On the Time-optimal Trajectory Planning along Predetermined Geometric Paths and Optimal Control Synthesis for Trajectory Tracking of Robot Manipulators

by

Pedro Reynoso Mora

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Roberto Horowitz
Professor Pieter Abbeel

Fall 2013

On the Time-optimal Trajectory Planning along Predetermined Geometric Paths and Optimal Control Synthesis for Trajectory Tracking of Robot Manipulators

Abstract

On the Time-optimal Trajectory Planning along Predetermined Geometric Paths and Optimal Control Synthesis for Trajectory Tracking of Robot Manipulators

by

Pedro Reynoso Mora

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

In this dissertation, we study two important subjects in robotics: (i) time-optimal trajectory planning, and (ii) optimal control synthesis methodologies for trajectory tracking. In the first subject, we concentrate on a rather specific sub-class of problems, the time-optimal trajectory planning along predetermined geometric paths. In this kind of problem, a purely geometric path is already known, and the task is to find out how to move along this path in the shortest time physically possible. In order to generate the true fastest solutions achievable by the actual robot manipulator, the complete nonlinear dynamic model should be incorporated into the problem formulation as a constraint that must be satisfied by the generated trajectories and feedforward torques. This important problem was studied in the 1980s, with many related methods for addressing it based on the so-called velocity limit curve and variational methods. Modern formulations directly discretize the problem and obtain a large-scale mathematical optimization problem, which is a prominent approach to tackle optimal control problems that has gained popularity over variational methods, mainly because it allows to obtain numerical solutions for harder problems.

We contribute to the referred problem of time-optimal trajectory planning, by extending and improving the existing mathematical optimization formulations. We successfully incorporate the complete nonlinear dynamic model, including viscous friction because for the fastest motions it becomes even more significant than Coulomb friction; of course, Coulomb friction is likewise accommodated for in our formulation. We develop a framework that guarantees exact dynamic feasibility of the generated time-optimal trajectories and feedforward torques. Our initial formulation is carefully crafted in a rather specific manner, so that it allows to naturally propose a convex relaxation that solves exactly the original problem formulation, which is non-convex and therefore hard to solve. In order to numerically solve the proposed formulation, a discretization scheme is also developed. Unlike traditional and modern formulations, we motivate the incorporation of additional criteria to our original formulation, with simulation and experimental studies of three crucial variables for a 6-axis industrial manipulator. Namely, the resulting applied torques, the readings of a 3-axis ac-

celerometer mounted at the manipulator end-effector, and the detrimental effects on the tracking errors induced by pure time-optimal solutions. We therefore emphasize the significance of penalizing a measure of total jerk and of imposing acceleration constraints. These two criteria are incorporated without destroying convexity. The final formulation generates near time-optimal trajectories and feedforward torques with traveling times that are slightly larger than those of pure time-optimal solutions. Nevertheless, the detrimental effects induced by pure time-optimality are eliminated. Experimental results on a 6-axis industrial manipulator confirm that our formulation generates the fastest solutions that can actually be implemented in the real robot manipulator.

Following the work done on near time-optimal trajectories, we explore two controller synthesis methodologies for trajectory tracking, which are more suitable to achieve trajectory-tracking under such fast trajectories. In the first approach, we approximate the discrete-time nonlinear dynamics of robot manipulators, moving along the state-reference trajectory, as an affine time-varying (ATV) dynamical system in discrete-time. Therefore, the problem of trajectory tracking for robot manipulators is posed as a linear quadratic (LQ) optimal control problem for a class of discrete-time ATV dynamical systems. Then, an ATV control law to achieve trajectory tracking on the ATV system is developed, which uses LQ methods for linear time-varying (LTV) systems. Since the ATV dynamical system approximates the nonlinear robot dynamics along the state-reference trajectory, the resulting time-varying control law is suitable to achieve trajectory tracking on the robot manipulator. The ATV control law is implemented in experiments for the 6-axis industrial manipulator, tracking the near time-optimal trajectory. Experimental results verify the better performance achieved with the ATV control law, but also expose its shortcomings.

The second approach to address trajectory tracking is related in spirit, but different in crucial aspects, which ultimately endow this approach with its superior features. In this novel approach, the highly nonlinear dynamic model of robot manipulators, moving along a state-reference trajectory, is approximated as a class of piecewise affine (PWA) dynamical systems. We propose a framework to construct the referred PWA system, which consists in: (i) choosing strategic operating points on the state-reference trajectory with their respective (local) linearized system dynamics, (ii) constructing ellipsoidal regions centered at the operating points, whose purpose is to facilitate the scheduling strategy of controller gains designed for each local dynamics. Likewise, in order to switch controller gains as the robot state traverses in the direction of the state-reference trajectory, a simple scheduling strategy is proposed. The controller synthesis near each operating point is an LQR-type that takes into account the local coupled dynamics. The referred PWA control law is implemented in experiments for the 6-axis manipulator tracking the near time-optimal trajectory. The experimental results show the feasibility and superiority of the PWA control law over the typical PID controller and the ATV control law.

Dedicated to my ever growing family...

Contents

List of Figures

List of Tables

Acknowledgments

I want to thank all the people who made it possible for me to conclude my Ph.D. studies at Berkeley. First of all, my sincerest thanks to Professor Masayoshi Tomizuka, for allowing me to join his research group and for being so patient with me in regards to concluding this dissertation. Likewise, Professor Tomizuka encouraged me to continue pursuing my goals in the Ph.D. program during the most difficult times I experienced in Berkeley. At this stage of my life, there are no words to express my gratitude.

I would also like to thank Professors Roberto Horowitz and Pieter Abbeel for their willingness to read this dissertation and for their critical but constructive comments to improve the final version. The course and lecture notes on Advanced Control Systems II, which I took under Professor Roberto Horowitz, heavily influenced my view on the subject of control systems. Professor Pieter Abbeel's course CS287 Advanced Robotics, which I took in the Fall 2011, has noticeably influenced the content of this dissertation. Using terms such as "dynamically feasible trajectory", is an instance of this influence.

Special acknowledgments to my sponsors, the National Council for Science and Technology of Mexico (CONACYT), and the University of California Institute for Mexico and the United States (UC MEXUS). These two institutions provided me with a five-year Ph.D. fellowship, in a joint effort to provide support to Mexican students who wish to pursue a Ph.D. in the University of California system. Likewise, the financial and technical supports from FANUC Corporation are acknowledged. The developments presented in this dissertation would be of little value without the experimental evaluations on the real robot manipulator, which was kindly donated by FANUC Corporation.

I greatly benefited from several courses offered at Berkeley. The most influential on my carrier are: Advanced Control Systems I/II taught respectively by Professor M. Tomizuka and Professor R. Horowitz, Intermediate/Advanced Dynamics by Professor Oliver O'Reilly, Control of Nonlinear Dynamic Systems by Professor Karl Hedrick, Convex Optimization by Professor Laurent El Ghaoui, Advanced Robotics by Professor Pieter Abbeel, and the final course I took at Berkeley, Hybrid Systems: computation and control, taught jointly by Professors Claire Tomlin and Shankar Sastry. I am very honored for having had the opportunity to be exposed to those topics from such respected and bright people.

It goes without saying that I have also benefited enormously from being a member of the Mechanical Systems Control Laboratory, MSC Lab for short. I want to thank all the members of the MSC Lab who were there before and after I joined. Special thanks to Wenjie Chen, Sanggyum Kim, Evan Chang-Siu, Mike Chan, and Takashi Nagata. I had many fruitful discussions on my research with Wenjie Chen and Sanggyum Kim. Wenjie's critical comments definitely influenced my thinking to improve the final results.

Last and foremost, I want to thank my ever-growing family. In Mexico, my sisters Patricia and Angélica, my brothers Arturo and Enrique, my grandfather Pedro, and my parents Alejandra and Eduardo. Thank you Mom for all your sacrifices so that I could reach my dreams. In the USA, my fiancée Britta, for your love and encouragement, and for giving me the greatest gift I could ever receive, our wonderful daughter Camila.

Chapter 1

Introduction

The field of robotics is an active research field that deals with the study of machines that can replace human beings in the execution of tasks. The term robot, which derives from the term robota that means executive labor, was coined by the Czech play-writer Karel Čapek, who wrote the play *Rossum's Universal Robots* in 1920 [1]. The concept of a robot has changed substantially over time, from the fictitious idea of a superhuman machine, to the reality of automated machines that perform a large variety tasks in industrial sectors. An important class of these automated machines are robot manipulators, which are the main subject of this dissertation. Robot manipulators are widely seen in many industrial sectors, such as the automotive industry [1, 2].

In this dissertation, two important topics for robot manipulators are addressed: (i) time-optimal trajectory planning, and (ii) optimal control synthesis methodologies for trajectory tracking. Essentially, these two topics together study how to generate the motion profiles, and how to make the robot manipulator perform those motion profiles to achieve specific tasks autonomously. The focus of this dissertation regarding trajectory planning is on a rather specific sub-problem, namely, the time-optimal trajectory planning along predetermined geometric paths. In this problem, it is assumed that a purely geometric path is already known, and essentially the task is to find out how to move optimally along that path so as to minimize the traveling time. On the other hand, the type of trajectory tracking algorithms we develop in this dissertation are based on optimal control synthesis techniques for special classes of linear systems. These control algorithms are targeted to achieve trajectory tracking for the near time-optimal trajectories, which are generated with our algorithms for time-optimal trajectory planning.

1.1 Time-optimal Trajectory Planning

In industrial applications, time-optimality of robot manipulators is crucial for maximizing robot productivity. In most applications of robot manipulators, such as palletizing and pick-and-place, an operator specifies a collision-free geometric path that the robot must follow

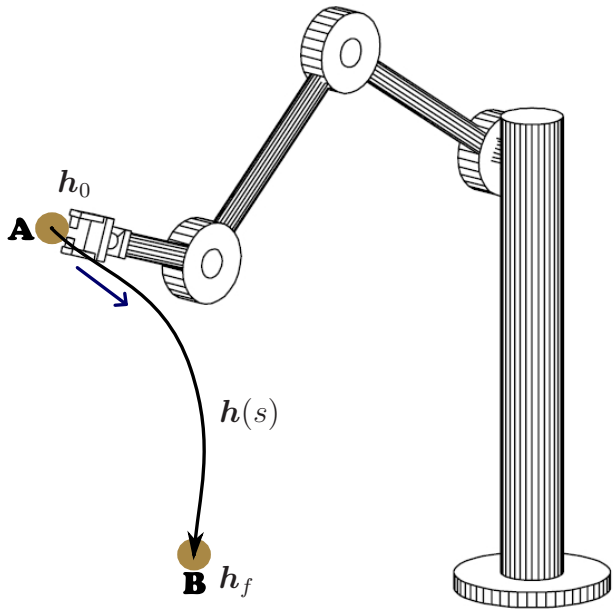


Figure 1.1: Illustration of purely geometric path.

in order to accomplish a particular task. This path specification is usually done through a so-called teach-pendant or through a path planning algorithm [3]. Once the geometric path has been specified, it is important to find out how to move the robot optimally along that path in the shortest time physically possible. Figure 1.1 illustrates the idea abstractly, where the robot manipulator is required to move from $\boldsymbol{A}$ to $\boldsymbol{B}$. In order to move from $\boldsymbol{A}$ to $\boldsymbol{B}$, it is assumed that a purely geometric path $\boldsymbol{h}(s)$ has been already obtained, where $\boldsymbol{h}_0$ and $\boldsymbol{h}_f$ represent respectively the geometric initial and final configurations. Therefore, the task is essentially to figure out the velocities, accelerations, and feedforward torques that guarantee motion along $\boldsymbol{h}(s)$ in minimum time. The feedforward torques are required to remain within physical bounds, i.e., actuator's torque limits. Additionally, these optimal motions are required to be feasible with respect to the nonlinear and coupled robot dynamics, which in this dissertation we shall refer to as dynamic feasibility.

1.1.1 Literature Review

First successful approaches to address this problem date back to the 1980s. Pioneering work is presented in the classic papers [4, 5, 6], where the minimum-time trajectory planning problem is developed for rigid manipulators that are fully actuated, governed by the following equations of motion:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ represents the joint positions for an n-degree-of-freedom manipulator, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of input torques, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is the positive-definite inertia matrix, and

$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ represents the combined effects of Coriolis/centrifugal forces and gravity forces. It is assumed that the actuators are subject to the following torque constraints:

$$\boldsymbol{\tau}_{\min}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{1.2}$$

The geometric path is assumed to be parameterized by a path parameter $s \in [s_0, s_f]$, and it is crucial that the joint positions, velocities, and accelerations can be expressed as functions of s, $\dot{s}$, and $\ddot{s}$:

$$\boldsymbol{q} = \boldsymbol{h}(s), \quad \dot{\boldsymbol{q}} = \dot{\boldsymbol{h}}(s, \dot{s}), \quad \ddot{\boldsymbol{q}} = \ddot{\boldsymbol{h}}(s, \dot{s}, \ddot{s}). \tag{1.3}$$

It is shown in these papers [4, 5, 6] that by combining (1.1)-(1.3), torque constraints can be expressed as constraints in the acceleration along the path $\ddot{s}$ as function of s and $\dot{s}$:

$$\ddot{s}_{\min}(s, \dot{s}) \leq \ddot{s} \leq \ddot{s}_{\max}(s, \dot{s}). \tag{1.4}$$

In [4] it is proposed that the time-optimal solution is to choose the acceleration to make the velocity as large as possible at every point without violating the constraints. It is then suggested that to minimize time, the acceleration always takes either its largest or its smallest possible value. Therefore, search algorithms were proposed that find switching points in the so-called velocity limit curve. At these switching points, instant changes from maximum acceleration to maximum deceleration and vice versa must occur. Another classic work on this problem is presented in [5] where additional properties of the velocity limit curve are rigourously analyzed, which allowed for simplification in the computation of the switching points. The development in [5] was apparently conceived in parallel to the work in [4]. Rather similar remarks are obtained in both papers, with subtle differences in the motivation, presentation, and proposed algorithms.

It is clear from (1.4) that no feasible acceleration along the path is possible when $\ddot{s}_{\min}(s, \dot{s}) > \ddot{s}_{\max}(s, \dot{s})$. The algorithms proposed in the classic papers [4, 5] both are based on the idea of avoiding the so-called inadmissible regions in the s-$\dot{s}$ plane, i.e., $\{(s, \dot{s}) \in \mathbb{R}^2_+ : \ddot{s}_{\min}(s, \dot{s}) > \ddot{s}_{\max}(s, \dot{s})\}$. The disadvantage is that searching for inadmissible regions over the entire plane s-$\dot{s}$ can become computationally rather expensive. Besides, pure time-optimal solutions produce torques that require sudden changes, which are impossible to handle by real servo-amplifiers whose bandwidth is limited. These techniques do not easily allow to incorporate additional criteria to generate solutions that are feasible for implementation.

Optimal control theory has also been used to tackle the referred problem of time-optimal trajectory planning, which in principle allows to incorporate additional criteria to produce solutions that are more feasible for implementation. In [6], additional criteria are added to trade off time-optimality against squared velocity and joint torques. To solve the optimization problem the authors apply a dynamic programming approach, where the plane s-$\dot{s}$ is discretized to obtain a two-dimensional grid. The benefit of the dynamic programming approach is clear, it naturally allows to include other performance criteria to generate solutions that are more feasible for implementations. The disadvantage of dynamic programming solution methods is the high computational burden associated with them.

Variational approaches in optimal control theory have also been applied to address the referred problem of time-optimal trajectory. In [7], time-optimality is traded off against a term that represents control energy. The method to solve the referred problem is based on variational methods, namely, the minimum principle of Pontryagin that requires solving a two-point boundary value problem using tedious shooting methods. Since this approach relies on shooting methods to solve the two-point boundary value problem, finding the optimal solution requires heavy computational burden.

Over the past decades, a prominent approach to address numerical optimal control problems has gained popularity over variational methods, mainly because it allows to obtain numerical solutions for harder problems, for which variational methods would simply be prohibitive [8]. In this approach, the optimal control problem is directly discretized to obtain a large-scale nonlinear optimization problem, allowing to impose not only equality constraints, but also more realistic and complicated inequality constraints, for which variational methods would fall short when attempting to obtain a numerical solution.

In the specific context of the problem of time-optimal trajectory planning along predefined paths, in [9] the problem is formulated as a large-scale nonlinear optimization problem. In that paper, a great deal of attention is paid to the adverse effects on the tracking errors induced by pure time-optimal solutions, and the authors incorporate inequality constrains on the torques time derivatives to produce smoother solutions. The formulation is a general nonlinear optimization, for which there are no guarantees for finding the global minimum.

More recently, motivated by the widespread reputation of convex optimization to efficiently solve engineering and science problems, a modern formulation of the time-optimal trajectory planning problem was proposed in [10], where theory and tools from convex optimization are utilized. The advantage of formulating a problem as a convex optimization problem is twofold: (i) theoretical and conceptual advantages, e.g., once an optimal solution is found, it represents the global optimal solution, (ii) the problem can be solved reliably and efficiently using mature interior-point methods or other special methods for convex optimization [11].

A main drawback of the approach presented in [10] is that viscous friction is neglected in order to have a convex and therefore tractable problem. This shortcoming has already been pointed out in [12]. It is clear that time-optimal trajectories represent the fastest trajectories, which implies that the manipulator will move at really high speeds. This means that the effects of viscous friction, which are velocity dependent, will be very significant (even more significant than Coulomb friction). Therefore, in order to really obtain the true fastest solutions that are dynamically feasible for the real robotic system, viscous friction should not be neglected. Nevertheless, none of the existing formulations take into account the complete dynamic model that includes both viscous and Coulomb friction.

1.1.2 Contributions

In this dissertation, we contribute to the referred problem of time-optimal trajectory planning, by extending and improving the existing mathematical optimization formulations.

More specifically:

1. We successfully incorporate the complete nonlinear dynamic model, including the effects of viscous friction. Clearly this is an important incorporation, since for the fastest motions, required by time-optimal solutions, viscous friction becomes even more significant than Coulomb friction. Notice nonetheless that Coulomb friction is likewise incorporated into our formulation.

2. Motivated by modern convex formulations with nice theoretical properties and efficient algorithms to solve them, we express explicit interest in pursuing a convex formulation. The convex formulation that we derive, guarantees exact dynamic feasibility of the time-optimal trajectories and feedforward torques, with respect to the full nonlinear dynamic model. We carefully construct the initial formulation in a rather specific manner, which turns out to be non-convex. This formulation allows to readily propose a convex relaxation that solves exactly the original non-convex problem. A discretization scheme is developed in order to numerically solve the proposed formulation using methods from numerical optimal control.

3. Traditional approaches seldom point out the detrimental effects on the system performance when implementing pure time-optimal solutions. We emphasize the importance of generating the fastest possible solutions, with crucial additions to the original time-optimal problem that will slightly increase the traveling time. Nevertheless, the final formulation generates near time-optimal solutions that can be implemented in the real manipulator, without seriously degrading the system performance.

4. We demonstrate that the nonzero accelerations at the beginning/end of the trajectory, required by pure time-optimal solutions, seriously degrade the system performance at those instants. We therefore develop a framework to incorporate acceleration constraints which guarantees smooth transitions from/to zero at the beginning/end of the trajectory. Likewise, we show how penalizing a measure of total jerk to trade off time-optimality will render the optimal solutions feasible for implementation, at the expense of a very modest increase in traveling time.

1.2 Optimal Control Synthesis Methodologies for Trajectory Tracking

We explore the use of optimal control synthesis methodologies to attain trajectory tracking for robot manipulators. The motivation is to develop control algorithms that are more suitable to achieve trajectory tracking under the near time-optimal trajectories developed in this dissertation. Initially, the kind of controller utilized to servo the manipulator under such fast trajectories is a general purpose PID plus feedforward controller. The PID portion of the controller is designed in a decentralized manner, where each robot joint is modeled as

a single-input single-output (SISO) linear time-invariant (LTI) system [13, 14]. However, it is well-known that the dynamic model of robot manipulators is highly coupled and nonlinear [1, 2]. Therefore, we are interested in developing control algorithms that use, or at least approximate reasonably well, the referred nonlinear dynamic model. This nonlinear dynamic model corresponds to the model we adopted for the development on time-optimal trajectory planning.

The problem of trajectory tracking that we deal with, considers a robot manipulator with n degrees of freedom (DOF) that has rigid links and revolute joints. The adopted dynamic model is the following highly nonlinear and coupled vector differential equation:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{D}_{\mathrm{v}}\dot{\boldsymbol{q}} + \boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1.5}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the vector of joint angles, $\boldsymbol{\tau} \in \mathbb{R}^n$ represents the vector of input torques, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis/centrifugal matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ represents the gravity torques, and the diagonal elements of $\boldsymbol{D}_{\mathrm{v}} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{F}_{\mathrm{C}} \in \mathbb{R}^{n\times n}$ represent the coefficients of viscous and Coulomb friction, respectively.

In the type of actuators that we utilize, the motors are mechanically connected to the corresponding robot links through gear boxes. We then make the distinction between link-side and motor-side variables. The link-side and motor-side positions are denoted by $\boldsymbol{q}$ and $\boldsymbol{\theta}$, respectively. In addition, the link-side and motor-side actuator torques are denoted by $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively. We assume that these motor-side and link-side variables are simply related by $\boldsymbol{\tau} = \boldsymbol{G}\boldsymbol{u}$ and $\boldsymbol{q} = \boldsymbol{G}^{-1}\boldsymbol{\theta}$, where $\boldsymbol{G}$ is a diagonal matrix with the gear ratios. The variables in dynamic model (1.5) are link-side variables. Since the controller implementation and sensing are done in the motor-side, it is convenient to write the nonlinear dynamic model (1.5) in the motor-side variables. For the purposes of controller synthesis in this dissertation, the following nonlinear dynamic model in motor-side is used:

$$\begin{aligned}\left[\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}\right]\ddot{\boldsymbol{\theta}} + \left[\boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta}, \boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) + \\ \left[\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{\theta}}) = \boldsymbol{u}.\end{aligned} \tag{1.6}$$

Notice that in practice, we always add the corresponding motor-side inertia $\boldsymbol{J}_{\mathrm{mot}}$ to the inertia matrix $\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}$, where $\boldsymbol{J}_{\mathrm{mot}}$ is diagonal and contains each motor's armature inertia. In other words, the actual inertia matrix that we use in (1.6) is: $\boldsymbol{J}_{\mathrm{mot}} + \boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}$. Similarly for the coefficients of viscous and Coulomb damping, the actual ones used for computing the friction torques are: $\boldsymbol{D}_{\mathrm{v,mot}} + \boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}$ and $\boldsymbol{F}_{\mathrm{C,mot}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}$, respectively. However, for simplicity of exposition, in this dissertation we have decided to present the more compact dynamics (1.6).

The problem of trajectory tracking is simply stated as follows: the robot is required to follow a desired joint-space reference trajectory in motor-side $\boldsymbol{\theta}_{\mathrm{d}}(t)$ for all $t \in [0, t_f]$. The problem of trajectory tracking therefore amounts to finding the manipulator joint torques in motor-side $\boldsymbol{u}(t) \in \mathbb{R}^n$ so that $\boldsymbol{\theta}(t) \approx \boldsymbol{\theta}_{\mathrm{d}}(t)$ for all $t \in [0, t_f]$. This problem has been studied for several decades, and therefore many sophisticated algorithms have been proposed

[15, 16]. However, the reality is that only the simplest algorithms are utilized in the real robot manipulators operating in industry. Essentially, all the nonlinear controllers that have been proposed, such as state feedback linearization also known as computed-torque control, require the real-time computation of the inverse dynamic model. These computations are too expensive to achieve in real-time for realistic robots with six or more degrees of freedom, even with the most efficient algorithms for inverse robot dynamics [17].

As a matter of fact, most industrial robots are controlled in practice by pre-computing a feedforward torque $\boldsymbol{u}_{\mathrm{d}}(t)$ from (1.6), which is entirely based on the reference trajectory $\boldsymbol{\theta}_{\mathrm{d}}(t)$, $\dot{\boldsymbol{\theta}}_{\mathrm{d}}(t)$, and $\ddot{\boldsymbol{\theta}}_{\mathrm{d}}(t)$. This feedforward torque is used to attempt to cancel out the nonlinearities. After the assumption of canceling out the nonlinearities through $\boldsymbol{u}_{\mathrm{d}}(t)$, decoupled linear time-invariant models are assumed for each joint [1, 2]. Then, a decoupled linear controller (typically a PID) is designed for the simplified LTI models. Notice that in this manner, there is no need for expensive real-time computations of the inverse dynamic model.

In this dissertation, we follow a similar philosophy. However, we do not assume a simple LTI model to describe the robot dynamics along the entire reference trajectory. Rather, we adopt a particular approach, in which it is strived to find models that approximate reasonably well the nonlinear robot dynamics. From nonlinear systems theory, a nonlinear dynamical system can be well approximated in the vicinity of an equilibrium point by a certain LTI system. This LTI system is obtained by linearizing the nonlinear dynamics at the specific equilibrium point [18, 19]. It is also well-known that linearization of the nonlinear dynamics can only be performed at the system's equilibria. The exception being a dynamically feasible trajectory, along which the linearization process leads to approximate the nonlinear dynamics by a linear time-varying (LTV) dynamical system [20].

The near time-optimal trajectory $(\boldsymbol{\theta}_{\mathrm{d}}(t), \dot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \ddot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$, developed in the first part of this dissertation, is dynamically feasible with respect to the nonlinear robot dynamics (1.6). This motivates to approximate the nonlinear robot dynamics along the near time-optimal reference trajectory as an LTV dynamical system. We can therefore synthesize controllers to achieve trajectory tracking on the LTV system, for which there are numerous efficient techniques from optimal control [21, 22]. Since the robot dynamics is well approximated along the reference trajectory by the LTV system, the resulting control law can be utilized to attain trajectory tracking in the actual robot manipulator. We explore these ideas and develop modifications that allow to implement the resulting controllers in experiments for the 6-axis industrial manipulator.

1.2.1 Contributions

We develop two related approaches to address the problem of trajectory tracking, where the reference trajectory corresponds to the fastest trajectory achievable by the real manipulator, i.e., the near time-optimal trajectory developed in the first part of this dissertation.

1. In the first approach, the continuous-time nonlinear robot dynamics (1.6) is discretized to obtain a discrete-time nonlinear dynamics. We show that due to discretizing the

continuous-time nonlinear model, the reference trajectory and feedforward torques are no longer exactly dynamically feasible with respect to the discrete-time nonlinear model. This leads us to introduce a time-varying "disturbance" term in position and velocity, and therefore approximate the discrete-time nonlinear robot dynamics along the reference trajectory as an affine time-varying (ATV) dynamical system.

2. Using results from optimal control for LTV systems, we synthesize a control law for trajectory tracking that considers the referred ATV dynamics, which implies that the effects of the "disturbance" are implicitly taken into account. The resulting control law is affine time-varying, which achieves trajectory tracking and compensates for the effects of the referred "disturbance", attributed to the non-exact dynamic feasibility of the reference trajectory. We develop the necessary tools and implement the resulting control law in experiments, for the 6-axis industrial manipulator.

3. In the second approach, we develop a gain scheduling strategy along the state-reference trajectory.[1] In this formulation, the continuous-time nonlinear robot dynamics (1.6) is not discretized, which implies that the reference trajectory is still exactly dynamically feasible with respect to the nonlinear model used for controller synthesis. The development is done entirely in continuous-time, although the implementation is clearly carried out in discrete-time. The continuous-time nonlinear dynamic model (1.6) is approximated along the state-reference trajectory by a special class of piecewise affine (PWA) dynamical system.

4. To construct the referred PWA system, we propose an algorithm that selects only strategic points on the state-reference trajectory. These points are chosen to guarantee that the linearized system dynamics for consecutive operating points exhibit a 1% dynamics "variation", with respect to a proposed simple metric. Once the operating points are selected, with their respective (local) linearized system dynamics, a novel approach is proposed to construct ellipsoidal regions around these operating points. The purpose of the ellipsoidal regions is to facilitate the scheduling strategy of feedback controller gains, designed for each local system dynamics, as the robot state traverses in the direction of the state-reference trajectory. We develop the necessary tools, so that the final PWA control law can be implemented and evaluated in experiments on the 6-axis industrial robot.

1.3 Dissertation Outline

The main contributions of this dissertation are presented from Chapter 2 to Chapter 5. In Chapter 2, we give a comprehensive development on the problem of time-optimal trajectory planning along predetermined geometric paths. Our main results in that chapter start with

[1] Related but different ideas on gain scheduling for nonlinear systems moving along a state-reference trajectory have been proposed in [23].

incorporating the complete nonlinear dynamic model of robot manipulators, and to show that it can be solved as a convex optimization problem. Theoretical and simulation results are presented on pure time-optimal trajectories, which leads us to conclude that pure time-optimal solutions need slight modifications before being implemented in experiments.

Then, in Chapter 3, we present the development of the necessary additions that ought to be incorporated into the pure time-optimal formulation. First, acceleration constraints are incorporated, and then, a term that penalizes a measure of total jerk is included to trade off time-optimality. The final formulation that we present in that chapter is also convex, and is used to efficiently compute the near time-optimal trajectories. The experimental results are also presented in that chapter, which verify the benefits of our contributions.

In Chapter 4, we present the development of the control law for trajectory tracking using LQ optimal control methods for LTV systems. Experimental results of the proposed algorithms in that chapter are also presented, which verify the feasibility and also expose certain shortcomings that motivate the development of the approach in Chapter 5. The second approach to achieve better trajectory tracking under the near time-optimal trajectory is presented in Chapter 5. The final control law proposed in that chapter is PWA, which is efficiently implemented in the experimental setup for the 6-axis industrial manipulator. We present the experimental results that confirm the superior features of our approach, which justifies its development.

Finally, concluding remarks and future recommendations are presented in Chapter 6. A brief description of the experimental setup for the 6-axis industrial robot FANUC M-16iB is presented in Appendix A.

Chapter 2

Time-optimal Trajectory Planning along Predetermined Paths

In this chapter, the problem of time-optimal trajectory planning along predetermined geometric paths is addressed. Since the approach in this dissertation relies heavily on the non-linear dynamic model of robot manipulators, a few of its properties will be reviewed, which will prove useful later in the chapter. Unlike existing methods, we incorporate the complete dynamic model, including the term representing viscous friction since for fast motions this term is not negligible at all. Then the problem formulation as an infinite-dimensional mathematical optimization is presented, followed by an analysis of the properties for the referred formulation. This analysis leads to proposing a formulation that we properly call convex relaxation, as it represents a convex and therefore tractable relaxation of the original problem which is non-convex. A discretization scheme is then developed that allows to obtain a numerical solution. Finally, an application to a 6-axis industrial manipulator is presented. From simulations on a realistic simulator, it is argued that pure time-optimal solutions need to be slightly modified before being implemented on the real manipulator.

2.1 Dynamic Model of Robotic Manipulators

The dynamic model of a manipulator provides a description of the relationship between the joint actuator torques and the motion of the manipulator. With Lagrange formulation, the equations of motion can be derived in a systematic manner independent of the reference coordinate frame.

2.1.1 Background on Lagrangian Dynamics

For a manipulator with n links, a set of coordinates q_i, $i = 1, \ldots, n$, known as the generalized coordinates, is chosen to effectively describe the link positions of the n degrees-of-freedom (DOF) manipulator [1]. The Lagrangian of the system is defined in terms of the generalized

coordinates:

$$\mathcal{L} = \mathcal{T} - \mathcal{U}, \tag{2.1}$$

where $\mathcal{U}$ and $\mathcal{T}$ denote respectively the total kinetic and potential energy of the system. Lagrange's equations of motion are usually written in the following form:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = Q_i, \quad i = 1, \ldots, n, \tag{2.2}$$

where Q_i is the generalized force associated with the generalized coordinate q_i. For a serial-chain industrial manipulator, q_i represents the relative angle of i-th link with respect to link $i-1$. The generalized forces Q_i, on the other hand, are given by the nonconservative forces, i.e., the joint actuator torques and the joint friction torques. Assuming viscous and Coulomb friction, $Q_i = \tau_i - d_{vi}\dot{q}_i - f_{ci}\,\mathrm{sign}(\dot{q}_i)$, where d_{vi}, f_{ci} are the coefficients of viscous and Coulomb friction, and τ_i is the actuator torque for the i-th joint.

A well-known fact from robot dynamics is that for an $n-$DOF manipulator, the total kinetic energy is written as the following quadratic form:

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{q}}^\top \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j, \tag{2.3}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the positive-definite inertia matrix, and $m_{ij}(\boldsymbol{q})$ is the ij-th element of $\boldsymbol{M}(\boldsymbol{q})$. It is a standard exercise in dynamics to show that the equations of motion (2.2) take the following form [1, 24]:

$$\sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_k\dot{q}_j + g_i(\boldsymbol{q}) = \tau_i - d_{vi}\dot{q}_i - f_{ci}\,\mathrm{sign}(\dot{q}_i), \quad i = 1, \ldots, n, \tag{2.4}$$

where $c_{ijk}(\boldsymbol{q})$ are known as the Christoffel symbols of the first kind,[1] and $g_i(\boldsymbol{q})$, $i = 1, \ldots, n$, represent the torques due to gravity.[2]

2.1.2 Robot Dynamics in Vector Form

Equations of motion (2.4) will prove useful later in this chapter, when formulating the time-optimal trajectory planning problem. These equations can be written in vector form, which is in fact the standard form given in most robotics textbooks [1, 2, 15]. Throughout this dissertation, the following vector differential equation will be used as the dynamic model for an n-DOF manipulator:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{D}_{\mathrm{v}}\dot{\boldsymbol{q}} + \boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{2.5}$$

[1] Defined in terms of the elements of the inertia matrix as $c_{ijk} := \frac{1}{2}\left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{jk}}{\partial q_i}\right)$.

[2] Defined in terms of the potential energy as $g_i(\boldsymbol{q}) := \frac{\partial \mathcal{U}}{\partial q_i}$.

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are the joint-space positions, velocities, and accelerations, respectively; matrices $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ and $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ are known as the inertia and Coriolis/centrifugal matrices, respectively; $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ represents the vector of gravitational torques; diagonal matrices $\boldsymbol{D}_{\mathrm{v}}, \boldsymbol{F}_{\mathrm{C}} \in \mathbb{R}^{n\times n}$ represent the coefficients of viscous and Coulomb friction, respectively.

2.2 Problem Formulation

As mentioned in Chapter 1, in most industrial applications of robot manipulators, a collision-free geometric path is specified through a teach-pendant or a path planning algorithm. Then, it is important to study how to move the manipulator along that path in the shortest time possible. The approach taken in this dissertation, is to design time-optimal trajectories and controls that are consistent with the complete dynamic model. In this manner, the generated solutions represent the true fastest motions that the manipulator can achieve, unlike methods based on kinematic models [25]. In order to generate trajectories and open-loop controls that are dynamically feasible, the dynamic model should take into account the most significant effects present in the physical system. Therefore, the equations of motion considered in this dissertation are given by the complete model (2.5) or equivalently (2.4).

2.2.1 Mathematical Formulation

Assume that the geometric path, along which the robot is required to move, has been already determined in joint space. In this dissertation, the geometric path is represented by $\boldsymbol{h}(s) \in \mathbb{R}^n$, where s is a monotonically increasing parameter $s(t) \in [0, 1]$, i.e., $\dot{s} > 0$. Parameter s can be thought of as the normalized distance traveled by the manipulator's end-effector. We are going to require that $\boldsymbol{h}(s)$ be twice continuously differentiable in s.

The goal is to determine how to move along the joint-space path $\boldsymbol{h}(s)$, in the shortest time dynamically possible without exceeding the maximum and minimum actuator torques, denoted here as $\overline{\boldsymbol{\tau}} \in \mathbb{R}^n$ and $\underline{\boldsymbol{\tau}} \in \mathbb{R}^n$, respectively. If the reference trajectory is denoted by $\boldsymbol{q}_{\mathrm{d}} \in \mathbb{R}^n$, it is then desired to obtain a 4-tuple $(\boldsymbol{q}_{\mathrm{d}}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}(t), \boldsymbol{\tau}_{\mathrm{d}}(t))$ that is dynamically feasible with respect to model (2.5) and that guaranties motion in the shortest time possible, while satisfying the torque limit constraints: $\underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}_{\mathrm{d}}(t) \leq \overline{\boldsymbol{\tau}}$.

From the equality constraint $\boldsymbol{q}_{\mathrm{d}} = \boldsymbol{h}(s)$, it is readily shown that [26]:

$$\begin{aligned} \dot{\boldsymbol{q}}_{\mathrm{d}} &= \boldsymbol{h}'(s)\dot{s} \\ \ddot{\boldsymbol{q}}_{\mathrm{d}} &= \boldsymbol{h}''(s)\dot{s}^2 + \boldsymbol{h}'(s)\ddot{s}, \end{aligned} \tag{2.6}$$

where $\boldsymbol{h}'(s) := d\boldsymbol{h}/ds$, $\boldsymbol{h}''(s) := d^2\boldsymbol{h}/ds^2$, whereas $\dot{s} = ds/dt$ and $\ddot{s} = d^2s/dt^2$ represent the pseudo-speed and pseudo-acceleration along the path, respectively. Since the 4-tuple $(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{\tau}_{\mathrm{d}})$ is required to be dynamically feasible with respect to (2.5), the expressions for $\boldsymbol{q}_{\mathrm{d}}$, $\dot{\boldsymbol{q}}_{\mathrm{d}}$ and $\ddot{\boldsymbol{q}}_{\mathrm{d}}$ are plugged into (2.5) to obtain:

$$\begin{aligned} \boldsymbol{\tau}_{\mathrm{d}} =& \boldsymbol{M}(\boldsymbol{h}(s)) \left[\boldsymbol{h}''(s)\dot{s}^2 + \boldsymbol{h}'(s)\ddot{s}\right] + \boldsymbol{C}\left(\boldsymbol{h}(s), \boldsymbol{h}'(s)\dot{s}\right)\boldsymbol{h}'(s)\dot{s} + \boldsymbol{g}(\boldsymbol{h}(s)) \\ &+ \boldsymbol{D}_{\mathrm{v}}\boldsymbol{h}'(s)\dot{s} + \boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\boldsymbol{h}'(s)\dot{s}), \end{aligned} \tag{2.7}$$

Since (2.4) and (2.5) are equivalent, the i-th element of vector $\boldsymbol{C}\left(\boldsymbol{h}(s), \boldsymbol{h}'(s)\dot{s}\right)\boldsymbol{h}'(s)\dot{s} \in \mathbb{R}^n$, is written as:

$$\begin{aligned}\left[\boldsymbol{C}\left(\boldsymbol{h}(s), \boldsymbol{h}'(s)\dot{s}\right)\boldsymbol{h}'(s)\dot{s}\right]_i &= \sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}(\boldsymbol{h}) h'_k \dot{s} h'_j \dot{s} \\ &= \left(\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}(\boldsymbol{h}) h'_k h'_j\right)\dot{s}^2, \quad i = 1, \ldots, n.\end{aligned}$$

It is then clear that the second term in (2.7) can be written as $\boldsymbol{C}\left(\boldsymbol{h}(s), \boldsymbol{h}'(s)\right)\boldsymbol{h}'(s)\dot{s}^2$. Likewise, since $\dot{s} > 0$ will be enforced, then $\boldsymbol{F}_{\mathrm{C}} \operatorname{sign}(\boldsymbol{h}'(s)\dot{s}) = \boldsymbol{F}_{\mathrm{C}} \operatorname{sign}(\boldsymbol{h}'(s))$.

With all the above provisos in mind, equation (2.7) is written as:

$$\boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{a}_1(s)\ddot{s} + \boldsymbol{a}_2(s)\dot{s}^2 + \boldsymbol{a}_3(s)\dot{s} + \boldsymbol{a}_4(s), \tag{2.8}$$

where $\boldsymbol{a}_i(s) \in \mathbb{R}^n$, $i = 1, \ldots, 4$, are defined as:

$$\begin{aligned}\boldsymbol{a}_1(s) &:= \boldsymbol{M}\left(\boldsymbol{h}(s)\right)\boldsymbol{h}'(s) \\ \boldsymbol{a}_2(s) &:= \boldsymbol{M}\left(\boldsymbol{h}(s)\right)\boldsymbol{h}''(s) + \boldsymbol{C}\left(\boldsymbol{h}(s), \boldsymbol{h}'(s)\right)\boldsymbol{h}'(s) \\ \boldsymbol{a}_3(s) &:= \boldsymbol{D}_{\mathrm{v}}\boldsymbol{h}'(s) \\ \boldsymbol{a}_4(s) &:= \boldsymbol{F}_{\mathrm{C}} \operatorname{sign}\left(\boldsymbol{h}'(s)\right) + \boldsymbol{g}\left(\boldsymbol{h}(s)\right).\end{aligned} \tag{2.9}$$

Note that $\boldsymbol{a}_i(s)$, $i = 1, \ldots, 4$, can be entirely pre-computed since $\boldsymbol{h}(s)$, $\boldsymbol{h}'(s)$, and $\boldsymbol{h}''(s)$ are already known. The unknowns in parametrization (2.8) are $\ddot{s}$, $\dot{s}^2$, $\dot{s}$, and $\boldsymbol{\tau}_{\mathrm{d}}$, which means we can optimize over these variables (pseudo-acceleration, pseudo-speed, and open-loop torques $\boldsymbol{\tau}_{\mathrm{d}}$), so as to minimize the total traversal time along $\boldsymbol{h}(s)$.

2.2.2 Formulation as a Mathematical Optimization Problem

Consider defining $a(s) := \ddot{s}$, $b(s) := \dot{s}^2$, $c(s) := \dot{s}$, and $\boldsymbol{\tau}_{\mathrm{d}}(s)$, which are to be determined as functions of s. Since a one-to-one relationship between s and t will be enforced (i.e., $\dot{s} > 0$), finding the unknowns as functions of s implies that they can be unambiguously recovered as functions of t. In this manner, $\boldsymbol{\tau}_{\mathrm{d}}(s)$ has a simple affine parametrization in $a(s)$, $b(s)$, and $c(s)$, namely:

$$\boldsymbol{\tau}_{\mathrm{d}}(s) = \boldsymbol{a}_1(s)a(s) + \boldsymbol{a}_2(s)b(s) + \boldsymbol{a}_3(s)c(s) + \boldsymbol{a}_4(s) \tag{2.10}$$

Likewise, with the definitions of $a(s)$, $b(s)$, and $c(s)$, two additional constraints must be incorporated: (i) $\dot{b}(s) = b'(s)\dot{s} = 2\dot{s}\ddot{s} \Leftrightarrow b'(s) = 2a(s)$ if $\dot{s} > 0$, (ii) $c(s) = \sqrt{b(s)}$.

On the other hand, the total traversal time, which is to be minimized, is denoted by t_f, and it can be rewritten in a different manner. Using the fact that $\dot{s} > 0$:

$$t_f = \int_0^{t_f} \mathrm{d}t = \int_0^1 \left(\frac{\mathrm{d}s}{\mathrm{d}t}\right)^{-1} \mathrm{d}s = \int_0^1 \frac{1}{c(s)}\,\mathrm{d}s. \tag{2.11}$$

Interpretation of (2.11) follows, i.e., in order to minimize the total traversal time t_f, the pseudo-speed along the path $c(s) = \dot{s}$ must be as large as possible. It is therefore clear at this point that the torque limit constraints, $\underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}_{\mathrm{d}}(s) \leq \overline{\boldsymbol{\tau}}$, are necessary to incorporate in order to have finite solutions that guarantee $t_f > 0$.

It is important to consider the case when the initial and final pseudo-speeds are zero, i.e., $\dot{s}_0 = \dot{s}_f = 0$. In that case, it is clear that the objective functional (2.11) is unbounded above. In order to overcome this limitation, the integral in (2.11) is defined instead in the interval $[0_+, 1_-]$, where 0_+ and 1_- will be formally defined in Section 2.4.

