Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2018

Sensor-Based Trajectory Planning in Dynamic Environments

Andreas Westerlund

Master of Science Thesis in Electrical Engineering

Sensor-Based Trajectory Planning in Dynamic Environments

Andreas Westerlund

LiTH-ISY-EX--18/5164--SE

Supervisor: **Hamed Haghshenas**
ISY, Linköping University
Giacomo Spampinato
ABB Robotics

Examiner: **Johan Löfberg**
ISY, Linköping University

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Motion planning is central to the efficient operation and autonomy of robots in the industry. Generally, motion planning of industrial robots is treated in a two-step approach. First, a geometric path between the start and goal position is planned where the objective is to achieve as short path as possible together with avoiding obstacles. Alternatively, a pre-defined geometric path is provided by the end user. Second, the velocity profile along the geometric path is calculated accounting for system dynamics together with other constraints. This approach is computationally efficient, but yield sub-optimal solutions as the system dynamics is not considered in the first step when the geometric path is planned.

In this thesis, an alternative to the two-step approach is investigated and a trajectory planner is designed and implemented which plans both the geometric path and the velocity profile simultaneously. The motion planning problem is formulated as an optimal control problem, which is solved by a direct collocation method where the trajectory is parametrised by splines, and the spline nodes and knots are used as optimization variables.

The implemented trajectory planner is evaluated in simulations, where the planner is applied to a simple planar elbow robot and ABB's SCARA robot IRB 910SC. Trade-off between computation time and optimality is identified and the results indicate that the trajectory planner yields satisfactory solutions. On the other hand, the simulations indicate that it is not possible to apply the proposed method on a real robot in real-time applications without significant modifications in the implementation to decrease the computation time.

Acknowledgments

First of all, I would like to thank ABB Robotics for giving me the opportunity to write this thesis. A special thanks to my supervisor at ABB Robotics, Giacomo Spampinato, for always taking time to answer my questions and showing great interest in my work. I would also like to thank Mikael Norrlöf for all the endless discussions and the great amount of support and encouragement I have received during this period. For this, I will always be grateful.

To my examiner, Johan Löfberg, and my supervisor at Linköping University, Hamed Haghshenas, thank you for the thorough feedback on my report and valuable inputs.

Finally, I would like to thank my family and friends for all their encouragement and support throughout the years.

Västerås, June 2018
Andreas Westerlund

Contents

1

Introduction

In the industry of robots, the motion planning is central to the operation of autonomy. It involves the generation of trajectories to move a robot manipulator from a start state to a goal state with the objective to minimize transition time or energy consumption while satisfying constraints like system dynamics, peak velocity and acceleration, components life-time as well as dynamically changing collision constraints. As such, there is a distinction between *motion planning* and *control* in that the former generates a feasible reference trajectory and the latter tracks that trajectory by introducing torques on the robot axes through current and voltage in the motors.

Traditionally, motion planning has been treated in a two-step approach, *path planning* and *trajectory generation* [6, 9]. Path planning concerns the search of time-independent continuous sequence of configurations, robot joint poses, from a start state to a goal state in order to achieve as short path as possible together with avoiding obstacles. In essence, path planning yields a collision-free geometric path without velocity profile. Alternately, a geometric path is given by the end user. Afterwards, in the trajectory generation, the planner calculates the optimal velocity profile along the path accounting for system dynamics and other constraints. As such, after the trajectory generation a trajectory in the state space is produced rather than just a geometric path. The two-step approach is an efficient procedure that can operate in real-time and in many scenarios it is sufficient due to the geometric constraints specified by the end user. However, the approach is sub-optimal since in the first step, the path planning, the procedure does not consider system dynamics.

An alternative approach is *trajectory planning* which finds path and velocity profile simultaneously obeying system dynamics as well as avoiding obstacles and other specified constraints. Trajectory planning is computationally more expensive but allows optimal solutions.

1.1 Problem Formulation

The objective of this master thesis is to investigate new techniques to dynamically optimize the trajectory planning for industrial robots in a dynamic changing environment. The trajectory planner should be able to run in real-time and take into account diversified kinds of constraints like system dynamics, energy, peak accelerations and velocities, components life-time as well as dynamically changing collision conditions.

The trajectory planner will assume that the states of the obstacles and targets, as well as the states of the system can be obtained through measurements along with state estimations. Moreover, it will assume the existence of a low-level controller that, given a feasible trajectory, computes the control values required to realize the trajectory.

Furthermore, the target applications for the trajectory planner are pick-and-place scenarios and conveyor tracking for industrial robot manipulators. Consequently, the trajectory planner should be able to generate feasible trajectories that allow reaching and tracking of moving targets.

1.2 Related Work

An approach to achieve time optimal trajectories within the *Model Predictive Control* (MPC) framework is described in [3]. The method is called *Time Optimal MPC* (TOMPC) and minimizes the transition time in a two layer optimization routine. The inner loop consists of a traditional MPC with quadratic objective function and the outer loop incrementally adjusts the prediction horizon in order to determine the shortest feasible horizon, resulting in a time optimal solution. In the vicinity of the target state the method behaves like a traditional MPC, due to a fixed lower bound on the time horizon, in order to guarantee stability. The initial guess of the prediction horizon is vital in order to achieve low runtime since it determine the number of outer loop iterations.

A hierarchical approach to the time optimal control in a MPC setting is presented in [8]. The framework formulates the time optimal control problem as a standard multi-objective problem, where the objectives are assigned with different level of priority. It states that the most vital thing, without violating the constraints, is to reach the target state in a predefined upper bound time horizon. Then incrementally reduce the time horizon and try, if possible, to reach the target state, until a predefined lower bound time horizon is attained. By this, each state between the lower and upper bound will be as close as possible to the goal state, yielding a time optimal trajectory.

O. Khatib et al. formed a basis for an effective framework, elastic band, to deal with real-time collision-free path planning [13]. An initial path between a start and end goal is considered as a deformable elastic band. The initial elastic band is deformed in real-time by virtual forces originated from the surrounding obstacles resulting in a collision-free path. The use of a deformable elastic band enables computational time appropriate for real-time applications as the path is

as short as possible, considering the obstacles, without employment of a global search. On the other hand, the framework does not consider system dynamics or optimal time transitions.

The elastic band framework was extended in [14], [16] introducing the *Timed Elastic Band* (TEB) where the time interval between poses are included into a joint trajectory. The concept of timed elastic bands enables not only geometric constraints on the path but also system dynamics constraints and optimal time transitions. The TEB framework was employed in a time optimal MPC setting in [15]. The TEB approach in an MPC context formulates the fixed horizon optimal control problem for point to point transitions as a *Nonlinear Program* (NLP). The NLP is solved by *Sequential Quadratic Programming* (SQP) based on the line-search SQP described in [12]. The approach looks promising as a real-time trajectory planner, but does not utilize the sparse structure of the NLP in order to reduce the computation time.

In [2], the TEB based time optimal nonlinear MPC is applied yielding real-time performance by utilizing the sparse structure of the optimization problem. The obstacle avoidance is included in the optimization problem by introducing penalizing terms in the objective function for points in the trajectory close to obstacles as well as hard constraints at closer distances to obstacles. In a vicinity close to the target state the objective function switches to a quadratic cost function to ensure asymptotic stability upon convergence to the goal state. The planning framework uses multiple starting guesses to initiate multiple planning procedures in parallel to account for the local optimums in the non-convex optimization problem.

C. Rösmann et al. extended the planning framework in [15] by introducing dynamic local time intervals between poses in the joint trajectory [17]. In the same paper, an approximate unconstrained optimization procedure to the TEB framework applied in an MPC setting is also introduced. The original optimization problem is reformulated by expressing the equality and inequality constraints as quadratic penalty functions in the objective, yielding an unconstrained least-square problem. The unconstrained optimization problem avoids Lagrange multipliers which reduces the dimension of the Hessian in the NLP significantly. Levenberg-Marquardt algorithm is utilized in order to solve the least-square problem [12]. The approximation and the original formulation are compared in a convergence analysis under limited computational resources which concludes that the least-square approximation matches nearly 90% of the optimal solution within a very short computational time. Both the global and the local time interval approach are also compared in the least-square framework under limited computational time, concluding that the local time interval procedure is better suited for higher order systems. However, the global time interval approach is more satisfactory in higher trajectory resolution scenarios due to the growth of optimization variables in the local time interval procedure.

The solution to many time optimal control problems are either bang-bang or small finite set of piece-wise constant controls. Therefore the problem formulation in [15] includes a lot of redundant control parameters since the number of effective control interventions at which the control signal changes is significantly

smaller than the temporal discretization of the MPC. The approach in [19] explicitly achieves a minimal intervention control, thereby minimal computational burden, by first applying multiple shooting and second, the temporal discretization is adapted [4]. In [20], C. Rösmann et al. extended the minimum intervention control algorithm even further to reduce the number of required NLP solutions by analyzing the previous solution in each iteration.

1.3 Approach

To construct a trajectory planner that is able to operate in real-time and take diversified kinds of constraints into account, this work will be based on the TEB framework in a time optimal MPC setting described in [15], together with the obstacle avoidance and switching strategies applied in [2]. The trajectory planner will not be limited to industrial robots, but the main focus will be to achieve a reactive trajectory planner for industrial robots with low computational effort. In order to reduce the number of optimization variables in the underlying NLP, the control signal in the TEB will be eliminated by expressing the control constraints as functions of the states by utilizing the system dynamics.

Both the local and global time interval approach will be investigated, utilizing direct collocation, to determine the most beneficial to industrial robots. The underlying optimization problem will be solved by the interior-point method and will be implemented by employing IPOPT [23].

2

Preliminaries

This chapter provides some mathematical preliminaries as a base for the next chapters. First, the main concepts and notations in robot modelling, together with descriptions of two robots later employed in the simulations, is provided. The robot modelling part is based on [22]. Next, the trajectory planning is formed as an optimal control problem and some main approaches to address these problems based on [5] are described. Finally, the framework of timed elastic bands is described.

2.1 Robot Modelling

Robot modelling is in essence concerned with formulations of various coordinate systems to represent the positions and orientations of rigid bodies together with transformations between these coordinate systems [22]. Two vital coordinate systems are the *configuration space* and the *operational space*. A configuration of a robot is a complete specification of the location of every point of the robot. In practice, a configuration is represented by the *joint* variables. A joint variable $\mathbf{q}$ represents the interconnection between two links and is typically either rotary or prismatic. The set of all achievable configurations is called the configuration space. The operational space constitutes the space where the manipulator operates in and is composed of Cartesian coordinates $\mathbf{p}$, including both positions and orientations of the end-effector of the robot. The *workspace* is the total volume reachable by the end-effector which is a subset of the operational space.

In this work, only serial manipulators are considered. As such, the connection between the configuration space and the operational space can be expressed by a homogeneous transformation, i.e. combining the operations of rotation and translation into a single matrix multiplication. The problem of determining the position and orientation of the end-effector from the joint variables is called the

forward kinematics. In other words, finding a function $\chi(\mathbf{q})$ that satisfies

$$\mathbf{p} = \chi(\mathbf{q}). \tag{2.1}$$

The inverse problem of determining the joint variables in terms of the end-effector's position and orientation is called the *inverse kinematics* and is expressed as

$$\mathbf{q} = \chi^{-1}(\mathbf{p}). \tag{2.2}$$

A position and orientation of the end-effector can typically be obtained by multiple configurations. Therefore the inverse kinematics concerns a more complex problem since the solution might not exist or it may not be unique whereas the forward kinematics problem always has a unique solution. For simple examples, there are systematic techniques that exploit the kinematic structure in order to yield closed form solutions. Generally, for more complex robot types, numerical approaches are applied.

The velocity relationship between the configuration space and the workspace is defined by the Jacobian of the forward kinematics function $\chi(\mathbf{q})$. Mathematically, the relationship between the joint velocities and the linear and angular velocities of the end-effector is

$$\dot{\mathbf{p}} = J(\mathbf{q})\dot{\mathbf{q}} \tag{2.3}$$

where $J(\mathbf{q})$ is the Jacobian of $\chi(\mathbf{q})$. The inverse relationship, i.e. connecting the linear and angular velocities of the end-effector with the joint velocities, is obtained by inverting the Jacobian matrix. The analytical Jacobian does not necessarily have full rank for all configurations. Configurations where the Jacobian drops rank are called singular configurations and at singularities bounded end-effector velocities may correspond to unbounded joint velocities and certain directions of motion may be unattainable. Therefore, identifying singular configurations is an important aspect of robot modelling. The relationship between end-effector accelerations and the joint accelerations are given by

$$\ddot{\mathbf{p}} = J(\mathbf{q})\ddot{\mathbf{q}} + \dot{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.4}$$

which is obtained by differentiating (2.3).

So far, the robot motion has been described by the kinematic equations without taking the torques generating the motion into account. To describe the dependency between torques and motion the Euler-Lagrange equations or the Newton-Euler formulations are utilized. The dynamics are expressed in the configuration space and the necessary joint torques $\boldsymbol{\tau}(t)$ to obtain a desired trajectory $(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$ are given by

$$M(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + C(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + G(\mathbf{q}(t)) = \boldsymbol{\tau}(t) \tag{2.5}$$

which is called the *inverse dynamic model*. The first term includes $M(\mathbf{q}(t))$ which is the inertia matrix, the second term includes $C(\mathbf{q}(t), \dot{\mathbf{q}}(t))$ which describes the Coriolis and centrifugal forces and the last term $G(\mathbf{q}(t))$ describes the external forces such as gravity.

2.1.1 Planar Elbow Robot

A two axes robot of *Selective Compliance Articulated Robot Arm* (SCARA) type called *planar elbow robot*, shown in Figure 2.1, which operates in a two-dimensional plane is used to simulate and test the trajectory planner.

