Optimal Control *of* Mechatronic Systems

A Differentially Flat Approach

Wannes Van Loock

August 2013

Submitted in partial fulfillment of the requirements
for the degree of Doctor in Engineering

Supervisors:
Prof. dr. ir. G. Pipeleers (supervisor)
Prof. dr. ir. J. Swevers (co-supervisor)

Members of the Examination Committee:
Prof. dr. ir. Y. Willems (chairman)
Prof. dr. ir. J. De Schutter
Prof. dr. ir. H. Bruyninckx
Prof. dr. M. Diehl
Prof. dr. ir. M. Norrlöf
Dr. ir. G. Pinte

This document was typeset using the LaTeX typesetting system originally developed by Leslie Lamport, based on TeX created by Donald Knuth.

The body text is set 11pt/12pt with Linux Libertine created by the Libertine Open Fonts Project, which aims to create free and open alternatives to proprietary typefaces. Figures and captions are set with Linux Biolinum, a complementary organic sans-serif typeface.

The book's design is inspired by *Motion Mountain*, the free physics textbook by Schiller (2013). Typographical decisions are based on *The Elements of Typographical Style* by Bringhurst (2004).

ISBN 978-94-6018-742-1
D/2013/7515/127

Dankwoord

De voorbije vier jaren zijn voor mij een geweldige en leerrijke ervaring geweest. Gedurende deze tijd hebben vele vrienden en collega's me geholpen en met raad en daad bijgestaan. Via deze weg dank ik jullie allemaal uit de grond van mijn hart.

Vooreerst, Goele en Jan, bedankt! Bedankt voor het vertrouwen dat jullie in mij gesteld hebben. Bedankt voor jullie dagelijkse steun en begeleiding. Met al mijn vragen, door jullie vaak onmiddelijk voorzien van antwoord, kon ik bij jullie steeds terecht. Bedankt voor het kritisch uitpluizen en nalezen van mijn schrijfsels. Onder jullie vleugels heb ik echt enorm veel geleerd.

Ook de overige leden van mijn begeleidingscommissie verdienen zeker een bedankje. Joris, jouw input bij het begin heeft me een vliegende start gegeven. Herman, bedankt voor de kritische vragen tijdens enkele interne presentaties. Ze hebben me gemotiveerd om mijn resultaten steeds te verbeteren. Moritz, dank je voor de introductie tot en het enthousiasme voor de wondere wereld van optimalisatie. Mede hierdoor koos ik voor dit doctoraat. Naast mijn begeleidingscommissie dank ik ook de overige juryleden. Gregory Pinte en Mikael Norrlöf, jullie commentaren hebben ongetwijfeld de kwaliteit van deze thesis verbeterd. A special thanks to Mikael Norrlöf for your interest in my work and for coming all the way from Linköping to complete my jury. Ook bedank ik professor Yves Willems voor zijn bereidwilligheid om de jury voor te zitten.

Uiteraard is de bijdrage van collega's en ex-collega's van de onderzoeksgroep en van het departement van niet te onderschatten belang. Niet alleen de vele diepgaande discussies hebben dit onderzoek ondersteund. Ook hun vriendschap en de talrijke ontspannende activiteiten zoals het wiezen, het squashen, het happy-hour, het klimmen, ... hebben ervoor gezorgd dat ik met volle goesting kwam werken. In het bijzonder dank ik Pieter, niet alleen voor het gezever tijdens koffiepauzes, maar ook voor zijn wetenschappelijke input. Onze samenwerking heeft ongetwijfeld een grote invloed op de kwaliteit van mijn onderzoek gehad. Verder wil ik

de ATP collega's bedanken voor voor hun administratieve en technische ondersteuning.

Tot slot wil ik mijn ouders, (schoon)familie en (muzikale) vrienden bedanken voor alle steun buiten het werk. Ik zeg het wellicht niet vaak genoeg, maar jullie zijn van onschatbare waarde! Eva, ik weet niet eens hoe ik moet beginnen jou te bedanken. Je steun en onophoudelijke liefde geven me de kracht om telkens opnieuw het beste van mezelf te geven. En jij ... Voorlopig vertoef je rustig in de buik van mama, nog zonder naam, aan jouw draag ik dit werk op.

Het is wellicht ironish dat na vier jaren onderzoek de laatste woorden die ik hier neerpen, misschien wel het meest gelezen zullen worden. Maar ze zijn ook belangrijk, want zonder jullie was dit doctoraat er wellicht niet geweest. Merci!

Wannes
Oktober, 2013

Abstract

Control systems are prevalent in modern technology with applications ranging from washing machines, hard drives and cars in domestics to high performance production machines in industry. The ever increasing customer demands have led to rapid advances in sensing, computing and actuation technology, in turn increasing the role of advanced control theory. This evolution has led to the introduction of optimization techniques to obtain superior systems.

A typical application of control theory is computing the signals required to perform a specific task, such as steering the system from one configuration to another or following a desired trajectory for one or more variables in the system. In addition, the system's performance, such as the execution time or energy consumption, and limitations must be taken into account, leading to so-called optimal control problems.

The problem formulation is key for finding solutions to these problems reliably and efficiently and constitutes the main focus of this thesis. To this end, both the mathematical structure of the system and the signal parameterizations are carefully chosen. A piecewise polynomial parameterization of the signals is adopted, allowing for a small dimensional optimization problem in which system limitations can be imposed reliably, either by necessary and sufficient semi-definite conditions or a series of sufficient linear relaxations. In addition, so-called differentially flat systems, a generalization of linear systems, exhibit a mathematical structure that is particularly well suited for the problems at hand.

For linear systems, the combination of both piecewise polynomials and differential flatness leads to a small-dimensional problem formulation, which can be solved reliably and efficiently. Additionally, time-optimality is achieved through a novel algorithm and system uncertainties are accounted for. For nonlinear differentially flat systems, the geometric path following problem is tackled by projecting the problem onto the path and by applying a time transformation. By allowing additional freedom in the geometric path, the more general path planning problem can also be solved.

Korte inhoud

Regelsystemen zijn alom tegenwoordig in moderne technologie. Je vindt ze niet alleen in hoogperformante productiemachines maar ook in een eenvoudige wasmachine, harde schijf of thermostaat. Bovendien hebben de steeds toenemende klantverwachtingen snelle evoluties in sensoren, actuatoren en rekenkracht teweeggebracht. Dit heeft geleid tot het gebruik van optimalisatie in het ontwerp van geavanceerde regelsystemen.

Een kenmerkende toepassing van regeltechniek is het berekenen van het vereiste actuatorsignaal om een specifieke taak uit te voeren, zoals het systeem van één configuratie naar een andere sturen of een gewenst pad voor één of meerdere variabelen laten volgen. Ook de gewenste performantie en de systeemlimieten moeten in rekening gebracht worden. Dit vereist de oplossing van een zogenaamd optimaal controle probleem.

Het efficient en betrouwbaar oplossen van zulke problemen vergt een zorvuldige formulering van het probleem en vormt een belangrijk aandachtspunt in deze thesis. Zowel de parametrizatie als de wisundige vorm van het systeem worden nauwgezet gekozen. Een stuksgewijze polynomiale parametrizatie van de signalen zorgt ervoor dat systeembeperkingen betrouwbaar opgelegd kunnen worden, hetzij door nodige en voldoende semidefiniete voorwaarden of eenvoudigere voldoende lineaire voorwaarden. Bovendien beschikken zogenaamde differentieel vlakke systemen, een veralgemening van lineaire systemen, over een wiskundige vorm die uitermate geschikt is voor de beschouwde problemen.

Voor lineaire systemen leidt de combinatie van zowel stuksgewijze veeltermen als differentieel vlakke systemen tot een probleemformulering met slechts enkele onbekenden die efficient en betrouwbaar opgelost kan worden. Bovendien worden tijdsoptimale trajecten met een vernieuwend algoritme bekomen en kunnen ook systeemonzekerheden in rekening gebracht worden. Voor niet-lineaire differentieel vlakke systemen, wordt het padvolg probleem aangepakt door achtereenvolgens de systeemdynamica langsheen het pad te projecteren en een transformatie van variabelen toe te passen. Door het pad variabel te beschouwen, wordt tevens het meer algemene padplanning probleem aangepakt.

Contents

1

Introduction

Control systems are prevalent in modern technology with applications ranging from washing machines, hard drives and cars in domestics, to high performance production machines in industry. The ever increasing customer demands have led to rapid advances in sensing, computing and actuation technology, in turn increasing the role of advanced control theory. This evolution has led to the introduction of optimization techniques to obtain superior systems.

Computing the inputs required to perform a specific task is a typical application of control theory. Such a task may be to steer the system from one state to another, so-called point-to-point problems, or to follow a desired trajectory for some or all states of the system, so-called path following problems. Additionally, one must trade off the performance of the system, whether it is execution time, energy or something else, for the actuation effort, leading to so-called optimal control problems.

The main focus of this thesis lies on solving the optimal point-to-point and optimal path following problem efficiently. Therefore, it is important to exploit the structure of the optimization problem, which is determined both by the mathematical structure of the system and the chosen parameterizations.

Linear systems exhibit the most common and simple structure. For such systems control theory is well established. However, due to the increasing complexity of systems, a linear system structure often falls short in describing the system dynamics accurately. So-called *differentially flat systems* are a generalization of linear systems and can model a much larger class of systems. Moreover, their mathematical structure is particularly well suited for the problems at hand. Polynomials constitute a well known parameterization for the control signals and can be constrained efficiently. However, they offer little flexibility and often result in poor performance. Piecewise polynomials on the other hand are equally efficient and offer more flexibility.

This research aims at developing efficient methodologies using both differential flatness and piecewise polynomials for solving optimal point-to-point motion problems and path following problems.

This chapter first introduces the reader to the concept of piecewise polynomials, followed by differential flatness and its application in motion planning problems, reviewing recent literature. Subsequently, an overview of the thesis and the contributions is provided. Finally, the notation used throughout the text is set.

1.1. PIECEWISE POLYNOMIALS

DEFINITION 1.1 · *Let $\xi = (\xi_0, \ldots, \xi_l)$ be a strictly increasing sequence of points. A function $s(x)$ defined on the finite interval $[\xi_0, \xi_l]$ is called a piecewise polynomial or* PP *function of degree $k \geq 0$ if $s(x)$ is a polynomial p_i of degree $\leq k$ on each interval $[\xi_i, \xi_{i+1})$:*

$$s(x) = p_i(x), \text{ if } \xi_i \leq x < \xi_{i+1},\ i = 0, 1, \ldots, l-1. \tag{1.1}$$

The points ξ are called the breaks *or* breakpoints *of s.*

The vector space of all PP functions of degree k and break sequence ξ is denoted by $\Pi_{k,\xi}$ with dimension $(k+1)l$, since each of the l polynomials has $k+1$ coefficients. The subspace $\Pi_{k,\xi,\nu}$ denotes the set of PP functions that satisfy the continuity conditions given by the sequence $\nu = (\nu_0, \ldots, \nu_l)$, where ν_i counts the number of continuity conditions required at ξ_i. In particular, $\nu_i = 0$ means that no continuity condition whatsoever is imposed at ξ_i. This thesis always takes $\nu_0 = \nu_l = 0$ such that the PP function vanishes outside $[\xi_0, \xi_l]$. The dimension of $\Pi_{k,\xi,\nu}$ is $g + k + 1$ with $g = (k+1)(l-1) - \sum_{i=0}^{l} \nu_i$.

Figure 1.1 illustrates a parabolic PP function ($k = 2$) with breakpoints $\xi = (0, 1, 3, 5, 6, 8, 9, 10)$. It features discontinuous jumps at the edges $x = 0$ and $x = 10$ as well as at the internal breaks $x = 1$ and $x = 9$. In the first derivative the PP function has discontinuous jumps at the sites $x = 3$ and $x = 9$. Finally the second derivative has jumps at $x = 4$ and $x = 5$. It follows that the continuity vector $\nu = (0, 0, 1, 2, 2, 1, 0, 0)$ and $g = 12$.

Chapter 2 details the definition of PP functions through B-splines and describes the most important properties used throughout the thesis.

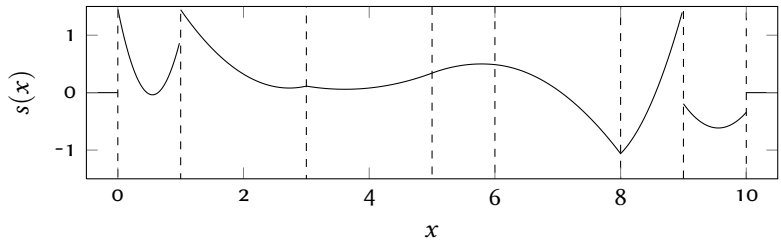

FIGURE 1.1.: A PP function of degree 2 and $l = 7$ with breakpoints $(0, 1, 3, 5, 6, 8, 9, 10)$

1.2. Differential flatness

The concept of differential flatness was initially introduced by Fliess, Lévine, et al. 1995. A system is said to be flat if there exists a set of outputs (equal to the number of inputs) such that all states and inputs can be determined from these outputs without integration.

Definition 1.2 (Differential flatness) · *A system*

$$\dot{x} = f(x, u) \tag{1.2}$$

with states $x \in \mathbb{R}^n$ and inputs $u \in \mathbb{R}^m$ is differentially flat if there exists a set of variables, the so-called flat outputs, $y \in \mathbb{R}^m$ of the form

$$y = g\left(x, u, \frac{\mathrm{d}u}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^q u}{\mathrm{d}t^q}\right)$$

such that

$$\begin{aligned} x &= \Phi\left(y, \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{r-1} y}{\mathrm{d}t^{r-1}}\right) \\ u &= \Psi\left(y, \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^r y}{\mathrm{d}t^r}\right), \end{aligned} \tag{1.3}$$

for some $q, r \in \mathbb{N}$.

Determining whether or not a system $\dot{x} = f(x, u)$ is flat remains an open problem. Fortunately, results exist for a wide variety of systems. The reader is referred to Martin, Murray, and Rouchon (2003) for an

overview and a catalog of flat systems. The following examples show some nonlinear mechanical systems that are flat. These systems will be used throughout the thesis.

Example 1.1 (Linear controllable systems) · *As differential flatness can be interpreted as a generalization of controllability for linear systems (Fliess, Lévine, et al. 1995), it is obvious that all linear controllable systems are flat. Consider a single-input single-output system with a (strictly proper) transfer function*

$$H(s) = \frac{b_0 + b_1 s + \ldots + b_{r-1} s^{r-1}}{a_0 + a_1 s + \ldots + s^r}.$$

This system has the following control canonical state space representation

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{r-1} \end{pmatrix} x + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} u,$$
$$z = \begin{pmatrix} b_0 & b_1 & \cdots & b_{r-1} \end{pmatrix} x,$$

with x the states and z the output. Obviously, a flat output $y = x_1$ due to the chain of integrators. Now, the input u and output z of the system can easily be written in terms of a flat output y as

$$u = a_0 y + a_1 \frac{\mathrm{d}y}{\mathrm{d}t} + \ldots + \frac{\mathrm{d}^r y}{\mathrm{d}t^r}$$
$$z = b_0 y + b_1 \frac{\mathrm{d}y}{\mathrm{d}t} + \ldots + b_{r-1} \frac{\mathrm{d}^{r-1} y}{\mathrm{d}t^{r-1}}.$$

Note that the flat output is not unique. Any multiple of y is also a flat output. It is convenient to scale the flat output by a factor $1/b_1$ such that the system output scales with the flat output when the system is at rest. For multiple-input multiple-output systems the interested reader is referred to Lévine and Nguyen (2003), Sira Ramírez and Agrawal (2004).

Example 1.2 (Overhead crane (Fliess, Lévine, et al. 1995)) · *Consider an overhead crane as displayed in Figure 1.2. A load with mass m is attached*

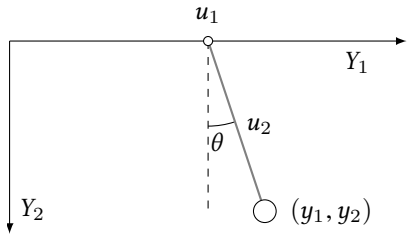

FIGURE 1.2.: On the overhead crane a load is attached to a moving trolley via a cable that can vary in length

via a cable to a moving trolley. Assume the trolley position and the cable length are the control variables. Neglecting damping, the equations of motion for the system are

$$m\ddot{y}_1 = -T \sin\theta$$
$$m\ddot{y}_2 = -T \cos\theta + mg$$
$$y_1 = u_2 \sin\theta + u_1$$
$$y_2 = u_2 \cos\theta,$$

where (y_1, y_2) is the coordinate of the load with respect to a reference frame Y_1Y_2, T the tension of the cable, θ the angle between the cable and the vertical axis, u_1 the trolley position and u_2 the cable length. From the first two equations we find

$$\tan\theta = \frac{\ddot{y}_1}{\ddot{y}_2 - g}.$$

Substituting θ in the fourth equation gives

$$u_2 = y_2 \sqrt{1 + \left(\frac{\ddot{y}_1}{\ddot{y}_2 - g}\right)^2}.$$

Then u_1 can be determined as

$$u_1 = y_1 - \frac{y_2 \ddot{y}_1}{\ddot{y}_2 - g}.$$

It is clear that the system inputs u_1, u_2 and the state θ can be determined from y_1, y_2 without integration. Hence the overhead crane is a differentially flat system.

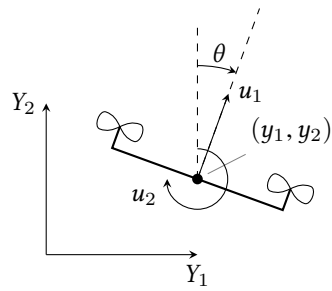

FIGURE 1.3.: Two-dimensional first principles quadrotor model

EXAMPLE 1.3 (QUADROTOR) · *Control of quadrotors and other unmanned aerial vehicles is an active research field nowadays. This example discusses flatness of a simplified first principles model in two dimensions from Hehn, Ritz, and D'Andrea (2012) as shown in Figure 1.3. It is controlled by two inputs, the thrust force u_1 and the pitch rate u_2. Furthermore it has three degrees of freedom, the horizontal and vertical position (y_1, y_2), with respect to the reference frame XY, and the angle θ.*

The equations of motion for the system are described by

$$\begin{aligned} m\ddot{y}_1 &= u_1 \sin\theta \\ m\ddot{y}_2 &= u_1 \cos\theta - g \\ \dot{\theta} &= u_2. \end{aligned}$$

Similar to the overhead crane, it is easy to see that

$$\begin{aligned} \tan\theta &= \frac{\ddot{y}_1}{\ddot{y}_2 + g} \\ u_1 &= m\sqrt{(\ddot{y}_2 + g)^2 + \ddot{y}_1^2} \\ u_2 &= \frac{\dddot{y}_1(\ddot{y}_2 + g) - \ddot{y}_1\dddot{y}_2}{(\ddot{y}_2 + g)^2 + \ddot{y}_1^2}. \end{aligned}$$

Hence, the two-dimensional first principles quadrotor model is differentially flat. The three-dimensional quadrotor model is also differentially flat with flat outputs the coordinate of the center of gravity and the yaw angle. A detailed derivation can be found in Mellinger and Kumar (2011).

1.3. Flatness and motion planning[1]

Differentially flat systems are particularly well suited for solving motion planning problems. Consider the problem of steering the system from an initial to a final state. Parameterizing the components of the flat output, $y_i, i = 1, \ldots, m$ as

$$y_i(t) = \sum_j A_{ij} b_j(t), \tag{1.4}$$

where $b_j(t)$ are basis functions, reduces the problem from finding a function in the infinite dimensional space of sufficiently smooth functions to finding a finite set of parameters A_{ij}.

For the point-to-point motion problem, given initial and final states, x_0 and x_T at times 0 and T, the corresponding values of the flat output and its derivatives are determined. To this end, the system of equations

$$\begin{aligned}
y_i(0) = \sum_j A_{ij} b_j(0),\ y_i(T) &= \sum_j A_{ij} b_j(T) \\
\frac{\mathrm{d}y_i}{\mathrm{d}t}(0) = \sum_j A_{ij} \frac{\mathrm{d}b_j}{\mathrm{d}t}(0),\ \frac{\mathrm{d}y_i}{\mathrm{d}t}(T) &= \sum_j A_{ij} \frac{\mathrm{d}b_j}{\mathrm{d}t}(T) \\
&\vdots \\
\frac{\mathrm{d}^{r-1}y_i}{\mathrm{d}t^{r-1}}(0) = \sum_j A_{ij} \frac{\mathrm{d}^{r-1}b_j}{\mathrm{d}t^{r-1}}(0),\ \frac{\mathrm{d}^{r-1}y_i}{\mathrm{d}t^{r-1}}(T) &= \sum_j A_{ij} \frac{\mathrm{d}^{r-1}b_j}{\mathrm{d}t^{r-1}}(T)
\end{aligned} \tag{1.5}$$

is solved for the coefficients A_{ij}. These equations are linear in the coefficients A. In fact, the coefficients A are required to be in an affine subspace defined by the above equations, implying that trajectory generation for flat systems can be reduced to simple algebra. When the system of equations is underdetermined, an optimal control problem is formulated to determine the best choice of coefficients.

[1] This section borrows many ideas from the excellent survey paper Martin, Murray, and Rouchon (2003)

Consider the standard optimal control problem

$$\begin{aligned} \underset{u(\cdot),x(\cdot)}{\text{minimize}} \quad & \int_0^T L(x(t),u(t))\,\mathrm{d}t \\ \text{subject to} \quad & \dot{x} = f(x,u) \\ & x(0) = x_0, x(T) = x_T, \end{aligned}$$

for given x_0, x_T and T. Assuming the system $\dot{x} = f(x,u)$ is flat, one can parameterize the flat output space as in (1.4) instead of discretizing the controls, resulting in the unconstrained [2] nonlinear program

$$\underset{y(\cdot)}{\text{minimize}} \quad \int_0^T L\left(\Phi\left(y, \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{r-1}y}{\mathrm{d}t^{r-1}}\right), \Psi\left(y, \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{r}y}{\mathrm{d}t^{r}}\right)\right)\mathrm{d}t,$$

where the flat output y is parameterized as in (1.4). Hence, numerical sensitivity issues during integration of the dynamics and the problem of satisfying $x(T) = x_T$ is eliminated. This methodology was first proposed in van Nieuwstadt and Murray (1998). In Fliess and Marquez (2000) it is considered in a predictive control setting and is subsequently applied on an industrial application in Petit, Creff, et al. (2001).

Adding state and input inequality constraints complicates the problem considerably. Consider a constraint $h(x(t),u(t)) \geq 0, \forall t \in [0,T]$. Classical constrained optimization approaches (Milam, Mushambi, and Murray 2000, Faiz, Agrawal, and Murray 2001, Louembet, Cazaurang, Zolghadri, et al. 2009, Petit, Milam, and Murray 2001) rely on collocation in which time sampling is used to impose the constraints. Therefore, in between time-samples constraints are not guaranteed to be satisfied such that a post-analysis is necessary. For linear systems several approaches have been proposed to ensure $h(x(t),u(t)) \geq 0$ for all $t \in [0,T]$. Henrion and Lasserre (2006) use a polynomial basis for the flat outputs and view the constrained motion planning problem as a polynomial nonnegativity problem requiring linear matrix inequalities. However, due to the limited degrees of freedom of a polynomial, the solution space is limited and higher degrees may lead to unwanted oscillations. Therefore, researchers have recently turned to piecewise polynomial functions (Louembet, Cazaurang, and Zolghadri 2010, Suryawan, De Doná, and Seron 2011, Suryawan,

[2] After eliminating the interpolation constraints (1.5)

De Doná, and Seron 2012). Louembet, Cazaurang, and Zolghadri (2010) generalize the approach of Henrion and Lasserre (2006) towards piecewise polynomial functions. The authors, however, do not take into account the finite support of the polynomial pieces and unwillingly, conservatism is introduced. Instead of resorting to linear matrix equalities, Suryawan, De Doná, and Seron (2011), Suryawan, De Doná, and Seron (2012) express the state and input constraints in a linear fashion based on the total positivity property of the basis functions. Naturally, these simplified constraints also introduce a possibly large amount of conservatism (de Boor and Daniel 1974).

For nonlinear systems, the point-to-point motion problem is much more challenging. To simplify the problem, it is often decoupled into a high level path planning stage in which a geometric trajectory is planned accounting only for geometric constraints, and a low level path following stage in which an optimal velocity profile along the path is determined taking into account system dynamics and limitations (Bobrow, Dubowsky, and Gibson 1985, Shin and McKay 1985). I will call this final step the *path following problem*. Raczy (1997), Raczy and Jacob (1999) were the first to recognize the applicability of path following problems to differentially flat systems. For these systems, the dynamics can be projected along the path onto a linear system, leading to a small dimensional optimization problem. These results were further generalized in Faulwasser, Hagenmeyer, and Findeisen (2011). For the specific model structure of a simplified robotic manipulator, Verscheure, Demeulenaere, et al. (2009) propose a problem transformation yielding a convex optimization problem, which can be solved reliably to global optimality.

1.4. Contributions and outline

This thesis aims at developing efficient methodologies using both differential flatness and piecewise polynomials for solving optimal point-to-point motion problems and path following problems. Figure 1.4 illustrates the relationship between the different themes that are treated in this thesis. Chapter 2 focuses on piecewise polynomials and develops both an exact and conservative method for imposing semi-infinite constraints. Chap-

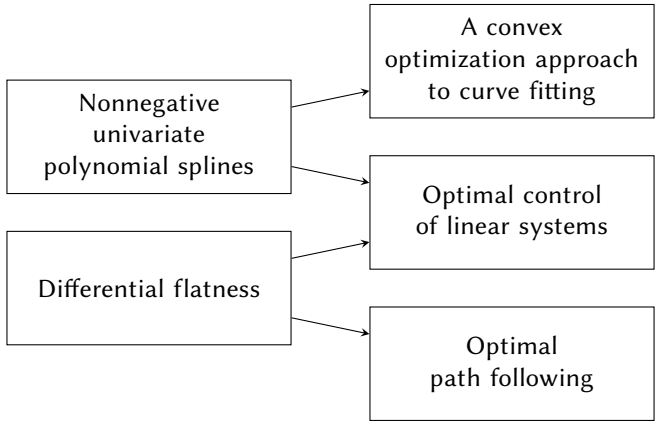

FIGURE 1.4.: The relationship between the different themes treated in this thesis

ter 3 briefly deviates from the central theme, i.e. optimal motion control, and uses the results from the previous chapter to focus on the spline approximation problem. Chapter 4 deals with optimal control problems for linear systems through differential flatness using PP functions. Chapter 5 continues with the path following problem for nonlinear differentially flat systems. A nonlinear change of variables is proposed, leading to an efficient problem formulation. Subsequently, the approach is extended towards path planning problems. The following subsections summarize the main contributions for each chapter.

1.4.1. Nonnegative univariate polynomial splines

Chapter 2 considers the problem of imposing a piecewise polynomial nonnegativity constraint $h(x) \geq 0$ for all x on the support of h. First, the basics of polynomial splines are covered and well-known results from polynomial optimization are reviewed. Then, based on recent results, an explicit and exact nonnegativity conditions for polynomial splines is derived. Furthermore, a novel linear relaxation method is presented. These linear conditions are easier to check but inevitably introduce conservatism.

1.4.2. A convex optimization approach to curve fitting

One challenge when optimizing splines is determining the locations of the breaks, which requires treating the breaks as variables resulting in a highly nonlinear and nonconvex optimization problem. Consequently, it is difficult to obtain and guarantee global optimality. Previously published results rely on a good initial estimate of the breaks and cannot incorporate prior knowledge straightforwardly. Instead of taking the breaks as variables, Chapter 3 follows an indirect approach by supplying many candidate breaks and using a regularization to favor solutions with few active breaks as proposed in Demeulenaere, Pipeleers, et al. (2009), resulting in a convex optimization problem, which can be solved efficiently to global optimality. By using a B-spline parameterization of the PP function numerical stability is improved. Moreover, by using an iterative reweighing procedure the sparsity is enhanced. Knowledge of the underlying function is also easily incorporated by adding constraints to the optimization problem.

1.4.3. Optimal control of linear systems through differential flatness

Inspired by Henrion and Lasserre (2006) and Louembet, Cazaurang, and Zolghadri (2010), Chapter 4 views optimal control problem as a nonnegativity problem with polynomial splines. Contrary to Suryawan, De Doná, and Seron (2011), Suryawan, De Doná, and Seron (2012), Louembet, Cazaurang, and Zolghadri (2010), exact solutions are found. Moreover, the linear relaxations, derived in Chapter 2 are able to approximate these solutions very closely at a low computational cost. For time-optimal motion problems, instead of resorting to classical binary search (Boyd and Vandenberghe 2004, Consolini and Piazzi 2009), a novel algorithm is adopted from Janssens et al. (2013b) that takes into account derivative information. Furthermore, by regarding the problem as a parametric linear program, solutions for the entire parameter range can be computed by solving only a limited set of problems. Moreover, by exploiting flatness, a robust counterpart for the robust input design problem for uncertain systems is derived. This way, the usual sampling of the uncertainty region as in Chew and Chuang (1995), De Caigny et al. (2008), De Caigny (2009) is avoided.

1.4.4. Optimal path following for differentially flat systems

Chapter 5 deals with the optimal path following problem for nonlinear differentially flat systems. For such systems, it has been shown that the projection of the dynamics along the geometric path onto a linear single-input system leads to a small dimensional optimal control problem (Raczy and Jacob 1999, Faulwasser, Hagenmeyer, and Findeisen 2011). Although the projection simplifies the problem to great extent, the resulting problem remains difficult to solve, in particular in the case of nonlinear system dynamics and time-optimal problems. Inspired by Verscheure, Demeulenaere, et al. (2009), this research proposes a nonlinear change of variables, using a time transformation, to arrive at a fixed end-time optimal control problem. Numerical simulations indicate that the proposed problem formulation is solved more efficiently particularly for highly nonlinear paths compared to results from literature. Although path following has many applications it is restrictive to constrain the output to follow a predetermined geometric path. Therefore, the proposed path following framework is extended to allow for some freedom in the geometric path. To this end, the geometric reference is represented as an unknown convex combination of two or more fixed boundary paths. In this way, optimal paths for differentially flat systems can be determined as well.

