

# Optimal Control of Switched Autonomous Systems: Theory, Algorithms, and Robotic Applications

A Thesis  
Presented to  
The Academic Faculty

by

**Henrik Axelsson**

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

School of Electrical and Computer Engineering  
Georgia Institute of Technology  
May 2006

# Optimal Control of Switched Autonomous Systems: Theory, Algorithms, and Robotic Applications

Approved by:

Dr. Magnus Egerstedt, Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Yorai Wardi, Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Erik Verriest  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Anthony Yezzi  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Dr. Spyros Reveliotis  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Dr. Ashraf Saad  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

*Date Approved: April 3rd, 2006*

# ACKNOWLEDGEMENTS

This thesis is the final result of many years of research at the Georgia Institute of Technology. During those years, I have met many people who influenced me, both in a professional and a personal way, whom I would like to thank. First and foremost, I would like to thank my advisers, professors Magnus Egerstedt and Yorai Wardi for their endless support and guidance. I also wish to thank professors Erik Verriest and Anthony Yezzi for serving on my reading committee, and professors Ashraf Saad and Spyros Reveliotis for being on my defense committee.

Furthermore, I would like to thank:

- My fellow graduate students for their great company and qualitative discussions: Muhammad Abubakr, Adam Austin, Shun-ichi Azuma, Muhamed Babaali, Staffan Björkenstam, Mauro Boccadoro, Florent Delmotte, Dennis Ding, Anders Gustafsson, Johan Isaksson, Meng Ji, Tejas Metha, Jeongseung Moon, Brian Smith, Ganesh Sundaramoorthi, and Dave Wooden.
- My friends in Atlanta for making my stay here enjoyable.
- My family for always being there for me.
- Finally, I would like to thank my fiancée for everything, I wouldn't have been where I am without you Annelise.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b> . . . . .	<b>iii</b>
<b>LIST OF TABLES</b> . . . . .	<b>vi</b>
<b>LIST OF FIGURES</b> . . . . .	<b>vii</b>
<b>SUMMARY</b> . . . . .	<b>x</b>
<b>I BACKGROUND AND INTRODUCTION</b> . . . . .	<b>1</b>
1.1 Optimal Switching Time Control . . . . .	3
1.2 Optimal Mode Scheduling . . . . .	6
1.3 Real Time Switching Time Control . . . . .	8
1.4 Optimal Mode Switching for Hybrid Systems with Unknown Initial State . . . . .	9
1.5 Reactive Robot Navigation using Optimal Switching Time Control . . . . .	10
<b>II OPTIMAL SWITCHING TIME CONTROL</b> . . . . .	<b>13</b>
2.1 Problem Formulation and Gradient Formula . . . . .	14
2.2 Optimality Condition and an Algorithm . . . . .	21
2.3 Numerical Example . . . . .	29
2.4 Conclusions . . . . .	30
<b>III OPTIMAL MODE SCHEDULING</b> . . . . .	<b>32</b>
3.1 Conceptual Framework for Algorithms' Convergence . . . . .	33
3.2 Problem Formulation . . . . .	36
3.3 Convergence Analysis . . . . .	42
3.4 Numerical Examples . . . . .	54
3.4.1 Linear System . . . . .	54
3.4.2 Nonlinear System . . . . .	55
3.5 Conclusions . . . . .	60
<b>IV REAL TIME SWITCHING TIME CONTROL</b> . . . . .	<b>61</b>
4.1 Problem Formulation . . . . .	62
4.2 Bounds on the Error In the State and Costate . . . . .	67
4.3 Real-Time Optimization . . . . .	70
4.4 Numerical Example . . . . .	73

4.5	Conclusion . . . . .	78
<b>V</b>	<b>OPTIMAL MODE SWITCHING FOR HYBRID SYSTEMS WITH UN-</b>	
	<b>KNOWN INITIAL STATE . . . . .</b>	<b>81</b>
5.1	Problem Formulation & Previous Results . . . . .	82
5.2	Minimax Optimization . . . . .	84
5.3	Numerical Example . . . . .	87
5.4	Conclusions . . . . .	91
<b>VI</b>	<b>REACTIVE ROBOT NAVIGATION USING OPTIMAL TIMING CON-</b>	
	<b>TROL . . . . .</b>	<b>93</b>
6.1	Behavior Based Robotics . . . . .	94
6.2	Optimal Control Derivation . . . . .	96
6.3	Guard Generation . . . . .	99
6.4	Multiple Obstacles . . . . .	103
6.5	Implementation . . . . .	105
6.6	Conclusions . . . . .	106
	<b>APPENDIX A — PROOFS . . . . .</b>	<b>109</b>
	<b>REFERENCES . . . . .</b>	<b>119</b>
	<b>VITA . . . . .</b>	<b>124</b>

# LIST OF TABLES

1	Switching vector, optimal simulation time, and robots position as a function of $t_c$ . . . . .	77
---	---	----

## LIST OF FIGURES

1	Switched autonomous system dynamics. . . . .	4
2	A mobile robot get its initial position via GPS from a Satellite. The specified position is $(x, y)$ but because of the error in the GPS reading, the robot can be anywhere inside the boxed area. . . . .	10
3	Mode transition occur as the guard predicates become true. . . . .	11
4	State-trajectories of $x_1$ (solid) and $x_\lambda$ (dotted) together with the current dynamic response functions. . . . .	20
5	Illustration of guaranteed descent using gradient descent algorithm when $\nabla J(\bar{\tau}) \in C^1$ and $\lambda \in [0, \bar{\lambda})$ . . . . .	22
6	Calculation of $r$ and $R$ for an example with five switches. . . . .	25
7	Calculation of descent direction $\bar{h}(\bar{\tau})$ , for an example with three blocks of switches. . . . .	28
8	Top figure: $J(\bar{\tau})$ is plotted as a function of the number of iterations of Algorithm 2.2.1. Bottom figure: The norm of the projected gradient is plotted as a function of the number of iterations of Algorithm 2.2.1. . . . .	30
9	Plot of $x(t)$ for different iterations of Algorithm 2.2.1. $x(t)$ is plotted with a dotted curve for every third iteration in Algorithm 2.2.1, and the final state-trajectory is plotted using a solid curve. $x_1(t)$ is plotted in blue and is the curve that starts by decreasing, $x_2(t)$ is plotted in red and starts by increasing. . . . .	31
10	Changes in the schedule obtained by inserting $f_{\sigma(\star)}$ centered at time $\tau$ for an interval of length $\lambda$ . . . . .	38
11	(a) Cost and norm of search direction as a function of the number of gradient descent steps. The red line in the bottom plot corresponds to $\ h\  = 0.1$ . (b) $D(\xi)$ every time we perform an insertion . . . . .	55
12	The double tank process. . . . .	56
13	Cost together with the dimension of the switching time vector as a function of the number of gradient descent iterations. . . . .	58
14	$D(\xi)$ every time we perform an insertion together with $-\omega$ (dotted). . . . .	58
15	Final state trajectories: $x_1(t)$ dotted, $x_2(t)$ solid, and $x_r(t)$ dashed. . . . .	59
16	Mode structure for the final schedule. . . . .	59
17	Simulated time horizon versus real time horizon: While the system evolves for $\Delta$ seconds, starting from time $t_c$ , $x(t)$ and $p(t)$ are simulated between $t_c$ and $t_c + T$ with step-length $\delta t$ , using Euler's method to solve for $x$ and $p$ . . . . .	62

18	Right hand side of Eq. (123). (a) $C_{CPU}$ is not small enough for our bound to be useful. The minimum value 4.67 is attained at $T = 2.79s$ . The norm of the gradient in our example is typically less than 1 and hence, the plot does not give any information. (b) $C_{CPU} = 10^{-6}$ and we see that it is optimal to simulate all the way until the end and that the error at the end is smaller than 0.005. If we simulate until the end we will get a step-length, $\delta t = 12\mu s$ which is very small. . . . .	76
19	Right hand side of (123) evaluated at (a) $t_c = 0$ , and (b) $t_c = 0.25$ . . . . .	78
20	Robots initial (dotted) and final (solid) trajectory . . . . .	79
21	Cost as a function of iterations of Algorithm 4.3.1. At iteration 1, $t_c=0$ , at iteration 2, $t_c = 0.25$ etc. . . . .	79
22	Mode switching occur when the state trajectory intersect a switching surface. In this case, the switching surface is a circle parameterized by the radius $a$ . . . . .	84
23	Calculation of $h$ given four initial states and their respective gradients. $x_1$ through $x_3$ are active initial states. . . . .	87
24	Initial states used: $\mathbb{X}_i$ contains the points with index $i, i - 1, \dots, 0$ . . . . .	89
25	(a) Change in maximum cost. (b) $\ h\ $ and $\delta$ as a function of the number of gradient descent iterations in Algorithm 5.2.2. . . . .	90
26	Change in $a$ and final trajectory: (a) Change in $a = (a_1, a_2)^T$ as a function of the number of gradient descent iterations in Algorithm 5.2.2. $a_1$ is the radius of the <i>obstacle-avoidance</i> switching surface, $a_2$ is the radius of the <i>go-to-goal</i> switching surface. (b) Final trajectory giving the maximum cost together with initial and final $a$ , and $\mathbb{X}_2$ . Initial $a_1$ dotted, initial $a_2$ solid, final $a_1$ and $a_2$ are equal to each other and are represented by the marked circle. . . . .	91
27	Different Initial States: (a) The optimal state trajectories for both initial states are shown. (b) $\lim_{\lambda \downarrow 0} \frac{dJ_{u_{\odot},t}}{d\lambda}$ is shown for both trajectories as function of $t$ when $B = (g)$ . . . . .	101
28	State trajectories and associated guard structure: (a) State trajectories for executing Algorithm 6.3.1 for a set of initial states along the $x_1$ -axis. (b) An approximation of the associated guards. Guard I corresponds to the region inside the stars and to the left of the vertical line between the goal and the obstacle, similar for Guard II. . . . .	102
29	The optimal solution to problem $P$ given in terms of guards associated with each point obstacle encountered in the robots path. . . . .	103
30	Optimal solution given in terms of guards associated with each obstacle encountered in the robots path for the case when the robot encounters several obstacles. . . . .	105
31	Switching surfaces for experimental comparison. The “standard” switching surface is defined by a semi-circle where the robot should evolve according to $\kappa_{\odot}$ if the robot is inside $G_{\odot}$ and vice versa for $G_{\ominus}$ . . . . .	106



32	Experimental setup for testing our real time reactive navigation method. . .	107
33	The simulated trajectory plotted using the odometry and sensor readings from the Magellan Pro for our Optimal Transitions System. . . . .	107

# SUMMARY

As control systems are becoming more and more complex, system complexity is rapidly becoming a limiting factor in the efficacy of established techniques for control systems design.

To cope with the growing complexity, control architectures often have a hierarchical structure. At the base of the system pyramid lie feedback loops with simple closed-loop control laws (gains, integrators, etc.). These are followed, at the higher level, by larger feedback laws based on adaptation, and finally, at the highest level, by discrete control logics. Such hierarchical systems typically have a *hybrid* nature, with a continuous state space at the lower level and a finite (logical) state space at the higher level.

A common approach to addressing these types of complexity consists of decomposing, in the time domain, the control task into a number of modes, i.e. control laws dedicated to carrying out a limited task. This type of control generally involves switching laws among the various modes, and its design poses a major challenge in many application domains. The primary goal of this thesis is to develop a *unified framework* for addressing this challenge. The framework is cast in the setting of *optimal control of hybrid systems*, and it concerns the development of fundamental concepts of switching control laws and algorithmic techniques for their implementation.

To this end, the contribution of this thesis is threefold:

1. An algorithmic framework for how to optimize the performance of *switched autonomous systems* is derived. The optimization concerns both the sequence in which different modes appear in and the duration of each mode. The optimization algorithms are presented together with detailed convergence analysis. In fact, one of the main contributions of this thesis is to define what convergence means when optimizing a system where the number of modes is not fixed.

2. Control strategies for how to optimize switched autonomous systems operating in *real time*, and when the *initial state* of the system is unknown, are presented. As the proposed optimization framework is cast in the setting of optimal control, it is assumed that the initial state of the systems is known and that the controller has sufficient time to calculate new control values. It will be shown how the framework can be modified to cover both the real time constraint and the situation when the initial state of the system is unknown.
3. A control strategy for how to optimally navigate an autonomous mobile robot in real-time is presented and evaluated on a mobile robotics platform. The control strategy uses optimal switching surfaces for when to switch between different modes of operations (behaviors). The switching surfaces are generated off-line, using the optimization framework, and are then transitioned to a real robotic platform operating in an unknown environment.

# CHAPTER I

## BACKGROUND AND INTRODUCTION

Over the last few decades, there has been a vast amount of research directed toward the control of complex systems characterized by discrete logical decision making at the highest level and continuous variable dynamics at the lowest level. These systems are commonly referred to as hybrid systems. In hybrid systems, the continuous-time dynamics are typically associated with dynamical laws (e.g., physical principles) and the discrete event dynamics can be logic devices, such as switches, digital circuitry, and software code. Examples where hybrid systems arise include manufacturing, where a discrete supervisory controller has to schedule work releases into a factory [54], air traffic control [66], and when a control module has to switch its attention among a number of subsystems [45], [60], [69]. Hybrid systems can also be used to model robotics systems [28] via the behavior-based paradigm [5].

This thesis addresses a particular class of hybrid systems, called switched autonomous systems, where the continuous time control variable is absent and the continuous time dynamics change at discrete times (*switching times*). These systems will be studied in a general sense, where we assume that we can control both the sequence in which different dynamic response functions (*modes*) appear, as well as the times when we switch between the modes. One of the major contributions of this thesis is the derivation of a framework for optimizing both over the switching times and over the sequence in which the modes appear. To this end, optimality conditions and optimization algorithms are derived. Having presented a general framework for how to optimize over both the sequence of modes and the switching times, we confine the discussion to the case when the mode sequence is fixed and consider two additional switching time optimization problems. The first problem is that of optimizing the switching times for a switched autonomous system performing in real time. As the system is to perform in real time, it is desirable that the controller produce adequate control values as fast as possible. However, the controller only has a finite amount

of computational power available. Hence there is a trade-off between the precision of the solution we get and the time the controller spends calculating new control values. This trade-off will be researched and a solution will be presented that dictates how long the controller should spend calculating new control values in order to achieve a satisfactory system performance. For the second problem we assume that the exact location of the initial state is not known, and we will optimize the switching times with respect to a set of initial states using a minimax [56] strategy.

Another major contribution of this thesis is the derivation of an optimal control inspired strategy for navigating an autonomous mobile robot in a cluttered environment. Viewing the navigation problem as an optimal switching time and mode sequencing problem, a control strategy is presented and evaluated on a real robotics platform.

Before progressing deeper into the above discussed problems, a short historical perspective is given to describe the events that lead up to the boom in research related to hybrid systems that started in the late 1980s. Frequently, in hybrid systems in the past, event-driven dynamics were studied separately from time-driven dynamics. The event-driven dynamics were approached via automata [24] or Petri net models [58], and the time-driven dynamics typically via differential or difference equations. By the end of the 1980s more and more control systems had a vast amount of computer code at their highest level, embedded control systems started to appear in everyday products such as coffee makers and cars, and digital controllers were used to control continuous systems via sampling. All the systems described above are hybrid in nature. To fully understand these systems' behavior and to be able to test whether or not performance specifications (e.g., stability, robustness, optimality,...) can be met, one needs to model all dynamics together with their interactions. Only then can problems such as optimization of the whole process be addressed properly. Furthermore, hybrid systems can also be used to reduce complexity in systems.

The above discussion justifies the need for hybrid systems for people working with control systems. However, the need for hybrid systems has also become apparent for computer scientists with emphasis on the verification of software models [49].

Next, some of the early work that has influenced the field of hybrid control will be

introduced.

A reasonable paper to start from, for the purpose of this thesis, is [19] written by Roger Brockett in 1993. In that paper, Brockett proposes four different models for hybrid systems. The paper can be thought of as providing a framework for modelling hybrid systems and it relates the proposed models to real-world systems. In [20], also by Brockett, the problem of controlling different processes with the use of one shared communication link is considered. The sequence in which the different processes are visited is assumed to be fixed and results relating to when the control system is stable are presented for the case of linear dynamics.

In [18], written by Michael Branicky *et al.*, a general modelling framework for hybrid systems is presented. The authors observe four phenomena that occur in a real-world system, including autonomous and controlled switching of the dynamics of the state variable and autonomous and controlled impulses on the state variable. They then propose a model covering the above mentioned phenomena and use it as a first step toward a general theory of optimal control of hybrid systems.

In [66], by Claire Tomlin *et al.*, the problem of designing safety zones for air traffic control is considered. The authors translate safety specifications into restrictions on the system's reachable sets of states. Then, with the help of game theory and optimal control analysis, Hamilton-Jacobi equations are derived that describe the boundaries of the reachable set. Although this result is theoretically strong, it is not all that applicable to real systems because of the complexity associated with the computation of the Hamilton-Jacobi equations.

The above papers are presented as a short survey to hybrid systems and serve as an introduction to the discussion about the optimal switching time control of switched autonomous systems that follows. Furthermore, many of the problems presented in the above mentioned papers will be touched upon in this thesis.

## ***1.1 Optimal Switching Time Control***

Throughout this section, we assume that the mode sequence (the sequence in which different dynamic response functions appear) and the initial state  $x_0$  are given and fixed. We can

then view our system as a switched mode system whose dynamic response changes among various modes according to a prescribed supervisory control law, as shown in Figure 1. Let  $\{x(t)\}_{t=0}^T$  denote the state trajectory of the system, and suppose that it evolves according to the equation  $\dot{x} = f(x, t)$ , where the dynamic response function  $f : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}^n$  comprises a sequential assignment of functions  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,  $i = 1, 2, \dots$ . Let us fix  $T > 0$  and suppose that the dynamic response changes  $N$  times in the interval  $[0, T]$ . Denoting the switching times by  $\tau_i$ ,  $i = 1, \dots, N$ , in increasing order, and defining  $\tau_0 := 0$  and  $\tau_{N+1} := T$ , we have that

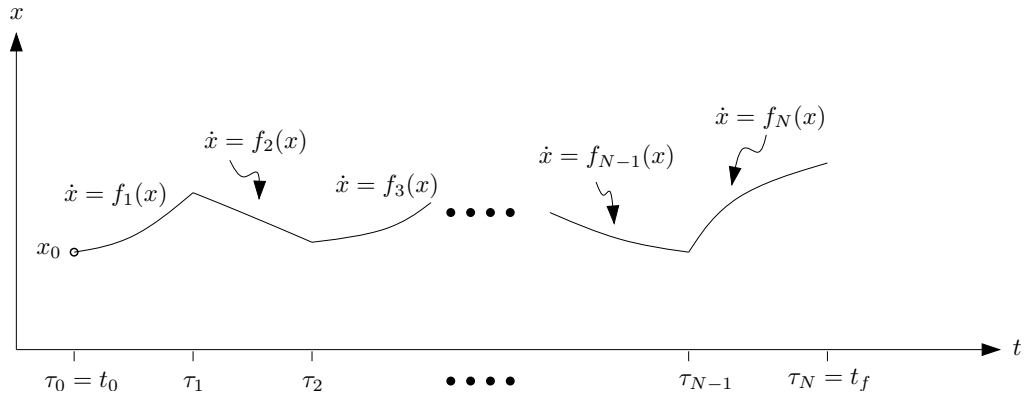
$$0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = T, \quad (1)$$

where (1) describes the set we will be optimizing over. We denote by  $\bar{\tau}$  the vector of switching times, i.e.,  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$ . Furthermore, suppose that  $f(x, t) = f_i(x)$  for every  $t \in [\tau_{i-1}, \tau_i)$  and for every  $i = 1, \dots, N+1$ . We then have the following differential equation defining the system's dynamics:

$$\dot{x}(t) = f_i(x(t)), \quad t \in [\tau_{i-1}, \tau_i), \quad i \in \{1, \dots, N+1\}, \quad (2)$$

i.e., the system first evolves according to  $\dot{x}(t) = f_1(x(t))$  between time  $\tau_0$  and time  $\tau_1$ . At time  $\tau_1$ , the dynamics of the system changes and the system will evolve according to  $\dot{x}(t) = f_2(x(t))$  between time  $\tau_1$  and  $\tau_2$ , and so on.

Looking at (2), we note that, given the initial state, the state trajectory of the system is completely determined by the switching times. A more general setting includes an exogenous



**Figure 1:** Switched autonomous system dynamics.

input  $u(t)$  in the right hand side of (2), but we consider only the autonomous case where such an input is absent.

These systems arise in a variety of applications including situations where a control module has to switch its attention among a number of subsystems [45, 60, 69], collect data sequentially from a number of sensory sources [20, 29, 40], or behavior-based robotic systems [5, 27]. In order to quantify what “good performance” means for the applications described above, one can define a cost function as a function of the state trajectory. The cost function can characterize numerous things, including the stability of the system and how close the system follows a given reference trajectory.

More formally, the problem we will discuss is that of determining the switching times in order to optimize a cost criterion of the form

$$J(\bar{\tau}) = \int_0^T L(x)dt, \quad (3)$$

for a given cost function  $L : \mathbb{R}^n \rightarrow \mathbb{R}$ . As noted before,  $x(t)$  is determined by the switching times and therefore we will minimize  $J$  with respect to  $\bar{\tau}$ . Although the set we are optimizing over (1) is convex,  $J(\bar{\tau})$  is generally not convex. Hence the algorithm we present only guarantees that the switching time vector converges to a local minima.

A lot of work has been done to solve the optimal switching time problem [14, 1, 2, 34, 37, 40, 45, 59, 60, 69, 72, 71]. In particular, for discrete time linear systems, [45] presents an algorithm that optimizes over both the mode sequence and the switching times based on a dynamic programming approach. For continuous time linear systems, [34] presents a solution to the switching time problem using a state feedback form where a switch occurs if the state enters a given region of the state space. Of particular importance to us is the work presented in [72] and [71] where general nonlinear autonomous systems are considered. In those two papers, nonlinear programming algorithms that compute the gradient and second order derivatives of the cost criterion are presented. Reference [72] can be thought of as providing the starting point for the results presented for the switching time optimization problem. However, we will develop a simpler formula for the gradient than the one presented in [72] that lends itself nicely to the case when we are optimizing over the mode sequence



as well.

Although a lot of work has been done on switching time optimization, our result presents a novel framework (including optimality criteria, descent direction, and a convergent algorithm) for how to optimize over the switching times for general nonlinear systems. Therefore, and to serve as a basis for the mode-scheduling problem, the switching time optimization problem is a well-motivated research subject.

## 1.2 *Optimal Mode Scheduling*

Once the switching time optimization problem has been solved, those results will be extended to the case when we consider optimizing over the mode schedule.

Given a cost functional defined on the state trajectory of the system, the mode scheduling problem amounts to the scheduling of the modes to minimize this cost functional. As noted in the previous section, such optimal scheduling problems arise in a number of application domains. The assumption that the mode sequence is given can only be justified for systems where we have enough knowledge to determine a good mode sequence by inspection, or if the mode sequence is given as a part of the systems' specification. Examples of systems when the mode sequence is given include various applications related to the gear-box of a car [41] (e.g., how to change the gears to accelerate a car as fast as possible) and scheduling in a manufacturing environment where a product has to go through a predefined number of steps before it is done [54]. If the system is complex enough, a good mode sequence might not be attained by simply inspecting the system. This can be the case when a control module acts as a communication link between a large number of processes. Optimizing over the mode sequence can also be used to add new modes to an already existing set of modes by observing the relative duration of each mode and designing new modes based on this information.

The mode sequence, being a scheduling parameter, is a discrete parameter and we refer to it as the *sequencing variable*. From the standpoint of optimization, it is generally much easier to deal with the switching times than with the sequencing variable. This is because the optimal sequencing problem is in general expected to have exponential complexity.

The subject of optimal mode scheduling is a well-studied subject, and several approaches to reduce the cost by altering the mode sequence have been reported in the literature [45, 13, 23, 64, 65, 62, 3, 8, 67]. In particular [23, 64, 3, 8] use the maximum principle to derive optimality conditions and optimization algorithms based on time discretization. The algorithm presented in [23] only optimizes over the sequencing variable for a mode sequence with a given number of modes, and the algorithm presented in [3] requires a certain predefined time to elapse between any two switches. Our algorithm does not have these restrictions.

From the discussion above, it is clear that no computationally reasonable framework has been presented for how to reduce the cost by inserting new modes into the current mode schedule. Such an approach is presented in Chapter III.

In particular, having the mode scheduling optimization problem in mind, we illustrate a way to improve on a given mode sequence by inserting a new mode over a brief period of time in such a way that the cost functional is reduced. This has motivated the development of an algorithm that alternates between two phases: in the first phase it solves the timing-optimization problem for a given mode sequence, and in the second phase it modifies the mode sequence by inserting to it a new mode at a suitable time. In the latter phase the algorithm uses local sensitivity information, and hence it is essentially a local search algorithm. Thus, it seeks only a local minimum to the scheduling problem (in a sense defined in Chapter III) and it appears thus to evade the complexity issue that is inherent in the problem of computing globally optimal schedules.

In the first phase, the results presented in the switching time optimization chapter (Chapter II) are used to present an algorithm that solves a constrained nonlinear programming problem whose dimension is related to the number of modes. This number tends to grow since a new mode is inserted each time the algorithm enters this phase. Thus, the algorithm can be viewed as operating not on a single variable space, but rather on a set of nested Euclidean spaces having increasing dimensions. Having a variable space of increasing dimension, the issue of convergence of the algorithm raises two interesting questions:

- (i) What is the proper meaning of “convergence” of such an algorithm?

(ii) Does the specific algorithm in question converge in that sense?

It is these two questions that we address and that constitute a significant part of this thesis.

### 1.3 *Real Time Switching Time Control*

Having presented the mode scheduling problem in the previous section, we now confine the discussion to the case when the mode sequence is fixed and consider the problem of real time switching time optimization.

The problem of controlling a process running in real time has received a lot of attention lately [7, 46, 38, 63]. This is partly because embedded controllers have been introduced *en masse* in a number of novel yet resource-intense applications, such as active structure control [32], advanced automotive processes [17], and autonomous robotics [42]. To this end, resource management has become a bottleneck.

We approach the resource management problem from a computational vantage point using the results derived in Chapters II and III for the switching time optimization problem. In particular, we investigate how computationally costly optimal control algorithms can be modified in such a way that they become applicable to real time scenarios.

By “real time” we understand hard constraints on the time the control processes have available to them before a result must be delivered. In the context of optimal control, these constraints will be translated into constraints on the accuracy of the numerical algorithms used to update the switching time vector. In particular, we investigate trade-offs between the horizon over which the solution is obtained and the precision of the numerical algorithm. Denoting the simulation horizon by  $T$  and the step-length (used when solving for the state variables in real time) by  $\delta t$ , the problem we solve is to minimize the cost with respect to  $T$  and  $\delta t$  subject to the real time constraint that the controller only has a finite amount of time, denoted by  $\Delta$ , available to calculate new switching times. To this end, we present a conservative bound of the norm of the difference between the real gradient and the gradient we calculate in real time. We will then minimize this upper bound with respect to  $T$  and  $\delta t$  subject to the real time constraint

$$g(T, \delta t) \leq \Delta,$$

where  $g$  is the computation time.

The problem of controlling a process running in real time has been studied in the context of receding horizon control [51, 48, 44] (also known as model-predictive control). There, the current control action is obtained by solving, at each sampling instant, a finite horizon open-loop optimal control problem using the current state of the system as the initial state. This optimization yields an optimal control sequence, and the first control in this sequence is applied to the plant. This procedure is then repeated.

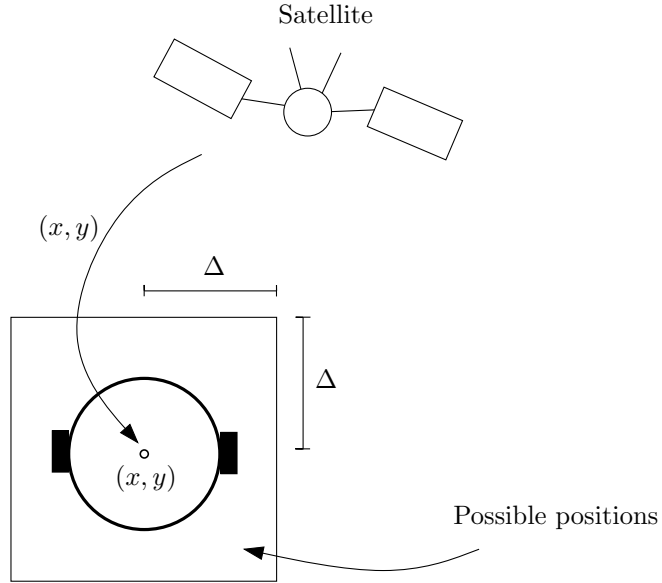
Although the receding horizon framework produces a sub-optimal solution in real time, the discussion of how far into the future to simulate versus the precision of the resulting solution is typically not studied. This is what makes our contribution novel since it tells us how far into the future to simulate and what precision we can expect doing that.

#### ***1.4 Optimal Mode Switching for Hybrid Systems with Unknown Initial State***

The results presented in Chapters II through IV all assume that the initial state of the system is given and fixed. For many applications this is not the case. An example of this is mobile robot navigation where the robot gets its position from a Global Positioning System (GPS). These systems typically have a nontrivial error associated with them. Hence, if the GPS indicates that the robot is at a point  $(x, y)$  the robot can be anywhere within the interval  $(x - \Delta, x + \Delta) \times (y - \Delta, y + \Delta)$ , for some positive constant  $\Delta$ . An illustration of this is shown in Figure 2.

As a result, solving the switching time optimization problem for a given fixed initial state might not give a good solution if the robot's position is given by a GPS. Instead, we will take an alternate approach and solve the optimization problem by minimizing the worst possible cost for all state trajectories starting in  $(x - \Delta, x + \Delta) \times (y - \Delta, y + \Delta)$ . To this end we have a minimax problem [56, 26]. Minimax problems have been considered for a long time and have been applied for example in Game Theory [35] and Optimal Control [47].

The novelty of our approach is the extension of our switching time optimization results



**Figure 2:** A mobile robot get its initial position via GPS from a Satellite. The specified position is  $(x, y)$  but because of the error in the GPS reading, the robot can be anywhere inside the boxed area.

to the case when the initial state is unknown.

### 1.5 *Reactive Robot Navigation using Optimal Switching Time Control*

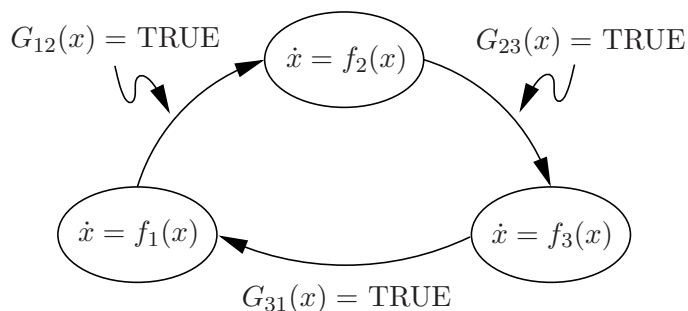
Once the optimal switching time/mode-scheduling framework has been derived in Chapters II and III, those results will be applied to a robotic navigation problems. In particular, supervisory control laws will be derived for an autonomous mobile robot navigating in a cluttered environment.

In the literature on robot navigation, two distinctly different approaches have emerged. The first approach, denoted as the *reactive* approach (following the terminology in [5]), consists of designing a collection of behaviors, or modes of operation, such as “avoid obstacle” or “approach goal” [4, 61, 21]. These different behaviors are defined through a particular control law dedicated to performing a specific task, and the robot switches between different behaviors as obstacles, landmarks, etc. are encountered in the environment. This way of structuring the navigation system has the major advantage that it simplifies the design task. Each controller is designed with only a limited set of objectives under consideration

and no elaborate world maps are needed. Unfortunately, very little can be said analytically about such systems, and we contrast them with the second approach under consideration here, namely, the *deliberative* approach [55, 28, 42]. Here the motion is carefully planned out in advance and care can be taken to minimize energy consumption and so on. This plan-based approach has proved useful in structured environments, e.g., in industrial settings, while unstructured environments pose a challenge. This is because there is normally a hefty computational burden associated with path planning and optimal control. Even if one is willing to pay this cost once, as soon as unmodeled obstacles are encountered, the cost will be incurred again.

In this thesis, we stay within the reactive navigation architecture but argue that optimality might still be relevant. Assuming that a number of control modes, or behaviors, have been designed, the question remains when to switch between them. This problem can be referred to as the guard design problem for hybrid systems, where a guard enables the transition between different modes of operation. Our approach is thus similar in spirit to the program developed in [66] where the guards were derived based on game theory to ensure safety in a multi-aircraft scenario. Formally, the state of the system evolves in mode  $i$  as  $\dot{x} = f_i(x)$  until  $G_{ij}(x) = \text{TRUE}$ , at which point the mode changes from  $i$  to  $j$ , as seen in Figure 3. The particular problem that we investigate is that of switching between *go-to-goal* and *avoid-obstacle* in an optimal manner.

Previously proposed guards typically involve a safety distance  $\Delta$  so that  $\dot{x} = f_g(x)$  (subscript  $g$  denotes go-to-goal) as long as  $\|x - x_{ob}\| > \Delta$ , where  $x_{ob}$  is the location of the



**Figure 3:** Mode transition occur as the guard predicates become true.

obstacle ([28, 42]). If  $\|x - x_{ob}\| \leq \Delta$ , then  $\dot{x} = f_o(x)$  (subscript  $o$  denotes avoid-obstacle) and hence the guard is defined through a circle centered at  $x_{ob}$  with radius  $\Delta$ . One can thus view the optimal control problem as a problem of determining the optimal radius  $\Delta$ . More generally, one can also optimize a parameterized surface  $g_\alpha(x) = 0$  with respect to  $\alpha$  (see [70, 15]). As an example of this, assume that the robot switches to an avoid-obstacle behavior if  $g_\alpha(x) < 0$  and it switches back to a go-to-goal behavior when  $g_\alpha(x) \geq 0$ . Then, by choosing  $g_\alpha = \|x_{ob} - x\| + \alpha$ , one can control the radius of the guard through changing  $\alpha$ .

Unfortunately, no guarantee can be given that we are optimizing over the right surface class (i.e., did we choose the right  $g_\alpha$ ) and therefore we take an alternate route and view the optimization problem as a switching time control problem. Subject to certain regularity conditions on the dynamics and the cost, it is shown how to obtain the optimal surface by applying the results presented in Chapters II and III. Once such a surface has been obtained in a potentially costly simulation, it can be implemented as a guard in a real time reactive navigation system.

## CHAPTER II

### OPTIMAL SWITCHING TIME CONTROL

This Chapter concerns the problem of optimizing the switching times for a switched dynamical system when the mode sequence is given and fixed. These systems are often described by differential inclusions of the form

$$\dot{x}(t) \in \{g_\alpha(x(t), u(t))\}_{\alpha \in A}, \quad (4)$$

where  $x(t) \in \mathbb{R}^n$  is the state variable,  $u(t) \in \mathbb{R}^k$  is the control variable, and  $\{g_\alpha : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n\}_{\alpha \in A}$  is a collection of continuously differentiable functions, parameterized by  $\alpha$  belonging to some given set  $A$  describing the dynamic response functions. Hence the different functions in the set  $A$  are the modes that the system can evolve according to. The time  $t$  is confined to a given finite-length interval  $[0, T]$ .

As mentioned in the introduction, these systems arise in numerous applications, including situations where a control module has to switch its attention among a number of subsystems [45, 60, 69], or collect data sequentially from a number of sensory sources [20, 29, 40]. In general, these systems occur whenever a process switches between different modes of operations corresponding to different dynamics of a plant. A supervisory controller is normally engaged for dictating the switching law, i.e., the rule for switching among the functions  $g_\alpha$  in the right-hand side of (4).

This Chapter only considers the special case of autonomous systems, where the continuous control term  $u(t)$  is absent and the control variable consists solely of the switching times. It should be mentioned that it might be possible to extend the results presented in this Chapter to the case when a continuous control signal is present. One possible way of doing this could be through a two-stage optimization process where the first stage optimizes  $u(t)$ , using calculus of variation, and the second stage optimizes the switching times using the results presented in this Chapter. Running these two stages iteratively should decrease



the cost. Furthermore, some results have been presented, in a different setting than ours, for the case when one optimizes over the switching times and the continuous control signal simultaneously, see [25], [23], and [8] for results of this nature.

In order to solve the switching time optimization problem, Section 2.1 formulates the problem and derives a formula for the gradient of the cost function. Section 2.2 derives an optimality condition having an intuitive appeal, and uses it to present a gradient descent algorithm. Section 2.3 presents two numerical results, and Section 2.4 concludes the Chapter.

## 2.1 Problem Formulation and Gradient Formula

Consider an autonomous switched mode dynamic system where the initial state  $x_0 \in \mathbb{R}^n$  and the final time  $T > 0$  are given. The functions  $g_\alpha$  in the right-hand side of (4) correspond to the modes of the system, and hence will be referred to as the *modal functions*. Suppose that the system switches between the modes (and their corresponding modal functions) a finite number of times  $N$  in the time-interval  $[0, T]$ . Let us denote the switching times by  $\tau_i$ ,  $i = 1, \dots, N$ , in nondecreasing order, and further define  $\tau_0 := 0$  and  $\tau_{N+1} := T$ . Then according to (4) and since the system is autonomous, for every  $i \in \{1, \dots, N+1\}$  there is an associated index term  $\alpha(i) \in A$  such that

$$\dot{x} = g_{\alpha(i)}(x), \quad \text{for all } t \in [\tau_{i-1}, \tau_i], \quad i = 1, \dots, N+1, \quad (5)$$

where at the boundary points  $\tau_{i-1}$  and  $\tau_i$  the derivative term  $\dot{x}(t)$  is replaced by the appropriate one-sided derivative. Note that the state trajectory  $x(t)$  is thus well defined and continuous throughout the interval  $[0, T]$ . Furthermore, we call the index-sequence  $\{\alpha(i)\}_{i=1}^{N+1}$  the *modal sequence*, and denote it by  $\sigma$ . Let  $L : \mathbb{R}^n \rightarrow \mathbb{R}$  be a given instantaneous cost function, and define the total cost  $J$  by

$$J = \int_0^T L(x(t))dt. \quad (6)$$

In order to proceed, we make the following assumption concerning the modal functions  $g_\alpha$  and the instantaneous cost function  $L$ .

**Assumption 1** (i) The functions  $g_\alpha$  and  $L$  are continuously differentiable on  $\mathbb{R}^n$ .

(ii) There exists a constant  $K_0 > 0$  such that, for every  $x \in \mathbb{R}^n$ , and for all  $\alpha \in A$ ,

$$\|g_\alpha(x)\| \leq K_0(\|x\| + 1). \quad (7)$$

It should be noted that the second assumption follows directly from Lipschitz continuity and boundedness of  $g_\alpha(x)$ . Observe that  $J$  is a function of the modal sequence  $\sigma = \{\alpha(i)\}_{i=1}^{N+1}$  as well as the switching times  $\tau_1, \dots, \tau_N$ . In this and the next sections we assume a fixed modal sequence  $\sigma$  and consider  $J$  as a function of the switching times. To simplify the notation, let us define the functions  $f_i$ ,  $i = 1, \dots, N + 1$ , by  $f_i = g_{\alpha(i)}$ . Then, (5) assumes the following form,

$$\dot{x}(t) = f_i(x(t)), \quad \text{for all } t \in [\tau_{i-1}, \tau_i], \quad i = 1, \dots, N + 1, \quad (8)$$

with the given initial condition  $x(0) = x_0$ . Furthermore, let us denote the set of switching times by  $\bar{\tau}$  in a vector form, i.e.,  $\bar{\tau} := (\tau_1, \dots, \tau_N)^T \in \mathbb{R}^N$ . Then  $J$  is a function of  $\bar{\tau}$  via (8) and (6), and hence, it is denoted by  $J(\bar{\tau})$ . Optimizing over the switching times we consider the following optimization problem, denoted by  $P_\sigma$ :

$P_\sigma$ : Minimize  $J(\bar{\tau})$  subject to the inequality constraints:  $0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = T$ .

This section derives a formula for the gradient  $\nabla J(\bar{\tau})$  that will be used later in a gradient descent algorithm. We first need a technical, preliminary result, Lemma 2.1.2, whose description and statement follow. Recall that the final time,  $T$ , is fixed. Given constants  $C > 0$ ,  $K_1 > 0$ ,  $K_2 > 0$ , and a convex compact set  $\Gamma \subset \mathbb{R}^n$ , we denote by  $H[C; K_1; K_2; \Gamma]$  the set of Lebesgue measurable functions  $h : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}^n$  having the following four properties:

1.  $\|h(x, t)\| \leq C$  for every  $(x, t) \in \Gamma \times [0, T]$ ,
2.  $h(x, t)$  is continuously differentiable in  $x \in \mathbb{R}^n$  for all  $t \in [0, T]$ ,
3.  $\|h(x_2, t) - h(x_1, t)\| \leq K_1\|x_2 - x_1\|$  for every  $x_1 \in \Gamma$ ,  $x_2 \in \Gamma$ , and  $t \in [0, T]$ ,

$$4. \quad \|\frac{\partial h}{\partial x}(x_2, t) - \frac{\partial h}{\partial x}(x_1, t)\| \leq K_2 \|x_2 - x_1\| \text{ for every } x_1 \in \Gamma, x_2 \in \Gamma, \text{ and } t \in [0, T].^1$$

We remark that the definition of  $H[C; K_1; K_2; \Gamma]$  does not require continuity of  $h(x, t)$  in its second variable,  $t$ . Hence,  $h(x, t)$  might be comprised of different modal functions at different instants of time, as needed for our presentation.

Now fix constants  $C > 0$ ,  $K_1 > 0$ , and  $K_2 > 0$ , and a convex compact set  $\Gamma \subset \mathbb{R}^n$ , and let  $h_1 \in H[C; K_1; K_2; \Gamma]$  and  $h_2 \in H[C; K_1; K_2; \Gamma]$  be two given functions. Let  $x_1(t)$  and  $x_2(t)$  be defined by the respective differential equations,  $\dot{x}_1(t) = h_1(x_1(t), t)$  and  $\dot{x}_2 = h_2(x_2(t), t)$ ,  $t \in [0, T]$ , with a common initial condition,  $x_1(0) = x_2(0) = x_0 \in \Gamma$ . Define  $\Delta h(x, t) := h_2(x, t) - h_1(x, t)$  and  $\Delta x(t) := x_2(t) - x_1(t)$ . Let  $\Phi(t, \tau) \in \mathbb{R}^{n \times n}$  denote the state transition matrix of the linearized system  $\dot{z} = \frac{\partial h_1}{\partial x}(x_1(t), t)z$ .