With all the above considerations, we are in a position to formulate the time-optimal trajectory planning problem stated previously in this chapter, as the following mathematical optimization problem:

$$
\begin{aligned}
\underset{a(s),b(s),c(s),\boldsymbol{\tau}_{\mathrm{d}}(s)}{\text{minimize}} \quad & \int_{0_+}^{1_-} \frac{1}{c(s)} \, \mathrm{d}s \\
\text{subject to} \quad & b(0) = \dot{s}_0^2, \; b(1) = \dot{s}_f^2 \\
& c(0) = \dot{s}_0, \; c(1) = \dot{s}_f \\
& \boldsymbol{\tau}_{\mathrm{d}}(s) = \boldsymbol{a}_1(s)a(s) + \boldsymbol{a}_2(s)b(s) + \boldsymbol{a}_3(s)c(s) + \boldsymbol{a}_4(s) \\
& \underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}_{\mathrm{d}}(s) \leq \overline{\boldsymbol{\tau}} \\
& \forall s \in [0, 1] \\
& b'(s) = 2a(s), \; c(s) = \sqrt{b(s)} \\
& b(s), c(s) > 0 \\
& \forall s \in [0_+, 1_-]
\end{aligned}
\tag{2.12}
$$

which essentially aims to obtaining the optimal pseudo-acceleration, pseudo-speed, and optimal torques along the path $\boldsymbol{h}(s)$. Therefore, if solved, problem (2.12) will yield optimal solutions that attain motion in the shortest time dynamically possible.

Problem (2.12) is an infinite dimensional optimization problem [27], but it can also be viewed as an optimal control problem: with control input $a(s)$, linear differential constraint $b'(s) = 2a(s)$, and algebraic state equalities and inequality constraints [21]. We turn our attention to analyzing some of the properties of problem (2.12), which will motivate the forthcoming development. The following properties are true about formulation (2.12):

1. The objective functional is convex in $c(s)$. To argue why, consider the function $f(c) = 1/c$, $c > 0$, which is trivially a convex function. Since the non-negative weighted sum of convex functions is convex [11], then it follows that the objective functional of problem (2.12) is convex.

2. All inequality constraints are affine.

3. The differential equality constraint is linear and the constrained robot dynamics is an affine equality constraint.

4. The only nonlinear equality constraint is $c(s) = \sqrt{b(s)}\ \forall s \in (0, 1)$.

It is then easy to argue that formulation (2.12) is a non-convex optimization problem [27, 8, 11]. The non-convexity of (2.12) is due to the non-linear equality constraint $c(s) = \sqrt{b(s)}$ $\forall s \in [0, 1]$, which shows up because viscous friction has been considered in the design. For fast motions of robot manipulators, we will see that viscous friction becomes rather significant, even more significant than Coulomb friction. Therefore, instead of ignoring this term, a convex relaxation that solves exactly the non-convex formulation (2.12) is proposed.

2.3 Convex Relaxation

In principle, even though formulation (2.12) is non-convex, we could directly discretize it and attempt to obtain a numerical solution using general nonlinear optimization algorithms, such as sequential convex optimization [11, 28]. However, there are several important reasons for obtaining a convex formulation. One such reason is that the optimal solution is a global minimum which is obtained rather fast since only one optimization is needed, as opposed to sequentially solve several optimizations. That being said, we develop a formulation that derives from (2.12) but that turns out to be convex, thereby the name convex relaxation. Essentially, each non-convex constraint is replaced with a looser, but convex constraint. Relaxing a problem to make it convex is a common mathematical optimization technique when dealing with non-convex hard problems. For instance, the two-way partitioning problem, which is considered very difficult to solve [11], is approximated by a convex relaxation that can therefore be solved efficiently. In general, the convex relaxation does not have to solve exactly the original non-convex problem, i.e., it all depends on the problem at hand.

Since $b(s), c(s) > 0$ for all $s \in [0_+, 1_-]$, the following chain of equivalences is true:[3]

$$\begin{aligned}
\sqrt{b(s)} = c(s) &\Leftrightarrow \frac{1}{\sqrt{b(s)}} = \frac{1}{c(s)} \\
&\Leftrightarrow \frac{1}{\sqrt{b(s)}} \leq \frac{1}{c(s)} \text{ and } \frac{1}{c(s)} \leq \frac{1}{\sqrt{b(s)}} \\
&\Leftrightarrow c(s)^2 - b(s) \leq 0 \text{ and } -c(s)^2 + b(s) \leq 0,
\end{aligned}$$

where the inequality constraint $c(s)^2 - b(s) \leq 0$ is convex for all $s \in [0_+, 1_-]$.[4] On the other hand, $-c(s)^2 + b(s) \leq 0$ is a concave inequality constraint.[5]

Having concave inequality constraints in an optimization problem implies that the overall problem is non-convex, even if the other constraints and the objective function are convex.

[3] We have used the simple fact that, satisfying any equality constraint, say $f_1(x) = f_2(x)$ is equivalent to satisfying two inequality constraints, namely: $f_1(x) \leq f_2(x)$ and $f_1(x) \geq f_2(x) \Leftrightarrow f_1(x) = f_2(x)$.

[4] This can be simply shown by considering $f(b, c) = c^2 - b$, whose Hessian matrix $\partial^2 f / \partial (b, c)^2$ is positive semidefinite, whereby we conclude $f(b, c)$ is a convex function of b, c [11].

[5] In an optimization problem, an inequality constraint of the form $g(x) \leq 0$ is convex if the function $g(x)$ is convex. If on the other hand the function $g(x)$ is concave, then the inequality constraint is concave.

The convex relaxation of (2.12) that we propose in this dissertation, consists in dropping the concave constraint. In other words, replace the equality constraint $c(s) = \sqrt{b(s)}, \forall s \in (0,1)$ in (2.12) with the convex inequality constraint $1/\sqrt{b(s)} \leq 1/c(s), \forall s \in [0_+, 1_-]$. The following is therefore a convex optimization problem:

$$
\begin{aligned}
\underset{a(s),b(s),c(s),\boldsymbol{\tau}_{\mathrm{d}}(s)}{\text{minimize}} \quad & \int_{0_+}^{1_-} \frac{1}{c(s)} \, \mathrm{d}s \\
\text{subject to} \quad & b(0) = \dot{s}_0^2, \; b(1) = \dot{s}_f^2 \\
& c(0) = \dot{s}_0, \; c(1) = \dot{s}_f \\
& \boldsymbol{\tau}_{\mathrm{d}}(s) = \boldsymbol{a}_1(s)a(s) + \boldsymbol{a}_2(s)b(s) + \boldsymbol{a}_3(s)c(s) + \boldsymbol{a}_4(s) \\
& \underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}_{\mathrm{d}}(s) \leq \overline{\boldsymbol{\tau}} \\
& \forall s \in [0,1] \\
& b'(s) = 2a(s), \; 1/\sqrt{b(s)} \leq 1/c(s) \\
& b(s), c(s) > 0 \\
& \forall s \in [0_+, 1_-]
\end{aligned}
\tag{2.13}
$$

which is the convex relaxation of problem (2.12).

We argue that an optimal solution $a^\star(s)$, $b^\star(s)$, $c^\star(s)$, $\boldsymbol{\tau}_{\mathrm{d}}^\star(s)$, to the convex relaxation (2.13), will also solve exactly the original non-convex problem (2.12). Because the functional in problem (2.13) is minimized and since $c(s) > 0$, at optimum the inequality $1/\sqrt{b(s)} \leq 1/c(s)$ must be active, i.e., $1/\sqrt{b^\star(s)} = 1/c^\star(s) \; \forall s \in [0_+, 1_-]$. This entails that for all $s \in [0,1]$ the solution $a^\star(s)$, $b^\star(s)$, $c^\star(s)$, $\boldsymbol{\tau}_{\mathrm{d}}^\star(s)$ to the convex relaxation (2.13) solves exactly the non-convex problem (2.12). This is the reason that motivated us to propose the original formulation in (2.12) precisely as it is.

2.3.1 Infinite-dimensional Second Order Cone Program Formulation

Here we obtain a convenient re-formulation of problem (2.13) that will allow to use available software for convex optimization problems. Before doing so, it is worthwhile explaining how come problem (2.13) is infinite dimensional. The reason for the term infinite dimensional is that the parameter s varies continuously in the open interval $(0,1)$, which implies that, say, the constraint $1/\sqrt{b(s)} \leq 1/c(s), \forall s \in (0,1)$ represents an infinite number of constraints. The same can be said about all the other constraints in problem (2.13).

However, if we grid the parameter s, into a finite set of points $s_1, s_2, \ldots, s_N$, then a finite number of constraints $1/\sqrt{b(s_k)} \leq 1/c(s_k)$, $k = 1, \ldots, N$, are obtained, which is essentially the method to discretize optimal control problems [8]. This is what will be done in the discretization section of this dissertation. However, instead of going straight to discretize optimization problem (2.13), a more convenient infinite dimensional formulation, which will make our discretization procedure straightforward and clear, will be developed first.

A second-order cone program (SOCP) is a special class of convex optimization problems of the form:

$$\begin{aligned} &\text{minimize } f^\top x \\ &\text{subject to } \|A_i x + b_i\|_2 \le c_i^\top x + d_i, \quad i = 1, \dots, m \\ &\qquad\qquad Fx = g, \end{aligned} \tag{2.14}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $A_i \in \mathbb{R}^{n_i \times n}$, and $F \in \mathbb{R}^{p \times n}$. The constraint of the form $\|Ax + b\|_2 \le c^\top x + d$, is called a second-order cone constraint.

We would like to express problem (2.13) as an infinite dimensional SOCP. To do so, it is realized first that the convex constraint $1/\sqrt{b(s)} \le 1/c(s), \forall s \in (0,1)$ can be expressed as an infinite dimensional second-order cone constraint.[6] Therefore, the following chain of equivalences is true, $\forall s \in [0_+, 1_-]$:

$$\begin{aligned} \frac{1}{\sqrt{b(s)}} \le \frac{1}{c(s)} &\Leftrightarrow c(s)^2 \le b(s) \cdot 1 \\ &\Leftrightarrow \left\| \begin{bmatrix} 2c(s) \\ b(s) - 1 \end{bmatrix} \right\|_2 \le b(s) + 1, \quad \forall s \in [0_+, 1_-], \end{aligned} \tag{2.15}$$

which is an infinite-dimensional SOC constraint. Likewise, in order to have a linear objective functional, assume there exists a "slack" function $d(s) > 0$ satisfying $d(s) \ge 1/c(s), \forall s \in [0_+, 1_-]$, then note that:

$$\begin{aligned} d(s) \ge \frac{1}{c(s)} &\Leftrightarrow 1 \le c(s)d(s) \\ &\Leftrightarrow \left\| \begin{bmatrix} 2 \\ c(s) - d(s) \end{bmatrix} \right\|_2 \le c(s) + d(s), \quad \forall s \in [0_+, 1_-], \end{aligned} \tag{2.16}$$

which is also second-order cone constraint. Therefore, problem (2.13) is transformed into the following equivalent problem:

$$\begin{aligned} \underset{a(s), b(s), c(s), \boldsymbol{\tau}_{\mathrm{d}}(s), d(s)}{\text{minimize}} \quad & \int_{0_+}^{1_-} d(s)\, \mathrm{d}s \\ \text{subject to} \quad & b(0) = \dot{s}_0^2,\ b(1) = \dot{s}_f^2 \\ & c(0) = \dot{s}_0,\ c(1) = \dot{s}_f \\ & \boldsymbol{\tau}_{\mathrm{d}}(s) = \boldsymbol{a}_1(s)a(s) + \boldsymbol{a}_2(s)b(s) + \boldsymbol{a}_3(s)c(s) + \boldsymbol{a}_4(s) \\ & \underline{\boldsymbol{\tau}} \le \boldsymbol{\tau}_{\mathrm{d}}(s) \le \overline{\boldsymbol{\tau}} \\ & \forall s \in [0, 1] \\ & b'(s) = 2a(s) \\ & \left\| \begin{bmatrix} 2c(s) \\ b(s) - 1 \end{bmatrix} \right\|_2 \le b(s) + 1 \end{aligned}$$

[6] Essentially the following simple fact is used [29]: $w^2 \le xy,\ x > 0,\ y > 0 \Leftrightarrow \left\| \begin{bmatrix} 2w \\ x - y \end{bmatrix} \right\|_2 \le x + y.$

$$\begin{aligned} &\left\| \begin{bmatrix} 2 \\ c(s) - d(s) \end{bmatrix} \right\|_2 \leq c(s) + d(s) \\ &b(s), c(s) > 0 \\ &\forall s \in [0_+, 1_-], \end{aligned} \tag{2.17}$$

which is in fact an infinite-dimensional version of an SOCP. Notice that formulation (2.17) has a linear objective functional, affine equality and inequality constraints, second-order cone constraints, and linear differential constraint. It is therefore expected that when discretized, the discretization will produce an SOCP of the form (2.14), which will then be readily coded using available software for second-order cone programs, such as CVX [30].

2.4 Problem Discretization

From a theoretical point of view, optimization problem (2.17) represents modest but important progress, since it solves the problem of time-optimal trajectory planning with the complete dynamic model (2.5). Likewise, it has many desirable properties that come with a convex optimization problem. However, from a practical perspective, it still represents a challenge to obtain analytic expressions for the optimal functions $a^\star(s)$, $b^\star(s)$, $c^\star(s)$ and $\tau^*(s)$ for $s \in [0, 1]$. We therefore have to resort to numerical methods. One method for numerically solving problem (2.17), requires the use of calculus of variations making use of the Maximum Principle [31, 27].

The method adopted in this dissertation consists in casting (2.17) as a large-scale convex optimization. This approach from numerical optimal control is termed direct collocation, which is a discretization scheme aimed to obtain a numerical solution [8]. As discussed in Chapter 3, it will be important to impose acceleration constraints to guarantee exact zero acceleration at the beginning and end of the trajectory. Unlike the approach in [10], where $a(s)$ was assumed piecewise constant and $b(s)$ piece-wise linear, we follow the approach to solve optimal control problems presented in [32]. This approach, also known as cubic collocation at Lobatto points, allows to enforce that $a(s)$ be piece-wise linear and that $b(s)$ be piece-wise quadratic. This makes the discretization procedure transparent and facilitates the incorporation of the required acceleration constraints in a straightforward manner.

2.4.1 Cubic Collocation at Lobatto Points

First, the independent parameter of (2.17), which in this case is s, must be discretized. This discretization is done by creating a grid of the path parameter s with N points, $s_1 = 0 < s_2 < \cdots < s_N = 1$. Then, the optimization variables are defined in the following manner: $a_1 = a(s_1), \ldots, a_N = a(s_N)$, $b_1 = b(s_1), \ldots, b_N = b(s_N)$, $c_1 = c(s_1), \ldots, c_N = c(s_N)$, $d_1 = d(s_1), \ldots, d_N = d(s_N)$, $\boldsymbol{\tau}^1 = \boldsymbol{\tau}_\mathrm{d}(s_1), \ldots, \boldsymbol{\tau}^N = \boldsymbol{\tau}_\mathrm{d}(s_N)$, which represent the functions $a(s), b(s), c(s), d(s)$, and $\boldsymbol{\tau}_\mathrm{d}(s)$ evaluated at the grid points $s_1 < s_2 < \cdots < s_N$. It is assumed that $b(s)$ and $a(s)$ are piecewise cubic and piecewise linear, respectively.

The pseudo-acceleration $a(s)$ in (2.17) being piecewise linear means:

$$a(s) = a_j + (a_{j+1} - a_j)\left(\frac{s - s_j}{s_{j+1} - s_j}\right), \quad s \in [s_j, s_{j+1}], \tag{2.18}$$

$j = 1, 2, \ldots, N-1$, from which it is noticed that indeed $a(s_j) = a_j$ and $a(s_{j+1}) = a_{j+1}$. The squared pseudo-speed $b(s)$ is chosen as piecewise cubic, i.e.,

$$b(s) = \sum_{k=0}^{3} \beta_{j,k}\left(\frac{s - s_j}{s_{j+1} - s_j}\right)^k, \quad s \in [s_j, s_{j+1}] \tag{2.19}$$

$j = 1, 2, \ldots, N-1$, where the polynomial coefficients $\beta_{j,0}, \beta_{j,1}, \beta_{j,2}$, and $\beta_{j,3}$ need to be determined explicitly. According to the method presented in [32], the above representation for $b(s)$ must satisfy $b(s_j) = b_j$, $b(s_{j+1}) = b_{j+1}$, $b'(s_j) = 2a(s_j)$, and $b'(s_{j+1}) = 2a(s_{j+1})$, which gives 4 equations for the four unknowns $\beta_{j,0}, \beta_{j,1}, \beta_{j,2}$, and $\beta_{j,3}$. After solving, this simple system of equations, it is obtained:

$$\begin{aligned}
\beta_{j,0} &= b_j, \\
\beta_{j,1} &= 2\Delta s_j a_j \\
\beta_{j,2} &= 3(b_{j+1} - b_j) - 2\Delta s_j(a_{j+1} + 2a_j) \\
\beta_{j,3} &= 2\Delta s_j(a_{j+1} + a_j) - 2(b_{j+1} - b_j),
\end{aligned} \tag{2.20}$$

where $\Delta s_j := s_{j+1} - s_j$, $j = 1, 2, \ldots, N-1$.

A requirement on the approximating functions of $a(s)$ and $b(s)$ in (2.18)-(2.19) is that they satisfy $b'(s) = 2a(s)$ at the grid points s_j, $j = 1, \ldots, N$, and at the mid grid points. In this dissertation, we define the mid grid points as $\bar{s}_j := (s_j + s_{j+1})/2$, $j = 1, \ldots, N-1$. The coefficients $\beta_{j,0}, \beta_{j,1}, \beta_{j,2}$, and $\beta_{j,3}$ given above already guarantee fulfillment of the constraints at grid points s_j. Therefore, the only constraints that need to be incorporated result from requiring $b'(\bar{s}_j) = 2a(\bar{s}_j)$, $j = 1, \ldots, N-1$. After standard algebraic simplifications, we show that $b'(\bar{s}_j) = 2a(\bar{s}_j)$ is equivalent to:

$$b_{j+1} - b_j = \Delta s_j(a_{j+1} + a_j), \quad j = 1, \ldots, N-1. \tag{2.21}$$

Interestingly enough, these constraints imply that the coefficient $\beta_{j,3}$ in (2.20) shall be zero, which means $b(s)$ will turn out to actually be piecewise quadratic. This result could have been anticipated since we are discretizing $b'(s) = 2a(s)$. Likewise, since $c(s)^2 = b(s)$, it is then reasonable to assume $c(s)$ is piecewise linear.

At this point, the only differential constraint in (2.17) has been discretized using the above procedure. Before moving on to discretizing the objective functional, we define $0_+ := (1-\alpha)s_1 + \alpha s_2$ and $1_- := (1-\alpha)s_N + \alpha s_{N-1}$, with $\alpha > 0$ being a small adjustable parameter. The objective functional in (2.17) is therefore discretized simply as follows:

$$\begin{aligned}
\int_{0_+}^{1_-} d(s)\,\mathrm{d}s = &\int_{0_+}^{s_2} d(s)\,\mathrm{d}s \\
&+ \sum_{k=2}^{N-2} \int_{s_k}^{s_{k+1}} d(s)\,\mathrm{d}s + \int_{s_{N-1}}^{1_-} d(s)\,\mathrm{d}s
\end{aligned}$$

$$\begin{aligned}
\approx & \frac{1}{2}\left[(1-\alpha)\Delta s_1(d(0_+)+d_2)\right. \\
& + \sum_{k=2}^{N-2}\Delta s_k(d_k+d_{k+1}) \\
& \left. + (1-\alpha)\Delta s_{N-1}(d_{N-1}+d(1_-))\right],
\end{aligned} \tag{2.22}$$

where $d(0_+) = (1-\alpha)d_1+\alpha d_2$ and $d(1_-) = \alpha d_{N-1}+(1-\alpha)d_N$. The rest of the constraints in problem (2.17) are discretized by simply being evaluated at the grid points s_j, $j = 1,\ldots,N$. For those constraints that are not defined at $s_1 = 0$ and $s_N = 1$, they are evaluated at 0_+ and 1_- instead. Therefore, the following convex optimization problem is obtained:

$$\begin{aligned}
\underset{a_k,b_k,c_k,\boldsymbol{\tau}^k,d_k}{\text{minimize}} \quad & \frac{1}{2}\left[(1-\alpha)\Delta s_1(d(0_+)+d_2)\right. \\
& + \sum_{k=2}^{N-2}\Delta s_k(d_k+d_{k+1}) \\
& \left. + (1-\alpha)\Delta s_{N-1}(d_{N-1}+d(1_-))\right] \\
\text{subject to} \quad & b_1 = \dot{s}_0^2,\ b_N = \dot{s}_f^2 \\
& c_1 = \dot{s}_0,\ c_N = \dot{s}_f \\
& \boldsymbol{\tau}^k = \boldsymbol{a}_1(s_k)a_k + \boldsymbol{a}_2(s_k)b_k + \boldsymbol{a}_3(s_k)c_k + \boldsymbol{a}_4(s_k) \\
& \underline{\boldsymbol{\tau}} \le \boldsymbol{\tau}^k \le \overline{\boldsymbol{\tau}} \\
& \left\| \begin{bmatrix} 2c_k \\ b_k - 1 \end{bmatrix} \right\|_2 \le b_k + 1 \\
& \text{for } k = 1,\ldots,N \\
& b_{j+1} - b_j = \Delta s_j(a_{j+1}+a_j) \\
& \text{for } j = 1,\ldots,N-1 \\
& b_l > 0,\ c_l > 0 \\
& \left\| \begin{bmatrix} 2 \\ c_l - d_l \end{bmatrix} \right\|_2 \le c_l + d_l \\
& \text{for } l = 2,\ldots,N-1 \\
& \left\| \begin{bmatrix} 2 \\ c(0_+) - d(0_+) \end{bmatrix} \right\|_2 \le c(0_+) + d(0_+) \\
& \left\| \begin{bmatrix} 2 \\ c(1_-) - d(1_-) \end{bmatrix} \right\|_2 \le c(1_-) + d(1_-),
\end{aligned} \tag{2.23}$$

where $c(0_+) = (1-\alpha)c_1 + \alpha c_2$ and $c(1_-) = \alpha c_{N-1} + (1-\alpha)c_N$. The discretized problem (2.23) is a second order cone program (SOCP), which is a special class of convex optimization problems [10, 11, 29].

When solved, problem (2.23) produces a "batch" solution to the original infinite dimensional problem (2.17). Therefore, by appropriately selecting the grid size Δs_j (small enough), the discrete solution represents an accurate approximation to the infinite-dimensional one. Clearly, there is a trade-off in selecting Δs_j, i.e., the smaller, the better the approximation. However, as Δs_j becomes smaller, the number of optimization variables and constraints in (2.23) increases, which in turn make the computation time for finding a solution larger. To better explain this fact, it is important to emphasize that problem (2.23) is simply a meaningful manner to express the standardly used model in convex optimization [11]:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \;\; & f_0(\boldsymbol{x}) \\ \text{subject to } & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & g_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, p \end{aligned}$$

where the optimization variable $\boldsymbol{x}$ in (2.23) is the following large-scale vector:

$$\boldsymbol{x} := \left[a_1, \ldots, a_N, b_1, \ldots, b_N, c_1, \ldots, c_N, \boldsymbol{\tau}^1, \ldots, \boldsymbol{\tau}^N, d_1, \ldots, d_N\right]^\top \in \mathbb{R}^{(4+n)N}.$$

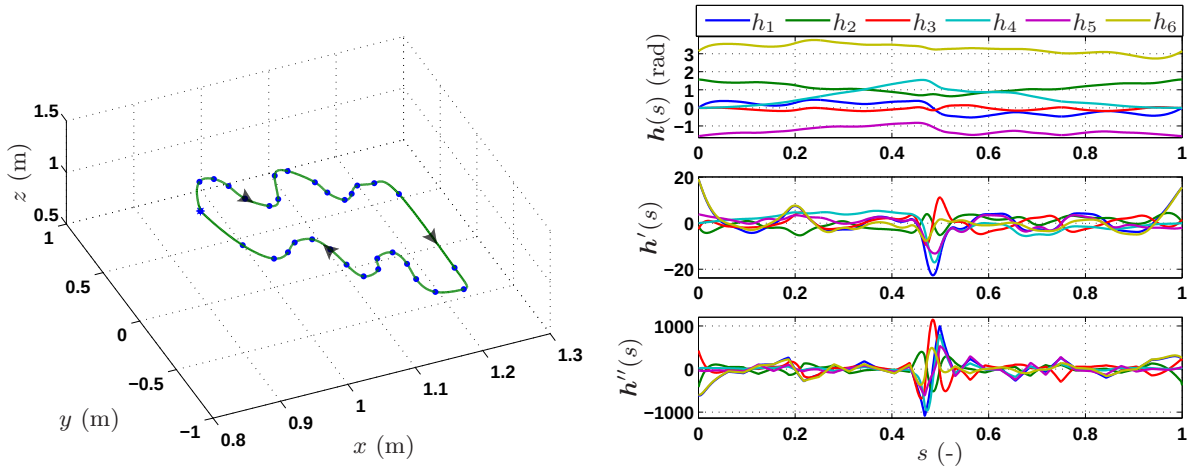
Optimization problem (2.23) is readily coded and solved using CVX, which is a MATLAB® software for discipline convex optimization [30]. Notice that by solving problem (2.23), the optimal functions $a^\star(s), b^\star(s), c^\star(s), d^\star(s)$, and $\boldsymbol{\tau}_{\rm d}^\star(s)$, evaluated at the grid $s_1 = 0 < s_2 < \cdots < s_N = 1$, are obtained. Recall however, that the original problem statement was to obtain the time-optimal 4-tuple $(\boldsymbol{q}_{\rm d}^\star(t), \dot{\boldsymbol{q}}_{\rm d}^\star(t), \ddot{\boldsymbol{q}}_{\rm d}^\star(t), \boldsymbol{\tau}_{\rm d}^\star(t))$. Since a one-to-one correspondence between s and t is enforced in problem (2.23) (i.e., $\dot{s} > 0$), there is no ambiguity in recovering the time-optimal 4-tuple from (2.6). In other words, $\dot{\boldsymbol{q}}_{\rm d}^\star = \boldsymbol{h}'(s)c^\star(s)$, $\ddot{\boldsymbol{q}}_{\rm d}^\star = \boldsymbol{h}''(s)b^\star(s) + \boldsymbol{h}'(s)a^\star(s)$.

To complete our algorithm, we must obtain the time t as a function of s. To do so, consider the following:

$$\begin{aligned} \frac{ds}{dt} = \dot{s} \Leftrightarrow \frac{dt}{ds} &= \frac{1}{c(s)}, \quad c(s) > 0 \\ &\Leftrightarrow t(s) = t(0_+) + \int_{0_+}^{s} \frac{1}{c(u)} \mathrm{d}u. \end{aligned}$$

Then,

$$\begin{aligned} t(s_2) &= t(0_+) + \int_{0_+}^{s_2} \frac{1}{c(u)} \mathrm{d}u \\ t(s_3) &= t(s_2) + \int_{s_2}^{s_3} \frac{1}{c(u)} \mathrm{d}u \\ &\vdots \\ t(s_k) &= t(s_{k-1}) + \int_{s_{k-1}}^{s_k} \frac{1}{c(u)} \mathrm{d}u, \quad k = 2, \ldots, N. \end{aligned}$$



(a) Cartesian-space path with control points

(b) Joint-space path $\boldsymbol{h}(s)$, $\boldsymbol{h}'(s)$ and $\boldsymbol{h}''(s)$

Figure 2.1: Non-trivial path used to test the time-optimal trajectory planning algorithm. A total of 33 control points are interpolated using cubic splines. This interpolation is done for both the Cartesian path as well as the Euler angle parametrization of end-effector orientation.

Likewise, at optimum, $1/c(s) = d(s)$, then:

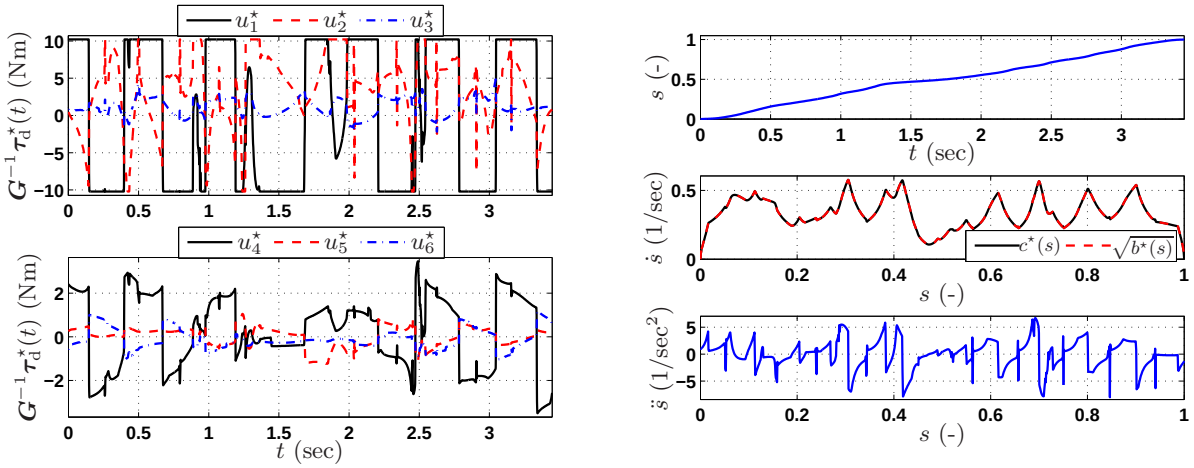
$$\begin{aligned} t(s_k) &= t(s_{k-1}) + \int_{s_{k-1}}^{s_k} d(u)\ \mathrm{d}u \\ &= t(s_{k-1}) + \frac{1}{2}\Delta s_{k-1}(d_{k-1} + d_k). \end{aligned} \tag{2.24}$$

Therefore, time t is recovered with the following steps: (i) initialize $t(s_1) = 0$, (ii) for $k = 2, \ldots, N$, compute $t(s_k)$ using the recursive formula (2.24).

2.5 Application to a 6-axis Industrial Manipulator

The time-optimal 4-tuple $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \boldsymbol{\tau}_{\mathrm{d}}^{\star}(t))$ is generated for a 6-axis industrial manipulator, namely, FANUC M-16*i*B. A non-trivial path with non-constant orientation of the end-effector is designed. The Cartesian-space path is shown in Fig. 2.1(a), where both the initial and final path points correspond to the home configuration, which is marked with an asterisk '*'. This path is designed by choosing 33 control points, marked with '·' in Fig. 2.1(a), which are then interpolated using cubic splines to satisfy the differentiability requirement on $\boldsymbol{h}(s)$ [33, 34]. Likewise, Euler angles are used to parameterize orientation of the end-effector, which are also interpolated using cubic splines from a set of control points that represent intermediate orientations.

A total of $N = 1200$ points are used to discretize the parameter s. In other words, from the 33 control points shown in Fig. 2.1(a), cubic splines are used to interpolate and therefore



(a) Motor-side time-optimal torques, $\boldsymbol{u}_{\rm d}^\star = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\rm d}^\star(t)$ (b) Pseudo-speed and pseudo-acceleration

Figure 2.2: Time-optimal solutions generated when solving problem (2.23). The time-optimal torques are presented in motor-side scale. Notice that the parameter s increases monotonically as t increases, i.e., $\dot{s} > 0\ \forall t \in (0, t_f)$.

obtain the path at 1200 points. Once these 1200 points are obtained in both Cartesian coordinates and Euler angles, the Robotics Toolbox for MATLAB® is used to carry out computations of inverse kinematics [35, 1]. In Fig. 2.1(b), the resulting joint-space path, which includes $\boldsymbol{h}(s)$, $\boldsymbol{h}'(s)$, $\boldsymbol{h}''(s)$, are presented for all six joints of the manipulator. Having $\boldsymbol{h}(s)$, $\boldsymbol{h}'(s)$, $\boldsymbol{h}''(s)$, implies that $\boldsymbol{a}_1(s_k)$, $\boldsymbol{a}_2(s_k)$, $\boldsymbol{a}_3(s_k)$, and $\boldsymbol{a}_4(s_k)$, $k = 1, \ldots, 1200$, defined in (2.9), can be computed from inverse dynamics computations using the Newton-Euler algorithm [35]. The average time to carry out these computations for the 6-axis manipulator is 0.7 seconds.

In general the initial and final pseudo-speeds $\dot{s}_0$ and $\dot{s}_f$ do not have to be zero, however, it is our interest in this dissertation to enforce them to be zero, i.e., $\dot{s}_0 = \dot{s}_f = 0$. The torque constraints are symmetric, i.e., $\underline{\boldsymbol{\tau}} = -\overline{\boldsymbol{\tau}}$, where $\overline{\boldsymbol{\tau}}$ are computed as follows: for the j-th motor, $\overline{\tau}_j = (1/4) r_j k_j i_j^{\max}$, where r_j is the gear ratio, k_j is the torque-current constant, and $i_j^{\max}$ the maximum current. The resulting vector is $\overline{\boldsymbol{\tau}} = (1782.4\ 1789.7\ 1647.2\ 97.2\ 108.5\ 79.1)^\top$ Nm, which represents the maximum torques at the link-side, (i.e., at the reducers output which couple directly to the robot links). The corresponding maximum torques at the motor-side, i.e., at the reducers input, are $\overline{\boldsymbol{u}} = \boldsymbol{G}^{-1}\overline{\boldsymbol{\tau}} = (10.21\ 10.21\ 8.60\ 4.30\ 1.58\ 1.58)^\top$ Nm, where $\boldsymbol{G}$ is the matrix of gear ratios, $\boldsymbol{G} = \mathrm{diag}(r_1, \ldots, r_6)$.

2.5.1 Algorithm Results

The time-optimal solutions, generated when solving problem (2.23), are presented in Fig. 2.2. The total traversal time is $t_f = 3.447$ seconds, which represents the fastest solution with the

assumed parameters and subject to dynamic model (2.5). Notice that all computations in problem (2.23) are carried out in the link-side, i.e., using $\underline{\boldsymbol{\tau}}$ and $\overline{\boldsymbol{\tau}}$. Therefore, $\boldsymbol{\tau}_{\mathrm{d}}^{\star}$ is generated when solving (2.23), however, we present the results in the motor-side, namely, $\boldsymbol{u}_{\mathrm{d}}^{\star} = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\mathrm{d}}^{\star}(t)$. The reason is that we want consistency when presenting simulation and experimental results later in this and subsequent chapters, all of which shall be presented in motor-side.

The motor-side time-optimal torques are shown in Fig. 2.2(a). Notice how these torques feature bang-bang behavior, i.e., there is always one actuator that saturates. For this specific scenario, either $u_1^{\star}$ or $u_2^{\star}$ saturates. In other words, $\forall t \in [0, t_f]$, $u_1^{\star}$ saturates when $u_2^{\star}$ does not, and vice versa. Also notice that even though the other actuators do not saturate, they are required to change suddenly exactly at those instants when $u_1^{\star}$ and $u_2^{\star}$ exchange being saturated. The corresponding $s(t)$, pseudo-speed, and pseudo-acceleration are all presented in Fig. 2.2(b). Both $c^{\star}(s)$ and $\sqrt{b^{\star}(s)}$ are plotted together, so that they can be compared. It is observed that $c^{\star}(s) = \sqrt{b^{\star}(s)}$ $\forall s \in [0, 1]$, as predicted by our earlier argument. This confirms that indeed, the proposed convex relaxation (2.17), or equivalently its discretized version (2.23), solves exactly the original non-convex problem (2.12).

The time-optimal trajectory $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t))$ is presented in Fig. 2.3. From the time-optimal acceleration in Fig. 2.3(c), we realize how time-optimal solutions require very large accelerations at the beginning/end of the trajectory, which is unrealistic since in actuality no actuators can produce such large accelerations instantly. Likewise, sudden acceleration changes are required at intermediate points. These two aspects will motivate our development in Chapter 3, but for now we continue analyzing the results and implications of purely time-optimal solutions.

2.5.2 Dynamic Feasibility

Recall the original problem statement, in which it was desired to obtain a 4-tuple $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \boldsymbol{\tau}_{\mathrm{d}}^{\star}(t))$ that achieves motion in the shortest time possible along a path $\boldsymbol{h}(s)$, and that guaranties dynamic feasibility with respect to the complete dynamic model (2.5). At this point, it is clear that we have developed a methodology that obtains a 4-tuple $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \boldsymbol{\tau}_{\mathrm{d}}^{\star}(t))$. However, this 4-tuple was obtained rather indirectly:

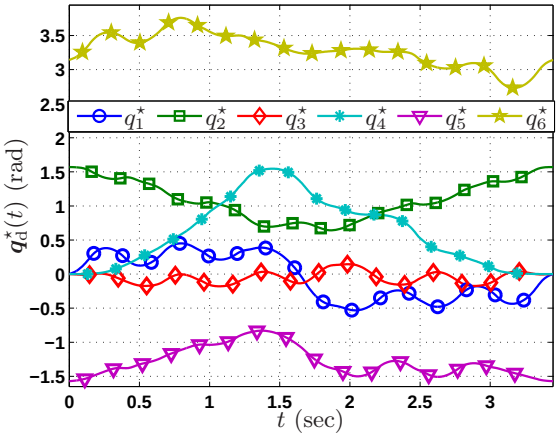
1. The problem was conveniently formulated as an infinite-dimensional optimal control problem (2.12), where the parameter s played the role of the independent variable.

2. An infinite-dimensional convex relaxation was proposed (2.17), that aimed at solving the original non-convex formulation.

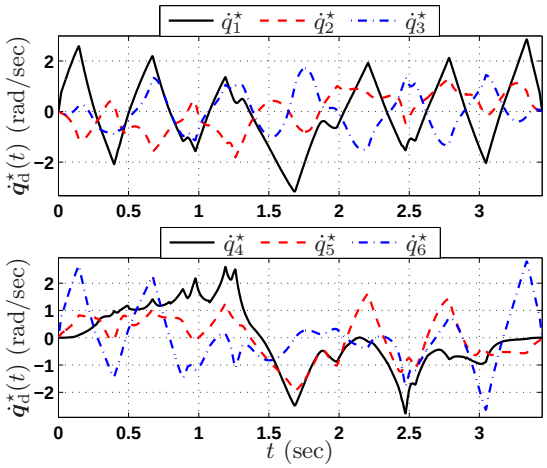
3. A discretization scheme was proposed (2.23), to obtain a numerical solution of (2.17).

4. The trajectory $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t))$ was recovered using (2.6), i.e., $\dot{\boldsymbol{q}}_{\mathrm{d}}^{\star} = \boldsymbol{h}'(s)c^{\star}(s)$, $\ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star} = \boldsymbol{h}''(s)b^{\star}(s) + \boldsymbol{h}'(s)a^{\star}(s)$.

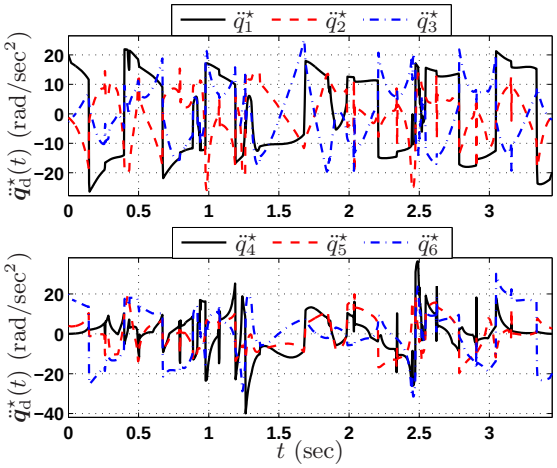
5. Time t was recovered from (2.24).



(a) Time-optimal position reference $\boldsymbol{q}_\mathrm{d}^\star(t)$



(b) Time-optimal velocity reference $\dot{\boldsymbol{q}}_\mathrm{d}^\star(t)$



(c) Time-optimal acceleration reference $\ddot{\boldsymbol{q}}_\mathrm{d}^\star(t)$

Figure 2.3: Time-optimal trajectory $(\boldsymbol{q}_\mathrm{d}^\star(t), \dot{\boldsymbol{q}}_\mathrm{d}^\star(t), \ddot{\boldsymbol{q}}_\mathrm{d}^\star(t))$.

It is then important to verify that the 4-tuple $(\boldsymbol{q}_\mathrm{d}^\star(t), \dot{\boldsymbol{q}}_\mathrm{d}^\star(t), \ddot{\boldsymbol{q}}_\mathrm{d}^\star(t), \boldsymbol{\tau}_\mathrm{d}^\star(t))$ is indeed dynamically feasible with respect to dynamics model (2.5). A substantial amount of effort was spent to ensure that dynamic feasibility is always achieved.

The dynamic feasibility of the 4-tuple $(\boldsymbol{q}_\mathrm{d}^\star(t), \dot{\boldsymbol{q}}_\mathrm{d}^\star(t), \ddot{\boldsymbol{q}}_\mathrm{d}^\star(t), \boldsymbol{\tau}_\mathrm{d}^\star(t))$ is verified by utilizing $(\boldsymbol{q}_\mathrm{d}^\star(t), \dot{\boldsymbol{q}}_\mathrm{d}^\star(t), \ddot{\boldsymbol{q}}_\mathrm{d}^\star(t))$ to compute the torques $\boldsymbol{\tau}(t)^\mathrm{feas}$ using the dynamic model (2.5); the resulting $\boldsymbol{\tau}(t)^\mathrm{feas}$ are then compared against $\boldsymbol{\tau}_\mathrm{d}^\star(t)$. It turns out, in all cases for this and next chapter, $\boldsymbol{\tau}_\mathrm{d}^\star(t) - \boldsymbol{\tau}(t)^\mathrm{feas} = \mathbf{0}\ \forall t \in [0, t_f]$, which means that our algorithm indeed generates optimal 4-tuples that are exactly dynamically feasible with respect to the dynamic model (2.5). Figure 2.4 shows $\boldsymbol{\tau}(t)^\mathrm{feas}$ and $\boldsymbol{\tau}_\mathrm{d}^\star(t)$ plotted together for all six joints of the manipulator.