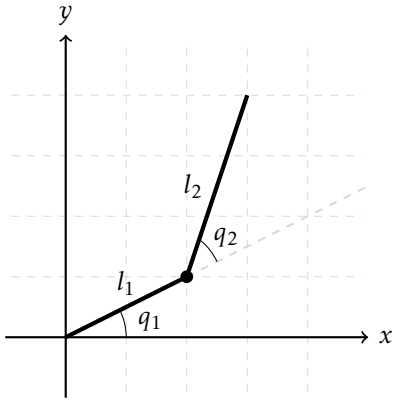


Figure 2.1: *Diagram of the planar elbow robot.*

The robot consists of two links with length l_1 and l_2, mass m_1 and m_2 and moment of inertia I_1 and I_2 respectively. Two revolute joint angles q_1 and q_2 are employed to model the motion of the robot and the forward kinematics are given by

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \chi(q_1, q_2) = l_1 \begin{pmatrix} \cos q_1 \\ \sin q_2 \end{pmatrix} + l_2 \begin{pmatrix} \cos q_1 & -\sin q_1 \\ \sin q_1 & \cos q_1 \end{pmatrix} \begin{pmatrix} \cos q_2 \\ \sin q_2 \end{pmatrix} \tag{2.6}$$

and the inverse kinematics are given by

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \chi^{-1}(p_1, p_2) = \begin{pmatrix} \frac{\arctan \frac{p_2}{p_1} + \arccos(l_1^2 - l_2^2 + p_1^2 + p_2^2)}{2l_1\sqrt{p_1^2+p_2^2}} \\ \frac{\arccos(l_1^2 + l_2^2 - p_1^2 - p_2^2) - \pi}{2l_1 l_2} \end{pmatrix}. \tag{2.7}$$

The joint variables are bounded to $-2\pi \leq q_1 \leq 2\pi$ and $-\pi \leq q_2 \leq \pi$ where q_1 is bounded to a greater interval in order to avoid discontinuity in the simulations. The inverse cosine terms yield two distinct solutions (q_1^1, q_2^1) and (q_1^2, q_2^2) for every pair of (p_1, p_2). Additionally, since q_1 is bounded to a larger interval, there will exist two extra solutions which are obtained by $(q_1^1 - 2\pi, q_2^1)$ and $(q_1^2 - 2\pi, q_2^2)$. In summary, for every pair of (p_1, p_2) it will exist four candidate configurations. The inverse dynamic model for the planar elbow robot is given by

$$M(q_1, q_2)\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + C(q_1, q_2, \dot{q}_1, \dot{q}_2)\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} + \begin{pmatrix} f_1 \dot{q}_1 \\ f_2 \dot{q}_2 \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} \tag{2.8}$$

where τ_1 and τ_2 are the torques applied on each link respectively, f_1 and f_2 are

additional linear friction terms. The inertia matrix is

$$M(q_1, q_2) = \begin{pmatrix} \frac{1}{4}m_1 l_1^2 + m_2(l_1^2 + \frac{1}{4}l_2^2 + l_1 l_2 \cos q_2 + I_1 + I_2) & m_2(\frac{1}{4}l_2^2 + \frac{1}{2}l_1 l_2 \cos q_2 + I_2) \\ m_2(\frac{1}{4}l_2^2 + \frac{1}{2}l_1 l_2 \cos q_2 + I_2) & \frac{1}{4}m_2 l_2^2 + I_2 \end{pmatrix}$$

and the Coriolis and centrifugal forces are given by

$$C(q_1, q_2, \dot{q}_1, \dot{q}_2) = \begin{pmatrix} h(q_1, q_2)\dot{q}_2 & h(q_1, q_2)\dot{q}_1 + h(q_1, q_2)\dot{q}_2 \\ -h(q_1, q_2)\dot{q}_1 & 0 \end{pmatrix}$$

where

$$h(q_1, q_2) = -\frac{1}{2}m_2 l_1 l_2 \sin q_2.$$

The values of the parameters used in the simulations are listed in Table 2.1.

Table 2.1: *Parameter values for the planar elbow robot used in the simulations.*

l_1	$1\ m$
l_2	$1\ m$
m_1	$1\ kg$
m_2	$1\ kg$
I_1	$0.5\ kgm^2$
I_2	$0.5\ kgm^2$
f_1	$1.5\ Nms/rad$
f_2	$1.5\ Nms/rad$

2.1.2 IRB 910SC (SCARA) Robot

ABB's IRB 910SC (SCARA), see Figure 2.2, is also used in the simulations to evaluate and analyze the behaviour of the trajectory planner on a more complex robot. The robot operates in three dimensions and consists of four joints represented by one prismatic joint q_3 and three rotary joints q_1, q_2, q_4, see Figure 2.3. Notice, the prismatic joint q_3 is defined to be positive along the positive z-axis and q_4 describes the orientation of the tool.

The position and orientation of the end-effector are denoted $\mathbf{p} = (p_1, p_2, p_3, p_4)^T$ where the first three variables represent the Cartesian position in the three-dimensional space and p_4 express the orientation with respect to the fixed world reference frame. Consequently, the forward kinematics are given by

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \chi(q_1, q_2, q_3, q_4) = \begin{pmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ q_3 + l_0 - l_3 \\ q_4 + q_1 + q_2 \end{pmatrix} \tag{2.9}$$

Figure 2.2: *ABB's IRB 910SC (SCARA). Photo courtesy of ABB.*

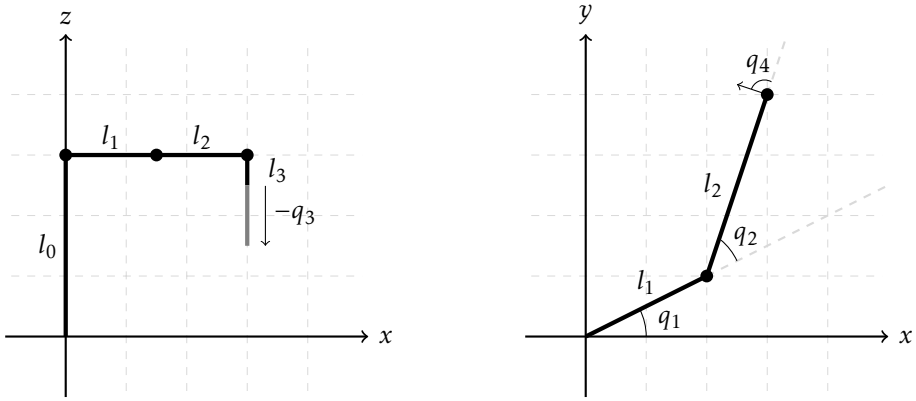


Figure 2.3: *Diagram of the SCARA robot in the world reference frame.*

and the inverse kinematics are given by

$$\begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \chi^{-1}(p_1, p_2, p_3, p_4) = \begin{pmatrix} \frac{\arctan\frac{p_2}{p_1} + \arccos(l_1^2 - l_2^2 + p_1^2 + p_2^2)}{2l_1\sqrt{p_1^2+p_2^2}} \\ \frac{\arccos(l_1^2 + l_2^2 - p_1^2 - p_2^2) - \pi}{2l_1 l_2} \\ p_3 - l_0 + l_3 \\ p_4 - q_1(p_1, p_2) - q_2(p_1, p_2) \end{pmatrix} \tag{2.10}$$

where $q_1(p_1, p_2)$ and $q_2(p_1, p_2)$ are row 1 and row 2, respectively, in order to simplify the expression. In the simulations, the joint variables are bounded to $-2\pi \leq q_1 \leq 2\pi$, $-\pi \leq q_2 \leq \pi$ and $-(l_0 + l_3) \leq q_1 \leq 0$. Analogous to the inverse kinematics of the planar elbow robot, for each $\mathbf{p}$ there exist four candidate configurations. The inverse cosine term yields two distinct solutions $(q_1^1, q_2^1, q_3^1, q_4^1)$ and

$(q_1^2, q_2^2, q_3^2, q_4^2)$, the additional two solutions are obtained by $(q_1^1 - 2\pi, q_2^1, q_3^1, q_4^1 - 2\pi)$ and $(q_1^2 - 2\pi, q_2^2, q_3^2, q_4^2 - 2\pi)$.

With knowledge of the mass, center of mass and inertia for each arm and tool, the inverse dynamic model

$$M(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + C(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + G(\mathbf{q}(t)) = \boldsymbol{\tau}(t) \tag{2.11}$$

can be derived.

2.2 Time Optimal Control Problem

Trajectory planning which determines both a path and a velocity profile simultaneously obeying diverse constraints such as system dynamics, collision conditions together with peak velocities and accelerations, falls under the category of problems called optimal control problems. Mathematically, it is expressed as

$$\begin{aligned} \min_{\mathbf{u}(t)} \quad & \phi(t_f, \mathbf{x}(t_f)) + \int_{t_i}^{t_f} f_0(t, \mathbf{x}(t), \mathbf{u}(t))dt \\ s.t. \quad & \begin{cases} \dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(t_i) \in S_i, \quad \mathbf{x}(t_f) \in S_f \\ \mathbf{x}(t) \in X_t, \quad \mathbf{u}(t) \in U_t(\mathbf{x}(t)) \end{cases} \end{aligned} \tag{2.12}$$

where t_i and t_f are the initial and terminal times, S_i and S_f are the initial and terminal manifolds, X_t and $U_t(\mathbf{x}(t))$ are the state constraint set respectively the control constraint set, $\phi(t_f, \mathbf{x}(t_f))$ is the terminal cost which penalizes deviation from a desired terminal state, $f_0(t, \mathbf{x}(t), \mathbf{u}(t))$ is the cost function and $f(t, \mathbf{x}(t), \mathbf{u}(t))$ is the system dynamics.

Generally, there are three main approaches to address optimal control problems; dynamic programming, indirect methods and direct methods [5]. Dynamic programming yields sufficient conditions for optimality by using the principle of optimality of subarcs to generate a feedback control. In the discrete time case this emerges to a backwards recursion problem and for the continuous time case this leads to the Hamilton-Jacobi-Bellman (HJB) equation. The HJB equation is a partial differential equation in the state space which is not necessary solvable as it requires the optimal cost function to be sufficiently smooth.

Indirect methods give necessary but not sufficient conditions for optimality by exploring local properties around the optimal control and state trajectories. This leads to the Pontryagin Minimum Principle which emerge to a Two Point Boundary Value Problem (TPBVP) in ordinary differential equations. Generally, the TPBVP must be solved numerically and, as it only yields necessary conditions for optimality, the obtained candidates must be further examined to provide the optimal solution. A common numerical method for solving the TPBVP is boundary condition iteration, an iterative method that finds the correct values of the unspecified initial/terminal conditions in the TPBVP. The major drawback is that

the ordinary differential equations in the TPBVP may embrace strong nonlinearity and instability causing numerical difficulties.

Direct methods discretize the optimal control problem (2.12) in order to transform the problem into a finite dimensional NLP which can be solved by numerical optimization methods. Common ways of transforming the optimal control problem into a NLP are collocation, single shooting and multiple shooting. In collocation, both the controls and the states are discretized on a fine grid. Normally, a piecewise constant approximation of the controls and the states are applied together with forward Euler approximation of the differential operator. In single shooting, only the control signals are discretized and the states are obtained by numerical simulation of the dynamic model. The states are then considered as dependent variables. Multiple shooting is closely related to single shooting. The difference is that in multiple shooting the numerical simulation is applied on each time step independently yielding trajectory pieces which later on form a complete trajectory.

In this work, the focus is on time optimal control problem for industrial robots, formulated as

$$\begin{aligned} \min_{\boldsymbol{\tau}(t)} \quad & t_f \\ s.t. \quad & \begin{cases} M(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + C(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + G(\mathbf{q}(t)) = \boldsymbol{\tau}(t) \\ \boldsymbol{\tau}_- \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_+ \\ \dot{\mathbf{q}}_- \leq \dot{\mathbf{q}}(t) \leq \dot{\mathbf{q}}_+ \\ \dddot{\mathbf{q}}_- \leq \dddot{\mathbf{q}}(t) \leq \dddot{\mathbf{q}}_+ \\ \mathbf{q}(t) \in Q \\ \mathbf{p}(0) = \mathbf{p}_i, \quad \dot{\mathbf{p}}(0) = \dot{\mathbf{p}}_i \\ \mathbf{p}(t_f) = \mathbf{p}_{t_f}, \quad \dot{\mathbf{p}}(t_f) = \dot{\mathbf{p}}_{t_f} \end{cases} \end{aligned} \tag{2.13}$$

where the objective is to minimize the total transition time between the start and end state subject to system dynamics, torque bounds, joint velocity bounds, joint jerk bounds and collision constraints.

2.3 Timed Elastic Band

In the *Timed Elastic Band* (TEB) framework, a direct method is applied to solve the time optimal control problem. The state vector $\mathbf{x}(t) \in \mathbb{R}^p$ and the input vector $\mathbf{u}(t) \in \mathbb{R}^q$ are discretized

$$[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$$
$$[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$$

with finite time differences

$$[\Delta T_1, \Delta T_2, \ldots, \Delta T_{n-1}].$$

The ΔT_k values are allowed to vary in order to deform in time and space. The differential operator is approximated by forward Euler

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta T_k}$$

yielding the discretized system dynamics

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta T_k} = f(\mathbf{x}_k, \mathbf{u}_k).$$

The sequence of states and time intervals are combined into a single set

$$\mathcal{B}_l := \{\mathbf{x}_1, \mathbf{u}_1, \mathbf{\Delta T}_1, \mathbf{x}_2, \mathbf{u}_2, \mathbf{\Delta T}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{u}_{n-1}, \mathbf{\Delta T}_{n-1}, \mathbf{x}_n\}$$

called TEB with local time discretization, where $\mathcal{B}_l \in \mathbb{R}^\rho$ with $\rho = np + (n-1)(q+1)$. The discretized time optimal control problem can be formulated as a NLP:

$$\min_{\mathcal{B}_l} \quad \sum_{k=1}^{n-1} \Delta T_k$$

$$s.t. \quad \begin{cases} \mathbf{x}_1 = \mathbf{x}_i,\ \mathbf{x}_n = \mathbf{x}_f,\ \Delta T_k > 0 \\ \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta T_k} = f(\mathbf{x}_k, \mathbf{u}_k) \\ g(\mathbf{x}_k, \mathbf{u}_k) \geq \mathbf{0} \qquad\qquad (k = 1, 2, \ldots, n-1) \end{cases}$$

where $\mathbf{x}_i$ is the initial state, $\mathbf{x}_f$ is the final state, $g(\mathbf{x}_k, \mathbf{u}_k)$ is the inequality constraints and the objective is to minimize the sum of the local time intervals ΔT_k.

