuous time was

$$\underset{u}{\text{minimize}} \int_0^{t_f} u(t)^2 \mathrm{d}t. \tag{6.23}$$

In this case, the solutions for the kinematic variables can be calculated analytically and turn out to be polynomials, where the polynomial describing the position is of fifth order, see [Ghazaei Ardakani, 2015]. With such an approach, constraints on the kinematic variables—i.e., position, velocity, acceleration, and jerk—obviously cannot be accounted for in the trajectory generation. This is also clear when examining the computed velocity in Fig. 6.12, see the blue line in the second upper plot. The second approach considered in the comparison was to compute the time-optimal solution, given the initial conditions, the final constraints, and the constraints on the kinematic variables, see Sec. 6.4. The cost function in this case is

$$\underset{u}{\text{minimize}}\ t_f. \tag{6.24}$$

The optimization problem was solved numerically using the JModelica.org platform [Åkesson et al., 2010]. The solution was subsequently scaled in time [Hollerbach, 1984] with a factor $\gamma \leq 1$ in order to reach the final state at the desired time t_f. The solution is visualized in green in Fig. 6.12, where the scaling factor $\gamma = 0.93$ has been applied. By definition, the scaling factor is equal to or less than one, otherwise the original trajectory-generation problem to be solved would not be feasible. It should also be noted that the minimum and maximum constraints on the kinematic variables are still satisfied, even though the state constraints are not necessarily active during the motion as a result of the scaling. After the scaling process, however, the non-zero final constraint on the velocity is not satisfied. This is clear in Fig. 6.12 for the velocity, which is slightly lower than the desired and hence does not fulfill the terminal constraint $v(t_f) = 0.5\,\text{rad/s}$.

Another characteristic of our method is that it improves the accuracy of the optimal trajectory as the system approaches the desired final state, by

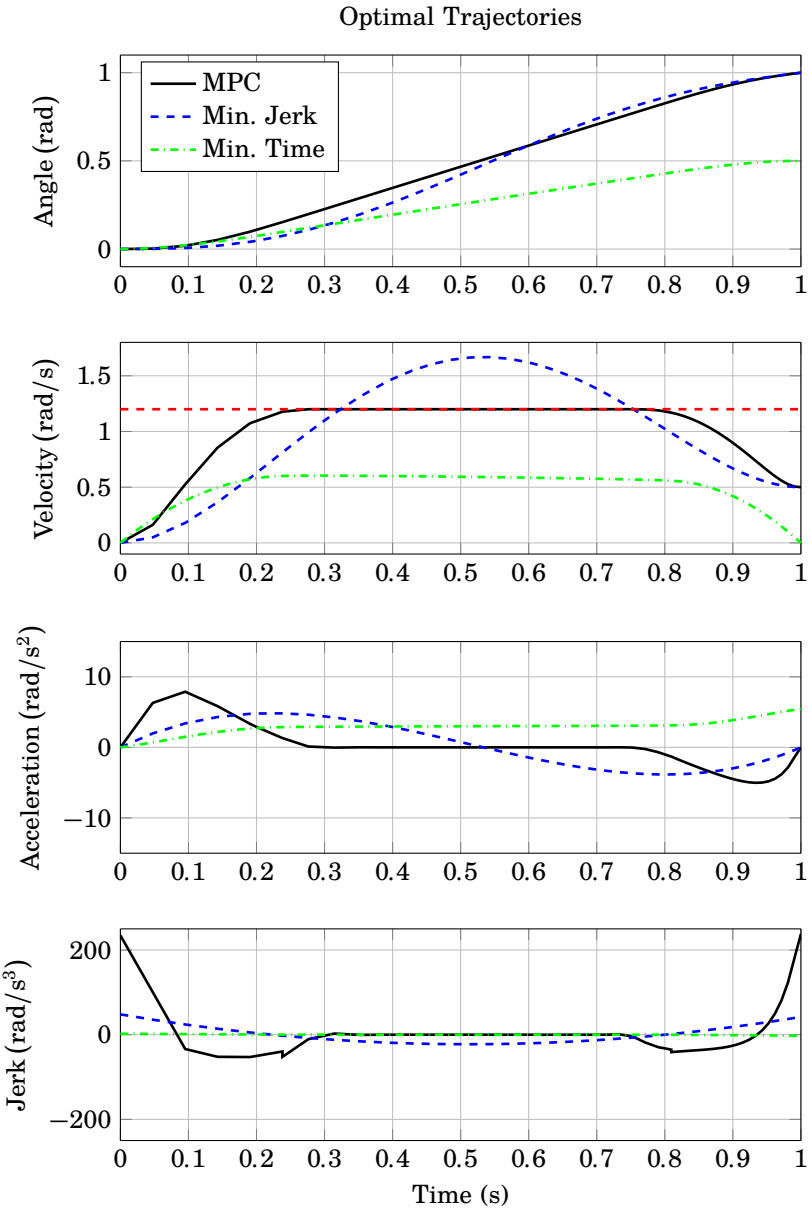


Figure 6.12 Comparison of the trajectories in Fig. 6.3 in Sec. 6.4 computed with the MPC-based optimization approach (solid black) with the corresponding trajectories obtained by minimizing the jerk (dashed blue) and by scaling the minimum-time solution (dashed-dotted green).

increasing the time-resolution. Hence, despite a small number of discretization points in the initial trajectory generation, the trajectory is successively being refined. This approach decreases the computational burden in the optimization and thus enables computation of optimal trajectories with real-time constraints.

We expect an improvement of the computation time with a more efficient implementation of the algorithms. Since the changes between two consecutive samples can be very small, the convergence of the optimization problem can be largely sped up by warm starting with the previous solution. In cases where each DOF can be independently controlled, such as for the model with a chain of integrators discussed in Sec. 6.3, an efficient way to reduce the time-complexity is to distribute the computation for each DOF on a separate core of the CPU. This will allow scaling of the described algorithm to a high number of DOF, since the major part of the computation time is spent on solving convex optimization problems. Reducing the computation time allows an increased resolution of the discretization grid, if prompted by the accuracy requirements of a task.

6.8 Conclusion

Model Predictive Control can offer a framework for generating trajectories, which goes beyond tracking problems. A subset of MPC problems—e.g., with quadratic cost functions and linear constraints—has the potential of being efficiently implemented. This fact allowed us to make a real-time implementation of fixed-time point-to-point trajectory planning for the task of ball-catching, with the feature of successive refinement of the planned trajectory. In extensive experiments, the method was evaluated in the demanding and time-critical task of ball-catching with online updates of the estimated target state from a vision system as the ball approached the robot. Several additional convex constraints and adaptations of the cost function that can be of interest in the trajectory generation using MPC were investigated and evaluated in extensive simulations.

7

Optimal Trajectories for Ball and Finger System

7.1 Introduction

Model-based design is the prevailing approach in engineering. The automotive and aerospace industries make extensive use of modeling and optimization tools to improve their products. Thanks to the increase of computation power, model-based optimization to address control problems has also gained popularity in recent years.

Practical systems often contain a combination of discrete and continuous state variables. Accurate modeling of such systems requires the incorporation of hybrid dynamics. Hybrid models can also appear in the context of processes with switched dynamics. For example in mechanics, establishing or breaking a contact changes the behavior of the system. Modeling the rapid changes in forces resulting from such phenomena leads to stiff differential equations, causing trouble for numerical methods. An alternative approach is to instead allow discontinuity in the state trajectories. The framework of hybrid systems allows both switched dynamics and discontinuous state variables.

Applying optimization techniques to hybrid models has resulted in the solution of various practical problems. For example, hybrid dynamic optimization for making discrete decisions in power plants have been considered [Fouquet et al., 2014]. In the bipedal locomotion planning, there are many examples where hybrid dynamic optimization methods have been used. For example, in [Felis and Mombaur, 2016] the full dynamics of a humanoid is considered to find a walking pattern.

A competing paradigm to the model-based approach is the model-free approach. Using a variety of model-free techniques, remarkable results for optimal control of robots interacting with the environment have been reported [Kumar et al., 2016; Rombokas et al., 2013; Tassa et al., 2012].

Despite the successful results, model-free approaches provide us with little or no insight about fundamental issues in the control of such systems.

This chapter contributes by presenting a simple, yet realistic enough, example system to capture the main issues arising in modeling and optimal control of dynamics with varying contacts. We propose a ball and finger system resembling a trackball interacted with by a human finger. *Unilateral* and *non-holonomic* constraints play an important role in this system. In order to derive analytic equations, simple 3D geometries are considered. The model enjoys the following features:

- Analytic 3D models with contacts
- Rolling of finger against ball
- Dynamics and slippage
- Suitable for simulation as well as optimization

There are elaborate mathematical models for studying contact stability [Yamada et al., 2012; Yamada and Yamamoto, 2013]. In contrast, the purpose of the model developed in this chapter is to capture contact transitions and to find optimal trajectories. The hybrid dynamic optimization required for this purpose has been performed utilizing a multiphase approach, i.e., a sequence of the mode changes of the system are mapped to phases.

The rest of the chapter is organized as follows. In Sec. 7.2, the mechanical model of the process is developed. We turn the mechanical model into a hybrid system in Sec. 7.3. This model is used for simulation as well as optimization. Section 7.4 discusses the trajectory-planning problem and the optimization approach to find optimal solutions. Various tools have been used for simulation and optimization. The tool chain is detailed in Sec. 7.5. We introduce instances of the trajectory-planning problems and give the results of the optimization and simulation in Sec. 7.6. Section 7.8 discusses various aspects of modeling and optimization. We draw conclusions and propose future research in Sec. 7.9. Additionally, Appendix A.1 and A.2 provide the details of the mathematical expressions for the mechanical models.

7.2 Mechanical Modeling

In this section, we build the mechanical model of the process shown in Fig. 7.1. To produce analytic equations, we consider an idealized model, which only admits contact between a sphere and a line segment. The following list summarizes, the modeling assumptions:

1. The finger is modeled as a rigid body with 3 degrees of freedom (DOF).

Table 7.1 DH parameters of the finger

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

2. The ball is a sphere with radius r and a spherical joint constraint.
3. The Coulomb friction model is considered with static (μ_s) and kinetic (μ_k) friction constants.
4. There is viscous friction in the joints and the socket with friction constants μ_v and D, respectively.
5. There is no torsional friction.
6. The restitution factor is equal to zero, i.e., when the finger collides with the ball it will not bounce back.
7. The links are idealized line segments.
8. In case of contact, single point contact either at the end-effector or along the last link is allowed. This is guaranteed by the choice of the geometry.
9. The centers of gravity (COG) are located at the middle of each link and inertia tensors are diagonal with respect to these points.

Geometry and Kinematics

Considering Fig. 7.1, the geometry of the system is described here. The ball with radius $r = 1$ is assumed to be located at the origin $o_b^T = [0, 0, 0]$. The base of the finger is specified by the transformation consisting of a rotation -90 degrees around the y-axis and a translation $o_f^T = [2, 0, 0]$. The kinematics of the finger is specified by the Denavit-Hartenberg (DH) parameters [Denavit and Hartenberg, 1955] according to Table 7.1. We use $q \in \mathbb{R}^3$ to denote the finger joint angles. Knowing the DH parameters and the base transformation, the forward kinematics and the geometric Jacobian with respect to the fingertip can be calculated. See Appendix A.1 for the details.

Collision

To detect collision between the ball and the finger, the closest distance between the ball and the last link is found. Define $p_m := p_e + \alpha \hat{e}_{x_3}$ as the

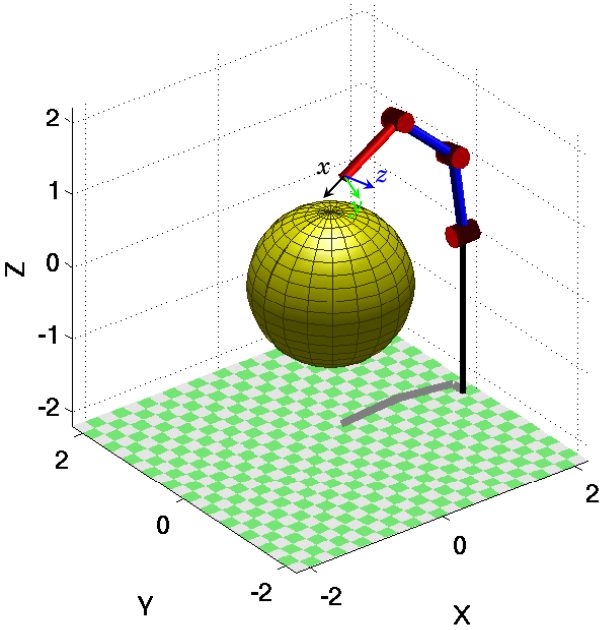


Figure 7.1 An illustration of the ball-and-finger system.

point on the line corresponding to the last link closest to the ball, where $\hat{e}_{x_3}$ is the unit vector parallel to link 3, p_e is the position of the fingertip, and α determines how far p_m lies from the fingertip. Then

$$(p_m - o_b) \cdot \hat{e}_{x_3} = 0, \tag{7.1}$$

which results in

$$\alpha = (o_b - p_e) \cdot \hat{e}_{x_3}. \tag{7.2}$$

Define also the potential collision point, i.e., the point on link 3 closest to the ball

$$p_c := p_e + \mathrm{sat}^0_{-a_3}(\alpha)\hat{e}_{x_3}, \tag{7.3}$$

where $\mathrm{sat}^U_L(x)$ denotes the saturation function with L as the lower limit and U as the upper limit and $a_3 = 1$ is the length of the third link. Hence, the shortest distance of the third link to the ball is

$$h(q) := \|r_c\| - r. \tag{7.4}$$

where $r_c := p_c - o_b$ is the vector to the closest point on the finger to the ball.

If the ball and the finger are impenetrable, it is required that $h(q) > 0$. Therefore, the condition for collision can be stated as

$$h(q) \leq 0. \tag{7.5}$$

The velocity of the third link at the potential contact point denoted by v_c can be derived from the velocity of the end point v_e using the velocity relation of rigid motion [Meriam and Kraige, 2012] according to

$$\begin{aligned} v_c &:= v_e + \omega_3 \times (p_c - p_e) \\ &= v_e - \mathrm{sat}^0_{-a_3}(\alpha)\hat{e}_{x_3} \times \omega_3 \\ &= \left(J_p - \mathrm{sat}^0_{-a_3}(\alpha)S(\hat{e}_{x_3})J_o\right)\dot{q} \\ &= J_c\dot{q}, \end{aligned} \tag{7.6}$$

where ω_3 is the angular velocity of the last link and $S(\cdot)$ denotes the skew-symmetric operator performing the cross product. We also define

$$J_c := J_p - \mathrm{sat}^0_{-a_3}(\alpha)S(\hat{e}_{x_3})J_o, \tag{7.7}$$

which is the Jacobian at the potential contact point. The expression for the Jacobian J_c can also be obtained by substituting a_3 with $a_3 + \mathrm{sat}^0_{-a_3}(\alpha)$ in the Jacobian expression (A.4). Note that v_c is the velocity of the link at the contact point and not the velocity of the contact point itself, i.e., $v_c \neq \dot{r}_c$.

Constrained Motion

When the finger is touching the ball, its motion is constrained to a manifold. Using (7.4), the finger touches the ball when

$$h(q) = 0. \tag{7.8}$$

This is equivalent to

$$\|r_c\|^2 - r^2 = 0. \tag{7.9}$$

Differentiating (7.9) w.r.t. time, results in

$$v_c \cdot r_c = 0. \tag{7.10}$$

This means that in the contact situation, the velocity of the contact point remains tangential to the ball.

Depending on whether there is an interaction force between the objects, the contact may be active or inactive. Let us denote the normal vector at the contact point outwards w.r.t. the ball by

$$n := \frac{r_c}{\|r_c\|}. \tag{7.11}$$

The normal component of the interaction force at the contact is then given by

$$\lambda_n := \lambda \cdot n, \tag{7.12}$$

where λ denotes the contact force exerted on the finger. Using the Coulomb friction model, interaction forces between two bodies are possible if and only if there is a normal force between the contacting surfaces. Therefore, an active contact implies $\lambda_n > 0$.

The relative velocity between the finger and the ball at the contact point is

$$\Delta v := \omega \times r_c - v_c. \tag{7.13}$$

When two bodies are sticking, the relative velocity between them is zero, i.e.,

$$\Delta v = 0. \tag{7.14}$$

This constraint is maintained by the static friction f_s. It is useful to define $v_\perp$ and $v_\parallel$ for the perpendicular and tangential decomposition of the vector v w.r.t. a plane with normal vector n, according to

$$v = v_\parallel + v_\perp, \tag{7.15a}$$
$$v_\perp = (v \cdot n)n = (n \otimes n)v = P_n v, \tag{7.15b}$$
$$v_\parallel = -n \times (n \times v) = v - v_\perp = (I - P_n)v = T_n v, \tag{7.15c}$$

where $\otimes$ denotes the outer product and we have defined

$$P_n := (n \otimes n), \tag{7.16}$$
$$T_n := I - P_n. \tag{7.17}$$

Let us define

$$\sigma := \mu_s \lambda_n - \left\| \lambda_\parallel \right\|. \tag{7.18}$$

Hence, according to the Coulomb model the condition for the stiction can be stated as

$$\sigma \geq 0 \Leftrightarrow \|f_s\| \leq \mu_s \lambda \cdot n. \tag{7.19}$$

When the finger is slipping against the ball, the relative velocity Δv is not zero anymore and

$$\lambda_\parallel = f_k := \mu_k \lambda_n \frac{\Delta v}{\|\Delta v\|}, \tag{7.20}$$

where f_k denotes the kinetic friction. The contact is inactive when $\lambda_n \leq 0$.

Depending on the contact point between the finger and ball may arise two types of contact: point and sphere or line and sphere. If the contact is not slipping, for the first type of contact, (7.14) can be integrated to result in a geometrical constraint, while the second type yields a rolling without slipping constraint which is non-holonomic.

Dynamics

In this section, we derive the equations of motion for the ball and finger system. The kinetic and potential energy of the finger can be described by

$$\mathcal{T}_f = \sum_{i=1}^{3} \frac{1}{2} m_i \dot{q}^T (J_P^i)^T J_P^i \dot{q} + \sum_i \frac{1}{2} \dot{q}^T (J_O^i)^T R_i I_i R_i^T J_O^i \dot{q}, \tag{7.21a}$$

$$\mathcal{U}_f = \sum_{i=1}^{3} m_i g_0^T p_i, \tag{7.21b}$$

respectively. Here, m_i and I_i denote the mass and inertia matrix of link i, respectively, J_P^i and J_O^i denote partial Jacobians up to link i, R_i is the rotation matrix from the world coordinate to link i, $g_0 = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$ is the gravity acceleration vector, and p_i denotes the position vector to the center of the gravity of link i. The Lagrangian of the finger is $\mathcal{L}_f := \mathcal{T}_f - \mathcal{U}_f$. Accordingly, we derive the equations of motion

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial \mathcal{L}_f}{\partial \dot{q}} \right) + \frac{\partial \mathcal{L}_f}{\partial q} = Q^e + Q_f^c. \tag{7.22}$$

The generalized external forces and the generalized forces due to the kinematical constraints are denoted by Q^e and Q_f^c, respectively. Substituting Q^e with the actuation torques, τ, minus the viscous friction, we obtain

$$M_f(q)\ddot{q} + C(q, \dot{q})\dot{q} + \mu_v \dot{q} + g(q) = \tau + Q_f^c, \tag{7.23}$$

where $M_f(q)$ is the mass matrix for the finger, $C(q, \dot{q})\dot{q}$ denotes the contribution of the centrifugal and Coriolis forces, μ_v is the viscous friction coefficient, and $g(q)$ is the gravitational force. The definitions of these functions are given in Appendix A.2.

For the ball, using the Newton-Euler equations, we find

$$I_b \dot{\omega} + \omega \times I_b \omega + D\omega = Q_f^c, \tag{7.24}$$

where $I_b = \frac{2}{5} m r^2 \mathbf{1}$ is the inertial tensor for the ball and D denotes the friction with the socket, and ω is the angular velocity of the ball.

In the rest of this section, we consider different regions depending on the active constraints imposed by the contact. Finally, we describe the overall system.

Free Motion In case of no contact, we have $Q_f^c = 0$. Hence,

$$M_f(q)\ddot{q} + C(q,\dot{q})\dot{q} + \mu_v\dot{q} + g(q) = \tau, \tag{7.25a}$$
$$I_b\dot{\omega} + \omega \times I_b\omega + D\omega = 0. \tag{7.25b}$$

Sticking The condition of no slipping (7.14) can be rewritten as

$$G\begin{bmatrix}\dot{q}\\ \omega\end{bmatrix} = 0. \tag{7.26}$$

where we have defined $G := [J_c\, S(r_c)] \in \mathbb{R}^{3\times 6}$. The generalized forces due to the constraints according to the Lagrange–d'Alembert theorem [Bloch, 2003] can be derived as

$$\begin{bmatrix}Q_f^c\\ Q_b^c\end{bmatrix} = G^T\lambda, \tag{7.27}$$

where λ are the Lagrange multipliers corresponding to the interaction forces. Therefore, (7.23) and (7.24) can be rewritten as

$$M_f(q)\ddot{q} + C(q,\dot{q})\dot{q} + \mu_v\dot{q} + g(q) = \tau + J_c^T\lambda, \tag{7.28a}$$
$$I_b\dot{\omega} + \omega \times I_b\omega + D\omega = -r_c \times \lambda. \tag{7.28b}$$

Solving for Interaction Forces The interaction forces λ may be solved for in order to reduce the DAE into an ODE. They could also be used to evaluate whether in a certain state sticking is possible without switching to new dynamics. The calculation may also be used to find out the direction of kinetic friction at the onset of slipping. Let us define

$$M := \text{blkdiag}\left(M_f(q), I_b\right). \tag{7.29}$$

We can rewrite (7.28) as

$$M\begin{bmatrix}\ddot{q}\\ \dot{\omega}\end{bmatrix} + f(q,\dot{q},\omega,\tau) = G^T\lambda. \tag{7.30}$$

where we have defined

$$f(q,\dot{q},\omega,\tau) := \begin{pmatrix} C(q,\dot{q})\dot{q} + \mu_v\dot{q} + g(q) - \tau \\ \omega \times I_b\omega + D\omega \end{pmatrix}. \tag{7.31}$$

The Lagrange multipliers λ can be calculated by first taking the derivative of the constraints (7.26). Afterward, we multiply (7.30) by GM^{-1} from the left and use the relation obtained from the derivative of the constraints.

Let us define $\Gamma := GM^{-1}G^T$. Assuming Γ^{-1} exists, $\lambda = \lambda(t, q, \dot{q}, \omega)$ can be calculated according to

$$\begin{aligned}\lambda &= \Gamma^{-1}\left(GM^{-1}f(q,\dot{q},\omega,\tau) - \sum_{k=1}^{n}\left(\frac{\partial G}{\partial q_k}\dot{q}_k\right)\begin{bmatrix}\dot{q}\\ \omega\end{bmatrix}\right)\\ &= \Gamma^{-1}\left(GM^{-1}f(q,\dot{q},\omega,\tau) + \omega \times \dot{r}_c - \dot{J}_c\dot{q}\right).\end{aligned} \tag{7.32}$$

Note that the matrix M is always invertible with

$$M^{-1} = \text{blkdiag}\left(M_f^{-1}(q),\, I_b^{-1}\right). \tag{7.33}$$

Slipping In this case, the constraint only defines the normal part of the interaction force, i.e., λ_n. By rewriting (7.10), we find

$$r_c \cdot J_c\dot{q} = 0. \tag{7.34}$$

We normalize (7.34) by dividing by $\|r_c\|$, to get

$$n^T J_c\dot{q} = 0. \tag{7.35}$$

Now, from the Lagrange–d'Alembert theorem, it is clear that the contribution of the constraint force is $Q_f^c = J_c^T n\lambda_n$. Including the kinetic friction forces f_k defined in (7.20), we conclude

$$M_f(q)\ddot{q} + C(q,\dot{q})\dot{q} + \mu_v\dot{q} + g(q) = \tau + J_c^T(f_k + n\lambda_n), \tag{7.36a}$$

$$I_b\dot{\omega} + \omega \times I_b\omega + D\omega = -r_c \times f_k. \tag{7.36b}$$

It is also possible to reformulate (7.36) using the total interaction force $\lambda := f_k + n\lambda_n$ such that we obtain the same relations as (7.28). In this case, (7.20) can be written as

$$\lambda \cdot \Delta v = \|\Delta v\|\,\mu_k(\lambda \cdot n), \tag{7.37a}$$

$$(n \times \Delta v) \cdot \lambda = 0, \tag{7.37b}$$

which additionally has no risk of division by zero as opposed to (7.20).

