MASTER

# Generic trajectory generation for industrial manipulators

van Dijk, N.J.M.

*Award date:*
2006

# Generic trajectory generation for industrial manipulators

N.J.M. van Dijk

DCT 2006.067

**Master of Science Thesis**

The work presented in this thesis is part of the New*Motion* project.

Committee: prof. dr. H. Nijmeijer[§] (Chairman)
dr. ir. N. van de Wouw[§]
prof. ir. O. Bosgra
ir. W. Pancras[‡]

[§] Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control

[‡] Bosch Rexroth B.V.
Business unit Electric Drives and Controls
Product area Semiconductor and medical

Eindhoven, June 13, 2006

# Preface

The work presented in this thesis is the result of a one year graduation traineeship that started at Nyquist industrial control B.V. and ended at Bosch Rexroth electric drives and control B.V.[1] The results obtained in this thesis would not be possible without the help of certain people. First of all, I would like to thank Wilco Pancras for his support throughout the past year. His insight in both theoretical and practical issues in the field of robot motion planning helped me to get familiar with the motion planning problem. Furthermore, I would like to thank professor Henk Nijmeijer and Nathan van de Wouw for the fruitful discussions and support during our monthly sessions. The discussions triggered me for being critical to obtained results and gave new energy for fulfilling my graduation assignment. Moreover, I would like to thank Nathan for his help in tackling several problems along the way and his many comments on this thesis. Without these comments the quality of this thesis would not be as high as it is now. Next, some thankful words are for Jiri Stembera, who developed and implemented the numerical inverse kinematics algorithm presented in Chapter 3 of this thesis. Last but certainly not least, I would like to thank the employees of Bosch Rexroth electric drives and control B.V. and especially the *motion engineers* (Geert, Huub, Menno, Robert, Wilco en Wim) for the help and enjoyment during my stay at Bosch Rexroth. Good luck with the further development of the NYCe4000 motion control platform. In my opinion it is a very competitive product and I would be surprised if it will not be successful.
Finally, it is interesting to see the effect of a company of around $50$ employees changing to a company of around $250,000$ employees, that is present all over the world. Furthermore, I experienced that solving robot motion planning problems is not evidently as it seems at forehand. Therefore, I would like to conclude this preface with the following quote that is applicable for my experience on robot motion planning during this graduation project:

> We can't solve problems by using the same kind of thinking we used when we created them.
> -Albert Einstein -

Niels van Dijk
Eindhoven, June 2006.

[1] During my traineeship, Bosch Rexroth electric drives and control B.V. has acquired Nyquist industrial control B.V.

# Abstract

One of the ultimate goals in robotics is the development of autonomous robots. In case of an autonomous robot, a user can specify the task a robot has to perform and the robot will then perform this task without any human intervention. One of the issues in developing autonomous robots is robot motion planning. Robot motion planning can roughly be split in two parts, namely path planning and trajectory generation. In this work, the focus lies in developing trajectory generation algorithms for a wide range of industrial manipulators, such that the design-in of motion control platforms into industrial applications can be shortened.

From literature it can be concluded that mainly two types of motions can be distinguished, namely pick and place motions and motions where the path is completely known in advance. A pick-and-place motion represents a movement of a robotic system for which a high level of freedom on the path between pick and place location exists, while for motions with a completely known path, it is desired for the end-effector to exactly follow a certain path.

A structure of a generic motion planner is proposed. The structure is based on results from literature and a requirements study. The requirements study is performed by interviewing motion engineers with a high-level of industrial experience.

Two trajectory generation algorithms are developed, that the results from the requirements study are satisfied. The first algorithm is applicable for pick-and-place cases and applications where simple contours need to be tracked. The algorithm determines linear segments through a set of user-defined points determined in the workspace of a manipulator. Furthermore, to prevent that the robot needs to stop at every point a user can specify spheres around each point where it is allowed to deviate from a linear segment. The second algorithm is applicable for path constrained motions. The problem results in a optimization strategy, which is solved by a combination of a genetic algorithm and a local optimization method.

The performance of the developed algorithms, compared with algorithms available from literature, is evaluated by means of simulations and experiments. Several testcases are developed such that a comparison between different algorithms can be made. Based on the results of the simulations and experiments it can be said that the developed algorithms are favorable above algorithms presented in literature. However, the choice between the algorithms depends on the application for which the algorithm will be used.

# Samenvatting

Een van de ultieme doelen in robotica is de ontwikkeling van autonome robotische systemen. Er wordt gesproken over een autonoom robot systeem wanneer een, door een gebruiker gespecificeerde, taak kan worden uitgevoerd zonder enige menselijke inbreng. Een van de punten om tot autonome robots te komen is het plannen van robotische bewegingen. Dit plannen van robotische bewegingen kan grofweg verdeeld worden in twee groepen, namelijk pad planning en traject generatie. In dit werk ligt de nadruk op het ontwikkelen van traject generatie algoritmes welke toepasbaar zijn voor een groot scala van industriële manipulatoren. Op deze manier kan het ontwerp traject van industriële regelaars voor industriële applicaties aanzienlijk worden verkort.

Uit de literatuur kan worden geconcludeerd dat twee type bewegingen zijn te onderscheiden, namelijk 'pick-and-place' bewegingen en bewegingen waarvan het pad vooraf helemaal bekend is. Een 'pick-and-place' beweging representeert een beweging van een robot systeem waar veel vrijheid bestaat tussen oppak en afleg locatie, terwijl voor een beweging met een vooraf bekend pad, de robot hand een specifiek pad exact moet volgen.

Een structuur voor een generieke beweging's planner is voorgesteld. De structuur is gebaseerd op resultaten uit de literatuur en een markt-onderzoek naar de eisen aan een dergelijke beweging's planner. Het markt-onderzoek is uitgevoerd door het afnemen van interviews met ingenieurs met veel industriële ervaring op het gebied van regeltechniek en robotica.

Twee traject generatie algoritmes zijn ontwikkeld zodanig dat wordt voldaan aan de resultaten van het markt-onderzoek. Het eerste algoritme is toepasbaar voor 'pick-and-place' toepassingen en toepassingen waar eenvoudige contouren gevolgd moeten worden. Het algoritme bepaald lineaire segmenten door een set van punten gegeven door een gebruiker, welke gespecificeerd worden in de werkruimte van een robot. Om te voorkomen dat op elk punt gestopt dient te worden, kan een gebruiker bollen rondom elk punt specificeren waarin afgeweken mag worden van een linear segment. Het tweede algoritme is toepasbaar voor problemen waar de robot hand exact een bepaald pad moet volgen. Dit resulteert in een optimalisatie probleem dat wordt opgelost door een combinatie van een genetisch algoritme en een lokale optimalisatie methode.

De prestaties van de ontwikkelde algoritmes zijn onderzocht door middel van simulaties en experimenten en de verkregen resultaten zijn vergeleken met algoritmes uit de literatuur. Diverse test situaties zijn ontwikkeld zodanig dat een vergelijk tussen verschillende algoritmes kan worden gemaakt. Uit de resultaten kan worden geconcludeerd dat de ontwikkelde algoritmes te prefereren zijn boven algoritmes uit de literatuur. De uiteindelijke keuze tussen de algoritmes is afhankelijk van de uiteindelijke toepassing waarvoor het algoritme gebruikt zal worden.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

A robot is a mechanical device equipped with actuators and sensors operating in a workspace, performing a certain task. The task a robot has to perform can be for example, welding two metal pieces to each other, picking an IC from a conveyor belt and placing it on a circuit board, milling a workpiece, moving products through a factory or even playing soccer with a team of robots. For carrying out a specific task, the robot can either be a mobile robot or a fixed base manipulator. In this thesis, we will only discuss issues regarding fixed base manipulators that merely can be found in an industrial environment. Since the early 1970s the first robotic systems have appeared in an industrial environment. The first microcomputer-controlled full electrical robot was introduced in 1974 by ASEA, see Figure 1.1.

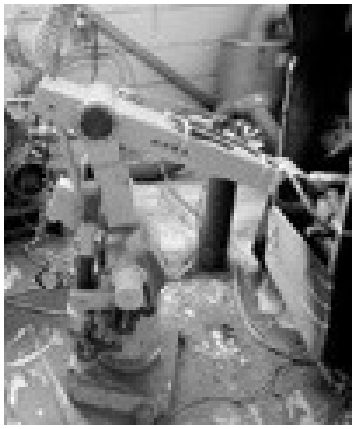


Figure 1.1: First microcomputer-controlled industrial robot (Courtesy of the AAB group).

One of the ultimate goals in robotics is to create autonomous robots [Sharir, 1989; Bruyninckx, 2004]. In case of an autonomous robot, a user can specify the task a robot has to perform and the robot will then perform this task without any human intervention. In developing autonomous robots, two major aspects can be distinguished, namely; robot motion planning and robot motion control (see Figure 1.2). The purpose of robot motion planning is to guide a robotic system through a field of (moving) obstacles, present in the workspace of the system, while respecting

the physical limitations of the system. The result of such motion planning is a set of profiles (desired trajectories) for each actuator of the system. These profiles serve as an input to the robot motion controller. Robot motion control has to make sure that given desired trajectories, the robot tracks these desired profiles with an accuracy as high as possible such that the system is robust for disturbances and possible unmodelled system properties. A huge amount of work has been done in the field of robot motion control. For a completer overview we refer to [Craig, 1986; Sciavicco, 1996; Kurfess, 2005]. The research in this report is focussed on developing robot motion planning algorithms.

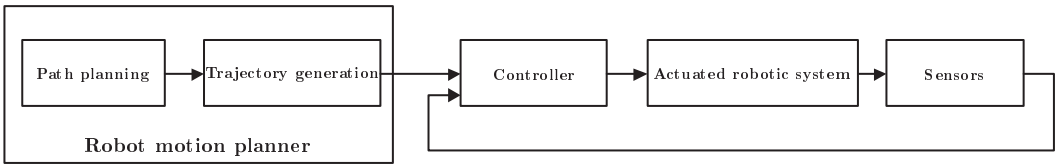



Figure 1.2: Components of a robotic system.

As stated above, several tasks can be performed with industrial manipulators. Each type of task asks for a specific planning solution. Each planning solution must result in an exact realisation of the prescribed task. Furthermore, in most cases the task must fulfill a certain performance criterium such as, for example, time-optimality[1] or energy-optimality. From a practical point of view it is not desired to design a specific trajectory generator for each application. The question then arises whether it is possible to design a generic trajectory generator, which can deal with several type of tasks and is able to deal with the (physical) limitations imposed by the application.

The robot motion planning problem can roughly be split into two parts, namely path planning and trajectory generation. Path planning handles the planning of a spatial path from the robot's initial configuration to the robot's goal configuration. During the transition from the initial to the goal configuration, no collisions may occur between the robot and (moving) objects present in the workspace of the robot. Furthermore, when a robotic systems consists of more than one link, no collisions may occur between each of the links. Trajectory generation consists of deciding on the velocity, acceleration and jerk profiles along the planned path as a function of time. The generated profiles must not exceed limitations on velocity, acceleration and jerks and in most cases must fulfill a certain performance criterium as stated above.

Since our main interest lies in developing generic trajectory generation algorithms, we assume that a collision free path is planned or can be constructed through a number of points. So, we are not interested in the path planning part. For more details on path planning we refer to [Latombe, 1991; LaValle, 2005; Ko, 1992; Keij, 2003; Sharir, 1989; Khatib, 1980; Canny, 1988; Borenstein, 1989; Mirolo, 1991; Yamamoto, 1994; Meng, 1988; Wang, 2004; Fan, 1994; Pajak, 2004; Rimon, 1992; Dijkstra, 1959; Ashiru, 1995; Davidor, 1990]. In the following section the problem statement will be discussed. After that the contributions of this thesis will be elaborated, and finally the outline of the thesis is presented.

[1] A trajectory is said to be time-optimal when during the trajectory, one of the constraints is always active [Žlajpah, 1996].

## 1.2 Problem statement

In this section, we will briefly discuss issues on generic trajectory generation before presenting the problem statement. A more extensive literature review on generic motion planning can be found in Chapter 2.
When considering a robotic system and the specific task it must perform, we can distinguish between two types of movements. Firstly, movements can be specified in which only certain points in the workspace of a manipulator must be visited and, secondly, movements can be specified in which a certain path needs to be followed exactly. For the first case we say that it represents a pick-and-place motion, while the second case represents an operation which is called path-constrained motion.

A pick and place motion represents a movement of a robotic system for which in principle, two desired positions are given. Namely, a pick position (i.e. the position where e.g. a product needs to be picked) and a place position (i.e. the position where e.g. a product needs to be placed). Such motions planning problems are typically seen in e.g. component mounting and packaging applications. Clearly, freedom in the design of the path exists between the pick and place position. As stated in the previous section, we are not interested in path planning algorithms. Therefore, the user of the motion planner is seen as the high-level path planner (providing both pick and place positions and possible intermediate positions to avoid collisions). Several algorithms can be found in literature that address this type of trajectory generation problem. However, the main problem in the obtained solutions is that the requirements on motion planner are taken into account too conservative or too rigorously. With this it is meant that e.g. a motion profile has an order that is larger (in case of a conservative solution) or smaller (in case of a too rigorous solution) than the required minimum profile order. Moreover, the solutions either account for geometric constraints or constraints on velocity, acceleration and possibly jerks acting in one domain (i.e. either in the workspace of a manipulator or on joint level).

In the case of path constrained motions, it is common to present the path as a function of a so-called path parameter. So the motion can be described in one degree-of-freedom [Pfeiffer, 1987]. Process and actuator constraints can be described as function of the path parameter. Several algorithms are available from literature that solve the path constrained motion problem. However, in most cases this results in motions that are not second-order continuous, so instant changes in torque levels occur. An instant change in torque level can, in practice, not be realized by an actuator due to it's electrical and mechanical dynamics. Furthermore, actuator limitations are mostly accounted for as maximum applicable torques (and in some cases also torque rates). This implies that dynamics equations of a robotic system must be known. This asks for a specific amount of knowledge of the system. However, from a practical point of view, this is not desired. Path constrained motions are typically addressed for in e.g. laser cutting and wafer inspection applications.

From the discussion above, we can conclude that each type of operation gives a different set of requirements on a trajectory. This will thus result in different trajectories. Moreover, the main results from literature are applicable only for specific applications, i.e. generally for pick-and-place operations or operations where a specific (complex) path needs to be tracked. From a manufacturers point of view this is not desirable, because for each application a new motion planner needs to be designed. This asks for a generic approach in motion planning. In order to be able to generate trajectories for a wide range of applications, algorithms must be developed which can cope with

applications with various requirements and must be applicable in an industrial environment. So, this leads to the following problem statement:

*Design and develop one (or more) trajectory generation algorithm(s) for pick and place motions and path constrained motions, which is (are) applicable for a wide range of (industrial) applications, regarding a set of industrial performance requirements and constraints.*

## 1.3 Contribution of the thesis

The main contribution of this thesis lies in the development of a generic trajectory generator. With 'generic' it is denoted that the motion planner should be able to calculate trajectories for a wide range of manipulators that have to perform different operations. As becomes clear from the discussion above, a large amount of work has been done in the field of robot motion planning (for a detailed literature overview, we refer to Chapter 2). Most of the developed algorithms are developed in a research environment. In this work the focus lies in developing algorithms for application in an industrial environment.

The proposed trajectory generator algorithms are developed while respecting practical usefulness. Often, the dynamics of robotics systems are evaluated to transform constraints at different levels (e.g. actuator space and/or Cartesian space) to one space. In this work, only the kinematic equations of the system are evaluated to obtain constraints in one space. The reason for this choice is the fact that the derivation of the dynamic equations of a robotic system requires specific knowledge on multibody dynamics, while the (forward) kinematic equations can be determined in a straightforward manner. This is not the case for the inverse kinematic description of a system. Therefore, a numerical algorithm is developed which determines the inverse kinematic description for the given system. Clearly, the fact that no a priori knowledge on the dynamics of the robot is required from the user adds significantly to user-friendliness of this approach.
Another important issue in trajectory generation is profile smoothness. A non-smooth profile will often result in manipulator wear. However, a smooth profile will result in motions that are not relatively fast, which is very important in case of pick-and-place applications. In this work, a compromise between fast motions and smooth motions is made.
Path constrained motion planning methods which fulfill industrial requirements will result in optimization strategies that are non-convex (as will be clear discussed in detail later on in this thesis). Present algorithms which solve theses type of motion planning problems do not present solutions that are able to deal with this fact. In this work a global optimization strategy for finding optimal solution for this type of motions is presented.

The performance evaluation and validation of trajectory generator algorithms is very application dependant. Typical performance evaluation criteria are total motion time and path following accuracy. Until now, no clear performance evaluation strategy is known. In this thesis an evaluation matrix is proposed, such that several trajectory generator algorithms can be evaluated and compared.

## 1.4 Outline of the thesis

In Chapter 2, a detailed literature overview on generic motion planning can be found. Chapter 3 presents an overview of the structure of the proposed generic motion planner. Before discussing this structure, results of a requirements study are presented. The requirements study is performed in order to obtain insight in the market demands for an industrial motion planner. The outcome of the requirements study provides boundary conditions on the structure of the generic motion planner, which is elaborated in Section 3.3.

After discussing the structure of the generic motion planner, algorithms for generating trajectories are presented in Chapter 4 and Chapter 5. In Chapter 4, an algorithm is presented that is able to generate pick-and-place motions, while Chapter 5 discusses an algorithm for determining path-constrained motions. The developed algorithms are mainly focused on (near) time-optimal motions.

After discussing the developed algorithms, Chapter 6 discusses the results of simulations and experiments performed using the developed trajectory generation algorithms. Before discussing the actual results, first the proposed evaluation matrix is presented. The matrix is used for evaluating the performance of different trajectory generator algorithms. The results are compared with results obtained with other trajectory generator algorithms by using the evaluation matrix. Finally conclusions and recommendations are presented in Chapter 7.

# Chapter 2

# Literature review on generic motion planning

## 2.1 Introduction

This chapter presents an overview of the literature available on generic motion planning. In the previous chapter, the basic results are already discussed. However, here the results will be elaborated in more detail. First, pick and place motions will be discussed. Next, we will discuss the literature on path constrained motions.

## 2.2 Pick-and-place motion

A pick-and-place motion represents a movement of a robotic system where a high level of freedom exist on the path between pick and place location. In some cases, the user, who specifies the motion, acts a high-level path planner. The input to a pick and place motion typically consists of two points and the task is to travel as fast as possible between these two points. Clearly, a motion planner must be able to plan paths without any collisions between manipulator and (moving) objects in the workspace of a manipulator. We refer to such planning of movement from pick to place location while avoiding collisions as high-level planning. Pick-and-place motions can for example be found in applications where a product must be placed into a storage device (such as a carton box) or movements between two welding/cutting operations are pursued. In some particular cases, it is possible that during a movement an update on the end position is given. This implies that the path needs to be recalculated *on the fly*, and consequently that the trajectories must be calculated online (i.e. during the movement of the robotic system). Another example where online trajectory generation is desired is the case in which segments are added to the path while the system is moving.

During the design of algorithms for online trajectory generation, assumptions or simplifications are made due to the fact that the kinematics and dynamics of a robotic system can be extremely complex [Park, 2005]. Furthermore, geometric simplifications are made by modeling robot links by lines [Liang, 1999; Zah, 2004]. These assumptions or simplifications are made in order to reduce the computational time needed for the trajectory generation.

For most applications, it is desired to determine the motion of a manipulator in the workspace of

the manipulator. In case of complex kinematical structures, motion planning in the workspace of the manipulator can be computationally very inefficient. However, in case of pick-and-place motions, where the initial- and end-configuration of the manipulator are specified, it is common to first transform these configurations, which are defined in the workspace of the manipulator, to joint variables by means of an inverse kinematic transformation [Sciavicco, 1996; Lin, 1983; Chang, 1992; Piazzi, 2000]. Several solutions then exist to determine joint trajectories which fulfill a performance criterium, such as time-optimality. Here, we will first discuss the case where a manipulator must move from an initial- to a final-configuration (known as the point-to-point motion). Next, we will discuss motion where also intermediate points are specified.

As stated above, point-to-point motions are motions where a manipulator moves from an initial configuration to a final configuration. The trajectory must respect certain constraints and must fulfill a performance criterium. For productional or packing manipulators a high throughput is desired such that costs of the process are minimized. Therefore, the most common performance criterium for pick-and-place motions is the minimum time criterium[1]. A method for obtaining smooth motions between two points is presented by Macfarlane [Macfarlane, 2003]. Here a concatenation of fifth-order polynomials (quintics) is used to obtain motions which are third-order continuous. Fifth-order polynomials are the lowest order polynomials for which it is possible to specify the begin and end point positions, velocities and accelerations. In order to prevent oscillating behaviour of the profile, a sine wave template is used to approximate the jerk profile such that the knotpoints (which are the points through which the spline passes) of the profile can be determined. The obtained point-to-point motions are near time-optimal. Lambrechts [Lambrechts, 2003] presents methods to determine time-optimal point-to-point trajectories. Equations for second-order and third-order continuous trajectories are derived analytically. Furthermore, an algorithm for higher-order point-to-point trajectory planning is given and time-optimal motion, while respecting constraints on each time-derivative of the position trajectory up to the order of the trajectory (i.e. velocity, acceleration and jerk in case of a trajectory that has orde three), is shown for all relevant cases.

In several applications, the motion of a manipulator is described by more than two points. Even for pick-and-place motions this can be the case, for example, when products must be placed into a box. Splines are probably the most regularly used functions for such applications. In the early history of spline theory, the following definition of a spline was formulated [Schoenberg, 1946]:

**Definition 2.1 (Spline)** *A spline function of order $k$ is defined to be a piecewise polynomial function of order $k$ on some (finite or infinite) interval with $(k-2)$ continuous derivatives.*

In other words, a "spline function" is defined as a piecewise polynomial function which is as smooth as it could be without simply reducing to a polynomial. However, it was found that piecewise polynomial functions with a less than this maximum smoothness are also quite interesting and useful (e.g. cubic Hermite splines and cubic Bessel splines are sometimes preferred above natural cubic splines). An adapted definition of a spline function is given by de Boor [de Boor, 1978]:

**Definition 2.2 (Spline (revised))** *A spline function of order $k$ with knot sequence $t$ is any linear combination of B-splines of order $k$ for the knot sequence $t$.*

[1] Sometimes, the minimum time criterium is combined with other performance criteria (such as energy), where weighing coefficients are assigned to specify the importance of each criterium.

In Definition 2.2 B-splines are a generalization of the Bezier curve[2]. With knot sequence the points through which the spline passes is denoted.

Cubic splines are often used for (smooth) trajectory generation, [Macfarlane, 2003]. The reason for this fact is that they can assure continuity of the velocity and acceleration, [Lin, 1983], and prevent large oscillations of the trajectory to occur, which can be a result of higher-order polynomials, [Sciavicco, 1996]. However, also higher-order splines are used. Park and Bobrow [Park, 2005] use linear combinations of finite-term quintic B-splines for representing joint displacements. If a sufficiently number of B-spline terms is used, the result will approximate the solution exactly. However, increasing the number of terms, leads to a proportional increase of CPU time.

A simplification in trajectory generation can be achieved by interpolating between $N$ path points using linear segments. In order to avoid discontinuities in velocity and/or acceleration at the path points, the (joint) trajectories can be blended. However, by blending the trajectories, we cannot assure that the trajectory passes through a path point exactly. While for certain applications (such as welding or cutting operations) it is desired to pass through a certain path point, still for many applications blending is allowed. A method for blending linear segments, which can ensure that a path point will be reached, is presented by Lloyd and Hayward [Lloyd, 1993].The blending method presented by Lloyd and Hayward is used by Macfarlane and Croft [Macfarlane, 2003], to blend a concatenation of fifth-order polynomials, which is near time-optimal.

Comparing splines with linear segments with blend trajectories, it can be said that for splines, the path is in general not predictable in advance. Furthermore, the motions (except for some specific cases) are near time-optimal, see [Lin, 1983]. Linear segments with blends provide a predictable path and only during blend-transition the motion are near time-optimal. When regarding computational issues, it can be said that both spline motions and linear segment with blends are computational efficient.

In this section, we have discussed a number of methods for determining so-called pick-and-place motions. For these type of motions the movement between points is not specified a priori. However, for certain applications it is desired to exactly follow a certain path. These type of motions are discussed in the next section.

## 2.3 Path-constrained motion

This section discusses methods available for determining profiles for applications where for the end-effector is desired to exactly follow a certain path. These type of motions are called path-constrained motions, since the desired path introduces a constraint on the motion of a manipulator. Examples of applications where constraints act on the path are cutting products from metal plates, milling three-dimensional objects and moving a wafer while inspecting it with an electron microscope. The path is often described as function of a so-called path parameter. This implies that the motion can be described in one degree-of-freedom [Pfeiffer, 1987]. Extensive work has been done in the field of trajectory generation for a path constrained motion. Here we will discuss the basic theory behind trajectory generation for path-constrained motions.

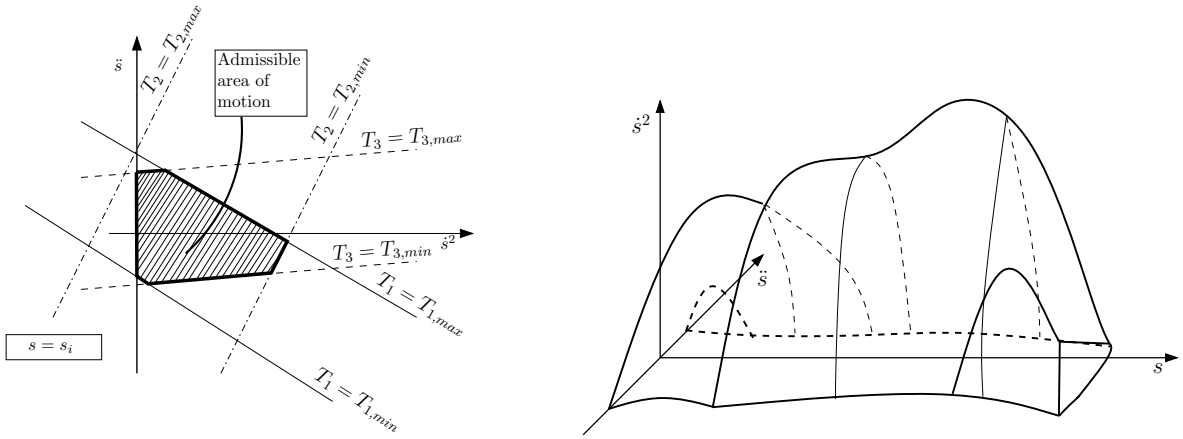
[2]For more information about Bezier curves we refer to [de Boor, 1978].

As discussed above, the problem is to determine trajectories for a prescribed path while respecting additional constraints. These additional constraints can either be given as maximum (and minimum) torque which can produced by an actuator ( [Shin, 1985], [Bobrow, 1985], [Pfeiffer, 1987], [Shiller, 1990], [Slotine, 1989]), or maximum velocities and accelerations that can be produced by the actuators or that are allowed by the process [van Tuijl, 1991]. Sometimes besides maximum actuator torques, also maximum actuator torque rates are taken into account [Constantinescu, 2000]. For the case in which torque (and possibly torque rate) constraints are specified, the dynamics of the robotic system must be evaluated. For the case in which the actuator constraints are represented as minimum and maximum allowable velocities and accelerations, the kinematic relations of the manipulator are used to rewrite the actuator constraints to constraints as function of the path parameter $s$. The relation between the path, as function of the path parameter, and the joint angles can be defined using the kinematic relations,

$$\underline{R}(\underline{q}) = \underline{P}(s), \tag{2.1}$$

where $R(\underline{q})$ represents the forward kinematics[3] (e.g. the position of the end-effector of the manipulator) of the robotic system as function of joint angles $\underline{q}$ and $\underline{P}(s)$ represents the same forward kinematics as function of the path parameter $s$.
By using either the dynamics of the system (in case of torque (and torque rate) constraints) or the kinematics of the system (in case of velocity and acceleration constraints), the constraints can be translated into constraints in terms of the path parameter and corresponding derivatives (which are functions of time). This results in a multidimensional phase space in which the motion must take place. An example of such a phase space can be found in Figure 2.1. In Figure 2.1a, an example of the polygon of admissible motion for a three-link manipulator, at a certain point $s_i$ of the path $\underline{P}(s)$ can be found. The polygon is constructed by the determining (straight) lines of maximum and minimum applicable torque as function of $\dot{s}^2$ and $\ddot{s}$, for each joint. In this case this results in six lines. Furthermore, the fact that $\dot{s}$ must be larger than zero is used to construct the polygon. Constructing the polygon of admissible motion for each $s$ results in the three-dimensional phase space as situated in Figure 2.1b.
The constrained phase space is simply connected. However, as stated by Shin and McKay [Shin, 1985], when viscous friction is taken into account in the dynamics of a manipulator, this can result in 'islands' in the constrained phase space. The problem for determining the trajectories, i.e. finding $s(t)$, is solved in the $(s, \dot{s})-$ space. In this phase space, lines of maximum acceleration and deceleration are drawn (see Figure 2.2).
The various algorithms presented in literature are rather similar. The first step in these algorithms is to start integrating from the initial condition ($s_0$, $\dot{s}_0$), which clearly must lie in the admissible phase space, with maximum acceleration until the boundary of the admissible region (boundary curve $g(s)$ is reached or the final value of the path parameter, $s = s_f$, is reached. Then, integrate backward, starting from the final condition, i.e. $(s_f, \dot{s}_f)$, and integrate until the boundary of the admissible region is reached or $s$ becomes less then zero. When the two curves intersect, the time-optimal trajectory is found, see Figure 2.3a.
When the two curves do not intersect (see Figure 2.3b), a search for a 'switching point' along the boundary curve is performed. A switching-point can be either a critical point, a tangency point or a discontinuous point at the boundary curve [Slotine, 1989]. At a critical point (or sometimes called a zero-inertia point), the acceleration at the limit curve is not unique. At a tangency point, the slope of the trajectory is equal to the slope of the boundary curve. This is the case when

[3] Sometimes the forward kinematics are referred to as direct kinematics.




(a) Polygon of admissible motion for fixed $s$.

(b) Constrained phase space.

Figure 2.1: Phase plane example for manipulator with three joints [Pfeiffer, 1987].

an extra actuator is saturated [Slotine, 1989]. Note that in order to be time-optimal, one of the actuators is already saturated [Bobrow, 1985]. Finally, discontinuous points exist when $\frac{\partial^2 P}{\partial s^2}$ is discontinuous. The several different switching-points are indicated in Figure 2.3c. At least one switching point must exist between the initial and final condition. This is the case because, as described in detail by Pfeiffer and Johanni [Pfeiffer, 1987], the forward integrated curve with maximum acceleration and backward integrated curve reach the limit curve. At the limit curve three situations occur (as can be seen from Figure 2.4), $\ddot{s}(\dot{s} = \dot{s}_{max}) > 0$, $\ddot{s}(\dot{s} = \dot{s}_{max}) < 0$ and $\ddot{s}$ is not uniquely defined. When the curve of maximum acceleration touches the limit curve, the situation, depicted in Figure 2.4a, is active, while Figure 2.4c presents the case when the maximum deceleration curve touches the limit curve. Pfeiffer and Johanni [Pfeiffer, 1987] denote the part of the limit curve where holds that,

$$\ddot{s}(\dot{s} = \dot{s}_{max}) > 0, \tag{2.2}$$

a trajectory sink (the acceleration forces the curve into the forbidden region above the velocity limit curve), and the part of the limit curve where,

$$\ddot{s}(\dot{s} = \dot{s}_{max}) < 0, \tag{2.3}$$

a trajectory sink (the curve is directed away from the forbidden region). Between these parts of the limit curve a transition point is present. At this point the acceleration at maximum velocity is not uniquely defined (see Figure 2.4b). So, at this point a switch between maximum acceleration and maximum deceleration must be made. Therefore, this point is denoted a switching point. From a switching point, a backward deceleration curve with maximum deceleration is constructed and a forward acceleration curve with maximum acceleration is constructed. If the forward acceleration curve intersects the final deceleration curve the time optimal trajectory is found. Otherwise, a next switching point is searched and the procedure described above is repeated (see Figure 2.3c).

The algorithms presented in literature, differ in finding the switching points. Bobrow et al.