1.5. NOTATION

In the remainder of the text $\partial_\tau^k f(\tau)$ denotes the k-th derivative of $f(\tau)$ with respect to τ. For $k = 1$, $\partial_\tau f(\tau)$ is used. The shorthand notation $\dot{f}$, $\ddot{f}$ denotes the first and second derivative with respect to time. $a^\intercal$ denotes a matrix or vector transpose. Furthermore, $\langle \cdot, \cdot \rangle$ denotes the standard inner product: for vectors, $a, b \in \mathbb{R}^n$ $\langle a, b \rangle = \sum_{i=1}^n a_i b_i = a^\intercal b$ and for matrices, $A, B \in \mathbb{R}^{n \times m}$, $\langle A, B \rangle = \sum_{i=1}^n \sum_{j=1}^m A_{ij} B_{ij} = \operatorname{tr}(A^\intercal B)$. The space of symmetric $n \times n$ matrices is denoted by $\mathbb{S}^n$. For $A \in \mathbb{S}^n$, $A \succeq 0$ indicates that the matrix A is positive semidefinite.

What to remember

- A function is a piecewise polynomial of degree k if it is a polynomial of at most degree k on each of its pieces.
- A system is said to be differentially flat if a set of variables equal to the number of inputs exists such that all states and inputs can be determined from these variables without integration.
- Differentially flat systems are particularly well suited for solving optimal control problems since the integration of the differential equations can be avoided.

2

Nonnegative univariate polynomial splines

Many engineering problems require determining whether a univariate function $p(x)$ is nonnegative on a subset of $\mathbb{R}$, see e.g. Ben-Tal and Nemirovskiĭ (2001), Lasserre (2010). Usually, nonnegativity is determined by checking a finite set of samples from the subset. However, this method gives no guarantee whatsoever in between samples. In case $p(x)$ is polynomial, however, nonnegativity can be guaranteed by solving a linear matrix inequality (LMI). This is due to the equivalence between nonnegativity and the existence of a sum of squares (SOS) decomposition (Laurent 2009). However, despite their ease of use, polynomials can be inflexible on large intervals and often generate excessive oscillations when used for interpolation, especially for high order curves. Therefore, nonnegative piecewise polynomial (PP) functions have recently gained interest (Louembet, Cazaurang, and Zolghadri 2010, Plaumann 2010), offering more flexibility than traditional polynomials even for low degrees. Rather than naively imposing nonnegativity of the polynomial pieces, which can result in an unnecessarily large LMI due to unaccounted continuity conditions, a sum of squares (of PP functions) decomposition is searched for. To this end, the seminal work Nesterov (2000) is combined with a sum of squares decomposition for PP functions, recently derived in Plaumann (2010). An explicit degree for the piecewise polynomials in the sum of squares decomposition from Plaumann (2010) is derived. Furthermore, it is conjectured that examining nonnegativity of a PP function only requires the solution to a sparse LMI instead of a dense one. As solving large LMI's remains computationally challenging to date, a linear sufficient condition for nonnegativity is determined and a series of linear relaxations is derived.

The chapter first reviews the basics on basis splines. The second section summarizes well-known results on nonnegativity conditions for univari-

ate polynomials. These results are generalized in the following section on piecewise polynomial nonnegativity. Both exact semidefinite conditions and a novel linear relaxation method are described.

2.1. BASIS SPLINES

Basis splines or B-splines are commonly used as a basis for PP functions in $\Pi_{k,\xi,\nu}$ (de Boor 2001). Given a nondecreasing knot sequence

$$\lambda = (\lambda_0, \ldots, \lambda_{g+2(k+1)}),$$

the normalized B-spline function of degree k, defined on $[\lambda_i, \lambda_{i+k+1}]$, is computed using the Cox-de Boor recursive formula (de Boor 2001)

$$b_{i,k,\lambda}(x) = \frac{x - \lambda_i}{\lambda_{i+k} - \lambda_i} b_{i,k-1,\lambda}(x) + \frac{\lambda_{i+k+1} - x}{\lambda_{i+k+1} - \lambda_{i+1}} b_{i+1,k-1,\lambda}(x), \tag{2.1}$$

starting with

$$b_{i,0,\lambda}(x) = \begin{cases} 1, \text{if } x \in [\lambda_i, \lambda_{i+1}), \\ 0, \text{if } x \notin [\lambda_i, \lambda_{i+1}). \end{cases} \tag{2.2}$$

For the space $\Pi_{k,\xi,\nu}$ the knot sequence λ can be constructed by repeating ξ_i exactly $k + 1 - \nu_i$ times in λ. As there are $l - 1$ internal breaks, g can then be interpreted as the number of internal knots, i.e. λ_i that do not lie on the boundary of the spline's support, in λ.

EXAMPLE 2.1 (BASIS SPLINES) · *Consider a parabolic spline ($k = 2$) with break sequence $\xi = (0, 1, 3, 6)$ and continuity requirements $\nu = (0, 2, 1, 0)$. Hence, the knot sequence $\lambda = (0, 0, 0, 1, 3, 3, 6, 6, 6)$ and $g = 3$. The six corresponding B-splines are plotted in Figure 2.1. Each of the B-splines is piecewise parabolic and at the breaks they clearly exhibit a discontinuity in either the function itself or its derivatives. The function $b_{0,k,\lambda}$ is discontinuous at $x = 0$, and $b_{1,k,\lambda}, b_{2,k,\lambda}, b_{3,k,\lambda}$ and $b_{4,k,\lambda}$ have a discontinuous first derivative at the sites $x = 0, 3$ and 6. Note that the discontinuity is tied to the multiplicity of the breakpoint in the defining knot sequence, e.g. the defining knot sequence for $b_{3,k,\lambda}$ is $(1, 3, 3, 6)$, which explains the discontinuous first derivative at $x = 3$.*

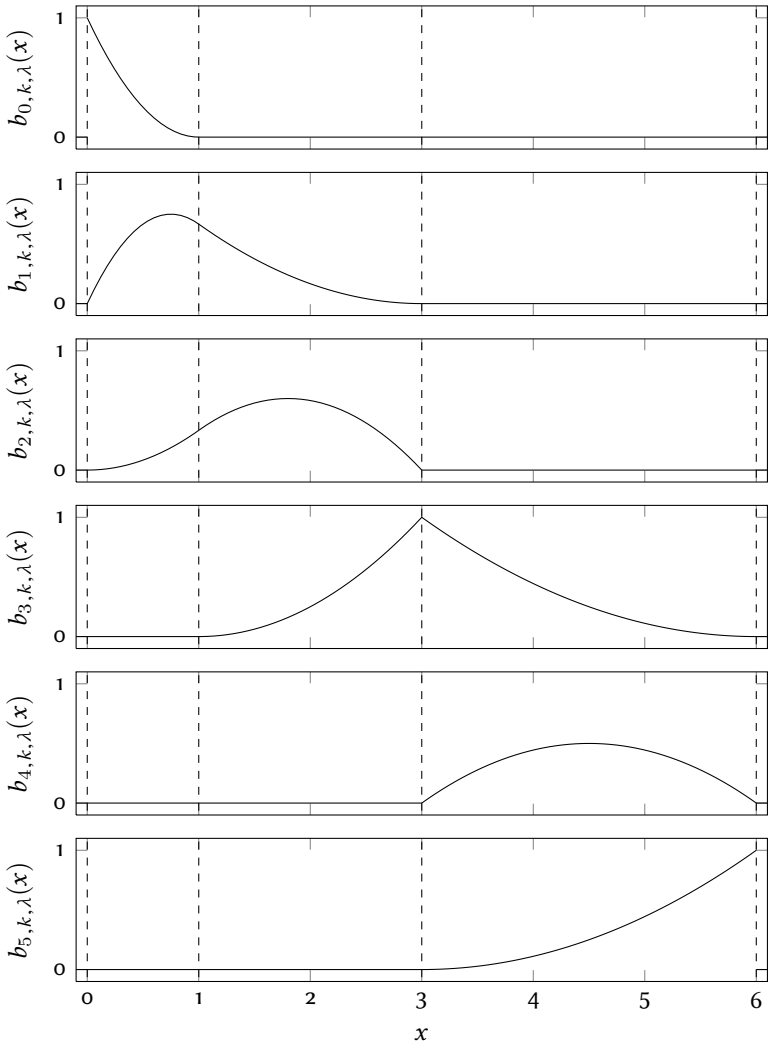

FIGURE 2.1.: Parabolic B-splines for the knot sequence $\lambda = (0, 0, 0, 1, 3, 3, 6, 6, 6)$

B-spline functions exhibit many interesting properties. In particular the positivity and partition of unity properties are of interest with regard to nonnegativity. Both properties can be discerned in Figure 2.1.

PROPERTY 2.1 (SUPPORT AND POSITIVITY) · *The basis function $b_{i,k,\lambda}$ is a PP function of degree $k+1$ with breaks $\lambda_i, \ldots, \lambda_{i+k+1}$, vanishes outside $[\lambda_i, \lambda_{i+k+1})$, and is positive on the interior of that interval.*

PROPERTY 2.2 (PARTITION OF UNITY) · *The basis $b_{k,\lambda}$ provides a partition of unity*

$$\sum_{i=0}^{g+k} b_{i,k,\lambda} = 1 \text{ on } [\lambda_0, \lambda_{g+k+1}).$$

Having defined the B-spline basis, every PP function s can be represented as a linear combination of B-spline basis functions

$$s = \sum_{i=0}^{g+k} c_i b_{i,k,\lambda} = \langle c, b_{k,\lambda} \rangle, \tag{2.3}$$

where $b_{k,\lambda}$ denotes the B-spline basis of degree k with knot sequence λ. The B-spline coefficients c are often called the *control points* of the spline and the *control polygon* is defined as the piecewise linear interpolant to the points (λ_i^*, c_i), where λ_i^* is the i-th knot average $\lambda_i^* = 1/k \sum_{j=i+1}^{i+k} \lambda_j$. In Figure 2.5 the thick black line represents the control polygon. A function, represented as a linear combination of B-splines, is called a *polynomial spline function* or a *spline*.

It follows from properties 2.1 and 2.2 that

PROPERTY 2.3 (CONVEX HULL) · *A spline is always contained in the convex hull of the control points. Hence, a scalar spline function is contained within the minimum and maximum value of its coefficients:*

$$s(x) = \langle c, b_{k,\lambda}(x) \rangle \in [\min(c), \max(c)], \forall x \in [\xi_0, \xi_l].$$

The convex hull property will prove very useful in Section 2.3.2.

The derivative of a spline is itself a spline. Its coefficients can be determined using following property:

Property 2.4 (Differentiation) · *The m-th derivative of a degree k spline s is itself a spline of degree k − m*

$$\partial_x^m s(x) = \prod_{i=1}^{m} (k+1-i) \sum_i c_i^{(m)} b_{i,k+1-m,\mu}(x)\,,$$

where

$$c_j^{(i)} = \begin{cases} c_j\,, & \text{if } i = 0\,, \\ \frac{c_j^{(i-1)} - c_{j-1}^{(i-1)}}{\lambda_{j+k+1-i} - \lambda_j} & \text{if } i > 0\,. \end{cases} \tag{2.4}$$

and the knot sequence μ is obtained by reducing the continuity with one over the breaks.

Property 2.5 (Addition) · *The sum of two pp functions in the space* Π_{k^1,ξ^1,v^1} *and* Π_{k^2,ξ^2,v^2} *is a pp function in the space* $\Pi_{k,\xi,v}$*, where* $k = \max(k^1, k^2)$*,* $\xi = \xi^1 \cup \xi^2$ *and*

$$v_i = \begin{cases} \min(v_n^1, v_m^2) & \text{if } \xi_n^1 = \xi_i, \xi_m^2 = \xi_i \text{ for some } n, m \\ v_n^1 & \text{if } \xi_n^1 = \xi_i \text{ for some } n \\ v_m^2 & \text{if } \xi_m^2 = \xi_i \text{ for some } m \end{cases}$$

Example 2.2 · *Consider two splines with degrees, breaks and continuity sequence* $k^1 = 3$*,* $\xi^1 = (0,1,2,4)$*,* $v^1 = (0,3,3,0)$ *and* $k^2 = 2$*,* $\xi^2 = (0,2,3,4)$*,* $v^2 = (0,2,2,0)$*. The sum of these splines has* $k = 3, \xi = (0,1,2,3,4)$ *and* $v = (0,3,2,2,0)$*.*

2.2. Nonnegative univariate polynomials

Before addressing nonnegative polynomial splines, let us first summarize the basics of nonnegative univariate polynomials. The material in this section is based on the survey paper Laurent (2009), to which the reader is referred for details and a more comprehensive study. Two methods for asserting nonnegativity are discussed. The first method is based on a sum of squares decomposition of the polynomial and yields exact results in the univariate case. The second method describes a series of linear, though conservative, relaxations.

2.2.1. *Sum of squares*

A crucial concept in this chapter is the notion of *sum of squares*. A polynomial p is said to be sum of squares of polynomials or SOS if p can be written as $p = \sum_{i=1}^{m} q_i^2$ for some polynomials $q_1, \ldots, q_m$. We are interested in determining the conditions for which p is nonnegative on a subset S of $\mathbb{R}$.

Obviously, a polynomial which is SOS is nonnegative on $\mathbb{R}$. Luckily, for univariate polynomials the converse holds as well:

LEMMA 2.1 · *Any nonnegative univariate polynomial is a sum of two squares*

Proof. See Laurent (2009). □

It is possible to determine whether a polynomial is SOS by verifying the existence of a positive definite matrix. This result was discovered independently by several authors e.g. Choi, Lam, and Reznick (1995), Powers and Wörmann (1998).

LEMMA 2.2 · *Let $p(x) = \sum_{i=0}^{2d} p_i x^i$ be a polynomial of degree $2d$. The following assertions are equivalent.*

1. *$p(x)$ is SOS*
2. *The following system in the matrix variable $X \in \mathbb{S}^{d+1}$ is feasible*

$$X \succeq 0, \quad \sum_{j,k \mid j+k=i} X_{j,k} = p_i. \tag{2.5}$$

Proof. Let $v(x) = (1, x, x^2, \ldots, x^d)^\intercal$ and define polynomials

$$q_i(x) = \langle q_i, v(x) \rangle.$$

Thus $\sum_i q_i(x)^2 = v(x)^\intercal \left(\sum_i q_i q_i^\intercal \right) v(x)^\intercal$. Therefore $p(x)$ is a sum of squares of polynomials if and only if $p(x) = v(x)^\intercal X v(x)$ for some positive semidefinite matrix X. By equating the coefficients of both polynomials $p(x)$ and $v(x)^\intercal X v(x)$, the system (2.5) is found. □

Example 2.3 · *Consider the polynomial* $p(x) = 4 + 4x - 3x^2 - 2x^3 + x^4$. *As* $p(x)$ *is a quartic polynomial, an* $X \succeq 0$ *satisfying*

$$p(x) = \begin{pmatrix} 1 & x & x^2 \end{pmatrix} \underbrace{\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}}_{X} \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

must be determined. By equating the coefficients, it can be seen that $X = \begin{pmatrix} 4 & 2 & c \\ 2 & -3-2c & -1 \\ c & -1 & 1 \end{pmatrix}$. *Therefore* $X \succeq 0$, *if and only if* $-2 \leq c \leq 2$. *For* $c = -2$,

$$X = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 2 & 1 & -1 \end{pmatrix},$$

which corresponds to the decomposition $p(x) = (2 + x - x^2)^2$.

2.2.2. Nonnegative polynomials on an interval

Often, you are only interested in determining whether a polynomial $p(x)$ is nonnegative on a $S \subseteq \mathbb{R}$. In this case the sos condition on $p(x)$ has to be relaxed. The theorems below are presented without proof. The interested reader is referred to Laurent (2009), Powers and Reznick (2000) for proofs and a more detailed treatment.

Theorem 2.1 (Pólya-Szegö) · *If* $p(x) \geq 0, \forall x \geq 0$ *then,* $p = f + xg$, *where* f *and* g *are* sos *and* $\deg(f), \deg(xg) \leq \deg(p)$.

Theorem 2.2 (Fekete, Markov-Lukácz) · *If* $p(x) \geq 0, \forall x\colon -1 \leq x \leq 1$ *then,*

1. $p = f + g(1-x^2)$, *where* f *and* g *are* sos *and* $\deg(f), \deg(g) \leq \deg(p)$ *(respectively* $\deg(p) + 1$*) if* $\deg(p)$ *is even (respectively odd).*

2. *If* $\deg(p)$ *is odd, then* $p = f(1 + x) + g(1 - x)$, *where* f *and* g *are* sos *and* $\deg(f(1 + x)), \deg(g(1 - x)) \leq \deg(p)$.

EXAMPLE 2.4 · *Consider the polynomial* $p(x) = 4 + 5x - 5x^2 - x^3 + x^4$ *on* $x \geq 0$. *To establish* $p(x) \geq 0, \forall x \geq 0$, *an* $X, Y \geq 0$ *and satisfying*

$$p(x) = \begin{pmatrix} 1 & x & x^2 \end{pmatrix} \underbrace{\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}}_{X} \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} + x \begin{pmatrix} 1 & x \end{pmatrix} \underbrace{\begin{pmatrix} g & h \\ h & i \end{pmatrix}}_{Y} \begin{pmatrix} 1 \\ x \end{pmatrix}$$

has to be found. Choose for example $X = \begin{pmatrix} 4 & 2 & -2 \\ 2 & 1 & -1 \\ -2 & -1 & 1 \end{pmatrix}$, $Y = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$, *which corresponds to the decomposition* $p(x) = (2 + x - x)^2 + x(1 - x)^2$.

2.2.3. Linear relaxations

As the degree of the polynomial increases, the matrix inequalities related to the nonnegativity conditions quickly grow in dimension and become increasingly difficult to solve. The necessary and sufficient semidefinite conditions for nonnegativity can then be replaced by linear sufficient conditions that are easier to solve. Naturally, the decrease in computational demand comes at the cost of added conservatism to the problem. For polynomial nonnegativity problems *Pólya's relaxation* is often employed (Pólya 1928, Laurent 2009). As this research only consider univariate polynomials, a dehomogenized version of Pólya's theorem is adopted, which can be attributed to Bernstein (1915).

THEOREM 2.3 · *If a univariate polynomial* $p > 0$ *on* $[a, b]$, *then* p *can be written as a positive linear combination of polynomials* $(x - a)^i (b - x)^j$ *for suitable integers* i *and* j.

This essentially means that any positive polynomial can be written as $\sum_{i=0}^{d} c_i (x - a)^i (b - x)^{d-i}$, with $c_i \geq 0$. Note that no bound on the degree d is given.

For polynomial optimization the inverse of Pólya's theorem is often used, i.e. a positive linear combination of polynomials $(x - a)^i (b - x)^j$ means positivity on $[a, b]$. By applying Pólya's relaxation, it is meant replacing the polynomial inequality by an inequality on the coefficients of $p(x) = \sum_{i=0}^{d} c_i (x - a)^i (b - x)^{d-i}$ for a fixed value of d.

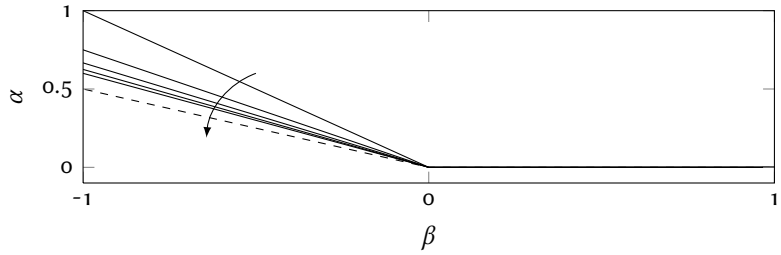

FIGURE 2.2.: Pólya's relaxations of $\alpha x^2 + \beta x(1-x) + \alpha(1-x)^2 > 0$ for degrees $d = 1, 3, 5, 7, 9$. The true bound is indicated by the dashed line.

EXAMPLE 2.5 · *Consider the polynomial* $p(x) = \alpha x^2 + \beta x(1-x) + \alpha(1-x)^2$ *on* $x \in [0, 1]$. *From Pólya's relaxation a sufficient condition for positivity of* $p(x)$ *is* $\alpha, \beta > 0$. *By multiplying the polynomial with* $x + (1-x)$, *one obtains*

$$p(x) = \alpha x^3 + (\alpha + \beta)x^2(1-x) + (\alpha + \beta)x(1-x)^2 + \alpha(1-x)^2$$

and the positivity condition $\alpha, \alpha + \beta > 0$, *which is less conservative. By repeatedly multiplying* p *with* $x + (1-x)$, *the constraints on the coefficients converge to the correct conditions, although it may require the degree* d *to go to infinity. Figure 2.2 illustrates the conditions for increasing* d. *Pólya's relaxation approaches for increasing degree the correct constraints* $\alpha > 0$ *and* $\alpha \geq -\beta/2$.

2.3. Nonnegative continuous univariate polynomial splines

Having covered nonnegative univariate polynomials and defined polynomial splines, we are now ready to tackle polynomial spline inequalities of the form $s(x) = \langle c, b_{k,\lambda}(x) \rangle \geq 0$ for $x \in [\lambda_0, \lambda_{g+k+1})$. As in the discussion on polynomial nonnegativity, both an exact semidefinite and a conservative linear method is determined.

2.3.1. *Sum of squares*

Rather than naively imposing nonnegativity of the polynomial pieces, which results in an unnecessarily large LMI due to unaccounted continuity conditions, a sum of squares decomposition is searched for. In Louembet, Cazaurang, and Zolghadri (2010) nonnegative PP functions were studied using the sum of squares formalism. However, as the finite support of the piecewise polynomial basis was not taken into account, the authors only derived sufficient conditions. By providing explicit necessary and sufficient conditions for nonnegative piecewise polynomials, the results in Louembet, Cazaurang, and Zolghadri (2010) are extended. To this end, a sum of squares decomposition for PP functions, which was recently derived in Plaumann (2010), is used.

THEOREM 2.4 · *Let μ denote the knot sequence with breakpoints $\xi = (\xi_0, \dots, \xi_l)$ and continuity sequence $\nu = (0, 1, 1, \dots, 1, 1, 0)$. The continuous PP function s with break sequence ξ is nonnegative on its support if and only if there exist sums of squares f_i of continuous PP functions such that*

$$s = f_0 + \sum_{i=0}^{l-1} b_{i,1,\mu} b_{i+1,1,\mu} f_{i+1}. \tag{2.6}$$

Note the resemblance with Theorem 2.2. In Plaumann (2010) only week degree bounds are given and the continuity of f_0 remains unclear. The following theorem fills in these voids.

THEOREM 2.5 · *The univariate continuous PP function $s = \langle c, b_{2d,\lambda} \rangle$ is nonnegative if and only if there exist matrices $Y_0 \in \mathbb{S}^{dl+1}$ and $Y_i \in \mathbb{S}^d$, $i = 1, \dots, l$, such that $Y_j \succeq 0, j = 0, \dots, l$ and*

$$s = b_{d,\mu}^\mathsf{T} Y_0 b_{d,\mu} + \sum_{i=0}^{l-1} b_{i,1,\mu} b_{i+1,1,\mu} b_{d-1,\mu}^\mathsf{T} Y_{i+1} b_{d-1,\mu}.$$

Proof. The degree of the spline basis follows from the application of Theorem 2.2 on the polynomial pieces of $s(x)$

$$s(x) = \begin{cases} g_1(x) + (x - \xi_0)(\xi_1 - x) f_1(x) & \text{if } x \in [\xi_0, \xi_1), \\ \vdots & \vdots \\ g_l(x) + (x - \xi_{l-1})(\xi_l - x) f_l(x) & \text{if } x \in [\xi_{l-1}, \xi_l), \end{cases}$$

where g_i and f_i are sum of squares of polynomials of degree at most d and $d-1$ respectively.

Equivalently

$$s(x) = f_0(x) + \sum_{i=0}^{l-1} b_{i,1,\mu}(x) b_{i+1,1,\mu}(x) f_{i+1}(x),$$

with $f_0(x) = \sum_{i=1}^{l} g_i(x)|_{x\in[\xi_i,\xi_{i+1})}$, a continuous PP function degree $\leq d$. The continuity of f_0 follows from the continuity of s since $(x-\xi_i)(\xi_{i+1}-x)f_i(x) = 0$ for $x = \xi_i, \xi_{i+1}$.

Furthermore, each $g_i(x) = h_i(x)^2 + l_i(x)^2$ can be written as a sum of two squares (cfr. Theorem 2.1). Define $\tilde{h}_i(x) = h_i(x)\sin\theta_i + l_i(x)\cos\theta_i$ and $\tilde{l}_i(x) = h_i(x)\cos\theta_i - l_i(x)\sin\theta_i$, such that $g_i(x) = \tilde{h}_i(x)^2 + \tilde{l}_i(x)^2$. Due to the continuity of $f_0(x)$, $g_i(\xi_i) = g_{i+1}(\xi_i)$. Therefore, there always exist a rotation θ_i such that $\tilde{h}_i(\xi_i) = h_{i+1}(\xi_i)$ and $\tilde{l}_i(\xi_i) = l_{i+1}(\xi_i)$. This shows that f_0 is also a sum of squares of *continuous* PP functions. □

Note that the f_i for $i = 1,\ldots,l$ are SOS of polynomials. Therefore, it is not necessary to use the complete basis $b_{d-1,\mu}$ in Theorem 2.5. The B-splines $b_{i(d-1),d-1,\mu}$ up to $b_{(i+1)(d-1),d-1,\mu}$ are sufficient to cover a degree $d-1$ polynomial basis over the knots $[\xi_i,\xi_{i+1}]$. For convenience, the notation from Theorem 2.5 will be used in the following.

With the necessary theorems in place, the following provides a convenient way to compute the matrices Y_i. First, it is convenient to express all PP functions in the same basis. Due to the Curry-Schoenberg theorem (de Boor 2001), $b_{2d,\mu}$ is a basis for all continuous PP functions of degree $\leq 2d$. Now, the spline s is reformulated as

$$s = \langle c, b_{2d,\lambda}\rangle = \langle \mathcal{T}c, b_{2d,\mu}\rangle,$$

where $\mathcal{T}$ is a linear transformation matrix.

Inspired by the seminal work of Nesterov (2000), the vector coefficient $l_{jk} \in \mathbb{R}^{2dl+1}$ are defined as

$$b_{j,d,\mu} b_{k,d,\mu} = \langle l_{jk}, b_{2d,\mu}\rangle.$$

Furthermore, the matrix valued linear operators $\Lambda_0(b_{2d,\mu}) : \mathbb{R}^{2dl+1} \mapsto \mathbb{S}^{dl+1}$ are defined, such that

$$\Lambda_0(b_{2d,\mu})^{(jk)} = \langle l_{jk}, b_{2d,\mu} \rangle.$$

Now, it holds that

$$b_{d,\mu} b_{d,\mu}^\intercal = \Lambda_0(b_{2d,\mu}),$$

such that $b_{d,\mu}^\intercal Y_0 b_{d,\mu} = \langle Y_0, b_{d,\mu} b_{d,\mu}^\intercal \rangle = \langle Y_0, \Lambda_0(b_{2d,\mu}) \rangle$.

In a similar fashion $\Lambda_i(b_{2d,\mu}) : \mathbb{R}^{2dl+1} \mapsto \mathbb{S}^d$ are defined as

$$b_{i,1,\mu} b_{i+1,1,\mu} b_{d-1,\mu} b_{d-1,\mu}^\intercal = \Lambda_{i+1}(b_{2d,\mu}).$$

The adjoint operator $\Lambda_0^*(Y) : \mathbb{S}^{dl+1} \mapsto \mathbb{R}^{2dl+1}$ is defined by:

$$\langle Y_0, \Lambda_0(b_{2d,\mu}) \rangle = \langle \Lambda_0^*(Y_0), b_{2d,\mu} \rangle, \forall Y_0 \in \mathbb{S}^{dl+1}$$

and similarly for the other Λ_i.

Then Theorem 2.5 can be reformulated as

THEOREM 2.6 · *The piecewise polynomial $s = \langle c, b_{2d,\lambda} \rangle$ is nonnegative on its support if and only if there exist matrices $Y_i \succeq 0$ with $Y_0 \in \mathbb{S}^{dl+1}$ and the remaining $Y_i \in \mathbb{S}^d$ such that*

$$\mathcal{T} c = \sum_{i=0}^{l} \Lambda_i^*(Y_i).$$

EXAMPLE 2.6 · *Let's illustrate the developed theory by means of a simple example with two polynomial pieces and provide a comparison with Louembet, Cazaurang, and Zolghadri (2010) and with the traditional polynomial nonnegativity Theorem 2.2.*

Consider the continuous degree 2 piecewise polynomial $s(x) = \langle c, b_{2,\lambda}(x) \rangle$ with knot sequence $\lambda = (0, 0, 0, 1, 1, 2, 2, 2)$. We are interested for which c the piecewise polynomial $s(x)$ is nonnegative.

First the bases $b_{2,\mu}(x)$ and $b_{1,\mu}(x)$ with continuity requirement one over the internal breakpoints are determined. These are illustrated in Figure 2.3.