Impacts

Collisions give rise to rapid changes of the states. By allowing discontinuities in the state trajectory, it is possible to model the effect of impacts right after the collision without dealing with the actual process. In this case, new values for post-impact quantities must be calculated. Since they can have jumps, we use superscripts $+$ and $-$ to denote the right and left limits of a quantity, respectively. Next, we consider various impact laws. Since after collision the finger is neither allowed to penetrate the ball nor to bounce off directly (restitution factor equal to zero), all the impact models ensure $(J_c\dot{q}^+)_\perp = n \cdot J_c\dot{q}^+ = 0$.

Newton's Impact Law If after impact the finger keeps its tangential velocity intact and only loses its normal velocity, we get

$$J_c\dot{q}^+ = \left(J_c\dot{q}\right)_{\|}^- = T_n J_c \dot{q}^-. \tag{7.38}$$

Accordingly, we can solve for the post-impact joint velocities. The impulse in this case is perpendicular to the ball and cannot change the velocity of the ball.

Sticking When sticking, the force impulse ι as a result of the impact makes the post-collision relative velocity $\Delta v^+ = 0$. From the equations of motion (7.28) and (7.13), we conclude

$$M_f(q)(\dot{q}^+ - \dot{q}^-) = J_c^T \iota, \tag{7.39a}$$
$$I_b(\omega^+ - \omega^-) = -r_c \times \iota, \tag{7.39b}$$
$$\omega^+ \times r_c - J_c\dot{q}^+ = 0, \tag{7.39c}$$

which can be solved for $\dot{q}^+$, ω^+, and ι. By combining (7.39a)–(7.39c), we find

$$\left(J_c M_f^{-1} J_c^T - S(r_c) I_b^{-1} S(r_c)\right) \iota = \omega^- \times r_c - J_c\dot{q}^- = \Delta v^-. \tag{7.40}$$

This shows that the impulse ι is a function of the contact point r_c, the configuration of the finger q, and the pre-impact relative velocity Δv^-, neither the absolute velocity of the finger nor the ball.

Slipping Considering the maximum dissipation principle, an impact law for this case can be formulated as minimizing the post-impact energy. The optimization problem to be solved is

$$\begin{aligned}
&\text{minimize} && \frac{1}{2}\omega^{+T} I_b \omega^+ + \frac{1}{2}\dot{q}^{+T} M(q)\dot{q}^+ && (7.41a)\\
&\text{with respect to} && \iota, \omega^+, \dot{q}^+ &\\
&\text{subject to} && M(q)(\dot{q}^+ - \dot{q}^-) = J_c^T \iota, && (7.41b)\\
& && I_b(\omega^+ - \omega^-) = -r_c \times \iota, && (7.41c)\\
& && n \cdot J_c\dot{q}^+ = 0, && (7.41d)\\
& && \left\|\iota_{\|}\right\| \le \mu_k \iota \cdot n. && (7.41e)
\end{aligned}$$

A heuristic way is to minimize the post-impact relative velocity. This allows to make a quick transition to sticking. The corresponding optimization

problem is

$$\begin{aligned}
&\text{minimize} && \left\|\Delta v^+\right\| && \text{(7.42a)}\\
&\text{with respect to} && \mu,\, \iota,\, \omega^+,\, \dot{q}^+ &&\\
&\text{subject to} && M(q)(\dot{q}^+ - \dot{q}^-) = J_c^T \iota, && \text{(7.42b)}\\
& && I_b(\omega^+ - \omega^-) = -r_c \times \iota, && \text{(7.42c)}\\
& && n \cdot J_c \dot{q}^+ = 0, && \text{(7.42d)}\\
& && \iota_\parallel = \mu \iota_n \frac{\Delta v_\parallel^-}{\left\|\Delta v_\parallel^-\right\|} && \text{(7.42e)}\\
& && 0 < \mu \leq \mu_k, && \text{(7.42f)}
\end{aligned}$$

where we have additionally assumed that the direction of the tangential part of the impulse is aligned with the pre-impact relative velocity. An analytic solution to problem (7.42) can be found. Subsequently, the equations can be used in an optimal control problem.

From (7.13) and using (7.42b)–(7.42c), we conclude

$$\begin{aligned}
\Delta v^+ &= \omega^+ \times r_c - J_c \dot{q}^+ \\
&= \omega^- \times r_c - J_c \dot{q}^- - \left(J_c M_f^{-1} J_c^T - S(r_c) I_b^{-1} S(r_c)\right) \iota \\
&= \Delta v^- - \left(J_c M_f^{-1} J_c^T - S(r_c) I_b^{-1} S(r_c)\right) \iota. \qquad \text{(7.43)}
\end{aligned}$$

Multiplying (7.42b) by $J_c M^{-1}$ from the left and using (7.42d), we obtain

$$-n \cdot J_c \dot{q}^- = n \cdot J_c M^{-1} J_c^T \iota. \qquad \text{(7.44)}$$

From (7.42e), it follows that ι lies in the subspace spanned by n and Δv^-. Thus, we can write

$$\iota = C\beta := [n,\, \Delta v^-]\beta \qquad \text{(7.45)}$$

where $\beta \in \mathbb{R}^2$ is an arbitrary vector. By substituting (7.45) back into (7.44) and (7.43), we obtain an expression for the post-impact velocities in terms of β as well as a constraint on β. Identifying β with x, we can rewrite the problem of minimizing the post-impact velocities as

$$\begin{aligned}
&\text{minimize} && \|y - Ax\| && \text{(7.46a)}\\
&\text{subject to} && Bx = z. && \text{(7.46b)}
\end{aligned}$$

where

$$\begin{aligned}
y &= \Delta v^- && \text{(7.47a)}\\
A &= \left(J_c M_f^{-1} J_c^T - S(r_c) I_b^{-1} S(r_c)\right) C && \text{(7.47b)}\\
B &= n^T J_c M^{-1} J_c^T C && \text{(7.47c)}\\
z &= -n \cdot J_c \dot{q}^- && \text{(7.47d)}
\end{aligned}$$

There is a generic solution to problem (7.46). However, in our case, we can simply solve for x_1 or x_2 from (7.46b) and substitute it into (7.46a) to obtain a quadratic one-dimensional unconstrained minimization problem. Consequently, the friction coefficient can be obtained from

$$\mu = \frac{\beta_2}{\beta_1} \|\Delta v^-\|. \tag{7.48}$$

If $\mu \leq \mu_k$, this solution is valid for the problem (7.42). Hence, the impulse and the post-impact relative velocity can be calculated from (7.44) and (7.43), respectively. Otherwise, we set $\mu = \mu_k$. Then, from (7.42e) we conclude

$$\iota = \iota_n n + \mu_k \iota_n \frac{\Delta v_{\|}^-}{\left\|\Delta v_{\|}^-\right\|}. \tag{7.49}$$

By substituting this into (7.44), we can solve for ι_n and hence find ι. Finally, Δv^+ is obtained by using (7.43).

Overall System

The complete system has three discrete modes, corresponding to free motion, sticking, and slipping. Let λ denote the interaction force and $x^T := \left(q^T, dq^T, \omega^T\right)$, where $\dot{q} = dq$. If for the moment we do not consider the transition between modes, we can unify all the modes with the first order differential equation

$$\begin{pmatrix} I & 0 \\ 0 & M(q) \end{pmatrix} \dot{x} + \begin{pmatrix} 0 & -I & 0 \\ 0 & C(q, dq) + \mu_v & 0 \\ 0 & 0 & S(\omega) I_b + D \end{pmatrix} x + \begin{pmatrix} 0 \\ g(q) \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \tau \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ J_c^T(q) \\ -S(r_c(q)) \end{pmatrix} \lambda. \tag{7.50}$$

The determination of λ is separate for each mode as follows.

- Free motion:

$$\lambda = 0. \tag{7.51}$$

- Sticking (valid when $\left\|\lambda_{\|}\right\| \leq \mu_s \lambda_n$):

$$J_c \dot{q} + r_c \times \omega = 0. \tag{7.52}$$

- Slipping (valid when $\Delta v \neq 0$):

$$r_c \cdot J_c \dot{q} = 0, \tag{7.53a}$$
$$\lambda \cdot \Delta v = \|\Delta v\| \, \mu_k \lambda_n, \tag{7.53b}$$
$$(n \times \Delta v) \cdot \lambda = 0. \tag{7.53c}$$

Hence, we have a differential-algebraic equation (DAE) describing the dynamics in each mode, which is low index in the case of free motion and of index 2 in the case of sticking and slipping.

7.3 System Modeling

The aim of this section is to transfer the mechanical system introduced in Sec. 7.2 into a hybrid dynamical system. To find the solution to the overall system, we must also consider an impact law and the so called Signorini's condition or corner law for λ_n (see [Brogliato, 2012]) :

$$\begin{aligned} &h \cdot \lambda_n = 0, \\ &h \geq 0, \ \lambda_n \geq 0. \end{aligned} \tag{7.54}$$

First, we discuss how these complementarity conditions lead to mode transitions. In the subsequent subsection, the complete hybrid system is presented.

Mode Transitions

The complementarity conditions (7.54) lead to mode transitions. Let us partition the state space depending on whether there is an active contact or not. The resulting binary partition is

$$\lambda_n = 0 \wedge (h > 0 \ \vee \ h = 0 \wedge \dot{h} > 0 \vee \ h = \dot{h} = 0 \wedge \ddot{h} \geq 0), \tag{7.55a}$$
$$\lambda_n > 0 \wedge h = 0. \tag{7.55b}$$

Note that those states where both λ_n and h are equal to zero, i.e., inactive contacts, are included in the first partition. Also note that $h = \dot{h} = 0 \wedge \ddot{h} < 0$ must be a transition point from the first partition since it leads to penetration, but at the same time $\exists \lambda_n > 0$ such that it results in $h = 0$.

To fulfill the complementarity condition, transition happens from the mode where $\lambda_n = 0$ to $\lambda_n > 0$ when

$$h < 0 \ \vee (h = 0 \wedge \dot{h} < 0) \ \vee (h = \dot{h} = 0 \wedge \ddot{h} < 0). \tag{7.56}$$

In the opposite direction, the transition happens when $\lambda_n \leq 0$.

Figure 7.2 illustrates the transition between these modes. Each mode can have its own set of dynamical equations and the conditions for the validity of a mode is given at the lowest part of the circle. The transition conditions are shown close to the tail of the arrows and the reset maps close to the arrowheads.

To account for switching to slipping and sticking and between them, we should consider their region of validity. Both sticking and slipping account for active contacts, i.e., $\lambda_n > 0$. The transition to them is however distinguished by the relative post-impact velocity Δv^+. If the static friction is not able to maintain the sticking constraint, i.e, according to (7.19) when $\sigma < 0$, the system makes a transition to slipping. In the opposite direction, the transition depends on the relative velocity. However, according to the Coulomb law the slipping region is not valid for zero relative velocity. This implies that there is no ideal transition condition between the slipping and sticking modes, although it is possible to calculate a continuous solution for the velocities. This can be resolved by conditioning the transition on $\|\Delta v\| \leq \epsilon$. The switching then happens arbitrarily close to the actual transition by tuning the value of ϵ. To avoid switching directly back we should also make sure that $\mathrm{d}\,\|\Delta v\| \,/\, \mathrm{d}t < 0$.

Note that switching on Δv larger than zero, implies that the transition to the sticking mode is not possible. Similarly in the opposite direction, by the end of the sticking mode, the relative velocity is still equal to zero and a transition to slipping mode is not possible. A solution to these problems is that to update $\dot{q}$ and ω such that the required condition on Δv^+ is satisfied.

Since numerically it is impossible to detect exactly an equality, in the implementation the equality conditions have to be replaced with inequality conditions on the residual of the errors. We consider a constant $\epsilon > 0$ defining the numerical tolerance, e.g., $\Delta v = 0$ is implemented as $\|\Delta v\| \leq \epsilon$. Note that for numerical reasons, we include negative values of $h(q)$ in the in-contact modes.

Hybrid Model

The transitions between the modes of (7.50)–(7.53c) happen according to the discussion of Sec. 7.3. To avoid extra complexity, the contact types (point and sphere or line and sphere) are treated in the same way and are not distinguished through additional modes. The final model is a hybrid, variable-index DAE system. Figure 7.3 illustrates the hybrid model. In the figure, post-impact relative velocities are used in the conditions for transition from the free motion mode. This is, however, just for saving space. It can be assumed that (7.56) is the transition condition to an "Active Contact" state where the impact law is applied. Consequently, depending on $\|\Delta^+\|$ the transition to either sticking or slipping happens.

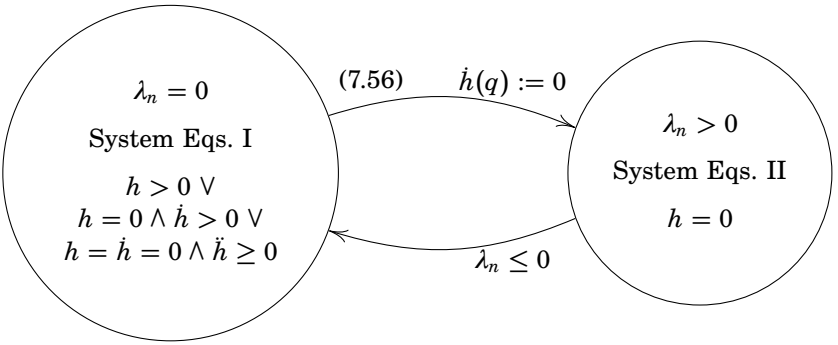


Figure 7.2 Hybrid system representation of the complementarity condition (7.54). For details of the notation, see [Lin and Antsaklis, 2014].

7.4 Optimization

In this section, we discuss how the hybrid DAE system in Fig. 7.3 can be used to find trajectories for the system using optimal control. In the optimization, the slipping mode and its transitions are not considered, i.e., we only consider free motion and sticking, which correspond to the parts in blue in Fig. 7.3. For many applications, this choice is motivated since slippage implies energy loss, which is preferably avoided. This hybrid DAE is ill-suited for straightforward application, due to the combination of nonlinear and hybrid dynamics. To deal with the hybrid dynamics, we use a mixture of multiple phases and smooth approximations to obtain a system model that is better suited for numerical optimization. The resulting equations are nonlinear and sufficiently differentiable to apply Newton-based methods.

Optimal Control Problem

To investigate optimal trajectories, we use the optimal control problem formulation

$$\begin{aligned}
&\text{minimize} && \phi(c(t_f)), && \text{(7.57a)}\\
&\text{with respect to} && x, y, u, t_f, x_0, &&\\
&\text{subject to} && \text{dynamics of Fig. 7.3 in blue,} && \text{(7.57b)}\\
& && \dot{c} = L(x, u), && \text{(7.57c)}\\
& && x(0) = x_0, \quad c(0) = 0, && \text{(7.57d)}\\
& && g(\dot{x}, x, y, u) \leq 0, && \text{(7.57e)}
\end{aligned}$$

where

$$x = (q, \dot{q}, \omega), \quad y = (h, \lambda, \lambda_n, \sigma), \quad u = \tau, \tag{7.58}$$

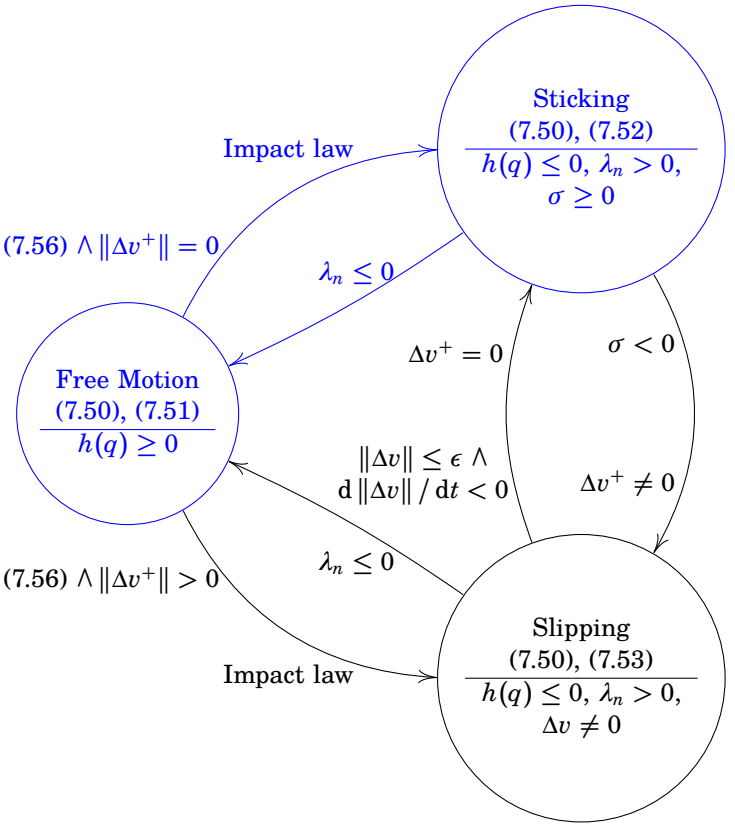


Figure 7.3 The hybrid system representing the ball and finger system. Only the subsystem shown in blue is used for finding optimal trajectories, limiting the possible solutions to free motion and sticking.

are the differential, algebraic, and control variables, respectively, c are additional state variables that model the objective defined by ϕ and L, which we describe further in Sec. 7.4, and g are path inequality constraints, which are described in Sec. 7.4. For simplicity, we assume that at $t = 0$ the system is in free motion, and so the initial state is determined by x_0, which may be chosen as an optimization variable. An arbitrary torque limit of 1 has been imposed.

This general formulation can be used to, e.g., find optimal solutions to the following problems:

- Reach the ball.
- Reorient the ball.

- Rotate the ball with a reference velocity.
- Rotate the ball along different axes as fast as possible.
- Find a periodic solution to keep the ball rotating.

From Hybrid Dynamics to Multiple Phases

The optimization problem (7.57) is not compatible with conventional methods for numerically solving optimal control problems due to the hybrid dynamics. To accurately treat the switching dynamics discussed in Sec. 7.3 in a differentiable way without making smooth approximations—which would lead to unacceptable inaccuracy—we introduce multiple phases as described in, e.g., [Betts, 2010; Becerra, 2010b]. The time horizon $[0, t_f]$ is thus divided into N phases

$$\left[t_0^{(1)}, t_f^{(1)}\right], \ldots, \left[t_0^{(N)}, t_f^{(N)}\right] \tag{7.59}$$

such that

$$t_0^{(1)} = 0, \quad t_f^{(N)} = t_f, \quad t_0^{(i+1)} = t_f^{(i)}, \quad i = 1, \ldots, N-1. \tag{7.60}$$

Within each phase only a single, prescribed mode is active, and the mode is only allowed to switch at the phase boundaries.

For ease of notation, let

$$z^{(i)} := \begin{bmatrix} \dot{x}^{(i)} & x^{(i)} & \dot{c}^{(i)} & c^{(i)} & y^{(i)} & u^{(i)} \end{bmatrix}^T. \tag{7.61}$$

The dynamics within phase i are then described by the non-hybrid DAE

$$F^{(i)}\left(z^{(i)}\right) = 0 \tag{7.62}$$

that corresponds to the mode that has been prescribed to the phase, that is, (7.51) or (7.52) as well as (7.50) and (7.57c).

The phase boundaries $t_0^{(i)}$ and $t_f^{(i)}$ are implicitly determined by event constraints $e^{(i)}$, which correspond to the transition conditions of Fig. 7.3, and are thus decision variables. The validity conditions are enforced by the path constraint $g^{(i)}$. State and control inequality constraints can also be incorporated into $g^{(i)}$. The system variables in the respective phases are connected by linkage constraints $\psi^{(i)}$ corresponding to the reset maps of Fig. 7.3.

The hybrid dynamic optimization problem (7.57) is thus transformed into the non-hybrid, multiple phase problem

$$\begin{aligned}
&\text{minimize} && \phi\big(c^{(N)}(t_f)\big), && \text{(7.63a)}\\
&\text{with respect to} && z^{(i)}, t_f^{(i)}, x_0^{(i)},\\
&\text{subject to} && F^{(i)}(z^{(i)}) = 0, && \text{(7.63b)}\\
& && e_L^{(i)} \le e^{(i)}\Big(z^{(i)}(t_f^{(i)})\Big) \le e_U^{(i)}, && \text{(7.63c)}\\
& && \psi_L \le \psi\Big(z^{(1)}(t_0^{(1)}), z^{(1)}(t_f^{(1)}),\\
& && \qquad\qquad \vdots\\
& && \qquad z^{(N)}(t_0^{(N)}), z^{(N)}(t_f^{(N)})\Big) \le \psi_U, && \text{(7.63d)}\\
& && g^{(i)}(z^{(i)}) \le 0, && \text{(7.63e)}\\
& && i = 1, \ldots, N,
\end{aligned}$$

where the initial conditions (7.57d) have been incorporated into the linkage constraints (7.63d).

Smooth Saturation

Although (7.63) is not hybrid, it is still not suitable for dynamic optimization due to the non-differentiable equations resulting from the saturation function in (7.3). While this can be handled by introducing further phase changes for whenever a junction point of the saturation function is crossed—i.e., when the potential contact point shifts between the fingertip to the side of the finger—this approach is computationally expensive and also makes it more difficult to determine the optimal phase sequence. We thus instead opt to approximate $\mathrm{sat}_L^U(x)$ by noting that

$$\min(a, b) = \frac{1}{2}(a + b - |a - b|), \tag{7.64}$$

$$\max(a, b) = -\min(-a, -b) = \frac{1}{2}(a + b + |a - b|), \tag{7.65}$$

$$\begin{aligned}
\mathrm{sat}_L^U(x) &:= \max(\min(x, U), L)\\
&= \left|\frac{x + U - |x - U|}{4} - \frac{L}{2}\right| + \frac{x + U - |x - U|}{4} + \frac{L}{2}
\end{aligned} \tag{7.66}$$

and then using the approximation

$$|x| \approx \sqrt{x^2 + \epsilon^2}, \tag{7.67}$$

where ϵ is a small number, which we set equal to 0.01. Figure 7.4 compares the smooth approximation with the ideal saturation as well as the hyperbolic

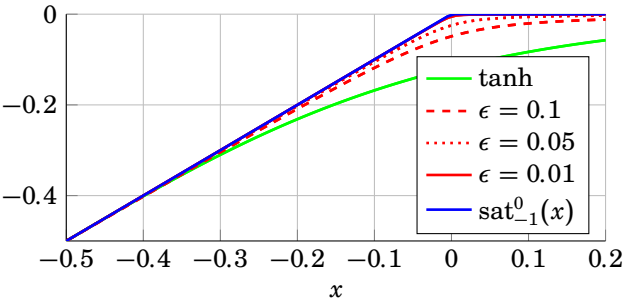


Figure 7.4 Approximation of the saturation function with the lower limit of -1 and the upper limit of 0. The curves have point symmetry around $(0.5, 0.5)$. Hence, only the first quadrant is shown. The green curve approximates the saturation function by scaling and shifting tanh. We observe that for $\epsilon = 0.01$, the smooth approximation of the saturation function is hardly distinguishable from it.

tangent function with an appropriate shifting and scaling. The hyperbolic tangent function is another function commonly used to make smooth approximation. Note that a significantly better approximation by scaling the hyperbolic tangent function is not possible since it requires changing either the slope or saturation limits.