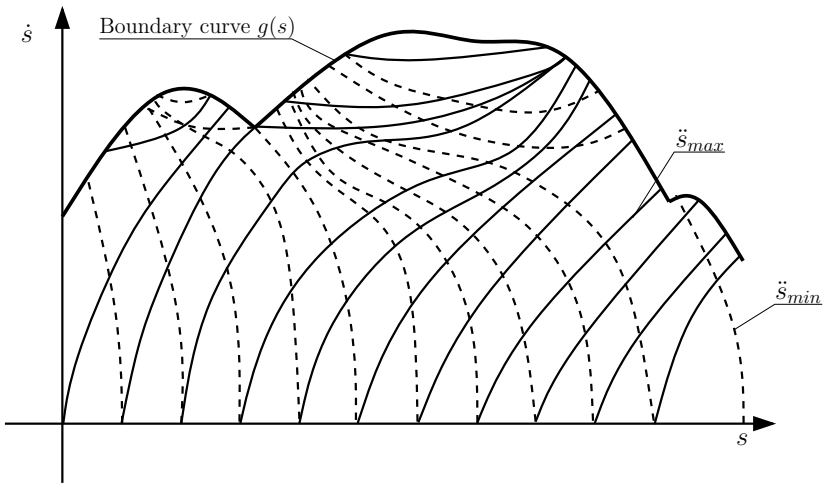


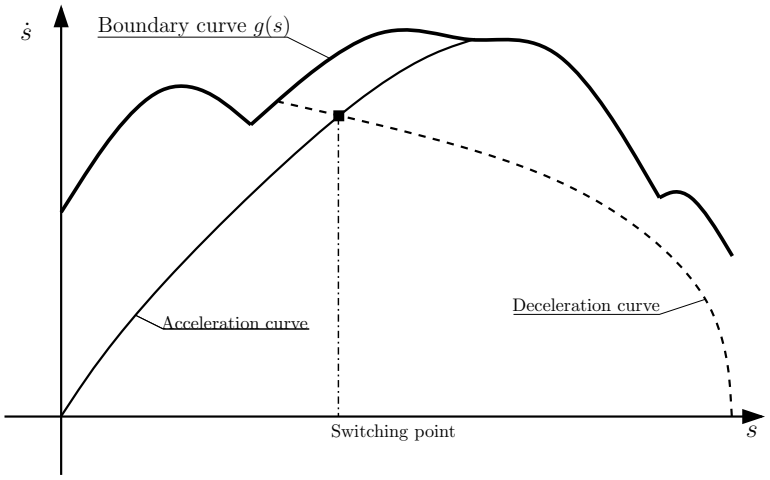

Figure 2.2: $(s, \dot{s})-$ phase space. The thick solid line represents the boundary curve of the constrained phase-space, the thin solid line represents curves of maximum acceleration while the dashed line represents curves of maximum deceleration. Figure inspired by [Pfeiffer, 1987].

[Bobrow, 1985] use an iterative search method to find a switching point. After constructing the acceleration curve from the initial condition $(s_0, \dot{s}_0)$ to the boundary curve, they lower the velocity $\dot{s}$, at the position where the curve touches the boundary curve, and integrate forward until the boundary curve is reached. The velocity is lowered until a curve is constructed which becomes tangent to the boundary curve. The point at which the acceleration curve is tangent to the boundary curve is a switching point. Shin and McKay [Shin, 1985] and Pfeiffer and Johanni [Pfeiffer, 1987] use similar methods to find switching points. They search along the boundary curve, starting at the point where the acceleration trajectory touches the boundary curve, for a point where the difference between the slope of the phase-plane trajectory and the slope of the boundary curve changes sign:

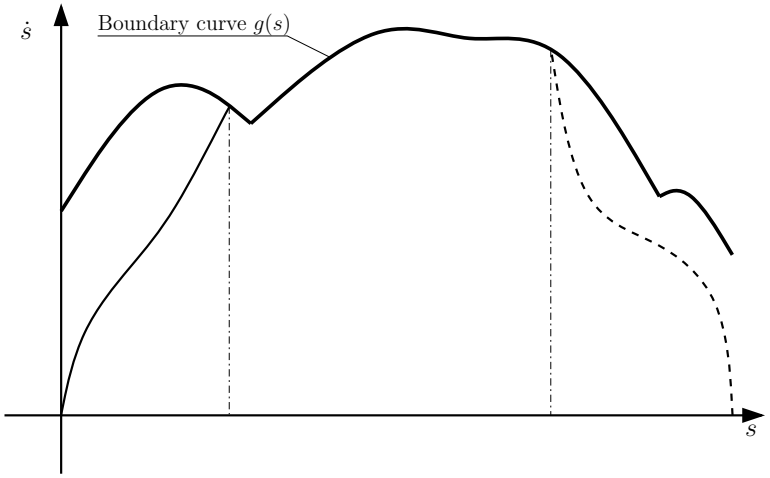
$$\kappa(s) = \frac{d\dot{s}}{ds} - \frac{dg(s)}{ds}. \tag{2.4}$$

Where, $\kappa(s)$ represents the slope difference and $g(s)$ presents the boundary curve. Slotine and Yang [Slotine, 1989] and Shiller and Lu [Shiller, 1990] present improvements in determining the switching points, in order to make the search for switching points more robust and computationally efficient.

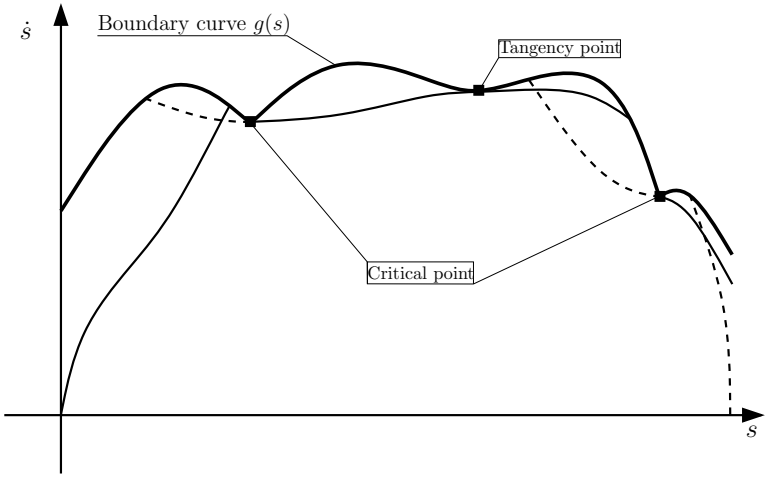
The algorithms discussed above result in trajectories which are not $C^2$ (i.e. not to times continuous differentiable), because the resulting torque trajectories simply switch between the maximum torque (acceleration) and minimum torque (maximum deceleration). In practice, however, an instant switch between maximum and minimum torque cannot be realized due to the dynamics of the actuator. So, an extra limitation must be taken into account, besides the maximum torques or forces, that can deal with this unmodelled dynamics. Constantinescu and Croft [Constantinescu, 2000] determine time optimal path-constrained motions, subjected not only to actuator torque limits but also to constraints on the first derivative of actuator torques, i.e. torque rates. Therefore, not only the dynamic equations of the manipulator are needed, but also its derivative with respect to time.




(a) Case when accelerating and decelerating curves intersect.

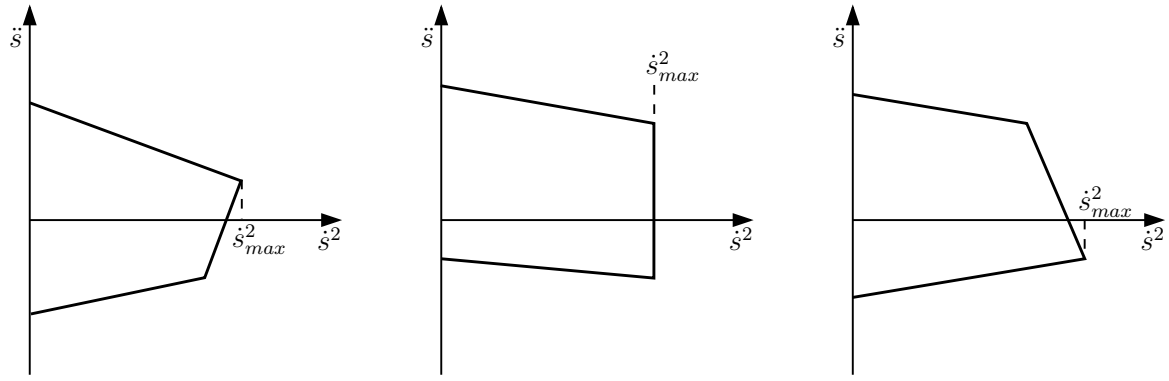



(b) Case when acceleration and deceleration curves do not intersect.




(c) Time optimal trajectory with two critical points.

Figure 2.3: Three steps for constructing a time-optimal trajectory. Figures inspired by [Shin, 1985], [Pfeiffer, 1987] and [Slotine, 1989].




(a) $\ddot{s}(\dot{s} = \dot{s}_{max}) > 0$.

(b) $\ddot{s}(\dot{s} = \dot{s}_{max})$ is not uniquely defined.

(c) $\ddot{s}(\dot{s} = \dot{s}_{max}) < 0$.

Figure 2.4: Polygons for trajectory sinks and sources. Figures inspired by [Pfeiffer, 1987]

The trajectory again is determined in the $(s, \dot{s})-$phase plane. First the switching points as described above are determined. Together with the slope $\frac{d\dot{s}}{ds}$ at $s_0$ and $s_f$, these points form the variables of an optimization problem. The optimization problem is subjected to the torque and torque rate constraints. The goal of the optimization problem is to minimize the difference between a trajectory with unlimited torque rates (as found by methods discussed above) and limited torque rates. The optimization problem is solved using the flexible tolerance method. Simulations show that with a certain choice of the torque rate limits, the method gives shorter motion times, than without torque rate limits. This can be explained by realizing that by not incorporating jerk constraints, the manipulator actual traveled path length will be larger than when jerk constraints are taken into account.

## 2.4 Discussion

In this chapter, an overview on the literature in the field of robot motion planning is discussed. First, pick and place motions are discussed. A pick-and-place motion represents a movement of a robotic system for which a high level of freedom on the path between pick and place location exists. For most applications, it is desired to determine the motion of a manipulator in the workspace of the manipulator. Cubic splines are often used to represent pick and place motions. A simplification can be made by using linear segments with blends. Comparing splines with linear segments with blend trajectories it can be said that, for splines, the path is in general not predictable in advance. Furthermore, the motions (except for some specific cases) are only near time-optimal, see [Lin, 1983]. Linear segments with blends provide a predictable path and only during the blend-transitions the motion is only near time-optimal. When regarding computational issues, it can be said that both spline motions and linear segments with blends are computational efficient.

Secondly, methods available for determining profiles for applications where it is desired for the end-effector to exactly follow a certain path are discussed. By using either the dynamics of the system (in case of torque (and torque rate) constraints) or the kinematics of the system (in case of velocity and acceleration constraints), the constraints can be translated into constraints in terms of the path parameter and corresponding derivatives. This results in a multidimensional phase space in which the motion must take place. The algorithms for obtaining time-optimal path constrained motions, discussed above, are similar. However, only one solution in which the trajectories have continuous acceleration trajectories is discussed.

# Chapter 3

# Generic motion planning

## 3.1 Introduction

In order to develop a generic motion planner, it must be clear what conditions must be fulfilled such that generic motions can be determined. Therefore, a requirements study is performed such that insight in these conditions is gathered. The results of this study are obtained by interviewing engineers with a high-level of industrial experience on motion planning and motion control of industrial manipulators. The list of interviewed people and typical questions, can be found in Appendix A. After discussing the requirements on generic motion planning in Section 3.2, a structure for a generic motion planner is proposed in Section 3.3.

## 3.2 Requirements on generic motion planning

This section elaborates the results of the requirements study performed. The set of requirements can be separated into five groups, namely;

- Constraint handling,
- Profile order,
- Path following accuracy,
- Input and output specification of the motion planner,
- I/O handling.

Each of these requirements will be elaborated in the sections below.

### 3.2.1 Constraint handling

As already stated above, trajectories generated by a motion planner must fulfill several conditions. An important part of these conditions consists of constraint handling. Constraints are conditions which can limit the behaviour of a manipulator (i.e. constraints are a result of the process which is performed with the manipulator), or constraints are imposed by physical limitations of the manipulator itself. Furthermore, we can distinguish constraints at different levels. Firstly, constraints can be active on the path planning level. Secondly, constraints can act on the trajectory

generation level. Finally, the motion control system which controls the manipulator can impose restrictions on the motion planner.

**Path planning constraints**
Manipulators may be operating in a very crowded workspace. Therefore, (moving) objects may be present in the workspace of a manipulator. Examples of such objects can be boxes in which products must be placed or other manipulators performing tasks. Moreover, common industrial manipulators consist of more than one link. The motion planner must account for this fact and ensure that no collisions between a manipulator and (moving) objects, present in the workspace, occur.

**Trajectory generation constraints**
Trajectory generation constraints are constraints which limit the velocity, acceleration and possibly jerk of the trajectory profiles. These constraints originate from different domains. Firstly, there are constraints on the actuator, i.e. maximum velocity, acceleration and possibly jerk[1]. By limiting jerk, an actuator experiences less wear, such that the life of an actuator is extended [Craig, 1986]. In most cases, manipulators are actuated by multiple, non-identical actuators. This implies that the actuator constraints may differ for each actuator.
Constraints, limiting the motion in another domain, are process constraints. Hereby, velocity, acceleration and jerk are limited in the workspace of a manipulator. For example, when a manipulator is performing a pick-and-place motion, the workpiece lying on a manipulator-hand should not fall off. The latter induces restrictions on the accelerations of the robot's end-effector. Other examples are laser cutting/welding operations, where a constant velocity during the operation is required.
The system which must perform the motion is, in practice, not infinitely stiff. The presence of such flexibilities limits the maximum allowable acceleration (applicable torque) of the system in order to avoid excessive vibrations of the manipulator.

**Motion control constraints**
As discussed in the beginning of this section, besides (physical) limitations on the manipulator and process limitations which limit the trajectory profiles, also the motion control system imposes restrictions on the motion planner. Namely, a controlled system has a limited bandwidth. When the determined profiles contain dominant frequencies which are within the bandwidth of the closed-loop system, the controlled system is able to follow the applied trajectories. However, when the trajectories contain frequencies which lie outside the bandwidth of the closed-loop system, these frequencies cannot be tracked accurately. This implies that trajectories, determined by the motion planner, must contain dominant frequencies which lie within the bandwidth of controlled system.

### 3.2.2 Profile order

An important issue in motion planning is the order of a trajectory profile. In this report, the following definition of the order of a profile is used.

**Definition 3.1 (Profile order)** *A profile is said to be of order n when the n-th derivative with respect to the depending parameter (e.g. time) is piecewise continuous.*

[1] Actuator constraints can also be specified as maximum torque and possibly maximum torque rates.

The profiles (either in the workspace of a manipulator or in joint space) must at least be second-order continuous, such that no discontinuities on acceleration-level occur. Discontinuities in the accelerations result in discontinuous torques. Such instant changes in desired torque levels cannot be generated by an actuator due to the electrical dynamics of an actuator. This implies that from the beginning a difference between the desired trajectory and the actual joint position will exist.
The profile may be of a higher order, but higher-order profiles exhibit some disadvantages [Macfarlane, 2003]. First, higher-order profiles require more computational power, which is important in the case of real-time motion planning. Furthermore, higher-order profiles have the tendency to oscillate; this may lead to trajectories which are not natural for a manipulator [Sciavicco, 1996]. On the other hand, higher-order profiles can generate smooth jerk profiles, thereby reducing manipulator wear.

### 3.2.3 Path following accuracy

For certain applications, time optimal solutions are less important than the accuracy of a generated motion. This is for instance the case for manipulators which have to carry out cutting, welding or gluing operations. For theses types of application is it important that a certain path is followed exactly. So a high accuracy is desired on following the path.
For pick-and-place motions, it is less important to exactly follow a path. However, here the manipulator must reach the specified end-point with an accuracy as high as possible, and moreover as fast as possible (without violating any of the constraints), while the obtained path accuracy in between pick and place location is of less importance.

### 3.2.4 Input and output specification of the motion planner

Since the goal of this work is to develop a generic motion planner, it must be possible to specify generic inputs to the motion planner. For a user, it is of great importance that inputs can be given in a coordinate system in which motion can be represented very easily. It must be possible to apply several forms of inputs to the motion planner, such as,

- points defined in Cartesian coordinates,
- points defined in Cylindrical coordinates,
- predefined shapes, such as a circle with radius $r$.

Since the motion planner provides input signals to the motion control system and the fact that most industrial motion controllers are digitalized nowadays, the output of the motion planner must provide at every sample instant, the desired position of a manipulator to the motion control system.
Since we are dealing with industrial manipulators which are produced in large series, not one manipulator has exactly the same configuration. This implies that the forward kinematics of the manipulator, used for the determination of the trajectories, need to be calibrated. When kinematic calibration is performed it means that the motion planner is able to deal with non-straightness of manipulator axes, influence of ball-bearings etc. Moreover, kinematic calibration improves tracking performance of the system. While kinematic calibration is of great practical importance, this work does not discuss kinematic calibration methods.

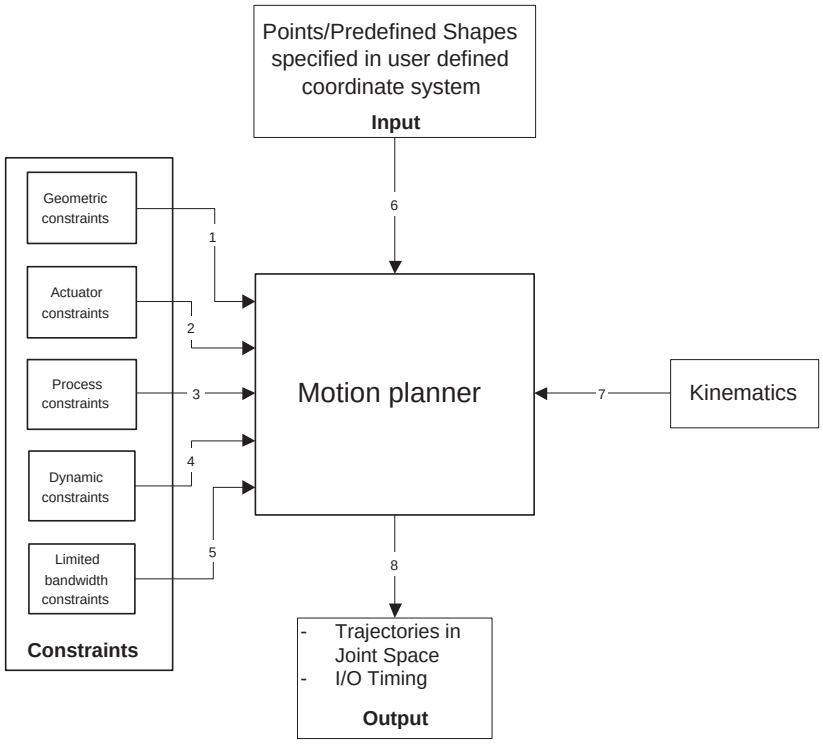



Figure 3.1: Basic overview of inputs and outputs of the generic motion planner.

### 3.2.5 I/O handling

In this report, with the term I/O, we indicate (digital)[2] inputs and outputs of a motion control system. (Digital) inputs are typically used for safety reasons, for example to check whether a valve is in open or closed position or, in the case of linear motors, to check whether the end of a stroke is reached. Other applications of (digital) inputs are for example used to provide an update on the position and orientation of an object (e.g. a product to be displaced, scanned, etc.) that influences the desired end-point of a trajectory. (Digital) outputs are typically used for functionality issues. These actions may consist of switching a laser on or off for welding or cutting operations. The motion planner must be able to interact with I/O, such that positions and timings are supplied to the I/O.

## 3.3 Structure of the generic motion planner

In this section, a structure for a generic motion planner is proposed. The structure is based on the set of requirements presented in the previous section. A basic overview of the inputs and outputs of a generic motion planner can be found in Figure 3.1.
There are different categories of inputs to the motion planner. The first type of inputs are plan-

[2]With digital we indicate that inputs and outputs can switch between two states, the high state and the low state.

ning inputs (see label 6 in Figure 3.1). As stated in the requirements, it must be possible to insert different type of planning inputs (e.g. points or predefined shapes, that can be specified in a user-defined coordinate system) to the motion planner. The second type of inputs are so-called safety inputs (labels 1 to 5 in Figure 3.1) . These inputs specify the constraints on the motion to be calculated. The constraints on the motion can be separated in several parts, namely geometric constraints, actuator constraints, process constraints, dynamic constraints and constraints due to a limited bandwidth of the motion controller. Geometric constraints specify forbidden spaces in the workspace of a manipulator, such that no collisions between a manipulator and objects in the workspace occur. Actuator constraints specify velocity, acceleration and jerk limits on the trajectories such that no actuator saturation occurs and the life of an actuator is extended. Process constraints limit the velocity, acceleration and jerk in the workspace of a manipulator. Dynamic constraints limit the torque produced by the actuators such that no unwanted vibrations occur. Finally it should be mentioned that the controlled system has a limited bandwidth. This implies that the generated trajectory must contain dominant frequencies below the bandwidth of the controlled system. The last type of inputs are so-called required inputs. These inputs must be supplied in order to make the algorithm work. The kinematics of a manipulator (label 7 in Figure 3.1) must be supplied to the trajectory generator in order to be able to calculate trajectories.
The output (label 8 in Figure 3.1) of the generic motion planner are trajectories in joint space (in the figures denoted as JS). The trajectories provide at each sample instant the desired position to the motion control system. Furthermore, certain I/O timings, such as for switching on a laser, may be given as an output.

### 3.3.1 Motion planner

In Figure 3.2, an overview of the structure of the generic motion planner is given. The numbered blocks correspond to the numbered inputs and outputs given in Figure 3.1.

**Coordinate transformation:** As stated by the requirements, the inputs can be specified in a user defined coordinate system. However, the motions will be calculated in one coordinate system, namely in Cartesian space (in the figures denoted as CS). The Cartesian space is chosen because the Cartesian coordinates can be straightforwardly interpreted. This implies that a coordinate transformation from the user-defined coordinate system to Cartesian space is required.

**Kinematical transformation:**
As can be seen from Figure 3.1 are the constraints defined in different domains. Since the motion planning strategy is performed in one domain, namely the Cartesian space, the constraints need to be transformed to this domain. This is done by a kinematical transformation.

**Trajectory generation algorithms:** As discussed in the previous chapter, the focus in this research lies on developing one (or more) trajectory generation algorithms. Therefore, after the coordinate transformation of the inputs applied to the motion planner, trajectories for the applications are determined. The trajectories are generated while respecting the constraints acting on the system, as discussed in the requirements section. Furthermore, the trajectories represent a path in the workspace of a manipulator. In the following chapters, the trajectory generation algorithms will be described in detail.

**Collision avoidance:** After planning a path in the workspace of a manipulator, an evaluation of

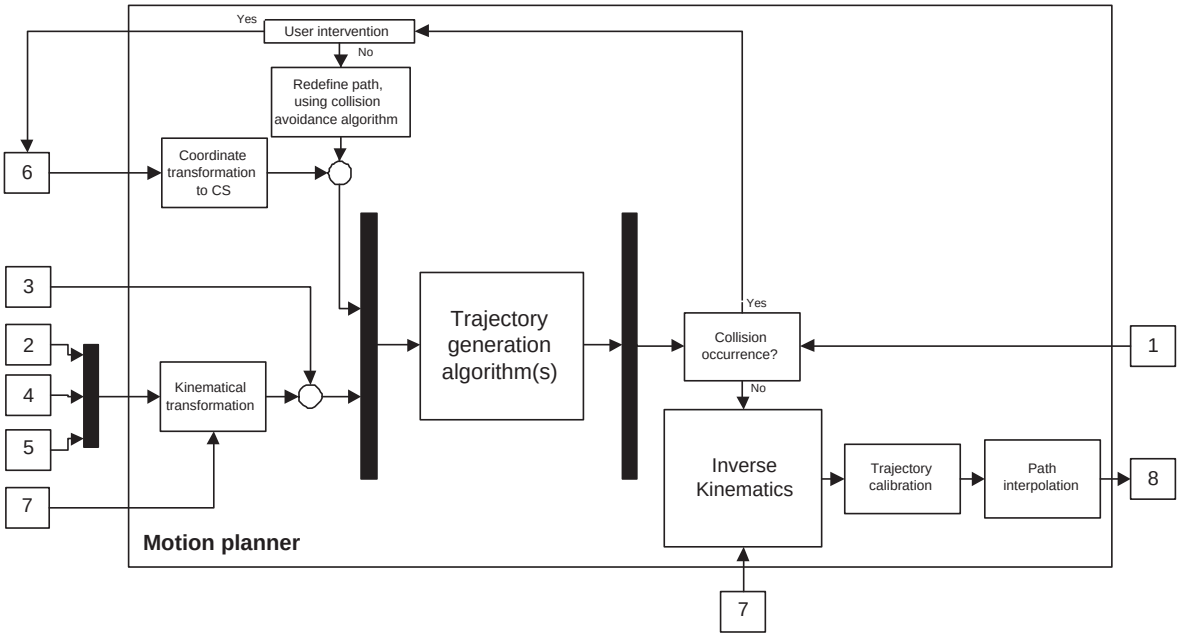



Figure 3.2: Structure of the generic motion planner.

the path is performed in order to check whether collisions between the manipulator and (moving) objects in the workspace of the manipulator occur. When this is the case, the path needs to be replanned. This can be done with or without user intervention. When user intervention is desired, the user has to specify new/extra inputs to the motion planner. If no user intervention is desired, an algorithm must be designed which determines new/extra inputs. Collision occurrence is evaluated after a motion trajectory is generated, since planning collision-free motion is labeled beyond the scope of this thesis and furthermore is seen as high-level motion planning (see Chapter 1).

**Inverse kinematics:** Until now, the total motion planning procedure is performed in Cartesian space. In order to execute the planned path we have to calculate the trajectories that should be sent to the actuators. Hereto, we need the inverse kinematic transformation. Because the trajectory generator must be applicable for several types of manipulators, it could be possible that the kinematics of a manipulator are described by complex nonlinear equations. Therefore, an algorithm is derived that can deal with several types of manipulators. The structure of the inverse kinematics algorithm will be presented in Section 3.3.2.

**Trajectory calibration:** The generated set of trajectories does not account for non-straight manipulator axes, disturbances induced by ball-bearings, etcetera. Therefore, a calibration of the calculated trajectories, which account for these disturbances and non-straightness of the axes, should be performed. The kinematic calibration can be provided in either an look-up table or a calibration function. As already mentioned above, trajectory calibration is beyond the scope of this work.

**Path interpolation:** In order to be able to process the path with a motion controller, the path

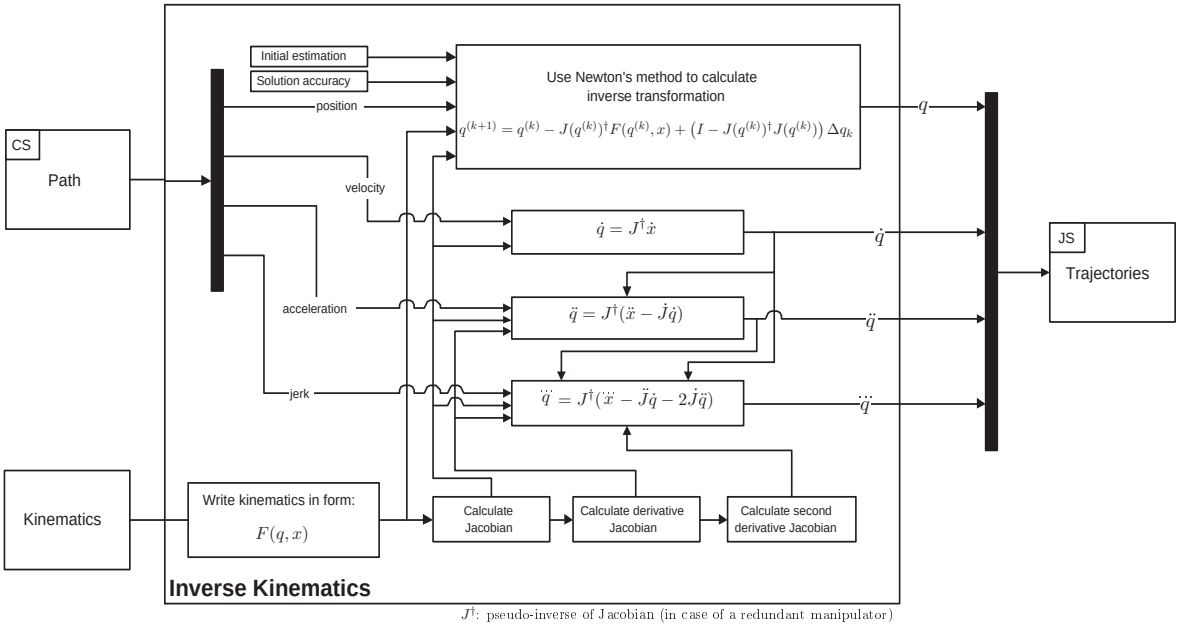



Figure 3.3: Structure of the inverse kinematics algorithm based on Newton's Method.

is interpolated such that at each sample the desired position of each actuator of a manipulator, is fed to the motion controller. Next to a feedback loop, in industrial motion control systems, also feedforward control for single axis motion control is present. This implies that next to desired positions, also desired velocities, accelerations and possibly jerks must be provided to the motion control system at every sample instant.

### 3.3.2 Inverse kinematics algorithm

In this section, we will present an inverse kinematics algorithm. The forward (or direct) kinematic equations relate joint variables to positions and orientations of the end-effector in the workspace space of a manipulator. The inverse kinematics problem is the opposite of the forward kinematic problem, namely given the positions and orientations of the end-effector in the workspace of a manipulator, determine the corresponding joint variables. For forward kinematics, the solution of the equations can be defined uniquely. This is not always the case for the inverse kinematics problem, since the equations can be highly nonlinear. Furthermore, multiple solutions may exist. Solutions can be determined in closed-form or by using numerical techniques. Computation of closed-form solutions requires either an algebraic-intuition to find the significant equations or geometric intuition to find the significant points on the structure to express positions and orientations as function of joint variables [Sciavicco, 1996]. So, determining a closed-form solution requires a lot of specific knowledge about the system. Therefore, we choose to determine the inverse kinematic solution numerically. In this way, a user only has to specify the forward kinematic equations, which can be determined very easily using for example the Denavit-Hartenberg convention [Denavit, 1955]. An overview of numerical algorithms available for determining solutions to the inverse kinematics problem can be found in [Stembera, 2005].

In Figure 3.3, an overview of the structure of the inverse kinematics algorithm can be found. As can be seen from this figure, the solutions of the positional joint variables are based on Newton's method. Newton's method is an iterative searching method based on the Taylor expansion of the forward kinematics written in the following form;

$$\underline{F}(\underline{q}, \underline{x}) = \underline{0}. \tag{3.1}$$

Where $\underline{F}(\underline{q}, \underline{x})$ is a function depending on joint variables $\underline{q}$ and end-effector variables $\underline{x}$ (which are given in the inverse kinematics problem). The Taylor expansion of (3.1) around $\underline{q} = \underline{q}^{(k)}$ results in,

$$\underline{F}(\underline{q}, \underline{x}) = \underline{F}(\underline{q}^{(k)}, \underline{x}) + \underline{J}(\underline{q}^{(k)}, \underline{x})\Delta\underline{q}^{(k)} + H.O.T., \tag{3.2}$$

where

$$\underline{J}(\underline{q}^{(k)}, \underline{x}) = \left.\frac{\partial \underline{F}(\underline{q}, \underline{x})}{\partial \underline{q}}\right|_{\underline{q}=\underline{q}^{(k)}} \tag{3.3}$$

is defined as the Jacobian of the function $\underline{F}(\underline{q}, \underline{x})$ with respect to $\underline{q}$ in $\underline{q} = \underline{q}_k$ and $(\cdot)^{(k)}$ denotes the result of the $k-$th iteration. Using (3.1) and (3.2) and ignoring higher-order terms, the update rule for the joint variables is defined as

$$\underline{q}^{(k+1)} = \underline{q}^{(k)} + \Delta\underline{q}^{(k)}, \tag{3.4}$$

with

$$\Delta\underline{q}^{(k)} = -\underline{J}^{-1}(\underline{q}^{(k)}, \underline{x})\underline{F}(\underline{q}^{(k)}, \underline{x}). \tag{3.5}$$

The method is said to have converged when the update (3.5) becomes less than a user-defined tolerance criterium $\epsilon$, i.e. $|\Delta\underline{q}^{(k)}| \leq \epsilon$, which has to be supplied to the algorithm together with an initial estimation $\underline{q}^{(0)}$ of $\underline{q}^{(0)}$ (see Figure 3.3). From (3.5), it can be seen that when the Jacobian in (3.3) becomes singular, no solution can be determined. Furthermore, when a manipulator is redundant the Jacobian is not square and the inverse of the Jacobian does not exist. However, as becomes clear in the problem statement in Chapter 1, it should also be possible to generate motions for redundant manipulators. Therefore, a pseudo-inverse of the Jacobian is determined using singular value decomposition (see Appendix B). The pseudo-inverse solution is not an exact inverse solution of the Jacobian. However, no exact solution of the inverse Jacobian is needed to solve the problem. Since only a *search-direction* is necessary to keep Newton's method converging to the inverse kinematic solution. A benefit of using a pseudo-inverse is that a solution can be found even when a manipulator is in a singular position. A numerical example of the inverse kinematics algorithm, which shows that no deadlocks occur when joint variables are determined, can be found in Appendix C.

After determining the joint positions using Newton's method, the joint velocities, accelerations and jerks can be determined. Instead of numerically differentiating the joint positions, the derivatives are determined using the Jacobian and it's time derivatives (see Figure 3.3). The derivatives of the Jacobian are determined approximated numerically. In this way, no numerical noise is introduced in the trajectory profiles.