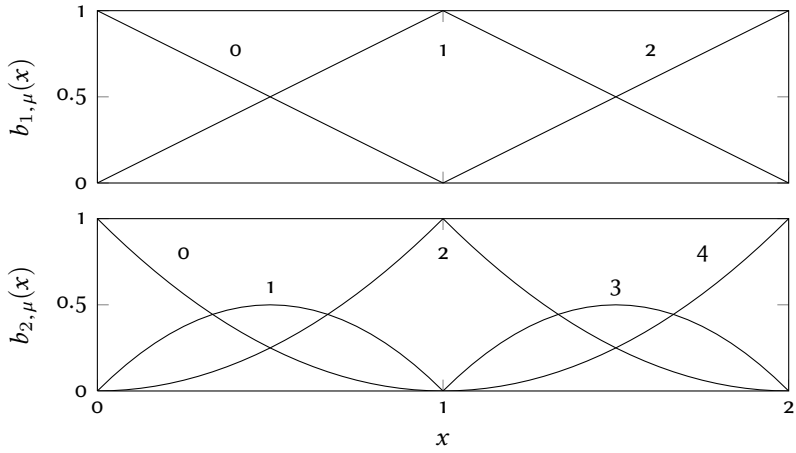

FIGURE 2.3.: The bases $b_{1,\mu}(x)$ and $b_{2,\mu}(x)$. Note the discontinuity in the first derivative at the internal breaks

Then the matrix valued linear operators Λ_i are determined:

$$\begin{pmatrix} b_{0,1,\mu}^2 & b_{0,1,\mu}b_{1,1,\mu} & 0 \\ b_{1,1,\mu}b_{0,1,\mu} & b_{1,1,\mu}^2 & b_{1,1,\mu}b_{2,1,\mu} \\ 0 & b_{2,1,\mu}b_{1,1,\mu} & b_{2,1,\mu}^2 \end{pmatrix}$$
$$= \begin{pmatrix} b_{0,2,\mu} & b_{1,2,\mu}/2 & 0 \\ b_{1,2,\mu}/2 & b_{2,2,\mu} & b_{3,2,\mu}/2 \\ 0 & b_{3,2,\mu}/2 & b_{4,2,\mu} \end{pmatrix} = \Lambda_0(b_{2,\mu}),$$

and

$$b_{0,1,\mu}b_{1,1,\mu}I_{[0,1]} = b_{1,2,\mu}/2 = \Lambda_1(b_{2,\mu}),$$
$$b_{1,1,\mu}b_{2,1,\mu}I_{[1,2]} = b_{3,2,\mu}/2 = \Lambda_2(b_{2,\mu}),$$

where the indicator function $I_{[\xi_i,\xi_{i+1}]} = 1$ for $t \in [\xi_i, \xi_{i+1}]$ and zero otherwise.

Now, the adjoints $\Lambda_0^*(Y)$ *and* $\Lambda_i^*(z_i)$ *are found as*

$$\begin{aligned}\Lambda_0^*(Y) &= (Y_{11}, Y_{12}, Y_{22}, Y_{23}, Y_{33})^\intercal,\\ \Lambda_1^*(z_1) &= (0, z_1/2, 0, 0, 0)^\intercal,\\ \Lambda_2^*(z_2) &= (0, 0, 0, z_2/2, 0)^\intercal.\end{aligned}$$

In this example, the transformation matrix $\mathcal{T}$ *is the identity matrix.*

We conclude that $s(x) \geq 0$ *for* $x \in [0, 2]$ *if and only if there exists* $Y \succeq 0 \in \mathbb{R}^{3\times 3}$ *and* $z_i \geq 0 \in \mathbb{R}, i = 1, 2$ *such that* $c = \Lambda_0^*(Y) + \sum_{i=1}^{2} \Lambda_i^*(z_i)$. *By eliminating the equality constraints, the non-negativity conditions*

$$\begin{pmatrix} c_0 & c_1 - z_1/2 & Y_{13} \\ c_1 - z_1/2 & c_2 & c_3 - z_2/2 \\ Y_{13} & c_3 - z_2/2 & c_4 \end{pmatrix} \succeq 0$$

and

$$z_1, z_2 \geq 0$$

are obtained.

Using polynomial positivity theory on both pieces one would find the equivalent yet larger LMI

$$\begin{pmatrix} c_0 & c_1 - z_1/2 & 0 & 0 \\ c_1 - z_1/2 & c_2 & 0 & 0 \\ 0 & 0 & c_2 & c_3 - z_2/2 \\ 0 & 0 & c_3 - z_2/2 & c_4 \end{pmatrix} \succeq 0$$

and

$$z_1, z_2 \geq 0,$$

whereas applying results from Louembet, Cazaurang, and Zolghadri (2010) yields

$$\begin{pmatrix} c_0 & c_1 & y \\ c_1 & c_2 & c_3 \\ y & c_3 & c_4 \end{pmatrix} \succeq 0.$$

Clearly this last result is conservative as for example $c = (1, 2, 1, 2, 1)$ *would be excluded from the feasible set.*

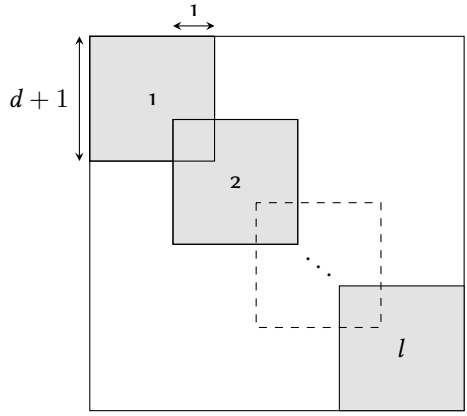

FIGURE 2.4.: Matrix structure of Y_0 for a fully continuous spline

Note that for a large number of breakpoints the matrix Y_0 grows large, which poses computational problems when solving the LMI, especially when matrices are dense. However, it appears that depending on the degree of continuity of $s(x)$ a specific sparsity pattern can be imposed on Y_0.

CONJECTURE 2.1 · *Consider a degree $2d$ spline with continuity $2d$ over all the internal breaks. Then the matrix Y_0 from Theorem 2.5 is a block matrix in which the first and last element of each block overlap, as illustrated in Figure 2.4.*

EXAMPLE 2.7 · *Consider a degree two spline with $\lambda = (0, 0, 0, 1, 2, 2, 2)$. Applying Theorem 2.6 results in the nonnegativity conditions*

$$\begin{pmatrix} c_0 & c_1 - t_1 & y \\ c_1 - t_1 & \frac{c_1+c_2}{2} & c_2 - t_2 \\ y & c_2 - t_2 & c_3 \end{pmatrix} \geq 0 \tag{2.7}$$

and $t_1, t_2 \geq 0$.

Conjecture 2.1 claims that $y = 0$, which can be verified when looking at the nonnegativity conditions on the individual intervals:

$$\begin{pmatrix} c_0 & c_1 - t_1 \\ c_1 - t_1 & \frac{c_1+c_2}{2} \end{pmatrix} \geq 0$$
$$\begin{pmatrix} \frac{c_1+c_2}{2} & c_2 - t_2 \\ c_2 - t_2 & c_3 \end{pmatrix} \geq 0$$

and $t_1, t_2 \geq 0$. To see that the above conditions yield (2.7) with $y = 0$, first consider $c_1 \geq 0$. In this case, simply take $t_1 = c_1$ such that (2.7) can be decoupled. If $c_1 < 0$, we take $t_1 = 0$. If $c_2 < 0$ the LMI (2.7) can no longer be decoupled. However, this would render the LMI infeasible as the diagonal element $\frac{c_1+c_2}{2} < 0$.

Up to now, the discussion focused on even degree splines. For odd degree polynomial splines the following corollary from Theorem 2.4 can be derived.

COROLLARY 2.1 · *A continuous piecewise polynomial s of odd degree is nonnegative on its domain if and only if there exist sum of squares g_i of continuous PP functions such that*

$$s = \sum_{i=1}^{l+1} b_{i,1,\mu} g_i. \tag{2.8}$$

Proof. Substituting the identity

$$b_{i,1,\mu} = b_{i,1,\mu}^2 + b_{i-1,1,\mu} b_{i,1,\mu} + b_{i,1,\mu} b_{i+1,1,\mu}$$

in (2.8) results in (2.6).

On the other hand, multiplying (2.6) with $\sum b_{1,\mu} = 1$, results in

$$f_0 \sum_{i=1}^{l+1} b_{i,1,\mu} + \sum_{i=1}^{l} \left(b_{i,1,\mu} b_{i+1,1,\mu}^2 + b_{i,1,\mu}^2 b_{i+1,1,\mu} \right) f_i \,.$$

where the property $b_{i,1,\mu} b_{j,1,\mu} b_{k,1,\mu} = 0$ for distinct i, j, k is used. It is readily verified that this result is of the form (2.8). □

Similarly to before the operators $\Lambda_i(b_{2d+1,\mu}) : \mathbb{R}^{2dl+2d+1} \mapsto \mathbb{S}^{d+2}$ are defined as

$$b_{i,1,\mu} b_{d,\mu} b_{d,\mu}^\intercal = \Lambda_i(b_{2d+1,\mu}).$$

THEOREM 2.7 · *A piecewise polynomial $s = \langle c, b_{2d+1,\lambda} \rangle$ is nonnegative if and only if there exist matrices $Y_i \in \mathbb{S}^{d+2}$ such that $Y_i \succeq 0$, $i = 0, \ldots, l$ and*

$$Tc = \sum_{i=0}^{l} \Lambda_i^*(Y_i).$$

2.3.2. Linear relaxations

As shown in the previous section, the LMI's can quickly grow in dimension for increasing number of knots and degree. Therefore, linear sufficient conditions similar to Pólya's relaxation for polynomial nonnegativity problems are very interesting from a computational point of view.

An obvious sufficient condition for nonnegativity follows from Property 2.1. Since all B-spline basis functions are nonnegative,

$$s(x) = \langle c, b_{k,\lambda}(x) \rangle \geq 0, \forall x \in [\lambda_0, \lambda_{g+k+1})$$

if

$$c \geq 0.$$

Although this is a simple condition, it may introduce a large amount of conservatism in the problem as shown by de Boor and Daniel (1974). To reduce the amount of conservatism a knot insertion method is proposed. By a refinement $\tilde{\lambda}$ of the knot sequence λ, the spline is reformulated as $s(x) = \langle \tilde{c}, b_{k,\tilde{\lambda}} \rangle$ and a relaxation $\tilde{c} \geq 0$ is obtained. The more refined the knot sequence, the closer the control polygon lies to the PP function as is illustrated in Figure 2.5 and hence the smaller the conservatism. Note that this method shows some resemblance to a sampling based approach, where instead of inserting a knot, one directly constrains the function value $s(x_i) \geq 0$ for some $x_i \in [\lambda_0, \lambda_{g+k+1}]$. In this approach the nonnegativity constraint is overly relaxed; there is no guarantee that the spline is nonnegative in between samples. By using knot insertion, nonnegativity is guaranteed albeit with the addition of some conservatism.

Suppose a single knot t is inserted in the knot span $(\lambda_j, \lambda_{j+1})$, then the new spline coefficients can easily be calculated with

$$\tilde{c}_i = (1 - a_i)c_{i-1} + a_i c_i, \text{ for } j - k + 1 \leq i \leq j,$$

where the ratio a_i is determined as

$$a_i = \frac{t - \lambda_i}{\lambda_{i+k} - \lambda_i} \text{ for } j - k + 1 \leq i \leq j.$$

For $i > j$ and $i < j - k + 1$ the coefficients remain unaltered

$$\tilde{c}_i = c_i, \text{ for } i < j - k + 1, \tilde{c}_i = c_{i+1} \text{ for } i > j.$$

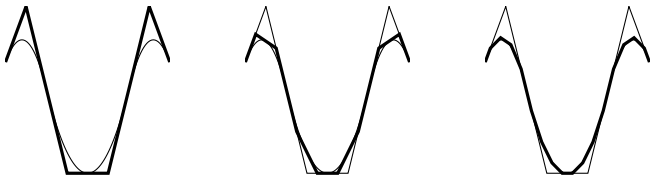

FIGURE 2.5.: The control polygon converges to the spline as the knot spacing decreases. After two midpoint refinements, the control polygon of the spline is already hard to distinguish from the spline

EXAMPLE 2.8 · *Consider a degree two polynomial on* $[0, 1]$*, which can be represented as a degree two spline with knot sequence* $\lambda = (0, 0, 0, 1, 1, 1)$ *and resulting (Bernstein) basis splines*

$$b_{0,2,\lambda} = (1-x)^2, b_{1,2,\lambda} = 2x(1-x), b_{2,2,\lambda} = x^2.$$

Consider the coefficients $c = (\alpha, \beta/2, \alpha)^\intercal$*, such that they match Example 2.5 on Pólya's relaxation.*

Due to the positivity Property 2.1, the obvious relaxation is $\alpha, \beta \geq 0$*, yielding the same relaxation as Pólya's theorem. However, inserting a knot at* $x = 0.5$*, yields the basis functions*

$$\begin{aligned} b_{0,2,\tilde{\lambda}} &= (1-2x)^2 I_{[0,0.5]}, \\ b_{1,2,\tilde{\lambda}} &= 2x(2-3x) I_{[0,0.5]} + 2(1-x)^2 I_{[0.5,1]}, \\ b_{2,2,\tilde{\lambda}} &= 2x^2 I_{[0,0.5]} - (1-2x)(2-3x)2(1-4x+3x^2) I_{[0.5,1]}, \\ b_{3,2,\tilde{\lambda}} &= (1-2x)^2 I_{[0.5,1]} \end{aligned}$$

and corresponding coefficients $\tilde{c} = (\alpha, 0.5(\alpha+\beta/2), 0.5(\alpha+\beta/2), \alpha)^\intercal$*, yielding the relaxation* $\alpha, 2\alpha + \beta \geq 0$*. This relaxation is exact, whereas Pólya's relaxation may require an infinite degree. A sampling based approach at* $x_i = 0, 0.5$ *also yields the correct constraints. However, a knot inserted at another location will not give exact results, e.g.* $\alpha, 2\alpha + 3\beta \geq 0$ *for a knot inserted at* $x = 0.25$*, which is more conservative than the Pólya's relaxation and the sampling based approach for* $x_i = 0, 0.25$*, yields* $\alpha, 10\alpha + 3\beta \geq 0$*, which is overly relaxed.*

2.4. Summary

This chapter sets the necessary basics for the following two chapters. PP functions are defined through B-spline basis functions and the problem of piecewise polynomial nonnegativity is tackled. Based on recent results, necessary and sufficient LMI conditions for nonnegativity are derived. Furthermore, a specific sparsity structure for the matrices is conjectured. Finally, sufficient linear conditions are also derived and by using a knot insertion method a series of relaxations is derived.

What to remember

- Piecewise polynomial functions can be represented as a linear combination of B-spline basis functions. Such a linear combination is called a spline.
- A spline is always contained in the convex hull of its control polygon.
- A univariate polynomial is nonnegative if it can be written as a sum of squares. Determining whether a polynomial can be written as a sum of squares requires solving a semidefinite program.
- A series of relaxations of linear sufficient conditions can also assert nonnegativity of a polynomial.
- Determining whether a piecewise polynomial function is nonnegative requires solving a large, sparse semidefinite program.
- The convex hull property of splines provides a sufficient linear condition for nonnegativity of piecewise polynomial functions. By using knot insertion, these conditions can be relaxed.

3

A convex optimization approach to curve fitting

A first application of spline based optimization in this thesis is function approximation. Approximating measured data by a smooth curve is a frequent problem in motion analysis, computer aided design, image processing and many other fields. When the data exhibit a complicated shape, simple polynomials often fall short. In this case, polynomial splines are often chosen as parametrization for the underlying function (Dierckx 1993, de Boor 2001).

One challenge when optimizing splines is determining the locations of the knots. This requires treating the knots as variables, resulting in a highly nonlinear and nonconvex optimization problem (Dierckx 1993). Consequently, it is difficult to obtain and guarantee global optimality.

In the literature many methods have been proposed to solve this problem. Most methods, however, require a good initial guess of the knot sequence (Jupp 1978), cannot guarantee global optimality (Hayes 1974, Dierckx 1993, Molinari, Durand, and Sabatier 2004) or produce redundant knots (Dierckx 1993). Also, it is often desired to incorporate knowledge of the underlying function in the optimization. Dierckx (1993) proposes algorithms to deal with simple constraints such as end-point derivatives, periodicity and convexity, but more general constraints cannot be incorporated.

Demeulenaere, Pipeleers, et al. (2009) propose a convex framework for optimizing rigid motion trajectories with splines. In their work, the spline knots are optimized indirectly by supplying many candidate knot locations and using a regularization to favor solutions with few active knots, i.e. knots for which the spline exhibits a discontinuity in one of its derivatives. This approach results in a convex optimization problem for which the global optimum is guaranteed to be found efficiently and reliably. In this chapter we apply their ideas to the curve fitting problem with

following essential modifications: (i) instead of integrating a piecewise linear function up to the spline degree, B-splines are used as a basis to evaluate polynomial splines, which enhances numerical stability and provides an elegant way to enforce semi-infinite constraints without gridding (cfr. Section 2.3.2), and (ii) to further increase the sparsity of the solution a reweighed ℓ_1 minimization (Candès, Wakin, and Boyd 2008) is added to the framework. Also, knowledge of the underlying function is easily incorporated by adding (convex) constraints to the optimization problem.

This chapter first introduces the optimization problem and describes the reweighing procedure. Subsequently, three examples of increasing complexity illustrate the power and versatility of the proposed method. The content of this chapter has also been published in Van Loock, Pipeleers, De Schutter, et al. 2011.

3.1. Optimization problem

The basic approximation problem that is considered can be formulated as follows: "given values y_r, $r = 1, \ldots, m$, corresponding to values $x_r \in [x_{\min}, x_{\max}]$, determine a function $y(x) := y(x;\theta)$ of known form but containing a vector θ of n disposable parameters to be determined such that $y(x_r) \approx y_r$" (Hayes 1974). Often, it is also desirable to be able to include knowledge of the underlying function to improve the quality of fit.

The classical way to solve this approximation problem with polynomial splines, is to treat both the spline coefficients c and the spline knots λ as the disposable parameters. However, this approach makes the spline approximation problem nonconvex, because the spline s is a nonconvex function of the knots. To overcome the nonconvexity we adopt an indirect spline knot optimization approach proposed by Demeulenaere, Pipeleers, et al. (2009). Many (typically 500 to 1000) fixed candidate knot locations are provided, leaving only the spline coefficients to be determined and in order to favor solutions with few active knots an ℓ_1 regularization is used. This strategy results in a convex problem of which the global optimum is guaranteed to be found efficiently and reliably using dedicated algorithms.

Given the spline degree k and the knot sequence λ, the spline is completely determined by its spline coefficients c, which are taken as the optimization variables. Given the measurements (x_r, y_r) the basic approximation problem is formulated as

$$\underset{c}{\text{minimize}} \quad \sum_{i=0}^{l-1} w_i \left| c_{i+1}^{(k)} - c_i^{(k)} \right| \tag{3.1a}$$

$$\text{subject to} \quad \sum_{r=1}^{m} \left(v_r (y_r - \langle c, b_{k,\lambda}(x_r) \rangle) \right)^2 \leq S \tag{3.1b}$$

$$g(c) = 0 \tag{3.1c}$$

$$h(c) \leq 0, \tag{3.1d}$$

with w_i and v_r chosen weights, S a fixed parameter which controls the quality of fit and $c^{(k)}$ are the spline coefficients of the k-th derivative of y as defined in (2.4). The objective function (3.1a) is a measure of the nonsmoothness of the fit. Additional constraints to the problem $g(c)$ and $h(c)$ can include prior knowledge of the underlying function. If $g(\cdot)$ is linear and $h(\cdot)$ is convex, the minimization problem is convex and can be solved reliably to global optimality.

Although problem (3.1) is similar to the so-called smoothing criterion for approximation (Dierckx 1975) some important differences should be noted:

- Instead of using the ℓ_1-norm in equation (3.1a), Dierckx (1975) uses the ℓ_2-norm

$$\sum_{i=0}^{l-1} \left(c_{i+1}^{(k)} - c_i^{(k)} \right)^2 . \tag{3.2}$$

  Where the ℓ_2-norm only smooths the fit, the ℓ_1-norm has a twofold effect:

  1. The ℓ_1-norm (3.1a) measures the nonsmoothness of $s(x)$.
  2. It is well-known in the area of function approximation that ℓ_1-norm minimization is likely to yield sparse solutions. Applied to the present problem, sparsity implies few nonzero elements in $\left(c_{i+1}^{(k)} - c_i^{(k)} \right)$, that is, few jumps in $s^{(k)}(x)$ or in

other words few active knots. This means the optimal number of knots and optimal knot locations are found automatically, whereas in other algorithms the number of knots must be chosen beforehand and often a good initial guess of the knot locations is required for the algorithm to converge.

- Since we solve a convex problem, arbitrary convex constraints (3.1c) and (3.1d) can be included without making the problem substantially more difficult. In Dierckx (1993) every other constraint requires a change in algorithm and therefore only algorithms exist for few different types of constraints (e.g. end-point derivative constraints and linear inequality constraints).

If still too many active knots are observed in the solution, a reweighted ℓ_1 minimization is performed to improve sparsity of the solution (Candès, Wakin, and Boyd 2008). Initially $w_i = 1$. In each reweighing iteration, the ℓ_1 optimization problem (3.1) is solved and the weights w_i are updated as:

$$w_i = \frac{1}{\left| c_{i+1}^{(k)} - c_i^{(k)} \right| + \varepsilon}. \tag{3.3}$$

In this way, small jumps in the k-th derivative are penalized more heavily. The value of ε should be chosen slightly smaller than the expected nonzero magnitudes of $c_{i+1}^{(k)} - c_i^{(k)}$.

3.2. Examples

3.2.1. An unconstrained example

As a first example the titanium heat data from de Boor and Rice (1968) are fitted. The data are known to be difficult to fit using traditional techniques due to the sharp peak in the data (Figure 3.1), and have therefore often been used to test spline fitting algorithms (de Boor and Rice 1968, Jupp 1978, Dierckx 1993, F. Yoshimoto, Harada, and Y. Yoshimoto 2003, Molinari, Durand, and Sabatier 2004). The basic optimization problem (3.1) is solved for a cubic spline ($k = 3$) with $S = 0.0073$, $v_r = 1, r = 1, \ldots, m$ and 1000 equidistant candidate knots. The value for S is close to the theoretical

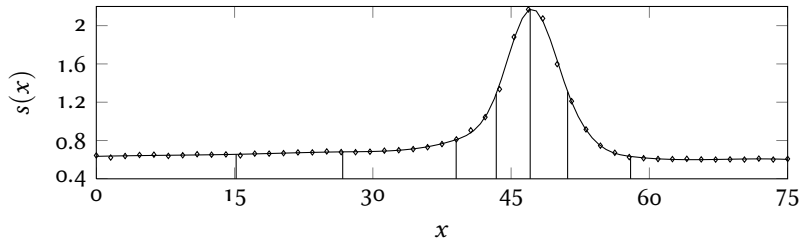

FIGURE 3.1.: A cubic spline ($k = 3$) fitted to the titanium heat data from de Boor and Rice (1968). The residual quadratic error is $S = 0.0073$. The vertical lines represent the locations of the active knots.

method	g	nonsmoothness (3.2)
Dierckx (1975)	8	0.0899
Jupp (1978)	5	0.0742
Convex approach	7	0.0433

TABLE 3.1.: Comparison with respect to the degree of nonsmoothness of different fits to the titanium heat data

minimum quadratic error for 5 knots (Jupp 1978) and corresponds to the value used in Dierckx (1975) so a comparison can be made.

Seven active knots are found after seven reweighing iterations (3.3) with $\varepsilon = 10$. Figure 3.1 shows the fit and the active knot locations found with our method. In Table 3.1, the degree of nonsmoothness (3.2) is shown for our solution and these of Jupp (1978) and Dierckx (1975). Our fit clearly outperforms the results of Jupp (1978) and Dierckx (1975) in terms of smoothness. Moreover, our method neither requires an (accurate) initial guess of the knot sequence, nor is it necessary to choose the number of knots beforehand.

3.2.2. Convexity Constraints

Prior knowledge of the underlying function, such as shape properties, periodicity and known function values, should be taken into account during the optimization to improve the quality of fit. In this example

we consider stress-strain data from Amos and Slater (1969), of which the underlying function is known to be concave.

Shape preserving conditions like positivity, monotonicity and convexity are semi-infinite constraints. In Chapter 2, we saw how we can impose such constraints either exactly with LMI's or conservatively with linear inequalities on the spline coefficients. Note that, due to the large amount of knots, the amount of conservatism will in general be small.

In the following example we consider a cubic spline and add the concavity constraint $\partial_x^2 s(x) \leq 0, \forall x \in [x_1, x_m]$. This is accomplished by the constraint $c^{(2)} \leq 0$ (cfr. equation (2.4)), which (for the case of a cubic spline) is exact and hence does not introduce conservatism. The smoothing parameter S is chosen $S = 0.0044$, as in Dierckx (1980) and the problem is solved for a 1000 equidistant candidate knots. Five knots are found after five reweighting iterations (3.3) with $\varepsilon = 10$. Figure 3.2 shows the fit (top) and its second derivative (bottom). The concavity constraint is active on the first and last polynomial segment, resulting in linear segments in $s(x)$.

Both the convex approach and Dierckx (1980) find five active knots, although the latter only allows knots at the measurement points. The convex framework yields a smoothness value of 500 whereas Dierckx (1980) finds 828.

3.2.3. Distance constrained fitting

In this section we discuss a more involved example including multiple functions with mutual constraints. Consider measurements of n marker positions P_i, $i = 1, \ldots, n$, at time samples t_k, $k = 1, \ldots, m$, on a rigid body moving in three dimensional space as illustrated in Figure 3.3 for $n = 3$. Through the measured coordinates, x_i, y_i and z_i of each marker, we wish to fit a spline, $s_{x_i}(t)$, $s_{y_i}(t)$ and $s_{z_i}(t)$, taking into account that the distance between two markers remains constant. In the presence of measurement errors, a constant distance might overly constrain the solution. Therefore we allow the distance to vary $\pm\delta$ around the distance μ_{ij} between the markers P_i and P_j ($i < j$). The distance μ_{ij} is approximated by averaging out the distance between the measurements of the marker positions over

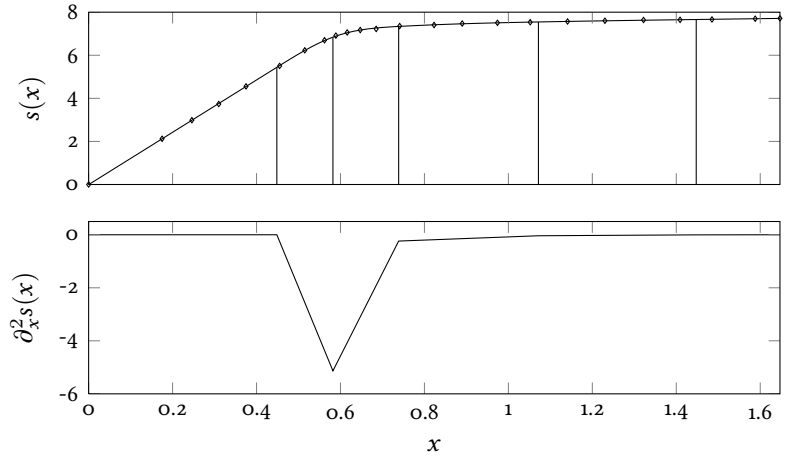

FIGURE 3.2.: A cubic spline (top) and its second derivative (bottom) fitted to the concave stress-strain data from Amos and Slater (1969), with a residual quadratic error $S = 0.0044$. The concavity constraint is active at the first and final segment. The vertical lines represent the locations of the active knots.

time. Now we wish to impose

$$(\mu_{ij}-\delta)^2 \leq \sum_{l=x,y,z} (s_{l_i}(t_k)-s_{l_j}(t_k))^2 \leq (\mu_{ij}+\delta)^2, \text{ for } k = 1,\ldots,m \quad (3.4)$$

with $i < j$.

However, this constraint is nonconvex due to the concavity of the lower bound. To overcome this problem, the lower bound in (3.4) is linearized. Figure 3.4 illustrates the constraints and linearization in two dimensions (constant z-coordinate). This allows for a simpler visualization. In the $(\Delta x, \Delta y) = (x_i - x_j, y_i - y_j)$-plane, constraint (3.4) is visualized by two concentric circles with radii $(\mu_{ij} \pm \delta)$. Now assume we have from the measurements a point $(dx, dy) = (x_i(t_k)-x_j(t_k), y_i(t_k)-y_j(t_k))$. Because of (3.1b), $(s_{x_i}(t_k) - s_{x_j}(t_k), s_{y_i}(t_k) - s_{y_j}(t_k))$ will lie close to this point. Therefore the concave lower bound is linearized at a point (n_x, n_y) on

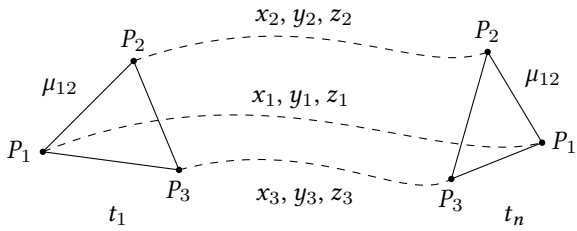

FIGURE 3.3.: Three markers, P_1, P_2 and P_3 measure the movement of a rigid body. At all time instances the distance between the markers is constant.

the lower bound that is closest to (dx, dy),

$$\begin{aligned} n_x &= \operatorname{sgn}(dx)(\mu_{ij} - \delta)\sqrt{\frac{dx^2}{dx^2 + dy^2}}\,, \\ n_y &= \operatorname{sgn}(dy)(\mu_{ij} - \delta)\sqrt{\frac{dy^2}{dx^2 + dy^2}}\,, \end{aligned} \tag{3.5}$$

resulting in the convex set C, indicated by the gray region in Figure 3.4. The region is defined by the constraints

$$(s_{x_i} - s_{x_j})^2 + (s_{y_i} - s_{y_j})^2 \leq (\mu_{ij} + \delta)^2\,, \tag{3.6}$$

$$n_x(s_{x_i} - s_{x_j}) + n_y(s_{y_i} - s_{y_j}) \geq (\mu_{ij} - \delta)^2\,. \tag{3.7}$$

Note the sign-function in (3.5) to ensure that the linearization is carried out in the correct quadrant.

To verify the above approach, fictitious measurements are generated for two markers on a rigid body, moving in a plane. The markers are initially located at $(1\,\mathrm{m}, 0\,\mathrm{m})$ and $(-1\,\mathrm{m}, 0\,\mathrm{m})$ and rotate 180° around the origin in $N = 100$ samples with $t_s = 0.01\,\mathrm{s}$. Gaussian noise with a standard deviation $\sigma = 0.02\,\mathrm{m}$ is added to each coordinate. Figure 3.5 illustrates the simulated marker measurements and the true trajectories for both markers.

