Efficient Methods for Stochastic Optimal Control

Thesis by

Matanya B. Horowitz

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

California Institute of Technology
Pasadena, California

2014
(Defended May 14, 2014)

For the moon of my life

“*I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.*”
– Alan Turing, Computing Machinery and Intelligence (1950).

Acknowledgments

I've been asked by many of my friends if I'd recommend graduate school. It's a tough question. Let there be no doubt, graduate school has its difficult moments, and it certainly takes endurance. However, the knowledge I've gained, and the contrast between myself when I first began and now, is extraordinary. As I finish my last weeks here, I can only say that yes, for me it was worth it.

I am humbled by the generosity and support of my main advisor, Joel Burdick. Throughout my time here his enthusiasm and good nature have provided me rich ground to explore and amass a wide variety of disparate interests. I am forever indebted. I also wish to thank John Doyle for his incredible support and vision during my first several years. It is rare that students, so early in the endeavor that is academia, are treated and challenged to think so widely and deeply. Some day I hope to communicate such a passion for the future. I also owe my gratitude to Richard, for his guidance and for providing such a welcoming environment, even for students that are not his own. I am also in Venkat's debt, as his clarity in both instruction and discussion led me down entirely new paths and a direction of research that I will follow well after graduate school.

In my spare moments away from my laptop, I have been blessed to work with and befriend astounding colleagues. Nikolai, Eric, Ivan, and Enoch you will be truly missed. I can't imagine having survived without you all. To my colleagues Nick, Jeremy, Jason, Tom, and Rangoli, thank you for helping me learn what robotics truly is, and helping me to see the future.

Graduate school for me began long before I arrived at Caltech, and both my arrival and completion are due to two mentors who guided me well before I had a clue. Todd

Murphey I owe nothing less than my entire career up to this point, and I thank Scott Douglas for arranging our collaboration. Jason Marden I also thank for giving me that first glimpse into true rigor and passion for mathematics.

Finally, none of this would have been possible without my wonderful family. Their love and support makes every holiday a chance to review and plan for the future. Above all else, I owe my sanity and health to my beautiful wife. Kelsey, with you, anything is possible.

Contents

Acknowledgments **iv**

Abstract **xvi**

1 Optimal Control Theory **1**

1.1 Introduction 1

1.1.1 Thesis Contribution and Outline 4

1.2 Dynamic Programming 6

1.2.1 Deterministic Hamilton-Jacobi-Bellman 8

1.2.2 Stochastic Hamilton-Jacobi-Bellman 10

1.2.3 Feedback Control Law 11

1.2.4 Existing Hamilton Jacobi Bellman Algorithms 12

1.3 Approximate Approaches to the Optimal Control Problem 13

1.3.1 System Linearization 14

1.3.2 Model Predictive Control 15

1.3.3 Lyapunov Theory 17

1.3.4 Method of Moments 19

1.3.5 Navigation Functions 20

1.3.6 Sampling based planners 21

1.4 The Linear Hamilton Jacobi Bellman Equation 23

1.4.1 Path Integral Control 26

1.5 Markov Decision Processes 27

1.5.1 Linear MDPs 30

2 Optimization Based Numerical Methods **31**

2.1 Semidefinite Programming for Optimal Control 31

2.1.1 Sum of Squares Programming 32

2.1.2 Sum-of-Squares Relaxation of the linear HJB PDE 36

2.1.3 Analysis of SOS Relaxation 41

2.1.4 Examples 43

2.2 Domain Partitioning 47

2.2.1 Alternating Direction Method of Multipliers 48

2.2.2 Decomposition of Stochastic Optimal Control 50

2.2.3 Parallelization 53

2.2.4 Decomposition Issues 55

2.2.5 Decomposition Analysis 56

2.2.6 Scalar Example 59

2.2.7 Two Dimensional System 61

2.2.8 Discussion 61

2.3 Uncertainty Quantification of Partial Differential Systems 63

2.3.1 Sum of Squares-Based Solution Bounds 64

2.3.2 Moment-Based Bounds 66

2.4 Discussion 74

3 High Dimensional Optimal Control **75**

3.1 High Dimensional Tensor Background 76

3.2 Separated Representations of Tensors 77

3.3 Alternating Least Squares 79

3.4 Separated Solution to the HJB 80

3.4.1 Separation Rank of the HJB 81

3.4.2 Representation of Interior Boundary Conditions 82

3.5 Implementation Details and Examples 84

3.5.1 Inverted Pendulum 85

3.5.2 VTOL Aircraft 86

3.5.3 Quadcopter 88

3.6 Discussion 92

4 Navigation Functions as Optimal Controllers 93

4.1 Navigation Functions Constructed from Optimal Control 95

4.2 Approximate Time-Optimal Navigation Functions 99

4.2.1 Analogy with electro-statics 99

4.2.2 Convergence of the solution trajectories 100

4.3 Navigation Examples 100

4.3.1 Problems with Analytical Solutions 100

4.3.2 Effect of Noise on Corridor Navigation 101

4.3.3 Maze 102

4.3.4 Grasping 103

4.4 Discussion 105

5 Temporal Task Planning 107

5.1 Preliminaries 109

5.1.1 System Model 109

5.1.2 Specification Language 110

5.1.3 Problem Statement 111

5.2 Solution 112

5.2.1 Dynamic programming 113

5.2.2 Stochastic constrained reachability 114

5.3 Composition of Solutions 115

5.3.1 Automata-Guided Task 118

5.3.2 Dynamic Programming 118

5.3.3 Temporal Boundary Conservatism 119

5.3.4 Introduction of New Propositions 120

5.3.5 Complexity 121

5.4 Temporal Examples 122

5.4.1 Two Goal Temporal Example 125

5.4.2 $n-$Goal Temporal Example 125

5.5 Conclusions 126

6 Implications and Future Work **128**

6.1 Control Lyapunov Functions 128

6.2 Sum of Squares Programming in High Dimensions 130

6.3 Linear Markov Decision Processes in High Dimensions 132

6.4 General Hamilton Jacobi Bellman Equations 132

6.5 Software 132

6.6 Conclusion 133

6.6.1 Final Thoughts 135

Bibliography **136**

List of Figures

2.1 Illustration of potential misalignment between the optimal and approximate value function, $V(x)$ and $\tilde{V}(x)$, gradients despite their proximity. The x-axis here denotes state space domain, while the y-axis denotes the cost-to-go at a particular state. 42

2.2 Plots of approximate and exact desirability and cost-to-go solutions for the scalar system (2.11) versus state x in the interval $x \in [-1, 1]$. The dashed red, dashed blue, and solid black lines represent the $deg(\Psi) = 4$, $deg(\Psi) = 6$, and $deg(\Psi) = 8$ approximations respectively. The *Positivstellensatz* polynomial multipliers in (2.3) were set to have matching degree, i.e. $deg(s_i) = deg(\Psi)$. 44

2.3 Approximate Desirability and Value solutions for the two dimensional example (2.12). The problem is solved with $deg(\Psi) = deg(s_i) = 14$. The upper bound had gap $\gamma_{up} = 0.0979$, and lower bound $\gamma_{lw} = 0.1049$. Ten simulated trajectories of the closed loop system, randomly sampled from $x, y \in [-.75, .75]^2$, are shown in black. 45