## 3.4 Discussion

In this chapter, the results of a requirements study on generic motion planning are presented. The results can be seen as requirements regarding constraints, order of the profile, path following accuracy, input and output of a motion planner and I/O handling. From the set of requirements a structure of a generic motion planner is proposed. An advantage of the proposed structure is the practical applicability. A user has to specify the desired path (by providing a path-function or a set of points), the constraints acting on the application and the forward kinematics of the manipulator in use.
In order to determine the trajectories for each actuator, an inverse kinematics algorithm is developed. The algorithm can deal both with redundant and non-redundant manipulators and uses Newton's method. Moreover, it is shown by means of an example (in Appendix C) that no deadlocks occur in singular positions of a manipulator.

In the following chapters, trajectory generation algorithms, see Figure 3.2 will be presented. First, an algorithm for determining pick-and-place motions will be discussed in Chapter 4. Next, a method for determining path-constrained motions will be presented in Chapter 5.

# Chapter 4

# Trajectory generation algorithm for Pick-and-Place motions

## 4.1 Introduction

This chapter discusses the development of a trajectory generation algorithm for the calculation of (time-optimal) pick-and-place motions. As discussed in Chapter 2, several methods are available for determining pick-and-place motions. Linear segments with blend trajectories are commonly used by industrial manipulators and machine tool trajectories for their fast motions and low computational complexity [Macfarlane, 2003]. Also splines are often used for motion planning. However, the path of a spline is not predictable in advance. Furthermore, the motion will in general be only near time-optimal. Therefore, we choose to determine pick-and-place operations by using linear segment with blend theory as discussed by Macfarlane and Croft [Macfarlane, 2003] and Lloyd and Hayward [Lloyd, 1993]. Linear segments with blends are defined as straight lines between a set of points, defined either in the workspace of a manipulator or in joint space, provided by an user. Around each point the user may specify a sphere with a certain radius. Inside this sphere, it is allowed to deviate from the straight lines. So, the user is seen as a high-level path planner which is able to specify path such that no collision occurs between a manipulator and (moving) objects in the workspace of a manipulator (i.e. the satisfaction of geometric constraints is guaranteed a priori).

This chapter is organized as follows. First, we will discuss the tools that are necessary to describe linear segment with blend trajectories. Next, the structure of the algorithm will be discussed.

## 4.2 Linear segments with blends

This section elaborates on the tools needed to describe linear segments with blends. The linear segments, defined between a set of user-defined waypoints, are described by using time-optimal second-order continuous point-to-point profiles. This type of profile is chosen because it is the least order that fulfills the profile order requirement and furthermore the requirement that the motion must be as fast as possible (see Section 3.2). Higher-order profiles mainly result in smoother but slower motions. Since during pick-and-place motions the accuracy is of less importance than the total motion time, the minimum allowed profile order is used.
In principle, the path is defined by linear segments. The requirements on the generic trajec-

tory generator state that the trajectories have to be at least $C^2$-continuous (see Section 3.2); this implies that a manipulator has to stop at every defined waypoint. In this way, the motion does not become as fast as possible. However, the motion can be regarded as time-optimal, since constraints are active during a stop-motion[1]. This can be tackled by defining a sphere, with a certain radius around each waypoint. Within the sphere, it is allowed to deviate from a linear segment (i.e. geometric constraints remain to be satisfied). Within this sphere, we can design a path, such that it is possible to travel from one linear segment to another without stopping. First, we will discuss the calculation of time-optimal second-order continuous linear segments. After that, a description of the blending between segments will be discussed.

### 4.2.1 Point-to-point description

As stated in the previous section, the path consists of linear segments defined between a set of user-defined waypoints. These segments are calculated by using second-order continuous point-to-point theory. In order to obtain a motion that is as fast as possible, the trajectories need to be time-optimal. Therefore, we need to calculate point-to-point trajectories for which at every time instant at least one constraint is active. Furthermore, for blend calculation purposes, it must be possible to define certain velocities at begin and end positions of a point-to-point. Begin and end accelerations of a point-to-point are set equal to zero, in order to guarantee a smooth transition from a linear segment to a blend segment.

We can define a linear segment $\underline{p}(t)$ connecting point $\underline{p}_i$ to point $\underline{p}_{i+1}$, where $\underline{p}_i,\ \underline{p}_{i+1} \in \mathbb{R}^3$, by the following parametric representation [Sciavicco, 1996]:

$$\underline{p}(t) = \underline{p}_i + \frac{\underline{p}_{i+1} - \underline{p}_i}{\|\underline{p}_{i+1} - \underline{p}_i\|} r(t), \tag{4.1}$$

where $\underline{p}(t)$ is defined as an $(3 \times 1)$ vector containing Cartesian positions, $r(t)$ is a scalar function which describes the way in which the trajectory is traveled in time and $\|\cdot\|$ represents the $l^2$-norm of vector $(\cdot)$. The begin and end conditions of $r(t)$ are defined as

$$\begin{aligned} r(t=0) &= 0, \\ r(t=T) &= \|\underline{p}_{i+1} - \underline{p}_i\|. \end{aligned} \tag{4.2}$$

The same parametric representation can be written for the angular coordinates $\underline{\phi}$:

$$\underline{\phi}(t) = \underline{\phi}_i + \frac{\underline{\phi}_{i+1} - \underline{\phi}_i}{\|\underline{\phi}_{i+1} - \underline{\phi}_i\|} r(t), \tag{4.3}$$

where $\underline{\phi}(t)$ is defined as a $(3 \times 1)$ vector containing angular coordinates. The orientations are described using RPY-angles (Roll-Pitch-Yaw angles). Roll-Pitch-Yaw angles are chosen because they are easy interpretable. RPY-angles are often used in the automotive and/or (aero)nautical field. The rotation resulting from the RPY angles can be obtained by a composition of rotations with respect to a fixed reference frame [Sciavicco, 1996; Craig, 1986]. The derivation of the direction cosine matrix for RPY-angles can be found in Appendix D.

[1] Recall here the definition of a time-optimal motion, that can be found in Chapter 1.

In order to obtain time-optimal motions, we need to determine the trajectory $r(t)$ such that one constraint is active over the entire time span. The trajectory $r(t)$ is determined using point-to-point theory as described in [Lambrechts, 2003]. However, in [Lambrechts, 2003] only point-to-points with begin and end velocity *equal* to zero are discussed. In our case, we need to extend the approach such that point-to-points with begin and end velocity *unequal* to zero, can be determined.

When regarding the acceleration profile of a point-to-point with equal begin and end velocities, we can distinct seven different timings, represented by time parameters $\tau_1$ to $\tau_7$. Time parameters $\tau_1$, $\tau_3$, $\tau_5$ and $\tau_7$ represent the time for which the jerk is constant, $\tau_2$ and $\tau_6$ the time for which the acceleration is constant and finally $\tau_4$ represents the time for which the velocity is constant. Since it must be possible to calculate time-optimal point-to-points, with specified begin and end velocities, several different sets of time parameters can be found. In total seventeen different sets of the time parameters can be found. An overview of the acceleration profiles for each set of the time parameters can be found in Appendix E. Below, the approach for determining the time-optimal point-to-points, where begin and end velocities can be unequal to zero, is discussed. A detailed derivation for a certain case where the profile is not symmetric can be found in Appendix F.

A distance $x_{max} = \|\underline{p}_{i+1} - \underline{p}_i\|$ is traveled, while respecting velocity, acceleration and jerk constraints indicated by $v_{max}$, $a_{max}$ and $j_{max}$, respectively. The point-to-point time parameters are determined using the requirement that the motion should be time-optimal. Thus, at all times one constraint needs to be active over the time span. Depending on the values of the set of constraints acting on the point-to-point, this results in one of the cases discussed in Appendix E. In order to determine the time parameters, we first assume that a constant velocity part exists. Furthermore, we assume that the total distance $x_{max}$ is large enough to overcome the defined velocity difference ($\Delta v = |v_e - v_0|$), where $v_0$ denotes the velocity defined at the begin of the point-to-point and $v_e$ denotes the velocity at the end of the point-to-point. This implies that

$$\tau_4 > 0. \tag{4.4}$$

The condition for the presence of a constant acceleration part, i.e. $\tau_2 > 0$, is given by

$$\frac{v_{max} - v_0}{a_{max}} - \frac{a_{max}}{j_{max}} \geq 0. \tag{4.5}$$

If condition (4.5) is satisfied, the time parameters $\tau_1$, $\tau_2$ and $\tau_3$ are given by,

$$\begin{aligned} \tau_1 = \tau_3 &= \frac{a_{max}}{j_{max}}, \\ \tau_2 &= \frac{v_{max} - v_0}{a_{max}} - \frac{a_{max}}{j_{max}}. \end{aligned} \tag{4.6}$$

When the condition given by (4.5) does not hold, the time parameters for the acceleration part are given by,

$$\begin{aligned} \tau_1 = \tau_3 &= \sqrt{\frac{v_{max} - v_0}{j_{max}}}, \\ \tau_2 &= 0. \end{aligned} \tag{4.7}$$

The same analysis can be performed for the deceleration part. The condition for a constant deceleration part, i.e. $\tau_6 > 0$, is given by

$$\frac{v_{max} - v_e}{a_{max}} - \frac{a_{max}}{j_{max}} \geq 0. \tag{4.8}$$

If condition (4.8) is satisfied, the deceleration time parameters ($\tau_5$, $\tau_6$ and $\tau_7$) can be determined by

$$\begin{aligned} \tau_5 = \tau_7 &= \frac{a_{max}}{j_{max}}, \\ \tau_6 &= \frac{v_{max} - v_e}{a_{max}} - \frac{a_{max}}{j_{max}}. \end{aligned} \tag{4.9}$$

If condition (4.8) does not hold, the deceleration time parameters are given by,

$$\begin{aligned} \tau_5 = \tau_7 &= \sqrt{\frac{v_{max} - v_e}{j_{max}}}, \\ \tau_6 &= 0. \end{aligned} \tag{4.10}$$

Finally, only the time parameter for which the velocity constraint is active, $\tau_4$ needs to be determined. This can be done by using the total amount of distance that needs to be traveled. Clearly, the total area under the velocity profile must be equal to the distance to travel, i.e.

$$x_{max} = \frac{1}{2}(v_{max} + v_0)(2\tau_1 + \tau_2) + v_{max}\tau_4 + \frac{1}{2}(v_{max} + v_e)(2\tau_5 + \tau_6). \tag{4.11}$$

Substituting the time parameters defined above provides the solution for $\tau_4$. For certain given parameters, it could be possible that $\tau_4 < 0$, so the maximum velocity $v_{max}$ cannot be reached with the given parameters. This implies that $\tau_4$ must be equal to zero. Next, we assume that the maximum acceleration and deceleration can be reached. This implies that $\tau_2$ and $\tau_6$ are larger than zero. We can now determine the time parameters as,

$$\begin{aligned} \tau_1 = \tau_3 &= \frac{a_{max}}{j_{max}}, \\ \tau_2 &= \frac{v(t_3) - v_0}{a_{max}} - \frac{a_{max}}{j_{max}}, \\ \tau_5 = \tau_7 &= \frac{a_{max}}{j_{max}}, \\ \tau_2 &= \frac{v(t_3) - v_e}{a_{max}} - \frac{a_{max}}{j_{max}}. \end{aligned} \tag{4.12}$$

with $t_3 = \tau_1 + \tau_2 + \tau_3$. The velocity $v(t_3)$ at $t = t_3$ can be determined by solving the following quadratic equation,

$$\frac{1}{2}(v(t_3) + v_0)(2\tau_1 + \tau_2) + \frac{1}{2}(v(t_3) + v_e)(2\tau_5 + \tau_6) = x_{max}. \tag{4.13}$$

What remains are the three cases for which $\tau_2$ or $\tau_6$ are less or equal to zero, i.e. $\tau_2 > 0,\ \tau_6 = 0$, $\tau_2 = 0,\ \tau_6 > 0$ and $\tau_2 = \tau_6 = 0$. For the first case, $\tau_2 > 0,\ \tau_6 = 0$, this results in solving the

following set of equations:

$$\begin{aligned} \tau_1 = \tau_3 &= \frac{a_{max}}{j_{max}}, \\ \tau_6 &= 0, \\ j_{max}\tau_5^2 &= (v_e - v_0) - \frac{a_{max}^2}{j_{max}} + a_{max}\tau_2, \\ \frac{1}{2}(v_e + v_0 - j_{max}\tau_5^2)(2\tau_1 + \tau_2) &+ (2v_e - j_{max}\tau_5^2)\tau_5 = x_{max}. \end{aligned} \tag{4.14}$$

For the second case, i.e. $\tau_2 = 0,\ \tau_6 > 0$, a similar set of equations can be derived:

$$\begin{aligned} \tau_5 = \tau_7 &= \frac{a_{max}}{j_{max}}, \\ \tau_2 &= 0, \\ j_{max}\tau_1^2 &= (v_e - v_0) - \frac{a_{max}^2}{j_{max}} + a_{max}\tau_6, \\ \frac{1}{2}(v_e + v_0 + j_{max}\tau_1^2)(2\tau_5 + \tau_6) &+ (2v_0 + j_{max}\tau_1^2)\tau_1 = x_{max}. \end{aligned} \tag{4.15}$$

Finally, the case for which both $\tau_2$ and $\tau_6$ are zero results in solving the following set of equations,

$$\begin{aligned} \tau_2 = \tau_6 &= 0, \\ v_0 + j_{max}\tau_1^2 &= v_e - j_{max}\tau_5^2, \\ (v_e + v_0 - j_{max}\tau_5^2)\tau_1 &+ (2v_e - j_{max}\tau_5^2)\tau_5 = x_{max}. \end{aligned} \tag{4.16}$$

One of the statements for determining the time optimal point-to-points was that the distance was large enough to overcome the total change in velocity. For the linear segments with blends algorithm this assumption could be violated. Such violation implies that in such case the begin or end velocity needs to be redefined (see Section 4.3 for more detail on this matter). The resulting acceleration profile will consist of an acceleration or deceleration profile depending on the values of the begin or end velocity.

In Figure 4.1, an example of a time-optimal point-to-point is given which is determined with a total traveling distance, $x_{max}$, of 1 $m$, and begin and end velocity defined as $v_0 = 0.25\ \frac{m}{s}$ and $v_e = 1.5\ \frac{m}{s}$, respectively. Furthermore, a velocity constraint of $v_{max} = 2\ \frac{m}{s}$, an acceleration constraint of $a_{max} = 10\ \frac{m}{s^2}$ and a jerk constraint of $j_{max} = 100\ \frac{m}{s^3}$, is used. We can see that the condition stated by (4.5) is satisfied, thus $\tau_2 > 0$, while condition (4.8) does not hold, thus $\tau_6 = 0$. Furthermore, $\tau_4$ is larger than zero as can be seen using (4.11). From the figure, it can be seen that during the entire time span of the point-to-point one of the constraints is active, resulting in a time-optimal motion.

### 4.2.2 Blend function description

In order to travel between two linear segments, a blend function $\underline{x}(t)$ is needed which describes the movement between two segments. As stated earlier, a blend sphere can be specified in which the path may deviate from the linear segments. In this section, we will discuss methods to blend between two linear segments.

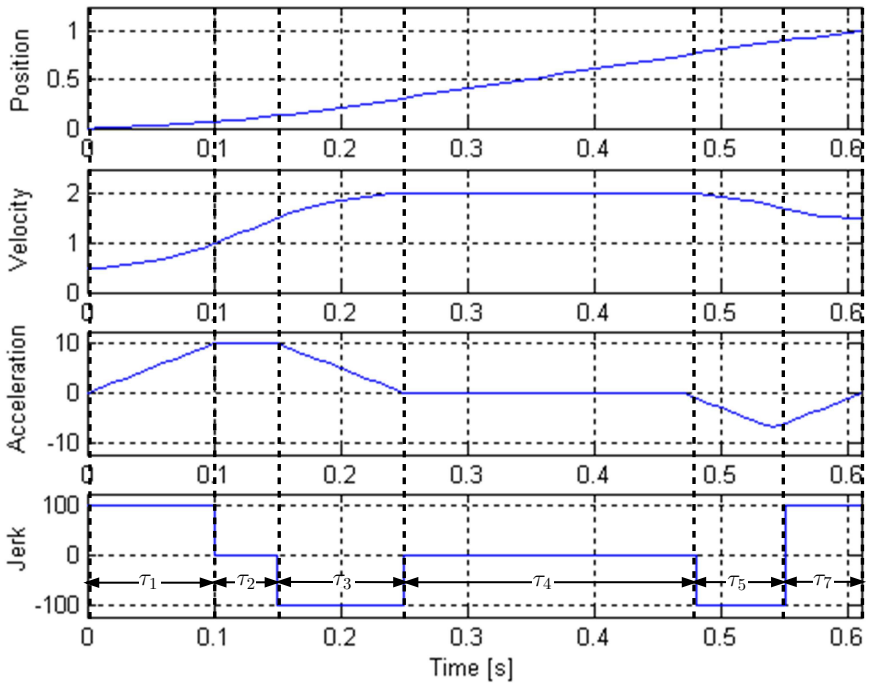



Figure 4.1: Third order point-to-point with different begin and end velocity.

Suppose we want to blend between two linear segments defined by waypoints $\underline{p}_1$, $\underline{p}_2$ and $\underline{p}_3$, with a blend sphere defined around waypoint $\underline{p}_2$ (see Figure 4.2). The blend radii defined at each point are given as $\varepsilon_1 = \varepsilon_3 = 0$ and $\varepsilon_2 > 0$. The blend start and end points, $\underline{p}_a$ and $\underline{p}_b$, are now defined by

$$\begin{aligned} \underline{p}_a &= \underline{p}_2 - \varepsilon_2 \frac{\underline{p}_2 - \underline{p}_1}{\|\underline{p}_2 - \underline{p}_1\|} \\ \underline{p}_b &= \underline{p}_2 + \varepsilon_2 \frac{\underline{p}_3 - \underline{p}_2}{\|\underline{p}_3 - \underline{p}_2\|}. \end{aligned} \tag{4.17}$$

At these points, the following boundary conditions must hold in order to guarantee $C^2$-continuity of the geometric path.

$$\begin{aligned} \underline{x}(t = t_1) &= \underline{p}_a, & \underline{x}(t = t_2) &= \underline{p}_b, \\ \tfrac{d\underline{x}}{dt}(t = t_1) &= \underline{v}_a, & \tfrac{d\underline{x}}{dt}(t = t_2) &= \underline{v}_b, \\ \tfrac{d^2\underline{x}}{dt^2}(t = t_1) &= \underline{a}_a, & \tfrac{d^2\underline{x}}{dt^2}(t = t_2) &= \underline{a}_b. \end{aligned} \tag{4.18}$$

These six conditions can be satisfied by connecting the blend start and end points by a fifth-order polynomial. The main disadvantage of this type of blends is the fact that there is no control of the transition shape, since the coefficients of the polynomial are determined by the boundary conditions and the travel time is determined by the acceleration and velocity constraints. Moreover, in

order to calculate the transition we need the velocity and acceleration boundary conditions at $\underline{p}_b$. This may not be possible if the path is tracking external sensor signals or control inputs [Lloyd, 1993].

In order to overcome the above mentioned problems, Lloyd and Hayward [Lloyd, 1993] have developed a blend procedure which uses a convex average of two geometrically defined paths. The procedure is applied by Macfarlane in order to obtain linear segments with blends trajectories [Macfarlane, 2003]. Below we will elaborate on the procedure as developed by Lloyd and Hayward.

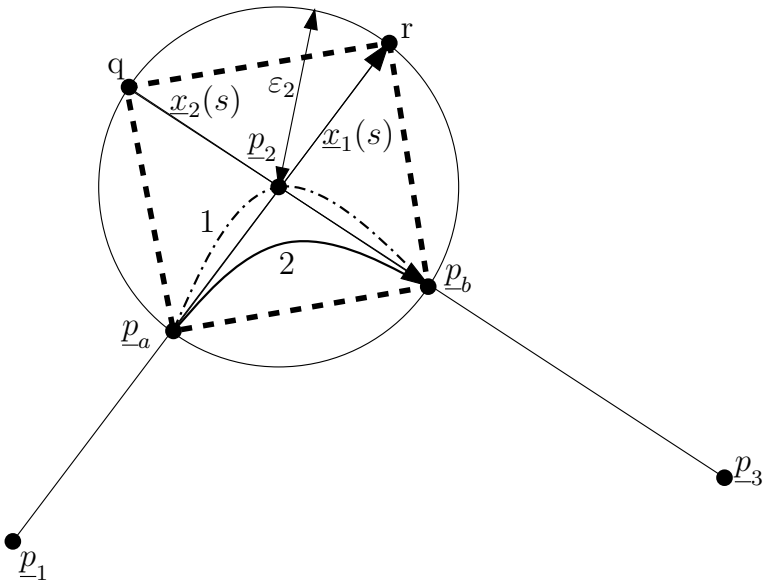



Figure 4.2: Overview of the blend procedure developed by Lloyd and Hayward [Lloyd, 1993].

The blend function $\underline{x}(t)$ is parameterized using a nondimensional time coordinate $s$ defined by

$$s = \frac{t - t_1}{2\tau}, \tag{4.19}$$

where $t$ represents time, $t_1$ the time at the start of the blend and $\tau$ the half blend time. Obviously, the blend occurs in the interval $s \in [0, 1]$.
As described by [Lloyd, 1993], during the transition interval, the two linear segments are represented by two linear geometric paths $\underline{x}_1(s)$ and $\underline{x}_2(s)$ (as can be seen in Figure 4.2),

$$\underline{x}_1(s) = \underline{b}_1 + \underline{m}_1 s, \tag{4.20}$$

$$\underline{x}_2(s) = \underline{b}_2 + \underline{m}_2 s, \tag{4.21}$$

where $\underline{b}_i$ and $\underline{m}_i$ are the position and direction of $\underline{x}_i(s)$ at $s = 0$, respectively, with $i = 1, 2$. In order to obtain smooth blend trajectories, the boundary conditions defined by (4.18) need to be satisfied. Furthermore, in order to obtain a smooth transition between linear segment and blend,

the acceleration at the begin and end of the blend are set to zero (namely this also holds at the start and end of the linear segments). The blend function is now defined as follows:

$$\underline{x}(s) = \underline{x}_1(s) + \alpha(s)(\underline{x}_2(s) - \underline{x}_1(s)), \tag{4.22}$$

where,

$$\alpha(s) = 6s^5 - 15s^4 + 10s^3. \tag{4.23}$$

Since the blend function is a result of a convex combination of $\underline{x}_1(s)$ and $\underline{x}_2(s)$, the transition shape lies in the dashed rectangle defined in Figure 4.2. A possible shape is indicated by the dash-dotted line with index 1. This possible shape indicates that the blend may tend to accelerate (and then decelerate) more than necessary during the transition. Consequently, we want the profile to be between the isosceles triangle determined by $\underline{p}_a - \underline{p}_2$, $\underline{p}_2 - \underline{p}_b$ and $\underline{p}_a - \underline{p}_b$. Later on, we will discuss how to create different transition shapes. Lloyd and Hayward compensate for this unnecessary acceleration (and deceleration) by adding an extra term $\beta(s)\underline{u}$ to (4.22), where $\beta(s)$ is some polynomial and $\underline{u}$ a fixed vector:

$$\underline{x}(s) = \underline{x}_1(s) + \alpha(s)(\underline{x}_2(s) - \underline{x}_1(s)) + \beta(s)\underline{u}. \tag{4.24}$$

The polynomial $\beta(s)$ is of order six. A fifth-order polynomial is required to fulfill the boundary conditions, $\beta(0) = \dot{\beta}(0) = \ddot{\beta}(0) = \beta(1) = \dot{\beta}(1) = \ddot{\beta}(1) = 0$. The sixth order is introduced in order to influence the shape of the transition. The parameters of $\beta(s)$ are determined using the boundary conditions stated above. This results in an underdetermined set of linear equations (since 7 boundary conditions are needed and only six boundary conditions are given). If one set of coefficients $\underline{c}_0$ can be found, then all other sets $\underline{c}$ must satisfy (4.25) (see [Lloyd, 1993]).

$$\underline{c} = \underline{c}_0 + N(H)\underline{k}, \tag{4.25}$$

where $N(H)$ represents the null space of the matrix $H$ containing the parameters of the set of equations, $\underline{k}$ is an arbitrary vector with length equal to the dimension of the matrix's null space (in this case $\underline{k}$ has dimension one, i.e. $\underline{k} = k \in \mathbb{R}$), since the problem is one boundary condition short). The rows of matrix $H$ are determined by the coefficients of

$$\beta(s) = \sum_{i=0}^{6} c_i s^i, \tag{4.26}$$

and its first two derivatives at $s = 0$ and $s = 1$, such that

$$\begin{bmatrix} \beta(0) \\ \dot{\beta}(0) \\ \ddot{\beta}(0) \\ \beta(1) \\ \dot{\beta}(1) \\ \ddot{\beta}(1) \end{bmatrix} = H\underline{c}, \tag{4.27}$$

with,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 2 & 6 & 12 & 20 & 30 \end{bmatrix}. \tag{4.28}$$

The boundary conditions are all equal to zero, implying that $\underline{c}_0 = \underline{0}$. This then results in,

$$\beta(s) = k(s^6 - 3s^5 + 3s^4 - s^3), \tag{4.29}$$

where $k$ is a free parameter which can be used to adjust the characteristics of the transition shape [Lloyd, 1993]. Lloyd and Hayward set $k = 1$ and adjust the transition shape by tuning the magnitude of $\underline{u}$. $\underline{u}$ is chosen such that the average acceleration is minimized,

$$min\left(\int_0^1 \|x''(s)\|^2 ds\right), \tag{4.30}$$

where $(\cdot)'$ denotes the derivative with respect to the parameter $s$, i.e. $\frac{d(\cdot)}{ds}$.
Such minimization results in the following choice for $\underline{u}$ (for a detailed derivation see [Lloyd, 1993]),

$$\underline{u} = -\frac{15}{2}\underline{v}_d, \tag{4.31}$$

with $\underline{v}_d = \underline{m}_2 - \underline{m}_1$ and consequently results in the following blend function:

$$\underline{x}(s) = \underline{x}_1(s) + \alpha(s)(\underline{x}_2(s) - \underline{x}_1(s)) - \kappa\beta(s)\underline{v}_d, \tag{4.32}$$

where $\kappa$ is the amount of acceleration compensation applied, with an optimal value of $15/2$. In Figure 4.2, the shape of the blend with acceleration compensation is indicated by index 2.
Since the blend function describes a geometric path, it is relatively easy to control the shape of the blend. In [Lloyd, 1993], an extension of (4.32) is given such that, by adjusting two parameters the blend shape can be controlled. This results in the following equation,

$$\underline{x}(s) = \underline{x}_1(s)+\alpha(s)(\underline{x}_1(s_h)-\underline{x}_1(s))+\alpha(s)(\underline{x}_2(s)-\underline{x}_2(s_s))-\kappa\beta(s)\underline{v}_d+(\underline{x}_1(0.5)-\underline{x}_1(s_h)), \tag{4.33}$$

with $\alpha(s)$ and $\beta(s)$ as defined by (4.23) and (4.29), respectively. Moreover, $s_s$ and $s_h$ are so-called start and halt parameters as described in detail by Lloyd and Hayward [Lloyd, 1993]. Although the procedure is described by Lloyd and Hayward, no details on respecting the velocity, acceleration and jerk constraints during the blend interval are discussed. Macfarlane [Macfarlane, 2003] discusses matters on respecting velocity, acceleration and jerk constraints during the blend interval, where the blend is described using the procedure described by Lloyd and Hayward. We will discuss such constrained blend design below. From now on when a blend is discussed, it is described using (4.33). The reason for choosing this blend description is that in this way the algorithm can be used for certain contouring tasks that can be described by linear segments (such as e.g. cutting rectangular objects from a workpiece).

The blend description requires a begin and end speed which are equal to each other, i.e. $\|\underline{v}_{a,i}\| = \|\underline{v}_{b,i}\| = spd_{blend,i}$, where $i$ indicates blend number in the sequence of linear segments with blends. This implies that the blend is symmetric around its midpoint. The maximum acceleration occurring during the blend is defined by [Macfarlane, 2003]

$$a_{blendmax,i} = \frac{5(spd_{blend,i})^2}{4\varepsilon_i}\sqrt{\frac{1+\cos\theta_i}{2}}, \tag{4.34}$$

with,

$$\cos\theta_i = -\frac{(\underline{p}_i - \underline{p}_{a,i})\cdot(\underline{p}_{b,i} - \underline{p}_i)}{\varepsilon_i^2}. \tag{4.35}$$

Therefore, we can conclude that, in order to limit the acceleration during the blend, the blend speed at the begin and end of the blend, $spd_{blend}$, needs to be upper-bounded. The maximum speed at the begin and end of the blend while respecting the acceleration constraint, $a_{max}$, is then defined as follows:

$$spd_{blendMaxAcc,i} = \sqrt{\frac{4a_{max}\varepsilon_i}{5\sqrt{\frac{1+\cos\theta_i}{2}}}}. \tag{4.36}$$

Similar limitations for the speed $spd_{blend,i}$ related to the maximum jerk can be formulated. The maximum jerk occurring during a blend occurs at the start and end of the blend and is given by [Macfarlane, 2003]

$$j_{blendmax,i} = \frac{15(spd_{blend,i})^3}{2\varepsilon_i^2}\cos\left(\frac{\theta_i}{2}\right). \tag{4.37}$$

So, the jerk during the blend can be limited by upper-bounding the blend speed at the begin and end of the blend. The blend speed for which the maximum jerk during the blend is equal to the maximum allowable jerk $j_{max}$ is defined as follows:

$$spd_{blendMaxJerk,i} = \left(\frac{2j_{max}\varepsilon_i^2}{15\cos\left(\frac{\theta_i}{2}\right)}\right)^{\frac{1}{3}}. \tag{4.38}$$

The maximum speed with which the blend can be started or ended, while respecting acceleration and jerk constraints is given by

$$spd_{blendMax,i} = min(spd_{blend,i}, spd_{blendMaxAcc,i}, spd_{blendMaxJerk,i}), \tag{4.39}$$

where $spd_{blend,i}$ is determined by the end-velocity of the linear segment at position $\underline{p}_a$. When $spd_{blendMax,i} \neq spd_{blend,i}$ the previous linear segment needs to be recalculated with the new blendspeed. This will be described in more detail in the next section, where the tools for determining linear segments with blends are discussed described in this section will be used to present the structure of the algorithm which determines linear segment with blend trajectories.

## 4.3 Structure of the algorithm

This section presents the structure of the algorithm for calculating (near time-optimal) linear segments with blends. The algorithm uses the tools as discussed in the previous section. First, a figure of the structure will be presented. After that, the components of the algorithm will be discussed.