Four splines are then fitted simultaneously, one for each coordinate of each marker. Problem (3.1) is therefore extended. First, for each coordinate a one-norm regularization is added to the goal function and

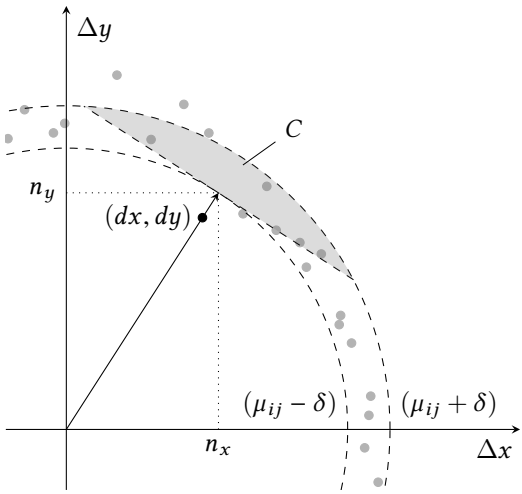

FIGURE 3.4.: Linearization of constraint (3.4) in two dimensions. $(s_{x_i} - s_{x_j}, s_{y_i} - s_{y_j})$ must lie in the ring delimited by the circles with radii $(\mu_{ij} \pm \delta)$. The gray dots indicate measured values.

constraints (3.1b) are repeated. Furthermore, constraints (3.6, 3.7) are added to constrain the distance between the markers. For the optimization S is chosen equal to $N\sigma^2$ since this is the expected value of the residual quadratic error, $\delta = 0.001\,\text{m}$, $k = 4$, $g = 100$, and no reweighting iterations are performed. The resulting splines all exhibit one or none active knots.

Figure 3.6 shows the resulting distance in function of time for fitted splines with (solid) and without (dashed) the distance constraints. The dotted line indicates the evolution of the measured distance. It is clear from the zoomed (bottom) plot that the constrained splines stay within their bounds indicated by the gray region. Observe that the true lower bound is never active due to the conservatism introduced by the linearization. Consequently δ should not be chosen too small. The unconstrained splines clearly exhibit more variation in the distance and seem to follow a slower variation of the measured distance. Figure 3.7 shows the radial difference Δ of the fitted spline with the true trajectories for both markers. Except for the beginning, the constrained spline (solid line) fits the trajectories more accurately.

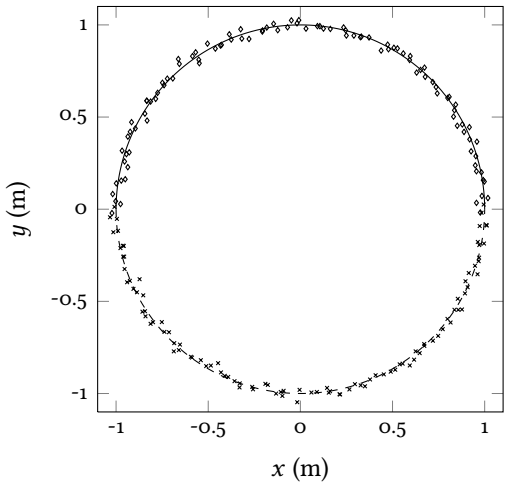

FIGURE 3.5.: The simulated marker measurements and true marker trajectories for both markers

3.3. Summary

Instead of taking the knots as variables, this chapter takes an indirect approach to the spline approximation problem by supplying many candidate knots and using an ℓ_1 smoothing criterion to favor solutions with few active knots. To further increase the sparsity, the ℓ_1 norm is iteratively reweighed. This way, the optimization problem is cast as a *convex* optimization problem in which prior knowledge can easily be included.

What to remember

- Treating the spline's knot locations as variables results in a highly nonlinear and nonconvex optimization problem.
- An indirect approach supplies many candidate knots and uses an ℓ_1 smoothing criterion to favor solutions with few active knots, resulting in a *convex* optimization problem.
- Iteratively reweighing the ℓ_1 norm can increase the sparsity even further.

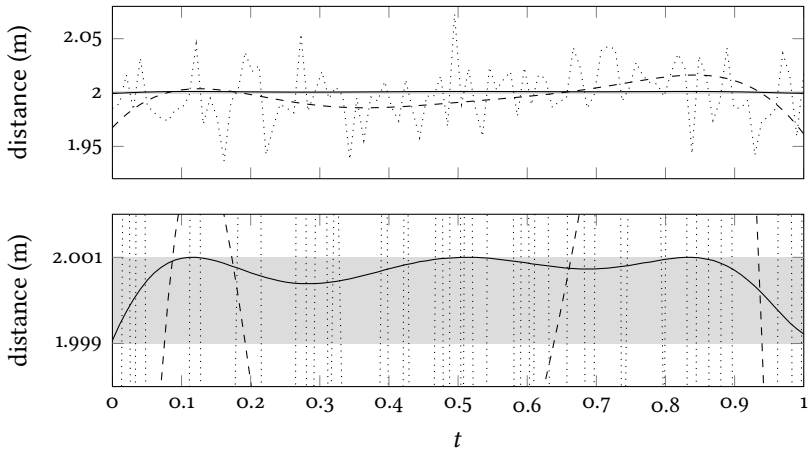

FIGURE 3.6.: Resulting distances in function of time for fitted splines with (solid line) and without (dashed line) distance constraints. The dotted line represents the variation of the measured distance. The bottom plot shows the constrained distance in more detail. The distance must lie within the gray region.

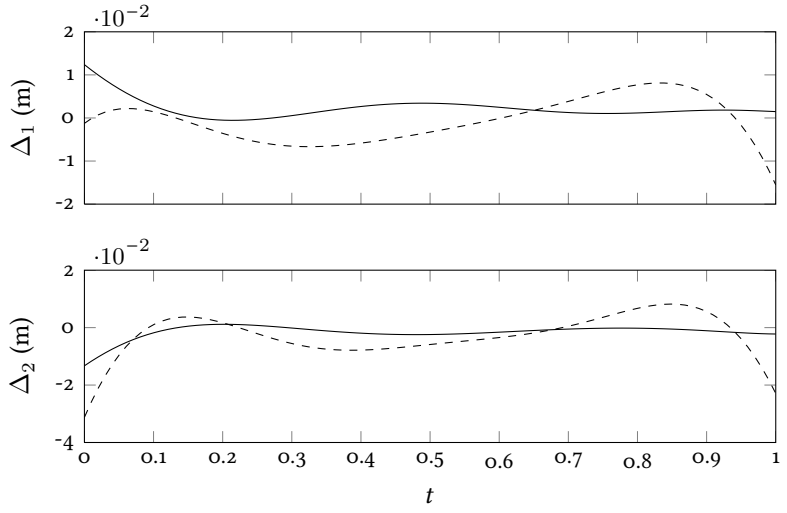

FIGURE 3.7.: Radial difference Δ of the fitted splines with the true trajectories for both markers, with (solid line) and without (dashed line) distance constraints.

4

Optimal control of linear systems through differential flatness

Determining an open-loop control law that optimally drives a system from an initial state to a terminal state without violating input and state constraints is a classical problem in control. In this motion planning problem, flatness is a popular concept as it avoids integration of the differential equations. Instead of imposing the state and input constraints on a finite number of samples, by parameterizing the flat output of a linear system as a piecewise polynomial, the optimization problem can be viewed as a piecewise polynomial nonnegativity problem as shown in Louembet, Cazaurang, and Zolghadri (2010), Suryawan, De Doná, and Seron (2011), Suryawan, De Doná, and Seron (2012). In Louembet, Cazaurang, and Zolghadri (2010) the nonnegativity constraints are imposed through semidefinite programming. The finite support of the polynomial pieces, however, is not taken into account, which introduces conservatism. Suryawan, De Doná, and Seron (2011), Suryawan, De Doná, and Seron (2012) use the convex hull Property 2.3 to develop a linear programming approach. Although, it is computationally cheaper compared to semidefinite programming, a large amount of conservatism is unwillingly introduced.

This chapter bridges both approaches by using either exact semidefinite conditions or linear relaxations for which the conservatism can be strongly reduced, as developed in Chapter 2. The first section develops the general problem formulation and benchmarks the proposed problem formulation to other recent results. Subsequently, in contrast to traditional binary search approaches (Demeulenaere, De Caigny, et al. 2009, Consolini and Piazzi 2009, Van den Broeck, Diehl, and Swevers 2011a, Van den Broeck, Diehl, and Swevers 2011b, Suryawan, De Doná, and Seron 2011), a novel iterative procedure based on Newton-Raphson's root finding algorithm

for the computation of time-optimal point-to-point motions is presented. Additionally, the parametric solution of such problems is treated. Flatness also proves useful for the computation of robust inputs for uncertain systems, as described in the final section. By eliminating the equality constraints imposing that all outputs originate from the same input, the robustness constraints amount to semi-infinite constraints for which a tractable robust counterpart can be derived. This way, gridding of the uncertainty region as in De Caigny et al. (2008), De Caigny (2009) and a reachability analysis is avoided.

4.1. General problem formulation

Consider a controllable linear time-invariant system

$$\dot{x}(t) = Ax(t) + Bu(t), z(t) = Cx(t)$$

with states $x \in \mathbb{R}^n$ and inputs $u \in \mathbb{R}^m$. We are interested in finding the control law $u(t), t \in [0, T]$ that steers the system from an initial state x_0 to a terminal state x_T and that minimizes a performance criterion

$$J(x, u, T) = G(x(T), u(T), T) + \int_0^T F(x(t), u(t), t)\, dt,$$

while at the same time obeying constraints on states and inputs

$$H(x(t), u(t)) \geq 0, \forall t \in [0, T].$$

We assume the functions $H(\cdot, \cdot)$ to be linear and $J(\cdot)$ convex. Two common performance criteria are time where $J = T$, such that a time-optimal solution is determined, and tracking where a given performance output $z^{\text{ref}}(t)$ is tracked as accurately as possible $J = \int_0^T \|z(t) - z^{\text{ref}}(t)\|^2\, dt$.

4.1.1. Spline parameterization of the flat output

First, this section formulates the problem in terms of a time-scaled flat output of the system. In Chapter 1, it is stated that a linear system is differentially flat if and only if it is controllable. For these systems the

states and inputs of the system are a linear combination of the flat output $y \in \mathbb{R}^m$ and its derivatives. Therefore, states and inputs are substituted by the flat output of the system. The initial and terminal states are translated to conditions on the flat output as in Section 1.3.

Furthermore, to cope with free end-time problems, a time scaling is adopted by substituting the time parameter $t \in [0, T]$ by $\tau = t/T \in [0, 1]$. Hence, the time derivatives of the flat output must be scaled with the end-time:

$$\partial_t y = \partial_\tau y \partial_t \tau = \partial_\tau y T^{-1}$$

Then the optimization problem can be written in terms of the time-scaled flat output of the system

$$\begin{aligned} \underset{y(\cdot),T}{\text{minimize}} \quad & J_y(y(\tau), \dots, \partial_\tau^r y(\tau), T) \\ \text{subject to} \quad & \partial_\tau^i y(0) = T^i y_0^{(i)}, \text{ for } i = 0, \dots, r \\ & \partial_\tau^i y(1) = T^i y_T^{(i)}, \text{ for } i = 0, \dots, r \\ & H_y(y(\tau), \dots, \partial_\tau^r y(\tau), T) \geq 0, \forall \tau \in [0, 1]. \end{aligned} \tag{4.1}$$

Let us now describe each of the components of y by a polynomial spline (cfr. Section 2.1), such that the problem reduces to determining the spline coefficients of the flat outputs:

$$y_i = \langle c_i, b_{k,\lambda} \rangle,$$

where the degree $k \geq r$ and λ is a chosen knot sequence.

As H_y is a linear combination of y and its derivatives, each element of H_y can be represented by a PP function (cfr. Property 2.5). The spline coefficients h_j of each element in H_y depend linearly on the coefficients of the flat outputs c:

$$h_j = \sum_i \mathcal{T}_i c_i,$$

with $\mathcal{T}_i$ suitable transformation matrices.

Now, by imposing either semidefinite constraints or linear constraints on h_j (cfr. Chapter 2), the semi-infinite nonnegativity requirement

$$H_y(y(\tau), \dots, \partial_\tau^r y(\tau), T) \geq 0, \forall \tau \in [0, 1]$$

can be fulfilled.

4.1.2. Benchmark example

This section considers the motorized base-stage high-precision positioning system, described in Lévine and Nguyen (2003) and subsequently treated in Henrion and Lasserre (2006) and Suryawan, De Doná, and Seron (2012). With the system as depicted in Figure 4.1, he transfer function matrices are given by

$$\begin{pmatrix} 25s^2 & 0 \\ 0 & 450s^2 + 1.1875 \times 10^4 s + 6.3955 \times 10^5 \end{pmatrix} \begin{pmatrix} w_\mathrm{s} \\ w_\mathrm{b} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} u,$$

where the input u is the force applied to the stage, w_s is the relative position of the center of mass of the stage with respect to a coordinate frame attached to the base with origin w_b, the position of the center of mass of the base in a fixed coordinate frame related to the ground (Lévine and Nguyen 2003). A flat output y of the system is given by $y = w_\mathrm{s} - 0.0186\dot{w}_\mathrm{s} + 9.18w_\mathrm{b} - 0.3342\dot{w}_\mathrm{b}$, such that

$$\begin{aligned} w_\mathrm{s} &= y + 0.0186\dot{y} + 7.0362 \times 10^{-4}\ddot{y} \\ w_\mathrm{b} &= -3.909 \times 10^{-5}\ddot{y} \\ u &= 25\ddot{y} + 0.4642\dddot{y} + 0.0176\ddddot{y}. \end{aligned} \tag{4.2}$$

The goal is to track a given reference as accurately as possible, while restricting the movement of the base between $-0.17\,\mathrm{mm}$ and $0.12\,\mathrm{mm}$. The system's initial state is given by $w_\mathrm{s}(0) = 0\,\mathrm{m}$, $\dot{w}_\mathrm{s}(0) = 0\,\mathrm{m\,s^{-1}}$,

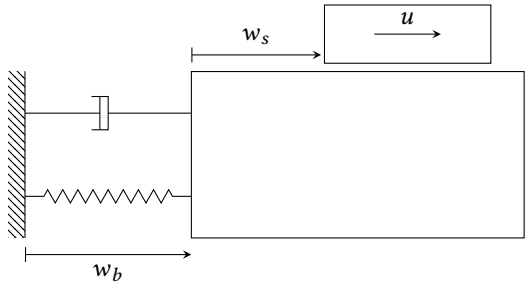

FIGURE 4.1.: Base stage high-precision positioning system

$w_b(0) = 0\,\mathrm{m}$, $\dot{w}_b(0) = 0\,\mathrm{m\,s^{-1}}$, $u(0) = 0\,\mathrm{N}$ and its terminal state by $w_s(0.2) = 0.02\,\mathrm{m}$, $\dot{w}_s(0.2) = 0\,\mathrm{m\,s^{-1}}$, $w_b(0.2) = 0\,\mathrm{m}$, $\dot{w}_b(0.2) = 0\,\mathrm{m\,s^{-1}}$, $u(0.2) = 0\,\mathrm{N}$. For the flat output this yields $y(0) = 0$, $y(0.2) = 0.02$ and zero for the derivatives up to order four at the boundaries. Hence, the minimal degree of an interpolating polynomial is 9. By using knot insertion (cfr. Section 2.3.2) the interpolating polynomial is expressed in the same basis as y. Its spline coefficients are denoted by c_d. Then, from (4.2) a tracking reference $w_{s,d}$ for the stage displacement is calculated.

The spline coefficients of the stage are related to c, the coefficients of the flat output, through the transformation matrix $\mathcal{T}_s$. The tracking performance criterion $\int_0^T \|w_s(t) - w_{s,d}(t)\|^2\,dt$ is reformulated in terms of the spline coefficients of the stage

$$J = (c - c_d)^\intercal \mathcal{T}_s{}^\intercal \mathcal{T}_s (c - c_d).$$

First, the semi-infinite constraints on the movement of the base are imposed by using the linear relaxations from Section 2.3.2. Without any knot refinement, it has been numerically established by gradually increasing the number of knots that minimally six equidistant internal breaks for a spline of degree nine are required for the optimization problem to be feasible. Optimizing over six uniformly distributed knots yields the optimal value is $J^* = 1.6083 \times 10^{-5}\,\mathrm{m^2}$. The resulting movements of the stage and of the base are shown in Figure 4.2 as the black line. Clearly, as the true bounds are not even hit, a large amount of conservatism is introduced by using these linear constraints. The dashed line shows the reference trajectory for the stage. By using one or two midpoint refinements of the knot sequence, the amount of conservatism is significantly reduced, as illustrated by the dark and light gray lines in Figure 4.2 with respectively one and two midpoint refinements and optimal values $J^* = 8.4015 \times 10^{-6}\,\mathrm{m^2}$ and $J^* = 6.3977 \times 10^{-6}\,\mathrm{m^2}$. Clearly, the less conservative constraints contribute to the improvement of the optimal value. For comparison a similar framework from Suryawan, De Doná, and Seron (2012) is also implemented. For this method minimally sixteen equidistant internal knots are required for the problem to be feasible. The optimal value is $J^* = 3.1933 \times 10^{-5}\,\mathrm{m^2}$, which is larger than our method using only six internal knots. Moreover the computational time is larger, due to

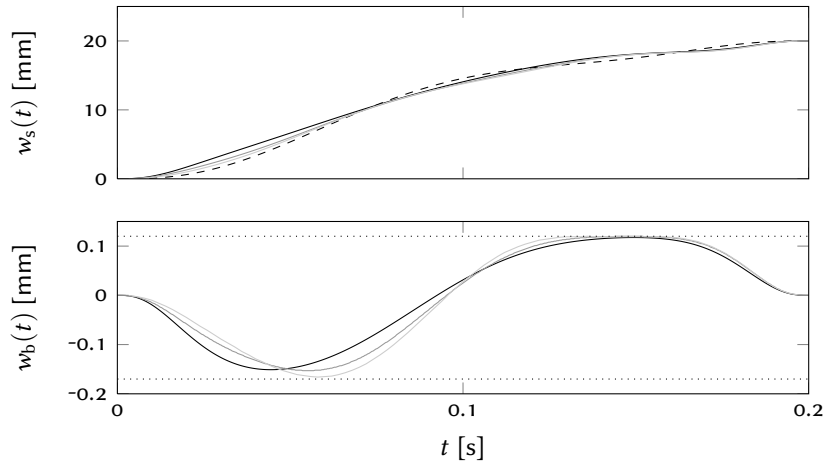

FIGURE 4.2.: Optimal tracking of the dashed reference for a base-stage high-precision positioning system, while limiting the displacement of the base. The black, gray and light gray lines show the solution for a spline with six internal knots using linear relaxations with respectively no, one and two midpoint refinements.

a larger number of constraints in the problem formulation. For the problem with sixteen internal knots, 3.37 s are required to solve the problem, whereas our approach only requires 1.62 s with sixteen internal knots and two refinements and finds a more accurate solution than Suryawan, De Doná, and Seron (2012). Both problem instances are solved using CVXOPT (Andersen, Dahl, and Vandenberghe 2013).

Figure 4.3 shows the stage and base displacements for the exact semidefinite approach for a degree nine spline with six internal knots. The optimal value is $J^* = 6.0405 \times 10^{-6}\,\mathrm{m}^2$, which outperforms the linear relaxations in terms of optimality. However, it should be noted that the linear approach with double midpoint refinement (gray line) lies close to the exact solution. The main drawback of the semidefinite approach is a larger computational effort required to solve the problem compared to the linear approach. For six internal knots, the semidefinite approach requires 8.44 s, whereas the linear approach with two knot refinements only requires 0.45 s.

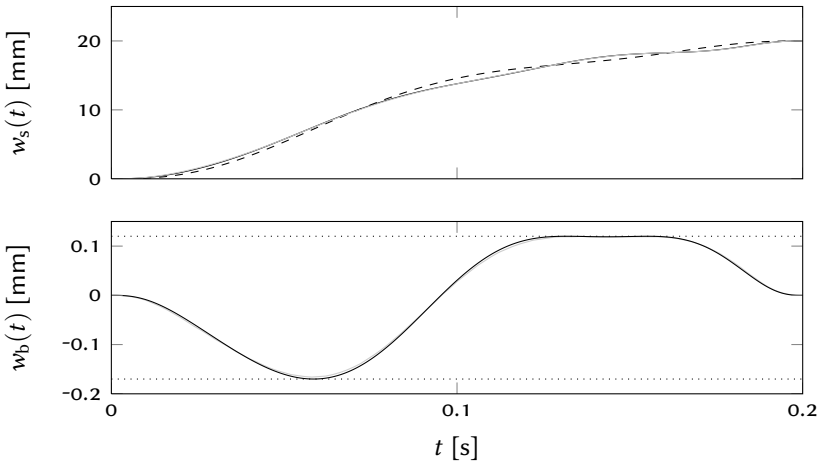

FIGURE 4.3.: Optimal stage and base displacements for a base-stage high-precision positioning system, using an exact semidefinite programming approach. For comparison the solution of the linear relaxation with 2 midpoint refinements is shown in gray. The difference is marginal.

In this section, the nonnegativity theory developed in Chapter 2 is applied to a typical optimal control problem. The resulting optimization problems only contain few variables and constraints, while the benchmark results indicate a significant improvement over other recently published results. Up to now, the end-time T was fixed. Many applications, however, expect a time-optimal trajectory. This requires T to be an optimization variable, which would destroy the convexity. As an alternative, the following section details how to compute such trajectories by solving a series of optimization problems.

4.2. Efficient computation of time-optimal point-to-point motion trajectories

The following sections consider time-optimal, i.e. $J = T$, rest-to-rest motion trajectories for linear time-invariant systems with a scalar output. Instead of treating T as a variable, a parametric optimization problem in

T is constructed. By repeatedly solving the problem for fixed T, the first section develops an iterative method based on Newton-Raphson's root finding algorithm to compute the optimal time efficiently. Subsequently, the algorithm's efficiency is illustrated by two benchmark examples. The final section computes a parametric solution to the optimization problem and avoids the iterative procedure altogether.

4.2.1. Safeguarded Newton-Raphson root-finding

Many of the ideas in this section have been published in Janssens et al. (2013a) and Janssens et al. (2013b). These publications focus on discrete time systems whereas this work considers continuous time systems and relies strongly on the concept of differential flatness.

Due to the quasi-convex nature of the problem, time-optimal point-to-point motion trajectories are typically solved using a binary feasibility search algorithm also known as bisection (see e.g. Demeulenaere, De Caigny, et al. (2009), Consolini and Piazzi (2009), Van den Broeck, Diehl, and Swevers (2011a), Van den Broeck, Diehl, and Swevers (2011b), Suryawan, De Doná, and Seron (2011)). Starting from an interval $[T_-, T_+]$, that is known to contain the optimal time T^*, binary search determines whether T^* is in the lower or upper half of the interval by solving a feasibility problem and updates the search interval accordingly (Boyd and Vandenberghe 2004).

Equivalently, the problem can be cast as a root finding problem of the function $d(T) = d^*(T) - d_\mathrm{d}$, where $d^*(T)$ denotes the maximum travel range of the system for a given time T and d_d is the desired travel distance. As the function $d(T)$ is monotonically nondecreasing, it has only a single root. Where the bisection algorithm only uses function values of $d(T)$ for a given trial time, our approach aims at also using the value of the derivative, as in Newton-Raphson's method, to increase the rate of convergence from linear to quadratic convergence near the solution (Burden and Faires 2010).

To be able to apply the above ideas, the rest-to-rest maximum reach

problem is defined

$$\begin{aligned} d^*(T) = \underset{y(\cdot),d}{\text{maximize}} \quad & d \\ \text{subject to} \quad & y(0) = 0, y(1) = d \\ & \partial_\tau^i y(0) = \partial_\tau^i y(1) = 0, \text{ for } i = 1, \dots, r \\ & H_y(y(\tau), \dots, \partial_\tau^r y(\tau), T) \geq 0, \forall \tau \in [0, 1], \end{aligned} \tag{4.3}$$

in which the reach of the system given the constraints H_y is maximized. The following discussion considers a single-input single-output system ($m = 1$) and assumes that the output of the system scales with the flat output. For a multiple input multiple output system the reader is referred to the examples in Janssens et al. (2013a), Janssens et al. (2013b).

By using the proposed spline parameterization for y and transforming the semi-infinite constraints H_y to linear or semidefinite constraints, problem (4.3) corresponds to a parametric optimization problem in T, which, in its most general form, can be cast as

$$\begin{aligned} d^*(T) = \underset{x}{\text{maximize}} \quad & c^\intercal x \\ \text{subject to} \quad & A_{\text{eq}}(T)x = b_{\text{eq}} \\ & A_{\text{in}}(T)x \leq b_{\text{in}} \\ & A_{\text{sdp}}(x, T) \succeq 0, \end{aligned} \tag{4.4}$$

where x represents the optimization variables, c, b_{eq} and b_{in} are vectors of coefficients, A_{eq} and A_{in} are matrices of coefficients (depending on T) and $A_{\text{sdp}}(x, T)$ is a matrix depending affinely on x.

Let x^* denote an optimal point of the problem for $T = T_i$ and $\lambda^*, \mu^*, \gamma^*$ the corresponding optimal Lagrange multipliers of the equality, linear inequality and linear matrix inequality constraints. The derivative of d^* with respect to T evaluated at T_i is given by (Freund 1985, Shapiro 1997):

$$\begin{aligned} \partial_T d^*(T_i) = & -(\lambda^*)^\intercal \partial_T A_{\text{eq}}(T_i)x^* - \\ & (\mu^*)^\intercal \partial_T A_{\text{in}}(T_i)x^* - \\ & \operatorname{tr}((\gamma^*)^\intercal \partial_T A_{\text{sdp}}(x^*, T_i)). \end{aligned}$$

Now, instead of using binary search, a derivative based root finding algorithm, such as Newton-Raphson's method, can be applied to the quasi-convex time-optimal point-to-point motion planning problem. Moreover, the evaluation of the derivatives is computationally cheap compared to solving an optimization or feasibility problem. Therefore, due to the quadratic convergence of Newton-Raphson's method, large performance improvements over binary search are expected.

In Newton-Raphson's method convergence is not guaranteed for each initial guess (e.g. when the derivative evaluates to zero), which is an important drawback. For this reason a safeguarded Newton-Raphson scheme is adopted (Janssens et al. 2013a, Janssens et al. 2013b, Hedge and Kacera 2006), that combines Newton-Raphson's method with binary search yielding a guaranteed convergence provided that the initial search interval $[T_-, T_+]$ contains the minimal motion time T^*. When the initial guess is sufficiently close to T^* the safeguarded method only uses Newton-Raphson iterations.

Each iteration computes the solution to the maximum range problem (4.3) for the current estimate T_i of the minimal motion time. Depending on the sign of $d(T_i)$ the upper or lower bound of the search interval $[T_-, T_+]$ is substituted by the current estimate T_i. Then, the next iteration's estimate T_{i+1} is computed according to Newton-Raphson's method:

$$T_{i+1} = T_i - \left(\partial_T d^*(T_i)\right)^{-1} d(T_i).$$

If $T_{i+1} \notin [T_-, T_+]$, a bisection step is performed instead:

$$T_{i+1} = \frac{T_- + T_+}{2}.$$

The algorithm terminates when $|d(T_i)| \leq \text{TOL}$, with TOL a predefined tolerance.

4.2.2. Benchmark examples

We consider two examples from Consolini and Piazzi (2009) and subsequently treated in Suryawan, De Doná, and Seron (2011). All presented results use an *equidistant* knot spacing for the spline parameterization.

The first example consists of a second order system

$$H(s) = \frac{10(s+2)}{(s+1)^2 + 9}, \tag{4.5}$$

with input constraints $u(t) \in [-1.8, 1.8]$ and overshoot constraints $z(t) \in [-0.1, 3.1]$ on the output, for which we want to compute a minimum time rest-to-rest transition from $z(0) = 0$ to $z(T) = 3$. Input and output are related to the flat output via

$$\begin{aligned} u(t) &= 0.5y(t) + 0.1\dot{y}(t) + 0.05\ddot{y}(t) \\ z(t) &= y(t) + 0.5\dot{y}(t). \end{aligned}$$

Therefore, the rest-to-rest conditions require $y(0) = 0, y(T) = 3$ and $\dot{y}(0) = \dot{y}(T) = 0$. Furthermore, $u(0) = 0$ and $u(T)H(0) = z(T)$ is required.

We first compare the newly developed Newton-Raphson iteration scheme with the traditional binary search method. In order to make a fair comparison in iteration count between both methods, the binary search method is also applied to the maximum reach problem instead of the feasibility problem as used in (Demeulenaere, De Caigny, et al. 2009, Consolini and Piazzi 2009, Van den Broeck, Diehl, and Swevers 2011a, Van den Broeck, Diehl, and Swevers 2011b, Suryawan, De Doná, and Seron 2011), such that the same stopping criterion can be used. Although the computational cost of solving an optimization problem is somewhat higher than a feasibility problem, it is assumed the difference is marginal compared to the total computation time of the binary search algorithm.

The flat output is parameterized by a spline of degree 4 with 10 internal knots. The semi-infinite constraints are imposed linearly, as described in Section 2.3.2 with two midpoint refinements. Figure 4.4 shows the maximum travel range as a function of the motion time for the chosen parameterization. The optimal time $T^* = 1.983$ s for a travel range $y(T) = 3$.

The time-optimal point-to-point problem is solved for a range of interval widths w from 0.001 s to 9 s.[1] The minimal motion time is located at

[1] In practice the interval is chosen as the smallest interval known to contain the optimal time.

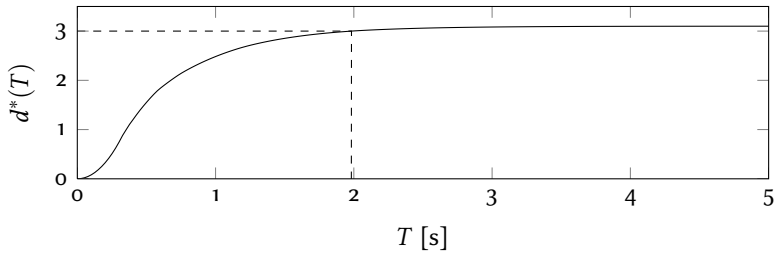

FIGURE 4.4.: Maximum travel range $d^*(T)$ as a function of the available time of the system (4.5) for a degree 4 spline with 10 internal knots

$\frac{1}{5}$ of the interval and for both methods the initial guess is located in the middle:

$$T_- = T^* - 0.2w, T_+ = T^* + 0.8w, T_1 = 0.5(T_- + T_+). \tag{4.6}$$

Figure 4.5 shows the number of iterations for both the binary search and Newton-Raphson's method for a tolerance TOL = 10^{-6}. The dashed line illustrates the number of safeguarding steps in Newton-Raphson's method. For intervals widths $w \geq 1.8$ s, one safeguarding step is required as the gradient to $d(T_1)$ tends to zero for increasing width (see Figure 4.4). As expected, the proposed algorithm clearly outperforms binary search. The average computation time per iteration amounts to 0.0040 s for a bisection step and 0.0046 s for a Newton step on a 2.6 GHz laptop PC using Python and GLPK (*Gnu linear programming kit*) as a solver.