2.4 Example of a particular grid domain decomposition, the checkerboard pattern, with the partitions grouped into white and black sets. As the sets of the same color require no consensus over their local variables, it is possible to perform the optimization over each set in parallel while maintaining the convergence properties of ADMM. 54

2.5 Approximations of a degree five polynomial $f(x, y) = 2x^5 + x^3(3y^2 - 5) + x(3 - 4y^2 + y^4)$ over four quadrants of the box $[-1, 1]^2$ without (left column) and with (right column) enforced continuity at the quadrant boundaries. Each row imposes a different degree bound d on the approximating polynomial and gives the approximation error. The degree of the Positivestellensatz multipliers are denoted as sos, and the optimization gap as gamma. 57

2.6 Evolution of the alternative value function over 10 alternating direction steps for the scalar system (2.31). Arrows show direction of evolution with each step of ADMM. 60

2.7 Values of the dual variables (top) and maximum approximation gap (bottom) with iteration number for the scalar system (2.31). 61

2.8 Results of multidimensional, nonlinear example system (2.34). On the left is the optimization after two iterations of ADMM, while the one on the right is the converged results, when using $deg(\Psi) = deg(s_i) = 8$, where s_i are the Positivstellensatz multiplers. At convergence the upper bound has distance $\gamma = 0.6321$. 62

2.9 Illustration of convolution of the unknown solution with Gaussian moments. 70

2.10 Example decomposition of planar problem domain with the resulting solution regions appearing explicitly as optimization variables. (a) On the left, the optimization problem (2.45) may be solved over the region ω with only the boundary data on Ω given. (b) On the right, the point solution ω or the interior boundaries b_{ij} appear explicitly as optimization variables and may also be bounded. 73

3.1 Illustrations of an inverted pendulum, a VTOL aircraft, and a quadcopter. 84

3.2 Desirability function for the inverted pendulum. 86

3.3 Cost to go for the inverted pendulum. The effects of the noise may be seen in the smoothing of the value function in comparison to the deterministic case seen in [134]. 87

3.4 Five principal separated components for the inverted pendulum desirability solution along the x_1, x_2 dimensions. 87

3.5 Sample trajectory when executing desirability for the VTOL aircraft. 88

3.6 Convergence and weighting for the VTOL solution. The red markers indicate at which iterations the ALS algorithm enriched the solution by adding a basis element. The weights correspond to the variables s_l in (3.1) 89

3.7 Complete basis function set for Quadcopter policy. 89

3.8 Convergence and weighting for the quadcopter solution. 91

3.9 Simulation of the closed loop quadcopter system. 91

4.1 Navigation function for a two dimensional point-mass robot calculated according to Eq. (4.5) with varying corridor widths. The goal location is the black square in the top right corner. 102

4.2 Navigation function for a two dimensional point-mass robot in a maze-like environment with equi-dimensional noise. On the top-left is the standard navigation function calculated according to (4.5). On the top-right, the minimum time-criteria is approximated by taking $\alpha = 100$, $\lambda = .04$ in (4.1). The bottom-left and -right have $\Sigma_t = 4I_2, 4.6I_2$, and $\lambda = .4, .46$ respectively. The obstacles and boundary are chosen to have penalty of 20 units, while the goal region has a penalty of 0 units. The goal is located at the center of the domain at $(x, y) = (5, 5)$ and is illustrated as a black rectangle. 104

4.3 An illustration of a nut grasping task. In (b) the square nut is shown in green while the gripper is shown in red. The blue region denotes a range of acceptable goal nut locations relative to the gripper. On the left in (a), cross sections of navigation function for the nut grasping task illustrated in Figure 4.3. Parameters used are $\alpha = 0.02, R = .02I_{3\times 3}, \Sigma_t = 5I_{3\times 3}$ and the boundary costs are set to $\phi = 1$. Spatial discretization in the $x-$ and $y-$coordinates are $h_x = h_y = 0.25$, and in the angular direction $h_\theta = 20°$. The goal region isosurface is displayed in dark blue. 105

5.1 Illustration of the construction of a solution primitive. On the left the solution is created for the boundary-only problem with the boundaries of the labeled regions set to zero, indicated by dashed lines. In the middle, the PDE is solved for one region activated while all other regions and boundary set to zero. In the last, a complete solution is shown with the solutions added. 118

5.2 Visualization of Algorithm 5. On the left, the solution for a given task is extracted along the boundary of the new region. In the middle, the solution is generated for the desired boundary conditions along the new region while all others are set to zero. In the last, the boundary is set to its desired value while all other boundaries are set to zero. 121

5.3 Results for last stage cost-to-go of Example 1. The domain is $x, y \in [0,1]^2 \backslash C$ for $C = \{x \in [0.2, 0.6], y \in [0.25, 0.4]\}$. The goals are $A = \{x \in [0.7, 0.85], y \in [0.7, 0.8]\}$, $B = \{x \in [0.7, 0.8], y \in [0.15, 0.25]\}$. A trajectory of the closed loop system begins at the grey square. 123

5.4 Individual desirability primitives for regions A, B on the left and right respectively. 123

5.5 Results of using composition method on the two region visit task of Example 1. 124

5.6 Exact and approximate cost-to-go before either region is visited on the left and right respectively. A trajectory, beginning from the grey square, is shown in black when following the induced policies. After visiting A the trajectories are continued in Fig. 5.3, 5.5. 124

5.7 Calculation times using exact and superposition method for visitation problem with n regions to visit. 126

List of Tables

1.1 Linear Desirability PDE for Various Stochastic Optimal Control Settings, from [63]. $L(\Psi) := f^T\left(\nabla_x \Psi\right) + \frac{1}{2} Tr\left(\left(\nabla_{xx}\Psi\right)\Sigma_t\right)$ 26

2.1 Solution quality γ of the desirability lower bound for the scalar system (2.11) with varying polynomial degree of solution Ψ and Positivestellensatz multipliers s_i of (2.3). 44

Abstract

The Hamilton Jacobi Bellman (HJB) equation is central to stochastic optimal control (SOC) theory, yielding the optimal solution to general problems specified by known dynamics and a specified cost functional. Given the assumption of quadratic cost on the control input, it is well known that the HJB reduces to a particular partial differential equation (PDE). While powerful, this reduction is not commonly used as the PDE is of second order, is nonlinear, and examples exist where the problem may not have a solution in a classical sense. Furthermore, each state of the system appears as another dimension of the PDE, giving rise to the *curse of dimensionality.* Since the number of degrees of freedom required to solve the optimal control problem grows exponentially with dimension, the problem becomes intractable for systems with all but modest dimension.

In the last decade researchers have found that under certain, fairly non-restrictive structural assumptions, the HJB may be transformed into a linear PDE, with an interesting analogue in the discretized domain of Markov Decision Processes (MDP). The work presented in this thesis uses the linearity of this particular form of the HJB PDE to push the computational boundaries of stochastic optimal control.