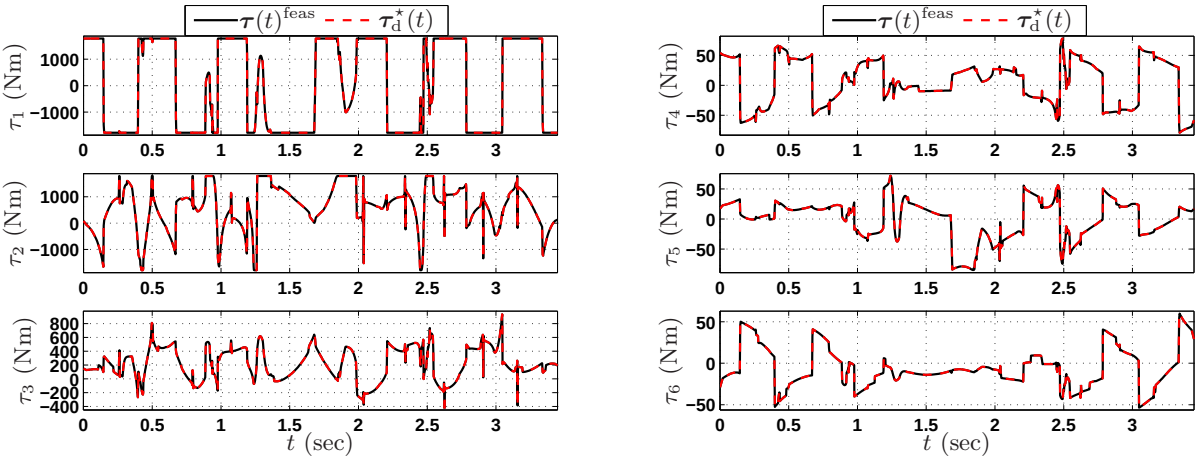


Figure 2.4: Dynamic feasibility verification of the time-optimal 4-tuple $(\boldsymbol{q}_{\text{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\text{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\text{d}}^{\star}(t), \boldsymbol{\tau}_{\text{d}}^{\star}(t))$. Exact dynamic feasibility is achieved for all six joints, i.e., $\boldsymbol{\tau}_{\text{d}}^{\star}(t) - \boldsymbol{\tau}(t)^{\text{feas}} = \mathbf{0}$ $\forall t \in [0, t_f]$

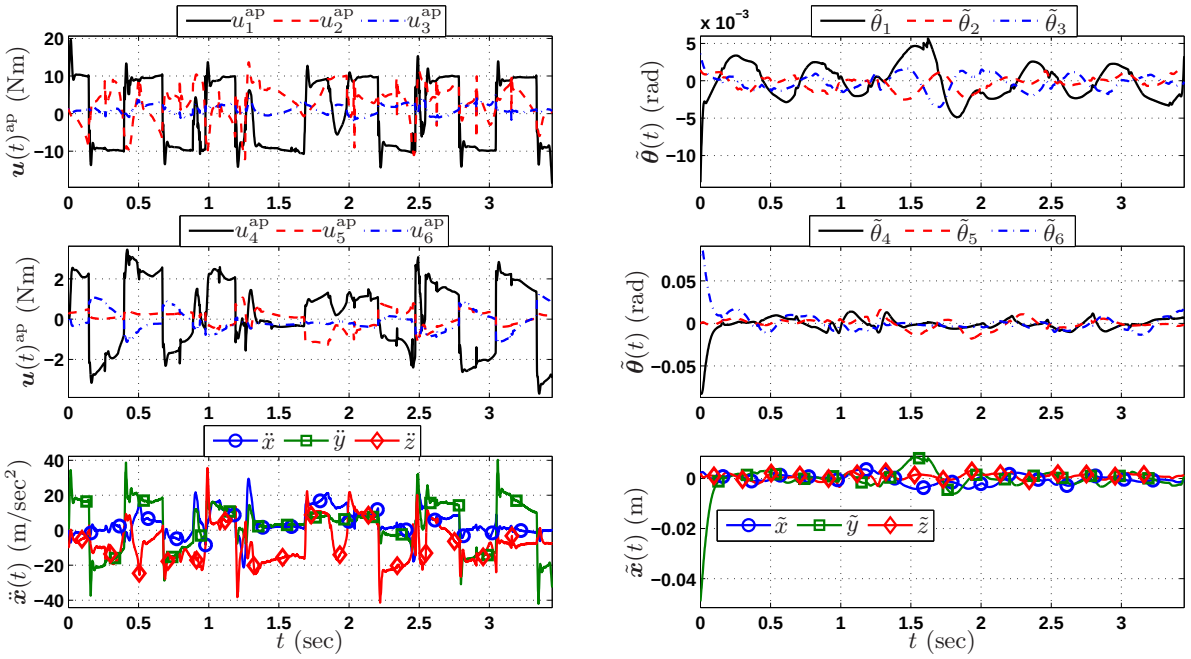
2.6 Simulation of Time-optimal Solution

In order to study the effects of implementing the time-optimal 4-tuple $(\boldsymbol{q}_{\text{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\text{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\text{d}}^{\star}(t), \boldsymbol{\tau}_{\text{d}}^{\star}(t))$, simulations are carried out. A Robot Simulator in MATLAB® that uses Simulink® and SimMechanics™ has been developed in our research group. This Simulator incorporates real robot dynamic effects that are not considered in the dynamic model (2.5), for instance, joint flexibility due to indirect drives. Throughout this dissertation, the link-side and motor-side positions shall be denoted by $\boldsymbol{q}$ and $\boldsymbol{\theta}$, respectively. On the other hand, the link-side and motor-side actuator torques will be denoted by $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively, which we have used already. The control law implemented in the motor-side is given by a feedforward torque plus a feedback:

$$\boldsymbol{u} = \boldsymbol{u}_{\text{d}}^{\star}(t) + \boldsymbol{K}_{\text{P}}\tilde{\boldsymbol{\theta}} + \boldsymbol{K}_{\text{V}}\dot{\tilde{\boldsymbol{\theta}}} + \boldsymbol{K}_{\text{I}}\int_0^t \tilde{\boldsymbol{\theta}}(v)\, \text{d}v, \tag{2.25}$$

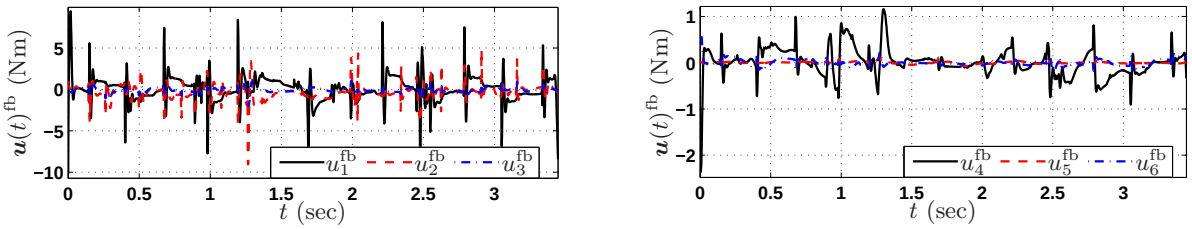
where the feedforward torques $\boldsymbol{u}_{\text{d}}^{\star}(t)$ represent the motor-side optimal torques, i.e., $\boldsymbol{u}_{\text{d}}^{\star}(t) = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\text{d}}^{\star}(t)$. The motor-side tracking error is defined as $\tilde{\boldsymbol{\theta}}(t) := \boldsymbol{\theta}_{\text{d}}(t) - \boldsymbol{\theta}(t)$, where the motor-side reference is simply $\boldsymbol{\theta}_{\text{d}}(t) = \boldsymbol{G}\boldsymbol{q}_{\text{d}}^{\star}(t)$. The feedback gains $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$, and $\boldsymbol{K}_{\text{I}}$ are constant for both simulations and experiments.

The motor-side applied torques, computed from control law (2.25) and denoted by $\boldsymbol{u}(t)^{\text{ap}}$, are presented in Fig. 2.5(a). Comparing to the feedforward torques $\boldsymbol{u}_{\text{d}}^{\star}(t)$ in Fig. 2.2(a), notice that the applied torques $\boldsymbol{u}(t)^{\text{ap}}$ feature large peaks at those instants when $\boldsymbol{u}_{\text{d}}^{\star}(t)$ change suddenly. This implies that the actuators limits, $\underline{\boldsymbol{u}}$, $\overline{\boldsymbol{u}}$, are exceeded, which in turn violates one of the original objectives, namely, $\underline{\boldsymbol{u}} \leq \boldsymbol{u}(t)^{\text{ap}} \leq \overline{\boldsymbol{u}}$. Likewise, the readings of a 3-axis accelerometer mounted at the end-effector are shown in Fig. 2.5(a). A similar effect is induced, i.e., at those instants when $\boldsymbol{u}_{\text{d}}^{\star}(t)$ change suddenly, the end-effector acceleration changes sud-



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors



(c) PID feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

Figure 2.5: Simulation results for the time-optimal trajectory.

denly and large peaks (overshoots) show up. These sudden changes in acceleration will most likely excite oscillation modes due to indirect drive trains for the real system.

Both the joint-space motor-side tracking error $\tilde{\boldsymbol{\theta}}(t)$ and the Cartesian-space tracking error $\tilde{\boldsymbol{x}}(t)$, are presented in Fig. 2.5(b). Due to the non-zero initial and final accelerations required by time-optimal solutions in Fig. 2.3(c), the initial and final tracking errors are comparatively large as evidenced in Fig. 2.5(b). These effects are seldom pointed out in any of the listed references that deal with time-optimal trajectories and controls. Instead, much attention is given to generating solutions that cope with sudden changes at intermediate points, but the effect on the tracking error due to initial/final large accelerations is not recognized. We shall deal with both these issues in Chapter 3.

In Fig. 2.5(c) the feedback PID torques are presented. It is these torques that contribute to such high peaks in the applied torques $\boldsymbol{u}(t)^{\text{ap}}$. Since the trajectory changes in accelerations are too fast to follow closely, large tracking errors are produced. This entails that the PID feedback controller, in trying to keep up with such fast-changing trajectory, produces the peaks exhibited in Fig. 2.5(c).

2.7 Summary

In this chapter we presented the problem of time-optimal trajectory planning of robot manipulators along predetermined geometric paths. In the formulation presented in this chapter, we aimed at improving existing algorithms, by requiring that the time-optimal trajectories and torques be dynamically feasible with respect to the complete nonlinear dynamic model (2.5). We presented the formulation as a mathematical optimization problem, which was deliberately different from existing formulations, so that it would allow to readily propose a convex relaxation that solved exactly the original problem which was non-convex. We also developed our own discretization scheme to actually obtain a numerical solution. It was then shown and verified that our proposed convex relaxation solved exactly the referred problem of time-optimal trajectory planning with full dynamic model. Throughout the chapter, we motivated any development by analyzing simulation results on a realistic robot simulator. The effects on three crucial variables were monitored: the tracking errors, applied torques, and a 3-axis accelerometer mounted at the robot's end-effector. Even though the presented formulation represents progress from a theoretical and numerical standpoint, it was concluded that additional criteria were necessary to incorporate before implementing these optimal solutions on the real robot manipulator.

Chapter 3

Near Time-optimal Trajectory Planning with Acceleration Constraints and Jerk Penalization

The theory presented in Chapter 2 represents important progress from a numerical standpoint, since it guarantees to obtain a numerical solution to the time-optimal trajectory planning problem introduced in that chapter, which considers the complete nonlinear dynamic model (2.5). However, as already discussed in Chapter 2, pure time-optimal solutions produce large nonzero accelerations at the beginning/end of the trajectory, and require sudden acceleration changes at several intermediate points of the trajectory. As suggested from our simulation results, when implemented on the robot manipulator, the referred pure time-optimal solutions will lead to undesired degradation of the system performance.[1] It is therefore necessary to incorporate additional criteria into the formulation before being implemented on the real system. Notice however that time-optimality is still very important in this chapter. In other words, we aim to achieve traveling times that are as close as possible to the traveling times attained by the purely time-optimal solutions presented in Chapter 2. In the present chapter, the time-optimal trajectory planning problem is extended to incorporate necessary criteria that will eliminate the drawbacks of pure time-optimal solutions. First, acceleration constraints will allow to impose exact zero accelerations with smooth transitions at the beginning/end of the trajectory. Then, we introduce a term that penalizes a measure of total jerk, which is thus used to slightly trade off time optimality yielding optimal solutions that are near time-optimal, but with no sudden acceleration changes. Extensive simulations and experimental studies will be presented to justify our development.

[1] Due to limited bandwidth, there are no actuators/servo-amplifiers that can produce such fast changes in accelerations, which clearly impacts the servo performance. Besides, the required sudden changes in acceleration are likely to excite high-frequency vibration modes.

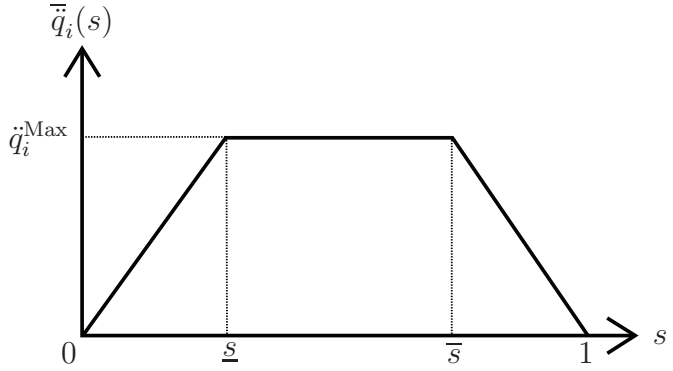


Figure 3.1: Profile of joint-space acceleration constraints.

3.1 Imposing Acceleration Constraints

Several criteria could be incorporated into the problem formulation [7, 10], however, we decide to incorporate only two that will keep the traversal time near time-optimal while eliminating the drawbacks of pure time-optimal solutions. It is evident from the analysis presented in Chapter 2 that non-zero initial/final accelerations lead to large tracking errors at the beginning/end of the trajectory. Therefore, it is necessary to incorporate constraints that guarantee exact zero acceleration at the initial/final point of the trajectory, with smooth growth/decay from/to zero.

3.1.1 Problem Formulation

Acceleration constraints were already pointed out in [10], however never really implemented. In this section we present our own version, which is necessary since it builds on top of our formulation presented in Chapter 2. One of the main advantages of the discretization scheme presented in Chapter 2 shall become evident, namely, it will allow us to impose zero accelerations at exactly the beginning/end of the trajectory, since we did not have to define a mid grid to discretize $a(s)$.

Consider incorporating the following joint-space acceleration constraint:

$$\underline{\ddot{\boldsymbol{q}}}(s) \leq \ddot{\boldsymbol{q}}_{\text{d}}(s) \leq \overline{\ddot{\boldsymbol{q}}}(s), \tag{3.1}$$

where the inequalities are understood componentwise. For the purposes of this dissertation, we propose symmetric bounds, i.e.,

$$-\overline{\ddot{\boldsymbol{q}}}(s) \leq \ddot{\boldsymbol{q}}_{\text{d}}(s) \leq \overline{\ddot{\boldsymbol{q}}}(s). \tag{3.2}$$

For each $\overline{\ddot{q}}_i(s)$, $i = 1, \ldots, n$, we propose the profile shown in Fig. 3.1, where $\ddot{q}_i^{\text{Max}}$ represents the maximum acceleration allowed for the i-th joint at intermediate points of the trajectory. On the other hand, $\underline{s}$ and $\overline{s}$ are adjustable parameters chosen preferably small.

Using the expression for $\ddot{\boldsymbol{q}}_{\mathrm{d}}(s)$ in (2.6), inequality (3.2) is written as:

$$-\overline{\ddot{\boldsymbol{q}}}(s) \leq \boldsymbol{h}''(s)b(s) + \boldsymbol{h}'(s)a(s) \leq \overline{\ddot{\boldsymbol{q}}}(s), \tag{3.3}$$

which is then discretized rather straightforwardly with the discretization scheme presented in Chapter 2. We simply evaluate (3.3) at the grid points, $s_1 = 0 < s_2 < \cdots < s_N = 1$, to obtain:

$$-\overline{\ddot{\boldsymbol{q}}}(s_k) \leq \boldsymbol{h}''(s_k)b_k + \boldsymbol{h}'(s_k)a_k \leq \overline{\ddot{\boldsymbol{q}}}(s_k), \quad k = 1, 2, \ldots, N. \tag{3.4}$$

Since a_k and b_k are defined at exactly the same points in the grid (i.e., s_k), inequalities (3.4) allow to enforce exactly zero acceleration at the initial/final points of the trajectory by simply enforcing symmetrically the profile of Fig. 3.1. This subtle fact is only possible thanks to our proposed discretization scheme presented in Chapter 2. Since the inequalities (3.4) are affine in a_k, b_k, they are incorporated into problem (2.23) without destroying convexity, i.e., still and SOCP:

$$\begin{aligned}
\underset{a_k, b_k, c_k, \boldsymbol{\tau}^k, d_k}{\text{minimize}} \quad & \frac{1}{2}\left[(1-\alpha)\Delta s_1(d(0_+) + d_2)\right. \\
& \quad + \sum_{k=2}^{N-2} \Delta s_k(d_k + d_{k+1}) \\
& \quad \left. + (1-\alpha)\Delta s_{N-1}(d_{N-1} + d(1_-))\right] \\
\text{subject to} \quad & b_1 = \dot{s}_0^2,\ b_N = \dot{s}_f^2 \\
& c_1 = \dot{s}_0,\ c_N = \dot{s}_f \\
& \boldsymbol{\tau}^k = \boldsymbol{a}_1(s_k)a_k + \boldsymbol{a}_2(s_k)b_k + \boldsymbol{a}_3(s_k)c_k + \boldsymbol{a}_4(s_k) \\
& \underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}^k \leq \overline{\boldsymbol{\tau}} \\
& \left\| \begin{bmatrix} 2c_k \\ b_k - 1 \end{bmatrix} \right\|_2 \leq b_k + 1 \\
& -\overline{\ddot{\boldsymbol{q}}}(s_k) \leq \boldsymbol{h}''(s_k)b_k + \boldsymbol{h}'(s_k)a_k \leq \overline{\ddot{\boldsymbol{q}}}(s_k) \\
& \text{for } k = 1, \ldots, N \\
& b_{j+1} - b_j = \Delta s_j(a_{j+1} + a_j) \\
& \text{for } j = 1, \ldots, N-1 \\
& b_l > 0,\ c_l > 0 \\
& \left\| \begin{bmatrix} 2 \\ c_l - d_l \end{bmatrix} \right\|_2 \leq c_l + d_l \\
& \text{for } l = 2, \ldots, N-1 \\
& \left\| \begin{bmatrix} 2 \\ c(0_+) - d(0_+) \end{bmatrix} \right\|_2 \leq c(0_+) + d(0_+) \\
& \left\| \begin{bmatrix} 2 \\ c(1_-) - d(1_-) \end{bmatrix} \right\|_2 \leq c(1_-) + d(1_-),
\end{aligned} \tag{3.5}$$

which is also readily coded in CVX.

3.1.2 Algorithm Results

The optimal 4-tuple $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \boldsymbol{\tau}_{\mathrm{d}}^{\star}(t))$ is generated for the 6-axis industrial manipulator FANUC M-16iB. The same parameters described in Section 2.5 are utilized, but optimization problem (3.5) is used instead to generate the optimal solutions. The parameters for acceleration constraints in Fig. 3.1 are chosen as follows: $\underline{s} = 0.02$, $\overline{s} = 0.98$, and $\ddot{\boldsymbol{q}}^{\mathrm{Max}} = (60\ 60\ 60\ 30\ 30\ 30)^{\top}$ rad/sec^2. Optimization problem (3.5) generates the optimal solutions presented in Fig. 3.2. The optimal traversal time $t_f = 3.987$ seconds, which means that this optimal solution is slower than the pure time-optimal by only 0.54 seconds. Notice from Fig. 3.2(d) that exact zero acceleration is indeed enforced at the beginning/end of the trajectory, with a smooth growth/decay. Note that only $\ddot{q}_4^{\star}$ saturates at some instants. But in general, at the intermediate points, the optimal accelerations in Fig. 3.2(d) are identical to the pure time-optimal accelerations discussed before in Fig. 2.3(c).

The effects on the optimal torques are shown in Fig. 3.2(b). Notice how the optimal torques do not saturate at the beginning/end of the trajectory. Instead, they grow/decay smoothly from/to the required gravity compensation at home position. At the intermediate points, nonetheless, they behave exactly as pure time-optimal, i.e., $u_1^{\star}$ saturates most of the time, and when it does not, $u_2^{\star}$ does. It is likewise realized from Fig. 3.2(a) that $c^{\star}(s) = \sqrt{b^{\star}(s)}, \forall s \in [0, 1]$, which is important to always hold, since it entails full dynamic feasibility with respect to (2.5) is not affected.

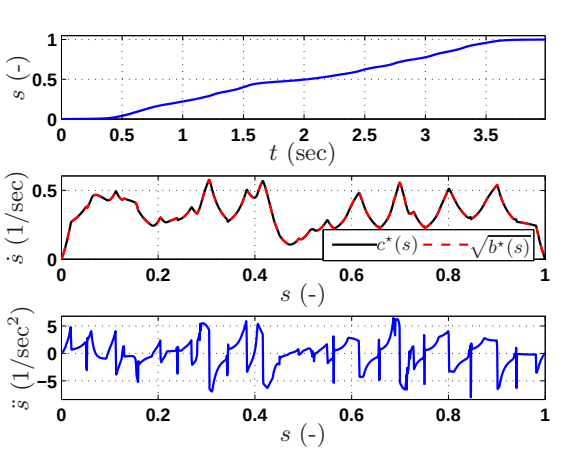
3.1.3 Simulation Results

Recall that the control law is implemented in the motor-side, given by a feedforward torque plus a feedback PID controller:

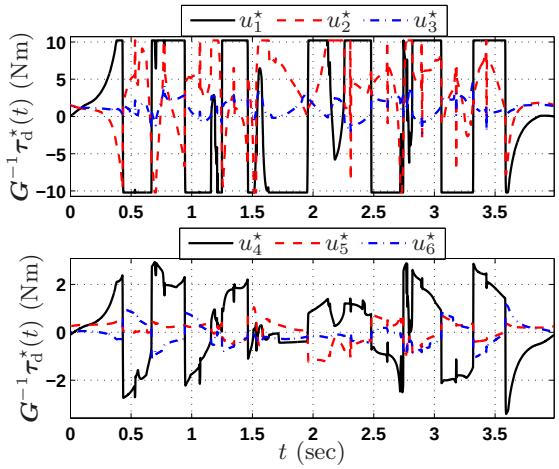
$$\boldsymbol{u} = \boldsymbol{u}_{\mathrm{d}}^{\star}(t) + \boldsymbol{K}_{\mathrm{P}}\tilde{\boldsymbol{\theta}} + \boldsymbol{K}_{\mathrm{V}}\dot{\tilde{\boldsymbol{\theta}}} + \boldsymbol{K}_{\mathrm{I}}\int_0^t \tilde{\boldsymbol{\theta}}(v)\ \mathrm{d}v, \tag{3.6}$$

where the feedforward torques $\boldsymbol{u}_{\mathrm{d}}^{\star}(t)$ represent the near time-optimal torques from Fig. 3.2(b). The simulation results are shown in Fig. 3.3. Observe from Fig. 3.3(a) how the applied torques $\boldsymbol{u}(t)^{\mathrm{ap}}$ closely resemble the near time-optimal torques of Fig. 3.2(b) at the initial/final transitions, thanks to the smooth acceleration growth/decay at the beginning/end of the trajectory as seen from the accelerometer readings. This implies that the PID feedback torques $\boldsymbol{u}(t)^{\mathrm{fb}}$ will not exhibit the large initial/final peaks, which featured in the pure time-optimal case; the PID feedback torques are shown in Fig. 3.3(c). It is also clear that tracking errors shall be benefited, i.e., excessively large initial/final tracking errors are not induced, as shown in Fig. 3.3(b).

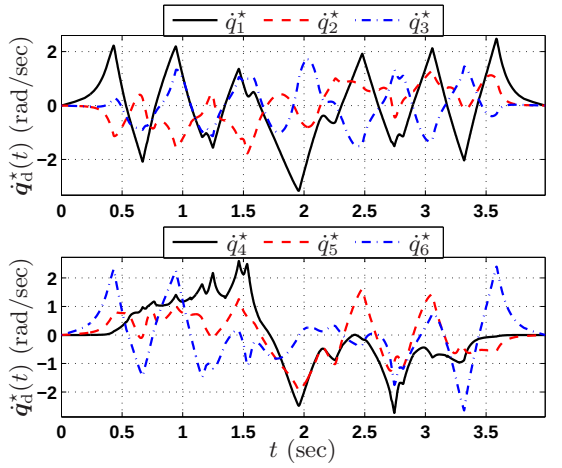
The presented solutions are not yet ready to be implemented on the actual robot manipulator. It is clear from Fig. 3.3 that at the intermediate points, sudden acceleration changes



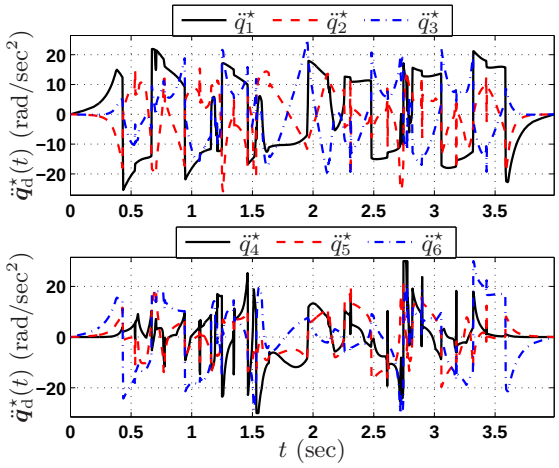
(a) s, $\dot{s}$, and $\ddot{s}$



(b) Motor-side optimal torques, $\boldsymbol{u}_{\mathrm{d}}^{\star} = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\mathrm{d}}^{\star}(t)$



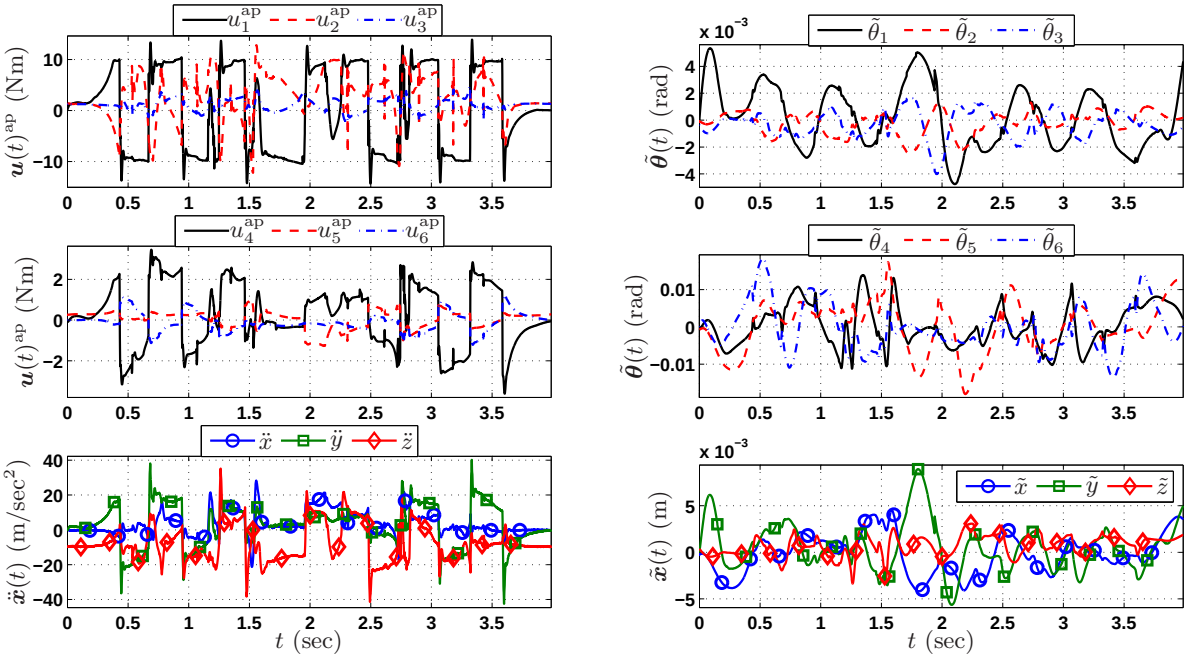
(c) Optimal velocity reference $\dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$



(d) Optimal acceleration reference $\ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$

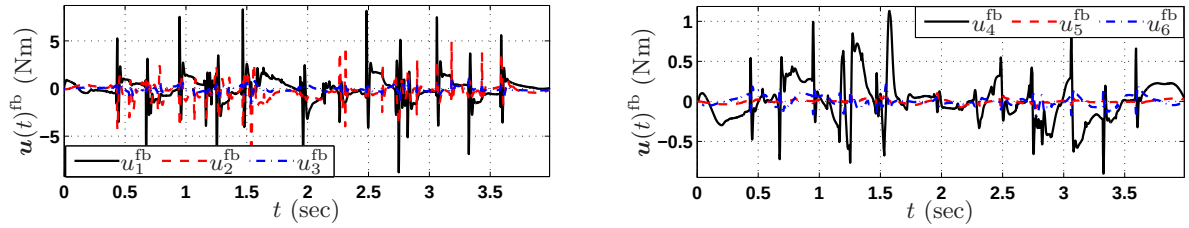
Figure 3.2: Optimal solutions generated when solving problem (2.23), which enforces acceleration constraints with the profile of Fig. 3.1. The optimal torques are presented in motor-side scale. Notice that the only differences, between this near time-optimal solution and the pure time-optimal solution in Chapter 2, are at the beginning/end transitions. At intermediate points of the trajectory, this solution is identical to the pure time optimal. Also notice that $c^{\star}(s) = \sqrt{b^{\star}(s)}, \forall s \in [0, 1]$.

still occur, which have undesirable effects on the system performance. The accelerometer readings suggest that a term that slightly penalizes jerk (i.e., change in acceleration) would help towards completing the algorithm.



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors



(c) PID feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

Figure 3.3: Simulation results for the near time-optimal solution, which enforces zero acceleration constraints at the beginning/end transitions of the trajectory, with smooth growth/decay. The benefits are clear, i.e., the drawbacks at the initial/final transitions of pure time-optimal solutions are eliminated.

3.2 Penalizing a Measure of Total Jerk

It is clear that time optimality alone leads to degradation of the system performance, nonetheless, time-optimal trajectories are important for increase of robot productivity. Therefore, we have discussed that it is still desirable to consider problem (2.23), and incorporate acceleration constraints that guarantee smooth acceleration growth/decay at the the initial and final points of the trajectory. Additionally, in this section, we shall add a term that

penalizes a measure of total jerk to slightly trade off time-optimality, which will prove useful to eliminate the sudden acceleration changes at intermediate points of the trajectory.

3.2.1 Jerk Penalization versus Torque Derivative Penalization

There is usually a question regarding which criterion would give a better trajectory solution, namely: (i) the derivative of acceleration or (ii) the derivative of the torques. For example, in an attempt to study how human-arm motions are actually generated, it is suggested in [36] that trajectories for human planar reaching motions are chosen so that they minimize the integral of the square norm of jerk:

$$J = \frac{1}{2}\int_{t_0}^{t_1} (\dddot{x}^2 + \dddot{y}^2)\, dt, \tag{3.7}$$

where (x, y) is the Cartesian hand position. Following these ideas, it was then suggested in [37] that the motions minimize the integral of the squared norm of the vector of torque derivatives, i.e., the cost function for the planar two-joint arm is of the form:

$$J = \frac{1}{2}\int_{t_0}^{t_1} (\dot{\tau}_1^2 + \dot{\tau}_2^2)\, dt. \tag{3.8}$$

It is mentioned in [25] that it remains an open question which paradigm best describes human-arm motions.

In this dissertation, we are interested in using an equivalent measure of either (3.7) or (3.8) to trade off time-optimality. We choose a measure of total jerk mainly because it has not been reported elsewhere to trade off time optimality. We must also mention that the effects obtained using total jerk will certainly be different from the ones it would be obtained with torque derivatives. That being said, we are interested in $\dddot{\boldsymbol{q}}$, but the forthcoming derivation follows similar lines for $\dot{\boldsymbol{\tau}}$ (see [10]).

3.2.2 Problem Formulation

We use a measure of total jerk that involves the 1-norm $\|\dddot{\boldsymbol{q}}\|_1$, as opposed to the more commonly used 2-norm $\|\dddot{\boldsymbol{q}}\|_2$. Mainly because the resulting objective function will not destroy convexity when incorporated to problem (3.5).[2] Consider therefore trading off the

[2]Even though the 2-norm $\|\dddot{\boldsymbol{q}}\|_2$ is convex in $\dddot{\boldsymbol{q}}$, it is important to realize that $\dddot{\boldsymbol{q}}$ is not the optimization variable, but can be written as a function of the optimization variables as:

$$\dddot{\boldsymbol{q}} = \boldsymbol{h}'''(s)b(s)c(s) + \boldsymbol{h}''(s)b'(s)c(s) + \boldsymbol{h}''(s)a(s)c(s) + \boldsymbol{h}'(s)a'(s)c(s),$$

which is not an affine transformation in $a(s)$, $b(s)$, and $c(s)$. Therefore using $\|\dddot{\boldsymbol{q}}\|_2$ in (3.9) will lead to a non-convex objective function in $a(s)$, $b(s)$, and $c(s)$. It turns out, we are able to overcome this limitation only with the 1-norm $\|\dddot{\boldsymbol{q}}\|_1$, as shown in the forthcoming derivation.

traversal time t_f against the following measure of total jerk:

$$J_{\text{jerk}} = \lambda \int_0^{t_f} \|\dddot{\boldsymbol{q}}\|_1 \, \mathrm{d}t, \tag{3.9}$$

where λ is a weighting parameter that will allow us to produce different results. The jerk is simply the time derivative of acceleration, i.e., for the i-th joint of the manipulator $\dddot{q}_i = \mathrm{d}\ddot{q}_i/\mathrm{d}t$. The 1-norm of the jerk vector $\dddot{\mathbf{q}}$ is simply $\|\dddot{\boldsymbol{q}}\|_1 = \sum_{i=1}^{n} |\dddot{q}_i|$. Therefore J_{jerk} is written as follows:

$$\begin{aligned}
J_{\text{jerk}} = \lambda \int_0^{t_f} \|\dddot{\boldsymbol{q}}\|_1 \, \mathrm{d}t &= \lambda \sum_{i=1}^{n} \int_0^{t_f} |\dddot{q}_i| \; \mathrm{d}t \\
&= \lambda \sum_{i=1}^{n} \int_0^{t_f} \left| \frac{\mathrm{d}\ddot{q}_i}{\mathrm{d}t} \right| \; \mathrm{d}t \\
&= \lambda \sum_{i=1}^{n} \int_0^{1} \left| \frac{\mathrm{d}\ddot{q}_i}{\mathrm{d}s} \right| \; \mathrm{d}s \\
&\approx \lambda \sum_{i=1}^{n} \sum_{j=1}^{N-1} |\ddot{q}_i(s_{j+1}) - \ddot{q}_i(s_j)| \\
&\propto \lambda \sum_{i=1}^{n} \sum_{j=1}^{N-1} \frac{|\ddot{q}_i(s_{j+1}) - \ddot{q}_i(s_j)|}{\ddot{q}_i^{\text{Max}}},
\end{aligned} \tag{3.10}$$

where the last step, i.e., dividing by $\ddot{q}_i^{\text{Max}}$ is necessary in order to nondimensionalize the objective function. Nondimensionalizing the objective function was noticed to really make a difference when trying out a wide range of values for λ, from very small to very large. In all cases no numerical stability problems arise, unlike the case of not dividing by $\ddot{q}_i^{\text{Max}}$.

The objective function (3.10) is nonlinear since the absolute value function $|\cdot|$ is nonlinear (actually piecewise linear). In order to have a linear objective function, consider introducing the following slack variables: e_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, N-1$, such that $\forall i, j$ $|\ddot{q}_i(s_{j+1}) - \ddot{q}_i(s_j)| \leq \ddot{q}_i^{\text{Max}} e_{ij}$. Thus (3.10) can be replaced with the linear objective function and inequality constraints:

$$\begin{aligned}
J_{\text{jerkLin}} &= \lambda \sum_{i=1}^{n} \sum_{j=1}^{N-1} e_{ij} \\
\text{subject to } & |\ddot{q}_i(s_{j+1}) - \ddot{q}_i(s_j)| \leq \ddot{q}_i^{\text{Max}} e_{ij} \\
& i = 1, \ldots, n, \quad j = 1, \ldots, N-1.
\end{aligned} \tag{3.11}$$

Therefore (3.11) is incorporated into problem (3.5) to trade off the traversal time t_f. The constraints $|\ddot{q}_i(s_{j+1}) - \ddot{q}_i(s_j)| \leq \ddot{q}_i^{\text{Max}} e_{ij}$, $i = 1, \ldots, n$, $j = 1, \ldots, N-1$, are expressed compactly

by defining $\boldsymbol{e}_j := (e_{1j}\ e_{2j}\ \cdots\ e_{nj})^\top \in \mathbb{R}^n$, $j = 1, \ldots, N-1$. Therefore, these constraints can be written in vector form:

$$-\boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}} \leq \ddot{\boldsymbol{q}}_{\mathrm{d}}(s_{j+1}) - \ddot{\boldsymbol{q}}_{\mathrm{d}}(s_j) \leq \boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}}, \quad j = 1, \ldots, N-1,$$

where we have defined $\boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}}$ to mean vector element-wise multiplication. By explicitly substituting $\ddot{\boldsymbol{q}}_{\mathrm{d}}(s) = \boldsymbol{h}''(s)b(s) + \boldsymbol{h}'(s)a(s)$, it is obtained:

$$\begin{aligned} -\boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}} \leq\ & \boldsymbol{h}''(s_{j+1})b_{j+1} + \boldsymbol{h}'(s_{j+1})a_{j+1} \\ & - \boldsymbol{h}''(s_j)b_j - \boldsymbol{h}'(s_j)a_j \leq \boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}}, \quad j = 1, \ldots, N-1. \end{aligned} \tag{3.12}$$

It is then clear that the objective function (3.10), which represents a measure of total jerk, can be replaced with the linear objective function (3.11) and the affine inequality constraints (3.12). With all the above provisos in mind, the following final formulation is obtained, which allows to impose acceleration constraints with the profile of Fig. 3.1, and which trades off time optimality against a measure of total jerk through the weighting parameter λ:

$$\begin{aligned} \underset{a_k, b_k, c_k, \boldsymbol{\tau}^k, d_k, \boldsymbol{e}_j}{\text{minimize}} \quad & \frac{1}{2}\left[(1-\alpha)\Delta s_1(d(0_\alpha) + d_2)\right. \\ & \quad + \sum_{k=2}^{N-2} \Delta s_k(d_k + d_{k+1}) \\ & \quad \left. + (1-\alpha)\Delta s_{N-1}(d_{N-1} + d(1_\alpha))\right] \\ & + \lambda \sum_{i=1}^{n}\sum_{j=1}^{N-1} e_{ij} \\ \text{subject to} \quad & b_1 = \dot{s}_0^2,\ b_N = \dot{s}_f^2 \\ & c_1 = \dot{s}_0,\ c_N = \dot{s}_f \\ & \boldsymbol{\tau}^k = \boldsymbol{a}_1(s_k)a_k + \boldsymbol{a}_2(s_k)b_k + \boldsymbol{a}_3(s_k)c_k + \boldsymbol{a}_4(s_k) \\ & \underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau}^k \leq \overline{\boldsymbol{\tau}} \\ & \left\| \begin{bmatrix} 2c_k \\ b_k - 1 \end{bmatrix} \right\|_2 \leq b_k + 1 \\ & -\overline{\ddot{\boldsymbol{q}}}(s_k) \leq \boldsymbol{h}''(s_k)b_k + \boldsymbol{h}'(s_k)a_k \leq \overline{\ddot{\boldsymbol{q}}}(s_k) \\ & \text{for } k = 1, \ldots, N \\ & b_{j+1} - b_j = \Delta s_j(a_{j+1} + a_j) \\ -\boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}} \leq\ & \boldsymbol{h}''(s_{j+1})b_{j+1} + \boldsymbol{h}'(s_{j+1})a_{j+1} \\ & \quad - \boldsymbol{h}''(s_j)b_j - \boldsymbol{h}'(s_j)a_j \leq \boldsymbol{e}_j * \ddot{\boldsymbol{q}}^{\mathrm{Max}} \\ & \text{for } j = 1, \ldots, N-1 \end{aligned} \tag{3.13}$$

$$
\begin{aligned}
& b_l > 0,\ c_l > 0 \\
& \left\| \begin{bmatrix} 2 \\ c_l - d_l \end{bmatrix} \right\|_2 \leq c_l + d_l \\
& \text{for } l = 2, \ldots, N-1 \\
& \left\| \begin{bmatrix} 2 \\ c(0_\alpha) - d(0_\alpha) \end{bmatrix} \right\|_2 \leq c(0_\alpha) + d(0_\alpha) \\
& \left\| \begin{bmatrix} 2 \\ c(1_\alpha) - d(1_\alpha) \end{bmatrix} \right\|_2 \leq c(1_\alpha) + d(1_\alpha)
\end{aligned}
$$

Formulation (3.13) still represents an SOCP, which is therefore readily coded in CVX. It is the final version for near time-optimal trajectory presented in this dissertation. In other words, no additional criteria shall be added to optimization problem (3.13).

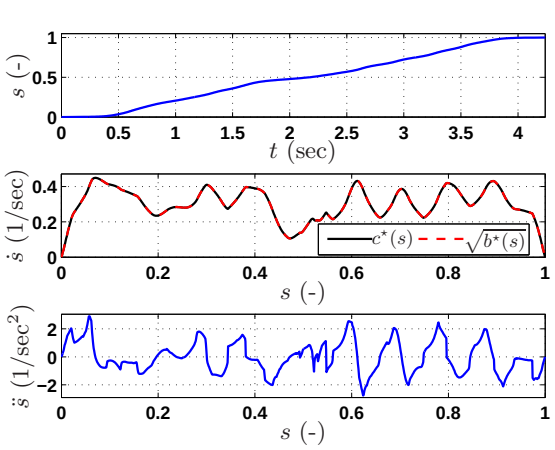
Although we could certainly incorporate more criteria as long as they do not destroy convexity, it would however contribute to increasing the total traversal time t_f. We really are interested in pushing the limits of near time-optimality as much as possible, which means we want to attain the fastest near time-optimal motions. We have pointed out that two factors contributed to degradation of system performance when testing pure time-optimal solutions: (i) the large nonzero initial/final accelerations and (ii) the sudden acceleration changes at intermediate points of the trajectory. As we shall discover soon, these two issues are resolved with formulation (3.13). We will also see that formulation (3.13) can be used to generate medium and low speed optimal trajectories, by appropriately choosing λ.

3.2.3 Algorithm Results

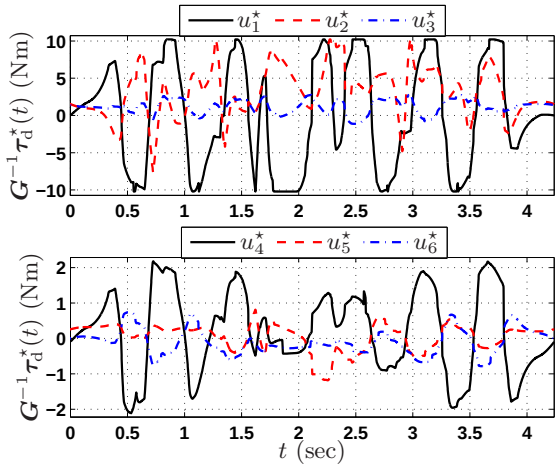
We generate near time-optimal 4-tuples $(\boldsymbol{q}_{\mathrm{d}}^{\star}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t), \boldsymbol{\tau}_{\mathrm{d}}^{\star}(t))$ for the 6-axis manipulator FANUC M-16iB, using formulation (3.13). The same baseline parameters are used as previously, i.e., the number of grid points $N = 1200$, $\overline{\boldsymbol{\tau}} = (1782.4\ 1789.7\ 1647.2\ 97.2\ 108.5\ 79.1)^{\top}$ Nm, which implies $\overline{\boldsymbol{u}} = \boldsymbol{G}^{-1}\overline{\boldsymbol{\tau}} = (10.21\ 10.21\ 8.60\ 4.30\ 1.58\ 1.58)^{\top}$ Nm. The acceleration constraint parameters $\underline{s} = 0.02$, $\overline{s} = 0.98$, and $\ddot{\boldsymbol{q}}^{\mathrm{Max}} = (60\ 60\ 60\ 30\ 30\ 30)^{\top}$ rad/sec^2.

We present the generated results for $\lambda = 0.02$ in Fig. 3.4. The first feature to realize is that even though an additional term was added to the objective function, it is still the case that $c^{\star}(s) = \sqrt{b^{\star}(s)}, \forall s \in [0, 1]$, which means that optimization problem (3.13) inherits the property from (3.5) of generating solutions that are indeed dynamically feasible with respect to the full dynamic model (2.5). Notice also how the pseudo-acceleration $\ddot{s}$ has been rendered smaller and smoother, as compared to the one previously presented in Fig. 3.2(a). This shall have a direct effect in $\ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}$ as shown in Fig. 3.4(d), of effectively eliminating the sudden acceleration changes.

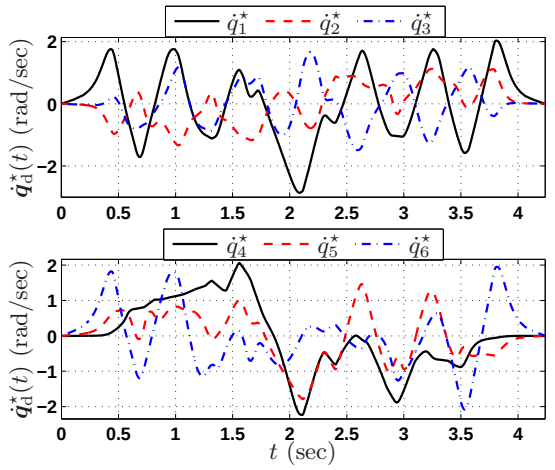
Regarding the optimal torques in Fig. 3.4(b), notice that at some time instants, $\tau_1^{\star}$ is still required to saturate since the solutions are near time-optimal. Nevertheless, the transitions between saturation levels are required to occur in a finite amount of time. These benefits come at the cost of a modest increase in the traversal time, i.e., $t_f = 4.238$ seconds, which



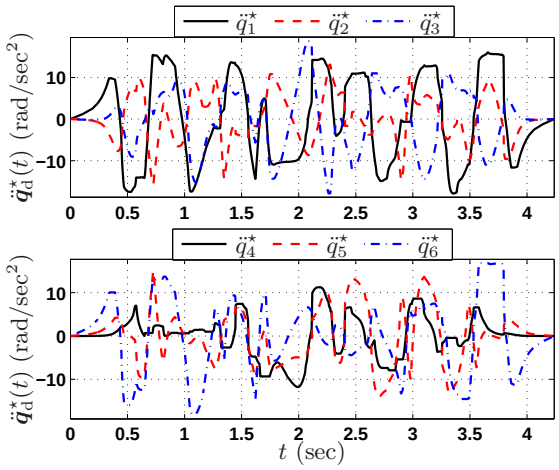
(a) s, $\dot{s}$, and $\ddot{s}$



(b) Motor-side optimal torques, $\boldsymbol{u}_{\mathrm{d}}^{\star} = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\mathrm{d}}^{\star}(t)$



(c) Optimal velocity reference $\dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$



(d) Optimal acceleration reference $\ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$

Figure 3.4: Near-time optimal solutions generated when solving problem (3.13) for $\lambda = 0.02$, which in addition to enforcing acceleration constraints with the profile of Fig. 3.1, it penalizes a measure of total jerk (change in acceleration) through the weighting parameter λ. Compared to the optimal solutions generated by (3.5), notice that the generated optimal accelerations/torques feature no sudden changes. Mercifully, it is again the case that $c^{\star}(s) = \sqrt{b^{\star}(s)}, \forall s \in [0, 1]$.

means that this solution is slower than the purely time-optimal by only 0.791 seconds. Nonetheless, the benefits in terms of non-degradation of the performance become a crucial factor to justify our development.