Lemma 2.1.2 amounts to a sensitivity-analysis result of solutions to differential equations. Whereas various sensitivity-analysis results are well-known (see for instance Section 5.6 of [56]), we have not seen one in the particular form of Lemma 2.1.2, nor could we prove the lemma as an immediate corollary of one of the established results. In the proof of Lemma 2.1.2, Bellman-Grönwall's inequality is used, formally presented in the following lemma.

**Lemma 2.1.1** (*Bellman-Grönwall's Inequality*) *Let  $\xi : [t_0, T] \rightarrow \mathbb{R}^+$  be an integrable function, and suppose that there exist constants  $C \geq 0$  and  $K \geq 0$  such that, for all  $t \in [t_0, T]$ ,*

$$0 \leq \xi(t) \leq K \int_{t_0}^T \xi(\tau) d\tau + C. \quad (9)$$

*Then, for every  $t \in [t_0, T]$ ,*

$$\xi(t) \leq C e^{K(t-t_0)}. \quad (10)$$

**Proof.** Please see [56], p. 713. ■

We are now in a position to present Lemma 2.1.2. This Lemma will be used when deriving the gradient formula and in proving that the gradient is Lipschitz continuous in  $\bar{\tau}$ .

---

<sup>1</sup>The norm in the left-hand side of the inequality is the induced matrix norm.

**Lemma 2.1.2** *There exist constants  $K > 0$  and  $\bar{K} > 0$ , depending only on  $C$ ,  $K_1$ ,  $K_2$ , and  $\Gamma$ , such that, for all  $h_1 \in H[C; K_1; K_2; \Gamma]$  and  $h_2 \in H[C; K_1; K_2; \Gamma]$  with the property that  $x_1(t) \in \Gamma$  and  $x_2(t) \in \Gamma$  for all  $t \in [0, T]$ , the following two inequalities are in effect:*

$$\|\Delta x(t)\| \leq K \int_0^T \|\Delta h(x_1(t), t)\| dt, \quad (11)$$

and

$$\begin{aligned} & \left\| \Delta x(t) - \int_0^t \Phi(t, \tau) \Delta h(x_1(\tau), \tau) d\tau \right\| \leq \\ & \leq \bar{K} \left( \int_0^T \|\Delta h(x_1(t), t)\| dt \right) \left( \int_0^T \|\Delta h(x_1(t), t)\| dt + \int_0^T \left\| \frac{\partial \Delta h}{\partial x}(x_1(t), t) \right\| dt \right). \end{aligned} \quad (12)$$

**Proof.** See Appendix A.

As an application of this lemma, consider a family of functions,  $h_\lambda \in H(C; K_1; K_2; \Gamma)$ , parameterized by  $\lambda \in [0, \bar{\lambda})$  for some  $\bar{\lambda} > 0$ , for given  $C > 0$ ,  $K_1 > 0$ ,  $K_2 > 0$ , and a compact set  $\Gamma \subset \mathbb{R}^n$ . Let  $x_\lambda(t)$  be defined by the differential equation  $\dot{x}_\lambda = h_\lambda(x_\lambda, t)$ ,  $t \in [0, T]$ , with a common initial condition  $x_0 \in \Gamma$ . For the special case where  $\lambda = 0$  we will use the notation  $h(x, t) = h_0(x, t)$  and  $x(t) = x_0(t)$ , and we define  $\Delta h_\lambda(x, t) = h_\lambda(x, t) - h(x, t)$ . Fix  $\tau_0 \in (0, T)$  such that  $\tau_0 + \bar{\lambda} \leq T$ , and let  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$  be a function satisfying Assumption 1. Suppose that  $\Delta h_\lambda(x, t)$  has the following form,

$$\Delta h_\lambda(x, t) = \begin{cases} g(x), & \text{if } \tau_0 \leq t \leq \tau_0 + \lambda \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

Let  $L : \mathbb{R}^n \rightarrow \mathbb{R}$  be a function satisfying Assumption 1(i) (i.e., it is continuously differentiable), and define the function  $J : [0, \bar{\lambda}] \rightarrow \mathbb{R}$  by

$$J(\lambda) = \int_0^T L(x_\lambda(t)) dt. \quad (14)$$

**Proposition 2.1.1** *If  $x_\lambda(t) \in \Gamma$  for every  $t \in [0, T]$  and for all  $\lambda \in [0, \bar{\lambda})$ , then  $J$  has the following right derivative at 0,*

$$\frac{dJ}{d\lambda^+}(0) = p(\tau_0)^T g(x(\tau_0)), \quad (15)$$

where the costate  $p(t)$  satisfies the differential equation

$$\dot{p}(t) = - \left( \frac{\partial h}{\partial x}(x(t), t) \right)^T p(t) - \left( \frac{\partial L}{\partial x}(x(t)) \right)^T, \quad (16)$$

with the boundary condition  $p(T) = 0$ .

The proof of Proposition 2.1.1 uses the mean value theorem and the sensitivity results of Lemma 2.1.2.

**Proof of Proposition 2.1.1.** Fix  $\lambda \in [0, \bar{\lambda})$ , and define  $\Delta J_\lambda = J(\lambda) - J(0)$ . By (14),  $\Delta J_\lambda = \int_0^T (L(x_\lambda(t)) - L(x(t)))dt$ , and by the mean value theorem,

$$\Delta J_\lambda = \int_0^T \frac{\partial L}{\partial x}(x(t) + s(t)\Delta x_\lambda(t))\Delta x_\lambda(t)dt \quad (17)$$

for some  $s(t) \in [0, 1]$ . By Assumption 1, there exists  $K_3 > 0$  such that

$$\left\| \frac{\partial L}{\partial x}(x(t) + s(t)\Delta x_\lambda(t)) - \frac{\partial L}{\partial x}(x(t)) \right\| \leq K_3 \|\Delta x_\lambda(t)\|. \quad (18)$$

Next, by lemma 2.1.2 (Eq. (11)), there exists  $K_4 > 0$  such that, for all  $t \in [0, T]$ ,

$$\|\Delta x_\lambda(t)\| \leq K_4 \int_0^T \|\Delta h_\lambda(x(t), t)\|dt. \quad (19)$$

By the definition of  $\Delta h_\lambda$  (Eq. (13)), there exists  $K_5 > 0$  such that

$$\int_0^T \|\Delta h_\lambda(x(t), t)\|dt \leq K_5 \lambda, \quad (20)$$

and

$$\int_0^T \left\| \frac{\partial \Delta h_\lambda}{\partial x}(x(t), t) \right\|dt \leq K_5 \lambda. \quad (21)$$

Combining (18), (19) and (20) we obtain,  $\left\| \left( \frac{\partial L}{\partial x}(x(t) + s(t)\Delta x_\lambda(t)) - \frac{\partial L}{\partial x}(x(t)) \right) \Delta x_\lambda(t) \right\| \leq K_3 \|\Delta x_\lambda(t)\|^2 \leq K_3 K_4^2 K_5^2 \lambda^2$ . By defining  $K_6 = K_3 K_4^2 K_5^2 T$ , we have that,

$$\int_0^T \left\| \left( \frac{\partial L}{\partial x}(x(t) + s(t)\Delta x_\lambda(t)) - \frac{\partial L}{\partial x}(x(t)) \right) \Delta x_\lambda(t) \right\|dt \leq K_6 \lambda^2. \quad (22)$$

Consequently, and by (17), we have that

$$\Delta J_\lambda = \int_0^T \frac{\partial L}{\partial x}(x(t))\Delta x_\lambda(t)dt + o(\lambda), \quad (23)$$

where  $o(\lambda)/\lambda \rightarrow 0$  as  $\lambda \rightarrow 0$ . Next, applying Lemma 2.1.2 (Eq. (12)) with  $h_1 = h$ ,  $x_1 = x$ ,  $h_2 = h_\lambda$ , and  $x_2 = x_\lambda$ , it follows (by (12), (20) and (21)) that  $\Delta x_\lambda(t) - \int_0^t \Phi(t, \tau)\Delta h_\lambda(x(\tau), \tau)d\tau = o(\lambda)$ , where the function  $o(\lambda)$  is independent of  $t \in [0, T]$  or of  $\lambda \in [0, \bar{\lambda})$ . Consequently, and by (23), we have that

$$\Delta J_\lambda = \int_0^T \frac{\partial L}{\partial x}(x(t)) \int_0^t \Phi(t, \tau)\Delta h_\lambda(x(\tau), \tau)d\tau dt + o(\lambda). \quad (24)$$

Changing the order of integration in (24) we obtain,

$$\Delta J_\lambda = \int_0^T \int_\tau^T \frac{\partial L}{\partial x}(x(t)) \Phi(t, \tau) dt \Delta h_\lambda(x(\tau), \tau) d\tau + o(\lambda). \quad (25)$$

Define the costate  $p(\tau) \in \mathbb{R}^n$  by  $p(\tau)^T = \int_\tau^T \frac{\partial L}{\partial x}(x(t)) \Phi(t, \tau) dt$ . Taking derivative, it is apparent that  $\dot{p}(\tau)^T = -p(\tau)^T \frac{\partial h}{\partial x}(x(\tau), \tau) - \frac{\partial L}{\partial x}(x(\tau))$ , and hence (16) is in effect; and also  $p(T)^T = 0$ . It now follows from (25) that  $\Delta J_\lambda = \int_0^T p(\tau)^T \Delta h_\lambda(x(\tau), \tau) d\tau + o(\lambda)$ . Hence, by (13),  $\Delta J_\lambda = \int_{\tau_0}^{\tau_0+\lambda} p(\tau)^T g(x(\tau)) d\tau + o(\lambda)$ . Dividing by  $\lambda$  and taking the limit  $\lambda \rightarrow 0$ , and noting that  $p(\tau)^T g(x(\tau))$  is a continuous function of  $\tau$ , we obtain that  $\frac{dJ}{d\lambda^+}(0) = p(\tau_0)^T g(x(\tau_0))$ . This completes the proof.  $\blacksquare$

We remark that the left derivative has the same formula, as can be seen by repeating the arguments of the proof of Proposition 2.1.1 with minor modifications. The purpose of Proposition 2.1.1 is to provide the basic building block for the gradient derivation formulated in Proposition 2.1.2.

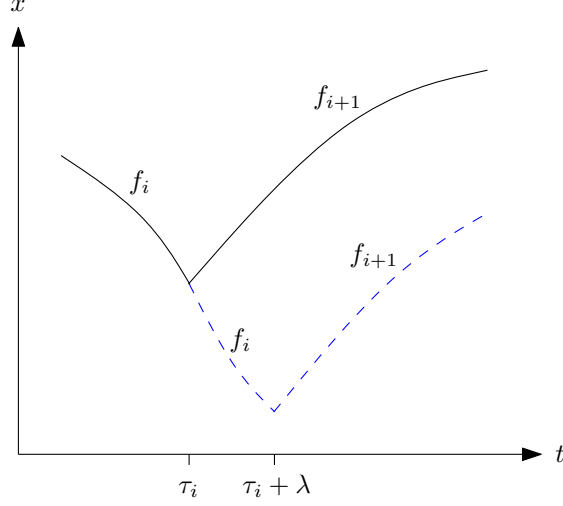
As an illustration of Proposition 2.1.2, Figure 4 is presented. In Figure 4, a system, represented by  $\dot{x}_1 = h(x_1, t)$ , has its  $i^{th}$  switch at time  $\tau_i$  and the  $i^{th}$  switch do not coincide with any other switch. Hence, the system evolves according to  $\dot{x}_1 = f_i(x_1(t))$  right before time  $\tau_i$  and  $\dot{x}_1 = f_{i+1}(x_1(t))$  right after time  $\tau_i$ . The system is then perturbed so that the  $i^{th}$  switch is at time  $\tau_i + \lambda$  for some positive  $\lambda$  satisfying  $\lambda < \tau_{i+1} - \tau_i$ . The perturbed system is represented by  $\dot{x}_2 = h_\lambda(x_2, t)$  and we define  $\Delta h_\lambda(x, t)$  as the difference between the original and the perturbed dynamics, i.e.,  $\Delta h_\lambda(x, t) = h(x, t) - h_\lambda(x, t) = f_i(x) - f_{i+1}(x)$  for  $t \in \tau_i \leq t \leq \tau_i + \lambda$  and zero elsewhere.

Consider the function  $J(\bar{\tau})$  as defined by (8) and (6). Define the feasible set, denoted by  $\Lambda$ , by  $\Lambda := \{\bar{\tau} = (\tau_1, \dots, \tau_N)^T : 0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = T\}$ . For every  $\bar{\tau} \in \Lambda$ , define the costate  $p(t) \in \mathbb{R}^n$  by the differential equation

$$\dot{p} = - \left( \frac{\partial f_{i+1}}{\partial x}(x, t) \right)^T p - \left( \frac{\partial L}{\partial x}(x) \right)^T, \quad t \in [\tau_i, \tau_{i+1}], \quad i = N, N-1, \dots, 0, \quad (26)$$

with the boundary condition  $p(T) = 0$ .

**Proposition 2.1.2** *Suppose that Assumption 1 is in effect. For every point  $\bar{\tau}$  in the interior*



**Figure 4:** State-trajectories of  $x_1$  (solid) and  $x_\lambda$  (dotted) together with the current dynamic response functions.

of  $\Lambda$ , and for all  $i = 1, \dots, N$ , the derivative  $\frac{dJ}{d\tau_i}(\bar{\tau})$  has the following form

$$\frac{dJ}{d\tau_i}(\bar{\tau}) = p(\tau_i)^T (f_i(x(\tau_i)) - f_{i+1}(x(\tau_i))). \quad (27)$$

**Proof.** Define the function  $h(x, t) : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}^n$  by  $h(x, t) = f_i(x)$  for all  $t \in [\tau_{i-1}, \tau_i]$ . Then  $\dot{x} = h(x, t)$  with the initial condition  $x(0) = x_0$ . By Assumption 1, there exists a convex compact set  $\Gamma \subset \mathbb{R}^n$  such that  $x(t) \in \Gamma$  for every feasible  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$  and for all  $t \in [0, T]$ . Moreover, by the same assumption there exist constants  $C > 0$ ,  $K_1 > 0$ , and  $K_2 > 0$ , such that  $h(x, t) \in H[C; K_1; K_2; \Gamma]$  for all  $\bar{\tau}$ . Given  $i \in \{1, \dots, N\}$  and  $\lambda \in [0, \tau_{i+1} - \tau_i]$ , define  $\Delta h_\lambda(x, t)$  as in (13), with  $\tau_i$  instead of  $\tau_0$ , and with  $g(x) = f_i(x) - f_{i+1}(x)$ . An application of Proposition 2.1.1 and the remark that follows it now yields (27). ■

At this point it should be noted that (27) could have been derived through standard variational principles, i.e., through perturbing the switching times of the initial system as described in [68]. However, the benefit of the derivation in this Chapter is that it lends itself to the case when we consider inserting a new modal function. This will be researched in the next Chapter.

We observe that the derivative  $dJ/d\tau_i$  may not be well defined on the boundary of  $\Lambda$ . The reason is that, if  $\tau_{i+1} = \tau_i$ , then changing these variables in a way that swaps their

order leaves unclear the identity of the modal function between them and hence the right-hand side of (27). However, the expression in the right-hand side of (27) is defined on the boundary of  $\Lambda$  where, in the event that  $\tau_{i+1} = \tau_i$ , the domain of the modal function  $f_{i+1}$  is the single point  $\tau_{i+1} = \tau_i$ . Let us define, for every  $\bar{\tau} \in \Lambda$ , by  $q_i(\bar{\tau})$  the right-hand side of (27), and define  $\bar{q}(\bar{\tau}) := (q_1(\bar{\tau}), \dots, q_N(\bar{\tau}))^T \in \mathbb{R}^N$ . Then the function  $\bar{\tau} \rightarrow \bar{q}(\bar{\tau})$  is well defined throughout  $\Lambda$ . Note that  $\bar{q}(\bar{\tau}) = \nabla J(\bar{\tau})$  in the interior of  $\Lambda$ . Furthermore, it is evident that the directional derivative of  $J$  at  $\bar{\tau} \in \Lambda$  in a feasible direction  $h$  is given by the inner product  $\langle \bar{q}(\bar{\tau}), h \rangle$ . This fact will be used in the analysis carried out in the next section.

At this point an expression for the gradient of the cost with respect to the switching times has been presented. In order to utilize this expression to optimize a cost function defined over the switching times, we will present, in the next section, a gradient descent algorithm that takes into account the fact that we are optimizing over a constraint set. Furthermore, an optimality criterion with an intuitive appeal will be derived.

## 2.2 Optimality Condition and an Algorithm

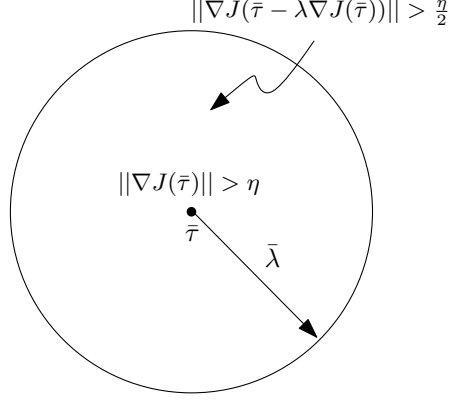
This section derives a special form of the Kuhn-Tucker optimality condition that is based on the structure of the constraint set  $\Lambda$ , and uses it to compute a descent direction. This descent direction will then be used in a gradient descent algorithm in order to optimize over the switching times.

In general, in order for a gradient descent algorithm to converge to a local optimum, it is sufficient that the gradient is continuously differentiable, i.e.,  $\nabla J(\bar{\tau}) \in C^1$ . To see that this is a sufficient condition, assume that  $\|\nabla J(\bar{\tau})\| > \eta$  for some  $\eta > 0$ . Since  $\nabla J(\bar{\tau}) \in C^1$  there exists a positive real number  $\bar{\lambda}(\bar{\tau}) > 0$  such that for all  $\lambda \in [0, \bar{\lambda}(\bar{\tau})]$ ,  $\|\nabla J(\bar{\tau} - \lambda \nabla J(\bar{\tau}))\| > \eta/2$ , as illustrated in Figure 5. If we take a sufficiently small step against the direction of the gradient, the cost is guaranteed to decrease according to the mean value theorem,

$$\begin{aligned} J(\bar{\tau} - \lambda \nabla J(\bar{\tau})) - J(\bar{\tau}) &= \nabla J(\bar{\tau} - \lambda \nabla J(\bar{\tau}) - s(\bar{\tau} - \lambda \nabla J(\bar{\tau}) - \bar{\tau})) \cdot (\bar{\tau} - \lambda \nabla J(\bar{\tau}) - \bar{\tau})^T = \\ &= -\lambda \nabla J(\bar{\tau} - \lambda s \nabla J(\bar{\tau})) \cdot \nabla J(\bar{\tau})^T \end{aligned} \quad (28)$$

for some  $s \in [0, 1]$ . As  $\|\nabla J(\bar{\tau} - \lambda \nabla J(\bar{\tau}))\| > \eta/2$ , it follows that  $\nabla J(\bar{\tau} - \lambda s \nabla J(\bar{\tau})) \cdot \nabla J(\bar{\tau})^T >$





**Figure 5:** Illustration of guaranteed descent using gradient descent algorithm when  $\nabla J(\bar{\tau}) \in C^1$  and  $\lambda \in [0, \bar{\lambda})$ .

$\eta^2/2$  and the right-hand side of (28) is less than  $-\lambda\eta^2/2$  for all  $\lambda \in [0, \bar{\lambda}(\bar{\tau}))$ , hence we are guaranteed a descent in the cost by moving against the direction of the gradient.

In order to continue with our analysis, the following result concerning continuity of  $\nabla J(\bar{\tau})$  is required.

**Proposition 2.2.1** *For every  $\bar{\tau} \in \mathbb{R}^N$  such that  $\bar{\tau} \in \Lambda$ , define  $q_i(\bar{\tau}) := p(\tau_i)^T(f_i(x(\tau_i)) - f_{i+1}(x(\tau_i)))$ ,  $\forall i \in \{1, \dots, N\}$ . Furthermore, let  $\bar{q}(\bar{\tau}) := (q_1(\bar{\tau}), \dots, q_N(\bar{\tau}))^T \in \mathbb{R}^N$ . Then the function  $\bar{q}(\bar{\tau}) : \Lambda \rightarrow \mathbb{R}^N$  is Lipschitz continuous throughout  $\Lambda$ .*

**Proof.** Given  $\bar{\tau} \in \Lambda$ , denote by  $x(t; \bar{\tau})$  and  $p(t; \bar{\tau})$  the state and costate variables defined by (8) and (26), respectively, with the switching time vector  $\bar{\tau}$ . By Assumption 1, there exist compact sets  $\Gamma_x \subset \mathbb{R}^n$  and  $\Gamma_p \subset \mathbb{R}^n$  such that  $x(t, \bar{\tau}) \in \Gamma_x$  and  $p(t; \bar{\tau}) \in \Gamma_p$  for every  $t \in [0, T]$  and for every  $\bar{\tau} \in \Lambda$ . Consider two points  $\bar{\tau}(1) = (\tau_1(1), \dots, \tau_N(1))^T \in \Lambda$  and  $\bar{\tau}(2) = (\tau_1(2), \dots, \tau_N(2))^T \in \Lambda$ . By Lemma 2.1.2 (Eq. (11)) applied first to  $x$  (Eq. (8)) and then to  $p$  (Eq. (26)), there exists a constant  $K_1 > 0$  such that, for every  $\bar{\tau}(1)$  and  $\bar{\tau}(2)$ , and for all  $t \in [0, T]$ ,  $\|x(t; \bar{\tau}(1)) - x(t; \bar{\tau}(2))\| \leq K_1 \|\bar{\tau}(1) - \bar{\tau}(2)\|$ , and  $\|p(t; \bar{\tau}(1)) - p(t; \bar{\tau}(2))\| \leq K_1 \|\bar{\tau}(1) - \bar{\tau}(2)\|$ . Next, by (26) and Assumption 1, there exists  $K_2 > 0$  such that, for every  $\bar{\tau} \in \Lambda$ ,  $\|\dot{p}(t; \bar{\tau})\| \leq K_2$  for every  $t \in [0, T]$ , and hence, for every  $t_1 \in [0, T]$  and  $t_2 \in [0, T]$ ,

$\|p(t_1; \bar{\tau}) - p(t_2; \bar{\tau})\| \leq K_2 |t_1 - t_2|$ . Consequently, for every  $i = 1, \dots, N$ , we have that,

$$\begin{aligned}
& \|p(\tau_i(1), \bar{\tau}(1)) - p(\tau_i(2), \bar{\tau}(2))\| \\
& \leq \|p(\tau_i(1), \bar{\tau}(1)) - p(\tau_i(1), \bar{\tau}(2))\| + \|p(\tau_i(1), \bar{\tau}(2)) - p(\tau_i(2), \bar{\tau}(2))\| \\
& \leq K_1 \|\bar{\tau}(1) - \bar{\tau}(2)\| + K_2 \|\tau_i(1) - \tau_i(2)\| \leq (K_1 + K_2) \|\bar{\tau}(1) - \bar{\tau}(2)\|.
\end{aligned} \tag{29}$$

This establishes that the mapping  $\tau_i \rightarrow p(\tau_i)$  is Lipschitz continuous in  $\tau_i$ . A similar (and actually, simpler) argument applies to the Lipschitz continuity of the function  $\tau_i \rightarrow x(\tau_i)$ . Consequently, and by (27),  $q_i(\bar{\tau}) : \Lambda \rightarrow \mathbb{R}^N$  is a Lipschitz-continuous function. This completes the proof.  $\blacksquare$

Note that the above proof relied on the fact that the product of two bounded Lipschitz continuous functions is Lipschitz continuous, as shown below. Assume  $f(x)$  and  $g(x)$  are two bounded Lipschitz continuous functions such that  $g, h : \mathbb{R}^N \rightarrow \mathbb{R}^k$ . Assume without loss of generality that they both have the same Lipschitz constant,  $K$  and that they both are bounded by  $M > 0$ . Then

$$\begin{aligned}
& \|f(x + \Delta x)^T g(y + \Delta y) - f(x)^T g(y)\| = \\
& \|f(x + \Delta x)^T [g(y + \Delta y) - g(y)] + f(x + \Delta x)^T g(y) - f(x)^T g(y)\| \leq \\
& \|f(x + \Delta x)\| \cdot \|g(y + \Delta y) - g(y)\| + \|g(y)\| \cdot \|f(x + \Delta x) - f(x)\| \leq \\
& MK \|\Delta y + \Delta x\|,
\end{aligned} \tag{30}$$

and the product between  $f, g$  is Lipschitz continuous with Lipschitz constant  $MK$ .

We next derive a special form of the Kuhn-Tucker optimality condition. Fix a point  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T \in \Lambda$ , and recall that we defined  $\tau_0 := 0$  and  $\tau_{N+1} = T$ . If  $\bar{\tau}$  is on the boundary of  $\Lambda$  then  $\tau_i = \tau_{i+1}$  for some  $i = 0, \dots, N$ . To account for this case we define, for all  $i \in \{0, \dots, N+1\}$ , the integer-quantities  $k(i)$  and  $n(i)$  as follows:  $k(i) := \min\{k \leq i : \tau_k = \tau_i\}$ , and  $n(i) := \max\{n \geq i : \tau_n = \tau_i\}$ . In other words,  $\tau_j = \tau_i$  for all  $j \in \{k(i), \dots, n(i)\}$ ; if  $\tau_i > 0$  then  $\tau_{k(i)-1} < \tau_{k(i)}$ ; and if  $\tau_i < T$  then  $\tau_{n(i)} < \tau_{n(i)+1}$ . Furthermore, define  $r_i(\bar{\tau}) := \sum_{j=k(i)}^i q_j(\bar{\tau})$  and  $R_i(\bar{\tau}) := \sum_{j=i}^{n(i)} q_j(\bar{\tau})$ . The following result characterizes Kuhn-Tucker points.

**Proposition 2.2.2** *Let  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$  be a local minimum for  $P_\sigma$ . Then, for every  $i \in \{1, \dots, N\}$ ,  $r_i(\bar{\tau}) \leq 0$  unless  $\tau_i = 0$ , and  $R_i(\bar{\tau}) \geq 0$  unless  $\tau_i = T$ .*

**Proof.** Let  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$  be a local minimum for  $P_\sigma$ . Consider  $k \in \{1, \dots, N\}$  and  $n \in \{k, \dots, N\}$  such that: (i)  $\tau_k = \tau_n$ ; (ii) either  $\tau_k = 0$  or  $\tau_{k-1} < \tau_k$ ; and (iii)  $\tau_n < \tau_{n+1}$ . We will prove that  $R_i(\bar{\tau}) \geq 0$  for all  $i = k, \dots, n$ ; since similar arguments apply to proving the reverse inequality regarding  $r_i$ , this will complete the proposition's proof.

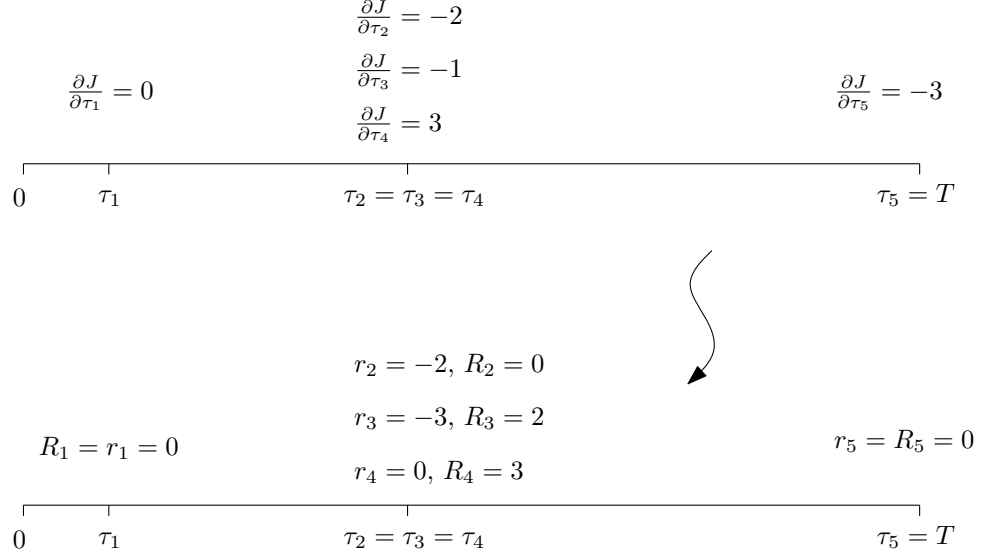
If  $k = n$  then certainly  $q_k(\bar{\tau}) = 0$  if  $\tau_k > 0$  and  $q_k(\bar{\tau}) \geq 0$  if  $\tau_k = 0$ , and hence  $R_k(\bar{\tau}) = q_k(\bar{\tau}) \geq 0$  in either case. Next, consider the case where  $k < n$ . For all  $j = k, \dots, n-1$ , since  $\tau_j = \tau_{j+1}$ , there exists a Lagrange multiplier  $\lambda_j \geq 0$  for the constraint  $\tau_j - \tau_{j+1} \leq 0$ . Moreover, if  $\tau_k = 0$  then there exists a Lagrange multiplier  $\mu_k \geq 0$  for the constraint  $-\tau_k \leq 0$ . From the Kuhn-Tucker optimality condition, it follows that (i)  $q_k(\bar{\tau}) + \lambda_k = 0$  if  $\tau_k > 0$ , and  $q_k(\bar{\tau}) + \lambda_k - \mu_k = 0$  if  $\tau_k = 0$ ; (ii)  $q_j(\bar{\tau}) - \lambda_{j-1} + \lambda_j = 0$  for all  $j = k+1, \dots, n-1$ ; and (iii)  $q_n(\bar{\tau}) - \lambda_{n-1} = 0$ . Fix  $i \in \{k, \dots, n\}$ . Summing up these equations for  $j = i, \dots, n$ , we obtain: (i) for  $i > k$ ,  $R_i(\bar{\tau}) = \lambda_{i-1}$ ; and (ii) for  $i = k$ ,  $R_k(\bar{\tau}) = 0$  if  $\tau_k > 0$ , and  $R_k(\bar{\tau}) = \mu_k$  if  $\tau_k = 0$ . In any event,  $R_i(\bar{\tau}) \geq 0$ . ■

**Corollary 2.2.1** *In the setting of Proposition 2.2.2, if  $\tau_{k(i)-1} < \tau_{k(i)}$  and  $\tau_{n(i)} < \tau_{n(i)+1}$ , then  $R_{k(i)}(\bar{\tau}) = r_{n(i)}(\bar{\tau}) = 0$ .*

**Proof.** Follows immediately from Proposition 2.2.2, since  $R_{k(i)}(\bar{\tau}) = r_{n(i)}(\bar{\tau})$ . ■

Proposition 2.2.2, and the corollary that follows, is illustrated through the example in Figure 6. In Figure 6, five switching times and their respective derivatives are depicted. To see if the switching times are optimal,  $r_1$  through  $r_5$ , and  $R_1$  through  $R_5$ , are calculated with the result that  $r_1 = R_1 = \frac{\partial J}{\partial \tau_1} = 0$ , hence  $\tau_1$  is an optimal switching time. Furthermore,  $r_2 = -2$ ,  $r_3 = -3$ ,  $r_4 = 0$  and  $R_2 = 0$ ,  $R_3 = 2$ ,  $R_4 = 3$ , hence the block of switching times at time  $\tau_2$  is optimal since  $r_i \leq 0$  and  $R_i \geq 0$ ,  $\forall i \in \{2, 3, 4\}$ . Finally,  $\tau_5 = T$  and  $r_5 \leq 0$  implies that  $\tau_5$  is optimal as well. Hence, the setting in Figure 6 corresponds to an optimal set of switching times.

In order to solve the problem  $P_\sigma$ , to the extent of computing a point satisfying the above optimality condition, a gradient-projection algorithm with Armijo step sizes is used. Given



**Figure 6:** Calculation of  $r$  and  $R$  for an example with five switches.

a point  $\bar{\tau} \in \Lambda$ , let  $\Psi(\bar{\tau})$  denote the set of feasible directions from the point  $\bar{\tau}$ , namely,

$$\Psi(\bar{\tau}) := \{\bar{h} \in \mathbb{R}^N \mid \text{for some } \tilde{\zeta} > 0, \text{ and for all } \zeta \in [0, \tilde{\zeta}), \quad \bar{\tau} + \zeta \bar{h} \in \Lambda\}.$$

Let  $\bar{h}(\bar{\tau})$  denote the projection of the vector  $-\bar{q}(\bar{\tau})$  onto  $\Psi(\bar{\tau})$ . The following algorithm is used in order to find an optimal switching time vector through using the Armijo step sizes in the direction of  $\bar{h}(\bar{\tau})$ .

**Algorithm 2.2.1** *Gradient-Projection Algorithm with Armijo Step Sizes.*

*Given:* Constants  $\alpha \in (0, 1)$ ,  $\beta \in (0, 1)$ , and  $\bar{z} > 0$ .

*Initialize.* Choose an initial point  $\bar{\tau}_0 \in \Lambda$ . Set  $i = 0$ .

*Step 1.* If  $\bar{\tau}_i + \bar{z}\bar{h}(\bar{\tau}_i) \notin \Lambda$  then compute  $z_{\max} := \max\{z \geq 0 \mid \bar{\tau}_i + z\bar{h}(\bar{\tau}_i) \in \Lambda\}$ ; otherwise set  $z_{\max} := \bar{z}$ .

*Step 2.* Compute the step size  $\zeta_i$  by

$$\zeta_i = \max\{z = z_{\max} \cdot \beta^k; k \geq 0 \mid J(\bar{\tau}_i + z\bar{h}(\bar{\tau}_i)) - J(\bar{\tau}_i) \leq \alpha z < \bar{h}(\bar{\tau}_i), \bar{q}(\bar{\tau}_i) >\}. \quad (31)$$

*Step 3.* Set  $\bar{\tau}_{i+1} := \bar{\tau}_i + \zeta_i \bar{h}(\bar{\tau}_i)$ , set  $i = i + 1$ , and go to Step 1. ■

Note that, if  $\bar{\tau}_i + \bar{z}\bar{h}(\bar{\tau}_i) \in \Lambda$  then  $z_{\max} = \bar{z}$ , and if  $\bar{\tau}_i + \bar{z}\bar{h}(\bar{\tau}_i) \notin \Lambda$  then  $z_{\max}$  is the maximum step size  $z$  for which  $\bar{\tau}_i + z\bar{h}(\bar{\tau}_i) \in \Lambda$ . Moreover, the step size computed in

Step 2 is  $\zeta_i := z_{\max} \cdot \beta^k$  for some integer  $k \geq 0$ . Ref. [56] contains an analysis of this algorithm and various alternative versions thereof. In particular, it proves that (i)  $\bar{h}(\bar{\tau}_i)$  indeed is a descent direction from  $\bar{\tau}_i$ , i.e.,  $J(\bar{\tau}_{i+1}) \leq J(\bar{\tau}_i)$ ; (ii) the step size  $\zeta_i$  is nonzero as long as  $\bar{\tau}_i$  does not satisfy the Kuhn-Tucker optimality condition (and hence the optimality condition established in Proposition 2.2.2), and (iii) every accumulation point of a sequence  $\{\bar{\tau}_i\}_{i=0}^\infty$ , computed by the algorithm, satisfies the optimality condition. Therefore, a practical stopping rule is to end the algorithm's run at a point  $\bar{\tau}_i$  whenever  $\|\bar{h}(\bar{\tau}_i)\| < \epsilon$  for an a-priori chosen value of  $\epsilon > 0$ . Moreover, the algorithm is stable in the sense that it will converge from every starting point, and it has a linear asymptotic convergence rate. Ref. [56, pp. 30-31] also gives recommendations for the choices of  $\alpha$ ,  $\beta$  and  $\bar{z}$ .

The reason for calculating  $\bar{z}$  and setting  $z_{\max} := \max\{z \geq 0 \mid \bar{\tau}_i + z\bar{h}(\bar{\tau}_i) \in \Lambda\}$  if  $\bar{\tau}_i + \bar{z}\bar{h}(\bar{\tau}_i) \notin \Lambda$ , is to make sure the algorithm does not converge to accumulation points that do not satisfy the optimality criteria.

To see how this could happen, assume the step size is chosen according to

$$\zeta_i = \max\{z = \beta^k; k \geq 0; \bar{\tau}_i + z\bar{h}(\bar{\tau}_i) \in \Lambda \mid J(\bar{\tau}_i + z\bar{h}(\bar{\tau}_i)) - J(\bar{\tau}_i) \leq \alpha z < \bar{h}(\bar{\tau}_i), \bar{q}(\bar{\tau}_i) >\}. \quad (32)$$

Furthermore, assume that we have two switches  $\tau_i$  and  $\tau_{i+1}$  such that  $\frac{dJ}{d\tau_i} < 0$  and  $\frac{dJ}{d\tau_{i+1}} > 0$ . Then, since the step-size can only attain certain values  $\beta^k$  for all natural numbers  $k$ , then a situation could occur where the algorithm converges to a point where  $\tau_i = \tau_{i+1}$  even though this point might not be optimal.

Finally, a word must be said about the computation of  $\bar{h}(\bar{\tau})$  for a given  $\bar{\tau} := (\tau_1, \dots, \tau_N)^T \in \Lambda$ . Let us define a *block* to be a contiguous integer-set  $\{k, \dots, n\} \subset \{1, \dots, N\}$  such that  $\tau_n = \tau_k$  (and hence  $\tau_i = \tau_k$  for all  $i \in \{k, \dots, n\}$ ). Observe that every set of contiguous integers that is a subset of a block is also a block. Furthermore, we say that a block is *maximal* if no superset thereof is a block. Obviously, the set  $\{1, \dots, N\}$  is partitioned into disjoint maximal blocks in a way that depends on  $\bar{\tau}$ .

The following computation of  $\bar{h}(\bar{\tau}) := (h_1(\bar{\tau}), \dots, h_N(\bar{\tau}))^T$  is done one-block-at-a-time in the following manner. Let  $\{k, \dots, n\}$  be a maximal block associated with  $\bar{\tau}$ .

**Algorithm 2.2.2** *Procedure for computing  $h_i(\bar{\tau})$ ,  $i = k, \dots, n$ .*

*Step 0.* Set  $\ell = k$ .

*Step 1.* Compute  $r_{max}$  defined by

$$r_{max} := \max \left\{ \frac{r_i(\bar{\tau})}{i - k + 1} \mid i = \ell, \dots, n \right\}. \quad (33)$$

Define  $m := \max\{i = k, \dots, n : \frac{r_i}{i - k + 1} = r_{max}\}$ .

*Step 2.* For all  $i \in \{\ell, \dots, m\}$ , define  $h_i(\bar{\tau})$  by  $-r_{max}$  unless either (i)  $\tau_m = 0$  and  $r_{max} > 0$ , or (ii)  $\tau_m = T$  and  $r_{max} < 0$ . In either case (i) or (ii), set  $h_i(\bar{\tau}) = 0$ .

*Step 3.* If  $m = n$ , exit. If  $m < n$ , set  $\ell := m + 1$  and go to Step 1. ■

Proving that the resulting vector  $\bar{h}(\bar{\tau})$  indeed is the projection of  $-\bar{q}(\bar{\tau})$  onto  $\Psi(\bar{\tau})$  is the topic of Proposition 2.2.3 that follows. In order to prove proposition 2.2.3, we define the quantity

$$r_{\ell,i} = \frac{1}{i - \ell + 1} \sum_{j=\ell}^i \frac{dJ(\bar{\tau})}{d\tau_j}, \quad (34)$$

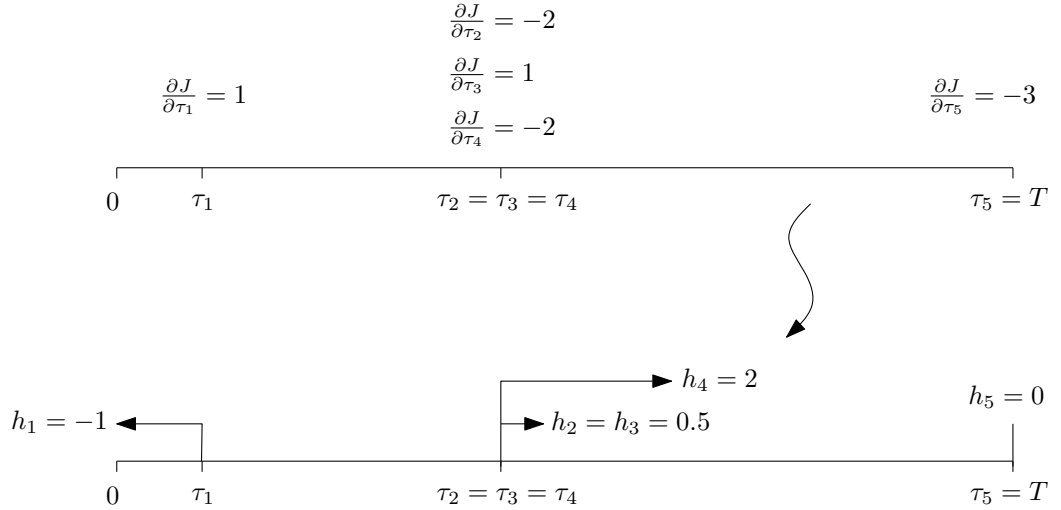
for every  $\ell \in \{k, \dots, n\}$ , and for every  $i \in \{\ell, \dots, n\}$ .

**Proposition 2.2.3** *The vector  $\bar{h}$  computed by Algorithm 2.2.2 for every maximal block is indeed the vector  $\bar{h}(\bar{\tau})$ , namely the projection of  $-\bar{q}(\bar{\tau})$  onto the feasible set  $\Psi(\bar{\tau})$ .*

**Proof.** See Appendix A.

It should be pointed out that the optimality condition established in Proposition 2.2.2 has the following associated intuitive geometric appeal. If the optimality condition is satisfied, then obviously  $\bar{h}(\bar{\tau}) = 0$ . If it is not satisfied, then Algorithm 2.2.2 indicates which variables  $\tau_i$ ,  $i \in \{k, \dots, n\}$ , should be increased and which ones should be decreased; in other words, a descent direction for  $J$  clearly emerges. In order to illustrate the computation on  $\bar{h}(\bar{\tau})$ , a simple example is presented in Figure 7. In Figure 7, there are three blocks of switching times. For the first block, Algorithm 2.2.2 gives that  $r_{max} = r_1 = -1$ , hence  $h_1(\bar{\tau}) = 1$ . For the second block, consisting of switching times  $\tau_2$  through  $\tau_4$ , we have that

$$\begin{cases} r_2 = -2, & \ell = 2, & r_{max} = \max\{\frac{-2}{1}, \frac{-1}{2}, \frac{-3}{3}\} = \frac{-1}{2} \Rightarrow m = 3 \Rightarrow h_2 = h_3 = 0.5 \\ r_3 = -\frac{1}{2}, \\ r_4 = -2, & \ell = 4, & r_{max} = -2 \Rightarrow m = 4 \Rightarrow h_4 = 2. \end{cases} \quad (35)$$



**Figure 7:** Calculation of descent direction  $\bar{h}(\bar{\tau})$ , for an example with three blocks of switches.

Since  $m = 3$  in the calculations in the first row of (35),  $h_2 = h_3$  and they are both equal to  $-r_{max} = 0.5$ . For  $h_4$  we have that  $r_{max} = -2$  hence,  $h_4 = 2$ . Finally, for  $h_5$  since  $r_{max} = -3 < 0$  and  $\tau_5 = T$ ,  $h_5 = 0$ .