In Figure 4.3 an overview of the algorithm structure is given. It can be seen that the algorithm consists of three main parts. The first part ('Check waypoints') checks whether the inputs supplied to the algorithm are valid. The second part ('Determine ptp') calculates time-optimal point-to-points, and the final part ('Determine blend') determines the blends.
The input to the algorithm consists of a set of points, defined in the workspace of the manipulator, through which the manipulator must travel. Furthermore, with each point (except with the

first and the last point) a blend sphere radius with a prescribed transition type (which defines the shape of the blend to be performed) can be specified. The first part of the algorithm checks whether two points lie on each other or whether the blend spheres of two successive points intersect. In the first case, where two points lie on each other, the point with the largest blend radius is deleted from the set of points. In the case of intersecting blend radii, i.e. the sum of the blend radii is larger than the distance between two points, i.e.

$$\|\underline{p}_{i+1} - \underline{p}_i\| < (\varepsilon_{i+1} + \varepsilon_i), \tag{4.40}$$

the radii of the two blend spheres are decreased. The size of the blend spheres are decreased while respecting the original radii ratio. The ratio between the blend radii is defined as,

$$\varepsilon^* = \frac{\varepsilon_i}{\varepsilon_{i+1} + \varepsilon_i}. \tag{4.41}$$

The new blend sphere radii are then defined as follows:

$$\varepsilon_{i,new} = \varepsilon^* \|\underline{p}_{i+1} - \underline{p}_i\|, \tag{4.42}$$

$$\varepsilon_{i+1,new} = (1 - \varepsilon^*)\|\underline{p}_{i+1} - \underline{p}_i\|. \tag{4.43}$$

The second part of the algorithm determines time-optimal linear segments from $\underline{p}_i$ to $\underline{p}_{i+1}$. Before determining the point-to-point parameters, the begin and end position ($\underline{p}_s$ and $\underline{p}_e$) and velocities ($v_s$ and $v_e$) of the linear segment need to be set. The velocities depend on either the constraints or the given blend radius $\varepsilon$. The velocity is determined by the unit-vector of the linear segment (see definition of a linear segment in (4.1)) and the constraints (velocity, acceleration and jerk) in each Cartesian direction.
The first and last point of the path, i.e. pick location and place location respectively, are set with a zero blend radius. This implies that the velocity at these points is equal to zero. Furthermore, when along the path a point has a blend radius equal to zero, the velocity at this point needs to be set to zero. This is done to avoid a discontinuous velocity profile. In all other cases, the velocity at the end of a segment is set to the maximum allowed velocity $v_{max}$. The maximum allowed velocity $v_{max}$ is determined by taking the the first derivative to time is given as from (4.1).

$$\dot{\underline{p}}(t) = \frac{(\underline{p}_{i+1} - \underline{p}_i)}{\|\underline{p}_{i+1} - \underline{p}_i\|}\dot{r}(t) \tag{4.44}$$

The positional constraints on velocity are given as,

$$|\dot{p}(t)_i| \leq \dot{p}_{max,i}, \text{ for } i = 1, \ldots, 3. \tag{4.45}$$

The same limitations can be defined for the angular coordinates and corresponding velocity constraints. The maximum allowed velocity on the time mapping trajectory, $r(t)$ is then determined as,

$$\dot{r}(t)_{max} = \min_i \left( \frac{(\dot{p}_{max,i} \cdot \|\underline{p}_{i+1} - \underline{p}_i\|)}{(\underline{p}_{i+1} - \underline{p}_i)}, \frac{(\dot{\phi}_{max,i} \cdot \|\underline{\phi}_{i+1} - \underline{\phi}_i\|)}{(\underline{\phi}_{i+1} - \underline{\phi}_i)} \right). \tag{4.46}$$

When the valid begin and end positions and velocities are set, the point-to-point parameters can be determined. This is done by the procedure described in section 4.2.1. Next, we need to check

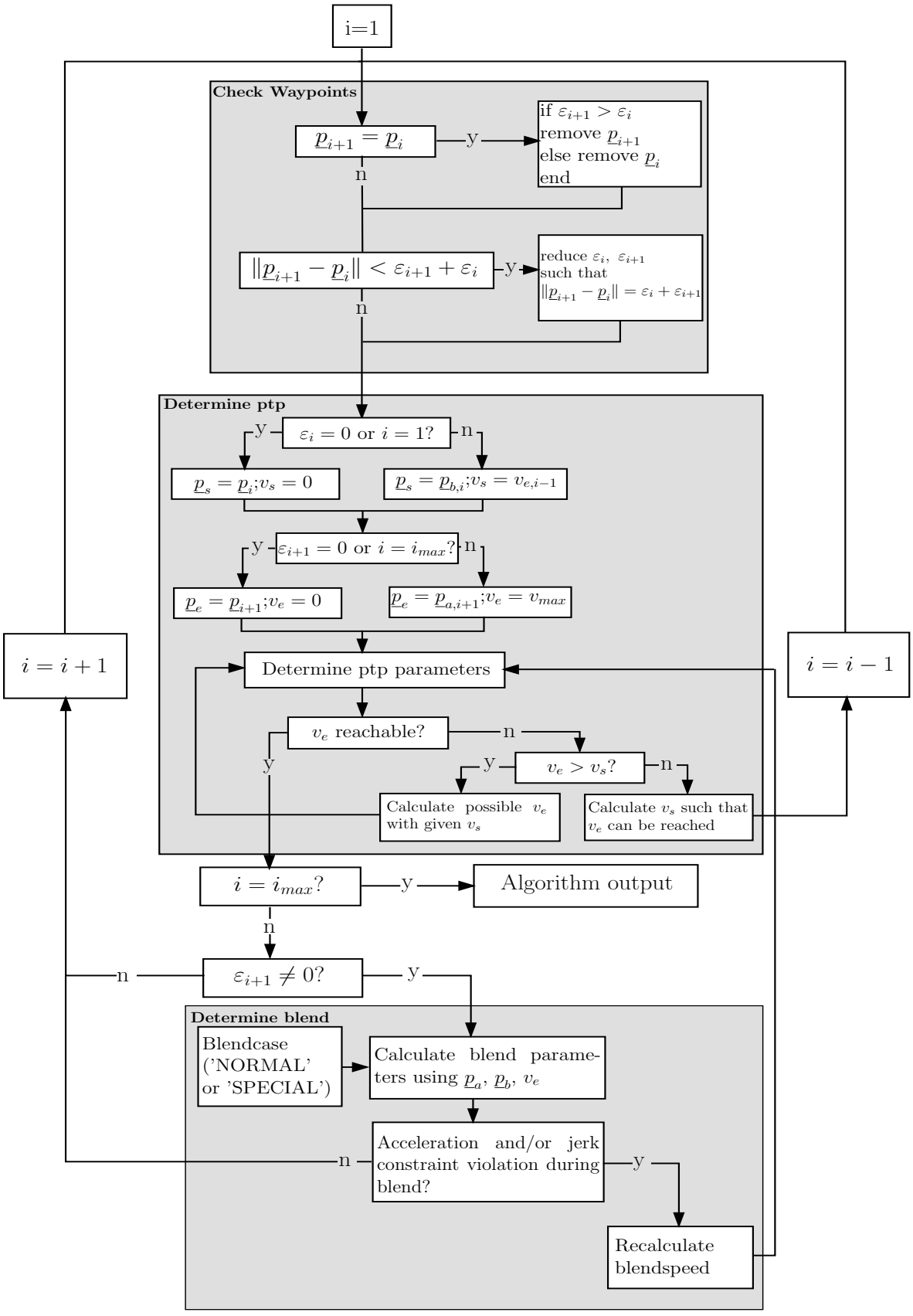



Figure 4.3: Structure of the algorithm. Subscripts $s$ and $e$ denote the start and end positions and velocities of linear segment $i$, respectively.

whether the specified velocity can be reached over the given distance from $\underline{p}_i$ to $\underline{p}_{i+1}$. If the specified velocity can be reached, we can proceed to the next part of the algorithm. Otherwise, we need to check whether the begin or end velocity of the segment needs to be recalculated. In such case, when the end velocity is larger than the begin velocity, we need to recalculate the end velocity of the segment, since there is not enough distance to overcome the change in velocity. On the other hand, if the end velocity is smaller than the begin velocity, we need to recalculate the begin velocity. The begin velocity needs to be adapted because the end velocity is the maximum allowed velocity such that the constraints on velocity, acceleration and jerk are not violated inside the next blend. Furthermore the end velocity can be equal to zero in case of a stop point (i.e. a point with a zero blend radius). Since the begin velocity of the segment is equal to the end velocity of the previous segment, this implies that we need to go back in the path calculation and recalculate the previous segment, i.e. $i = i - 1$ (see Figure 4.3).
When the calculated segment is not equal to the final segment and the specified blend radius is unequal to zero, we can proceed to the next part. Otherwise, the algorithm is finished or, in case of a zero blend radius, we can proceed to the next segment.

The last part of the algorithm determines the blend parameters (i.e. $\underline{b}_i$, $\underline{m}_i$, $spd_{blend}$ and $\tau$ with $i = 1, 2$), using the blend begin and end positions determined by (4.17) and the end velocity of the calculated segment. Furthermore, the blend parameters depend on the specified transition shape, determined by the parameters $s_s$ and $s_h$. The type of transition shape can either be set to 'NORMAL' or 'SPECIAL', where 'NORMAL' provides a blend which lies within the isosceles triangle determined by $\underline{p}_a - \underline{p}_2$, $\underline{p}_2 - \underline{p}_b$ and $\underline{p}_a - \underline{p}_b$ and 'SPECIAL' provides a blend which lies between isosceles triangle determined by $q - \underline{p}_2$, $\underline{p}_2 - r$ and $q - r$ and passes through point $\underline{p}_2$ exactly (see Figure 4.2). When the blend parameters are determined, we need to check whether the acceleration and jerk constraints are violated by using (4.36), (4.38) and (4.39). When the outcome of (4.39) is not equal to $spd_{blend}$ we need to recalculate the blend speed at the start and end of the blend. Furthermore, this implies that we need to recalculate the linear connecting segment (see Figure 4.3). Otherwise, the next linear segment can be calculated, i.e. $i = i + 1$.

The process continues until $i$ is equal to the number of segments to be calculated, i.e. $i_{max}$. The outcome of the algorithm then consists of a set of (time-optimal) point-to-point and blend parameters which can be interpolated such that at each sample-instant the position is fed to the motion control system (see further Section 3.3).

## 4.4 Discussion

As can be seen from this chapter, linear segments and blend theory can be used to describe pick-and-place motions. Hereby, it is assumed that a user acts like a high-level path planner, which is able to plan a desired path without any collisions between manipulator and (moving) objects in the workspace of a manipulator.

The algorithm presented in this chapter determines paths in the workspace of a manipulator. The method can be used to determine (near) time-optimal pick and place motions, while respecting velocity, acceleration and jerk constraints given in the workspace of the manipulator. Only constraints supplied in the domain of calculation are taken into account. Handling constraints acting in different domains will result in a complex optimization problem, as will be discussed

in detail in the next chapter. The satisfaction of possible geometric constraints is left to the user, who can specify waypoints with a certain blend radius characterizing spheres in the space in which it is allowed to deviate from a linear segment. Furthermore, the method is applicable in case of certain simple contouring applications , i.e. where tracking of straight lines is desired (e.g. cutting rectangular objects from a workpiece in laser cutting applications). Using a special blend transition shape, discontinuities in velocity and acceleration are prevented. The determination of this blend shape was already available from literature, however the application in such algorithm structure is new.
The presented method can easily be extended to determine linear segments with blends in the joint space. Determining linear segments with blends in joint space has the benefit that only the given waypoints need to be transformed to joint space instead of a whole path. A disadvantage is the fact that the path in the workspace of a manipulator is not known in advance, so that is it not clear whether geometric constraints are satisfied.

The method is not applicable in cases where a prescribed path does not consist of, or can be approximated by, linear segments. Therefore, in the following chapter a method is presented for determining (near time-optimal) motions for paths which cannot be described by linear segments. Simulations and experiments performed using the algorithm presented in this chapter, are discussed in Chapter 6.

# Chapter 5

# Path-constrained motion planning

## 5.1 Introduction

In the previous chapter, a trajectory generation algorithm is presented which can deal with paths that consist of, or can be approximated by, linear segments. Since one of the goals of this work is to generate trajectories which are applicable for a wide range of applications, also paths that do not consist of or cannot be approximated by linear segments must be determined. Therefore, in this chapter a method is presented that is able to determine (time-optimal) motions for paths that cannot be determined by the algorithm presented in the previous chapter.
In general we refer to this type of motions as path-constrained motions. With path-constrained motion it is meant that the path which the end-effector of a manipulator has to follow is fully known. The path is described as a function of the path parameter $s$. So, the motion can be described in one degree-of-freedom (DOF) [Pfeiffer, 1987]. Moreover, the feasible motion is constrained as a consequence of actuator limitations and limitations present in the workspace of a manipulator (as discussed in Chapter 3). These limitations are defined in different spaces, namely joint space and Cartesian space. In order to respect each of these constraints, we need to redefine them in terms of the path parameter $s$. This subject will be treated in Section 5.2. Besides constraints, the motion must fulfil a certain requirement. Here the requirement that the motions need to be time-optimal is treated. So, after rewriting the constraints, the optimization problem to obtain time-optimal motions will be formulated in Section 5.3. Finally, the algorithm to solve the optimization problem will be discussed in Section 5.4.

## 5.2 Constraint formulation

As discussed in the previous section, the overall problem is to find time-optimal trajectories subjected to constraints on the path (geometrical constraints such that the end effector of a manipulator will stay on the path), actuator constraints and process constraints. The path, that must be followed by the manipulator's end-effector, can be represented by a six-dimensional vector $\underline{P}$, which consists of a translational component $\underline{P}_t$ (with three translational coordinates) and an angular component $\underline{P}_o$ (with three angular coordinates), relative to some reference frame,

$$\underline{P} = (\underline{P}_t^T, \underline{P}_o^T)^T. \tag{5.1}$$

As stated above, the path $\underline{P}$(s) will be parameterized by a single parameter $s$,

$$\underline{P} = \underline{P}(s). \tag{5.2}$$

The actuator constraints are defined as,

$$\begin{aligned} &|\dot{q}_i| \leq \dot{q}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_j, \\ &|\ddot{q}_i| \leq \ddot{q}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_j, \\ &|\dddot{q}_i| \leq \dddot{q}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_j, \end{aligned} \tag{5.3}$$

with $q_i$ the displacement of joint $i$ and $n_j$ the number of joints. Here the assumption that a joint only allows a rotation or translation in one degree-of-freedom is used. The process constraints are defined as,

$$\begin{aligned} &|\dot{X}_i| \leq \dot{X}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_c, \\ &|\ddot{X}_i| \leq \ddot{X}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_c, \\ &|\dddot{X}_i| \leq \dddot{X}_{i,max}, \text{ for } i = 1,\ 2,\ \ldots,\ n_c, \end{aligned} \tag{5.4}$$

where $n_c$ represents the number of degrees of freedom in the workspace of the manipulator ($1 \leq n_c \leq 6$). In the following sections, we will transform the constraints defined above to constraints on the path parameter $s$. First, the actuator constraint transformation will be discussed. Second, the process constraints will be treated and finally the results will be combined to derive the set of constraints on the path parameter.

### 5.2.1 Actuator constraints

As already discussed in Chapter 2, actuator constraints can be transformed to constraints in terms of the path parameter, using the dynamic equations or the kinematic relations of a robotic system. Here, we will derive constraints on the path parameter based on actuator limitations using the kinematic relations of a robotic system. The reason for this choice is the fact that the derivation of the dynamic equations of a robotic system requires specific knowledge on multibody dynamics, while the (forward) kinematical equations can be determined in a straightforward manner. In this way, the industrial applicability of the strategy is increased. The actuator limitations can be e.g. determined using the actuator data sheet or by performing some experiments. By using velocity, acceleration and jerk constraints instead of torque and torque rate constraints the method will be conservative. However, as already discussed the industrial applicability will be highly increased. The path, defined by (5.2), can be written in terms of the joint angle displacements using the kinematic relations of the manipulator, $\underline{R}(\underline{q})$:

$$\underline{P}(s(t)) = \underline{R}(\underline{q}(t)), \tag{5.5}$$

with $\underline{q} = [q_1, \ldots, q_{n_j}]$ and $\underline{R}(\underline{q}) = [R_1(\underline{q}), \ldots, R_{n_c}(\underline{q})]$. In order to determine the constraints on the path velocity $\dot{s}$, induced by the actuator constraints, (5.5) is differentiated with respect to time.

$$\underline{R}_{\underline{q}}(\underline{q})\dot{\underline{q}} = \underline{P}_s(s)\dot{s}, \tag{5.6}$$

with

$$(\underline{R}_{\underline{q}}(\underline{q}))_{ij} = \frac{\partial R_i(\underline{q})}{\partial q_j} \text{ and } \underline{P}_{i,s}(s) = \frac{dP_i(s)}{ds}. \tag{5.7}$$

When the inverse of the Jacobian $\underline{R}_{\underline{q}}(\underline{q})$ exists and the manipulator is non-redundant we can determine the joint velocity angles as,

$$\dot{\underline{q}} = (\underline{R}_{\underline{q}}(\underline{q}))^{-1}\underline{P}_s(s)\dot{s}. \tag{5.8}$$

Using the actuator velocity constraints from (5.3), the constraint on $\dot{s}$ can be determined as,

$$\dot{s}_{i,ACvmax} \leq \frac{\dot{q}_{i,max}}{|(\underline{R}_{\underline{q}}(\underline{q}))^{-1}\underline{P}_s(s)|_i}. \tag{5.9}$$

So, the constraint on $\dot{s}(t)$ depends on $s(t)$ (and thus also on $\underline{q}(t)$, see (5.5)). Differentiating (5.5) twice with respect to time results, for $i-$th element of $\underline{R}(\underline{q}(t))$ and $\underline{P}(s(t))$, in

$$\sum_{k=1}^{n_j}\left(\sum_{j=1}^{n_j}\frac{\partial^2 R_i(\underline{q})}{\partial q_k \partial q_j}\dot{q}_k\dot{q}_j + \frac{\partial R_i(\underline{q})}{\partial q_k}\ddot{q}_k\right) = P_{i,ss}(s)\dot{s}^2 + P_{i,s}(s)\ddot{s},\ i = 1,\ldots,n_c. \tag{5.10}$$

where $P_{i,ss}(s) = \frac{dP_i}{ds^2}$. Using the notation $(\underline{R}_{\underline{q}}(\underline{q}))_{ij} = \frac{\partial R_i(\underline{q})}{\partial q_j}$ and $(R_{i,\underline{qq}}(\underline{q}))_{jk} = \frac{\partial^2 R_i(\underline{q})}{\partial q_k \partial q_j}$, we can rewrite (5.10) as follows:

$$\underline{\dot{q}}^T R_{i,\underline{qq}}(\underline{q})\underline{\dot{q}} + R_{i,\underline{q}}(\underline{q})\underline{\ddot{q}} = P_{i,ss}(s)\dot{s}^2 + P_{i,s}(s)\ddot{s},\ i = 1,\ldots,n_c. \tag{5.11}$$

Combining 5.8 and (5.11), we obtain

$$\underline{\ddot{q}} = \underline{R}_{\underline{q}}^{-1}(\underline{q})\left[(\underline{P}_{ss}(s) - \underline{Q})\dot{s}^2 + \underline{P}_s(s)\ddot{s}\right], \tag{5.12}$$

with

$$Q_i = \underline{P}_s^T(s)\underline{R}_{\underline{q}}^{-T}(\underline{q})R_{i,\underline{qq}}(\underline{q})\underline{R}_{\underline{q}}^{-1}(\underline{q})\underline{P}_s(s) \text{ for } i = 1,\ 2,\ \ldots,\ n_c. \tag{5.13}$$

For the case in which $\ddot{s} = 0$, i.e. when the net acceleration capacity of the axes is available for centripetal acceleration [van Tuijl, 1991], this gives a restriction on $\dot{s}$. Using $\ddot{s} = 0$ and substituting the acceleration constraints on joint level (i.e. $\underline{\ddot{q}}_{max,i},\ i = 1,\ldots,n_j$) into (5.12), the limitation on $\dot{s}$ due to maximum acceleration is given as

$$\dot{s}_{ACamax,i} = \sqrt{\frac{\ddot{q}_{max,i}}{|\underline{R}_{\underline{q}}(\underline{q})^{-1}\left(\underline{P}_{ss}(s) - \underline{Q}\right)|_i}}. \tag{5.14}$$

Consequently, using (5.9) and (5.14) the overall admissible $\dot{s}$ due to actuator velocity and acceleration constraints, can be determined as,

$$\dot{s}_{max,ac} = \min_i(\dot{s}_{ACamax,i}, \dot{s}_{ACvmax,i}). \tag{5.15}$$

Next, when $\ddot{s} \neq 0$ the limitation on $\ddot{s}$ is determined as,

$$\frac{-\ddot{q}_{max,i} + |\underline{R}_{\underline{q}}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i\dot{s}^2}{|\underline{R}_{\underline{q}}(\underline{q})^{-1}\underline{P}_s(s)|_i} \leq \ddot{s}_i \leq \frac{\ddot{q}_{max,i} - |\underline{R}_{\underline{q}}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i\dot{s}^2}{|\underline{R}_{\underline{q}}(\underline{q})^{-1}\underline{P}_s(s)|_i}, \tag{5.16}$$

and, therefore the minimum and maximum values for $\ddot{s}$ are given by:

$$\ddot{s}_{min,ac} = \max_i\left(\frac{-\ddot{q}_{i,max} + |\underline{R}_{\underline{q}}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i\dot{s}^2}{|\underline{R}_{\underline{q}}(\underline{q})^{-1}\underline{P}_s(s)|_i}\right), \tag{5.17}$$

$$\ddot{s}_{max,ac} = \min_i\left(\frac{\ddot{q}_{i,max} - |\underline{R}_{\underline{q}}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i\dot{s}^2}{|\underline{R}_{\underline{q}}(\underline{q})^{-1}\underline{P}_s(s)|_i}\right). \tag{5.18}$$

As can be seen from (5.16), the constraints on $\ddot{s}$ depend on the maximum joint acceleration levels, path parameter $s$ and pseudo-velocity $\dot{s}$. The limits on $\ddot{s}$ are determined by evaluating (5.16) at each point along the path, for pseudo-velocities varying from $0 \leq \dot{s} \leq \dot{s}_{max}$, where $\dot{s}_{max}$ will be defined later on when the limits due to actuator constraints and process constraints will be combined. So, at each point along the path with a certain pseudo-velocity $\dot{s}_i$ a maximum and minimum pseudo-acceleration can be determined.
Finally, the constraints due to the jerk limitations on the actuator need to be determined. Differentiating (5.10) with respect to time results in,

$$\sum_{l=1}^{n_j}\sum_{k=1}^{n_j}\sum_{j=1}^{n_j}\frac{\partial^3 R_i(\underline{q})}{\partial q_l \partial q_k \partial q_j}\dot{q}_j\dot{q}_k\dot{q}_l + 3\sum_{k=1}^{n_j}\sum_{j=1}^{n_j}\frac{\partial^2 R_i(\underline{q})}{\partial q_k \partial q_j}\ddot{q}_k\dot{q}_j + \sum_{k=1}^{n_j}\frac{\partial R_i(\underline{q})}{\partial q_k}\dddot{q}_k = P_{i,sss}(s)\dot{s}^3 + 3P_{i,ss}(s)\dot{s}\ddot{s} + P_{i,s}(s)\dddot{s}. \tag{5.19}$$

Equation (5.19) can be written in a matrix notation as follows:

$$R_{i,\underline{qqq}}(\underline{q},\dot{\underline{q}})\dot{\underline{q}} + 3\ddot{\underline{q}}^T R_{i,\underline{qq}}(\underline{q})\dot{\underline{q}} + R_{i,\underline{q}}(\underline{q})\dddot{\underline{q}} = P_{i,sss}(s)\dot{s}^3 + 3P_{i,ss}(s)\dot{s}\ddot{s} + P_{i,s}(s)\dddot{s}, \tag{5.20}$$

where $\dot{q}$ and $\ddot{q}$ can be determined using (5.8) and (5.12), respectively, and

$$\left(R_{i,\underline{qqq}}(\underline{q},\dot{\underline{q}})\right)_j = \dot{\underline{q}}^T R_{i,\underline{qqq}j}(\underline{q})\dot{\underline{q}} \text{ with } \left(R_{i,\underline{qqq}j}(\underline{q})\right)_{kl} = \frac{\partial^3 R_i(\underline{q})}{\partial q_l \partial q_k \partial q_j}. \tag{5.21}$$

From (5.20) we can derive limitations on $\dddot{s}$. This then results in,

$$\frac{-|R_{i,\underline{q}}\dddot{q}_{max,i} + Z_i|}{|P_{i,s}(s)|} \leq \dddot{s} \leq \frac{|R_{i,\underline{q}}\dddot{q}_{max,i} + Z_i|}{|P_{i,s}(s)|}, \tag{5.22}$$

with

$$Z_i = R_{i,\underline{qqq}}\underline{R}_{\underline{q}}^{-1}(\underline{q})\underline{P}_s\dot{s} + 3\left[\underline{R}_{\underline{q}}^{-1}(\underline{q})\left(\left(\underline{P}_{ss}(s) - \underline{Q}\right)\dot{s}^2 + \underline{P}_s\ddot{s}\right)\right]^T R_{i,\underline{qq}}\underline{R}_{\underline{q}}^{-1}(\underline{q})\underline{P}_s\dot{s} - 3P_{i,ss}\dot{s}\ddot{s} - P_{i,sss}\dot{s}^3 \tag{5.23}$$

where $\underline{Q}$ is defined as in (5.13). At this point, it assumed that the influence of the jerk joint constraints on the limitation of pseudo-velocity $\dot{s}$ and pseudo-acceleration $\ddot{s}$ can be neglected. This completes the derivation of the limitations in terms of the path parameter $s$ based on constraints on the actuators. In the next section, the derivation of limitations in terms of the path parameter is extended with limitations on velocity, acceleration and jerk in terms of process constraints.

### 5.2.2 Process constraints

The process constraints acting on the path can be determined in the same manner as the actuator constraints, that are discussed above. Since the path is defined in the workspace of a manipulator, the kinematics of the manipulator are not needed. The path in the workspace is defined as

$$\underline{X}(t) = \underline{P}(s(t)). \tag{5.24}$$

Differentiating this description with respect to time results in,

$$\dot{\underline{X}}(t) = \underline{P}_s(s(t))\dot{s}(t). \tag{5.25}$$

Using the constraints from (5.4), we can determine the maximum admissible value of $\dot{s}$ due to the maximum velocity allowed in the workspace,

$$\dot{s}_{PCvmax,i} = \begin{cases} \frac{\dot{X}_{i,max}}{|P_{i,s}(s)|}, \text{ if } P_{i,s}(s), \neq 0 \\ \infty, \text{ if } P_{s,i}(s) = 0. \end{cases} \tag{5.26}$$

By determining the second derivative with respect to time from (5.24), and regarding the case for which $\ddot{s} = 0$, the maximum pseudo-velocity $\dot{s}_{max,pc}$ due to the maximum process acceleration can be determined as,

$$\dot{s}_{PCamax,i} = \sqrt{\frac{\ddot{X}_{i,max}}{|P_{i,ss}(s)|}} \tag{5.27}$$

$$\dot{s}_{max,pc} = \min_i(\dot{s}_{PCvmax,i}, \dot{s}_{PCamax,i}). \tag{5.28}$$

The maximum pseudo-acceleration $\ddot{s}_{max,pc}$ is determined as,

$$\frac{-\ddot{X}_{i,max} + |P_{i,ss}(s)|\dot{s}^2}{|\underline{P}_{i,s}(s)|} \leq \ddot{s}_{max,pc,i} \leq \frac{\ddot{X}_{i,max} - |\underline{P}_{i,ss}(s)|\dot{s}^2}{|\underline{P}_{i,s}(s)|}. \tag{5.29}$$

Finally, the limitations on the third derivative of the path parameter $s$ with respect to time, induced by jerk process constraints are determined. The third derivative of (5.24) with respect to time is defined as,

$$\dddot{\underline{X}} = \underline{P}_{sss}(s)\dot{s}^3 + 3\underline{P}_{ss}(s)\dot{s}\ddot{s} + \underline{P}_s(s)\dddot{s}. \tag{5.30}$$

The limitations on $\dddot{s}$ are now defined as follows,

$$\frac{-(\dddot{X}_{i,max} - |\underline{P}_{sss}(s)\dot{s}^3 + 3\underline{P}_{ss}(s)\dot{s}\ddot{s}|_i)}{|P_{i,s}(s)|} \leq \dddot{s}_{max,pc,i} \leq \frac{(\dddot{X}_{i,max} - |\underline{P}_{sss}(s)\dot{s}^3 + 3\underline{P}_{ss}(s)\dot{s}\ddot{s}|_i)}{|P_{i,s}(s)|}. \tag{5.31}$$

In the next section, the results obtained in this section and previous section are combined such that the limitations on the path parameters are determined uniquely.

### 5.2.3 Combining actuator and process constraints

Here, the constraints on the pseudo-velocity, -acceleration and -jerk determined using the actuator constraints and process constraints are gathered such that one constraint definition is determined. For the pseudo-velocity this results in,

$$0 < \dot{s} \leq \min(\dot{s}_{max,ac}, \dot{s}_{max,pc}). \tag{5.32}$$

The same can be stated for the pseudo-acceleration constraint:

$$a(\dot{s}, s) \leq \ddot{s} \leq b(\dot{s}, s) \tag{5.33}$$

with,

$$a(\dot{s}, s) = \max_i(a_{ac,i}(\dot{s}, s), a_{pc,i}(\dot{s}, s)), \tag{5.34}$$

$$b(\dot{s}, s) = \min_i(b_{ac,i}(\dot{s}, s), b_{pc,i}(\dot{s}, s)). \tag{5.35}$$

The values $a_{ac,i}$, $b_{ac,i}$, $a_{pc,i}$ and $b_{pc,i}$ are defined as

$$a_{ac,i} = \frac{-\ddot{q}_{i,max} + |\underline{R_q}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i \dot{s}^2}{|\underline{R_q}(\underline{q})^{-1}\underline{P}_s(s)|_i}, \tag{5.36}$$

$$b_{ac,i} = \frac{\ddot{q}_{i,max} - |\underline{R_q}(\underline{q})^{-1}(\underline{P}_{ss}(s) - \underline{Q})|_i \dot{s}^2}{|\underline{R_q}(\underline{q})^{-1}\underline{P}_s(s)|_i}, \tag{5.37}$$

$$a_{pc,i} = \frac{-\ddot{X}_{i,max} + |\underline{P}_{ss}(s)|_i \dot{s}^2}{|\underline{P}_{i,s}(s)|}, \tag{5.38}$$

$$b_{pc,i} = \frac{\ddot{X}_{i,max} - |\underline{P}_{ss}(s)|_i \dot{s}^2}{|\underline{P}_{i,s}(s)|}. \tag{5.39}$$

The total pseudo-jerk limitation $\dddot{s}$ is defined as follows,

$$c(s, \dot{s}, \ddot{s}) \leq \dddot{s} \leq d(s, \dot{s}, \ddot{s}), \tag{5.40}$$

with,

$$c(s, \dot{s}, \ddot{s}) = \max_i(c_{ac,i}, c_{pc,i}), \tag{5.41}$$

$$d(s, \dot{s}, \ddot{s}) = \min_i(d_{ac,i}, d_{pc,i}). \tag{5.42}$$

The variables $c_{ac,i}, c_{pc,i}, d_{ac,i}, d_{pc,i}$ are defined by,

$$c_{ac,i} = \frac{-|R_{i,\underline{q}}\ddot{q}_{max,i} + Z_i|}{|P_{i,s}(s)|}, \tag{5.43}$$

$$c_{pc,i} = \frac{-(\dddot{X}_{i,max} - |\underline{P}_{sss}(s)\dot{s}^3 + 3\underline{P}_{ss}(s)\dot{s}\ddot{s}|_i)}{|P_{i,s}(s)|}, \tag{5.44}$$

$$d_{ac,i} = \frac{|R_{i,\underline{q}}\ddot{q}_{max,i} + Z_i|}{|P_{i,s}(s)|}, \tag{5.45}$$

$$d_{pc,i} = \frac{(\dddot{X}_{max,i} - |\underline{P}_{sss}(s)\dot{s}^3 + 3\underline{P}_{ss}(s)\dot{s}\ddot{s}|_i)}{|P_{i,s}(s)|}, \tag{5.46}$$

where $\underline{Z_i}$ is defined by (5.23). Since the constraints on the path parameter and corresponding time-derivatives are defined, the optimization problem can be defined. This optimization problem is discussed in the following section. In Section 5.4, the solution to the optimization problem is discussed.

## 5.3 Optimization problem formulation

This section will state the optimization problem which needs to be solved. In principle, the problem of path-constrained motion planning is to find a function $s(t)$ satisfying the constraints, as defined in the previous section, and the requirement that the motion is time-optimal. This problem can be written as a Time-Optimal Control (TOC) problem. The general optimal control problem is formulated as follows. Consider the system

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, t), \tag{5.47}$$

subject to a performance criterium,

$$J = \Phi(\underline{x}(T), T) + \int L(\underline{x}, \underline{u}, t)dt, \tag{5.48}$$

with $\underline{x}$ the state variables of the system, $\underline{u}$ the control inputs and $t$ the time variable. Furthermore, $\Phi(\underline{x}(T), T)$ and $L(\underline{x}, \underline{u}, t)$ represent some performance criteria functions in terms of the system variables. $T$ represents the total motion time. In case of a Time-Optimal Control problem, the goals is to find input $\underline{u}$ while respecting constraints on input $\underline{u}$, and possibly states $\underline{x}$, such that the total motion time it takes to move from the initial location $\underline{x}_0$ to the final location $\underline{x}_f$ is minimized. In case of a Time-Optimal Control problem, the performance criterium in (5.48) can be reduced to,

$$J = \int_{t_0}^{t_f} dt. \tag{5.49}$$

Constantinescu and Croft [Constantinescu, 2000] show that in case of path-constrained motion, the system can be written as

$$\dot{\underline{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{5.50}$$

with the states of the system defined as

$$\underline{x} = \left[s,\ \dot{s},\ \ddot{s}\right]^T, \tag{5.51}$$

and control input,

$$u = \dddot{s}. \tag{5.52}$$

The problem is now to find input $\underline{u}$ such that (5.49) is minimized, subject to the dynamics defined by (5.50), with boundary conditions,

$$\begin{aligned} \underline{x}_0 &= \left[s_0,\ \dot{s}_0,\ \ddot{s}_0\right], \\ \underline{x}_f &= \left[s_f,\ \dot{s}_f,\ \ddot{s}_f\right], \end{aligned} \tag{5.53}$$

while respecting the constraints on the states $\underline{x}$, and the control saturation levels $\underline{u}_{min} \leq \underline{u} \leq \underline{u}_{max}$ derived in Section 5.2. In order to solve the problem, an optimization algorithm needs to be solved. The algorithm must be able to handle the constrained optimization problem directly. Time-optimal control problems with bounded controls, have been solved analytically using

Pontryagin's minimum principle[1] [Athans, 1966; Lee, 1967; Bryson, 1975]. However, these solutions are limited to second-order systems. Since here the system has an order greater than two, this principle cannot be used. Despite the research done in the field of time optimal control, time optimal control for high-order systems is still an open issue [Penev, 2005]. However, for higher-order systems, or systems where both control and state constraints are present, the time-optimal control problem can be solved numerically. Malmgren and Nordström [Malmgren, 1995] and Kim et al. [Kim, 2002] present numerical solutions for the time optimal control problem for second-order systems with constrained inputs and states. The control law is defined as a state feedback control law. Malmgren and Nordström solve the problem using $l_\infty$-contraction control theory, while Kim et al. [Kim, 2002] show that the time-optimal trajectory is constructed by two curves, a forward velocity limitation curve and a backward velocity limitation curve, which are obtained by solving two scalar ordinary differential equations. The ODE's are solved using well-known Euler or Runge-Kutta integration methods. Penev and Christov [Penev, 2005] present a method to solve the time-optimal control problem for high-order systems with a fixed structure. They consider a linear system of order $k$,

$$\underline{\dot{x}}_k = \underline{A}_k \underline{x}_k + \underline{B}_k \underline{u}_k \tag{5.54}$$

Where matrix $\underline{A}_k$ has a diagonal structure and every element of $\underline{B}_k$ is nonzero. The time-optimal control problem is solved by first solving the problem analytically for the case where $k = 1$, then a numerical procedure is performed which increases the order $k = k + 1$ up to the total order of the system, and in each step determines the input $\underline{u}_k$ for the time-optimal control problem. Sim et al. [Sim, 2000] use a combination of genetic algorithms and the shooting method to determine the solution to the time-optimal control problem. The shooting method is used since a genetic algorithm in general does not determine the optimal solution. Therefore, a genetic algorithm is used to determine initial guesses for the shooting method. The difficulty in applying this method lies in setting up a fitness function for a genetic algorithm, such that state and control constraints are not violated.