A nice property to point out is that when $\lambda = 0$, optimization problem (3.13) generates exactly the same optimal solutions as (3.5), even though they are actually two different

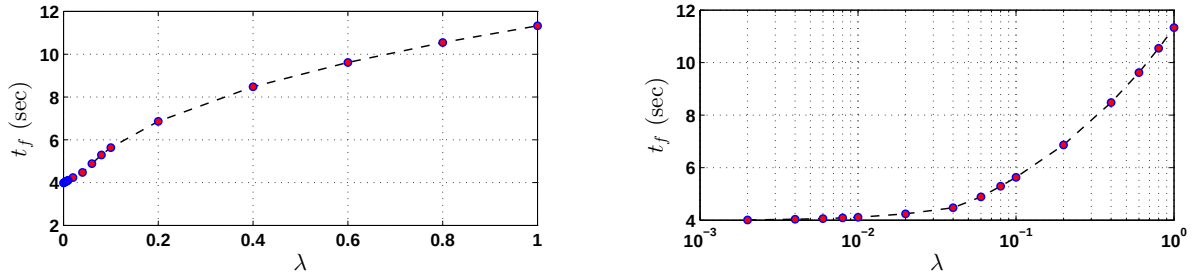


Figure 3.5: Optimal traversal time t_f for a range of values of the weighting parameter λ. Normal scale (on the left) and semi-logarithmic scale (on the right).

problems. In other words, both are large-scale optimization problems but with different number of optimization variables and constraints. Recall that problem (3.5) is a large-scale optimization problem with optimization vector:

$$\left[a_1, \ldots, a_N, b_1, \ldots, b_N, c_1, \ldots, c_N, \boldsymbol{\tau}^1, \ldots, \boldsymbol{\tau}^N, d_1, \ldots, d_N\right]^\top \in \mathbb{R}^{(4+n)N},$$

which gives a total number of optimization variables $(4+n)N = 12,000$. On the other hand, problem (3.13) is a large-scale optimization problem with optimization vector:

$$\left[a_1, \ldots, a_N, b_1, \ldots, b_N, c_1, \ldots, c_N, \boldsymbol{\tau}^1, \ldots, \boldsymbol{\tau}^N, d_1, \ldots, d_N, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{N-1}\right]^\top \in \mathbb{R}^{(4+2n)N-n},$$

giving a total number of optimization variables $(4+2n)N - n = 19,194$. Note that the number of constraints is also different for the two optimization problems.

Before continuing onto presenting the simulation and experimental results for $\lambda = 0.02$, it is interesting to plot the traversal time t_f as λ is increased from zero. The referred plot is presented in Fig. 3.5. For a more quantitative presentation of the relationship between λ and t_f, Table 3.1 displays some of the values shown in Fig. 3.5.

λ	0	.004	.008	.01	.02	.06	.1	.4	.8	1
t_f (sec)	3.987	4.034	4.085	4.110	4.238	4.887	5.633	8.478	10.545	11.331

Table 3.1: Some numerical values of λ and resulting optimal traversal times t_f.

For very small values of λ, i.e., $\lambda \leq 0.008$, the traversal time t_f increases but not so significantly. For larger values of λ, the increase in t_f becomes more significant. It is therefore meaningful to present this relation in a semi-logarithmic fashion as shown in the right sub-plot of Fig. 3.5.

3.3 Simulation and Experimental Results

We implement again control law (3.6) with the same parameters as before, but for the optimal solutions generated with problem (3.13) using $\lambda = 0.02$. The simulation results are shown

in Fig. 3.6 from which it is seen that the applied torques $\boldsymbol{u}(t)^{\text{ap}}$ are rather consistent with the feedforward optimal torques $\boldsymbol{u}_{\text{d}}^{\star}(t)$ in Fig. 3.4(b). Therefore, the applied torques $\boldsymbol{u}(t)^{\text{ap}}$ do not exceed the torque limits, which was one of the original requirements for optimal trajectory planning presented in Chapter 2. Even though this requirement had been fulfilled in open-loop, e.g., Figs. 2.2(a) and 3.2(b), when closing the loop the corresponding $\boldsymbol{u}(t)^{\text{ap}}$ always exceeded the torque limits due to large peaks. In contrast, problem (3.13) generates optimal solutions that are near time-optimal (i.e., near fastest) and which result in applied torques $\boldsymbol{u}(t)^{\text{ap}}$ that do not exceed the torque limits.

The corresponding motor-side and Cartesian-space tracking errors are presented in Fig. 3.6(b), all of which are slightly better than the ones presented before in Fig. 3.3(b), and significantly better than the ones in Fig. 2.5(b). Even though the tracking errors in Fig. 3.6(b) are just slightly better than in Fig. 3.3(b), the real advantages of the optimal solutions in Fig. 3.4 come from the accelerometer readings in Fig. 3.6(a) and from the PID feedback torques in Fig. 3.6(c). In other words, when carrying out simulations with control law (3.6), the accelerometer readings exhibit high accelerations with no overshooting. Likewise, the PID feedback torques $\boldsymbol{u}(t)^{\text{fb}}$ in Fig. 3.6(c) should be compared against Fig. 3.3(c), it is clear that the feedback controller is not required to generate large torque peaks in order to track this fast trajectory. It is therefore at this point that we turn from simulations to carry out experiments on the actual robot manipulator.

3.3.1 Experimental Results

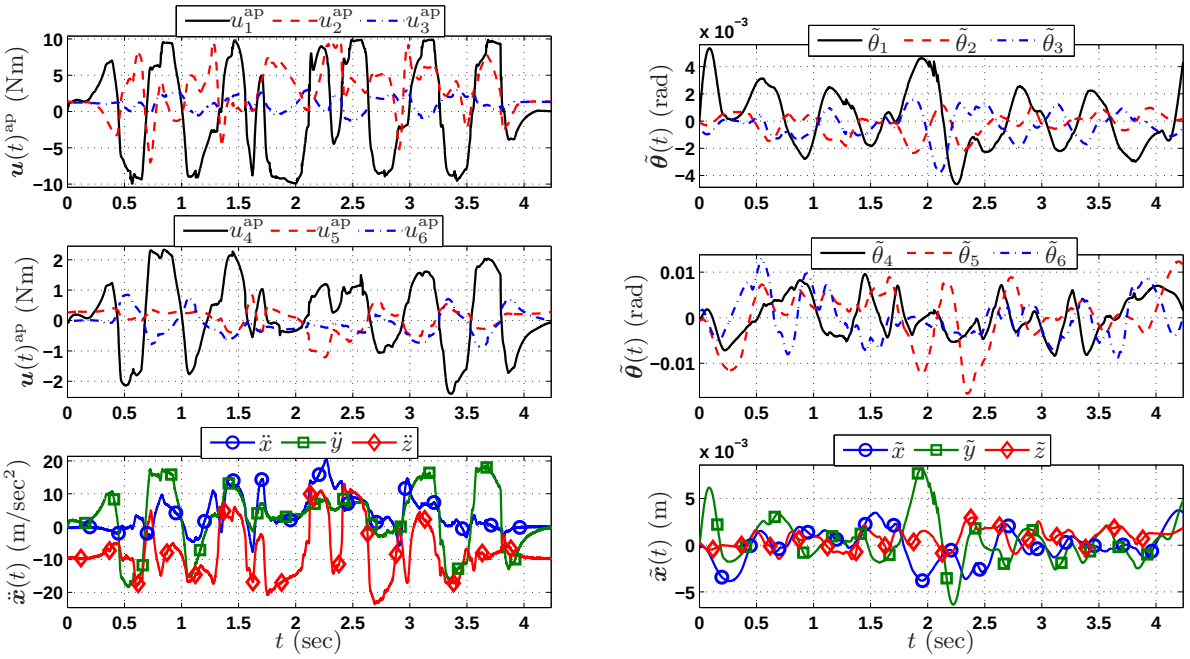
Motivated by the above observations, we proceed to carry out experiments on the actual 6-axis industrial manipulator FANUC M16*i*B. The optimal solution presented in Fig. 3.4 is implemented on the real system which runs at a 1-millisecond sampling period. The control law is repeated here for convenience:

$$\boldsymbol{u} = \boldsymbol{u}_{\text{d}}^{\star}(t) + \boldsymbol{K}_{\text{P}}\tilde{\boldsymbol{\theta}} + \boldsymbol{K}_{\text{V}}\dot{\tilde{\boldsymbol{\theta}}} + \boldsymbol{K}_{\text{I}}\int_0^t \tilde{\boldsymbol{\theta}}(v)\ \text{d}v, \tag{3.14}$$

where the feedback gains $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$, and $\boldsymbol{K}_{\text{I}}$ are exactly the same as for simulations. Details on the experimental setup are presented in Appendix A.

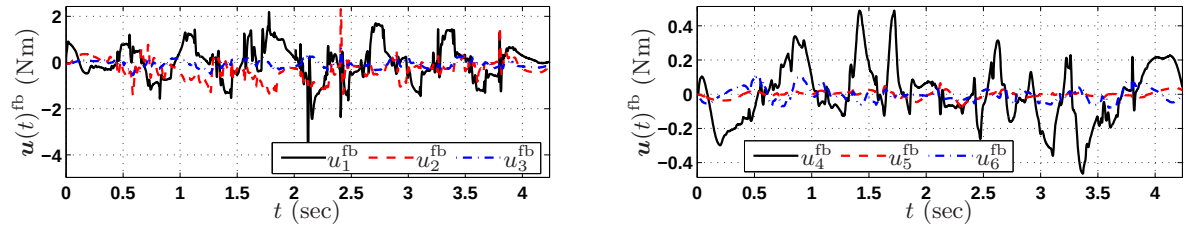
The experimental results are presented in Fig. 3.7 whereby several similarities and differences, with the simulation results, should be pointed out. The experimental applied torques $\boldsymbol{u}(t)^{\text{ap}}$ from Fig. 3.7(a) are rather consistent with the applied torques from simulation in Fig. 3.6(a). Even though the optimal torques and trajectories represent the fastest motions achievable by the actual manipulator, the experimental applied torques do not exceed the maximum and minimum torque limits.

The joint-space motor-side tracking errors $\tilde{\boldsymbol{\theta}}(t)$ are presented in Fig. 3.7(b). It is observed that for the first three joints, the experimental tracking errors $\tilde{\theta}_1$, $\tilde{\theta}_2$, and $\tilde{\theta}_3$, are rather consistent with the corresponding tracking errors obtained from simulations in Fig. 3.6(b). The Cartesian-space tracking errors $\tilde{\boldsymbol{x}}(t)$ in Fig. 3.7(b), measured with CompuGauge 3D measurement system, are also rather consistent with the tracking errors obtained in simulation.



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors

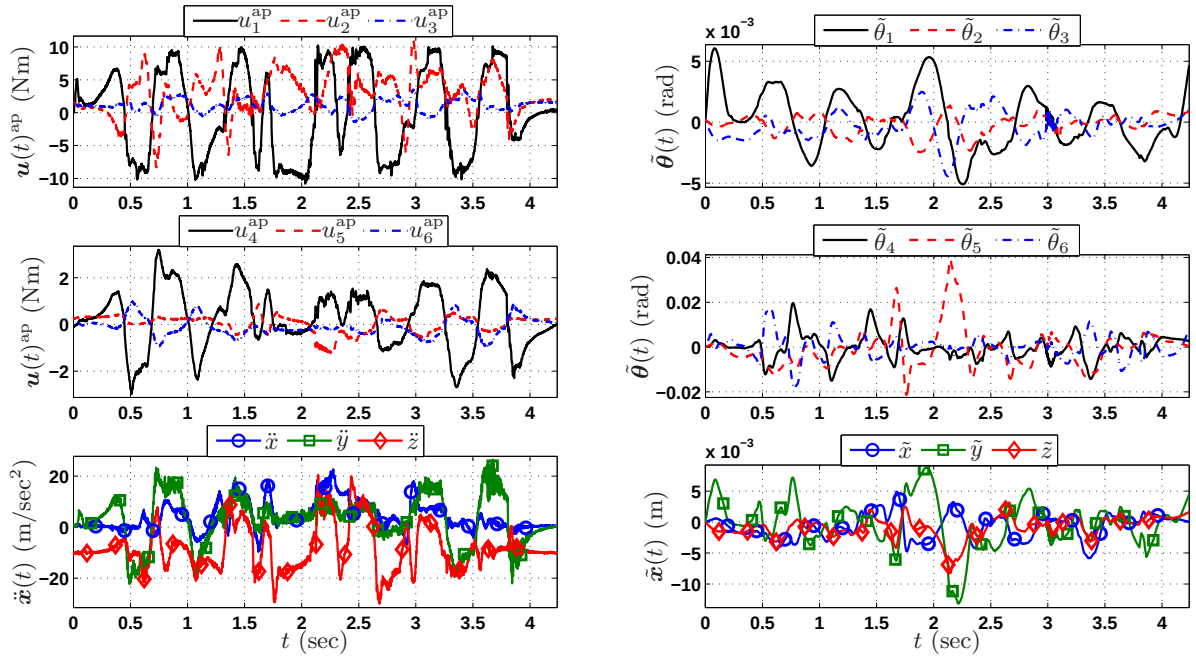


(c) PID feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

Figure 3.6: Simulation results for the near time-optimal solutions with acceleration constraints and penalization of a measure of total jerk. The optimal solutions are generated using problem (3.13) with $\lambda = 0.02$. Clearly, the drawbacks of pure time-optimality are eliminated, at the cost of a modest increase in traversal time.

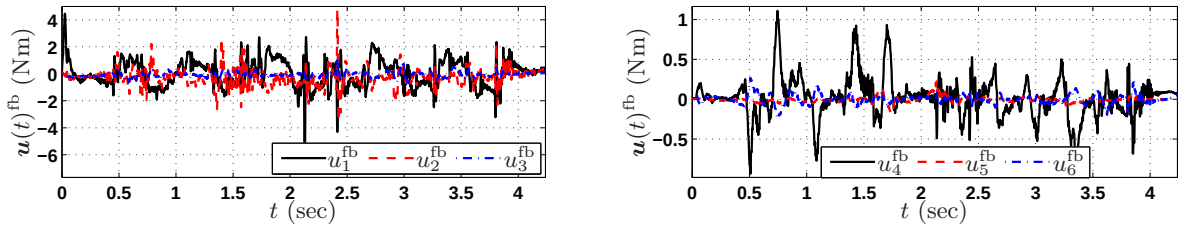
For the last three joints though (especially the 5th joint), the joint-space motor-side tracking errors $\tilde{\theta}_4$, $\tilde{\theta}_5$, and $\tilde{\theta}_6$, are larger for experiments than they are for simulations. This is likely due to modeling errors and parameter uncertainty.

Finally, the experimental PID feedback torques $\boldsymbol{u}(t)^{\text{fb}}$ are presented in Fig. 3.7(c). Since the role of the feedback part of the control law is to compensate for uncertainty, clearly, these torques shall be different in experiments and simulations, as uncertainty and unexpected disturbances are different for real experiments.



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors

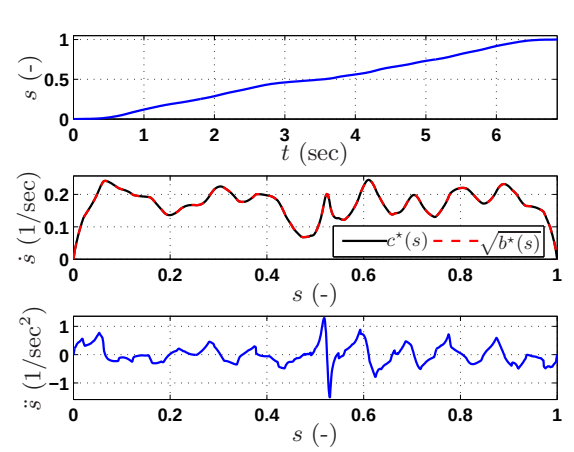


(c) PID feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

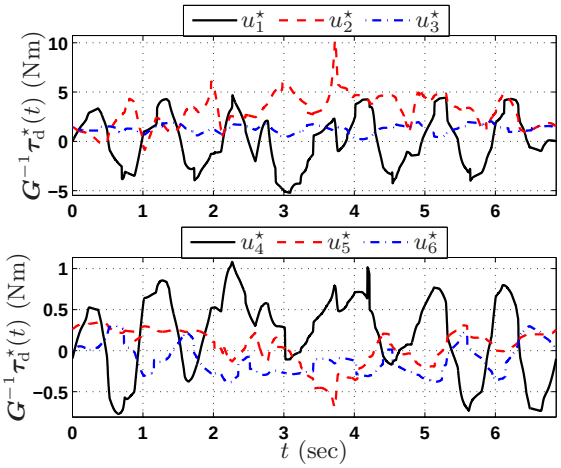
Figure 3.7: Experimental results for the near time-optimal solutions with acceleration constraints and penalization of a measure of total jerk, using $\lambda = 0.02$.

The magnitude of the feedback torque due to parametric uncertainty depends on how much parametric uncertainty and to some extent on how fast the trajectory is. For instance, if we solve again the optimization problem (3.13) but this time with $\lambda = 0.2$, the generated optimal solution is presented in Fig. 3.8, with an optimal traversal time $t_f = 6.863$ seconds. The experimental results for this medium-speed optimal solution are presented in Fig. 3.9. Observe how the applied torques $\boldsymbol{u}(t)^{\text{ap}}$ are again rather consistent with the optimal feedforward torques $\boldsymbol{u}_{\text{d}}^{\star}(t)$. Regarding the experimental PID feedback torques $\boldsymbol{u}(t)^{\text{fb}}$ in Fig. 3.9(c), we notice that the magnitude is substantially reduced as compared to Fig. 3.7(c).

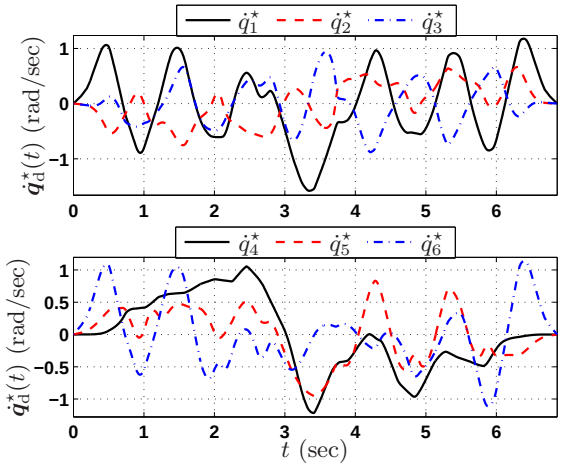
The tracking errors are clearly benefited from implementing a slower trajectory. This sug-



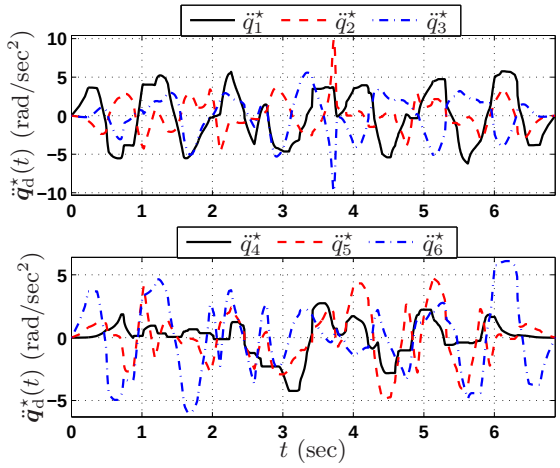
(a) s, $\dot{s}$, and $\ddot{s}$



(b) Motor-side optimal torques, $\boldsymbol{u}_{\mathrm{d}}^{\star} = \boldsymbol{G}^{-1}\boldsymbol{\tau}_{\mathrm{d}}^{\star}(t)$



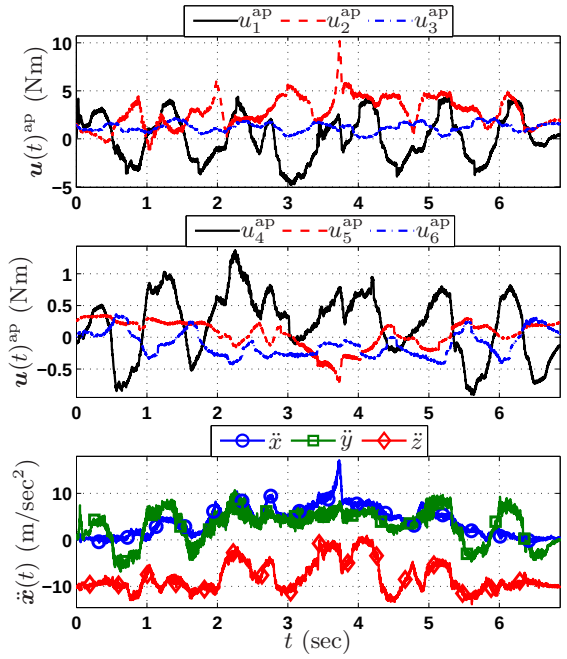
(c) Optimal velocity reference $\dot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$



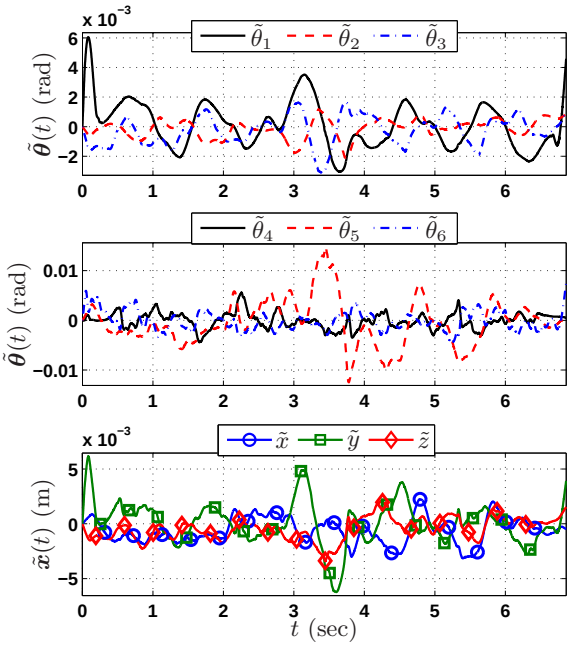
(d) Optimal acceleration reference $\ddot{\boldsymbol{q}}_{\mathrm{d}}^{\star}(t)$

Figure 3.8: Optimal solutions generated by solving optimization problem (3.13) for $\lambda = 0.2$. Notice that it is still the case that $c^{\star}(s) = \sqrt{b^{\star}(s)}, \forall s \in [0, 1]$.

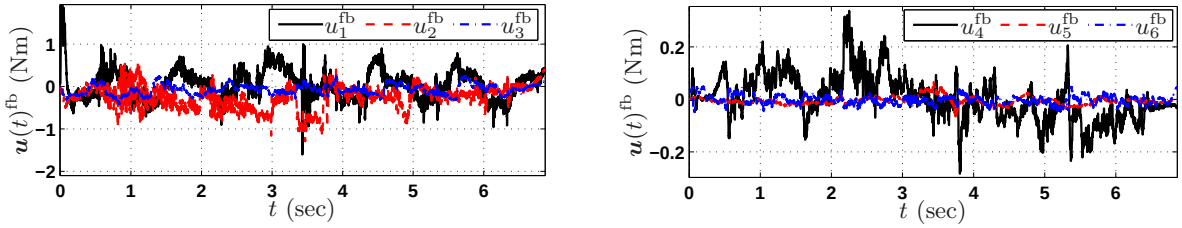
gests us that in order to achieve smaller tracking errors for the near time-optimal (fastest) trajectory, more sophisticated control algorithms should be explored. In other words, optimization problem (3.13) can generate the fastest optimal trajectory solutions that the actual robot manipulator can achieve, without degrading its performance. It is thus our next task to explore more advanced feedback control algorithms, in order to improve the performance for the near time-optimal solutions. We explore two related ideas in the next two chapters of this dissertation.



(a) Applied torques and accelerometer readings



(b) Motor-side and Cartesian-space tracking errors



(c) PID feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

Figure 3.9: Experimental results for the medium-speed optimal solutions which uses $\lambda = 0.2$.

3.4 Summary

In this chapter we presented an extension to the work done in Chapter 2. We developed a problem formulation that incorporates acceleration constraints and trades off time-optimality against a measure of total jerk. The resulting formulation generates optimal trajectories and torques that are near time-optimal. These near time-optimal solutions represent the fastest optimal solutions achievable by the real robot manipulator. Initially, from the formulation in Chapter 2, pure time-optimality was considered. Then, acceleration constraints and penalization of a measure of total jerk were incorporated, both of which proved necessary from real experiments on the 6-axis industrial manipulator. In all cases, the resulting optimal

trajectories and torques turned out to always be dynamically feasible with respect to the full nonlinear dynamic model (2.5). This brings not only a modest theoretical contribution, but also an important practical extension to existing algorithms. As discussed in the chapter, our methodology generates the actual fastest solutions that can be implemented in the real system, without significantly degrading the system performance. It was also explained how optimization problem (3.13) can be used to generate medium-speed optimal solutions.

Chapter 4

LQ-based Control Synthesis for Trajectory Tracking of Robot Manipulators

In Chapter 3, we have developed a formulation to generate near time-optimal trajectories and feedforward torques, which are dynamically feasible with respect to the complete nonlinear dynamic model (2.5). We studied how the generated optimal solutions do not seriously degrade the performance of the robot manipulator, even though they represent the fastest achievable solutions. Thus, an immediate question to ask in this dissertation follows: is it possible to improve the performance for the near time-optimal trajectories, by implementing feedback controllers that are more suited for trajectory tracking? In other words, we are still interested in the fastest achievable solutions generated by optimization problem (3.13), but attempt to replace the PID feedback portion of control law (3.14), with a more adequate algorithm for trajectory tracking. The main insight will be to linearize the nonlinear robot dynamics, and use the linearized model for controller synthesis. In the controller synthesis, we attempt to keep the tracking error as small as possible, while keeping the applied controls as close as possible to the open-loop optimal torques. In this regard, the Linear Quadratic (LQ) optimal control setting is appealing, since it will allow for the possibility of trading off tracking error and control effort [31, 38, 22]. In the present and the next chapter of this dissertation, we explore the trajectory tracking control problem for robot manipulators using LQ methods. Two different but related schemes will be developed, leading to a affine time-varying (ATV) controller in the first case, and to a piecewise affine (PWA) controller in the second case. The ATV results are presented in this chapter, whereas the PWA results are presented in Chapter 5. In both approaches, we study how under suitable assumptions, the LQ results for linear systems can be tailored to address the problem of trajectory tracking of robot manipulators, with dynamic model (2.5) which is inherently nonlinear.

4.1 Nonlinear Dynamic Model

Consider again the nonlinear dynamic model (2.5), repeated here for convenience:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{D}_{\mathrm{v}}\dot{\boldsymbol{q}} + \boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{4.1}$$

which is commonly used for nonlinear controller synthesis such as computed torque control [15, 19]. Let us write dynamic model (4.1) in the motor-side since all controller syntheses will be developed in the motor side. Recall that the link-side and motor-side positions are denoted by $\boldsymbol{q}$ and $\boldsymbol{\theta}$, respectively. In addition, the link-side and motor-side actuator torques are denoted by $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively. These motor-side and link-side variables are related by $\boldsymbol{\tau} = \boldsymbol{G}\boldsymbol{u}$ and $\boldsymbol{q} = \boldsymbol{G}^{-1}\boldsymbol{\theta}$, where $\boldsymbol{G}$ is a diagonal matrix with the gear ratios. Therefore, the following nonlinear dynamic model is used as the governing model in motor side:

$$\begin{aligned}\left[\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}\right]\ddot{\boldsymbol{\theta}} + \left[\boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta},\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) + \\ \left[\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{\theta}}) = \boldsymbol{u}.\end{aligned} \tag{4.2}$$

Since the inertia matrix $\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})$ is in general positive definite, we can write (4.2) in terms of the state vector $[\boldsymbol{\theta}^\top\ \dot{\boldsymbol{\theta}}^\top]^\top \in \mathbb{R}^{2n}$ as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{\theta}\\ \dot{\boldsymbol{\theta}}\end{bmatrix} = \boldsymbol{f}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{u}), \tag{4.3}$$

where the map $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^{2n}$ is defined as:

$$\boldsymbol{f}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{u}) = \begin{bmatrix}\dot{\boldsymbol{\theta}}\\ \boldsymbol{G}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})^{-1}\boldsymbol{G}\left\{\begin{array}{r}\boldsymbol{u} - \boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta},\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}} - \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) + \cdots\\ -\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}} - \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{\theta}})\end{array}\right\}\end{bmatrix} \tag{4.4}$$

Since the forthcoming development will require a nonlinear discrete-time model, the continuous time model (4.3) must be discretized [39]. There are several ways to obtain discrete-time nonlinear dynamic models from continuous-time nonlinear dynamic models [40]. In this dissertation, we use the first-order Euler's approximation, which produces a simple nonlinear discrete-time model that can be used for controller synthesis:

$$\begin{bmatrix}\boldsymbol{\theta}_{k+1}\\ \dot{\boldsymbol{\theta}}_{k+1}\end{bmatrix} = \boldsymbol{f}_d(\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k,\boldsymbol{u}_k), \tag{4.5}$$

where $\boldsymbol{f}_d : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^{2n}$ is defined as:

$$\boldsymbol{f}_d(\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k,\boldsymbol{u}_k) \approx \begin{bmatrix}\boldsymbol{\theta}_k\\ \dot{\boldsymbol{\theta}}_k\end{bmatrix} + T_s\boldsymbol{f}(\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k,\boldsymbol{u}_k), \tag{4.6}$$

with T_s being the servo sampling period.

4.2 LQ-based Trajectory Tracking

Even though the Linear Quadratic (LQ) optimal control setting is by definition a control synthesis technique for linear systems [38], we can tailor its results to solve the problem of trajectory tracking for robot manipulators with nonlinear dynamic model (4.5). The basic idea is that if the desired trajectory is known a-priori (which is actually the case for manipulators operating in industrial applications), we can linearize the nonlinear dynamics (4.5) along the desired trajectory [18, 20, 19]. This procedure will lead to approximate (4.5) by a linear time-varying (LTV) system, allowing thus the possibility of using LQ results for LTV systems.

4.2.1 The LQ Optimal Control Problem for LTV Systems

The discrete-time linear quadratic (LQ) optimal control problem for linear time-varying (LTV) systems, with finite horizon H, considers the discrete-time LTV dynamics:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k, \quad \boldsymbol{x}_0 = \boldsymbol{x}^{\text{init}}, \tag{4.7}$$

and aims to choosing the control sequence $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{H-1}$, that minimize the following quadratic cost function:

$$J(U) = \sum_{k=0}^{H-1} \left(\boldsymbol{x}_k^\top \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{R} \boldsymbol{u}_k \right) + \boldsymbol{x}_H^\top \boldsymbol{Q}_f \boldsymbol{x}_H, \tag{4.8}$$

where $U = (\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{H-1})$ and $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0$, $\boldsymbol{Q}_f = \boldsymbol{Q}_f^\top \succeq 0$, $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$. The first term measures state deviation from zero, the second measures input size or actuator authority, and the last term measures final state deviation. The LQ problem aims to finding $\boldsymbol{u}_0^\star, \ldots, \boldsymbol{u}_{H-1}^\star$, $\boldsymbol{x}_0^\star, \ldots, \boldsymbol{x}_H^\star$, that minimize $J(U)$ subject to the LTV dynamics (4.7). In other words, it can be seen as the following convex optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u}_k, \boldsymbol{x}_k}{\text{minimize}} \quad & \sum_{k=0}^{H-1} \left(\boldsymbol{x}_k^\top \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{R} \boldsymbol{u}_k \right) + \boldsymbol{x}_H^\top \boldsymbol{Q}_f \boldsymbol{x}_H \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k, \quad \boldsymbol{x}_0 = \boldsymbol{x}^{\text{init}} \\ & k = 0, 1, \ldots, H-1. \end{aligned} \tag{4.9}$$

An efficient way to solve the LQ problem (4.9) is through dynamic programming, which makes use of the principle of optimality [41, 42]. It is a standard exercise in optimal control, to show that the LQ solution via dynamic programming can be computed backwards in time:

1. set $\boldsymbol{P}_H = \boldsymbol{Q}_f$
2. for $k = H-1, \ldots, 0$,

$$\boldsymbol{P}_k = \boldsymbol{Q} + \boldsymbol{A}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k - \boldsymbol{A}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k (\boldsymbol{R} + \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k)^{-1} \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k$$

3. for $k = 0, \ldots, H-1$, define the time-varying state-feedback gain matrix $\boldsymbol{K}_k = (\boldsymbol{R} + \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k)^{-1} \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k$

4. for $k = 0, \ldots, H-1$, the optimal control law is given in feedback form $\boldsymbol{u}_k^\star = -\boldsymbol{K}_k \boldsymbol{x}_k$.

4.2.2 Trajectory Tracking of Robotic Manipulators as LQ for Affine Time-varying Systems

Suppose we want an n-DOF robot manipulator with discrete-time nonlinear dynamics (4.5), follow an optimal trajectory solution $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$ for all $k = 0, 1, \ldots, H$, where H is the finite time horizon. The optimal trajectory solution $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$ is generated through formulation (3.13) presented in Chapter 3, i.e., $\boldsymbol{\theta}_k^\star = \boldsymbol{\theta}^\star(T_s k)$, $\dot{\boldsymbol{\theta}}_k^\star = \dot{\boldsymbol{\theta}}^\star(T_s k)$, $\ddot{\boldsymbol{\theta}}_k^\star = \ddot{\boldsymbol{\theta}}^\star(T_s k)$, $\boldsymbol{u}_k^\star = \boldsymbol{u}^\star(T_s k)$, $k = 0, 1, \ldots, H$.

Assume that the robot dynamics is governed by the discrete-time nonlinear dynamics (4.5). Likewise assume that the robot state $(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k)$ and control input $\boldsymbol{u}_k$ will remain close to the nominal optimal state $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star)$ and control input $\boldsymbol{u}_k^\star$, respectively. Therefore, it is possible to linearize the nonlinear map $\boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k)$ along the nominal optimal solution $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$. The first-order Taylor series expansion of $\boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k)$ along $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$ gives:

$$\begin{aligned} \boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k) \;\approx\; & \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) + \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{\theta}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^\star) \\ & + \frac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{\theta}}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_k^\star) + \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{u}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\boldsymbol{u}_k - \boldsymbol{u}_k^\star), \end{aligned} \tag{4.10}$$

therefore the nonlinear discrete-time robot dynamics (4.5) can be approximated as:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\theta}_{k+1} \\ \dot{\boldsymbol{\theta}}_{k+1} \end{bmatrix} \;=\; & \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) + \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{\theta}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^\star) \\ & + \frac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{\theta}}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_k^\star) + \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{u}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)(\boldsymbol{u}_k - \boldsymbol{u}_k^\star). \end{aligned} \tag{4.11}$$

Consider subtracting $[\boldsymbol{\theta}_{k+1}^{\star\top} \;\; \dot{\boldsymbol{\theta}}_{k+1}^{\star\top}]^\top$ from both sides of (4.11), defining $\tilde{\boldsymbol{\theta}}_k := \boldsymbol{\theta}_k - \boldsymbol{\theta}_k^\star$, $\tilde{\boldsymbol{u}}_k := \boldsymbol{u}_k - \boldsymbol{u}_k^\star$, and defining the following time-varying matrices for $k = 0, 1, \ldots, H-1$:

$$\begin{aligned} \boldsymbol{A}_{\boldsymbol{\theta}_k} &:= \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{\theta}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) \in \mathbb{R}^{2n \times n}, \quad \boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k} := \frac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{\theta}}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) \in \mathbb{R}^{2n \times n} \\ \boldsymbol{B}_k &:= \frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{u}_k}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) \in \mathbb{R}^{2n \times n}, \quad \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix} := \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) - \begin{bmatrix} \boldsymbol{\theta}_{k+1}^\star \\ \dot{\boldsymbol{\theta}}_{k+1}^\star \end{bmatrix}. \end{aligned} \tag{4.12}$$

Therefore,

$$\begin{bmatrix} \tilde{\boldsymbol{\theta}}_{k+1} \\ \dot{\tilde{\boldsymbol{\theta}}}_{k+1} \end{bmatrix} = \boldsymbol{A}_{\boldsymbol{\theta}_k} \tilde{\boldsymbol{\theta}}_k + \boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k} \dot{\tilde{\boldsymbol{\theta}}}_k + \boldsymbol{B}_k \tilde{\boldsymbol{u}}_k + \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix}, \tag{4.13}$$

It is then clear that by defining $\boldsymbol{A}_k := [\boldsymbol{A}_{\boldsymbol{\theta}_k} \ \boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k}] \in \mathbb{R}^{2n\times 2n}$, the discrete-time nonlinear robot dynamics (4.5) is approximated by the following affine time-varying (ATV) dynamics:

$$\begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k+1}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k+1}\end{bmatrix} = \boldsymbol{A}_k \begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k}\end{bmatrix} + \boldsymbol{B}_k\tilde{\boldsymbol{u}}_k + \begin{bmatrix}\boldsymbol{v}_k\\ \boldsymbol{w}_k\end{bmatrix}. \tag{4.14}$$

Ideally, by the definition in (4.12), $\forall k$ the vector $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top$ in (4.14) should be $\boldsymbol{0}_{2n}$ if $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$ is dynamically feasible with respect to the non-linear discrete-time dynamics (4.5). However, it will not be the case. To see the reason, recall that the 4-tuple optimal solution $(\boldsymbol{\theta}^\star(T_sk), \dot{\boldsymbol{\theta}}^\star(T_sk), \ddot{\boldsymbol{\theta}}^\star(T_sk), \boldsymbol{u}^\star(T_sk))$ is indeed dynamically feasible with respect to the complete continuous-time dynamic model (4.2)-(4.3). However, due to the discretization approximation process to obtain (4.5), the trajectory solution $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$, $k = 1, \ldots, H$, will not be exactly dynamically feasible with respect to the discrete-time nonlinear dynamic model (4.5). To prove it, we simply use the definition of $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top$ in (4.14), and also use the definition of $\boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k)$ in (4.6):

$$\begin{aligned}
\begin{bmatrix}\boldsymbol{v}_k\\ \boldsymbol{w}_k\end{bmatrix} &= \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) - \begin{bmatrix}\boldsymbol{\theta}_{k+1}^\star\\ \dot{\boldsymbol{\theta}}_{k+1}^\star\end{bmatrix}\\
&= \begin{bmatrix}\boldsymbol{\theta}_{k}^\star\\ \dot{\boldsymbol{\theta}}_{k}^\star\end{bmatrix} + T_s\boldsymbol{f}(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) - \begin{bmatrix}\boldsymbol{\theta}_{k+1}^\star\\ \dot{\boldsymbol{\theta}}_{k+1}^\star\end{bmatrix}\\
&= \begin{bmatrix}\boldsymbol{\theta}_{k}^\star\\ \dot{\boldsymbol{\theta}}_{k}^\star\end{bmatrix} + \begin{bmatrix}T_s\dot{\boldsymbol{\theta}}_{k}^\star\\ T_s\ddot{\boldsymbol{\theta}}_{k}^\star\end{bmatrix} - \begin{bmatrix}\boldsymbol{\theta}_{k+1}^\star\\ \dot{\boldsymbol{\theta}}_{k+1}^\star\end{bmatrix}\\
&= \begin{bmatrix}\boldsymbol{\theta}_{k}^\star + T_s\dot{\boldsymbol{\theta}}_{k}^\star - \boldsymbol{\theta}_{k+1}^\star\\ \dot{\boldsymbol{\theta}}_{k}^\star + T_s\ddot{\boldsymbol{\theta}}_{k}^\star - \dot{\boldsymbol{\theta}}_{k+1}^\star\end{bmatrix} \neq \begin{bmatrix}\boldsymbol{0}_n\\ \boldsymbol{0}_n\end{bmatrix}
\end{aligned}$$

We are therefore interested in solving the LQ optimal control problem with affine time-varying dynamics (4.14), namely:

$$\begin{aligned}
\underset{\tilde{\boldsymbol{u}}_k, \tilde{\boldsymbol{\theta}}_k, \dot{\tilde{\boldsymbol{\theta}}}_k}{\text{minimize}} \ & \sum_{k=0}^{H-1}\left(\begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{Q}_2\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k}\end{bmatrix} + \tilde{\boldsymbol{u}}_k^\top\boldsymbol{R}\tilde{\boldsymbol{u}}_k\right) + \begin{bmatrix}\tilde{\boldsymbol{\theta}}_{H}\\ \dot{\tilde{\boldsymbol{\theta}}}_{H}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{Q}_{f_1} & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{Q}_{f_2}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\theta}}_{H}\\ \dot{\tilde{\boldsymbol{\theta}}}_{H}\end{bmatrix}\\
\text{subject to} \ & \begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k+1}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k+1}\end{bmatrix} = \boldsymbol{A}_k\begin{bmatrix}\tilde{\boldsymbol{\theta}}_{k}\\ \dot{\tilde{\boldsymbol{\theta}}}_{k}\end{bmatrix} + \boldsymbol{B}_k\tilde{\boldsymbol{u}}_k + \begin{bmatrix}\boldsymbol{v}_k\\ \boldsymbol{w}_k\end{bmatrix}, \quad k = 0, 1, \ldots, H-1,
\end{aligned} \tag{4.15}$$

where $\boldsymbol{Q}_1 \succeq 0$ penalizes position tracking error $\tilde{\boldsymbol{\theta}}_k$ deviation from zero, $\boldsymbol{Q}_2 \succeq 0$ penalizes velocity tracking error $\dot{\tilde{\boldsymbol{\theta}}}_k$ deviation from zero, and $\boldsymbol{R} \succ 0$ penalizes $\tilde{\boldsymbol{u}}_k$ which represents deviation from the nominal optimal control input $\boldsymbol{u}_k^\star$. Therefore, the optimal control problem (4.15), if solved, allows the possibility to design feedback controllers that guarantee a position tracking error $\tilde{\boldsymbol{\theta}}_k$ as small as possible, while keeping the applied controls $\boldsymbol{u}_k$ as close as possible to the open-loop optimal torques $\boldsymbol{u}_k^\star$.