Objective

Out of the various archetype problems mentioned in Sec. 7.4, we will focus on the final one of finding a periodic trajectory that achieves a certain rotation of the ball. We use the objective function

$$\phi(c(t_f)) := \frac{c_1(t_f)}{c_2(t_f)}, \tag{7.68}$$

where we will consider two different ways of defining c_1 and c_2. The first is cost of transport, which can be expressed by the two cost states

$$\dot{c}_1^{\text{COT}} := \|u\|^2, \tag{7.69}$$

which quantifies the cost of control action, and a measure of the amount of rotation in a desired direction

$$\dot{c}_2^{\text{COT}} := -w^{\text{COT}} \left\| \underline{\omega}^{\text{ref}} \right\|^2 + \omega \cdot \hat{\omega}^{\text{ref}}, \tag{7.70}$$

where $\hat{\omega}^{\text{ref}}$ is a unit vector determining the desired direction of angular velocity,

$$\underline{\omega}^{\text{ref}} := \omega - \left(\omega \cdot \hat{\omega}^{\text{ref}}\right) \hat{\omega}^{\text{ref}} \tag{7.71}$$

is the component of the angular velocity perpendicular to the desired direction, and w^{COT} is used to weight the two quantities. The second objective is a more conventional quadratic penalty on the input weighted by w^{L^2} and on the deviation from a reference rotational velocity, ω^{ref}, expressed by the cost states

$$\dot{c}_1^{L^2} := w^{L^2} \|u\|^2 + \left\| \omega - \omega^{\mathrm{ref}} \right\|^2, \tag{7.72}$$

and a state for calculating the total period

$$\dot{c}_2^{L^2} := 1. \tag{7.73}$$

The cost of transport defined here is motivated by the energy efficiency of the rotation and is similar to the energy index used for comparing the efficiency of various transport systems [Shi et al., 2008]. The average rotational speed is thus determined by the optimization. On the other hand, the quadratic objective function is intended for achieving a certain average rotational speed while allowing some trade-off for the average power. The fact that the quadratic cost function favors angular velocities close to the constant reference velocity leads to short periods. This is understood by noting that the speed of the ball cannot be influenced in the free motion and drops rapidly.

Periodicity, i.e., $x_0^{(1)} = x^{(N)}(t_f)$, is enforced via the linkage constraint ψ.

7.5 Implementation

The equations of Appendix A.1 and A.2 have been derived with the use of Maple. The multiphase problem (7.63) is implemented in PSOPT [Becerra, 2010a]. The Legendre-Gauss pseudospectral collocation of PSOPT is used to transcribe the problem into a (twice continuously differentiable) nonlinear program (NLP) with 30 collocation points for each phase. The NLP is solved with IPOPT [Wächter and Biegler, 2006] and the linear solver MA27 [HSL, 2016].

Simulation of the system including slipping is performed using Modelica [Fritzson, 2015] and Dymola [Dymola, 2016]. Dymola does currently not support hybrid systems with variable index, although there has been recent work to resolve this [Mattsson et al., 2015]. One possible way to manually resolve this is to index reduce the high-index modes, but this is challenging for the considered system due to the impossibility of statically selecting suitable state variables. Hence, we instead adopt a more cumbersome approach where three instances of the system are simulated simultaneously, one low-index for free motion and two high-index (with dynamic state selection [Mattsson et al., 2000]) for modes in contact. A correct

system model can thus be created by appropriate communication between the three instances during mode switching.

The trajectories from both optimization and simulation are finally imported into MATLAB for visualization. For 3D visualization of the finger and its movements, Peter Corke's robotic toolbox [Corke, 2011] is used.

7.6 Results

This section presents results for the problem of finding a periodic solution for rotating the ball. We first present optimal trajectories obtained by neglecting slipping, and then briefly show the solutions obtained by simulating the system with the open-loop optimal inputs while taking into account slipping.

Optimal Control

We consider the two objectives discussed in Sec. 7.4. We divide each period into three phases: free motion, sticking, and finally free motion again with the constraint $x^{(3)}(t_f) = x^{(1)}(0)$. The initial state as well as the terminal and switching times $t_f^{(i)}$ are free. We also impose the initial constraint $h(0) \geq 1.3r$ for robustness, i.e., the finger should go sufficiently far away from the ball before making a new contact. The weights w^{COT} and w^{L^2} are optionally set to 10 and 1, respectively. As the desired direction of angular velocity $\hat{\omega}^{\mathrm{ref}}$, we use $1/\sqrt{2}(1,-1,0)^T$ for cost of transport and we use $\omega^{\mathrm{ref}} = (0,0,2.2)^T$ as the angular velocity reference for the quadratic objective.

A simple initial guess is sufficient to numerically solve the problem. The initial guess consists of a constant non-zero $q \equiv q^0$ and $t_f^{(i)} = i$, with all the remaining variables being 0.

We present the result of three experiments. In the first two experiments, we solve the problem of optimizing the COT. The difference between the experiments is that in the first one we include no constraint on the joint angles, while in the second one the constraint that $q_i \geq 0$ for $i \in \{2,3\}$ is enforced in order to comply with the limitations of the human finger. Fig. 7.5 illustrates the optimal path obtained for the robotic finger. In Fig. 7.6, we see the resulting optimal angular velocities of the ball and switching times. The discontinuities at the impact point are visible.

The minimum distance h between the finger and the ball is shown in Fig. 7.7. Moreover, the distance between the contact point and fingertip is plotted. As we see, the contact point is somewhere on the link and shifts slightly during the rotation. The required torques and interaction forces are given in Figs. 7.8 and 7.9, respectively.

Figure 7.10 illustrates the human finger in its initial position. In this case, the fingertip remains always the closest point to ball. The path ob-

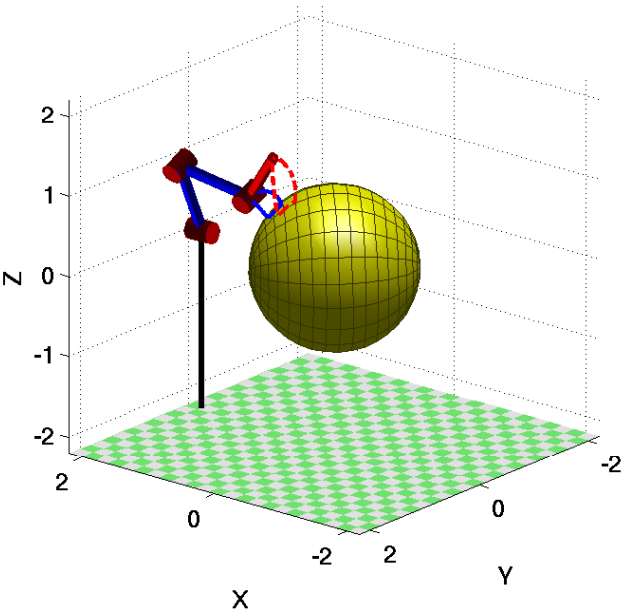


Figure 7.5 Visualization of the ball and finger system. The blue curve shows the optimal path of p_c for COT when there is no joint limit. The red dashed curve corresponds the optimal path for the fingertip.

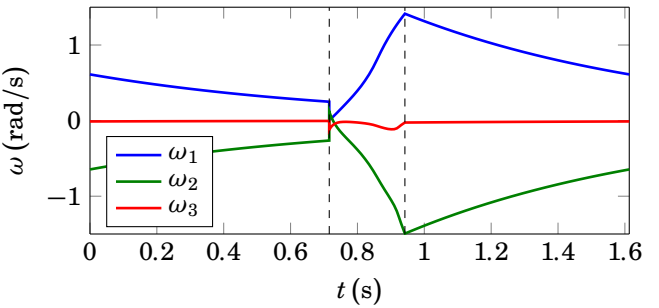


Figure 7.6 The optimal solution for COT: the angular velocity of the ball ω. The vertical lines show the switching time for the phases.

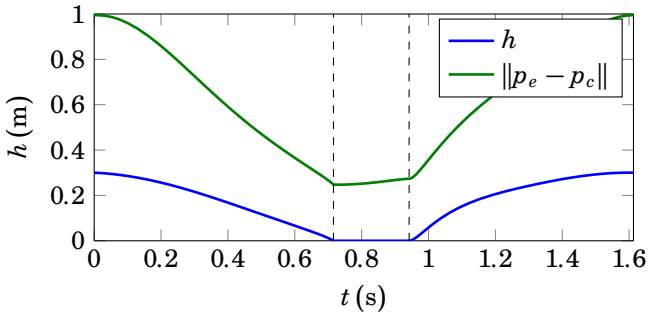


Figure 7.7 The minimum distance to the ball, h, and the distance of the contact point to the fingertip. The finger is in contact with the ball in the interval between the vertical lines. The side of the finger is used for rotating the ball and the contact point shifts slightly during rotation.

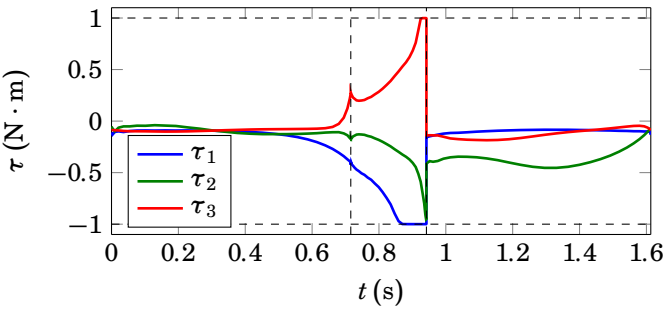


Figure 7.8 The optimal control signal for minimizing COT: The vertical lines show the switching time for the phases. Towards the end of the contact phase, the bounds on the control signals are active.

tained from the movement of the closest point is shown in blue. The paths of the closest points of the human finger as well as the robotic finger are compared in Fig. 7.11.

Assuming the COT, the robotic finger can rotate the ball in the desired direction more efficiently. It uses the side of the finger, while the human finger contacts with the fingertip only.

Figure 7.12 shows the finger in its initial position and the path obtained for rotating the ball around the z-axis using the quadratic objective function. The angular velocity obtained is illustrated in Fig. 7.13. As we can see, the average angular velocity matches the desired ω_{ref}. The period of the solution compared to Fig. 7.6 is shorter. The minimum distance as a function of time and distance of the contact point to the fingertip is shown in Fig. 7.14. In this

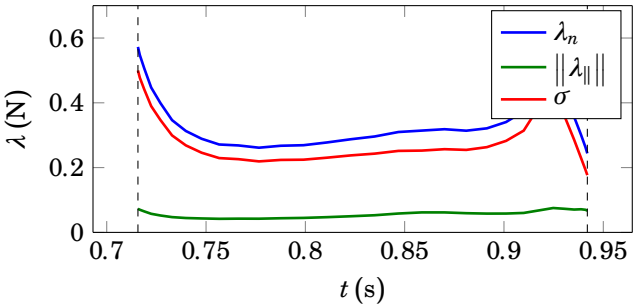


Figure 7.9 Interaction forces for COT: The tangential and normal component of the interaction are shown. In this case, there is always some margin to slipping.

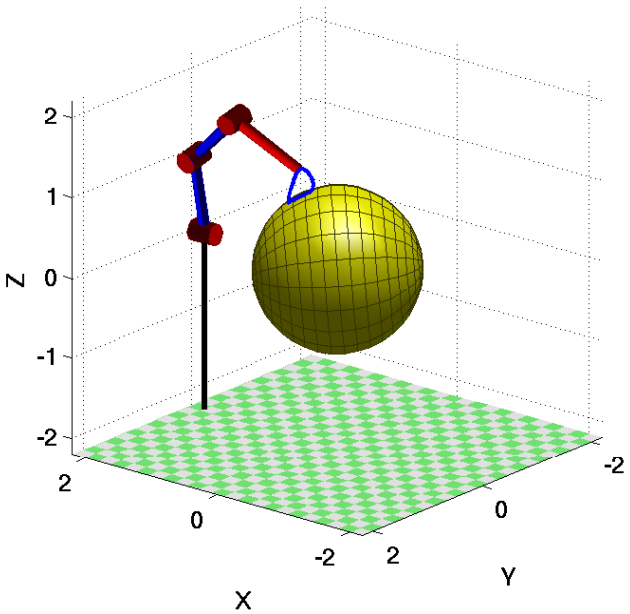


Figure 7.10 Visualization of the ball and finger system. The blue curve shows the optimal path of p_c complying with human joint limits for COT. The contact point is always the fingertip.

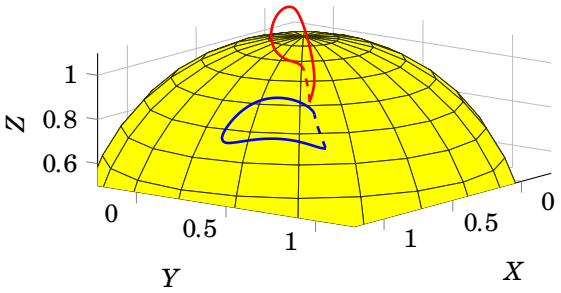


Figure 7.11 The paths of p_c obtained using the COT optimization criterion. The red and blue curves correspond to human and robotic fingers, respectively. The parts of the paths in contact with the ball are dashed.

solution, only the fingertip is used for rotating the ball. Finally, the torques and interaction forces are depicted in Figs. 7.15 and 7.16, respectively. As we see, the constraint on the friction cone is always active during contact. Hence, the motion is always on the verge of slipping.

Open-Loop Simulation

We now use the open-loop optimal torques for the COT criterion without non-negativity bounds on the joints from Fig. 7.8 to simulate the full system in Fig. 7.3 in Dymola. The resulting angular velocities are shown in Fig. 7.17 and compared with the optimal angular velocities. We see that open-loop angular velocities match well at the beginning, but drift from the optimum as time passes. There is a short window of time immediately following collision in which slipping occurs, which is almost always the case when employing Newton's impact law. The dynamics quickly switch to sticking because of the high interaction forces. There is a jump in the post-impact velocities in the optimization solution, while using the Newton's impact law, the velocities are continuous in the simulation result.

7.7 Extension to Multifinger Setup

The equations of motion for extra fingers can be added to the model. The procedure is almost trivial by considering (7.50), i.e., for each finger, the equations are repeated with new variables and all the interaction forces are summed up to calculate the effect on the ball. Accordingly, the optimization method described in Sec. 7.4 is still applicable. However, this approach is limited by the fact that the order of phases is not known in advance. As more fingers are added to the model, the combination of phases grows exponentially. A solution to this problem is to use a hierarchical planning

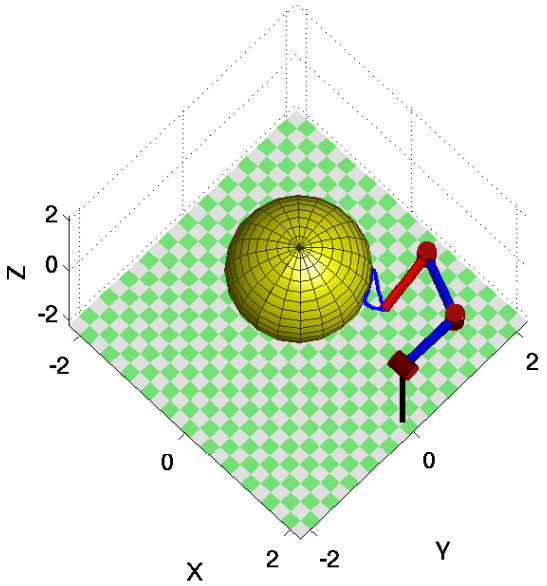


Figure 7.12 Visualization of the ball and finger system. The blue curve shows the optimal path of p_c for the quadratic cost. The objective is to rotate the ball around the z-axis.

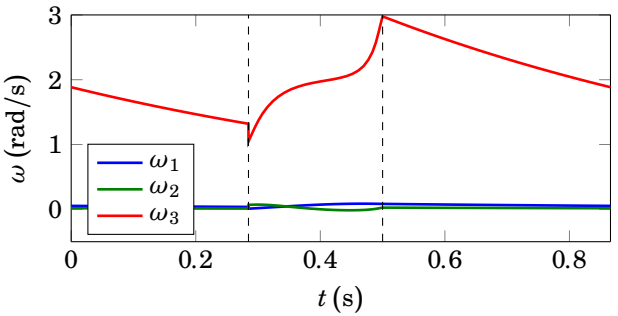


Figure 7.13 The optimal solution to ϕ_{L^2}: angular velocity of the ball ω and the minimum distance to the last link h. The vertical lines show the switching time for the phases.

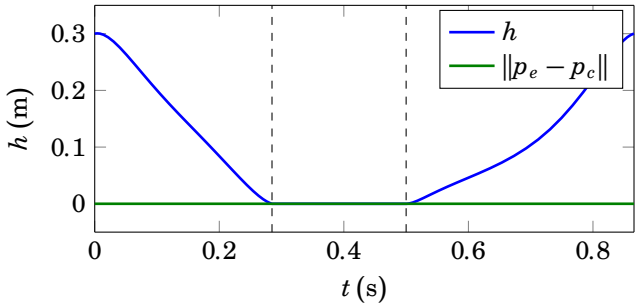


Figure 7.14 The minimum distance to the ball, h, and the distance of the contact point to the fingertip. The finger is in contact with the ball in the interval between the vertical lines. Only the fingertip is used for rotating the ball.

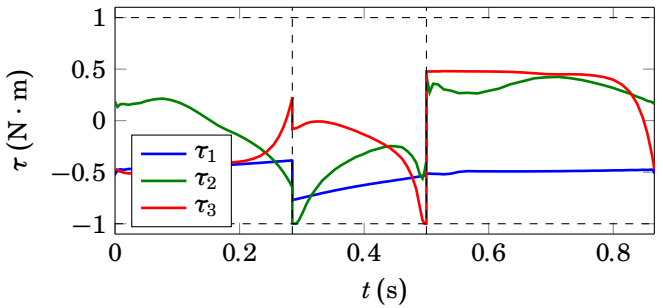


Figure 7.15 The optimal control signal for minimizing ϕ_{L^2}: The vertical lines show the switching time for the phases. Note that, the bounds on the control signal are only active when making contact or breaking it.

scheme where in the highest level the finger gating and consequently the phases are determined. For finger gating, often a more abstract and less detailed model suffices [Saut et al., 2011].

In this section, we study the sequence of finger movements for a three-finger gripper for rotating a ball without letting it fall. A reinforcement learning (RL) type of algorithm [Sutton and Barto, 1998] is used to find an optimal coordination between fingers to achieve the maximum average velocity of the rotation of a ball in the counterclockwise direction. The obtained sequence defines the required phases for finding trajectories for a system with three fingers according to the method of Sec. 7.4.

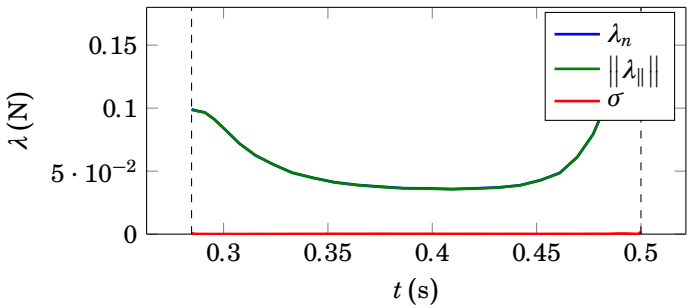


Figure 7.16 Interaction forces for the quadratic cost: The tangential and normal component of the interaction are shown. In this case, there is no margin to slipping.

Methods

An abstract model of the process is developed wherein the states and the actions are high-level representations. There are several ways to obtain and describe such models. Here, we use a discrete model which is based on the high-level understanding of the significant events.

In every time step, each finger can independently perform an action. The fingers have a limited reach and speed. The abstract actions are staying still, closing (making contact) or opening (breaking contact), and moving left or right with respect to the center of the ball. Denoting the action for finger i by u^i, the actions and their effects can be encoded as

u^i	Effect
0	Null
1	Close or Open
2	Left or Right

Given that there are three fingers, there are totally $3^3 = 27$ possible actions.

We define the abstract state of each finger as its relative position to the ball. For example, if r_0 denotes the radius of the ball and ϕ_0 the nominal position of a finger, using the polar coordinate we can describe

$$\text{Left Open} = \{(r, \phi) | r > r_0 \wedge 0 < \phi - \phi_0 \leq 15\pi/180\}. \tag{7.74}$$

Accordingly, for each finger we define the following states:

x^i	Definition
0	Right Open
1	Right Close
2	Left Open
3	Left Close

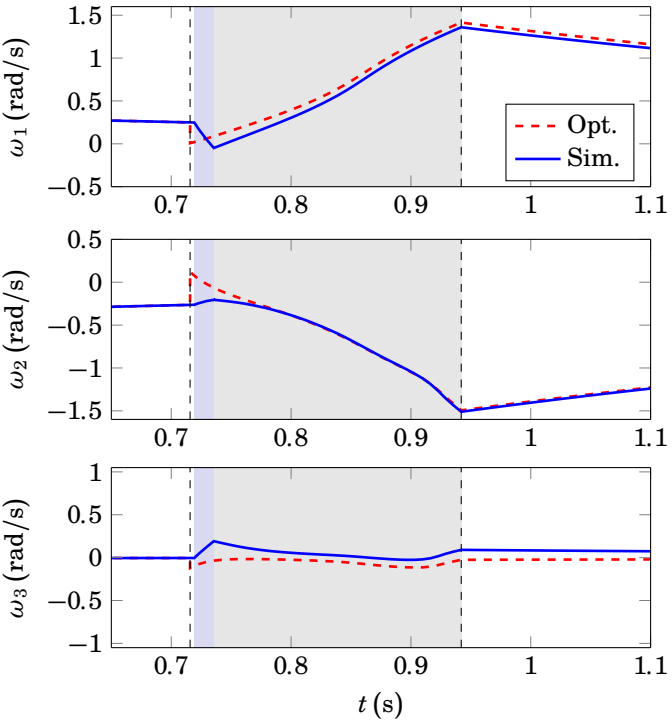


Figure 7.17 The open-loop simulation of the complete system in Fig. 7.3 using the control signals in Fig. 7.8. The angular velocities of the ball (solid blue curves) are compared with the optimization result (red dashed curves). The vertical lines show the switching times obtained from the optimization. The time regions corresponding to the slipping and sticking modes of the simulated result are shaded in purple and gray, respectively. We see that the contact times differ slightly. However, there is no difference in the time that the finger leaves the ball.

The total number of states is $4^3 = 64$.

Given the definitions of the states and the actions, the system dynamics are nonlinear

$$x(k+1) = f(x(k), u(k)), \tag{7.75}$$

where

$$f_i = \mathrm{shl}(\mathrm{XOR}(\mathrm{shr}(x_i \wedge 10_2), \mathrm{shr}(u_i \wedge 10_2))) \\ \wedge \mathrm{XOR}(x_i \wedge 01_2, u_i \wedge 01_2). \tag{7.76}$$

Here $\wedge$ denotes bitwise AND operator and XOR$(\cdot,\cdot)$ denotes exclusive OR operation, shr and shl denote shift to right and shift to left operations, respectively. The numbers with subscript 2 are given in the binary system. The system dynamics can be understood by paying attention to how the states and the actions are encoded. The lower bit of x_i determines if the finger is closed or open while the upper bit determines the position. The Close or Open action, with lower bit set to one ($1 = 01_2$), therefore just flips the lower bit of the state, while the Left or Right action, with higher bit set to one ($2 = 10_2$), flips the higher bit of the state.