This is done by crafting together previously disjoint lines of research in computation. The first of these is the use of Sum of Squares (SOS) techniques for synthesis of control policies. A candidate polynomial with variable coefficients is proposed as the solution to the stochastic optimal control problem. An SOS relaxation is then taken to the partial differential constraints, leading to a hierarchy of semidefinite relaxations with improving sub-optimality gap. The resulting approximate solutions are shown to be guaranteed over- and under-approximations for the optimal value function. It is shown that these results extend to arbitrary parabolic and elliptic PDEs, yielding a novel method for Uncertainty Quantification (UQ) of systems governed by partial differential constraints. Domain decomposition techniques are also made available, allowing for such problems to be solved via parallelization and low-order polynomials.

The optimization-based SOS technique is then contrasted with the Separated Representation (SR) approach from the applied mathematics community. The technique allows for systems of equations to be solved through a low-rank decomposition that results in algorithms that scale linearly with dimensionality. Its application in stochastic optimal control allows for previously uncomputable problems to be solved quickly, scaling to such complex systems as the Quadcopter and VTOL aircraft. This technique may be combined with the SOS approach, yielding not only a numerical technique, but also an analytical one that allows for entirely new classes of systems to be studied and for stability properties to be guaranteed.

The analysis of the linear HJB is completed by the study of its implications in application. It is shown that the HJB and a popular technique in robotics, the use of navigation functions, sit on opposite ends of a spectrum of optimization problems, upon which tradeoffs may be made in problem complexity. Analytical solutions to the HJB in these settings are available in simplified domains, yielding guidance towards optimality for approximation schemes. Finally, the use of HJB equations in temporal multi-task planning problems is investigated. It is demonstrated that such problems are reducible to a sequence of SOC problems linked via boundary conditions. The linearity of the PDE allows us to pre-compute control policy primitives and then compose them, at essentially zero cost, to satisfy a complex temporal logic specification.

Chapter 1

Optimal Control Theory

1.1 Introduction

Developments in robotics and artificial intelligence suggest a new wave of innovation is breaking. Advancements in computational power and memory costs are paralleled by improved theory of optimization algorithms. In nearly every component of artificial intelligence, from control theory to computer vision, our understanding of increasingly autonomous tasks has grown significantly. It appears increasingly likely that the next two decades will finally see the diffusion of robotic and artificially intelligent technology across all spheres of life first envisioned in the 20th century and earlier by such luminaries as Ada Lovelace, Isaac Asimov, and many others.

Contrary to popular opinion, these advancements are only secondarily related to the increase in computational power. An illustrative example lies in the field of linear programming [1]. Since 1988, linear program solvers have had their computational time fall by a factor of roughly 43 million. While Moore's law may be credited for roughly a factor of 1,000x, the remaining factor of 43,000x due to improved algorithms. Similar patterns are present in nearly every component of artificial intelligence. The algorithms of the past decade are impractical for solving problems routinely considered now, regardless of the computational resources available. The implications of the algorithmic advancements in the last decade are therefore all the more acute.

A crucial component of autonomous systems is their ability to comprehend and interact with signals from the physical world. This gives rise to the field of control theory, which seeks to drive systems towards achieving specified goals. The domain of problems ranges from the development of *feedback laws* that control the system directly from low-level signals, all the way to the development of abstract plans to solve a high-level task, overlapping with areas of conventional Artificial Intelligence.

The model in such problems is the concept of a system state, capturing all mutable and influential components of the system, represented at a particular point in time t by a variable $x(t)$. The evolution of the system's state is in turn governed by a set of constraints, called the system *dynamics*. When such a system can be said to be influenced by some control signal u, the task in control theory is to design this control signal to generate desirable behavior relative to a goal. The design of this signal is labeled the control law or policy, depending on the context. Feedback, which allows for the control signal to depend on the current state of the system, $u \triangleq u(x(t))$, is the most powerful tool available in this endeavor. Control theory primarily answers two questions: how to design the control law, and how to analyze its properties.

Efficiency and robustnesss are frequently significant factors in control law design. These criteria are incorporated by assigning a *cost* to the system state and control signal. Common is the desire to penalize excessive use of energy, as well as a weighting on certain system states, allowing for time away from some desired goal state to be minimized. The framework is general, allowing for most any concrete goal that involves the state of the system to be specified.

At the core of solving these problems is a theoretical object named the *value function*. This quantity, if known, represents the choice of action by the system that will *optimally* solve the problem at hand. Not only is the construction of this object possible, but a general equation is known that specifies the value function for the majority of systems of interest. This formula, called the *Hamilton Jacobi Bellman* equation, is quite general, and is the fundamental answer to many problems in control theory and planning.

Unfortunately, although there exists an explicit equation, solving the Hamilton Jacobi Bellman equation is difficult primarily for two reasons. The first is that it may be intractable due to discontinuities in the solution. This has led to a relaxed notion of a solution, dubbed *viscosity solutions*, and is an active object of study to this day. However, the second is far greater an obstacle, the *curse of dimensionality.* This obstacle, discovered by the creator of the value function, Richard Bellman in 1957, refers to the fact that the number of degrees of freedom of the value function increases exponentially with the dimensionality of the system if there is no assumption on structure. The result is that solving for systems with even a moderate number of degrees of freedom quickly becomes too computationally and memory intensive for all but the roughest of approximations. This has stymied the study of Hamilton Jacobi Bellman equations, leading researchers towards other approaches that either approximate or neglect optimality, or to assume certain structure in the system.

Control theory is itself a mature field, with many critical questions not only answered, but definitively solved provably optimal solutions. *Linear* systems, are an example of systems whose control has largely been solved, with many tractable algorithms for even large and sophisticated systems. A remaining area that is yet open is the development of optimal solutions to nonlinear systems with tractable algorithms. This thesis presents advancements in two key areas. The first is the development of algorithms that grapple with nonlinearities directly, incorporating the natural dynamics of the system rather than trying to remove them. The second is the development of algorithms that scale economically with the complexity, measured by the number of degrees of freedom, of the system. This advances the state of the art in the motion planning of complex systems, allowing for new classes of problems to be solved in a scalable manner. Indeed, such advancements are necessary, as traditional numerical methods that don't incorporate the underlying structure become impractical for solving problems with high dimensional systems no matter the computational resources available.

Foremost in this thesis is the study of the Hamilton Jacobi Bellman equation. By advancing the theory involved in solving this equation, this general optimal control approach comes closer to wide applicability and practicality. In particular, it is the generality, rather than the optimality, that is of key importance. Generality corresponds directly to the ability to automate, allowing for difficult planning problems that would previously require the design experience of multiple skilled practitioners to be exchanged for computation.

This thesis advances the theory of value functions in several ways, but two are primary. First, an optimization technique called Sum of Squares is adapted to solve the problem. These value functions have a number of advantages, including a parameterization of the solution that avoids fine discretization; theoretical guarantees; and an ability to relax solution quality to obtain less computationally intensive procedures. The second is the use of a technique to solve high dimensional problems, allowing for the curse of dimensionality to be mitigated for problems of interest. This opens the door for a new class of problems to be solved using optimal control theory. The method appears quite attractive, solving previously impossible problems quickly in practice. These two techniques, the Sum of Squares and high dimensional frameworks, are then fused. Various augmentations are then proposed, and a number of more specific applications are studied.

1.1.1 Thesis Contribution and Outline

This thesis is concerned with calculating solutions to the linear Hamilton Jacobi Bellman equation, and the applications thereof. The contributions of the following chapters are based on a number of publications [2, 3, 4, 5, 6], indicated below.