4.2.3 Reformulation as Standard LQ for LTV Systems

We could derive the solution to the optimal control problem (4.15). However, in this dissertation, we are interested in a method for solving problem (4.15) using the standard LQ solution to problem (4.9). The key insight resides in noticing that the affine time-varying dynamics in (4.15) is equivalent to

$$\begin{bmatrix} \tilde{\boldsymbol{\theta}}_{k+1} \\ \dot{\tilde{\boldsymbol{\theta}}}_{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_k & \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix} \\ \boldsymbol{0}_{2n}^\top & 1 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\theta}}_k \\ \dot{\tilde{\boldsymbol{\theta}}}_k \\ 1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_k \\ \boldsymbol{0}_n^\top \end{bmatrix} \tilde{\boldsymbol{u}}_k.$$

Therefore we can define the following time-varying matrices and state vector $\bar{\boldsymbol{x}}_k$:

$$\begin{aligned} \bar{\boldsymbol{A}}_k := \begin{bmatrix} \boldsymbol{A}_k & \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix} \\ \boldsymbol{0}_{2n}^\top & 1 \end{bmatrix} \in \mathbb{R}^{(2n+1)\times(2n+1)}, \quad \bar{\boldsymbol{B}}_k := \begin{bmatrix} \boldsymbol{B}_k \\ \boldsymbol{0}_n^\top \end{bmatrix} \in \mathbb{R}^{(2n+1)\times n} \\ \bar{\boldsymbol{x}}_k := \begin{bmatrix} \tilde{\boldsymbol{\theta}}_k \\ \dot{\tilde{\boldsymbol{\theta}}}_k \\ 1 \end{bmatrix} \in \mathbb{R}^{2n+1}, \quad k = 0, 1, \ldots, H. \end{aligned} \tag{4.16}$$

With this definition, the affine time-varying dynamics in problem (4.15) is equivalent to the linear time-varying dynamics:

$$\bar{\boldsymbol{x}}_{k+1} = \bar{\boldsymbol{A}}_k \bar{\boldsymbol{x}}_k + \bar{\boldsymbol{B}}_k \tilde{\boldsymbol{u}}_k, \quad k = 0, 1, \ldots, H-1. \tag{4.17}$$

On the other hand, the objective function in problem (4.15) can be expressed in terms of the enlarged state vector $\bar{\boldsymbol{x}}_k$ by defining:

$$\bar{\boldsymbol{Q}} = \begin{pmatrix} \boldsymbol{Q}_1 & \boldsymbol{O} & \boldsymbol{0} \\ \boldsymbol{O} & \boldsymbol{Q}_2 & \boldsymbol{0} \\ \boldsymbol{0}^\top & \boldsymbol{0}^\top & 0 \end{pmatrix}, \quad \bar{\boldsymbol{Q}}_f = \begin{pmatrix} \boldsymbol{Q}_{f_1} & \boldsymbol{O} & \boldsymbol{0} \\ \boldsymbol{O} & \boldsymbol{Q}_{f_2} & \boldsymbol{0} \\ \boldsymbol{0}^\top & \boldsymbol{0}^\top & 0 \end{pmatrix},$$

which allows to finally re-formulate problem (4.15) as a standard LQ optimal control problem for linear time-varying systems:

$$\begin{aligned} \underset{\tilde{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k}{\text{minimize}} \quad & \sum_{k=0}^{H-1} \left(\bar{\boldsymbol{x}}_k^\top \bar{\boldsymbol{Q}} \bar{\boldsymbol{x}}_k + \tilde{\boldsymbol{u}}_k^\top \boldsymbol{R} \tilde{\boldsymbol{u}}_k \right) + \bar{\boldsymbol{x}}_H^\top \bar{\boldsymbol{Q}}_f \bar{\boldsymbol{x}}_H \\ \text{subject to} \quad & \bar{\boldsymbol{x}}_{k+1} = \bar{\boldsymbol{A}}_k \bar{\boldsymbol{x}}_k + \bar{\boldsymbol{B}}_k \tilde{\boldsymbol{u}}_k \\ & k = 0, 1, \ldots, H-1. \end{aligned} \tag{4.18}$$

There is no qualitative difference between the general LQ formulation (4.9) for LTV systems, and the formulation of the tracking problem for robot manipulators in (4.18). Therefore, problem (4.18) can be solved exactly. Alluding to the backward solution presented

beforehand for the general LQ for LTV systems, the optimal solution $\tilde{\boldsymbol{u}}_k^\star$ takes a feedback form of the enlarged state vector $\bar{\boldsymbol{x}}_k$, i.e.,

$$\tilde{\boldsymbol{u}}_k^\star = -\bar{\boldsymbol{K}}_k \bar{\boldsymbol{x}}_k, \quad k = 0, 1, \ldots, H-1, \tag{4.19}$$

where the time-varying feedback gain matrices $\bar{\boldsymbol{K}}_k \in \mathbb{R}^{n\times(2n+1)}$, $k = 0, 1, \ldots, H-1$, are pre-computed backwards in time, i.e., set $\boldsymbol{P}_H = \bar{\boldsymbol{Q}}_f$ and then for $k = H-1, \ldots, 1, 0$, iterate:

$$\begin{aligned} \bar{\boldsymbol{K}}_k &= (\boldsymbol{R} + \bar{\boldsymbol{B}}_k^\top \boldsymbol{P}_{k+1} \bar{\boldsymbol{B}}_k)^{-1} \bar{\boldsymbol{B}}_k^\top \boldsymbol{P}_{k+1} \bar{\boldsymbol{A}}_k \\ \boldsymbol{P}_k &= \bar{\boldsymbol{Q}} + \bar{\boldsymbol{K}}_k^\top \boldsymbol{R} \bar{\boldsymbol{K}}_k + (\bar{\boldsymbol{A}}_k + \bar{\boldsymbol{B}}_k \bar{\boldsymbol{K}}_k)^\top \boldsymbol{P}_{k+1} (\bar{\boldsymbol{A}}_k + \bar{\boldsymbol{B}}_k \bar{\boldsymbol{K}}_k). \end{aligned} \tag{4.20}$$

4.2.4 Time-varying Affine Control Law

To actually carry out trajectory tracking for the discrete-time nonlinear robot model (4.5), we need to spell out what the actual control law $\boldsymbol{u}_k$, $k = 0, 1, \ldots, H-1$, should be. To do that, let us conveniently partition the time-varying feedback gain matrix $\bar{\boldsymbol{K}}_k \in \mathbb{R}^{n\times(2n+1)}$ as:

$$\bar{\boldsymbol{K}}_k = \begin{bmatrix} \boldsymbol{K}_1(k) & \boldsymbol{K}_2(k) & \boldsymbol{\alpha}_k \end{bmatrix}, \quad k = 0, 1, \ldots, H-1, \tag{4.21}$$

where $\boldsymbol{K}_1(k) \in \mathbb{R}^{n\times n}$, $\boldsymbol{K}_2(k) \in \mathbb{R}^{n\times n}$, and $\boldsymbol{\alpha}_k \in \mathbb{R}^n$, are time varying. From the definition of the enlarged vector $\bar{\boldsymbol{x}}_k$ in (4.16), the control law in (4.19) takes the following form:

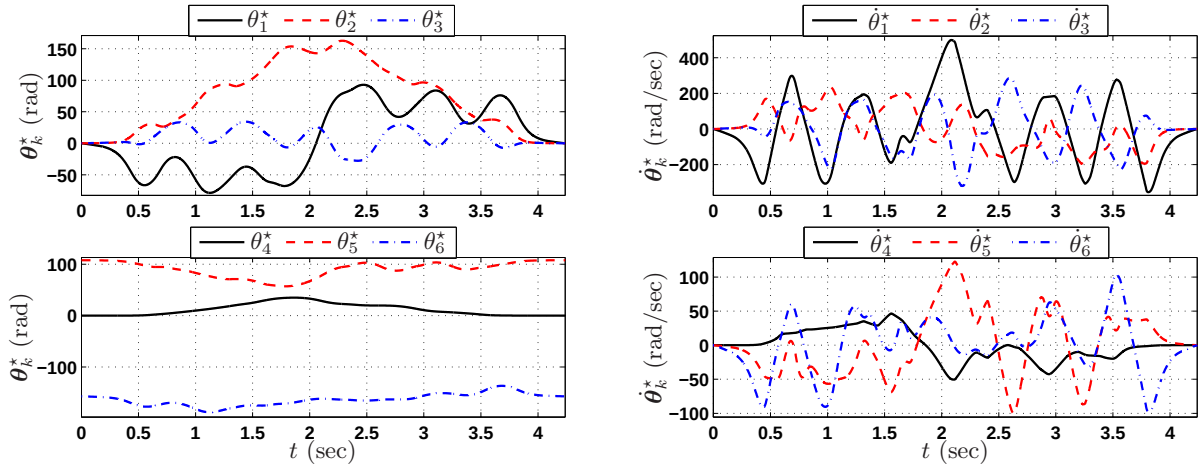
$$\begin{aligned} \tilde{\boldsymbol{u}}_k^\star &= -\boldsymbol{K}_1(k)\tilde{\boldsymbol{\theta}}_k - \boldsymbol{K}_2(k)\dot{\tilde{\boldsymbol{\theta}}}_k - \boldsymbol{\alpha}_k \\ &= -\boldsymbol{K}_1(k)(\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^\star) - \boldsymbol{K}_2(k)(\dot{\boldsymbol{\theta}}_k - \dot{\boldsymbol{\theta}}_k^\star) - \boldsymbol{\alpha}_k \\ &= \boldsymbol{K}_1(k)(\boldsymbol{\theta}_k^\star - \boldsymbol{\theta}_k) + \boldsymbol{K}_2(k)(\dot{\boldsymbol{\theta}}_k^\star - \dot{\boldsymbol{\theta}}_k) - \boldsymbol{\alpha}_k. \end{aligned}$$

Recalling that $\tilde{\boldsymbol{u}}_k^\star = \boldsymbol{u}_k - \boldsymbol{u}_k^\star$, the final control law is given by:

$$\boldsymbol{u}_k = \boldsymbol{u}_k^\star - \boldsymbol{\alpha}_k + \boldsymbol{K}_1(k)(\boldsymbol{\theta}_k^\star - \boldsymbol{\theta}_k) + \boldsymbol{K}_2(k)(\dot{\boldsymbol{\theta}}_k^\star - \dot{\boldsymbol{\theta}}_k). \tag{4.22}$$

The term $\boldsymbol{u}_k^\star$ represents the nonlinear optimal feedforward torque, generated from optimization problem (3.13). Note that since $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top \neq \boldsymbol{0}_{2n}$, an additional term in the optimal feedforward torque $\boldsymbol{u}_k^\star$ is subtracted, i.e., $\boldsymbol{\alpha}_k$. The term $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top$ can actually be thought of as a known disturbance, for when trying to model the robot dynamics with the affine time-varying dynamics (4.14). The term $-\boldsymbol{\alpha}_k$ in control law (4.22), therefore aims at compensating for the effect of the "disturbance" term $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top$.

On the other hand, notice that the feedback part of the control law (4.22) can be thought of as a PD controller with time-varying matrices $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, $k = 0, 1, \ldots, H-1$. These time-varying matrices, however, are not diagonal, unlike those constant feedback gains $\boldsymbol{K}_\mathrm{P}$, $\boldsymbol{K}_\mathrm{V}$ in control law (2.25) from Chapter 2. The off-diagonal terms in time-varying matrices $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, $k = 0, 1, \ldots, H-1$, are nonzero due to the Multiple-input Multiple-output (MIMO) property of optimal control problem (4.15).



(a) Optimal trajectory position $\boldsymbol{\theta}_k^\star$

(b) Optimal trajectory velocity $\dot{\boldsymbol{\theta}}_k^\star$

(c) Optimal feedforward torques $\boldsymbol{u}_k^\star$

Figure 4.1: Near time-optimal trajectory positions, velocities, and torques, generated with optimization problem (3.13) for $\lambda = 0.02$. Along this optimal trajectory, the discrete-time nonlinear dynamics (4.5) is approximated as affine time-varying dynamics (4.14).

4.3 Controller Synthesis for 6-axis Manipulator

In this section, we present the necessary tools to implement control law (4.22) for the 6-DOF industrial manipulator FANUC M-16*i*B. In this case, the degrees of freedom $n = 6$, which means that the state vector $[\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top]^\top \in \mathbb{R}^{12}$. Since both the simulator and experimental setup run at a 1-millisecond sampling period (i.e., $T_s = 1$ millisecond), the number of linearization points will be relatively large. For instance, in Fig. 4.1 the optimal reference positions and velocities $\boldsymbol{\theta}_k^\star$, $\dot{\boldsymbol{\theta}}_k^\star$ and optimal torques $\boldsymbol{u}_k^\star$ are presented. Along this optimal trajectory, the linearization of the nonlinear dynamics (4.14) is performed. This near time-optimal trajectory solution is generated by solving optimization problem (3.13) for $\lambda = 0.02$ and then performing the appropriate conversions to motor-side. The total traversal time is $t_f = 4.238$ seconds, which means that the linearization is performed and stored for 4,238 points. I.e., a total of 4,238 matrices $\boldsymbol{A}_{\boldsymbol{\theta}_k} \in \mathbb{R}^{12\times 6}$, $\boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k} \in \mathbb{R}^{12\times 6}$, $\boldsymbol{B}_k \in \mathbb{R}^{12\times 6}$, $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top \in \mathbb{R}^{12}$, defined in (4.12), need to be numerically computed and stored for subsequent controller synthesis.

4.3.1 Linearization along the Reference Trajectory

Since $n = 6$, the map $\boldsymbol{f} : \mathbb{R}^6 \times \mathbb{R}^6 \times \mathbb{R}^6 \mapsto \mathbb{R}^{12}$ defined in (4.4), needs to be computed efficiently to carry out linearization at a large number of points. These computations are done very fast using the recursive Newton-Euler algorithm [17], which has been coded in C and therefore a MEX file can be executed within MATLAB® to substantially speed up the computations [35]. Likewise, since the sign$(\cdot)$ function is discontinuous at zero, i.e.,

$$\text{sign}(x) := \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}, \tag{4.23}$$

we approximate it by a smooth continuous function, satur$(\cdot)$, defined as follows:

$$\text{satur}(x) := \frac{\tan^{-1}(\eta x) - \tan^{-1}(-\eta x)}{\pi}, \tag{4.24}$$

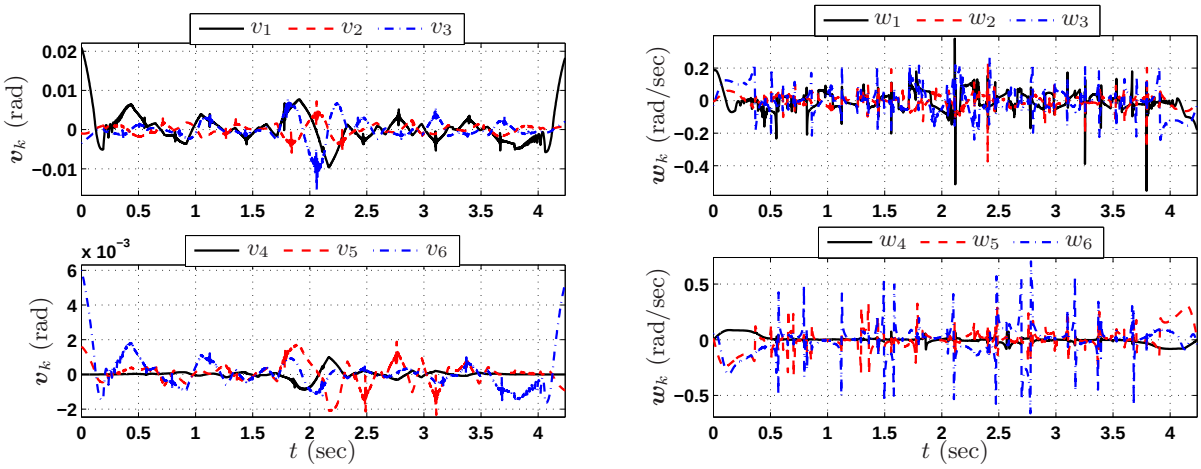
where $\eta > 0$ is a control parameter that can be used to refine the approximation to the sign$(\cdot)$ function, namely, the larger η, the more accurate the approximation. In this dissertation, we use $\eta = 25$ since we need the function satur$(\cdot)$ to be differentiable at $\dot{\theta}_i = 0$, $i = 1, 2, \ldots, 6$. The reason is that the velocity reference $\dot{\boldsymbol{\theta}}_k^\star$, from Fig. 4.1(b), crosses zero at different instants of time. Therefore, for linearization purposes along this optimal reference trajectory, the following nonlinear map is used:

$$\boldsymbol{f}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \boldsymbol{G}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})^{-1}\boldsymbol{G} \left\{ \begin{array}{r} \boldsymbol{u} - \boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta}, \boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}} - \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) + \cdots \\ -\boldsymbol{G}^{-1}\boldsymbol{D}_{\text{v}}\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}} - \boldsymbol{G}^{-1}\boldsymbol{F}_{\text{C}}\,\text{satur}(\dot{\boldsymbol{\theta}}) \end{array} \right\} \end{bmatrix}. \tag{4.25}$$

For a given point $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{u})$, the recursive Newton-Euler algorithm needs to be invoked eight times to do one computation of $\boldsymbol{f}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{u})$. Namely, one call to compute $\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) \in \mathbb{R}^6$, one call for $\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta}, \boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}} \in \mathbb{R}^6$, and six calls to calculate $\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) \in \mathbb{R}^{6\times 6}$. From the definition of the discrete-time nonlinear map $\boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k)$ in (4.6), the same number of calls are needed for one evaluation of $\boldsymbol{f}_d(\boldsymbol{\theta}_k, \dot{\boldsymbol{\theta}}_k, \boldsymbol{u}_k)$.

The computation of $\boldsymbol{A}_{\boldsymbol{\theta}_k} \in \mathbb{R}^{12\times 6}$, $\boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k} \in \mathbb{R}^{12\times 6}$, $\boldsymbol{B}_k \in \mathbb{R}^{12\times 6}$, defined in (4.12), is done numerically column-by-column as follows: for $i = 1, \ldots, 6$, the i-th columns of $\boldsymbol{A}_{\boldsymbol{\theta}_k}$, $\boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k}$, $\boldsymbol{B}_k$, are approximated by the following directional derivatives:

$$\begin{aligned} \boldsymbol{A}_{\boldsymbol{\theta}_k}(:, i) &\approx \frac{\boldsymbol{f}_d(\boldsymbol{\theta}_k^\star + \varepsilon\boldsymbol{e}_i, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star) - \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star - \varepsilon\boldsymbol{e}_i, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)}{2\varepsilon} \\ \boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k}(:, i) &\approx \frac{\boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star + \varepsilon\boldsymbol{e}_i, \boldsymbol{u}_k^\star) - \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star - \varepsilon\boldsymbol{e}_i, \boldsymbol{u}_k^\star)}{2\varepsilon} \\ \boldsymbol{B}_k(:, i) &\approx \frac{\boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star + \varepsilon\boldsymbol{e}_i) - \boldsymbol{f}_d(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star - \varepsilon\boldsymbol{e}_i)}{2\varepsilon}, \end{aligned} \tag{4.26}$$



(a) Position "disturbance" $\boldsymbol{v}_k$ (b) Velocity "disturbance" $\boldsymbol{w}_k$

Figure 4.2: Disturbance terms $\boldsymbol{v}_k \in \mathbb{R}^6$ and $\boldsymbol{w}_k \in \mathbb{R}^6$, which respectively represent position "disturbance" and velocity "disturbance" due to non-exact dynamic feasibility. These terms actually show up because of: (i) the approximation performed to obtain a nonlinear discrete-time dynamic model from a nonlinear continuous-time dynamic model and (ii) from high-order terms in the nonlinear discrete-time dynamics that are ignored when linearizing along the nominal trajectory.

where $\varepsilon = 10^{-4}$, and $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_6\}$ is the canonical basis for $\mathbb{R}^6$ [43]. These computations are carried out along the entire optimal trajectory $(\boldsymbol{\theta}_k^\star, \dot{\boldsymbol{\theta}}_k^\star, \boldsymbol{u}_k^\star)$, $k = 0, 1, 2, \ldots, H$.

The near time-optimal trajectory presented in Fig. 4.1 requires a time horizon $H = 4,238$. It means that 4238 matrices $\boldsymbol{A}_{\boldsymbol{\theta}_k} \in \mathbb{R}^{12\times 6}$, $\boldsymbol{A}_{\dot{\boldsymbol{\theta}}_k} \in \mathbb{R}^{12\times 6}$, and $\boldsymbol{B}_k \in \mathbb{R}^{12\times 6}$, must be computed from (4.26), and then stored for controller synthesis. In addition, the "disturbance" term $[\boldsymbol{v}_k^\top \ \boldsymbol{w}_k^\top]^\top$ is computed from its definition in (4.12). The total computation time to obtain the referred matrices using MATLAB® is around 55 seconds, on an Intel(R) Core(TM)2 Duo CPU @ 2.5 Ghz.

The resulting disturbance terms $\boldsymbol{v}_k \in \mathbb{R}^6$, $\boldsymbol{w}_k \in \mathbb{R}^6$ are shown in Fig. 4.2. From their definition in (4.12), and from the manner in which they appear in the time-varying affine dynamics (4.14), $\boldsymbol{v}_k$ can be interpreted as a "disturbance" in position whereas $\boldsymbol{w}_k$ as a "disturbance" in velocity. Clearly from Fig. 4.2, these disturbances are small (but not zero) compared to the optimal trajectory positions and velocities $\boldsymbol{\theta}_k^\star$, $\dot{\boldsymbol{\theta}}_k^\star$ presented in Fig. 4.1.

With all the above provisos in mind, matrices $\bar{\boldsymbol{A}}_k$, $\bar{\boldsymbol{B}}_k$, $k = 0, 1, \ldots, H-1$, are constructed from their definition in (4.16). To complete the parameters needed to solve optimal control problem (4.18), it remains to choose the weighting matrices $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_{f_1}$, $\boldsymbol{Q}_{f_2}$, and $\boldsymbol{R}$. A first choice for the matrices $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{R}$ is given by the Bryson's rule [44]: select $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$,

and $\boldsymbol{R}$ diagonal with the following entries:

$$[\boldsymbol{Q}_1]_{ii} = \frac{1}{\text{maximum acceptable value of } \left[\tilde{\boldsymbol{\theta}}_k\right]_i^2}, \quad i = 1, 2, \ldots, 6$$

$$[\boldsymbol{Q}_2]_{ii} = \frac{1}{\text{maximum acceptable value of } \left[\dot{\tilde{\boldsymbol{\theta}}}_k\right]_i^2}, \quad i = 1, 2, \ldots, 6$$

$$\boldsymbol{R}_{ii} = \frac{1}{\text{maximum acceptable value of } [\tilde{\boldsymbol{u}}_k]_i^2}, \quad i = 1, 2, \ldots, 6.$$

Essentially this rule scales each variable in the objective function (4.15) so that the maximum acceptable value for each term is one. Although Bryson's rule gives sometimes good results, it is just the starting point to a trial-and-error procedure until desirable properties of the closed-loop system are attained.

In this dissertation, we use Bryson's rule in a way so as to produce time-varying feedback matrices $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, in (4.22), that are "comparable" to the constant feedback gains $\boldsymbol{K}_\mathrm{P}$, $\boldsymbol{K}_\mathrm{V}$, in control law (2.25) from Chapter 2. The constant feedback gain matrices $\boldsymbol{K}_\mathrm{P}$, $\boldsymbol{K}_\mathrm{V}$, in control law (2.25) are:

$$\boldsymbol{K}_\mathrm{P} = \mathrm{diag}(4.4717,\ 4.0786,\ 0.8297,\ 1.4449,\ 0.0399,\ 0.1415)$$
$$\boldsymbol{K}_\mathrm{V} = \mathrm{diag}(0.4255,\ 0.3910,\ 0.0810,\ 0.0622,\ 0.0025,\ 0.0069),$$

which are diagonal since these gains are designed in a decentralized manner. On the other hand, the optimal control formulation (4.15) is intrinsically MIMO, and therefore will produce time-varying gains $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, with nonzero off-diagonal elements to account for the coupled dynamics. We thus use Bryson's rule but tune the weighting matrices so that the diagonal elements of $\boldsymbol{K}_1(0)$, $\boldsymbol{K}_2(0)$ (i.e., at $k = 0$) are approximate to those of $\boldsymbol{K}_\mathrm{P}$, $\boldsymbol{K}_\mathrm{V}$. Of course, the off-diagonal terms of $\boldsymbol{K}_1(0)$, $\boldsymbol{K}_2(0)$ will not be zero due to the coupled dynamics. Besides, $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$ are time-varying, therefore the referred design requirement will not be necessarily true for $k = 1, 2, \ldots, H-1$.

With these considerations, weighting matrices $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$, which respectively penalize deviation from zero of $\tilde{\boldsymbol{\theta}}_k$ and $\dot{\tilde{\boldsymbol{\theta}}}_k$, are conveniently set as:

$$\boldsymbol{Q}_1 = \mathrm{diag}\left(\frac{1}{1^2},\ \frac{1}{1^2},\ \frac{1}{1^2},\ \frac{1}{1^2},\ \frac{1}{1^2},\ \frac{1}{1^2}\right) = \boldsymbol{I}$$
$$\boldsymbol{Q}_2 = \mathrm{diag}\left(\frac{1}{10^2},\ \frac{1}{10^2},\ \frac{1}{10^2},\ \frac{1}{10^2},\ \frac{1}{10^2},\ \frac{1}{10^2}\right) = 0.01\boldsymbol{I},$$

where $\boldsymbol{I}$ is the 6×6 identity matrix. In this way, the weighting matrix $\boldsymbol{R}$, which penalizes $\tilde{\boldsymbol{u}}_k$ deviation from zero (i.e., $\boldsymbol{u}_k$ deviation from the nominal optimal torque $\boldsymbol{u}_k^\star$), is tuned as:

$$\boldsymbol{R} = \mathrm{diag}\left(\frac{1}{5.06^2},\ \frac{1}{4.63^2},\ \frac{1}{0.93^2},\ \frac{1}{1.55^2},\ \frac{1}{0.06^2},\ \frac{1}{0.21^2}\right).$$

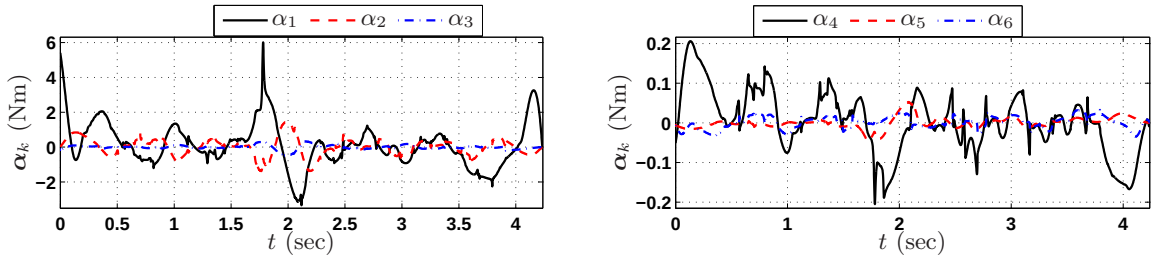


Figure 4.3: Resulting compensation torque $\boldsymbol{\alpha}_k$ in control law (4.22). This term could be zero only when the disturbance terms $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are zero.

For simplicity, $\boldsymbol{Q}_{f_1} = \boldsymbol{Q}_1$, and $\boldsymbol{Q}_{f_2} = \boldsymbol{Q}_2$, however, for convenience these weighting matrices are tuned as $\boldsymbol{Q}_{f_1} = 100 \cdot \boldsymbol{Q}_1$, and $\boldsymbol{Q}_{f_2} = 6 \cdot \boldsymbol{Q}_2$, which completes the parameters needed in the optimal control problem. Solving the referred optimal control problem (4.18), yields $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, and the torques $\boldsymbol{\alpha}_k$, $k = 0, 1, \ldots, H-1$. The resulting torques $\boldsymbol{\alpha}_k$, which result due to the terms $\boldsymbol{v}_k$, $\boldsymbol{w}_k$, are shown in Fig. 4.3. Note that these torques are nonzero because $\boldsymbol{v}_k, \boldsymbol{w}_k \neq \boldsymbol{0}$. The resulting feedback gain matrices at time $k = 0$, i.e., $\boldsymbol{K}_1(0)$, $\boldsymbol{K}_2(0)$, are:

$$\boldsymbol{K}_1(0) = \begin{pmatrix} 4.4724 & 0.0026 & 0.0032 & 0.1320 & -0.0079 & 0.3676 \\ 0.0120 & 4.0712 & -1.0495 & -0.0016 & 0.1784 & 0.0026 \\ -0.0014 & 0.2924 & 0.8258 & 0.0050 & -0.0548 & -0.0002 \\ -0.0057 & -0.0009 & -0.0032 & 1.4448 & -0.0070 & -0.1121 \\ -0.0001 & -0.0040 & 0.0042 & -0.0002 & 0.0415 & 0.0001 \\ -0.0371 & 0.0001 & -0.0003 & 0.0172 & 0.0002 & 0.1491 \end{pmatrix},$$

$$\boldsymbol{K}_2(0) = \begin{pmatrix} 0.4254 & 0.0007 & 0.0001 & -0.0014 & -0.0003 & 0.0757 \\ 0.0017 & 0.4044 & -0.1043 & 0.0001 & 0.0576 & 0.0002 \\ -0.0002 & 0.0355 & 0.0769 & 0.0007 & -0.0077 & 0.0000 \\ -0.0062 & 0.0001 & -0.0003 & 0.0838 & 0.0000 & -0.0140 \\ 0.0000 & -0.0005 & 0.0008 & -0.0000 & 0.0016 & 0.0000 \\ -0.0046 & 0.0000 & -0.0000 & 0.0041 & 0.0000 & 0.0064 \end{pmatrix}.$$

The diagonal elements of $\boldsymbol{K}_1(0)$ and $\boldsymbol{K}_2(0)$ are numerically approximate to the constant feedback gains $\boldsymbol{K}_\mathrm{P}$ and $\boldsymbol{K}_\mathrm{V}$. Note however that the off-diagonal elements of $\boldsymbol{K}_1(0)$ and $\boldsymbol{K}_2(0)$ are nonzero since our controller synthesis considers the robot coupled dynamics. Besides, $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k)$, are time-varying, which means that for certain time instants, say j, the feedback gain matrices $\boldsymbol{K}_1(j)$ and $\boldsymbol{K}_2(j)$, might not be feasible for implementation on the experiments.[1] For this trajectory, a total of 4237 matrices $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$ are generated. We wish to present all these matrices $\forall k$. A compact, yet informative way, is through their maximum and minimum singular values, i.e., $\sigma_{\max}(\boldsymbol{K}_1(k))$, $\sigma_{\min}(\boldsymbol{K}_1(k))$, $\sigma_{\max}(\boldsymbol{K}_2(k))$ and $\sigma_{\min}(\boldsymbol{K}_2(k))$, $k = 0, 1, \ldots, 4237$. The resulting singular values are shown in Fig. 4.4, which for reference purposes also includes $\sigma_{\max}(\boldsymbol{K}_\mathrm{P})$, $\sigma_{\min}(\boldsymbol{K}_\mathrm{P})$, $\sigma_{\max}(\boldsymbol{K}_\mathrm{V})$ and $\sigma_{\min}(\boldsymbol{K}_\mathrm{V})$.

This manner of presenting the controller synthesis allows to visualize graphically the maximum gain amplifications of $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$ (i.e., $\sigma_{\max}(\boldsymbol{K}_1(k))$ and $\sigma_{\max}(\boldsymbol{K}_2(k))$). A couple

[1] It was indeed the case, when first carrying out experiments using the referred time-varying feedback gains, that large commanded torques caused our first experiments to fail. It was then necessary to plot the maximum singular values of the feedback gain matrices to determine the cause of these undesired effects.

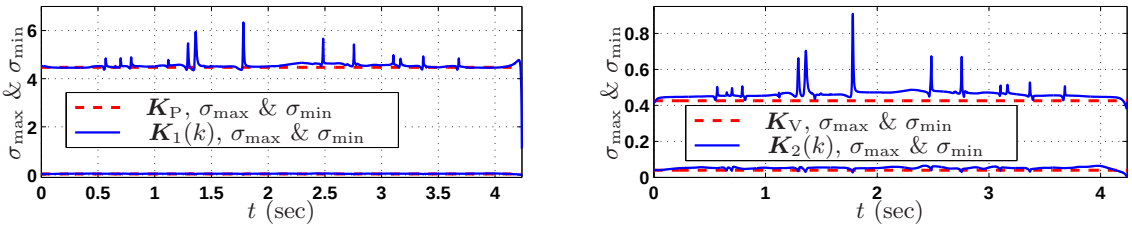


Figure 4.4: Maximum singular values of $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$ for all $k = 0, 1, \ldots, 4237$. For comparison purposes, the maximum singular values of the constant feedback matrices $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$ are also included.

of inconveniences can be anticipated from Fig. 4.4, namely, $\sigma_{\max}(\boldsymbol{K}_1(k))$ and $\sigma_{\max}(\boldsymbol{K}_2(k))$ exhibit sudden changes to large values at several time instants. The reason for this undesirable effect in the controller synthesis is related to linearization. Concretely, in trying to approximate the sign$(\cdot)$ function, with the smooth function satur$(\cdot)$ defined in (4.24), the parameter η should be chosen smaller. The reason for this is to make the approximation not as accurate since linearization requires the derivative of satur$(\dot{q})$ at several points where $\dot{q} = 0$. Changing the value of the parameter to $\eta = 5$, resolves this issue yielding results that are feasible for implementation on the actual robot.

Using $\eta = 5$ in approximation (4.24), yields the controller synthesis results shown in Fig. 4.5. In this case, the maximum gain amplifications $\sigma_{\max}(\boldsymbol{K}_1(k))$ and $\sigma_{\max}(\boldsymbol{K}_2(k))$ are more feasible for controller implementation on the actual robot. Note that $\sigma_{\max}(\boldsymbol{K}_1(k))$ and $\sigma_{\max}(\boldsymbol{K}_2(k)))$ slightly vary around $\sigma_{\max}(\boldsymbol{K}_{\mathrm{P}})$ and $\sigma_{\max}(\boldsymbol{K}_{\mathrm{V}})$, respectively. However, for all time k, the matrices $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$ should clearly be different from $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$, respectively. For instance, the feedback matrices corresponding to controller synthesis of Fig. 4.5 at time $k = 0$ are:

$$\boldsymbol{K}_1(0) = \begin{pmatrix} 4.2582 & 0.0031 & 0.0018 & 0.4185 & -0.0002 & 1.3887 \\ 0.0106 & 3.9774 & -1.1555 & -0.0030 & 0.8215 & 0.0024 \\ -0.0017 & 0.3436 & 0.7884 & 0.0020 & -0.1721 & 0.0011 \\ 0.0030 & 0.0015 & -0.0007 & 1.3923 & -0.0059 & -0.3530 \\ -0.0001 & -0.0099 & 0.0142 & -0.0005 & 0.0506 & 0.0003 \\ -0.0768 & -0.0000 & -0.0007 & 0.0378 & 0.0002 & 0.1758 \end{pmatrix},$$

$$\boldsymbol{K}_2(0) = \begin{pmatrix} 0.4008 & 0.0007 & 0.0001 & 0.0260 & 0.0000 & 0.1268 \\ 0.0014 & 0.3851 & -0.1045 & -0.0001 & 0.0780 & 0.0004 \\ -0.0002 & 0.0379 & 0.0765 & 0.0005 & -0.0142 & 0.0001 \\ -0.0065 & 0.0005 & -0.0003 & 0.1293 & -0.0003 & -0.0272 \\ 0.0000 & -0.0011 & 0.0023 & -0.0000 & 0.0040 & 0.0000 \\ -0.0087 & -0.0000 & -0.0001 & 0.0067 & 0.0000 & 0.0118 \end{pmatrix}.$$

Before presenting the experimental results, a couple of comments are made on the synthesis results from Fig. 4.5. The maximum gain amplification $\sigma_{\max}(\boldsymbol{K}_1(k))$ drops to a small value for the final transition of the trajectory. This is always the case, even when we substantially increase the weighting matrix $\boldsymbol{Q}_{f_1}$. Apparently, the optimal control problem formulation (4.15) of the trajectory tracking problem, implies that at time $k = H$ the control input

$\tilde{\boldsymbol{u}}_H$ should drop. This makes sense, since the objective function in (4.15) penalizes on the tracking error $\tilde{\boldsymbol{x}}_k$ up to $k = H$, which means that $\tilde{\boldsymbol{u}}_{H-1}$ is the last control input that can be utilized to affect $\tilde{\boldsymbol{x}}_H$.

4.4 Experimental Evaluations

We implement control law (4.22) to carry out experiments on the FANUC M16*i*B industrial robot. The compensation torque $\alpha_1(k)$ from Fig. 4.5 is too large for implementation on the actual robot. Therefore, only torques $\alpha_2(k), \alpha_3(k), \ldots, \alpha_6(k)$, are implemented on the experiments. In our experimental setup, additional 0.5 seconds are required after the end of the trajectory to allow for activation of mechanical breaks. Therefore, we switch to the feedback gains $\boldsymbol{K}_{\mathrm{P}}$, $\boldsymbol{K}_{\mathrm{V}}$, for $k = H+1, H+2, \ldots$. In any of the forthcoming plots regarding experiments, we always include what happens for $k = H+1, H+2, \ldots, H+10$, so that we can monitor the effect of switching to the feedback gains $\boldsymbol{K}_{\mathrm{P}}$, $\boldsymbol{K}_{\mathrm{V}}$, at the end of the trajectory.

The referred experimental results are all shown in Fig. 4.6. In Fig. 4.6(a) we present the applied torques $\boldsymbol{u}(t)^{\mathrm{ap}}$ and the accelerometer readings $\ddot{\boldsymbol{x}}(t)$. Note that the applied torques $\boldsymbol{u}(t)^{\mathrm{ap}}$ represent the total torques, computed with control law (4.22), composed of feedforward and feedback portions. Of interest to us in this Chapter are the resulting tracking errors, shown in Fig. 4.6(b). The joint-space motor-side tracking errors $\tilde{\boldsymbol{\theta}}(t)$ are compared against the corresponding tracking errors $\tilde{\boldsymbol{\theta}}(t)$ in Fig. 3.7(b) presented in Chapter 3. This is a fair comparison since from Fig. 4.5 the maximum singular values $\sigma_{\max}(\boldsymbol{K}_1(k))$ and $\sigma_{\max}(\boldsymbol{K}_2(k))$, $k = 0, 1, \ldots, H$, are of comparable magnitude to $\sigma_{\max}(\boldsymbol{K}_{\mathrm{P}})$ and $\sigma_{\max}(\boldsymbol{K}_{\mathrm{V}})$. The difference

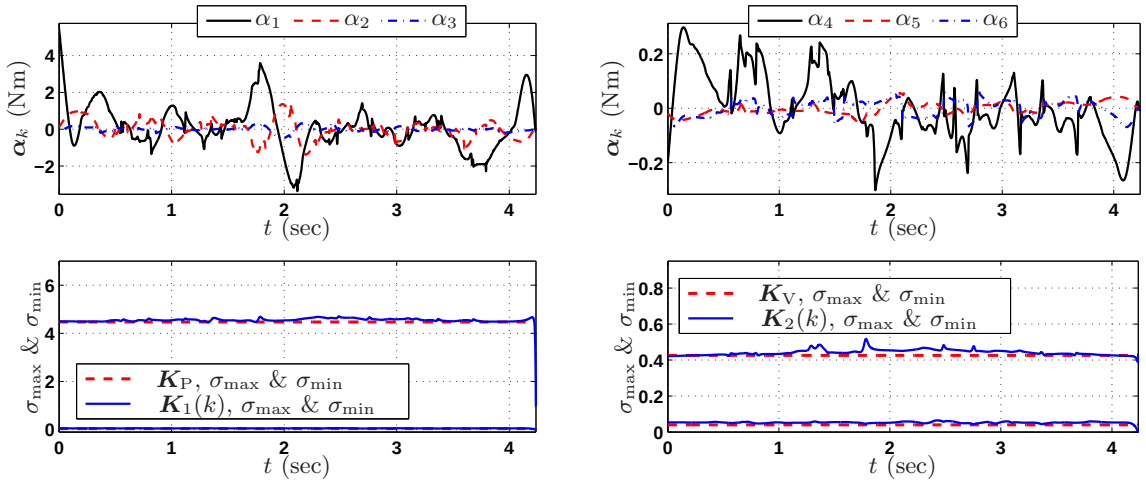
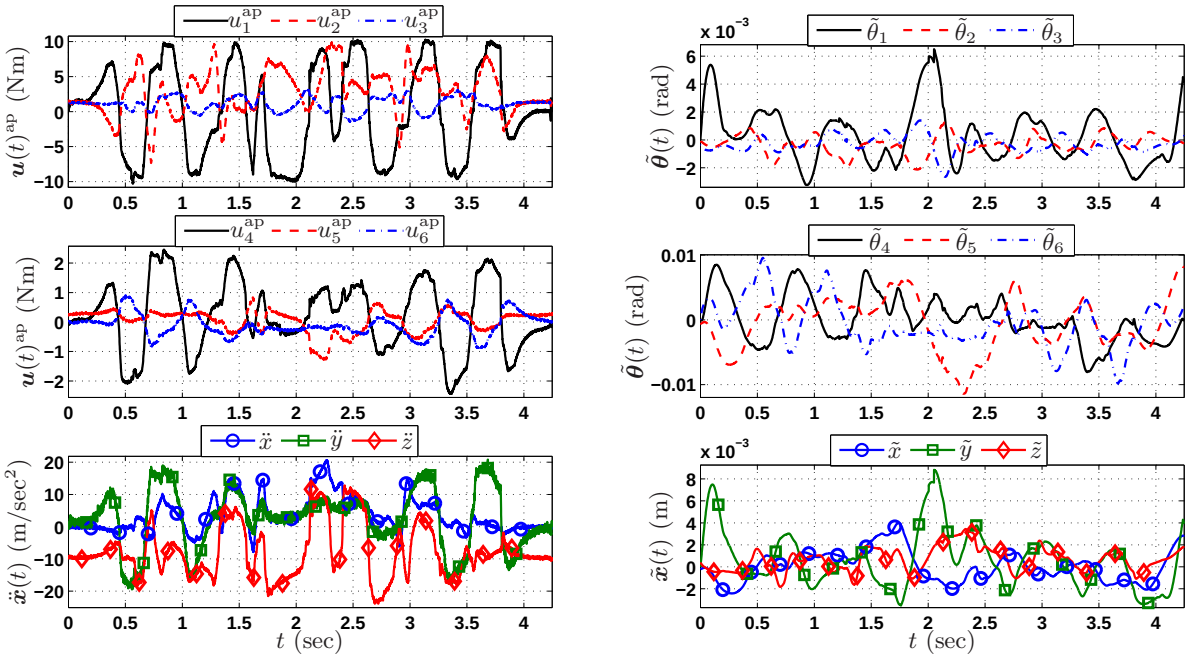


Figure 4.5: Controller synthesis results when choosing $\eta = 5$, so that the approximation to the sign$(\cdot)$ function is not as accurate, since for linearization purposes the derivative of satur$(\dot{q})$ is required at several points where $\dot{q} = 0$.



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors

(c) Feedback torques, $\boldsymbol{u}(t)^{\text{fb}}$

Figure 4.6: Experimental results when implementing the ATV control law (4.22). The reference trajectory corresponds to the near time-optimal trajectory presented in Chapter 3, (i.e., for $\lambda = 0.02$).

is that $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$ are time-varying, and have non-zero off-diagonal terms to account for the inherent coupled dynamics of robotic arms.

The root mean square (RMS) value for each of the joint tracking errors $\tilde{\theta}_1(k), \ldots, \tilde{\theta}_6(k)$, are presented in Table 4.1 for both the PID control law and the ATV control law (4.22). Also in Table 4.1 are presented the RMS values for the Cartesian-space tracking errors $\tilde{x}(k)$, $\tilde{y}(k)$, $\tilde{z}(k)$. For all axes, the RMS values associated to the ATV control law (4.22) are better than the corresponding values for PID control law (3.6). Finally, it is important to look at the feedback torques $\boldsymbol{u}(t)^{\text{fb}}$ shown in Fig. 4.6(c). Note that the ATV control law (4.22) is able to

	$\tilde{\theta}_1(k)$	$\tilde{\theta}_2(k)$	$\tilde{\theta}_3(k)$	$\tilde{\theta}_4(k)$	$\tilde{\theta}_5(k)$	$\tilde{\theta}_6(k)$	$\tilde{x}(k)$	$\tilde{y}(k)$	$\tilde{z}(k)$
PID	.0024	.0008	.0012	.0054	.0089	.0050	.0022	.0038	.0019
ATV	.0020	.0008	.0007	.0036	.0042	.0036	.0014	.0027	.0013

Table 4.1: Comparison of RMS values of the tracking errors achieved by the PID control law and the proposed ATV controller.

achieve better performance than control law (3.6) with smaller feedback torques (compare Figs. 4.6(c) and 3.7(c)). This might be attributed to the time-varying and multi-variable nature of control law (4.22). Likewise, the compensation torques $\boldsymbol{\alpha}_k$, which are feedforward, might be alleviating some burden on the feedback control portion.

Even though the ATV control law (4.22) features superior properties, the following comments should be pointed out:

- The methodology requires the storage of a large number of feedback matrices $\boldsymbol{K}_1(k)$, $\boldsymbol{K}_2(k) \in \mathbb{R}^{6\times 6}$, $k = 0, \ldots, H-1$. For instance, in the specific case of the trajectory from Fig. 4.1, for which the traversal time is 4.238 seconds with a sampling time $T_s = 1$ milliseconds, a total of 4237×2 matrices, $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$, are needed. Even though we were able to successfully implement the referred methodology in our experimental setup, for trajectories with longer traversal times (e.g., 12 seconds would require 24,000 matrices), the memory usage may become prohibitive.