This section is closed by an extension to Proposition 2.1.1 concerning the case when we consider altering the mode sequence by inserting a new function into the current sequence of functions, as described in Chapter III.

Fix a modal sequence  $\sigma = \{\alpha(i)\}_{i=1}^{N+1}$  and the associated switching time vector  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$ . Recall that the state trajectory  $\{x(t)\}$  evolves according to (5), and by defining  $f_i = g_{\alpha(i)}$ , (5) is transformed into (8). Let  $\{p(t)\}$  be the costate trajectory defined by (26). Now fix  $\alpha \in A$ ,  $\tau \in (0, T)$ , and  $\lambda > 0$  such that  $\tau + \lambda < T$ , and consider inserting the modal function  $g_\alpha$  in the time-interval  $[\tau, \tau + \lambda]$ . This insertion will result in a modification of the modal sequence  $\sigma$  by adding to it the index  $\alpha$ . Recall the cost functional  $J$  as defined by (6), and consider it as a function of  $\lambda$ , hence to be denoted by  $J(\lambda)$ . Then the following is an immediate corollary of Proposition 2.1.1.

**Proposition 2.2.4** *Let  $\tau \in [\tau_{i-1}, \tau_i)$  for some  $i \in \{1, \dots, N+1\}$ . Then, the one-sided derivative  $\frac{dJ_{g_\alpha, \tau}}{d\lambda^+}(0)$  has the following form,*

$$\frac{dJ_{g_\alpha, \tau}}{d\lambda^+}(0) = p(\tau)^T (g_\alpha(x(\tau)) - g_{\alpha(i)}(x(\tau))). \quad (36)$$

**Proof.** Follows directly from Proposition 2.1.1. ■

If the above insertion takes place at a point  $\tau \in (\tau_{i-1}, \tau_i)$  then, for  $\lambda < \tau_i - \tau$ , the switching time vector becomes  $(\tau_1, \dots, \tau_{i-1}, \tau, \tau + \lambda, \tau_i, \dots, \tau_N)^T \in \mathbb{R}^{N+2}$ , and the associated, modified modal sequence becomes  $\{\alpha(1), \dots, \alpha(i), \alpha, \alpha(i), \alpha(i+1), \dots, \alpha(N+1)\}$ . We point out that when the above term has to be computed for a number of insertion points  $\tau$ , the costate trajectory need be computed only once. Proposition 2.2.4 and the sensitivity formula (36) will be used in the next chapter for computing insertion points.

### 2.3 Numerical Example

To illustrate the viability of Algorithm 2.2.1, and our gradient formula, we consider optimizing the switching times of a linear system. The system in question switches between three different modes of operation, denoted *mode 1*, *mode 2*, and *mode 3*. Each mode  $i$  has the dynamic representation given by  $\dot{x} = A_i x$  for  $i \in \{1, 2, 3\}$ , where  $x \in \mathbb{R}^2$  and  $A_1, A_2$ , and  $A_3$  are given by

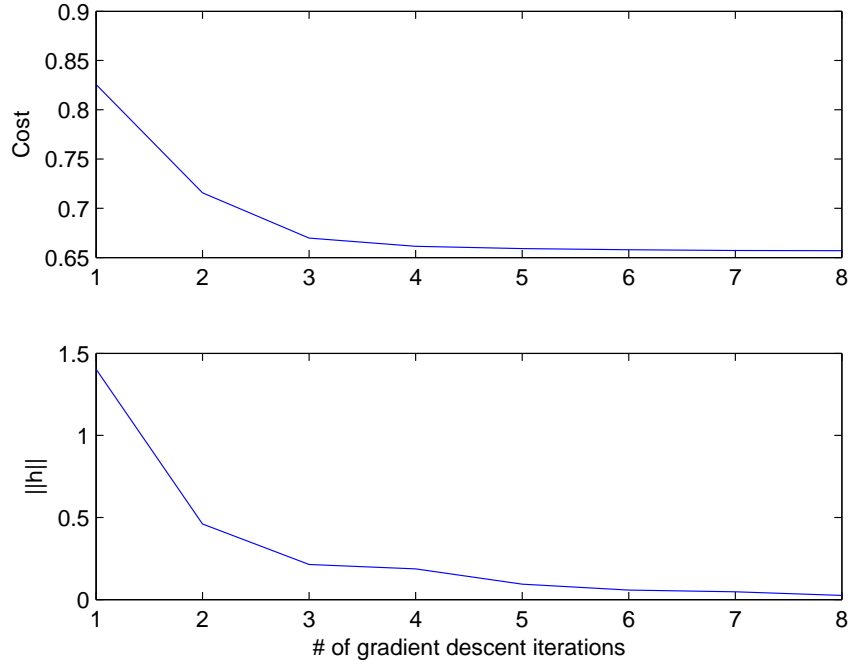
$$A_1 = \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix}, \quad \text{and} \quad A_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

We observe that both of the  $A_1$  and  $A_2$  matrices have one positive eigenvalue and one negative eigenvalue, and the respective eigenvectors of the negative eigenvalues do not coincide. As for  $A_3$ , we observe that it has two positive eigenvalues. Consequently, the switching is used to manage the unstable parts of the state trajectories and we do not expect mode 3 to be active. In fact, given the “energy” cost functional, we expect an optimal switching vector to switch frequently between mode 1 and mode 2 in order to minimize the norm of the state-trajectory.

According to our notation, we let  $A := \{1, 2, 3\}$  be the index set of our modes, and we let  $g_1(x) = A_1 x$ ,  $g_2(x) = A_2 x$ , and  $g_3(x) = A_3 x$ . The time interval we are optimizing over is  $[0, T]$  with  $T = 1$ , and the initial condition is  $x_0 = x(0) = (0.5, 0.5)^T$ . The cost criterion (functional) is given by

$$J = \frac{1}{2} \int_0^1 \|x(t)\|^2 dt.$$





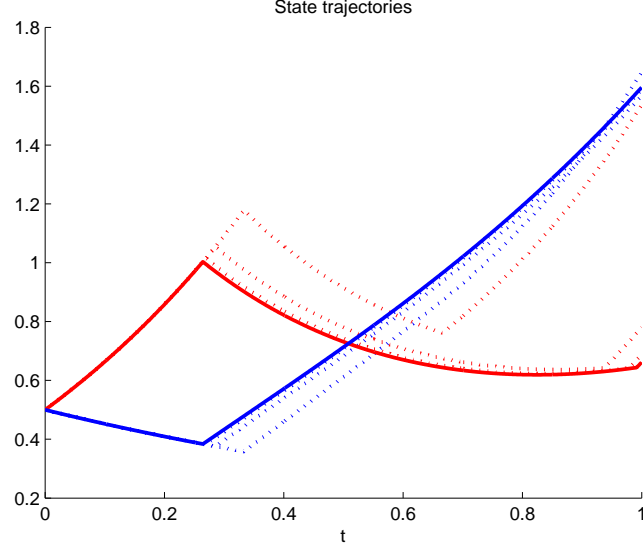
**Figure 8:** Top figure:  $J(\bar{\tau})$  is plotted as a function of the number of iterations of Algorithm 2.2.1. Bottom figure: The norm of the projected gradient is plotted as a function of the number of iterations of Algorithm 2.2.1.

We initialize the switching vector to be  $\bar{\tau} = (0, 0.33, 0.66, 1)$  with mode 1 active between time  $t_0$  and  $\tau_1$ , mode 2 active between time  $\tau_1$  and  $t_2$ , and mode 3 active between time  $\tau_2$  and  $t_f$ . The results of running Algorithm 2.2.1 with  $\alpha = \beta = 0.5$  and  $\bar{z} = 1$  is shown in Figures 8 and 9. As can be seen from Figure 9,  $\tau_2 \rightarrow t_f$  which implies that mode 3 is not used when the algorithm terminates. Furthermore, it is clear from Figure 8 that the cost is reduced. We choose to terminate the algorithm when  $\|h\| < 0.01$ . At that point  $\bar{\tau} = (0, 0.26, 1, 1)$ .

It will be seen how the cost can be reduced further by inserting new switches into the current mode structure in Chapter III.

## 2.4 Conclusions

This Chapter concerned an optimal control problem defined on switched mode dynamical systems, whose variational parameter consists of the switching times. It first derived a



**Figure 9:** Plot of  $x(t)$  for different iterations of Algorithm 2.2.1.  $x(t)$  is plotted with a dotted curve for every third iteration in Algorithm 2.2.1, and the final state-trajectory is plotted using a solid curve.  $x_1(t)$  is plotted in blue and is the curve that starts by decreasing,  $x_2(t)$  is plotted in red and starts by increasing.

formula for the gradient of the cost functional with respect to a given sequence of switching times. This formula was then utilized in a gradient descent algorithm. A necessary optimality criterion was presented, and it was concluded that the algorithm presented in order to find an optimal switching time vector, Algorithm 2.2.1, always converged to a point satisfying the optimality criteria. The results are derived in a general setting where continuous differentiability, with respect to the state variable, of the instantaneous cost function and the modal functions was assumed.

In conclusion, under the assumptions mentioned above, the results presented in this Chapter, in particular Algorithm 2.2.1, solve the problem of finding optimal switching times for a given sequence of modes. Most of the results presented in this Chapter can be found in [31, 30, 11] coauthored by the author of this work.

In order to illustrate the utilization of the different algorithms presented, and to testify to the potential viability of the proposed approach, a numerical example was presented.

The next Chapter extends the results presented in this Chapter to the case when we consider optimizing over the sequence of modes as well.

## CHAPTER III

### OPTIMAL MODE SCHEDULING

This chapter extends the results presented in the previous chapter, concerning the optimization of switched mode dynamical systems, to cover the case when we are optimizing over the mode sequence as well. In particular, we consider a dynamic system that switches among various modes during the course of its operation. Given a cost functional defined on the state trajectory of the system, the problem addressed in this chapter is how to schedule the modes in order to minimize the cost functional. For this problem, the scheduling variable is comprised of two parameters: one discrete and the other continuous. We refer to these parameters as the *sequencing variable* and the *timing variable*, respectively. The discrete parameter consists of the sequence of modes, while the continuous parameter consists of the amount of time each mode is in effect. The optimization of the continuous parameter, for the case when the mode sequence was fixed, was dealt with extensively in the previous chapter and except for a few remarks, in order to make the current chapter self-contained, it will not be discussed in this chapter. Having said that, it should be noted that the optimization of the sequencing and timing variables will not be decoupled.

Several results have been presented regarding this problem (please see Chapter I, Section 1.2, for a list of papers) and most of the results presented in this chapter have been published, or submitted, in the following papers [30, 12, 9, 10] coauthored by the author of this work. In particular, having the scheduling optimization problem in mind, [30] and later [10], illustrated a way to improve on a given mode sequence by inserting a new mode over a brief period of time in such a way that the cost functional is reduced. This has motivated the development of an algorithm that alternates between two phases: in the first phase it solves the timing-optimization problem for a given mode sequence, and in the second phase it modifies the sequence by inserting to it a new mode at a suitable time. Since the number of modes, and therefore the number of switching times, increases each time a new mode

is inserted, the concept of convergence of the algorithm is not clearly defined. Defining convergence is one of the main contributions of this Chapter and it will be proven that the algorithm presented converges to a local minimum in a certain sense.

The rest of the Chapter is organized as follows. Section 3.1 establishes an abstract setting for the optimization problem and defines a suitable concept of algorithms' convergence. Section 3.2 formulates the optimal mode scheduling problem and recalls some preliminary results. Section 3.3 presents a convergent algorithm, and Section 3.4 contains two examples. Finally, Section 3.5 concludes the Chapter.

### 3.1 *Conceptual Framework for Algorithms' Convergence*

Consider a problem of minimizing a function  $J : \Xi \rightarrow \mathbb{R}$ , where  $\Xi$  is a set. Let  $\Gamma$  denote an optimality condition, and let  $\Sigma \subset \Xi$  denote the set of points  $\xi \in \Xi$  where  $\Gamma$  is satisfied. Let  $\theta : \Xi \rightarrow \mathbb{R}^-$  be a nonpositive-valued function such that  $\theta(\xi) = 0$  if and only if  $\xi \in \Sigma$ .<sup>1</sup> Such a function is called *optimality function associated with  $\Gamma$*  (see [56], p.19). Typically  $\theta$  is defined so as to provide a quantitative measure of the extent to which a point  $\xi \in \Xi$  satisfies the condition  $\Gamma$ .

As a simple example of an optimality function, consider minimizing the function  $f(x) = x^2$  over  $\mathbb{R}$ . Then  $\theta = -|2x|$  could act as an optimality function and  $f(x)$  would be optimized if and only if  $\theta = 0$ . Likewise, in the previous chapter,  $-||\bar{h}(\bar{\tau})||$  could act as an optimality function.

The optimality function's upper semicontinuity in the case where  $\Xi$  is a topological space ensures that if  $\hat{\xi}$  is an accumulation point of a sequence  $\{\xi_j\}_{j=1}^\infty$  computed by an algorithm, and if  $\lim_{j \rightarrow \infty} \theta(\xi_j) = 0$ , then  $\theta(\hat{\xi}) = 0$  and hence  $\hat{\xi} \in \Sigma$ .

Ref. [56] has used the concept of optimality functions to develop a unified approach to analysis and design of optimization algorithms, providing simple proofs of their asymptotic convergence. For the standard setting of nonlinear programming, where the parameter space is an Euclidean space, an algorithm's convergence typically means that every accumulation point of a sequence of iteration points, computed by the algorithm, satisfies an (usually

---

<sup>1</sup>If  $\Xi$  is a topological space then  $\theta$  has to be upper-semicontinuous as well.

necessary) optimality condition. Thus, every bounded sequence of iteration points would yield at least one point satisfying the optimality condition. On the other hand, if the parameter space is infinite-dimensional, then bounded sequences of iteration points might not have accumulation points, and hence a different notion of convergence is needed. Such a notion was proposed in [57], and it requires that  $\limsup_{j \rightarrow \infty} \theta(\xi_j) = 0$ , where  $\{\xi_j\}_{j=1}^\infty$  is a sequence of iteration points computed by the algorithm.

The optimization problem confronting us is different. Its parameter set does not consist of a single infinite-dimensional space, but rather of an infinite union of certain monotone-increasing sets. Specifically, let  $\Phi$  be a given finite set; for every  $N = 0, 1, 2, \dots$ , let  $\Xi_N$  be a subset of  $\Phi \times \Phi \times \dots \times \Phi$  (the product  $N + 1$  times), and let  $\Xi := \cup_{N=1}^\infty \Xi_N$ .<sup>2</sup> Then  $\Xi$  is the parameter set of our optimization problem. Let  $\Gamma_N$  be a suitable optimality condition defined on  $\Xi_N$ , and let  $\theta_N$  be the corresponding optimality function. The algorithm presented in the next section computes a sequence  $\xi_j$ ,  $j = 1, 2, \dots$ , where  $\xi_j \in \Xi_{N(j)}$ , and  $N(j+1) > N(j)$  for all  $j = 1, 2, \dots$ . Its convergence will be characterized by computing either a point  $\xi_j \in \Xi_{N(j)}$  such that  $\theta_{N(j)}(\xi_j) = 0$  (in which case the algorithm stops), or a sequence  $\{\xi_j\}_{j=1}^\infty$  such that  $\lim_{j \rightarrow \infty} \theta_{N(j)}(\xi_j) = 0$ . This characterization of convergence extends the concepts developed in [57] from the setting of an infinite-dimensional parameter space to the setting of the present Chapter. Its justification will be made clear once the algorithm is presented and analyzed.

We will adopt the common approach developed in [56] to proving convergence of descent algorithms. It is based on the principle of *sufficient descent*, defined as follows.

**Definition 3.1.1** *An algorithm defined on  $\Xi$  has sufficient descent with respect to the optimality functions  $\theta_k$ ,  $k = 1, 2, \dots$ , if for every  $\delta > 0$  there exists  $\eta > 0$  such that, for every  $j = 1, 2, \dots$ , and for every iteration point  $\xi_j$  computed by the algorithm, if  $\theta_{N(j)}(\xi_j) < -\delta$ , then*

$$J(\xi_{j+1}) - J(\xi_j) < -\eta, \quad (37)$$

where  $\xi_{j+1}$  is the next iteration point.

---

<sup>2</sup>It will become apparent that there is a natural embedding of  $\Xi_N$  into  $\Xi_{N+1}$ .

Definition 3.1.1 says that if the optimality function is negative, i.e., we are not optimal, then there exists a direction of descent in our parameter space such that the cost will be reduced. It will be seen later on in this chapter that the optimality function employed to characterize local optimality is based on inserting a new mode into the current sequence of modes.

Now convergence can be ensured as follows.

**Proposition 3.1.1** *Suppose that there exists a constant  $D$  such that  $J(\xi) \geq D$  for every  $\xi \in \Xi$ . If a descent algorithm has the property of sufficient descent, then for every infinite sequence  $\{\xi_j\}_{j=1}^{\infty}$  it computes,  $\lim_{j \rightarrow \infty} \theta_{N(j)}(\xi_j) = 0$ .*

*Proof.* Immediate from Definition 3.1.1. ■

Having defined the abstract setting of our optimization problem, a short summary of the results that will be derived in the subsequent sections is presented in order to guide the reader. The function  $\frac{dJ_{f,\tau}}{d\lambda^+}$ , as defined in Chapter II Proposition 2.2.4, will serve as the optimality function when we are optimizing over the sequence of modes. Recall that this function gave a derivative measure to the extent of how the cost would change if the mode  $f$  was inserted at time  $\tau$ . In this setting  $\Phi \times \Phi \times \dots \Phi$  (the product  $N$  times), corresponds to the modal sequence  $\sigma$ , as defined in the next section, and the length of this sequence is increased by one each time a new mode is inserted. A simple algorithm based on inserting new modes by evaluating  $\frac{dJ_{f,\tau}}{d\lambda^+}$  for all modal functions  $f \in \Phi$  and for all times  $\tau \in [0, T]$ , where  $T$  is a given final time, will be presented. It will then be proven that the proposed algorithm has the property of sufficient descent, i.e. it converges in the sense that  $\lim_{j \rightarrow \infty} \theta_{N(j)}(\xi_j) = 0$ . Said in words, convergence of our algorithm implies that if the algorithm has converged, i.e.  $\theta_{N(j)}(\xi_j) = 0$ , then, by inserting a new mode at a time  $\tau$  and then optimize over the resulting switching times we would not be guaranteed a reduction in cost.

### 3.2 Problem Formulation

As in the previous chapter, we consider the following time-varying dynamical system,

$$\dot{x} = F(x, t), \quad (38)$$

where  $x \in \mathbb{R}^n$  is the state,  $[0, T]$  is the time horizon for a given  $T > 0$ , the initial state  $x_0 := x(0)$  is assumed given, and  $F : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}^n$  is a function satisfying assumptions sufficient to ensure a unique, continuous, piecewise-differentiable solution to (38). Let  $L : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}$  be a cost function, and consider the cost functional  $J$ , defined by

$$J = \int_0^T L(x, t) dt. \quad (39)$$

As in the previous chapter,  $F$  is assumed to be in a class of functions having the following form. Let  $\Phi$  be a given, finite set of functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ . Let  $N \geq 0$  be any integer, and corresponding to  $N$ , let  $s := (\tau_1, \dots, \tau_N)^T \in \mathbb{R}^N$  be any vector satisfying the inequalities

$$0 \leq \tau_1 \leq \dots \leq \tau_N \leq T. \quad (40)$$

We define, for convenience,  $\tau_0 := 0$  and  $\tau_{N+1} := T$ . Let  $\{f_{\sigma(i)}, i = 1, \dots, N+1\}$ , be any collection of  $N+1$  functions in  $\Phi$ . Then, the class of functions  $F$  that we consider in (38) has the form

$$F(x, t) = f_{\sigma(i)}(x) \quad \text{for all } t \in [\tau_{i-1}, \tau_i), \quad \text{and for all } i = 1, \dots, N+1. \quad (41)$$

Note that, with this form of  $F$ , (38) assumes the following expression,

$$\dot{x} = f_{\sigma(i)}(x) \quad \text{for all } t \in [\tau_{i-1}, \tau_i), \quad \text{and for all } i = 1, \dots, N+1. \quad (42)$$

To ensure a unique solution to (42) and other properties, we make the following mild assumption (also made in Chapter II).

**Assumption 2** (i). The functions  $f \in \Phi$ , and  $L$ , are continuously differentiable on  $\mathbb{R}^n$ .  
(ii). There exists a constant  $K > 0$  such that, for every  $x \in \mathbb{R}^n$ , and for every  $f \in \Phi$ ,

$$\|f(x)\| \leq K(\|x\| + 1). \quad (43)$$

The system is called a *switched mode hybrid system*, and the various functions  $f \in \Phi$  represent its various modes, and hence are called *modal functions*. We define  $\sigma$  by  $\sigma := \{\sigma(1), \dots, \sigma(N+1)\}$ , and refer to it as the *modal sequence*. The vector  $s = (\tau_1, \dots, \tau_N)^T$  is called the *vector of switching times*, and the pair  $(\sigma, s)$ , denoted by  $\xi$ , is referred to as the *modal schedule*, or the *system's schedule*. Given  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\}$ , it is possible that  $f_{\sigma(i)} = f_{\sigma(k)}$  for  $i \neq k$ , in other words, a modal sequence may have a specific function  $f \in \Phi$  multiple times. Consequently, we can associate a modal sequence  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\}$  with an element in the product-set  $\Phi^{N+1}$ , defined as the set-product of  $\Phi$  with itself  $N+1$  times.

We consider the problem of minimizing  $J$  as a function of the schedule  $\xi = (\sigma, s)$ , where we note that  $N$  is a part of the variable  $\sigma$ . For future reference, we denote this problem by  $P$ . This problem may have fairly general constraints on the modal sequence  $\sigma$ , including, but not limited to the following type that often arises in applications: For every  $f \in \Phi$ , let  $\Psi_f \subset \Phi$  be a set of “permissible” modal functions that are allowed to follow the function  $f$  in any modal sequence. Thus, for every  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\}$ , it is required that  $f_{\sigma(i+1)} \in \Psi_{f_{\sigma(i)}}$  for all  $i = 1, \dots, N$ . Generally, we denote by  $\Psi$  the set of all feasible modal sequences, i.e., we require that  $\sigma \in \Psi$ . For a given  $\sigma \in \Psi$ , the problem of minimizing  $J$  as a function of  $s$  subject to the constraints set forth in (40), is referred to as  $P_\sigma$  and solved in Chapter II. Observe that the constraint in (40) allows for the situation where  $\tau_{i-1} = \tau_i$ . In this case the interval  $[\tau_{i-1}, \tau_i)$  is empty and hence the modal function  $f_{\sigma(i)}$  plays no role in the evolution of the state trajectory via (42). However, we allow for this case since we bring to bear upon the problem  $P_\sigma$  the theory of nonlinear programming which generally requires closed constraint sets.

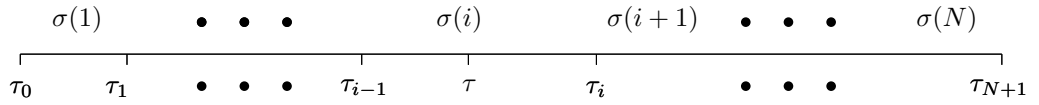
Let us fix a modal sequence  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\}$  for a moment. Let  $s = (\tau_1, \dots, \tau_N)^T$  be a Kuhn-Tucker point for  $P_\sigma$ , and define  $\xi = (\sigma, s)$ . When computed by a Algorithm 2.2.1 in Chapter II,  $s$  is a local minimum for  $P_\sigma$ . However,  $\xi$  may not be a local minimum for  $P$  in the following sense. Consider a time-point  $\tau \in [0, T]$  and a modal function  $f \in \Phi$ , henceforth denoted by  $f = f_{\sigma(*)}$ . Given  $\lambda > 0$ , consider the insertion of the modal function  $f_{\sigma(*)}$  in the interval  $[\tau - \frac{\lambda}{2}, \tau + \frac{\lambda}{2}) \cap [0, T]$ , and denote the corresponding value of  $J$  by



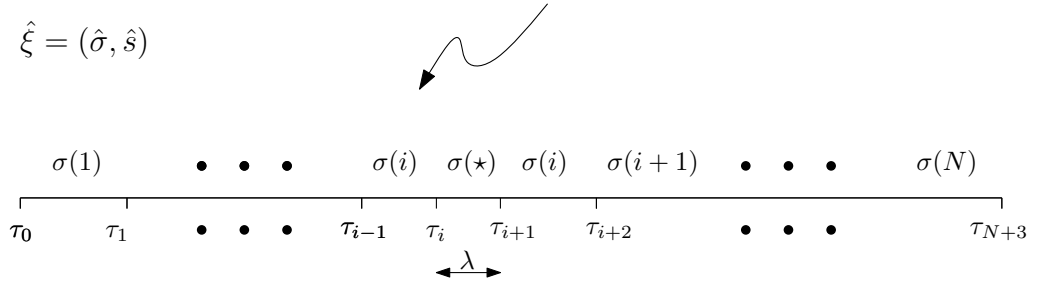
$\hat{J}(\lambda)$ . Let  $D_{f,\tau}(\xi)$  denote the one-sided derivative of  $\hat{J}$  at  $\lambda = 0$ , namely,  $D_{f,\tau}(\xi) = \frac{d\hat{J}(0)}{d\lambda^+}$ . By a direct application of Proposition 2.2.4 in Chapter II, it follows that  $D_{f,\tau}(\xi)$  is well defined for all  $\tau \in [0, T]$ , and it is continuous there in  $\tau$  except at the switching points  $\tau_i$ . If  $D_{f,\tau}(\xi) < 0$  then an insertion of  $f$  in a “small-enough” interval centered at  $\tau$  would result in a reduction in  $J$ .

Such a mode insertion modifies the modal sequence  $\sigma$  and the dimension of the switching time vector  $s$ . For example, consider the case where  $(\tau - \frac{1}{2}\lambda, \tau + \frac{1}{2}\lambda) \subset (\tau_{i-1}, \tau_i)$  for some  $i = 1, \dots, N+1$ . Then the above insertion changes the schedule  $\xi = (\sigma, s)$  to a new schedule,  $\hat{\xi} = (\hat{\sigma}, \hat{s})$ , defined by  $\hat{\sigma} = \{\sigma(1), \dots, \sigma(i), \sigma(\star), \sigma(i), \dots, \sigma(N+1)\}$ , and  $\hat{s} = (\tau_1, \dots, \tau_{i-1}, \tau - \frac{1}{2}\lambda, \tau + \frac{1}{2}\lambda, \tau_i, \dots, \tau_N)^T$ . Note that  $\hat{s} \in \mathbb{R}^{N+2}$ . For an illustration, see Figure 10. Now we will be primarily interested in the case where  $\lambda \rightarrow 0$ , and in this case will refer to  $\hat{\xi}$  as the *schedule obtained by inserting  $f = f_{\sigma(\star)}$  to the schedule  $\xi$  at the time  $\tau$* .

$$\xi = (\sigma, s)$$



$$\hat{\xi} = (\hat{\sigma}, \hat{s})$$



**Figure 10:** Changes in the schedule obtained by inserting  $f_{\sigma(\star)}$  centered at time  $\tau$  for an interval of length  $\lambda$ .

The possibility of inserting new modes to a given schedule allows us to construct an algorithm that appends the sequencing variable  $\sigma$  by first-order sensitivity information. The idea is to solve the problem  $P_\sigma$  for a given  $\sigma$ , and then to insert a new modal function

$f$  at a time  $\tau$  such that  $D_{f,\tau}(\xi) < 0$ . In fact, let us define

$$D(\xi) := \min\{D_{f,\tau}(\xi) \mid f \in \Phi \text{ such that } \hat{\sigma} \in \Psi, \tau \in [0, T]\}, \quad (44)$$

where  $\hat{\sigma}$  is the modal sequence resulting from inserting  $f$  to  $\xi$  at time  $\tau$  and (recall that)  $\Psi$  is the set of feasible modal sequences. Let  $(g, t) \in \Phi \times [0, T]$  be an argmin of the term in (44). Observe that if  $D(\xi) < 0$  then the insertion of the modal function  $g$  in a small interval centered at time  $t$  would result in a reduced value of  $J$ . On the other hand, there is no indication of such a reduction in  $J$  if  $D(\xi) = 0$ . The case where  $D(\xi) > 0$  is not possible since, for  $f = f_{\sigma(i)}$  and for all  $\tau \in (\tau_{i-1}, \tau_i)$ ,  $D_{f,\tau}(\xi) = 0$ , and hence, by definition (44),  $D(\xi)$  is always less than or equal to zero. Based on all of this, our algorithm has the following form.

### Algorithm 3.2.1

*Data:* A modal sequence  $\sigma_0$ .

*Step 0:* Set  $j = 0$ .

*Step 1:* Solve the problem  $P_{\sigma_j}$  to the extent of computing a switching time vector  $s_j$  that is a Kuhn-Tucker point for  $P_{\sigma_j}$ . Denote the resulting schedule by  $\xi_j := (\sigma_j, s_j)$ .

*Step 2:* If  $D(\xi_j) = 0$ , stop and exit. Otherwise, compute  $g \in \Phi$  and  $t \in [0, T]$  such that  $D_{g,t}(\xi_j) = D(\xi_j)$ .

*Step 3:* Define  $\hat{\xi}_{j+1} := (\sigma_{j+1}, \hat{s}_{j+1})$  to be the schedule obtained by inserting  $g$  to the schedule  $(\sigma_j, s_j)$  at time  $t$ .

*Step 4:* Set  $j = j + 1$ , and goto Step 1. ■

A few remarks are due.

**Remark 3.2.1** The formula for  $D_{f,\tau}(\xi)$  will be presented later in this Chapter, based on the results derived in Chapter II. We will mention straightforward extensions thereof to simultaneous insertions of multiple modal functions at a given time  $\tau$ . Such multiple insertions can replace the single-mode insertion in Step 2, but we prefer to present the analysis only in the setting of the single insertion in order to keep it as simple as possible.

**Remark 3.2.2** *Step 1 and Step 2 generally require infinite-loop procedures. In Step 1, solving  $P_{\sigma_j}$  typically involves an iterative nonlinear-programming algorithm, i.e., Algorithm 2.2.1. In Step 2, the computation of  $D(\xi_j)$  involves maximization of  $D_{g,t}(\xi_j)$  over the infinite set  $\Phi \times [0, T]$ . Implementations must involve practical stopping rules that yield approximations to the desired quantities. We do not discuss here specific approximation schemes, but rather assume that they are available and carry out the convergence analysis by assuming exact computations.*

For every  $N = 0, 1, \dots$ , we define  $\Xi_N$  to be the set of schedules  $\xi = (\sigma, s)$  such that  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\} \in \Psi$ , and  $s = (s_1, \dots, s_N)^T$  is a Kuhn-Tucker point for  $P_\sigma$ . Hence, in the sense of problem  $P_\sigma$  (as defined in Chapter II), when we only considered optimizing the switching times over a fixed schedule, we are at an optimal point. This is a subtle point in the proof of sufficient descent of the algorithm that follows. Observe that Algorithm 3.2.1 computes, in Step 1, a schedule  $\xi_j \in \Xi_{N(j)}$  for some integer  $N(j)$ . Moreover, we have the inequality  $N(j+1) > N(j)$  for all  $j = 1, 2, \dots$ ; in fact, either  $N(j+1) = N(j) + 2$  or  $N(j+1) = N(j) + 1$ . We will consider these schedules as the iteration points computed by the algorithm, and it is in their terms that we shall characterize convergence.

The condition  $D(\xi_j) = 0$  can be viewed as a necessary condition for a local optimality, and hence it acts as the stopping rule in Step 2 of the algorithm. Formally, we label this condition *stationarity*, defined as follows.

**Definition 3.2.1** *A schedule  $\xi = (\sigma, s)$  is said to be a stationary schedule for the problem  $P$  if*

- (i)  $\sigma$  is a feasible sequence, i.e.,  $\sigma \in \Psi$ ;
- (ii)  $s$  is a Kuhn-Tucker point for the problem  $P_\sigma$ , and
- (iii)  $D(\sigma, s) = 0$ . ■

Thus, Algorithm 3.2.1 stops at a stationary schedule, and we expect it to converge to stationary points in the sense of Section 3.1. To characterize this kind of convergence, let us define the optimality function  $\theta_N : \Xi_N \rightarrow \mathbb{R}^-$  by  $\theta_N(\xi) = D(\xi)$ , and further defining

$\Sigma_N \subset \Xi_N$  to be the set of stationary schedules, we observe that  $\theta_N(\xi) = 0$  if and only if  $\xi \in \Sigma_N$ . Now the main result of the chapter is that, under suitable assumptions, if the algorithm computes a sequence of schedules  $\xi_j \in \Xi_{N(j)}$ ,  $j = 1, 2, \dots$ , then

$$\lim_{j \rightarrow \infty} \theta_{N(j)}(\xi_j) = 0. \quad (45)$$

Whether this result is in force depends on specific details of the algorithm, and especially on the procedure used in Step 1 to solve  $P_{\sigma_j}$ . This will be discussed in Section 3.3, where the convergence analysis will be nontrivial due to the fact that the switching time vectors  $s_j$  lie in successive Euclidean spaces of increasing dimensions.

Before closing this section some results are recalled concerning the derivative terms  $\frac{dJ}{d\tau_i}$  and  $D_{f,\tau}(\xi)$ . As for the characterization of Kuhn-Tucker points for  $P_\sigma$  the reader is referred to Chapter II. Let us fix  $\sigma = \{\sigma(1), \dots, \sigma(N+1)\}$  and consider  $J$  as a function of  $s = (\tau_1, \dots, \tau_N)^T \in \mathbb{R}^N$  ( $s$  need not be a Kuhn-Tucker point for  $P_\sigma$ ). Recall (41). Define  $x_i$  ( $i = 0, \dots, N+1$ ) by  $x_i := x(\tau_i)$ , and note that  $x_i$  is well defined since  $x(\cdot)$  is continuous in  $t$ . Furthermore, define the costate vector  $p(t)$  by the following equation,

$$\dot{p}(t) = - \left( \frac{\partial F}{\partial x}(x, t) \right)^T p(t) - \left( \frac{\partial L}{\partial x}(x, t) \right)^T, \quad (46)$$

with the boundary condition  $p(T) = 0$ , and define  $p_i$  by  $p_i = p(\tau_i)$  ( $p_i$  is well defined since  $p(\cdot)$  is continuous). Now the derivative  $\frac{dJ}{d\tau_i}$  ( $i = 1, \dots, N$ ) has the following form:<sup>3</sup>

$$\frac{dJ}{d\tau_i} = p_i^T (f_{\sigma(i)}(x_i) - f_{\sigma(i+1)}(x_i)), \quad (47)$$

as presented in Chapter II. Moreover, this derivative term is continuous in  $\tau_i$  for all  $s$  satisfying the constraint inequalities in (40).

The following remark will be needed subsequently.

**Remark 3.2.3** *We call a set of contiguous integers  $\{m, \dots, n\}$  a block if  $m = m(i)$  and  $n = n(i)$  for some  $i \in \{m, \dots, n\}$ . For a block of switches, the optimality condition presented for  $P_\sigma$  in Chapter II means that  $\sum_{k=m}^n \frac{dJ}{d\tau_k}(s) = 0$  for every block  $\{k, \dots, n\} \subset \{1, \dots, N\}$ .*

---

<sup>3</sup>We use the derivative notation  $\frac{dJ}{d\tau_i}$  and not  $\frac{\partial J}{\partial \tau_i}$  since we will focus on the total derivative of  $J$  as a function of  $\tau_i$ .

Finally, consider the insertion of a modal function  $f \in \Phi$  at a given time  $\tau \in (0, T)$ . Suppose first that  $\tau \in (\tau_{i-1}, \tau_i)$  for some  $i = 1, \dots, N + 1$ . Then (from Chapter II), with  $\xi := (\sigma, s)$ , we have that

$$D_{f,\tau}(\xi) = p(\tau)^T (f(x(\tau)) - f_{\sigma(i)}(x(\tau))). \quad (48)$$

For the case where  $\tau = \tau_i$ ,

$$D_{f,\tau}(\xi) = p(\tau_i)^T \left( f(x_i) - \frac{f_{m(i)}(x_i) + f_{n(i)+1}(x_i)}{2} \right). \quad (49)$$

The next section will establish conditions on Step 1 of Algorithm 3.2.1 guaranteeing convergence in the sense of Section 3.1.

### 3.3 Convergence Analysis

In order to complete the description of Algorithm 3.2.1 we have to specify the procedure for implementing Step 1. Dubbed Procedure 3.3.1, its purpose is to solve  $P_{\sigma_j}$  to the extent of computing a Kuhn-Tucker point. We assume that it is comprised of an iterative feasible descent algorithm which starts at the point  $\hat{s}_j$  computed in Step 3 of the previous iteration of Algorithm 3.2.1. Consequently Algorithm 3.2.1 is a descent algorithm as well, and hence  $J(\xi_{j+1}) \leq J(\xi_j)$  for all  $j = 0, 1, \dots$ . Whether it converges in the sense of Section 3.1 depends on the details of Procedure 3.2.1. In our initial investigation we considered a descent algorithm in the opposite direction of the projection of the gradient onto the feasible set, and attempted to prove convergence by establishing the property of sufficient descent. This, however, we could not prove. The stumbling block was the increasing dimensions of the programs  $P_{\sigma_j}$  which, although providing descent, failed to afford sufficient descent, thus leading us to believe that these algorithms do not converge.

To get around this difficulty we replaced the above descent direction by a *descent curve* in  $\mathbb{R}^{N(j)}$ . Parameterizing it by  $\lambda \geq 0$ , we denote this curve by  $\{C(\lambda)\}_{\lambda \geq 0}$ . It is piecewise linear and continuous, and hence not differentiable everywhere. It yields sufficient descent for  $J$  only at the first step of Procedure 3.3.1, but not in later steps. This, however, is sufficient to guarantee sufficient descent of Algorithm 3.2.1 as long as we require that all the later steps of Procedure 3.3.1 result in a descent in  $J$ . Thus, the first step of Procedure

3.3.1 has to be specified in detail, while regarding subsequent steps all that we say is that they yield descent.

To describe the first step of Procedure 3.3.1 and establish the relevant notation for its analysis, it is instructive to track one iteration of Algorithm 3.2.1. In Step 1 it has  $\sigma_j := \{\sigma_j(1), \dots, \sigma_j(N(j))\}$ , and it solves the problem  $P_{\sigma_j}$  to the extent of computing a Kuhn-Tucker point  $s_j := (\tau_{j,1}, \dots, \tau_{j,N(j)})^T$ . At Step 2, suppose that  $D(\xi_j) < 0$ , so that the algorithm computes  $g \in \Phi$  and  $t \in [0, T]$  such that  $D_{g,t}(\xi_j) = D(\xi_j)$ . Assume that  $t \in (\tau_{j,i-1}, \tau_{j,i})$  for some  $i = 1, \dots, N(j) + 1$ ; the analysis for the case where  $t = \tau_{j,i}$  is similar (as will be evident later) and hence it is omitted. Let us use the notation  $g = f_{\sigma(\star)}$ . Next, consider Step 3, here we have that

$$\sigma_{j+1} = \{\sigma_j(1), \dots, \sigma_j(i), \sigma(\star), \sigma_j(i), \dots, \sigma_j(N(j) + 1)\}, \quad (50)$$

and

$$\hat{s}_{j+1} = (\tau_{j,1}, \dots, \tau_{j,i-1}, t, t, \tau_{j,i}, \dots, \tau_{j,N(j)})^T \in \mathbb{R}^{N(j+1)}, \quad (51)$$

where  $N(j + 1) = N(j) + 2$ . Note that  $t$  is the time of two switching points and the modal function  $g$  is inserted between them for an interval of length 0. Moreover, if we use the notation  $\hat{s}_{j+1} := (\tau_{j+1,1}, \dots, \tau_{j+1,N(j+1)})^T$ , then  $\tau_{j+1,k} = \tau_{j,k}$  for all  $k = 1, \dots, i - 1$ ;  $\tau_{j+1,i} = \tau_{j+1,i+1} = t$ ; and  $\tau_{j+1,k} = \tau_{j,k-2}$  for all  $k = i + 2, \dots, N(j + 1)$ .

The algorithm now returns to Step 1, where it solves  $P_{\sigma_{j+1}}$  by Procedure 3.3.1, starting from  $\hat{s}_{j+1}$ . The first step of this procedure consists of the Armijo step size (whose details are described below) along the curve  $\{C(\lambda)\}_{\lambda \geq 0}$ , next defined. For every  $\lambda \geq 0$ ,  $C(\lambda) \in \mathbb{R}^{N(j+1)}$ , and we denote its coordinates by  $C(\lambda) = (c_1(\lambda), \dots, c_{N(j+1)}(\lambda))^T$  (the dependence on  $j + 1$  is clear from the context and hence implicit). Its starting point is  $C(0) = \hat{s}_{j+1}$  and hence  $c_k(0) = \tau_{j+1,k}$  for every  $k = 1, \dots, N(j + 1)$ . In particular we have that  $c_k(0) = \tau_{j,k}$  for all  $k = 1, \dots, i - 1$ ;  $c_i(0) = c_{i+1}(0) = t$ ; and  $c_k(0) = \tau_{j,k-2}$  for all  $k = i + 2, \dots, N(j + 1)$ . The curve is defined as follows. For every  $\lambda \geq 0$ ,  $c_i(\lambda) = \max\{t - \lambda, 0\}$  and  $c_{i+1}(\lambda) = \min\{t + \lambda, T\}$ ; for every  $k = 1, \dots, i - 1$ ,  $c_k(\lambda) = \min\{c_i(\lambda), \tau_{j+1,k}\}$ ; and for every  $k = i + 2, \dots, N(j + 1)$ ,  $c_k(\lambda) = \max\{c_{i+1}(\lambda), \tau_{j+1,k}\}$ . To put it in words, the  $i$ th and  $(i + 1)$ th coordinates, starting both at  $t$ , move away from each other in opposite directions at the rate

of 1 until they reach 0 and  $T$ , respectively, where they stay thereafter. Along the way they “bump” into other coordinates of  $\hat{s}_{j+1}$ , and then they drag them along. Eventually, for  $\lambda$  large enough,  $c_k(\lambda) = 0$  for all  $k = 1, \dots, i$ , and  $c_k(\lambda) = T$  for all  $k = i + 1, \dots, N(j + 1)$ . This curve is piecewise linear. Initially it moves in the two-dimensional plane defined by its  $i$ th and  $(i + 1)$ st coordinates, and it moves in spaces of increasing dimensions each time it “bumps” into one of the coordinates of  $\hat{s}_{j+1}$ . We will denote the points where the curve changes directions by  $\lambda_\nu$ ,  $\nu = 1, 2, \dots$ , in increasing order, and we define  $\lambda_0 := 0$  for convenience.