In Chapter 1 Dynamic Programming is reviewed for stochastic optimal control, leading to the derivation of the Hamilton Jacobi Bellman equation. A review is then given of the significant and popular alternative techniques that avoid solving this partial differential equation directly.

The contributions of this thesis begin in Chapter 2. First presented in [2], a method to calculate improving sub- and super-solutions of the Hamilton Jacobi Bellman PDE via sum-of-squares relaxations is proposed. Polynomial candidate solutions are optimized, and guarantees on the quality of the approximate solutions are provided. This technique is augmented via domain partitioning in Section 2.2, based on the work of [3], allowing for a collection of low-order polynomials to capture local phenomena without excessive growth in the order of the candidate solutions. Finally, in Section 2.3, the power of the method is applied to general Elliptic and Parabolic PDEs, allowing for uncertainty quantification over solutions even when the coefficients may be unknown.

In Chapter 3 an alternative method to solve the Hamilton Jacobi Bellman equation is proposed. Based on [4], a method to obtain an approximate low rank decomposition of the Hamilton Jacobi Bellman operator is developed, and used to calculate low rank approximations to the optimal solution. These techniques demonstrate linear growth with dimension, allowing for high dimensional problems to be solved.

The work proceeds by examining several application areas of optimal control theory, connecting the linear Hamilton Jacobi Bellman equation to the broader literature. First, the Hamilton Jacobi Bellman equation is connected to the literature on Navigation Functions for robotics in Chapter 4, arising from the work in [5]. It is demonstrated that the Navigation Function may be seen as a particular form of the Hamilton Jacobi Bellman equation, and existing PDE techniques in the literature may be directly derived from optimal control.

This is followed by the study of temporal task planning problems in Chapter 5, built upon the results of [6]. It is shown that temporal problems encoded in Linear Temporal Logic consist of distinct optimal control problems connected via boundary conditions. The linearity of the Hamilton Jacobi Bellman equation is leveraged to rapidly compute solutions in these domains.

Finally, Chapter 6 rounds out the work presented in this thesis. A brief review of the topics is given, along with a discussion of how the computability of the Hamilton Jacobi Bellman equation affects those applications developed here, as well as those not touched upon. The thesis ends with a discussion of potential developments in the study of Hamilton Jacobi Bellman solutions.

1.2 Dynamic Programming

The Hamilton Jacobi Bellman equation is fundamental object to the study of stochastic optimal control theory, arising from Bellman's *Principle of Optimality* [7]. During the execution of a controlled system, at any point in time the system trajectory may be split in two. The optimal future trajectory is independent of the past, as the system's state captures the full history of what occurred, and thus each trajectory segment is itself optimal.

Let the discrete time system trajectory[1] be denoted by $x_0, \ldots, x_N \in \mathbb{R}^n$, where x_k is the state at time t_k, the control inputs are similarly denoted $u_0, \ldots, u_{N-1} \in \mathbb{R}^m$, as are perturbations to the system $w_0, \ldots, w_{N-1}$. The evolution of the trajectory can be captured in the state transition function $f_k : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}^n$, with $x_{k+1} = f_k(x_k, u_k, w_k)$. At each discrete time, the state is penalized with a cost $\ell_k : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ is assigned, which one seeks to minimize. The control input, u_k, may further be restricted to lie in some set $u_k \in U_k(x_k)$. The basic stochastic optimal control problem with horizon length N is therefore

$$\begin{array}{ll} \text{minimize} & \mathbb{E}\left[\sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k) + \ell_N(x_N)\right] \\ \text{subject to} & x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, \ldots, N-1 \\ & u_k \in U_k(x_k), \quad k = 0, \ldots, N-1, \end{array} \tag{1.1}$$

where ℓ_N is a terminal cost, the decision variables are x_k, u_k, and the expectation is over (random) disturbances $w_0, \ldots, w_{N-1} \in \mathbb{R}^r$. For any feasible trajectory, the cost is expressed as the function

$$J(x, u) \triangleq \mathbb{E}\left[\sum_{k=0}^{N-1} \ell_k(x_k, u_k, w_k) + \ell_N(x_N)\right] \tag{1.2}$$

Supposing that the minimizing policy was in fact known, the optimal solution is

[1]Thanks to Ivan Papusha for a preliminary version of these notes

defined as the *value function*, also known as the *cost to go* $V(x)$,

$$V^*(x) \triangleq \min_u J(x,u) = J(x, u^*(x,t)), \tag{1.3}$$

where $u^*(x,t)$ is the (as yet unknown) optimal control strategy. Clearly, the optimal solution at future points in time does not depend on actions in the past. Therefore, the optimal choice at any point in time is to choose the action that brings the system into the accessible state with the lowest future cost to go. This result is known as Bellman's principle of optimality.

Theorem 1 (Bellman's Principle of Optimality [7])**.** *If $u^*(x,\tau)$ is optimal over the interval $[t, t_N]$ starting at state $x(t)$ then $u^*(x,\tau)$ is necessarily optimal over the subinterval $[t, t+\Delta t]$ for any Δt such that $T - t \geq \Delta t \geq 0$.*

For every initial state x_0, the optimal cost $V^*(x_0)$ of the basic problem is given by $V_0(x_0)$ in the last step of the following algorithm [8, §1.3], which proceeds backward from period $N-1$ to period 0:

$$\begin{aligned} V_N(x_N) &= \ell_N(x_N), \\ V_k(x_k) &= \min_{u_k \in U_k(x_k)} \mathbb{E}_{w_k}\left[\ell_k(x_k, u_k, w_k) + V_{k+1}\big(f_k(x_k, u_k, w_k)\big)\right], \quad k = 0, \ldots, N-1. \end{aligned}$$

The optimal policy consists of choosing a minimizing control action u_k^*,

$$u_k^* \in \operatorname*{argmin}_{u_k \in U_k(x_k)} \mathbb{E}_{w_k}\left[\ell_k(x_k, u_k, w_k) + V_{k+1}\big(f_k(x_k, u_k, w_k)\big)\right], \quad k = 0, \ldots, N-1.$$

Note the generality of the approach. The next section studies more specific variants of the Hamilton Jacobi Bellman equation for deterministic and stochastic systems with affine disturbances.

1.2.1 Deterministic Hamilton-Jacobi-Bellman

The basic continuous-time control problem with horizon length T is

$$\begin{array}{ll} \text{minimize} & \int_0^T \ell(x(t), u(t))\, dt + \ell_T(x(T)) \\ \text{subject to} & \dot{x}(t) = f(x(t), u(t)), \quad 0 \le t \le T, \\ & x(0) = x_0 \end{array}$$

If $V(t, x)$ a continuously differentiable (in t and x) solution to the Hamilton Jacobi Bellman equation, [9],

$$-\frac{\partial}{\partial t} V(t, x) = \min_{u \in U} \left[\ell(x, u) + \nabla_x V(t, x)^T f(x, u) \right], \tag{1.4}$$

$$V(T, x) = \ell_T(x), \tag{1.5}$$

then it is the optimal cost-to-go function and a control policy obtained using the minimization is optimal. The function $V : [0, T] \times \mathbb{R}^n \to \mathbb{R}$ is the *value function* for this problem type. The derivation is given in the following section.