Since the total transition time can be partitioned in different ways by the local time intervals the NLP is under-constrained. If the NLP solver does not handle under-constrained problems well an additional regularization term $r : \mathbb{R}^+ \to \mathbb{R}$ weighted by $\lambda \in \mathbb{R}^+$ can be added to the objective function. The NLP is now formulated as

$$\min_{\mathcal{B}_l} \quad \sum_{k=1}^{n-1} [\Delta T_k + \lambda r(\Delta T_k)]$$

$$s.t. \quad \begin{cases} \mathbf{x}_1 = \mathbf{x}_i,\ \mathbf{x}_n = \mathbf{x}_f,\ \Delta T_k > 0 \\ \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta T_k} = f(\mathbf{x}_k, \mathbf{u}_k) \\ g(\mathbf{x}_k, \mathbf{u}_k) \geq \mathbf{0} \qquad\qquad (k = 1, 2, \ldots, n-1). \end{cases}$$

In contrast to TEB with local time discretization $\mathcal{B}_l$, the finite time differences can be defined as $\Delta T_k := \Delta T$ which yield the single set

$$\mathcal{B}_g := \{\mathbf{x}_1, \mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{u}_{n-1}, \mathbf{x}_n, \Delta T\}$$

called TEB with global time discretization, where $\mathcal{B}_g \in \mathbb{R}^\rho$ with $\rho = np + (n-1)q + 1$. Note that in $\mathcal{B}_g$ there is only one finite time difference to optimize, which reduces the number of optimization variables. The discretized time optimal control

problem with global time interval can be formulated as

$$\min_{\mathcal{B}_g} \quad (n-1)\Delta T$$

$$s.t. \quad \begin{cases} \mathbf{x}_1 = \mathbf{x}_i,\ \mathbf{x}_n = \mathbf{x}_f,\ \Delta T > 0 \\ \frac{\mathbf{x}_{k+1}-\mathbf{x}_k}{\Delta T} = f(\mathbf{x}_k, \mathbf{u}_k) \\ g(\mathbf{x}_k, \mathbf{u}_k) \geq \mathbf{0} \qquad\qquad (k = 1, 2, \ldots, n-1). \end{cases}$$

Notice, if $r(\Delta T_k) = \Delta T_k^2$ and $\lambda \gg 1$ the objective is approximately $\sum_{k=1}^{n-1} \Delta T_k^2$. Thus, as described in [20], if T^* is the optimal total transition time then the optimal time intervals are $\Delta T_k^* = \frac{T^*}{n}$ and the solution exactly corresponds to the TEB with global time interval.

3

Spline-Based Trajectory Planner

In this chapter, a spline-based trajectory planner is introduced. First, a spline-based framework is presented which serves as an alternative to the TEB approach. Then, the spline-based approach is applied to industrial robots and the problems of collision avoidance and tracking of moving targets are addressed. Finally, the software implementation is described.

3.1 Spline-Based Timed Elastic Band

In the original TEB formulation, the differential operator is approximated by forward Euler to transform the continuous time problem into a finite dimensional NLP. Hence, a large time step ΔT_k yields a large approximation error. Therefore, in order for the approximation error to be sufficiently small and to guarantee a high resolution, a large band length n has to be used. Obviously, the computational time is strongly correlated with the number of optimization variables. In this work, an alternative approach based on splines is presented. The approach assumes that there exists a function $\mathbf{u}(t) = \tilde{f}(\mathbf{x}(t))$ that given the state evolution $\mathbf{x}(t)$ uniquely reconstruct the applied control signal $\mathbf{u}(t)$. Particularly, there exists a unique solution of the problem of inverting $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$ with respect to $\mathbf{u}(t)$. Consequently, in the optimal control problem, the control signal can be substituted by $\tilde{f}(\mathbf{x}(t))$.

The overall time interval is partitioned into n subintervals

$$\begin{aligned} 0 = t_1 \leq t_1 + \Delta T_1 &= t_2, \\ t_2 \leq t_2 + \Delta T_2 &= t_3, \\ &\vdots \\ t_{n-1} \leq t_{n-1} + \Delta T_{n-1} &= t_n. \end{aligned}$$

and let $\mathbf{q}(t) \in \mathbb{R}^j$ consist of splines of degree p with knot vector

$$\mathbf{t} = (t_1, t_2, \ldots, t_n)$$

and nodes

$$\begin{aligned} \mathbf{q}(t_k) &= \mathbf{q}_k \\ \dot{\mathbf{q}}(t_k) &= \dot{\mathbf{q}}_k \\ &\vdots \\ \mathbf{q}^{(p)}(t_k) &= \mathbf{q}_k^{(p)} \end{aligned} \tag{3.1}$$

for $k = 1, 2, \ldots, n$. Hence,

$$\mathbf{q}(t) = \mathbf{s}_k(t) \quad \text{for} \quad t_k \leq t < t_{k+1}$$

where

$$\mathbf{s}_k(t) = \frac{\mathbf{q}_k^{(p)}}{p!}(t - t_k)^p + \frac{\mathbf{q}_k^{(p-1)}}{(p-1)!}(t - t_k)^{(p-1)} + \ldots + \dot{\mathbf{q}}_k(t - t_k) + \mathbf{q}_k \tag{3.2}$$

with continuity conditions

$$\begin{aligned} \mathbf{s}_k(t_{k+1}) &= \mathbf{s}_{k+1}(t_{k+1}) \\ \dot{\mathbf{s}}_k(t_{k+1}) &= \dot{\mathbf{s}}_{k+1}(t_{k+1}) \\ &\vdots \\ \mathbf{s}_k^{(p-1)}(t_{k+1}) &= \mathbf{s}_{k+1}^{(p-1)}(t_{k+1}). \end{aligned} \tag{3.3}$$

Note, defining the spline coefficients as in (3.2) ensures that the spline nodes (3.1) are satisfied. Also, as

$$\mathbf{s}_k^{(p-1)}(t_{k+1}) = \mathbf{s}_{k+1}^{(p-1)}(t_{k+1}) \Leftrightarrow \mathbf{q}_k^{(p)} \Delta T_k + \mathbf{q}_k^{(p-1)} = \mathbf{q}_{k+1}^{(p-1)}$$

let

$$\mathbf{q}_k^{(p)} := \frac{\mathbf{q}_{k+1}^{(p-1)} - \mathbf{q}_k^{(p-1)}}{\Delta T_k} \tag{3.4}$$

to reduce the number of unknown variables. Now, the state vector $\mathbf{x}(t) \in \mathbb{R}^{pj}$ is defined as

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \\ \vdots \\ \mathbf{q}^{(p-1)}(t) \end{pmatrix}$$

and let $\mathbf{x}_k := \mathbf{x}(t_k)$. For sake of readability, $\mathbf{x}(t; \mathbf{s}_k)$ is defined as

$$\mathbf{x}(t; \mathbf{s}_k) := \begin{pmatrix} \mathbf{s}_k(t) \\ \dot{\mathbf{s}}_k(t) \\ \vdots \\ \mathbf{s}_k^{(p-1)}(t) \end{pmatrix}.$$

As such, the notation $\mathbf{x}(t;\mathbf{s}_k)$ emphasizes $\mathbf{x}(t)$ associated with the polynomial $\mathbf{s}_k(t)$ (see Figure 3.1). The continuity conditions (3.3) can now be written as

$$\mathbf{x}_{k+1} = \mathbf{x}(t_k + \Delta T_k; \mathbf{s_k}), \qquad (k = 1, 2, \ldots, n-1).$$

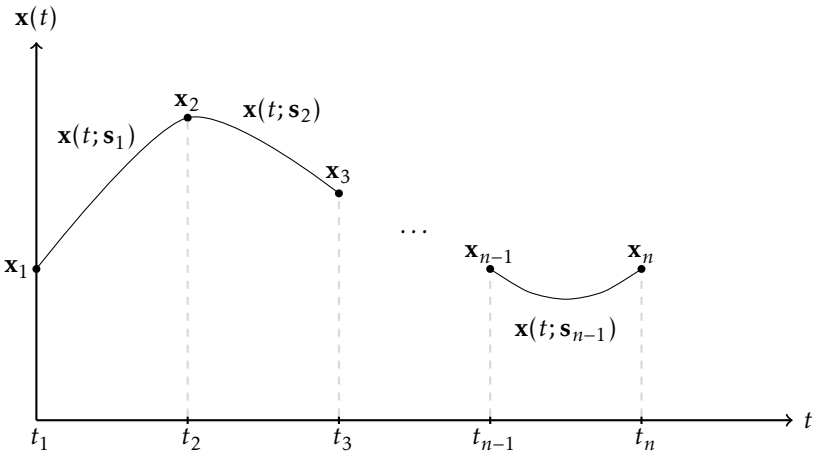


Figure 3.1: *Generic illustration of the continuity of one component of* $\mathbf{x}(t)$.

The sequence of nodes $\mathbf{x}_k$ and time intervals ΔT_k are combined into a single set

$$\mathbf{b}_l = \{\mathbf{x}_1, \mathbf{\Delta T}_1, \mathbf{x}_2, \mathbf{\Delta T}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{\Delta T}_{n-1}, \mathbf{x}_n\}$$

defined as *Timed Elastic Nodes* (TEN) with non-uniform knots, where $\mathbf{b}_l \in \mathcal{R}^{\varsigma}$ with $\varsigma = npj + (n-1)$. The discretized time optimal control problem with non-uniform knots is formulated as

$$\begin{aligned} \min_{\mathbf{b}_l} \quad & \sum_{k=1}^{n-1} [\Delta T_k + \lambda r(\Delta T_k)] \\ s.t. \quad & \begin{cases} \mathbf{x}_1 = \mathbf{x}_i,\ \mathbf{x}_n = \mathbf{x}_f,\ \Delta T_k > 0 \\ \mathbf{x}_{k+1} = \mathbf{x}(t_k + \Delta T_k; \mathbf{s}_k) \\ g(\mathbf{x}_k, \Delta T_k) \geq \mathbf{0} \qquad\qquad (k = 1, 2, \ldots, n-1). \end{cases} \end{aligned} \tag{3.5}$$

where it is assumed that the inequality constraints $g(\mathbf{x}_k, \Delta T_k)$ are satisfied for all intermediate states on the time interval $[t_k, t_k + \Delta T_k]$. Thus, $\mathbf{x}_k$ can be substituted by any $\mathbf{x}(t)$ from $[t_k, t_k + \Delta T_k]$ without violating the inequality constraints. In general, this assumption is not satisfied and various approaches to alleviate this issue is introduced in this work.

Analogously to TEB with global time discretization, the knots can be constrained to be uniformly distributed by defining $\Delta T_k := \Delta T$, which yield the single set

$$\mathbf{b}_g = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \Delta T\}$$

called TEN with uniform knots, where $\mathbf{b}_g \in \mathcal{R}^\sigma$ with $\sigma = npj + 1$. The discretized time optimal control problem with uniform knots is formulated as

$$\begin{aligned} \min_{\mathbf{b}_g} \quad & (n-1)\Delta T \\ s.t. \quad & \begin{cases} \mathbf{x}_1 = \mathbf{x}_i,\ \mathbf{x}_n = \mathbf{x}_f,\ \Delta T > 0 \\ \mathbf{x}_{k+1} = \mathbf{x}(t_k + \Delta T; \mathbf{s}_k) \\ g(\mathbf{x}_k, \Delta T) \geq \mathbf{0} \qquad\qquad (k = 1, 2, \ldots, n-1). \end{cases} \end{aligned} \tag{3.6}$$

The difference between uniform and non-uniform knots as well as the effect of λ and $r(\Delta T_k)$ is further discussed in Section 4.1.1.

In contrast to the original TEB formulation, the TEN framework yields a time continuous trajectory which allows the resolution of the trajectory, sent to the low-level controller, to be controlled independently of band length n. Also, since there are no approximations, the time intervals ΔT_k are allowed to be arbitrarily large which implies that feasible trajectories can be generated without utilizing a large band length n with the potential advantage to reduce the computation time.

3.2 Timed Elastic Nodes Applied to Industrial Robots

Now, the TEN framework is applied to the time optimal control problem for industrial robots (2.13). Cubic splines ($p = 3$) are employed to ensure continuity in the joint accelerations and thus continuous torques, see (2.5).

With $p = 3$, the state vector $\mathbf{x}(t) \in \mathcal{R}^{3j}$ is

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \end{pmatrix}$$

with spline nodes

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{q}_k \\ \dot{\mathbf{q}}_k \\ \ddot{\mathbf{q}}_k \end{pmatrix}.$$

Also, the polynomials $\mathbf{s}_k(t) \in \mathcal{R}^j$ are given by

$$\mathbf{s}_k(t) = \frac{\ddot{\mathbf{q}}_{k+1} - \ddot{\mathbf{q}}_k}{6\Delta T_k}(t - t_k)^3 + \frac{\ddot{\mathbf{q}}_k}{2}(t - t_k)^2 + \dot{\mathbf{q}}_k(t - t_k) + \mathbf{q}_k$$

and

$$\mathbf{x}(t; \mathbf{s}_k) = \begin{pmatrix} \mathbf{s}_k(t) \\ \dot{\mathbf{s}}_k(t) \\ \ddot{\mathbf{s}}_k(t) \end{pmatrix}.$$

The inequality constraints $g(\mathbf{x}_k, \Delta T_k)$ are given by

$$g(\mathbf{x}_k, \Delta T_k) = \begin{pmatrix} g_1(\mathbf{x}_k) \\ g_2(\mathbf{x}_k) \\ g_3(\mathbf{x}_k, \Delta T_k) \end{pmatrix}$$

where $g_1(\mathbf{x}_k)$ denote the torque bounds

$$g_1(\mathbf{x}_k) = \begin{pmatrix} M(\mathbf{q}_k)\ddot{\mathbf{q}}_k + C(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + G(\mathbf{q}_k) - \boldsymbol{\tau}_- \\ \boldsymbol{\tau}_+ - (M(\mathbf{q}_k)\ddot{\mathbf{q}}_k + C(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + G(\mathbf{q}_k)) \end{pmatrix},$$

$g_2(\mathbf{x}_k)$ denote the joint velocity bounds

$$g_2(\mathbf{x}_k) = \begin{pmatrix} \dot{\mathbf{q}}_k - \dot{\mathbf{q}}_- \\ \dot{\mathbf{q}}_+ - \dot{\mathbf{q}}_k \end{pmatrix},$$

and finally, $g_3(\mathbf{x}_k, \Delta T_k)$ denote the joint jerk bounds

$$g_3(\mathbf{x}_k, \Delta T_k) = \begin{pmatrix} \frac{\ddot{\mathbf{q}}_{k+1} - \ddot{\mathbf{q}}_k}{\Delta T_k} - \dddot{\mathbf{q}}_- \\ \dddot{\mathbf{q}}_+ - \frac{\ddot{\mathbf{q}}_{k+1} - \ddot{\mathbf{q}}_k}{\Delta T_k} \end{pmatrix}.$$

The initial position $\mathbf{q}_i$, initial velocity $\dot{\mathbf{q}}_i$, initial acceleration $\ddot{\mathbf{q}}_i$ and target position $\mathbf{p}_f$ are assumed to be known through sensor readings and/or state estimation. At this point, assume that the target velocity $\dot{\mathbf{p}}_f$ is zero. Non-zero target velocities are addressed in Section 3.7. To limit the scope, the target acceleration $\ddot{\mathbf{p}}_f$ will always be zero in this work. Consequently, the boundary states are given by

$$\mathbf{x}_i = \begin{pmatrix} \mathbf{q}_i \\ \dot{\mathbf{q}}_i \\ \ddot{\mathbf{q}}_i \end{pmatrix}, \ \mathbf{x}_f = \begin{pmatrix} \chi^{-1}(\mathbf{p}_f) \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{3.7}$$

As discussed, there might be multiple solutions of the inverse kinematics. As such, it is not obvious to decide which one that should be used as a goal configuration in the planner to achieve the shortest transition time. In this work, a naive approach is applied where the goal configuration is chosen to be the one closest to the current configuration of the robot. However, this might not yield the time-optimal trajectory. Further consideration is required to solve the issue of multiple solutions of the inverse kinematics, for example by investigating all candidate configurations, which can be done in parallel. This approach is used in [2].