- The LTV feedback gains are essentially time-varying because it is assumed that the system dynamics matrices $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$, vary with time, as a result of linearizing along the reference trajectory and torques. In other words, because of the time-varying feedback matrices $\boldsymbol{K}_1(k)$ and $\boldsymbol{K}_2(k)$, the ATV control law (4.22) can be regarded as a specific instance of gain scheduling controller for nonlinear systems, with time index k representing the scheduling variable [45, 23].

- The sudden drop of $\sigma_{\max}(\boldsymbol{K}_1(k))$ to almost zero at the end of the trajectory (see Fig. 4.5) is undesired, since we then need to switch to the feedback gains, $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$, for $k = H+1, H+2, \ldots$, in order to be able to control the robot state at that constant position. Switching from the almost zero $\sigma_{\max}(\boldsymbol{K}_1(H))$ to the nonzero $\sigma_{\max}(\boldsymbol{K}_{\mathrm{P}}(H+1))$ can result in sudden changes of the feedback torques $\boldsymbol{u}(k)^{\mathrm{fb}}$ (see 4.6(c)).

It seems intuitive that we might be able to attain similar performance with a gain scheduling scheme that requires the storage of significantly less feedback gain matrices. For example, if we let the actual state vector $[\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top]^\top$ be the scheduling variable, we could then choose a few strategic points on the trajectory where local controllers can be synthesized based on the local linear dynamics. To choose which controller to utilize, we would constantly monitor the actual robot state $[\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top]^\top$ and identify which controller to apply. In this manner, we do not rely on the robot state to have a specific value at an exact instant of time. In the

next chapter, we explore this idea, which builds on the afore discussed development that led to the ATV control law (4.22). As we shall see in Chapter 5, however, the resulting control law in that case will be piecewise affine (PWA) as opposed to affine and time-varying.

4.5 Summary

In this chapter we explored a controller synthesis methodology to achieve trajectory tracking on robotic arms. We developed a framework for trajectory tracking, based on the LQ optimal control problem for a class of affine time-varying (ATV) dynamical systems. Even though the dynamic model of robot manipulators is inherently nonlinear, we showed how the nonlinear dynamics can be approximated along the reference trajectory as an ATV dynamical system. Therefore a control law to achieve trajectory tracking was developed using LQ methods for linear time-varying (LTV) systems. When discretizing the continuous-time nonlinear model to obtain a discrete-time nonlinear model, the reference trajectory and torques are no longer exactly dynamically feasible, which motivated us to introduce a "disturbance" term in position and velocity. Even though the "disturbance" terms are rather small compared to the actual desired positions and velocities, our controller synthesis scheme takes these terms into account. The outcome is a time-varying trajectory tracking controller that "compensates" for the referred "disturbance" terms. The final control law was implemented in our experimental setup for the 6-axis industrial manipulator, which verified the feasibility (and also exposed the shortcomings) of the proposed method.

Chapter 5

Piecewise Affine Modeling and Control Synthesis for Trajectory Tracking

In this chapter we study the control synthesis for a class of piecewise affine systems to address the problem of trajectory tracking of robots, whose dynamic model is inherently nonlinear. Although the dynamic model of robot manipulators is highly nonlinear, we study that when the reference trajectory is known, it is possible to linearize the nonlinear dynamics along several operating points on the reference trajectory. Unlike the proposed method in Chapter 4, which led to the ATV controller and required linearization at every point on the reference trajectory, in the forthcoming development the linearization is only performed at specific points on the reference trajectory. The approximation will lead to a piecewise affine (PWA) dynamical system. If we assume that each affine dynamics is valid within an ellipsoidal region centered at a corresponding operating point, we can synthesize controllers for each ellipsoidal region based on the local dynamics. Each controller then guarantees that the corresponding equilibrium is asymptotically stable. Unlike our previous development on ATV controller in discrete-time, the controller synthesis for each ellipsoidal region will be done using the local continuous-time model. In fact, the forthcoming development is done entirely in continuous-time, but of course the implementation is carried out in discrete-time. Prior to implementing the proposed controller in experiments for the 6-axis industrial robot, throughout the chapter we illustrate our ideas on simulation studies for a 1-DOF and 2-DOF manipulators. Then, experimental results on the 6-axis industrial manipulator are presented to evaluate the effectiveness of the proposed control law.

5.1 Continuous-time Nonlinear Dynamic Model

Consider again the general nonlinear dynamic model for a robot manipulator with n degrees of freedom. Let the vector $\boldsymbol{\theta} \in \mathbb{R}^n$ represent the motor-side joint positions, and let $\boldsymbol{u} \in$

$\mathbb{R}^n$ represent the motor-side actuator torques. For the sake of clarity in the forthcoming development, the dynamic model (4.2) is restated here:

$$\begin{aligned}\left[\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}\right]\ddot{\boldsymbol{\theta}} + \left[\boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta},\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}})\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}) + \\ \left[\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\dot{\boldsymbol{\theta}}) = \boldsymbol{u},\end{aligned} \tag{5.1}$$

where $\boldsymbol{G}$ is a diagonal matrix containing the reducer's gear ratios, $\boldsymbol{M}(\cdot)$ is the positive-definite inertia matrix, $\boldsymbol{C}(\cdot,\cdot)$ is the Coriolis/centrifugal matrix, $\boldsymbol{g}(\cdot)$ is the vector of gravitational torques, and $\boldsymbol{D}_{\mathrm{v}}, \boldsymbol{F}_{\mathrm{C}} \in \mathbb{R}^{n\times n}$ representing respectively the coefficients of viscous and Coulomb damping. As already mentioned in Chapter 1, we always add the corresponding motor-side inertia $\boldsymbol{J}_{\mathrm{mot}}$ to the inertia matrix $\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}$, where $\boldsymbol{J}_{\mathrm{mot}}$ is diagonal and contains each motor's armature inertia. In other words, the total inertia matrix in (5.1) is: $\boldsymbol{J}_{\mathrm{mot}} + \boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta})\boldsymbol{G}^{-1}$. Similarly for the coefficients of viscous and Coulomb damping, the actual ones used in (5.1) are: $\boldsymbol{D}_{\mathrm{v,mot}} + \boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}$ and $\boldsymbol{F}_{\mathrm{C,mot}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}$, respectively. However, for simplicity of exposition, in this dissertation we have always presented the more compact dynamics in (5.1).

Assume that the manipulator is required to follow a motor-side joint-space reference trajectory $\boldsymbol{\theta}_{\mathrm{d}}(t)$ for all $t \in [0, t_f]$. A common technique used in industrial manipulators is to pre-compute a feedforward torque $\boldsymbol{u}_{\mathrm{d}}(t)$ from dynamic model (5.1), which is entirely based on the reference trajectory $\boldsymbol{\theta}_{\mathrm{d}}(t)$ [1]. This feedforward torque is then used to attempt to cancel out the nonlinearities in dynamic model (5.1). After the assumption of canceling out the nonlinearities through $\boldsymbol{u}_{\mathrm{d}}(t)$, a linear controller (e.g., a PI plus lead compensator) is designed under the assumption that a linear plant results due to the feedforward torque. In this chapter, we follow a similar philosophy, but we do not assume a simple linear model for the entire trajectory. Rather, we propose that the problem can be formulated in the context of controller synthesis for a class of piecewise affine systems, by selecting strategic points along the reference trajectory and approximating the nonlinear dynamics near those points as affine dynamical systems.

Given the reference trajectory, first and second time derivatives $(\boldsymbol{\theta}_{\mathrm{d}}(t), \dot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \ddot{\boldsymbol{\theta}}_{\mathrm{d}}(t))$, the referred feedforward torque $\boldsymbol{u}_{\mathrm{d}}(t)$ is simply:

$$\begin{aligned}\boldsymbol{u}_{\mathrm{d}}(t) = \left[\boldsymbol{G}^{-1}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{\theta}_{\mathrm{d}})\boldsymbol{G}^{-1}\right]\ddot{\boldsymbol{\theta}}_{\mathrm{d}} + \left[\boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{\theta}_{\mathrm{d}},\boldsymbol{G}^{-1}\dot{\boldsymbol{\theta}}_{\mathrm{d}})\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}}_{\mathrm{d}} + \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{\theta}_{\mathrm{d}}) \\ + \left[\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\right]\dot{\boldsymbol{\theta}}_{\mathrm{d}} + \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{satur}(\dot{\boldsymbol{\theta}}_{\mathrm{d}}).\end{aligned} \tag{5.2}$$

We refer to the trajectory 4-tuple $(\boldsymbol{\theta}_{\mathrm{d}}(t), \dot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \ddot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$ as a dynamically feasible trajectory with respect to (5.1). Notice that the near time-optimal 4-tuple trajectory $(\boldsymbol{\theta}^{\star}(t), \dot{\boldsymbol{\theta}}^{\star}(t), \ddot{\boldsymbol{\theta}}^{\star}(t), \boldsymbol{u}^{\star}(t))$ generated from algorithm (3.13) in Chapter 3, is indeed dynamically feasible with respect to dynamics (5.1). It should also be emphasized that in this chapter, the continuous-time model (5.1) will not be discretized to obtain a nonlinear discrete-time model. In this manner, any near time-optimal trajectory $(\boldsymbol{\theta}^{\star}(t), \dot{\boldsymbol{\theta}}^{\star}(t), \ddot{\boldsymbol{\theta}}^{\star}(t), \boldsymbol{u}^{\star}(t))$ generated from algorithm (3.13) will remain dynamically feasible.

The nonlinear dynamic model (5.1) should be expressed in state-space form. Let us define the state vector $\boldsymbol{x} = (\boldsymbol{x}_1^\top \ \boldsymbol{x}_2^\top)^\top := (\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top)^\top \in \mathbb{R}^{2n}$, then the nonlinear dynamic model (5.1) is compactly written:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{5.3}$$

where the map $\boldsymbol{f} : \mathbb{R}^{2n} \times \mathbb{R}^n \mapsto \mathbb{R}^n$ is defined as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{G}\boldsymbol{M}(\boldsymbol{G}^{-1}\boldsymbol{x}_1)^{-1}\boldsymbol{G} \left\{ \begin{array}{r} \boldsymbol{u} - \boldsymbol{G}^{-1}\boldsymbol{C}(\boldsymbol{G}^{-1}\boldsymbol{x}_1, \boldsymbol{G}^{-1}\boldsymbol{x}_2)\boldsymbol{G}^{-1}\boldsymbol{x}_2 - \boldsymbol{G}^{-1}\boldsymbol{g}(\boldsymbol{G}^{-1}\boldsymbol{x}_1) + \\ -\boldsymbol{G}^{-1}\boldsymbol{D}_{\mathrm{v}}\boldsymbol{G}^{-1}\boldsymbol{x}_2 - \boldsymbol{G}^{-1}\boldsymbol{F}_{\mathrm{C}}\,\mathrm{sign}(\boldsymbol{x}_2) \end{array} \right\} \end{bmatrix}. \tag{5.4}$$

In this context, a dynamically feasible pair $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$ with respect to (5.3)-(5.4) means $\dot{\boldsymbol{x}}_{\mathrm{d}} = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{d}}, \boldsymbol{u}_{\mathrm{d}})$ for all $t \in [0, t_f]$. Clearly dynamic feasibility with respect to (5.1) implies dynamic feasibility with respect to (5.3)-(5.4).

5.1.1 Linearization along the Reference Trajectory

To linearize (5.3)-(5.4) along the continuous-time trajectory pair $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$, we obtain the first-order Taylor expansion of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ along $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$ for all $t \in [0, t_f]$, giving:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \approx \ & \boldsymbol{f}(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t)) \\ & + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{x}_{\mathrm{d}}(t) \\ \boldsymbol{u} = \boldsymbol{u}_{\mathrm{d}}(t)}} (\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{d}}(t)) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{x}_{\mathrm{d}}(t) \\ \boldsymbol{u} = \boldsymbol{u}_{\mathrm{d}}(t)}} (\boldsymbol{u}(t) - \boldsymbol{u}_{\mathrm{d}}(t)). \end{aligned}$$

Therefore, by defining the following time-varying matrices and vector:

$$\begin{aligned} \boldsymbol{A}(t) &:= \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{x}_{\mathrm{d}}(t) \\ \boldsymbol{u} = \boldsymbol{u}_{\mathrm{d}}(t)}} \in \mathbb{R}^{2n \times 2n}, \quad \boldsymbol{B}(t) := \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x} = \boldsymbol{x}_{\mathrm{d}}(t) \\ \boldsymbol{u} = \boldsymbol{u}_{\mathrm{d}}(t)}} \in \mathbb{R}^{2n \times n}, \\ \boldsymbol{b}(t) &:= \boldsymbol{f}(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t)) - \boldsymbol{A}(t)\boldsymbol{x}_{\mathrm{d}}(t) - \boldsymbol{B}(t)\boldsymbol{u}_{\mathrm{d}}(t), \end{aligned} \tag{5.5}$$

leads to approximate (5.3)-(5.4) along $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$, as the following affine time-varying dynamical system:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x} + \boldsymbol{b}(t) + \boldsymbol{B}(t)\boldsymbol{u}, \quad \forall t \in [0, t_f]. \tag{5.6}$$

5.2 Piecewise Affine Modeling

As a result of assuming that the actual state of the robot $\boldsymbol{x}(t)$ will pass through $\boldsymbol{x}_{\mathrm{d}}(t)$ exactly at time t, approximation (5.6) contains time-varying matrices $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, and $\boldsymbol{b}(t)$. This approximation is closely related to the one presented in (4.11)-(4.12) from Chapter 4, which eventually led to the ATV control law (4.22). However, an important difference is that the

non-linear map used in this chapter to define $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, $\boldsymbol{b}(t)$, is the continuous-time vector field $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, as opposed to the discrete-time nonlinear map defined in (4.6).

One reason not to discretize the continuous-time nonlinear dynamics in this chapter, is to preserve the exact dynamic feasibility of any optimal trajectory $(\boldsymbol{\theta}_{\mathrm{d}}(t), \dot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \ddot{\boldsymbol{\theta}}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$, generated through algorithm (3.13). As already seen, when discretizing the continuous-time nonlinear dynamics, the optimal trajectory is no longer exactly dynamically feasible with respect to the discrete-time dynamics. This observation led us to introduce the concept of a "disturbance" term due to the non-exact dynamic feasibility of the trajectory, namely, $\boldsymbol{v}_k$, $\boldsymbol{w}_k$ defined in (4.12). In the present chapter, there will be no need to introduce this "disturbance" term, since the developments are carried out using the continuous-time dynamics.

We argued in Chapter 4 that having a control law whose feedback gains depend on the actual state of the robot $\boldsymbol{x}(t)$ (i.e., state-dependent gain scheduling), might be more convenient than a time-varying control law (i.e., time-dependent gain scheduling). To this end, instead of approximating the nonlinear dynamics (5.3)-(5.4) by the affine time-varying dynamics (5.6), we propose a piecewise affine (PWA) dynamics, which means that the system matrices depend actually on the state $\boldsymbol{x}(t)$ [46]. This allows to define operating regions in the spate space whereby the corresponding approximation is valid.

Consider $\left(\boldsymbol{A}_{\alpha(\boldsymbol{x})}, \boldsymbol{b}_{\alpha(\boldsymbol{x})}, \boldsymbol{B}_{\alpha(\boldsymbol{x})}\right)$, with the state-dependent index $\alpha : \mathbb{R}^{2n} \mapsto \{1, \ldots, L\}$, where L represents the number of operating regions (or discrete modes) carefully selected along the reference trajectory. Each operating region $\mathcal{R}_i \subset \mathbb{R}^{2n}$ satisfies $\mathcal{R}_i = \{\boldsymbol{x} \in \mathbb{R}^{2n} : \alpha(\boldsymbol{x}) = i\}$. We therefore propose to approximate the nonlinear dynamics (5.3)-(5.4) along the reference trajectory, by the following piecewise affine (PWA) dynamical system:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i + \boldsymbol{B}_i \boldsymbol{u}, \quad \text{if } \boldsymbol{x} \in \mathcal{R}_i, \quad i \in \{1, 2, \ldots, L\}. \tag{5.7}$$

The question on how to choose L, how to construct the system matrices $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$, and the corresponding operating regions $\mathcal{R}_i$, $i = 1, \ldots, L$, is addressed next.

5.2.1 Constructing $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$ and $\mathcal{R}_i$

As already pointed out, instead of considering all points in the entire reference trajectory $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$ $\forall t \in [0, t_f]$, we only choose L operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$ along the reference pair $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t))$. On the other hand, the operating regions are denoted $\mathcal{E}_i$ and are constructed as ellipsoids centered at the corresponding operating point $\boldsymbol{x}_{\mathrm{c}}^{(i)}$, $i = 1, \ldots, L$. An abstract idea of the kind of construction that we want to attain is illustrated in Fig. 5.1, where the state-space reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$, operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$, and ellipsoidal regions $\mathcal{E}_1, \ldots, \mathcal{E}_L$ are included.

Essentially, to achieve such a construction, we need to first devise a method to choose the operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$. Once these operating points are known, constructing the corresponding ellipsoids $\mathcal{E}_1, \ldots, \mathcal{E}_L$ should be straightforward, since we know already their center, namely, $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$. In this dissertation, we use three equivalent parametriza-

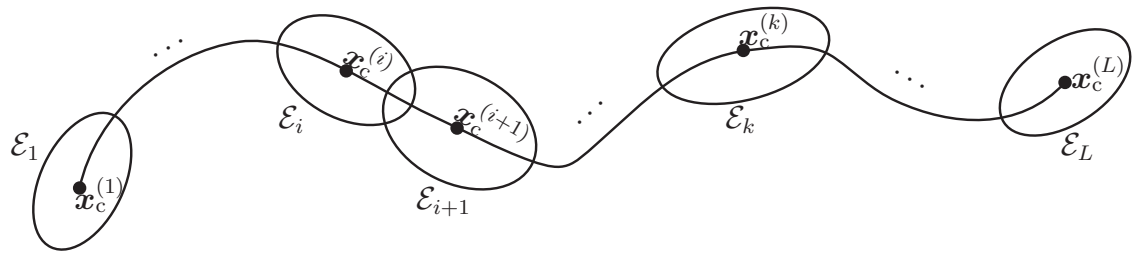


Figure 5.1: Illustration of reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$, operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$, and ellipsoidal regions $\mathcal{E}_1, \ldots, \mathcal{E}_L$.

tions of an ellipsoid $\mathcal{E}_i \subset \mathbb{R}^{2n}$ [46, 47]:

$$\begin{aligned}
\mathcal{E}_i &= \left\{\boldsymbol{x} \in \mathbb{R}^{2n} : (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}^{(i)})^\top \boldsymbol{R}_i^{-1} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}^{(i)}) \leq 1\right\} \\
\mathcal{E}_i &= \left\{\boldsymbol{x}_{\mathrm{c}}^{(i)} + \boldsymbol{L}_i \boldsymbol{z} : \|\boldsymbol{z}\|_2 \leq 1\right\} \\
\mathcal{E}_i &= \left\{\boldsymbol{x} \in \mathbb{R}^{2n} : \|\boldsymbol{S}_i \boldsymbol{x} + \boldsymbol{s}_i\|_2 \leq 1\right\}
\end{aligned} \tag{5.8}$$

where $\boldsymbol{R}_i \in \mathbb{R}^{2n \times 2n}$, $\boldsymbol{R}_i = \boldsymbol{R}_i^\top \succ 0$; $\boldsymbol{L}_i \in \mathbb{R}^{2n \times 2n}$ is a nonsingular matrix; $\boldsymbol{S}_i \in \mathbb{R}^{2n \times 2n}$, $\boldsymbol{S}_i = \boldsymbol{S}_i^\top \succ 0$, $\boldsymbol{s}_i \in \mathbb{R}^{2n}$. It is easy to show that these three parametrizations are equivalent, i.e., given one parametrization it is possible to recover the other two. For instance, given $\boldsymbol{R}_i$ and $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ in the first parametrization, the other two parametrizations of $\mathcal{E}_i$ are simply given by $\boldsymbol{L}_i = \boldsymbol{R}_i^{1/2}$, and $\boldsymbol{S}_i = \boldsymbol{R}_i^{-1/2}$, $\boldsymbol{s}_i = -\boldsymbol{R}_i^{-1/2} \boldsymbol{x}_{\mathrm{c}}^{(i)}$.

To parameterize the ellipsoids $\mathcal{E}_i$ with known center $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ of Fig. 5.1, the first parametrization is more convenient:

$$\mathcal{E}_i = \left\{\boldsymbol{x} \in \mathbb{R}^{2n} : (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}^{(i)})^\top \boldsymbol{R}_i^{-1} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}^{(i)}) \leq 1\right\}, \tag{5.9}$$

from which it is clear that constructing the ellipsoid $\mathcal{E}_i$ with known center $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ is equivalent to finding the parameterizing positive-definite matrix $\boldsymbol{R}_i$. We shall explain how to obtain $\boldsymbol{R}_i$ shortly, but for now we concentrate on explaining the selection of the operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$.

Procedure to Choose the Operating Points

The following simple algorithm is proposed and implemented to generate the set of operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$:

1. Starting from the first point of the reference trajectory $(\boldsymbol{x}_{\mathrm{d}}(0), \boldsymbol{u}_{\mathrm{d}}(0))$, linearize the nonlinear dynamics (5.3)-(5.4) at $(\boldsymbol{x}_{\mathrm{d}}(0), \boldsymbol{u}_{\mathrm{d}}(0))$. This results in system matrices $\boldsymbol{A}(0)$ and $\boldsymbol{B}(0)$. Define the current operating point as $\boldsymbol{x}_{\mathrm{current}} = \boldsymbol{x}_{\mathrm{d}}(0)$.

2. Define system matrix $\bar{\boldsymbol{A}}_{\mathrm{current}} := [\boldsymbol{A}(0) \;\; \boldsymbol{B}(0)]$.

3. Move forward along the reference trajectory to the next point, say $(\boldsymbol{x}_{\rm d}(t_j), \boldsymbol{u}_{\rm d}(t_j))$, and linearize the nonlinear dynamics (5.3)-(5.4). Resulting in system matrices $\boldsymbol{A}(t_j)$ and $\boldsymbol{B}(t_j)$.

4. Define system matrix $\bar{\boldsymbol{A}}_{\rm candidate} := [\boldsymbol{A}(t_j) \;\; \boldsymbol{B}(t_j)]$.

5. Compare the systems $\bar{\boldsymbol{A}}_{\rm current}$ and $\bar{\boldsymbol{A}}_{\rm candidate}$.

6. If $\bar{\boldsymbol{A}}_{\rm current}$ and $\bar{\boldsymbol{A}}_{\rm candidate}$ are not sufficiently different, keep moving forward along $(\boldsymbol{x}_{\rm d}(t), \boldsymbol{u}_{\rm d}(t))$ and keep redefining $\bar{\boldsymbol{A}}_{\rm candidate}$ until it is sufficiently different from $\bar{\boldsymbol{A}}_{\rm current}$. We use the following criterion of 1% relative error:

$$\frac{\|\bar{\boldsymbol{A}}_{\rm current} - \bar{\boldsymbol{A}}_{\rm candidate}\|_2}{\|\bar{\boldsymbol{A}}_{\rm current}\|_2} \geq 0.01, \tag{5.10}$$

   where $\|\cdot\|_2$ is the maximum-singular-value matrix norm.

7. When this criterion is satisfied, choose the new operating point $\boldsymbol{x}_{\rm current} := \boldsymbol{x}_{\rm d}(t_j)$ as the starting operating point, define $\bar{\boldsymbol{A}}_{\rm current} := \bar{\boldsymbol{A}}_{\rm candidate}$, and go back to step 3.

8. Continue iterating until reaching the end of the trajectory.

When the end of the reference trajectory is reached, this algorithm outputs the set of operating points $\{\boldsymbol{x}_{\rm c}^{(1)}, \ldots, \boldsymbol{x}_{\rm c}^{(L)}\}$ with their corresponding system matrices $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$, $i = 1, \ldots, L$.

Obtaining $\boldsymbol{R}_i$ to Parameterize $\mathcal{E}_i$

As already mentioned, for the construction of $\mathcal{E}_i$, we use the first parametrization in (5.8) since the center $\boldsymbol{x}_{\rm c}^{(i)}$ is already known. Therefore, the task here is simply: obtain the matrices $\boldsymbol{R}_i \in \mathbb{R}^{2n\times 2n}$, that parameterize $\mathcal{E}_i$, $i = 1, \ldots, L$. Essentially, we propose a simple method to achieve this task, based on the eigenvalue decomposition (EVD) theorem of symmetric matrices [48].[1] To generate ellipsoids with the shape and orientations depicted in Fig. 5.1, it is intuitive to require that the first semi-axis of $\mathcal{E}_i$ be in the direction of $2n$-dimensional vector $\boldsymbol{x}_{\rm c}^{(i+1)} - \boldsymbol{x}_{\rm c}^{(i)}$. The remaining $2n-1$ semi-axes should then be in orthogonal directions.

Let us denote the eigenvalues of $\boldsymbol{R}_i$ as $\lambda_1^{(i)}, \ldots, \lambda_{2n}^{(i)}$, with corresponding eigenvectors $\boldsymbol{v}_1^{(i)}, \ldots, \boldsymbol{v}_{2n}^{(i)} \in \mathbb{R}^{2n}$. The above requirement, on the semi-axes of $\mathcal{E}_i$, is equivalent to require that eigenvector $\boldsymbol{v}_1^{(i)}$ be in the direction of $\boldsymbol{x}_{\rm c}^{(i+1)} - \boldsymbol{x}_{\rm c}^{(i)}$. The corresponding eigenvalue $\lambda_1^{(i)}$ should be chosen so that

$$\sqrt{\lambda_1^{(i)}} = \beta \|\boldsymbol{x}_{\rm c}^{(i+1)} - \boldsymbol{x}_{\rm c}^{(i)}\|_2, \tag{5.11}$$

[1] We can decompose any symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{2n\times 2n}$ with the symmetric eigenvalue decomposition

$$\boldsymbol{A} = \sum_{i=1}^{2n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top, \quad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_{2n}),$$

where the matrix $\boldsymbol{U} = [\boldsymbol{u}_1 \;\cdots\; \boldsymbol{u}_{2n}]$ is orthogonal (that is, $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}_{2n}$).

with $\beta \in (0.5, 1]$ an adjustable parameter; generally we choose $\beta = 1$. The remaining $2n-1$ eigenvalues are set so that

$$\sqrt{\lambda_2^{(i)}} = \sqrt{\lambda_3^{(i)}} = \cdots = \sqrt{\lambda_{2n}^{(i)}} = \frac{1}{\gamma} \max_{i \in \{1,\ldots,L\}} \sqrt{\lambda_1^{(i)}}, \tag{5.12}$$

i.e., a fraction of the maximum first semi-axis among all the ellipsoids $\mathcal{E}_i$, $i = 1, \ldots, L$; generally $\gamma \in \{5, \ldots, 10\}$. Having defined the eigenvalues of $\boldsymbol{R}_i$, it is convenient to define $\boldsymbol{\Lambda}_i := \text{diag}(\lambda_1^{(i)}, \lambda_2^{(i)}, \ldots, \lambda_{2n}^{(i)})$.

Since the first eigenvector $\boldsymbol{v}_1^{(i)}$ is required to be in the direction of $\boldsymbol{x}_\text{c}^{(i+1)} - \boldsymbol{x}_\text{c}^{(i)}$, and the remaining $2n-1$ eigenvectors $\boldsymbol{v}_2^{(i)}, \ldots, \boldsymbol{v}_{2n}^{(i)}$ being in orthogonal directions, we simply perform a modified Gram-Schmidt procedure to the following set of $2n+1$ vectors [49]:

$$\left\{ \boldsymbol{x}_\text{c}^{(i+1)} - \boldsymbol{x}_\text{c}^{(i)}, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{2n} \right\},$$

where the set of vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{2n}\}$ is the canonical basis for $\mathbb{R}^{2n}$. Therefore, the output of this procedure is the orthonormal set of $2n$ vectors $\{\boldsymbol{v}_1^{(i)}, \boldsymbol{v}_2^{(i)}, \ldots, \boldsymbol{v}_{2n}^{(i)}\}$, with $\boldsymbol{v}_1^{(i)}$ in the direction of $\boldsymbol{x}_\text{c}^{(i+1)} - \boldsymbol{x}_\text{c}^{(i)}$. It is then convenient to construct the following orthogonal matrices $\boldsymbol{U}_i := [\boldsymbol{v}_1^{(i)} \ \cdots \ \boldsymbol{v}_{2n}^{(i)}] \in \mathbb{R}^{2n \times 2n}$, $i = 1, 2, \ldots, L$.

From the eigenvalue decomposition theorem of symmetric matrices, we thus propose to build matrices $\boldsymbol{R}_i$ simply as:

$$\boldsymbol{R}_i := \sum_{j=1}^{2n} \lambda_j^{(i)} \boldsymbol{v}_j^{(i)} \boldsymbol{v}_j^{(i)\top} = \boldsymbol{U}_i \boldsymbol{\Lambda}_i \boldsymbol{U}_i^\top, \quad i = 1, 2, \ldots, L, \tag{5.13}$$

with $\boldsymbol{U}_i = [\boldsymbol{v}_1^{(i)} \ \cdots \ \boldsymbol{v}_{2n}^{(i)}]$, $\boldsymbol{\Lambda}_i = \text{diag}(\lambda_1^{(i)}, \ldots, \lambda_{2n}^{(i)})$, $i = 1, 2, \ldots, L$, constructed with the above development.

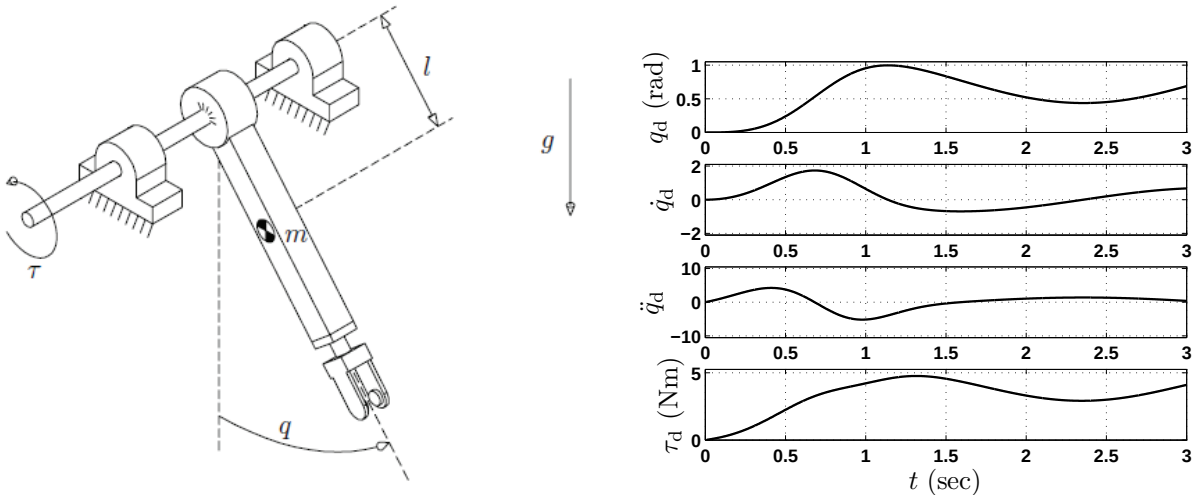
5.3 Case Studies on the Simplest Manipulators

In order to illustrate the ideas developed in Section 5.2, two case studies are presented: a 1-DOF and 2-DOF planar manipulators, which are meant to study the feasibility of the aforementioned ideas before applying them to the 6-DOF industrial manipulator. In these two toy examples, it is assumed for simplicity that the gear ratios are equal to one, i.e., $\boldsymbol{G} = \boldsymbol{I}$, and therefore $\boldsymbol{\theta} = \boldsymbol{q}$, $\boldsymbol{u} = \boldsymbol{\tau}$. It is also assumed that Coulomb friction is zero.

5.3.1 1-DOF Manipulator

We start off with the simplest robot manipulator, a 1-DOF manipulator shown in Fig. 5.2(a). The nonlinear dynamic model for this system is simply given by:

$$J\ddot{q} + mgl \sin(q) + d\dot{q} = \tau(t), \tag{5.14}$$



(a) A 1-DOF simple manipulator

(b) Reference trajectory $(q_{\rm d}(t), \dot{q}_{\rm d}(t), \ddot{q}_{\rm d}(t), \tau_{\rm d}(t))$

Figure 5.2: First toy example, a 1-DOF planar manipulator and its reference trajectory.

where the parameters $J = 0.1843$ kg $\cdot$ m^2, $m = 6.5225$ kg, $l = 0.2600$ m, $d = 0.05$ Nm $\cdot$ s. The reference trajectory $q_{\rm d}(t)$ is created at a 1-millisecond sampling rate using the following primitive:

$$q_{\rm d}(t) = b_1(1 - \exp(-2t^3)) + c_1(1 - \exp(-2t^3))\sin(\omega_1 t),$$

with parameters $b_1 = \pi/4$, $c_1 = \pi/9$, $\omega_1 = 2$ rad/s, for $t \in [0, 3]$ seconds. The complete trajectory including $\dot{q}_{\rm d}(t)$, $\ddot{q}_{\rm d}(t)$, and the corresponding feedforward torques $\tau_{\rm d}(t)$ are shown in Fig. 5.2(b).

We apply the algorithms presented in Section 5.2 to generate the operating points $\{\boldsymbol{x}_{\rm c}^{(1)}, \ldots, \boldsymbol{x}_{\rm c}^{(L)}\}$, corresponding system matrices $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$, and ellipsoids-parameterizing matrices $\boldsymbol{R}_i$, $i = 1, \ldots, L$. The parameter γ in (5.12) is set to $\gamma = 8$. The number of operating points L that the algorithm generates is $L = 121$. The reference trajectory $(q_{\rm d}(t), \dot{q}_{\rm d}(t))$ on the phase-plane $(q, \dot{q})$, together with the operating points $\{\boldsymbol{x}_{\rm c}^{(1)}, \ldots, \boldsymbol{x}_{\rm c}^{(L)}\}$ and the corresponding ellipses $\mathcal{E}_i$, are shown in Fig. 5.3.

5.3.2 2-DOF Planar Manipulator

We move on to a more complicated manipulator, the 2-DOF robotic arm shown in Fig. 5.4. The nonlinear dynamic model has the general form (5.1), with no Coulomb friction, $\boldsymbol{G} = \boldsymbol{I}$ entailing $\boldsymbol{\theta} = \boldsymbol{q}$ and $\boldsymbol{u} = \boldsymbol{\tau}$. Matrices $\boldsymbol{M}(\boldsymbol{q}), \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{2\times 2}$ and vector $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^2$ are readily available from standard robotics textbooks [15, 1].

The parameters used here are realistic, which have been taken from [15]: for the first link, $I_1 = 0.1213$ kg $\cdot$ m^2, $m_1 = 6.5225$ kg, $l_1 = 0.26$ m, $l_{\rm c1} = 0.0983$ m, $d_1 = 0.05$ Nm $\cdot$ s; for the second link, $I_2 = 0.0116$ kg $\cdot$ m^2, $m_2 = 2.0458$ kg, $l_2 = 0.26$ m, $l_{\rm c2} = 0.0229$ m,

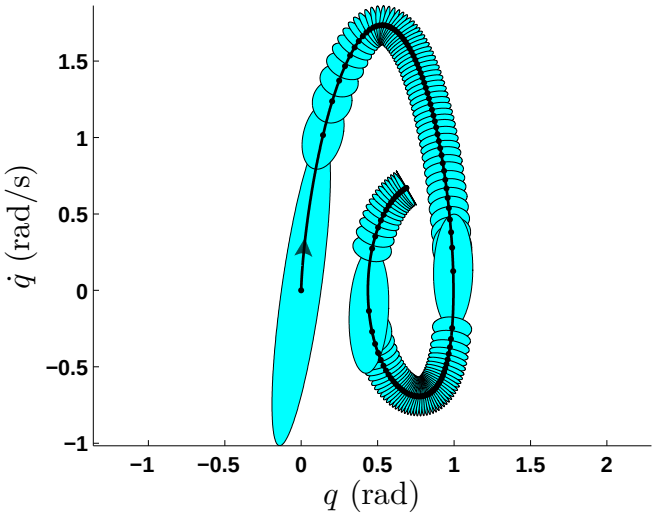


Figure 5.3: Reference trajectory, resulting operating points (marked with '·'), and resulting ellipsoids for the 1-DOF manipulator.

$d_2 = 0.01$ Nm · s. The reference trajectory $\boldsymbol{q}_{\mathrm{d}}(t) \in \mathbb{R}^2$ is generated at a 1-millisecond sampling rate for $t \in [0, 3]$ seconds. The following primitives are used:

$$\begin{aligned} q_{\mathrm{d}1}(t) &= b_1(1 - \exp(-2t^3)) + c_1(1 - \exp(-2t^3))\sin(\omega_1 t) \\ q_{\mathrm{d}2}(t) &= b_2(1 - \exp(-2t^3)) + c_2(1 - \exp(-2t^3))\sin(\omega_2 t), \end{aligned}$$

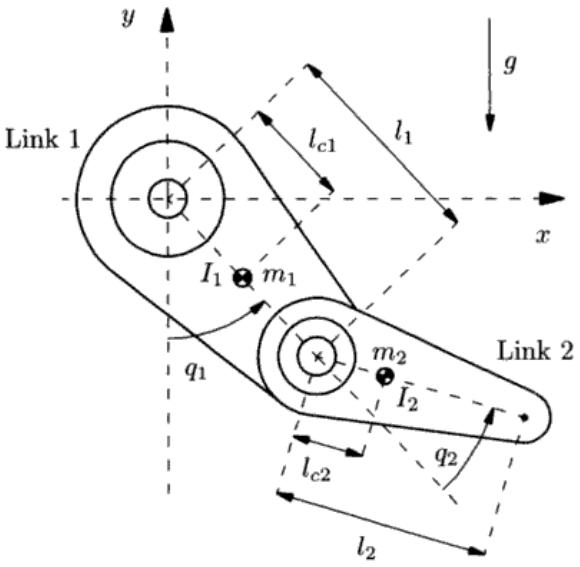


Figure 5.4: A 2-DOF planar manipulator used as a toy example.

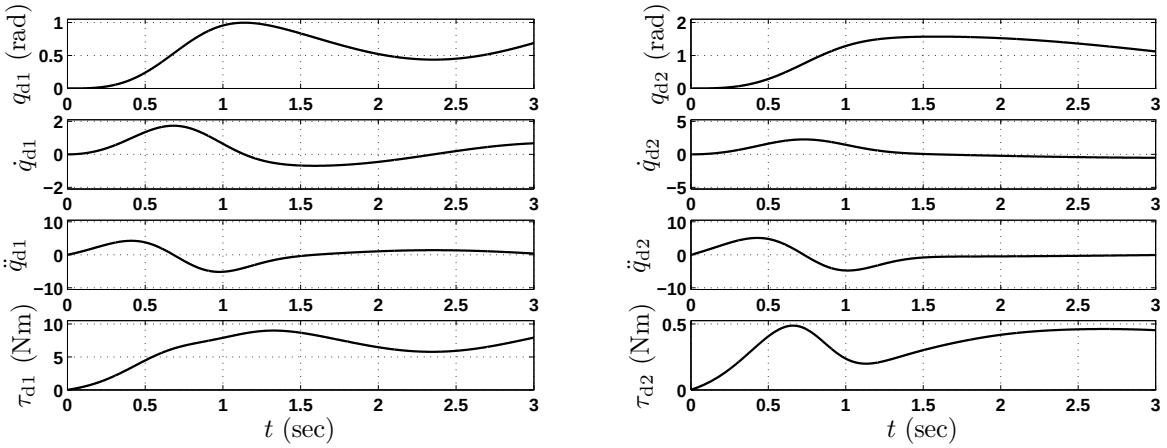


Figure 5.5: Reference trajectory $(\boldsymbol{q}_{\rm d}(t), \dot{\boldsymbol{q}}_{\rm d}(t), \ddot{\boldsymbol{q}}_{\rm d}(t), \boldsymbol{\tau}_{\rm d}(t))$ for the 2-DOF planar manipulator.

with parameters $b_1 = \pi/4$, $c_1 = \pi/9$, $\omega_1 = 2$ rad/s, $b_2 = \pi/3$, $c_2 = \pi/6$, $\omega_2 = 1$ rad/s. The reference trajectory including the feedforward torques $\boldsymbol{\tau}_{\rm d}(t)$ are shown in Fig. 5.5.

In this case, the number of operating points L turns out $L = 96$. Since the state vector $\boldsymbol{x} = (\boldsymbol{q}^\top \ \dot{\boldsymbol{q}}^\top)^\top \in \mathbb{R}^4$, we cannot draw the 96 ellipsoids in this case. We simply draw the reference state-trajectory in the $(q_1, \dot{q}_1)$ and $(q_2, \dot{q}_2)$ planes, with the resulting operating points $\boldsymbol{x}_{\rm c}^{(1)}, \ldots, \boldsymbol{x}_{\rm c}^{(96)}$ (marked with '$\cdot$'), as shown in Fig. 5.6.

The two toy examples presented in this section, represent positive indication that our

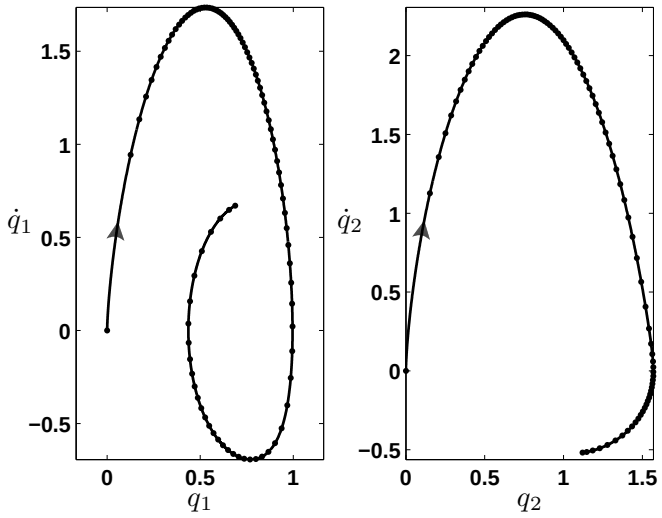


Figure 5.6: Reference trajectory and corresponding operating points $\boldsymbol{x}_{\rm c}^{(1)}, \ldots, \boldsymbol{x}_{\rm c}^{(L)}$, marked with '$\cdot$', for the 2-DOF planar manipulator, in which case $L = 96$.

algorithms to generate the operating points $\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}$ and ellipsoidal regions $\mathcal{E}_1, \ldots, \mathcal{E}_L$, are feasible to apply on the 6-DOF industrial manipulator. In the 6-DOF case, the state vector $\boldsymbol{x} \in \mathbb{R}^{12}$, and therefore the ellipsoids $\mathcal{E}_i \subset \mathbb{R}^{12}$. We defer the exposition of results for the 6-DOF case, and present them later in this chapter, after the controller synthesis strategy for each ellipsoidal region has been discussed.

5.4 Controller Synthesis

In Section 5.2, we have proposed a modeling framework, in which the problem of trajectory tracking for robot manipulators is approached by casting robot modeling into the context of a class of piecewise affine (PWA) dynamical systems. Specifically, assuming that the actual state of the robot $\boldsymbol{x} = (\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top)^\top \in \mathbb{R}^{2n}$ remains "near" the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t) \in \mathbb{R}^{2n}$, the nonlinear robot dynamics (5.3)-(5.4) can be reasonably well approximated by the following piecewise affine dynamical system:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i + \boldsymbol{B}_i \boldsymbol{u}, \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}, \tag{5.15}$$

where $\mathcal{E}_i$ is an ellipsoidal region centered at an operating point $\boldsymbol{x}_{\mathrm{c}}^{(i)}$, $i = 1, \ldots, L$. The operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$ are selected points on the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$. The assumption that the actual state $\boldsymbol{x} = (\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top)^\top$ remains "near" the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$ $\forall t \in [0, t_f]$, is achieved by pre-computing the feedforward torque $\boldsymbol{u}_{\mathrm{d}}(t)$ from (5.2).