Derivation Using Dynamic Programming

The following derivation follows that of [8, §3.2]. Divide the time horizon $[0, T]$ into N pieces using the discretization interval $\delta = \frac{T}{N}$, and define

$$x_k \triangleq x(k\delta), \quad u_k \triangleq u(k\delta), \quad k = 0, \dots, N.$$

The first order approximation to the continuous system and its cost function are

$$x_{k+1} = x_k + f(x_k, u_k) \cdot \delta$$

$$J = \sum_{k=0}^{N-1} \ell(x_k, u_k) \cdot \delta + h(x_N).$$

Let $J^*(t, x)$ be the optimal cost to go at time t and state x for the continuous-time problem, and $J_d^*(t, x)$ be the optimal cost-to-go for the discrete-time approximation.

The DP equations are

$$\begin{aligned} J_d^*(N\delta, x) &= h(x), \\ J_d^*(k\delta, x) &= \min_{u\in U}\left[\ell(x,u)\cdot\delta + J_d^*\big((k+1)\cdot\delta, x + f(x,u)\cdot\delta\big)\right], \quad k = 0,\ldots,N-1. \end{aligned}$$

Expanding J_d^* as a Taylor series around $(k\delta, x)$ one obtains

$$J_d^*\big((k+1)\cdot\delta, x+f(x,u)\cdot\delta\big) = J_d^*(k\delta,x)+\nabla_t J_d^*(k\delta,x)\cdot\delta+\nabla_x J_d^*(k\delta,x)^T f(x,u)\cdot\delta+o(\delta),$$

where $\lim_{\delta\to 0} o(\delta)/\delta = 0$. After substituting back into the DP equations,

$$J_d^*(k\delta,x) = \min_{u\in U}\left[\ell(x,u)\cdot\delta + J_d^*(k\delta,x) + \nabla_t J_d^*(k\delta,x)\cdot\delta + \nabla_x J_d^*(k\delta,x)^T f(x,u)\cdot\delta + o(\delta)\right].$$

Cancelling $J_d^*(k\delta,x)$ from both sides, divide by δ, and take the limit as $\delta \to 0$. Assuming the discrete-time cost-to-go function yields in the limit its continuous-time counterpart, *i.e.*,

$$\lim_{k\to\infty,\,\delta\to 0,\,k\delta=t} J_d^*(k\delta,x) = J^*(t,x), \quad \text{for all } t, x,$$

this derivation yields the Hamilton-Jacobi-Bellman equation for the optimal cost-to-go $J^*(t,x)$,

$$\begin{aligned} 0 &= \min_{u\in U}\left[g(x,u) + \nabla_t J^*(t,x) + \nabla_x J^*(t,x)^T f(x,u)\right], \quad \text{for all } t, x, \\ h(x) &= J^*(T,x), \quad \text{for all } x. \end{aligned}$$

Letting $V(t,x) \triangleq J^*(t,x)$, and removing the terms that do not depend on u out of the minimum reveals (1.4).

1.2.2 Stochastic Hamilton-Jacobi-Bellman

The basic stochastic control problem with horizon length T is

$$\begin{aligned}
\text{minimize} \quad & \mathbb{E}\left[\int_0^T \ell(x_t, u_t)\,dt + h(x_T)\right] \\
\text{subject to} \quad & dx_t = f(x_t, u_t)\,dt + \sigma(x_t)\,d\omega_t, \quad 0 \le t \le T, \\
& x|_{t=0} = x_0.
\end{aligned}$$

The dynamics of the state x_t are governed by an Itō drift-diffusion process in $\mathbb{R}^n$, where $\{\omega_t \mid t \ge 0\}$ is a standard Wiener process in $\mathbb{R}^q$ and $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times q}$ is a noise feedthrough function. The stochastic Hamilton Jacobi Bellman equation is

$$\begin{aligned}
-\frac{\partial}{\partial t}V(t,x) &= \min_{u\in U}\left[\ell(x,u) + \nabla_x V(t,x)^T f(x,u) + \frac{1}{2}\,\mathbf{Tr}\left(\nabla_x^2 V(t,x)\cdot\sigma(x)\sigma(x)^T\right)\right], \\
V(T,x) &= h(x).
\end{aligned}$$

The extra Hessian term that differs from the deterministic case arises from Itō's formula. The definitive sources are [10] with a derivation given in the following section.

Derivation Using Dynamic Programming

The first order approximation to the continuous system with stochastic perturbations is similar, but according to the rules of Itō includes a stochastic forcing term

$$x_{k+1} = x_k + f(x_k, u_k)\cdot\delta + \sigma(x_k)\cdot\epsilon_k\cdot\delta^{1/2},$$

where $\epsilon_k \sim \mathcal{N}(0, I_q)$ are iid standard normal variables on $\mathbb{R}^q$ inherited from the Wiener process. The DP equations are

$$\begin{aligned}
J_d^*(N\delta, x) &= h(x), \\
J_d^*(k\delta, x) &= \min_{u \in U} \mathbb{E}\left[\ell(x,u)\cdot\delta + J_d^*\big((k+1)\cdot\delta, x + f(x,u)\cdot\delta + \sigma(x)\cdot\epsilon\cdot\delta^{1/2}\big)\right] \\
&= \min_{u \in U} \left[\ell(x,u)\cdot\delta + \mathbb{E}J_d^*\big((k+1)\cdot\delta, x + f(x,u)\cdot\delta + \sigma(x)\cdot\epsilon\cdot\delta^{1/2}\big)\right]
\end{aligned}$$

for $k = 0, \ldots, N-1$. Expand J_d^* as a Taylor series around $(k\delta, x)$ to the second order:

$$\begin{aligned}
&J_d^*\big((k+1)\cdot\delta, x + f(x,u)\cdot\delta + \sigma(x)\cdot\epsilon\cdot\delta^{1/2}\big) = \\
&\qquad J_d^*(k\delta, x) + \nabla_t J_d^*(k\delta, x)\delta + \nabla_x J_d^*(k\delta, x)^T \left(f(x,u)\delta + \sigma(x)\epsilon\delta^{1/2}\right) \\
&\qquad\qquad + \frac{1}{2} Tr\left(\nabla_x^2 J_d^*(k\delta, x)\cdot\sigma(x)\epsilon\epsilon^T\sigma(x)^T\delta\right) + o(\delta^{3/2})
\end{aligned}$$

Using $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}\left[\epsilon\epsilon^T\right] = I_q$, take the expected value of both sides to obtain

$$\begin{aligned}
&\mathbb{E}\left[J_d^*\big((k+1)\cdot\delta, x + f(x,u)\cdot\delta + \sigma(x)\cdot\epsilon\cdot\delta^{1/2}\big)\right] = J_d^*(k\delta, x) + \nabla_t J_d^*(k\delta, x)\delta \\
&\qquad + \nabla_x J_d^*(k\delta, x)^T f(x,u)\delta + \frac{1}{2} Tr\left(\nabla_x^2 J_d^*(k\delta, x)\cdot\sigma(x)\sigma(x)^T\delta\right) + o(\delta^{3/2}).
\end{aligned}$$

Finally, substitute this expression back into the DP equations, subtract $J_d^*(k\delta, x)$ from both sides, divide by δ, and take the limit as $\delta \to 0$ to obtain the stochastic Hamilton Jacobi Bellman equations, [10].

$$\begin{aligned}
-\nabla_t J^*(t,x) &= \min_{u \in U}\left[\ell(x,u) + \nabla_x J^*(t,x)^T f(x,u) + \frac{1}{2} Tr\left(\nabla_x^2 J^*(t,x)\cdot\sigma(x)\sigma(x)^T\right)\right], \\
h(x) &= J^*(T,x).
\end{aligned}$$

1.2.3 Feedback Control Law

The Hamilton Jacobi Bellman solution has up to this point been developed as a method to generate optimal trajectories. However, as a consequence of Eq. (1.23), the method also provides an optimal feedback controller. The result is an architecture that is both robust and far-sighted, with the feedback controller and planner optimal and integrated.