Due to the limitations of the chosen parameterization, the trajectory does not exhibit generalized bang-bang behavior as in Consolini and Piazzi (2009) and therefore it is not truly time-optimal. On the other hand, the smoothness of the control signals, which depends on the chosen spline parameterization, puts lower strain on the actuators and avoid excitation of higher system dynamics. In this example the true minimal motion time is $T_{\text{opt}} = 1.812$ s. By increasing the number of knots, the spline-based solution approaches the true solution. Naturally, this comes at a higher computational cost, as the optimization problem grows in size. Figure 4.6 shows the evolution of the optimal time for an increasing number of (equidistant) knots from 3 to 50. Initially, adding knots leads to a major

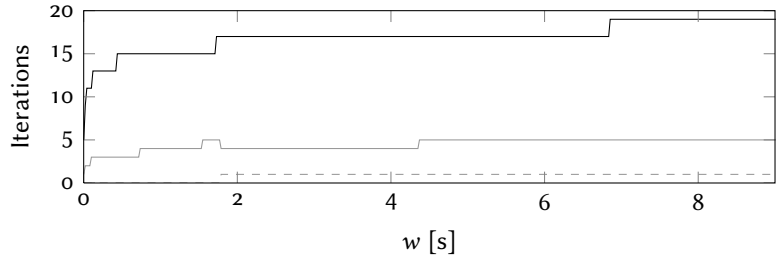

FIGURE 4.5.: Number of iterations for the binary search algorithm (black) and the proposed safeguarded Newton-Raphson's method (gray) as a function of the interval width w. The dashed line shows the number of safeguarding steps.

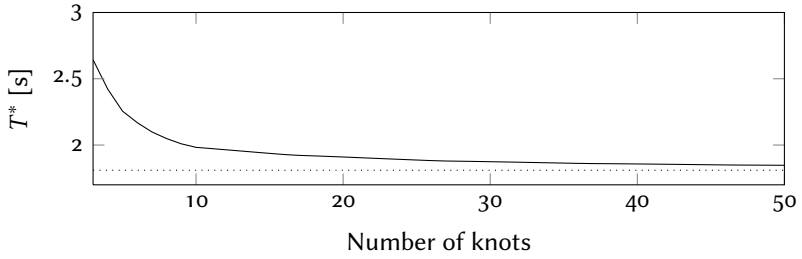

FIGURE 4.6.: The optimal time as a function of the number of knots. For an increasing number of knots the solution approaches the theoretical minimum (dotted line)

performance increase, which subsides when the number of knots grows larger. Figure 4.7 show the computed inputs and outputs for splines with 10, 20, 30, 40 and 50 internal knots. By increasing number of knots, the solution tends more to the generalized bang-bang solution. In Suryawan, De Doná, and Seron (2011) a degree 4 spline with 52 knots yields an optimal time of 1.885 s, while in our approach, 26 knots are sufficient to find $T^* = 1.882$ s. With 52 knots the optimal time $T^* = 1.846$ s.

The semidefinite programming approach (cfr. Chapter 2) yields similar convergence results as the linear approach, which was used in the discussion above. However, for the same number of knots, the semidefinite approach finds slightly more optimal results. Figure 4.8 shows the relative

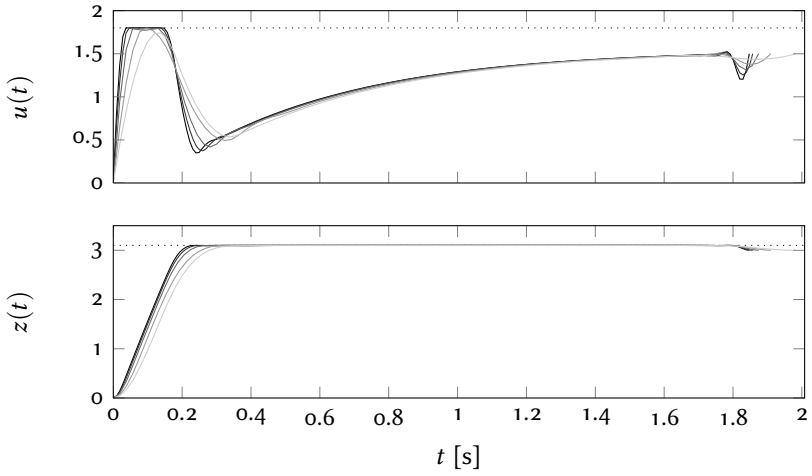

FIGURE 4.7.: The optimal system inputs and outputs for splines of degree 4 with 10, 20, 30, 40, and 50 internal knots (gray to black). For an increasing number of knots the solution tends to a generalized bang-bang solution

difference between the optimal time found with linear programming, T^*_{lin}, (using 2 midpoint refinements) and semidefinite programming, T^*_{sdp}, for an increasing number of knots. For three knots, the difference amounts to 1 %, but as the number of knots increases, the difference between both approaches becomes negligible. Moreover, the relative small improvement comes at a significantly higher computational cost as the LMI's quickly grow in size. For 10 knots the average computation time per Newton iteration amounts to 5.0300 s using Python and CVXOPT as a solver, whereas the linear approach only takes 0.0046 s.

The second example considers a fourth order system with transfer function

$$H(s) = \frac{10(3.5 - s)(s^2 + 25)}{(s + 2)(s + 3)(s + 4)(s + 5)}. \tag{4.7}$$

The desired transition is from $z(0) = 0$ to $z(T) = 3$, while constraining the input $u(t) \in [-2, 2]$ and the output $z(t) \in [-0.1, 3.1]$. Input and output

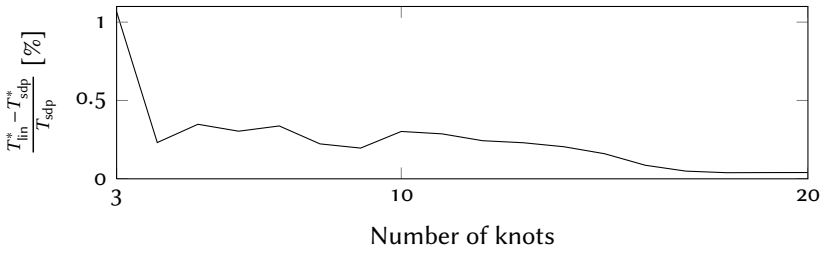

FIGURE 4.8.: The relative difference between the optimal time found with linear programming using two midpoint refinements and semidefinite programming for an increasing number of knots. As the number of knots increases, the difference between both approaches becomes negligible

are related to a flat output through

$$u(t) = \frac{1}{875}\left(y^{(4)}(t) + 14y^{(3)}(t) + 71\ddot{y}(t) + 154\dot{y}(t) + 120y(t)\right)$$
$$z(t) = \frac{1}{875}\left(-10y^{(3)}(t) + 35\ddot{y}(t) - 250\dot{y}(t) + 875y(t)\right).$$

As in the previous example, the flat output is parameterized by a spline of degree 4 with 10 internal knots. The semi-infinite constraints are imposed linearly with two midpoint refinements. Figure 4.9 shows the maximum travel range as a function of the motion time for the chosen parameterization. Note, in contrast to previous example, the sudden change in slope once the maximum travel range hits the bound on the output $z(t) \leq 3.1$. For a travel range $y(T) = 3$, the optimal time $T^* = 1.459$ s.

The time-optimal point-to-point problem is solved for 500 different intervals with an interval width w ranging from 0.001 s to 7 s. The initial search interval and initial guess are again defined as in (4.6). Figure 4.10 shows the number of iterations for both the binary search and Newton-Raphson's method for a tolerance TOL $= 10^{-6}$. The dashed line illustrates the number of safeguarding steps in Newton-Raphson's method. Because in this example the optimal time lies close to where the slope of the maximum travel range is zero, up to three safeguarding steps are required. Even in this case, the proposed algorithm outperforms binary search. The

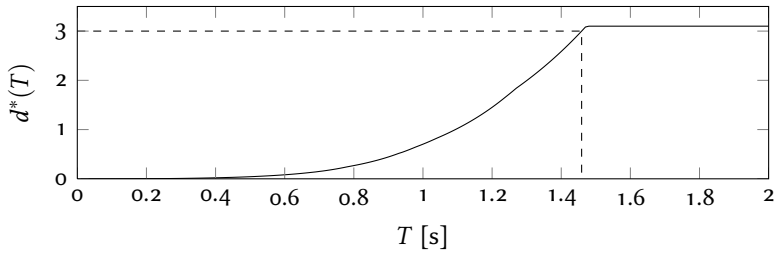

FIGURE 4.9.: Maximum travel range $d^*(T)$ as a function of the available time of the system (4.7) for a degree 4 spline with 10 internal knots. Note the sudden change in slope once the maximum travel range hits the bound on the output $z(t) \leq 3.1$

average computation time per iteration amounts to 0.0082 s for a bisection step and 0.010 s for a Newton step.

Compared to Suryawan, De Doná, and Seron (2011), much less conservatism is introduced by the parameterization and the linear constraints. To accomplish an optimal time $T^* = 1.459\,\text{s}$ for a degree 6 spline, the proposed approach only requires 10 knots (with two midpoint refinements for constraint imposition) whereas Suryawan, De Doná, and Seron (2011) require eighty knots. With eighty knots, our approach yields an optimal time $T^* = 1.394\,\text{s}$, which lies much closer to the theoretical minimum $T_{\text{opt}} = 1.382\,\text{s}$ using generalized bang-bang control.

Both benchmarks illustrate a significant performance increase of the proposed algorithm compared to classical binary search algorithms. Due to the higher computational effort of solving the semidefinite nonnegativity problem, the linear problem formulation is preferred. Moreover, by using a linear relaxation with two midpoint refinements, the amount of added conservatism is negligible.

4.2.3. Parametric solution to time-optimal control problems

Instead of iteratively searching for the optimal motion time, this section aims at determining a parametric solution to the maximum reach problem (4.3). By only solving a few optimization problems, the function $d(T)$ and corresponding solution can be determined analytically as a function

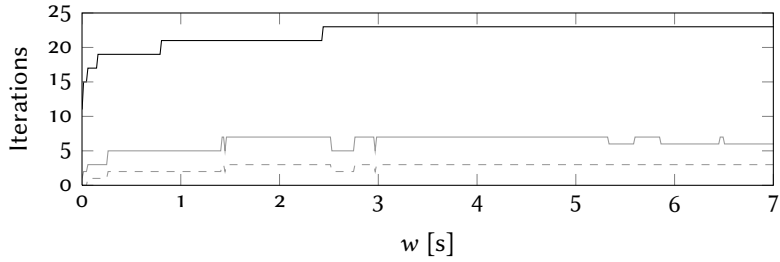

FIGURE 4.10.: Number of iterations for the binary search algorithm (black) and the proposed safeguarded Newton-Raphson's method (gray) as a function of the interval width w. The dashed line shows the number of safeguarding steps.

of the travel time T. This allows to precompute and store the solution offline.

Consider a linear program as in (4.4)[2] and its dual

$$\begin{aligned} d^*(T) = \underset{\lambda,\mu}{\text{minimize}} \quad & -b_{\text{eq}}^\intercal \lambda - b_{\text{in}}^\intercal \mu \\ \text{subject to} \quad & A_{\text{eq}}(T)^\intercal \lambda + A_{\text{in}}(T)^\intercal \mu + c = 0 \\ & \mu \leq 0. \end{aligned} \tag{4.8}$$

Assume a solution $x^*(T_i)$ to (4.4) at a specific value T_i. Denote by $A_{\text{in}}^i(T)$ the active constraint matrix at T_i and associated b_{in}^i, the right-hand side of the active constraints. Around T_i the solution can be parameterized as

$$x^*(T) = \begin{pmatrix} A_{\text{eq}}(T) \\ A_{\text{in}}^i(T) \end{pmatrix}^{-1} \begin{pmatrix} b_{\text{eq}} \\ b_{\text{in}}^i \end{pmatrix} = A^i(T)^{-1} b^i \tag{4.9}$$

and the optimal value

$$d^*(T) = c^\intercal x^*(T) = c^\intercal A^i(T)^{-1} b^i,$$

when $A^i(T)$ is invertible. Note that if the problem is well posed, $A^i(T)$ will usually be square. In the rare case of (4.9) being overly determined, a subset of active constraints must be chosen.[3]

[2] After omitting the semidefinite constraint

[3] In practice (4.9) will never be underdetermined as it implies that the optimization problem is unbounded.

The parametric solution only holds as long as (4.8) and (4.4) remain feasible, i.e.

$$A_{\text{in}}(T)A^i(T)^{-1}b^i \leq b_{\text{in}} \text{ and } -A^i(T)^{-\intercal}c \geq 0. \tag{4.10}$$

Alternatively, instead of requiring a matrix inverse, an expression for the optimal value can be found using determinants (Zuidwijk 2005):

$$\begin{aligned} 1 + d^*(T) &= 1 + c^\intercal A^i(T)^{-1}b^i \\ &= \det\left(1 + c^\intercal A^i(T)^{-1}b^i\right) \\ &= \det\left(I + A^i(T)^{-1}b^i c^\intercal\right) \\ &= \det\left(A^i(T)^{-1}\left(A^i(T) + b^i c^\intercal\right)\right) \\ &= \frac{\det\left(A^i(T) + b^i c^\intercal\right)}{\det\left(A^i(T)\right)}. \end{aligned} \tag{4.11}$$

In a similar fashion the rows of $x^*(T)$ and of the conditions in (4.10) can be found.

To illustrate the methodology consider again the system (4.5). To keep the problem dimension small and manageable on paper, consider a degree 4 spline with only two internal knots. Using the linear approach with no knot refinement and eliminating all equality constraints the following parametric linear program in two variables is derived

$$\begin{aligned} d^*(T) = \underset{x_1, x_2}{\text{maximize}} \quad & x_2 \\ \text{subject to} \quad & \left(\tfrac{1}{12}T^2 + \tfrac{23}{120}T + \tfrac{3}{8}\right)x_1 + \left(\tfrac{1}{60}T + \tfrac{3}{20}\right)x_2 \leq 2T^2 \\ & \tfrac{5}{9}Tx_1 + \left(\tfrac{2}{9}T + 1\right)x_2 \leq 3.1T \\ & \left(\tfrac{1}{3}T - \tfrac{17}{12}\right)x_1 + \left(\tfrac{2}{3}T + \tfrac{19}{12}\right)x_2 \leq 3.1T \\ & -\tfrac{1}{2}x_1 + \left(T + \tfrac{1}{2}\right)x_2 \leq 3.1T \\ & \left(\tfrac{1}{10}T - \tfrac{23}{40}\right)x_1 - \left(\tfrac{1}{2}T^2 + \tfrac{1}{10}T - \tfrac{5}{8}\right)x_2 \leq 2T^2. \end{aligned} \tag{4.12}$$

We want to determine the maximum reach curve $d^*(T)$ analytically for $T \in [0, 6]$. We first determine a solution for $T_0 = 6$ s. The third and fourth

constraint are active at the solution. Hence,

$$A^0(T) = \frac{1}{T}\begin{pmatrix} \frac{1}{3}T - \frac{17}{12} & \frac{2}{3}T + \frac{19}{12} \\ -\frac{1}{2} & T + \frac{1}{2} \end{pmatrix} \text{ and } b^0 = \begin{pmatrix} 3.1 \\ 3.1 \end{pmatrix}.$$

Using (4.11) we determine the maximum reach curve

$$d^*(T) = \frac{\det\left(A^0(T) + b^0 c^\intercal\right)}{\det\left(A^0(T)\right)} - 1 = \frac{124T^2 - 341T}{40T^2 - 110T + 10}.$$

Mind that this solution is only valid as long as the conditions in (4.10) are met. We find that for $T \leq 4.25\,\text{s}$ the dual solution is no longer feasible.

Next, we determine a new solution for $T_1 = 4\,\text{s}$, at which the second and third constraint are active. Similarly, we find

$$d^*(T) = \frac{744T^2 + 4743T}{320T^2 + 930T + 1530}$$

for $0.516\,\text{s} \leq T \leq 4.25\,\text{s}$, where the lower bound on T is the point where the primal solution is no longer feasible.

The process is repeated until the entire interval from 0 to 6 s has been covered. We end up with the piecewise function

$$d^*(T) = \begin{cases} \frac{124T^2-341T}{40T^2-110T+10} & \text{for } 4.25 < T \leq 6 \\ \frac{744T^2+4743T}{320T^2+930T+1530} & \text{for } 0.516 < T \leq 4.25 \\ \frac{-14700T^3+123390T^2+41850T}{2000T^3+8826T^2+18975T+29025} & \text{for } 0.369 < T \leq 0.516 \\ \frac{800T^4+880T^3+9120T^2}{-200T^4-500T^3-750T^2+369T+1539} & \text{for } 0 < T \leq 0.369 \end{cases}.$$

The function is shown in Figure 4.11. Note its resemblance with Figure 4.4, which shows the maximum range curve for a spline with 10 knots. For a given travel distance, the motion planning problem now boils down to a one-dimensional root finding problem to determine the corresponding optimal time T^*. Once T^* is known, the optimal solution can be evaluated. Such a strategy can easily be implemented on simple hardware.

We effectively determined the maximum reach function by solving only 4 linear programs. Unfortunately, for an increasing number of knots,

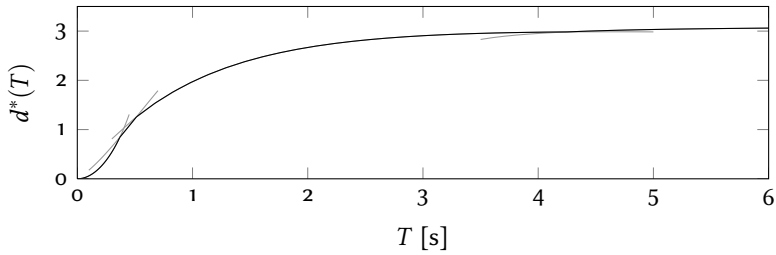

FIGURE 4.11.: By solving only 4 linear programs the maximum reach curve (in black) of problem (4.12) is computed parametrically as a function of T. The gray lines show the individual components of the maximum range curve.

the number of optimization problems to be solved grows quickly, as it depends on the number of active set changes for varying T. Moreover, calculating the solution explicitly for larger matrices, requires a symbolic computation of the determinant or the inverse, which for growing matrix size quickly becomes expensive. Alternatively, the active constraint set could be stored instead of its inverse. This way, the maximum range curve could still be evaluated numerically, without requiring the solution of an optimization problem.

Both the iterative as the parametric procedure effectively yield solutions to the time-optimal point-to-point motion problem. Although the iterative algorithm yields a significant performance increase compared to classical binary search, it might still be computationally too demanding for online use depending on the initial width of the bracketing interval. The parametric procedure, on the other hand, is ideally suited for online use since the entire map of solutions is computed offline. Here, the difficulty lies both in computing the map offline for a larger number of knots and efficiently determining the region of operation online.

4.3. Robust splines for dynamic systems

In the previous sections, we discussed how to design optimal inputs for a given system. In practice, however, the model of the system contains uncertainty. Applying such an optimized input to the uncertain system

can result in residual vibrations, which naturally are undesirable. This section considers the design of robust inputs which limit these residual vibrations for the entire uncertainty range of the system.

4.3.1. Residual vibration of dynamic systems

In a flexible motion system, input u and output z are in the simplest case related through a second order system:

$$\ddot{z}(t) + 2\zeta\omega_0\dot{z}(t) + \omega_0^2 z(t) = \omega_0^2 u(t),$$

where ω_0 represents the undamped resonance frequency of the system and ζ the corresponding damping ratio. We are interested to move the output over a finite distance while starting and stopping at rest. Therefore, the input consists of a rise portion for $t \leq T$ and a dwell portion for $t > T$, where the input is kept constant at a fixed value $\overline{u}$. During the dwell portion of the input, the output is a damped oscillation around $\overline{u}$ as illustrated in Figure 4.12. To quantify this residual oscillation, it is convenient to express the system in dimensionless coordinates as

$$\ddot{\chi}(\tau) + 2\zeta(2\pi\sigma)\dot{\chi}(\tau) + (2\pi\sigma)^2\chi(\tau) = (2\pi\sigma)^2\theta(\tau), \tag{4.13}$$

by introducing

$$\tau = \frac{t}{T}, \theta(\tau) = \frac{u(Tt)}{\overline{u}}, \chi(\tau) = \frac{z(Tt)}{\overline{u}} \text{ and } \sigma = T\frac{\omega_0}{2\pi}.$$

A good measure for the residual dwell vibration is the amplitude of its exponential envelope at $\tau = 1$ (Figure 4.12). It is given by (Thomson 1997)

$$A = \frac{1}{\sqrt{1-\zeta}}\sqrt{(\chi(1)-1)^2 + 2\zeta(\chi(1)-1)\left(\frac{\dot{\chi}(1)}{2\pi\sigma}\right) + \left(\frac{\dot{\chi}(1)}{2\pi\sigma}\right)^2}. \tag{4.14}$$

It is clear that the amplitude of residual vibration depends on σ, which relates to ω_0 and T, the damping factor ζ, and the position and velocity of the output at $\tau = 1$, which in turn depend on the input trajectory and again on σ and ζ.

This work assumes σ to be an uncertain parameter, which is often the case when working with identified data, especially for lightly damped

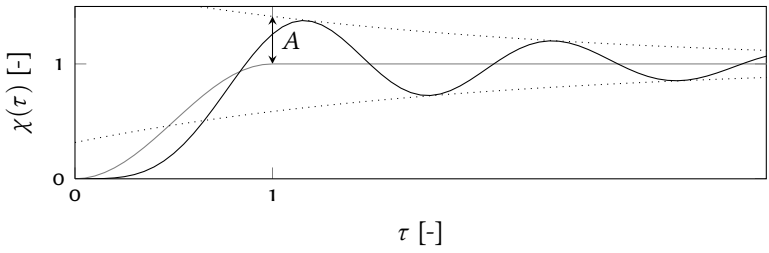

FIGURE 4.12.: During the dwell portion of the input, the output is a damped oscillation.

systems. We are interested in determining an input law $\theta(\tau)$ such that the amplitude of residual vibration is guaranteed to be below a threshold ε for all possible values of $\sigma \in [\underline{\sigma}, \overline{\sigma}]$:

$$A(\sigma) \leq \varepsilon. \tag{4.15}$$

Let $\chi_\sigma(\tau)$ and $\dot{\chi}_\sigma(\tau)$ denote a perturbed output position and velocity. Then, in Demeulenaere, De Caigny, et al. (2009) it is shown that (4.15) can be accomplished by the linear constraints

$$\begin{gathered} -\frac{\varepsilon}{\eta(\zeta)} \leq \chi_\sigma(1) - 1 \leq \frac{\varepsilon}{\eta(\zeta)}, \\ -2\pi\sigma\frac{\varepsilon}{\eta(\zeta)} \leq \dot{\chi}_\sigma(1) \leq 2\pi\sigma\frac{\varepsilon}{\eta(\zeta)}, \\ \text{for all } \sigma \in [\underline{\sigma}, \overline{\sigma}], \end{gathered} \tag{4.16}$$

with

$$\eta(\zeta) = \sqrt{\frac{2 + 2\zeta}{1 - \zeta^2}}.$$

4.3.2. A flatness based approach

In order to be able to impose the semi-infinite constraints (4.16) we choose to parameterize the output of the system at the nominal value of $\sigma = \sigma_0$ as a PP function

$$\chi_{\sigma_0} = \langle c_{\sigma_0}, b_{k,\lambda} \rangle.$$

Naturally, all outputs χ_σ are required to originate from the same input θ. Equation (4.13) together with a spline parameterization of the output yields

$$\langle\left(\frac{1}{2\pi\sigma}\right)^2 \ddot{b}_{k,\lambda}+\frac{2\zeta}{2\pi\sigma}\dot{b}_{k,\lambda}+b_{k,\lambda}, c_\sigma\rangle = \langle\left(\frac{1}{2\pi\sigma_0}\right)^2 \ddot{b}_{k,\lambda}+\frac{2\zeta}{2\pi\sigma_0}\dot{b}_{k,\lambda}+b_{k,\lambda}, c_{\sigma_0}\rangle,$$

with c_σ the spline coefficients of a perturbed output. Equivalently, by determining the spline collocation matrices (de Boor and Daniel 1974) $B_{k,\lambda}, \dot{B}_{k,\lambda}, \ddot{B}_{k,\lambda}$ at appropriate sites τ_i the above equation is written as a matrix-vector product.

$$\left(\left(\frac{1}{2\pi\sigma}\right)^2 \ddot{B}_{k,\lambda} + \frac{2\zeta}{2\pi\sigma}\dot{B}_{k,\lambda} + B_{k,\lambda}\right) c_\sigma =$$
$$\left(\left(\frac{1}{2\pi\sigma_0}\right)^2 \ddot{B}_{k,\lambda} + \frac{2\zeta}{2\pi\sigma_0}\dot{B}_{k,\lambda} + B_{k,\lambda}\right) c_{\sigma_0}$$

Moreover, by also eliminating the system's initial state explicitly from c_σ, the spline coefficients of a perturbed output can be uniquely determined by

$$\begin{aligned} c_\sigma = &\left(\left(\frac{1}{2\pi\sigma}\right)^2 \ddot{B}_{k,\lambda} + \frac{2\zeta}{2\pi\sigma}\dot{B}_{k,\lambda} + B_{k,\lambda}\right)^{-1} \cdot \\ &\left(\left(\frac{1}{2\pi\sigma_0}\right)^2 \ddot{B}_{k,\lambda} + \frac{2\zeta}{2\pi\sigma_0}\dot{B}_{k,\lambda} + B_{k,\lambda}\right) c_{\sigma_0} \end{aligned} \tag{4.17}$$

Constraints (4.16) can then be expressed as

$$\begin{aligned} -\frac{\varepsilon}{\eta(\zeta)} \le b_{k,\lambda}(1)^\intercal c_\sigma - 1 \le \frac{\varepsilon}{\eta(\zeta)}, \\ -2\pi\sigma\frac{\varepsilon}{\eta(\zeta)} \le \dot{b}_{k,\lambda}(1)^\intercal c_\sigma \le 2\pi\sigma\frac{\varepsilon}{\eta(\zeta)}, \\ \text{for all } \sigma \in [\underline{\sigma}, \overline{\sigma}], \end{aligned} \tag{4.18}$$

with c_σ as defined in (4.17). It is clear that these constraints are a rational function of the perturbation σ. We will express these constraints in the following general form

$$F(x, \Delta) \ge 0, \forall \Delta \in \mathcal{D},$$

where $F(x, \Delta)$ is linear in the decision variables x and rational in the uncertainty Δ, and $\mathcal{D}$ is a subspace of $\mathbb{R}^{p\times q}$. In the following section, well-known results from robust optimization are applied to determine linear matrix inequalities for the constraints (4.18) to hold for all possible σ.

4.3.3. Robust solutions using semidefinite programming

We will use following well-known result which traces back to Packard and Doyle (1993)

Lemma 4.1 · *Let $F = F^\intercal, L, R, D$ be real matrices of appropriate size and $\Delta = \delta I$. Let $\mathcal{S}$ (resp. $\mathcal{G}$) be the set of symmetric (resp. skew-symmetric) matrices that commute with each element of Δ.*

We have $\det(I - D\Delta) \neq 0$ *and*

$$F + L\Delta(I - D\Delta)^{-1}R + R^\intercal(I - D\Delta)^{-T}\Delta^\intercal L^\intercal \geq 0 \tag{4.19}$$

for every Δ, if and only if there exist $S \in \mathcal{S}$ and $G \in \mathcal{G}$ such that $S \succ 0$ and

$$\begin{pmatrix} F - LSL^\intercal & R^\intercal - LSD^\intercal + LG \\ R - DSL^\intercal - GL^\intercal & \rho^{-2}T - GD^\intercal + DG - DSD^\intercal \end{pmatrix} \succ 0. \tag{4.20}$$

A more general formulation of the lemma, in which $\Delta \in \mathbb{R}^{n\times n}$, can be found in El Ghaoui and Lebret (1997) and El Ghaoui, Oustry, and Lebret (1998).

In order to apply Lemma 4.1, a linear-fractional representation, as in (4.19), for each of the constraints in (4.18) is determined. First, we transform the uncertainty parameter

$$\frac{1}{2\pi\sigma} = \delta_0 + \delta,$$

with $\delta_0 = \frac{1}{2\pi\sigma_0} = \frac{1}{4\pi\overline{\sigma}} + \frac{1}{4\pi\underline{\sigma}}$ and $\delta \in [\frac{1}{4\pi\overline{\sigma}} - \frac{1}{4\pi\underline{\sigma}}, \frac{1}{4\pi\underline{\sigma}} - \frac{1}{4\pi\overline{\sigma}}]$. For notational convenience, we define the matrices

$$N_0 = \delta_0^2\ddot{B}_{k,\lambda} + 2\zeta\delta_0\dot{B}_{k,\lambda} + B_{k,\lambda}, N_1 = 2\delta_0\ddot{B}_{k,\lambda} + 2\zeta\dot{B}_{k,\lambda} \text{ and } N_2 = \ddot{B}_{k,\lambda}.$$

Then, following linear fractional representations for c_σ and $\frac{c_\sigma}{2\pi\sigma} = (\delta_0 + \delta)c_\sigma$ can then be derived (see e.g. Zhou and Doyle (1998))

$$c_\sigma = \left(I + \begin{pmatrix} I & 0 \end{pmatrix} \Delta \left(I - \begin{pmatrix} -N_0^{-1}N_1 & I \\ -N_0^{-1}N_2 & 0 \end{pmatrix} \Delta \right)^{-1} \begin{pmatrix} -N_0^{-1}N_1 \\ -N_0^{-1}N_2 \end{pmatrix} \right) c_{\sigma_0} \tag{4.21}$$

and

$$(\delta_0+\delta)c_\sigma = \left(\delta_0 I + \begin{pmatrix} I & 0 \end{pmatrix} \Delta \left(I - \begin{pmatrix} -N_0^{-1}N_1 & I \\ -N_0^{-1}N_2 & 0 \end{pmatrix} \Delta \right)^{-1} \begin{pmatrix} I - \delta_0 N_0^{-1}N_1 \\ -\delta_0 N_0^{-1}N_2 \end{pmatrix} \right) c_{\sigma_0}, \tag{4.22}$$

with $\Delta = \delta I$. By multiplying the above equations (4.21) and (4.22) with respectively $b_{k,\lambda}(1)^\intercal$ and $\dot{b}_{k,\lambda}(1)^\intercal$, the linear fractional representations for the constraints in (4.18) follow straightforwardly. Hence, the semi-infinite constraints (4.18) can be imposed by repeatedly applying the LMI (4.20).