Therefore, the next question to ask is how to design a piecewise affine state-feedback control law of the form:

$$\boldsymbol{u}(\boldsymbol{x}) = -\boldsymbol{K}_i \boldsymbol{x} + \boldsymbol{k}_i, \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}, \tag{5.16}$$

that guarantees asymptotic stability under switching between operating regions $\mathcal{E}_i$, when closing the loop on the PWA system (5.15).

5.4.1 Closed-loop Dynamics

The closed-loop dynamics of PWA system (5.15) under the PWA control law (5.16) features L different equilibria. These equilibria can be conveniently placed at $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ and made asymptotically stable, by properly synthesizing $\boldsymbol{K}_i$, $\boldsymbol{k}_i$, $i = 1, \ldots, L$. To find out how, consider the PWA closed-loop dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i + \boldsymbol{B}_i(-\boldsymbol{K}_i \boldsymbol{x} + \boldsymbol{k}_i), \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \tag{5.17}$$

From the definition of $\boldsymbol{b}(t) \in \mathbb{R}^{2n}$ in (5.5), $\boldsymbol{b}_i$ is clearly:

$$\boldsymbol{b}_i = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)}) - \boldsymbol{A}_i \boldsymbol{x}_{\mathrm{c}}^{(i)} - \boldsymbol{B}_i \boldsymbol{u}_{\mathrm{c}}^{(i)}, \quad i = 1, 2, \ldots, L, \tag{5.18}$$

where each $\boldsymbol{u}_{\mathrm{c}}^{(i)}$, $i = 1, 2, \ldots, L$, satisfies $\dot{\boldsymbol{x}}_{\mathrm{c}}^{(i)} = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)})$. Therefore,

$$\begin{aligned} \dot{\boldsymbol{x}} = \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{f}(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)}) - \boldsymbol{A}_i\boldsymbol{x}_{\mathrm{c}}^{(i)} - \boldsymbol{B}_i\boldsymbol{u}_{\mathrm{c}}^{(i)} - \boldsymbol{B}_i\boldsymbol{K}_i\boldsymbol{x} + \boldsymbol{B}_i\boldsymbol{k}_i, \\ \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \end{aligned} \tag{5.19}$$

It is then clear that by choosing $\boldsymbol{k}_i = \boldsymbol{K}_i\boldsymbol{x}_{\mathrm{c}}^{(i)} + \boldsymbol{u}_{\mathrm{c}}^{(i)}$, $i = 1, \ldots, L$, the closed-loop PWA dynamics (5.19) becomes:

$$\begin{aligned} \dot{\boldsymbol{x}} = (\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{K}_i)\boldsymbol{x} - (\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{K}_i)\boldsymbol{x}_{\mathrm{c}}^{(i)} + \boldsymbol{f}(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)}), \\ \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \end{aligned} \tag{5.20}$$

Notice that each operating point $(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)})$ is a specific point on the (exactly dynamically feasible) reference trajectory $(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{u}_{\mathrm{d}}(t)$, then each $(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)})$ exactly satisfies $\dot{\boldsymbol{x}}_{\mathrm{c}}^{(i)} = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{c}}^{(i)}, \boldsymbol{u}_{\mathrm{c}}^{(i)})$, $i = 1, \ldots, L$. Thus, the PWA closed-loop dynamics is reduced to:

$$\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\mathrm{c}}^{(i)} = (\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{K}_i)(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}^{(i)}), \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \tag{5.21}$$

It is then clear that each $\boldsymbol{K}_i$, $i = 1, 2, \ldots, L$, must be synthesized so that $(\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{K}_i)$ are Hurwitz [50], which will imply that for each $i = 1, 2, \ldots, L$, $\boldsymbol{x}(t) \to \boldsymbol{x}_{\mathrm{c}}^{(i)}$ asymptotically if $\boldsymbol{x}(t) \in \mathcal{E}_i$. The referred control law for the PWA system is therefore simply:

$$\boldsymbol{u} = -\boldsymbol{K}_i\boldsymbol{x} + \left(\boldsymbol{K}_i\boldsymbol{x}_{\mathrm{c}}^{(i)} + \boldsymbol{u}_{\mathrm{c}}^{(i)}\right) \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \tag{5.22}$$

It is important to point out that control law (5.22) guarantees asymptotic regulation of each equilibrium point for the PWA system (5.15). Clearly, if we were to implement the PWA control law (5.22) on the actual manipulator and $\boldsymbol{x}(t) \in \mathcal{E}_i$, the state $\boldsymbol{x}(t)$ would reach asymptotically $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ and stay there. However, for the actual manipulator with nonlinear dynamics (5.3)-(5.4), the objective is to track each point in the reference trajectory. Therefore, control law (5.22) must be tailored to achieve tracking on the actual robot manipulator. The referred PWA control law proposed in this dissertation to achieve trajectory tracking, which is a tailored version of (5.22), takes the following final form:

$$\boldsymbol{u}(\boldsymbol{x}) = -\boldsymbol{K}_i\boldsymbol{x} + (\boldsymbol{K}_i\boldsymbol{x}_{\mathrm{d}}(t) + \boldsymbol{u}_{\mathrm{d}}(t)) \quad \text{if } \boldsymbol{x} \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \tag{5.23}$$

The difference between (5.22) and (5.23) is that (for all time) the center points $\boldsymbol{x}_{\mathrm{c}}^{(i)}$ are replaced with the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$, and that the points $\boldsymbol{u}_{\mathrm{c}}^{(i)}$ are replaced with the feedforward torques $\boldsymbol{u}_{\mathrm{d}}(t)$. Note however that (5.23) uses the state-feedback gains $\boldsymbol{K}_i$ designed for control law (5.22).

5.4.2 Synthesizing State-feedback Gains $\boldsymbol{K}_i$

There are several options to synthesize the state-feedback controller gains $\boldsymbol{K}_i \in \mathbb{R}^{n\times 2n}$, $i = 1, 2, \ldots, L$, so that $(\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{K}_i)$ are Hurwitz [50]. We choose the linear quadratic

regulator (LQR) approach [38], because it will allow us to trade off between: (i) deviation of $\boldsymbol{x}(t)$ from $\boldsymbol{x}_{\rm c}^{(i)}$, defined $\tilde{\boldsymbol{x}}_i(t) := \boldsymbol{x}(t) - \boldsymbol{x}_{\rm c}^{(i)}$ and (ii) deviation of $\boldsymbol{u}(t)$ from $\boldsymbol{u}_{\rm c}^{(i)}$, defined $\tilde{\boldsymbol{u}}_i(t) := \boldsymbol{u}(t) - \boldsymbol{u}_{\rm c}^{(i)}$. Besides, it will give a fair comparison against the LQ formulation of the trajectory tracking problem presented in Chapter 4. After all, the development in the present chapter is motivated to overcoming the drawbacks of the approach discussed in Chapter 4.

The continuous-time infinite-horizon LQ optimal control problem, better known as the linear quadratic regulator (LQR), is used here to independently synthesize the state-feedback gains $\boldsymbol{K}_i$, for each pair $(\boldsymbol{A}_i, \boldsymbol{B}_i)$, $i = 1, 2, \ldots, L$. In other words, independently for each $i = 1, 2, \ldots, L$, the problem is to minimize the continuous-time infinite-horizon quadratic cost functional:

$$J_i = \frac{1}{2} \int_0^\infty \left(\tilde{\boldsymbol{x}}_i(t)^\top \boldsymbol{Q}_i \tilde{\boldsymbol{x}}_i(t) + \tilde{\boldsymbol{u}}_i(t)^\top \boldsymbol{V}_i \tilde{\boldsymbol{u}}_i(t) \right) \, dt, \tag{5.24}$$

subject to linear time-invariant dynamics:

$$\dot{\tilde{\boldsymbol{x}}}_i(t) = \boldsymbol{A}_i \tilde{\boldsymbol{x}}_i(t) + \boldsymbol{B}_i \tilde{\boldsymbol{u}}_i(t), \quad \tilde{\boldsymbol{x}}_i(0) = \tilde{\boldsymbol{x}}_i^{\rm init},$$

where the (constant) weighting matrices $\boldsymbol{Q}_i = \boldsymbol{Q}_i^\top \succeq 0$ and $\boldsymbol{V}_i = \boldsymbol{V}_i^\top \succ 0$. In this infinite-horizon formulation, in order to guarantee that the cost functional (5.24) remains bounded, each (constant) system matrix pair $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ must be controllable.

In classical optimal control theory, it is a standard exercise to show that the optimal state-feedback gains can be computed by $\boldsymbol{K}_i = \boldsymbol{V}_i^{-1} \boldsymbol{B}_i^\top \boldsymbol{P}_i$, where each $\boldsymbol{P}_i$, $i = 1, 2, \ldots, L$, satisfies the algebraic Riccati equation (ARE) [38, 22]:

$$\boldsymbol{O} = \boldsymbol{A}_i^\top \boldsymbol{P}_i + \boldsymbol{P}_i \boldsymbol{A}_i + \boldsymbol{Q}_i - \boldsymbol{P}_i \boldsymbol{B}_i \boldsymbol{V}_i^{-1} \boldsymbol{B}_i^\top \boldsymbol{P}_i.$$

Even though the above methodology allows to use different weighting matrices $(\boldsymbol{Q}_i, \boldsymbol{V}_i)$ for each ellipsoidal region $\mathcal{E}_i$, in this dissertation we use the same weighting matrices $(\boldsymbol{Q}, \boldsymbol{V})$ for all $\mathcal{E}_i$, $i = 1, 2, \ldots, L$. Of course, since the local system dynamics matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, are different for each ellipsoidal region $\mathcal{E}_i$, then each state-feedback gain $\boldsymbol{K}_i$ will be different for each $\mathcal{E}_i$, $i = 1, 2, \ldots, L$.

5.4.3 Controller Switching Strategy

A total of L different state-feedback controller gains $\boldsymbol{K}_i$ are obtained, and are to be scheduled by the PWA control law (5.23). Therefore a switching strategy is needed to switch controller gains, as the actual state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t)^\top \ \dot{\boldsymbol{\theta}}(t)^\top)^\top$ transitions from one ellipsoidal region to another. The main idea is simple and intuitive, and is summarized as follows:

1. There are a total of L different feedback-controller gains, $\boldsymbol{K}_i$, corresponding to each operating ellipsoid $\mathcal{E}_i$, $i = 1, \ldots, L$. In other words, there are L discrete modes of operation.

2. At every time instant t, we measure the actual state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ and identify the ellipsoidal region it belongs to.

3. Say, $\boldsymbol{x}(t) \in \mathcal{E}_j$, then the j-th feedback-controller gain $\boldsymbol{K}_j$ is used, until the measured state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ transitions into any other ellipsoidal region.

4. Say that at some instant, the measured state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ crosses the boundary of $\mathcal{E}_k$, then the feedback-controller gain is switched from $\boldsymbol{K}_j$ to $\boldsymbol{K}_k$.

5. In situations when the measured state $\boldsymbol{x}(t)$ belongs to several ellipsoids, say, $\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k, \mathcal{E}_l$, choose the controller gain $\boldsymbol{K}_m$, where the controller index $m = \max\{i, j, k, l\}$.

6. For the unlikely event in which the measured state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ does not belong to any ellipsoidal region, the PD feedback gain matrices $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$ are utilized.

In the actual experimental setup, the above tasks must be executed within one sampling period, $T_s = 1$ millisecond. Namely, measuring the actual state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$, identifying the ellipsoidal region it belongs to, looking up the respective feedback-controller gain matrix, and then from (5.23) compute the required control torque. In order to achieve these computations in real time, the task of searching for the ellipsoidal region that $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ belongs to, should be done only within a reduced "window" of ellipsoids. In other words, suppose that at time $t = kT_s$, $\boldsymbol{x}(t) \in \mathcal{E}_j$, then for the next sampling period $t = (k+1)T_s$, the search is performed only within $\{\mathcal{E}_{j-10}, \ldots, \mathcal{E}_j, \ldots, \mathcal{E}_{j+10}\}$. Querying whether $\boldsymbol{x}(t) \in \mathcal{E}_j$ is a very simple computation aided by using the third parametrization in (5.8). Namely,

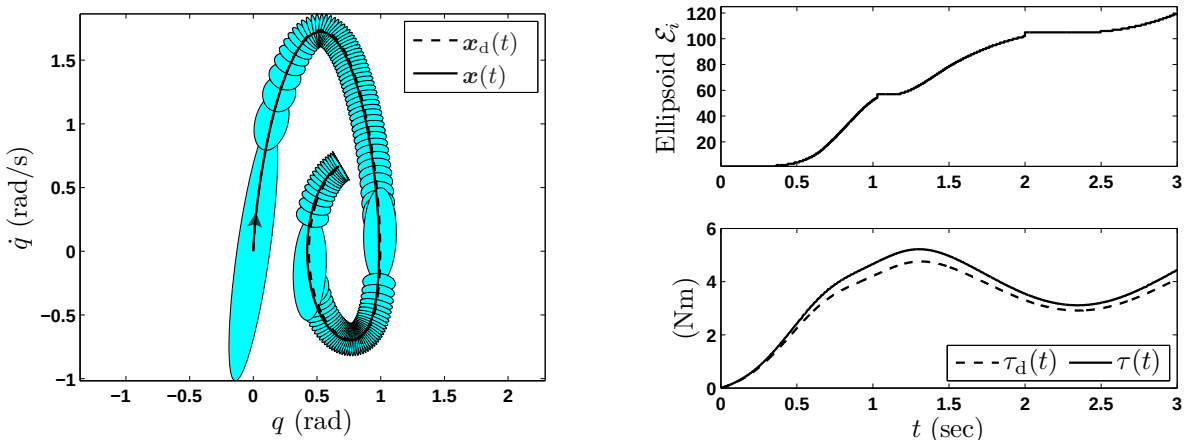
$$\text{if } \|\boldsymbol{S}_j\boldsymbol{x} + \boldsymbol{s}_j\|_2 \leq 1, \text{ then } \boldsymbol{x} \in \mathcal{E}_j, \text{else } \boldsymbol{x} \notin \mathcal{E}_j.$$

5.4.4 Case Study, 1-DOF Manipulator

We apply the synthesis methodology to the 1-DOF manipulator of Fig. 5.2, introduced in Section 5.3. In that case Coulomb friction was assumed zero, and the gear ratios are equal to one, implying $\theta = q$, $u = \tau$. We presented the results of generating the operating points $\{\boldsymbol{x}_{\mathrm{c}}^{(1)}, \ldots, \boldsymbol{x}_{\mathrm{c}}^{(L)}\}$, corresponding system matrices $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$, and ellipsoids-parameterizing matrices $\boldsymbol{R}_i$, $i = 1, \ldots, L$. If we recall, for that case L turned out 121, which means that a total of 121 state-feedback gains $\boldsymbol{K}_i \in \mathbb{R}^{1\times 2}$ are generated.

The weighting matrices in this case $\boldsymbol{Q} \in \mathbb{R}^{2\times 2}$, $V \in \mathbb{R}$, are chosen the same for all $\mathcal{E}_i$, $i = 1, 2, \ldots, L$. Clearly, since $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, are different for each ellipsoidal region $\mathcal{E}_i$, the state-feedback gains $\boldsymbol{K}_i$ will be different for each $\mathcal{E}_i$, $i = 1, 2, \ldots, L$. We set $\boldsymbol{Q} = 500 \cdot \boldsymbol{I}_2$, $V = 1$, and carry out simulations, for which PWA control law (5.23) is implemented with the controller switching strategy described in sub-section 5.4.3.

A 5% parameter uncertainty is introduced in all parameters of the nonlinear dynamic model (5.14). Both the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$ and the actual state $\boldsymbol{x}(t)$ are shown in Fig. 5.7(a). As seen from the figure, the actual state $\boldsymbol{x}(t)$ follows closely the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$, and never leaves the ellipsoidal regions as time marches forward.



(a) State-reference $\boldsymbol{x}_{\mathrm{d}}(t)$ and robot state $\boldsymbol{x}(t)$ (b) Ellipsoid # $\mathcal{E}_i$, feedforward and applied torques

Figure 5.7: Simulation results of the PWA control law for the 1-DOF manipulator.

In Fig. 5.7(b), we show graphically the evolution of ellipsoidal region number (or discrete state) as a function of time. Plotting this discrete state gives us information on which ellipsoid and controller gain were utilized at a particular time, which ultimately certifies that each single scheduled controller was utilized. Notice the non-decreasing fashion in which the ellipsoid number (or discrete state) evolves, which means that each of the scheduled controllers were utilized as planned.

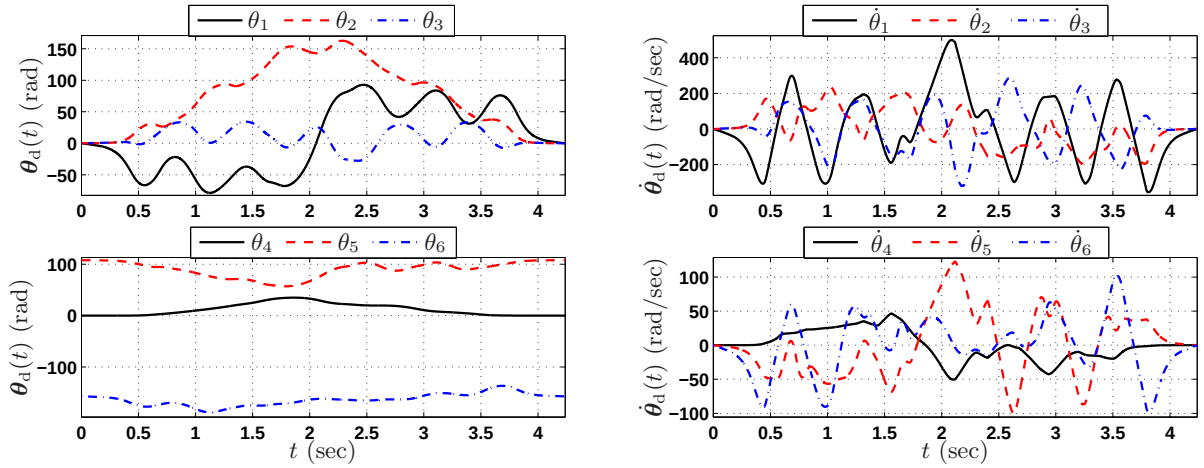
The corresponding feedforward and applied torques are shown in Fig. 5.7(b). It is observed that due to the parameter uncertainty introduced in the nonlinear dynamic model, the applied torque $\tau(t)$ deviate from the feedforward torque $\tau_{\mathrm{d}}(t)$. Notice that switching between controller gains (as the state transitions ellipsoids) does not introduce discontinuity in the control signal.

5.5 Application to 6-axis Industrial Manipulator

In this section, we present the application of PWA modeling and controller synthesis to achieve trajectory tracking on the 6-DOF industrial manipulator, FANUC M-16iB. First we give a detailed exposition through the design process and synthesis results. Then, experiments are conducted to evaluate the feasibility and effectiveness of the methodology proposed in this chapter.

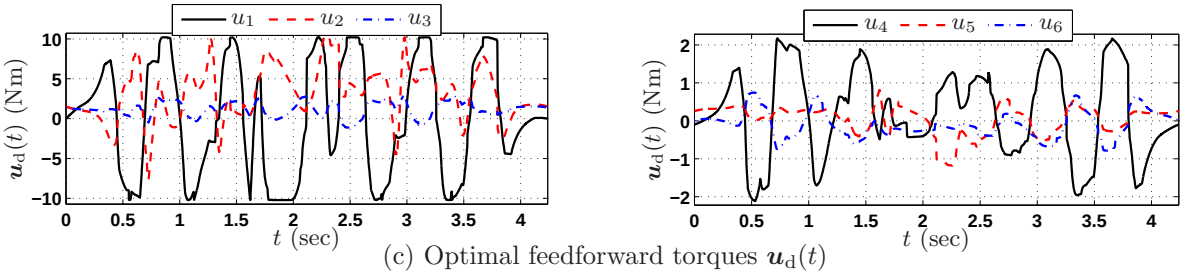
5.5.1 Near Time-optimal Trajectory

We apply the proposed PWA modeling and controller synthesis to the 6-axis manipulator under the near time-optimal trajectory. This optimal trajectory solution was presented in



(a) Reference trajectory position $\boldsymbol{\theta}_{\rm d}(t)$

(b) Reference trajectory velocity $\dot{\boldsymbol{\theta}}_{\rm d}(t)$



(c) Optimal feedforward torques $\boldsymbol{u}_{\rm d}(t)$

Figure 5.8: Motor-side near time-optimal trajectory positions $\boldsymbol{\theta}_{\rm d}(t)$, velocities $\dot{\boldsymbol{\theta}}_{\rm d}(t)$, and torques $\boldsymbol{u}_{\rm d}(t)$.

Chapter 3, and then utilized in Chapter 4 to evaluate the feasibility and effectiveness of the LQ-based trajectory tracking scheme, developed in that chapter. For the sake of a self-contained chapter, we present again the optimal trajectory and some of its key parameters, which at times will seem to overlap with comments already made in Chapter 4. That being said, in this section we are interested in synthesizing and experimentally implementing the PWA control law (5.23) for the 6-axis manipulator.

For this manipulator the degrees of freedom n is 6, which means that the state vector in nonlinear dynamic model (5.3)-(5.4) is 12-dimensional, $\boldsymbol{x} = (\boldsymbol{x}_1^\top \ \boldsymbol{x}_2^\top)^\top = (\boldsymbol{\theta}^\top \ \dot{\boldsymbol{\theta}}^\top)^\top \in \mathbb{R}^{12}$. The near time-optimal trajectory is generated by first using algorithm (3.13) with $\lambda = 0.02$, and then performing the appropriate conversions to motor-side. The motor-side optimal-reference trajectory pair $(\boldsymbol{x}_{\rm d}(t), \boldsymbol{u}_{\rm d}(t))$ is presented in Fig. 5.8, where $\boldsymbol{x}_{\rm d} = (\boldsymbol{\theta}_{\rm d}^\top \ \dot{\boldsymbol{\theta}}_{\rm d}^\top)^\top$ and the feedforward torque $\boldsymbol{u}_{\rm d}(t)$ satisfy (5.2).

The optimal trajectory in Fig. 5.8 represents the fastest dynamically feasible trajectory achievable by the real manipulator, with a total traversal time $t_f = 4.238$ seconds. In order to linearize the nonlinear dynamics (5.3)-(5.4) at several points of the reference trajectory,

the recursive Newton-Euler algorithm efficiently coded in C as a toolbox for MATLAB® [35], is utilized. Since the sign$(\cdot)$ function is discontinuous at zero, for the purposes of linearization it is approximated by the smooth function satur$(\cdot)$, defined as:

$$\text{satur}(x) := \frac{\tan^{-1}(\eta x) - \tan^{-1}(-\eta x)}{\pi}, \tag{5.25}$$

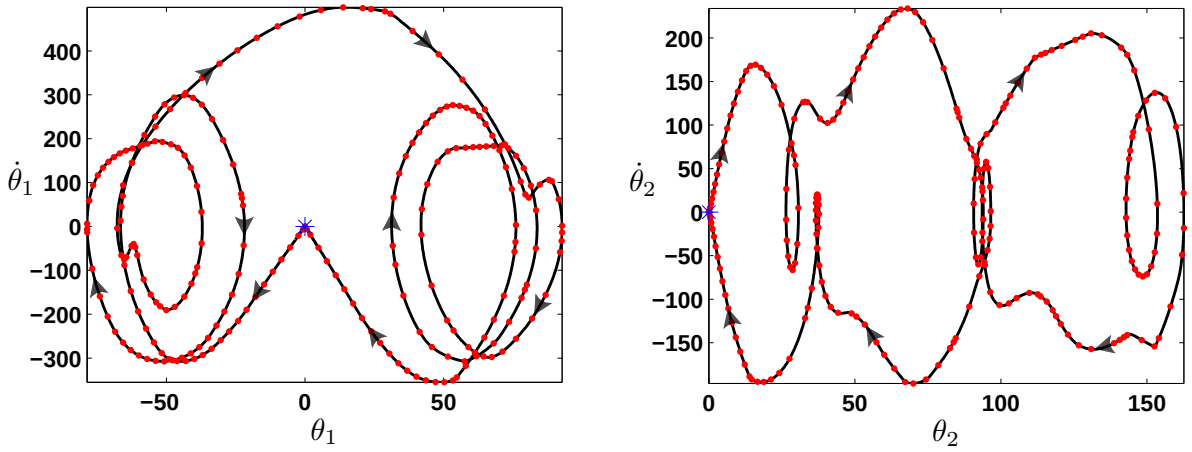
with $\eta = 5$, which ensures differentiability near zero velocities.

5.5.2 Piecewise Affine Modeling Synthesis

The first part of our synthesis methodology is to choose the operating points $\boldsymbol{x}_{\text{c}}^{(i)} \in \mathbb{R}^{12}$, $i = 1, \ldots, L$, and corresponding system matrices $\boldsymbol{A}_i \in \mathbb{R}^{12\times 12}$, $\boldsymbol{b}_i \in \mathbb{R}^{12}$, $\boldsymbol{B}_i \in \mathbb{R}^{12\times 6}$. Then, constructing the positive-definite matrices $\boldsymbol{R}_i \in \mathbb{R}^{12\times 12}$, $i = 1, \ldots, L$, that parameterize the ellipsoidal regions $\mathcal{E}_i \subset \mathbb{R}^{12}$. We apply the procedures presented in Section 5.2: **Algorithm to Choose the Operating Points** and **Obtaining $\boldsymbol{R}_i$ to Parameterize $\mathcal{E}_i$**.

The resulting number of operating points is $L = 224$, which means that 224 system matrices $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{B}_i)$ are also generated. These local system dynamics are guaranteed to exhibit a 1% dynamics "variation" with respect to the simple metric in (5.10). The resulting operating points, $\{\boldsymbol{x}_{\text{c}}^{(1)}, \boldsymbol{x}_{\text{c}}^{(2)}, \ldots, \boldsymbol{x}_{\text{c}}^{(224)}\}$, are presented as projections onto the planes $(\theta_j, \dot{\theta}_j)$, $j = 1, \ldots, 6$. The referred projections together with the state-reference trajectory are presented in Fig. 5.9, which is split into two parts in consecutive pages. The reason we present projections of the state-reference trajectory $\boldsymbol{x}_{\text{d}}(t) \in \mathbb{R}^{12}$ and the corresponding operating points $\boldsymbol{x}_{\text{c}}^{(i)} \in \mathbb{R}^{12}$, $i = 1, \ldots, 224$, is so that we can visualize how spread the operating points have been chosen by our algorithm. The initial and final points of the state-reference trajectory in Fig. 5.9 are marked with an asterisk '*'. Both the initial and final points correspond to the home position of the industrial robot.

Once we have obtained the operating points $\boldsymbol{x}_{\text{c}}^{(i)}$, $i = 1, \ldots, 224$, constructing the ellipsoidal regions $\mathcal{E}_i$ is done by obtaining the parameterizing matrices $\boldsymbol{R}_i$, $i = 1, \ldots, 224$. These



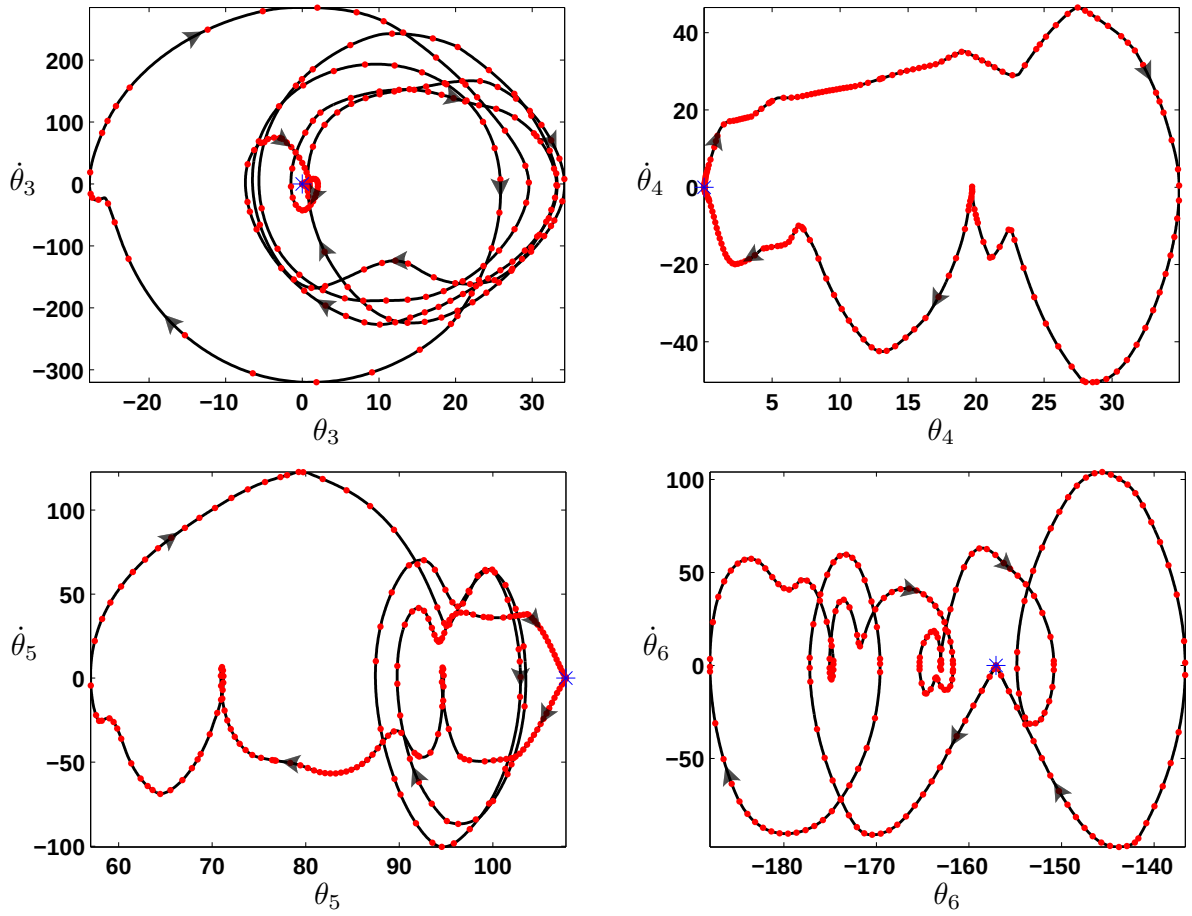


Figure 5.9: Projections of the state-reference trajectory $\boldsymbol{x}_{\mathrm{d}}(t)$ and operating points $\boldsymbol{x}_{\mathrm{c}}^{(i)}$, $i = 1, \ldots, 224$, onto the planes $(\theta_j, \dot{\theta}_j)$, $j = 1, \ldots, 6$.

matrices are readily built with the procedure presented in Section 5.2: **Obtaining $\boldsymbol{R}_i$ to Parameterize $\mathcal{E}_i$.**

5.5.3 Synthesis of State-feedback Controller Gains

In order to implement the PWA control law (5.23), we need to synthesize the state-feedback controller gains $\boldsymbol{K}_i \in \mathbb{R}^{6\times 12}$, $i = 1, \ldots, 224$. These gains are efficiently computed with the procedure presented in Section 5.4. The same weighting matrices $\boldsymbol{Q}_i = \boldsymbol{Q}$ and $\boldsymbol{V}_i = \boldsymbol{V}$ are used for all $i = 1, \ldots, 224$. Clearly, since the system dynamics pairs $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ are guaranteed to be different, then the resulting controllers $\boldsymbol{K}_i$, $i = 1, \ldots, 224$, are different even though $\boldsymbol{Q}_i = \boldsymbol{Q}$ and $\boldsymbol{V}_i = \boldsymbol{V}$.

We tune the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{V}$ using the Bryson's rule [44], in a similar manner as it was done for Chapter 4. First, it should be realized that PWA control law (5.23) can be

conveniently re-written, by partitioning $\boldsymbol{K}_i := [\boldsymbol{K}_i^{\text{pos}} \ \boldsymbol{K}_i^{\text{vel}}]$, $i = 1, \ldots, L$, where $\boldsymbol{K}_i^{\text{pos}} \in \mathbb{R}^{6\times 6}$, $\boldsymbol{K}_i^{\text{vel}} \in \mathbb{R}^{6\times 6}$. Thus, the actual way in which control law (5.23) is implemented:

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) &= \boldsymbol{K}_i^{\text{pos}}(\boldsymbol{\theta}_{\text{d}}(t) - \boldsymbol{\theta}(t)) + \boldsymbol{K}_i^{\text{vel}}(\dot{\boldsymbol{\theta}}_{\text{d}}(t) - \dot{\boldsymbol{\theta}}(t)) + \boldsymbol{u}_{\text{d}}(t) \\ &\text{if } \boldsymbol{x} = (\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \mathcal{E}_i, \quad i \in \{1, 2, \ldots, L\}. \end{aligned} \tag{5.26}$$

We break down $\boldsymbol{Q}$ into a term that penalizes position deviation $\boldsymbol{Q}_{\text{pos}} \in \mathbb{R}^{6\times 6}$, and a term that penalizes velocity deviation $\boldsymbol{Q}_{\text{vel}} \in \mathbb{R}^{6\times 6}$, i.e., $\boldsymbol{Q} = \text{diag}(\boldsymbol{Q}_{\text{pos}}, \boldsymbol{Q}_{\text{vel}})$. Then, we choose $\boldsymbol{Q}_{\text{pos}}$, $\boldsymbol{Q}_{\text{vel}}$, and $\boldsymbol{V}$ diagonal with:

$$[\boldsymbol{Q}_{\text{pos}}]_{jj} = \frac{1}{\text{maximum acceptable value of } [\boldsymbol{\theta}_{\text{d}}(t) - \boldsymbol{\theta}(t)]_j^2}, \quad j = 1, 2, \ldots, 6$$

$$[\boldsymbol{Q}_{\text{vel}}]_{jj} = \frac{1}{\text{maximum acceptable value of } \left[\dot{\boldsymbol{\theta}}_{\text{d}}(t) - \dot{\boldsymbol{\theta}}(t)\right]_j^2}, \quad j = 1, 2, \ldots, 6$$

$$\boldsymbol{V}_{jj} = \frac{1}{\text{maximum acceptable value of } [\boldsymbol{u}_{\text{d}}(t) - \boldsymbol{u}(t)]_j^2}, \quad j = 1, 2, \ldots, 6.$$

In this dissertation, we use Bryson's rule in a way so as to generate state-feedback gains, $\boldsymbol{K}_i^{\text{pos}}$, $\boldsymbol{K}_i^{\text{vel}}$, $i = 1, \ldots, L$, that are "comparable" to the constant feedback gains $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$. The specific values of $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$ are given as:

$$\begin{aligned} \boldsymbol{K}_{\text{P}} &= \text{diag}(4.4717,\ 4.0786,\ 0.8297,\ 1.4449,\ 0.0399,\ 0.1415) \\ \boldsymbol{K}_{\text{V}} &= \text{diag}(0.4255,\ 0.3910,\ 0.0810,\ 0.0622,\ 0.0025,\ 0.0069), \end{aligned}$$

which are clearly diagonal, since these gains are designed in a decentralized manner for each robot joint. On the other hand, our PWA controller synthesis approach takes into account the (local) coupled dynamics. Therefore, it will produce local feedback gains $\boldsymbol{K}_i^{\text{pos}}$, $\boldsymbol{K}_i^{\text{vel}}$, $i = 1, \ldots, L$, with nonzero off-diagonal elements to account for the (local) coupled dynamics. We thus use Bryson's rule but tune the weighting matrices $\boldsymbol{Q}_{\text{pos}}$, $\boldsymbol{Q}_{\text{vel}}$, and $\boldsymbol{V}$ such that:

$$\forall i = 1, 2 \ldots, L, \quad \sigma_{\max}(\boldsymbol{K}_i^{\text{pos}}) \approx \sigma_{\max}(\boldsymbol{K}_{\text{P}}), \quad \sigma_{\max}(\boldsymbol{K}_i^{\text{vel}}) \approx \sigma_{\max}(\boldsymbol{K}_{\text{V}}), \tag{5.27}$$

where $\sigma_{\max}(\cdot)$ is the maximum-singular-value matrix norm. In this manner, we will generate controllers that are "comparable" (in the above sense) to the constant feedback gains $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$. Of course, since $\boldsymbol{K}_i^{\text{pos}}$, $\boldsymbol{K}_i^{\text{vel}}$, $i = 1, \ldots, L$, are synthesized taking into account not only the coupling, but most importantly the local dynamics $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ at strategic points of the reference trajectory, it is expected achieve better results.

With the above provisos in mind, weighting matrices $\boldsymbol{Q}_{\text{pos}}$ and $\boldsymbol{Q}_{\text{vel}}$, which respectively penalize position and velocity deviation from the reference trajectory, are conveniently tuned as:

$$\begin{aligned} \boldsymbol{Q}_{\text{pos}} &= \text{diag}\left(\frac{1}{1^2}, \ \frac{1}{1^2}, \ \frac{1}{1^2}, \ \frac{1}{1^2}, \ \frac{1}{1^2}, \ \frac{1}{1^2}\right) = \boldsymbol{I} \\ \boldsymbol{Q}_{\text{vel}} &= \text{diag}\left(\frac{1}{11.18^2}, \ \frac{1}{11.18^2}, \ \frac{1}{11.18^2}, \ \frac{1}{11.18^2}, \ \frac{1}{11.18^2}, \ \frac{1}{11.18^2}\right) = 0.008\boldsymbol{I}, \end{aligned}$$

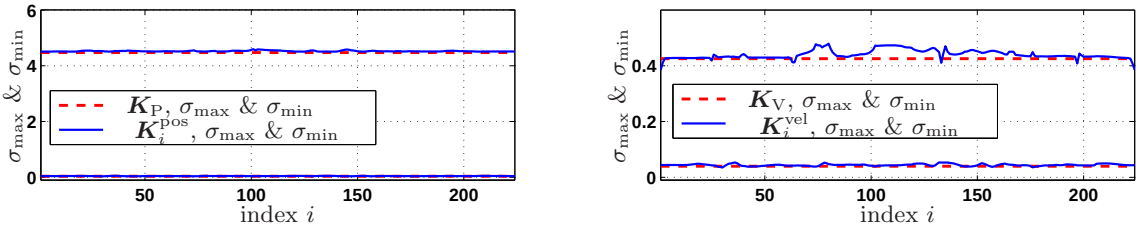


Figure 5.10: Maximum singular values of $\boldsymbol{K}_i^{\mathrm{pos}}$ and $\boldsymbol{K}_i^{\mathrm{vel}}$ for all $i = 1, \ldots, 224$. For reference purposes, the singular values of the feedback gain matrices $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$ are also included.

where $\boldsymbol{I}$ is the 6×6 identity matrix. The weighting matrix $\boldsymbol{V}$, which penalizes torque command deviation from the nominal feedforward torque, is tuned as:

$$\boldsymbol{V} = \mathrm{diag}\left(\frac{1}{4.51^2}, \frac{1}{4.22^2}, \frac{1}{0.86^2}, \frac{1}{1.51^2}, \frac{1}{0.05^2}, \frac{1}{0.15^2}\right).$$

Since for this case $L = 224$, a total of 224 state-feedback matrices, $\boldsymbol{K}_i^{\mathrm{pos}}$ and $\boldsymbol{K}_i^{\mathrm{vel}}$, are synthesized. A convenient way to present all these matrices is by plotting their maximum and minimum singular values, i.e., $\sigma_{\max}(\boldsymbol{K}_i^{\mathrm{pos}})$, $\sigma_{\min}(\boldsymbol{K}_i^{\mathrm{pos}})$, $\sigma_{\max}(\boldsymbol{K}_i^{\mathrm{vel}})$ and $\sigma_{\min}(\boldsymbol{K}_i^{\mathrm{vel}})$, $i = 1, 2, \ldots, 224$. In this manner, we can visualize the minimum and maximum gain amplifications. As discussed in Chapter 4, this visualization is important to ensure the feasibility of implementing these state-feedback gains in experiments. The referred plot of singular values is presented in Fig. 5.10, where we have also included the maximum and minimum singular values of $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$. We can conclude from Fig. 5.10, that for all $i = 1, \ldots, 224$, the state-feedback matrices, $\boldsymbol{K}_i^{\mathrm{pos}}$ and $\boldsymbol{K}_i^{\mathrm{vel}}$, are feasible to carry out experiments on the real manipulator.

It is interesting to present the actual matrices $\boldsymbol{K}_i^{\mathrm{pos}}$ and $\boldsymbol{K}_i^{\mathrm{vel}}$ for some i, say for $i = 1$,

$$\boldsymbol{K}_1^{\mathrm{pos}} = \begin{pmatrix} 4.4864 & -0.0038 & 0.0065 & 0.0841 & -0.0014 & 0.4528 \\ 0.0051 & 4.0832 & -1.1545 & -0.0013 & 0.1286 & -0.0001 \\ -0.0011 & 0.2394 & 0.8253 & 0.0034 & -0.0698 & -0.0000 \\ -0.0091 & -0.0012 & -0.0058 & 1.4467 & 0.0008 & -0.2011 \\ 0.0000 & -0.0027 & 0.0021 & -0.0000 & 0.0436 & 0.0000 \\ -0.0153 & 0.0000 & -0.0001 & 0.0028 & 0.0000 & 0.1481 \end{pmatrix},$$

$$\boldsymbol{K}_1^{\mathrm{vel}} = \begin{pmatrix} 0.3780 & 0.0001 & 0.0006 & -0.0060 & -0.0003 & 0.0728 \\ 0.0009 & 0.3654 & -0.1022 & 0.0002 & 0.0502 & -0.0001 \\ -0.0001 & 0.0280 & 0.0688 & 0.0006 & -0.0069 & 0.0000 \\ -0.0056 & 0.0001 & -0.0004 & 0.0733 & 0.0001 & -0.0150 \\ 0.0000 & -0.0003 & 0.0006 & -0.0000 & 0.0012 & 0.0000 \\ -0.0022 & -0.0000 & -0.0000 & 0.0023 & 0.0000 & 0.0038 \end{pmatrix}.$$

Note that the diagonal elements are approximate to those of $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{V}}$. Likewise, the off-diagonal entries are nonzero to account for the (local) coupled dynamics. Of course, for different i, these state-feedback matrices are different since the local dynamics $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ change from one operating point to another.

Likewise, it should be pointed out from Fig. 5.10, that for all $i = 1, \ldots, 224$, the maximum gain amplifications $\sigma_{\max}(\boldsymbol{K}_i^{\text{pos}})$, $\sigma_{\max}(\boldsymbol{K}_i^{\text{vel}})$, do not change suddenly to large or small values, but remain consistent as i increases.

5.5.4 Experimental Evaluations

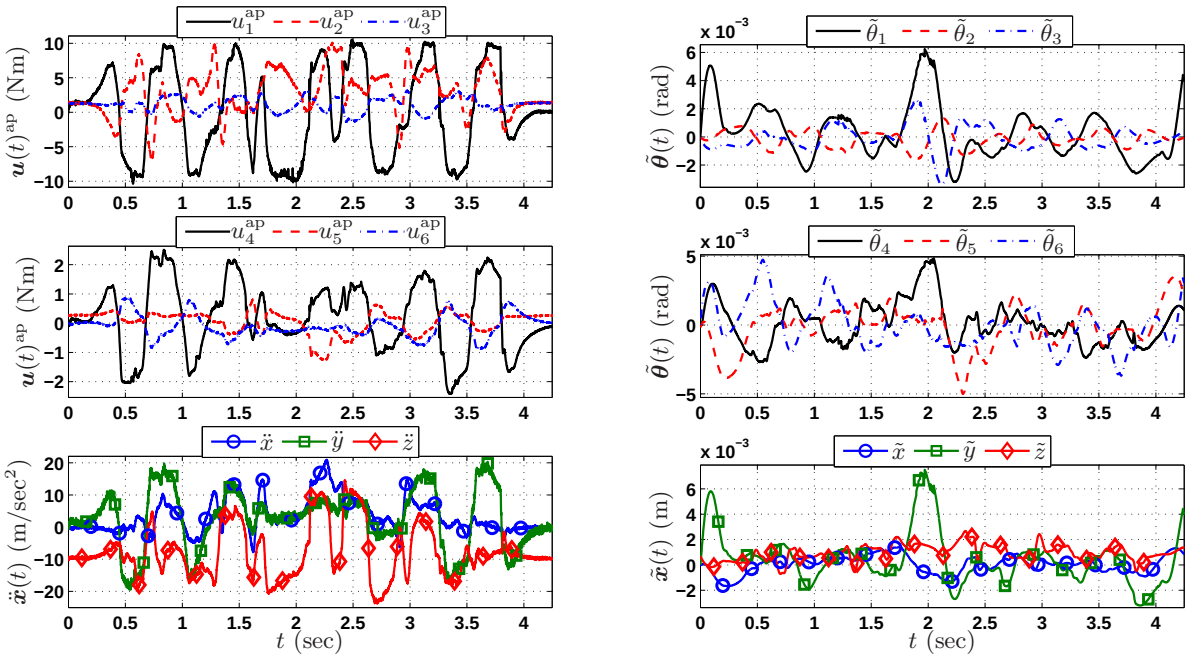
We are in a position to implement the PWA control law (5.26), since all the associated controller parameters have been synthesized. Similar to the experiments presented in Chapter 4, additional 0.5 seconds are required after the end of the trajectory. Therefore, we switch to the feedback gains $\boldsymbol{K}_{\text{P}}$, $\boldsymbol{K}_{\text{V}}$, for all $t > t_f$. In all of the forthcoming plots regarding experiments, we always include what happens right after the end of the trajectory for $10 \cdot T_s$ seconds, where T_s is the sampling period in our experimental setup. In this manner, we can monitor the effect of switching to the feedback gains, $\boldsymbol{K}_{\text{P}}$ and $\boldsymbol{K}_{\text{V}}$.