Define the function  $h : \mathbb{R}^+ \rightarrow \mathbb{R}$  by

$$h(\lambda) := J(C(\lambda)), \quad (52)$$

i.e., the cost functional  $J$  along the curve  $C(\lambda)$ . Then  $h(\cdot)$  is continuous and piecewise differentiable, and it is differentiable at all but the points  $\lambda_\nu$ ,  $\nu = 1, 2, \dots$ . We will denote its derivative by  $h'(\lambda)$  whenever it exists, and use the notation  $h'(\lambda^+)$  and  $h'(\lambda^-)$  for the right derivative and left derivative, respectively, which exist for every  $\lambda \in [0, T]$ .

Procedure 3.3.1 now has the following form.

**Procedure 3.3.1**

*Parameters:* A constant  $\alpha \in (0, 1)$ , and a monotone-decreasing sequence of positive numbers  $\{\lambda(\ell)\}_{\ell=0}^\infty$  such that  $\lim_{\ell \rightarrow \infty} \lambda(\ell) = 0$ .

*Starting point:*  $\hat{s}_{j+1} = C(0)$ .

*First step:* Compute  $\bar{\ell}$  defined as

$$\bar{\ell} := \min\{\ell = 0, 1, \dots, : h(\lambda(\ell)) - h(0) \leq \alpha \lambda(\ell) h'(\lambda(\ell)^+)\}. \quad (53)$$

*Define  $\bar{s}_{j+1}$  by  $\bar{s}_{j+1} := C(\lambda(\bar{\ell}))$ .*

*Subsequent steps:* Use any feasible descent algorithm, starting from  $\bar{s}_{j+1}$ , to compute  $s_{j+1}$ ,

*A Kuhn-Tucker point for  $P_{\sigma_{j+1}}$ .* ■

We recognize  $\lambda(\bar{\ell})$  as the Armijo step-size along the curve  $\{C(\lambda)\}$ , and we refer the reader to [6, 56] for its analysis under general assumptions as well as its practical deployment in

nonlinear programming. The first step of the procedure gives us the sufficient descent that we seek, and proving this fact is the focus of the rest of the section.

To start with the analysis, we first characterize the derivative term  $h'(\lambda)$ . For every  $\lambda \geq 0$  and for every  $k = 0, \dots, N(j+1)$ , we define the integer quantities  $m(\lambda; k)$  and  $n(\lambda; k)$  by

$$m(\lambda; k) = \min\{m \leq k : c_m(\lambda) = c_k(\lambda)\}, \quad (54)$$

and

$$n(\lambda; k) = \max\{n \geq k : c_n(\lambda) = c_k(\lambda)\}. \quad (55)$$

Note that  $m(\lambda; k)$  and  $n(\lambda; k)$  are extensions of the quantities  $m(i)$  and  $n(i)$  defined earlier at a point  $s$ . In particular, for  $\lambda = 0$ ,  $C(0) = \hat{s}_{j+1}$ , and since by assumption  $\tau_{j,i-1} < t < \tau_{j,i}$ , we have that  $m(0; i) = m(0, i+1) = i$  and  $n(0; i) = n(0, i+1) = i+1$ . Observe that, for a given  $k$ ,  $m(\lambda; k)$  is monotone non-increasing in  $\lambda$  and  $n(\lambda; k)$  is monotone nondecreasing in  $\lambda$ , and these functions change their values only at the points  $\lambda_\nu$ ,  $\nu = 1, 2, \dots$

Next, consider the state equation (38) and the costate equation (46), which were defined for a given schedule  $\xi = (\sigma, s)$ . Now we have a family of schedules parameterized by  $\lambda$ ,  $\{\xi(\lambda)\}$ , and hence we have to parameterize the above equations by  $\lambda$  as well. For every  $\lambda \geq 0$ , let us denote by  $F_\lambda(x, t)$  the function  $F$  in (38) corresponding to the schedule  $\xi(\lambda)$ , and denote by  $x_\lambda(t)$  its corresponding state trajectory; namely,

$$\dot{x}_\lambda = F_\lambda(x_\lambda, t), \quad x_\lambda(0) = x_0. \quad (56)$$

Likewise, we denote by  $p_\lambda(t)$  the corresponding costate trajectory obtained by (46) with the schedule  $\xi(\lambda)$ , namely

$$\dot{p}_\lambda = -\frac{\partial F_\lambda}{\partial x}(x_\lambda, t)^T p_\lambda - \frac{\partial L}{\partial x}(x_\lambda, t), \quad p_\lambda(T) = 0. \quad (57)$$

Then, for a fixed  $\lambda \geq 0$ , in analogy with (47) we have the following equation,

$$\frac{dJ}{dc_k(\lambda)} = p_\lambda(c_k(\lambda))^T \left( f_{\sigma_{j+1}(k)}(x_\lambda(c_k(\lambda))) - f_{\sigma_{j+1}(k+1)}(x_\lambda(c_k(\lambda))) \right). \quad (58)$$

Let us define  $\chi_{[c_k(\lambda) > 0]}$  to be the characteristic function of the event that  $c_k(\lambda) > 0$ , and likewise, we define  $\chi_{[c_k(\lambda) < T]}$  to be the characteristic function of the event that  $c_k(\lambda) < T$ .

We now have the following characterization of  $h'(\lambda)$ .



**Lemma 3.3.1** For every  $\nu = 0, 1, \dots$ , and for every  $\lambda \in (\lambda_\nu, \lambda_{\nu+1})$ ,

$$\begin{aligned} h'(\lambda) &= p_\lambda(c_i(\lambda))^T \left( f_{\sigma_{j+1}(i+1)}(x_\lambda(c_i(\lambda))) - f_{\sigma_{j+1}(m(\lambda;i))}(x_\lambda(c_i(\lambda))) \right) \chi_{[c_i(\lambda)>0]} \\ &+ p_\lambda(c_{i+1}(\lambda))^T \left( f_{\sigma_{j+1}(i+1)}(x_\lambda(c_{i+1}(\lambda))) - f_{\sigma_{j+1}(n(\lambda;i+1)+1)}(x_\lambda(c_{i+1}(\lambda))) \right) \chi_{[c_{i+1}(\lambda)<T]}. \end{aligned} \quad (59)$$

*Proof.* Fix  $\nu = 0, 1, \dots$ , and fix  $\lambda \in (\lambda_\nu, \lambda_{\nu+1})$ . For all  $k = m(\lambda; i), \dots, i$ : if  $c_i(\lambda) = 0$  then  $c_k(\lambda) = 0$  and  $\frac{dc_k(\lambda)}{d\lambda} = 0$ , and if  $c_i(\lambda) > 0$  then  $c_k(\lambda) = t - \lambda$  and hence  $\frac{dc_k(\lambda)}{d\lambda} = -1$ . Similarly, for all  $k = i + 1, \dots, n(\lambda; i + 1)$ : if  $c_{i+1}(\lambda) = T$  then  $c_k(\lambda) = T$  and  $\frac{dc_k(\lambda)}{d\lambda} = 0$ , and if  $c_{i+1}(\lambda) < T$  then  $c_k(\lambda) = t + \lambda$  and hence  $\frac{dc_k(\lambda)}{d\lambda} = 1$ . Moreover, for all  $k \in \{1, \dots, m(\lambda; i) - 1\} \cup \{n(\lambda; i + 1) + 1, \dots, N(j + 1)\}$ ,  $c_k(\lambda) = \tau_{j+1,k}$  and hence  $\frac{dc_k(\lambda)}{d\lambda} = 0$ . Consequently, and by an application of the chain rule, we have that

$$h'(\lambda) = -\chi_{[c_i(\lambda)>0]} \sum_{k=m(\lambda;i)}^i \frac{dJ}{dc_k(\lambda)} + \chi_{[c_{i+1}(\lambda)<T]} \sum_{k=i+1}^{n(\lambda;i+1)} \frac{dJ}{dc_k(\lambda)}. \quad (60)$$

Plug (58) in (60) to obtain,

$$\begin{aligned} h'(\lambda) &= -\chi_{[c_i(\lambda)>0]} \sum_{k=m(\lambda;i)}^i p_\lambda(c_k(\lambda))^T \left( f_{\sigma_{j+1}(k)}(x_\lambda(c_k(\lambda))) - f_{\sigma_{j+1}(k+1)}(x_\lambda(c_k(\lambda))) \right) \\ &+ \chi_{[c_{i+1}(\lambda)<T]} \sum_{k=i+1}^{n(\lambda;i+1)} p_\lambda(c_k(\lambda))^T \left( f_{\sigma_{j+1}(k)}(x_\lambda(c_k(\lambda))) - f_{\sigma_{j+1}(k+1)}(x_\lambda(c_k(\lambda))) \right). \end{aligned} \quad (61)$$

By definition of  $m(\lambda; i)$  and  $n(\lambda; i + 1)$ , we have that  $c_k(\lambda) = c_i(\lambda)$  for all  $k = m(\lambda; i), \dots, i$ , and  $c_k(\lambda) = c_{i+1}(\lambda)$  for all  $k = i + 1, \dots, n(\lambda; i + 1)$ . Using this in (61) renders the two sums telescopic and hence yields (59).  $\blacksquare$

Recall that  $m(\lambda; k) = m(\lambda_\nu; k)$  and  $n(\lambda; k) = n(\lambda_\nu; k)$  for all  $\lambda \in (\lambda_\nu, \lambda_{\nu+1})$ . Therefore  $h'(\lambda)$  is discontinuous only at the points  $\lambda_\nu$  as can be seen from the presence of the terms  $m(\lambda; i)$  and  $n(\lambda; i + 1)$  in (59). Consequently  $h'(\lambda_\nu^+)$  is given by (59) with  $\lambda = \lambda_\nu$ , namely,

$$\begin{aligned} h'(\lambda_\nu^+) &= p_{\lambda_\nu}(c_i(\lambda_\nu))^T \left( f_{\sigma_{j+1}(i+1)}(x_{\lambda_\nu}(c_i(\lambda_\nu))) - f_{\sigma_{j+1}(m(\lambda_\nu;i))}(x_{\lambda_\nu}(c_i(\lambda_\nu))) \right) \chi_{[c_i(\lambda_\nu)>0]} \\ &+ p_{\lambda_\nu}(c_{i+1}(\lambda_\nu))^T \left( f_{\sigma_{j+1}(i+1)}(x_{\lambda_\nu}(c_{i+1}(\lambda_\nu))) - f_{\sigma_{j+1}(n(\lambda_\nu;i+1)+1)}(x_{\lambda_\nu}(c_{i+1}(\lambda_\nu))) \right) \chi_{[c_{i+1}(\lambda_\nu)<T]}. \end{aligned} \quad (62)$$

Of a particular interest is the case  $\nu = 0$ , namely the term  $h'(0^+)$ , for which we have the following formula.

**Lemma 3.3.2** *The derivative term  $h'(0^+)$  has the following form,*

$$h'(0^+) = 2D(\xi_j). \quad (63)$$

*Proof.* Recall that  $C(0) = \hat{s}_{j+1}$ , and hence, and by (51),  $c_i(0) = t$  and  $c_{i+1}(0) = t$ . Also, since by assumption  $t \in (\lambda_\nu, \lambda_{\nu+1})$ ,  $m(0; i) = i$  and  $n(0; i+1) = i+1$ . Therefore, we get from (62) that

$$\begin{aligned} h'(0^+) &= p_0(t)^T \left( f_{\sigma_{j+1}(i+1)}(x_0(t)) - f_{\sigma_{j+1}(i)}(x_0(t)) \right) + \\ &\quad p_0(t)^T \left( f_{\sigma_{j+1}(i+1)}(x_0(t)) - f_{\sigma_{j+1}(i+2)}(x_0(t)) \right). \end{aligned} \quad (64)$$

By (50),  $f_{\sigma_{j+1}(i)} = f_{\sigma_j(i)}$ ,  $f_{\sigma_{j+1}(i+1)} = g$ , and  $f_{\sigma_{j+1}(i+2)} = f_{\sigma_j(i)}$ . Plug this in (64) to obtain,

$$h'(0^+) = 2p_0(t)^T \left( g(x_0(t)) - f_{\sigma_j(i)}(x_0(t)) \right). \quad (65)$$

Finally, (63) follows from an application of (48) with  $g$  and  $t$  instead of  $f$  and  $\tau$ , and the fact that  $D_{g,t}(\xi_j) = D(\xi_j)$ . ■

We next turn to proving convergence of Algorithm 3.2.1. The proof is based on the following successive steps.

1. We first establish that the functions  $x_\lambda(\cdot)$  and  $p_\lambda(\cdot)$ , viewed as elements in  $L^\infty[0, T]$ , are Lipschitz continuous in  $\lambda$ .
2. We prove that  $h'(\lambda^+)$  has a continuity property, uniformly with respect to  $\xi_j$ , at  $\lambda = 0$ . This is despite the fact that  $h'(\lambda)$  is generally discontinuous in  $\lambda$ . What makes this statement true is the way the point  $C(0) = \hat{s}_{j+1}$  was constructed in Step 3 of Algorithm 3.2.1.
3. We prove that Algorithm 3.2.1 with Procedure 3.3.1 in its first step has the sufficient descent property.

The result progress through a sequence of preliminary results where most the proofs have been presented in Chapter II, and are therefore omitted.

**Lemma 3.3.3** *There exists a constant  $D > 0$  such that, for every schedule  $\hat{\xi}_{j+1}$  computed in Step 3 of Algorithm 3.2.1, and for every  $\lambda \geq 0$ ,  $\|x_\lambda\|_\infty \leq D$  and  $\|p_\lambda\|_\infty \leq D$ , where  $\|\cdot\|$  denotes the  $L^\infty$  norm of a function.*

**Lemma 3.3.4** *There exists a constant  $L_1 > 0$  such that, for every schedule  $\hat{\xi}_{j+1}$  computed in Step 3 of Algorithm 3.2.1, for every  $\lambda > 0$ , and for every  $\tau \in [0, T)$  and  $\delta > 0$ ,  $|x_\lambda(\tau + \delta) - x_\lambda(\tau)| \leq L_1\delta$  and  $|p_\lambda(\tau + \delta) - p_\lambda(\tau)| \leq L_1\delta$ .*

**Lemma 3.3.5** *There exists a constant  $L > 0$  such that, for every schedule  $\hat{\xi}_{j+1}$  computed by Algorithm 3.2.1, and for every  $\lambda > 0$ ,  $\|x_\lambda - x_0\|_\infty \leq L\lambda$  and  $\|p_\lambda - p_0\|_\infty \leq L\lambda$ .*

**Lemma 3.3.6** *There exists a constant  $K > 0$  such that, for every schedule  $\hat{\xi}_{j+1}$  computed by Algorithm 3.2.1, and for every  $\lambda > 0$ ,*

$$|h'(\lambda^+) - h'(0^+)| \leq K\lambda. \quad (66)$$

The proof is complicated by the fact that when the switches are pushed apart (i.e. when  $\lambda$  increase), several switches may occur at the same time. This makes it hard to prove Lipschitz continuity and it is here the fact that we are inserting the new mode at an already optimized switching time vector is applied.

*Proof of Lemma 3.3.6.* Fix  $\lambda > 0$ . We can assume, without loss of generality, that  $c_i(\lambda) > 0$  and  $c_{i+1}(\lambda) < T$ . Then, applications of (59) and (62) with  $\nu = 0$  yield,

$$\begin{aligned} & h'(\lambda^+) - h'(0^+) \\ &= p_\lambda(c_i(\lambda))^T \left( f_{\sigma_{j+1}(i+1)}(x_\lambda(c_i(\lambda))) - f_{\sigma_{j+1}(m(\lambda; i))}(x_\lambda(c_i(\lambda))) \right) \\ &+ p_\lambda(c_{i+1}(\lambda))^T \left( f_{\sigma_{j+1}(i+1)}(x_\lambda(c_{i+1}(\lambda))) - f_{\sigma_{j+1}(n(\lambda; i+1)+1)}(x_\lambda(c_{i+1}(\lambda))) \right) \\ &- p_0(c_i(0))^T \left( f_{\sigma_{j+1}(i+1)}(x_0(c_i(0))) - f_{\sigma_{j+1}(m(0; i))}(x_0(c_i(0))) \right) \\ &- p_0(c_{i+1}(0))^T \left( f_{\sigma_{j+1}(i+1)}(x_0(c_{i+1}(0))) - f_{\sigma_{j+1}(n(0; i+1)+1)}(x_0(c_{i+1}(0))) \right). \end{aligned} \quad (67)$$

Defining  $U_1$ ,  $U_2$ ,  $U_3$ , and  $U_4$  by (69)-(72), below, and rearranging the various terms in (67), we obtain that

$$h'(\lambda^+) - h'(0^+) = U_1 + U_2 + U_3 + U_4, \quad (68)$$

where

$$U_1 = p_\lambda(c_i(\lambda))^T f_{\sigma_{j+1}(i+1)}(x_\lambda(c_i(\lambda))) - p_0(c_i(0))^T f_{\sigma_{j+1}(i+1)}(x_0(c_i(0))), \quad (69)$$

$$U_2 = p_\lambda(c_i(\lambda))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_\lambda(c_i(\lambda))) - p_0(c_i(0))^T f_{\sigma_{j+1}(m(0;i))}(x_0(c_i(0))), \quad (70)$$

$$U_3 = p_\lambda(c_{i+1}(\lambda))^T f_{\sigma_{j+1}(i+1)}(x_\lambda(c_{i+1}(\lambda))) - p_0(c_{i+1}(0))^T f_{\sigma_{j+1}(i+1)}(x_0(c_{i+1}(0))), \quad (71)$$

$$U_4 = p_\lambda(c_{i+1}(\lambda))^T f_{\sigma_{j+1}(n(\lambda;i+1)+1)}(x_\lambda(c_{i+1}(\lambda))) - p_0(c_{i+1}(0))^T f_{\sigma_{j+1}(n(0;i+1)+1)}(x_0(c_{i+1}(0))). \quad (72)$$

Recall that  $c_i(0) = t$  and  $c_i(\lambda) = t - \lambda$ , and hence  $c_i(\lambda) - c_i(0) = -\lambda$ . Therefore, and by Lemma 3.3.5, Lemma 3.3.4, and Lemma 3.3.3, there exists a constant  $K_1 > 0$  (independent of  $\hat{\xi}_{j+1}$  or  $\lambda$ ) such that,

$$|U_1| \leq K_1 \lambda. \quad (73)$$

Similarly,  $c_{i+1}(0) = t$  and  $c_{i+1}(\lambda) = t + \lambda$ , and hence  $c_{i+1}(\lambda) - c_{i+1}(0) = \lambda$ ; therefore there exists a constant  $K_3 > 0$  such that,

$$|U_3| \leq K_3 \lambda. \quad (74)$$

We need similar inequalities for  $U_2$  and  $U_4$ . These, however, are more problematic. The reason, regarding  $U_2$ , can be seen in the right-hand-side of (70), whose two additive terms contain different modal functions, namely  $f_{\sigma_{j+1}(m(\lambda;i))}$  and  $f_{\sigma_{j+1}(m(0;i))}$ , and hence their difference does not indicate a bound like the one in (73). A similar issue arises in (72). However, we shall see that such bounds are indeed in force, and this is due to the particular structure of the algorithm, and especially to the fact that the point  $s_j$  was a Kuhn-Tucker point for  $P_{\sigma_j}$ .

Let us consider  $U_2$  as defined in (70). Adding and subtracting the same terms, and defining  $U_{2,1}$ ,  $U_{2,2}$ , and  $U_{2,3}$  by (76)-(78), below, we have that

$$U_2 = U_{2,1} + U_{2,2} + U_{2,3}. \quad (75)$$

The various terms in the right-hand-side of (75) are defined as follows.

$$U_{2,1} = p_\lambda(c_i(\lambda))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_\lambda(c_i(\lambda))) - p_0(c_i(0))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_0(c_i(0))), \quad (76)$$

$$U_{2,2} = p_0(c_i(0))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_0(c_i(0))) - p_0(c_{m(\lambda;i)}(0))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_0(c_{m(\lambda;i)}(0))), \quad (77)$$

$$U_{2,3} = p_0(c_{m(\lambda;i)}(0))^T f_{\sigma_{j+1}(m(\lambda;i))}(x_0(c_{m(\lambda;i)}(0))) - p_0(c_i(0))^T f_{\sigma_{j+1}(m(0;i))}(x_0(c_i(0))). \quad (78)$$

regarding  $U_{2,1}$ , similarly to (73) and (74), there exists a constant  $K_{2,1} > 0$  such that

$$|U_{2,1}| \leq K_{2,1}\lambda. \quad (79)$$

Concerning  $U_{2,2}$ , recall that  $c_i(0) = t$ ;  $c_i(\lambda) = t - \lambda$ ; by (54)  $c_{m(\lambda;i)}(\lambda) = c_i(\lambda)$ ;  $m(\lambda;i) \leq i$  and hence  $c_{m(\lambda;i)}(0) \leq c_i(0)$ ; and  $c_{m(\lambda;i)}(\cdot)$  is monotone non-increasing in  $\lambda$ . Therefore, we have that  $0 \leq c_i(0) - c_{m(\lambda;i)}(0) \leq c_i(0) - c_{m(\lambda;i)}(\lambda) = t - (t - \lambda) = \lambda$ , and hence,  $|c_i(0) - c_{m(\lambda;i)}(0)| \leq \lambda$ . Consequently and by (77), in a way similar to (73) and (74), there exists a constant  $K_{2,2} > 0$  such that

$$|U_{2,2}| \leq K_{2,2}\lambda. \quad (80)$$

$U_{2,3}$  is where the problem lies, because the two additive terms in the right-hand-side of (78) involve two different modal functions. Recall that  $m(0;i) = i$ , and hence (and by (78)) we can write  $U_{2,3}$  as

$$U_{2,3} = \sum_{k=m(\lambda;i)}^{i-1} \left( p_0(c_k(0))^T f_{\sigma_{j+1}(k)}(x_0(c_k(0))) - p_0(c_{k+1}(0))^T f_{\sigma_{j+1}(k+1)}(x_0(c_{k+1}(0))) \right). \quad (81)$$

Note that, in the special case where  $m(\lambda;i) = i$ , the range of the sum in (81) is vacuous and hence (81) suggests that  $U_{2,3} = 0$ , which is consistent with (78). Thus, we can assume without loss of generality that  $m(\lambda;i) < i$ . Defining  $V_{2,3,k}$  and  $W_{2,3,k}$  by

$$V_{2,3,k} = p_0(c_k(0))^T \left( f_{\sigma_{j+1}(k)}(x_0(c_k(0))) - f_{\sigma_{j+1}(k+1)}(x_0(c_k(0))) \right) \quad (82)$$

and

$$W_{2,3,k} = p_0(c_k(0))^T f_{\sigma_{j+1}(k+1)}(x_0(c_k(0))) - p_0(c_{k+1}(0))^T f_{\sigma_{j+1}(k+1)}(x_0(c_{k+1}(0))), \quad (83)$$

we have that

$$U_{2,3} = \sum_{k=m(\lambda;i)}^{i-1} (V_{2,3,k} + W_{2,3,k}). \quad (84)$$

Recall (51) that for every  $k = m(\lambda; i), \dots, i-1$ ,  $c_k(0) = \tau_{j,k}$ , where  $s_j = (\tau_{j,1}, \dots, \tau_{j,N(j)})^T$  was the Kuhn-Tucker point for  $P_{\sigma_j}$  last computed in Step 1 of Algorithm 3.2.1. Moreover, by applying (58) with  $\lambda = 0$  we obtain, for all  $k = m(\lambda; i), \dots, i-1$ , that

$$V_{2,3,k} = \frac{dJ}{d\tau_{j,k}}(s_j). \quad (85)$$

Recall the definition of a *block*, made in Remark 3.2.3. We now show that for every  $\lambda > 0$ , the set  $\{m(\lambda; i), \dots, i-1\}$  is comprised of the union of contiguous blocks associated with  $C(0)$ . To this end, all we need to show is that  $c_{m(\lambda; i)-1}(0) < c_{m(\lambda; i)}(0)$  and  $c_{i-1}(0) < c_i(0)$ , since in this case  $m(\lambda; i) - 1$  is not in the block containing  $m(\lambda; i)$ , and  $i-1$  is not in the block containing  $i$ , according to the curve  $C(0)$ . The latter inequality follows from our assumption that  $\tau_{j,i-1} < \tau_{j,i} = t$ . Regarding the former inequality, by definition (54),  $c_{m(\lambda; i)-1}(\lambda) < c_{m(\lambda; i)}(\lambda)$ ; and by the definition of the curve,  $c_{m(\lambda; i)-1}(\lambda) = c_{m(\lambda; i)-1}(0)$ . This implies that  $c_{m(\lambda; i)-1}(0) < c_{m(\lambda; i)}(\lambda)$ . Since  $c_{m(\lambda; i)}(\lambda) \leq c_{m(\lambda; i)}(0)$ , it follows that  $c_{m(\lambda; i)-1}(0) < c_{m(\lambda; i)}(0)$ .

Now remark 3.2.3, together with (85) and the fact that the set  $\{m(\lambda; i), \dots, i-1\}$  is the union of blocks associated with  $C(0)$ , imply that

$$\sum_{k=m(\lambda; i)}^{i-1} V_{2,3,k} = 0. \quad (86)$$

Finally, consider the term  $W_{2,3,k}$ , defined in (83). Similarly to the proof of (73), there exists a constant  $K_{2,3} > 0$  such that for every  $k = m(\lambda; i), \dots, i-1$ ,

$$|W_{2,3,k}| \leq K_{2,3}(c_{k+1}(0) - c_k(0)). \quad (87)$$

But  $c_i(0) = t$  and  $c_{m(\lambda; i)}(0) \geq c_{m(\lambda; i)}(\lambda) \geq t - \lambda$ , and hence  $c_i(0) - c_{m(\lambda; i)}(0) \leq \lambda$ . Consequently, (and since  $c_i(0) - c_{m(\lambda; i)}(0) \geq 0$ ) we have that

$$\sum_{k=m(\lambda; i)}^{i-1} |W_{2,3,k}| \leq K_{2,3} \sum_{k=m(\lambda; i)}^{i-1} (c_{k+1}(0) - c_k(0)) = K_{2,3}(c_i(0) - c_{m(\lambda; i)}(0)) \leq K_{2,3}\lambda. \quad (88)$$

Putting it all together we obtain, by (84), (86), and (88), that

$$|U_{2,3}| \leq K_{2,3}\lambda. \quad (89)$$

Defining  $K_2 := K_{2,1} + K_{2,2} + K_{2,3}$ , Eqs. (75), (79), (80), and (89) imply that

$$|U_2| \leq K_2 \lambda. \quad (90)$$

In a similar way (see (72)), there exists a constant  $K_4 > 0$  such that,

$$|U_4| \leq K_4 \lambda. \quad (91)$$

At last, by (68), (73), (90), (74), and (91), we have that  $|h'(\lambda^+) - h'(0^+)| \leq (K_1 + K_2 + K_3 + K_4)\lambda$ . Defining  $K := K_1 + K_2 + K_3 + K_4$ , (66) follows.  $\blacksquare$

**Remark 3.3.1** *The main argument of the proof is based on the bounds derived in (73), (74), (79), (80), and (89). All but the latter one are applications of the Lipschitz continuity of  $x_\lambda(t)$  and  $p_\lambda(t)$  in  $\lambda$  and  $t$  (see Lemma 3.3.4 and Lemma 3.3.5). On the other hand, (89) is based on (86), and this equation is due to the particular starting point of the curve  $\{C(\lambda)\}$ , namely  $\hat{s}_{j+1}$ . Starting the curve from any other point generally would not yield an equation like (86) since the derivative term  $h'(\lambda)$  is discontinuous. Consequently, a descent algorithm may not yield sufficient descent, and Algorithm 3.2.1 might not converge in the sense of Section 3.1.*

There is another subtlety that is ironed out by the particular choice of the curve  $\{C(\lambda)\}$ . The crucial equation (86) is true regardless of the number  $i - m(\lambda; i)$ . A different descent curve, even though starting from  $\hat{s}_{j+1}$ , might have a positive but not zero upper bound on the term  $\sum_{k=m(\lambda; i)}^{i-1} V_{2,3,k}$  in the left-hand-side of (86). This bound may be “small” but grows with  $i - m(\lambda; i)$ . The latter number, in turn, may grow with the dimension of the curve,  $N(j+1)$ . This, in turn, could prevent the sufficient-descent property and hence the convergence of Algorithm 3.2.1. For this reason we were unable to prove convergence of a feasible gradient-descent algorithm in lieu of Procedure 3.3.1, nor do we believe that it is true. Descent we certainly could get but without its “sufficient” qualification, and this is due to the growing dimensions of the successive problems  $P_{\sigma_j}$ . The particular choice of the curve  $\{C(\lambda)\}$  not only provides a measure of continuity of the derivative  $h'(\lambda)$  at the starting point of the curve, but also avoids the problem associated with its growing dimensionality in successive iterations.

The next lemma pertains to the first step of Procedure 3.3.1. Recall that Algorithm 3.2.1 enters Procedure 3.3.1 with a point  $\hat{\xi}_{j+1}$  and Step 1 computes the point  $\bar{\xi}_{j+1}$ .

**Lemma 3.3.7** *For every  $\delta > 0$  there exists  $\eta > 0$  such that, if Algorithm 3.2.1 enters Procedure 3.3.1 with  $\hat{\xi}_{j+1}$  such that  $h'(0^+) < -\delta$ , then, for the computed point  $\bar{\xi}_{j+1}$ ,*

$$J(\bar{\xi}_{j+1}) - J(\hat{\xi}_{j+1}) \leq -\eta. \quad (92)$$

*Proof.* The proof is based on standard arguments in the analysis of gradient descent algorithms. Let Procedure 3.3.1 start at a point  $\hat{\xi}_{j+1}$ . Fix  $\delta > 0$  and suppose that  $h'(0^+) < -\delta$ . Recall  $\alpha \in (0, 1)$  as given by Procedure 3.3.1, and let  $K$  be the Lipschitz constant given by Lemma 3.3.6 (Eq. (66)). By the mean value theorem, for every  $\lambda \geq 0$  there exists  $\zeta \in \text{conv}\{h'(\tilde{\lambda}^+)|\tilde{\lambda} \in [0, \lambda]\}$  ( $\text{conv}(\cdot)$  indicates convex hull) such that  $h(\lambda) - h(0) = \zeta\lambda$ . Since by (66) we have that  $|\zeta - h'(0^+)| \leq K\lambda$ , we have that

$$h(\lambda) - h(0) \leq (h'(0^+) + K\lambda)\lambda. \quad (93)$$

Define  $\bar{\lambda} := (1 - \alpha)\delta/K$ . Since  $h'(0^+) < -\delta$  by assumption, it follows that for every  $\lambda \in [0, \bar{\lambda}]$ ,

$$h'(0^+) + K\lambda \leq -\delta + K\bar{\lambda} = -\delta + (1 - \alpha)\delta = -\alpha\delta < \alpha h'(0^+).$$

Therefore, and by (93), we have that  $h(\lambda) - h(0) \leq \alpha\lambda h'(0^+)$ . An examination of (53) reveals that  $\bar{\ell}$  is bounded from below by  $\hat{\ell} := \min\{\ell = 0, 1, \dots : \lambda(\ell) < \bar{\lambda}\}$ , and by (53) and the fact that the sequence  $\{\lambda(\ell)\}$  is monotone decreasing,  $h(\lambda(\ell)) - h(0) \leq \alpha\lambda(\hat{\ell})h'(0^+) \leq -\alpha\lambda(\hat{\ell})\delta$ . Defining  $\eta := \alpha\lambda(\hat{\ell})\delta$  and recalling the definition of  $\bar{s}_{j+1}$  (following (53)) and the fact that  $h(\lambda) := J(C(\lambda))$ , (92) follows. ■

Finally, we present the main result of the chapter.

**Theorem 3.3.1** *Suppose that Algorithm 3.2.1, with Procedure 3.3.1 for its Step 1, computes a sequence of schedules  $\{\xi_j\}_{j=1}^\infty$ . Then,*

$$\lim_{j \rightarrow \infty} D(\xi_j) = 0. \quad (94)$$



*Proof.* By (39) and Lemma 3.3.3,  $J$  is bounded from below. By Procedure 3.3.1, Algorithm 3.2.1 is a descent algorithm and hence  $J(\xi_{j+1}) \leq J(\xi_j)$  for all  $j = 1, 2, \dots$ . Fix  $\delta > 0$ . By Lemma 3.3.2 and Lemma 3.3.7 there exists  $\eta > 0$  such that, for every  $j = 1, 2, \dots$ , if  $D(\xi_j) < -\delta$  then  $J(\xi_{j+1}) - J(\xi_j) \leq -\eta$ . Consequently Algorithm 3.2.1 has the sufficient descent property with the optimality function  $\theta(\xi_j) := D(\xi_j)$ , and by Proposition 3.1.1, (94) is satisfied. ■

Having presented Theorem 3.3.1, we have proved that Algorithm 3.2.1 converges in the sense we defined it.

### 3.4 Numerical Examples

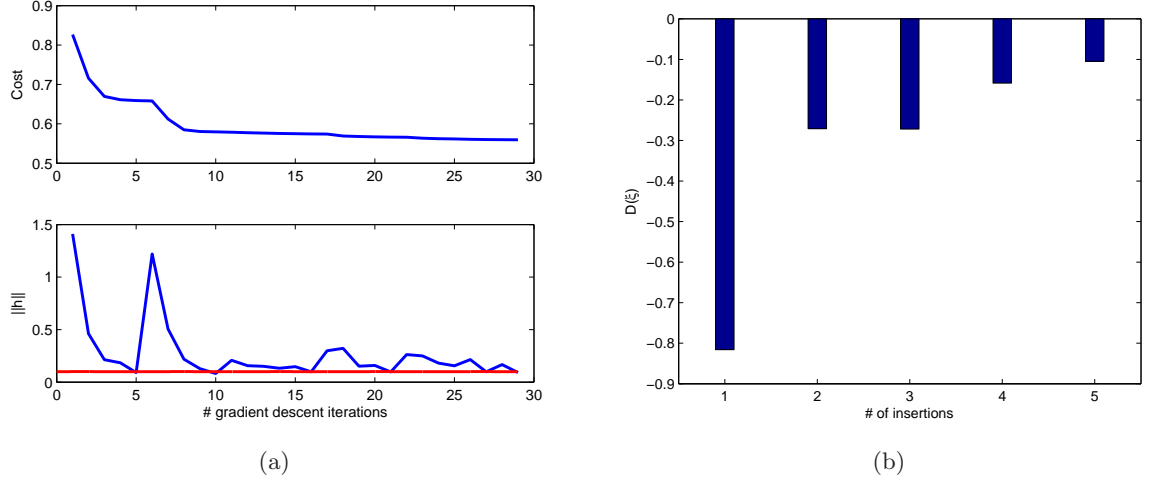
In order to show the utilization of the proposed framework, we apply it to two different problems. First, we will see how the cost can be reduced for the linear system, presented in the example in Chapter II, by inserting new modes to the current modal structure. Then, a more complex nonlinear system is considered.

#### 3.4.1 Linear System

In Chapter II, a linear system that switched between three different modes was presented. The system was initialized with each mode active one-third of the time. After we optimized the system with respect to the two switching times that corresponded to the above mode sequencing, we got a final cost of 0.66.

We will now perform five insertions and see how the cost and the modal structure changes. An insertion is performed whenever  $\|h\| < 0.1$ . The results obtained when running Algorithm 3.2.1 for five insertions, i.e., we execute Step 1 through 5 five times, are presented in Figure 11. From Figure 11 is clear that the cost is reduced by the insertion of new modes. Furthermore, it can be seen that the reduction in cost decreases as  $D(\xi)$  gets smaller. An insertion is performed at iteration 5, 10, 16, 21, and 27 as can be seen by the increase of  $\|h\|$ , in Figure 11(a), at those iteration points.

The final schedule is given by  $\sigma = \{1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 3\}$  with  $\tau = \{0, 0.034, 0.129, 0.264, 0.372, 0.372, 0.380, 0.386, 0.501, 0.634, 0.806, 0.877, 1.000, 1\}$ , where we see that mode

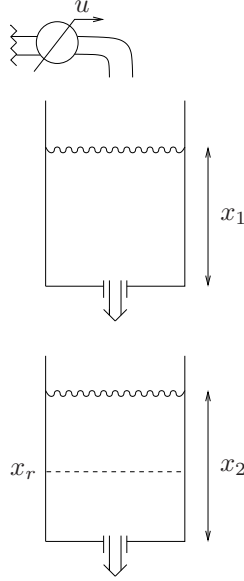


**Figure 11:** (a) Cost and norm of search direction as a function of the number of gradient descent steps. The red line in the bottom plot corresponds to  $\|h\| = 0.1$ . (b)  $D(\xi)$  every time we perform an insertion

3 is not active. In total we have 12 switches, but the last switch corresponds to a mode with zero duration. From this we conclude that it is optimal to use mode 1 and mode 2 interchangeably.

### 3.4.2 Nonlinear System

We consider the problem of controlling the fluid level in a tank by adjusting the input flow rate at an auxiliary tank, as shown in Figure 12. This problem was initially considered in [50]. There, two different controllers were designed, one PID and one time-optimal controller, and a state-based switching strategy for when to switch between the two controllers, in order to control the fluid level in the second tank, was presented. Although we consider the same system, we will approach the problem from the standpoint of our optimization framework and find a good modal sequence and the corresponding switching time vector. As can be seen in the figure, fluid from the upper tank flows to the lower tank through a valve, and fluid from the lower tank flows out through another valve. From Torricelli's principle it follows that fluid is discharged from either tank at a rate that is proportional to the square-root of the fluid level in the tanks. Denoting by  $x_1$  and  $x_2$  the fluid levels in the upper and the lower tank respectively, there exist two positive constants  $\alpha_1$  and  $\alpha_2$



**Figure 12:** The double tank process.

such that the flow rate between the two tanks is  $\alpha_1\sqrt{x_1}$ , and the flow rate out of the lower tank is  $\alpha_2\sqrt{x_2}$ . Suppose that the control parameter is the inflow rate to the upper tank, denoted by  $u$ . Then, the state  $x(t)$ , defined by  $x(t) = (x_1(t), x_2(t))^T$ , evolves according to the following differential equation,

$$\dot{x} = f(x, u) = \begin{bmatrix} -\alpha_1\sqrt{x_1} + u \\ \alpha_1\sqrt{x_1} - \alpha_2\sqrt{x_2} \end{bmatrix}. \quad (95)$$

Furthermore, suppose that the input valve can be in either one of three states: closed, half-open, or fully open. Correspondingly, the input flow rate  $u(t)$  attains one of the following three values, 0,  $0.5u_{max}$ , and  $u_{max}$ , for some given  $u_{max} > 0$ . Corresponding to the three values of  $u(t)$  we denote the right-hand side of (95) by  $f_1(x)$  when  $u = u_{max}$ , by  $f_2(x)$  when  $u = 0.5u_{max}$ , and by  $f_3(x)$  when  $u = 0$ . Hence,  $\Phi = \{f_1, f_2, f_3\}$ , and we assume that there are no constraints on the order in which the different modes can appear in.

Given an initial state  $x(0)$  and a final time  $T > 0$ , the objective of our switching-control strategy is to have the fluid level in the lower tank track a given reference value, denoted by  $x_r(t)$ . Thus, the performance functional that we minimize is

$$J = K \int_0^T (x_2(t) - x_r(t))^2 dt, \quad (96)$$

for a suitable constant  $K > 0$ .

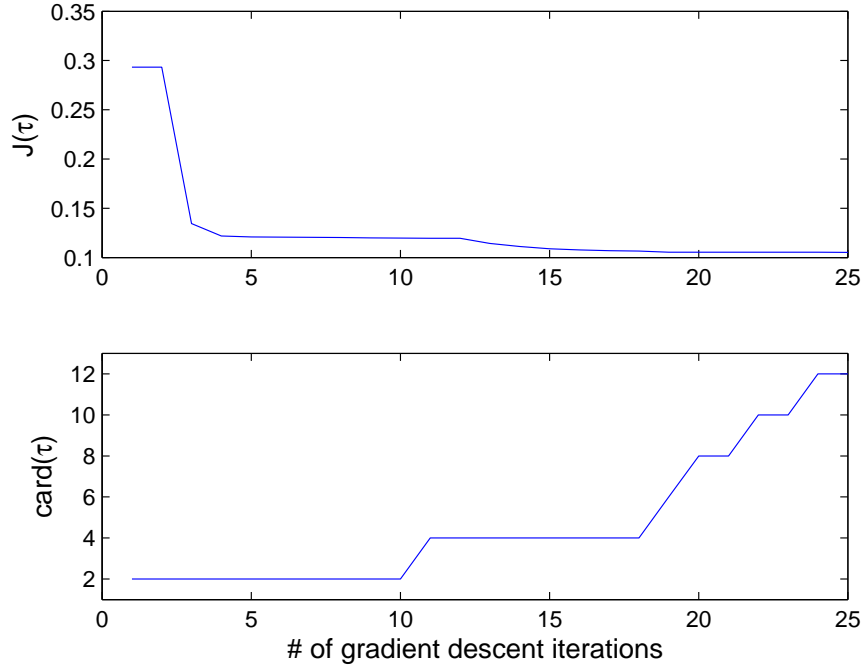
In order to optimize both over the modal sequence  $\sigma$  and the switching time vector  $s$ , Algorithm 3.2.1. was employed starting at  $\sigma_0 = \{f_1\}$ , with  $t_0 = 0$  and  $T = 5$ .

Since  $\sigma_0$  consists of only one mode, the algorithm goes directly to Step 2 in its first iteration. Entering Step 2 of the algorithm, we search to see if it is beneficial to perform an insertion. Note that we could have inserted several new modes, but for the sake of the simplicity of our exposition, we only consider inserting one new mode. To this end, we evaluate (44) and let  $(g, t) \in \Phi \times [0, T]$  be an argmin of (44). Furthermore, we only perform an insertion if  $D$ , as specified in (44), satisfies  $D < -\omega$  for a positive constant  $\omega$ , where we choose  $\omega = 0.01$ . If  $D < -\omega$ , we add the two new switching times at time  $t$  and the modal function  $g$  between them, we update  $\sigma$  and  $s$  and go back to Step 1. On the other hand, if  $D \geq -\omega$ , we consider ourselves done and terminate the algorithm.