3.3 Obstacle Avoidance

An important aspect of real-time trajectory planning is collision avoidance. For simplicity, the trajectory planner only aims to keep the end-effector of the robot free from colliding. One could imagine that the robot arm of the planar elbow robot is placed on a level above the obstacles, thus safe from collision. In contrast, the end-effector of the robot carries a tool that reaches down to the obstacles. This simplified scenario is used in this work as proof of concept, in order to investigate how the planning framework performs in environments involving simple obstacles.

The obstacles are modelled as spherical bodies that the end-effector of the robot cannot enter. Hence, each obstacle j is parametrized by its radius r_j and

center $\mathcal{O}_j$ and the avoidance of obstacles can be formulated as inequality constraints

$$\|\mathbf{p}(t) - \mathcal{O}_j\|^2 \geq (s + r_j)^2$$

or equivalently

$$\|\chi(\mathbf{q}(t)) - \mathcal{O}_j\|^2 \geq (s + r_j)^2$$

for $(j = 1, 2, \ldots, m)$, where m is the number of obstacles and s is a predefined safety distance. This is incorporated in the TEN framework by augmenting the inequality constraints $g(\mathbf{x}_k, \Delta T_k)$ with

$$g_4(\mathbf{x}_k) = \begin{pmatrix} \|\chi(\mathbf{q}_k) - \mathcal{O}_1\|^2 - (s + r_1)^2 \\ \|\chi(\mathbf{q}_k) - \mathcal{O}_2\|^2 - (s + r_2)^2 \\ \vdots \\ \|\chi(\mathbf{q}_k) - \mathcal{O}_m\|^2 - (s + r_m)^2 \end{pmatrix}.$$

Consequently, $g(\mathbf{x}_k, \Delta T_k)$ is now given by

$$g(\mathbf{x}_k, \Delta T_k) = \begin{pmatrix} g_1(\mathbf{x}_k) \\ g_2(\mathbf{x}_k) \\ g_3(\mathbf{x}_k, \Delta T_k) \\ g_4(\mathbf{x}_k) \end{pmatrix}.$$

3.4 Inequality Constraints Satisfied for Intermediate States

As mentioned, the assumption that the inequality constraints $g(\mathbf{x}_k, \Delta T_k)$ are fulfilled for all intermediate states in the time interval $[t_k, t_k + \Delta T_k]$ will in general not be satisfied. Obviously, the assumption might be satisfied by increasing the band length n, but this significantly increase the computation time. Thus alternative approaches are required. First, a method to ensure that the joint velocity satisfies the bounds is presented and second, a method to reduce the violation of the torque bound as well as ensuring collision-free trajectories is introduced. Note, the joint jerk bounds are fulfilled for all intermediate states since the joint jerk is constant during the time interval $[t_k, t_k + \Delta T_k]$.

3.4.1 Conservative Joint Velocity Constraints

For cubic splines, the joint accelerations are linear in each time interval $[t_k, t_{k+1}]$ and the extreme values of the joint velocities are either located at t_k, t_{k+1} or possibly at the time instance where the joint accelerations change sign. Since the time instances t_k and t_{k+1} corresponds to the spline nodes, which are constrained to satisfy the joint velocity bounds, the problem of violating the joint velocity bounds arises when joint accelerations change sign within an interval $[t_k, t_{k+1}]$. Thus as a first thought, introducing additional constraints on the intermediate joint velocities where the zero-crossings of the joint accelerations occur would be enough

to ensure that all joint velocities satisfy the bounds. However, to formulate a function of the optimization variables that locate the zero-crossings of the joint accelerations is difficult and would require usage of non-continuous functions as there are different cases depending on the sign of $\ddot{q}_k$ and $\ddot{q}_{k+1}$. As such, the constraint function would be non-continuous and most common optimization solvers are not suited for non-continuous functions [18]. Instead, a conservative approach is applied where it is assumed that zero-crossings occur at the end of each time interval $t_k \leq t \leq t_{k+1}$.

The intermediate joint velocity $\dot{q}(t)$ in time interval $[t_k, t_{k+1}]$ is given by

$$\dot{q}(t) = \dot{s}_k(t)$$

where

$$\dot{s}_k(t) = \frac{\ddot{q}_{k+1} - \ddot{q}_k}{2\Delta T_k}(t - t_k)^2 + \ddot{q}_k(t - t_k) + \dot{q}_k.$$

Let

$$\dot{\tilde{s}}_k(t) = \frac{-\ddot{q}_k}{2\Delta T_k}(t - t_k)^2 + \ddot{q}_k(t - t_k) + \dot{q}_k$$

where $\ddot{q}_{k+1} = 0$ is assumed and define $\dot{\tilde{q}}_{k+1} := \dot{\tilde{s}}_k(t_{k+1})$. Thus, if the joint acceleration $\ddot{q}(t)$ changes sign in the time interval $[t_k, t_{k+1}]$ the following inequalities hold

$$\begin{cases} \dot{\tilde{q}}_{k+1} > \dot{q}(t), & \text{if } \ddot{q}_k > 0,\ \ddot{q}_{k+1} < 0 \\ \dot{\tilde{q}}_{k+1} < \dot{q}(t), & \text{if } \ddot{q}_k < 0,\ \ddot{q}_{k+1} > 0 \end{cases}$$

for $t \in [t_k, t_{k+1}]$. Also, if no zero-crossing occurs in the time interval the following inequalities hold

$$\dot{q}_k \leq \dot{\tilde{q}}_{k+1} \leq \dot{q}_{k+1}. \tag{3.8}$$

An illustration of these cases are shown in Figure 3.2. Hence, introducing additional inequality constraints

$$\dot{\mathbf{q}}_- \leq \dot{\tilde{\mathbf{q}}}_{k+1} \leq \dot{\mathbf{q}}_+, \qquad (k = 1, 2, \ldots, n-1)$$

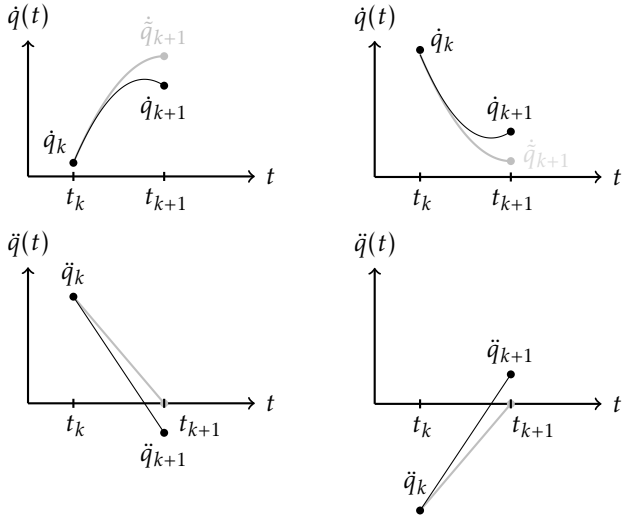
in the planning framework ensures that the intermediate joint velocity satisfies the bounds. As

$$\dot{\tilde{\mathbf{q}}}_{k+1} = \dot{\tilde{\mathbf{s}}}_k(t_{k+1}) = \dot{\tilde{\mathbf{s}}}_k(t_k + \Delta T_k) = \frac{\ddot{\mathbf{q}}_k}{2}\Delta T_k + \dot{\mathbf{q}}_k$$

the inequality constraints $g_2(\mathbf{x}_k, \Delta T_k)$ for the joint velocity bounds are augmented according to

$$g_2(\mathbf{x}_k, \Delta T_k) = \begin{pmatrix} \dot{\mathbf{q}}_k - \dot{\mathbf{q}}_- \\ \dot{\mathbf{q}}_+ - \dot{\mathbf{q}}_k \\ \frac{\ddot{\mathbf{q}}_k}{2}\Delta T_k + \dot{\mathbf{q}}_k - \dot{\mathbf{q}}_- \\ \dot{\mathbf{q}}_+ - \left(\frac{\ddot{\mathbf{q}}_k}{2}\Delta T_k + \dot{\mathbf{q}}_k\right) \end{pmatrix}.$$

Note, if the joint variables are accelerating/decelerating during the whole time interval $[t_k, t_{k+1}]$, the inequalities (3.8) hold and the additional constraints will not intrude the time optimality.



(a) *Acceleration to deceleration.*

(b) *Deceleration to acceleration.*

(c) *Acceleration.*

(d) *Deceleration.*

Figure 3.2: *Intermediate joint velocity and joint acceleration in four different cases. Note, in (a) $\dot{\tilde{q}}_{k+1} > \dot{q}(t)$, in (b) $\dot{\tilde{q}}_{k+1} < \dot{q}(t)$ and in (c) and (d) $\dot{q}_k \leq \dot{\tilde{q}}_{k+1} \leq \dot{q}_{k+1}$.*

3.4.2 Oversampling of the State Trajectory

Due to the spline formulation, the intermediate states are obtained in a computationally efficient way as it only requires evaluation of polynomials. Thus, an alternative approach to increasing the band length n, is to add additional inequality constraints on a predefined number of intermediate states. This approach will of course also increase the computation time, but not in the same extent as increasing the band length n. The number of constrained intermediate states will serve as a trade-off between feasibility and computational time.

In detail, each time interval ΔT_k is divided into $m+1$ equidistant time intervals of length $\Delta t_k = \frac{\Delta T_k}{m+1}$. Inequality constraints are imposed on intermediate states $\mathbf{x}(t_k + l\Delta t_k; \mathbf{s}_k)$ for $l = 1, 2, \ldots, m$. Notice, the cases $l = 0$ and $l = m+1$ correspond to the spline nodes and should not be included. The inequality constraint function $g(\mathbf{x}_k, \Delta T_k)$ is now given by

$$g(\mathbf{x}_k, \Delta T_k) = \begin{pmatrix} g_1(\mathbf{x}_k) \\ g_1(\mathbf{x}(t_k + \frac{l_\tau}{m_\tau+1}\Delta T_k; \mathbf{s}_k)) \\ g_2(\mathbf{x}_k, \Delta T_k) \\ g_3(\mathbf{x}_k, \Delta T_k) \\ g_4(\mathbf{x}_k) \\ g_4(\mathbf{x}(t_k + \frac{l_{obs}}{m_{obs}+1}\Delta T_k; \mathbf{s}_k)) \end{pmatrix}$$

for $l_\tau = 1, 2, \ldots, m_\tau$ and $l_{obs} = 1, 2, \ldots, m_{obs}$, where m_τ is the number of intermediate states per time interval for the input constraints and m_{obs} is the number of intermediate states per time interval for the obstacle constraints.

3.5 Trajectory Initialization

The initial guess is of great importance for many optimization solvers in order to find a feasible solution. In this work, an initial trajectory is formed by calculating a feasible velocity profile along a fixed geometric path. To make it simple, the geometric path is a straight line in configuration space between the initial joint variables and the goal joint variables. For simple scenarios, the proposed geometric path is sufficient and yields feasible trajectories. On the other hand, if obstacles are located in the workspace the initial trajectory might not be collision-free and one might have to apply some existing path-planning to find a collision-free path, for example potential field methods or probabilistic roadmap methods [22].

To find a feasible velocity profile, an iterative approach is applied where the desired total transition time is gradually increased until the torque and joint velocity bounds are satisfied. For simplicity, the initial trajectory consists of four spline nodes with uniform knots $[0, t_s, 2t_s, 3t_s]$. Hence, given the initial state $\mathbf{x}_i$, the goal state $\mathbf{x}_f$ as well as a requested total transition time $3t_s$, let

$$q(t) = q_i + l(t)(q_f - q_i)$$

for each joint variable $q(t)$, where

$$l(t) = \begin{cases} l_1(t), & 0 \le t \le t_s \\ l_2(t), & t_s \le t \le 2t_s \\ l_3(t), & 2t_s \le t \le 3t_s \end{cases}$$

and

$$l_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i, \quad i = 1, 2, 3$$

where $l(t) \in [0, 1]$, $l(t) \in C^2$. By this formulation, the joint variables are fixed to the straight line path between the initial joint variables and the goal joint variables. Now, the objective is to determine the spline coefficients a_i, b_i, c_i, d_i in order to satisfy the initial and final state and the continuity conditions. As

$$\dot{q}(t) = (q_f - q_i)\dot{l}(t), \quad \ddot{q}(t) = (q_f - q_i)\ddot{l}(t)$$

the initial state yields three equations to satisfy

$$l(0) = 0, \quad \dot{l}(0) = \frac{\dot{q}_i}{q_f - q_i}, \quad \ddot{l}(0) = \frac{\ddot{q}_i}{q_f - q_i}$$

and the final state yields three additional equations

$$l(3t_s) = 1, \quad \dot{l}(3t_s) = \frac{\dot{q}_f}{q_f - q_i}, \quad \ddot{l}(3t_s) = \frac{\ddot{q}_f}{q_f - q_i}.$$

As $l(t) \in C^2$ the following six equations need to be satisfied

$$l_1(t_s) = l_2(t_s), \quad \dot{l}_1(t_s) = \dot{l}_2(t_s), \quad \ddot{l}_1(t_s) = \ddot{l}_2(t_s)$$

$$l_2(2t_s) = l_3(2t_s), \quad \dot{l}_2(2t_s) = \dot{l}_3(2t_s), \quad \ddot{l}_2(2t_s) = \ddot{l}_3(2t_s).$$

Hence, all 12 unknown spline coefficients can be determined by utilizing the 12 independent equations. When a complete trajectory $(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$ is obtained, the torques required to realize the trajectory are given by

$$M(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + C(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + g(\mathbf{q}(t)) = \boldsymbol{\tau}(t).$$

The spline nodes and intermediate states are examined with a similar approach as described in Section 3.4 and if the input or the joint velocity bounds are violated, the total transition time is prolonged and the process is restarted. Hence, the resulting trajectory will satisfy the initial and final state, the torque bounds as well as the joint velocity bounds. Finally, if the band length n is greater than four, the remaining spline nodes and knots are equally distributed along the interval $[t_s, 2t_s]$.