As becomes clear from the discussion above, the time-optimal control problem defined by (5.49) and (5.50) cannot be solved analytically. Numerical methods can be used to solve the problem. Most of the numerical methods have been developed for only a limited class of linear systems. However, a method which uses a combination of a genetic algorithm and a local optimization method can be used for several classes of linear systems.
Here, in order to find the function $s(t)$ we combine the methods presented by Constantinescu [Constantinescu, 2000] and [Sim, 2000]. This implies that the trajectory planning problem is solved in the $(s, \dot{s})-$plane using a combination of a global and (followed by) a local optimization strategy. The motivation for this choice is the fact that in the $(s, \dot{s})-$plane the begin and end-points are fixed, while in the time-domain the end-time is not known. Moreover, the solution space is non-convex. Furthermore, as discussed above, no analytic/numerical solution for the time-optimal control problem can be found for this typical linear system. In the next section, the strategy for solving the optimization problem is elaborated.

[1] In Russian literature it is referred to as Pontryagin's maximum principle due to a different sign convention.

## 5.4 Solution to the optimization problem

Instead of solving the time-optimal control problem, presented in the previous section, directly, we opt to solve the trajectory planning problem in the $(s, \dot{s})-$plane. The same approach as presented by Constantinescu and Croft [Constantinescu, 2000] is followed, however the solution is obtained in a different manner. Constantinescu and Croft first solve the problem in which no constraints exist on the pseudo-jerk, i.e. $\dddot{s}_{max} = \infty$. From this procedure the obtained switching points (a switching point is a point in the $(s, \dot{s})-$plane where the pseudo-acceleration switches from maximum acceleration to minimum acceleration or from minimum acceleration to maximum acceleration) in the $(s, \dot{s})-$plane are connected by cubic splines. Cubic splines are chosen because they are the lowest degree polynomials which results in a smooth motion, i.e. continuous and differentiable everywhere [Constantinescu, 2000]. Since the obtained switching points cannot be connected using cubic splines without violating the constraints (in the actual problem the pseudo-jerk limitation is not infinite), an optimization algorithm is used to determine *new* switching points such that the pseudo-velocity, -acceleration, -jerk and furthermore the performance criterium are satisfied. The algorithm used by Constantinescu and Croft is the Flexible Tolerance method [Paviani, 1969] that is based on the Nelder-Mead method [Nelder, 1965]. A disadvantage of this optimization algorithm is that it can converge to local minima in case of non-convex optimization problems. Constantinescu and Croft try to overcome this disadvantage by running the optimization four times. The best solution is then used as initial estimation for a final optimization run. However, this approach still does not guarantee that a global optimum is found. Since it is not guaranteed that the remaining solution space of the trajectory planning problem is convex the obtained solution may be a local minimum. Therefore, here a different approach is suggested.

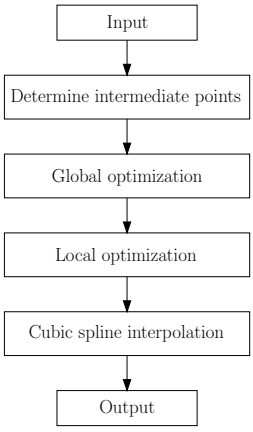



Figure 5.1: Optimization strategy to solve the path-constrained trajectory planning problem.

As discussed above here the trajectory planning problem is, as in Constantinescu and Croft, solved in the $(s, \dot{s})-$plane. However, the optimization strategy will be somewhat different. The goal of the optimization strategy is to find a set of points in the $(s, \dot{s})-$plane that are interpolated using cubic splines. An overview of the optimization strategy is depicted in Figure 5.1. As can be seen from this figure, the chosen optimization strategy is a hybrid optimization strategy. Such strategy is chosen since the solution space may be a non-convex space. A global optimization

strategy that determines points in the $(s, \dot{s})-$plane which are interpolated using cubic splines is proposed. Since in general the outcome of a global optimization strategy does not give the optimum solution, the global optimization strategy is followed by a local optimization strategy to obtain a more accurate optimum solution.

The optimization problem is defined as

$$\min \ f(\underline{O}), \ \underline{O} \in \mathbb{R}^n, \tag{5.55}$$

where $\underline{O}$ represents the set of optimization variables such that fitness function $f(\underline{O})$ is minimized. The definition of the fitness function is discussed at the end of this section. The optimization variables to this optimization problem are the parameters for determining a cubic spline. The variables are given as

$$\underline{O} = \left[ \left( \frac{d\dot{s}}{ds} \right)_0, \dot{s}_1, \ldots, \dot{s}_p, \left( \frac{d\dot{s}}{ds} \right)_f \right]. \tag{5.56}$$

As can be seen from the optimization variables, here a directional cubic spline is used instead of a natural cubic spline[2]. A visualization of the optimization variables for a case with $p = 2$ can be found in Figure 5.2. As already stated above, the optimization variables serve as points through which a cubic spline is interpolated. The variables $\left(\frac{d\dot{s}}{ds}\right)_0$ and $\left(\frac{d\dot{s}}{ds}\right)_f$ represent the slope of the directional cubic spline at the start ($s = s_0$) and end point ($s = s_f$), respectively. The slopes are part of the optimization variables for two reasons. First, often a begin and/or end pseudo-velocity is required, so these pseudo-velocities cannot be part of the set of optimization variables. Second, the start and end slopes represent the velocity at which the actuator torques leave or approach the static equilibrium values. Therefore, steeper slopes represent faster motions [Constantinescu, 2000].

The intermediate knotpoint positions for cubic spline interpolation, are determined using a strategy discussed by Shin and McKay [Shin, 1985] and Pfeiffer and Johanni [Pfeiffer, 1987], which is already discussed in Chapter 2. A search along the boundary curve (i.e. the curve which limits the maximum pseudo-velocity $\dot{s}(s)$) is performed for a point where the difference between the slope of the $(s, \dot{s})-$plane trajectory and the slope of the boundary curve changes sign:

$$\kappa(s) = \frac{d\dot{s}}{ds} - \frac{dg(s)}{ds}, \tag{5.57}$$

where, $\kappa(s)$ represents the slope difference and $g(s)$ presents the boundary curve.

As can be seen from Figure 5.1 the strategy requires certain inputs. These inputs consist of constraints on pseudo-velocity, -acceleration and -jerk, defined in Section 5.2, the initial conditions ($s_0, \ \dot{s}_0, \ \ddot{s}_0$) and final conditions ($s_f, \ \dot{s}_f, \ \ddot{s}_f$). After defining the inputs and in order to make the strategy more efficient, the intermediate knot-point positions (i.e. $s_i$ for $i = 1, \ldots, n_s$, with $n_s$ the number of intermediate points) of the cubic splines along the path are determined using the strategy discussed above.

[2] More information on spline theory can be found in [Bartels, 1987]

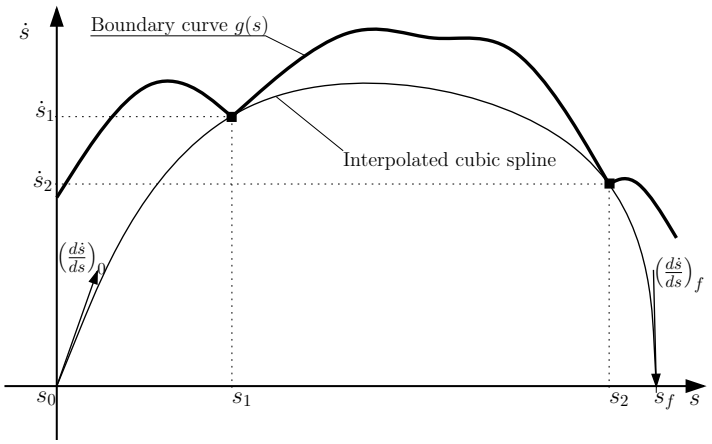



Figure 5.2: Visualization of optimization strategy.

After the intermediate points are determined, the global optimization strategy is performed. Several global optimization algorithms are available (for a brief overview and references for further reading on global optimization algorithms see [Weisstein, 2006]). Here the global optimization method suggested by Sim et al. [Sim, 2000], namely genetic algorithms, is used. Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics [Goldberg, 1989]. A genetic algorithm repeatedly modifies a population of individual solutions by a random selection. Although randomized, genetic algorithms are not simply a random walk method. They exploit historical information to determine a new population with expected improvement of performance. A genetic algorithm differs from a standard optimization algorithm in two ways. A standard algorithm generates a single point at each iteration, while a genetic algorithm generates a population of points at each iteration. Furthermore, in a standard optimization algorithm the next point in the sequence is chosen by a deterministic computation. A genetic algorithm, on other hand, selects the next population by computations that involve random choices. Simple genetic algorithms are composed of three operators, namely reproduction (or sometimes called selection), crossover and mutation. During reproduction, a selection out of an initial population of solutions is made. The selection is based on the function values of an objective function $F$. Once a solution is selected for reproduction, it enters a mating pool, i.e. a tentative new population, for further genetic operator action. Crossover proceeds in two steps. First, the solutions in the mating pool are mated at random. Next, each of the mated solutions undergoes a crossover. The crossover point is a random number between 1 and the number of optimization parameters minus one. Both the reproduction and crossover operators require the bulk of the processing power of the genetic algorithm. Therefore, there is a lot of confusion about the role of the mutation operator [Goldberg, 1989]. A mutation is a random walk through the string space. When a string is binary coded, this simply means changing a 1 to a 0 and vice versa. The mutation operator plays a secondary role in a genetic algorithm. Normally, one mutation per thousand bit takes place.

In our approach, the genetic algorithm has a population of 100 solutions. Furthermore, a crossover rate of 20 % and a mutation rate of 25 % is used. A maximum of 40 generations is used. However, when the solution stalls for 10 generations (i.e. the best solution remains the same for 10 genera-

tions) the global optimization is terminated. The settings for the genetic algorithm are verified by performing reproduction experiments. The outcome of the global optimization strategy is a set of optimization variables which in general will not result in the optimum solution. Therefore, an optimization algorithm is used that is suitable for local optimization problems. Many optimization algorithms for this type of optimization problem exist. For an overview of local optimization methods one can look at Papalambros and Wilde [Papalambros, 2000]. Here, the Nelder-Mead method is chosen [Nelder, 1965]. The Nelder-Mead method is chosen above the Steepest-Descent and line search methods. In comparison the latter methods are computationally more efficient, however they show slow convergence near the optimum solution.

The Nelder-Mead method is a so-called direct search method for finding a local minimum of several variables without using numerical or analytic gradients. For $n$ optimization variables the simplex is a $n+1-$dimensional space. For the simple case of two variables the simplex is a triangle. In each search step the function values at the vertices of the simplex are compared to the value at a point in or near the simplex. The worst vertex, where the function value is largest, is rejected and a new triangle is formed and the search continues. During the search for the local optimum, the shape of the triangle may differ. Furthermore, the size of the triangle reduces until the coordinates of the local minimum are found. The method is finished once a tolerance criterium is reached.
The outcome of the local optimization strategy is interpolated using cubic splines, see Figure 5.1. This completes the optimization strategy. The optimization algorithms discussed above evaluate a certain function, here denoted as *Fitness function*. Below, we will discuss the evaluation of the fitness function used for the optimization strategy discussed above.

The Fitness function defined for the optimization algorithms determines the total motion time. The input to the function are the optimization variables, i.e. the slopes of the trajectory in the $(s,\dot{s})-$plane and the pseudo-velocities at the intermediate points (see (5.56)). The optimization variables are interpolated using cubic splines. First, it is checked whether the constraints on pseudo-velocity, -acceleration and -jerk are violated, since both genetic algorithms and the Nelder-Mead method are unconstrained optimization strategies. When a constraint is violated the motion time for the set of optimization variables is set to,

$$T = 1 \cdot 10^{99}, \tag{5.58}$$

with $T$ the total motion time. This will result in a rejection of that typical set of optimization variables. When no constraints are violated, the total motion time can be determined. This is done in the same manner as done by Constantinescu and Croft. It is assumed that between two consecutive points along the trajectory the pseudo-jerk is constant. The time taken by the manipulator to move from point $s_i,\ \dot{s}_i,\ \ddot{s}_i$ to point $s_{i+1},\ \dot{s}_{i+1},\ \ddot{s}_{i+1}$ is determined by solving the following system,

$$\begin{aligned} s_{i+1} &= s_i + \dot{s}_i \cdot t + \frac{\ddot{s}_i \cdot t^2}{2} + \frac{\dddot{s}_i \cdot t^3}{6}, \\ \dot{s}_{i+1} &= \dot{s}_i + \ddot{s}_i \cdot t + \frac{\dddot{s}_i \cdot t^2}{2}, \end{aligned} \tag{5.59}$$

where $\dddot{s}_i$ and $t$ are the unknown variables. The pseudo-acceleration is updated by,

$$\ddot{s}_{i+1} = \ddot{s}_i + \dddot{s}_i \cdot t. \tag{5.60}$$

The total motion time $T$ can then be determined by summing all calculated sub-timings.

## 5.5 Discussion

In this chapter, a method is presented for determining paths which cannot be described or approximated by linear segments. This type of motions is referred to as a path-constrained motion. For path-constrained motions, the path is fully known is advance and can be described in one parameter; the path parameter $s$. Since the velocity, acceleration and jerk constraints are defined in terms of time and in several domains, the constraints are redefined in terms of pseudo-velocity $\dot{s}(s)$, pseudo-acceleration $\ddot{s}(s)$ and pseudo-jerk $\dddot{s}(s)$. Hereby, it is assumed that joint jerk constraints do not influence the limitation of pseudo-velocity and pseudo-acceleration.

After redefining the constraints, the optimization problem is presented. The optimization problem can be defined in terms of a Time-Optimal Control (TOC) problem. Since the order of the optimization problem is three, the TOC problem cannot be solved analytically. Several numerical methods have been developed in literature. However, these methods are mostly applicable for specific classes of linear systems. Therefore, here a different approach has been followed.

The optimization problem is solved in the $(s, \dot{s})-$plane using a strategy presented in literature. However, the optimization problem is solved in a different way due to the fact that the optimization problem can be non-convex. A hybrid optimization strategy, which consists of a genetic algorithm and the Nelder-Mead method, is presented. The motion planning strategy presented in this chapter will be evaluated by means of simulations and experiments. The results are presented in the following chapter.

# Chapter 6

# Simulations and Experiments

## 6.1 Introduction

The trajectory generation algorithms presented in Chapters 4 and 5 will be evaluated in this chapter. The evaluation is performed by means of simulations and experiments. Before discussing the actual results a performance evaluation matrix, which can be used to evaluate and compare several algorithms, is proposed. Using this evaluation matrix, it is possible to use both quantitative and qualitative requirements to gain insight in how well the motion planner requirements are fulfilled. By applying weighing coefficients, one can make distinctions between the requirements. After discussing the performance evaluation matrix, the results of simulations using the earlier presented algorithms will be discussed. After that, some experimental results will be presented.

## 6.2 Testcase development for evaluating motion planning algorithms

This section discusses the development of testcases for evaluation the performance of motion planning algorithms. Until now, no clear benchmark is available for testing and comparing several algorithms.
The problem statement for developing a testcase is defined as:

> How and on the basis of which criteria should an algorithm be assessed?

The first part of the problem statement, 'How should an algorithm be assessed', can easily be answered. Namely, by performing simulations and experiments and evaluating the results using an analysis method. The second part of the problem statement, 'On the basis of which criteria…' can be somewhat difficult to answer. First, we want to test the functionalities of the algorithm, or 'Does the algorithm work?'. Secondly, we want to evaluate the performance of the algorithm and compare it with the performance of other algorithms. The question then arises, how can the performance of an algorithm be measured? One way is looking at the total motion time. However, this is not a correct criterion. Clearly, the smallest motion time between two points is zero, gained with a zero order profile. But then the motion planner requirements, presented in Chapter 3, are not satisfied. So, we have to take more issues into account when testing algorithms for performance, such as the order of the profile, path following accuracy, etc. Therefore, we will derive a performance evaluation matrix which will be used to compare algorithms based

on a set of criteria. These criteria are based on the requirements which must be satisfied by the algorithm, and are derived from the overall requirements which must hold for the motion planner (see Chapter 3). An overview of the evaluation criteria is given in Table 6.1. As can be seen from this table, the evaluation criteria are split into two groups, namely quantitative criteria and qualitative criteria. The quantitative criteria are criteria to which a variable can be assigned. The total motion time is the time it takes to move from the first setpoint sample to the last setpoint sample. Path following accuracy is defined as the root mean square (RMS) value of the difference between the determined path and the path performed by the manipulator, while the end-point settling criterion is the steady-state error between the desired end-point and the actual reached end-point. The qualitative criteria cannot be characterized by a value, such as total motion time in seconds, but are evaluated using a certain scale. This scale can e.g. be a ranking scale between the evaluated algorithms. In this case, the number one algorithm satisfies a certain criterion the most. Another evaluation scale can be a five-point scale, where five pluses indicate that a certain criterion is completely satisfied and one plus indicates a poor satisfaction of a criterion. The joint acceleration smoothness criterion reflects the robot and actuator wear. The constraint handling criterion is added to see if constraints (e.g. velocity, acceleration and jerk constraints in Joint space or Cartesian space) are taken into account by the algorithm. Finally, a usability criterion is evaluated. With usability here the specific amount of knowledge required to generate trajectories is addressed and furthermore a "user-friendliness" index can be addressed. The evaluation between the algorithms, using the evaluation matrix presented in Table 6.1, is based on the multi-criteria analysis [TDO, 2004]. The multi-criteria analysis is a selection tool for evaluating problems with both qualitative and quantitative criteria. A weighing factor is assigned to each criterion to define the importance of the criterion. In this way it is possible to evaluate different motion planning algorithms for certain cases (each case resembles a different set of weighing factors). The values to the quantitative criteria are assigned using the results obtained from simulations and experiments. The values to the qualitative criteria are assigned by a five point scale where five pluses indicate that either a acceleration profile is smooth, all constraints (as specified in Section 3.2) are handled and the algorithm under evaluation is applicable in an industrial environment. On the other hand, one plus indicates that the algorithm under evaluation has a non-smooth acceleration profile, non constraints are handled and the algorithm is not applicable in an industrial environment, respectively. The calculation of the scores using the requirements evaluation matrix with assigned weighing factors is described in detail in [TDO, 2004]. An example of a multi criteria analysis is provided in Appendix G.

Table 6.1: Requirements evaluation matrix.

| Quantitative criteria |
|---|
| Total motion time |
| Path following accuracy |
| End-point settling |

(a) Table containing quantitative criteria

| Qualitative criteria |
|---|
| Joint acceleration smoothness |
| Constraint handling |
| Usability |

(b) Table containing qualitative criteria

Now that the evaluation matrix is presented the testcases which will be used throughout this chapter will be discussed. Since the algorithms presented in the previous chapter are applicable for different applications, the testcases will be different. Here, three testcases are chosen. The first

testcase resembles a pick-and-place motion and the second testcase represents a motion where the path is fully known in advance and cannot be approximated by linear segments. Finally, the third testcase is a straight line profile. In this way, the performance between the algorithms is evaluated. The evaluation matrices corresponding to the three testcases and corresponding weighing coefficients can be found in Appendix G.

## 6.3 Simulations

As discussed above, first the results obtained by performing simulations will be discussed. The results of the simulations will be evaluated using the evaluation matrices as discussed in the previous section. First we will discuss the results from the testcase that resembles a pick-and-place motion. Next, the result for the case in which a path is fully known is treated and finally the two algorithms will be compared for the case in which a straight line needs to be tracked. The results will be compared and evaluated using the earlier presented evaluation matrices, with results obtained from literature.

The simulations will be performed using model of a Scorbot ER VII robot. This robot is chosen because [Constantinescu, 2000] and [Macfarlane, 2003] use this robot to evaluate their algorithms. An schematic overview of the Scorbot ER VII robot is given in Figure 6.1. In the figure, the physical interpretations of the Denavit-Hartenberg parameters (except $\alpha_i$) of the Scorbot given in Table 6.3 can be found. $\alpha_i$ represents the angle between axes $z_{i-1}$ and $z_i$ rotated around $x_i$, where $x_i$ and $z_i$ are the axes of the reference frame assigned to link $i$. While the Scorbot ER VII is a five degree-of-freedom robot, here only the positional degrees of freedom are considered. Therefore, the Scorbot ER VII is considered as a three-link robot, where degrees-of-freedom $q_4$ and $q_5$ are eliminated. The simulations are performed using MATLAB/Simulink [The Mathworks Inc., 2006]. The simulation model uses the dynamic model of the Scorbot ER VII robot as presented in [Constantinescu, 2000]. Friction is modeled using Coulomb and viscous friction, with Coulomb friction coefficients of 2 $[Nm]$ and viscous friction coefficients of $0.2$ $[Nms]$ for each joint. A torque saturation level of $10$ $[Nm]$ is used. The robot is controlled using a joint independent position feedback control-loop that is tuned for critical damping and a rise-time of $200$ $ms$. In this way, the results obtained by Constantinescu [Constantinescu, 2000] and Macfarlane [Macfarlane, 2003] can be easily compared with the results obtained in this thesis. The reference trajectories generated for each simulation case are generated using the constraints listed in Table 6.2, resulting in similar torque and torque rate demands as listed in [Constantinescu, 2000].

Table 6.2: Constraints on velocity, acceleration and jerk for the simulation cases.

| Constraint | Value |
|---|---|
| Velocity $[\frac{m}{s}]$ | 0.37 |
| Acceleration $[\frac{m}{s^2}]$ | 0.89 |
| Jerk $[\frac{m}{s^3}]$ | 4.45 |

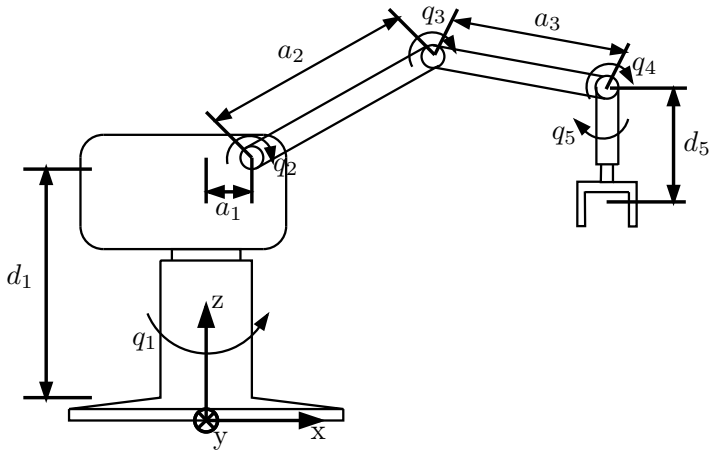



Figure 6.1: Schematic overview of the Scorbot ER VII robot.

Table 6.3: Denavit-Hartenberg parameters of Scorbot ER VII robot.

| Link | $q_i$ [rad] | $a_i$ [m] | $d_i$ [m] | $\alpha_i$ [rad] |
|---|---|---|---|---|
| 1 | $q_1$ | $a_1 = 0.05$ | $d_1 = 0.3585$ | $\alpha_1 = -\frac{\pi}{2}$ |
| 2 | $q_2$ | $a_2 = 0.3$ | $d_2 = -0.037$ | $\alpha_2 = 0$ |
| 3 | $q_3$ | $a_3 = 0.25$ | $d_3 = 0$ | $\alpha_3 = 0$ |

### 6.3.1 Results

In this section, the results obtained by performing simulations using the simulation environment as presented above.

#### Simulation case 1: Pick and place application

As discussed above, first the results obtained from simulations for a case that resembles a pick-and-place operation will be discussed. The trajectories will be determined using the linear segments with blend algorithm presented in Chapter 4. The obtained results will be compared with results obtained by using the quintic algorithm, developed by MacFarlane and Croft [Macfarlane, 2003], and an industrial pick and place algorithm[1]. In Table 6.4 the pick and place position are given. Furthermore, the intermediate points with corresponding blend radii that are supplied to the algorithms are given. These intermediate points are given such that no collisions occur with the manipulator base and a fictive object which is located in the workspace of the robot. For the industrial algorithm, only the pick and place location and the second intermediate point is used.

The linear segments with blends path is given in Figure 6.2. Moreover, the fictive obstacle present in the workspace is given, see box in Figure 6.2. The reference trajectories with the simulated output trajectories for the algorithms are presented in Figures 6.3, 6.4 and 6.5. The total motion

[1]The industrial pick and place algorithm is developed at Bosch Rexroth electric drives and control B.V. for a dedicated application.

Table 6.4: Pick and place locations with intermediate point positions and corresponding blend radii.

| X-coordinate [m] | Y-coordinate [m] | Z-coordinate [m] | blend radius [m] | |
|---|---|---|---|---|
| -0.20 | -0.4 | 0.15 | 0 | pick location |
| -0.20 | -0.4 | 0.3585 | 0.1 | |
| -0.25 | 0 | 0.3585 | 0.1 | |
| 0.05 | 0.5 | 0.3585 | 0.1 | |
| 0.05 | 0.5 | 0.15 | 0 | place location |

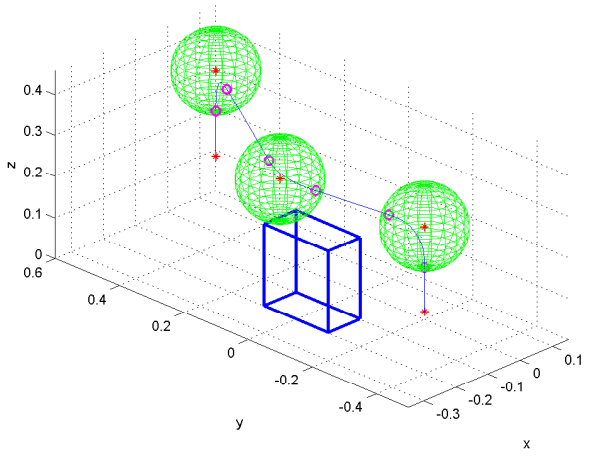
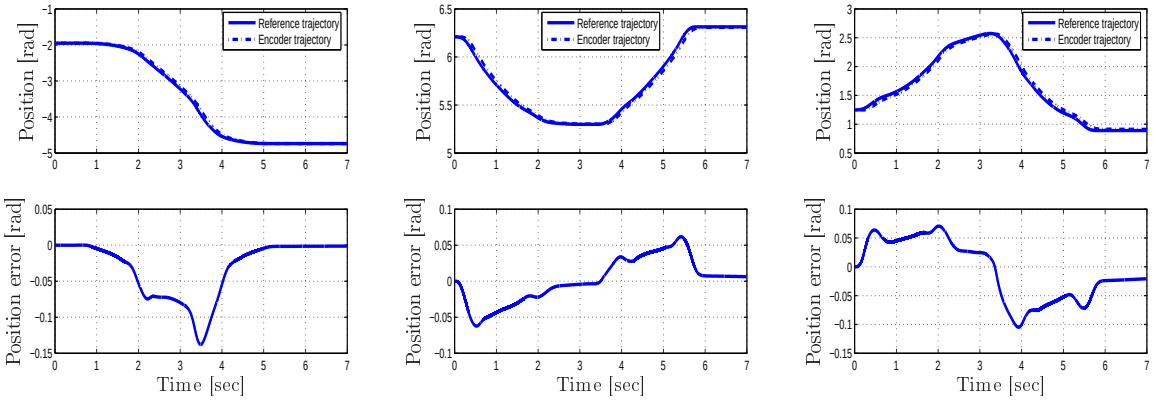



Figure 6.2: Pick and place path with obstacle in workspace of the manipulator.




(a) Axis 1 (b) Axis 2 (c) Axis 3

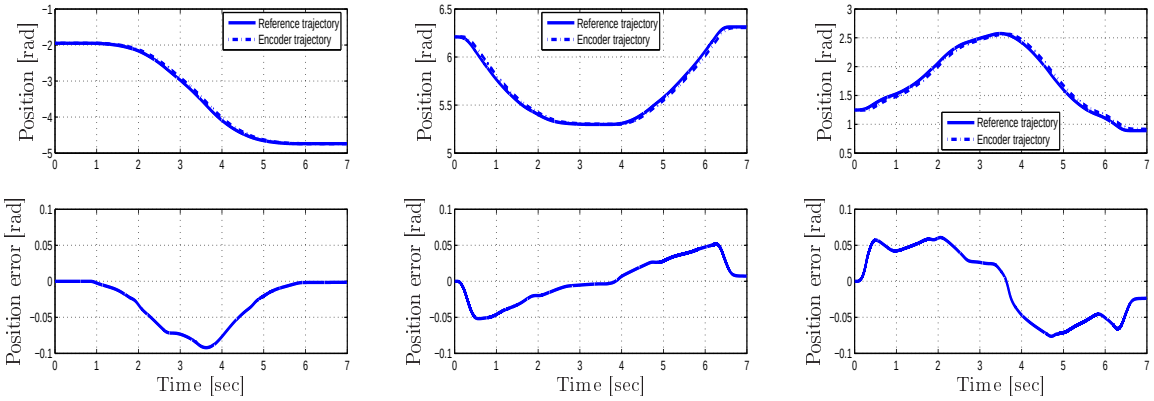
Figure 6.3: Reference and simulated trajectories for pick and place case using linear segments with blends algorithm.

Table 6.5: Simulation results of pick and place case using linear segments with blends (LSwB) algorithm, quintic algorithm and industrial algorithm.

| | LSwB algorithm | Quintic algorithm | Industrial algorithm |
|---|---|---|---|
| Total motion time [sec] | 5.8325 | 6.0750 | 4.0315 |
| Path following accuracy axis 1 [deg] | 2.8193 | 2.5139 | 2.9167 |
| Path following accuracy axis 2 [deg] | 1.7661 | 1.6710 | 2.3662 |
| Path following accuracy axis 3 [deg] | 3.005 | 2.8094 | 3.4479 |
| End-point settling axis 1 [deg] | 0.0712 | 0.0791 | 0.0533 |
| End-point settling axis 2 [deg] | −0.3636 | −0.4096 | −0.2612 |
| End-point settling axis 3 [deg] | 1.1864 | 1.3385 | 0.8537 |

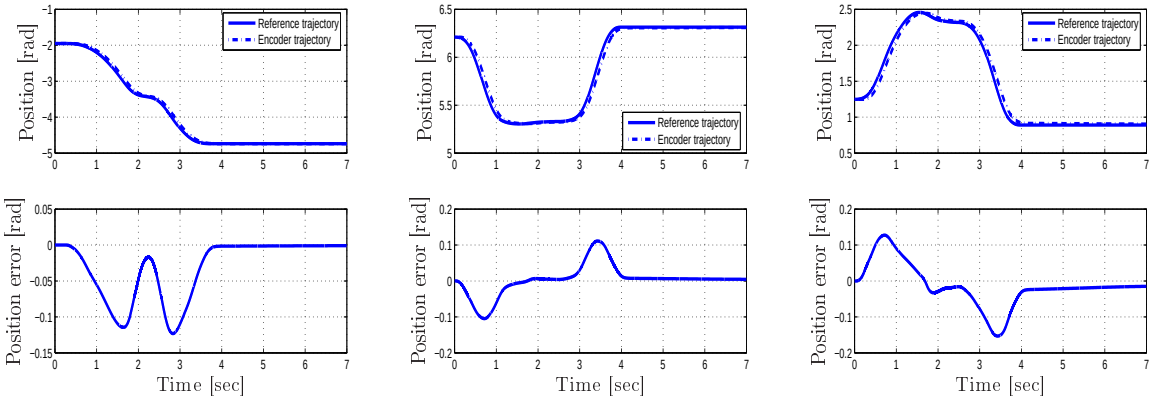
times, root mean square values of the position errors and settling error of each axis used for performance evaluation, are given in Table 6.5.