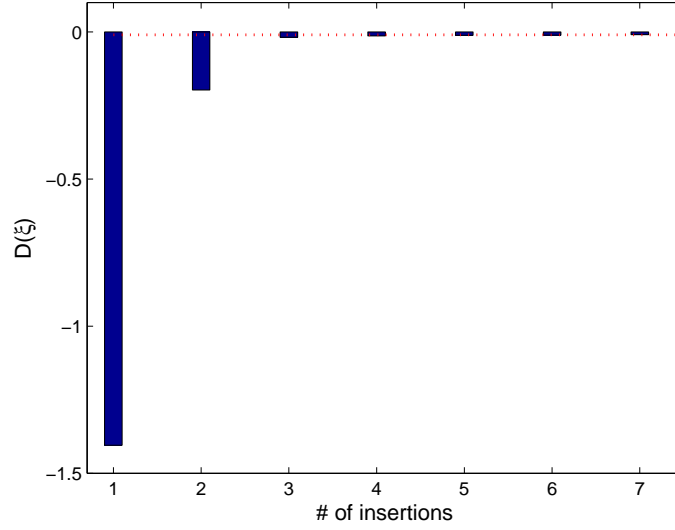
When the algorithm enters Step 1, it optimizes over  $s$  using a gradient-descent algorithm with Armijo step-size [6]. This choice of step-size guarantees, asymptotically, that the projected gradients of the iteration sequence converge to 0 (see [56]), but for our problem, we stop its execution once the magnitude of the projected gradient is smaller than a given positive constant  $\epsilon$ . In our case we use  $\epsilon = 0.01$ , and hence, we enter Step 2 of Algorithm 3.2.1 when the norm of the projected gradient falls below 0.01.

An execution for our problem when  $K = 10$ ,  $\alpha_1 = \alpha_2 = u_{max} = 1$ , and we want to track  $x_r(t) = 0.5 + 0.25\frac{t}{T}$  is depicted in Figures 13 -16. Figure 13 shows the cost and the number of switches as functions of the number of gradient descent steps. As can be seen from Figure 13, Algorithm 3.2.1 effectively reduces the cost by introducing new modes and optimizing over the switching times.

Figure 14 shows the value of  $D(\xi)$ , as defined in (44), as function of the number of insertions we have performed, together with  $-\omega$  (dotted). When the algorithm is about to perform its 7th insertion,  $D(\xi)$  is above  $-\omega$  and we consider ourselves done. Figure 15 plots the final state trajectories together with the reference signal. Finally, the modal schedule at the point when the algorithm has terminated is plotted in Figure 16. There we see that mode 1 is active between 0 and 1.75 and hence the system evolves according to  $\dot{x} = f_1(x)$

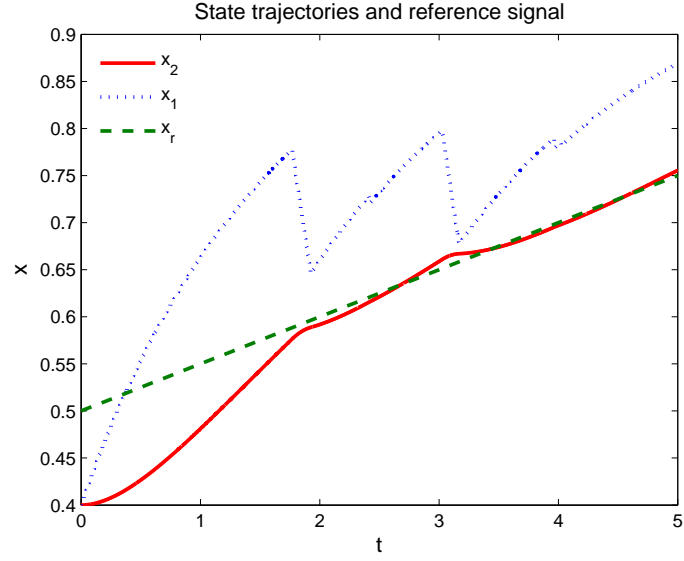


**Figure 13:** Cost together with the dimension of the switching time vector as a function of the number of gradient descent iterations.

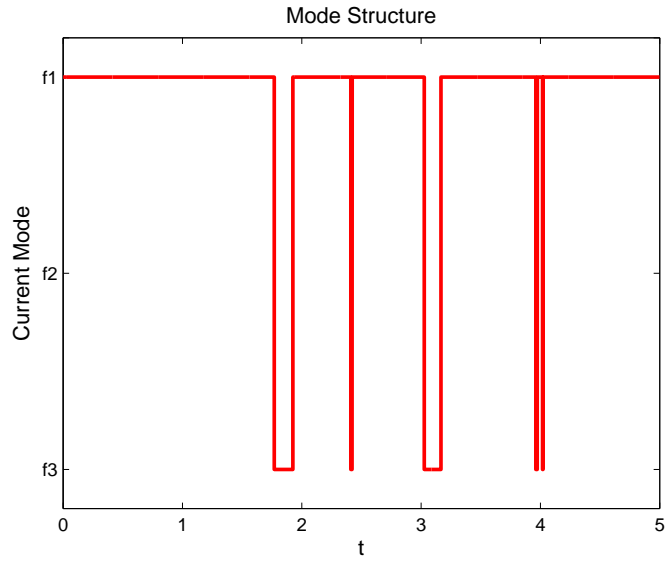


**Figure 14:**  $D(\xi)$  every time we perform an insertion together with  $-\omega$  (dotted).

between those times. At time 1.75 it then switches to mode 3 and evolves according to  $\dot{x} = f_3(x)$  for a short period of time, and so on. It can be seen that mode 1 is active 93%



**Figure 15:** Final state trajectories:  $x_1(t)$  dotted,  $x_2(t)$  solid, and  $x_r(t)$  dashed.



**Figure 16:** Mode structure for the final schedule.

of the time, mode 2 is inactive, and mode 3 is active 7% of the time.

It should be noted that if we start with  $\sigma_0 = \{f_2\}$ , instead of  $\sigma_0 = \{f_1\}$  as above, the final costs when the optimization algorithm terminates are close to being the same. In fact, the final cost when the algorithm started with  $\sigma_0 = \{f_1\}$  was 0.105 and the final cost when the algorithm started with  $\sigma_0 = \{f_2\}$  was 0.107. Finally, if we start with  $\sigma_0 = \{f_1, f_2\}$ ,

each being active half the time, the final cost is 0.106. Hence, in our case the final cost is about the same, independent of what mode structure we start with.

### ***3.5 Conclusions***

This Chapter presented a gradient descent approach to optimal mode scheduling in hybrid dynamical systems. Based on the derivative formula presented in Chapter II, it was shown how the cost can be reduced by inserting new modes into the current modal schedule for an infinitesimal amount of time. Once a new mode has been inserted, the algorithm presented in Chapter II, for optimizing over the modal sequence, was used along a descent curve in order to decrease the cost. Furthermore, convergence of the algorithm presented in this chapter to a local minimum was defined, and it was proven that the algorithm had the property of sufficient descent which in turn implied that the algorithm is provably convergent. In particular, convergence meant that at a local minimum one can not insert a new mode for an infinitesimal amount of time and be guaranteed a reduction in cost by optimizing over the corresponding switching time vector. Two examples testified to the usefulness of the algorithm.

In summary, the results presented in this Chapter give an algorithmic solution to the mode scheduling problem.

## CHAPTER IV

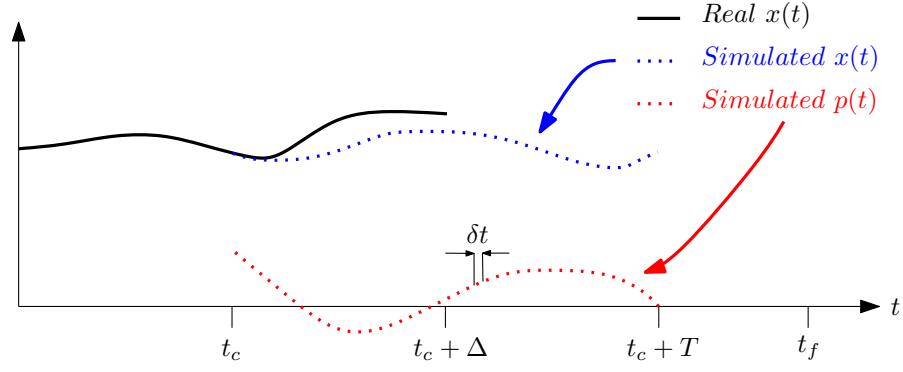
### REAL TIME SWITCHING TIME CONTROL

This Chapter concerns the derivation of a control strategy for how to optimize switched autonomous systems in real time. In particular, it is assumed that the mode sequence is given and fixed and the problem under consideration is to optimize the switching times in real time. The optimization will be performed through a gradient descent algorithm (as derived in Chapter II). However, due to the computational constraint that the controller only has a certain time available before it must deliver updated control values, exact solutions will not be available. Therefore, there is a trade-off between the precision of the solution obtained in real time and the computation horizon. It is this trade-off that will be studied in detail and a solution will be presented based on minimizing a conservative bound between the true gradient and the gradient calculated in real time.

In particular, assume that one want to optimize the switching times of a switched autonomous system with initial time  $t_0$  and final time  $t_f$ . To this end, one could calculate the state forward in time between 0 and  $t_f$ , and the costate backward between  $t_f$  and 0, using Euler's method with step size  $\delta t$ . Given the simulation horizon  $t_f$ , the step size would then be determined by the time the controller has available to it before it must deliver a control value. Now, instead of calculating the state and costate trajectories over  $(0, t_f)$ , we will see how the performance of the gradient descent algorithm changes if we calculate them over a shorter time interval, i.e. over  $(0, T)$  for some  $T < t_f$ . Note that this implies that we will start calculating the costate backward from time  $T$ . This will allow us to get a smaller  $\delta t$  potentially rendering better performance of our gradient descent algorithm. Characterizing the trade-off between  $T$  and  $\delta t$  is the main subject of this Chapter.

Already at this point it should be mentioned that there are two time horizons for this problem, the “real” time horizon that the system operates in, and the “simulation” time horizon used for simulating  $x$  and  $p$  over. This is illustrated in Figure 17 where a system





**Figure 17:** Simulated time horizon versus real time horizon: While the system evolves for  $\Delta$  seconds, starting from time  $t_c$ ,  $x(t)$  and  $p(t)$  are simulated between  $t_c$  and  $t_c + T$  with step-length  $\delta t$ , using Euler's method to solve for  $x$  and  $p$ .

runs for  $\Delta$  seconds while simulating  $x$  and  $p$ ,  $T$  seconds into the future. Once  $\Delta$  seconds have passed, the system do the same thing over again but start the calculation of  $x(t)$  from  $x(t_c + \Delta)$ .

The outline of this Chapter is as follows: In Section 4.1, the switching time optimization problem is presented and some initial results are given. Moreover, the real time approach to solving this problem is presented. In particular, we will define the performance metric in terms of bounds on the error in the gradient as a function of the solution horizon and the numerical precision. In order to compute these bounds the behavior of the state and costate equations must be characterized, which is the topic of Section 4.2. This characterization is then put to use in Section 4.3 for optimizing the aforementioned performance metric, followed by a robot navigation example (Section 4.4) in which an autonomous mobile robot must switch between different modes of operation (or behaviors) in real time in order to navigate an unknown area. Finally, conclusions are given in Section 4.5.

## 4.1 Problem Formulation

In order to make the chapter self-containing, a few results from Chapter II are repeated below.

Let  $\{x(t)\}_{t=0}^{t_f}$  denote the state trajectory of the system, and suppose that it evolves according to the equation  $\dot{x} = f(x, t)$ , where the dynamic-response function  $f : \mathbb{R}^n \times [0, t_f] \rightarrow$

$\mathbb{R}^n$  whose temporal dependence is defined through a sequential assignment of functions  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,  $i = 1, 2, \dots$ . Let us fix  $t_f > 0$  and suppose that the dynamic response changes  $N$  times in the interval  $[0, t_f]$ . Denoting the switching times by  $\tau_i$ ,  $i = 1, \dots, N$ , in increasing order, and defining  $\tau_0 := 0$  and  $\tau_{N+1} := t_f$ , we have that  $0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = t_f$ . Furthermore, suppose that  $f(x, t) = f_i(x)$  for every  $t \in [\tau_{i-1}, \tau_i)$  and for every  $i = 1, \dots, N+1$ . We then have the following differential equation defining the system's dynamics,

$$\dot{x}(t) = f_i(x(t)), \quad t \in [\tau_{i-1}, \tau_i), \quad i \in \{1, \dots, N+1\}. \quad (97)$$

We assume throughout that the initial condition  $x(0)$  is given and fixed.

The optimal control problem considered is that of optimizing the switching times, denoted in vector form by  $\bar{\tau} = (\tau_1, \dots, \tau_N)$ , in order to minimize a cost-function  $J$  over a fixed time interval:

$$\begin{aligned} P: \quad & \min_{\bar{\tau}} J(\bar{\tau}) = \int_0^{t_f} L(x(t)) dt \\ & s.t. \quad \dot{x}(t) = f_i(x(t)), \quad t \in [\tau_i, \tau_{i+1}), \end{aligned}$$

where  $L$  is the instantaneous cost. An expression for the gradient of  $J$  was presented in Chapter II, and for the sake of completeness, we recall it below. The derivative of the cost-functional with respect to a switching time  $\tau_i$ , where  $i \in \{1, \dots, N\}$ , is given by

$$\frac{\partial J}{\partial \tau_i}(\bar{\tau}) = p(\tau_i)^T (f_{i-1}(x(\tau_i)) - f_i(x(\tau_i))), \quad (98)$$

where  $p(t) \in \mathbb{R}^n$  is the costate and satisfies the following differential equation,

$$\dot{p} = - \left( \frac{\partial f_{i+1}}{\partial x}(x(t)) \right)^T p - \left( \frac{\partial L}{\partial x}(x(t)) \right)^T, \quad t \in (\tau_i, \tau_{i+1}], \quad (99)$$

for  $i \in \{1, \dots, N\}$  with the final condition  $p(t_f) = 0$ . Once the state trajectory  $x(t)$  and the costate trajectory  $p(t)$  are calculated, (98) is easily evaluated for all switching times.

The problem considered in this Chapter is that of solving problem  $P$  under the real time constraint that we only have a finite amount of time available to calculate  $x(t)$ ,  $p(t)$  and to evaluate the gradient before we need to update the switching times. The calculation of  $x(t)$  and  $p(t)$  is done through the Euler method with step-length  $\delta t$ . Assume that the current

time is  $t_c$ , where  $t_c \in [0, t_f]$ , and that we simulate  $T \in (0, t_f - t_c]$  seconds into the future using a step-length of  $\delta t$  seconds. The real time constraint considered in this Chapter has the following form

$$g(T, \delta t) \leq \Delta, \quad (100)$$

where  $\Delta$  is the time we are allowed to simulate before we must update our switching time vector, and  $g$  is the time it takes to simulate  $T$  seconds with a step-length of  $\delta t$ . We will refer to  $g$  as the *simulation-time* function. In general  $g : \mathbb{R}^2 \rightarrow \mathbb{R}^+ \cup \{0\}$  will depend on the processor speed and it will be an increasing function in  $T$ , and decreasing in  $\delta t$ . We make the following simplifying assumption regarding the simulation-time function  $g$ :

**Assumption 3** *The simulation-time function is given by*

$$g(T, \delta t) = C_{CPU} \frac{T}{\delta t}, \quad (101)$$

where  $C_{CPU}$  is a constant relating the CPU speed to the numerical simulation time.

At this point it should be mentioned that we assume that the state of the system is observable. This implies that every  $\Delta$  seconds we can start the numerical integration of  $x(t)$  using  $x$  at that time as the initial state. For example, assume that we were at time  $\Delta$  and we want to calculate  $x(t)$  forward in time. Then, we initialize our numerical integration with  $x(\Delta)$  and evaluate  $x(t)$  using Euler's method. Note that there are two time variables here, the real time that the system is operating in, and the simulation horizon  $T$  that the system calculates  $x$  and  $p$  over in order to get an expression for the gradient.

Assuming we are given the dynamics, i.e., the sequence of  $f'_i$ s and the corresponding switching time vector  $\bar{\tau} = (\tau_1, \dots, \tau_N)$ , our problem is then to solve  $P$  by updating the switching times every  $\Delta$  seconds while choosing  $T$  and  $\delta t$  to satisfy the real time constraint (100), where  $g$  is given by (101). The switches are updated through a gradient descent algorithm, with step-size  $\gamma$ , using the calculated gradient denoted by  $\nabla \tilde{J}$ .

The calculated gradient is given by evaluating (98), where  $x(t)$  and  $p(t)$  have been replaced by the state and costate trajectories obtained when solving (97) and (99) numerically. This is done through a numerical integration  $T$  seconds into the future using a step-length

of  $\delta t$  seconds, as mentioned earlier. Hence the calculated gradient  $\nabla \tilde{J}$ , at any given time, will depend on  $T$ ,  $\delta t$  and  $\bar{\tau}$ . Furthermore, the calculated gradient will also depend on the current time  $t_c$  as the state is updated every  $\Delta$  seconds, thereby changing the initial condition in the calculation of the state trajectory. However, the dependence on  $t_c$  in  $\nabla \tilde{J}$  will be suppressed for notational convenience whenever it is clear from the context.

Having motivated why the calculated gradient depends on  $T$  and  $\delta t$ , our approach to optimize the cost in real time can be presented. As mentioned earlier, we will optimize the cost in real time through a gradient descent algorithm, with fixed step-size  $\gamma$ , using the calculated gradient. We want the difference in cost between updating the switching times with the true gradient as compared to updating them with the calculated gradient, to be as small as possible. This formulation allows us to minimize the cost with respect to  $T$  and  $\delta t$ . Hence, we will choose  $T$  and  $\delta t$  to minimize the following term every  $\Delta$  seconds:

$$J_{diff}(\bar{\tau}, T, \delta t) = |J(\bar{\tau} - \gamma \nabla J(\bar{\tau})) - J(\bar{\tau} - \gamma \nabla \tilde{J}(\bar{\tau}, T, \delta t))|, \quad (102)$$

subject to (100).

Based on the second order Taylor expansion of the two terms in (102), together with the mean-value theorem, we get that

$$J(\bar{\tau} - \gamma \nabla J(\bar{\tau})) = J(\bar{\tau}) - \gamma \langle \nabla J(\bar{\tau}), \nabla J(\bar{\tau}) \rangle + \frac{\gamma^2}{2} \nabla J^T(\bar{\tau}) \cdot \frac{\partial^2 J}{\partial \bar{\tau}^2}(d) \cdot \nabla J(\bar{\tau}), \quad (103)$$

for some vector  $d$  on the line segment between  $\tau$  and  $\tau - \gamma \nabla J(\bar{\tau})$ . Likewise, for the second term of (102), we get that

$$\begin{aligned} J(\bar{\tau} - \gamma \nabla \tilde{J}(\bar{\tau}, T, \delta t)) = & J(\bar{\tau}) - \gamma \langle \nabla \tilde{J}(\bar{\tau}, T, \delta t), \nabla J(\tau) \rangle + \\ & + \frac{\gamma^2}{2} \nabla \tilde{J}^T(\bar{\tau}, T, \delta t) \cdot \frac{\partial^2 J}{\partial \bar{\tau}^2}(d') \cdot \nabla \tilde{J}(\bar{\tau}, T, \delta t), \end{aligned} \quad (104)$$

for some vector  $d'$  on the line segment between  $\bar{\tau}$  and  $\bar{\tau} - \gamma \nabla \tilde{J}(\bar{\tau}, T, \delta t)$ . In order to proceed, some regularity assumptions on the state dynamics and the instantaneous cost are needed:

#### Assumption 4

(i). The functions  $f_i$ ,  $i \in \{1, \dots, N\}$ , and the instantaneous cost  $L$ , are twice continuously differentiable on  $\mathbb{R}^n$ .

(ii). There exists a constant  $K > 0$  such that, for every  $x \in \mathbb{R}^n$ , and for every  $i \in \{1, \dots, N\}$ ,

$$\|f_i(x)\| \leq K(\|x\| + 1). \quad (105)$$

It should be mentioned that a similar assumption was made in Chapters II and III. Assumption 4 guarantees that both  $f(x, t)$  and  $\frac{\partial f(x, t)}{\partial x}(x(t))$  are Lipschitz with Lipschitz constants  $L_f$  and  $L_{f'}$ . The same is true for  $L(x(t))$  and  $\frac{\partial L}{\partial x}$ , with Lipschitz constants  $L_L$  and  $L_{L'}$ .

Given Assumption 4, it was shown in Chapter II that both  $J(\bar{\tau})$  and  $\nabla J(\bar{\tau})$  are Lipschitz continuous in  $\bar{\tau}$  with Lipschitz constants  $L_J$  and  $L_{J'}$  and that  $\|\nabla J(\bar{\tau})\|$  is bounded from above by some constant  $D$ . It then follows that the second derivative is bounded by some constant  $K_{J^2}$ , and the last term of (103) and (104) are bounded by  $\gamma^2 C_1$ , where  $C_1 = K_{J^2} D^2 / 2$ . Summarizing, it follows that

$$J_{diff}(\bar{\tau}, T, \delta t) = \left| J(\bar{\tau}) - \gamma \|\nabla J(\bar{\tau})\|^2 - J(\bar{\tau}) + \gamma \langle \nabla J(\bar{\tau}), \nabla \tilde{J}(\bar{\tau}, T, \delta t) \rangle \right| + o(\gamma^2) C_1, \quad (106)$$

where  $o(\gamma^2)$  represents little  $o$  in the normal way, i.e.  $\frac{o(\gamma^2)}{\gamma} \rightarrow 0$  as  $\gamma \rightarrow 0$ . Assuming we pick a small step-size,  $\gamma$ , we can approximate (102) by the following equation,

$$\begin{aligned} J_{diff}(\bar{\tau}, T, \delta t) &\approx \gamma \left| \langle \nabla J(\bar{\tau}), \nabla J(\bar{\tau}) - \nabla \tilde{J}(\bar{\tau}, T, \delta t) \rangle_Q \right| \\ &\leq \gamma \|\nabla J(\bar{\tau})\|_Q \|\nabla \tilde{J}(\bar{\tau}, T, \delta t) - \nabla J(\bar{\tau})\|_Q, \end{aligned} \quad (107)$$

where the last step follows from Cauchy-Schwartz inequality, and the norm  $\|\cdot\|_Q$  is a function of the switching vector and the current time  $t_c$ , to be defined in Section 4.3.

Considering the right hand side of (107), we see that the only term that can be controlled explicitly is  $\left\| \nabla \tilde{J}(\bar{\tau}, T, \delta t) - \nabla J(\bar{\tau}) \right\|_Q$ . This, since our control strategy is to minimize the difference between  $\nabla J$  and  $\nabla \tilde{J}$  by choosing  $T$  and  $\delta t$ , hence we do not explicitly control  $\bar{\tau}$ . Therefore, we will minimize (102) by minimizing the supremum of the above term over  $T$  and  $\delta t$ . We thus have the following problem:

$$\begin{aligned} P_{RT} \quad & \min_{T, \delta t} \left\| \nabla \tilde{J}(\bar{\tau}, T, \delta t) - \nabla J(\bar{\tau}) \right\|_Q, \\ \text{s.t.} \quad & g(T, \delta t) \leq \Delta, \end{aligned}$$

where the subscript  $RT$  denotes "real time" and  $g$  is given in (101).

In order to solve problem  $P_{RT}$ , an expression for the upper bound of the norm of the error between the simulated state/costate and the true state/costate will be derived as a function of  $T$  and  $\delta t$ . To this end, Bellman-Grönwall's Lemma (see [56]) is used. Bellman-Grönwall's Lemma states that  $\|x(t)\|$ ,  $t \in (0, t_f)$  is bounded above by  $(\|x_0\| + Kt_f)e^{Kt_f}$ , where  $K$  is as in Assumption 4. Using Assumption 4 again, we get that  $\|f\| \leq C_f$ , where  $C_f := K(\|x_0\| + Kt_f)e^{Kt_f} + 1$ . We are now in position to derive the upper bound on the error between the state  $x$  and the simulated state, denoted by  $x_s$ . Before doing that we note that there exists a bounded compact set  $X$  such that  $x(t) \in X$ ,  $\forall t \in [0, t_f]$  since  $x(t)$  is bounded.

## 4.2 Bounds on the Error In the State and Costate

The simulated state, obtained by solving (97) numerically, is denoted by  $x_s[t, \delta t]$ . Here,  $\delta t$  is the time step between each evaluation of (97) and  $t$  is the time variable. As mentioned earlier, we solve for  $x_s$  using Euler's method. More elaborate methods can be imagined, e.g., Runge-Kutta's method [36], but for the sake of simplicity, we will only consider Euler's method. Starting at time 0,  $x_s$  is initialized to be equal to  $x_0$ , then, for  $\Delta$  seconds, the real time process calculates the state and costate trajectories and then evaluates the calculated gradient in order to update the switching times. After  $\Delta$  seconds, the process updates  $x_s$  according to  $x_s[\lfloor \Delta \rfloor, \delta t] = x(\Delta)$  and repeat the above steps for  $\Delta$  seconds. Here, and in the subsequent part of this paper,  $\lfloor \Delta \rfloor$  denotes the closest multiple of  $\delta t$  smaller than or equal to  $\Delta$ . Defining the set  $M(T, \delta t) := \{0, \delta t, 2\delta t, \dots, \lfloor T \rfloor - \delta t, \lfloor T \rfloor\}$ , we can then define the least upper bound of the norm of the error between the real state  $x$  and the simulated state  $x_s$ , at a time  $t \in M(t_f, \delta t)$ , to be the function  $E[t, \delta t]$ , where  $E : t \times \delta t \rightarrow \mathbb{R}_+$ . The following proposition gives an upper bound on  $E$  that will be used when solving  $P_{RT}$ .

**Proposition 4.2.1** *Assume we are given a switching vector  $\bar{\tau} \in \mathbb{R}^N$ , and the associated dynamic response functions  $f_i$ ,  $i \in \{1, 2, \dots, N+1\}$ , where each  $f_i$  satisfies Assumption 4. Let the state trajectory  $x$  be given by solving (97) with a fixed initial condition,  $x(0) = x_0$ . Let  $x_s[t, \delta t]$  denote the simulated state at time  $t \in M(t_f, \delta t)$  obtained through solving (97)*

using Euler's method, with step-size  $\delta t$ , starting at  $x_s[0, \delta t] = x_0$ . Let  $E(t, \delta t)$  be the least upper bound on the norm of the error between  $x$  and  $x_s$  at time  $t \in M(t_f, \delta t)$ . Then the following inequality holds:

$$\|x(t) - x_s[t, \delta t]\| \leq E(t, \delta t) \leq (e^{L_f t} - 1) C_f \delta t. \quad (108)$$

**Proof.** See the Appendix.

In order to solve  $P_{RT}$ , a bound on the norm of the error between the real costate  $p(t)$  and the simulated costate, denoted by  $p_s[t, T, \delta t]$  and calculated through Euler's method, must be derived. Regarding the error in the simulated costate, there are three sources of errors described below:

1. Firstly, according to (99),  $\dot{p}$  depends on  $x(t)$ , in our case we only have access to  $x_s[t, \delta t]$ ,  $\forall t \in M(t_f, \delta t)$ .
2. Secondly,  $p_s[t, T, \delta t]$  is calculated using Euler's method.
3. Finally, if  $t_c + T < t_f$  then we do not have access to the state after time  $t_c + T$ , hence we can not use (99) to solve for  $p_s$  between  $[t_c + T, t_f]$  as we need access to  $x_s$  in that interval.

We start by assuming that the simulation length  $T$  satisfies  $t_c + T = t_f$  and derive the error due to the first two cases described above. To this end we note that, by Assumption 4, there exist two constants  $C_{L'}$  and  $C_{f'}$  such that  $\|\frac{dL}{dx}\| \leq C_{L'}$ , and  $\|\frac{df}{dx}\| \leq C_{f'}, \forall x \in X$ . These two bounds will give us the bounds on  $\|p\|$  and  $\|\dot{p}\|$  derived below,

$$\|p(t)\| = \left\| \int_t^{t_f} \left( \frac{df(x(s), s)^T}{dx} p(s) + \frac{dL(x(s))^T}{dx} \right) ds \right\| \leq (t_f - t) C_{L'} + \int_t^{t_f} C_{f'} \|p(s)\| ds \quad (109)$$

Eq. (109) together with Bellman-Gronwalls lemma gives the following bound for  $\|p(t)\|$  for all  $t \in (0, t_f)$ ,

$$\|p(t)\| \leq (t_f - t) C_{L'} e^{C_{f'}(t_f - t)} \leq t_f C_{L'} e^{C_{f'} t_f}, \quad (110)$$

where we define the right hand side of (110) as  $C_p$  and trivially deduce the bound of  $\|\dot{p}\|$  from (99) to be  $C_{\dot{p}} := C_{L'} + C_{f'} C_p$ . As done above for the error between  $x$  and  $x_s$ , we define the least upper bound of the norm of the error between the real costate  $p$  and the

simulated costate  $p_s$ , at a time  $t \in M(t_f, \delta t)$ , to be the function  $\tilde{E}[t, T, \delta t]$ . The following proposition gives an upper bound on  $\tilde{E}$  that will be used when solving  $P_{RT}$ .

**Proposition 4.2.2** *Assume we are given a switching vector  $\bar{\tau} \in \mathbb{R}^N$ , the associated dynamic response functions  $f_i$ ,  $i \in \{1, 2, \dots, N+1\}$ , and a cost function  $L$ , where each  $f_i$  and  $L$  satisfies Assumption 4. Let the costate trajectory  $p$  be given by (99) with a fixed final condition,  $p(t_f) = 0$ . Let the simulation length  $T$  satisfy  $T + t_c = t_f$ , i.e., we simulate until the final time  $t_f$ . Let  $p_s[t, T, \delta t]$  denote the simulated costate at time  $t \in M(t_f, \delta t)$  obtained through solving (99) backwards using Euler's method with step-size  $\delta t$  starting at  $p_s[t_f, t_f, \delta t] = 0$ . Let  $\tilde{E}(t, T, \delta t)$  denote the least upper bound of the norm of the error between the real costate  $p$  and the simulated costate  $p_s$ , at a time  $t \in M(t_f, \delta t)$ . Then the following inequality holds:*

$$\|p(t) - p_s[t, t_f, \delta t]\| \leq \tilde{E}(t, t_f, \delta t) \leq S_1(\delta t)^2 + S_2\delta t, \quad (111)$$

where the system constants,  $S_1$  and  $S_2$ , are given by

$$S_1 := e^{C_{f'}t_f} [L_{L'}(e^{L_f t_f} - 1)C_f + C_f) + 2C_{f'}C_{\dot{p}}], \quad (112)$$

$$S_2 := [L_{L'}C_f + C_fC_{\dot{p}} + C_pC_{f'}C_f + (L_{L'} + C_pL_{h'}) \times (e^{L_f t_f} - 1)C_f] \frac{e^{C_{f'}t_f} - 1}{C_{f'}}. \quad (113)$$

**Proof.** See the Appendix.

If we are currently at time  $t_c \in (0, t_f)$  when a simulation started, then by replacing  $t_f$  by  $t_f - t_c$  in the expressions for  $S_1$  and  $S_2$ , we get a bound for the error in the costate that reflects the current time.

Regarding the error due to the third source of error described earlier, when we do not simulate all the way until  $t_f$ , we set  $p_s[t, T, \delta t] = 0$ ,  $\forall t \in (t_c + T, t_f)$ , and integrate backward in time starting from  $t_c + T$ . The error due to this can be found by simply noting that by (110),  $\|p(t_c + T)\| \leq (t_f - (t_c + T))C_{L'}e^{C_{f'}(t_f - (t_c + T))}$  and therefore,

$$\|p_s[t, T, \delta t] - p(t)\| \leq (t_f - (t_c + T))C_{L'}e^{C_{f'}(t_f - (t_c + T))}, \quad (114)$$

for all  $t \in [t_c + T, t_f]$ . The total error in  $\|p - p_s\|$  is bounded above by the sum of the two errors presented in (111) and (114). This results in the following upper bound for  $\tilde{E}(t, T, \delta t)$



for a given step size  $\delta t$  and a given simulation length  $T$  and  $\forall t \in \{(t_c, t_f) \cap M(t_f, \delta t)\}$ .

$$\tilde{E}(t, T, \delta t) \leq S_1(\delta t)^2 + S_2\delta t + (t_f - (t_c + T))C_{L'}e^{C_{f'}(t_f - (t_c + T))}, \quad (115)$$

where  $S_1$  and  $S_2$  are given by (112) and (113) respectively, possibly replacing  $t_f$  with  $t_f - t_c$ .

Having derived upper bounds for the norm of the errors in the state  $E(T, \delta t)$  and costate  $\tilde{E}(t, T, \delta t)$ , we are now in position to solve problem  $P_{RT}$  to the extent of deriving an upper bound of the last term in the right hand side of (107) depending only on  $T$  and  $\delta t$ .

### 4.3 Real-Time Optimization

In order to minimize the cost in real time subject to (100), we showed in Section 4.1 that this problem can be approached by minimizing  $\|\nabla \tilde{J}(\bar{\tau}, T, \delta t) - \nabla J(\bar{\tau})\|_Q$ , subject to  $T$  and  $\delta t$ , every  $\Delta$  second. We assume that it is more important to optimize switches that are close to the current time  $t_c$  than switches far away in the future. Hence we let our norm reflect this assumption. This is a valid assumption since the switches close to  $t_c$  will be updated fewer times than the switches far away, as a simulation takes  $\Delta$  seconds. Therefore, we might only get a few chances to update the switches close to  $t_c$ . Define

$$F(\tau_i, T, \delta t) := \left\| \frac{\partial \tilde{J}}{\partial \tau_i}(\bar{\tau}, T, \delta t) - \frac{\partial J}{\partial \tau_i}(\bar{\tau}) \right\|, \quad (116)$$

for  $\forall \tau_i \in \bar{\tau}$  s.t.  $\tau_i \geq t_c + \Delta$ , where the simulated derivative is given by

$$\frac{\partial \tilde{J}}{\partial \tau_i}(\bar{\tau}, T, \delta t) = p_s[\tau_i, T, \delta t]^T (f_{i-1}(x_s[\tau_i, \delta t]) - f_i(x_s[\tau_i, \delta t])), \quad (117)$$

and the real derivative is given by (98). Since we are only interested in changing the switches that have not already passed, i.e., we only care about switches that are after  $t_c + \Delta$ ,  $F(\tau_i, T, \delta t)$  is not defined for  $\tau_i$ 's such that  $\tau_i < t_c + \Delta$ . Likewise, we note that  $\frac{\partial \tilde{J}}{\partial \tau_i}(\bar{\tau}, T, \delta t) = 0$  if  $\tau_i > t_c + T$  since we set  $p_s[t, T, \delta t] = 0$  for  $t > t_c + T$  in order to minimize the error between  $p_s$  and  $p$ . Rewriting (116), we get

$$\begin{aligned} F(\tau_i, T, \delta t) &= \|p(\tau_i)^T (f_{i-1}(x(\tau_i)) - f_i(x(\tau_i))) \\ &\quad - p_s[\tau_i, T, \delta t]^T (f_{i-1}(x_s[\tau_i, \delta t]) - f_i(x_s[\tau_i, \delta t]))\| \\ &= \|[p(\tau_i)^T - p_s[\tau_i, T, \delta t]^T][f_{i-1}(x_s[\tau_i, \delta t]) - f_i(x_s[\tau_i, \delta t])] \\ &\quad - p(\tau_i)^T [f_{i-1}(x_s[\tau_i, \delta t]) - f_i(x_s[\tau_i, \delta t]) - f_{i-1}(x(\tau_i)) + f_i(x(\tau_i))]\|, \end{aligned} \quad (118)$$

where an upper bound of (118) is given by,

$$F(\tau_i, T, \delta t) \leq 2C_f \tilde{E}(\tau_i, T, \delta t) + 2C_p L_f E(\tau_i, \delta t), \quad (119)$$

for all  $\tau_i < t_c + T$ . Likewise,

$$F(\tau_i, T, \delta t) \leq 2C_f(t_f - \tau_i)C_{L'}e^{C_{f'}(t_f - \tau_i)}, \quad (120)$$

if  $\tau_i > t_c + T$ , where the right hand side of (120) is the upper bound of  $\|\frac{dJ}{d\tau_i}\|$  using (110).

Using (108) and (115), we can get an upper bound for the right hand side of (119) that do not depend on the switching time  $\tau_i$ . This upper bound is denoted by  $F_1(T, \delta t)$ , where

$$\begin{aligned} F_1(T, \delta t) := & 2C_f \left\{ ((t_f - (t_c + T))C_{L'}e^{C_{f'}(t_f - (t_c + T))}) + S_1(\delta t)^2 + S_2\delta t \right\} \\ & + 2C_p L_f [e^{L_f T} - 1] C_f \delta t, \end{aligned} \quad (121)$$

and satisfies  $F(\tau_i, T, \delta t) \leq F_1(T, \delta t)$ ,  $\forall \tau_i \in \bar{\tau}$  s.t.  $\tau_i \geq t_c + \Delta$ . We note that since  $F_1(T, \delta t)$  do not depend on  $\bar{\tau}$ ,  $F_1(T, \delta t)$  can easily be calculated off-line and stored in order for the controller to look it up when the process is running in real time. This saves computation time, and this is why  $F_1(T, \delta t)$  was introduced. We also define the right hand side of (120) by  $F_2(\tau_i)$ , i.e.,  $F_2(\tau_i) := 2C_f(t_f - \tau_i)C_{L'}e^{C_{f'}(t_f - \tau_i)}$ .

We are now in position to define  $\|\cdot\|_Q$  as

$$\|\nabla J(\bar{\tau}) - \nabla \tilde{J}(\bar{\tau}, T, \delta t)\|_Q = \sum_{\substack{\tau_i \in \bar{\tau} \text{ s.t.} \\ \tau_i > t_c + \Delta}} \left\| \frac{\partial \tilde{J}}{\partial \tau_i}(\bar{\tau}, T, \delta t) - \frac{\partial J}{\partial \tau_i}(\bar{\tau}) \right\| \beta^{\tau_i - (t_c + \Delta)}, \quad (122)$$

where  $\beta \in (0, 1]$  describes how important it is to optimize switches close to  $t_c$  relative to switches far away from  $t_c$ . If  $\beta = 1$  then all switches are equally important to optimize. If  $\beta$  is close to zero, we consider it more important to optimize switches close to  $t_c$ . An upper bound of (122), using  $F_1(T, \delta t)$  and  $F_2(\tau_i)$ , is given by

$$\|\nabla J(\bar{\tau}) - \nabla \tilde{J}(\bar{\tau})\|_Q \leq F_1(T, \delta t) \sum_{\substack{\tau_i \in \bar{\tau} \text{ s.t.} \\ t_c + \Delta < \tau_i < t_c + T}} \beta^{\tau_i - (t_c + \Delta)} + \sum_{\substack{\tau_i \in \bar{\tau} \text{ s.t.} \\ \tau_i > t_c + T}} F_2(\tau_i) \beta^{\tau_i - (t_c + \Delta)}. \quad (123)$$

Since we assume that we simulate as long as the real time constraint allow us to, it follows from (101) that  $T = \frac{\Delta \cdot \delta t}{C_{CPU}}$ , and  $F_1(\delta t, T) = F_1(C_{CPU} \frac{T}{\Delta}, T) = F_1(T)$  is only a function of the simulation length. Looking at (123), we see that the limits of the summation for

the two sums changes each time  $t_c + T$  passes by a switching time. The two sums do not change when  $T$  is between any two switching points  $\tau_i$  and  $\tau_{i+1}$ , therefore, for each interval  $(\tau_i, \tau_{i+1})$  we only need to consider the minimum value of  $F_1(T)$  in that interval. This can be done in real time since it only requires looking up the  $T$  value that minimizes  $F_1(T)$  in given intervals  $(\tau_i, \tau_{i+1})$ .

There is no benefit to evaluate  $T$  more than every  $\delta t$  seconds, i.e.,  $T \in M(t_f, \delta t)$ , since we only have access to  $x_s$  and  $p_s$  every  $\delta t$  second. From a practical point of view, a significantly coarser grid can be chosen to evaluate  $T$  on, i.e.  $T \in M(t_f, j \cdot \delta t)$  for some integer  $j \gg 1$ .

In order to solve problem  $P_{RT}$ , the following algorithm can be employed:

#### Algorithm 4.3.1

*Given:* A simulation time  $\Delta$ , a processor constant  $C_{CPU}$ , an initial  $\bar{\tau}$ , a  $\beta$ , a positive integer  $j$  specifying how often the right hand side of (123) should be evaluated, the step size  $\gamma$ , and the system parameters, i.e. the order of the modal sequence and the cost function to be minimized.

*Init:* Calculate  $F_1(T)$ ,  $\forall T \in M(t_f, j \cdot \delta t)$ , set  $t_c = 0$ .

*Step 1.* Evaluate  $i^* = \arg \min \{\tau_i \in \bar{\tau} \mid \tau_i > t_c + \Delta\}$ . For  $\forall i \in \{i^*, \dots, \text{card}(\bar{\tau})\}$ , let  $T_i^o = \min\{F(T) \mid T \in (\tau_i, \tau_{i+1}) \cap M(t_f, j \cdot \delta t)\}$ .

*Step 2.* For all  $T_i^o$  calculated in Step 1, evaluate (123), define  $T^*$  to be the  $T_i^o$ ,  $i \in \{i^*, \dots, \text{card}(\bar{\tau})\}$  that minimizes (123).

*Step 3.* Using Euler's method, calculate  $x_s$  from time  $t_c$  to  $T^* + t_c$  and  $p_s$  from time  $T^* + t_c$  to  $\tau_{i^*}$ . Evaluate  $\nabla \tilde{J}(\bar{\tau}, T, \delta t)$ .

*Step 4.* For all  $\tau_i \in \bar{\tau}$  s.t.  $t_c + \Delta < \tau_i < t_c + T^*$ , update the switch by

$$\tau_i := \max \left\{ \tau_i - \gamma \frac{d\tilde{J}}{d\tau_i}, t_c + \Delta \right\}. \quad (124)$$

*Step 5.* If  $t_c + \Delta < t_f$ , set  $t_c := t_c + \Delta$  and go to Step 1, otherwise STOP.

Having presented Algorithm 4.3.1, a few remarks are due:

**Remark 4.3.1** *In Step 4 of Algorithm 4.3.1, the switching times are updated without taking into consideration that we are optimizing over the constraint set  $\tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq t_f$ . The algorithm can easily be changed to take this into account, but for simplicity of the exposition we do not present that case.*

**Remark 4.3.2** *The upper bound being minimized in Algorithm 4.3.1 was derived through the following steps:*

1. *Upper bounds of  $\|x(t) - x_s[t, \delta t]\|$  and  $\|p(t) - p_s[t, T, \delta t]\|$  were presented as functions of  $T$  and  $\delta t$ .*
2.  *$\|\cdot\|_Q$  was defined in order to make it more important to optimize switches close to  $t_c$  than switches far away.*
3. *An upper bound of  $|\nabla J(\tau_i) - \nabla \tilde{J}(\tau_i, T, \delta t)|$  was derived that only depended on the simulation horizon  $T$ .*
4. *This upper bound was then put into use for minimizing  $\|\nabla J(\bar{\tau}) - \nabla \tilde{J}(\bar{\tau}, T, \delta t)\|$*

As the optimization is done through minimizing conservative bounds, the question how conservative these bounds are must be considered for each system being optimized. To this end, and to show the usefulness of Algorithm 4.3.1, a numerical example is presented.