4.3.4. *Example*

To illustrate the developed methodology, consider an uncertain dimensionless second order system (4.13) with $\zeta = 0.01$ and $\sigma \in [1.7, 2.3]$. A robust input is determined for $\varepsilon = 0.02$. Hereto, the output is parameterized as a degree 4 spline with 9 internal knots. Using (4.21) and (4.22) a linear fractional representation for each of the constraints in (4.18) is determined. Subsequently, these semi-infinite constraints are imposed using (4.20). Solving the LMI with a feasibility problem results in the robust input shown in Figure 4.13 (top). The corresponding outputs for $\sigma \in [1.7, 2.3]$ are all contained in the black region of Figure 4.13 (bottom).

The proposed approach effectively avoids sampling of the uncertainty region. However, for an increasing number of knots the computations quickly become too expensive. For the chosen parameterization, four matrix inequalities of dimension 21 and four of dimension 20 have to be solved, which is already considered as a large LMI problem. Moreover, when adding a performance objective instead of solving a feasibility problem the solver often fails to converge as the solution is pushed to the edge of feasibility. This is a fundamental problem of current semidefinite solvers.

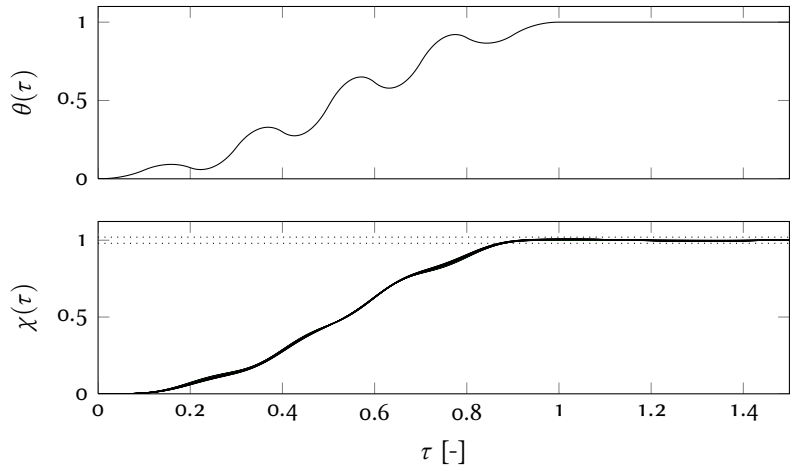

FIGURE 4.13.: A degree 4 robust input with 9 internal knots (top) for an uncertain second order system with $\zeta = 0.01$ and $\sigma \in [1.7, 2.3]$. The corresponding set of outputs is shown in black (bottom).

This section introduced a novel method for computing robust inputs. By parameterizing all possible outputs and constraining them to originate from the same input, classical reachability analysis is avoided. For a second order system with uncertainty on the resonance frequency, a robust counterpart is derived. The proposed method avoids sampling of the uncertainty region. A numerical example illustrated the opportunities of the approach. However, we are currently unable to fully reap its rewards due to the computational issues related to solving the LMI's.

4.4. Summary

This chapter illustrates the power and versatility of splines for solving optimal control problems for linear systems. By using a spline parameterization of the flat output, the optimal control problem is cast as a piecewise polynomial nonnegativity problem. As seen in Chapter 2, such constraints can be imposed either using exact semidefinite conditions or conservative linear conditions. By using the linear approach with mid-

point refinements, the amount of conservatism is greatly reduced. This approach is shown to outperform other recently proposed spline-based methods (Suryawan, De Doná, and Seron 2011, Suryawan, De Doná, and Seron 2012) both in terms of optimality and efficiency.

Instead of resorting to classical binary search for time-optimal control problems, a Newton-Raphson iteration scheme is proposed based on a maximum reach problem. This approach results in a significant decrease in computational effort. By using a limited number of knots the true time-optimal generalized bang-bang solution is approximated by smooth input signals. Moreover, for a limited problem dimension a time-optimal solution can also be determined parametrically.

A novel approach for robust input design parameterizes all possible flat outputs and constrains them to originate from the same input. This way, a robust counterpart for the robust input design problem for uncertain systems is derived, such that sampling of the uncertainty region as in De Caigny et al. (2008), De Caigny (2009) is avoided.

What to remember

- For linear systems, the general optimal control problem can be cast as a piecewise polynomial nonnegativity problem.
- Using the linear relaxation with two midpoint refinements gives almost identical results to the exact semidefinite conditions.
- By formulating a maximum range problem instead of the classical feasibility problem for time-optimal point-to-point motions, sensitivity information can be used for the iterative computation of the optimal time.
- By regarding the time-optimal problem as a parametric optimization problem, the maximum range curve can also be expressed analytically by solving a finite number of linear programs.
- A new approach for robust input design parameterizes all possible outputs and constrains them to originate from the same input, allowing for the derivation of a robust counterpart.

5

Optimal path following for differentially flat systems through a geometric transformation

As opposed to trajectory tracking where a reference specified in time is tracked, path following deals with the problem of following a geometric path without any preassigned timing information. Many industrial tasks are in fact path following problems, such as robot path following in e.g. welding or painting (Shin and McKay 1985, Verscheure, Demeulenaere, et al. 2009, Bobrow, Dubowsky, and Gibson 1985, Debrouwere et al. 2013), control of autonomous vehicles (Skjetne, Fossen, and Kokotović 2004, Hoffmann, Wasl, and Tomlin 2008), but also control of chemical reactors (Faulwasser, Hagenmeyer, and Findeisen 2011) and batch crystallization (Nagy 2008). In addition path following is often considered as the low level stage in a decoupled motion planning problem (Shin and McKay 1985, Bobrow, Dubowsky, and Gibson 1985). In a first step, a high level planner determines a geometric path accounting only for geometric path constraints. Subsequently, an (optimal) velocity profile along the geometric path that takes into account the system dynamics and limitations, such as actuator saturation, is determined.

Most path following methods employ the fact that motion along a path can be described by a single coordinate, commonly denoted by s. For differentially flat systems, by applying the chainrule, the dynamics are projected along the path onto a single-input system. Using this projection, Raczy and Jacob (1999) derive a small dimensional optimal control problem for an overhead crane. In Faulwasser, Hagenmeyer, and Findeisen (2011) this approach is generalized for differentially flat systems. Furthermore, the authors provide easy-to-check conditions for determining whether a

path is exactly followable. Despite its small dimension, the optimization problem remains difficult to solve and requires a good initialization, especially for free end-time problems. Moreover, the optimization variables enter through the definition of the path into the optimization problem. Hence, even for linear systems, a nonlinear reference path quickly complicates the problem, as shown in Section 5.1.

These problems are largely overcome by assuming a nonnegative velocity along the path such that time can be eliminated from consideration and the coordinate s can be used as independent variable (Hauser and Saccon 2006), thus, arriving at a geometric problem formulation. A similar transformation is used in Verscheure, Demeulenaere, et al. (2009). In their work a clever time transformation renders the time-optimal path following problem convex for the case of a simplified robotic manipulators. In Debrouwere et al. (2013), the authors apply sequential convex programming to broaden the set of models and constraints. The method however relies on finding a convex-concave decomposition of the constraints, which is difficult in many cases, hereby limiting its applicability. This chapter presents a generalization of the approach of Verscheure, Demeulenaere, et al. (2009) for differentially flat systems using the results from Faulwasser, Hagenmeyer, and Findeisen (2011). Although in general the convexity of the problem is lost, the proposed problem formulation remains appealing since (i) the problem dimension remains small, (ii) the optimization variables no longer enter through the definition of the path, (iii) the problem is transformed into a fixed end-time problem and (iv) numerical experiments reveal that the solver no longer requires an accurate initial guess and computation times remain comparable to the convex problem as obtained in Verscheure, Demeulenaere, et al. (2009). Moreover, the proposed problem formulation allows for an intuitive understanding of the conditions for which a path is exactly followable.

Although path following has many applications it is restrictive to constrain the output to follow a predetermined path. Therefore, in the second part of this chapter the proposed path following framework is extended to allow for freedom in the geometric path. To this end, the geometric reference is represented as an unknown convex combination of two or more fixed boundary paths. In this way, optimal paths for differentially flat systems can be determined as well.

The content of this chapter is based on the papers Van Loock, Pipeleers, Diehl, et al. (2013), Van Loock, Bellens, et al. (2013), Van Loock, Pipeleers, and Swevers (2013a), Van Loock, Pipeleers, and Swevers (2013b).

5.1. Problem statement

Consider a desired geometric reference $y_d(s) \in C^r$, a parametrized curve as a function of a scalar path coordinate s for the flat output.[1] The time dependency follows from $s(t)$. Without loss of generality, it is assumed that the trajectory starts at $t = 0$, ends at $t = T$ and $s(0) = 0 \leq s(t) \leq s(T) = 1$. Furthermore, it is assumed that the velocity along the path is non-negative, i.e. $\dot{s}(t) \geq 0$, that the system performs a rest-to-rest motion and that the boundary conditions x_0, x_T are consistent with the reference path, such that $x_0 = \Phi(y_d(s(0)), 0, \ldots, 0)$ and $x_T = \Phi(y_d(s(T)), 0, \ldots, 0)$.

In this Chapter, the goal is to determine an input signal $u(t)$ such that (i) the desired geometric reference is followed exactly by the flat output, i.e.

$$y(t) = y_d(s(t)), \tag{5.1a}$$

(ii) constraints on states and inputs

$$x(t) \in \mathcal{X} \text{ and } u(t) \in \mathcal{U}, \tag{5.1b}$$

are respected and (iii) the cost function

$$J = T + \int_0^T F(x(\tau), u(\tau))d\tau \tag{5.1c}$$

is minimized, where the duration T is weighed against a performance criterion $F(\cdot)$, e.g. energy consumption or other regularization terms.

For an arbitrary nonlinear system this problem is challenging to solve as it requires integrating a nonlinear state space model and imposing the

[1] Recall from Definition 1.2 that r is the number of derivatives of the flat output required to describe the input u.

algebraic equations $y(t) = y_d(s(t))$. However, when considering differentially flat systems, the system dynamics can be projected along the path onto a linear single-input system that is trivial to integrate (Faulwasser, Hagenmeyer, and Findeisen 2011). By applying the chainrule, we find the time derivatives of $y_d(s(t))$

$$\begin{aligned} \dot{y}_d &= \partial_s y_d \dot{s}, \\ \ddot{y}_d &= \partial_s^2 y_d \dot{s}^2 + \partial_s y_d \ddot{s}, \end{aligned} \tag{5.2}$$

and so on. Then by substituting (5.2) into (1.3) we rewrite the states and inputs of the system as

$$\begin{aligned} x &= \Phi_s(s, \dot{s}, \ldots, \partial_t^{r-1} s) = \Phi_s(\sigma) \\ u &= \Psi_s(s, \dot{s}, \ldots, \partial_t^{r} s) = \Psi_s(\sigma, v), \end{aligned}$$

where $\sigma(t) = (\sigma_0, \ldots, \sigma_{r-1})^\intercal = (s, \dot{s}, \ldots, \partial_t^{r-1} s)^\intercal$ and $v(t) = \partial_t^r s$. The rest-to-rest path following problem can now be reformulated as the smaller dimensional optimal control problem with states σ and control v

$$\begin{aligned} \underset{\sigma(\cdot), v(\cdot), T}{\text{minimize}} \quad & T + \int_0^T F(\sigma(\tau), v(\tau)) d\tau \\ \text{subject to} \quad & \dot{\sigma}(t) = \begin{pmatrix} 0 & I_{r-1} \\ 0 & 0 \end{pmatrix} \sigma(t) + (0, \ldots, 0, 1)^\intercal v(t) \\ & \sigma(0) = (0, \ldots, 0)^\intercal, \; \sigma(T) = (1, 0 \ldots, 0)^\intercal \\ & T \geq 0 \\ & \sigma_1(t) \geq 0, \forall t \in [0, T] \\ & \Phi_s(\sigma(t)) \in \mathcal{X}, \; \forall t \in [0, T] \\ & \Psi_s(\sigma(t), v(t)) \in \mathcal{U}, \; \forall t \in [0, T], \end{aligned} \tag{5.3}$$

Although the projection simplifies the problem (5.1) to a great extent, it suffers from some important drawbacks.

1. The variable $\sigma_0 = s$ enters the optimization problem through $y_d(s(t))$. Problems with nonlinear paths (i.e. nonlinear functions $y_d(\cdot)$) will therefore introduce more nonlinearity into the problem.

2. The problem is a free end-time optimal control problem for which the solution can vary quite nonlinearly with changes in T, which usually results in slower convergence compared to fixed end-time problems.

3. Due to the above two reasons, an accurate initial guess is often needed to ensure convergence.

In the following section we show how to overcome these difficulties by a transformation of variables.

5.2. Time transformation

The key idea is to transform the problem such that, instead of the time t, the path coordinate s becomes the independent variable. In this way, the problem is transformed into a fixed end-time optimal control problem. Moreover, the optimization variables no longer enter the problem through the reference path $y_d(s)$. The transformation, proposed in Verscheure, Demeulenaere, et al. (2009), consists of parameterizing the velocity along the path as a function of the path coordinate:

$$b(s) := \dot{s}^2. \tag{5.4}$$

By taking the time derivative on both sides of (5.4), we obtain $\partial_s b(s)\dot{s} = 2\ddot{s}\dot{s}$ or

$$\ddot{s} = \frac{\partial_s b(s)}{2}.$$

We also require derivatives of higher order, which are obtained by repeatedly applying the chainrule

$$\partial_t^3 s = \frac{\partial_s^2 b(s)\dot{s}}{2} = \frac{\partial_s^2 b(s)\sqrt{b(s)}}{2},$$
$$\partial_t^4 s = \frac{\partial_s^2 b(s)\ddot{s}}{2} + \frac{\partial_s^3 b(s)\dot{s}^2}{2} = \frac{\partial_s^2 b(s)\partial_s b(s)}{4} + \frac{\partial_s^3 b(s) b(s)}{2},$$

and so on.

States and inputs can now be reformulated as a function of s and $b(s)$ and its derivatives:

$$\begin{aligned} x &= \Phi_b(s, b, \partial_s b, \ldots, \partial_s^{r-2} b) = \Phi_b(s, \beta) \\ u &= \Psi_b(s, b, \partial_s b, \ldots, \partial_s^{r-1} b) = \Psi_b(s, \beta, w), \end{aligned}$$

where

$$\beta(s) = (\beta_0, \ldots, \beta_{r-2})^\intercal = (b, \partial_s b, \ldots, \partial_s^{r-2} b)^\intercal \text{ and } w(s) = \partial_s^{r-1} b. \quad (5.5)$$

Furthermore, using $dt = \frac{ds}{\dot{s}} = \frac{ds}{\sqrt{b}}$, the objective function is reformulated as

$$J = \int_0^T 1 + F(x(\tau), u(\tau)) d\tau = \int_0^1 \frac{1 + F_b(s, \beta(s), w(s))}{\sqrt{\beta_0(s)}} ds. \quad (5.6)$$

With the transformation (5.4), problem (5.3) is reformulated as the optimal control problem with pseudotime s, differential states β and control w

$$\begin{aligned} \underset{\beta(\cdot), w(\cdot)}{\text{minimize}} \quad & \int_0^1 \frac{1 + F_b(s, \beta(s), w(s))}{\sqrt{\beta_0(s)}} ds \\ \text{subject to} \quad & \partial_s \beta(s) = \begin{pmatrix} 0 & I_{r-2} \\ 0 & 0 \end{pmatrix} \beta(s) + (0, \ldots, 0, 1)^\intercal w(s) \\ & \beta_0(s) \geq 0, \forall s \in [0, 1] \\ & \Phi_b(s, \beta(s)) \in \mathcal{X}, \ \forall s \in [0, 1] \\ & \Psi_b(s, \beta(s), w(s)) \in \mathcal{U}, \ \forall s \in [0, 1]. \end{aligned} \quad (5.7)$$

The initial and terminal values for β are omitted deliberately. These will be discussed in more detail in Section 5.2.1.

Our problem formulation holds several advantages over (5.3). First, note that (5.7) has one differential state fewer than (5.3). Therefore, it contains fewer optimization variables. Second, the optimal control problem has a fixed (pseudo) end-time, namely $s = 1$, which simplifies the problem to great extent. Third, as s is the independent variable, the optimization variables no longer enter the problem through $y_d(s)$. Instead, only the

coefficients of the terms in Φ_b and Ψ_b depend on the reference path. Therefore, nonlinear paths will not introduce more nonlinearity.

Problem formulation (5.7) can be seen as a generalization of the path following problem for robotic manipulators, described in Verscheure, Demeulenaere, et al. 2009. In their formulation, $k = 2$, Φ_b and Ψ_b are linear in the optimization variables, $\mathcal{X}$ and $\mathcal{U}$ are convex and only $F(\beta, w)$ that result in a convex objective are allowed. For this specific case, the optimization problem is convex and is solved globally and efficiently. As we will illustrate in the examples below, the more general problem can also be solved efficiently, but global optimality cannot be guaranteed.

5.2.1. *Singularities*

In order for the system to perform a rest-to-rest transition, the initial and terminal states must be set to zero. This could be accomplished by imposing zero initial and terminal values for β in (5.7). However, these constraints will often result in the integral (5.6) being undefined, as shown in the proposition below.

Lemma 5.1 · *If $b(0) = 0$, $\partial_s b(0) = 0$ and $\partial_s^2 b(0) = c$ then the integral*

$$\int_0^1 \frac{1}{\sqrt{b(s)}} ds$$

is undefined.

Proof. Consider $b(s)$ near 0. Since $\partial_s^2 b(0) = c$, there always exists an η such that $b(s) \leq cs^2$ on $[0, \eta]$ or, equivalently

$$\frac{1}{\sqrt{b(s)}} \geq \frac{1}{\sqrt{c}s} \text{ for } s \in [0, \eta].$$

Consequently

$$\int_0^\eta \frac{1}{\sqrt{b(s)}} ds \geq \int_0^\eta \frac{1}{\sqrt{c}s} ds = \infty.$$

and hence the integral $\int_0^1 \frac{1}{\sqrt{b(s)}} ds$ is undefined. □

In other words, from $r \geq 3$ onwards, zero initial or terminal values for β render (5.6) undefined and hence they cannot be imposed. Therefore, rest-to-rest conditions are imposed indirectly by choosing a suitable parametrization for $y_d(s)$. Indeed, by ensuring $\partial_s^i y_d(0) = 0$ for $i = 1, \ldots, r-1$, we can impose $\partial_t^i y_d(s(0)) = 0$ regardless of the value of $\beta(0)$ (cfr. (5.2)). Any path $y_d(s)$ can easily be reparameterized by $y_d(q(s))$ with $q(\cdot)$ an odd degree polynomial of sufficient degree such that $q(0) = 0$, $q(1) = 1$, $\partial_s q(s) \geq 0$ for $s \in [0, 1]$ and $\partial_s^i q(0) = \partial_s^i q(1) = 0$ for $i = 1, \ldots, r-1$. In the following section we use this reparameterization to prove under which conditions a given reference is exactly followable.

5.2.2. Path followability

Sufficient conditions to determine whether a given geometric reference is exactly followable are already derived in Faulwasser, Hagenmeyer, and Findeisen 2011. However, by using the proposed time transformation and a suitable reparameterization of the path these conditions allow for an intuitive understanding and a simplification of the proof. Therefore, we repeat them here.

Theorem 5.1 (Exact path followability) · *Assume that the system* (1.2) *starts and stops in steady state and the maps* Φ_b *and* Ψ_b *are continuous. If for all* $s \in [0, 1]$

$$\Phi_b(s, 0, \ldots, 0) \in \operatorname{int}(\mathcal{X}) \text{ and } \Psi_b(s, 0, \ldots, 0) \in \operatorname{int}(\mathcal{U}), \tag{5.8}$$

then $y_d(s)$ *can be followed exactly by the flat system* (1.2)*. Furthermore, the minimal transition time is finite for the time-optimal case.*

Proof. The proof relies on finding a feasible solution to (5.7). First, the path is reparameterized such that the system starts and stops in steady state, as described in the previous section.

Now choose $b(s) = \epsilon > 0$, where the constant ϵ is chosen small enough such that for all $s \in [0, 1]$, due to (5.8) and the continuity of the maps Φ_b and Ψ_b,

$$\Phi_b(s, \epsilon, 0, \ldots, 0) \in \mathcal{X}$$

and

$$\Psi_b(s, \epsilon, 0, \ldots, 0) \in \mathcal{U}.$$

Hence, $b(s) = \epsilon$ yields a feasible solution to (5.7). Moreover, when considering the time-optimal case, i.e. $F(\cdot) = 0$, the minimal transition time $T^* \leq 1/\sqrt{\epsilon}$ is finite. □

Intuitively, the theorem states that $y_d(s)$ is exactly followable if each point of the reference path can be visited in steady state while staying in the interior of the constraint set. The proof shows this can be accomplished by traveling with very low constant speed $b(s) = \epsilon$ along the path. Note the theorem only provides a sufficient condition.

5.3. Examples

The robotic manipulator and the quadrotor serve as illustration for the proposed framework. We consider a robotic manipulator with and without viscous friction in the joints. In the latter case the optimization problem is convex. We show that calculation times remain comparable for both cases. For highly nonlinear systems as the quadrotor, solutions are also calculated efficiently without requiring a user-defined initial guess. Instead, we rely on automatic initialization of the solver. As the time-optimal problem is considered more difficult, we consider the case $F = 0$ in (5.1c), in both examples.

To simplify defining path following problems, a software package, based on CasADi (Andersson, Åkesson, and Diehl 2012) and Ipopt (Wächter and Biegler 2006) is developed. All problems are discretized using direct transcription (Betts 2001). A tutorial for the software package is provided in Appendix A. For future benchmarking, the code to solve the examples is also made available. The problems are solved on a 2.66 GHz PC with 4 GB of RAM memory. The computation times are reported as the sum of the time spent in Ipopt and in nonlinear functions calls.

5.3.1. Robotic manipulator

The equations of motion of an n-DOF robotic manipulator with joint angles $q \in \mathbb{R}^n$, can be written as a function of the applied joint torques $\tau \in \mathbb{R}^n$ (Spong and Hutchinson 2005)

$$\tau = D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q),$$

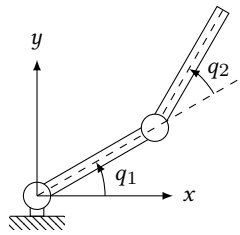

FIGURE 5.1.: Planar elbow manipulator with joints q_1 and q_2

where $D(q)$ is a positive definite mass matrix, $C(q, \dot{q})$ is a matrix accounting for Coriolis and centrifugal effects and $g(q)$ denotes the gravity vector. Verscheure, Demeulenaere, et al. (2009) make the critical assumption that $C(q, \dot{q})$ is linear in the joint velocities, which renders their problem convex. However, when a matrix $B(q)$ accounting for viscous friction in the joints is added, we find

$$\tau = D(q)\ddot{q} + (C(q, \dot{q}) + B(q))\dot{q} + g(q).$$

Now $C(q, \dot{q}) + B(q)$ is affine in $\dot{q}$ and convexity of the path following problem is destroyed.

In this example, we consider a two-link planar elbow manipulator in a vertical plane, as in Fig. 5.1, with and without a viscous friction matrix $B(q)$, which allows us to compare the convergence of both the convex and nonconvex problem for the same problem size. Obviously, the joint angles $(q_1, q_2)^\mathsf{T}$ are a flat output for this system. We consider the geometric reference for both joints

$$y_d(s) = \left(\frac{\pi}{2}s, -\pi s\right)^\mathsf{T}, \tag{5.9}$$

such that for a robot with equal link lengths the end-effector traces a line along the x-axis. In order to start with zero joint velocity the reference is reparameterized as described in Section 5.2.1. The joint torques are constrained to

$$\tau \in [-20, 20]\,\mathrm{N\,m} \times [-10, 10]\,\mathrm{N\,m}.$$

The discretized problem contains 400 variables, 199 equality and 400 inequality constraints. The solutions of both the convex and nonconvex

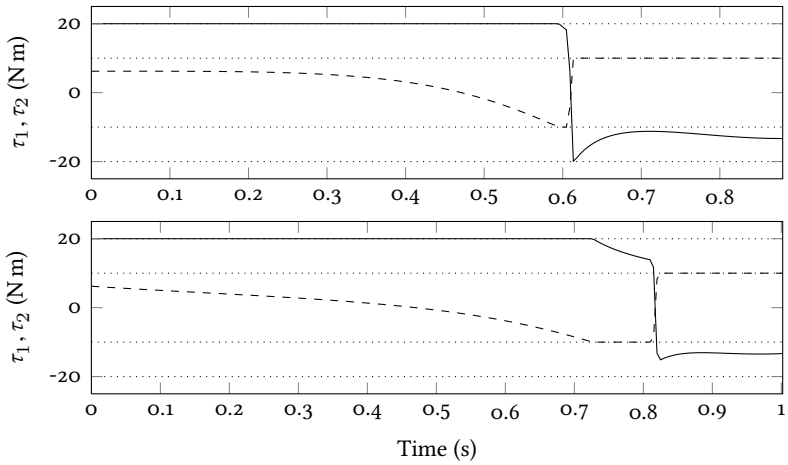

FIGURE 5.2.: Optimized torques (τ_1: solid, τ_2: dashed) for a planar elbow manipulator with (bottom) and without (top) viscous friction in the joints following the reference (5.9)

problem are obtained in 24 iterations or 0.12 s. For the convex problem 0.036 s are spent in Ipopt and 0.084 s in nonlinear function calls, whereas for the nonconvex problem the computation times are 0.032 s and 0.088 s respectively. In our approach there is little difference in computation times whereas in Debrouwere et al. 2013, where a sequential convex programming approach is followed, the cost for solving the nonconvex problem is approximately four times the cost of the convex problem.

The optimal actuator torques are shown in Fig. 5.2. Both for the convex (top) and nonconvex (bottom) problem, at least one of the constraints is active at each time step, which is a typical property for time-optimal solutions. Also note that the execution time for the nonconvex problem is slightly larger due to the viscous friction in the joints.

The convergence of problem (5.7) is compared to (5.3) by increasing the nonlinearity of the reference path. To this end, we define a nonlinear recursive function

$$r_i(s) = \frac{1}{1 + e^{\cos \pi r_{i-1}(s)}}, \tag{5.10}$$

with $r_0(s) = s$ and the reference path $y_d(s) = (r_i(s), -2r_i(s))^\intercal$. Table 5.1

TABLE 5.1.: Number of iterations for formulations (5.3) and (5.7) of the path following problem with path $y_d(s) = (r_i(s), -2r_i(s))$ and $r_i(s)$ as defined in (5.10)

i	Problem (5.3)	Problem (5.7)
1	97	28
2	38	32
3	78	30
4	79	32
5	91	31
6	94	32
7	64	34
8	–	32
9	105	32
10	189	34

compares the number of iterations for different values of i for both problem formulations. It is clear that our approach consistently outperforms (5.3). Moreover, the number of required iterations hardly changes, whereas for (5.3) it varies strongly with an increasing trend for increasing i, though with some outliers. For $i = 8$, the solver was unable to converge for formulation (5.3). These findings confirm that, contrary to Faulwasser, Hagenmeyer, and Findeisen 2011, in our approach the nonlinearity of the path has little influence on the convergence.

5.3.2. Quadrotor

To illustrate that the proposed framework copes well with strongly nonlinear problems as well, this section considers aggressive time-optimal maneuvers for quadrotors along a predetermined path. Figure 5.3 shows the notation used. The coordinate $(x, y, z)^\intercal$ represents the coordinate of the quadrotor's center of gravity with respect to a world frame and the XYZ Euler angles $(\phi, \theta, \psi)^\intercal$ denote the roll, pitch and yaw angle. A flat output for the quadrotor is $(x, y, z, \psi)^\intercal$. The reader is referred to Mellinger and Kumar 2011, for further information concerning flatness. The control input of the quadrotor $u = (u_1, u_2, u_3, u_4)^\intercal$ consists of the net body force u_4 and the three body moments u_1, u_2, u_3 and can also be related to the angular velocities of the rotors Mellinger and Kumar 2011.

A geometric reference trajectory is usually planned only in $(x, y, z)^\intercal$ and therefore allows some freedom in the reference trajectory for the

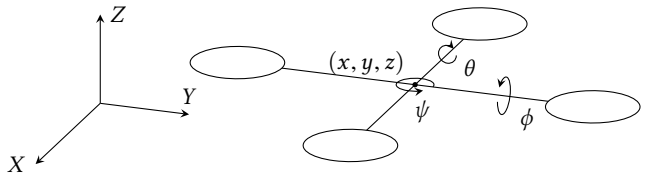

FIGURE 5.3.: The quadrotor with coordinates (x, y, z) of the center of gravity and roll, pitch, yaw angles (θ, ϕ, ψ)

yaw angle ψ. One possibility is to align one of the quadrotor's arm with the tangent dy/dx, i.e. $\psi = \arctan \frac{\partial_s y(s)}{\partial_s x(s)}$. In this example we consider following reference

$$y_d(s) = \left(\cos(2\pi s), \sin(2\pi s), \left(0.9(e^s - 1) + 0.1\sin(2\pi s)\right)^2, 2\pi s\right)^\mathsf{T} \quad (5.11)$$

The reference is reparameterized as described in Section 5.2.1 such that the quadrotor's initial and terminal states are zero. The inputs are constrained to[2]

$$u \in [-8, 8]\,\mathrm{N\,m} \times [-8, 8]\,\mathrm{N\,m} \times [-8, 8]\,\mathrm{N\,m} \times [1, 32]\,\mathrm{N}.$$

A solution is obtained in 22 iterations or 5.824 s, of which 0.12 s in Ipopt and 5.704 s in nonlinear function calls for a problem with 750 variables, 600 inequality and 596 equality constraints. The optimized control inputs are shown in Fig. 5.5. Note that at each time instant one of the constraints is active. Fig. 5.4 illustrates the position and orientation of the quadrotor for 20 equidistant time steps.