In the context of stabilization, the feedback design problem may be framed as the problem of simply achieving a goal expressed as an equilibrium point. In tasks with more complex and arbitrary goals, the feedback arises from the application of the Hamilton Jacobi Bellman solution from any point in the state space, and further from the inclusion of stochasticity within the planning process. In contrast to other techniques, the control design based Hamilton Jacobi Bellman equation does not involve an approximation, allowing for the effects of stochasticity to be used when beneficial, and for various control design criteria to be incorporated.

The controllers resulting from this design principle have several appealing properties. In contrast to Model Predictive Control-based schemes (further explored in 1.3.2), no online computation is required. The policy produced is a mapping from system state to control input, and once produced can often be implementedinvolves nothing more than as a table lookup.

The use of the optimal policy in this context parallels another tool of control theory, that of *gain scheduling* [11]. These approaches develop a feedback controller by partitioning the state space of the nonlinear problem. A linear approximation to the dynamics is then made over each individual partition, and the resulting controller is used when the state of the system occupies that partition. Key to the gain scheduling control design approach is that once the local controllers have been developed, their properties may be investigated through Lyapunov theory. The resulting controller is also fixed, and has no online computation. The controller may then be exhaustively simulated, a necessity in aerospace applications.

1.2.4 Existing Hamilton Jacobi Bellman Algorithms

While a great deal of research has gone into techniques that approximate properties of the Hamilton Jacobi Bellman solution, considerable effort has been devoted to numerical methods of solving the Hamilton Jacobi Bellman equation.

Level-Set Methods The work of [12] demonstrates the use of Hamilton Jacobi Bellman solutions for solving reachability-type problems, as well as the difficulties in the general use of this approach for control theory. In particular, reachability analysis proves to be key in the development of control algorithms for hybrid systems. The authors of [12] subsequently focused on the development of level-set techniques for computing Hamilton Jacob solutions [13, 14]. In this framework, the reachable set is described implicitly as the zero level set to a variable quantity. The level set boundary layer is then evolved according to the Hamilton Jacobi Bellman equation. The work has also been extended to dynamic games [15].

Albrecht Method (Local Taylor Expansion) In [16], a method to solve the Hamilton Jacobi Bellman equations was proposed based on a Taylor series expansion of the solution. It is shown that if a solution to the first order or higher terms is obtained, this will stabilize the system to some origin. By increasing the number of terms used, the performance may be further improved and the dynamics of the system more fully captured [17]. However, the basis of attraction for such a suboptimal controller is unknown a priori. In [18, 19] the method was augmented by incorporating many such local solutions, typically called "patches", where the boundaries are determined by studying the invariant manifolds of the closed loop dynamics. Recent results using high order approximations are given in [20], and a discrete time variant of this approach in [21].

Max-Plus McEaney [22, 23] has developed a novel framework with which to analyze the Hamilton Jacobi Bellman equation. The insight of the method is that while the Hamilton Jacobi Bellman equation is quadratic in the gradient, it is linear in a max-plus algebra. This insight is used to solve a number of control and filtering problems.

The method has also given rise to the only other curse-of-dimensionality free method beyond that presented in this thesis that this author is aware of, specifically the method of [24]. This technique has complexity that scales with the number of basis solutions, each requiring the solution to a Riccati system. Of note is the fact that this alternative also has cubic growth with respect to state space dimension.

1.3 Approximate Approaches to the Optimal Control Problem

Historically, the difficulty of solving the Hamilton Jacobi Bellman equation directly has prevented its use in practice. Indeed, for all but the recently developed Max-Plus approach, none of the techniques detailed in the previous section can readily scale beyond modest dimensions (namely, five or so continuous system states). This is due to the curse of dimensionality, wherein the number of optimization variables scales exponentially with the number of system dimensions. This difficulty had led researchers to consider alternative methods. While optimality may be lost for nonlinear systems, these methods have come to define the modern control landscape.

1.3.1 System Linearization

The overwhelming success of linear control theory has led the control community to embrace linearization techniques. Given a system of the form

$$\dot{x} = f(x, u) \tag{1.6}$$

$$y = h(x, u) \tag{1.7}$$

The dynamics matrices may be obtained by taking the Taylor expansion about an equilibrium points $x = x_0$ and retaining the first order term

$$\delta\dot{x} = \nabla_x f(x_0, u_0)\delta x + \nabla_u f(x_0, u_0)\delta u \tag{1.8}$$

$$\delta y = \nabla_x h(x_0, u_0)\delta x + \nabla_u h(x_0, u_0)\delta u \tag{1.9}$$

Replacing the linearization of the dynamics with matrices $A \triangleq \nabla_x f(x_0, u_0)$, $B \triangleq \nabla_u f(x_0, u_0)$, and $C = \nabla_x h(x_0, u_0)$ and $D = \nabla_u h(x_0, u_0)$. The tools of linear control theory, such as the Linear Quadratic Regulator [25], can then be applied to design a controller, and stabilization will be retained on the true system for some neighborhood of the origin. The basin of attraction created may be verified through a number of tools of nonlinear theory, chiefly the construction of a Lyapunov function [26].

The above process may be repeated over a partition of the state space of the system, obtaining a collection of linearizations, each valid in the neighborhood of some linearization point. The result gives rise to gain scheduling, already mentioned in Section 1.2.3. While in the continuous limit such approximations approach the underlying dynamics of the nonlinear system, note that the number of partitions necessary to achieve a certain fidelity grows exponentially with state space size.

1.3.2 Model Predictive Control

The Hamilton Jacobi Bellman equation solves the control problem from *every system initial condition*. However, many applications only require a solution beginning from a *particular* initial condition. Indeed, in many problem instances, it is exceedingly unlikely that the system will visit vast regions of the state space domain. This has given rise to a group of techniques under the names of Model Predictive Control, Receding Horizon Control, and is related to Differential Dynamic Programming.