## 4.4 Numerical Example

As an application, we consider a robotics example where a mobile robot is moving towards a goal, denoted by  $x_g$ , while avoiding an obstacle located at  $x_{ob}$ . This must be achieved by choosing among different behaviors and the times to switch between them. Furthermore, the dynamics of the robot is assumed to be a single integrator, i.e.  $\dot{x} = u \in \mathbb{R}^2$  for some control signal or feedback law  $u$ . We consider using the following three standard behaviors: *go-to-goal*, *go-around-obstacle-clockwise*, and *go-around-obstacle-counterclockwise*. The respective behaviors are denoted  $f_g$ ,  $f_{\odot}$  and  $f_{\ominus}$ , and are given by

$$f_g(x) = v(x_g - x), \quad f_{\odot}(x) = v \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \frac{x_{ob} - x}{\|x_{ob} - x\|}, \quad f_{\ominus}(x) = v \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \frac{x_{ob} - x}{\|x_{ob} - x\|},$$

where  $v$  is a positive scalar equal to 1 and  $f_{\odot}(x)$  and  $f_{\ominus}(x)$  correspond to the robot moving in a circle around the obstacle in the given direction.

If we assume a given sequence of behaviors, the problem is to find good switching times in real time. For example, assume we evolve according to  $f_g(x)$  until time  $\tau_1$ , at which point the robot encounters an obstacle to the left, and therefore switches to  $f_{\odot}(x)$ . It then evolves according to  $f_{\odot}(x)$  until time  $\tau_2$ , where it switches back to  $f_g(x)$  and evolves according to this behavior until the final time  $t_f = 3$  seconds. The trajectory of the robot is then given by the following differential equation,

$$\dot{x} = \begin{cases} f_g(x), & 0 \leq t < \tau_1, \\ f_{\odot}(x), & \tau_1 \leq t < \tau_2, \\ f_g(x), & \tau_2 \leq t < t_f, \end{cases}$$

with  $x_0 = (0.05, 0)^T$ , and where  $0 \leq \tau_1 \leq \tau_2 \leq t_f$ .

The state trajectory is completely determined by the switching times,  $\tau_1$  and  $\tau_2$ , and can therefore be calculated analytically. In fact, we have that

$$x(t) = \begin{cases} e^{-t}x_0 + (1 - e^{-t})x_g, & t \in [0, \tau_1), \\ R(\theta) \begin{bmatrix} \sin(t - \tau_1) \\ -\cos(t - \tau_1) \end{bmatrix} \|x_{ob} - x(\tau_1)\| + x_{ob}, & t \in [\tau_1, \tau_2), \\ e^{-t}x(\tau_2) + (1 - e^{-t})x_g, & t \in [\tau_2, t_f], \end{cases}$$

where  $R(\theta)$  is the rotation matrix in  $\mathbb{R}^2$ .  $R(\theta)$  is given by

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix},$$

where  $\theta$  is the angle between the obstacle and the robot at time  $\tau_1$  with reference to the  $x_2$ -axes. Hence, in this example there are no errors associated with the calculation of the state trajectory and therefore  $E(t, \delta t) = 0$ .

In order for the robot not to collide with any obstacles while moving towards the goal, the cost-function  $L(\cdot) : \mathbb{R}^2 \rightarrow \mathbb{R}^+ \cup \{0\}$  should include a term that penalizes the robot for being far away from the goal and a term that incurs a cost when the robot is close to an obstacle. To satisfy this, the following cost function is used,

$$L(x(t)) = \rho \|x_g - x(t)\|^2 + \alpha e^{-\frac{\|x_{ob} - x(t)\|^2}{\beta}}, \quad (125)$$

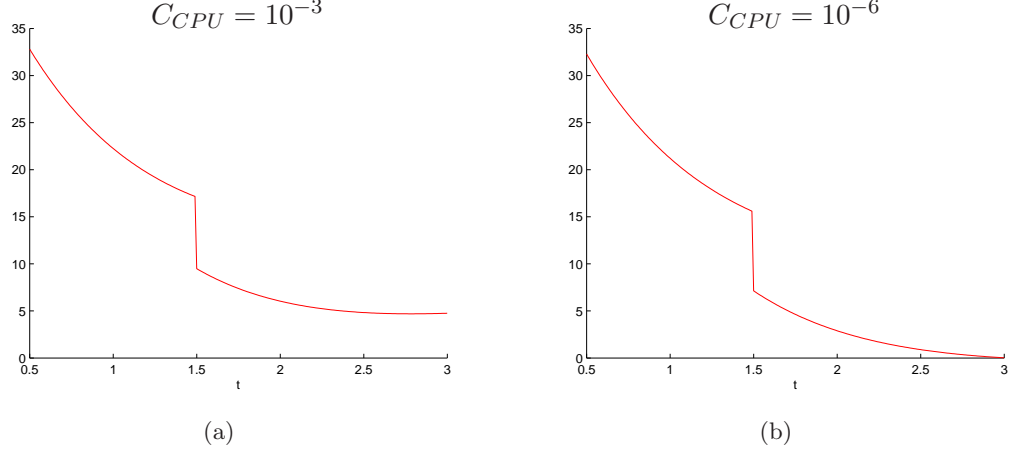
where  $\rho = 0.01$  is the gain of the goal attraction term,  $\alpha = 2$  is the gain of the obstacle avoidance term, and  $\beta = 0.1$  is the shaping parameter for the range of the obstacle avoidance term.

As noted earlier, the robot does not know the exact location of the obstacle before it starts running, therefore it can not determine good switching times. However, we do assume that the robot knows that if there is an obstacle, it will be to the left of its trajectory. Therefore, the robot is initialized to evolve according to the following sequence of behaviors  $\{f_g, f_{\odot}, f_g\}$  and the switching vector is initialized to be  $\bar{\tau} = (0, 0.5, 1.5, 3)$ . The robot starts at  $x_0 = (0.05, 0)^T$  and the goal is located at  $x_g = (0, 2.5)^T$ . We assume that we can only simulate 0.25 seconds into the future before we have to update our switching times, hence  $\Delta = 0.25$ .

In order to determine how far into the future to simulate, i.e. what  $T$  to choose, we will minimize the upper bound off the error between the true gradient and the simulated gradient with respect to how far into the future we simulate according to the right hand side of (123). However, if we apply naive bounds on the constants needed to calculate  $F_1(T)$ , as described in (121), and  $F_2(\tau_i)$ , the bound will be very conservative in two ways. Firstly, if we do not have enough computing power, i.e.  $C_{CPU}$  is not small enough, the minimum of the right hand side of (123) might be orders of magnitude bigger than the actual gradient value and minimizing the upper bound would not give us any additional information. Secondly, if we have enough computing power, minimizing the right hand side of (123) will typically tell us to simulate all the way to the end, i.e.  $T$  will be such that  $t_c + T = t_f$ . An illustration of this for our example is given in Figure 25. Note that the discontinuity in Figure 25 is due to index in the summation of the right hand side of (123) changes at  $T = \tau_1$ .

The problem is that if we do not simulate all the way until  $t_f$  the error in  $p$  will grow very fast since we have not made any assumption regarding the dynamics of the state. To this end, we note that after time  $\tau_3$ , we have that

$$\dot{p} = -\frac{df_g^T}{dx}(x(t))p(t) - \frac{dL^T}{dx}(x(t)), \quad (126)$$



**Figure 18:** Right hand side of Eq. (123). (a)  $C_{CPU}$  is not small enough for our bound to be useful. The minimum value 4.67 is attained at  $T = 2.79s$ . The norm of the gradient in our example is typically less than 1 and hence, the plot does not give any information. (b)  $C_{CPU} = 10^{-6}$  and we see that it is optimal to simulate all the way until the end and that the error at the end is smaller than 0.005. If we simulate until the end we will get a step-length,  $\delta t = 12\mu s$  which is very small.

for  $t \in (\tau_2, 3]$  where we note that  $\frac{df_g}{dx}$  is the identity matrix and therefore,  $-\frac{df_g}{dx}$  is negative definite and the first term in the right hand side of (126) corresponds to stable dynamics. Regarding the second term in the right hand side of (126), we assume that  $\|x_{ob} - x(t)\| \geq 0.65$ , which is reasonable since we want the robot to avoid the obstacle, and that  $\|x_g - x(t)\| < 2$ ,  $\forall t \in [\tau_2, 3]$ , we then get that  $\|\frac{dL^T}{dx}(x(t))\| = 2\rho\|x_g - x(t)\| + \frac{2\alpha}{\beta}\|x_{ob} - x(t)\|e^{-\frac{\|x_{ob} - x(t)\|^2}{\beta}}\| < 4\rho + \frac{2\alpha}{\beta}\|x_{ob} - x(t)\|e^{-\frac{\|x_{ob} - x(t)\|^2}{\beta}} \leq 4\rho + \frac{2\alpha}{\beta} \max_z\{z \geq 0.65 \mid ze^{-\frac{z^2}{\beta}}\} = 0.42$ ,  $\forall t \in [\tau_2, t_f]$ . If we define  $c_1 = 0.42$  and  $c_2 = 1$ , and denote the bound on the costate in  $(\tau_2, t_f]$  by  $p_b$ , we get that  $\dot{p}_b = c_1 + p_b c_2$ . Solving for  $p_b$  backwards it follows that,

$$p_b(t) = \frac{c_1}{c_2}(1 - e^{(t-t_f)c_2}), \quad \forall t \in (\tau_2, t_f].$$

Using this bound for the costate after time  $\tau_2$  implies that  $\|p_s[t, T, \delta t] - p(t)\| \leq \frac{c_1}{c_2}(1 - e^{(t-t_f)c_2})$  for all  $t \geq T + t_c$ . This bound enable us to get a good result when evaluating (123) using reasonable  $C_{CPU}$  values. To summarize, we have the following expression for the bound of the costate,

$$C_p(t) = \begin{cases} \frac{c_1}{c_2}(1 - e^{(t-t_f)c_2}), & \tau_2 \leq t \leq t_f, \\ \frac{c_1}{c_2}(1 - e^{(\tau_2-t_f)c_2}) + (\tau_2 - \tau_1)C_{L'}e^{C_{f'}(\tau_2-\tau_1)}, & \tau_1 \leq t \leq \tau_2, \end{cases}$$

**Table 1:** Switching vector, optimal simulation time, and robots position as a function of  $t_c$ .

$t_c$	$\bar{\tau} = (\tau_1, \tau_2)^T$	$T + t_c$	$x(t_c)$
0	$(0.5, 1.50)^T$	1.5	$(0.05, 0)^T$
0.25	$(0.39, 1.50)^T$	3	$(0.04, 0.55)^T$
0.50	$(0.39, 1.50)^T$	3	$(0.11, 0.87)^T$
0.75	$(0.39, 1.51)^T$	3	$(0.27, 1.08)^T$
1	$(0.39, 1.51)^T$	3	$(0.36, 1.31)^T$
1.25	$(0.39, 1.52)^T$	3	$(0.37, 1.57)^T$
1.50	$(0.39, 1.52)^T$	3	$(0.32, 1.81)^T$

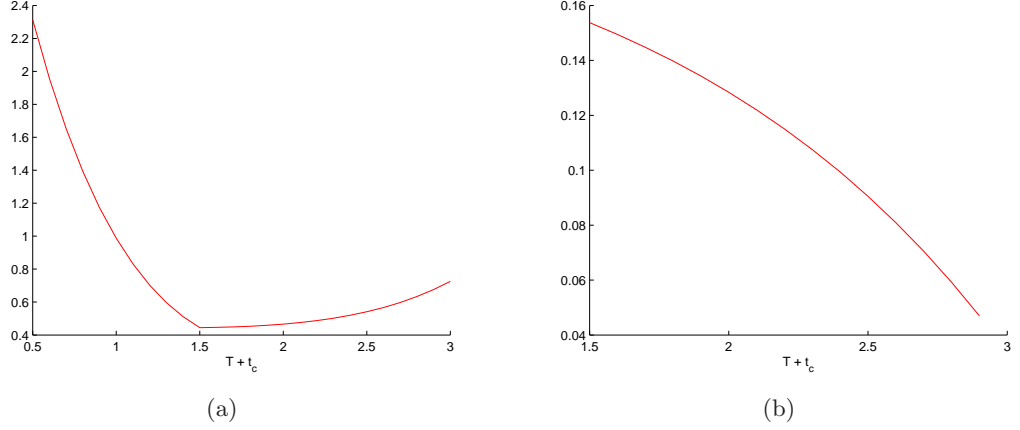
and we do not care about  $p$  before time  $\tau_1$ , since  $\tau_1$  is the first switching time. Furthermore, we will introduce a time dependence on  $S_1$  and  $S_2$  and evaluate them for different switching times. We then have that

$$\begin{cases} S_{1,\tau_i}(T) = e^{C_{f'}(t_c+T-\tau_i)} (2C_{f'}C_{\dot{p}}(t) + L_{L'}C_f), \\ S_{2,\tau_i}(T) = \frac{e^{C_{f'}(t_c+T-\tau_i)} - 1}{C_{f'}} [L_{L'}C_f + C_{f'}C_{\dot{p}}(t) + \\ C_p(t)L_{h'}C_f], \end{cases}$$

since  $E(t, \delta t) = 0$ , and  $C_{\dot{p}}(t)$  follows from  $C_p(t)$  and (99). Note that having time dependent bounds on the costate and on  $S_{1,\tau_i}$  and  $S_{2,\tau_i}$  is a departure from the general setting where we calculated  $F_1(T)$  off-line and did not have any  $T$  dependence in  $S_1$  and  $S_2$ . However, as noted earlier,  $F_1(T)$ ,  $S_{1,\tau_i}(T)$  and  $S_{2,\tau_i}(T)$  do not need to be evaluated every  $\delta t$  seconds, instead it might be sufficient to evaluate them at a significant lower rate since we are only looking for a good value of  $T$  in order to minimize the norm between the true gradient and the calculated gradient. In our case, we evaluate  $T$  every 0.1 seconds. Figure 27(a) shows a plot of the right hand side of (123) when we are at time  $t_c = 0$ . At this point, we assume the robot sees the obstacle located at  $x_{ob} = (-0.5, 1.5)^T$ . If it did not see the obstacle then we would set  $x_{ob}$  to be a big number and in practice we would not minimize with respect to the second term of (125). The value of the system constants are as follows:  $C_f = 1$  since we are only interested in times  $t \geq \tau_1$ , likewise  $C_{f'} = 1$ ,  $C_L = \frac{2^2}{\rho} + 2e^{-\frac{\|x_{ob}-x\|^2}{\beta}} \big|_{\|x_{ob}-x\|=0.65} = 0.07$ ,  $C_{L'} = \frac{2^2}{\rho} + 2^2 \frac{0.65}{\beta} e^{-\frac{0.65}{\beta}} = 0.42$ ,  $L_L = C_{L'}$ ,  $L_{L'} = \frac{2}{\rho} + \frac{2^2}{\beta} e^{-\frac{0.65^2}{\beta}} = 0.60$ ,  $L_f = 1$ ,  $L_{h'} = 1$ .

It can be seen that the minimum in Figure 27(a) is obtained at  $T = \tau_2$ . Therefore the robot calculates  $p_s$  backwards from time  $\tau_2$ , where  $p_s[\tau_2, \tau_2, \delta t] = 0$ , until time  $\tau_1$  and then evaluate the gradient and update the switching vector. Table 1 shows where the minimum





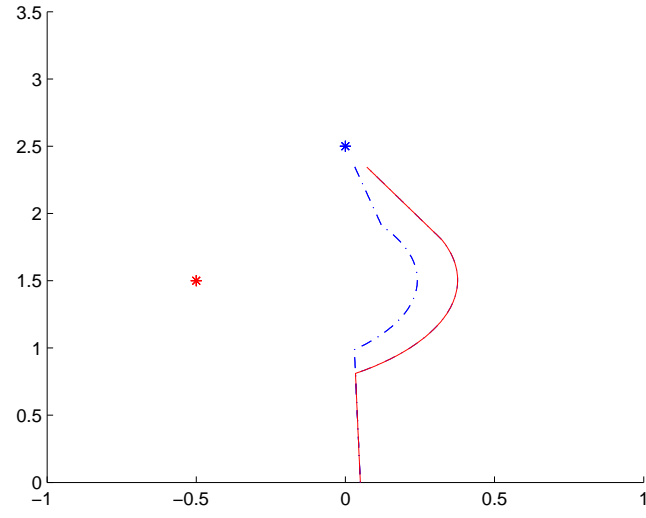
**Figure 19:** Right hand side of (123) evaluated at (a)  $t_c = 0$ , and (b)  $t_c = 0.25$ .

$T$  is attained for different  $t'_c$ s, how the switching vector is updated, and the robots position at time  $t_c$ . At time  $t_c = 0.25$ , we see that  $t_c + \Delta > \tau_1$ , hence the first switching time would have passed before we have a chance to update it. Therefore, in the right hand side of (123) the summation changes and hence, we get a new time  $T$  that minimizes the right hand side of (123), this is shown in Figure 27(b). Finally, Figure 20 shows the robots final trajectory together with the initial trajectory (dotted) and Figure 21 shows how the total cost for the robots trajectory is changing for every new  $t_c$  value, i.e., every 0.25 seconds. As can be seen from Figure 21 the robot effectively reduces the cost in real time.

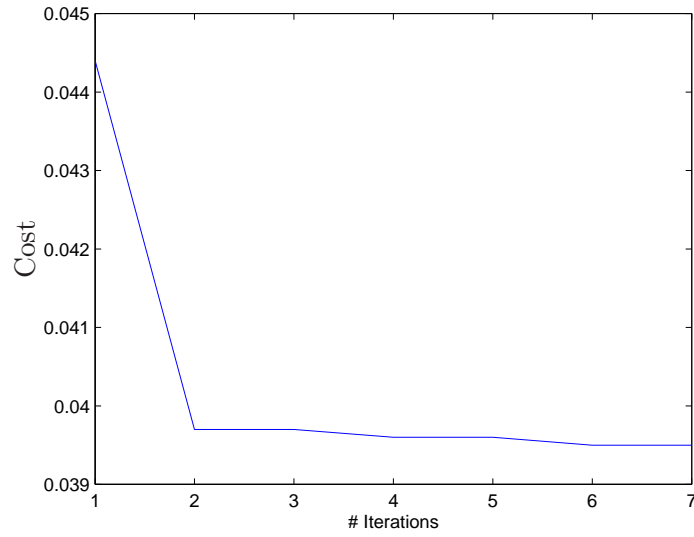
## 4.5 Conclusion

In this Chapter, an approach to real time optimization of the switching times in autonomous hybrid systems is proposed. The approach is based on minimizing an upper bound of the norm between the true gradient and the gradient we obtain by simulating the state and the costate trajectories. To this end, upper bounds for the error between the true state/costate and the simulated state/costate trajectories are presented as a function of the computation horizon and the time available to the controller before a result must be delivered. The bound of the norm between the true gradient and the simulated gradient has a simple form and, therefore, we can minimize this norm in real time with little computational effort.

The feasibility of the approach is verified through a robotics example where a mobile



**Figure 20:** Robots initial (dotted) and final (solid) trajectory



**Figure 21:** Cost as a function of iterations of Algorithm 4.3.1. At iteration 1,  $t_c=0$ , at iteration 2,  $t_c = 0.25$  etc.

robot has to move towards a goal position while avoiding an obstacle.

It should be noted that although we did not explicitly talk about how to modify the mode sequence in real time, the results in this Chapter can be extended to cover this case as well. This is true since performing an insertion only amounts to evaluating the gradient formula at the points when an insertion is considered and this can be accounted for in the simulation-time function.

In summary, the results given in this Chapter present a computationally appealing way of extending the switching time optimization results, presented in Chapter II, to the case when we are optimizing in real time.

## CHAPTER V

# OPTIMAL MODE SWITCHING FOR HYBRID SYSTEMS WITH UNKNOWN INITIAL STATE

The results presented in Chapters II-IV for the mode-sequencing/switching-time optimization problem all assumed that the initial state of the system was given. In many situations this is an unreasonable assumption. Rather, one knows that the initial state is within a certain region, i.e., if  $x \in \mathbb{R}^3$  then it might be given that  $x_0 \in S \subset \mathbb{R}^3$ . A common approach to minimizing the cost for this case is to minimize the maximum cost for all trajectories starting in  $S$ . This chapter presents a method to optimize the switching times for switched autonomous systems based on minimizing the maximum cost associated with all initial states in  $S$ . Hence, the theory of minimax optimization will be utilized. Minimax optimization is a well studied subject (see for example [56, 26]) and the reader is referred to these references for an introduction to the subject. In this chapter, we will only present the results necessary for our problem, including an algorithm that will provide a solution to the switching time optimization problem for an unknown initial state.

The setting in this chapter is slightly different from previous chapters in that the switching times are not controlled directly. Instead, we assume that the system switches whenever it intersects given switching surfaces parameterized by a switching parameter  $a$ , to be defined later. This problem was initially considered in [16] and we will refer the reader to that paper for results relating to the gradient of the cost-functional with respect to the switching surface parameter.

The outline of this Chapter is as follows: In Section 5.1, the problem at hand is introduced together with some previous results relating to the gradient formula. Section 5.2 presents our solution using a minimax strategy, and a robotics example is presented in Section 5.3. Finally, conclusions are given in 5.4.

## 5.1 Problem Formulation & Previous Results

The underlying system under consideration in this chapter is the same as in Chapters II and III, i.e. we consider nonlinear switched dynamical systems satisfying the following equation,

$$\dot{x}(t) = f_i(x(t), t \in [\tau_{i-1}, \tau_i)), i \in \{1, \dots, N+1\}, \quad (127)$$

where we assume that the system switches  $N$  times. The modal functions are chosen from a given set  $\{f_\alpha\}_{\alpha \in A}$ . However, in this chapter we assume that the switching times are not controlled directly. Instead, a switch occurs whenever the state trajectory intersects a *switching* surface. This problem was initially considered in [16] by Boccadoro *et al.* for a fixed initial state. We will follow the presentation of [16] in order to set the stage for our minimax problem when the initial state is not completely known.

We assume that the switching times and the modal functions are determined recursively in the following way. Given  $f_i$  and  $\tau_{i-1} > 0$  for some  $i = 1, 2, \dots$ , let  $A(i) \subset A$  be a given finite set of modes, labelled *the set of modes enabled by  $f_i$* . Hence, there might be a restriction on the mode sequence. For every  $\alpha \in A(i)$ , we let  $S_\alpha \subset \mathbb{R}^n$  be the  $n-1$  dimensional surface enabling the switch to mode  $\alpha$ . Then, the next switch is defined by

$$\tau_i = \min\{t > \tau_{i-1} : x(t) \in \cup_{\alpha \in A(i)} S_\alpha\} \quad (128)$$

and we note that it is possible to have  $\tau_i = \infty$ . If  $\tau_i < \infty$  then we pick  $\tilde{\alpha} \in A(i)$  such that  $x(\tau_i) \in S_{\tilde{\alpha}}$ , and we set  $f_{i+1} = f_{\tilde{\alpha}}$ . The system is initialized by setting  $\tau_0 = 0$  and choosing what mode the system should start with.

The time when the state trajectory intersects a surface defines  $\tau_i$ , and the index of the surface  $S_{\tilde{\alpha}}$  defines  $f_{i+1}$ . In this chapter, the surfaces  $S_\alpha$  are defined by the solution points of parameterized equations from  $\mathbb{R}^n$  to  $\mathbb{R}$ . We denote the parameter by  $a$  and suppose that  $a \in \mathbb{R}^k$  for some integer  $k \geq 1$ . For every  $\alpha \in A$ , we let  $g_\alpha : \mathbb{R}^n \times \mathbb{R}^k \rightarrow \mathbb{R}$  be a continuously differentiable function. For a given fixed value of  $a \in \mathbb{R}^k$ , denoted here by  $a_\alpha$ , the switching curve  $S_\alpha$  is defined by the solution points  $x$  of the equation  $g_\alpha(x, a_\alpha) = 0$ . Note that under mild assumption,  $S_\alpha$  is a smooth  $(n-1)$  dimensional manifold in  $\mathbb{R}^n$ , and  $a_\alpha$  can be viewed as a control parameter of the surface. Using the terminology defined earlier, we will replace

the index  $\alpha$  by  $i$ ; thus,  $S_i$  is the solution set of the equation

$$g_i(x, a_i) = 0, \quad (129)$$

which is parameterized by the control variable  $a_i \in \mathbb{R}^k$ .

To summarize, the system changes dynamics whenever the state trajectory intersect a switching curve  $g(x, a) = 0$  parameterized by a control variable  $a$ , as illustrated in Figure 22. In order to minimize a cost criterion of the form

$$J = \int_0^T L(x(t))dt, \quad (130)$$

where  $L : \mathbb{R}^n \rightarrow \mathbb{R}$ , we need to determine the optimal switching surface parameters  $a$  since the state trajectory depends on  $a$ . To this end [16] presented an expression of the gradient of the cost functional with respect to switching surface parameter  $a$ . This gradient was presented under the assumption that the functions  $f_i, g_i$   $i = 1, \dots, N + 1$ , and  $L$  were continuously differentiable with respect to all its variables. Furthermore, it was assumed that  $f_i$   $i = 1, \dots, N + 1$ , was uniformly Lipschitz. Note that the same assumption were made in Chapter II, Assumption 1.

We define  $x_i = x(\tau_i)$ , and the terms  $R_i$  and  $L_i$  by

$$R_i = f_i(x_i) - f_{i+1}(x_i), \quad (131)$$

and

$$L_i = \frac{\partial g_i}{\partial x}(x_i, a_i) f_i(x_i), \quad (132)$$

where we recognize  $L_i$  as the Lie derivative of  $g_i$  in the direction of  $f_i$ .

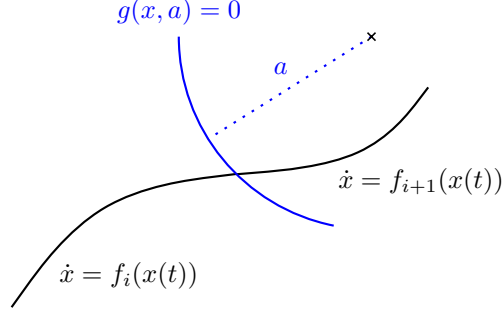
Now, in order to ensure that the gradient exists, the following assumption is presented;

**Assumption 5** For all  $i = 1, \dots, N$ ,  $L_i \neq 0$ .

Given Assumption 5, reference [16] related the derivative  $\frac{dJ}{da_i}$  to the derivative term  $\frac{dJ}{d\tau_i}$  in the following way:

**Proposition 5.1.1** The following equation is in force,

$$\frac{dJ}{da_i} = -\frac{1}{L_i} \frac{dJ}{d\tau_i} \frac{\partial g_i}{\partial a_i}(x_i, a_i). \quad (133)$$



**Figure 22:** Mode switching occur when the state trajectory intersect a switching surface. In this case, the switching surface is a circle parameterized by the radius  $a$ .

where the costate equation is given by

$$\dot{p}(t) = - \left( \frac{\partial f_{i+1}}{\partial x}(x(t), t) \right)^T p(t) - \left( \frac{\partial L}{\partial x}(x(t)) \right)^T ; t \in [\tau_i, \tau_{i+1}), i = 1, \dots, N, \quad (134)$$

with terminal condition  $p^T(t_N) = 0$  when the final time is fixed, and reset conditions

$$p(\tau_i^-) = (I - \frac{1}{L_i} R_i \frac{\partial g_i}{\partial x}(x_i, a_i))^T p(\tau_i^+), \quad i = 1, \dots, N. \quad (135)$$

**Proof:** See [16].

The expression for  $\frac{dJ}{d\tau_i}$  is the same as the one given in Chapter II with the costate replaced with the above equations.

Having presented the expression for the gradient, as derived in [16], we can now proceed to present the minimax solution to our switching surface parametrization problem.

## 5.2 Minimax Optimization

Given a set of possible initial points  $S \subset \mathbb{R}^n$ , a set of switching surfaces parameterized by some vector  $a$ , and an instantaneous cost  $L$ , the total cost, starting at  $x_0 \in S$ , is given by

$$J_{x_0}(a) = \int_0^T L(x(t)) dt, \quad (136)$$

where  $T$  is a fixed final time and subscript  $x_0$  indicates the initial condition. Our problem, denoted by  $P_S$ , can be stated as

$P_S$ : Given a set of initial states  $S$  and a set of switching surfaces parameterized by  $a$ , find the surface parameter  $a$  such that

$$\max\{J_x(a) \mid x \in S\} \quad (137)$$

is minimized.

As mentioned earlier, the theory of minimax optimization and consistent approximations [56] will be utilized in order to implement and solve this problem.

Given a set of possible initial states  $S \subset \mathbb{R}^n$ , we will choose a sequence of sets of initial points,  $\{\mathbb{X}_i\}_{i=0}^\infty$ . This sequence will satisfy the following three conditions: Firstly,  $\mathbb{X}_i \subset S$   $i = 1, 2, \dots$ ; secondly, the number of elements in  $\mathbb{X}_i$  is bigger than the number of elements in  $\mathbb{X}_{i-1}$ ; thirdly, every point in  $S$  will be arbitrarily close to a point in  $\mathbb{X}_i$ , as  $i$  goes to infinity. Choosing  $\{\mathbb{X}_i\}_{i=0}^\infty$  in this way enables us to find the solution to (137) by solving a sequence of optimization problems, each one with a different set of initial states.

For each  $\mathbb{X}_i$  we will find the optimal switching parameter  $a_i^o$  that minimizes  $\max\{J_x(a_i) \mid x \in \mathbb{X}_i\}$  through a gradient descent algorithm, as described below. After we have found the optimal  $a_i^o$ , we will solve  $\max\{J_x(a_{i+1}) \mid x \in \mathbb{X}_{i+1}\}$  by initializing  $a_{i+1}$  to  $a_i^o$ . This gives a good starting point for the gradient descent algorithm.

For each  $\mathbb{X}_i$  we will find the optimal  $a_i^o$  by executing the following gradient descent algorithm with Armijo step size [6]. We assume that  $\mathbb{X}_i$  have  $N(i)$  elements, i.e.  $\mathbb{X}_i = \{x_1, \dots, x_{N(i)}\}$  for some  $x_1, \dots, x_{N(i)}$  in  $S \subset \mathbb{R}^n$ .

**Algorithm 5.2.1** *Gradient Projection Algorithm with Armijo Stepsize*

*Given:* The Armijo constants  $\alpha, \beta$  in  $(0, 1)$ . Two constants  $\delta > 0$ , and  $\epsilon > 0$  and the set of initial points  $\mathbb{X} = \{x_1, \dots, x_N\} \subset S$ .

*Initialize:* Choose a feasible initial guess on the switching surface parameter  $a$ .

*Step I:* Calculate the maximum cost for the given set of initial states, denoted

$$F(\mathbb{X}, a) = \max_x \{J_x(a) \mid x \in \mathbb{X}\}, \quad (138)$$

where  $J_x$  is given by (136). Let  $I(\mathbb{X}, a)$  denote the index set of active constraints, i.e.

$$I(\mathbb{X}, a) = \{j \in \{1, \dots, N\} \mid F(\mathbb{X}, a) - J_j(a) < \epsilon\}. \quad (139)$$

Calculate the generalized gradient

$$\partial F(\mathbb{X}, a) = \text{conv}\{\nabla J_j(a) \mid j \in I(\mathbb{X}, a)\}, \quad (140)$$



where  $\text{conv}$  denotes the convex hull. Find the point in  $\partial F(\mathbb{X}, a)$  closest to the origin and denote it by  $h$ . If  $\|h\| < \delta$  then STOP. Else, goto Step II.

Step II: Calculate the step-length  $\lambda$  according to Armijo's rule i.e.

$$\lambda = \max\{z = \beta^k; k \geq 0 \mid F(\mathbb{X}, a - zh) - F(\mathbb{X}, a) \leq -\alpha z \|h\|^2\}.$$

Update  $a$  according to  $a = a - \lambda h$ , goto Step I.

A few remarks concerning Algorithm 5.2.1 are due.

**Remark 5.2.1** The index set of active constraints,  $I(\mathbb{X}, a)$ , is introduced in order to determine what initial states in  $\mathbb{X}$  we should take into consideration for a given  $a$ . If the index of an initial state is in the index set, then the gradient of the cost associated with that initial state is current in the calculation of the generalized gradient,  $\partial F(\mathbb{X}, a)$ . If  $\epsilon = 0$  in (139), i.e., we only optimize with respect to the initial state corresponding to the maximal cost, it is conceivable that we can only take a very small descent step since the index set changes when  $a$  changes.

**Remark 5.2.2** In order to find the optimal  $a$  for a given set of initial states, we would have to set the constants  $\delta$  and  $\epsilon$  to 0. However, doing this when we solve for a sequence of initial states,  $\{\mathbb{X}_i\}_{i=0}^{\infty}$ , would not give any additional benefit, instead we only require that for each consecutive problem we will solve,  $\delta$  and  $\epsilon$  will decrease, and in the limit when  $i \rightarrow \infty$ , they will be zero.

**Remark 5.2.3** Solving for  $h$  is a standard quadratic optimization problem over a convex set, and can be solved using a variety of optimization algorithms.

**Remark 5.2.4** In the robotics example presented in Section 5.3, a simple constraint is introduced on  $a$ . Hence we need to initialize  $a$  to be in the set of feasible points.

In order to illustrate the calculation of  $h$ , a simple example is presented. Assume that we have four different initial states,  $x_1$  through  $x_4$  in  $\mathbb{R}^2$ . In Figure 23, their respective gradients are plotted and it is assumed that  $x_1$  through  $x_3$  are active initial states for the

given switching surface parameter  $a$ . The shaded region in Figure 23 corresponds to the convex hull of the gradients of the active initial states, and  $h$  is the closest vector in this set from the origin.

Having presented Algorithm 5.2.1. and the remarks that follow it, we are now in the position to present Algorithm 5.2.2 that will solve problem  $P_S$ .

**Algorithm 5.2.2** *Minimax optimization for unknown initial state:*

*Given:* A sequence of initial sets  $\{\mathbb{X}_i\}_{i=0}^{\infty} \in S \subset \mathbb{R}^n$ , where  $\mathbb{X}_i = \{x_1, \dots, x_{N(i)}\}$  and  $N(i) > N(i-1)$ . Two positive sequences  $\{\epsilon_i\}_{i=0}^{\infty}$  and  $\{\delta_i\}_{i=0}^{\infty}$  such that in the limit when  $i \rightarrow \infty$ , both are 0.

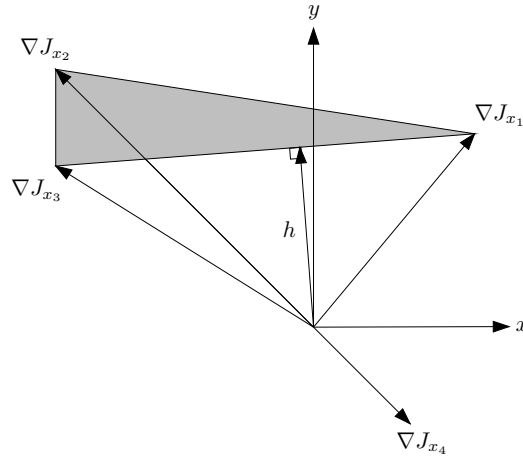
*Init:* Set  $i = 0$ , pick a feasible initial guess on  $a_0$ .

*Step I:* Use Algorithm 5.2.1. to optimize over  $a$  with  $\mathbb{X} = \mathbb{X}_i$ ,  $\delta = \delta_i$ ,  $\epsilon = \epsilon_i$ . Initialize  $a$  with  $a_{i-1}$  if  $i \neq 0$ , and with  $a_0$  if  $i = 0$ .

*Step II:* Set  $a_i$  to  $a$  given from Algorithm 3.1. Increase  $i$  by one, goto Step I.

### 5.3 Numerical Example

In order to show the usefulness of Algorithm 5.2.2, we consider a mobile robot navigation problem. The task of the robot is to get to a goal point  $x_g \in \mathbb{R}^2$  while avoiding an obstacle, located at  $x_{ob} \in \mathbb{R}^2$ , in its path. It has to do this by switching between two different



**Figure 23:** Calculation of  $h$  given four initial states and their respective gradients.  $x_1$  through  $x_3$  are active initial states.

behaviors, one *go-to-goal* and one *obstacle-avoidance* behavior. These different behaviors are denoted by  $f_g$  and  $f_o$  respectively. We model the robot having unicycle dynamics

$$\begin{cases} \dot{x}_1 = v \cos(\phi), \\ \dot{x}_2 = v \sin(\phi), \\ \dot{\phi} = f_q(x_1, x_2, \phi), \end{cases} \quad (141)$$

where  $(x_1, x_2)$  is the position of the robot,  $\phi$  is its heading, and  $q \in \{g, o\}$  is the current behavior the robot evolves according to. We assume that the translational velocity  $v$  is constant. Our control variable is then given by the switching surface parameters of the goal and avoid obstacle guards that dictate what behavior the robot should evolve according to. A standard pair of “approach-goal” and “avoid-obstacle” behaviors are given by

$$f_g(x_1, x_2, \phi) = c_g(\phi_g - \phi), \quad (142)$$

$$f_o(x_1, x_2, \phi) = c_o(\pi + \phi_{ob} - \phi). \quad (143)$$

Here,  $c_g$  and  $c_o$  are the gains associated with each behavior, and  $\phi_g$  and  $\phi_{ob}$  are the angles to the goal and nearest obstacle respectively. Both of these angles are measured with respect to the  $x$ -axis and can be expressed as

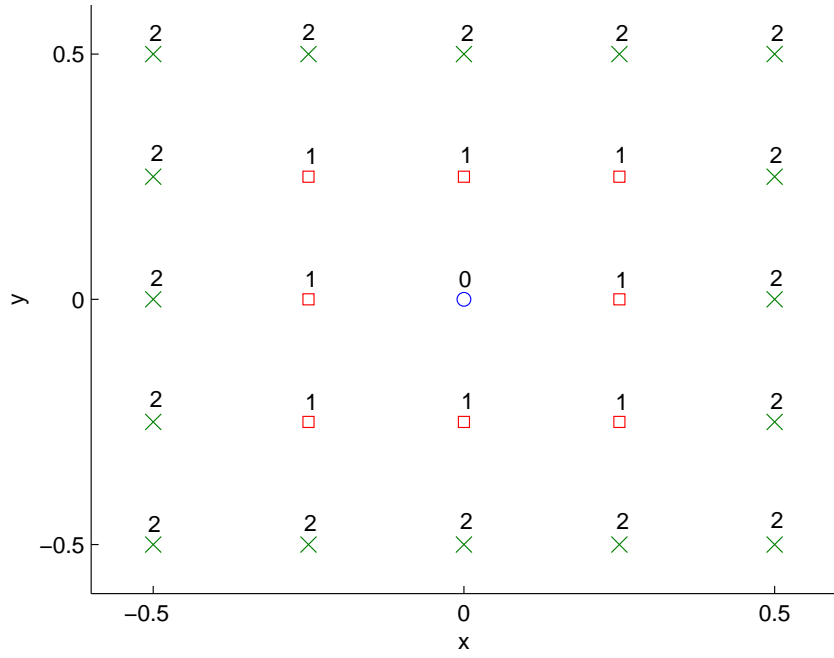
$$\phi_g = \arctan\left(\frac{x_{g2} - x_2}{x_{g1} - x_1}\right), \quad (144)$$

$$\phi_{ob} = \arctan\left(\frac{x_{ob2} - x_2}{x_{ob1} - x_1}\right), \quad (145)$$

where  $(x_{g1}, x_{g2})$  and  $(x_{ob1}, x_{ob2})$  are the Cartesian coordinates of the goal and the nearest obstacle respectively.

The instantaneous cost  $L$  is the same as in Chapter IV, Equation (125).

For a given initial position  $x_0 \in \mathbb{R}^3$  the total cost is given by (136). However, many mobile robots get their position from GPS readings which has an error associated with them. In our example, we assume that the robot get the initial position  $x_0 = (0, 0, \cdot)^T$  from the GPS and that the error associated with the GPS is 0.5 meters (note that GPS do not give the direction of a stationary robot). In order to simplify our exposition, we assume that the robot is always directed towards the goal, hence we will only show the  $(x_1, x_2)$



**Figure 24:** Initial states used:  $\mathbb{X}_i$  contains the points with index  $i, i-1, \dots, 0$ .

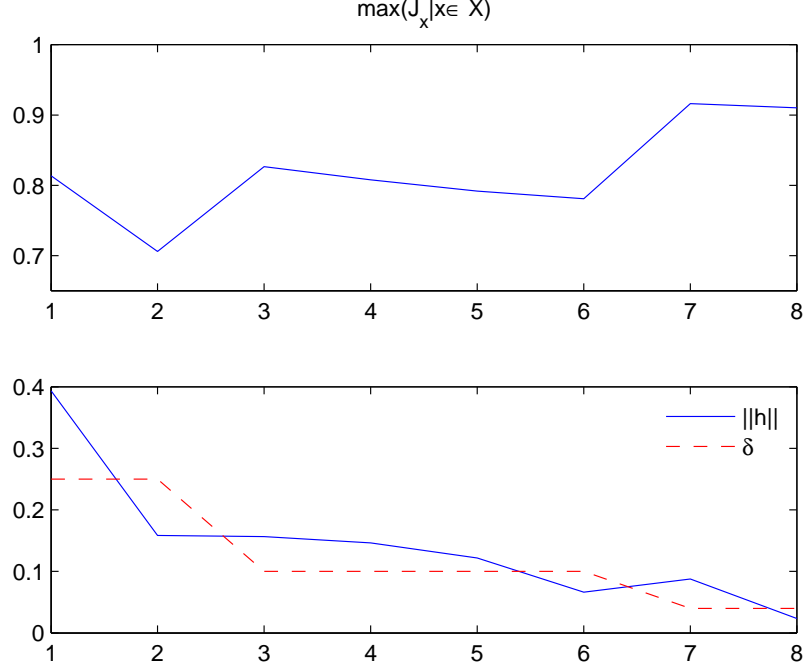
components in  $\mathbb{X}_i$ ,  $i = 0, 1, 2$ . This is a reasonable assumption if the robot can see the goal, which we assume.

Due to the error in the GPS reading, the robot can be anywhere in the interval  $[-0.5, -0.5] \times [0.5, 0.5]$ . Therefore we initialize Algorithm 5.2.2 with only one initial state  $\mathbb{X}_0 = (0, 0)^T$  and we then extend the set of initial states, in a somewhat arbitrary fashion, as shown in Figure 24. In this example, we stop the algorithm after its third iteration, i.e. when  $\|h\| < \delta_2$ , therefore we do not define  $\mathbb{X}_i$  for  $i = 3, 4, \dots$

The switching surfaces for when to switch from  $f_g$  to  $f_o$ , and when to switch from  $f_o$  to  $f_g$ , are given by two circles with radius  $a_1$  and  $a_2$  respectively, where we require  $a_1 \leq a_2$ . Both circles are centered at the obstacle  $x_o = (2, 1.25)^T$ . At this point it should be noted that having circular guards might not correspond to an optimal guard shape (for a discussion about how to find the optimal guard shape, see Chapter VI).