In order to evaluate the data presented in Table 6.5, the evaluation strategy presented in the previous section is applied. As discussed the method used is the multi-criteria analysis method. For the qualitative data listed in Table 6.1, the rewarded scores are listed in Table 6.6. The scores for the joint acceleration smoothness are based on the order of the profile. The higher the order of the profile, the higher the score. he linear segments with blends and the industrial algorithm both have that are second-order continuous, while the quintic algorithm has profile that are up to fourth-order continuous. The scores regarding constraint handling, it can be said that all three algorithms only account for constraints given in one domain. However, with the industrial algorithm it can not be guaranteed by forehand that no collisions occur. Usability scores are assigned regarding the user-friendliness of the algorithm. The linear segments with blends algorithm and quintic algorithm are easy in use and therefore gain the highest score. The industrial algorithm is awarded less, in comparison with the linear segments with blends algorithm and quintic al-




(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.4: Reference and simulated trajectories for pick and place case using quintic algorithm.




(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.5: Reference and simulated trajectories for pick and place case using industrial pick and place algorithm.

Table 6.6: Qualitative data scores for linear segments with blends algorithm (LSwB), quintic algorithm and industrial algorithm.

| Criterion | LSwB algorithm | Quintic algorithm | Industrial algorithm |
|---|---|---|---|
| Joint acceleration smoothness | +++ | ++++ | +++ |
| Constraint handling | +++ | +++ | ++ |
| Usability | ++++ | ++++ | +++ |

Table 6.7: Overall dominance scores for pick and place simulation case.

| | LSwB algorithm | Quintic algorithm | Industrial algorithm |
|---|---|---|---|
| LSwB algorithm | | −2.0388 | −6 |
| Quintic algorithm | 2.0388 | | −3.9612 |
| Industrial algorithm | 6 | 3.9612 | |

gorithm, due to the fact that the path motion needs to be tuned in order to make sure that no collisions occur. The performance evaluation is performed using the evaluation matrix and corresponding weighing factors presented in appendix G. The multi-criteria analysis then results in the matrix of overall dominance scores presented in Table 6.7. A negative dominance score at $ij$ in the matrix, means that algorithm $j$ perform better than algorithm $i$.

From this we can conclude that for this evaluation matrix with corresponding weighing factors the industrial pick and place algorithm performs better than the linear segments with blends algorithm and quintic algorithm. The motion time of the industrial algorithm is almost $1.5$ times lower that the motion time of the linear segments with blends and the quintic algorithm. This is mainly due to the fact that by defining linear segments with blends the high level of freedom that exists on a pick and place motion is lost. However, the industrial algorithm is applicable only for a very limited number of pick and place applications. Therefore, for more complex pick and place applications (e.g. a application where the workspace contains several obstacles) the linear segments with blends algorithm and quintic algorithm will be in favor. Moreover, this type of algorithm can be used for simple contouring tasks.

### Simulation case 2: path constrained motion

In this simulation case, trajectories are generated using the path constrained motion planning algorithm presented in Chapter 5. The path to be tracked will be a helix path defined in the workspace of the manipulator. Due to the fact that such path cannot be approximated by linear segments with blends or the quintic algorithm, this simulation will be used as proof of principle case. The performance evaluation of the path constrained motion planning algorithm will be done in the third simulation case, where a straight line will be tracked.

As discussed above, in this simulation case a helix path will be tracked. The path is given by,

$$\underline{P}(s) = \begin{bmatrix} 0.25\cos(2\pi s) \\ 0.25\sin(2\pi s) \\ 0.15 + (0.3585 - 0.15)s \end{bmatrix}, \; s \in [0,1]. \tag{6.1}$$

In Figure 6.6, the joint reference trajectories together with the simulated output trajectories are presented. Furthermore, in Figure 6.7 the corresponding paths in the Cartesian space are given. From this simulation case, it can be seen that the algorithm tracks the path accurately. The performance of the algorithm compared to other algorithms will be compared in the next section, where simulation results of the straight line tracking case will be presented.

### Simulation case 3: Straight line tracking

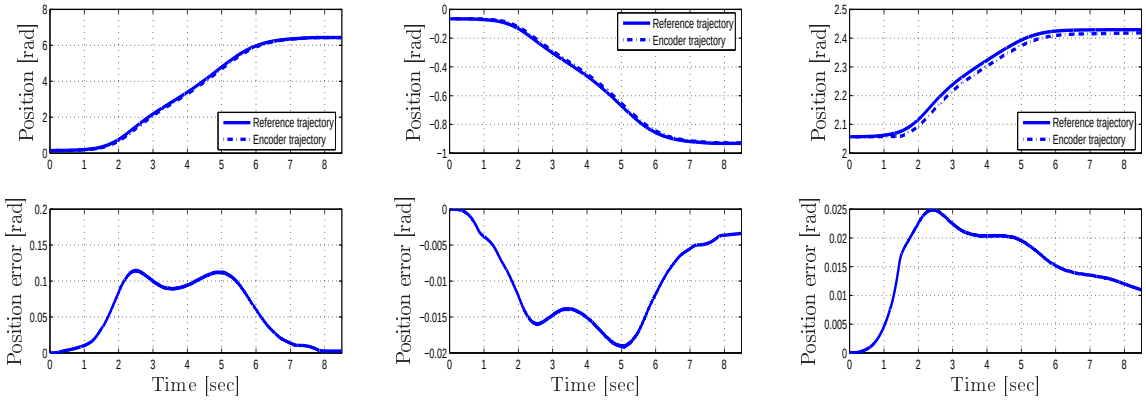
The final simulation case considers the tracking of a straight line in the workspace of the manipulator. The straight line is given by,

$$\underline{P}(s) = \begin{bmatrix} 0.4 \\ 0.3s - 0.1 \\ 0.2s + 0.3 \end{bmatrix}, \; s \in [0,1]. \tag{6.2}$$

The trajectories corresponding to this straight line are generated using the linear segments with blends algorithm, the path constrained motion algorithm and the quintic algorithm. Furthermore, the obtained results are compared to the results obtained by [Constantinescu, 2000]. In this way, insight in the performance of different trajectory generation algorithms is gained. The determined and simulated trajectories corresponding to the straight line are presented in Figures 6.8, 6.9 and 6.10. In Table 6.9 the data corresponding to this simulation case is given. Furthermore, the results for the same simulation that are obtained by [Constantinescu, 2000] are given. However, no information available on the end point settling errors is listed by [Constantinescu, 2000]. Therefore, for this case the end point settling error is omitted from this analysis.

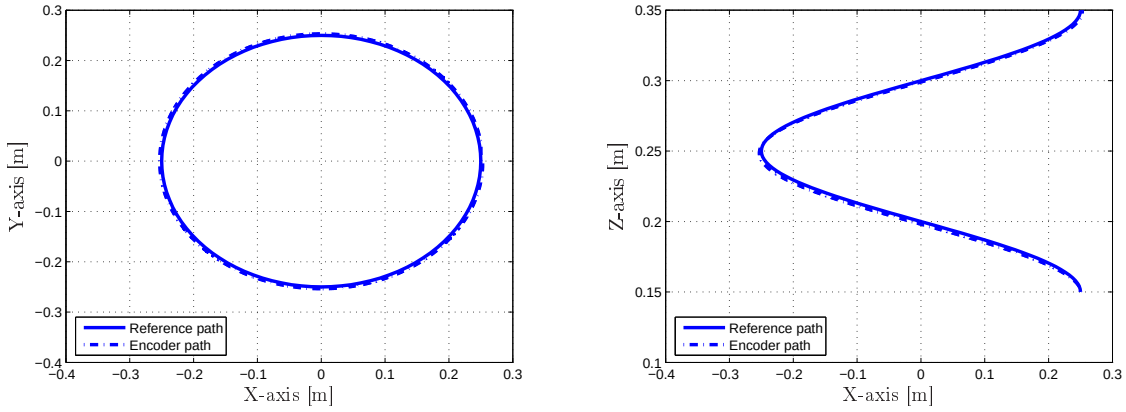
The qualitative scores for the algorithms can be found in Table 6.10. The scores for the joint acceleration smoothness are based on the order of the profile. The higher the order of the profile, the higher the score. The scores regarding constraint handling, are highest for the path constrained algorithm and the SPCTOM algorithm since these algorithms can cope with both process and actuator constraints, while the linear segments with blends and quintic algorithm only account for constraints in one domain. Usability scores are assigned regarding the user-friendliness of the algorithm. The linear segments with blends algorithm and quintic algorithm are easy in use and therefore gain the highest score. The path constrained motion algorithm get's a higher score than the SPCTOM algorithm because kinematic relations instead of dynamic equations are used for constraint evaluation. Based on the listed quantitative and qualitative data the dominance scores can be made up (see Table 6.11). From this, the overall algorithm rank can be found in Table 6.12.

From this simulation case, we can conclude that the developed algorithms perform better than the algorithms presented in literature. The path constrained motion algorithm is favorable above the linear segments with blends algorithm. This is mainly caused by the smaller tracking errors and the fact that both actuator constraints and process constraints can be taken into account.




(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.6: Reference and simulated trajectories for helix path using path constrained motion algorithm.

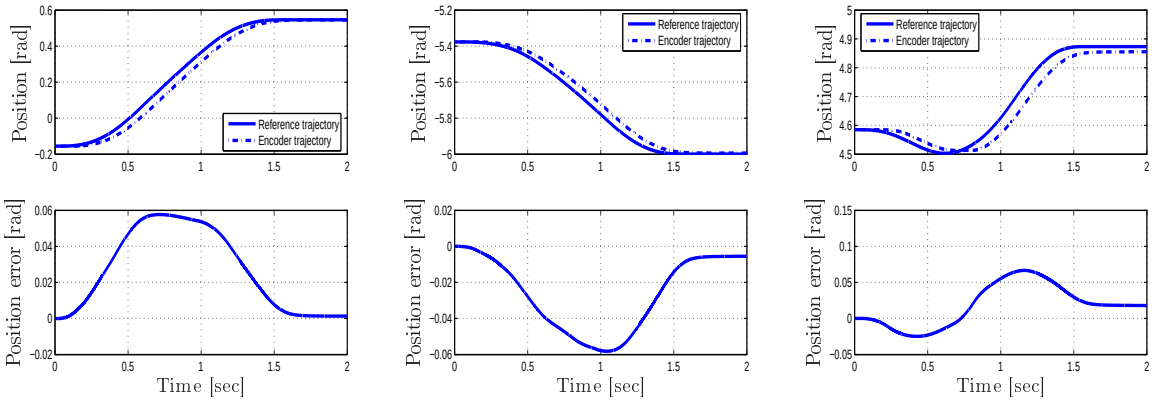



(a) X-Y plane (b) X-Z plane

Figure 6.7: Reference and simulated trajectories for helix path using path constrained motion algorithm.

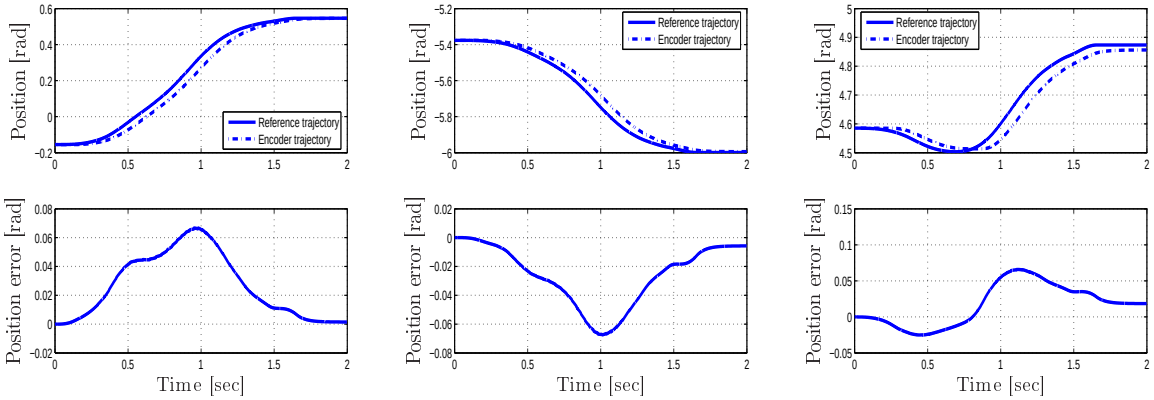
Table 6.8: Simulation data for helix path.

| | Path constrained motion algorithm |
|---|---|
| Total motion time [sec] | 7.8160 |
| Path following accuracy axis 1 [deg] | 4.0730 |
| Path following accuracy axis 2 [deg] | 0.6702 |
| Path following accuracy axis 3 [deg] | 0.9658 |
| End point settling axis 1 [deg] | −0.1286 |
| End point settling axis 2 [deg] | 0.1910 |
| End point settling axis 3 [deg] | −0.6277 |




(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.8: Reference and simulated trajectories for straight line tracking case using linear segments with blends algorithm.

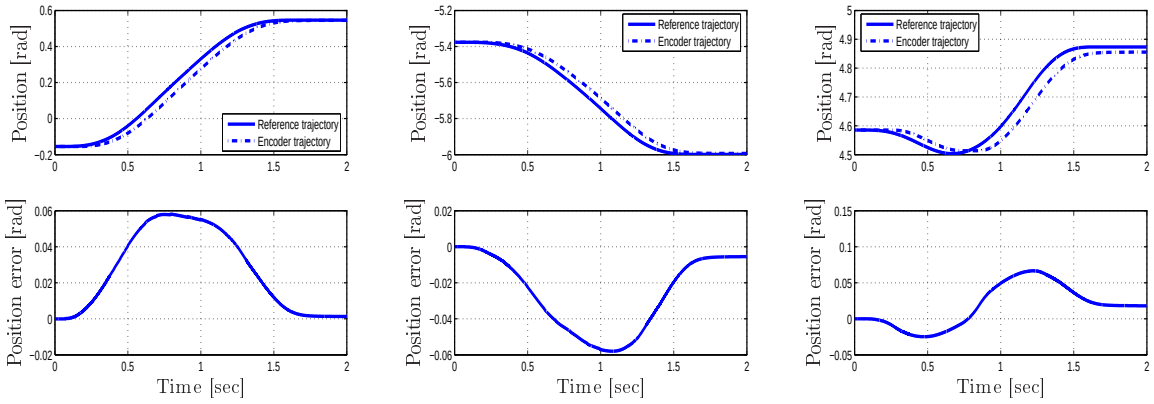



(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.9: Reference and simulated trajectories for straight line tracking case using path constrained motion algorithm.

Table 6.9: Simulation results of straight line tracking using linear segments with blends (LSwB) algorithm, path constrained motion (PCM) algorithm, quintic algorithm and the SPCTOM algorithm developed by Constantinescu.

| | LSwB | PCM | Quintic | SPCTOM |
|---|---|---|---|---|
| Total motion time [sec] | 1.5895 | 1.6555 | 1.6870 | 1.5 |
| Path following accuracy axis 1 [deg] | 2.0101 | 1.9957 | 2.0099 | 2.42 |
| Path following accuracy axis 2 [deg] | 1.8973 | 1.8838 | 1.8963 | 1.97 |
| Path following accuracy axis 3 [deg] | 1.9560 | 1.9102 | 1.9482 | 1.48 |




(a) Axis 1 (b) Axis 2 (c) Axis 3

Figure 6.10: Reference and simulated trajectories for straight line tracking case using quintic algorithm.

Table 6.10: Qualitative data scores for linear segments with blends algorithm (LSwB), quintic algorithm, path constrained motion algorithm (PCM) and the SPCTOM algorithm developed by Constantinescu.

| Criterion | LSwB algorithm | Quintic algorithm | PCM algorithm | SPCTOM |
|---|---|---|---|---|
| Joint acceleration smoothness | +++ | ++++ | +++++ | +++++ |
| Constraint handling | +++ | +++ | +++++ | +++++ |
| Usability | ++++ | ++++ | +++ | ++ |

Table 6.11: Overall dominance scores for straight line case.

| | LSwB | PCM | Quintic | SPCTOM |
|---|---|---|---|---|
| LSwB algorithm | | −0.5795 | 0.3809 | 0.1026 |
| PCM algorithm | 0.5795 | | 1.8498 | 1.4321 |
| Quintic algorithm | −0.3809 | −1.8498 | | −1.1677 |
| SPCTOM algorithm | −0.1026 | −1.4321 | 1.1677 | |

Table 6.12: Overall rank score for straight line simulation case.

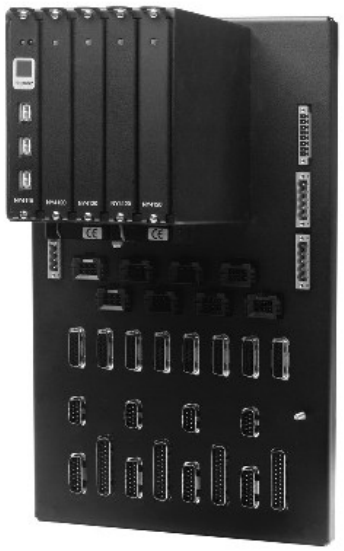
| LSwB algorithm | PCM algorithm | Quintic algorithm | SPCTOM algorithm |
|---|---|---|---|
| 2 | 1 | 4 | 3 |

## 6.4 Experiments

This section discusses the results of experiments performed using the earlier presented algorithms. The experimental results will be compared with existing algorithms, as done in the previous section. Due to the fact that useful mechanics are not available, the experiments will mainly provide a proof of principle of the algorithms.

The experiments are performed on a robotic system consisting of two stacked independently driven linear axes. The system is controlled using a NYCe4000 industrial motion control system. An overview of the setup is presented in Figure 6.11. A schematic representation can be found in Figure 6.12. Each axis is controlled by a tuned discrete PID-controller. Furthermore, friction feedforward compensation is applied. The position feedback loop has a sample frequency of $8\ kHz$. The setpoint generator generates reference values at $2\ kHz$. The position of each axis is measured separately using an incremental encoder with a resolution of $87\ \frac{increments}{mm}$, yielding a resolution of $1.15 \cdot 10^{-2}\ \frac{mm}{increment}$. The constraints on velocity, acceleration and jerk are determined by performing experiments, while regarding the saturation level of the drive current and the dynamical behavior of the robotic system. The obtained constraints can be found in Table 6.13. While the mechanics of the robotic system are very straightforward, the experiments are very useful due to the fact that the robotic system is controlled using an industrial motion control system.

(a) Robotic system consisting of two independent driven axes.



(b) NYCe4000 industrial motion control system

Figure 6.11: Overview of setup used during experiments.

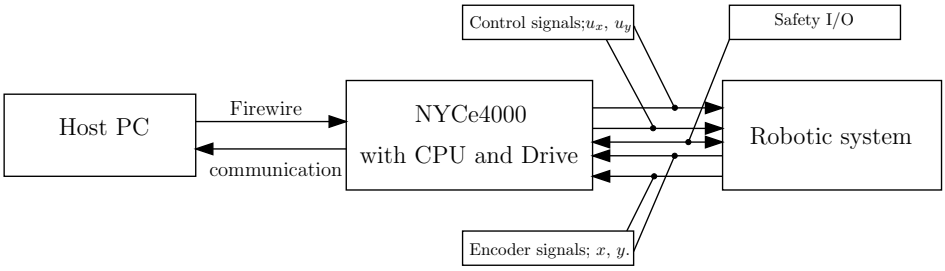



Figure 6.12: Schematic representation of experimental setup.

Table 6.13: Constraints on velocity, acceleration and jerk for the experimental setup.

| Constraint | X-axis | Y-axis |
|---|---|---|
| Velocity $[\frac{mm}{s}]$ | 450 | 450 |
| Acceleration $[\frac{mm}{s^2}]$ | 2500 | 2500 |
| Jerk $[\frac{mm}{s^3}]$ | 25000 | 25000 |

In total three experiments are performed. First a straight line will be tracked. In this case both the developed algorithms will be used to generate trajectories. Furthermore, the results will be compared with an existing algorithm. The second experiment is a case which is typical a case for the linear segments with blends algorithm, while the final experiment is a case which can be tackled using the path constrained motion planning algorithm presented in Chapter 5.

### 6.4.1 Results

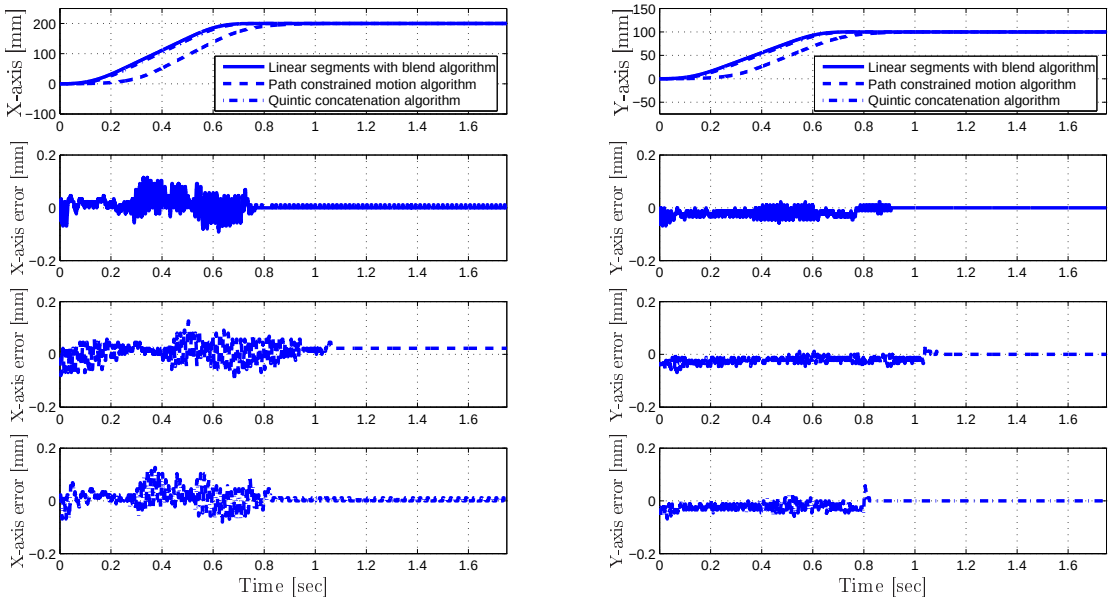
In this section, the results obtained by performing experiments using the experimental setup presented above.

#### Experiment 1: Straight line tracking

As discussed above, the first experiment represents the tracking of a straight line between two points, i.e:

$$\underline{p}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \underline{p}_e = \begin{bmatrix} 200 \\ 100 \end{bmatrix}, \text{in mm} \tag{6.3}$$

The trajectories corresponding to the straight line are determined using the algorithms presented in Chapter 4 and 5. The obtained results are evaluated and compared with results obtained with the algorithm developed by [Macfarlane, 2003]. The latter algorithm is able to generate linear segments using a concatenation of fifth order splines. For the path constrained motion planning algorithm of Chapter 5 the path must be provided in total (and not by only start and end points).




(a) (b)

Figure 6.13: Experimental results of straight line tracking using linear segments with blends algorithm, path constrained motion algorithm and quintic algorithm.

The path between start and end points (see (6.3)) is defined as,

$$\underline{P}(s) = \begin{bmatrix} 200 \\ 100 \end{bmatrix} s, \; s \in [0, 1], \tag{6.4}$$

where $s$ represents the path parameter. The results are presented in Figure 6.13 and Table 6.14.

Table 6.14: Experimental results of straight line tracking using linear segments with blends (LSwB) algorithm, path constrained motion (PCM) algorithm and quintic algorithm.

| | LSwB algorithm | PCM algorithm | Quintic algorithm |
|---|---|---|---|
| Total motion time [sec] | 0.7635 | 1.0480 | 0.8110 |
| Path following accuracy X-axis [mm] | $4.00 \cdot 10^{-2}$ | $3.71 \cdot 10^{-2}$ | $4.02 \cdot 10^{-2}$ |
| Path following accuracy Y-axis [mm] | $2.62 \cdot 10^{-2}$ | $2.54 \cdot 10^{-2}$ | $2.68 \cdot 10^{-2}$ |
| End-point settling X-axis [mm] | $1.15 \cdot 10^{-2}$ | $2.30 \cdot 10^{-2}$ | $1.15 \cdot 10^{-2}$ |
| End-point settling Y-axis [mm] | 0.00 | 0.00 | 0.00 |

From the results of Table 6.14, we can conclude that the linear segments with blends and quintic algorithm generate motions that result in a much lower motion time than the path constrained motion algorithm. When considering the path following accuracy, we can see that for the linear

segments with blends and the quintic algorithm result in similar accuracy values, while the path constrained motion algorithm result in better tracking accuracies. However, for the X-axis the end-point settling the path constrained motion has a greater steady-state error. The reason for this can be found when considering the mechanics of the experimental setup. The X-axis of the setup suffers from a certain amount of backlash, while for the Y-axis the backlash is minimized by a backlash free gearbox. Therefore, the end-point settling for the Y-axis lies between one encoder increment (since one encoder increment is equal to $1.15 \cdot 10^{-2}\ mm$). Based on the evaluation matrix for the straight line case by using the scores for the qualitative criteria as already presented in Table 6.10 and the weighing scores from Section G.3, the overall dominance scores are determined by applying the multi-criteria analysis. The overall dominance scores for the straight line case can be found in Table 6.15.

Table 6.15: Overall dominance scores for straight line case.

| | LSwB algorithm | PCM algorithm | Quintic algorithm |
|---|---|---|---|
| LSwB algorithm | | $-0.2619$ | 0.2365 |
| PCM algorithm | 0.2619 | | 2.0293 |
| Quintic algorithm | $-0.2365$ | $-2.0293$ | |

Based on the overall scores, the overall rank for the first set of experiments can be made up and can be found in Table 6.16.

Table 6.16: Overall rank score for straight line case.

| LSwB algorithm | PCM algorithm | Quintic algorithm |
|---|---|---|
| 2 | 1 | 3 |

From Table 6.16, it can be seen that the same conclusions can be drawn as for the straight line simulation case. The path constrained motion algorithm performs better than the linear segments with blens and the quintic algorithm. This is mainly due to smaller tracking errors and the fact that both actuator constraints and process constraints can be taken into account.

### Experiment 2: Multi Waypoint experiment

The second set of experiments performed is a case of typical linear segments and blends. Since the robotic system used for the experiments has only two actuated axes, no typical pick and place task can be performed. Therefore, here a contouring problem consisting of multiple waypoints is considered. The set of points with corresponding blending radii are given in Table 6.17. In Figure 6.14 an overview of the points connected by linear segments and corresponding blend sphere is presented.

The trajectories corresponding to this path are generated using the linear segments with blends algorithm and the quintic algorithm. The measured axis positions corresponding to the generated trajectories are given in Figure 6.15.

Table 6.17: Multi waypoint data.

| X-coordinate [mm] | Y-coordinate [mm] | Blend radius [mm] |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 300 | 25 |
| 250 | 300 | 25 |
| 300 | 150 | 25 |
| 250 | 0 | 25 |
| 0 | 0 | 0 |

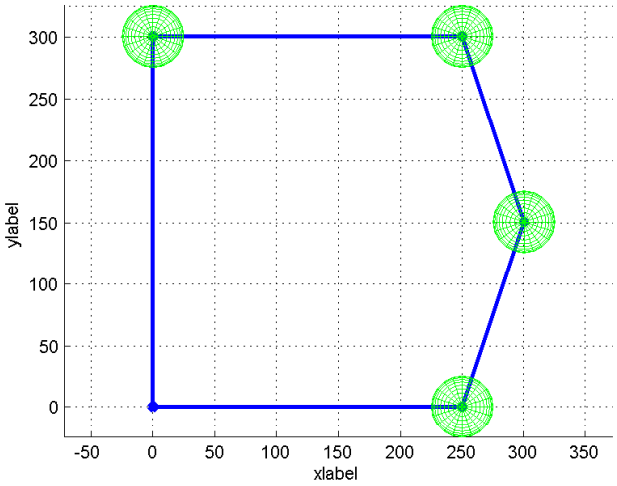


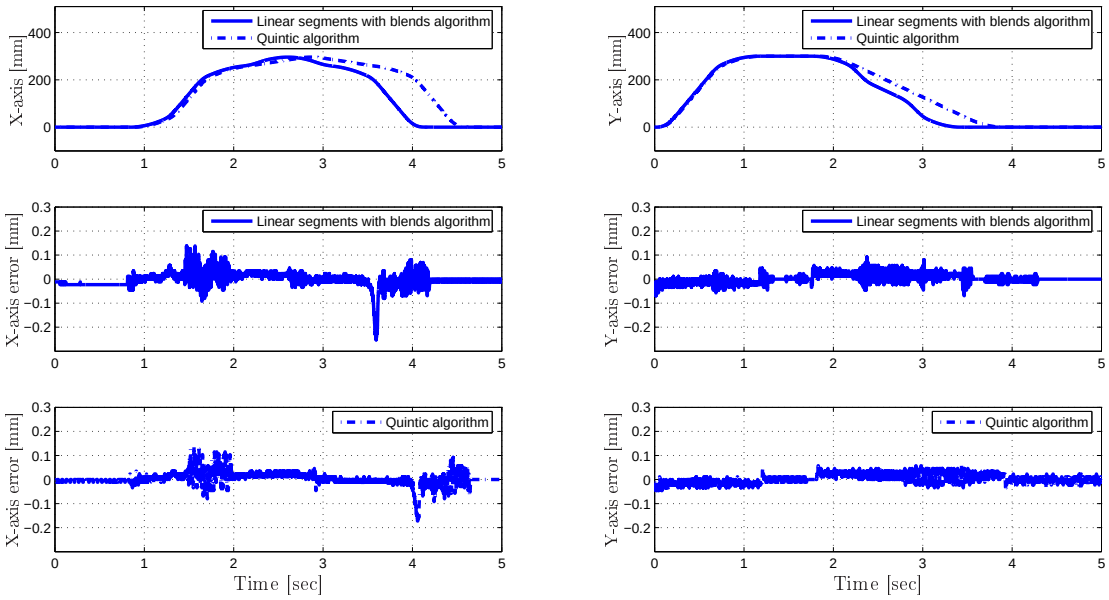

Figure 6.14: Multi waypoint overview in the workspace of the manipulator.

In Table 6.18, the data corresponding to the figures can be found. Instead of evaluating the data by using the evaluation matrix for pick and place testcases, here the evaluation matrix for path constrained motions is applied. This approach is applied because here the algorithms are used for a contouring problem. The dominance scores (see Table 6.19) obtained by performing the multi-criteria analysis with the qualitative and quantitative data from Table 6.10 and 6.18, respectively. From this we can conclude that in this case the quintic algorithm performs better than the linear segments with blends algorithm. This is mainly due to the fact that a fifth order polynomial is used to describe the linear segments. Although this results in a larger motion time, it reduces wear of the actuator due to a smoother acceleration profile.

### Experiment 3: Spiral path tracking

For the final experiment a spiral formed path is tracked. The path is defined by,

$$\underline{P}(s) = \begin{bmatrix} rs\cos(4\pi s) \\ rs\sin(4\pi s) \end{bmatrix}, \; r = 50\; mm, \; s \in [0,1]. \tag{6.5}$$




(a) (b)

Figure 6.15: Experimental results of multi waypoint experiment using linear segments with blends algorithm and quintic algorithm.

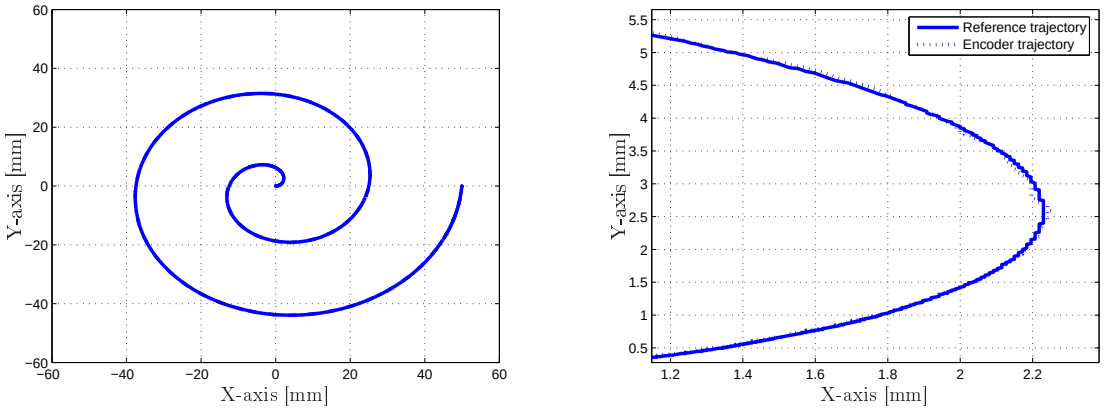
Table 6.18: Experimental results of multi waypoint experiment using linear segments with blends (LSwB) algorithm and quintic algorithm.

| | LSwB algorithm | Quintic algorithm |
|---|---|---|
| Total motion time [sec] | 4.1460 | 4.6095 |
| Path following accuracy X-axis [mm] | $2.71 \cdot 10^{-2}$ | $2.32 \cdot 10^{-2}$ |
| Path following accuracy Y-axis [mm] | $1.53 \cdot 10^{-2}$ | $1.69 \cdot 10^{-2}$ |
| End-point settling X-axis [mm] | $-1.15 \cdot 10^{-2}$ | $1.15 \cdot 10^{-2}$ |
| End-point settling Y-axis [mm] | $1.15 \cdot 10^{-2}$ | $1.15 \cdot 10^{-2}$ |

Table 6.19: Overall dominance scores for multi waypoint experiment.