The ball falls if it is in contact with one or no fingers and it will rotate in the direction determined by the closed fingers when no finger is resisting the rotation or work against it. Otherwise, it remains still. The states of the ball are encoded according to the following table:

x^b	Definition
0	Dropped
1	Rotating CW
2	Rotating CCW
3	Still

We would like to design a reward function $r(x, u)$ that encourages counterclockwise (ccw) and discourages clockwise (cw) rotations. Moreover, unnecessary movements and dropping must be punished. Consequently, we define:

$$r(x,u) = \begin{cases} 1, & \text{ccw rotation} \\ -1, & \text{cw rotation} \\ -1, & \text{one or 2 closed fingers are still, while the rest are moving} \\ -2, & \text{all closed fingers move but not all in the same direction} \\ -10, & \text{unstable grip, i.e., less than 2 fingers in contact} \end{cases} \tag{7.77}$$

For a given policy Π and an initial state x_k, the objective is to maximize the cumulative discounted reward according to

$$V_\Pi(x_k) = \sum_{m=k}^{\infty} \gamma^{m-k} r(x_m, u_m), \qquad 0 < \gamma < 1. \tag{7.78}$$

We use Q-learning to solve the problem with the following update equation [Watkins and Dayan, 1992]

$$Q^{i+1}(x_k, u_k) = Q^i(x_k, u_k) + \eta \left[r + \gamma Q^i(x_{k+1}, \Pi(x_{k+1})) - Q^i(x_k, u_k) \right] \tag{7.79}$$

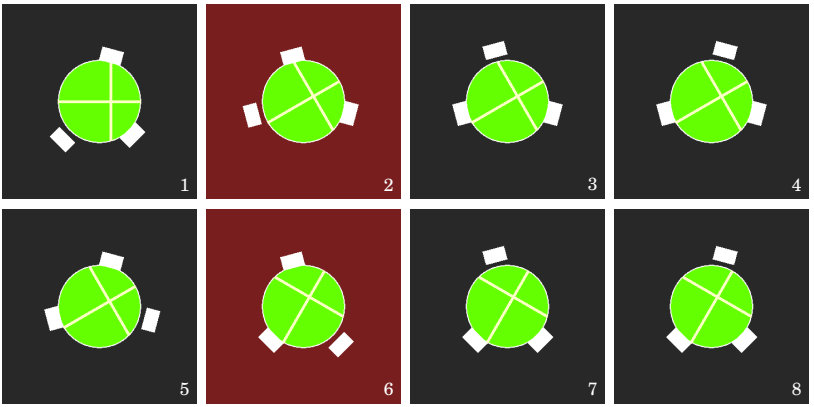


Figure 7.18 The optimal sequence with a duty cycle of 2/8: Two steps out of eight cause a ccw rotation. White boxes represent the fingers and the green circle in the middle corresponds to a ball. The yellow lines on the ball are to make the rotation visible. The states right after a rotation are highlighted in red.

The optimal policy is obtained by calculating

$$\Pi(x) = \arg\max_u Q(x, u). \tag{7.80}$$

In the learning phase, we use an ϵ-greedy policy. It means that with probability ϵ instead of

$$u_k = \Pi(x_k), \tag{7.81}$$

a random action is taken.

Results

We chose the following parameters for the Q-learning algorithm

$$\gamma = 0.9,\ \eta = 0.3,\ \epsilon = 0.01. \tag{7.82}$$

After approximately 100, 000 iterations, the optimal sequence was found. Figure 7.18 illustrates the sequence. Starting from $x_0^T = [0, 0, 0]$, i.e., all fingers in the Right Open position, the optimal policy results in:

k	x_k^T	u_k^T		x_{k+1}^T	r_{k+1}
0	[0, 0, 0]	[1, 1, 2]	→	[1, 1, 2]	0
1	[1, 1, 2]	[2, 2, 2]	→	[3, 3, 0]	1
2	[3, 3, 0]	[1, 0, 1]	→	[2, 3, 1]	0
3	[2, 3, 1]	[2, 0, 0]	→	[0, 3, 1]	0
4	[0, 3, 1]	[1, 1, 0]	→	[1, 2, 1]	0
5	[1, 2, 1]	[2, 2, 2]	→	[3, 0, 3]	1
6	[3, 0, 3]	[1, 1, 0]	→	[2, 1, 3]	0
7	[2, 1, 3]	[2, 0, 0]	→	[0, 1, 3]	0
8	[0, 1, 3]	[1, 0, 1]	→	[1, 1, 2]	0

As can be observed, the optimal sequence has a duty cycle of 2/8, i.e., 2 rotations in a sequence of 8 actions.

Note that the period does not determine the average velocity and the duty cycle is therefore the correct indicator. Figure 7.19 illustrates an example of a non-optimal sequence with 1/6 duty cycle. Note that in this case the period is 25% shorter but the cumulative reward is half compared to the optimal sequence. Starting from $x_0^T = [0, 0, 0]$ and following the policy, the state trajectory, the inputs, and the rewards are as follows:

k	x_k^T	u_k^T		x_{k+1}^T	r_{k+1}
0	[0, 0, 0]	[2, 1, 1]	→	[2, 1, 1]	0
1	[2, 1, 1]	[0, 2, 2]	→	[2, 3, 3]	1
2	[2, 3, 3]	[1, 0, 1]	→	[3, 3, 2]	0
3	[3, 3, 2]	[0, 0, 2]	→	[3, 3, 0]	0
4	[3, 3, 0]	[0, 1, 1]	→	[3, 2, 1]	0
5	[3, 2, 1]	[0, 2, 0]	→	[3, 0, 1]	0
6	[3, 0, 1]	[1, 1, 0]	→	[2, 1, 1]	0

7.8 Discussion

The considered system in this chapter has several modes, each described with a different set of equations. In contrast to some of the results for finding optimal trajectories reported in the literature (see for example [Kumar et al., 2014; Rombokas et al., 2013]), in our approach the connection between different modes happens through phases. Thus, it is not required to compromise the optimal solution by biasing the cost function toward making contact. On the other hand, we have fixed the order of the phases. This can be relaxed by trying all the combinations. For more complex systems, a hierarchical planning scheme similar to the one discussed in Sec. 7.7 can be used in order to determine the phases.

There are several possible approaches to treating hybrid dynamics in dynamic optimization other than the multiphase framework. A natural al-

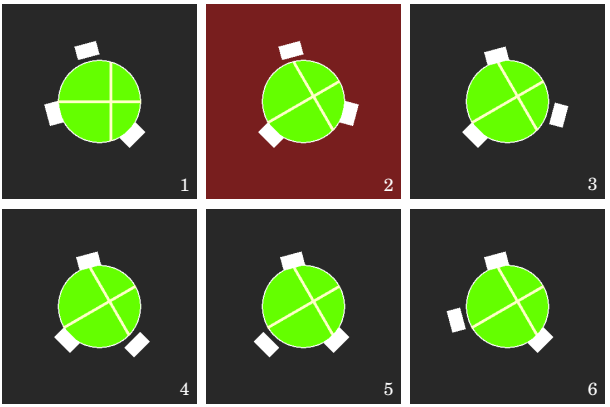


Figure 7.19 A non-optimal sequence with a duty cycle of 1/6: One steps out of six cause a ccw rotation. White boxes represent the fingers and the green circle in the middle corresponds to a ball. The yellow lines on the ball are to make the rotation visible.

ternative in the considered case is the incorporation of the complementarity constraints [Baumrucker and Biegler, 2009] in the optimization problem. However, it is not always easy to rewrite the non-smooth behavior using complementarity conditions, e.g., switching between slipping and sticking when the friction coefficients are not the same. Other possible approaches are mixed-integer nonlinear programming based on branch-and-bound [Belotti et al., 2009] or sum-up-rounding [Kirches and Lenders, 2016] or widely applicable derivative-free methods such as genetic algorithms [Thieriot et al., 2011]. The various approaches each have their respective strengths and weaknesses. The strength of the multiphase framework is the efficiency of the required computations, which may enable online solution of these problems. A significant drawback is that the phase sequence needs to be determined a priori.

In optimization, it is not always suitable to neglect the slipping completely. Ignoring the slipping works well when the interaction forces are high. However, when the slipping margin σ is small, it becomes more important to consider the effects of slipping. If slipping is neglected, it is also important to ensure that the collision itself does not give rise to slipping. When using the Newton's impact law, this is possible by constraining the relative tangential velocity at the collision point to be zero. This will guarantee a transition to the sticking mode upon collision, though it might not be the optimal solution. Better solutions can however be obtained by considering a more sophisticated impact law than Newton's, see Sec. 7.2, and then constraining the resulting impulse ι so that the friction is sufficient to

support the sticking mode.

Another option for handling impacts is to add flexible components such as a non-rigid ball or a padded arm. This leads to a smooth approximation of discontinuities. For example, the normal force can be considered proportional to the depth of penetration of p_c, i.e., $\lambda_n = -k \min(h(q), 0)n$ could be employed. At the same time, in order to get realistic behavior the model tends to become stiff.

To account for the transient forces in the optimization, the resulting impulse can be calculated. Furthermore, to avoid slipping right after collision, we have to constrain ι such that the friction is sufficient to support the sticking mode. We could also make sure that the relative tangential velocity at the collision point is zero by introducing a constraint on it. This will guarantee a transition to the sticking mode upon collision.

Open-loop control is generally not satisfactory, even if slipping can be neglected. Feedback is needed to handle model uncertainties, such as inaccurate collision model, as well as discretization errors. Different controllers depending on the mode of the system may be employed. Specifically, in free motion, a position controller with feedforward torques can be used. As soon as the finger comes into contact with the ball, the controller can be switched to a hybrid position/force controller. To maintain the sticking condition, the normal force should be regulated while the velocity in the tangential direction is being tracked.

In simulation, inconsistent solutions due to the Painlevé paradox [Brogliato, 2012] may arise when the kinetic friction constant between the ball and the finger is large. However, this will not be an issue in the optimization because of the infeasibility of such solutions. The optimization results presented in Sec. 7.6 were obtained using simple initial guesses. However, warm starting the optimization from previous runs with simplified cost functions can often be beneficial. This way it is possible to direct the solution gradually to the region of interest.

The path constraint related to the validity condition $h \geq 0$ is difficult to treat numerically. In phase 1, the natural formulation is $h^{(1)} \geq 0$ and $h^{(1)}(t_f) = 0$. However, such a formulation violates constraint qualifications, leading to unbounded dual variables [Nocedal and Wright, 2006]. Ideally, this would be handled by only enforcing $h^{(1)} \geq 0$ in all but the last collocation point of phase 1, which instead is determined by the event constraint $h^{(1)}(t_f^{(1)}) = 0$, but this is not readily supported by PSOPT. Hence, we work around this issue by replacing the event constraint by $0 \leq h^{(1)}(t_f) \leq \epsilon := 10^{-4}$. Further problems related to this are encountered in phase 3. In phase 2, $\dot{h} = 0$ is a solution invariant, which is however not preserved during discretization. Numerical drift-off can thus lead to $h^{(2)}(t_f) < 0$ (enforcing $h^{(2)} \geq 0$ would once again lead to violation of con-

straint qualifications). To circumvent this, we replace the validity condition of phase 3 by $h^{(3)} \geq -\epsilon$.

In Sec. 7.7, we could have defined actions and states differently to make the notion of the state abstraction more clear (e.g., by thresholding the actual inputs and states), which also results in simpler state transitions (linear plus saturation). However, this adds unnecessary possibilities (e.g., trying to open a finger while it is open) hence increasing the dimension of the Q-function without adding any benefit for finding the optimal solution.

The speed of convergence to the optimal solution depends on the choice of the parameters and the realization of the ϵ-policy. In general, there is no guarantee to converge to the optimal solution. In our problem, due to its symmetry, the solution is not even unique. Therefore, different runs of the algorithm might converge to different optimal solutions.

The proposed framework is not robust with respect to avoiding contact in the free motion mode, as it allows for free motion arbitrarily close to the ball. A simple remedy for the considered examples is to also enforce $\dot{h} \leq -\epsilon$ in the first phase and $\dot{h} \geq \epsilon$ in the third phase, which however may sacrifice optimality. Future work is to design a more sophisticated solution by considering robust approaches such as those in [Ben-Tal et al., 2009; Mordatch et al., 2015]. We also consider implementing a controller and evaluating the optimal trajectories in the simulation with the complete model.

7.9 Conclusion

In this chapter, we propose a ball-and-finger system as a realistic example to study common robotic tasks involving *interaction* and varying dynamics. The result of modeling is a hybrid, variable-index differential-algebraic equation system. To find the optimal trajectories for performing a task, we took the integrated approach, i.e., there is no separate process for finding the required path. A multiphase formulation was used to handle changes in the dynamics in each mode. The benefit of this approach is that there is no need to compromise the desired objective function in order to make contact.

As the result of the optimization, we find not only the trajectory for the motion but also the interaction forces. This is an important feature considering the fact that robots are often employed for manipulation tasks, which involves interaction with other objects. The extension of our model with several fingers opens up the opportunity to perform automatic motion planning for dexterous manipulation, e.g., for assembly tasks.

8

Iterative Learning Control

8.1 Introduction

The main application of iterative learning control (ILC) is to improve the reference tracking performance of a system. In order to reduce the tracking error, the control signal to the system is adjusted in each iteration by using feedback information from previous iterations. In effect, ILC finds an approximate system inverse for a specific reference [Moore et al., 1989]. An advantage of ILC is that it does not require an explicit model of the transfer function or even linearity of the system for finding the inverse. Instead, it often uses the actual system as a part of the algorithm. ILC has found successful applications in many different fields [Ahn et al., 2007; Freeman et al., 2012; Sörnmo et al., 2016], where accurate models of the system and disturbances are difficult to obtain.

While the frequency domain is the preferred approach for filter design and analysis of linear ILC [Wang et al., 2014], the widely used convergence criterion, see (8.4) below, applies only to strictly monotone convergence of the algorithm (the 2-norm of the error between the current control signal and its final value strictly decreases in each iteration). Moreover, it is not theoretically clear to what extent the frequency criterion is applicable to a practical ILC system where each iteration runs only over a finite-time interval and to ILC systems with non-causal filters. To motivate this study, we demonstrate examples for which the ILC converges but the classical frequency condition cannot provide any indication of the convergence property. Our analysis gives an explanation for this mode of convergence.

We extend the work of [Norrlöf and Gunnarsson, 2002] by introducing a less conservative criterion, hence reducing the gap between the existing time-domain and frequency-domain criteria. We also provide an analysis of the transient behavior of the algorithm, which proves useful when the convergence is not monotone. The contributions of this chapter can be summarized as follows:

- Analysis of "convergence on finite-time interval" motivated by practical ILC where the trial length is finite.

- A less conservative frequency domain convergence criterion than the one by [Norrlöf and Gunnarsson, 2002] is derived (see Theorem 14)

$$\inf_{\rho>0} \sup_{\omega} \left| G(\rho e^{i\omega}) \right| < 1.$$

  The criterion is applicable to ILC systems with *causal* as well as *non-causal* filters and includes the classical result of strictly monotone convergence as a special case.

- The connection between time-domain and frequency-domain criteria is established in a rigorous manner using Toeplitz operators.

- A frequency domain tool for understanding the transient behavior of ILC—i.e., the wave of convergence/divergence—is introduced.

- A strategy to limit the growth of the transient errors when the convergence is not monotone is proposed.

Previous Work

ILC is a two-dimensional process, in the sense that the dynamics are indexed by both time and iteration variables [Kurek and Zaremba, 1993]. A standard approach to analysis of linear and a certain class of nonlinear ILC algorithms relies on the lifted-system framework, i.e., considering a time series as a vector [Bristow et al., 2006]. [Norrlöf, 2000] has extensively studied the theory and applications of linear ILC. Time-domain criteria as well as a classical frequency-domain criterion for the convergence of the linear ILC algorithm have been derived by [Norrlöf and Gunnarsson, 2002].

There have been many attempts to understand and improve the convergence properties of the linear ILC. [Longman and Huang, 2002] have noted that the algorithm might practically diverge after an initial substantial decay of the tracking error. [Elci et al., 2002] have introduced a non-causal filter, namely a zero-phase filter, in the algorithm to improve the transient behavior. The transient properties of the convergence have been studied in more detail by [Longman and Huang, 2002] and [Wang et al., 2014]. [Longman, 2000] and [Norrlöf and Gunnarsson, 2002] have commented on the potential convergence of the algorithm despite a transient growth of the norm of the error, i.e., when the classical frequency condition is not fulfilled.

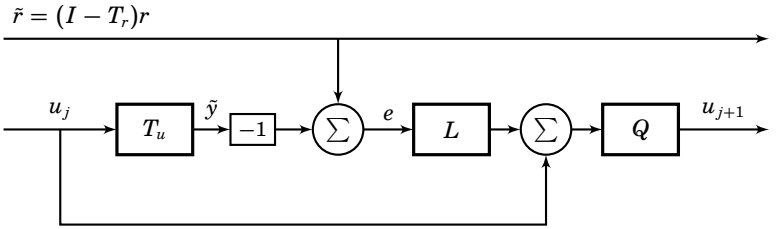


Figure 8.1 Block diagram of an iterative learning controller. Here, $\tilde{y} = y - T_r r$.

Problem Description

A general form of the discrete linear first-order ILC algorithm is

$$y_j = T_r r + T_u u_j \tag{8.1}$$
$$e_j = r - y_j \tag{8.2}$$
$$u_j = Q\left(u_{j-1} + L e_{j-1}\right); \tag{8.3}$$

see [Norrlöf, 2000]. Here $j \in \mathbb{Z}_{\geq 0}$ is a non-negative integer iteration index, r, y_j, and $e_j \in \ell_2$ are the reference, output, and tracking error signals, respectively, $u_j \in \ell_2$ is the control signal. The stable systems from reference to output and control signal to output are denoted by T_r and T_u, respectively, and Q and L are filters to be designed. The choice of u_0 is free. Figure 8.1 depicts the ILC algorithm. Note that in practice the trial length is finite, i.e., the system is stopped after N samples and signal values at time $n \in \{0, \ldots, N-1\}$ are stored. The filters Q and L do not need to be causal since they operate on the signals of the previous iteration.

Let us define $G(e^{i\omega}) := Q(e^{i\omega})(1 - L(e^{i\omega})T_u(e^{i\omega}))$. The classical sufficient condition for strictly monotone convergence of ILC requires that (see for example [Norrlöf and Gunnarsson, 2002])

$$|G(e^{i\omega})| < 1, \ \ \forall \omega \in [0, 2\pi), \tag{8.4}$$

where $L(e^{i\omega})$, $T_u(e^{i\omega})$, and $Q(e^{i\omega})$ are the frequency representations of the respective filters.

Given the definition of the ILC algorithm in (8.1)–(8.3) and the fact that each iteration runs only over a finite-time interval, $n \in \{0, \ldots, N-1\}$, our purpose is to find less restrictive conditions for G that guarantee the convergence of the algorithm, i.e., that the limits $u_j \to u_\infty$ and $e_j \to e_\infty$ exist for the finite trial length.

The rest of this chapter is organized as follows: In Sec. 8.2, we present a motivating example for which the ILC converges but the classical condition cannot provide any indication of the convergence property. Section 8.3

clarifies the notation and summarizes the mathematical background. The iteration-domain dynamics for ILC are derived in Sec. 8.4 before we delve into the issue of convergence. Section 8.5 starts with a formal definition of convergence for iterative procedures and states our convergence results. In Sec. 8.6, practical aspects concerning the transient behavior of ILC when the convergence is non-monotone is discussed. We propose qualitative measures that characterize the convergence, and discuss the gap between the time- and frequency-domain criteria in Sec. 8.9. We draw conclusions in Sec. 8.10.

8.2 Motivating Example

Let us consider the following transfer functions

$$T_u(s) = \frac{1}{(s+1)(s^2+0.8s+16)}, \quad T_r(s) = 0, \tag{8.5}$$

$$Q(s) = \frac{10}{s+10}, \quad L_d(z) = 10k(1-0.9z^{-1})z^a. \tag{8.6}$$

We discretize (8.5) and filter $Q(s)$ by the zero-order-hold (ZOH) method [Åström and Wittenmark, 2011] with sampling time $h = 0.1$ s. Figure 8.2 compares the time responses of the systems corresponding to two ILC scenarios where in 1) $k = 0.8$, $a = 5$ (System I) and in scenario 2) $k = 0.5$, $a = 8$ (System II). After discretization, Q is implemented as a zero-phase filter and hence we get

$$G(e^{\mathrm{i}\omega}) = Q_d(e^{\mathrm{i}\omega})Q_d(e^{-\mathrm{i}\omega})\left(1 - L_d(e^{\mathrm{i}\omega})T_{ud}(e^{\mathrm{i}\omega})\right). \tag{8.7}$$

In Fig. 8.3, the Bode plots for $G(e^{\mathrm{i}\omega})$ are illustrated. We see that in both scenarios the condition $|G(e^{\mathrm{i}\omega})| < 1$ is violated. Nevertheless, System I appears to converge, at least for the time region of interest, while System II does not. The Bode diagrams corresponding to convergent and non-convergent scenarios may seem counterintuitive at first glance since the one with the highest peak in the gain $|G(e^{i\omega})|$ corresponds to the convergent case.

Our result in Theorem 14 later in this chapter explains the situation and says that if there exists a $\rho > 0$ such that $\sup_\omega |G(\rho e^{\mathrm{i}\omega})| < 1$, then we have convergence in the sense that u_∞ and e_∞ exist on the finite interval $[0, \cdots, N)$. In Fig. 8.5, where $\sup_\omega |G(\rho e^{i\omega})|$ is plotted against ρ, it can be seen that the curve for System I goes below 1 for some ρ and thus the ILC algorithm converges.

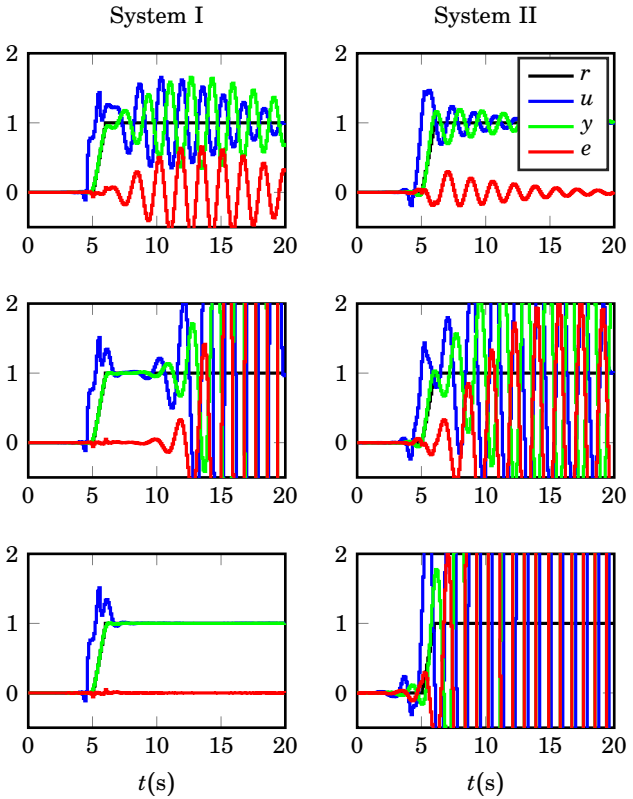


Figure 8.2 The 4th, 12th, and 50th iterations for example (8.5)–(8.6): The left column shows System I (blue curve in Fig. 8.3) and the right column shows System II (green curve in Fig. 8.3). The black, green, blue, and red curves correspond to the reference r, output y, control signal u, and error e, respectively. In the left column, the error signal looks like a growing wave which moves toward plus infinity as the iteration number increases. However, in the right column the wave does not move and the signals grow unbounded in the time region of the trial.