We initialize  $a$  to be  $(1, 1.5)^T$  and for the constants in  $L$ , we set  $\rho = 0.01$ ,  $\alpha = 10$  and  $\beta = 0.1$  and we use  $c_g = c_o = 1$  for the feedback gains in (142) and (143). The velocity

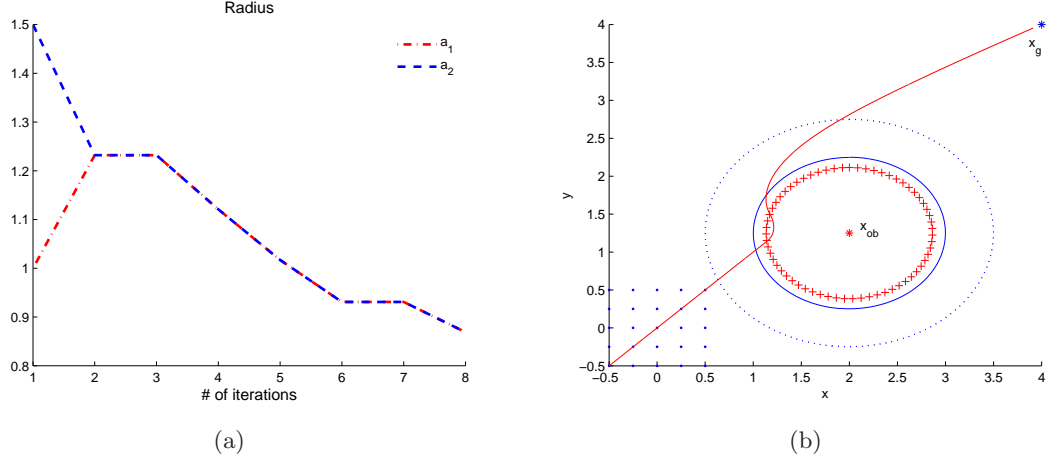
of the robot is set to  $v = 0.5$  and the goal is located at  $x_g = (4, 4)^T$ . For the constants in the Armijo procedure, we use  $\alpha = \beta = 0.5$ . The sequences of  $\epsilon_j$  and  $\delta_j$  used is given by  $\delta_j = \frac{\delta_{j-1}}{2.5}$  with  $\delta_0 = 0.25$ , and  $\epsilon_j = \frac{\epsilon_{j-1}}{2.5}$  with  $\epsilon_0 = 0.1$



**Figure 25:** (a) Change in maximum cost. (b)  $\|h\|$  and  $\delta$  as a function of the number of gradient descent iterations in Algorithm 5.2.2.

A plot of how the cost changes together with the norm of  $h$  and  $\delta$  is shown in Figure 25. As can be seen in the figure, Algorithm 5.2.2 effectively reduces the maximum cost for a given set of initial states. Once the norm of  $h$  falls below  $\delta$ , we update  $\delta, \epsilon$  and the set of initial states,  $\mathbb{X}$ .

Once we have updated  $\mathbb{X}_0$  to  $\mathbb{X}_1$  after iteration three, we see that the maximum of the cost increases, just as should be expected since  $\mathbb{X}_1$  has more initial states than  $\mathbb{X}_0$ . Figure 26(a) shows how the switching surface parameters change. The optimum is obtained when  $a_1 = a_2$ , i.e. both radii are the same. Finally, Figure 26(b) shows the final set of initial states  $\mathbb{X}_2$  together with the initial and final switching surfaces and the trajectory corresponding to the maximum cost for  $\mathbb{X}_2$ .



**Figure 26:** Change in  $a$  and final trajectory: (a) Change in  $a = (a_1, a_2)^T$  as a function of the number of gradient descent iterations in Algorithm 5.2.2.  $a_1$  is the radius of the *obstacle-avoidance* switching surface,  $a_2$  is the radius of the *go-to-goal* switching surface. (b) Final trajectory giving the maximum cost together with initial and final  $a$ , and  $\mathbb{X}_2$ . Initial  $a_1$  dotted, initial  $a_2$  solid, final  $a_1$  and  $a_2$  are equal to each other and are represented by the marked circle.

## 5.4 Conclusions

This Chapter presented a novel way of getting rid of the dependence on the initial condition when optimizing the switching times for a switched autonomous system. The system considered switched between different modes as the state trajectory intersected different parameterized switching surfaces. Hence, the optimization was done with respect to the switching surface parameter.

The dependence on the initial condition was dealt with by minimizing the switching parameter over the maximum cost for a given set of initial states. To this end, the theory of minimax optimization was used as it applied to our problem.

An algorithm was presented that effectively minimized the maximum cost for the system given that the initial state of the system was confined within a given region in the state space.

The contribution made in this chapter ties in nicely with the previous results presented in the thesis since the gradient formula presented in Chapter II can be directly applied in Algorithms 5.2.1 and 5.2.2. Furthermore, the inherit dependence on the initial state

when using calculus of variations was overcome by introducing our minimax strategy of minimizing the cost.

## CHAPTER VI

# REACTIVE ROBOT NAVIGATION USING OPTIMAL TIMING CONTROL

This chapter concerns the application of the switching time/mode scheduling optimization, presented in Chapters II and III, to a robotics problem. The problem in question is that of a mobile robot moving towards a goal point while avoiding potential obstacles in its path. Furthermore, the robot has to do this in real time. Using Algorithm 3.2.1, presented in Chapter III for optimizing over the mode sequence, it will be shown how a switching surface can be generated for when the robot should switch between a *go-to-goal* and an *avoid-obstacle* behavior. This switching surface will then be used on a real robotics platform operating in an unknown environment. One of the main contributions of this Chapter is the derivation of the general shape of the switching surface, for when to switch between the different behaviors.

The approach taken in this chapter differs from the one in Chapter IV although we consider a similar problem. In this Chapter, a suboptimal solution to the *go-to-goal avoid-obstacle* problem will be presented based on deriving guards off line, hence no online optimization is required. The benefit of this approach is that the robot can perform well with small computational overhead. However, this does not mean that the legitimacy of the results of Chapter IV are questioned as the robotics system is a very special system and there are no hopes of deriving guards for more complicated systems.

It should be noted, already at this point, that even though the solution to the guard generation problem is obtained by optimizing over a well-defined and known environment, the resulting navigation strategy will be transitioned onto a real robotic platform operating in an unknown environment. As such, it may no longer be optimal but rather correspond to a performance enhancing design strategy.

The outline of this Chapter is as follows: In Section 6.1 we formalize our problem,



and the behavior-based formalism is introduced. Section 6.2 is devoted to the solution to our problem and Section 6.3 is concerned with the development of guards, suitable for implementation. We then extend our solution in Section 6.4 to include the case when we have multiple obstacles and the Chapter concludes with a systematic evaluation of our solution on a real robotics platform in Section 6.5.

## 6.1 Behavior Based Robotics

In order to formally characterize the main design challenges associated with the coordination of a set of behaviors under consideration here, some comments about the basic behavioral building blocks must be made. Assume that the robot dynamics are given by

$$\dot{x} = f(x, u), \quad (146)$$

where  $x \in \mathbb{X}$  is the robot state and  $u \in \mathbb{U}$  is the control input. We identify individual behaviors with feedback laws defined with respect to a particular task, data source, or operating point. In other words, the set of behaviors available to us are given by the set  $\{\kappa_1, \dots, \kappa_k\}$ , where each  $\kappa_i$  is a feedback mapping from  $\mathbb{X}$  to  $\mathbb{U}$ . Note that in the setting of Chapters II and III, a behavior corresponds to a distinct mode.

As an example we consider the unicycle dynamics introduced in Chapter V,

$$\begin{cases} \dot{x}_1 = v \cos(x_3), \\ \dot{x}_2 = v \sin(x_3), \\ \dot{x}_3 = u, \end{cases} \quad (147)$$

where  $(x_1, x_2)$  is the position of the robot and  $x_3$  is its heading. Assume that the translational velocity  $v$  is constant, and the angular velocity  $u$  is our control variable. In this Chapter we consider the following three behaviors

$$u_g = \kappa_g(x) = c_g(\phi_g - x_3), \quad (148)$$

$$u_{\odot} = \kappa_{\odot}(x, x_{ob}) = c_{ob}(\phi_{ob} - \frac{\pi}{2} - x_3), \quad (149)$$

$$u_{\ominus} = \kappa_{\ominus}(x, x_{ob}) = c_{ob}(\phi_{ob} + \frac{\pi}{2} - x_3), \quad (150)$$

where  $u_g$  is a standard “approach-goal” behavior,  $u_{\odot}$  and  $u_{\ominus}$  are “avoid-obstacle” behaviors that makes the robot move in a circle around the closest obstacle in the given direction

(clockwise and counter-clockwise respectively). Furthermore,  $c_g$  and  $c_{ob}$  are the gains associated with each behavior and  $\phi_g$  and  $\phi_{ob}$  are the angles to the goal and nearest obstacle respectively. Both of these angles are measured with respect to the  $x$ -axis and were expressed in Equations (144) and (145) in Chapter V.

Having designed a set of behaviors, the initial task that we want the robot to achieve is to reach a given goal location  $x_g \in \mathbb{R}^2$ , while staying clear of a point-obstacle located at  $x_{ob} \in \mathbb{R}^2$ . Even though the point-obstacle assumption is clearly unrealistic in a real environment, it still provides us with the appropriate design tools since obstacles are represented as points by almost all range sensors. Furthermore, it will be shown in Section 6.4 that the guard, derived for a single point-obstacle, can be extended to the case where several obstacles are present.

The instantaneous cost  $L : \mathbb{R}^2 \rightarrow \mathbb{R}^+$  is as defined in Chapter IV, Equation (125). There, the state variable were in  $\mathbb{R}^2$  while  $x \in \mathbb{R}^3$  here. However, the instantaneous cost is still only a function of  $x_1$  and  $x_2$  and for notational convenience, we will still write  $x - x_g$  when we mean  $(x_{g1} - x_1, x_{g2} - x_2)^T$  whenever the dimensions are clear from the context. Different instantaneous costs can be imagined but, as the main focus of this Chapter is to show that feasible robotic controllers can be provided in real time without giving up on optimality, we do not elaborate on that here.

The total cost associated with a particular trajectory then becomes

$$J = \int_0^T L(x(t))dt, \quad (151)$$

where  $T$  is the total time of the run.

As noted in the introduction, we will minimize the total cost by finding the best sequence of behaviors (given a bound on the number of switches) and optimal times when to switch between these behaviors. To formalize this, we let  $\mathcal{K} = \{g, \circlearrowleft, \circlearrowright\}$  and let  $\mathcal{K}^*$  denote the set of all finite length strings over  $\mathcal{K}$ . Hence,  $\mathcal{K}$  is the index set of behaviors. Furthermore, assuming that we switch  $N$  times, we denote by  $\tau_i, i \in \{1, \dots, N\}$  the time of the  $i$ 'th switch and let  $\bar{\tau} = (\tau_1, \dots, \tau_N)^T$  be the vector of switching times.

Then the switching time problem, denoted by  $P$ , can be cast as

$$\begin{aligned}
P : \quad & \min_{B, \bar{\tau}} J = \int_0^T L(x(t)) dt \\
s.t. \quad & \dot{x} = \begin{cases} f(x, u_{B_1}), & 0 \leq t < \tau_1, \\ \vdots & \vdots \\ f(x, u_{B_N}), & \tau_{N-1} \leq t < T, \end{cases} \\
& x(0) = x_0, \\
& 0 \leq \tau_1 \leq \dots \leq \tau_{N-1} \leq T,
\end{aligned} \tag{152}$$

where the dimension of  $\bar{\tau}$  is induced by  $B = (B_1, B_2, \dots, B_N)$ ,  $B \in \mathcal{K}^*$ , and the robot starts at  $x_0$ .

Now, given a collection of behaviors  $\mathcal{K}$ , solving Problem  $P$  will give a solution to the “approach-goal” and “avoid-obstacle” problem based on switching between behaviors in an optimal manner. However, other solutions can be imagined. For example, one could try to use several behaviors at the same time [52],[53]. Doing this in an optimal way would result in trajectory with a lower cost than using a pure switching control strategy since the solution given from the optimal switching controller can be obtained through combining the different modes. However, such a combination-based strategy destroys some of the desirable modular aspects of designing specific behaviors for specific tasks. If the behaviors do not have complete control over the robot, their functionality and performance can no longer be guaranteed [27]. Therefore, this Chapter only considers the switching time problem  $P$ .

## 6.2 Optimal Control Derivation

In order to obtain a (locally optimal) solution to Problem  $P$  we need to both find a good sequence of behaviors and the optimal times when to switch between them. To this end, we will approach the Problem  $P$  by answering the following two questions:

1. What is a good sequence of behaviors?
2. Given a sequence of behaviors  $B$ , how do we find the optimal switching times?

The answer to Question 2 above was presented in detail in Chapter II. In particular, Algorithm 2.2.1 will find a locally optimal switching time vector for a given sequence of behaviors. Likewise, in order to find a good sequence of behaviors, Algorithm 3.2.1 will be used, as presented in Chapter III. The reader is referred to these Chapters for details about the different Algorithms.

Note that Algorithm 3.2.1 may insert several new behaviors. However, it turns out that with our particular choice of instantaneous cost and behaviors, we do not get any significant reduction in cost by performing a second insertion after we have optimized over  $\bar{\tau}$  given by the first insertion. Therefore, we only consider performing one insertion in Algorithm 3.2.1.

The solution to  $P$  given by executing Algorithm 3.2.1 will indeed be locally optimal but it is not applicable to real time robotics problems since we need to calculate  $x(t)$  and  $p(t)$  for each iteration of Algorithm 3.2.1 which is time consuming. We would like to obtain a suboptimal solution where the optimal switches between the different behaviors are given by a geometric guard (switching surface) defined around the obstacle. Moreover, the structure of the guard should only depend on the distance between the obstacle and the goal. Hence, independent of where the robot starts, the guard should be the same.

In order to arrive at this result, it first needs to be proven that the solution is *invariant along trajectories*, given that the final time  $T$  is big enough. Invariance along trajectories means that the optimal solution starting at  $x_0$  switches at the same point in the state space as the optimal solution starting along the trajectory of the solution starting at  $x_0$ .

To this end, Assumption 6 and Lemma 6.2.1 are presented.

**Assumption 6** *Given Problem  $P$ , assume that the instantaneous cost  $L$  and the dynamic representation  $f(x, u)$ , associated with the different behaviors, are continuously differentiable and that  $L$  is bounded from above. Furthermore, assume that there exists a finite time  $t_1 < T$  such that the robot evolves according to a behavior  $\kappa_b$ , after time  $t_1$ , where  $\frac{\partial f(x, u_b)}{\partial x}$  is negative definite.*

Note that the statements in Assumption 6 are reasonable since we can assume that the goal and the obstacle are separated for the problem to be meaningful. Furthermore, we need

to choose  $T$  big enough to guarantee that the robot reaches the goal, hence the robot will evolve according to  $\kappa_g$  after some time  $t_1$ . The assumption that the robot evolves according to a behavior  $b$  that satisfies the constraint that  $\frac{\partial f(x, u_b)}{\partial x}$  is negative definite, should be seen as if we are requiring the robot to converge towards the goal position after a certain time.

We are now in position to present Lemma 6.2.1.

**Lemma 6.2.1** *Under Assumption 6: For an initial state  $x_0$ , denote by  $x(t)$  and by  $p(t)$  the state and the costate trajectories obtained when  $x(t)$  evolves according to (152) for a given switching vector  $\bar{\tau}^1$ . Denote by  $\bar{x}(t)$  and by  $\bar{p}(t)$  the state and the costate trajectories associated with the same system, but with an initial condition along the trajectory of  $x(t)$ , i.e.,  $\bar{x}(0) = x(\Delta)$  for some  $\Delta \in (0, T)$ . Furthermore, assume that the switching vector for the second system  $\bar{\tau}^2$  is identical to  $\bar{\tau}^1$  but with all switches increased by  $\Delta$ . Then the state and the costate trajectories satisfy the following two equations*

$$\bar{x}(t - \Delta) = x(t), \quad t \in [\Delta, T], \quad (153)$$

$$\lim_{T \rightarrow \infty} \bar{p}(t - \Delta) = p(t), \quad t \in [\Delta, T]. \quad (154)$$

*Proof:* See the Appendix.

Lemma 6.2.1 provides us with the result needed in order to show that Problem  $P$  satisfies the property of invariance along trajectories. To this end, assume that the robot starts at  $x_0 \in \mathbb{R}^3$  and evolves according to  $\kappa_g$ . We denote the trajectory corresponding to this by  $x^1(t)$ . Likewise, let  $x^2(t)$  be the state trajectory when we start at  $x^1(\Delta)$  for some time  $\Delta \in (0, T)$ , but evolve according to the same behavior. From Lemma 6.2.1 we know that (153) and (154) are in force and hence  $x^1(t) = x^2(t - \Delta)$  and  $p^1(t) = p^2(t - \Delta)$  for all finite times  $t \in [\Delta, T]$  as  $T \rightarrow \infty$ . Denote the cost associated with  $x^1(t)$  by  $J^1$ , and the cost associated with  $x^2(t)$  by  $J^2$ . We define  $\bar{\Delta}$  to be the vector with the same dimension as  $\bar{\tau}$  with each element equal to  $\Delta$ . Since  $\nabla J(\bar{\tau})$  and  $\lim_{\lambda \downarrow 0} \frac{dJ}{d\lambda}$  only depend on the state  $x(t)$  and the costate  $p(t)$  it follows that

$$\|\nabla J^1(\bar{\tau}) - \nabla J^2(\bar{\tau} - \bar{\Delta})\| = 0, \quad (155)$$

$$\left| \lim_{\lambda \downarrow 0} \frac{dJ^1_{b,t}}{d\lambda} - \lim_{\lambda \downarrow 0} \frac{dJ^2_{b,t-\Delta}}{d\lambda} \right| = 0, \quad (156)$$

are close to zero for all finite times  $t \in (\Delta, T)$  and for all behaviors  $b \in \{g, \circlearrowleft, \circlearrowright\}$ , as  $T \rightarrow \infty$ .

The implication of (155) and (156) is that  $\frac{dJ_{b,t}^1}{d\lambda}$  will be minimized at the same time  $\bar{t} \in (\Delta, T)$  and with the same behavior  $b$  as  $\frac{dJ_{b,t-\Delta}^2}{d\lambda}$ . Hence, given that  $\frac{dJ_{b,t}^1}{d\lambda} < 0$ , the insertion of a new behavior will occur at the same point in the state space for both systems. After the insertions,  $\bar{\tau}$  consists of two switching times but we still have  $\nabla J^1(\bar{\tau}) = \nabla J^2(\bar{\tau} - \Delta)$ . Hence Algorithm 3.2.1 will terminate at two distinct switching vectors:  $\bar{\tau}^1$  associated with  $x^1(t)$  and  $\bar{\tau}^2$  associated with  $x^2(t)$  such that  $\bar{\tau}^1 = \bar{\tau}^2 - \bar{\Delta}$ . Moreover, the switches occur at the same point in the state space.

Thus, we have shown that the solution to problem  $P$  is invariant along trajectories, e.g. if we start along an optimal trajectory, it is optimal to switch at the same points in the state space independently of where on the trajectory we start. This is exactly the result needed in order to generate the guards for when to switch from the *go-to-goal* behavior to the *obstacle-avoidance* behavior.

### 6.3 Guard Generation

In order to be able to generate the guard, we need to make sure that Assumption 6 is satisfied. In particular, we need to ensure that the robot evolves according to a behavior  $b$  such that  $\frac{\partial f(x, u_b)}{\partial x}$  is negative definite after a certain time. Unfortunately,  $u_g$  does not satisfy the above constraint due to the fact that it has a constant translational velocity  $v$ .

Therefore, and in order for the robot to arrive smoothly at the goal, a new behavior is introduced. The behavior is active whenever the robot is closer to the goal than one meter, and the behavior is such that its translational velocity is decreasing proportional to the distance to the goal, until zero velocity at the goal. Furthermore, it is assumed that the robot is heading straight towards the goal meaning that  $x_3 = \phi_g$ , where  $\phi_g$  is the angle to the goal, as defined in (144), and is assumed to be constant. This is a reasonable assumption since we assume that the goal and the obstacle are far apart, hence we can assume that the robot will be heading towards the goal once it is within a distance of one meter from the

goal. The behavior is defined by

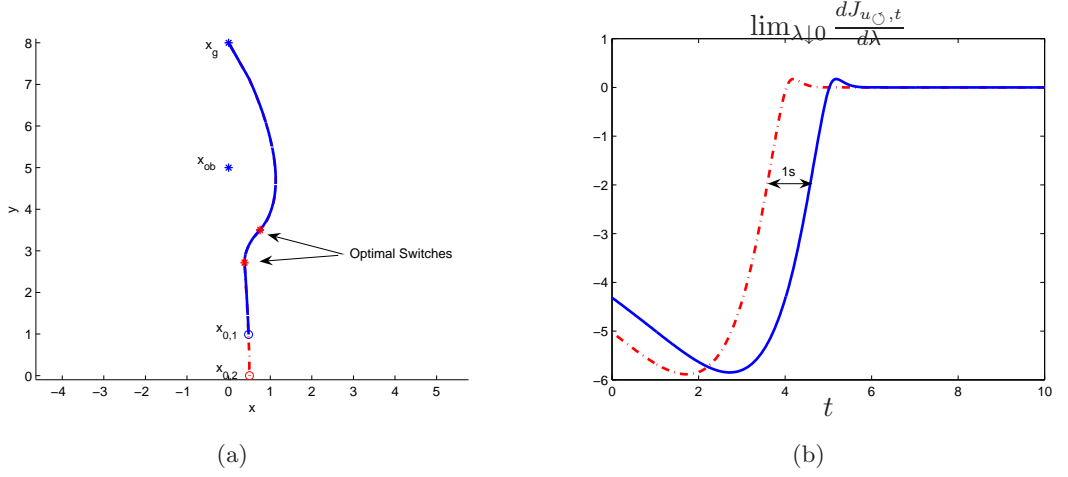
$$\begin{cases} \dot{x}_1 = \|y - x_g\| \cos(\phi_g), \\ \dot{x}_2 = \|y - x_g\| \sin(\phi_g), \\ \dot{x}_3 = 0, \end{cases}$$

where  $y = (x_1, x_2)^T$ . Here the control variable is the translational velocity  $v$  instead of  $u$ , a departure from the setting of Section 6.1, where  $v$  was assumed to be constant. Denoting the system above by  $\dot{x} = f(x, u_q)$  and taking the derivative with respect to  $x$ , we get that

$$\frac{\partial f(x, u_q)}{\partial x} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (157)$$

where we see that  $\frac{\partial f(x, u_q)}{\partial x}$  is only negative semi-definite. However,  $L$  does not depend on  $x_3$ , hence  $\frac{\partial L}{\partial x} = (\cdot, \cdot, 0)$ , has a zero as its third component. This guarantees that (154) holds even though  $\frac{\partial f(x, u_q)}{\partial x}$  is not negative definite.

Having defined  $f(x, u_q)$ , we have shown that all the claims made in Assumption 6 can be fulfilled for our system. What remains to be done is to mention that the property of invariance along trajectories, for our problem, is still true for all practical purposes even though the final time  $T$  is finite. To illustrate this, a solution to problem  $P$  for two different initial states (denoted by a  $x_{0,1}$  and  $x_{0,2}$ ) along the same trajectory is depicted in Figure 27(a). Both trajectories initially evolve according to  $B = (g)$ . An insertion is then performed. Here, the final time is 10 seconds,  $x_{ob} = (0, 5)^T$ ,  $x_g = (0, 8)^T$ , and the values for  $\rho$ ,  $\alpha$  and  $\beta$  are  $\rho = 1/100$ ,  $\alpha = 8$ , and  $\beta = 0.2$ . The switching times are marked with red stars. It is clear, from Figure 27(a), that the solution to problem  $P$  for both initial states switches at the same point in the state space even though the final time is finite. Furthermore, in Figure 27(b),  $\lim_{\lambda \downarrow 0} \frac{dJ_{u_\zeta, t}}{d\lambda}$  is depicted for both state trajectories. As can be seen from the figure, both curves are similar in shape but there is a time offset between them corresponding to the fact that the solid curve starts 1 second in along the trajectory of the dotted curve. From Figure 27(b) it is moreover clear that the insertion will occur at the same point in the state space. We can now proceed to derive the suboptimal geometric



**Figure 27:** Different Initial States: (a) The optimal state trajectories for both initial states are shown. (b)  $\lim_{\lambda \downarrow 0} \frac{dJ_{u_O, t}}{d\lambda}$  is shown for both trajectories as function of  $t$  when  $B = (g)$ .

guard for problem  $P$ . In order to get the data needed to generate the guard for a given distance between the obstacle and the goal we execute the following Algorithm:

**Algorithm 6.3.1** *Guard-generation Algorithm:*

*Given:* The parameters of Problem  $P$ .

*Init:* Select a finite set of representative initial states  $X_0$ .

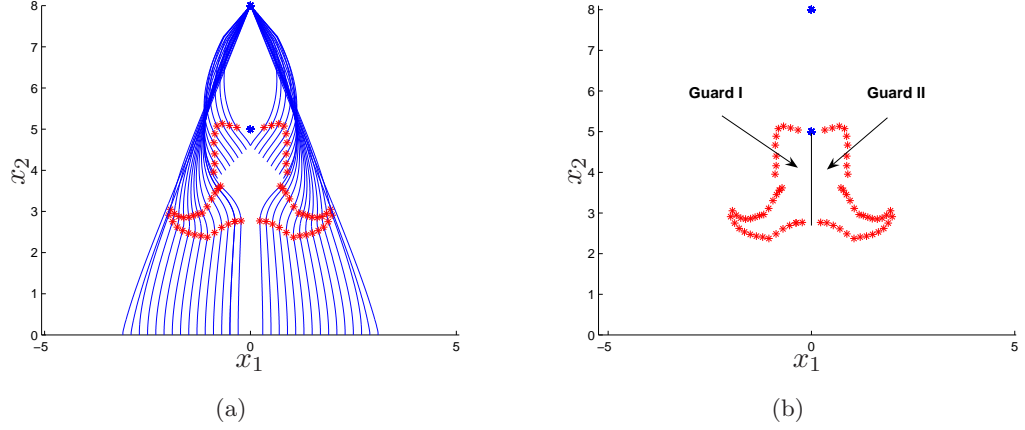
*Step 1:* If  $X_0 = \emptyset$  STOP. Else, select an initial state  $x_0 \in X_0$  and execute Algorithm 3.2.1 for one insertion, starting from this state. Save the switching positions obtained for the optimal trajectory. Remove  $x_0$  from  $X_0$ . Go to Step 1.

Note that we assume that Algorithm 3.2.1 only performs one insertion in Algorithm 6.3.1.

An example of the guard obtained when executing Algorithm 6.3.1 is shown in Figure 28. There,  $x_{ob} = (0, 5)^T$ ,  $x_g = (0, 8)^T$ , and the values for  $\rho$ ,  $\alpha$  and  $\beta$  are as before. As can be seen, the guard lies in  $\mathbb{R}^2$ , corresponding to the robots position, even though  $x \in \mathbb{R}^3$ . Hence, the robots direction is implicit in Figure 28(a). The justification for this is that we assume that the robot is directed towards the goal when it encounters the obstacle. It should be noted that a guard in  $\mathbb{R}^3$  can be generated, but for simplicity of our exposition, we do not present such a guard.

By examining Figure 28(a), we see that whenever the robot is inside the region denoted





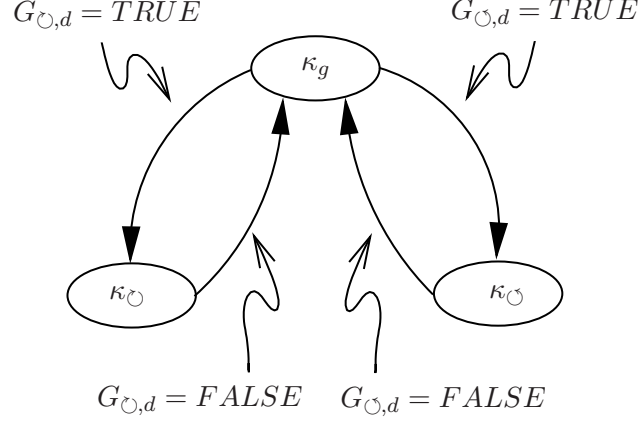
**Figure 28:** State trajectories and associated guard structure: (a) State trajectories for executing Algorithm 6.3.1 for a set of initial states along the  $x_1$ -axis. (b) An approximation of the associated guards. Guard I corresponds to the region inside the stars and to the left of the vertical line between the goal and the obstacle, similar for Guard II.

by Guard I in Figure 28(b), it is optimal to let the robot evolve according to  $\kappa_{\odot}$ . Likewise, if the robot is inside the region denoted by Guard II in Figure 28(b), it is optimal to let the robot evolve according to  $\kappa_{\odot}$ . Everywhere else it is optimal to use  $\kappa_g$ . The regions where it is beneficial to evolve according to  $\kappa_{\odot}$  or  $\kappa_{\odot}$  can approximately be described by a set of linear matrix inequalities together with some additional logic (the additional logic is needed since the guards are not convex). To this end, we let  $A_{\odot,d} \cdot y \leq b_{\odot,d}$ , where  $y = (x_1, x_2)^T$ , denote the linear matrix inequality corresponding to Guard I in Figure 28(b) and similar for Guard II.

At this point it should be noted that given our instantaneous cost and our behaviors, the structure of the guards depends *only* on the distance between the goal and the obstacle, denoted by  $d$ . As we change  $d$  it might be conceivable that we get a big change in the  $A$  and  $b$  matrices but simulation shows that this is not the case for our range of distances. Hence, we denote by  $A_{\odot,d}$  and  $b_{\odot,d}$  the linear matrix inequality associated with Guard I in Figure 28(b) where subscript  $d$  denoted the distance between the goal and the obstacle. Then, under the assumption that the obstacle and the goal lie on the  $x_2$ -axis, the guards associated with  $\kappa_{\odot}$  and  $\kappa_{\odot}$ , denoted by  $G_{\odot,d}$  and  $G_{\odot,d}$  can be expressed as functions of  $A_{\odot,d}$ ,  $b_{\odot,d}$  and  $A_{\odot,d}$ ,  $b_{\odot,d}$  respectively. If the goal and obstacle do not line up, a simple

rotation and translation is needed.

From the assumption that  $x_{ob}$  and  $x_g$  are far enough apart, we argued earlier that it was enough to consider the following three sequences of behaviors  $B = (g)$ ,  $B = (g, \circlearrowleft, g)$  and  $B = (g, \circlearrowright, g)$ . From this it follows that we never switch between  $\kappa_{\circlearrowleft}$  and  $\kappa_{\circlearrowright}$ . Therefore the optimal solution is given in terms of the guards  $G_{\circlearrowleft, d}$  and  $G_{\circlearrowright, d}$  and the optimal solution can be cast on the form of Figure 29.



**Figure 29:** The optimal solution to problem  $P$  given in terms of guards associated with each point obstacle encountered in the robots path.

Once we have calculated  $G_{\circlearrowleft, d}$  and  $G_{\circlearrowright, d}$  for a range of distances  $d$ , this solution is suitable for real time applications since the guards are easily stored and evaluated. For simplicity of our exposition, we do not plot the state corresponding to  $\kappa_g$ , i.e. when the robot is closer to the goal than one meter.

## 6.4 Multiple Obstacles

In the previous section, the guards for when to switch between the different behaviors were derived under the assumption that the robot only encountered a single point-obstacle in its path. In order to extend the results to the multiple obstacle scenario, some further remarks are due.

Having several obstacles and several range sensor readings, several obstacles might be detected at the same time. For each obstacle detected, a guard will be generated around it. If the robot currently is in the go-to-goal behavior, it will switch to an obstacle-avoidance

behavior if it is inside any of the guards generated around the obstacles associated with the sensor readings. In particular, if the robot is inside more than one guard, it will switch to the obstacle avoidance behavior that is associated with the closest obstacle. This way of structuring the switching law ensures that the robot will avoid the closest obstacle first, which makes the switching law robust. For each obstacle encountered in the robots path, it will continue to evolve according to the behavior it first switched to, e.g. if the robot switched to  $\kappa_{\odot}$  when it first encountered the obstacle, it will continue to evolve according to  $\kappa_{\odot}$  until it is outside all guards associated with any sensor reading and then switch back to  $\kappa_g$ . Hence, the robot will move in a given direction going around the obstacle. This way of designing the switching law both guarantees that the robot will move around the obstacle, if there is a traversable path, and that the robot has a clear course in front of it when it switches back to the go to goal behavior. The inspiration of this switching law comes from [39]. The switching law described above gives a solution to the go-to-goal avoid-obstacle problem but the resulting solution might no longer be optimal. Instead, we view the switching law as a performance enhancing design strategy.

In order to formalize the discussion above, assume that the robot is equipped with  $k$  sensors and let  $S = \{s_1, s_2, \dots, s_k\}$  denote the sensor readings. From each reading  $s_j \in S$ , we get the position of the obstacle  $x_{ob}^j \in \mathbb{R}^2$  associated with sensor  $j$ , and every obstacle  $x_{ob}^j$  has the two guards  $G_{\odot,d,j}$  and  $G_{\ominus,d,j}$  associated with it. Note that the distance  $d$  follows from the index  $j$  of the current sensor since we know the position of the goal. Hence, the  $d$  dependence in the guards will be omitted. Assume that the robot evolves according  $\kappa_g$ . We then let the robot switch to the behavior associated with the shortest sensor reading, i.e. if  $s_j \leq s$ ,  $\forall s \in S$  and  $x \in G_{\odot,j}$ , then the robot should switch to  $\kappa_{\odot}$ . We denote the index of the sensor corresponding to the closest obstacle by  $c^*$  where  $c^* = \arg \min\{s_j, j \in S\}$ .

From the definition of  $\kappa_{\odot}$ , when we only have one obstacle  $x_{ob}$ ,  $\kappa_{\odot}$  is a function of both  $x$  and  $x_{ob}$ . When there are several obstacles, we let  $\kappa_{\odot}(x, x_{ob}^{c^*})$  so that the robot moves in a circle around the obstacle that is closest to the robot.

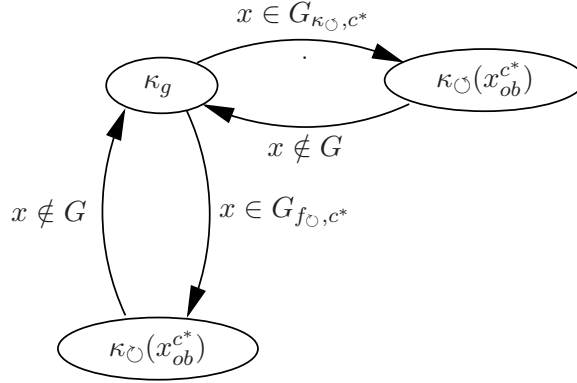
The robot will continue to be in the obstacle avoidance behavior until the robot is outside all guards associated with any sensor reading and then switch back to  $\kappa_g$ . To this

end, we define

$$G = \bigcup_{i=1}^k G_{\odot,i} \cup G_{\odot,i} \quad (158)$$

to be the union of all switching surfaces at a given point  $x$ . Note that the robot will have a clear course when it switches back to  $\kappa_g$  since it will be outside  $G$ .

The behavior of the robot can be described by the automaton presented in Figure 30, where we see that the optimal solution can be cast in terms of the guards  $G_{\odot,j}$  and  $G_{\odot,j}$ ,  $\forall j \in \{1, \dots, k\}$ .



**Figure 30:** Optimal solution given in terms of guards associated with each obstacle encountered in the robots path for the case when the robot encounters several obstacles.

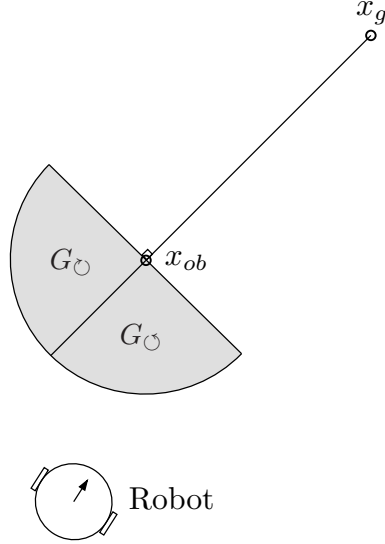
## 6.5 Implementation

In order to verify that the proposed navigation strategy perform well when implemented on a real robotics platform, it was tested on the Magellan Pro platform from iRobot with the setup shown in Figure 32.

Note that the optimal switching surfaces were designed for the known environment, presented in Section 6.3, where we only considered a single-point obstacle. Even though we extended this result in Section 6.4, the resulting control strategy will not be optimal in the setting above. Instead, the controller is expected to produce a good, suboptimal cost-reducing strategy.

To illustrate this point, we compare the performance of the optimal strategy with a

situation where the switching surface was given by a semi-circle, with a given radius, generated around the obstacle. The semi-circle has the arced segment on the opposite side of the goal, while the line segment is perpendicular to the goal, as illustrated in Figure 31. The robot will thus switch to the appropriate go-around-obstacle behavior when it gets inside this semi-circle.



**Figure 31:** Switching surfaces for experimental comparison. The “standard” switching surface is defined by a semi-circle where the robot should evolve according to  $\kappa_{\circlearrowleft}$  if the robot is inside  $G_{\circlearrowleft}$  and vice versa for  $G_{\circlearrowright}$ .

The result of this test is shown in Figure 33. The resulting trajectories as well as the detected obstacles are plotted using the odometry and sensor readings from the robot. Figure 33 shows the resulting trajectory using a switching controller with our optimal switching surface and the standard semi-circle switching surface, with a radius of 0.5 meters.

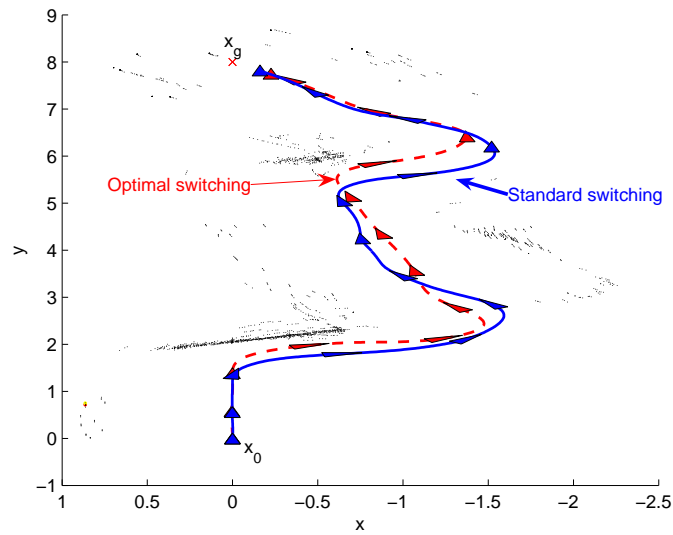
The costs for each trajectory computed according to (151) were 16.64 and 10.64 for the semi-circle approach and our optimal guard approach respectively. Hence, we see that the optimal method gives a lower cost than the standard counterpart, as should be expected.

## 6.6 Conclusions

In this Chapter, an optimal control inspired solution for how to control a robot moving towards a goal point while avoiding obstacles was presented. The solution was based on



**Figure 32:** Experimental setup for testing our real time reactive navigation method.



**Figure 33:** The simulated trajectory plotted using the odometry and sensor readings from the Magellan Pro for our Optimal Transitions System.

deriving suboptimal guards for when to switch between a go-to-goal behavior and two obstacle avoidance behaviors. Hence, the solution combines results from both optimal control theory and behavior based robotics. Once the guards were calculated, through simulations off line, they were approximated and transitioned onto a real robot platform.

The guards can then easily be used on line by the robot, and therefore the solution lends itself nicely to real time implementations.

One of the main contributions of this Chapter was the derivation of the general shape for how the suboptimal guard, for when to switch to an obstacle avoidance behavior, should look like. Another contribution was to extend this suboptimal guard to the case when we had several range-sensor readings. Furthermore, the Chapter served as a real world application to the switching time/mode-scheduling optimization results presented in Chapter II and III.

## APPENDIX A

### PROOFS

**Proof of Lemma 2.1.2.** The equations defining  $x_1(t)$  and  $x_2(t)$ , and the fact that they have a common initial condition  $x_0$ , imply that  $\Delta x(t) = \int_0^t (h_2(x_2(\tau), \tau) - h_1(x_1(\tau), \tau)) d\tau$ , and adding and subtracting  $h_2(x_1(\tau), \tau)$ , we obtain,

$$\Delta x(t) = \int_0^t (h_2(x_2(\tau), \tau) - h_2(x_1(\tau), \tau) + \Delta h(x_1(\tau), \tau)) d\tau. \quad (159)$$

Property 3 in the definition of  $H[C; K_1; K_2; \Gamma]$  asserts that  $\|h_2(x_2(\tau), \tau) - h_2(x_1(\tau), \tau)\| \leq K_1 \|\Delta x(\tau)\|$  for some  $K_1 \geq 0$  and for all  $\tau \in [0, T]$ . Consequently, and by (159), we have that  $\|\Delta x(t)\| \leq K_1 \int_0^t \|\Delta x(\tau)\| d\tau + \int_0^t \|\Delta h(x_1(\tau), \tau)\| d\tau$ . Bellman-Gronwall's inequality now implies (11) with  $K := e^{K_1 T}$  and the first inequality in the lemma has been proven.