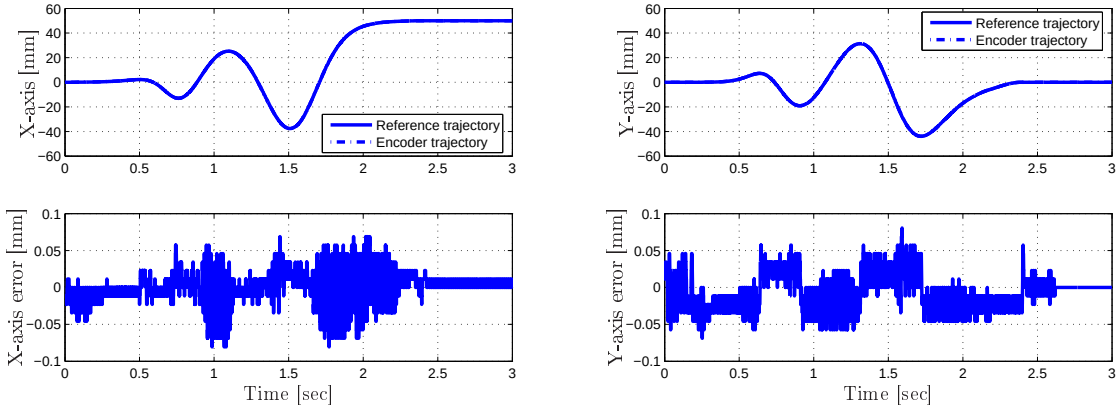
| | LSwB algorithm | Quintic algorithm |
|---|---|---|
| LSwB algorithm | | $-30.5$ |
| Quintic algorithm | $-30.5$ | |

The trajectories for this problem are determined using the path constrained motion algorithm. Since this path cannot be approximated by linear segments with blends or the quintic algorithm this case serves as a proof of principle case. It demonstrates the working of the path constrained motion algorithm. The experimental results corresponding to this case can be found in Figures 6.16 and 6.17 and Table 6.20.




(a) Total spiral path. (b) Zoom of the spiral path.

Figure 6.16: Path profile of spiral tracking using path constrained motion algorithm.




(a) (b)

Figure 6.17: Setpoint and trajectories profiles and corresponding position errors of spiral tracking using path constrained motion algorithm.

Since only the path constrained motion planning is considered in this experiment, no further analysis is performed. As can be seen from the results the algorithm tracks the prescribed path accurately. The performance of the algorithm has already been proven in the straight line experiments.

Table 6.20: Experimental results of spiral tracking using path constrained motion algorithm.

| | PCM algorithm |
|---|---|
| Total motion time [sec] | 2.3860 |
| Path following accuracy X-axis [mm] | $1.46 \cdot 10^{-2}$ |
| Path following accuracy Y-axis [mm] | $1.56 \cdot 10^{-2}$ |
| End point settling accuracy X-axis [mm] | $1.15 \cdot 10^{-2}$ |
| End point settling accuracy Y-axis [mm] | 0.00 |

## 6.5 Discussion

In this chapter, the results obtained by performing simulations and experiments are discussed. In total, three simulations and three experiments are discussed. In order to evaluate the performance of the developed algorithms, a performance evaluation approach is presented. The analysis of the performance is based on the multi-criteria analysis.

In total three simulations are performed. The first simulation resembles a pick and place application. From the results we can conclude that for this simulation an industrial algorithm under consideration results in motions with smaller motion times almost 1.5 times smaller than the linear segments with blends algorithm and quintic algorithm. For pick and place applications the total motion time is the most important criterion, as can be seen from the weighing factor for total motion time in the performance evaluation matrix. However, the industrial algorithm is applicable for only a very limited number of pick and place applications. The second simulation case resembles a path constrained motion case, that serves as a proof of principle case. The reason for this is that the tracked path cannot be approximated by the linear segments with blends algorithm or the quintic algorithm. The performance of the path constrained motion algorithm and linear segments with blends algorithm is evaluated in the third simulation case, in which a straight line is tracked. The obtained results are compared with two algorithms from literature. From the outcome of the multi-criteria analysis, it can be seen that, based on the presented evaluation matrix, the developed algorithms perform better than the algorithms presented in literature.

After discussing the simulation results, the experimental results are presented. The performed experiments are used for the same comparison as done by performing simulations. However, the experiments are performed using a simple mechanical structure, namely a robotic system that consist of two stacked axes. The obtained results are compared using the multi-criteria analysis with the defined evaluation matrices. Based on the outcome of the multi-criteria analysis, it can be said that the presented trajectory generation algorithms (i.e. linear segments with blends and path constrained motion algorithm) are favorable above algorithms presented in literature. However, the choice between the algorithms depends on the application for which it will be used.

As discussed above, the performance evaluation is carried out using multi-criteria analysis. The outcome of the analysis depends on the assigned weighing factors. Furthermore, the total sum of the qualitative weighing factors and quantitative weighing factor, respectively indicates the importance of these type of criteria. In order to obtain more information on the influence of the weighing factors on the final score, a sensibility analysis can be performed.

# Chapter 7

# Conclusions and Recommendations

In order to shorten the design-in of motion control platforms into industrial applications, research is performed to investigate the development of trajectory generation algorithms that are applicable for a wide range of applications. Overall we can conclude that research in the field of generic motion planning is not evidently. Furthermore, a generic solution cannot be obtained by one single algorithm that solves the problem regarding the complete set of requirements. In this chapter conclusions (Section 7.1) and recommendations for future research (Section 7.2) will be presented.

## 7.1 Conclusions

From literature, it can be concluded that basically two motion planning strategies can be distinguished, namely pick-and-place motions and path constrained motions. A pick-and-place motion represents a movement of a robotic system for which a high level of freedom on the path between pick and place locations exists, while during a path constrained motion it is desired for the end-effector of a manipulator to exactly follow a certain path. Hereby, the user of the motion planner is seen as the high-level path planner (providing both pick and place positions and possible intermediate positions to avoid collisions).

In order to gain insight in what conditions must be fulfilled such that generic motions can be generated, a study on the requirements for generic motion planning is performed. The results are based on interviewing engineers with a high-level experience in the field of industrial motion control. Based on the results of the requirements study and the results already presented in literature, a structure for a generic motion planner is proposed. One of the main advantages of the proposed structure is its practical applicability. This practical applicability is mainly achieved by making sure that no a priori knowledge about the dynamics of the manipulator is needed. Furthermore, the transformation from paths in the workspace of the manipulator to trajectories for each joint is achieved by an numerical inverse kinematics algorithm. The algorithm is applicable for both redundant and non-redundant manipulators.

Trajectory generation algorithms are developed that are able to deal with pick and place applications and path constrained motion applications.
In case of pick and place applications a trajectory generation algorithm is presented, which is capable of calculating linear segments with blends. Linear segment with blends trajectories are

commonly used by industrial manipulators and machine tool trajectories for their fast motions and low computational complexity. The theory of describing linear segments with blends is discussed. The linear segments are described using classical point-to-point theory for determining time-optimal third-order point-to-points. A blend function is introduced which describes the transition between two linear segments. The shape of the transition can easily be controlled by setting two parameters. The structure of the developed algorithm is presented. The algorithm mainly consists of three parts. The first part of the algorithm checks whether the user-defined points lie on each other or whether the blend spheres of two successive points intersect. The second and third part determine the parameters of the point-to-points and blends, respectively. The developed algorithm can easily be extended for determining linear segments with blends in joint space. Furthermore, the algorithm is also applicable for simple contouring problems (i.e. where paths consists of linear segments).

For cases in which a manipulator must perform a path that does not consist or can be approximated by linear segments, another algorithm is developed. The algorithm is able to account for both actuator constraints and process constraints. By transforming the constraints to one domain, a time-optimal control problem can be formulated. However, no analytic solution for this kind of time-optimal control problem can be found so far. Therefore, a different (numerical) approach is followed to solve the optimization problem. The resulting algorithm consists of a combination of a genetic algorithm and the Nelder-Mead method. In this way a global optimum, for a path constrained motion with a minimum time criterion, can be found.

The performance of the algorithms has been evaluated by means of simulations and experiments. A performance evaluation strategy has been developed, since no clear benchmark was available in literature for testing and comparing trajectory generation algorithms. The evaluation strategy is based on multi-criteria analysis which uses an evaluation matrix that may consist of both qualitative and quantitative evaluation criteria. Furthermore, to each criterion a weighing factor can be assigned to indicate the importance of the criterion in a certain testcase.
Based on the outcome of the multi-criteria analysis for both simulations and experiments it can be concluded that the presented trajectory generation algorithms (i.e. linear segments with blends and path constrained motion algorithm) are favorable above algorithms presented in literature. However, when performance is evaluated compared to an industrial algorithm developed for a dedicated application (as is done in this work for a pick and place application), it can be seen that the dedicated algorithm performs better than the developed pick and place algorithm. This due to the fact that by defining linear segments with blends the high level of freedom that exists on a pick and place motion is lost. Where the industrial algorithm is only applicable for a very limited number of applications, the developed algorithms are applicable for more generic applications. Moreover, the choice between the algorithms depends on the application for which it will be used.

## 7.2 Recommendations for future research

In this section, some recommendations for future research is proposed. First, some recommendations regarding the improvement of the developed algorithms will be discussed. Next, some other recommendations are proposed.

As already discussed in the previous section, the developed pick and place algorithm does not

account for the high level of freedom that exists on a pick and place path. So this limits the entire motion. Therefore, it is recommended to extend the developed algorithm such that the pick and place algorithm is able to deal with a high level of path freedom.

Regarding the path constrained motion algorithm, the recommendations include:

- Reduction of the computational time, in order to increase the industrial applicability of the algorithm,
- Investigate the performance of Genetic algorithm compared to other global optimization strategies, such as simulated annealing,
- The application of other optimization criteria (e.g. energy optimality),
- Investigation of the influence of the assumption that the joint jerk constraints do not influence the maximum pseudo-velocity $\dot{s}$ and pseudo-acceleration $\ddot{s}$.

An overall recommendation is research in the development of a high-level motion planner. A high-level motion planner should plan motions for a certain application without user intervention, such that:

- no collisions occur between the manipulator and objects in the workspace of the manipulator,
- at certain points and/or segments along a path a certain constant velocity is reached (this is an important issue in laser cutting or welding applications),
- trajectory calibration is performed,
- I/O actions can be handled.

This high-level motion planner can be inspired by the motion planning structure presented in Section 3.3. The algorithms developed in this thesis can serve as a basis for the high-level motion planner. So, the high-level motion planner is a tool that should call the developed algorithms such that the desired motions are obtained.

In this work, the developed algorithms are evaluated by simulations and experiments. The experiments are performed using a simple mechanical structure. However, for further experimental performance evaluation it is recommended to perform experiments on industrial applications with more challenging mechanics, such as a robotic system of double-scara type. Furthermore, it can be interesting to apply the developed algorithms on applications other than industrial manipulators. For example, a mobile robotic system, with non-holonomic constraints.
The performance analysis is evaluated using multi-criteria analysis. The outcome of the analysis depends on the assigned weighing factors. Furthermore, the total sum of the qualitative weighing factors and quantitative weighing factor, respectively indicates the importance of these type of criteria. In order to obtain more information on the influence of the weighing factors on the final score, it is recommended to perform a sensibility analysis.

A final recommendation is to gain more insight in solving the time optimal control problem for path constrained motions. As already stated in Chapter 5, until now no analytical solution is found for the time optimal control problem for this specific problem with bounded input and states.

# Appendix A

# List of people interviewed and questions for requirements study

## A.1 Interviewed people

**Bosch Rexroth Electric drives and control**

- Wim de Graaff (Senior motion engineer)
- Menno Haring (Motion engineer)
- Wilco Pancras (Motion engineer)
- Huub Uijen (Motion engineer)

**FEI Company**

- Andrea Pasqualini (Optics)
- Frank Roes (Project leader)
- Edwin Verschuren (Motion engineer)

## A.2 Questions

- What applications must be served by the motion planner?
- What are typical problems when generating motions?
- What are the shortcomings of the present motion planner?
- What issues must be served by a new motion planner?
  - constraint handling,
  - on-line versus off-line trajectory generations,
  - collision avoidance,
  - trajectory optimization,
  - profile order,
  - input / output of trajectory generator.

# Appendix B

# Pseudo-inverse of a matrix by using singular value decomposition

This appendix discusses the determination of the inverse of a matrix $\underline{A}$ using singular value decomposition. The usage of singular value decomposition can be very useful when the matrix is singular or when $\underline{A}$ is not a square matrix. Singular value decomposition is a way of representing a matrix (other ways of representing a matrix can be e.g. eigenvalue decomposition, LU decomposition, Cholesky decomposition, etc.). In case of singular value decomposition of a $m \times n$ matrix $\underline{A}$, the matrix is represented using three matrices,

$$\underline{A} = \underline{U}\,\underline{\Sigma}\,\underline{V}^T. \tag{B.1}$$

where $\underline{U}$ is an orthogonal $m \times m$ matrix, $\underline{V}$ is an orthogonal $n \times n$ matrix and $\underline{\Sigma}$ a $m \times n$ diagonal matrix, with

$$\sigma_{ij} = \begin{cases} 0 & \text{for } i \neq j \\ \sigma_i \geq 0 & \text{for } i = j \end{cases} \tag{B.2}$$

The diagonal entries $\sigma_i$ are called the singular values of matrix $A$, and are usually ordered so that $\sigma_i \geq \sigma i + 1$, for $i = 1, \ldots, n-1$. Since $\underline{U}$ and $\underline{V}$ are orthogonal the following properties hold:

1. $\underline{U}\,\underline{U}^T = \underline{U}^T\underline{U} = \underline{I}$, $\underline{V}\,\underline{V}^T = \underline{V}^T\underline{V} = \underline{I}$,
2. The vectors in $\underline{U}$ and $\underline{V}$ have length one and are perpendicular to each other (This could also be seen from the property above).

In [Heath, 1997] a pseudo-inverse of a scalar $\gamma$ is defined as $\frac{1}{\gamma}$ if $\gamma \neq 0$, and zero otherwise. The pseudo-inverse is now defined by transposing the matix and taking the pseudo-inverse of each entry $\sigma_{ij}$, for $i = 1, \ldots, m$, $j = 1, \ldots, n$. Using singular value decomposition this implies that the pseudo-inverse is determined by transposing the matrix multiplication $\underline{U}\,\underline{\Sigma}\,\underline{V}^T$ and by taking the pseudo-inverse of the entries of $\underline{\Sigma}$. So, the pseudo-inverse is determined by

$$\underline{A}^\dagger = \underline{V}\,\underline{\Sigma}^\dagger\,\underline{U}^T. \tag{B.3}$$

From the definition above it can be seen that the pseudo-inverse always exists despite the fact that a matrix can be non-square or does not have full rank. When matrix $\underline{A}$ is square and has full rank, the pseudo-inverse is the same as the usual matrix inverse, $\underline{A}^{-1}$.

# Appendix C

# Numerical example of inverse kinematics algorithm

The goal of this appendix is to provide a numerical example of the inverse kinematics algorithm presented in Section 3.3.2 such that no deadlocks in singular positions occur. For this example, we use a two-link manipulator as depicted in Figure C.1. We assume that the manipulator links have the same length, namely $L_1 = L_2 = L$. In this example, we will determine the joint variables ($q_1$

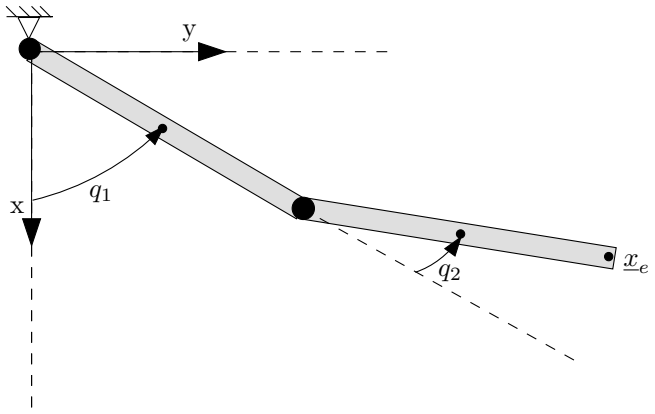



Figure C.1: A two-link manipulator.

and $q_2$) for a given path depending on the operational space variables ($x$ and $y$). As can be seen in Figure 3.3 in Section 3.3.2, the forward kinematics need to be supplied to the algorithm. Here, the forward kinematics are defined as the relation between the joint variables ($\theta_1$ and $\theta_2$) and the position at the end of link 2 (i.e. $\underline{x}_e$). A full derivation of the kinematic relations for a two-link manipulator can be found in [van de Wouw, 2001]. So, the relation between joint variables and the position at the end of link 2 can be written as,

$$\underline{x}_e = \underline{R}(\underline{q}), \tag{C.1}$$

with,

$$\underline{R}(\underline{q}) = \begin{bmatrix} L\cos(q_1) + L\cos(q_1 + q_2) \\ L\sin(q_1) + L\sin(q_1 + q_2) \end{bmatrix} \tag{C.2}$$

Here, a link-length of $L = 1$ is used. Furthermore, an estimation accuracy of $\epsilon = 10^{-4}$ is required. The joint variables are determined for the path given in Figure C.2. The initial estimation provided to the algorithm is set to $\underline{q}_0 = [1\ 1]^T$.

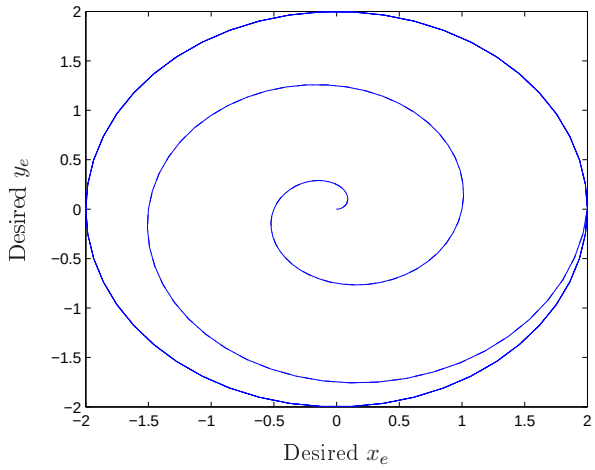



Figure C.2: Path for inverse kinematics algorithm example.

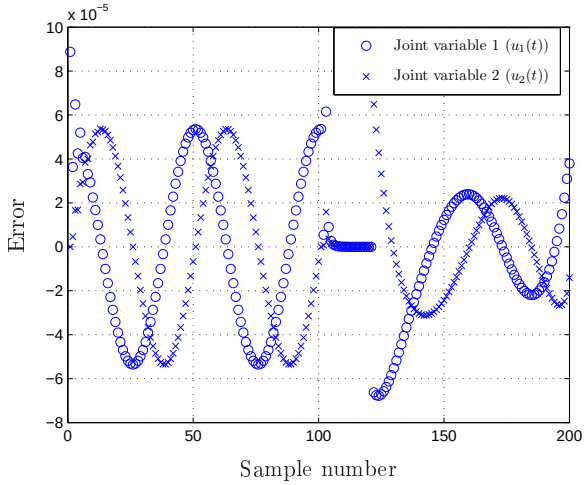



Figure C.3: Difference between desired and calculated path.

The performance of the algorithm is evaluated by inserting the output of the algorithm into (C.1) and comparing the outcome with the specified path of Figure C.2. In Figure C.3, the difference between the desired path and the output of the algorithm inserted in the forward kinematics can be found. From Figure C.3, we can conclude that the error stays between the specified estimation accuracy. Furthermore, we can see that no deadlocks occur, while the manipulator moves from one singular position(two links on top of each other) to another(links are fully-stretched).

# Appendix D

# Derivation of direction cosine matrix using Roll-Pitch-Yaw angles

This appendix will discuss the derivation of the direction cosine matrix for using Roll-Pitch-Yaw angles. Roll-Pitch-Yaw angles are often used in automotive and/or (aero)nautical field and describe rotations about the axes of a (body-)fixed frame. The orientation of a body-fixed frame $\underline{\vec{e}}^{\,3}$ with respect to the fixed frame $\underline{\vec{e}}^{\,0}$ by using RPY-angles can be described as follows:

> Start with the coinciding frames $\underline{\vec{e}}^{\,0}$ and $\underline{\vec{e}}^{\,3}$. First rotate $\underline{\vec{e}}^{\,3}$ about $\vec{e}_x^{\,0}$ by an angle $\alpha$, then rotate about $\vec{e}_y^{\,0}$ by an angle $\beta$ and finally rotate about $\vec{e}_z^{\,0}$ by an angle $\gamma$.

We denote the rotation about axis $\vec{e}_x^{\,0}$ with angle $\alpha$ as the *roll* angle, the rotation about $\vec{e}_y^{\,0}$ with angle $\beta$ as the *pitch* angle and, finally, the rotation about $\vec{e}_z^{\,0}$ with angle $\gamma$ as the *yaw* angle,

$$\begin{aligned}
&\underline{\vec{e}}^{\,0} \xrightarrow[\vec{e}_x^{\,0}]{\alpha} \underline{\vec{e}}^{\,1}, \\
&\underline{\vec{e}}^{\,1} \xrightarrow[\vec{e}_y^{\,0}]{\beta} \underline{\vec{e}}^{\,2}, \\
&\underline{\vec{e}}^{\,2} \xrightarrow[\vec{e}_z^{\,0}]{\gamma} \underline{\vec{e}}^{\,3}.
\end{aligned} \tag{D.1}$$

In literature, the direction cosine matrix is derived by stating that RPY rotations about a fixed frame give the same orientation as ZYX rotations about a moving frame, [Bruyninckx, 2004; Craig, 1986; Sciavicco, 1996]. This then results in the following direction cosine matrix,

$$\begin{aligned}
\underline{A}^{30} &= \underline{A}^{32}\underline{A}^{21}\underline{A}^{10} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & s_\alpha \\ 0 & -s_\alpha & c_\alpha \end{bmatrix} \begin{bmatrix} c_\beta & 0 & -s_\beta \\ 0 & 1 & 0 \\ s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} c_\gamma & s_\gamma & 0 \\ -s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_\gamma c_\beta & s_\gamma c_\beta & -s_\beta \\ c_\gamma s_\beta s_\alpha - s_\gamma c_\alpha & s_\gamma s_\beta s_\alpha + c_\gamma c_\alpha & c_\beta s_\alpha \\ c_\gamma s_\beta c_\alpha + s_\gamma s_\alpha & s_\gamma s_\beta c_\alpha - c_\gamma s_\alpha & c_\beta c_\alpha \end{bmatrix},
\end{aligned} \tag{D.2}$$

where $c_\alpha$ denotes $\cos\alpha$, and $s_\alpha$ denotes $\sin\alpha$, etc. Here, we will show that RPY rotations about the axes of a fixed frame indeed exactly leads to the direction cosine matrix in (D.2). The corresponding derivation will be performed below.

We will derive the direction cosine matrix presented above, by using quaternions (or Euler parameter) theory. The Euler theorem for describing orientations by Euler parameters is stated as [van de Wouw, 2001]:

**Theorem D.1** *Two arbitrarily oriented frames $\underline{\vec{e}}^{\,r}$ and $\underline{\vec{e}}^{\,s}$ with common origin $O$ can be made to coincide with one another by rotating one of them through a certain angle $\chi$ about an axis which is passing through $O$ and which has the direction of the eigenvector $\vec{n}$ determined by the eigenvalue problem (with eigenvalue 1)*

$$\underline{A}^{sr}\underline{n}^{r} = \underline{n}^{r} \; with \; \vec{n} = \underline{n}^{rT}\underline{\vec{e}}^{\,r}, \tag{D.3}$$

*with $\underline{A}^{sr}$ the direction cosine matrix describing the rotation from $\underline{\vec{e}}^{\,r}$ to $\underline{\vec{e}}^{\,s}$: $\underline{\vec{e}}^{\,s} = \underline{A}^{sr}\underline{\vec{e}}^{\,r}$.*

The direction cosine matrix $\underline{A}^{sr}$ is determined by four parameters $q_0$, $q_1$, $q_2$ and $q_3$, defined by;

$$q_0 = \cos\frac{\chi}{2}, \tag{D.4}$$

and,

$$\vec{q} = \vec{n}\sin\frac{\chi}{2} = [q_1 \; q_2 \; q_3]^T\underline{\vec{e}}, \tag{D.5}$$

according to

$$\underline{A}^{sr} = \begin{bmatrix} 2(q_0^2+q_1^2)-1 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 2(q_0^2+q_2^2)-1 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 2(q_0^2+q_3^2)-1 \end{bmatrix}. \tag{D.6}$$

The first direction cosine matrix, for rotating about $\vec{e}_x^{\,0}$, can be obtained by taking

$$\begin{aligned} &q_0 = \cos(\tfrac{\alpha}{2}) \\ &\vec{q} = \vec{n}\sin(\tfrac{\alpha}{2}) = [1\;0\;0]\sin(\tfrac{\alpha}{2})\underline{\vec{e}}^{\,0} = \sin(\tfrac{\alpha}{2})\underline{e}_x^{\,0} \end{aligned} \tag{D.7}$$

This then results in:

$$\begin{aligned} \underline{A}^{10} &= \begin{bmatrix} 2(c^2(\frac{\alpha}{2})+s^2(\frac{\alpha}{2}))-1 & 0 & 0 \\ 0 & 2c^2(\frac{\alpha}{2})-1 & 2c(\frac{\alpha}{2})s(\frac{\alpha}{2}) \\ 0 & -2c(\frac{\alpha}{2})s(\frac{\alpha}{2}) & 2c^2(\frac{\alpha}{2})-1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & s_\alpha \\ 0 & -s_\alpha & c_\alpha \end{bmatrix}. \end{aligned} \tag{D.8}$$

For the second rotation, we rotate $\underline{\vec{e}}^{\,1}$ about $\vec{e}_y^{\,0}$ to obtain $\underline{\vec{e}}^{\,2}$. We again can use the quaternion theory and the direction cosine matrix obtained above, to determine the direction cosine matrix for rotating from frame $\underline{\vec{e}}^{\,1}$ to frame $\underline{\vec{e}}^{\,2}$. The Euler parameters are defined as;

$$\begin{aligned} &q_0 = \cos(\tfrac{\beta}{2}) \\ &\vec{q} = \vec{n}\sin(\tfrac{\beta}{2}) = [0\;1\;0](\underline{A}^{10})^T\sin(\tfrac{\beta}{2})\underline{\vec{e}}^{\,1}. \end{aligned} \tag{D.9}$$

Substituting $\underline{q} = [q_1\ q_2\ q_3]^T = [0\ \cos(\alpha)\sin(\frac{\beta}{2})\ -\sin(\alpha)\sin(\frac{\beta}{2})]^T$ into the direction cosine matrix defined by (D.6), gives the direction cosine matrix for rotating from frame $\underline{\vec{e}}^{\,1}$ to frame $\underline{\vec{e}}^{\,2}$;

$$\underline{A}^{21} = \begin{bmatrix} 2c^2(\frac{\beta}{2})-1 & -2c(\frac{\beta}{2})s(\frac{\beta}{2})s(\alpha) & -2c(\frac{\beta}{2})s(\frac{\beta}{2})c(\alpha) \\ 2c(\frac{\beta}{2})s(\frac{\beta}{2})s(\alpha) & 2(c^2(\frac{\beta}{2})+s^2(\frac{\beta}{2})c^2(\alpha))-1 & -2s^2(\frac{\beta}{2})c(\alpha)s(\alpha) \\ 2c(\frac{\beta}{2})s(\frac{\beta}{2})c(\alpha) & -2s^2(\frac{\beta}{2})c(\alpha)s(\alpha) & 2(c^2(\frac{\beta}{2})+s^2(\frac{\beta}{2})s^2(\alpha))-1 \end{bmatrix}. \quad \text{(D.10)}$$

Using standard goniometric equations, we can rewrite (D.10) as follows,

$$\underline{A}^{21} = \begin{bmatrix} c(\beta) & -s(\beta)s(\alpha) & -s(\beta)c(\alpha) \\ s(\beta)s(\alpha) & c(\beta)+2s^2(\frac{\beta}{2})c^2(\alpha) & -s^2(\frac{\beta}{2})c(\alpha)s(\alpha) \\ s(\beta)c(\alpha) & -s^2(\frac{\beta}{2})c(\alpha)s(\alpha) & c(\beta)+2s^2(\frac{\beta}{2})s^2(\alpha) \end{bmatrix}. \quad \text{(D.11)}$$

The final rotation is about $\vec{e}_z^{\,0}$. In order to determine the corresponding direction cosine matrix $\underline{A}^{32}$, we first need to define the Euler parameters. The Euler parameters are determined by using the direction cosine matrices $\underline{A}^{10}$ and $\underline{A}^{21}$ determined above:

$$\begin{aligned} q_0 &= \cos(\tfrac{\gamma}{2}) \\ \vec{q} &= \vec{n}\sin(\tfrac{\gamma}{2}) = [0\ 1\ 0](\underline{A}^{21}\underline{A}^{10})^T \sin(\tfrac{\gamma}{2})\underline{\vec{e}}^{\,2}. \end{aligned} \quad \text{(D.12)}$$

Substituting $\underline{A}^{21}$ and $\underline{A}^{10}$ into (D.12) gives,

$$\begin{aligned} \vec{q} &= \begin{bmatrix} -s(\frac{\gamma}{2})s(\beta) \\ s(\frac{\gamma}{2})((c(\beta)+2s^2(\frac{\beta}{2})c^2(\alpha))s(\alpha) - s^2(\frac{\beta}{2})c^2(\alpha)s(\alpha)) \\ s(\frac{\gamma}{2})(-s^2(\frac{\beta}{2})c(\alpha)s^2(\alpha) + c(\beta)c(\alpha) + 2s^2(\frac{\beta}{2})s^2(\alpha)c(\alpha)) \end{bmatrix} \underline{\vec{e}}^{\,2} \\ &= [-s(\tfrac{\gamma}{2})s(\beta),\ s(\tfrac{\gamma}{2})c(\beta)s(\alpha),\ s(\tfrac{\gamma}{2})c(\beta)c(\alpha)]^T\underline{\vec{e}}^{\,2} \end{aligned} \quad \text{(D.13)}$$

The direction cosine matrix $\underline{A}^{32}$ can then be determined by substituting the parameters into the direction cosine matrix presented in (D.6):

$$\underline{A}^{32} = \begin{bmatrix} c(\gamma)+2s^2(\frac{\gamma}{2})s^2(\beta) & 2(c(\frac{\gamma}{2})s(\frac{\gamma}{2})c(\beta)c(\alpha) - s^2(\frac{\gamma}{2})s(\beta)c(\beta)s(\alpha)) & -2(s^2(\frac{\gamma}{2})s(\beta)c(\beta)c(\alpha) + c(\frac{\gamma}{2})s(\frac{\gamma}{2}c(\beta)s(\alpha))) \\ -2(s^2(\frac{\gamma}{2})s(\beta)c(\beta)s(\alpha) + c(\frac{\gamma}{2})s(\frac{\gamma}{2})c(\beta)c(\alpha)) & c(\gamma)+2s^2(\frac{\gamma}{2})c^2(\beta)s^2(\alpha) & 2(s^2(\frac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) - c(\frac{\gamma}{2})s(\frac{\gamma}{2})s(\beta)) \\ 2(c(\frac{\gamma}{2})s(\frac{\gamma}{2}c(\beta)s(\alpha) - s^2(\frac{\gamma}{2})s(\beta)c(\beta)c(\alpha)) & 2(s^2(\frac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) + c(\frac{\gamma}{2})s(\frac{\gamma}{2})s(\beta)) & (c(\gamma)+2s^2(\frac{\gamma}{2})c^2(\beta)c^2(\alpha)) \end{bmatrix} \quad \text{(D.14)}$$

The final direction cosine matrix, describing 3D rotation parameterised by RPY-angles, can now be obtained by multiplying the above obtained direction cosine matrices:

$$\underline{A}^{30} = \underline{A}^{32}\underline{A}^{21}\underline{A}^{10}. \quad \text{(D.15)}$$

Each element of the overall direction cosine matrix will be elaborated below:

$$
\begin{aligned}
\underline{A}^{30}_{11} &= c(\gamma)c(\beta) + 2s^2(\tfrac{\gamma}{2})s^2(\beta)c(\beta) + 2s(\beta)s(\alpha)\left(c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)c(\alpha) - s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)s(\alpha)\right) \\
&\quad -2s(\beta)c(\alpha)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)c(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)s(\alpha)\right) \\
&= c(\gamma)c(\beta), \\
\\
\underline{A}^{30}_{21} &= -2c(\beta)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)s(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)c(\alpha)\right) + s(\beta)s(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)s^2(\alpha)\right) \\
&\quad +2s(\beta)c(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) - c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) \\
&= c(\gamma)s(\beta)s(\alpha) - s(\gamma)c(\alpha), \\
\\
\underline{A}^{30}_{31} &= 2c(\beta)\left(c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)s(\alpha) - s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)c(\alpha)\right) + 2s(\beta)s(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) \\
&\quad +s(\beta)c(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)c^2(\alpha)\right) \\
&= c(\gamma)s(\beta)c(\alpha) + s(\gamma)s(\alpha), \\
\\
\underline{A}^{30}_{12} &= 2c(\alpha)\left(c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)c(\alpha) - s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)s(\alpha)\right) + 2s(\alpha)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)c(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)s(\alpha)\right) \\
&= s(\gamma)c(\beta), \\
\\
\underline{A}^{30}_{22} &= c(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)s^2(\alpha)\right) - 2s(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)c(\alpha)s(\alpha) - c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) \\
&= c(\gamma)c(\alpha) + s(\gamma)s(\beta)s(\alpha), \\
\\
\underline{A}^{30}_{32} &= 2c(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)c(\alpha)s(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) - s(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)c^2(\alpha)\right) \\
&= s(\gamma)s(\beta)c(\alpha) - c(\gamma)s(\alpha), \\
\\
\underline{A}^{30}_{31} &= -s(\beta)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})s^2(\beta)\right) + 2c(\beta)s(\alpha)\left(c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)c(\alpha) - s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)s(\alpha)\right) \\
&\quad -2c(\beta)c(\alpha)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)c(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)s(\alpha)\right) \\
&= -s(\beta), \\
\\
\underline{A}^{30}_{23} &= 2s(\beta)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)s(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)c(\alpha)\right) + c(\beta)s(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)s^2(\alpha)\right) \\
&\quad +2c(\beta)c(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) - c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) \\
&= c(\beta)s(\alpha), \\
\\
\underline{A}^{30}_{33} &= 2s(\beta)\left(s^2(\tfrac{\gamma}{2})s(\beta)c(\beta)c(\alpha) - c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})c(\beta)s(\alpha)\right) + 2c(\beta)s(\alpha)\left(s^2(\tfrac{\gamma}{2})c^2(\beta)s(\alpha)c(\alpha) + c(\tfrac{\gamma}{2})s(\tfrac{\gamma}{2})s(\beta)\right) \\
&\quad +c(\beta)c(\alpha)\left(c(\gamma) + 2s^2(\tfrac{\gamma}{2})c^2(\beta)c^2(\alpha)\right) \\
&= c(\beta)c(\alpha),
\end{aligned}
\tag{D.16}
$$

From the results presented above, we can conclude that deriving the direction cosine matrix for RPY-angles using Euler parameters, gives the same results as presented in literature (see (D.2)).