8.3 Background and Preliminaries

Some useful results from systems theory are presented in this section. Familiar readers may skip to the end of the section, where a short summary of the notations used in this chapter is given. We define

$$\ell_p := \left\{ f : \mathbb{Z} \to \mathbb{R} \;\middle|\; \|f\|_{\ell_p}^p := \sum_{n=-\infty}^{\infty} |f(n)|^p < \infty \right\},$$

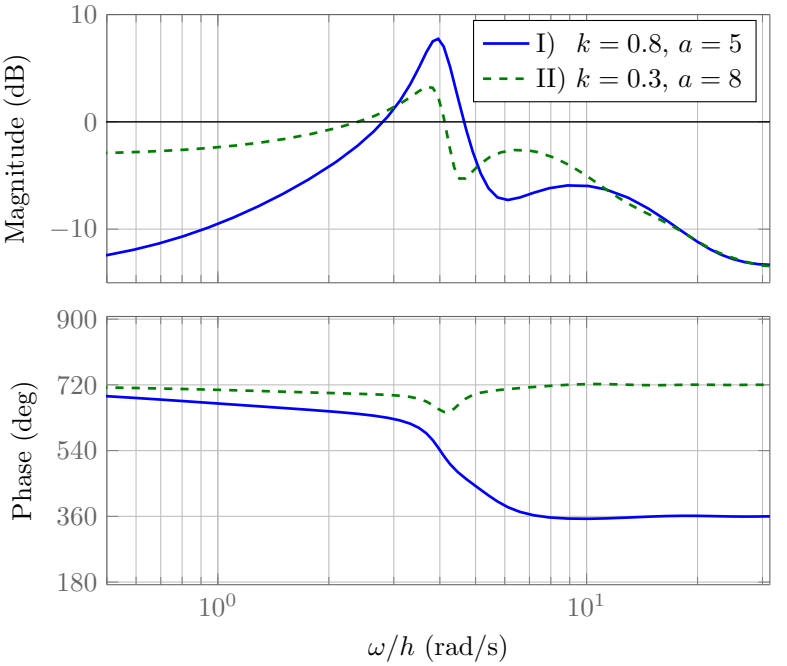


Figure 8.3 Comparison of Bode diagrams for $G(e^{i\omega})$: The frequency response of System I (left column in Fig. 8.2) is in blue and System II in green (right column in Fig. 8.2), respectively. Both systems violate (8.4).

where $|\cdot|$ denotes the absolute value. A linear operator $G : \ell_2 \to \ell_2$ is said to be *bounded* if

$$\|G\|_{\ell_2 \to \ell_2} := \sup_{\|x\|_{\ell_2}=1} \|Gx\|_{\ell_2} < \infty.$$

The product of $x \in \ell_2$ and $y \in \ell_2$ is defined as

$$(xy)[n] := x[n]y[n].$$

Given an $x : \mathbb{Z} \to \mathbb{R}$, denote the z-transform of x by

$$X(z) = \mathcal{Z}x := \sum_{n=-\infty}^{\infty} x[n]z^{-n},$$

and the inverse z-transform denoted by $\mathcal{Z}^{-1}$ satisfies $x = \mathcal{Z}^{-1}X$. The set of values of z for which the z-transform converges absolutely is called the *region of convergence* (ROC) of the z-transform.

Given a linear *time-invariant* (LTI) $G : \ell_2 \to \ell_2$ that is possibly non-causal, denote by $g : \mathbb{Z} \to \mathbb{R}$ the impulse response/convolution kernel of G.

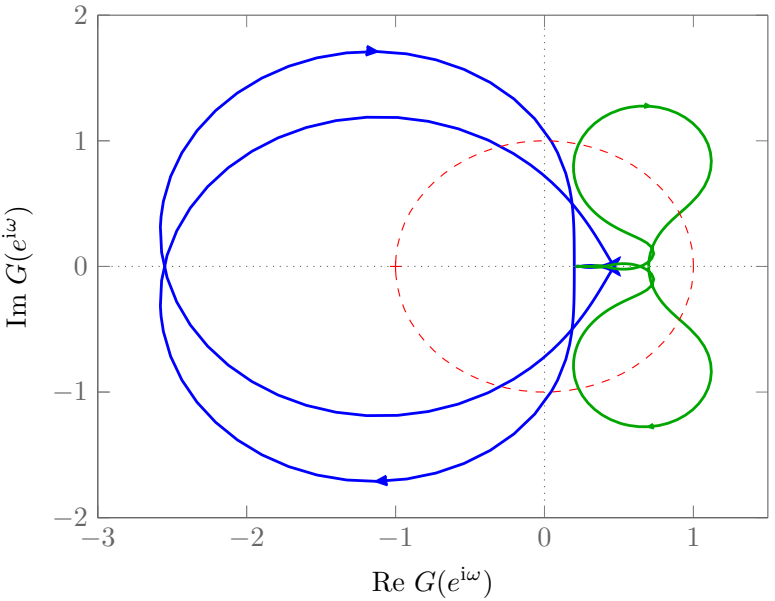


Figure 8.4 Comparison of Nyquist Diagrams for $G(z)$: The blue and green curves correspond to System I and System II, respectively. The unit circle is shown in dashed red. The blue curve corresponding to the convergent system makes a longer journey outside of the unit circle compared to the green one.

In particular, note that $Gx = g * x$ for any $x \in \ell_2$, where $*$ denotes the convolution operator defined as

$$(x * y)[n] = \sum_{m=-\infty}^{\infty} x[m]y[n-m].$$

We define $G(z) := \mathcal{Z}g$. If G is a bounded LTI operator, then

$$\{|z| = 1\} \subset \text{ROC of } G(z).$$

Define

$$L_\infty := \left\{ G(z) \;\middle|\; \|G(z)\|_\infty := \sup_{\omega \in [0,2\pi)} |G(e^{i\omega})| < \infty \right\}.$$

Hereafter, we denote by $G(z)$ the transfer function corresponding to a bounded LTI operator, i.e., $G(z) \in L_\infty$.[1]

[1] In this thesis, we do not explicitly specify the ROC for stable systems since it can unambiguously be determined.

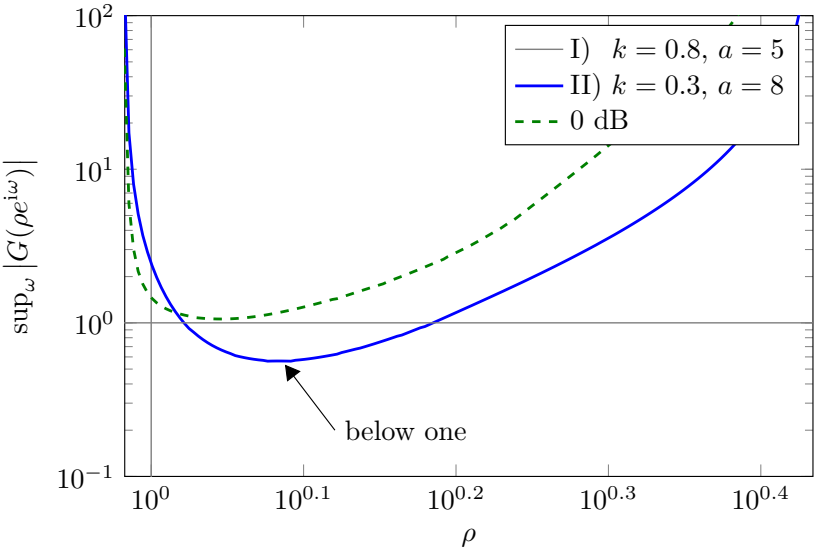


Figure 8.5 Comparison of $\|G(\rho z)\|_\infty$ for $0.96 < \rho < 2.72$: The blue and green curves correspond to System I and System II, respectively. We prove in Theorem 14 that if the plot of $\|G(\rho z)\|_\infty$ vs. $\rho > 0$ goes below one, ILC converges on finite time intervals.

Define G^j as

$$\begin{aligned} G^0 &= I, \\ G^{j+1} &= G \circ G^j, \end{aligned} \tag{8.8}$$

where I denotes the identity operator and $\circ$ the composition of two operators.

Lemma 3— [Kreyszig, 1989]
Given a bounded linear time-invariant operator $G : \ell_2 \to \ell_2$, it holds that

$$\|G\|_{\ell_2 \to \ell_2} = \|G(z)\|_\infty . \qquad \square$$

Define the truncation operator as

$$(\Pi_k x)[n] = \begin{cases} x[n], & n < k \\ 0, & \text{otherwise} \end{cases} \tag{8.9}$$

Given the definition of the truncation operator (8.9), an operator is causal if for all $k \in \mathbb{Z}$,

$$\Pi_k G(I - \Pi_k) = 0. \tag{8.10}$$

For an operator $G : \ell_2 \to \ell_2$, we define the truncated operator

$$\bar{G} := \Pi_N(I - \Pi_0)G\Pi_N(I - \Pi_0), \tag{8.11}$$

where $\Pi_N(I - \Pi_0)$ sets the values of a signal outside of $\{0, \ldots, N-1\}$ to zero and the finite Toeplitz matrix

$$T_{N\{i,j\}}(g) = g[i-j],\ i, j = 1, \ldots, N, \tag{8.12}$$

where g is the impulse response of G.

Lemma 4
Given $x \in \ell_2$ and $\bar{G}$ as defined in (8.11), if $y = \bar{G}x$ then

$$\mathbf{y} = T_N(g)\mathbf{x}, \tag{8.13}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ are the input and output signals converted into vectors, e.g., $\mathbf{x} = \{x[0], \cdots, x[N-1]\}$. □

Proof Define $\bar{x} := \Pi_N(I - \Pi_0)x$. For $0 \le i < N$, we have

$$y[i] = \sum_{j=-\infty}^{\infty} g[i-j]\bar{x}[j] = \sum_{j=0}^{N-1} g[i-j]x[j] = \sum_{j=0}^{N-1} T_{N\{i,j\}}(g)\mathbf{x}_{\{j\}}, \tag{8.14}$$

which proves the result. □

Note that the operator $\bar{G}$ can be represented by a matrix by converting the input and output signals to vectors. In this case, $\bar{G} : \ell_2 \to \ell_2$ is a linear *time-varying* system with a Toeplitz matrix representation $T_N(g) \in \mathbb{R}^{N\times N}$. The infinite-dimensional Toeplitz matrix corresponding to the operator G is denoted by $T_\infty(g)$.

Lemma 5
If G is causal, then $\bar{G}^j = \Pi_N G^j(I - \Pi_0)$. □

Proof By setting $k = 0$ and $k = N$ in (8.10), it follows that

$$(I - \Pi_0)G(I - \Pi_0) = G(I - \Pi_0),$$
$$\Pi_N G\Pi_N = \Pi_N G,$$

respectively. By multiple applications of these two rules to the expression of $\bar{G}^j$ obtained from (8.11) and noting that the order of truncation can be interchanged, the result follows. □

The spectral radius and the largest singular value of $T \in \mathbb{R}^{N\times N}$ are defined, respectively, as

$$\text{rad}(T) := \max_i |\lambda_i(T)|, \tag{8.15}$$

$$\bar{\sigma}(T) := \sqrt{\text{rad}(TT^*)}. \tag{8.16}$$

where $\lambda_i(T)$ denotes an eigenvalue and T^* the transpose of T.

Lemma 6—[Horn and Johnson, 2012, Sec. 5.6]
If $T \in \mathbb{R}^{N\times N}$, then

$$\|T\|_2 := \sup_{\mathbf{u}\neq 0} \frac{\|T\mathbf{u}\|}{\|\mathbf{u}\|} = \bar{\sigma}(T) \tag{8.17}$$

□

The set of strictly positive and non-negative integers are denoted by $\mathbb{Z}_{>0}$ and $\mathbb{Z}_{\geq 0}$, respectively.

Lemma 7—[Kreyszig, 1989, Theorem 7.5-2]
Given a matrix $T \in \mathbb{R}^{N\times N}$,

$$\text{rad}\, T \leq \|T^n\|^{\frac{1}{n}}, \quad \forall n \in \mathbb{Z}_{>0}$$ □

Lemma 8—[Horn and Johnson, 2012, Sec. 5.6]
Given a matrix $T \in \mathbb{R}^{N\times N}$, if $\|T\|_2 < 1$, then

$$\lim_{j\to\infty} \sum_{n=0}^{j} T^n = (I - T)^{-1}.$$ □

Lemma 9—[Böttcher and Grudsky, 2005, p. 177]
For a transfer function $G(z)$ and its corresponding Toeplitz matrix $T_N(g)$ and for all $n \in \mathbb{Z}_{\geq 0}$, it holds

$$\|T_N^n(g)\|_2 \leq \|T_N(g)\|_2^n \leq \|T_\infty(g)\|_2^n = \|G(z)\|_\infty^n. \tag{8.18}$$

□

Lemma 10—[Böttcher and Grudsky, 2005, Theorem 8.1]
If $g \in \ell_1$, then

$$\lim_{N\to\infty} \|T_N^n(g)\|_2 = \|T_\infty(g)\|_2^n = \|G(z)\|_\infty^n. \tag{8.19}$$

□

For an arbitrary function f, by $f[\cdot]$ we denote a mapping $\mathbb{Z} \to \mathbb{R}$ which when evaluated at a point $n \in \mathbb{Z}$ results in $f(n)$. For instance, $\rho^{-[\cdot]} : \mathbb{Z} \to \mathbb{R}$ evaluated at n is equal to ρ^{-n}.

To summarize the notation, we use the symbols G, $\bar{G}$, $G(z)$, g, and $T_N(g)$, which read as a bounded linear time-invariant operator $\ell_2 \to \ell_2$, a bounded linear time-varying operator $\ell_2 \to \ell_2$ operating on signals with a finite support $n \in \{0, \ldots, N-1\}$, the transfer function of a stable LTI system, which gives us the frequency response when evaluated on the unit circle $z = e^{i\omega}$, the impulse response of G obtained by the inverse z-transform of $G(z)$, and the finite Toeplitz matrix corresponding to $\bar{G}$, respectively. For more detailed descriptions and some fundamental results, the readers are advised to look at the background and preliminaries in Sec. 8.3.

8.4 Iteration-Domain Dynamics

In order to analyze the convergence of the ILC system (8.1)–(8.3), we derive the dynamics of the system in the iteration domain. Furthermore, to take into account the assumption of the finite-time intervals, we use the truncated counterparts of the original operators as defined in (8.11). Using the truncated operators, system equations (8.1)–(8.3) can be rewritten as

$$u_{j+1} = \bar{G}u_j + \bar{H}\tilde{r} \tag{8.20}$$

$$e_j = -\bar{T}_u u_j + \tilde{r}, \tag{8.21}$$

where $\bar{G} := \bar{Q}(I - \bar{L}\bar{T}_u)$, $\bar{H} := \bar{Q}\bar{L}$ and $\tilde{r} := (I - \bar{T}_r)r$, hence

$$u_j = \sum_{i=0}^{j-1} \bar{G}^i \bar{H}\tilde{r} + \bar{G}^j u_0 \tag{8.22}$$

$$e_j = (I - \bar{T}_u \sum_{i=0}^{j-1} \bar{G}^i \bar{H})\tilde{r} - \bar{T}_u \bar{G}^j u_0, \tag{8.23}$$

where $\bar{G}^j$ denotes j times composition of $\bar{G}$ by itself defined in (8.8).

It is desired that the outcome of the algorithm be independent of the choice of initial input u_0. Therefore, as j tends to infinity, the response to the initial conditions $\bar{T}_u \bar{G}^j u_0$ must vanish. Additionally, the forced response due to the reference signal $(I - \bar{T}_u \sum_{n=0}^{j-1} \bar{G}^n \bar{H})\tilde{r}$ should be bounded on the desired interval.

Assuming $\|T_N(g)\|_2 < 1$, (8.22) and (8.23) converge respectively to

$$u_\infty = (I - \bar{G})^{-1}\bar{H}\tilde{r} \tag{8.24}$$

$$e_\infty = (I - \bar{T}_u(I - \bar{G})^{-1}\bar{H})\tilde{r}, \tag{8.25}$$

where $(I - \bar{G})^{-1}$ is the operator $\ell_2 \to \ell_2$ corresponding to $(I - T_N(g))^{-1}$. The result can be derived by using Lemmas 4 and 8. Generally, whether the residual e_∞ is acceptable or not depends on the length of the experiment, the signal $\tilde{r}$, and the norm of the operator in (8.25).

Note that if $\bar{T}_u$ is right invertible, we can express the error as

$$e_\infty = \bar{T}_u(I - \bar{G})^{-1}(I - \bar{Q})\bar{T}_u^{-1}\tilde{r}$$

Similarly, if $\bar{L}$ is left invertible and $\bar{Q} = \bar{q}I$, we have

$$e_\infty = \bar{L}^{-1}(I - \bar{G})^{-1}(I - \bar{Q})\bar{L}\tilde{r}$$

Therefore, $\bar{Q} = I$ results in $e_\infty \equiv 0$. However, this choice may not fulfill the convergence condition and degrades the robustness of ILC [Roover, 1996]. Hence, there is a trade-off between the convergence property and the residual error.

8.5 Convergence of Iterative Procedures

Analysis of linear iterative procedures such as (8.20) can be well understood using linear system theory [Norrlöf and Gunnarsson, 2002]. First, we formalize the notion of convergence analogously to linear systems. Thereafter, we state time and frequency domain criteria for convergence.

Definition 1
We say an iterative procedure $\bar{G}$, $\bar{H} \neq 0$

$$u_{j+1} = \bar{G}u_j + \bar{H}r_j \tag{8.26}$$

is convergent if and only if for all $u_0 \in \ell_2$ and iteration-independent inputs $r_j \equiv r \in \ell_2$, there exists an equilibrium signal u_e such that for any given $\epsilon > 0$, there exists $J = J(\epsilon, u_0, r) \in \mathbb{Z}_{\geq 0}$ such that

$$\forall j > J \Rightarrow \|u_j - u_e\|_2 < \epsilon. \qquad \square$$

Definition 2
We say the convergence is *strictly monotone* if in addition to Definition 1,

$$\|u_{j+1} - u_e\|_2 < \|u_j - u_e\|_2 . \tag{8.27}$$
$\square$

Remark 2
Definition 1 translates to uniform asymptotic stability of the lifted system described by matrices $\mathbf{G} := T_N(g)$ and $\mathbf{H} := T_N(h)$,

$$\mathbf{u}_{j+1} = \mathbf{G}\mathbf{u}_j + \mathbf{H}\mathbf{r}, \tag{8.28}$$

where $T_N(g)$ is defined in (8.12) and $\mathbf{u}, \mathbf{r} \in \mathbb{R}^N$ are defined as

$$\mathbf{u} := (u[0], \cdots, u[N-1])^T,$$
$$\mathbf{r} := (r[0], \cdots, r[N-1])^T.$$

Note that the time references of inputs and outputs are assumed to be the same and do not change from one iteration to the next. Therefore, the representation of the lifted system is independent of the relative degree of the system. □

Proposition 11
The iterative procedure (8.26) converges according to

$$\text{Definition 1} \iff \operatorname{rad}\mathbf{G} < 1 \tag{8.29}$$
$$\text{Definition 2} \iff \bar{\sigma}(\mathbf{G}) < 1 \tag{8.30}$$

□

Proof The first statement is a well-known result for asymptotic stability of linear systems (8.28); see [Rugh, 1996]. For the proof of sufficiency of the second statements see [Norrlöf and Gunnarsson, 2002]. To show necessity, suppose that (8.29) holds, i.e., $\mathbf{u}_j$ in (8.28) converges to the limit $\mathbf{u}_e$ satisfying $\mathbf{u}_e = \mathbf{G}\mathbf{u}_e + \mathbf{H}\mathbf{r}$. Suppose to the contraposition that $\bar{\sigma}(\mathbf{G}) \geq 1$ and let $\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be a singular decomposition, where $\mathbf{V} = [\mathbf{V}_1, \ldots, \mathbf{V}_n]$, $\mathbf{U} = [\mathbf{U}_1, \ldots, \mathbf{U}_n]$ and $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ with $\sigma_1 \geq \ldots \geq \sigma_n$. By choosing $\mathbf{u}_0 = \mathbf{u}_e + \mathbf{V}_1$, it follows that $\mathbf{u}_1 = \mathbf{u}_e + \sigma_1\mathbf{U}_1$, whereby (8.27) is violated. □

Lemma 12
Given $x \in \ell_2$ and $\rho \in \mathbb{R}$, if $y = \bar{G}x$, then

$$\rho^{-[\cdot]}y = \bar{G}_\rho(\rho^{-[\cdot]}x), \tag{8.31}$$

where $\bar{G}_\rho$ is the truncated operator corresponding to the system $G_\rho(z) := G(\rho z)$ using the def. (8.11) and $\rho^{-[\cdot]} : \mathbb{Z} \to \mathbb{R}$ evaluated at n is equal to ρ^{-n}. □

Proof Let us denote the impulse response of G_ρ by g_ρ. From definition (8.12), we conclude $\forall i, j \in [1, N]$

$$T_{N\{i,j\}}(g_\rho) = g_\rho[i-j] = \rho^{-i+1}g[i-j]\rho^{j-1}.$$

With $P := \mathrm{diag}(1, \rho, \ldots, \rho^{N-1})$, we get

$$T_N(g_\rho) = P^{-1} T_N(g) P. \tag{8.32}$$

Using the matrix representation of $y = \tilde{G}x$ according to Lemma 4 and multiplying both sides from the left by P^{-1}, we obtain

$$P^{-1}\mathbf{y} = P^{-1}T_N(g)PP^{-1}\mathbf{x} = T_N(g_\rho)P^{-1}\mathbf{x},$$

which is (8.31) in its lifted form. □

Lemma 13
Assume $G(z)$ is a (not necessarily causal) LTI system and there exists a $\rho > 0$ such that $\{|z| = \rho\} \subset$ ROC of $G(z)$ and $\|G(\rho z)\|_\infty < 1$. Then

$$\lim_{j\to\infty} \tilde{G}^j u_0 = 0, \ \ \forall u_0 \in \ell_2. \tag{8.33}$$

□

Proof We prove the result for the equivalent Toeplitz matrix representation of the system $\tilde{G}$ according to Lemma 4. Note that because of the similarity transformation (8.32), $T_N(g)$ has the same spectrum as $T_N(g_\rho)$. From Lemmas 7 and 9, it follows

$$\text{rad } T_N(g) = \text{rad } T_N(g_\rho) \leq \left\|T_N(g_\rho)\right\|_2 \leq \|G(\rho z)\|_\infty < 1. \tag{8.34}$$

The proof is completed by applying (8.29). □

Remark 3
Considering Prop. 11 and Lemma 6, (8.34) guarantees strictly monotone convergence for the truncated operators if $\|G(\rho z)\|_\infty < 1$ for $\rho = 1$. Therefore, the standard result for the convergence of ILC [Norrlöf and Gunnarsson, 2002] for causal filters is also applicable to non-causal filters. □

The following theorem establishes a frequency-domain criterion for the convergence of the ILC scheme (8.1)–(8.3). It is more widely applicable than the standard criterion in the literature (see for example [Bristow et al., 2006; Norrlöf and Gunnarsson, 2002; Longman, 2000]) in the sense that it is not an asymptotic result for the case of $N \to \infty$, covers ILC with non-causal filters, and applies to the convergence according to Definition 1 as well as Definition 2 by setting $\rho = 1$.