Next, consider (12). Recall that  $\Phi(t, \tau)$  is the state transition function of the linearized system  $\dot{z} = \frac{\partial h_1}{\partial x}(x_1(t), t)z$ , and hence,  $\Phi(t, \tau) = \int_\tau^t \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \Phi(\zeta, \tau) d\zeta + I$ . Consequently, we have that  $\int_0^t \Phi(t, \tau) \Delta h_1(x_1(\tau), \tau) d\tau = \int_0^t (\int_\tau^t \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \Phi(\zeta, \tau) d\zeta + I) \Delta h(x_1(\tau), \tau) d\tau$ . Changing the order of integration we obtain,

$$\begin{aligned} \int_0^t \Phi(t, \tau) \Delta h(x_1(\tau), \tau) d\tau &= \int_0^t \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \int_0^\zeta \Phi(\zeta, \tau) \Delta h(x_1(\tau), \tau) d\tau d\zeta + \\ &\quad \int_0^t \Delta h(x_1(\zeta), \zeta) d\zeta. \end{aligned} \quad (160)$$

Now by (159) and (160) we obtain the following equation (notice that  $\tau$  is being replaced by  $\zeta$  in the integrand of (159)):

$$\begin{aligned} &\|\Delta x(t) - \int_0^t \Phi(t, \tau) \Delta h(x_1(\tau), \tau) d\tau\| = \\ &= \|\int_0^t (h_2(x_2(\zeta), \zeta) - h_2(x_1(\zeta), \zeta) + \Delta h(x_1(\zeta), \zeta) - \\ &\quad \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \int_0^\zeta \Phi(\zeta, \tau) \Delta h(x_1(\tau), \tau) d\tau - \Delta h(x_1(\zeta), \zeta)) d\zeta\| = \\ &= \|\int_0^t (h_2(x_2(\zeta), \zeta) - h_2(x_1(\zeta), \zeta) - \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \int_0^\zeta \Phi(\zeta, \tau) \Delta h(x_1(\tau), \tau) d\tau) d\zeta\|. \end{aligned} \quad (161)$$



By the mean value theorem, there exists  $s(\zeta) \in [0, 1]$  such that,

$$h_2(x_2(\zeta), \zeta) - h_2(x_1(\zeta), \zeta) = \frac{\partial h_2}{\partial x}(x_1(\zeta) + s(\zeta)\Delta x(\zeta), \zeta)\Delta x(\zeta). \quad (162)$$

Furthermore,

$$\begin{aligned} \frac{\partial h_2}{\partial x}(x_1(\zeta) + s(\zeta)\Delta x(\zeta), \zeta) &= \left( \frac{\partial h_2}{\partial x}(x_1(\zeta) + s(\zeta)\Delta x(\zeta), \zeta) - \frac{\partial h_2}{\partial x}(x_1(\zeta), \zeta) \right) + \\ &+ \left( \frac{\partial h_2}{\partial x}(x_1(\zeta), \zeta) - \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \right) + \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta). \end{aligned} \quad (163)$$

Therefore, and by (161) and (162), we have that

$$\begin{aligned} &\|\Delta x(t) - \int_0^t \Phi(t, \tau)\Delta h(x_1(\tau), \tau)d\tau\| = \\ &= \|\int_0^t \left( \left( \frac{\partial h_2}{\partial x}(x_1(\zeta) + s(\zeta)\Delta x(\zeta), \zeta) - \frac{\partial h_2}{\partial x}(x_1(\zeta), \zeta) \right)\Delta x(\zeta) + \frac{\partial \Delta h}{\partial x}(x_1(\zeta), \zeta)\Delta x(\zeta) + \right. \\ &\quad \left. + \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta)\Delta x(\zeta) - \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \int_0^\zeta \Phi(\zeta, \tau)\Delta h(x_1(\tau), \tau)d\tau \right) d\zeta\|. \end{aligned} \quad (164)$$

By property 4 of the definition of  $H[C; K_1; K_2; \Gamma]$  we have that  $\|\frac{\partial h_2}{\partial x}(x_1(\zeta) + s(\zeta)\Delta x(\zeta), \zeta) - \frac{\partial h_2}{\partial x}(x_1(\zeta), \zeta)\| \leq K_2\|\Delta x(\zeta)\|$ , and by (11),  $\|\Delta x(\zeta)\| \leq K \int_0^T \|\Delta h(x_1(\zeta), \zeta)\|d\zeta$ . Consequently, and by (164), we have that,

$$\begin{aligned} &\|\Delta x(t) - \int_0^t \Phi(t, \tau)\Delta h(x_1(\tau), \tau)d\tau\| \leq \\ &\leq K_2TK^2 \left( \int_0^T \|\Delta h(x_1(\zeta), \zeta)\|d\zeta \right)^2 + K \int_0^T \left\| \frac{\partial \Delta h}{\partial x}(x_1(\zeta), \zeta) \right\|d\zeta \int_0^T \|\Delta h(x_1(\zeta), \zeta)\|d\zeta + \\ &+ \int_0^t \left\| \frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta) \right\| \left( \|\Delta x(\zeta) - \int_0^\zeta \Phi(\zeta, \tau)\Delta h(x_1(\tau), \tau)d\tau\| \right) d\zeta. \end{aligned} \quad (165)$$

By property 3 of the definition of  $H[C; K_1; K_2; \Gamma]$ ,  $\|\frac{\partial h_1}{\partial x}(x_1(\zeta), \zeta)\| \leq K_1$ . Therefore, and by (165), there exists  $\tilde{K} > 0$  such that,

$$\begin{aligned} &\|\Delta x(t) - \int_0^t \Phi(t, \tau)\Delta h(x_1(\tau), \tau)d\tau\| \leq \\ &\leq \tilde{K} \left( \int_0^T \|\Delta h(x_1(\zeta), \zeta)\|d\zeta \right) \left( \int_0^T \|\Delta h(x_1(\zeta), \zeta)\|d\zeta + \int_0^T \left\| \frac{\partial \Delta h}{\partial x}(x_1(\zeta), \zeta) \right\|d\zeta \right) + \\ &+ K_1 \int_0^T \left( \|\Delta x(\zeta) - \int_0^\zeta \Phi(\zeta, \tau)\Delta h(x_1(\tau), \tau)d\tau\| \right) d\zeta. \end{aligned} \quad (166)$$

Eq. (12) now follows by Bellman-Gronwall's inequality with  $\bar{K} := \tilde{K}e^{K_1T}$ . ■

**Proof of Proposition 2.2.3.** Let us denote by  $\ell_q$  the value of  $\ell$  when Algorithm 2.2.2 enters its Step 1 in the  $q^{th}$  time,  $q = 1, 2, \dots$ . Likewise, let  $m_q$  denote the value of  $m$

computed in Step 1 at the  $q^{th}$  iteration. Thus,  $\ell_1 = k$ ,  $m_1$  is computed by Step 1 in the first iteration,  $\ell_2 = m_1 + 1$  unless  $m_1 = n$ , etc. Correspondingly, we denote by  $r_{max,q}$  the value of  $r_{max}$  computed via (33) in the  $q^{th}$  iteration of the algorithm.

Let  $\bar{h} = (h_1, \dots, h_N)^T$  be the vector computed by Algorithm 2.2.2 as applied to every maximal block. We first ascertain that  $\bar{h} \in \Psi(\bar{\tau})$ . Observe that  $\bar{h} \in \Psi(\bar{\tau})$  if and only if, for every maximal block such as  $\{k, \dots, n\}$ , the following three equations are in force.

$$h_k \geq 0 \text{ if } \tau_k = 0. \quad (167)$$

$$h_i \leq h_{i+1} \text{ for all } i \in \{k, \dots, n-1\}. \quad (168)$$

$$h_n \leq 0 \text{ if } \tau_n = T. \quad (169)$$

We next prove that these three conditions are indeed in force.

Suppose first that  $0 < \tau_k$  and  $\tau_n < T$ ; the cases where  $\tau_k = 0$  or  $\tau_n = T$  will be considered later. Then, only (168) has to be ascertained. For all  $q = 1, 2, \dots$ , and for all  $i \in \{\ell_q, \dots, m_q\}$ ,  $h_i = -r_{\ell_q, m_q}$  by Step 1. Therefore, (168) will be established once we show that

$$r_{\ell_q, m_q} > r_{\ell_{q+1}, m_{q+1}}. \quad (170)$$

This is what we next do. By Step 1,  $r_{\ell_q, m_q} > r_{\ell_q, m_{q+1}}$ . This means, by (34), that

$$\frac{1}{m_q - \ell_q + 1} \sum_{j=\ell_q}^{m_q} \frac{dJ(\bar{\tau})}{d\tau_j} > \frac{1}{m_{q+1} - \ell_q + 1} \sum_{j=\ell_q}^{m_{q+1}} \frac{dJ(\bar{\tau})}{d\tau_j}. \quad (171)$$

Consequently, and after some algebra, we obtain that

$$\frac{1}{m_{q+1} - m_q} \sum_{j=m_q+1}^{m_{q+1}} \frac{dJ(\bar{\tau})}{d\tau_j} < \frac{1}{m_q - \ell_q + 1} \sum_{j=\ell_q}^{m_q} \frac{dJ(\bar{\tau})}{d\tau_j}. \quad (172)$$

Applying the fact that  $\ell_{q+1} = m_q + 1$  (by definition) to the left-hand side of (172), (170) follows by (34).

Next, consider the case where  $\tau_{m_q} = 0$ . By Step 2, we see that, for all  $i \in \{\ell_q, \dots, m_q\}$ ,  $h_i = \max\{-r_{m_q}, 0\}$ , and this ascertains that both (167) and (168) are in force. Finally, it follows in a similar way that, if  $\tau_n = T$ , then (168) and (169) hold true. This establishes that  $\bar{h} \in \Psi(\bar{\tau})$ .

We next prove that  $\bar{h}$  indeed is the projection of  $-\nabla J(\bar{\tau})$  onto  $\Psi(\bar{\tau})$ . To be the projection,  $\bar{h}$  has to solve the following quadratic program, denoted by  $Q$ .

$$Q : \min \left\{ \frac{1}{2} \|\tilde{h} + \nabla J(\bar{\tau})\|^2 : \tilde{h} \in \Psi(\bar{\tau}) \right\}. \quad (173)$$

Associated with the vector  $\bar{h}$ , let us define the vector  $\bar{h}_{k,n} \in R^{n-k+1}$  by  $\bar{h}_{k,n} = (h_k, \dots, h_n)^T$ . Similarly, we define the vector  $\nabla_{k,n} J(\bar{\tau}) \in R^{n-k+1}$  by  $\nabla_{k,n} J(\bar{\tau}) = \left( \frac{dJ(\bar{\tau})}{d\tau_k}, \dots, \frac{dJ(\bar{\tau})}{d\tau_n} \right)^T$ . Now the condition  $\bar{h} \in \Psi(\bar{\tau})$  is equivalent to the constraint that, for every maximal block like  $\{k, \dots, n\}$ , (167)-(169) are in force. Therefore,  $\bar{h}$  solves the quadratic program  $Q$  if and only if, for every maximal block like  $\{k, \dots, n\}$ ,  $\bar{h}_{k,n}$  solves the following quadratic program, denoted by  $Q_{k,n}$ .

$$\begin{aligned} Q_{k,n} : \quad & \min \left\{ \frac{1}{2} \|\tilde{h}_{k,n} + \nabla_{k,n} J(\bar{\tau})\|^2 \right. \\ & : \text{ Eqs. (167) - (169) are satisfied} \}. \end{aligned} \quad (174)$$

This is a quadratic program that has a unique solution point, which is also the only point satisfying the Lagrange multiplier rule for  $Q_{k,n}$ . We now show that this point is  $\bar{h}_{k,n}$ .

We observe that for every  $q = 1, 2, \dots$ , and for every  $i \in \{\ell_q, \dots, m_q - 1\}$ ,  $h_i - h_{i+1} = 0$ , i.e., the constraint  $h_i - h_{i+1} \leq 0$  is active. Moreover, by Step 2 and (170),  $h_{m_q} < h_{m_q+1}$ , and hence the constraint  $h_{m_q} - h_{m_q+1} \leq 0$  is not active. Therefore, to satisfy the Lagrange multiplier rule for  $Q_{k,n}$ , there must exist a multiplier  $\lambda_i \geq 0$  associated with the inequality constraint  $h_i - h_{i+1} \leq 0$ ,  $i = \ell_q, \dots, m_q - 1$ ; in the case where  $q = 1$  and hence  $\ell_q = k$ , if  $\tau_k = 0$  and  $h_k = 0$  then there exists a multiplier  $\mu \geq 0$  associated with the inequality  $-h_k \leq 0$ ; and in the case where  $m_q = n$ , if  $\tau_n = T$  and  $h_n = 0$  then there exists a multiplier  $\nu \geq 0$  associated with the constraint  $h_n \leq 0$ . The latter two situations cannot arise simultaneously since  $\tau_k = \tau_n$  (due to the fact that the set  $\{k, \dots, n\}$  constitutes a maximal block), therefore we can assume, without loss of generality, that  $\tau_1 > 0$  and consider only the possible case where  $\tau_n = T$ .

Consider first the case where either  $\tau_n < T$  or  $h_n < 0$ . Then, the above Lagrange multiplier rule means that the following three equations are in force,

$$h_{\ell_q} + \frac{dJ(\bar{\tau})}{d\tau_{\ell_q}} + \lambda_{\ell_q} = 0, \quad (175)$$

$$h_i + \frac{dJ(\bar{\tau})}{d\tau_i} + \lambda_i - \lambda_{i-1} = 0 \quad \text{for all } i = \ell_q + 1, \dots, m_q - 1, \quad (176)$$

$$h_{m_q} + \frac{dJ(\bar{\tau})}{d\tau_{m_q}} - \lambda_{m_q-1} = 0. \quad (177)$$

For every  $j \in \{\ell_q + 1, \dots, m_q - 1\}$ , summing up (175) with (176) for all  $i = \ell_q + 1, \dots, j$ , we obtain,

$$\sum_{i=\ell_q}^j h_i + \sum_{i=\ell_q}^j \frac{dJ(\bar{\tau})}{d\tau_i} + \lambda_j = 0. \quad (178)$$

Likewise, summing up over all  $i \in \{\ell_q + 1, \dots, m_q - 1\}$  and adding (175) and (177), we get that

$$\sum_{i=\ell_q}^{m_q} h_i + \sum_{i=\ell_q}^{m_q} \frac{dJ(\bar{\tau})}{d\tau_i} = 0. \quad (179)$$

In fact, it is readily seen that the condition defined by (175)-(177) is equivalent to the condition defined by (175), (178) for all  $j \in \{\ell_q + 1, \dots, m_q - 1\}$ , and (179). Since  $\lambda_i \geq 0$ , the latter condition amounts to the condition defined by the following two equations.

$$\sum_{i=\ell_q}^j h_i + \sum_{i=\ell_q}^j \frac{dJ(\bar{\tau})}{d\tau_i} \leq 0, \quad j = \ell_q, \dots, m_q - 1; \quad (180)$$

$$\sum_{i=\ell_q}^{m_q} h_j + \sum_{i=\ell_q}^{m_q} \frac{dJ(\bar{\tau})}{d\tau_i} = 0. \quad (181)$$

By (34) and Step 2, (180) amounts to

$$-(j - \ell_q + 1)r_{\ell_q, m_q} - \ell_q, m_q + (j - \ell_q + 1)r_{\ell_q, j} \leq 0, \quad (182)$$

and (181) means that

$$-(m_q - \ell_q + 1)r_{\ell_q, m_q} + (m_q - \ell_q + 1)r_{\ell_q, m_q} = 0. \quad (183)$$

(182) is satisfied by the maximality of  $m_q$  (Eq. (33)), and (183) is certainly true. This shows that  $\bar{h}_{k,n}$  satisfies the Lagrange multiplier rule for  $Q_{k,n}$ , and hence it is the solution point for that quadratic program.

Finally, consider the case where  $m_q = n$ ,  $\tau_n = T$ , and  $h_n = 0$ . Then we have the additional Lagrange multiplier  $\nu \geq 0$  associated with the active inequality constraint  $\tau_n - T \leq 0$ . Consequently, the optimality condition amounts to (180) for all  $i = \ell_q, \dots, n - 1$ ,

but not (181); instead, we have the condition  $h_n = 0$ . By Step 2, the latter condition means that  $h_i = 0$  for all  $i = \ell_q, \dots, n$ , and, moreover, this can occur if and only if  $r_{\ell_q, n} \leq 0$ . In this case it is readily seen that (180) is in force, since  $h_i = 0$  and (by (34))

$$\sum_{i=\ell_q}^j \frac{dJ(\bar{\tau})}{d\tau_i} \leq (j - \ell_q + 1)r_{\ell_q, j} \leq (j - \ell_q + 1)r_{\ell_q, n} \leq 0$$

for all  $j = \ell_q, \dots, n$ . This shows that  $\bar{h}_{k, n}$  indeed solves the quadratic program  $Q_{k, n}$ , which completes the proof.  $\blacksquare$

**Proof of Proposition 4.2.1.** For a given step size  $\delta t$ , the error after the first step is given by,

$$E(\delta t, \delta t) = \sup \|x_s[\delta t, \delta t] - x(\delta t)\|, \quad (184)$$

where the supremum represents the fact that  $x$  and  $x_s$  are not known before the simulation is done. Using Assumption 4, the following upper bound on the right hand side of (184) is obtained,

$$\begin{aligned} \|x_s[\delta t, \delta t] - x(\delta t)\| &= \|x_0 + \delta t f(x_s[0, \delta t], 0) - \\ &(x_0 + \int_0^{\delta t} f(x, t) dt)\| \leq L_f \int_0^{\delta t} \|x(t) - x_s[0, \delta t]\| dt. \end{aligned} \quad (185)$$

Furthermore,  $\|x(t) - x_s[0, \delta t]\| \leq \|x_0 + \int_0^{\delta t} C_f dt - x_0\| \leq C_f \delta t$ ,  $\forall t \in [0, \delta t]$ , and putting this into (185), it follows that  $E(\delta t, \delta t) = L_f C_f (\delta t)^2$ . A similar analysis for the least upper bound of  $\|x(2\delta t) - x_s[2\delta t, \delta t]\|$  results in,

$$\begin{aligned} E(2\delta t, \delta t) &= L_f C_f (\delta t)^2 + L_f [C_f L_f (\delta t)^2 + \delta t C_f] = \\ &E(\delta t, \delta t) + L_f (E(\delta t, \delta t) + C_f \delta t). \end{aligned} \quad (186)$$

Likewise,  $E(3\delta t, \delta t) = E(2\delta t, \delta t) + L_f (E(2\delta t, \delta t) + C_f \delta t)$ . From  $E(\delta t, \delta t)$ ,  $E(2\delta t, \delta t)$ , and  $E(3\delta t, \delta t)$  it can be deduced that the general form for  $E$  at time  $t \in M(t_f, \delta t)$  is given by

$$\begin{aligned} E(t, \delta t) &= \sum_{i=0}^{\frac{t}{\delta t}-1} (1 + L_f \delta t)^i L_f C_f (\delta t)^2 = \\ &\left[ (1 + L_f \delta t)^{\frac{t}{\delta t}} - 1 \right] C_f \delta t, \end{aligned} \quad (187)$$

where we note that the right hand side of (187) is bounded by  $[e^{L_f t} - 1] C_f \delta t$ , hence  $\|x_s[t, \delta t] - x(t)\| \leq E(t, \delta t) \leq (e^{L_f t} - 1) C_f \delta t$ , which completes the proof.  $\blacksquare$

**Proof of Proposition 4.2.2.** As  $T + t_c = t_f$ ,  $\tilde{E}(t, T, \delta t) = \tilde{E}(t, t_f - t_c, \delta t)$  does not depend on  $T$ , and in order to simplify notation, an auxiliary variable  $D$  is introduced defined by  $D(t, \delta t) := \tilde{E}(t, t_f - t_c, \delta t)$ . Furthermore, we define  $t'_f := \lfloor t_f \rfloor$ , and suppress the  $\delta t$  and  $T$  dependence in  $x_s$  and  $p_s$ . Evaluating the simulated costate at time  $t'_f - \delta t$ , setting  $p_s[t'_f] = 0$  since there is no final constraint, we get that  $p_s[t'_f - \delta t] = \delta t \frac{dL}{dx}(x_s[t'_f])^T$ . From this, it follows that

$$\begin{aligned} \|p_s[t'_f - \delta t] - p(t'_f - \delta t)\| &= \left\| \int_{t'_f - \delta t}^{t'_f} \left[ \frac{dL}{dx}(x_s[t'_f])^T - \frac{dL}{dx}(x(t))^T - \frac{df}{dx}(x(t), t)^T p(t) \right] dt \right\| \leq \\ &\leq L_{L'} \int_{t'_f - \delta t}^{t'_f} \|x_s[t'_f] - x(t)\| dt + C_{f'} \int_{t'_f - \delta t}^{t'_f} \|p(t)\| dt, \end{aligned} \quad (188)$$

where the Lipschitz continuity of  $\frac{dL}{dx}$  was used and the fact that  $\frac{df}{dx}$  is bounded. The integrand of the first term in the last part of (188) is bounded by  $E(t'_f, \delta t) + C_f \delta t$  where the  $C_f \delta t$  term is due to the fact that the integral ranges from  $t'_f - \delta t$  to  $t'_f$ . For the last term of (188), we note that  $p(t_f) = 0$  does not imply that  $p(t'_f) = 0$  since  $t_f \neq t'_f$  in general. Hence  $\|p(t)\| \leq 2C_{\bar{p}} \delta t$ ,  $\forall t \in [t'_f - \delta t, t'_f]$ . Hence, we define

$$D(t'_f - \delta t, \delta t) := L_{L'}[E(t'_f, \delta t) + C_f \delta t] \delta t + 2C_{f'} C_{\bar{p}} (\delta t)^2. \quad (189)$$

At time  $t'_f - 2\delta t$ , we get that,  $p_s[t'_f - 2\delta t] = p_s[t'_f - \delta t] - \delta t [-\frac{dL}{dx}^T(x_s[t'_f - \delta t]) - \frac{df}{dx}^T(x_s[t'_f - \delta t], t'_f - \delta t)p_s[t'_f - \delta t]]$  and  $p(t'_f - 2\delta t)$  is given by (99). It then follows that,

$$\begin{aligned} \|p_s[t'_f - 2\delta t] - p(t'_f - 2\delta t)\| &\leq \|p_s[t'_f - \delta t] - p(t'_f - \delta t)\| \\ &+ \int_{t'_f - 2\delta t}^{t'_f - \delta t} \left\| \frac{dL}{dx}^T(x_s[t'_f - \delta t], t'_f - \delta t) - \frac{dL}{dx}^T(x(t), t) + \frac{df}{dx}^T(x(t), t)p(t) \right\| dt, \end{aligned} \quad (190)$$

where the first term of the right hand side of (190) is bounded by  $D(t'_f - \delta t, \delta t)$  and the integral term is bounded by  $L_{L'}[E(t'_f - \delta t, \delta t) + C_h \delta t] + \int_{t'_f - 2\delta t}^{t'_f - \delta t} I dt$ , where  $I$  is given below,  $I = \frac{df}{dx}^T(x_s[t'_f - \delta t], t'_f - \delta t)p_s[t'_f - \delta t] - \frac{df}{dx}^T(x(t), t)p(t) = \frac{df}{dx}^T(x_s[t'_f - \delta t], t'_f - \delta t)[p_s[t'_f - \delta t] - p(t)] + p(t)[\frac{df}{dx}^T(x_s[t'_f - \delta t], t'_f - \delta t) - \frac{df}{dx}^T(x(t), t)]$ . If we assume that  $x_s \in X$ , then

$\frac{df}{dx}(x_s[t'_f - \delta t], t'_f - \delta t) \leq C_{f'}$  resulting in  $I \leq C_{f'}[D(t'_f - \delta t, \delta t) + C_{\dot{p}}\delta t] + C_p L_{h'}[E(t'_f - \delta t, \delta t) + C_f \delta t]$ . Putting everything together, we get that

$$\begin{aligned} D(t'_f - 2\delta t, \delta t) &:= D(t'_f - \delta t, \delta t)[1 + C_{f'}\delta t] + \\ &E(t'_f - \delta t, \delta t)[L_{L'}\delta t + C_p L_{h'}\delta t] + \\ &L_{L'}C_f(\delta t)^2 + C_f C_{\dot{p}}(\delta t)^2 + C_p C_{f'} C_f(\delta t)^2. \end{aligned}$$

By the same token, we have that

$$\begin{aligned} D(t'_f - 3\delta t, \delta t) &:= D(t'_f - 2\delta t, \delta t)[1 + C_{f'}\delta t] + E(t'_f - 2\delta t, \delta t)[L_{L'}\delta t + C_p L_{h'}\delta t] + \\ &L_{L'}C_f(\delta t)^2 + C_f C_{\dot{p}}(\delta t)^2 + C_p C_{f'} C_f(\delta t)^2, \end{aligned}$$

and in general, the least upper bound of the norm of the error in the costate is given recursively by the following equation,

$$\begin{aligned} D(t'_f - (j+1)\delta t, \delta t) &:= D(t'_f - j\delta t, \delta t)[1 + C_{f'}\delta t] + \\ &E(t'_f - j\delta t, \delta t)[L_{L'}\delta t + C_p L_{h'}\delta t] + \\ &+ L_{L'}C_f(\delta t)^2 + C_f C_{\dot{p}}(\delta t)^2 + C_p C_{f'} C_f(\delta t)^2, \end{aligned} \quad (191)$$

for any  $j \in \{1, \dots, \frac{t'_f}{\delta t} - \delta t\}$ . Defining,  $a := 1 + C_{f'}\delta t$ ,  $b := L_{L'}\delta t + C_p L_{h'}\delta t$ , and  $c := [L_{L'}C_f + C_f C_{\dot{p}} + C_p C_{f'} C_f](\delta t)^2$ , (191) can be expressed as,  $D(t'_f - (j+1)\delta t, \delta t) = aD(t'_f - j\delta t, \delta t) + bE(t'_f - j\delta t, \delta t) + c$ . Through recursion, we get that  $D(t'_f - j\delta t, \delta t) = a^{j-1}D(t'_f - \delta t, \delta t) + c \sum_{i=1}^{j-1} a^{i-1} + b \sum_{i=1}^{j-1} E(t'_f + (i-j)\delta t, \delta t)a^{i-1}$ , where an upper bound for  $E(t'_f + (i-j)\delta t, \delta t)$  was given in (108). For the sake of simplicity of our argument we use the following conservative bound  $E(t'_f + (i-j)\delta t, \delta t) \leq E(t'_f, \delta t) \leq (e^{L_f t'_f} - 1)C_f \delta t$ . Evaluating the sums, we get that,

$$\sum_{i=1}^{j-1} a^{i-1} = \frac{a^{j-1} - 1}{a - 1} = \frac{(1 + C_{f'}\delta t)^{j-1} - 1}{C_{f'}\delta t},$$

this together with the facts that  $D(t'_f - j\delta t, \delta t) \leq D(0, \delta t)$  and  $a^{j-1} \leq a^{t'_f/\delta t} \leq e^{C_{f'} t'_f}$ ,  $\forall j \in \{1, \dots, \frac{t'_f}{\delta t} - \delta t\}$ , finally results in the following conservative bound for  $\|p_s[t] - p(t)\|$ ,  $\forall t \in M(t_f, \delta t)$ ,

$$D(t'_f - j\delta t, \delta t) \leq e^{C_{f'} t'_f} D(t'_f - \delta t, \delta t) + c \frac{e^{C_{f'} t'_f} - 1}{C_{f'} \delta t} + b \left[ e^{L_f t'_f} - 1 \right] C_f \frac{e^{C_{f'} t'_f} - 1}{C_{f'}},$$

where we note that the first term of (192) is of order  $(\delta t)^2$ , since  $D(t'_f - \delta t, \delta t)$  is of order  $(\delta t)^2$ , while the last two terms is of order  $\delta t$ . By defining the constants  $S_1$  and  $S_2$  as in the proposition, the proof follows.  $\blacksquare$

It should be mentioned that the bounds on the error in the state and the costate trajectories presented above are well known results in numerical analysis, see for example [33] for results of this nature.

In the proof of Lemma 6.2.1, we will assume that  $\dot{x} = h(x, t)$  without loss of generality.

**Proof of Lemma 6.2.1:** Equation (153) holds since  $h$  does not depend on the initial state and  $\bar{x}(0) = x(\Delta)$ , hence  $x(t) = \bar{x}(t - \Delta)$  for all  $t \in (\Delta, T)$  regardless of  $T$ . As for (154), we note that the costate for the first system is given by

$$p(t) = \int_t^T \frac{dL}{dx}(x(s))\Phi(s, t)ds,$$

where  $\Phi(s, t)$  is the state transition matrix of the linear, time-varying dynamical system  $\dot{z} = \frac{\partial h(x, t)}{\partial x}z$ . For the second system, we have that  $\bar{p}(t) = \int_t^T \frac{dL}{d\bar{x}}(\bar{x}(s))\bar{\Phi}(s, t)ds$ , where  $\bar{\Phi}(s, t)$  is the state transition matrix of the linear, time-varying dynamical system  $\dot{z} = \frac{\partial h(\bar{x}, t)}{\partial \bar{x}}z$ . Furthermore

$$\bar{p}(t - \Delta) = \int_{t-\Delta}^T \frac{dL}{d\bar{x}}(\bar{x}(s))\bar{\Phi}(s, t - \Delta)ds = \int_t^{T+\Delta} \frac{dL}{d\bar{x}}(\bar{x}(s - \Delta))\bar{\Phi}(s - \Delta, t - \Delta)ds,$$

by a change of variables. Noting that  $\bar{x}(s - \Delta) = x(s)$  and  $\bar{\Phi}(s - \Delta, t - \Delta) = \Phi(s, t)$ , we get that  $\bar{p}(t - \Delta) = \int_t^{T+\Delta} \frac{dL}{dx}(x(s))\Phi(s, t)ds$ , where we define  $x(s) = x(T)$ ,  $\forall s \in [T, T + \Delta]$  since the system is assumed to have converged at time  $T$ . The following expression for the difference holds for all times  $t \in (\Delta, T)$ ,

$$\begin{aligned} \|p(t) - \bar{p}(t - \Delta)\| &= \left\| \int_t^T \frac{dL}{dx}(x(s))\Phi(s, t)ds - \int_t^{T+\Delta} \frac{dL}{dx}(x(s))\Phi(s, t)ds \right\| \leq \\ &\leq C \int_T^{T+\Delta} \|\Phi(s, t)\|ds. \end{aligned} \quad (192)$$

The following equation is in force for  $\Phi(s, t)$ ,  $\forall t \in [0, T]$ ,

$$\frac{d}{ds}\Phi(s, t) = \frac{dh}{dx}(x, s)\Phi(s, t) \quad (193)$$

$$\Phi(t, t) = I. \quad (194)$$



As  $\frac{dh}{dx}(x, s)$  is negative definite for  $t > T_1$ ,  $\lim_{s \rightarrow \infty} \Phi(s, t) = 0$  by (193) and (194). Since  $\Delta$  and  $t$  are finite it follows from (192) that  $\bar{p}(t - \Delta) = p(t)$ , as  $T \rightarrow \infty$ . ■

It should be noted that results similar to Lemma 6.2.1 have appeared in the literature ([22, 43]). Nevertheless it is important for our presentation and is therefore presented.

## REFERENCES

- [1] A.BEMPORAD, BORELLI, F., and MORARI, M., “The explicit solution of constrained lp-based receding horizon control,” *Proceedings of the 39th Conference on Decision and Control*, Sydney, Australia, December 2000.
- [2] A.BEMPORAD and MORARI, M., “Control of systems integrating logic, dynamics, and constraints,” *Automatica*, vol. 35, pp. 407–427, 1999.
- [3] ALAMIR, M. and ATTIA, S., “On solving optimal control problems for switched nonlinear systems by strong variations algorithms,” *6th IFAC symposium on Nonlinear Control Systems*, September 2004.
- [4] ARKIN, R., “Motor schema-based mobile robot navigation,” *The International Journal of Robotics Research*, vol. 8, pp. 92–112, 1989.
- [5] ARKIN, R., *Behavior-Based Robotics*. Cambridge, Massachusetts: MIT Press, 1998.
- [6] ARMIJO, L., “Minimization of functions having lipschitz continuous first-partial derivatives,” *Pacific Journal of Mathematics*, vol. 16, pp. 1–3, 1966.
- [7] ARZEN, K. and ANTON, C., “Control and embedded computing; survey of research directions,” *Proc 16th IFAC World Congress*, Prague, The Czech Republic, 2005.
- [8] ATTIA, S., ALAMIR, M., and DE WIT, C. C., “Sub optimal control of switched nonlinear systems under location and switching constraints,” *IFAC World Congress*, 2005.
- [9] AXELSSON, H., EGERSTEDT, M., and WARDI, Y., “A gradient-descent approach to optimal mode scheduling in hybrid dynamical systems,” *In preparation*.
- [10] AXELSSON, H., EGERSTEDT, M., and WARDI, Y., “Convergence of gradient-descent algorithms for mode-scheduling problems in hybrid systems,” *Submitted to Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 2006.
- [11] AXELSSON, H., EGERSTEDT, M., WARDI, Y., and VACHTSEVANOS, G., “Algorithm for switching-time optimization in hybrid dynamical systems,” *ISIC*, Cyprus,, July 2005.
- [12] AXELSSON, H., WARDI, Y., and EGERSTEDT, M., “Transition-time optimization for switched systems,” *IFAC World Congress*, Prague, The Czech Republic, July 2005.
- [13] BEMPORAD, A., GIUA, A., and SEATZU, C., “A master-slave algorithm for the optimal control of continuous-time switched affine systems,” *41th IEEE Conf. on Decision and Control*, pp. 1976–1981, 2002.
- [14] BEMPORAD, A., GIUA, A., and SEATZU, C., “An iterative algorithm for the optimal control of continuous-time switched linear systems,” *6th Int. Work. on Discrete Event Systems (WODES)*, Zaragoza, Spain, Oct. 2002.

- [15] BOCCADORO, M., EGERSTEDT, M., and WARDI, Y., "Optimal control of switching surfaces in hybrid dynamic systems," *IFAC Workshop on Discrete Event Systems*, Reims, France, Sept. 2004.
- [16] BOCCADORO, M., WARDI, Y., EGERSTEDT, M., and VERRIEST, E., "Optimal control of switching surfaces in hybrid dynamical systems," *Journal of Discrete Event Dynamic Systems*, vol. 15, pp. 433–448, Dec. 2005.
- [17] BOHN, C., KARKOSCH, H., MARIEFELD, P., and SVARICEK, F., *Automotive Applications of Rapid Prototyping for Active Vibration Control*. 1029-1039: IFAC Advances in Automotive Control Engineering Practice, 2004.
- [18] BRANICKY, M., BORKAR, V., and MITTER, S., "A unified framework for hybrid control: Model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, pp. 31–45, 1998.
- [19] BROCKETT, R., "Hybrid models for motion control systems," *In: Perspectives in Control*, H. Trentelman and J. C. Willems, Eds, Birkh, Boston, pp. 29–54, 1993.
- [20] BROCKETT, R., "Stabilization of motor networks," *IEEE Conference on Decision and Control*, pp. 1484–1488, 1995.
- [21] BROOKS, R., "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, p. 1423, March 1986.
- [22] BRYSON, A. and HO, Y., *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Pub. Corp.
- [23] CAINES, P. E. and SHAIKH, M. S., "Optimality zone algorithms for hybrid systems computation and control: From exponential to linear complexity," *Proceedings of the 2005 International Symposium on Intelligent Control/ 13th Mediterranean Conference on Control and Automation*.
- [24] CASSANDRAS, C. and LAFORTUNE, S., *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [25] DELMOTTE, F., EGERSTEDT, M., and VERRIEST, E., "Hybrid function approximation: A variational approach," *IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005.
- [26] DU, D. D.-Z. and PARDALOS, P. M., eds., *Minimax and Applications*. New York: Springer, 1995.
- [27] EGERSTEDT, M., "Behavior based robotics using hybrid automata," *Lecture Notes in Computer Science: Hybrid Systems III: Computation and Control*, pp. 103–116, Pittsburgh, PA, Springer-Verlag, March 2000.
- [28] EGERSTEDT, M. and HU, X., "A hybrid control approach to action coordination for mobile robots," *Automatica*, vol. 38, pp. 125–130, Jan. 2002.
- [29] EGERSTEDT, M. and WARDI, Y., "Multi-process control using queuing theory," *IEEE Conference on Decision and Control*, Dec. 2002.

- [30] EGERSTEDT, M., WARDI, Y., and AXELSSON, H., "Transition-time optimization for switched systems," *IEEE Transactions on Automatic Control*, vol. 51, pp. 110–115, Jan. 2006.
- [31] EGERSTEDT, M., WARDI, Y., and AXELSSON, H., "Optimal control of switching times in hybrid systems," *MMAR'2003*, Miedzyzdroje, Poland, Aug. 2003.
- [32] GAWRONSKI, W. K., *Advanced Structural Dynamics and Active Control of Structures*. New-York: Springer-Verlag, 2004.
- [33] GEAR, C., *Numerical Initial Value Problems in Ordinary Differential Equations*. Englewood Cliffs, N.J: Prentice-Hall, 1971.
- [34] GUIA, A., SEATZU, C., and DER MEE, C. V., "Optimal control of switched autonomous linear systems," *Proceedings of the 40th Conference on Decision and Control*, pp. 1816–1821, Phoenix, Arizona, December 1999.
- [35] GUILLERMO, O., *Game Theory*. London: Academic Press, 3rd ed., 1995.
- [36] HEATH, M. T., *Scientific Computing: An Introductory Survey*. McGraw-Hill, 1997.
- [37] HEDLUND, S. and RANTZER, A., "Optimal control of hybrid systems," *Proceedings of the 38th IEEE Conference on Decision and Control*, pp. 3972–3977, 1999.
- [38] HELLERSTEIN, J. L., DIAO, Y., PAREKH, S., and TILBURY, D. M., *Feedback Control of Computing Systems*. Wiley-IEEE Press, 2004.
- [39] HOPCROFT, J. and WILFONG, G., "Motion of objects in contact," *The International Journal of Robotics Research*, vol. 4, pp. 32–45, 1986.
- [40] HRISTU-VARSAKELIS, D., "Feedback control systems as users of shared network: Communication sequences that guarantee stability," *IEEE Conference on Decision and Control*, pp. 3631–3631, FL, 2001.
- [41] JOHANSSON, R. and RANTZER, A., *Nonlinear and Hybrid Systems in Automotive Control*. London: Springer-Verlag, 2003.
- [42] KORTENKAMP, D., BONASSO, R., and MURPHY, R., *Artificial Intelligence and Mobile Robots*. Cambridge, Massachusetts: MIT Press, 1998.
- [43] LEE, E. and MARKUS, L., *Foundations of Optimal Control Theory*. New York: Wiley, 1967.
- [44] LEE, J. and COOLEY, B., "Recent advances in model predictive control and other related areas," In *J.C. Kantor, C.E. Garcia, and B. Carnahan (Eds.), Fifth international conference on chemical process control, CACHE, AIChE*, pp. 201–216, 1997.
- [45] LINCOLN, B. and RANTZER, A., "Optimizing linear systems switching," *IEEE Conference on Decision and Control*, pp. 2063–2068, Orlando, FL, 2001.
- [46] LIU, J. and LEE, E., "Timed multitasking for real-time embedded software," *IEEE Control Systems Magazine*, vol. 22(6), Dec. 2002.

- [47] LUENBERGER, D., "Optimization by vector space methods," *John Wiley and Sons, Inc*, New York, 1969.
- [48] MAGNI, L. and SEPULCHRE, R., "Stability margin of nonlinear receding horizon control via inverse optimality," *System & Control Letters*, vol. 32, pp. 241–245, 1997.
- [49] MALER, O., "Control from computer science," *IFAC Symposium Nonlinear Control*, 2001.
- [50] MALMBORG, J. and EKER, J., "Hybrid control of a double tank system," *IEEE Conference on Control Application, Hartford, Connecticut*, 1997.
- [51] MAYNE, D., RAWLINGS, J., RAO, C., and SOKKAERT, P., "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, Jan. 2000.
- [52] MEHTA, T., DELMOTTE, F., and EGERSTEDT, M., "Motion alphabet augmentation based on past experience," *IEEE Conference on Decision and Control, Seville, Spain*, 2005.
- [53] MEHTA, T. and EGERSTEDT, M., "An optimal control approach to mode generation in hybrid systems," *Nonlinear Analysis: Theory, Methods and Applications*, 2006.
- [54] MOON, J., WARDI, Y., and KAMEN, E., "Optimal release times in single-stage manufacturing systems with finite production inventory," *IEEE Conference on Decision and Control*, pp. 2506–2511, December 2002.
- [55] OGREN, P. and LEONARD, N., "A provable convergent dynamic window approach to obstacle avoidance," *IFAC World Conference, Barcelona, Spain*, 2002.
- [56] POLAK, E., *Optimization Algorithms and Consistent Approximations*. New York, New York: Springer-Verlag, 1997.
- [57] POLAK, E. and WARDI, Y., "A study of minimizing sequences," *SIAM Journal on Control and Optimization*, vol. 22, pp. 599–609, 1984.
- [58] RAMADGE, P. and WONHAM, W., "The control of discrete event systems," *IEEE Transaction on Automatic Control*, vol. 77, pp. 81–98, Jan. 1989.
- [59] RANTZER, A. and JOHANSSON, M., "Piecewise linear quadratic optimal control," *Proceedings of the American Control Conference*, 1997.
- [60] REHBINDER, H. and SANFRIDSON, M., "Scheduling of a limited communication channel for optimal control," *IEEE Conference on Decision and Control, Sidney, Australia* Dec. 2000.
- [61] REIF, J. and WANG, H., "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, pp. 171–194, 1999.
- [62] RIEDINGER, P., ZANNE, C., and KRATZ, F., "Timeoptimal control of hybrid systems," *Proceedings of the American Control Conference, San Diego*, June 1999.
- [63] SANFRIDSON, M., "Quality of control and real-time scheduling," *PhD thesis. Department of Machine Design, Royal Institute of Technology (KTH), Sweden*, 2004.

- [64] SHAIKH, M. and CAINES, P., “On the optimal control of hybrid systems: Optimization of trajectories, switching times and location schedules,” *Proceedings of the 6th International Workshop on Hybrid Systems: Computation and Control*, Prague, The Czech Republic, 2003.
- [65] SUSSMANN, H., “A maximum principle for hybrid optimal control problems,” *Proc. 38th IEEE Int. Conf. Decision and Control*.
- [66] TOMLIN, C., LYGEROS, J., and SASTRY, S., “A game theoretic approach to controller design for hybrid systems,” *IEEE Transaction on Automatic Control*, vol. 88, pp. 949–970, July 2000.
- [67] VERRIEST, E., “Optimal control for switched distributed delay systems with refractory period,” *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1421–1426, 2006.
- [68] VERRIEST, E. and DELMOTTE, F., “Optimal control for switched point delay systems with refractory period,” *IFAC World Congress*, Prague, Czech Republic, July 2005.
- [69] WALSH, G., YE, H., and BUSHNELL, L., “Stability analysis of networked control systems,” *American Control Conference*, pp. 2876–2880, 1999.
- [70] WARDI, Y., EGERSTEDT, M., BOCCADORO, M., and VERRIEST, E., “Optimal control of switching surfaces,” in *43rd IEEE Conference on Decision and Control*, Atlantis, Bahamas, 2004.
- [71] XU, X. and ANTSAKLIS, P., “An approach to switched systems optimal control based on parameterization of the switching instants,” *Proceedings of the IFAC World Congress*, Barcelona, Spain, 2002.
- [72] XU, X. and ANTSAKLIS, P., “Optimal control of switched autonomous systems,” *IEEE Conference on Decision and Control*, Las Vegas, NV, Dec. 2002.

## VITA

Henrik Axelsson was born in Gislaved, Sweden, on February 10, 1979. He received a M.S. degree in Electrical Engineering from Chalmers University of Technology, Sweden, and from Georgia Institute of Technology in 2002 and 2003 respectively. In 2006, he graduated from the Georgia Institute of Technology with a Ph.D. in Electrical and Computer Engineering. His research focused on optimal control theory and its applications to mobile robotics.