# Appendix E

# Overview point-to-point cases

This appendix presents an overview of the possible acceleration profiles when determining time-optimal point-to-points with defined begin and end velocities. The overview of these acceleration profiles can be found in Figure E.1.

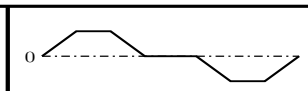
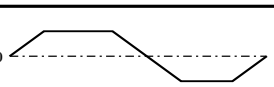
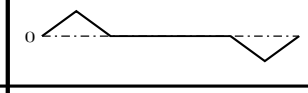
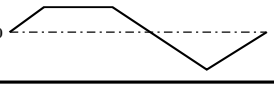
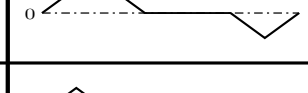
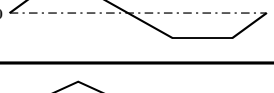
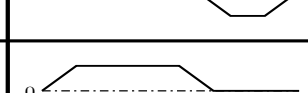
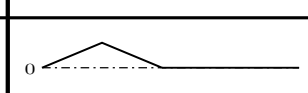
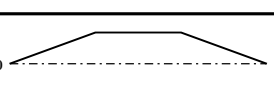
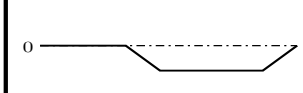
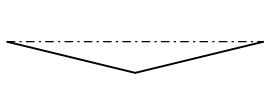
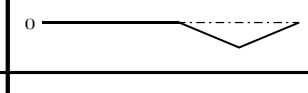
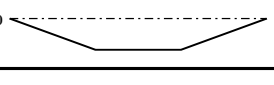
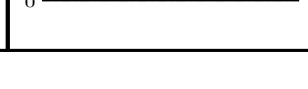
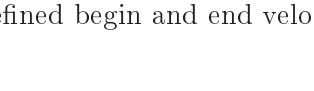
| | | | | | |
|---|---|---|---|---|---|
| 1 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\tau_4 > 0,$ $\tau_5 > 0,\ \tau_6 > 0,\ \tau_7 > 0.$ | 10 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 > 0,\ \tau_7 > 0.$ |
| 2 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\tau_4 > 0,$ $\tau_5 > 0,\ \tau_t = 0,\ \tau_7 > 0.$ | 11 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 = 0,\ \tau_7 > 0.$ |
| 3 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 > 0,\ \tau_6 = 0,\ \tau_7 > 0.$ | 12 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 > 0,\ \tau_7 > 0.$ |
| 4 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 > 0,\ \tau_5 > 0,\ \tau_7 > 0.$ | 13 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 = 0,\ \tau_7 > 0.$ |
| 5 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 = 0,\ \tau_6 = 0,\ \tau_7 = 0.$ | 14 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 = 0,\ \tau_6 = 0,\ \tau_7 = 0.$ |
| 6 |  | $\tau_1 > 0,\ \tau_2 = 0,\ \tau_3 > 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 = 0,\ \tau_6 = 0,\ \tau_7 = 0.$ | 15 |  | $\tau_1 > 0,\ \tau_2 > 0,\ \tau_3 > 0,$ $\tau_4 = 0,$ $\tau_5 = 0,\ \tau_6 = 0,\ \tau_7 = 0.$ |
| 7 |  | $\tau_1 = 0,\ \tau_2 = 0,\ \tau_3 = 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 > 0,\ \tau_6 > 0,\ \tau_7 > 0.$ | 16 |  | $\tau_1 = 0,\ \tau_2 = 0,\ \tau_3 = 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 = 0,\ \tau_7 > 0.$ |
| 8 |  | $\tau_1 = 0,\ \tau_2 = 0,\ \tau_3 = 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 > 0,\ \tau_6 = 0,\ \tau_7 > 0.$ | 17 |  | $\tau_1 = 0,\ \tau_2 = 0,\ \tau_3 = 0,$ $\tau_4 = 0,$ $\tau_5 > 0,\ \tau_6 > 0,\ \tau_7 > 0.$ |
| 9 |  | $\tau_1 = 0,\ \tau_2 = 0,\ \tau_3 = 0,$ $\boldsymbol{\tau_4 > 0,}$ $\tau_5 = 0,\ \tau_6 = 0,\ \tau_7 = 0.$ | | | |

Figure E.1: Overview of possible acceleration profiles for determining point-to-points with defined begin and end velocity.

# Appendix F

# Derivation of Point-to-Point time parameters

This appendix presents the derivation of point-to-point time-parameters for a case which results in a non-symmetric profile. The derivation of other cases, as situated in Appendix E, can be solved in the same manner. The case presented here, is already presented in the example in Section 4.2.1. As can be seen from the figure in Section 4.2.1, the one point-to-point time parameter is equal to zero, namely $\tau_6 = 0$. The remaining parameters,

$$\tau_i > 0, \text{ for } i = 1, \ldots, 5, 7, \tag{F.1}$$

are larger than zero. This implies that a constant velocity part exists, and thus that the velocity at $v(t_3) = v_{max}$, with $t_3 = \tau_1 + \tau_2 + \tau_3$. This also holds for $v(t_4)$, with $t_4 = \tau_1 + \tau_2 + \tau_3 + \tau_4$. We first will determine the parameters for the speedramp from $v(t = 0)$ to $v(t = t_3)$. The velocity and acceleration profiles corresponding to this situation is depicted in Figure F.1.

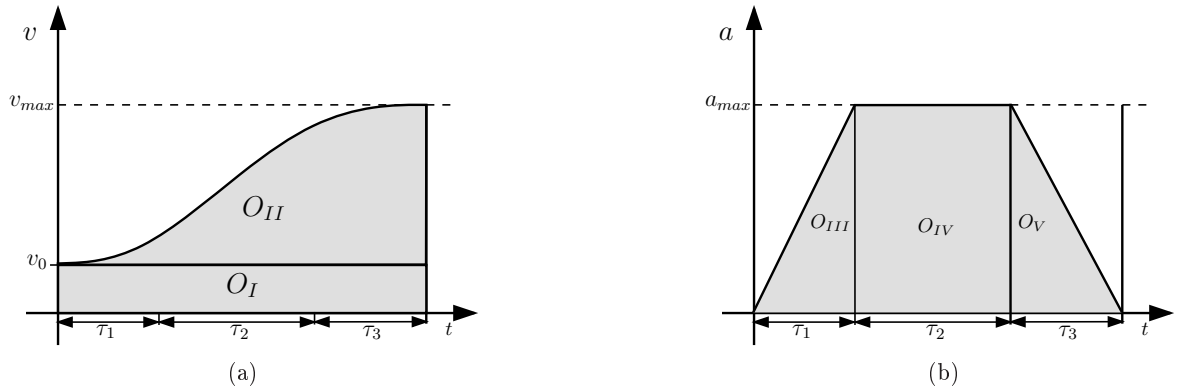



Figure F.1: Velocity (a) and acceleration (b) profile for speedramp from $v(t = 0)$ to $v(t = t_3)$.

In case that a constant acceleration part exists, the total velocity change (i.e. $\Delta v_1 = v_{max} - v_0$) is reached by an acceleration profile that reaches the maximum allowable acceleration $a_{max}$. The velocity change $\Delta v_1$ is obtained by the acceleration profile given in Figure F.1b. The area under this profile must thus equal the velocity change $\Delta v_1$,

$$v_{max} - v_0 = O_{III} + O_{IV} + O_V, \tag{F.2}$$

with $O_{III} = \frac{a_{max}}{2}\tau_1$, $O_{IV} = a_{max}\tau_2$ and $O_V = \frac{a_{max}}{2}\tau_3$. Since the acceleration profile is symmetric with identic begin and end values, $O_{III}$ and $O_V$ must be equal to each other. From this and the fact that the motion is defined to be time-optimal (during time instants $\tau_1$ and $\tau_3$ the jerk constraint is active) we can determine $\tau_1$ and $\tau_3$.

$$j_{max}\tau_1 = a_{max}. \tag{F.3}$$

Since $O_{III} = O_V$, this means that,

$$\tau_1 = \tau_3 = \frac{a_{max}}{j_{max}}. \tag{F.4}$$

Now, $\tau_2$ can be determined by combining (F.2) and (F.4) and rearranging the terms,

$$\tau_2 = \frac{v_{max} - v_0}{a_{max}} - \frac{a_{max}}{j_{max}}. \tag{F.5}$$

The same derivation for the parameters $\tau_5$ and $\tau_7$ can be made. However, in this case there is not a constant acceleration part. So, the maximum acceleration is not reached. This case is depicted in Figure F.2.

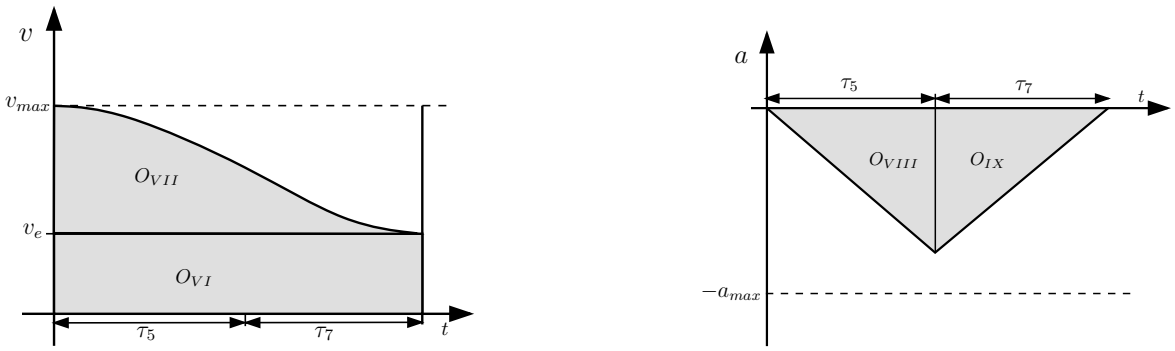



Figure F.2: Velocity (a) and acceleration (b) profile for speedramp from $v(t = t_5)$ to $v(t = t_7)$.

So in this case the velocity change $\Delta v_2 = v_{max} - v_e$ can be determined by evaluating the area under the acceleration profile depicted in Figure F.2b.

$$v_{max} - v_e = O_{VI} + O_{VII}, \tag{F.6}$$

with $O_{VIII} = \frac{j_{max}}{2}\tau_1^2$ and $O_{IX} = \frac{j_{max}}{2}\tau_5^2$. Also here, the areas $O_{VIII}$ and $O_{IX}$ must be equal to each other, otherwise the acceleration cannot be equal to zero at the end point. This implies that,

$$\tau_5 = \tau_7. \tag{F.7}$$

Using this the point-to-point time parameters $\tau_5$ and $\tau_7$ can be solved as,

$$\tau_5 = \tau_7 = \sqrt{\frac{v_{max} - v_e}{j_{max}}}. \tag{F.8}$$

Finally, $\tau_4$ needs to be determined. This can be done by taking the total distance to travel $x_{max}$ subtracted by the distance covered due to the speedramp from $v_0$ to $v_{max}$ and the speedramp

from $v_{max}$ to $v_e$. The distance covered by the two speedramps is given by the total area under the velocity profiles, i.e. $O_{dx} = O_I + O_{II} + O_{VI} + O_{VII}$. When $O_{dx}$ is known, $\tau_4$ is determined as follows,

$$\tau_4 = \frac{x_{max} - O_{dx}}{v_{max}}. \tag{F.9}$$

As stated above $O_{dx}$ is determined by calculating the areas under the velocity profiles of the speedramp from $v_0$ to $v_{max}$ and the speedramp from $v_{max}$ to $v_e$. The areas are given as

$$\begin{aligned} O_I &= v_0(\tau_1 + \tau_2 + \tau_3) = v_0(2\tau_1 + \tau_2), \\ O_{II} &= \tfrac{1}{2}(v_{max} - v_0)(\tau_1 + \tau_2 + \tau_3) = \tfrac{1}{2}(v_{max} - v_0)(2\tau_1 + \tau_2), \\ O_{VI} &= v_e(\tau_5 + \tau_7), \\ O_{VII} &= \tfrac{1}{2}(v_{max} - v_e)(\tau_5 + \tau_7) = (v_{max} - v_0)\tau_5. \end{aligned} \tag{F.10}$$

The derivation presented in this appendix can be followed for each of the cases presented in Appendix E.

# Appendix G

# Testcase evaluation matrices

This appendix presents the evaluation matrices with corresponding weighing factors that are used for the evaluation of simulation and experimental results. First, the evaluation matrix for the pick-and-place case is presented. Next, the case in which a path is fully known in advance is discussed. After that, the evaluation matrix for a straight line segment is discussed. Finally, an example of a multi-criteria analysis will be given.

## G.1 Pick-and-place evaluation matrix

For pick-and-place motions, the total motion time is the most important issue, since reducing the motion time results in a higher throughput of the manipulator. Therefore, the total motion time has the highest weighing factor, as can be seen from Table G.1. The path between pick location and place location is of less importance than the accuracy with which the end-point (pick or place position) is reached. This is distinguished by a difference in weighing factors. Regarding the quantitative criteria, weighing factors of 3, 5, 5 are given for joint acceleration smoothness, constraint handling and usability, respectively.

Table G.1: Requirements evaluation matrix for pick-and-place motion.

| Criteria | Weighing factor |
|---|---|
| Total motion time [sec] | 10 |
| Path following accuracy [mm] | 1 |
| End-point settling [mm] | 7 |
| Joint acceleration smoothness [-] | 3 |
| Constraint handling [-] | 5 |
| Usability [-] | 5 |

## G.2 Path constrained motion evaluation matrix

While in case of a pick-and-place motion the total motion time is the most important criterium; in case of a path constrained motion the path following accuracy is the most important criterium.

Therefore, this criteria is assigned with the highest weighing factor. This also holds for the end-point settling criterium. Since the performance criterium used in this work is total motion time, the weighing factor is relatively chosen. The qualitative criteria are chosen as in the pick-and-place case.

Table G.2: Requirements evaluation matrix for path constrained motion.

| Criteria | Weighing factor |
|---|---|
| Total motion time | 8 |
| Path following accuracy | 10 |
| End-point settling | 10 |
| Joint acceleration smoothness | 3 |
| Constraint handling | 5 |
| Usability | 5 |

## G.3 Straight line evaluation matrix

The straight line testcase is used to provide insight in the difference between the developed algorithms. Since, both algorithms are are able to determine straight line segments. Therefore, in this case all the weighing factors are chosen equal to one.

Table G.3: Requirements evaluation matrix for straight line motion.

| Criteria | Weighing factor |
|---|---|
| Total motion time | 1 |
| Path following accuracy | 1 |
| End-point settling | 1 |
| Joint acceleration smoothness | 1 |
| Constraint handling | 1 |
| Usability | 1 |

## G.4 Example of a multi-criteria analysis

In this section an example of a multi-criteria analysis is presented. In Table G.4 the data for the example is provided. In this example we want to evaluate the performance of two algorithms.
As can be seen, the data in Table G.4 consists of quantitative data and qualitative data. For each set of data(i.e. qualitative and quantitative) a dominance score is determined. Based on these dominance score for qualitative and quantitative data and the weighing factors, an overall dominance score is determined. First, we will discuss the determination of the quantitative dominance scores. Next, the determination of the qualitative dominance scores is determined. Finally, the overall dominance score and corresponding overall rank is presented.

Table G.4: Example data for multi-criteria analysis example.

| Criteria | Weighing factor | Algorithm 1 | Algorithm 2 |
|---|---|---|---|
| Total motion time | 8 | 1.5 | 2 |
| Path following accuracy | 10 | 0.01 | 0.005 |
| End-point settling | 10 | 0.005 | 0.001 |
| Joint acceleration smoothness | 3 | +++ | ++++ |
| Constraint handling | 5 | +++ | +++ |
| Usability | 5 | ++++ | +++ |

### G.4.1 Determination of quantitative dominance scores

As discussed in detail in [TDO, 2004], quantitative data contains a lot of information of which option scores better and how much better. However, in order to make the data comparable with the qualitative data the scores are first standardized by,

$$e_i = \frac{n_i - n_{lowest}}{n_{highest} - n_{lowest}}, \tag{G.1}$$

where $e_i$ represents the standardized score for option $i$, $n_i$ the actual score for option $i$, $n_{lowest}$ the lowest recorded score for the criterion and $n_{highest}$ the highest recorded score for the criterion. Since, the criteria used for this evaluation have a negative direction (e.g. the highest motion time must result in the worst score) a correction must be applied. The new standardized scores are determines by,

$$e_{new,i} = 1 - e_i, \tag{G.2}$$

where $e_{new,i}$ represents the new standardized score and $e_i$ the standardized score presented in (G.1). Using the presented scoring strategy the quantitative data for example data of Table G.4, the standardized scores result in, After determining the standardized scores the next step is to

Table G.5: Standardized quantitative scores for example data.

| Criteria | Weighing factor | Algorithm 1 | Algorithm 2 |
|---|---|---|---|
| Total motion time | 8 | 1 | 0 |
| Path following accuracy | 10 | 0 | 1 |
| End-point settling | 10 | 0 | 1 |

calculate the dominance scores for the quantitative data. A dominance score indicates to what extend algorithm $i$ is performs better or worse than algorithm $j$. The dominance score is determined using the standardized scores, the defined criteria and corresponding weighing factors. This can be given as,

$$d_{ij} = \sum_{k=1}^{l} w_k(e_{k,i} - ek, j) \tag{G.3}$$

with $d_{ij}$ the quantitative dominance score of algorithm $i$ compared to algorithm $j$, $w_k$ the weighing factor of criterion $k$, $e_{k,i}$ the standardized score of algorithm $i$ for criterion $k$, $e_{k,j}$ the standardized score of algorithm $j$ for criterion $k$, and $l$ the number of quantitative criteria. For the example of this section, this results in the following dominance score;

$$d_{12} = 8(1-0) + 10(0-1) + 10(0-1) = -12. \tag{G.4}$$

It can be seen that in this case the dominance score is negative. Therefore, it can be concluded that for algorithm 2 performs better than algorithm 1, based on the quantitative scores.

### G.4.2 Determination of qualitative dominance scores

Next, the dominance scores for qualitative data are determined. No standardization for qualitative data is needed, since this data is already standardized. Therefore, we can directly determine the dominance scores in case of qualitative data. This is defined by,

$$a_{ij} = \sum_{k=1}^{m} w_k \cdot sign(n_{k,i} - nk, j), \tag{G.5}$$

with $a_{ij}$ the qualitative dominance score of algorithm $i$ compared to algorithm$j$, $w_k$ again the weighing factor of criterion $k$, $n_{k,i}$ the qualitative score of algorithm $i$ and $m$ the number of qualitative criteria. Furthermore, the $sign(\ldots)$ function is defined as:

$$\begin{aligned} &= +, \text{ if } n_{k,i} \text{ has more + scores than } n_{k,j}, \\ &= 0, \text{ if } n_{k,i} \text{ has the same number of + scores than } n_{k,j}, \\ &= -, \text{ if } n_{k,i} \text{ has fewer + scores than } n_{k,j}. \end{aligned} \tag{G.6}$$

So the $sign(\ldots)$ function shows whether a weighing factor should be added, neglected or subtracted from the dominance score. For the qualitative data in this example, the following qualitative dominance score is then determined,

$$a_{12} = 3(-) + 5(0) + 5(+) = 2. \tag{G.7}$$

Based on the qualitative data it is shown that algorithm 1 performs better than algorithm 2. How the overall dominance score is determined using the result form this and previous section is presented in the following section.

### G.4.3 Determination of the overall dominance score

Based on the quantitative and qualitative dominance score, determined in the previous sections, the overall dominance score is determined. The overall dominance score is defined as,

$$m_{ij} = w_{qt} D_{ij} + w_{qw} A_{ij}, \tag{G.8}$$

with $m_{ij}$ the overall dominance score of algorithm $i$ with respect to algorithm $j$, $w_{qt}$ the summation of the quantitative weighing factors (i.e. $w_{qt} = \sum_{k=1}^{l} w_k$), $w_{qw}$ the summation of the qualitative weighing factors (i.e. $w_{qw} = \sum_{k=1}^{m} w_k$), $D_{ij}$ is the quantitative dominance score of algorithm $i$ with respect to algorithm 2 divided by the absolute value of all quantitative dominance scores, i.e.

$$D_{ij} = \frac{d_{ij}}{\sum_{i=1}^{c} \sum_{j=1}^{c} |d_{ij}|}. \tag{G.9}$$

with $c$ the number of algorithms under evaluation. The same formula can be written down for the quantitative dominance score $A_{ij}$;

$$A_{ij} = \frac{a_{ij}}{\sum_{i=1}^{c} \sum_{j=1}^{c} |a_{ij}|}. \tag{G.10}$$

For the qualitative and quantitative dominance scores of the example data presented in Table G.4, the overall dominance score is calculated as follows,

$$m_{12} = 28\frac{-12}{24} + 13\frac{2}{4} = -7.5. \tag{G.11}$$

So, from the overall dominance score of (G.11) we can conclude that for this set of data and weighing factors algorithm 2 performs better than algorithm 1.

# Bibliography


[TDO, 2004] (2004). Technology and sustainability. Notes for the " technology for sustainable development 4P800)" course, Eindhoven, University of Technology.

[Ashiru, 1995] Ashiru, I. and Czarnecki, C. (1995). Optimal motion planning for mobile robots using genetic algorithms. In *International Conference on industrial automation and control*, pages 297–300, Hyderabad. IEEE.

[Athans, 1966] Athans, M. and Falb, P. L. (1966). *Optimal control: an introduction to the theory and its applications*. London: McGraw-Hill.

[Bartels, 1987] Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann Publishers.

[Bobrow, 1985] Bobrow, J. E., Dubowsky, S., and Gibson, J. S. (1985). Time-optimal control of robotic manipulators along specified paths. *International journal of robotic research*, 4:3–17.

[Borenstein, 1989] Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. volume 9, pages 1179–1187.

[Bruyninckx, 2004] Bruyninckx, H. and Schutter, J. D. (2004). Introduction to intelligent robotics. Notes for the "robotics course", Katholieke universiteit Leuven.

[Bryson, 1975] Bryson, A. E. and Ho, Y.-C. (1975). *Applied optimal control : optimization, estimation and control*. S.l.: Hemisphere. Revised edition.

[Canny, 1988] Canny, J. F. (1988). *The complexity of robot motion planning*. MIT Press, Cambridge, Massachusetts.

[Chang, 1992] Chang, Y.-H., Lee, T.-L., and Liu, C.-H. (1992). On-line approximate cartesian path trajectory planning for robotic manipulators. In *Transactions on sytems, man, and cybernetics*, volume 22, pages 542–547. IEEE.

[Constantinescu, 2000] Constantinescu, D. and Croft, E. A. (2000). Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of robotic systems*, 17(5):233–249.

[Craig, 1986] Craig, J. J. (1986). *Introduction to robotics*. New York: Addision-Wesley.

[Davidor, 1990] Davidor, Y. (1990). Robot motion planning using a genetic algorithm. In *International Conference on Computer Systems and Software Engineering*, pages 186–191, Tel Aviv. IEEE.

[de Boor, 1978] de Boor, C. (1978). *A practical guide to splines*. Springer-Verlag, New York.

[Denavit, 1955] Denavit, J. and Hartenberg, R. S. (1955). A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics*, 22:215–221.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[Fan, 1994] Fan, K. and Lui, P. (1994). Solving find path problem in mapped environments using modified A* algorithm. *Transactions on systems, man and cybernetics*, 24(9):1390–1396.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Amsterdam.

[Heath, 1997] Heath, M. T. (1997). *Scientific computing: an introductory survey*. McGraw-Hill.

[Keij, 2003] Keij, J. J. A. M. (2003). Obstacle avoidance for wheeled mobile robotic systems. DCT 2003.11, Eindhoven University of Technology.

[Khatib, 1980] Khatib, O. (1980). *Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles*. PhD thesis, Ecole nationale supérieure de l'aéronautique et de l'espace, Toulouse. (In french).

[Kim, 2002] Kim, S.-J., Choi, D.-S., and Ha, I.-J. (2002). Time-optimal control of state-constrained second-order systems and an application to robotic manipulators. In *Proceedings of the American control conference*, pages 1478–1483, Anchorage.

[Ko, 1992] Ko, N. Y., Lee, B. H., Ko, M. S., and Nam, Y. S. (1992). Robot motion planning for time-varying obstacle avoidance using view-time concept. In *Proceedings of the international symposium on industrial electronics*, volume 1, pages 366–370, Xian, China. IEEE.

[Kurfess, 2005] Kurfess, T. R. (2005). *Robotics and automation handbook*. CRC Press, London.

[Lambrechts, 2003] Lambrechts, P. F. (2003). Trajectory planning and feedforward design for electromechanical motion systems. Technical Report DCT 2003-18, Eindhoven, University of Technology.

[Latombe, 1991] Latombe, J. C. (1991). *Robot motion planning*. Kluwer Academic Publisher.

[LaValle, 2005] LaValle, S. M. (2005). *Planning algorithms*. [Online]. Available at http://msl.cs.uiuc.edu/planning/.

[Lee, 1967] Lee, E. B. and Markus, L. (1967). *Foundations of optimal control theory*. London: Wiley.

[Liang, 1999] Liang, T. C. and Liu, J. S. (1999). An improved trajectory planner for redundant manipulators in constrained workspace. *Journal of robotic systems*, 16(6):339–351.

[Lin, 1983] Lin, C. S., Chang, P. R., and Luh, J. Y. S. (1983). Formulation and optimization of cubic polynomial joint trajectories for industrial robots. In *Transactions on automatic control*, volume 28, pages 1066–1074. IEEE.

[Lloyd, 1993] Lloyd, J. and Hayward, V. (1993). Trajectory generation for sensor-driven and time-varying tasks. *International journal of robotics research*, 12(4):380–393.

[Macfarlane, 2003] Macfarlane, S. and Croft, E. (2003). Jerk-bounded manipulator trajectory planning: design for real-time applications. *Transactions on robotics and automation*, 19(1):42–52.

[Malmgren, 1995] Malmgren, A. and Nordström, K. (1995). Minimum time state feedback control with constrained inputs and states. In *Conference on decision and control*, pages 802–808, New Oreleans, LA.

[Meng, 1988] Meng, A. C. C. (1988). Free space modelling and geometric motion planning under unexpected obstacles. In *Proceedings of the fourth conference on artificial intelligence for applications*, pages 82–87, San Diego, CA. IEEE.

[Mirolo, 1991] Mirolo, C. and Pagello, E. (1991). Local geometric issues for spatial reasoning in robot motion planning. *International workshop on intelligent robots and systems*, pages 569–574.

[Nelder, 1965] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7(4):308–313.

[Pajak, 2004] Pajak, G., Pajak, I., and Galicki, M. (2004). Trajectory planning of multiple manipulators. *Fourth international workshop on robot motion and control*, pages 121–126.

[Papalambros, 2000] Papalambros, P. Y. and Wilde, D. J. (2000). *Principles of optimal design: modeling and computation*. Cambridge University Press. 2nd edition.

[Park, 2005] Park, J. K. and Bobrow, J. E. (2005). Reliable computation of minimum-time motions for manipulators moving in obstacle fields using a successive search for minimum-overload trajectories. *Journal of robotic systems*, 22(1):1–14.

[Paviani, 1969] Paviani, D. and Himmelblau, D. M. (1969). Constrained nonlinear optimization by heuristic programming. *Operations Research*, 17:872–882.

[Penev, 2005] Penev, B. G. and Christov, N. D. (2005). A fast time-optimal control synthesis algorithm for a class of linear systems. In *American control conference*, pages 883–888, Portland, OR.

[Pfeiffer, 1987] Pfeiffer, F. and Johanni, R. (1987). A concept for manipulator trajectory planning. *IEEE journal of robotics and automation*, 3(2):115–123.

[Piazzi, 2000] Piazzi, A. and Visioli, A. (2000). Global minimum-jerk trajectory planning of robot manipulators. *Transactions on industrial electronics*, 47(1):140–149.

[Rimon, 1992] Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. In *IEEE transactions on robotics and automation*, volume 8, pages 501–518.

[Schoenberg, 1946] Schoenberg, I. J. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. applied mathematics*, 4:45–99 and 112–141.

[Sciavicco, 1996] Sciavicco, L. and Siciliano, B. (1996). *Modeling and control of robot manipulators*. McGraw-Hill Companies Inc.

[Sharir, 1989] Sharir, M. (1989). Algorithmic motion planning in robotics. *Computer*, 22(3):9–19.

[Shiller, 1990] Shiller, Z. and Lu, H.-H. (1990). Robust computation of path constrained time optimal motions. In *IEEE International conference on Robotics and Automation*, volume 1, pages 144–149. IEEE.

[Shin, 1985] Shin, K. G. and McKay, N. D. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE transactions on automatic control*, 30(6):531–541.

[Sim, 2000] Sim, Y. C., Leng, S. B., and Subramaniam, V. (2000). A combined genetic algorithms-shooting method approach to solving optimal control problems. *International journal of systems science*, 31(1):83–89.

[Slotine, 1989] Slotine, J.-J. E. and Yang, H. S. (1989). Improving the efficiency of time-optimal path-following algorithms. *IEEE transactions on robotics and automation*, 5(1):118–124.

[Stembera, 2005] Stembera, J. (2005). Inverse kinematical algorithms of redundant robots. Technical report, Czech Technical University in Prague.

[The Mathworks Inc., 2006] The Mathworks Inc. (2006). *Matlab User's Guide*. Natick, Massachusetts. version 7.

[van de Wouw, 2001] van de Wouw, N. (2001). Multibody dynamics. Notes for the "multibody dynamics (4J400)" course, Eindhoven, University of Technology.

[van Tuijl, 1991] van Tuijl, F., Lagewaard, S., Rienstra, S., and Reusken, A. (1991). Toward look-ahead feed and cubic hermite interpolation. Technical Report DER 6-32.1-4785B007, Philips machine tool controls.

[Wang, 2004] Wang, J., Qu, Z., Guo, Y., and Yang, J. (2004). A reduced-order analytical solution to mobile robot trajectory generation in the presence of moving obstacles. In *Proceedings of the 2004 IEEE international conference on robotics and automation*, pages 4301–4307, New Orleans, LA. IEEE.

[Weisstein, 2006] Weisstein, E. W. (1999-2006). Global optimization. From Mathworld–A wolfram web resource http://mathworld.wolfram.com/GlobalOptimization.html.

[Yamamoto, 1994] Yamamoto, M., Isshiki, Y., and Mohri, A. (1994). Collision free minimum time trajectory planning for manipulators using global search and gradient method. In *Proceedings of the IEEE/RSJ/GI international conference on intelligent robots and systems*, pages 2184–2191, Munich, Germany. IEEE.

[Zah, 2004] Zah, X. F. and Chen, X. Q. (2004). Trajectory coordination planning and control for robot manipulators in automated material handling and processing. *International journal of advanced manufacturing technologies*, 23:831–845.

[Žlajpah, 1996] Žlajpah, L. (1996). On time optimal path control of manipulators with bounded joint velocities and torques. In *Proceedings of the international conference on robotics and automation*, pages 1572–1577, Minneapolis, Minnesota. IEEE.