THEOREM 14
Given an LTI system $G(z)$ with the impulse response g, assume that there exists a $\rho > 0$ such that $\{|z| = \rho\} \subset \text{ROC of } G(z)$ and

$$\|G(\rho z)\|_\infty < 1. \tag{8.35}$$

Then the following limits for ILC system (8.1)–(8.3) with interval length of N hold:

$$\lim_{j\to\infty} u_j = u_\infty = (I - \bar{G})^{-1}\bar{H}\tilde{r}$$
$$\lim_{j\to\infty} e_j = e_\infty = (I - \bar{T}_u(I - \bar{G})^{-1}\bar{H})\tilde{r}$$

where $\bar{G} := \bar{Q}(I - \bar{L}\bar{T}_u)$, $\bar{H} := \bar{Q}\bar{L}$ and $\tilde{r} =: (I - \bar{T}_r)r$. □

Proof The limits of (8.22) and (8.23) need to be calculated. For the proof, we use the corresponding matrix representation of the truncated systems, e.g., $T_N(g)$ instead of $\bar{G}$ according to Lemma 4. First note that

$$\begin{aligned}\lim_{j\to\infty}\sum_{i=0}^{j-1} T_N^i(g) &= P^{-1}\left(\lim_{j\to\infty}\sum_{i=0}^{j-1}\left(PT_N(g)P^{-1}\right)^i\right)P \\ &= P^{-1}\left(I - PT_N(g)P^{-1}\right)^{-1}P = \left(I - T_N(g)\right)^{-1}. \end{aligned}\tag{8.36}$$

To derive (8.36), we have used the fact that $\left\|PT_N(g)P^{-1}\right\| \le \|G(\rho z)\|_\infty < 1$, and Lemma 8. Moreover, according to Lemma 13,

$$\lim_{j\to\infty} T_N^j(g) = 0. \tag{8.37}$$

Substituting (8.36) and (8.37) into the limits of (8.22) and (8.23) as $j \to \infty$ completes the proof. □

A Special Case

As a special case, we consider causal and stable rational systems

$$G(z) = \frac{\sum_{n=0}^{\infty} b_n z^{-n}}{\sum_{n=0}^{\infty} a_n z^{-n}}.$$

In this case, a milder condition than that of Theorem 14 is both sufficient and necessary for convergence on finite intervals, only the direct term matters. Notice that the causality condition on the filters is, however, quite restrictive for the performance of the ILC algorithm.

THEOREM 15
Given a causal, stable, proper, and rational system $G(z)$, the iterative procedure (8.26) is convergent as per Definition 1 if and only if the direct term satisfies

$$g_0 := \lim_{|z|\to+\infty} |G(z)| = b_0/a_0 < 1. \tag{8.38}$$

□

Proof For sufficiency, due to the rationality of $G(z)$ and (8.38), there exists a sufficiently large ρ such that $\|G(\rho z)\|_\infty < 1$. Therefore, the iterative procedure (8.20) is convergent according to Theorem 14.

For necessity, note that if (8.38) is violated we can write $G(z)$ as

$$G(z) = g_0 + z^{-1}G_1(z), \tag{8.39}$$

where $G_1(z)$ is proper and $|g_0| \geq 1$. For any $j \geq 1$ we get

$$G^j(z) = g_0^j + z^{-1}P_j(z), \tag{8.40}$$

where $P_j(z)$ is proper. Since G is causal, by Lemma 5, $\bar{G}^j = \Pi_N G^j(I - \Pi_0)$. Having $|g_0| \geq 1$ would contradict $|g_0^j| \to 0$, which is needed for convergence with the initial signal $x_0 = \delta[\cdot]$. □

COROLLARY 16
An iterative procedure (8.20) corresponding to a causal and *strictly* proper transfer function is convergent according to Definition 1, since $g_0 = 0$ in this case. □

8.6 Practical Considerations

When $\|G(z)\|_\infty > 1$, the ILC lacks the strictly monotone convergence property. Hence, the norm of the signals may grow as iterations proceed. This section provides some insights into this transient behavior and suggests a solution to upper bound the growth of the error under a certain condition.

THEOREM 17
Given $G(z)$ with the impulse response g, let $u_{j+1} = \bar{G}u_j$ for some u_0 and $j \in \mathbb{Z}_{\geq 0}$. Then for every $\rho \geq 1$ for which $\{|z| = \rho\} \subset$ ROC of $G(z)$, there exists a $C > 0$ such that

$$\forall n \in \mathbb{Z}, \forall j \in \mathbb{Z}_{\geq 0}, \quad \|\Pi_n u_j\|_2 \leq C\rho^n \|G(\rho z)\|_\infty^j. \tag{8.41}$$

□

Proof We split $\rho^{-[\cdot]}u_j$ such that

$$\left\|\rho^{-[\cdot]}u_j\right\|_2 = \left\|\Pi_n\rho^{-[\cdot]}u_j\right\|_2 + \left\|(I-\Pi_n)\rho^{-[\cdot]}u_j\right\|_2. \tag{8.42}$$

Furthermore, note that when $\rho \geq 1$,

$$\rho^{-n}\left\|\Pi_n u_j\right\|_2 \leq \left\|\Pi_n\rho^{-[\cdot]}u_j\right\|_2. \tag{8.43}$$

Combining (8.42) and (8.43) results in

$$\rho^{-n}\left\|\Pi_n u_j\right\|_2 \leq \left\|\rho^{-[\cdot]}u_j\right\|_2 - \left\|(I-\Pi_n)\rho^{-[\cdot]}u_j\right\|_2. \tag{8.44}$$

Considering (8.44) and Lemma 12, we obtain

$$\left\|\Pi_n u_j\right\|_2 \leq \rho^n\left\|\rho^{-[\cdot]}u_j\right\|_2 \leq \rho^n\left\|T_N(g_\rho)\right\|_2^j\left\|\rho^{-[\cdot]}u_0\right\|_2. \tag{8.45}$$

Note that $\|\rho^{-[\cdot]}u_0\|_2$ is constant. By considering Lemma 9, the final result follows. □

Remark 4
If for some $\rho > 1$, $\|G(\rho z)\|_\infty < 1$, Theorem 17 intuitively means that the iteration operator $\bar{G}$ can shift the energy distribution of a signal only toward plus infinity. Define $\tilde{u}_j := u_j - u_e$, which results in $\tilde{u}_{j+1} = \bar{G}\tilde{u}_j$. By equating the right hand side of (8.41) to ϵ, for a given $n \in \mathbb{Z}_{\geq 0}$ and $\forall\epsilon > 0$, we can find J such that $\left\|\Pi_n\tilde{u}_j\right\|_2 \leq \epsilon$ for all $j > J$. For $j > J' > J$, (8.41) implies that n can be increased. Hence, the norm of a larger interval is guaranteed to be less than ϵ. This resembles a wave traveling to the right. Assuming $\|G(\rho z)\|_\infty < 1$ for $0 < \rho < 1$, a similar relation to (8.41) can be derived for $\|(I-\Pi_n)\tilde{u}_j\|_2$, so the wave moves toward minus infinity. □

With the result of Theorem 17 in mind, let us reexamine the example in Sec. 8.2. In Fig. 8.2 for the convergent case, the growing wave is pushed toward the right as the number of iterations increases while for the non-convergent case the signals grow unbounded in the time region of the trial.

Assuming the conditions of Theorem 17 are fulfilled for $\rho > 1$, we can employ a strategy to only feed "safe inputs" and set the remaining inputs to a bounded signal. This way, a possibly growing tail of the control signal can be truncated, since we know that its energy distribution can only be shifted to plus infinity. Additionally, since we have assumed that (8.1) is stable, setting the removed part of the signal to a bounded signal is harmless. More specifically, a good choice is

$$u_j[n] = \begin{cases} Q\left(u_{j-1} + Le_{j-1}\right)[n], & 0 \leq n < j\hat{d} \\ T_u(1)^{-1}(1-T_r(1))r[n], & j\hat{d} \leq n < N, \end{cases} \tag{8.46}$$

where the steady-state solution to (8.1) is used for the unsafe region.

To determine the range of safe inputs, consider the model $G(z) = cz^{-d}$, i.e., $g[n] = c\delta[n-d]$ where $\delta[\cdot]$ is the Dirac delta function. According to this model,

$$g^j[n] = c^j \delta[n - jd]. \tag{8.47}$$

Thus, the norm of the signal is multiplied by c and shifted d steps along the time axis in each iteration. Obviously, after sufficiently many iterations, the energy of the signal lies outside of the interval of $n \in \{0, \ldots, N-1\}$. By setting the right hand side of (8.41) in Theorem 17 to less than or equal to ϵ, we derive

$$n \leq \frac{\ln \epsilon - \ln C}{\ln \rho} - j\frac{\ln \|G(\rho z)\|_\infty}{\ln \rho}.$$

By comparison with model (8.47), we conclude that a good choice for $\hat{d}$ in (8.46) is

$$\hat{d} = -\frac{\ln \|G(\rho z)\|_\infty}{\ln \rho}. \tag{8.48}$$

Therefore, a rule of thumb for calculating $\hat{d}$ is obtained by substituting ρ with $\rho^* := \arg\min_\rho \|G(\rho z)\|_\infty$ in (8.48).

Figure 8.6 shows an example of the application of the safe-feed strategy (8.46). The filter $Q(s)$ is the same as in (8.6), and the rest of the transfer functions are

$$\begin{aligned} T_u(s) &= \frac{s-5}{(s+1)(s^2+0.8s+16)},\ T_r(s) = 0, \\ L_d(z) &= -6\frac{1 - 2.7z^{-1} + 2.5z^{-2} - 0.8z^{-3}}{1 - 0.5z^{-1} - 0.3z^{-2}} z^5. \end{aligned} \tag{8.49}$$

The Bode plots of $T_u(s)$ after discretization is shown in Fig. 8.7. From the phase plot, it is evident that the system is non-minimum phase. Figure 8.8 confirms that the system converges according to Theorem 14. Without the safe-feed strategy, the amplitude of signals grows rapidly as shown in the left pane of Fig. 8.6. However, by employing the safe-feed strategy the transient behavior is kept under control.

Early termination schemes are also advisable since, as it has been reported by other authors, divergence might appear after many iterations [Longman and Huang, 2002]. In the case of $\|G(z)\|_\infty \geq 1$, more precautions must be taken considering the inherent lack of robustness.

Robustness

Here we present an example that shows the lack of robustness of ILC when $\|G(z)\|_\infty > 1$ and N is large. According to (8.22), the control signal is

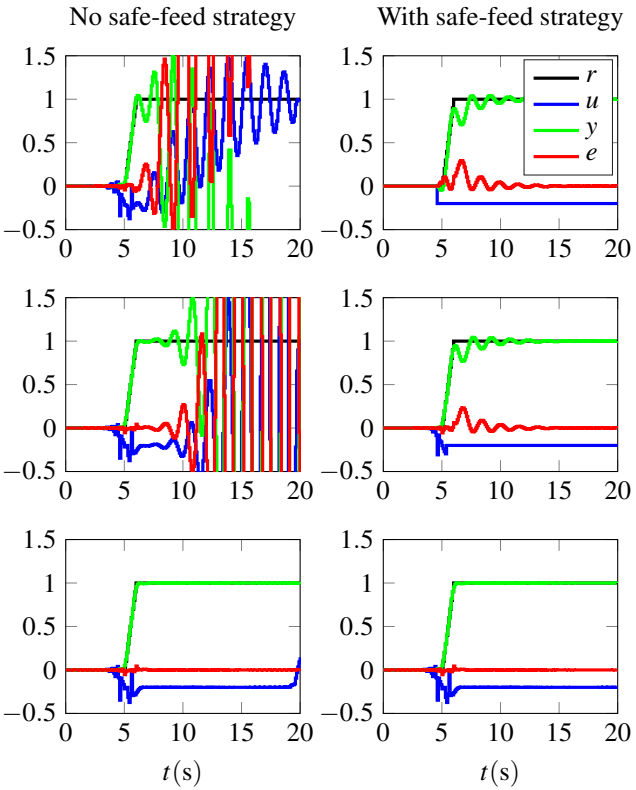


Figure 8.6 Safe-feed strategy for non-minimum phase system (8.49): From top to bottom, iterations 5, 10, and 100 are illustrated. The signals shown are the reference (black), input (blue), output (green), and error (red). For the right pane, the safe-feed strategy (8.46) is employed with $\hat{d} = 5$, while for the left pane it is not.

comprised of two terms; one is the response to the initial control signal $\tilde{G}^j u_0$ and the other is entirely due to the reference signal $\sum_{i=0}^{j-1} \tilde{G}^i \tilde{H} \tilde{r}$. In this example, we see while the term related to the initial control signal vanishes, the geometric series resulting from the response to the reference signal may not converge.

Consider

$$G(z) = 0.2z^{-3} - 0.6z^{-2} + 0.4z^{-1} - 0.1 + 0.1z + 0.12z^2. \tag{8.50}$$

Figure 8.9 confirms that $\operatorname{rad} T_N(g) < 1$ while $\|G(z)\|_\infty > 1$. Figure 8.10 compares the impulse response of the infinite series $\sum_{i=0}^{\infty} \tilde{G}^i \tilde{H} \tilde{r}$ calculated iteratively and the closed-form solution (8.36). As seen, the responses match

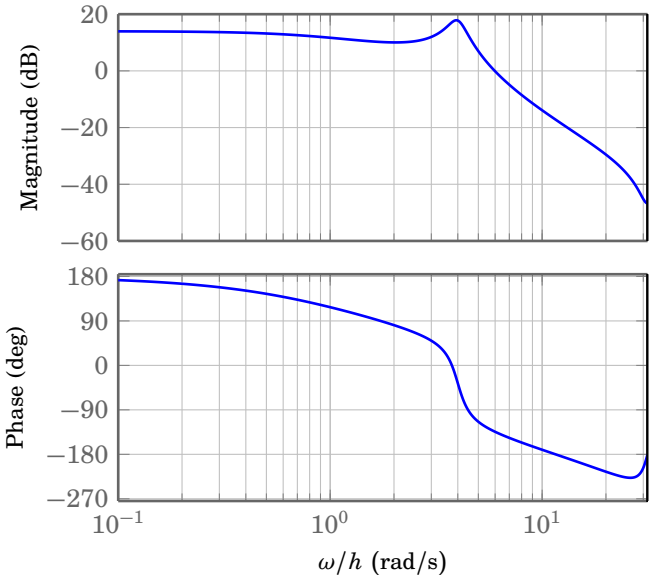


Figure 8.7 Example (8.49): Bode diagrams of $T_{ud}(e^{i\omega})$ shows that the system is non-minimum phase.

in the beginning but the blue curve corresponding to the infinite series diverges after a while.

To explain the issue, let us have a look at the numerical values of the spectral radius and the norm of the infinite series for various N. The infinite series is approximated by a finite series, but with a large number of terms. In Fig. 8.11, we see that at approximately $N = 246$ the norm of the approximation of the infinite series of the system starts to deviate quickly from the closed-form solution (8.36). At the same N, we see that the numerically computed spectral radius becomes larger than one. Nevertheless irrespective of N, $\|T_N(g)\| > 1$ and $\left\|T_N^{5000}(g)\right\| \approxeq 0$ within the machine precision.

First of all, we observe that the closed-form solution for the calculation of the infinite series does not hold for the numerical results when $\|G(z)\|_\infty > 1$ and N is large. Additionally, from the comparison of the spectral radius of $T_N(g)$ and $PT_N(g)P^{-1}$ we see that they diverge even earlier. In theory, we do not expect that the eigenvalues of the similarity-transformed matrices of finite size differ. However, in practice when N becomes large, they are not the same because of numerical errors.

In conclusion, as the trial length N becomes large, the result of Theorem 14 does not hold in practice. The issue must indeed be traced back

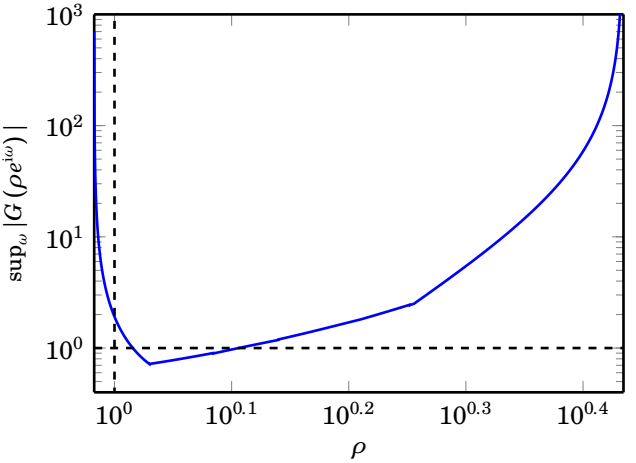


Figure 8.8 Plot of $\|G(\rho z)\|_\infty$ vs. ρ for the non-minimum phase system (8.49).

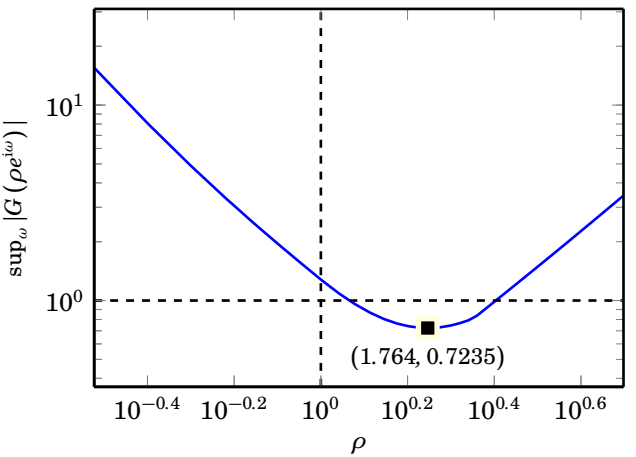


Figure 8.9 Plot of $\|G(\rho z)\|_\infty$ for the example given in (8.50).

to (8.29), i.e., $\operatorname{rad}\mathbf{G} < 1$ is practically not sufficient for the convergence according to Definition 1. This suggests using a more robust criterion such as the pseudo-spectrum [Reichel and Trefethen, 1992] instead of (8.29). Nevertheless, depending on the disturbance in the system we can observe convergence for a short period even if each trial length is quite long. As the vertical line in Fig. 8.10 indicates, we can expect a match between the

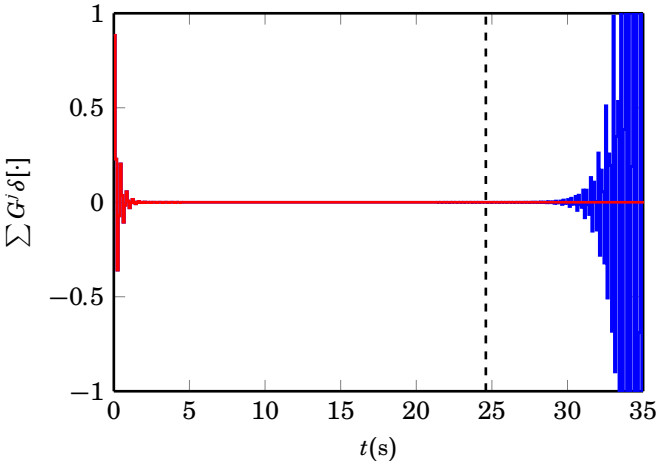


Figure 8.10 Comparison of the impulse response of $\sum G^j$ (blue line) and the closed-form solution (8.36) (red line).

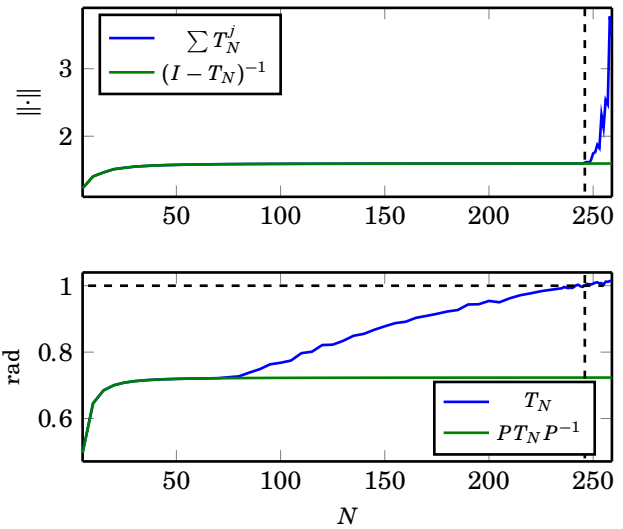
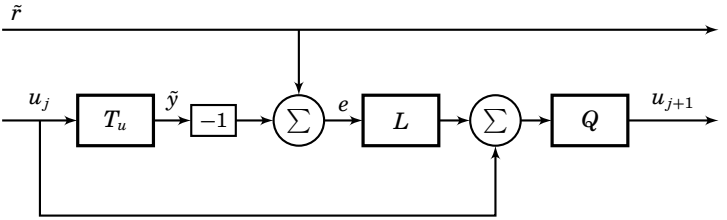
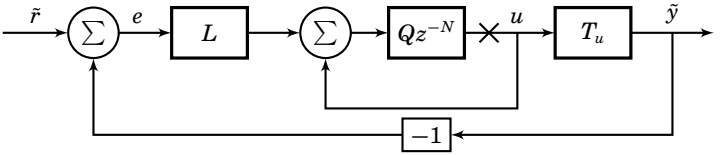


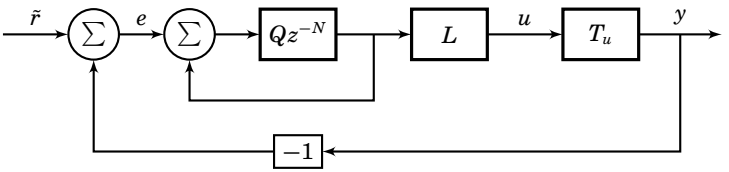
Figure 8.11 The 2-norm of the series $T_N^0(g) + \cdots + T_N^{5000}(g)$ compared with $(I - T_N(g))^{-1}$ and the numerically calculated spectral radius of $T_N(g)$ compared with $PT_N(g)P^{-1}$. Matrix P corresponds to $\rho^* \approxeq 1.764$. For large time interval N, the approximation of the infinite series diverges from its closed-form solution. At the same interval length, we observe that the numerically calculated rad T_N becomes larger than 1, despite the fact the spectral radius of the its similarity-transformed matrix remains less than 1.



(a) Iterative Learning Controller



(b) Connecting the delayed u_{j+1} to u_j. The cross shows the point of interconnection.



(c) Repetitive Controller

Figure 8.12 Relation between ILC and repetitive control.

ideal case and the numerical result even for the non-truncated operator G up to Kh, where K corresponds to the size of largest matrix such that the numerical rad $T_K(g) < 1$.

8.7 Relation to the Repetitive Control

Consider the block diagram of ILC in Fig. 8.12a. Noting that the next iteration can be started at an arbitrary reference time, it is possible to add any amount of delay to the output of each stage. Specifically, if we consider cyclic signals, adding any multiple of the period of the reference signal will not affect r. Now, if we reuse the same structure and connect delayed u_{j+1} to u_j, we arrive at Fig. 8.12a. The familiar block diagram of the repetitive control in Fig. 8.12c is obtained by changing the position of filter L in the loop. This reveals the connection between repetitive control and ILC algorithm. Namely, ILC is an unfolded version of the repetitive control where each stage of the cascaded system has a similar initial condition.

Note that for repetitive control, we should examine the closed-loop prop-

erty of $G(z)z^{-N}$, where N is the period of the reference signal. Therefore, the criterion for the stability of the repetitive control is more restrictive than for the ILC.