The referred experimental results are shown in Fig. 5.11, where the applied torques $\boldsymbol{u}(t)^{\text{ap}}$ are the commanded torques computed with PWA control law (5.26). The accelerometer readings are simply to make sure that no sudden changes in acceleration occur, since the trajectory is near time-optimal. The feedback torques $\boldsymbol{u}(t)^{\text{fb}}$ correspond to the feedback portion of PWA control law (5.26).

The corresponding evolution of ellipsoidal region number as a function of time is shown in Fig. 5.12. This plot gives us information on the ellipsoid and controller gain that were utilized at a particular time, which certifies that each single scheduled controller was utilized. The non-decreasing fashion in which the ellipsoid number evolves means that each of the scheduled controllers were utilized as planned.

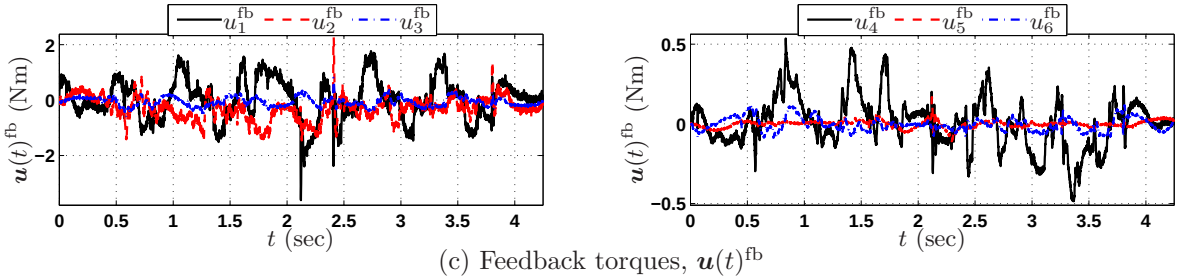
The results in Fig. 5.11 should be compared against the corresponding results in Fig. 3.7 and Fig. 4.6, which correspond respectively to implementing control laws (3.14) and (4.22) on the same near time-optimal trajectory solution. The experimental results in Fig. 5.11 exhibit several superior features that are pointed out as follows:

- The PWA control law requires the storage of significantly less feedback matrices than the ATV control law presented in Chapter 4.
- When switching to $\boldsymbol{K}_{\text{P}}$ and $\boldsymbol{K}_{\text{V}}$ at the end of the trajectory, no sudden changes in the feedback torques occur.
- Regarding the tracking errors, there is improvement compared to the ATV controller. This improvement is particularly noticeable for the last three joints, i.e., $\tilde{\theta}_4$, $\tilde{\theta}_5$, $\tilde{\theta}_6$.
- The RMS values for each joint tracking errors and Cartesian-space tracking errors are presented in Table 5.1. In this table, the results for the three major controllers used in this dissertation are presented, i.e., PID control law (3.14), ATV control law (4.22), and PWA control law (5.26).



(a) Applied torques and accelerometer readings

(b) Motor-side and Cartesian-space tracking errors



(c) Feedback torques, $\boldsymbol{u}(t)^{\rm fb}$

Figure 5.11: Experimental results when implementing the PWA control law (5.26). The reference trajectory corresponds to the dynamically feasible near time-optimal trajectory presented in Chapter 3 and utilized in Chapter 4.

- In all cases, PWA control law (5.26) exhibits the best performance. The controller comparisons are fair, since it was ensured that the maximum gain amplifications are of similar magnitudes for the three controllers, as seen from Fig. 4.5 and Fig. 5.10.
- Without doubt, the reason for the superior results is related to having taken into account the local and coupled dynamics for the synthesis of PWA control law (5.26).

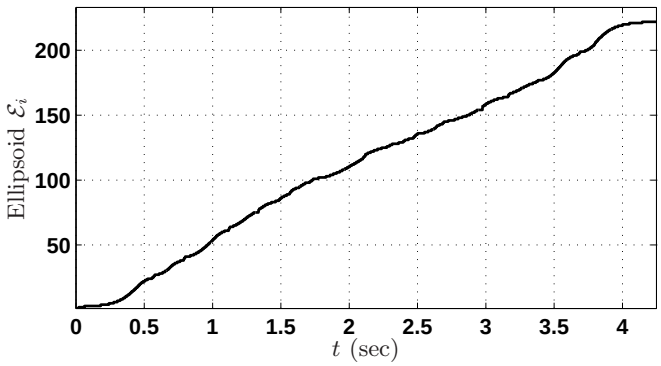


Figure 5.12: Evolution of ellipsoidal region number as a function of time. This plot shows that the state $\boldsymbol{x}(t) = (\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t))$ always belonged to at least one of the synthesized ellipsoidal regions. This means in turn that as the state $\boldsymbol{x}(t)$ evolved, the scheduled controller gains $\boldsymbol{K}_i$ were indeed utilized from beginning to end of the trajectory.

	$\tilde{\theta}_1(k)$	$\tilde{\theta}_2(k)$	$\tilde{\theta}_3(k)$	$\tilde{\theta}_4(k)$	$\tilde{\theta}_5(k)$	$\tilde{\theta}_6(k)$	$\tilde{x}(k)$	$\tilde{y}(k)$	$\tilde{z}(k)$
PID	.0024	.0008	.0012	.0054	.0089	.0050	.0022	.0038	.0019
ATV	.0020	.0008	.0007	.0036	.0042	.0036	.0014	.0027	.0013
PWA	.0020	.0006	.0009	.0016	.0016	.0016	.0007	.0021	.0011

Table 5.1: RMS values comparison for the tracking errors of PID control law (3.14), ATV control law (4.22), and PWA control law (5.26).

5.6 Summary

In this chapter we presented a novel control synthesis methodology to achieve trajectory tracking of robot manipulators. In this approach, the highly nonlinear dynamic model of robot manipulators, which move along a state-reference trajectory, was approximated as a class of piecewise affine (PWA) dynamical systems. We proposed a framework to construct the referred PWA dynamical system, which consists of the steps: (i) choose strategic operating points on the state-reference trajectory with their respective (local) linearized system dynamics, (ii) construct ellipsoidal regions centered at the operating points, whose purpose is to facilitate the scheduling strategy of controller gains designed for each local dynamics. Likewise, in order to switch controller gains as the robot state traverses in the direction of the state-reference trajectory, a simple scheduling strategy was proposed. The controller synthesis for each operating point takes into account the local coupled dynamics, and collocates the local closed-loop equilibrium at the operating point. The resulting PWA control law was designed for the PWA dynamical system. Nonetheless, a tailored version of the PWA control law was proposed to attain trajectory tracking on the robot manipulator. Throughout the chapter, the developed ideas were illustrated with two toy examples, a 1-DOF and 2-DOF

manipulators. These toy examples were rather crucial in gaining insight before moving on to the more complicated 6-DOF industrial manipulator. Finally, the referred PWA control law was implemented in our experimental setup, which showed its feasibility and superiority over the typical PID controller and the ATV controller developed in Chapter 4.

Chapter 6

Conclusions

In this dissertation we have explored two important topics in robotics: (i) optimal trajectory planning was studied in Chapters 2 and 3, (ii) optimal control synthesis methodologies for trajectory tracking of robot manipulators was explored in Chapters 4 and 5.

Time-optimal Trajectory Planning

Regarding the optimal trajectory planning, we concentrated on a very specific problem, namely, the time-optimal trajectory planning of robot manipulators along predetermined geometric paths. In this kind of problem, a purely geometric path is already known, and the task is to find out how to move along this predefined path in the shortest time possible. The resulting optimal solutions are required to be dynamically feasible with respect to the full nonlinear dynamic model (2.5). Likewise, this optimal solution must satisfy physical constraints in the form of minimum and maximum toques. In Chapter 2, we presented a comprehensive treatment on this problem. A formulation was developed that guarantees time-optimal solutions that are dynamically feasible with respect to the complete dynamic model (2.5), which brings a modest but important contribution to existing algorithms.

The referred formulation was presented as a mathematical optimization problem. Unlike the existing mathematical optimization formulations, ours was carefully expressed in a rather specific form, so that it would allow to naturally propose a convex relaxation, which solves exactly the original non-convex problem. It was then shown and verified to be indeed the case, i.e., our proposed convex relaxation solves exactly the referred problem of time-optimal trajectory planning with full dynamic model. We also developed our own discretization scheme to solve optimal control problems, so that an actual numerical solution could be generated and utilized in both simulations and experiments.

We then went further and discussed simulation results on a realistic robot simulator. We studied the effects of implementing time-optimal trajectories and torques on three crucial variables: (i) the tracking errors, (ii) applied torques, and (iii) a 3-axis accelerometer mounted at the robot's end-effector. The findings were that even though the formulation

represents important progress from theoretical and numerical standpoints, it was not ready to be implemented on the experimental setup. Additional criteria were emphasized necessary to incorporate into our formulation. Simulation results aided to argue that implementing the pure time-optimal solutions on the robot manipulator would degrade system performance, or even damaging the hardware setup.

Subsequently, in Chapter 3, we introduced a problem formulation incorporating criteria to overcome the limitations of pure time-optimality. The resulting formulation generates optimal trajectories and torques that are near time-optimal. It was argued that these near time-optimal solutions represent the fastest motions achievable by the real robot manipulator. Starting from the formulation of pure time-optimality, acceleration constraints and penalization of a measure of total jerk were incorporated, both of which proved necessary from real experiments on a 6-axis industrial manipulator. In all cases, the resulting optimal torques and trajectories were always dynamically feasible with respect to the full nonlinear dynamic model (2.5). It was also evidenced from experiments, that the final formulation (3.13) generates the fastest optimal solutions without seriously degrading the system performance. This brings not only a modest theoretical extension to existing algorithms, but also practical insight that is usually disregarded in most previous works. In addition, we showed the versatility of optimization problem (3.13), by utilizing this formulation to generate medium-speed optimal solutions.

Recommendations for future research, i.e., extending the work done in this dissertation regarding optimal trajectory planning along predetermined geometric paths, include:

- Incorporating flexible-joint model in the nonlinear dynamics of robot manipulators. In this dissertation, we have assumed rigid joints. However, it would be interesting and useful to generate the optimal trajectories and torques in the motor-side considering joint compliance. We did some preliminary analysis and development, which leads to a non-convex optimization formulation that seems fairly difficult to convexify. In that attempt, we aimed at obtaining both motor-side and load-side trajectories and torques as the optimization variables (all of them in one optimization formulation).

- However, from the author's current viewpoint, if we break down the problem into two separate sub-problems we would obtain a tractable formulation. Namely, (i) utilize the formulation presented in this dissertation to generate near time-optimal trajectories and torques in the load-side, (ii) use the load-side optimal solutions and formulate the problem of finding the motor-side trajectories and torques as another optimization problem. In step (ii), only the joint dynamics should be considered. Since the joint dynamics is linear, it is very plausible to obtain a convex formulation. In this formulation, the motor-side trajectories and torques can be generated in such a fashion, so as to minimize the torsional deflection of the springs collocated at each joint to model compliance. One could for instance, minimize the potential energy that these springs are known to store when subjected to deflection.

Optimal Control Synthesis Methodologies for Trajectory Tracking

Following the work done on near time-optimal trajectories and torques, our immediate goal was to develop control algorithms suitable for trajectory tracking under such fast trajectories. In Chapter 4, we explored an approach based on the LQ optimal control for LTV dynamical systems. Even though the dynamic model of robot manipulators is inherently nonlinear, in order to address the problem of trajectory tracking, we developed a control synthesis methodology for a class of affine time-varying (ATV) dynamical systems. We showed how the nonlinear dynamics can be approximated along the reference trajectory as an affine time-varying dynamical system. Then, a time-varying control law to achieve trajectory tracking on the ATV system was developed, which used LQ methods for LTV systems. Since the ATV dynamical system approximates the nonlinear robot dynamics along the reference trajectory, the resulting time-varying control law is suitable to achieve trajectory tracking on the robot manipulator.

The LQ approach for LTV systems, undertaken in Chapter 4, was based on the discrete-time nonlinear dynamic model. Therefore, it was necessary to first discretize the continuous-time nonlinear dynamic model. The optimal trajectory and torques, generated with the algorithm in Chapter 3, are exactly dynamically feasible with respect to the continuous-time nonlinear dynamics. However, it was observed in Chapter 4, that when discretizing the continuous-time nonlinear model to obtain a discrete-time nonlinear model, the optimal reference trajectory and torques were no longer exactly dynamically feasible with respect to the discrete-time non-linear model. This led us to introduce a "disturbance" term in position and velocity, which results from the non-exact dynamic feasibility of the reference trajectory and feedforward torques. Even though the "disturbance" terms are rather small compared to the actual desired positions and velocities, the controller synthesis takes these terms into account. Therefore, the resulting time-varying control law achieves trajectory tracking and compensates for the effects of the referred "disturbance" terms.

We also presented the details on the synthesis process, including numerical techniques for the efficient implementation of linearization along the entire reference trajectory. In order to tune the controller parameters, we used Bryson's rule in such a manner, so as to generate time-varying feedback gains that are "comparable" to the constant feedback gains of a decentralized PD controller. By "comparable", we found insightful to use the maximum singular values of the time-varying feedback gain matrices and to make them approximately match the maximum singular values for the constant PD gains. This allowed us to implement the resulting time-varying control law in the experimental setup of the 6-axis industrial robot, which verified the feasibility of the proposed method. The results also suggested the need to further improve the proposed methodology. In other words, they exposed the main limitations of the proposed algorithm, and motivated the development of the ensuing methodology presented in Chapter 5.

As a recommendation of future research, it would be interesting to develop the obvious

variant. Namely, do not discretize the continuous-time nonlinear dynamics, and instead approximate the robot dynamics along the state-reference trajectory as a continuous-time LTV dynamical system. Then, formulate the tracking problem for the continuous-time LTV dynamics (the non-exact dynamic feasibility of the trajectory would not be a problem anymore). At the end, a continuous-time Riccati equation would obtained, which ultimately has to be discretized at the servo sampling rate, in order to be implemented in the digital computer.

Finally, in Chapter 5 we presented a novel control synthesis methodology to achieve trajectory tracking of robot manipulators. The proposed control algorithm was aimed at overcoming the drawbacks of the closely related ATV controller, presented in Chapter 4. In this novel approach, the highly nonlinear dynamic model of robot manipulators, moving along a state-reference trajectory, was approximated by a class of piecewise affine (PWA) dynamical systems. To construct the referred PWA dynamical system, an algorithm was proposed to choose only strategic points on the state-reference trajectory, as opposed to utilizing all points. These points were chosen to guarantee that the linearized system dynamics for consecutive operating points exhibit a 1% dynamics "variation", with respect to a proposed simple metric.

Once the operating points were selected, with their respective (local) linearized system dynamics, a novel approach was proposed to construct ellipsoidal regions around these operating points. The purpose of the ellipsoidal regions was to facilitate the scheduling strategy of controller gains, designed for each local linear system dynamics. A scheduling (or switching) strategy was proposed as the robot state traverses in the direction of the state-reference trajectory. The advantages of using ellipsoidal regions to perform controller switching are that: (i) they are easy to construct, (ii) it is simple and computationally efficient to identify which ellipsoidal region the current robot-state belongs to.

For each operating point, a local stabilizing MIMO controller is synthesized using an LQR formulation, which takes into account the local coupled dynamics. The resulting local control law guarantees: (i) placing the closed-loop equilibrium at the corresponding operating point, (ii) the local closed-loop equilibrium is asymptotically stable. It was also pointed out that the resulting PWA control law is designed for the PWA dynamical system. Nonetheless, since the PWA system is an approximation of the nonlinear robot dynamics along the state-reference trajectory, a tailored version of the PWA control law was proposed to attain trajectory tracking on the robot manipulator. Throughout the chapter, the referred methodology was illustrated using two toy examples, a 1-DOF and 2-DOF manipulators, which were rather crucial in gaining insight, before moving on to the more complicated 6-DOF industrial manipulator. Finally, the referred PWA control law was implemented in our experimental setup, which showed its feasibility and superiority over the typical PID controller and the ATV controller developed in Chapter 4. It was likewise emphasized that the comparison of controllers is a fair comparison, since the maximum gain amplifications of the feedback gains for the three different controllers are of comparable magnitude.

Recommendations of future research to build on the work done in this dissertation, regarding the control synthesis methodologies for trajectory tracking of robot manipulators,

include:

- Stability analysis of the PWA closed-loop dynamics (5.21) considering switching between ellipsoidal regions. One would expect that a PWA dynamical system for which each discrete state's continuous system is stable would be stable. But this is not necessarily the case. For instance, it is discussed in [46] that a piecewise linear dynamical system, which switches between two independently stable linear systems $\dot{\boldsymbol{x}} = \boldsymbol{A}_1\boldsymbol{x}$ and $\dot{\boldsymbol{x}} = \boldsymbol{A}_2\boldsymbol{x}$, might become unstable under certain switching policies, even though $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are Hurwitz. This is unlikely to happen in the PWA closed-loop dynamics (5.21) since the (local) plant dynamics used for controller synthesis exhibit at the most a 1% relative dynamics "variation", with respect to the metric in (5.10). However, it would be instructive to verify this claim formally through a careful stability analysis. There is background work to prove stability of piecewise linear systems in [51], which relies on linear matrix inequalities (LMI's) to find piecewise quadratic Lyapunov functions. This work might be adopted to our particular setting.

- From the results obtained in this dissertation, the proposed PWA modeling framework to approximate the highly nonlinear and coupled dynamic model of robot manipulators, moving along a state-reference trajectory, is not only feasible but also a powerful concept. The essential ideas are: (i) constructing the PWA dynamical system (i.e., strategic operating points, local system dynamics matrices, and ellipsoidal regions parametrizations), and (ii) the scheduling methodology which switches local controllers as the robot state, which traverses in the direction of the state-reference trajectory, transitions from one ellipsoid to another(s). This opens up the possibility to try out synthesis methodologies for the local controllers from the vast body of powerful techniques available for linear dynamical systems. For instance, the local controllers can be designed based on the local coupled dynamics to satisfy requirements of robust stability and performance using H_∞ optimization methods.

Bibliography

[1] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. 1st. Advanced Textbooks in Control and Signal Processing. Springer-Verlag, 2009.

[2] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons Hoboken^ eNJ NJ, 2006.

[3] S. M. LaValle. *Planning Algorithms*. Available at http://planning.cs.uiuc.edu/. Cambridge, U.K.: Cambridge University Press, 2006.

[4] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. "Time-Optimal Control of Robotic Manipulators Along Specified Paths". In: *The International Journal of Robotics Research* 4.3 (1985), pp. 3–17. DOI: 10.1177/027836498500400301.

[5] K G Shin and N D McKay. "Minimum-time control of robotic manipulators with geometric path constraints". In: *IEEE Transactions on Automatic Control* 30.6 (1985), pp. 531–541.

[6] F. Pfeiffer and R. Johanni. "A concept for manipulator trajectory planning". In: *Robotics and Automation, IEEE Journal of* 3.2 (1987), pp. 115–123. ISSN: 0882-4967. DOI: 10.1109/JRA.1987.1087090.

[7] Zvi Shiller. "Time-energy optimal control of articulated systems with geometric path constraints". In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE. 1994, pp. 2680–2685.

[8] John T. Betts. *Practical methods for optimal control using nonlinear programming*. Vol. 3. Advances in Design and Control. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2001, pp. x+190. ISBN: 0-89871-488-5.

[9] D Costantinescu and EA Croft. "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths". In: *Journal of Robotic Systems* 17.5 (2000), pp. 233–249.

[10] Diederik Verscheure et al. "Time-Optimal Path Tracking for Robots: a Convex Optimization Approach". In: *IEEE Transactions on Automatic Control* (2008).

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. 1st. [Online]. Available: http://www.stanford.edu/ boyd/cvxbook/. Cambridge University Press, 2004.

[12] Denise Lam, Chris Manzie, and Malcolm Good. "Model predictive contouring control". In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE. 2010, pp. 6137–6142.

[13] Gene F Franklin, J David Powell, and Abbas Emami-Naeini. "Feedback control of dynamics systems". In: *Pretince Hall Inc* (1986).

[14] Karl Johan Aström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[15] Rafael Kelly, Victor Santibáñez, and Antonio Loría. *Control of robot manipulators in joint space*. Springer, 2005.

[16] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC PressI Llc, 1994.

[17] Roy Featherstone and David Orin. "Robot dynamics: equations and algorithms". In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 1. IEEE. 2000, pp. 826–834.

[18] Hassan K Khalil. *Nonlinear systems*. Upper Saddle River, NJ: Prentice Hall, 2002.

[19] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 1. 1. Prentice hall New Jersey, 1991.

[20] Russ Tedrake. "Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines Course Notes for MIT 6.832". In: *Working draft edition* (2009).

[21] Arthur Earl Bryson. *Applied optimal control: optimization, estimation, and control*. Taylor & Francis, 1975.

[22] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. Wiley, 2012.

[23] Atsushi Fujimori, Svante Gunnarsson, and Mikael Norrlöf. *A gain scheduling control of nonlinear systems along a reference trajectory*. Linköpings universitet, 2005.

[24] Oliver M O'Reilly. *Intermediate dynamics for engineers: a unified treatment of Newton-Euler and Lagrangian mechanics*. Cambridge University Press, 2008.

[25] Milos Zefran. "Continuous methods for motion planning". In: *IRCS Technical Reports Series* (1996), p. 111.

[26] Oliver M O'Reilly. *Engineering Dynamics: A Primer*. Springer Verlag, 2001.

[27] Magnus Rudolph Hestenes. *Calculus of variations and optimal control theory*. Wiley New York, 1966.

[28] Dimitri P Bertsekas. "Nonlinear programming". In: (1999).

[29] Miguel Sousa Lobo et al. "Applications of Second-order Cone Programming". In: *Linear Algebra and its Applications* (1998), pp. 193–228.

[30] M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 1.21*. Apr. 2011.

[31] Michael Athans and Peter L Falb. *Optimal control: an introduction to the theory and its applications*. McGraw-Hill New York, 1966.

[32] Oskar Von Stryk. *Numerical Solution of Optimal Control Problems by Direct Collocation*. 1993. URL: `http://citeseer.ist.psu.edu/69756.html;http://www-m2.mathematik.tu-muenchen.de/~stryk/paper/1991-dircol.ps.gz`.

[33] C De Boor. "A Practical guide to splines (rev. Ed)(Applied Mathematical Sciences, Vol. 27) POD". In: (2001).

[34] Cleve B Moler. *Numerical computing with MATLAB*. Society for Industrial and Applied Mathematics, 2010.

[35] P.I. Corke. "A Robotics Toolbox for MATLAB". In: *IEEE Robotics and Automation Magazine* 3.1 (1996), pp. 24–32.

[36] Tamar Flash and Neville Hogan. "The coordination of arm movements: an experimentally confirmed mathematical model". In: *The journal of Neuroscience* 5.7 (1985), pp. 1688–1703.

[37] Yoji Uno, M Kawato, and R Suzuki. "Formation and control of optimal trajectory in human multijoint arm movement". In: *Biological cybernetics* 61.2 (1989), pp. 89–101.

[38] Brian D. O. Anderson and John B. Moore. *Linear Optimal Control*. Prentice-Hall, Inc., 1971.

[39] Karl J Aström and Bjorn Wittenmark. *Computer-controlled systems: theory and design*. Courier Dover Publications, 2011.

[40] Charles P Neuman and Vassilios D Tourassis. "Discrete dynamic robot models". In: *Systems, Man and Cybernetics, IEEE Transactions on* 2 (1985), pp. 193–204.

[41] Richard Ernest Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Vol. 7962. Princeton University Press, 1966.

[42] Dimitri P Bertsekas et al. *Dynamic programming and optimal control*. Vol. 1. 2. Athena Scientific Belmont, 1995.

[43] Felix R Gantmacher. *The theory of matrices. 1*. Vol. 131. Chelsea publishing company, 2000.

[44] Joao P Hespanha. "LQG/LQR controller design". In: *California, USA: University of California, Department of Electrical and Computer Engineering* (2007).

[45] Douglas J Leith and William E Leithead. "Survey of gain-scheduling analysis and design". In: *International Journal of Control* 73.11 (2000), pp. 1001–1025.

[46] Mikael Johansson. *Piecewise linear control systems - a computational approach*. Vol. 284. Electronic version available at http://springerlink.com/content/6e2xn7v5n4be/. Heidelberg, Germany: Lecture notes in control and information sciences, Springer Verlag, 2002.

[47] Alex A. Kurzhanskiy and Pravin Varaiya. *Ellipsoidal Toolbox*. Tech. rep. UCB/EECS-2006-46. EECS Department, University of California, Berkeley, 2006. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-46.html`.

[48] Gilbert Strang. *Introduction to linear algebra*. SIAM, 2003.

[49] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*. 50. Siam, 1997.

[50] Thomas Kailath. *Linear systems*. Vol. 1. Prentice-Hall Englewood Cliffs, NJ, 1980.

[51] Arash Hassibi and Stephen Boyd. "Quadratic stabilization and control of piecewise-linear systems". In: *American Control Conference, 1998. Proceedings of the 1998*. Vol. 6. IEEE. 1998, pp. 3659–3664.

[52] H. Asada and J.-J. E. Slotine. *Robot Analysis and Control*. 1st. John Wiley and Sons, 1986.

[53] Carlos Canudas De Wit, Georges Bastin, and Bruno Siciliano. *Theory of robot control*. Springer-Verlag New York, Inc., 1996.

[54] Jos F. Sturm. *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*. Optimization Methods and Software 11. 1999.

[55] Mikael Johansson and Anders Rantzer. "Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems". In: *IEEE Transactions on Automatic Control* 43 (1998), pp. 555–559.

[56] Pedro Reynoso-Mora and Masayoshi Tomizuka. "LQ-based Trajectory Tracking of Robotic Manipulators with "Near" Dynamically Feasible Time-optimal Trajectory". In: *ASME/ISCIE International Symposium on Flexible Automation*. 2012.

[57] Pedro Reynoso-Mora, Wenjie Chen, and Masayoshi Tomizuka. "On the Time-optimal Trajectory Planning and Control of Robotic Manipulators Along Predefined Paths". In: *American Control Conference*. 2013.

[58] Analog Devices (ADIS16400). 2009. URL: `http://www.analog.com/en/mems/imu/adis16400/products/product.html`.

[59] CompuGauge 3D. 2009. URL: `http://www.dynalog-us.com/solutions`.

Appendix A

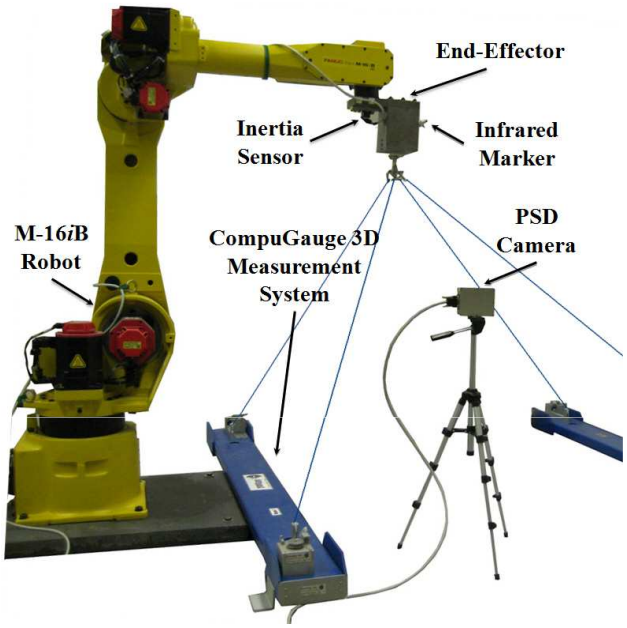
Experimental Setup for the 6-axis Industrial Manipulator

In this dissertation, we constantly evaluate the effectiveness of the proposed algorithms in experiments on a 6-axis industrial manipulator, courtesy of FANUC Corporation. Figure A.1(a) shows the robot together with additional hardware components. The model of this multi-joint manipulator is the M-16*i*B/20, which is a medium-size industrial robot capable of carrying objects with weights up to 20 kg at a maximum speed of 2000 mm/sec. We have attached to the robot's end-effector an "L"-shape payload, made from steel and weighting 18.37 kg. This robot is mainly used in high-speed applications such as: painting/coating, glueing and sealing, spot and arc welding, material handling, and water-jet cutting.

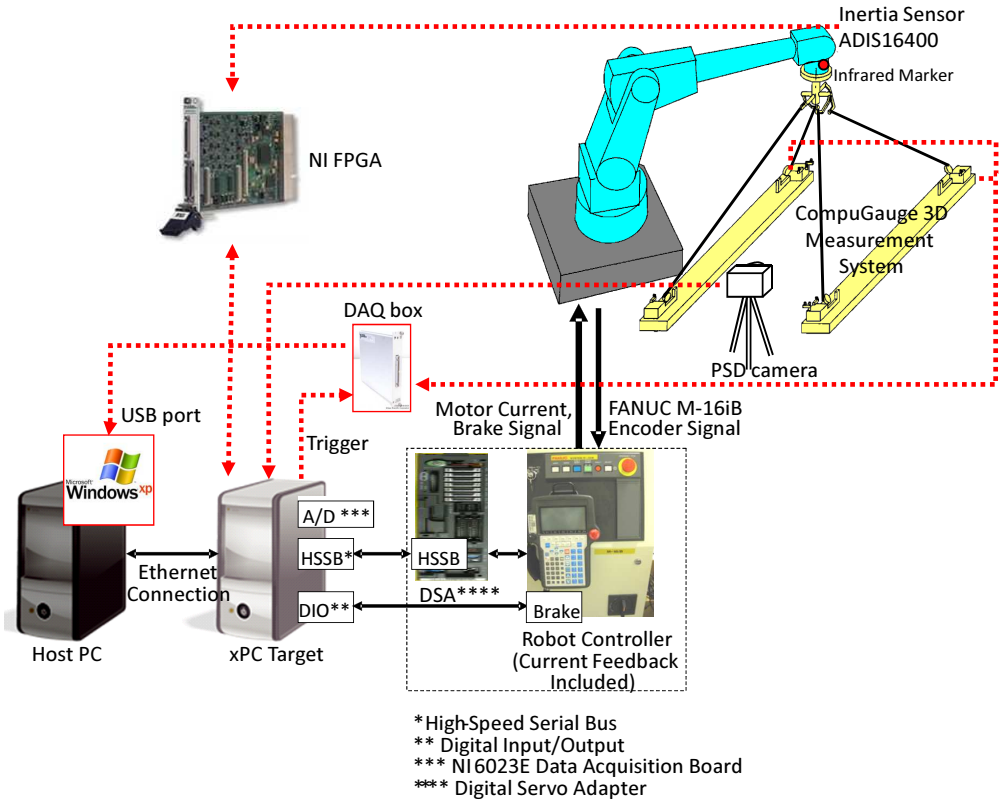
A.1 Hardware Configuration

Each motor of the M-16*i*B robot is equipped with a built-in encoder to measure joint position. The M-16*i*B robot commercial controller utilizes position and velocity feedback for control. The desired trajectory is programmed through the teach pendant, and then the commercial controller ensures the robot follows the desired motion. A disadvantage (from the research point of view) is that the structure of the commercial robot controller is fixed, which means it does not offer flexibility in the modification of control algorithms. Therefore, in order to implement our own control algorithms for research, the commercial robot controller must be bypassed. For rapid prototyping of new algorithms, it is convenient to design and implement control algorithms using MATLAB®.

The connection diagram of hardware components that allows to bypass the commercial robot controller is illustrated in Figure A.1(b). The control algorithms can be designed in MATLAB® on the host computer, and then implemented for experiments on the target computed. A digital servo adapter (DSA) is utilized to connect the real-time target computer with the FANUC robot controller. By using DSA, the motor torque commands are computed by our own control algorithms, running on the target computer, and then converted to the



(a) FANUC M-16*i*B robot with additional hardware



(b) Hardware connection diagram depicting the main components

Figure A.1: FANUC M-16*i*B industrial robot and its connection diagram of hardware.

current command for the FANUC robot controller to deliver such a current. Thus the commercial feedback controller is bypassed. Notice nonetheless that the FANUC robot controller still plays an important role in current feedback control and brake control. The current feedback loop is active in the FANUC robot controller to deliver the motor current that is commanded. The brake function is controlled with a digital input/output board installed on the target computer as depicted in Fig. A.1(b). The break turns off when the experiment starts and turns on after the experiment ends. In case of emergency (e.g., robot motion speed becomes too high), the FANUC robot controller automatically activates the brake of the motors. The operator can also activate the brake with the emergency button on the robot teach pendant.

Throughout this dissertation, a 3-axis accelerometer mounted on the robot's payload is used to monitor the effects of near time-optimal trajectories. For this purpose, we use an inertia measurement unit (IMU) from Analog Devices ADIS16400 [58], which includes a 3-axis accelerometer and a 3-axis gyroscope. Likewise, a three-dimensional position measurement system of the Cartesian coordinates (x, y, z) is utilized to measure the end-effector tool center point (TCP), which is ground truth for performance evaluation. This position measurement system, known as CompuGauge 3D [59], features a repeatability of 0.02 mm, accuracy of 0.15 mm, resolution of 0.01 mm, measurement space of $1.5 \times 1.5 \times 1.5$ m^3, tracking rate of up to 5 m/s, and the sampling frequency of up to 1000 Hz. Notice the CompuGauge is only used for performance evaluation of the Cartesian-space tracking error, i.e., the robot is entirely servoed using the motor encoders.

A.2 Real-time System

In real-time systems, an exact sampling period is required in order to avoid uncertainties caused by timing errors. However, most popular operating systems do not guarantee real-time operation due to unexpected interferences such as virus checking and event logging. To overcome such implementation issues, xPC Target™ is used, which is a toolbox of the MATLAB® product family. In this system, two computers are used, Host PC and Target PC, which are connected via Ethernet as depicted by the hardware diagram in Fig. A.1(b). MATLAB® is installed in the Host PC running on a Windows platform. The controller is designed in the Host PC by using Simulink®, another MATLAB® product, and MATLAB® coding functions as well. An advantage of using MATLAB® and xPC Target™ is that there is no need to rewrite code when changes in controller structure occur. After the controller design is done in the Host PC, the corresponding real-time code is loaded into the Target PC through Ethernet.

Notice that the Target PC has access to the motor position, the motor velocity, and the motor current through a high-speed serial buss (HSSB). When the control algorithms are running on the Target PC, the connection between the two computers is automatically disabled such that real-time execution of the algorithms in the Target PC is guaranteed. The sampling rates of all the sensor signals as well as the real-time controller implemented

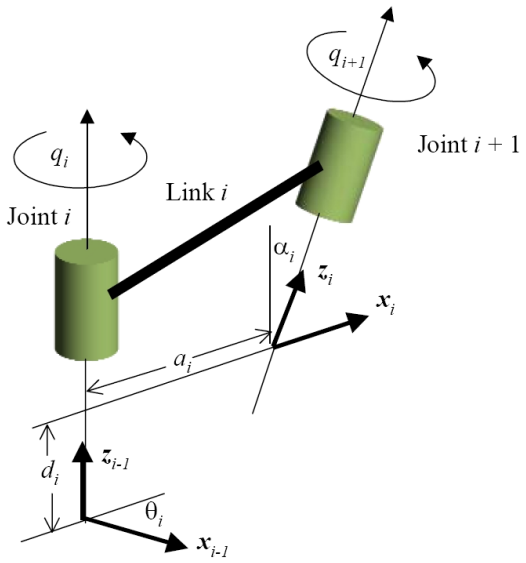


Figure A.2: Illustration of the Denavit-Hartenberg notation and parameters.

through MATLAB® xPC Target™ are set to 1 kHz.

A.3 Robot Kinematic Parameters

Throughout this dissertation, we use kinematics concepts extensively, namely, forward and inverse kinematics. We present the basic concepts on kinematics of robot manipulators and then use these concepts to perform kinematic modeling on the FANUC M-16*i*B robot. Two kinds of kinematics are recognized, forward and inverse kinematics, which map from joint to Cartesian and from Cartesian to joint spaces, respectively. In the forward and inverse kinematics, a suitable kinematic parametrization is given by the so-called Denavit-Hartenberg (DH) parametrization [1, 52]. This convention uses four parameters, known as the DH parameters, to completely parameterize the relative position and orientation of adjacent links. Figure A.2 illustrates the referred DH parameters.

A.3.1 DH Notations and Parameters

The DH procedure consists in characterizing the configuration of link i with respect to link $i-1$ by a 4×4 homogeneous transformation matrix, representing each link's coordinate system [52]. This matrix can be denoted by $\boldsymbol{T}_i^{i-1}$. The procedure for establishing coordinate frames at each link, as in Fig. A.2, is described by the following steps:

1. Name each joint starting with $1, 2, \ldots, n$, where n is the number of degrees of freedom.

2. Attach the z_{i-1} axis along the axis of rotation of the i-th joint.

3. Attach the x_i axis normal to the z_{i-1} axis (and of course to z_i) with direction from joint i to joint $i+1$.

4. The y_i axis is selected so that (x_i, y_i, z_i) is orthonormal and righthanded. Thus, it is common not to include this axis in the analysis.

The location of the zero reference frame is selected anywhere along the first axis of rotation. The last reference frame is attached arbitrarily to the last link, usually at the end-effector. However, the x_n axis must intersect the last axis of rotation at a right angle. The four parameters $(a_i, d_i, \alpha_i, \theta_i)$ in Fig. A.2 are the celebrated DH parameters for each pair of adjacent links. It can be shown that the homogeneous matrix relating adjacent frames $i-1$ and i is given by:

$$\boldsymbol{T}_i^{i-1} = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.1}$$

This matrix depends solely on the four DH parameters. For a revolute joint, α_i, a_i, and d_i are constant while θ_i varies. As a result, once α_i, a_i, and d_i are obtained for a specific link, the relative location of the corresponding adjacent links is only a function of θ_i.

A.3.2 Kinematic Modeling of FANUC M-16*i*B Robot

A suitable home position must be established first. The home configuration denotes the starting pose of the serial linkage, which we have chosen for the FANUC M-16*i*B robot as the posture shown in Fig. A.3. In order to obtain the DH parameters for the FANUC M-16*i*B robot, Figure A.4(a) provides accurate information on crucial lengths for establishment of

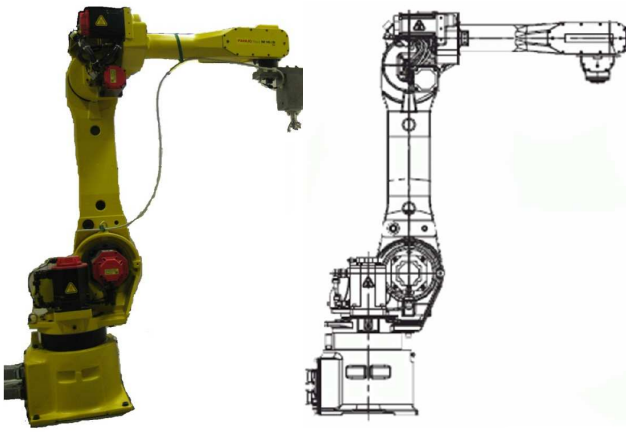
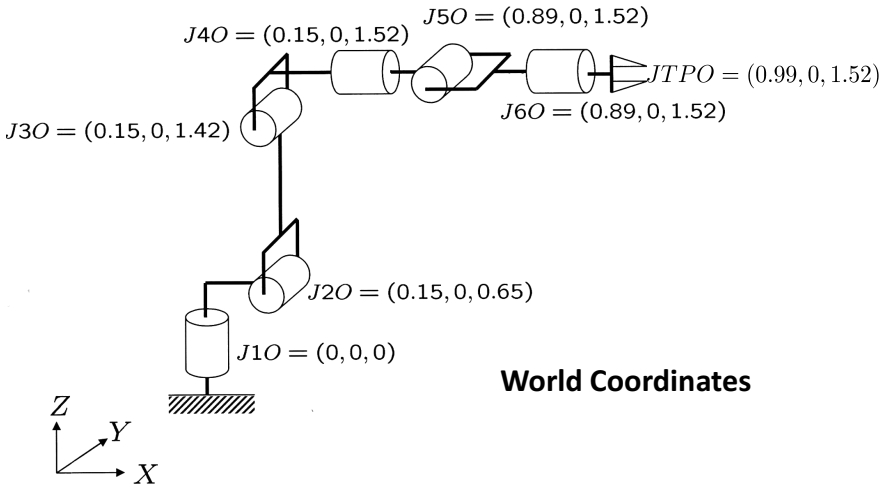
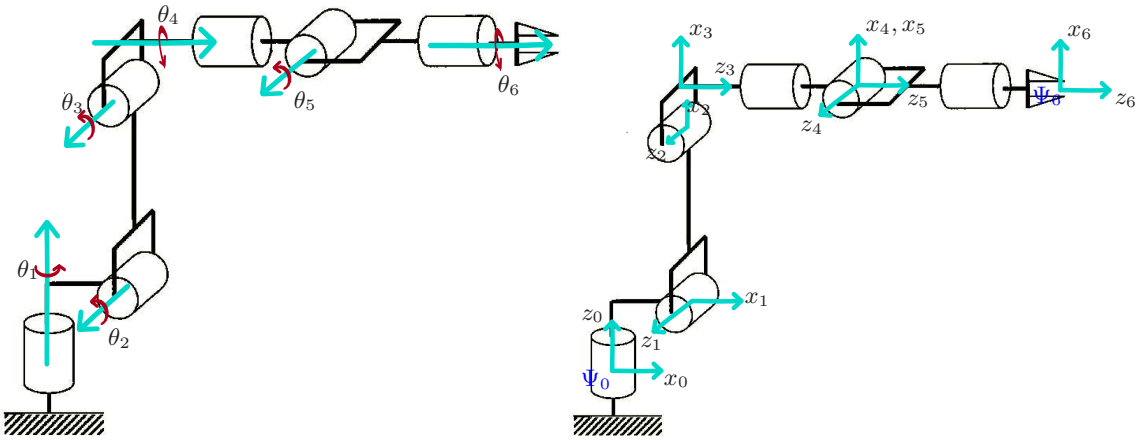


Figure A.3: Designated home position for FANUC M-16*i*B robot.



(a) Reference configuration for FANUC M-16iB robot



(b) Angle conventions (left) and attached frames (right)

Figure A.4: Crucial lengths, positive angle conventions, and attached coordinate frames for obtaining the DH parameters of FANUC M-16iB robot.

the DH parameters. It is important to explain the meaning of J_{10},J_{20},...,J_{06} in Fig. A.4(a); they represent the absolute Cartesian coordinates of each joint relative to the World reference frame. The World reference frame shall be denoted by Ψ_0.

An important step is to establish the positive-angle convention for each joint together with their respective axis of rotations. The corresponding positive-angle convention and rotation axes are shown in Fig. A.4(b). Finally, defining and attaching each of the reference frames to the corresponding links is done; the resulting frames are shown in Fig. A.4(b). These reference frames are attached to the corresponding links starting from the base link. Once these frames have been specified, the remaining part of kinematic modeling is straightforward. The DH parameters for FANUC M-16iB are obtained and presented on Table A.1.

i	α_i	a_i	θ_i	d_i
1	$\pi/2$	0.15	0	0.65
2	0	0.77	$\pi/2$	0
3	$\pi/2$	0.1	0	0
4	$-\pi/2$	0	0	0.74
5	$\pi/2$	0	0	0
6	0	0	0	0.10

Table A.1: DH Parameters for FANUC M-16iB robot