The deterministic version of this problem is

$$
\begin{aligned}
min._{u(\cdot)} \quad & \int_0^T \left(q(x) + u^T u\right) dt + \varphi\left(x(T)\right) && (1.10)\\
s.t. \quad & \dot{x} = f(x) + g(x)u && (1.11)\\
& x(0) = x_0 && (1.12)
\end{aligned}
$$

A Lagrange multiplier is introduced to enforce the equality constraints, creating the new optimization problem

$$
\min_{u(\cdot)} \int_0^T H(x(t), u(t), \lambda(t)) dt \tag{1.13}
$$

where the scalar *Hamiltonian* function $H(x(t), u(t), \lambda(t))$ is defined as

$$
H(x(t), u(t), \lambda(t)) = q(x(t)) + u^T(t)u(t) + \lambda^T(t)\left[f(x(t)) + g(x(t))u(t)\right] \tag{1.14}
$$

Integration by parts of (1.13) yields

$$\varphi\left(x\left(T\right)\right)-\lambda^{T}(T)x(T)+\lambda^{T}(0)x(0)+\int_{0}^{T}\left(H\left(x\left(t\right),u\left(t\right),\lambda\left(t\right)\right)+\dot{\lambda}\left(t\right)x\left(t\right)\right)dt$$

resulting in an equation that is independent of constraints. The problem is then to determine a stationary point of this cost equation, where the optimization variables now include $\lambda(t)$. Examining the differential with respect to the system variables $x(t)$ and $u(t)$ it is apparent that a necessary condition for optimality is that the expression

$$\left[\left(\frac{\partial\varphi}{\partial x}-\lambda^{T}\right)\delta x\right]_{t=T}+\left[\lambda^{T}\delta x\right]_{t=0}+\int_{0}^{T}\left[\left(\frac{\partial H}{\partial x}+\dot{\lambda}^{T}\right)\delta x+\frac{\partial H}{\partial u}\delta u\right]dt$$

must be equal to zero along the trajectory. The result is the set of equations

$$\begin{aligned}\dot{x} &= f(x)+g(x)u\\ \dot{\lambda} &= -\left(\frac{\partial H}{\partial x}\right)^{T}\\ \frac{\partial H}{\partial u} &= 0\end{aligned}$$

which now correspond to a two point boundary value problem, with boundary values $x(0)$ given, and

$$\lambda^{T}(T)=\left.\frac{\partial\varphi}{\partial x}\right|_{t=T}$$

The evolution of the variable $\lambda(t)$ is called the *co-state* or *adjoint* equation, and may be integrated backwards in time without knowledge of the control input. The control input $u(t)$ is then obtained by integration forwards in time.

The approach has shown notable success, as its solution is obtained via an ordinary, rather than partial, differential equation, and may therefore be calculated quickly. However, the resulting control law is not typically state dependent, and is instead executed open loop over some finite horizon. Furthermore, only necessary conditions have been used in the derivation, giving rise to trajectories that are in fact only locally optimal and depend on the choice of initial trajectory used.

The above considerations have given rise to a number of approaches. Primarily, state dependence has been re-introduced through receding horizon schemes. Therein, some finite horizon is planned over, as before, but only some small segment of the trajectory is executed. Periodically, the controller re-plans, based on its most recent state, which has likely drifted from the planned trajectory due to unmodeled dynamics or stochastic forcing. Such approaches are popular [27], but have few performance guarantees on their own and may even be unstable [28]. As will be seen subsequently in Section 6.1, these techniques may be augmented with particular terminal costs to increase their robustness.

This analysis has served as a foundation for a number of powerful algorithms [29, 30, 31]. There exist a number of approaches that vary in their parameterization of time, either through discretization or some spline-basis. A degree of algorithmic freedom also exists in the order of the dynamics that needs to is captured, trading model accuracy for computational effort.

Perhaps the most significant advance of these approaches is the growing body of literature that seeks to augment the trajectory optimization with other state variables. This includes hybrid state variables [32], including those that model contact [33]. One example in particular, the choice of appropriate relaxations for contact constraints between surfaces, yields the ability for systems to form plans that incorporate grasping [34], or bipedal motion [35]. The results from these works suggest that intelligent planning of complex, multi-stage behavior comes ever close to reality.

Interestingly, if Gaussian noise is assumed to perturb the system, the typical modeling assumption, the resulting trajectories of maximal likelihood of the optimization are identically equal to those of the deterministic case. This is one additional motivation for a Hamilton Jacobi Bellman-based approach. One would expect, particularly for robotic systems with impact dynamics, that if noise in the system were to increase, more cautious trajectories would be desirable.

1.3.3 Lyapunov Theory

In the study of nonlinear systems, the primary avenue for investigation has been that of Lyapunov theory, wherein an energy like function is used to show some measure of distance from a stability point decays over time. The beginning of the theory lies in the study of autonomous systems,

$$\dot{x} = f(x(t)) \tag{1.15}$$

where f is locally Lipschitz in a domain $D \subset \mathbb{R}^n$. Lyapunov theory is concerned with the study of such systems around an equilibrium point. Suppose x^* is such an equilibrium, typically taken to be the origin of the domain without loss of generality by a simple coordinate transform. The system may have the following two properties:

Definition 2. *The equilibrium $x = 0$ of* (1.15) *is:*

- *Stable, if for each $\epsilon > 0$ there is a $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\| < \delta \implies |x(t)| < \epsilon, \quad \forall t \geq 0 \tag{1.16}$$

- *Asymptotically stable if it is stable and δ can be chosen such that*

$$\|x(0)\| < \delta \implies \lim_{t\to\infty} x(t) = 0. \tag{1.17}$$

The study of the above two properties is facilitated by the development of an "energy-like" function, called a *Lyapunov function* [26]. The use of these functions becomes apparent as a consequence of the following theorem:

Theorem 3. *([26]) Consider the system* (1.15)*, and let $D \subseteq \mathbb{R}^n$ be a neighborhood of the origin. If there is a continuously differentiable function $V : D \to \mathbb{R}$ such that the following two conditions are satisfied:*

1. *$V(x) > 0$ for all $x \in D \backslash \{0\}$ and $V(0) = 0$, i.e., $V(x)$ is positive definite in D.*

2. $-\dot{V}(x) = -\frac{\partial V}{\partial x} f(x) \geq 0$ *for all* $x \in D$*, i.e.,* $\dot{V}(x)$ *is negative semidefinite in* D*.*

then the origin is a stable equilibrium. If in condition (2) $\dot{V}(x)$ *is negative definite in* D *then the origin is asymptotically stable. If* $D = \mathbb{R}^n$ *and* $V(x)$ *is radially unbounded, i.e.,* $V(x) \to \infty$ *as* $\|x\| \to \infty$*, then the result holds globally.*

The construction of Lyapunov Functions that certify system stability has advanced considerably due to the introduction of Sums of Squares Programming [36]. Previously, automatic algorithms were largely restricted to systems with linear dynamics, i.e., where $f(x) = Ax$ for a linear operator A. This has allowed for Lyapunov Functions to be synthesized for polynomial systems, demonstrated in [36], or more general vector fields, as in [37].

Unfortunately, the problem of control design for stabilization, rather than the analysis of an existing closed loop system, has proved more difficult. It is possible to generalize Lyapunov functions to incorporate control inputs, a generalization is known as Control Lyapunov Function (CLF) [38, 39, 40, 41, 42]. The presence of a CLF is sufficient for the construction of a stabilizing controller, as the control law is now implicit in the Lyapunov function. However, the synthesis of a CLF for a general system remains an open question.