8.8 Point-to-Point Trajectory Generation

For point-to-point trajectory generation, no explicit reference signal is defined. Therefore, we have a valid solution as long as the start and the end-point constraints are satisfied. We handle this case by setting the error in the interval between these points to zero. Note that in the lifted form, i.e., when considering $T_N(\cdot)$, setting the error to zero at certain points is equivalent of setting certain columns of $T_N(L)$ matrix to zero. Therefore, the analysis presented in this chapter with the modified L is applicable.

Figure 8.13 illustrates an example for planning a one-dimensional point-to-point movement. The reference signal (black line) is only valid before 7 s and after 10 s. Hence, it is required to plan a movement between these two points. The control signal (blue line) was initialized with the reference signal. However, after a number of iterations it was updated to reduce the error signal as it is seen in the lowest pane of the figure. As a preliminary observation, the convergence of the algorithm is more difficult to be ensured, but the residual error might be less than the case where the reference is fully specified. Further study of the nature of the converged solution is required.

8.9 Discussions

A number of qualitative measures for the convergence of (8.26) can be deduced from the plot of $\|G(\rho z)\|_\infty$ versus ρ. First of all, according to (8.34), the value $\inf_{\rho>0} \|G(\rho z)\|_\infty$ is an upper bound for the spectral radius of the finite Toeplitz matrix, indicating the exponential decay rate of the slowest transient mode of ILC applied on a finite-time interval. Based on (8.48), the larger ρ^* or $\|G(\rho^* z)\|$ the longer it would take for the growing wave to disappear. We can also define

$$\Omega := \{\rho \,|\, \|G(\rho z)\|_\infty < 1 \wedge |z| = \rho \subset \text{ROC of } G(z)\},$$
$$\rho_\ell := \inf \Omega, \quad \rho_h := \sup \Omega,$$

i.e., ρ_ℓ is the first crossing of 0 dB and ρ_h is the last one. If the control signal is weighted by an exponential function with rate $\rho < \rho_\ell$, the weighted signal appears to grow and shift toward plus infinity. On the other hand, weighting by $\rho > \rho_h$ results in a growing signal that moves toward minus infinity.

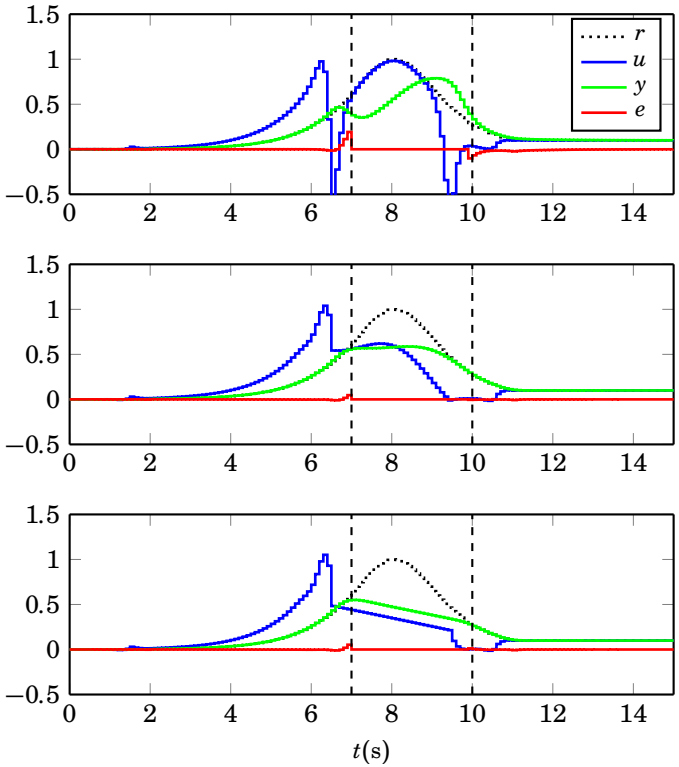


Figure 8.13 Point-to-point planning using ILC: From top to bottom, iterations 3, 50, and 300 are illustrated. The black dotted curve is the reference signal. The input is in blue, the output is in green, and the error signal in red. The reference signal in the region between the vertical dashed black lines is not relevant and it is only used as the initial guess for the control signal. ILC improves the control signal such that the output and the reference signal match closely at the dashed lines. I.e., a control signal has been found to transfer the state of the system from an initial point to a final point.

As an example we consider a binomial case where $G(z) = a_{-1}z^{-1} + a_1 z^1$ and $a_{-1} \geq 0$, $a_1 > 0$. The minimum of $\|G(\rho z)\|_\infty$ is achieved for $\rho^* = \sqrt{a_{-1}/a_1}$. Substituting ρ^* into $G(\rho z)$, results in

$$G(\rho^* z) = \sqrt{a_{-1}a_1}z^{-1} + \sqrt{a_{-1}a_1}z^1 = a(z^{-1} + z^1), \tag{8.51}$$

and $\|G(\rho^* z)\|_\infty = 2a$, where $a := \sqrt{a_{-1}a_1}$. Therefore, the coefficients are

retrieved by considering

$$a_1 = \frac{1}{2} r^* \left\| G(\rho^* z) \right\|_\infty, \tag{8.52}$$

$$a_{-1} = \frac{1}{2} \frac{1}{\rho^*} \left\| G(\rho^* z) \right\|_\infty. \tag{8.53}$$

Observe that

$$\begin{aligned} G(\rho^* z)^n &= a^n (z^{-1} + z^1)^n \\ &= a^n z^n (1 + z^{-2})^n \\ &= a^n (z^{-n} + n z^{-n+2} + \cdots + n z^{n-2} + z^n). \end{aligned} \tag{8.54}$$

Using Stirling's approximation when n is large, we have

$$\begin{aligned} \ln \binom{n}{m} \approx \left(n + \frac{1}{2} \right) \ln(n) - \left(m + \frac{1}{2} \right) \ln(m) \\ - \left(n - m + \frac{1}{2} \right) \ln(n - m) - \frac{1}{2} \ln(2\pi). \end{aligned} \tag{8.55}$$

If we further define $k \in \mathbb{Z}$ such that $m = \lfloor n/2 \rfloor + k$, and approximate $\ln(\cdot)$ by its fourth-order Taylor series expansion around $1/2$, we find

$$\ln \binom{n}{n/2 + k} \approx -\frac{16}{13} \frac{k^4}{n^3} - 2 \left(\frac{1}{n} - \frac{1}{n^2} \right) k^2 + (n+1) \ln(2) - \frac{1}{2} \ln(2\pi). \tag{8.56}$$

From (8.56), it becomes clear that the coefficients grow with the order of $O(2^n)$. Thus, $\left\| G(\rho^* z) \right\|_\infty = 2a < 1$ guarantees the convergence to zero. Additionally, we see that the effect of ρ is that it balances the tails of the polynomials in this case.

As a numerical example consider $G(z) = 0.225 z^1 + 0.9 z^{-1}$. The plot of $\left\| G(\rho z) \right\|_\infty$ against ρ is shown in Fig. 8.14. Figure 8.15 illustrates how the wave of divergence is affected by different values of ρ.

The frequency domain criterion provides an intuitive way to judge the convergence of ILC and facilitates the design process. Moreover, its evaluation can be computationally advantageous compared to the time-domain criteria. Iterative algorithms are often employed to solve for the largest eigenvalue or to calculate the H_∞ norm. Given a system with state dimension n, the cost of computing the H_∞ norm scales with $O(n^3)$ per iteration [Vandenberghe et al., 2005]. On the other hand, to compute the largest eigenvalue of a lifted system for the interval length of N, the cost per iteration for general algorithms is roughly $O(N^2)$. Thus, the time-domain

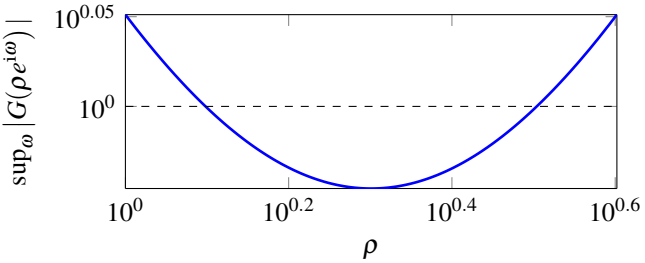


Figure 8.14 Plot of $\|G(\rho z)\|_\infty$ vs. ρ for a binomial case $G(z) = 0.225z^1 + 0.9z^{-1}$.

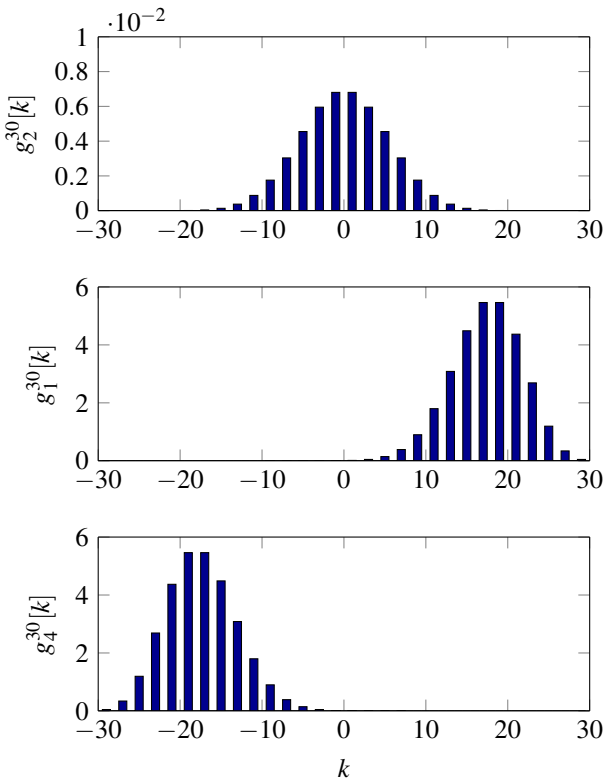


Figure 8.15 The effect of a weighting function on $g_\rho^n[k]$. For the top plot $\rho = \rho^* = 2$, for the middle plot, $\rho = 1 < \rho_\ell$ and for the bottom plot $\rho = 4 < \rho_h$. When $\rho < \rho_\ell$, the wave of divergence moves toward minus infinity and when $\rho > \rho_h$, it moves toward plus infinity.

criteria scale poorly with the interval length N, while the the frequency domain computation is independent of N.

Even when ρ is not set to one, the search over a range of ρ can be done efficiently. Firstly, note that very large values of ρ are not interesting since they imply poor transients. Secondly, the valid range of ρ is constrained by $\{|z| = \rho\} \subset$ ROC of $G(z)$. Let $|p_1| \leq |p_2| \leq \ldots \leq |p_n|$ denote the poles of $G(z)$. Since the ROC must be a connected region, if G is causal $|p_n| < \rho$ and if G is non-causal, $|p_i| < \rho < |p_{i+1}|$ for some p_i. Thirdly, note that as a consequence of the Hadamard three-circle theorem [Lang, 2013], $\log ||G(\rho z)||_\infty$ is strictly convex in $\log(\rho)$ if $G(z)$ is not a pure delay or forward shift. These facts allow for a fast evaluation of (8.35) using, for example, a bisection method.

While the frequency domain representation is widely accepted as an approximation for a practical ILC system where each iteration has a finite duration [Norrlöf and Gunnarsson, 2002; Longman, 2000], its implication for the frequency domain criterion (8.4) has not been clarified. From Lemma 9, it becomes evident how $\|G(z)\|_\infty$ approximates $\bar{\sigma}(\mathbf{G})$. Namely, the maximum singular value of a finite Toeplitz matrix is upper bounded by $\bar{\sigma}(T_\infty(g))$, which is equal to the infinity norm of the system. Thus, the criterion (8.4) is sufficient even for the convergence of a practical ILC algorithm. Moreover, when $g \in \ell_1$ according to Lemma 10, as $N \to \infty$ we have equality and hence the criterion (8.4) in this case is both sufficient and necessary for monotone convergence.

The criterion for non-monotone convergence is closely related to the results by [Schmidt and Spitzer, 1960], which states that if G has a finite impulse response (FIR), then

$$\liminf_{N\to\infty} \operatorname{rad} T_N(g) \leq \limsup_{N\to\infty} \operatorname{rad} T_N(g) \leq \inf_{\rho>0} \|G(\rho z)\|_\infty .$$

However, this criterion cannot be used for evaluating the convergence of long sequences, i.e., when N is large. The reason is that (8.29) does not guarantee robustness against perturbations. Specifically, the spectral radius cannot reliably describe the convergence behavior of large Toeplitz matrices and instead, pseudo-spectrum should be considered [Reichel and Trefethen, 1992]. In our opinion, this can clarify some of the gaps observed between the theoretical and practical convergence of ILC reported in the literature. The lack of robustness can manifest itself as the divergence of ILC that starts at a distant sample in time after a number of iterations. Hence, either a shorter time interval or an early termination strategy should be considered.

8.10 Conclusion

The time domain criterion states that ILC converges if and only if the spectral radius of $T_N(g)$ corresponding to the lifted system fulfills $\operatorname{rad} T_N(g) < 1$. For strictly monotone convergence, it is necessary and sufficient that the largest singular value is less than one, i.e., $\bar{\sigma}\left(T_N(g)\right) < 1$. These conditions have sufficient frequency domain counterparts applicable to ILC systems with causal as well as non-causal filters with no restriction to minimum-phase systems.

Our main result states that (non-monotone) convergence of ILC on finite-time intervals is implied by

$$\inf_{\rho>0} \sup_{\omega} \left|G(\rho e^{\mathrm{i}\omega})\right| < 1,$$

which is less conservative than $\sup_\omega |G(e^{\mathrm{i}\omega})| < 1$. In other words, for non-monotone convergence, the criterion $\|G(z)\|_\infty < 1$ can be evaluated on any circle in the region of convergence with radius larger than zero. The criterion implies that the spectral radius of the corresponding finite Toeplitz matrix $T_N(g)$ is less than one. In this case, repeated composition with the operator G shifts the energy distribution of the control signal toward plus or minus infinity if $\|G(z)\|_\infty \geq 1$ but $\|G(\rho z)\|_\infty < 1$ for some $\rho > 0$.

Our treatment clarified the source of approximation in using the frequency domain criterion. Furthermore, we elaborated on the connection between repetitive control and ILC. Specially, it was shown that the convergence criterion of repetitive control is more restrictive compared to ILC. We proposed an idea to use ILC for point-to-point trajectory planning problems.

Practical considerations were discussed, whereby it is advisable to employ safe-feed and early-termination strategies in an ILC system. Rigorous extensions of the main results to the multi-input-multi-output (MIMO) case as well as analyzing scenarios with non-repetitive disturbances [Mishra et al., 2007; Ruan et al., 2008] are parts of future research.

9

Conclusions and Future Research

A dual-arm haptic interface was proposed for collecting human-generated trajectories and interaction forces in a natural way. This model for Human Robot Interface (HRI) can solve the problem of mapping between a human demonstration and a robot behavior in addition to capturing interaction forces. While in the single-arm lead-through programming there is an unavoidable mechanical coupling between the operator, the manipulator, and the workpiece, using the teleoperation model 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 reduces unwanted demonstration side-effects compared to typical single-arm lead-through approaches.

The results of the rest of the thesis are applicable to automatic trajectory generation. There are many applications in which only initial and final states matter for defining a task. To motivate the use of simplistic models of robots for trajectory generation, we compared kinematic and dynamic models. Provided that kinematic constraints are conservative enough, our examples showed that kinematic models can be sufficient for trajectory planning. Having this in mind, we considered point-to-point trajectory planning and proposed a number of solutions. We derived an analytic solution to the problem of fixed-time optimal trajectory planning with maximum velocity and maximum acceleration constraints using the Pontryagin 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 fixed-time 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. We consider trajectory generation and path planning as interrelated problems, since they cannot always be treated separately without loss of performance. Hence, we incorporated geometric constraints in the MPC approach by approximating the workspace.

Manipulation tasks require interaction with a work-piece or other robots. For example, assembly relies on making contacts and controlling the robot in the contact situation. We developed a model for a ball-and-finger system. This model allowed us to study more challenging scenarios for trajectory generation, where interaction with another object plays an important role. Because of varying contacts and friction properties, the modeling resulted in a hybrid model with differential algebraic equation systems. We found optimal trajectories and paths for rotating the ball despite providing the algorithm with virtually no information about how to achieve the goal. The only information explicitly provided was the need for making and breaking contact. The optimization algorithm figured out where and when a contact has to be made and to be broken, in addition to finding the control and state trajectories. The proposed system opens up new opportunities for research, both in the modeling part and in the optimization. An extension of this work is to include extra fingers, where the problem of coordination and planning becomes of major importance.

Iterative Learning Control (ILC) as a means to obtain the control signal for realizing a trajectory was also studied in this thesis. We provided less restrictive criteria than the existing ones in the literature for evaluating the convergence of an ILC system, which are also applicable to non-monotone convergence of the ILC algorithm. Some suggestions for making non-monotone convergence practically useful were proposed. We also gave an example of how ILC can be modified for point-to-point trajectory generation. Finding directly control signals for point-to-point movements using an ILC algorithm, especially in multidimensional problems, is an interesting topic of research.

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. Thus, the border between motion planning and trajectory generation is blurry. Continuing on the analogy, it is also important to pay attention to the source of the changes when a trajectory controller is designed. Hence, a trajectory controller with two degrees of freedom is required to ensure different responses to the high-level goal and measurement disturbances.

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 [Ghazaei Ardakani, 2015], we can specify sensor-based piece-wise trajectories to accomplish a task. In essence, this is a form of the so-called procedural programming. On the other hand, we can specify 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 reality of robots, e.g., by introducing an 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 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.

Ahn, H.-S., Y.-Q. Chen, and K. Moore (2007). "Iterative learning control: brief survey and categorization". *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews* **37**:6, pp. 1099–1121. ISSN: 1094-6977.

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.

Anderson, R. and M. Spong (1989). "Bilateral control of teleoperators with time delay". *IEEE Trans. Automatic Control* **34**:5, pp. 494–501.

Arabyan, A. and F. Wu (1998). "An improved formulation for constrained mechanical systems". *Multibody System Dynamics* **2**:1, pp. 49–69.

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.

Åström, K. J. and B. Wittenmark (2011). *Computer-Controlled Systems: Theory and Design*. Courier Dover Publications, Mineola, New York.

Bauchau, O. A. and A. Laulusa (2008). "Review of contemporary approaches for constraint enforcement in multibody systems". *Journal of Computational and Nonlinear Dynamics* **3**:1, p. 011005.

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.

Baumrucker, B. T. and L. T. Biegler (2009). "MPEC strategies for optimization of a class of hybrid dynamic systems". *Journal of Process Control* **19**:8, pp. 1248–1256.

Becerra, V. M. (2010a). "Solving complex optimal control problems at no cost with PSOPT". In: *Proc. IEEE Multi-conf. Systems and Control, Sep. 7–10*. Yokohama, Japan, pp. 1391–1396.

Becerra, V. (2010b). *PSOPT optimal control solver user manual (release 3)*. Available: `http://code.google.com/p/psopt/downloads/list`.

Bellman, R. and R. E. Kalaba (1965). *Dynamic Programming and Modern Control Theory*. Academic Press, New York.

Belotti, P., J. Lee, L. Liberti, F. Margot, and A. Wächter (2009). "Branching and bounds tighteningtechniques for non-convex MINLP". *Optimization Methods & Software* **24**:4–5, pp. 597–634.

Ben-Tal, A., L. El Ghaoui, and A. Nemirovski (2009). *Robust Optimization*. Princeton University Press, Princeton, NJ.

Betts, J. T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd. SIAM, Philadelphia, PA.

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.

Billard, A., S. Calinon, R. Dillmann, and S. Schaal (2008). "Robot programming by demonstration". In: *Handbook of Robotics*. Springer, Berlin Heidelberg, pp. 1371–1394.

Bloch, A. M. (2003). *Nonholonomic Mechanics and Control*. Vol. 24. Springer Science & Business Media, New York.

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", IEEE Int. Conf. Robotics and Automation (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.

Borghesan, G., B. Willaert, T. De Laet, and J. De Schutter (2012a). "Teleoperation in presence of uncertainties: a constraint-based approach". In: *Proc. 10th IFAC Symposium on Robot Control*, pp. 385–392.

Borghesan, G., B. Willaert, and J. De Schutter (2012b). "A constraint-based programming approach to physical human-robot interaction". In: *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3890–3896.

Böttcher, A. and S. M. Grudsky (2005). *Spectral Properties of Banded Toeplitz Matrices*. SIAM, Philadelphia.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. 6th ed. Available for download: `https://web.stanford.edu/~boyd/cvxbook`. Cambridge University Press, Cambridge, UK.

Bristow, D., M. Tharayil, and A. Alleyne (2006). "A survey of iterative learning control". *IEEE Control Systems* **26**:3, pp. 96–114. ISSN: 1066-033X.

Brogliato, B. (2012). *Nonsmooth Mechanics: Models, Dynamics and Control*. 3rd ed. Springer Science & Business Media, Switzerland.

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

Chen, Y., J. Zhang, C. Yang, and B. Niu (2007). "The workspace mapping with deficient-DOF space for the PUMA 560 robot and its exoskeleton arm by using orthogonal experiment design method". *Robotics and Computer-Integrated Manufacturing* **23**:4, pp. 478–487.

Choset, H. M. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT press, Cambridge, MA.

Chotiprayanakul, P. and D. Liu (2009). "Workspace mapping and force control for small haptic device based robot teleoperation". In: *Proc. IEEE Int. Conf. Information and Automation, ICIA'09*, pp. 1613–1618.

Constantinescu, D. (2008). "The feel of virtual mechanisms". In: *Product Engineering*. Springer Netherlands, pp. 231–242.

Constantinescu, D., S. E. Salcudean, and E. A. Croft (2006). "Haptic manipulation of serial-chain virtual mechanisms". *J. Dynamic Systems, Measurement, and Control* **128**:1, pp. 65–74.

Corke, P. I. (2011). *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, Berlin, Heidelberg. ISBN: 978-3-642-20143-1.

Cuadrado, J., J. Cardenal, and E. Bayo (1997). "Modeling and solution methods for efficient real-time simulation of multibody dynamics". *Multibody System Dynamics* **1**:3, pp. 259–280.

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.

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.

Denavit, J. and R. S. Hartenberg (1955). "A kinematic notation for lower-pair mechanisms based on matrices." *Trans. of the ASME. J. Applied Mechanics* **22**, pp. 215–221.

Dulęba, I. (1997). "Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution". *Robotica* **15**:5, pp. 555–562.

Dymola (2016). *Multi-Engineering Modeling and Simulation - Dymola - CATIA*. URL: `http://www.3ds.com/products-services/catia/products/dymola` (visited on 2016-09-01).

Elci, H., R. W. Longman, M. Q. Phan, J.-N. Juang, and R. Ugoletti (2002). "Simple learning control made practical by zero-phase filtering: applications to robotics". *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications* **49**:6, pp. 753–767.

Erez, T., K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov (2013). "An integrated system for real-time model predictive control of humanoid robots". In: *13th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*. IEEE, pp. 292–299.

Felis, M. L. and K. Mombaur (2016). "Synthesis of full-body 3-D human gait using optimal control methods". In: *Proc. Int. Conf. Robotics and Automation (ICRA)*. Stockholm, Sweden, pp. 1560–1566.

Flash, T. and N. Hogan (1985). "The coordination of arm movements: an experimentally confirmed mathematical model". *J. Neuroscience* **5**:7, pp. 1688–1703.

Fouquet, M., H. Guéguen, D. Faille, and D. Dumur (2014). "Hybrid dynamic optimization of power plants using sum-up rounding and adaptive mesh refinement". In: *Proc. IEEE Conf. Control Applications*, pp. 316–321.

Freeman, C. T., E. Rogers, A.-M. Hughes, J. H. Burridge, and K. L. Meadmore (2012). "Iterative learning control in health care: electrical stimulation and robotic-assisted upper-limb stroke rehabilitation". *IEEE Control Systems* **32**:1, pp. 18–43.