5.3.3. *Discussion*

An efficient problem formulation is derived in two steps. First, the differential flatness property allows to project the system dynamics along the path onto a linear single-input system, resulting in a small dimensional

[2]Note that the rotational speed of the rotors are the true constraints for the system. These are related to the body forces and moments through a linear transformation matrix (Mellinger and Kumar 2011).

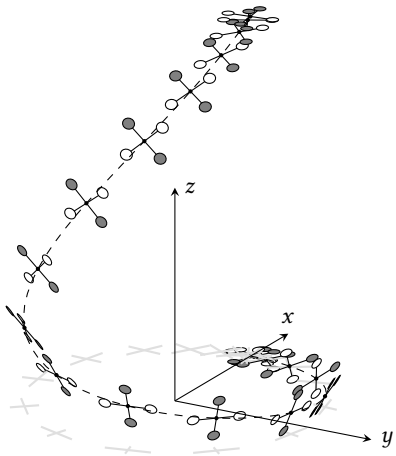

FIGURE 5.4.: Quadrotor's position and orientation in 20 equidistant time steps

optimal control problem. Despite its small dimension, the optimization problem remains difficult to solve and often requires a good initialization, especially for free end-time problems. Moreover, the optimization variables enter through the definition of the path into the optimization problem, making the convergence dependent on the geometric reference path. In a second step, a nonlinear change of variables that expresses the problem in terms of the progress along the path overcomes the former issues. These steps are summarized in Figure 5.7.

Compared to the problem formulation (5.3) from Faulwasser, Hagenmeyer, and Findeisen (2011), we believe our geometric problem formulation has two important advantages. First, the problem is transformed into a fixed end-time problem, hereby simplifying the problem to a large extent. Second, the optimization variables no longer enter the problem through $y_d(s)$, as the path coordinate s becomes the independent variable instead of the time. Numerical experiments illustrate that our formulation shows little dependence on the reference path as opposed to Faulwasser, Hagenmeyer, and Findeisen (2011).

Both numerical examples illustrate that solutions are obtained efficiently within a few CPU seconds, making the method practical in academic and industrial practice. Moreover, we rely on the automatic initialization of Ipopt and the user is not required to supply an initial guess,

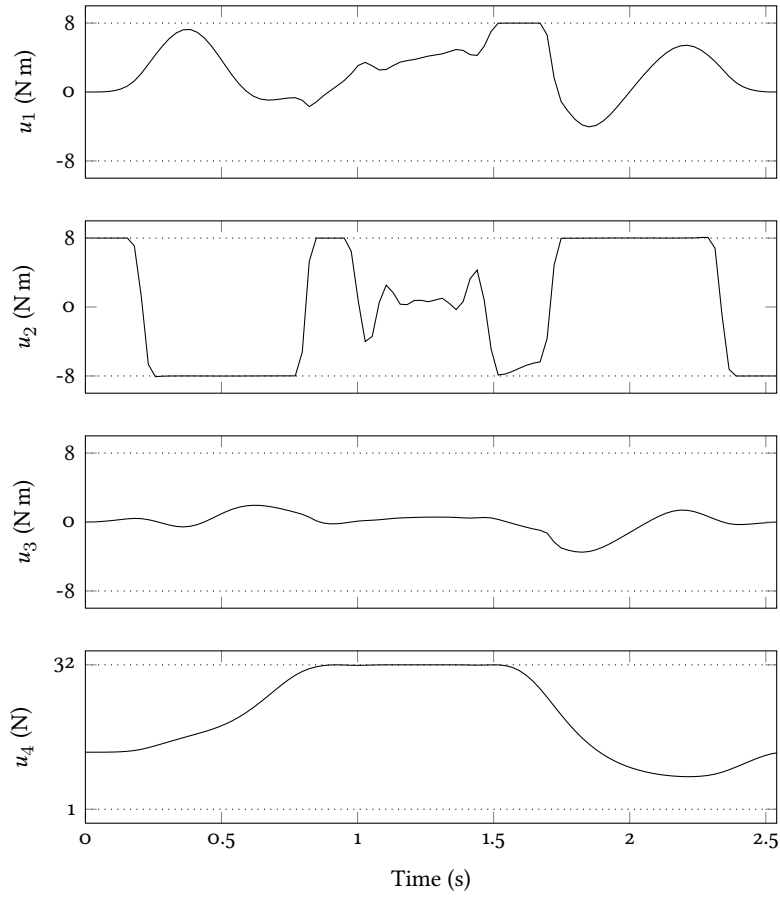

FIGURE 5.5.: Time-optimal control inputs for a quadrotor following (5.11)

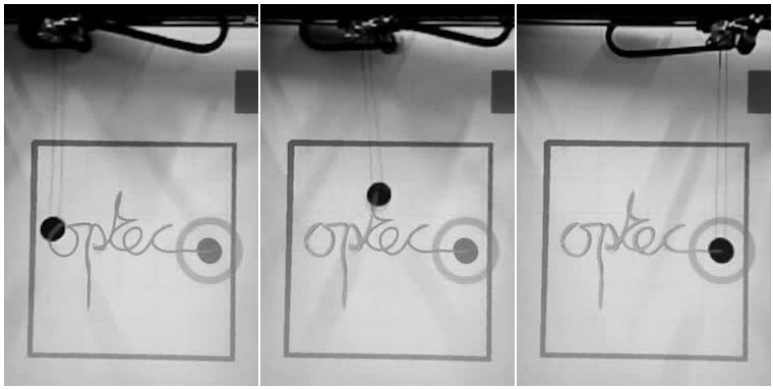

FIGURE 5.6.: The overhead crane is writing "optec" time-optimally

as opposed to Lai, Yang, and Wu (2006) where for quadrotors a time-consuming genetic algorithm is used to generate an initial guess. The proposed problem formulation (5.7) also allows for other objectives than time, such as energy consumption, and arbitrary constraints, such as torque rate constraints in robotic applications, which cannot be included in the framework of Verscheure, Demeulenaere, et al. (2009).

To illustrate the efficiency and effectiveness of the proposed approach, a path following demonstrator on an overhead crane is developed in van Bergen (2013). A user is able to draw an arbitrary reference path for the load. Subsequently, the developed software computes the time-optimal inputs required to follow the load. In a final step, the inputs are applied to the setup in feedforward. Figure 5.6 illustrates the overhead crane writing "optec".

5.4. Optimal path planning

In the decoupled motion planning approach (Bobrow, Dubowsky, and Gibson 1985), a high level planner first determines a geometric path accounting only for geometric constraints. Subsequently, in the path following stage, which was the focus of the previous sections, an optimal velocity profile along this path is determined. Inspired by this decoupled

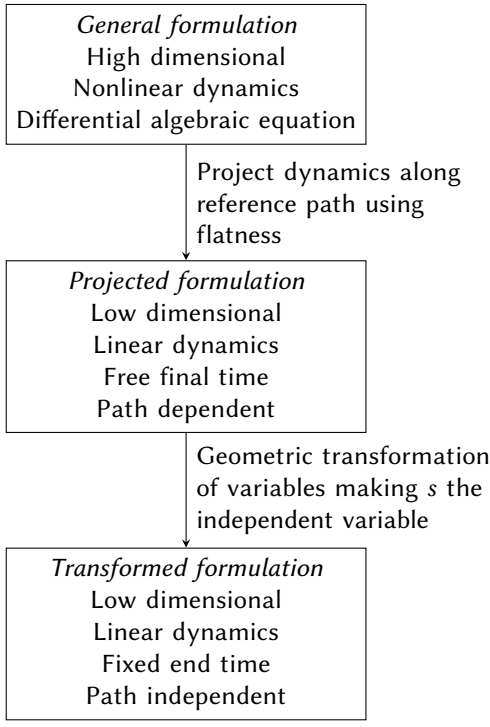

FIGURE 5.7.: Different steps towards the proposed geometric problem formulation

approach, we now tackle the optimal path planning problem. Instead of fixing the geometric path as before, it is now represented as a convex combination of two or more fixed boundary paths.

We first reformulate the problem formulation from Section 5.1 to accommodate for the unknown path. We want to find an input signal $u(t)$ and the a geometric path $y_d(s)$ that allows for the fastest execution time such that (i) the geometric path is followed exactly by the system output:

$$z(t) = y_d(s(t)),$$

(ii) without violating constraints on states and inputs

$$u(t) \in \mathcal{U},\ x(t) \in \mathcal{X},$$

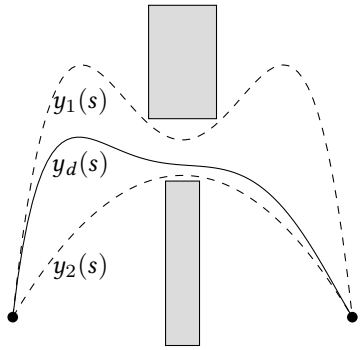

FIGURE 5.8.: Geometric reference $y_d(s)$ as a convex combination of two paths $y_1(s)$ and $y_2(s)$ in a constrained environment

(iii) without violating geometric constraints

$$y_d(s) \in \mathcal{Y},$$

and (iv) in minimal time T.

Due to the possible complex geometry of the environment, imposing $y_d(s) \in \mathcal{Y}$ directly in the optimization problem can pose difficulties when solving the problem. To circumvent this difficulty, the geometric reference path $y_d(s)$ is defined as a convex combination of l feasible paths $y_i(s) \in \mathcal{Y}, i = 1, \ldots, l$:

$$y_d(s) = \sum_{i=1}^{l} y_i(s) p_i(s),$$

with

$$\sum_{i=1}^{l} p_i(s) = 1 \text{ and } p_i(s) \geq 0, \forall s \in [0, 1],$$

and $p_i(s), y_i(s) \in C^r$. Figure 5.8 shows two functions $y_1(s)$ and $y_2(s)$, which stay clear of the geometric constraints in the hatched areas. The geometric reference $y_d(s)$ is defined as the convex combination of both functions. In this example $p_1(s) = 1 - \sqrt{s}$ and $p_2(s) = \sqrt{s}$.

Similarly as in previous sections, the system dynamics can be projected

along the path onto a linear single-input system:

$$\begin{aligned}
\dot{y}_d(s) &= \sum_{i=1}^{l} \left(\partial_s y_i(s) p_i(s) + \partial_s p_i(s) y_i(s)\right) \sqrt{b(s)}, \\
\ddot{y}_d(s) &= \sum_{i=1}^{l} \left(\partial_s y_i(s) p_i(s) + \partial_s p_i(s) y_i(s)\right) \partial_s b(s)/2 \\
&+ \sum_{i=1}^{l} \left(\partial_s^2 y_i(s) p_i(s) + 2\partial_s p_i(s) \partial_s y_i(s) + \partial_s^2 p_i(s) y_i(s)\right) b(s),
\end{aligned} \tag{5.12}$$

and so on, such that we can rewrite the states and inputs to the system as

$$\begin{aligned}
x &= \Phi_p(s, p_i, \partial_s p_i, \ldots, \partial_s^{r-1} p_i, b, \ldots, \partial_s^{r-2} b) = \Phi_p(s, \rho_i, \beta), \\
u &= \Psi_p(s, p_i, \partial_s p_i, \ldots, \partial_s^{r} p_i, b, \ldots, \partial_s^{r-1} b) = \Psi_p(s, \rho_i, \varrho_i, \beta, w),
\end{aligned}$$

with $\rho_i = (p_i, \partial_s p_i, \ldots, \partial_s^{r-1} p_i)^\mathsf{T}$, $\varrho_i = \partial_s^r p_i$, $i = 1, \ldots, l$ and w, β as defined in (5.5).

The path following problem (5.7) is then extended with the control ϱ_i and states ρ_i:

$$\begin{aligned}
\underset{\beta(\cdot), \rho_i(\cdot), w(\cdot), \varrho_i(\cdot)}{\text{minimize}} \quad & \int_0^1 \frac{1}{\sqrt{b(s)}} ds \\
\text{subject to} \quad & \partial_s \beta(s) = \begin{pmatrix} 0 & I_{k-2} \\ 0 & 0 \end{pmatrix} \beta(s) + (0, \ldots, 0, 1)^\mathsf{T} w(s) \\
& \partial_s \rho_i(s) = \begin{pmatrix} 0 & I_{k-1} \\ 0 & 0 \end{pmatrix} \rho_i(s) + (0, \ldots, 0, 1)^\mathsf{T} \varrho_i(s), \\
& i = 1, \ldots, l \\
& \sum_{i=1}^{l} (\rho_i)_0(s) = 1, \forall s \in [0, 1] \\
& \beta_0(s) \geq 0, \forall s \in [0, 1] \\
& (\rho_i)_0(s) \geq 0, \forall s \in [0, 1], i = 1, \ldots, l \\
& \Phi_p(s, \rho_i, \beta) \in \mathcal{X}, \forall s \in [0, 1] \\
& \Psi_p(s, \rho_i, \varrho_i, \beta, w) \in \mathcal{U}, \forall s \in [0, 1].
\end{aligned} \tag{5.13}$$

x_T [m]	T [s]	T^* [s]
3	0.8902	0.8977
6	1.223	1.231
9	1.478	1.488
12	1.694	1.705
15	1.885	1.895

TABLE 5.2.: Optimal times for horizontal displacements

Again, note that we have a fixed end-time problem where only a trivial integration has to be performed. However, due to the path being free, Φ_p, Ψ_p are vastly more nonlinear compared to the path following problem (5.7). However, the example below still shows good convergence, albeit more slowly compared to path following due to the increased nonlinearity.

To benchmark against an analytical method from Hehn, Ritz, and D'Andrea (2012), consider the simplified two-dimensional quadrotor model from Section 1.2. The thrust input is constrained to $1\,\mathrm{m\,s^{-2}} \leq F_T/m \leq 20\,\mathrm{m\,s^{-2}}$ and the rotational rate to $-10\,\mathrm{rad\,s^{-1}} \leq \omega \leq 10\,\mathrm{rad\,s^{-1}}$. In all examples, we start and stop in steady state, which is accomplished by a suitable reparameterization of the outer paths as proposed in Section 5.2.1.

First, we consider purely horizontal displacements from $(0, 0)$ to $(x_T, 0)$. The outer paths are defined as

$$y_1 = (x_T s, -1)^\intercal \text{ and } y_2 = (x_T s, 1)^\intercal,$$

The constraints $p_i(0) = p_i(1) = 0.5$ for $i = 1, 2$ are added to the optimization problem such that the quadrotor starts and stops at at a height $z = 0$.

Optimal times T are calculated for five distances $x_T = 3, 6, 9, 12, 15\,\mathrm{m}$ and compared to the travel times T^* reported in Hehn, Ritz, and D'Andrea (2012) in table 5.2. The calculated times consistently outperform the results of Hehn, Ritz, and D'Andrea (2012) by a small amount. This is probably due to the explicit parametrization of the controls in Hehn, Ritz, and D'Andrea (2012), which are either at their maximum or minimum value or zero. The solver time for these trajectories is around 0.6 s for problem sizes around 1000 variables.

Assuming the quadrotor is taking off from the ground, these maneuvers

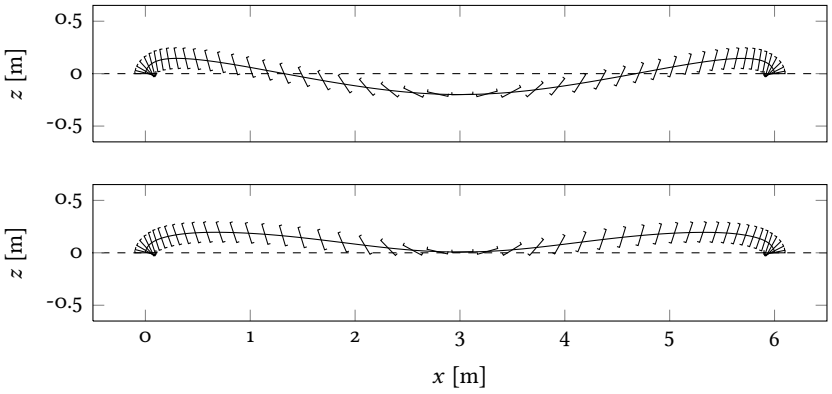

FIGURE 5.9.: Optimal horizontal maneuver with (bottom) and without (top) ground constraint. The quadrotors are drawn in equal time steps.

would require the quadrotor to go underground as z becomes negative as is clear from Figure 5.9 (top). Our method can easily cope with this constraint by defining the outer paths as

$$y_1 = (x_T s, 0)^\mathsf{T} \text{ and } y_2 = (x_T s, 1)^\mathsf{T},$$

ensuring a positive value for z. In Hehn, Ritz, and D'Andrea (2012), these geometric constraints cannot be taken into account. For $x_T = 6\,\text{m}$ both trajectories are shown in Fig. 5.9. The trajectory time for the positive trajectory is 1.227 s, which is only marginally more compared to the unconstrained case.

Now, consider general two-dimensional displacements from $(0, 0)$ to (x_T, z_T). For geometrically unconstrained problems we define the outer paths as

$$y_1 = (x_T s, -c)^\mathsf{T} \text{ and } y_2 = (x_T s, z_T + c)^\mathsf{T},$$

where c is a chosen constant and constraints are added on $p_i(s)$ for $s = 0, 1$ such that the trajectory start at $(0, 0)$ and stops at (x_T, z_T).

As a benchmark, an optimal trajectory is calculated for $x_T = z_T = 5\,\text{m}$. The constant c is chosen $c = 0\,\text{m}$. The optimal trajectory is shown in Fig. 5.10. Our computations lead to a final time of 1.289 s, which is shorter than the 1.4 s reported in Hehn, Ritz, and D'Andrea (2012).

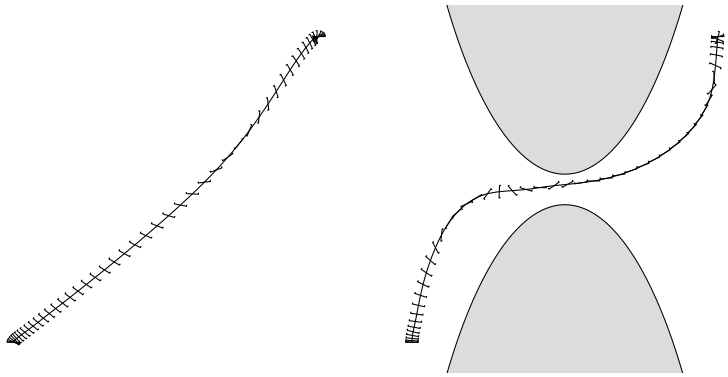

FIGURE 5.10.: Optimal general and constrained maneuver to (5 m, 5 m). The quadrotors are drawn in equal time steps.

Moreover, in the proposed framework geometric constraints can be easily added to the optimization problem. Consider flying the quadrotor to $x_T = z_T = 5\,\mathrm{m}$ through a hoop with diameter 0.5 m, positioned vertically with its center at (2.5 m, 2.5 m). To ensure the quadrotor flies through the hoop, we define the outer paths as

$$y_1 = \left(x_T s, (x_T s - 2.5)^2 + 2.75\right)^\mathsf{T}$$

and

$$y_2 = \left(x_T s, -(x_T s - 2.5)^2 - 2.25\right)^\mathsf{T},$$

The optimal trajectory is shown in Figure 5.10 along with the outer paths. The corresponding controls are shown in Figure 5.11. Note the aggressive controls causing the quadrotor to flip and use its thrust for breaking.

These examples show the versatility of the approach. Geometric constraints can easily be added by a suitable choice of the outer paths, without complicating the optimization problem. Moreover, solution are calculated efficiently while relying on automatic initialization of the solver and benchmark results for the quadrotor show improvements over an analytical method presented in Hehn, Ritz, and D'Andrea (2012).

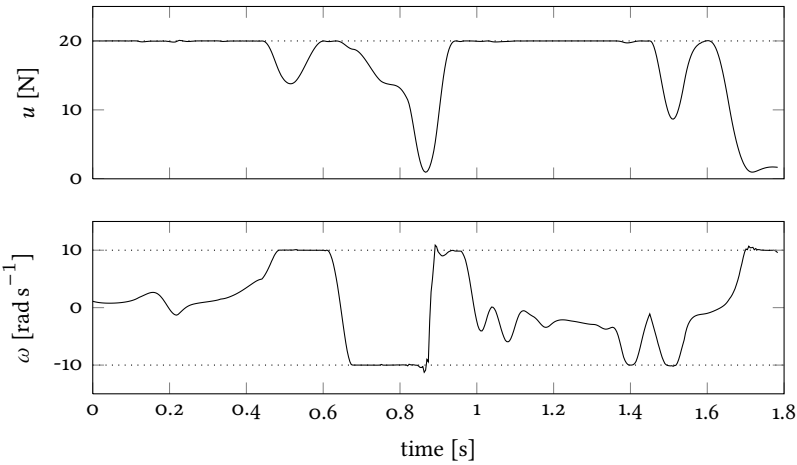

FIGURE 5.11.: Optimal controls for a constrained maneuver to (5 m, 5 m).

5.5. Summary

This chapter deals with path following problems for differentially flat systems. For such systems, an efficient problem formulation is found in two steps. First, due to differential flatness of the system, the dynamics can be projected along the path onto a linear single-input system, resulting in a small dimensional optimal control problem. Despite its small dimension, the optimization problem remains difficult to solve and often requires a good initialization, especially for free end-time problems. Moreover, the optimization variables enter through the definition of the path into the optimization problem, making the convergence dependent on the geometric reference path. To overcome these issues, the second step consists of a nonlinear change of variables that expresses the problem in terms of the progress along the path. This way, the problem is rid from its time dependence, making the convergence independent of the reference path. Due to the transformation, singularities may appear in the problem formulation. These can easily be overcome by a reparameterization of the path. Furthermore, we prove that if each point of the geometric path can be visited in steady state, the path can be followed exactly. Two numerical

examples illustrate the efficiency and practicality of the proposed method.

By also considering the geometric path as an optimization variable, the same idea is subsequently used in optimal path planning. To avoid hard geometric constraints such as collision avoidance, the geometric path is represented as an unknown convex combination of two or more geometrically feasible reference paths. By means of a numerical example the versatility of the proposed approach is illustrated.

What to remember

- Writing the path following problem in terms of the evolution along the path eliminates the time from consideration and results in an efficient problem formulation.
- By also considering the geometric path as a variable, the path following problem becomes a path planning problem.
- Representing the geometric path as a convex combination of outer paths avoids hard geometric constraints.

6

Concluding remarks

Over the course of five chapters, this thesis detailed contributions to nonnegative polynomial splines, with application in data approximation and optimal control in linear systems, and optimal path following problems. A central theme for optimal motion synthesis was the use of differential flatness to simplify the optimization problems. In this final chapter we summarize our contributions and provide suggestions for future research.

6.1. Summary

Chapter 2 detailed our main contributions with respect to polynomial spline optimization. Based on results regarding nonnegative polynomials, we derived an exact condition for nonnegativity of piecewise polynomials. We conjectured that the resulting LMI's exhibit a block diagonal structure with overlapping cliqes. Furthermore, based on the positivity property of the B-spline basis, a novel linear relaxation method using knot insertion was presented. Although this strategy introduces conservatism, it is computationally much cheaper.

Chapter 3 focused on the spline approximation problem. Here, the main challenge lies on finding the appropriate knot locations. Based on recent results from literature we tackled the issue indirectly by supplying many candidate knots and using a regularization to favor solution with only a few active knots. In this way, the optimization problem is convex, allowing for global solutions to be calculated efficiently. By using a reweighting algorithm, the sparsity is enhanced even further. Two benchmarks illustrated the performance compared to the established literature. Furthermore, prior knowledge on the function could easily be incorporated by adding (convex) constraints to the optimization problem.

Another application of spline based optimization was detailed in Chapter 4. Here, we looked at optimal control problems for linear systems

through differential flatness. Inspired by recent results, the optimal control problem is viewed as nonnegativity problem involving piecewise polynomials. Using the theory developed in Chapter 2, the problem is cast both as an exact semidefinite and a conservative linear problem. Benchmark results indicate improvements with respect to both conservatism and computational effort compared to previously published results. Instead of using binary search for time-optimal problems, derivative information is incorporated using a Newton-Raphson method by computing the sensitivity of the maximum travel range with respect to the available motion time. In this way, computational performance is vastly improved compared to binary search. By taking the method even further, a parametric description of the maximum range curve and corresponding solutions is obtained by solving only a few optimization problems. Finally, for uncertain systems a robust input design problem is set up. Using differential flatness, the robustness constraints are expressed as a rational function of the uncertainty parameter, allowing us to derive a robust counterpart for the optimization problem. This way sampling of the uncertainty region is avoided. This method, however, still suffers from computational difficulties due to the large LMI's resulting from the problem formulation.

In Chapter 5 we turned our attention to the so-called optimal path following problem, for which a geometric trajectory is to be followed without any preassigned timing information. A projection of the dynamics along the geometric path onto a linear single-input system simplifies the optimization problem to great extent. The problem, however, remains difficult to solve particularly for the case of time-optimal problems and highly nonlinear reference paths. By applying a transformation of variables, we arrived at a fixed end-time optimal control problem that can be solved efficiently and shows little dependence on the given reference path. Two examples illustrated the efficiency of the proposed approach. In a subsequent step, we extended this approach towards path planning problems. We represented the geometric reference as an unknown convex combination of two or more fixed boundary paths. In this way, we were able to determine optimal paths as well, without requiring an accurate initial guess. To easily model path following and planning problems for a variety of systems, a software tool was developed. It is covered in more detail in Appendix A.

6.2. Ideas for future research

In this thesis some theoretical issues remained unsolved and deserve further attention. Furthermore, per chapter we propose new ideas, which may provide a basis for future research.

6.2.1. Nonnegative univariate polynomial splines

To strengthen the theoretical results in Chapter 1, a proof for Conjecture 2.1 is necessary. As shown for the degree two case, the LMI conditions for positivity can always be decoupled. We believe that for higher degrees, a similar decoupling can be found. However, already for splines of degree four, we are unable to prove that such a decoupling is possible.

With respect to the linear relaxations as discussed in Section 2.3.2, it is interesting to extend the results for other totally positive bases such as trigonometric *B*-splines (Lyche and Winther 1979). This would provide us with easy interpretable linear sufficient conditions for nonnegativity for a variety of functions. Another valuable extension are multivariate functions, providing an alternative to Pólya's relaxation and at the same time broadening the scope of possible applications.

To efficiently solve the LMI's resulting from nonnegativity constraints, dedicated solvers that exploit the structure of the problem are necessary. The solver SMCP (Andersen, Dahl, and Vandenberghe 2010) shows promising results for semidefinite programs with overlapping cliqes and with band structure. The solver should be tested on our nonnegativity problems.

6.2.2. Optimal control of linear systems through differential flatness

It would be interesting to research the effect of other basis functions. In literature, very often the basis is described by B-splines. Solutions for other piecewise bases, such as a trigonometric basis, may have desirable properties and should be researched.

We would also like to extend our results towards linear parameter varying systems. For such systems, the state space matrices depend linearly on a varying parameter. Given spline parameterizations for the

varying parameter and the flat output, we can still express states and inputs as polynomial splines, which we can constrain efficiently using the results from section 2.3.2. Although the optimization problem will now be nonconvex, its dimensions remain small and we expect to solve it efficiently.

For the computation of time-optimal trajectories we used first order information to efficiently find the time optimal solution. By incorporating higher order information, we could possibly speed up the calculations. However, it should be noted that for higher order derivatives a matrix inversion is required, which could possibly slow down computations for larger problems.

Finally, with respect to computing robust inputs, instead of the linear conditions (4.16), the exact condition (4.14) should also be considered. Here, the difficulty lies in finding a linear fractional representation of this constraint. Another difficulty related to the method are the computational issues related to solving the LMI's. Linear relaxations can be explored to counter these difficulties at the cost of introducing conservatism. The proposed method can also be extended towards higher order uncertain systems by viewing the system as a sum of first and second order systems. Then, by constraining the residual vibration for each subsystem, the total overshoot can be limited.

6.2.3. Optimal path following for differentially flat systems

In each application of the path following framework described in this chapter, model-plant mismatch is inevitable. As a result the geometric path is not followed exactly and the optimized motion trajectory may be suboptimal or even infeasible for the true plant. Assuming the system executes the same task repeatedly, the performance can be improved using iterative learning control: the system learns from its previous trials to improve upon the current trial. Conventional learning controllers, require the reference to be specified in time and are not able to update the timing of the motion along a geometric path. Recently, a new iterative learning control approach for path following problems was proposed in Janssens et al. (2013c) for the specific case of an xy positioning stage. Each iteration, a nonparametric model correction is computed, based on the

model-plant mismatch observed in the previous iteration. Subsequently, the optimal path following problem is solved for the updated model. A generalization of this approach for differentially flat systems is currently being researched. Possibly, it could also be applied on the path planning problem. In this case, the path can also be iteratively computed based on the updated model.

Another extension aims at developing code that can be run online. In such a way, when the geometric path is replanned while already being executed, e.g. to avoid collision with a moving object, we can continue to follow these paths optimally. For robotic manipulators, such an approach has been presented in Verscheure, Diehl, et al. 2009. Here, approximate solutions are determined through a log-barrier batch solution method. A recursive variant allows the computation of optimal controls for the already available path data.