3.6 Sensor Reading and Planner Update

As the trajectory planner should be running in real-time, the integration of the open-loop optimization based trajectory planner and closed-loop planning should be discussed. In real applications, after the trajectory planner has finished the optimization phase, the resulting trajectory is sent to the low-level controller. At the next planning cycle, given the current state of the system and the location of the target obtained from sensor measurements and/or state estimations, a time-shifted version of the previous trajectory is used to warm-start the next optimization phase. In the simulations presented in the next chapter, it is assumed that the low-level controller can perfectly realize the given trajectory.

Let ΔT_s denote the sampling time and thus also the time elapsed since last planning cycle i. At planning cycle $i+1$, the first state and target state are updated according to sensor readings. As the time ΔT_s have elapsed since planning cycle i, the time intervals are updated according to

$$\Delta T_k^{(i+1)} = \Delta T_k^{(i)} - \frac{\Delta T_s}{n-1}$$

which implies that

$$\sum_{k=1}^{n-1} \Delta T_k^{(i+1)} = \sum_{k=1}^{n-1} \Delta T_k^{(i)} - \Delta T_s.$$

Hence, the predicted total transition time is reduced by ΔT_s and the intermediate spline nodes are updated according to

$$\mathbf{x}_k^{(i+1)} = \mathbf{x}^{(i)}(t_k^{(i+1)} + \Delta T_s), \quad (k = 2, 3, \ldots, n-1).$$

Obviously, $\Delta T_k > 0$ should hold, but too small time intervals should also be avoided as they cause numerical instabilities. To alleviate this issue, if some $\Delta T_k < \Delta T_s$, the first state in the trajectory is removed before the trajectory is shifted. However, the number of states is not allowed to be less than n_{min}.

Finally, when the end-effector is approaching the target the time intervals ΔT_k will approach to zero. Also, there is no guarantee that the system is stabilized at the target. Under mild conditions suboptimal model predictive controllers are stabilizing [21]. One condition is that each iteration yields a sufficient decrease of the objective function, which is ensured by utilizing quadratic cost functions. To utilize this result within the TEN framework, a switching strategy as in [2] is introduced. When the end-effector is within a pre-defined distance from the target, σ_t, the objective function is switched to a tracking formulation

$$\sum_{k=1}^{n-1} \|\mathbf{x}_k - \mathbf{x}_f\|^2 \tag{3.9}$$

and $\Delta T_k = \Delta T_s$ for $k = 1, 2, \ldots, n-1$. Hence, the time intervals are not allowed to vary and are excluded from the optimization vector in order to not intrude the convergence properties of the quadratic formulation. As long as the planning process (3.5) or (3.6) is able to guide the manipulator close to the target state, the quadratic formulation is utilized to convergence to the target state.

3.7 Tracking Moving Targets

If the objective is to track a moving target, the final state in (3.7) needs further consideration. Let $\mathbf{p}^{tg}(t)$ represent the movement of the target in Cartesian space, reaching the target is then equivalent to

$$\mathbf{p}(t_f) = \mathbf{p}^{tg}(t_f),\ \dot{\mathbf{p}}(t_f) = \dot{\mathbf{p}}^{tg}(t_f).$$

Consequently, the corresponding end condition in the TEN formulation are obtained by

$$\mathbf{x}_f = \begin{pmatrix} \chi^{-1}(\mathbf{p}^{tg}(t_f)) \\ J^{-1}(\chi^{-1}(\mathbf{p}^{tg}(t_f)))\dot{\mathbf{p}}^{tg}(t_f) \\ \mathbf{0} \end{pmatrix}$$

utilizing the inverse kinematics and the corresponding Jacobian. As described before in Section 2.1, all singularities of the Jacobian should be identified before implementation to ensure that the target does not pass through them. Scenarios where that is not possible to avoid requires further consideration. If the velocity of the target is not too large, the previous generated trajectory will serve as a feasible initial guess since the new optimization problem will be similar to the previous one. Note, as previously mentioned it is assumed that the target acceleration is zero. One could imagine that the objective is to track a target placed on a linear conveyor.

If the entire trajectory of the target $\mathbf{p}^{tg}(t)$ or $\dot{\mathbf{p}}^{tg}(t)$ is supplied, a more satisfactory approach can be utilized to the problem. In that case, the current estimated transition time and the trajectory of the target can be used to predict where the target will be located at the time instance when the end-effector is close to the target. Hence, instead of aiming at the target's current position, the goal can be set to the predicted location of the target. For instance, if the target velocity $\mathbf{v}$ is constant, the goal position can be set to

$$\mathbf{p}^{tg}(t_i) + \sum_{k=1}^{n-1} \Delta T_k \mathbf{v}$$

where $\mathbf{p}^{tg}(t_i)$ is the current position of the target and $\sum_{k=1}^{n-1} \Delta T_k$ is the current estimated transition time. As such, instead of first reaching the target and then having to catch up, the robot will be able to reach the target with a matching velocity. However, there is an apparent drawback with this approach as the predicted future position of the target might be located outside the workspace of the end-effector. In some scenarios, this implies that the robot would not have been able to reach the target. On the other hand, this could also occur if the current estimated transition time is not accurate. Further consideration is required to solve this problem and to distinguish these two scenarios.

3.8 Implementation

The trajectory planner is implemented in C++ as an object-oriented program and is based on the software package used in [2]. The main classes are *TrajectoryPlanner*, *TENSolver* and *IPSolver*. The *TrajectoryPlanner* provides an API for end users and setup the planning problem as well as update the planner according to sensor readings. The *TENSolver* handles the setup of the underlying optimization problem and stores the current TEN in *TimedElasticNodes*. Next, the *IPSolver* is used to solve the optimization problem with the interior-point method and uses *FGeval* to store and evaluate the objectives and constrains. Finally, another notable class is *TrajectoryProblem* which is used to read and store the configuration parameters obtained by a mandatory configuration file, as well as the class *RobotModel* that includes the kinematics and dynamics of the target robot. An example of a configuration file is given in Appendix A and a class diagram is shown in Figure 3.3.

Several third-party libraries are used by the trajectory planner. The *Eigen* [7] library is used for linear algebra operations and to store vectors and matrices. The optimization problem is solved using IPOPT [23] which applies the interior-point method. IPOPT uses automatic differentiation, provided by the *CppAD* [1] library, to compute required gradients, Jacobians and Hessians.

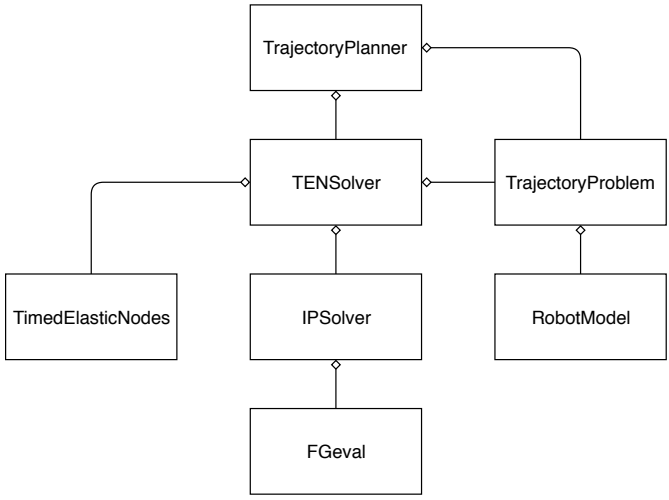


Figure 3.3: *Class diagram that describes the software structure.*

4

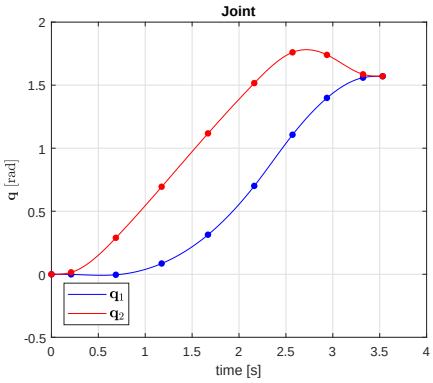
Evaluation of the Trajectory Planner

In this section the trajectory planner is evaluated in simulations. Mainly, the trajectory planner is applied to the planar elbow robot, but simulations are also made on ABB's IRB 910SC to analyze the performance on a more complex robot. First, simulations are made in a static environment and strengths together with weaknesses of the trajectory planner are discussed. Second, the trajectory planner is employed in dynamic environments where the trajectory is updated during execution.

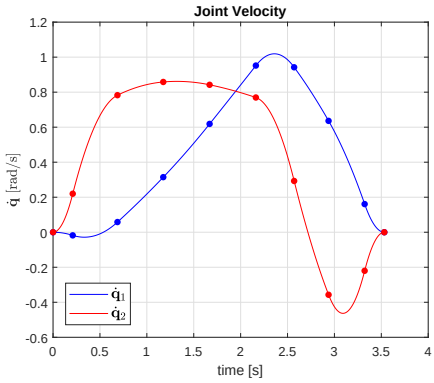
4.1 Static Environment

For all scenarios in this section, the planar elbow robot is initially located with the end-effector at $(2, 0)$ with joint angles, joint velocities and joint accelerations equal to zero, see Figure 4.1f. Further, the target is located at $(-1, 1)$ and the planner is configured according to Appendix A, if nothing else is stated. Also, to alleviate the issue that the underlying NLP for the TEN with non-uniform knots (3.5) is under-constrained the regularization term is set to $r(\Delta T_k) = \Delta T_k^2$, as suggested in [20].

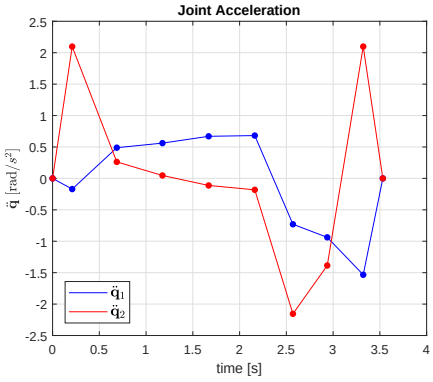
As a first example, a trajectory between the start state and the target state is planned. Figure 4.1 shows the planned trajectory together with the robot movement. As expected from a time-optimal trajectory, at least one constraint is active during each time interval. The joint jerk bounds are active in the initial and final phase and the torques are of bang-bang structure. This indicates that the solution might be close to the solution of the time optimal control problem (2.13).



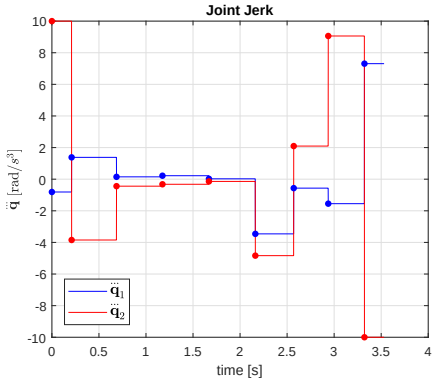
(a) Joint angle.



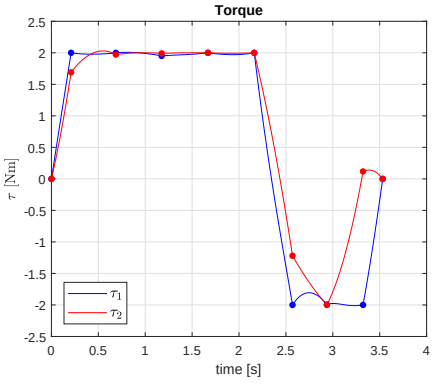
(b) Joint velocity.



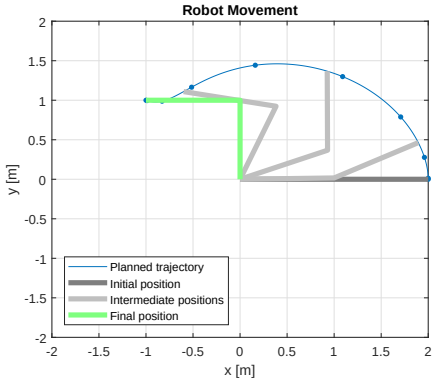
(c) Joint acceleration.



(d) Joint jerk.



(e) Torques required to realize the trajectory.



(f) Planned trajectory and robot movement.

Figure 4.1: *Planned trajectory and robot movement where the dots correspond to the spline nodes. The planner was configured according to Appendix A. The total transition time is 3.53 s and the computation time was 0.146 s.*

4.1.1 Uniform Knots Versus Non-Uniform Knots

Now, TEN with non-uniform knots (3.5) and TEN with uniform knots (3.6) are compared. Figure 4.3 shows the planned trajectory and robot movement when uniform knots are employed. In comparison with Figure 4.1, the approach with uniform knots yields trajectories with longer total transition time. The main difference between the two trajectories is seen at the switching points of the torques, where the non-uniform knots approach can switch more abrupt. In other words, for uniform knots the bang-bang structure is reduced as the time to switch between the limits is prolonged.