Freidovich, L., A. Robertsson, A. Shiriaev, and R. Johansson (2008). "Periodic motions of the pendubot via virtual holonomic constraints: theory and experiments". *Automatica* **44**:3, pp. 785–791.

Fritzson, P. (2015). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3*. 2nd edition. Wiley-IEEE Press, Piscataway, NJ.

Ghazaei Ardakani, M. M. (2015). *Topics in Trajectory Generation for Robots*. Licentiate Thesis TFRT--3265--SE. Dept. Automatic Control, Lund University, Sweden.

Ghazaei Ardakani, M. M., B. Olofsson, A. Robertsson, and R. Johansson (2015). "Real-time trajectory generation using model predictive control". In: *Proc. IEEE Int. Conf. Auto. Sci. Eng. (CASE), Gothenburg*. DOI: 10.1109/CoASE.2015.7294220, pp. 942–948.

Glad, T. and L. Ljung (2000). *Control Theory: Multivariable and Nonlinear Methods*. Taylor & Francis, London. ISBN: 0-7484-0878-9.

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.

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

Griewank, A. and A. Walther (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia.

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), Sep. 22-26, 2008*. Nice, France, pp. 3248–3253.

Hatanaka, T., N. Chopra, and M. W. Spong (2015). "Passivity-based control of robots: historical perspective and contemporary issues". In: *Proc. 54th, IEEE Conf. Decision and Control (CDC), Osaka, Japan*. IEEE, pp. 2450–2452.

Hehn, M. and R. D'Andrea (2011). "Quadrocopter trajectory generation and control". In: *Proc. of the 18th IFAC World Congress, Aug. 28–Sep. 2, 2011*. Vol. 18. 1. Milano, Italy, pp. 1485–1491.

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.

Horn, R. A. and C. R. Johnson (2012). *Matrix Analysis*. Cambridge University Press, New York, NY 10013-2473, USA.

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.

HSL (2016). *A collection of Fortran codes for large scale scientific computation*. URL: http://www.hsl.rl.ac.uk (visited on 2016-09-01).

Hunt, K. and F. Crossley (1975). "Coefficient of restitution interpreted as damping in vibroimpact". *J. Applied Mechanics* **42**:2, pp. 440–445.

Isidori, A. (1995). *Nonlinear Control Systems*. 3rd. Springer-Verlag, London.

JModelica.org (2015). *Open source Modelica platform for modeling, simulation and optimization*. `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.

Kirches, C. and F. Lenders (2016). "Approximation properties and tight bounds for constrained mixed-integer optimal control". *Mathematical Programming*. Submitted for publication. Available online: `http://www.optimization-online.org/DB_FILE/2016/04/5404.pdf`.

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.

Kreyszig, E. (1989). *Introductory Functional Analysis with Applications*. Vol. 81. Wiley, New York.

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. 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. Robotics* **26**:1, pp. 94–111.

Kumar, V., E. Todorov, and S. Levine (2016). "Optimal control with learned local models: application to dexterous manipulation". In: *Proc. IEEE Int. Conf. Rob. and Auto. (ICRA)*. Stockholm, Sweden, pp. 378–383.

Kumar, V., Y. Tassa, T. Erez, and E. Todorov (2014). "Real-time behaviour synthesis for dynamic hand-manipulation". In: *Proc. IEEE Int. Conf. Rob. and Auto. (ICRA)*. Hong Kong, China, pp. 6808–6815.

Kurek, J. and M. Zaremba (1993). "Iterative learning control synthesis based on 2-D system theory". *IEEE Trans. Automatic Control* **38**:1, pp. 121–125. ISSN: 0018-9286.

LabComm (2015). *LabComm protocol*. Dept. Automatic Control, Lund University. URL: `http://www.control.lth.se/Research/tools.html` (visited on 2016-09-01).

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. Autom. (ICRA), 2011*. Shanghai, China, pp. 3719–3726.

Lang, S. (2013). *Complex Analysis*. Graduate Texts in Mathematics. Springer, New York. ISBN: 9781475730838.

Laulusa, A. and O. A. Bauchau (2008). "Review of classical approaches for constraint enforcement in multibody systems". *Journal of Computational and Nonlinear Dynamics* **3**:1, p. 011004.

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.

Lawson, C. L. and R. J. Hanson (1995). *Solving Least Squares Problems*. Vol. 15. SIAM, Philadelphia.

Lee, D. and M. Spong (2006). "Passive bilateral teleoperation with constant time delay". *IEEE Trans. Robotics* **22**:2, pp. 269–281.

Lee, S. H., I. H. Suh, S. Calinon, and R. Johansson (2015). "Autonomous framework for segmenting robot trajectories of manipulation task". *Autonomous Robots* **38**:2, pp. 107–141.

Liarokapis, M. V., P. K. Artemiadis, and K. J. Kyriakopoulos (2012). "Functional anthropomorphism for human to robot motion mapping". In: *IEEE Int. Symp. Robot and Human Interactive Communication (RO-MAN),* IEEE, pp. 31–36.

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.

Lin, H. and P. J. Antsaklis (2014). "Hybrid dynamical systems: an introduction to control and verification". *Foundations and Trends in Systems and Control* **1**:1, pp. 1–172. ISSN: 2325-6818. DOI: 10.1561/2600000001.

Linderoth, M. (2013). *On Robotic Work-Space Sensing and Control*. Thesis No. TFRT–1098–SE. PhD thesis. Dept. 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. Autom. (ICRA), 2010*. Anchorage, AK, pp. 4525–4530.

Linderoth, M., A. Stolt, A. Robertsson, and R. Johansson (2013). "Robotic force estimation using motor torques and modeling of low velocity friction disturbances". In: *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 3550–3556.

Longman, R. W. (2000). "Iterative learning control and repetitive control for engineering practice". *Int. J. Control* **73**:10, pp. 930–954.

Longman, R. W. and Y.-C. Huang (2002). "The phenomenon of apparent convergence followed by divergence in learning and repetitive control". *Intell. Autom. & Soft Comput.* **8**:2, pp. 107–128.

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. Control* **4**:1, pp. 139–152.

Martin, B. J. and J. E. Bobrow (1997). "Minimum effort motions for open chain manipulators with task-dependent end-effector constraints". In: *Proc. IEEE Int. Conf. Rob. and Autom., Apr. 20–25, 1997*. Vol. 3. Albuquerque, NM, pp. 2044–2049.

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

Mattsson, S. E., H. Olsson, and H. Elmqvist (2000). "Dynamic selection of states in dymola". In: *Proc. Modelica Workshop*, pp. 61–67.

Mattsson, S. E., M. Otter, and H. Elmqvist (2015). "Multi-mode DAE systems with varying index". In: *Proc. 11th Int. Modelica Conf.* Paris, France, pp. 89–98.

Mattsson, S. E. and G. Söderlind (1993). "Index reduction in differential-algebraic equations using dummy derivatives". *SIAM J. Scientific Computing* **14**:3, pp. 677–692.

Maurer, H. (1977). "On optimal control problems with bounded state variables and control appearing linearly". *SIAM J. Control and Optimization* **15**:3, pp. 345–362.

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.

Meriam, J. L. and L. G. Kraige (2012). *Engineering Mechanics: Dynamics*. 7th ed. Vol. 2. John Wiley & Sons, New Jersey, USA.

Mishra, S., J. Coaplen, and M. Tomizuka (2007). "Precision positioning of wafer scanners segmented iterative learning control for nonrepetitive disturbances [applications of control]". *IEEE Control Systems* **27**:4, pp. 20–25.

Moore, K. L., M. Dahleh, and S. Bhattacharyya (1989). "Learning control for robotics". In: *Advances in Computing and Control*. Springer Berlin Heidelberg, pp. 240–251.

Mordatch, I., K. Lowrey, and E. Todorov (2015). "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Rob. and Sys. (IROS)*, pp. 5307–5314.

Müller, A. (2014). "Implementation of a geometric constraint regularization for multibody system models". *Archive of Mechanical Engineering* **61**:2, pp. 365–383.

Nocedal, J. and S. Wright (2006). *Numerical Optimization*. 2nd edition. Springer, New York, NY.

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. Linköping university, Linköping, Sweden.

Norrlöf, M. (2000). *Iterative Learning Control: Analysis, Design, and Experiments*. Thesis No. 653. Dep. E.E., Linköping university, Linköping, Sweden.

Norrlöf, M. and S. Gunnarsson (2002). "Time and frequency domain convergence properties in iterative learning control". *Int. J. Control* **75**:14, pp. 1114–1126.

Nottensteiner, K., T. Bodenmueller, M. Kassecker, M. A. Roa, A. Stemmer, T. Stouraitis, D. Seidel, and U. Thomas (2016). "A complete automated chain for flexible assembly using recognition, planning and sensor-based execution". In: *Proc. 47th Int. Symp. Rob. (ISR)*. Munich, Germany, pp. 1–8.

Otter, M., H. Elmqvist, and S. E. Mattsson (2003). "The new Modelica multibody library". In: *Proc. 3rd Int. Modelica Conf.* Linköping, Sweden, pp. 311–330.

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. Accession no. AD0785071. 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.

Privat Deschanel, A. (1884). *Elementary Treatise on Natural Philosophy Part I. Mechanics, Hydrostatics, and Pneumatics*. Appleton and Company, New York. The original illustration retrieved Nov. 1, 2016, from http://etc.usf.edu/clipart/galleries/236-pulleys.

Rebelo, J. and A. Schiele (2012). "Master-slave mapping and slave base placement optimization for intuitive and kinematically robust direct teleoperation". In: *12th Int. Conf. Control, Automation and Systems (ICCAS)*. IEEE, pp. 2017–2022.

Reichel, L. and L. N. Trefethen (1992). "Eigenvalues and pseudo-eigenvalues of Toeplitz matrices". *Linear algebra and its applications* **162**, pp. 153–185.

Rombokas, E., M. Malhotra, E. A. Theodorou, E. Todorov, and Y. Matsuoka (2013). "Reinforcement learning and synergistic control of the act hand". *IEEE/ASME Trans. Mechatronics* **18**:2, pp. 569–577.

Roover, D. de (1996). "Synthesis of a robust iterative learning controller using an H_∞ approach". In: *Proc. 35th IEEE Conf. Decision and Control*. Vol. 3. Kobe, Japan, pp. 3044–3049.

Ruan, X., Z. Z. Bien, and K.-H. Park (2008). "Decentralized iterative learning control to large-scale industrial processes for nonrepetitive trajectory tracking". *IEEE Tran. Systems, Man, and Cybernetics-Part A: Systems and Humans* **38**:1, pp. 238–252.

Rugh, W. J. (1996). *Linear System Theory*. Prentice Hall, Upper Saddle River, NJ.

Salvietti, G., L. Meli, G. Gioioso, M. Malvezzi, and D. Prattichizzo (2013). "Object-based bilateral telemanipulation between dissimilar kinematic structures". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5451–5456.

Saut, J.-P., A. Sahbani, and V. Perdereau (2011). "Generic motion planner for robot multi-fingered manipulation". *Advanced Robotics* **25**:1-2, pp. 23–46.

Schmidt, P. and F. Spitzer (1960). "The Toeplitz matrices of an arbitrary Laurent polynomial". *Math. Scand.* **8**, pp. 15–28.

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.

Shi, W., D. Stapersma, and H. T. Grimmelius (2008). "Comparison study on moving and transportation performance of transportation modes". *Int J Energy Environ* **2**:4, pp. 106–20.

Shiller, Z. (1994a). "Time-energy optimal control of articulated systems with geometric path constraints". In: *Proc. IEEE Int. Conf. Rob. and Autom., May 8-13, 1994*. San Diego, CA, pp. 2680–2685.

Shiller, Z. (1994b). "Time-energy optimal control of articulated systems with geometric path constraints". In: *Proc. IEEE Int. Conf. Robot. and Autom. (ICRA)*. Vol. 4. San Diego, CA USA.

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.

Shiriaev, A. S., L. B. Freidovich, A. Robertsson, R. Johansson, and A. Sandberg (2007). "Virtual-holonomic-constraints-based design of stable oscillations of furuta pendulum: theory and experiments". *IEEE Trans. Robotics* **23**:4, pp. 827–832.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009). *Robotics: Modelling, Planning and Control*. Springer Verlag, London.

Sörnmo, O., B. Bernhardsson, O. Kröling, P. Gunnarsson, and R. Tenghamn (2016). "Frequency-domain iterative learning control of a marine vibrator". *Control Engineering Practice* **47**, pp. 70–80.

Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006). *Robot Modeling and Control*. Vol. 3. Wiley, New York.

Stolt, A., F. B. Carlson, M. M. Ghazaei Ardakani, I. Lundberg, A. Robertsson, and R. Johansson (2015a). "Sensorless friction-compensated passive lead-through programming for industrial robots". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. Hamburg, Germany, pp. 3530–3537.

Stolt, A., A. Robertsson, and R. Johansson (2015b). "Robotic force estimation using dithering to decrease the low velocity friction uncertainties". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Seattle, USA, pp. 3896–3902.

Stryk, O. von and M. Schlemmer (1994). "Optimal control of the industrial robot manutec R3". In: Bulirsch, R. et al. (Eds.). *Computational Optimal Control*. Birkhauser Verlag, Basel, Switzerland, pp. 367–382. ISBN: 0-8176-5015-6.

Sundararajan, D. (2001). *The Discrete Fourier Transform: Theory, Algorithms and Applications*. World Scientific, Singapore. ISBN: 9789812810298.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT press, Cambridge, MA.

Tassa, Y., T. Erez, and E. Todorov (2012). "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *IEEE/RSJ Int. Conf. Intell. Rob. and Sys. (IROS)*. Vilamoura, Portugal, pp. 4906–4913.

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.

Thelander Andrén, M. and A. Cervin (2016). "Event-based state estimation using an improved stochastic send-on-delta sampling scheme". In: *Proc. IEEE 2nd Int. Conf. Event-Based Control, Communication and Signal Processing (EBCCSP)*, pp. 1–8.

Thieriot, H., M. Nemura, M. Torabzadeh-Tari, P. Fritzson, R. Singh, and J. J. Kocherry (2011). "Towards design optimization with OpenModelica emphasizing parameter optimization with genetic algorithms". In: *Proc. 8th Int. Modelica Conf.* Dresden, Germany, pp. 756–762.

Thomas, U., T. Stouraitis, and M. A. Roa (2015). "Flexible assembly through integrated assembly sequence planning and grasp planning". In: *Proc. IEEE Int. Conf. Auto. Sci. Eng. (CASE)*. Gothenburg, Sweden, pp. 586–592.

Tolani, D., A. Goswami, and N. I. Badler (2000). "Real-time inverse kinematics techniques for anthropomorphic limbs". *Graphical Models* **62**:5, pp. 353–388.

Vandenberghe, L., V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh (2005). "Interior-point algorithms for semidefinite programming problems derived from the KYP lemma". In: Henrion, D. et al. (Eds.). *Positive Polynomials in Control*. Springer, Berlin, Heidelberg, pp. 195–238. ISBN: 978-3-540-31594-0.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2008). "Time-energy optimal path tracking for robots: a numerically efficient optimization approach". In: *10th Int. Workshop Advanced Motion Control*. Trento, Italy.

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.

Wächter, A. and L. T. Biegler (2006). "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming". *Math. Program.* **106**, pp. 25–57.

Wahrburg, A., S. Zeiss, B. Matthias, and H. Ding (2014). "Contact force estimation for robotic assembly using motor torques". In: *Proc. IEEE Int. Conf. Auto. Sci. Eng. (CASE)*. Gothenburg, Sweden, pp. 1252–1257.

Wang, D., Y. Ye, and B. Zhang (2014). *Practical Iterative Learning Control with Frequency Domain Design and Sampled Data Implementation*. Springer, Singapore.

Watkins, C. J. and P. Dayan (1992). "Q-learning". *Machine learning* **8**:3, pp. 279–292.

Westervelt, E. R., J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris (2007). *Feedback Control of Dynamic Bipedal Robot Locomotion*. Vol. 28. CRC press, Boca Raton, FL.

Yamada, T., M. Yamada, and H. Yamamoto (2012). "Stability analysis of multiple objects grasped by multifingered hands with revolute joints in 2D". In: *Proc. IEEE Int. Conf. Mechatronics and Automation (ICMA)*. Chengdu, China, pp. 1785–1792.

Yamada, T. and H. Yamamoto (2013). "Grasp stability analysis of multiple objects including contact surface geometry in 3D". In: *Proc. IEEE Int. Conf. Mechatronics and Automation (ICMA)*. Takamatsu, Japan, pp. 36–43.

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.

Zahariev, E. and J. Cuadrado (2011). "Dynamics of mechanisms in over constrained and singular configurations". *Journal of Theoretical and Applied Mechanics* **41**:1, pp. 3–18.

A

Supplementary Equations for Ball and Finger System

A.1 Forward kinematics and Jacobian

Considering Fig. 7.1, the position of the fingertip, p_e, and the orientation of the 3rd link, described as a rotation matrix, are

$$p_e = \begin{bmatrix} -s_{23} - s_2 + 2 \\ \frac{1}{2}\left(s_{123} + s_{1\bar{2}\bar{3}} + s_{12} + s_{1\bar{2}}\right) + s_1 \\ \frac{1}{2}\left(c_{1\bar{2}\bar{3}} + c_{123} + c_{1\bar{2}} + c_{12}\right) + c_1 \end{bmatrix}, \tag{A.1}$$

$$[\hat{e}_{x_3}\ \hat{e}_{y_3}\ \hat{e}_{z_3}] = \begin{bmatrix} -s_{23} & -c_{23} & 0 \\ \frac{1}{2}\left(s_{123} + s_{1\bar{2}\bar{3}}\right) & \frac{1}{2}\left(c_{123} - c_{1\bar{2}\bar{3}}\right) & -c_1 \\ \frac{1}{2}\left(c_{1\bar{2}\bar{3}} + c_{123}\right) & \frac{1}{2}\left(s_{1\bar{2}\bar{3}} - s_{123}\right) & s_1 \end{bmatrix}, \tag{A.2}$$

where $s_{ij\ldots k}$ denotes $\sin(q_i + q_j + \cdots + q_k)$ and $c_{ij\ldots k}$ denotes $\cos(q_i + q_j + \cdots + q_k)$. A variable with a bar denotes the variable with a negative sign, e.g., $\bar{i}$ represents $-q_i$.

The geometric Jacobian for the fingertip is

$$J = \begin{bmatrix} J_p \\ J_o \end{bmatrix}, \tag{A.3}$$

where the translational part, J_p, is given by

$$J_p = \begin{bmatrix} 0 & -c_{23} - c_2 & -c_{23} \\ J_{21} & \frac{1}{2}\left(c_{123} - c_{1\bar{2}\bar{3}} - c_{1\bar{2}} + c_{12}\right) & \frac{1}{2}\left(c_{123} - c_{1\bar{2}\bar{3}}\right) \\ J_{31} & \frac{1}{2}\left(s_{1\bar{2}\bar{3}} - s_{123} - s_{12} + s_{1\bar{2}}\right) & -\frac{1}{2}\left(s_{1\bar{2}\bar{3}} - s_{123}\right) \end{bmatrix}, \tag{A.4}$$

where

$$J_{21} = \frac{1}{2}\left(c_{1\bar{2}\bar{3}} + c_{123} + c_{1\bar{2}} + c_{12}\right) + c_1,$$

$$J_{31} = -\frac{1}{2}\left(s_{123} + s_{1\bar{2}\bar{3}} + s_{12} + s_{1\bar{2}}\right) - s_1,$$

and the rotational part, J_o, is given by

$$J_o = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -c_1 & -c_1 \\ 0 & s_1 & s_1 \end{bmatrix}. \tag{A.5}$$

A.2 Dynamics

In this section, the expressions for the dynamical properties of the finger used in (7.23) are given. Every link is characterized by the mechanical properties mass m_i and moments of inertia $I_i = \text{diag}\left(I_{xx_i}, I_{yy_i}, I_{zz_i}\right)$ where i denotes the link number.

The expression for the gravitational force is

$$g(q) = -\frac{g_0}{4}\begin{bmatrix} g_{11} \\ m_2\left(s_{11} - s_{1\bar{1}}\right) + \left(s_{111} - s_{1\bar{1}\bar{1}} + 2s_{11} - 2s_{1\bar{1}}\right) m_3 \\ m_3\left(s_{111} - s_{1\bar{1}\bar{1}}\right) \end{bmatrix}, \tag{A.6}$$

$$\begin{aligned} g_{11} = {} & 2m_1 s_1 + \left(s_{11} + s_{1\bar{1}} + 4s_1\right) m_2 \\ & + \left(s_{111} + s_{1\bar{1}\bar{1}} + 2s_{11} + 2s_{1\bar{1}} + 4s_1\right) m_3. \end{aligned} \tag{A.7}$$

Here s_{xyz} denotes $\sin(xq_1 + yq_2 + zq_3)$ and c_{xyz} denotes $\cos(xq_1 + yq_2 + zq_3)$. A variable with a bar denotes the variable with a negative sign, e.g., $\bar{x}$ represents $-x$. The trailing zeros are omitted.

The mass matrix for the finger is

$$\begin{aligned} M_f(q) &= M_f^T(q) \\ &= \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_3 c_{001} + \frac{5}{4}m_3 + \frac{1}{4}m_2 + I_{zz_2} + I_{zz_3} & \frac{1}{4}\left(2c_{001} + 1\right) m_3 + I_{zz_3} \\ 0 & \frac{1}{4}\left(2c_{001} + 1\right) m_3 + I_{zz_3} & \frac{1}{4}m_3 + I_{zz_3} \end{bmatrix}, \end{aligned} \tag{A.8}$$

where

$$\begin{aligned} m_{11} = {} & \frac{1}{8}\left(-4I_{xx_3} + 4I_{yy_3} + m_3\right) c_{022} + \frac{1}{2}m_3 c_{021} + \frac{1}{8}\left(8c_{01} + 9\right) m_2 \\ & + \frac{1}{8}\left(4m_3 + m_2 - 4I_{xx_2} + 4I_{yy_2}\right) c_{020} + \frac{1}{2}\left(I_{yy_2} + I_{xx_3} + I_{yy_3} + I_{xx_2}\right) \\ & + \frac{1}{8}\left(16c_{01} + 4c_{001} + 8c_{011} + 13\right) m_3 + I_{yy_1} + \frac{1}{4}m_1. \end{aligned} \tag{A.9}$$

The matrix $C(q, \dot{q})$ can be expressed as

$$C_{ij} = \sum_{k=1}^{3} \psi_{ijk}\dot{q}_k, \tag{A.10}$$

where $\psi_{ijk} = 0$ except for

$$\begin{aligned}\psi_{112} = \psi_{121} = -\psi_{211} = {} & \frac{1}{8}\left(4I_{xx_3} - 4I_{yy_3} - m_3\right)s_{022} - \frac{1}{2}m_3 s_{021} \\ & - \frac{1}{8}\left(4m_3 + m_2 - 4I_{xx_2} + 4I_{yy_2}\right)s_{02} - \frac{1}{2}m_3 s_{011} \\ & - \left(\frac{1}{2}m_2 + m_3\right)s_{01}, \end{aligned} \tag{A.11}$$

$$\begin{aligned}\psi_{113} = \psi_{131} = -\psi_{311} = {} & \frac{1}{8}\left(4I_{xx_3} - 4I_{yy_3} - m_3\right)s_{022} \\ & - \frac{1}{4}m_3\left(2s_{011} + s_{001} + s_{021}\right), \end{aligned} \tag{A.12}$$

$$\psi_{223} = \psi_{232} = \psi_{233} = -\psi_{322} = -\frac{1}{2}m_3 s_{001}. \tag{A.13}$$