Path following also has many applications in the machining industry such as laser cutting and milling. To specify the paths, G-code is most widely used. Typically, this code specifies straight lines and circular arcs along which the machine has to move with a specified speed. A valuable extension consists of interpreting this G-code in our path following framework to determine the optimal inputs given the dynamic constraints of the machine. It can be taken even further by also accounting for the required machining tolerances for which the proposed path planning framework will provide a solution.

With respect to path the path planning problem, in another approach we consider to still fix the geometric path for the flat outputs. But instead of optimizing over one common path coordinate, each coordinate is given its own parameter. This way, optimal point-to-point motions can, for instance, be determined by linear geometric trajectories for each flat output.

Finally, experiments provide the ultimate validation of any contribution. Therefore, all simulation performed in this thesis have to prove their usefulness in practice. Initial experimental results for the proposed path following methods on an overhead crane are promising.

A

Software tutorial

In Chapter 5 an interesting formulation was developed for optimal path following problems for differentially flat systems. As it requires tedious manipulations to arrive at this formulation, a software package was developed to aid the user in defining his own path following or path planning problems. The goal of this chapter is to give a short overview of the software's functionality and illustrate it with two examples.

A.1. Installation

The software is developed in the Python programming language (*Python programming language*) and uses CasADi (Andersson, Åkesson, and Diehl 2012) as a framework for symbolic computations. Furthermore CasADi provides an interface to Ipopt (Wächter and Biegler 2006), a software package for large-scale nonlinear optimization. For installation instructions regarding these software packages, the user is referred to the CasADi homepage https://github.com/casadi/casadi/wiki. CasADi now offers binaries, which simplify the installation procedure considerably.

Our software is available via http://www.kuleuven.be/optec/software. After downloading the software it can easily be installed by executing

```
> sudo python setup.py install
```

A.2. Software design

The code exports three classes. The class `FlatSystem` supports the definition of differentially flat systems. For a given differentially flat system the class `PathFollowing` models path following problems and the class `PathPlanning` models path planning problems. Both classes provide functionality for defining the path and (path dependent) constraints, solving

the problem and plotting the solution. Each class' functionality is described below.

A.3. Flat systems

The class `FlatSystem` defines a flat system. A flat system S with m flat outputs using derivatives up to order k is defined as

```
S = FlatSystem(m, k)
```

Its symbolic flat output variables are stored in the instance attribute `y`. For example, the j-th derivative of flat output i of system S is `S.y[i, j]`. The instance method `set_state` allows to express the state x of the system S as a function of the flat outputs:

```
x = S.set_state(expr, name=None)
```

Optionally, the state can be given a name. The computation of time derivatives is facilitated with the instance method `dt`:

```
xdot = S.dt(x)
```

As an example, we model the overhead crane, described in Section 1.2.

```
from casadi import *
from pathfollowing import *

S = FlatSystem(2, 4)
G = 9.81
y1, dy1, ddy1 = S.y[0, 0], S.y[0, 1], S.y[0, 2]
y2, dy2, ddy2 = S.y[1, 0], S.y[1, 1], S.y[1, 2]
u1 = S.set_state(y1 + y2 * ddy1 / (G - ddy2), 'u1')
u2 = S.set_state(y2 * sqrt(1 + (ddy1 / (G - ddy2)) ** 2), 'u2')
theta = S.set_state(arctan(-ddy1 / (G - ddy2)), 'theta')
du1 = S.set_state(S.dt(u1), 'du1')
du2 = S.set_state(S.dt(u2), 'du2')
```

A.4. Path following

For the flat system S an instance of a path following problem is created via

```
P = PathFollowing(S)
```

The symbolic path coordinate and its time derivatives are stored in the instance attribute `s`. The reference path is set by the instance method

```
set_path(expr[, r2r=False])
```

where `expr` is a list in which each component of the flat output is expressed as a function of `s[0]`. When `r2r` is `True`, the geometric path is reparameterized as in 5.2.1 such that a rest-to-rest motion is imposed.

Inequality constraints are set using the instance method

```
set_constraint(expr, lb, ub)
```

where `lb` and `ub` are the lower and upper bounds. Furthermore, various solver options are set through

```
set_options({'option': value})
```

Aside from all supported options of IPOPT in CasADi, the following options are available

- `'N'`: The number of discretization steps
- `'Nt'`: The number of returned time points in the solution
- `'reg'`: A regularization factor added to the goal function to avoid singular arcs in the solution

Finally, the instance method

```
solve()
```

solves the problem. The solution is stored in the instance attribute `sol`. The instance method

```
plot()
```

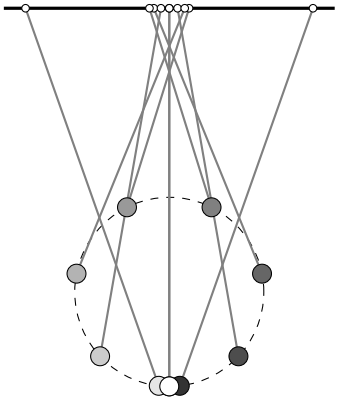

FIGURE A.1.: Time optimal path following of a circle for the overhead crane in 10 equal time steps

plots all states as defined in the flat system and inequality constraints.

Continuing the example from previous section, we define a path following problem for the overhead crane tracking a circular trajectory with its load. The velocities of the trolley and hoisting mechanisms are constrained to $[-5, 5]\,\mathrm{m\,s^{-1}}$ and $[-2.5, 2.5]\,\mathrm{m\,s^{-1}}$ respectively.

```
P = PathFollowing(S)
path = [
    0.25 * sin(2 * pi * P.s[0]),
    0.25 * cos(2 * pi * P.s[0]) + 0.5
]
P.set_path(path)
P.set_constraint(du1, -5, 5)
P.set_constraint(du2, -2.5, 2.5)
P.set_options({'reg': 1e-10})
P.solve()
P.plot()
```

Figure A.2 shows the resulting velocities of the trolley and hoisting mechanisms as a function of the path coordinate. At each time instant at least one of the constraints is active, indicating a time-optimal trajectory. Figure A.1 illustrates the movement of the crane required to follow the circular trajectory in 10 equal time steps.

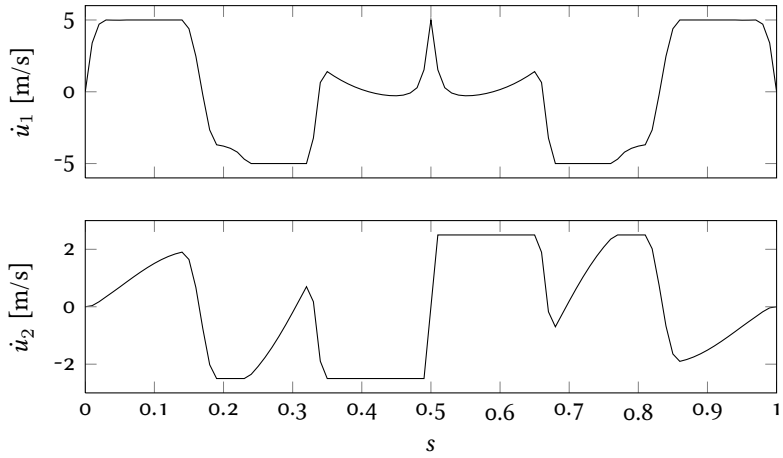

FIGURE A.2.: Time-optimal velocity signals as a function of the path coordinate for an overhead crane following a circular trajectory

A.5. Path planning

A subclass of `PathFollowing` allows to define path planning problems. An instance of a path planning problem for a differentially flat system S is created via

```
Q = PathPlanning(S)
```

Similarly, the symbolic path coordinate and its derivatives are stored in the instance attribute `s`.

The outer paths are set with the instance method

```
set_path(expr[, r2r=False])
```

where `expr` is a list. Each element parameterizes one of the outer paths by another list, in which each element contains a component of the flat output as a function of `s[0]`.

As before, constraints are set with the instance method

```
set_contstraint(expr, lb, ub)
```

and solver options are set with the instance method

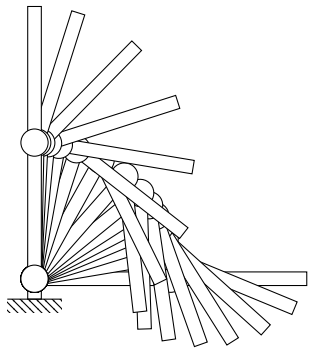

FIGURE A.3.: Time optimal movement of a planar manipulator in 15 equal time steps

```
set_options({'option': value})
```

An added option for path planning problems is `'Nc'`, which controls the number of spline coefficients used for the convex combination functions $p_i(s)$ (cfr. Section 5.4).

Finally, calling the instance method

```
solve()
```

solves the path planning problem.

Let's look at a path planning example for the two degree of freedom robotic manipulator from Section 5.3.1. Similar to previous example, we first define the robot as a flat system, for which the code can be found in the online examples. Now, we want to move the robot time optimally from an initial configuration $(0, 0)^T$ to $(\pi/2, 0)^T$. To this end, we define the outer paths, before reparameterization, as

$$y_1 = \left(\frac{\pi}{2}s, -32(s-0.5)^2 + 8\right)^T$$
$$y_2 = \left(\frac{\pi}{2}s, 32(s-0.5)^2 - 8\right)^T.$$

Note that, for simplicity, we only allow freedom in the movement of the second joint.

Suppose we have modeled the robot as the flat system `S`. Then the path planning problem can be modeled as follows:

```
P = PathPlanning(S)
s = P.s[0]
p = s ** 2 * (s + 3 * (1 - s))
y1 = [np.pi / 2 * p, -32 * (p - 0.5) ** 2 + 8]
y2 = [np.pi / 2 * p,  32 * (p - 0.5) ** 2 - 8]
P.set_path([y1, y2])
P.set_constraint('tau1', -20, 20)
P.set_constraint('tau2', -10, 10)
P.set_options({'N': 99, 'Nc': 10})
P.solve()
```

Figure A.3 shows the movement of the robot in 15 equal time steps. As expected, in order to move as fast as possible, the inertia for the first joint is lowered by drawing in the second joint.

Bibliography

Amos, D. E. and M. L. Slater (1969). "Polynomial and spline approximation by quadratic programming". In: *Commun. ACM* 12.7, pp. 379–380.

Andersen, M. S., J. Dahl, and L. Vandenberghe (2013). *CVXOPT: A Python package for convex optimization*. Version 1.1.6. URL: cvxopt.org.

Andersen, M. S., J. Dahl, and L. Vandenberghe (2010). "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones". In: *Mathematical Programming Computation* 2.3-4, pp. 167–201.

Andersson, J., J. Åkesson, and M. Diehl (2012). "CasADi: a symbolic package for automatic differentiation and optimal control". English. In: *Recent Advances in Algorithmic Differentiation*. Vol. 87, pp. 297–307.

Ben-Tal, A. and A. Nemirovskiĭ (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Society for Industrial and Applied Mathematics.

Bernstein, S. (1915). "Sur la représentation des polynômes positif". In: *Soobshch Har'k Mat Obshch* 14, pp. 227–228.

Betts, J. (2001). *Practical Methods for Optimal Control Using Nonlinear Programming*. Society for Industrial and Applied Mathematics.

Bobrow, J., S. Dubowsky, and J. Gibson (1985). "Time-optimal control of robotic manipulators along specified paths". In: *The International Journal of Robotics Research* 4.3, pp. 3–17.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge University Press. 727 pp.

Burden, R. L. and J. D. Faires (2010). *Numerical analysis*. Brooks/Cole.

Candès, E. J., M. B. Wakin, and S. Boyd (2008). "Enhancing sparsity by reweighted ℓ_1 minimization". In: *Journal of Fourier Analysis and Applications* 14.5, pp. 877–905.

Chew, M. and C. H. Chuang (1995). "Minimizing residual vibrations in high-speed cam-follower systems over a range of speeds". In: *Journal of Mechanical Design* 117.1, pp. 166–172.

Choi, M.-D., T.-Y. Lam, and B. Reznick (1995). "Sums of squares of real polynomials". In: *Proceedings of Symposia in Pure Mathematics*. Vol. 58. 2, pp. 103–126.

Consolini, L. and A. Piazzi (2009). "Generalized bang-bang control for feedforward constrained regulation". In: *Automatica* 45.10, pp. 2234–2243.

de Boor, C. (2001). *A practical guide to splines*. revised. Springer-Verlag. 346 pp.

de Boor, C. and J. W. Daniel (1974). "Splines with nonnegative b-spline coefficients". In: *Mathematics of Computation* 28.126, pp. 565–568.

de Boor, C. and J. R. Rice (1968). *Least squares cubic spline approximation ii — variable knots*. Tech. rep. CSD TR 21. Purdue University.

De Caigny, J. (2009). "Contributions to the Modeling and Control of Linear Parameter-Varying Systems (Bijdragen tot de modellering en controle van lineaire parameter-variërende systemen)". thesis. KU Leuven.

De Caigny, J., B. Demeulenaere, J. Swevers, and J. De Schutter (2008). "Design of dynamically optimal spline motion inputs: Experimental results". In: *Proceeding of the 2008 American Control Conference*. (Seattle, WA, USA), pp. 3269–3274.

Debrouwere, F., W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers (2013). "Time-optimal path following for robots with convex-concave constraints using sequential convex programming". In: *IEEE Transactions on Robotics*. Accepted.

Demeulenaere, B., J. De Caigny, G. Pipeleers, J. De Schutter, and J. Swevers (2009). "Optimal splines for rigid motion systems: benchmarking and extensions". In: *Journal of Mechanical Design* 131.10, 101005, 13 pages.

Demeulenaere, B., G. Pipeleers, J. De Caigny, J. Swevers, Joris De Schutter, and L. Vandenberghe (2009). "Optimal splines for rigid motion systems: a convex programming framework". In: *Journal of Mechanical Design* 131.10, 101004, 11 pages.

Dierckx, P. (1975). "An algorithm for smoothing, differentiation and integration of experimental data using spline functions". In: *Journal of Computational and Applied Mathematics* 1.3, pp. 165–184.

— (1980). "An algorithm for cubic spline fitting with convexity constraints". In: *Computing* 24, pp. 349–371.

— (1993). *Curve and surface fitting with splines*. Oxford University Press, Inc. 285 pp.

El Ghaoui, L., F. Oustry, and H. Lebret (1998). "Robust solutions to uncertain semidefinite programs". In: *SIAM Journal on Optimization* 9.1, pp. 33–52.

El Ghaoui, L. and H. Lebret (1997). "Robust solutions to least-squares problems with uncertain data". In: *SIAM Journal on Matrix Analysis and Applications* 18.4, pp. 1035–1064.

Faiz, N., S. K. Agrawal, and R. M. Murray (2001). "Trajectory planning of differentially flat systems with dynamics and inequalities". In: *Journal of Guidance, Control and Dynamics* 24.2, pp. 219–227.

Faulwasser, T., V. Hagenmeyer, and R. Findeisen (2011). "Optimal Exact Path-Following for Constrained Differentially Flat Systems". In: *Proceedings of 18th IFAC World Congress*. (Milano, Italy), pp. 9875–9880.

Fliess, M. and R. Marquez (2000). "Continuous-time linear predictive control and flatness: a module-theoretic setting with examples". In: *International Journal of Control* 73.7, pp. 606–623.

Fliess, M., J. Lévine, P. Martin, and P. Rouchon (1995). "Flatness and defect of nonlinear systems: Introductory theory and examples". In: *International Journal of Control* 61, pp. 1327–1361.

Freund, R. M. (1985). "Postoptimal analysis of a linear program under simultaneous changes in matrix coefficients". In: *Mathematical Programming Essays in Honor of George B. Dantzig Part I*. Vol. 24, pp. 1–13.

GLPK. *Gnu linear programming kit*. URL: www.gnu.org/software/glpk.

Hauser, J. and A. Saccon (2006). "A Barrier Function Method for the Optimization of Trajectory Functionals with Constraints". In: *Proceedings of the IEEE Conference on Decision and Control*. (San Diego, CA, USA), pp. 864–869.

Hayes, J. (1974). "Algorithms for curve and surface fitting". In: *Software for numerical mathematics*, pp. 219–233.

Hedge, S. and S. Kacera (2006). *Safeguarded zero-finding methods*. report. New York University, Courant Institute of Mathematical Sciences, Computer Science Department.

Hehn, M., R. Ritz, and R. D'Andrea (2012). "Performance benchmarking of quadrotor systems using time-optimal control". English. In: *Autonomous Robots* 33 (1-2), pp. 69–88.

Henrion, D. and J. B. Lasserre (2006). "LMIs for constrained polynomial interpolation with application in trajectory planning". In: *Systems & Control Letters* 55.6, pp. 473–477.

Hoffmann, G. M., S. L. Wasl, and C. J. Tomlin (2008). "Quadrotor helicopter trajectory tracking control". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. (Honolulu, Hawaii).

Janssens, P., W. Van Loock, G. Pipeleers, and J. Swevers (2013a). "An efficient algorithm for solving time-optimal point-to-point motion control problems". In: *Proceedings of the 2013 IEEE International Conference on Mechatronics*. (Vicenza, Italy), pp. 682–687.

— (2013b). "Efficient computation of time-optimal point-to-point motion trajectories". In: *IEEE Transactions on Control Systems Technology*. In review.

— (2013c). "Iterative learning control for optimal path following problems". In: *Proceedings of the 52nd IEEE Conference on Decision and Control*. (Firenze, Italy). Accepted.

Jupp, D. L. B. (1978). "Approximation to data by splines with free knots". In: *SIAM Journal on Numerical Analysis* 15.2, pp. 328–343.

Lai, L., C. Yang, and C. Wu (2006). "Time-optimal control of a hovering quad-rotor helicopter". In: *Journal of Intellegent and Robotic Systems* 45.2, pp. 115–135.

Lasserre, J. B. (2010). *Moments, positive polynomials and their applications*. Imperial College Press. 385 pp.

Laurent, M. (2009). "Sums of squares, moment matrices and optimization over polynomials". en. In: *Emerging Applications of Algebraic Geometry*. Vol. 149, pp. 157–270.

Lévine, J. and D. Nguyen (2003). "Flat output characterization for linear systems using polynomial matrices". In: *Systems & Control Letters* 48.1, pp. 69–75.

Louembet, C., F. Cazaurang, A. Zolghadri, C. Charbonnel, and C. Pittet (2009). "Path planning for satellite slew manoeuvres: a combined flatness and collocation-based approach". In: *Control Theory Applications, IET* 3.4, pp. 481–491.

Louembet, C., F. Cazaurang, and A. Zolghadri (2010). "Motion planning for flat systems using positive *b*-splines: An LMI approach". In: *Automatica* 46.8, pp. 1305–1309.

Lyche, T. and R. Winther (1979). "A stable recurrence relation for trigonometric B-splines". In: *Journal of Approximation Theory* 25.3, pp. 266–279.

Martin, P., R. M. Murray, and P. Rouchon (2003). *Flat systems, equivalence and trajectory generation*. Centre Automatique et Systèmes, École des Mines de Paris.

Mellinger, D. and V. Kumar (2011). "Minimum snap trajectory generation and control for quadrotors". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. (Shanghai, China), pp. 2520–2525.

Milam, M. B., K. Mushambi, and R. M. Murray (2000). "A new computational approach to real-time trajectory generation for constrained mechanical systems". In: *Proceedings of the IEEE Conference on Decision and Control*. (Sydney, Australia). Vol. 1, pp. 845–851.

Molinari, N., J.-F. Durand, and R. Sabatier (2004). "Bounded optimal knots for regression splines". In: *Computational Statistics & Data Analysis* 45.2, pp. 159–178.

Nagy, Z. K. (2008). "Model based control approach for batch crystallization product design". In: *Proceedings of the 17th IFAC World Congress*. Vol. 17.

Nesterov, Y. (2000). "Squared functional systems and optimization problems". In: *High performance optimization*. Ed. by H. Frenk, K. Roos, T. Terlaky, and S. Zhang. Springer.

Packard, A. and J. Doyle (1993). "The complex structured singular value". In: *Automatica* 29.1, pp. 71–109.

Petit, N., Y. Creff, L. Lemaire, and P. Rouchon (2001). "Minimum time constrained control of acid strength on a sulfuric acid alkylation unit". In: *Chemical Engineering Science* 56.8, pp. 2767–2774.

Petit, N., M. B. Milam, and R. M. Murray (2001). "Inversion based constrained trajectory optimization". In: *Proceedings of 5th IFAC symposium on nonlinear control systems (NOLCOS)*. (St. Peterburg, Russia).

Plaumann, D. (2010). *Positivity of continuous piecewise polynomials*. Version 3. arXiv: 1004.3104v3 [math.OC].

Pólya, G. (1928). "Über positive Darstellung von Polynomen". In: *Vierteljahresschrift der Naturforschenden Gesellschaft in Zürich* 73, pp. 141–145. Reprinted in *Collected papers*. Ed. by R. P. Boas. MIT Press, Cambridge, MA, 1974, pp. 309–313.

Powers, V. and B. Reznick (2000). "Polynomials that are positive on an interval". English. In: *Transactions of the American Mathematical Society* 352.10, pp. 4677–4692.

Powers, V. and T. Wörmann (1998). "An algorithm for sums of squares of real polynomials". In: *Journal of Pure and Applied Algebra* 127, pp. 99–104.

Python programming language. Python software foundation. URL: www.python.org.

Raczy, C. (1997). "Commandes optimales en temps pour les systèmes différentiellement plats". thesis. Université des Sciences et Technologies de Lille.

Raczy, C. and G. Jacob (1999). "Fast and smooth controls for a trolley crane". In: *Journal of Decision Systems* 8.3, pp. 367–388.

Shapiro, A. (1997). "First and second order analysis of nonlinear semidefinite programs". In: *Mathematical Programming* 77, pp. 301–320.

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

Sira Ramírez, H. J. and S. K. Agrawal (2004). *Differentially flat systems*. Marcel Dekker. 467 pp.

Skjetne, R., T. I. Fossen, and P. V. Kokotović (2004). "Robust output maneuvering for a class of nonlinear systems". In: *Automatica* 40.3, pp. 373–383.

Spong, M. and S. Hutchinson (2005). *Robot modeling and control*. Wiley. 496 pp.

Suryawan, F., J. De Doná, and M. Seron (2011). "Minimum-time trajectory generation for constrained linear systems using flatness and B-splines". In: *International Journal of Control* 84.9, pp. 1565–1585.

— (2012). "Splines and polynomial tools for flatness-based constrained motion planning". In: *International Journal of Systems Science* 43.8, pp. 1396–1411.

Thomson, W. T. (1997). *Theory of vibrations with applications*. 5th ed. Pearson Education. 534 pp.

van Bergen, N. (2013). "Tijdsoptimaal padvolgen voor differentieel vlakke systemen: toepassing op een portaalkraan met visiegebaseerde evaluatie". thesis. KU Leuven.

Van den Broeck, L., M. Diehl, and J. Swevers (2011a). "A model predictive control approach for time optimal point-to-point motion control". In: *Mechatronics* 21.7, pp. 1203–1212.

— (2011b). "Model predictive control for time-optimal point-to-point motion control". In: *Proceedings of the 18th IFAC World Congress*, pp. 2458–2463.

Van Loock, W., S. Bellens, G. Pipeleers, J. De Schutter, and J. Swevers (2013). "Time-optimal parking and flying: solving path following problems efficiently". In: *Proceedings of the 2013 IEEE International Conference on Mechatronics*. (Vicenza, Italy), pp. 840–845.

Van Loock, W., G. Pipeleers, J. De Schutter, and J. Swevers (2011). "A convex optimization approach to curve fitting with B-splines". In: *Proceedings of the 18th IFAC World Congress*. (Milano, Italy), pp. 2290–2295.

Van Loock, W., G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers (2013). "Optimal path following for differentially flat robotic systems through a geometric problem formulation". In: *IEEE Transactions on Robotics*. Accepted.

Van Loock, W., G. Pipeleers, and J. Swevers (2013a). "Time-optimal path planning for flat systems with application to a wheeled mobile robot". In: *Proceedings of the 9th International Workshop on Robot Motion and Control*. (Wasowo Palace, Wasowo, Poland), pp. 192–196.

— (2013b). "Time-optimal quadrotor flight". In: *Proceedings of the European Control Conference*. (Zurich, Switzerland), pp. 1788–1792.

van Nieuwstadt, M. and R. M. Murray (1998). "Real time trajectory generation for differentially flat systems". In: *International Journal of Robust and Nonlinear Control* 8, pp. 995–1020.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2009). "Time-optimal path tracking for robots: a convex optimization approach". In: *IEEE Transactions on Automatic Control* 54.10, pp. 2318–2327.

Verscheure, D., M. Diehl, J. De Schutter, and J. Swevers (2009). "Recursive log-barrier method for on-line time-optimal robot path tracking". In: *Proceedings of the American Control Conference*. (St. Louis, MO, USA), pp. 4134–4140.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106 (1), pp. 25–57.

Yoshimoto, F., T. Harada, and Y. Yoshimoto (2003). "Data fitting with a spline using a real-coded genetic algorithm". In: *Computer-Aided Design* 35.8, pp. 751–760.

Zhou, K. and J. Doyle (1998). *Essentials of robust control*. Prentice Hall International.

Zuidwijk, R. (2005). *Linear Parametric Sensitivity Analysis of the Constraint Coefficient Matrix in Linear Programs*. report ERS-2005-055-LIS. Erasmus Research Institute of Management (ERIM).

Author's contributions

Articles in international peer reviewed journals

Debrouwere, F., W. Van Loock, G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers (2013). "Convex time-optimal robot path following with cartesian acceleration and inertial force and torque constraints". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*. Accepted.

Debrouwere, F., W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers (2013). "Time-optimal path following for robots with convex-concave constraints using sequential convex programming". In: *IEEE Transactions on Robotics*. Accepted.

Janssens, P., W. Van Loock, G. Pipeleers, and J. Swevers (2013). "Efficient computation of time-optimal point-to-point motion trajectories". In: *IEEE Transactions on Control Systems Technology*. In review.

Van Loock, W., G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers (2013). "Optimal path following for differentially flat robotic systems through a geometric problem formulation". In: *IEEE Transactions on Robotics*. Accepted.

Articles in international conference proceedings

Debrouwere, F., W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers (2012). "Time-optimal robot path following with cartesian acceleration constraints: a convex optimization approach". In: *Proceedings of the 13th Mechatronics Forum*. (Linz, Austria), pp. 469–475.

Debrouwere, F., W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers (2013a). "Optimal robot path following for minimal time versus energy loss trade-off using sequential convex programming". In: *Proceedings of the 2013 IEEE International Conference on Mechatronics*. (Vicenza, Italy), pp. 316–320.

— (2013b). "Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. (Karlsruhe, Germany), pp. 1908–1913.

Debrouwere, F., W. Van Loock, G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers (2013). "Time-optimal path following for robots with object collision avoidance using lagrangian duality". In: *Proceedings of the 9th International Workshop on Robot Motion and Control*. (Wasowo Palace, Wasowo, Poland), pp. 186–191.

Janssens, P., W. Van Loock, G. Pipeleers, and J. Swevers (2013a). "An efficient algorithm for solving time-optimal point-to-point motion control problems". In: *Proceedings of the 2013 IEEE International Conference on Mechatronics*. (Vicenza, Italy), pp. 682–687.

— (2013b). "Iterative learning control for optimal path following problems". In: *Proceedings of the 52nd IEEE Conference on Decision and Control*. (Firenze, Italy). Accepted.

Van Loock, W., S. Bellens, G. Pipeleers, J. De Schutter, and J. Swevers (2013). "Time-optimal parking and flying: solving path following problems efficiently". In: *Proceedings of the 2013 IEEE International Conference on Mechatronics*. (Vicenza, Italy), pp. 840–845.

Van Loock, W., G. Pipeleers, J. De Schutter, and J. Swevers (2011). "A convex optimization approach to curve fitting with B-splines". In: *Proceedings of the 18th IFAC World Congress*. (Milano, Italy), pp. 2290–2295.

Van Loock, W., G. Pipeleers, and J. Swevers (2011). "Optimal input design for flat systems using B-splines". In: *Proceedings of the 2011 American Control Conference*. (San Francisco, CA, USA), pp. 3281–3282.

— (2013a). "Time-optimal path planning for flat systems with application to a wheeled mobile robot". In: *Proceedings of the 9th International Workshop on Robot Motion and Control*. (Wasowo Palace, Wasowo, Poland), pp. 192–196.

— (2013b). "Time-optimal quadrotor flight". In: *Proceedings of the European Control Conference*. (Zurich, Switzerland), pp. 1788–1792.

Vukov, M., W. Van Loock, B. Houska, H. J. Ferreau, J. Swevers, and M. Diehl (2012). "Experimental validation of nonlinear mpc on an overhead crane using automatic code generation". In: *Proceedings of the 2012 American Control Conference*. (Montreal, Canada), pp. 6264–6269.

Abstracts in international conference proceedings

Debrouwere, F., W. Van Loock, M. Diehl, J. De Schutter, and J. Swevers (2012). "Time-optimal robot path tracking with cartesian acceleration constraint". In: *Proceedings of the 31st Benelux Meeting on Systems and Control*. (Heijen, The Netherlands).

Debrouwere, F., W. Van Loock, G. Pipeleers, and J. Swevers (2013). "Time-optimal path following for robots with object collision avoidance using lagrangian duality". In: *Proceedings of the 32nd Benelux Meeting on Systems and Control*. (Houffalize, Belgium).

Janssens, P., W. Van Loock, G. Pipeleers, and J. Swevers (2013). "Iterative learning control for time-optimal path following problems". In: *Proceedings of the 32nd Benelux Meeting on Systems and Control*. (Houffalize, Belgium).

Van Loock, W., G. Pipeleers, and J. Swevers (2011a). "Optimal motion planning for flat systems using B-splines". In: *Proceedings of the 30th Benelux Meeting on Systems and Control*. (Lommel, Belgium).

— (2011b). "Time optimal input design with B-splines: linear versus semidefinite programming". In: *Proceedings of the SIAM Conference on Optimization*. (Darmstadt, Germany).

— (2012). "Time optimal point-to-point motion trajectories: a convex optimization approach". In: *Proceedings of the 31st Benelux Meeting on Systems and Control*. (Heijen, The Netherlands).

— (2013). "Dynamically optimal polynomial splines for uncertain flexible motion systems: an LMI approach". In: *Proceedings of the 32nd Benelux Meeting on Systems and Control*. (Houffalize, Belgium).