To further compare the two approaches, trajectories are planned for various band lengths n and regularization weights λ. The computation time and the total transition time are shown in Figure 4.2. As seen, the approach with non-uniform knots yields trajectories with shorter total transition time. Even though the approach with non-uniform knots includes more optimization variables for the same band length n, there is no significant increase in computation time. One reason is that the uniform knots approach shares a common ΔT that introduces a dense row in the Hessian of the underlying optimization problem, since all spline nodes depend on the same parameter. This is not the case for the non-uniform knots approach as each ΔT_k only influences the neighbouring spline nodes, leading to a sparse Hessian. Also, as discussed in Section 2.3, for larger regularization weight λ the non-uniform knots approach converges towards the uniform knots approach.

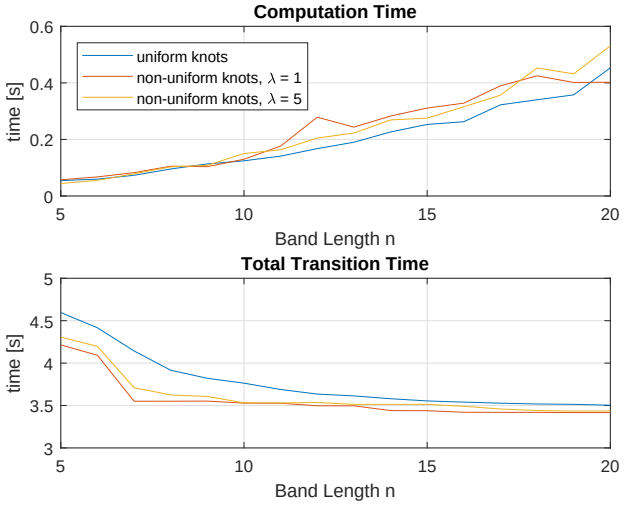
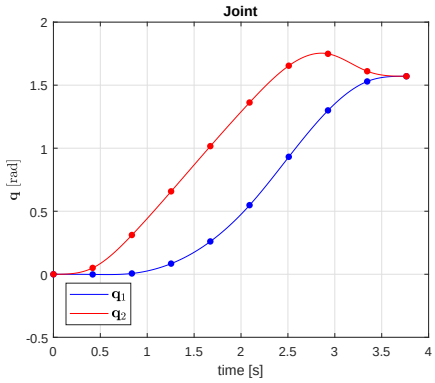
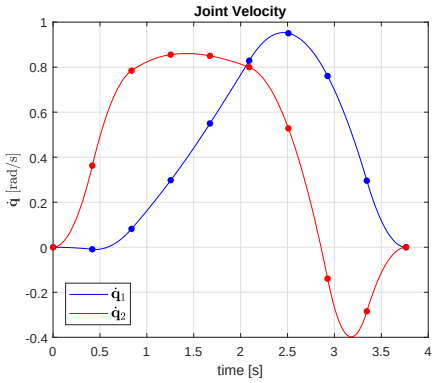


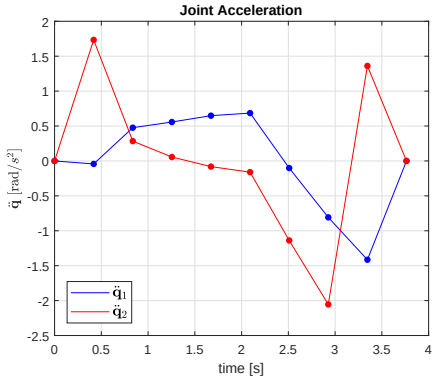
Figure 4.2: *A comparison of the non-uniform knots approach* (3.5) *and the uniform knots approach* (3.6)*. The planner was configured according Appendix A, but the band length and the regularization weight were varied according to the figure.*



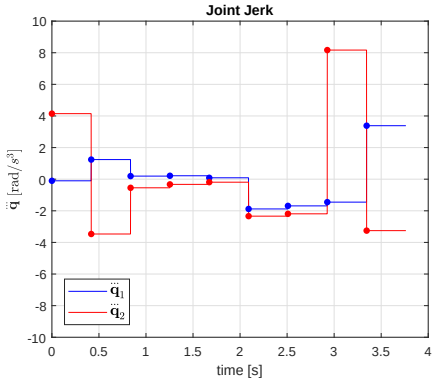
(a) Joint angle.



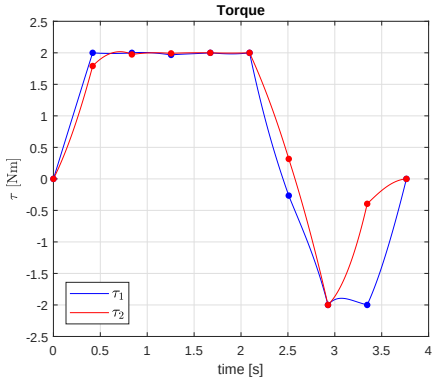
(b) Joint velocity.



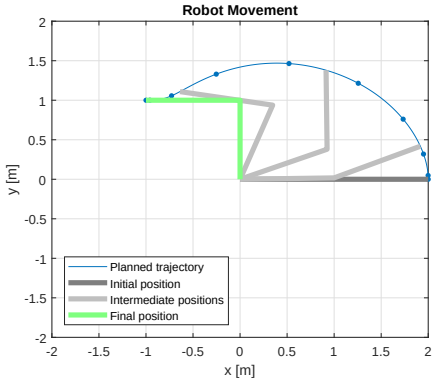
(c) Joint acceleration.



(d) Joint jerk.



(e) Torques required to realize the trajectory.



(f) Planned trajectory and robot movement.

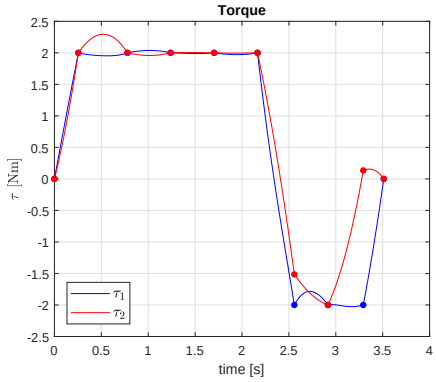
Figure 4.3: *Planned trajectory and robot movement where the dots correspond to the spline nodes. The planner was configured according to Appendix A, but with uniform knots set to true. The total transition time is 3.76 s and the computation time was 0.121 s.*

4.1.2 Violation of the Torque Bounds

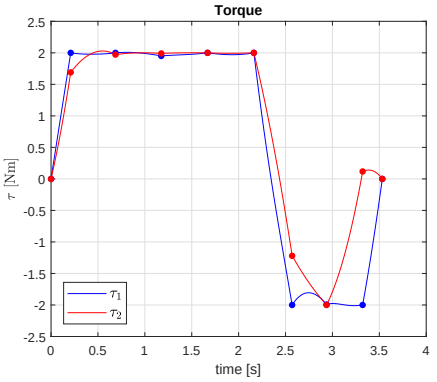
The importance of including additional inequality constraints to intermediate states is now discussed. Figure 4.4 shows the torques required to realize the trajectories when the number of constrained intermediate states for the torque bounds and band length are varied. As seen, the intermediate states violate the torque bounds in all four scenarios. The violation can be reduced by either introducing constrained intermediate states or increasing the band length, but at the expense of computational time. To identify this trade-off, the band length n and the number of constrained intermediate states for the torque bounds m_τ are varied and the obtained total transition time T^*, computation time T_c and maximum violation of the torque bounds $\boldsymbol{\tau}_{violation}$ are gathered in Table 4.1. The results are as expected, $\boldsymbol{\tau}_{violation}$ is reduced by increasing n and/or m_τ. However, increasing m_τ is computationally cheaper and has more impact on $\boldsymbol{\tau}_{violation}$. From a total transition time perspective it is preferable to increase n, but additional constraints on intermediate states are still needed to reduce the constraint violations.

Table 4.1: *Computation time T_c, total transition time T^* and maximum violation of the torque bounds $\boldsymbol{\tau}_{violation}$ for various band lengths n and number of constrained intermediate states for the torque bounds m_τ. The planner was configured according to Appendix A, except that n and m_τ was varied according to the table.*

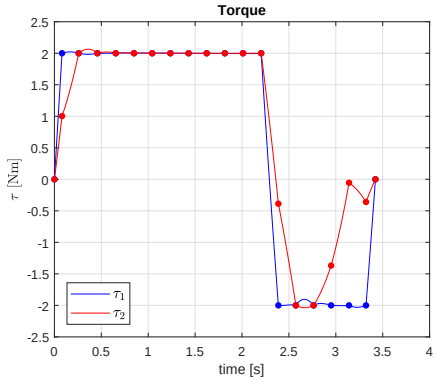
n	m_τ	$T_c\,[s]$	$T^*\,[s]$	$\boldsymbol{\tau}_{violation}\,[Nm]$
10	0	0.0598	3.5114	0.2950
10	1	0.1404	3.5319	0.0296
10	2	0.2720	3.5362	0.0031
10	3	0.3810	3.5374	0.0017
15	0	0.1219	3.4729	0.0802
15	1	0.2881	3.5130	0.0066
15	2	0.5201	3.5113	0.0024
15	3	0.9722	3.5128	0.0023
20	0	0.1912	3.4225	0.0645
20	1	0.5227	3.4361	0.0071
20	2	0.9260	3.4369	0.0022
20	3	1.4823	3.4370	0.0021



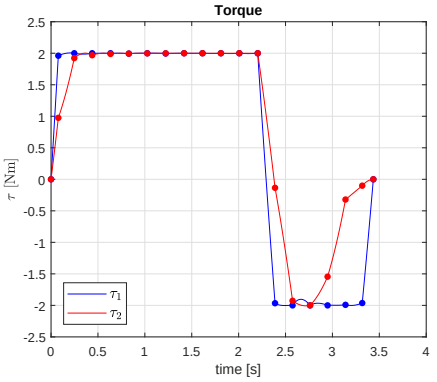
(a) Band length $n = 10$ and no additional torque bound constraints.



(b) Band length $n = 10$ and one additional torque bound constraint per time interval.



(c) Band length $n = 20$ and no additional torque bound constraints.



(d) Band length $n = 20$ and one additional torque bound constraint per time interval.

Figure 4.4: *The torques required to realize the planned trajectories. The planner was configured according to Appendix A, but band length and the number of constrained intermediate states for the torque bounds were varied.*

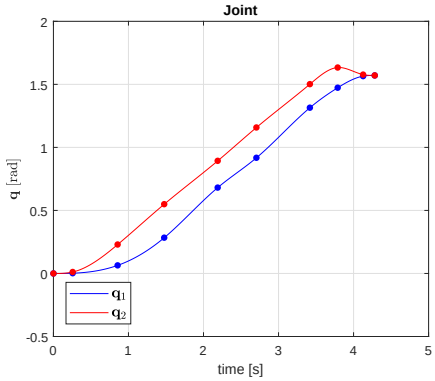
4.1.3 Conservative Joint Velocity Constraints

As discussed in Section 3.4.1, without the conservative joint velocity constraints there is no guarantee that the states between the spline nodes will satisfy the joint velocity bounds. To exemplify this, the joint velocity bounds are set to 0.5 rad/s instead of 2 rad/s as in the configuration file. First, the conservative joint velocity constraints are disregarded. The planned trajectory, together with the spline nodes, are shown in Figure 4.5. As seen, the joint velocity constraints are fulfilled in the spline nodes. However, the intermediate values violate the constraints and the violation only occurs in time intervals where the joint acceleration change sign. Figure 4.6 shows the planned trajectory when the conservative joint velocity constraints are included. Now, the intermediate values satisfy the joint velocity bounds with only a minor increase in computation time.

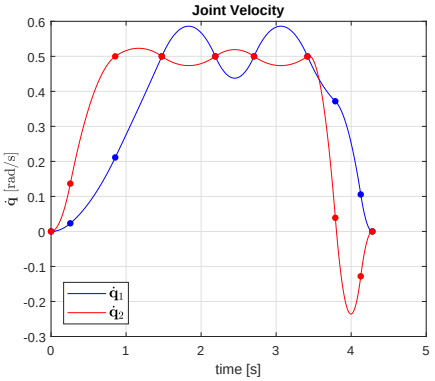
4.1.4 Planning with Obstacles

The trajectory planner is now evaluated in environments with obstacles. As mentioned, the planner only aims to keep the end-effector of the robot from colliding. First, one obstacle, located at $(-0.2, 1.1)$ with radius 0.3 m, is included in the workspace. Figure 4.7a shows the planned trajectory and robot movement when no additional obstacle constraint per time interval is used. The planner fails to find a collision-free trajectory as the planner only aims to keep the spline nodes free from colliding and does not consider the intermediate positions. As discussed, a collision-free trajectory might be found by oversampling the state trajectory and imposing constraints on the intermediate states. The result of including one additional obstacle constraint per time interval is shown in Figure 4.7b. As seen, the planner successfully finds a collision-free trajectory. An alternative solution is shown in Figure 4.7c, where the band length is increased instead of introducing additional constraints. Also in this solution, the trajectory planner finds a collision-free trajectory. The total transition time is shorter but the computation time is more than doubled.

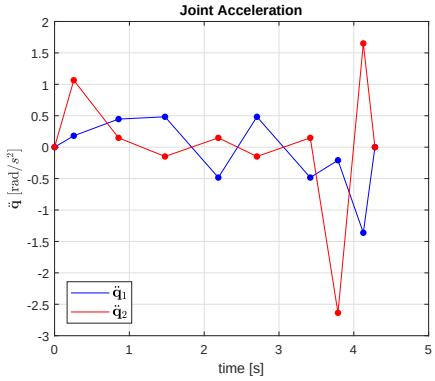
One more obstacle, located at $(0.6, 1.8)$ with radius 0.4 m, is now included in the workspace. The planned trajectory when one additional obstacle constraint per time interval is included is shown in Figure 4.8a. The planner fails to find a collision-free trajectory and for this band length, one additional obstacle constraint per time interval is not enough. Now, two additional obstacle constraint per time interval is included and the resulting trajectory is shown in Figure 4.7b. The planned trajectory becomes collision-free and the computation time is only marginally increased. To summarize, the number of additional obstacle constraints is chosen as a trade-off between computation time and feasibility. Also, for more complex scenarios, more additional obstacle constraints are required to find a collision-free trajectory.



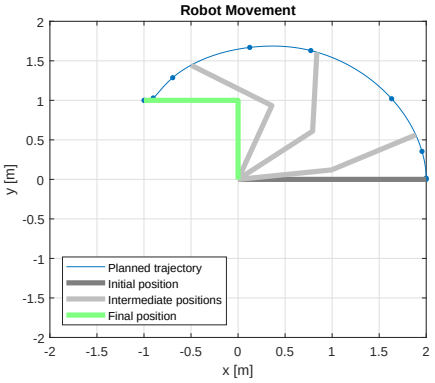
(a) Joint angles.



(b) Joint velocities.

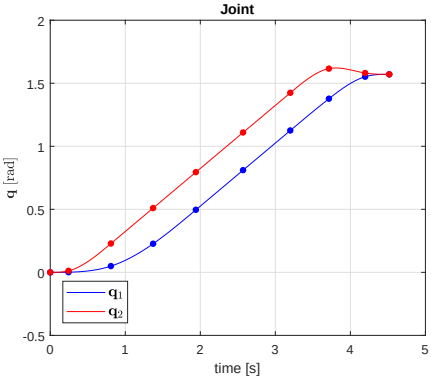


(c) Joint accelerations.

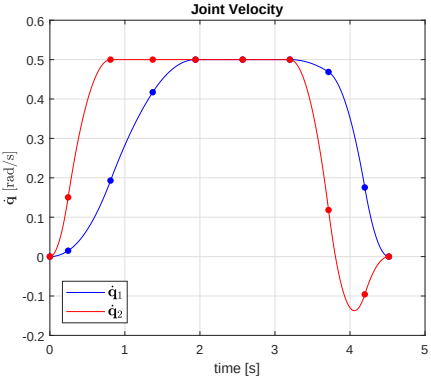


(d) Robot movement.

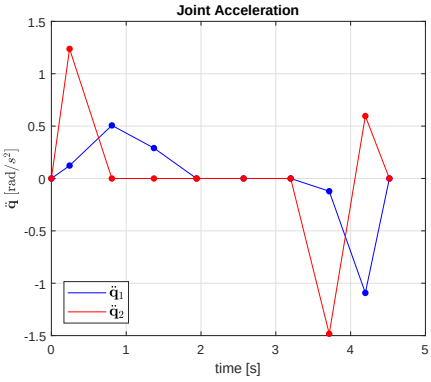
Figure 4.5: *Planned trajectory and robot movement where the dots correspond to the spline nodes. The planner was configured according to Appendix A, but the joint velocity bounds were set to 0.5 rad/s and the conservative joint velocity constraints were disregarded. The total transition time is 4.28 s and the computation time was 0.116 s. However, the joint velocity bounds are violated.*



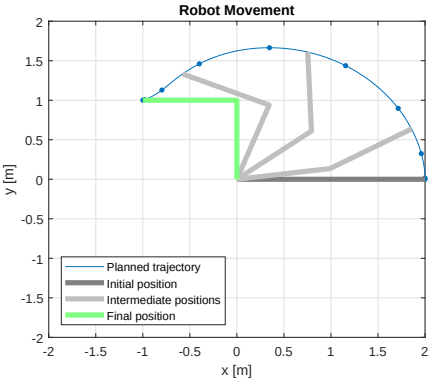
(a) Joint angle.



(b) Joint velocity.

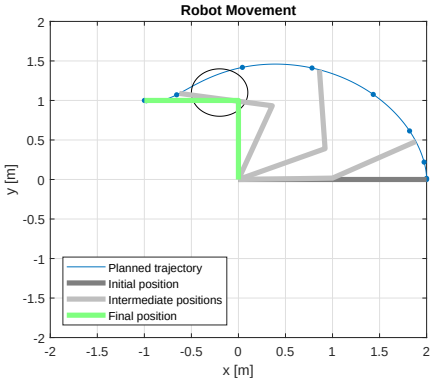


(c) Joint acceleration.

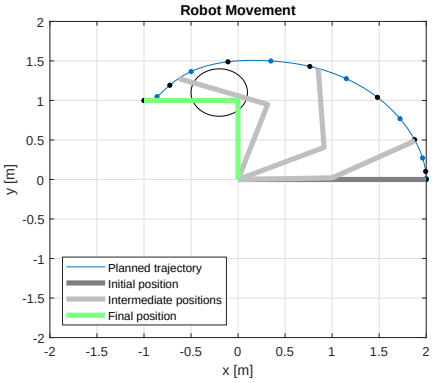


(d) Robot movement.

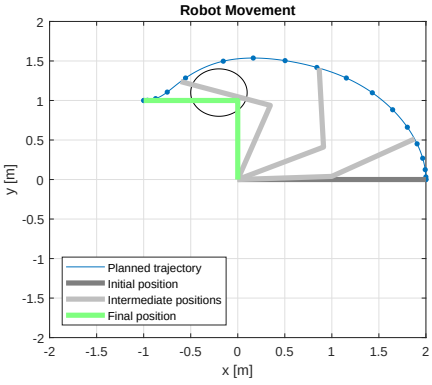
Figure 4.6: *Planned trajectory and robot movement where the dots correspond to the spline nodes. The planner was configured according to Appendix A, but the joint velocity bounds were set to 0.5 rad/s. The total transition time is 4.52 s and the computation time was 0.119 s.*



(a) *No additional obstacle constraint. The total transition time is 3.60 s and the computation time was 0.139 s, however the trajectory is not collision-free.*

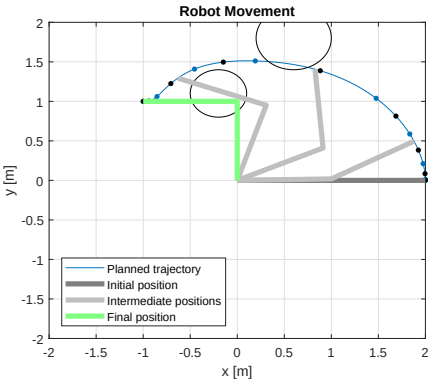


(b) *One additional obstacle constraint per time interval. The total transition time is 3.63 s and the computation time was 0.221 s.*

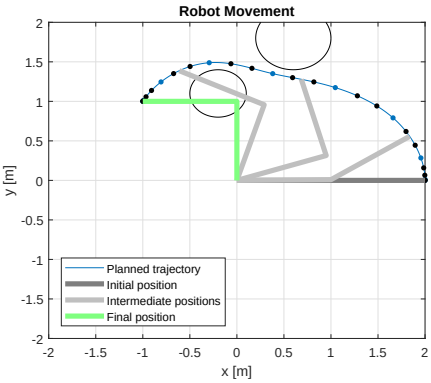


(c) *No additional obstacle constraint, but band length $n = 20$. The total transition time is 3.51 s and the computation time was 0.544 s.*

Figure 4.7: *Planned trajectories and robot movements when one obstacle, located at $(-0.2, 1.1)$ with radius 0.3 m, is included in the workspace. The planner was configured according to Appendix A, but the number of constrained intermediate states for the obstacle constraints (black dots) and band length were varied.*



(a) *One additional obstacle constraint per time interval. The total transition time is 3.667 s and the computation time was 0.337 s, however the trajectory is not collision-free.*

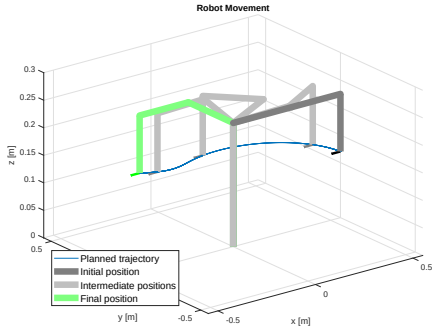


(b) *Two additional obstacle constraints per time interval. The total transition time is 3.966 s and the computation time was 0.368 s.*

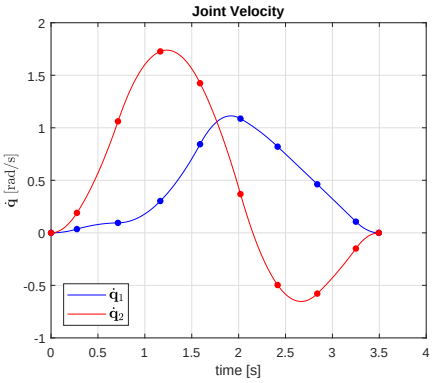
Figure 4.8: *Planned trajectories and robot movements when two obstacles, one located at* $(-0.2, 1.1)$ *with radius* 0.3 *m and the other located at* $(0.6, 1.8)$ *with radius* 0.4 *m, are included in the workspace. The planner was configured according to Appendix A, but the number of constrained intermediate states for the obstacle constraints (black dots) were varied.*

4.1.5 IRB 910SC (SCARA)

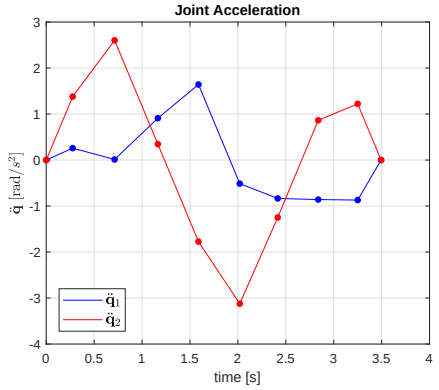
The trajectory planner is now applied to ABB's IRB 910SC. The dynamic model of the system is much more complex because it is a realistic description of an actual robot. For the sake of comparison with the previous examples in Section 4.1.2, joint three and four and are not moving and the initial state and target position in joint space are the same. The planner was configured according to Appendix A. The planned trajectory and robot movement can be seen in Figure 4.9 and the computation time was 8.467 s. As a consequence of the higher complexity, the computation time is significantly increased. This indicates that it is not possible to apply the proposed method to a real system in dynamic environments without significant modifications in the implementation to increase the performance.



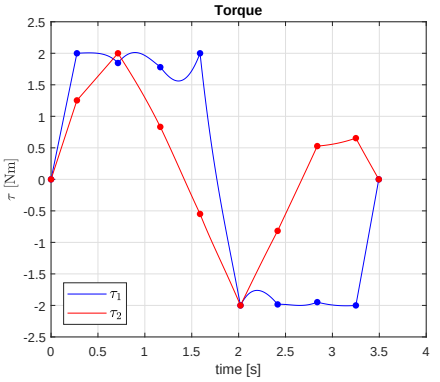
(a) Robot movement.



(b) Joint velocity.



(c) Joint acceleration.



(d) Torques required to realize the trajectory.

Figure 4.9: *Planned trajectory, robot movement and torques required to realize the trajectory. The total transition time is 3.49 s and the computation time was 8.467 s and the planner was configured according to Appendix A.*

4.2 Dynamic Environment

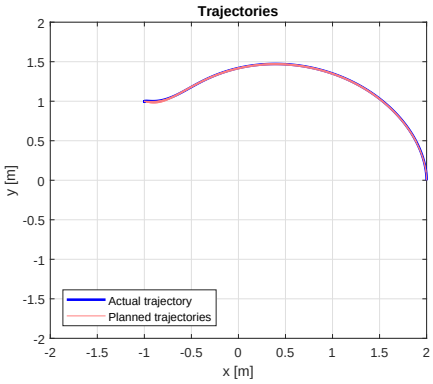
The trajectory planner is now applied to the planar elbow robot in dynamic environments where the planner has to update the trajectory during execution. The planar elbow robot is initially located with the end-effector at (2,0) with joint angles, joint velocities and joint accelerations equal to zero. The planner is configured according to Appendix A, if nothing else is stated.

4.2.1 Stationary Target

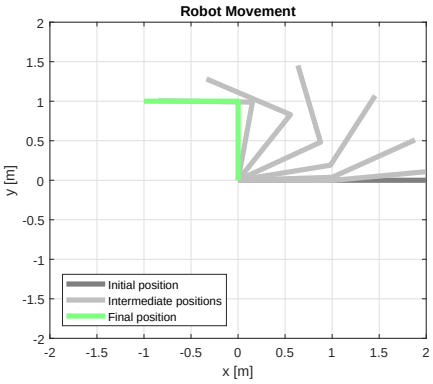
As a first simple example, the target is fixed at $(-1, 1)$ and the results are shown in Figure 4.10. The target is considered reached when the end-effector is within a distance of 10^{-4} m from the target. The planning procedure successfully guides the robot to the target and as expected, the actual realized trajectory is similar to the planned trajectory shown in Figure 4.1. Also, the intermediate planned trajectories during execution are nearly identical to the actual realized trajectory, which is as expected as the environment is static. The main difference is the switch to tracking formulation (3.9) to alleviate the issue of small time intervals ΔT_k when the end-effector is approaching the target. The tracking formulation intrude the time-optimality, which can be seen by comparing the total transition time in Figure 4.1 and Figure 4.10. The transition time is similar until the initiation of the tracking. Note, the computation time is barely notable during the tracking. However, the computation time exceeds the sampling time which is 0.1 s during the planning procedure. An alternative solution is shown in Figure 4.11, where the initial band length is set to $n = 7$. Now the computation time never exceeds the sampling time, but the total transition time is prolonged.

4.2.2 Moving Target

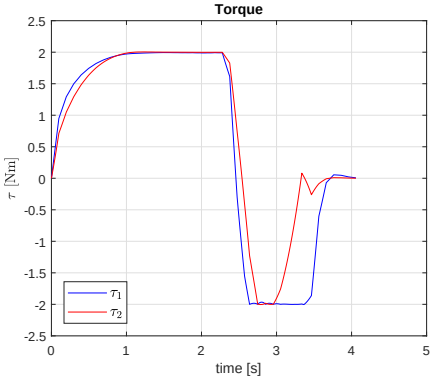
In this section, the target is moving with linear velocity. Initially the target is located at $(-1, 1)$ and moves in direction $(0, -1)$ with velocity 0.1 m/s. The result of the planning procedure is shown in Figure 4.12. First, a trajectory is planned to the initial position of the target. Then the current estimated total transition time is used to predict where the target will be located when the end-effector is approaching. Hence, the robot can reach the target with matching velocity, instead of reaching the target and then having to catch up. As mentioned in Section 3.7, this method has an apparent disadvantage as the predicted future location of the target might be located outside of the workspace of the robot and the planner will fail to find a solution.



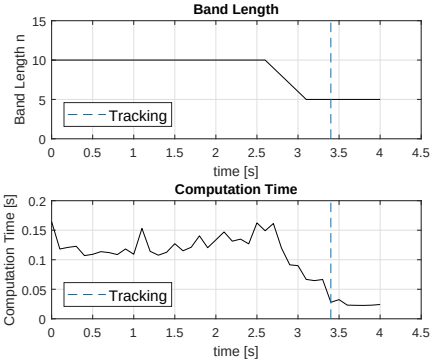
(a) *Actual realized trajectory and snapshots of intermediate planned trajectories.*



(b) *Robot movement.*

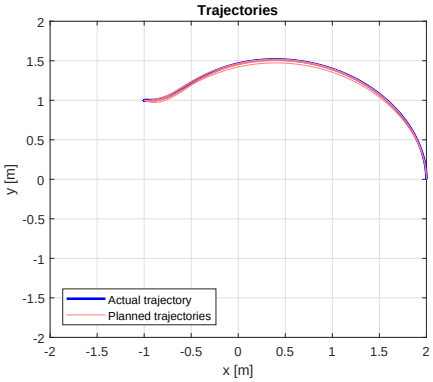


(c) *Torques required to realize the trajectory.*

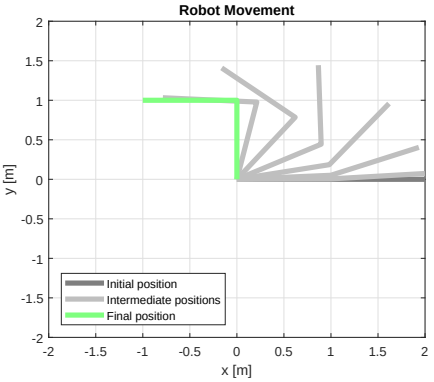


(d) *Band length, computation time and where the switch to tracking formulation (3.9) occurs during the planning procedure.*

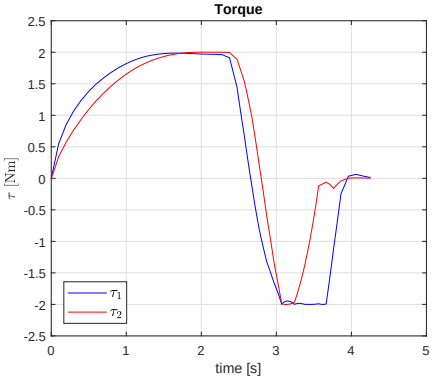
Figure 4.10: *Results of the planning procedure to a stationary target located at $(-1, 1)$ when the planner updates the trajectory during execution. The planner was configured according to Appendix A and the tracking vicinity is reached after 3.4 s and the total transition time is 4 s.*



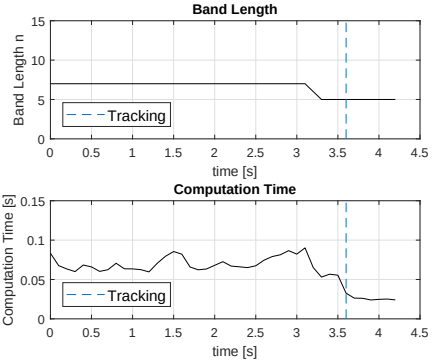
(a) Actual realized trajectory and snapshots of intermediate planned trajectories.



(b) Robot movement.

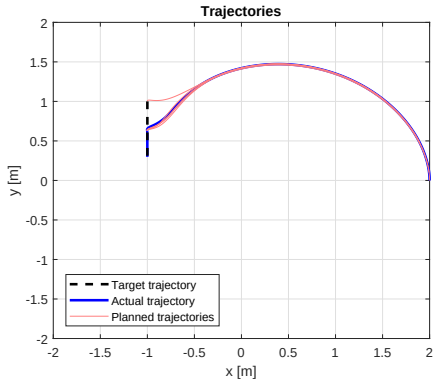


(c) Torques required to realize the trajectory.

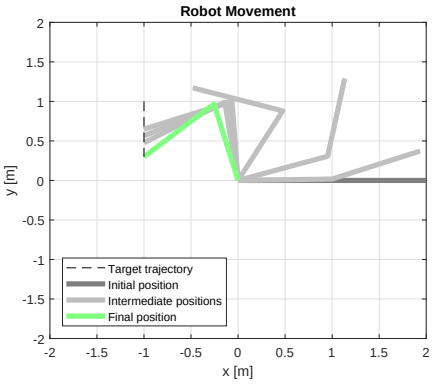


(d) Band length, computation time and where the switch to tracking formulation (3.9) occurs during the planning procedure.

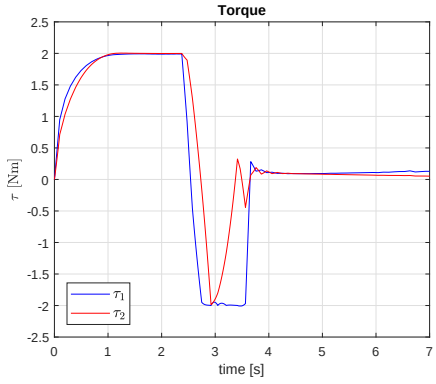
Figure 4.11: *Results of the planning procedure to a stationary target located at* $(-1, 1)$ *when the planner updates the trajectory during execution. The planner was configured according to Appendix A, but with initial band length* $n = 7$*. The tracking vicinity is reached after 3.6 s and the total transition time is 4.2 s.*



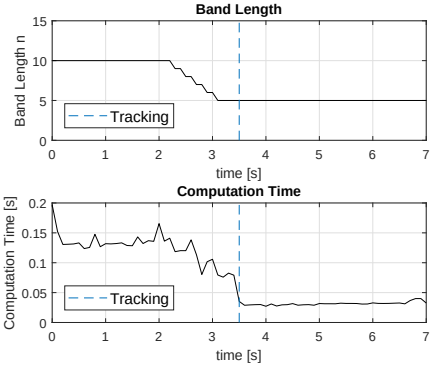
(a) Actual realized trajectory and snapshots of intermediate planned trajectories.



(b) Robot movement.



(c) Torques required to realize the trajectory.



(d) Band length and computation time during the planning procedure.

Figure 4.12: *Results of the planning procedure with a target initially located at* $(-1, 1)$ *and moving in direction* $(0, -1)$ *with velocity* 0.1 *m/s. The planner was configured according to Appendix A.*

5
Conclusions

In this thesis techniques for time optimal trajectory planning for industrial robots in dynamics environments have been investigated. In order to find optimal solutions, both the path and velocity profile is planned simultaneously while satisfying system dynamics as well as avoiding obstacles and other specified constraints. The investigated approach is based on splines where the spline nodes and knots are optimization variables. In this chapter the results are summarized and suggestions for future work are discussed.

5.1 Conclusions

To summarize the results from the previous chapter, the spline-based trajectory planner yield trajectories that seems to be close to the solution of the time optimal control problem for industrial robots (2.13). It should be noted that the spline-based approach is not restricted to industrial robots and can be applied to other systems as well. The main challenges of the proposed approach and implemented trajectory planner are the violations of inequality constraints between spline nodes and the computation time.

The method of conservative joint velocity constraints are deemed successful as it guarantees that the joint velocity bounds are satisfied between the spline nodes with only a slight increase in computation time. On the other hand, the additional joint velocity constraints might affect the time-optimality as it expresses a conservative upper/lower bound when the joint acceleration becomes zero within a time interval. Further investigation is required to evaluate this effect. Also, the results indicate that the approach of oversampling the state trajectory and including additional inequality constraints on intermediate states requires less computational time in contrast to increasing the band length. However, the choice of the number of additional inequality constraints is crucial and thus the implemented

planner is sensitive to the choice of configuration parameters (m_τ, m_{obs}). In each scenario the planner has to be tailored to the right parameters to find a feasible solution with low computation time. Especially in the cases with obstacles located in the workspace. In some cases, a few additional inequality constraints for the obstacles are sufficient to find a collision-free trajectory and in other cases more additional constraints are required. It is not trivial to distinguish these scenarios in advance and predict a sufficient number of additional constraints.

5.2 Future Work

There are several possible directions for future work. As mentioned, one of the main issues of the implemented planner is the computation time. As such, an idea is to investigate alternative approaches to solve the underlying non-linear optimization problem. For example, the procedure used in [17] could be investigated. In [17] the original problem is reformulated by expressing the equality and inequality constraints as quadratic penalty functions in the objective function, thus an unconstrained least-square problem is obtained. The dimension of the Hessian in the NLP is thereby significantly reduced as the Lagrange multipliers are avoided. Another suggestion is to investigate alternative methods to the interior-point method, such as the SQP method.

Alternative parametrisations of the splines to reduce the number of optimization variables and thus possibly reduce the computation time could also be investigated. For instance, the parametrisation used in [10] where each cubic polynomial in the spline are parametrised by the time interval length as well as the acceleration and the position in the spline nodes. Hence, there is no need to include the velocity in the spline nodes as explicit optimization variables. However, the velocity is obtained in a more complex procedure and it should be investigated if this parametrisation reduces the computation time.

Also, B-spline parametrisation could be investigated. By utilizing the convex hull property of the B-splines the spline constraints, for instance velocity, can be guaranteed to be satisfied in all time intervals by simply introducing constraints on the B-spline coefficients [11]. On the other hand, further consideration is required to guarantee that the torque bounds are satisfied.

Appendix

A

Description of Parameters

An example of a configuration file is given below and a description of the available parameters is given in Table A.1.

```
{
    "trajectoryProblem" : {
        "sampleTime" : 0.1,
        "initialBandLength" : 10,
        "nmin" : 5,
        "regularizationWeight" : 5,
        "intermediateInputConstraints" : 1,
        "intermediateObstacleConstraints" : 2,
        "unifromKnots" : false,
        "trackingVicinity" : 0.1,
        "safetyDistance" : 0.1,
        "bounds" : [
            {"type" : "Joint", "component" : 1, "lowerBound" : -6.28, "upperBound" : 6.28},
            {"type" : "Joint", "component" : 2, "lowerBound" : -3.14, "upperBound" : 3.14},
            {"type" : "JointVelocity", "component" : 1, "lowerBound" : -2, "upperBound" : 2},
            {"type" : "JointVelocity", "component" : 2, "lowerBound" : -2, "upperBound" : 2},
            {"type" : "JointJerk", "component" : 1, "lowerBound" : -10, "upperBound" : 10},
            {"type" : "JointJerk", "component" : 2, "lowerBound" : -10, "upperBound" : 10},
            {"type" : "Input", "component" : 1, "lowerBound" : -2, "upperBound" : 2},
            {"type" : "Input", "component" : 2, "lowerBound" : -2, "upperBound" : 2}
        ]
    }
}
```

Table A.1: *A description of the parameters used in the planner.*

ΔT_s	The inherent sampling time
n_{init}	The number of splines nodes in the timed elastic nodes set
n_{min}	The minimum allowed number of splines nodes in the timed elastic nodes set
λ	The regularization weight in the timed elastic nodes with non-uniform knots framework
m_τ	The number of constrained intermediate states per time interval for the torque bounds
m_{obs}	The number of constrained intermediate states per time interval for the obstacle constraints
σ_t	The distance from the target that defines the tracking vicinity
s	The safety distance that should be maintained from each obstacle

Bibliography

[1] Bradley M. Bell and Coin-Or Foundation. Cppad. https://www.coin-or.org/CppAD/. Cited on page 27.

[2] M. Biel and M. Norrlöf. Efficient trajectory reshaping in a dynamic environment. *IEEE AMC2018*, 2018. Cited on pages 3, 4, 19, 25, and 27.

[3] L. Van den Broeck, M. Diehl, and J. Swevers. A model predictive control approach for time optimal point-to-point motion control. *Mechatronics*, pages 1203–1212, 2011. Cited on page 2.

[4] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, pages 577–585, 2002. Cited on page 4.

[5] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*. Springer, Berlin, Heidelberg, 2006. ISBN 9783540361190. Cited on pages 5 and 10.

[6] P. Fiorini and Z. Shiller. Time optimal trajectory planning in dynamic environments. *Proceedings of IEEE International Conference on Robotics and Automation*, 2:1553–1558, 1996. Cited on page 1.

[7] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. Cited on page 27.

[8] Saed Al Homsi, Alexander Sherikov, Dimitar Dimitrov, and Pierre-Brice Wieber. A hierarchical approach to minimum-time control of industrial robots. *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2368–2374, 2016. Cited on page 2.

[9] Y. Kim and B. K. Kim. Time-optimal trajectory planning based on dynamics for differential-wheeled mobile robots with a geometric corridor. *IEEE Transactions on Industrial Electronics*, 64:5502–5512, 2017. Cited on page 1.

[10] Huashan Liu, Xiaobo Lai, and Wenxiang Wu. Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints. *Robotics and Computer-Integrated Manufacturing*, 29(2):309–317, 2013. Cited on page 46.

[11] Tim Mercy, Erik Hostens, and Goele Pipeleers. Online motion planning for autonomous vehicles in vast environments. *IEEE AMC2018*, 2018. Cited on page 46.

[12] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer-Verlag New York, second edition, 1999. Cited on page 3.

[13] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807, 1993. Cited on page 2.

[14] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. Trajectory modification considering dynamic constraints of autonomous robots. *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6, 2012. Cited on page 3.

[15] C. Rösmann, F. Hoffmann, and T. Bertram. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. *Control Conference (ECC), 2015 European*, pages 3352–3357, 2015. Cited on pages 3 and 4.

[16] C. Rösmann, F. Hoffmann, and T. Bertram. Planning of multiple robot trajectories in distinctive topologies. *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6, 2015. Cited on page 3.

[17] C. Rösmann, F. Hoffmann, and T. Bertram. Convergence analysis of time-optimal model predictive control under limited computational resources. *Control Conference (ECC), 2016 European*, pages 465–470, 2016. Cited on pages 3 and 46.

[18] C. Rösmann, F. Hoffmann, and T. Bertram. Kinodynamic trajectory optimization and control for car-like robots. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686, 2017. Cited on page 21.

[19] C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. Time-optimal nonlinear model predictive control with minimal control interventions. *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 19–24, 2017. Cited on page 4.

[20] C. Rösmann, A. Makarow, F. Hoffmann, and T. Bertram. Sparse shooting at adaptive temporal resolution for time-optimal model predictive control. *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 5551–5556, 2017. Cited on pages 4, 13, and 29.

[21] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, Mar 1999. Cited on page 25.

[22] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, third edition, 2005. ISBN 9780471649908. Cited on pages 5 and 23.

[23] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. Cited on pages 4 and 27.