Yet, for several large and important classes of systems, CLFs are in fact known and may be used for stabilization, with a review of this theory available in [40]. The drawback is that these CLFs are hand-constructed and may be shown to be arbitrarily suboptimal, using excess control effort and possibly actuating against the natural dynamics of the system unnecessarily. A way to alleviate this issue is through the incorporation of Receding Horizon Control (RHC), wherein the Euler-Lagrange equations are used to construct a locally optimal trajectory [43]. There it has been shown that by setting the terminal cost in the RHC problem in accordance with a CLF, a stabilization guarantee is produced. By utilizing the RHC framework, a cost function may be approximated over a finite horizon of the trajectory, improving the closed loop system cost. The stability properties of Lyapunov theory are then married with RHC through a trade-off with (local) optimality.

1.3.4 Method of Moments

Methods to calculate the solution to the Hamilton Jacobi Bellman equation via semidefinite programming have been proposed by Lasserre et al. [44, 45]. In their work, the solution and the optimality conditions are integrated against monomial test functions, producing an infinite set of moment constraints. By truncating to any finite list of monomials, the optimal control problem is reduced to one of semidefinite optimization. The method is quite general, applicable to any system with polynomial nonlinearities.

These moment techniques are intimately related to sum of squares programming. It can be shown that the two problems are in fact convex duals of one another [46]. The success and generality of these techniques have led to the development of a number of software tools that allow for policy synthesis, and scalability has been further improved by studying the sparsity structure of coefficient matrices [47]. The overall moment and sum of squares based techniques is not limited to optimal control, and in fact are being used to tackle difficult problems in combinatorics and other fields [48]. These techniques provide an interesting contrast with those presented later in this thesis, which require additional structure on the systems of interest.

1.3.5 Navigation Functions

Navigation functions can be viewed as a relaxed variant of the value function. These functions were introduced by Koditschek and Rimon [49, 50, 51] to remedy the local minima problem in the classical potential field method of robot motion planning [52]. Their early work on navigation functions focused primarily on the existence and discovery of potential functions whose gradient would lead a point mass model of a robot from any point in the robot's configuration space to a desired goal. Later work extended the navigation function concept to incorporate multiple agents [53], and sensory input [54]. Formally, the definition is as follows.

Definition 4. *(From [50]) Let q_d be a goal configuration in $\mathcal{F}$, the free configuration space of a system. A map $\varphi : \mathcal{F} \rightarrow [0, 1]$ is a* navigation function *if it is*

1. *smooth on $\mathcal{F}$ (at least a $C^{(2)}$ function);*
2. *polar at q_d, i.e., has a unique minimum at q_d on the path-connected component of $\mathcal{F}$ containing q_d;*
3. *admissable on $\mathcal{F}$, i.e., uniformly maximal on the boundary of $\mathcal{F}$;*
4. *a Morse function, i.e., the Hessian at critical points is nonsingular.*

Navigation functions have been successful in part due to their rapid computability and transparent nature. The navigation function provides both a global plan, as well as a feedback controller that follows the gradient of the navigation function. This allows the total motion planning and execution problem to be abstracted into a trajectory planner (the path followed by the gradient of the navigation function) and a path following controller [51]. The price to pay for this convenience is optimality: ignoring dynamics in the quest to follow the gradient will rarely result in a procedure that minimizes control effort. This deficiency is in part due to the fact that the system dynamics do not enter into the navigation functions calculation, and may result in unexpected and unstable behavior in some contexts [55]. The work presented later in this thesis shows the connection between navigation functions and more general optimal control theory, allowing for system dynamics and stochasticity to be included if this is desirable. It also becomes possible to include more sophisticated weighting of various goals, such as the desire for minimum-time trajectories.

In the construction and intuition behind navigation functions, implicit is the idea of robustness. Since navigation functions are defined over the entire free configuration space, small deviations from the desired path place the robot in nearby locations where the desirable behavior is similar. Indeed, smoothness of the solution is typically enforced. This work furthers the understanding of robustness properties of navigation functions which has largely heretofore been only analyzed ad hoc.

Finally, some approaches for finding navigation functions require difficult calculations and may not extend to complex obstacle geometries, while others have difficulty scaling to large configuration space sizes. The methods introduced in this thesis help to minimize the computational difficulty in large state-spaces and alleviate some of these issues.

1.3.6 Sampling based planners

An alternative set of methods has been developed based on methods to generate *feasible* solutions to motion planning problems by sampling from the system state space. Among the most popular of these techniques are Rapidly Exploring Random Trees (RRT) [56], and Probabilistic Roadmaps (PRM) [57].

These techniques rely on decomposing the configuration space at two levels. At the higher level is a graph, where each node represents a region of the system state space, and each edge represents a plan from between the two regions. This decomposition allows for plans to be generated in non-convex state spaces with difficult kinematics, while keeping computational effort low as each edge is a local path planning problem.

The two methods diverge in that RRTs sample and explore nodes near a tree of already connected states. In PRMs, the graph nodes are created a priori and then connected by a local planner. In each case, optimality may be induced by weighting the edges of these graphs, and then doing a search for the shortest path, typically using Djisktra's algorithm. Additionally, the sampling may be biased towards the goal region, allowing for computational effort to be concentrated in a goal seeking manner. The methods also have probabilistic guarantees, ensuring that feasibility and optimality may be guaranteed in the sampling limit. However, sampling an infinitely dense set of points may be burdensome.

By giving up optimality, these algorithms have achieved remarkably rapid execution times and are the de-facto choice of robotic researchers in many motion planning problems. As their underlying mechanism is Monte Carlo sampling, these techniques also scale favorably with dimensionality, even allowing their use on sophisticated manipulation systems [58]. However, the neglect of dynamics and stochasticity is to some extent responsible for the limitations of these techniques, and also for the typical "jerky" motions that result. Typically, motion planning takes place in constrained environments, with motions limited to moderate speed.

It is only recently that optimal control techniques based on convex programming and Model Predictive Control-type algorithms have come to have comparable performance [59, 60]. These alternative approaches have a number of benefits beyond optimality for the model system: they can also incorporate stochasticity and various relevant task criteria. Indeed, many of the approaches reviewed here are now coming to be seen to be the only approaches capable of capturing such necessary nonlinearities as impact dynamics and friction [35].

1.4 The Linear Hamilton Jacobi Bellman Equation

The study of Linear Hamilton Jacobi Bellman equation largely began with Kappen [61, 62], who discovered that particular assumptions on the structure of a dynamical system allows the transformation of the optimal control equation to a linear form. The work focused on calculating solutions via *path integral* techniques, popular in the physics community. This underlying structure was also discovered by Todorov in parallel [63], who began with analysis of particular Markov decision processes, and who showed the connection between the two paradigms. This was built upon by Theodorou et al. [64] into the Path Integral framework in use with Dynamic Motion Primitives. Therein, sampling of system trajectories is augmented with the use of suboptimal policies, producing better estimates of the dynamics when executing an optimal policy. The resulting sample trajectories can then be used to in turn improve the policy, and then the process is iterated. These results have been developed in a number of compelling directions [65, 66, 67, 68, 69].

The linear solvabiity of the Hamilton Jacobi Bellman equation arises as follows. Let $x_t \in \mathbb{R}^n$ as the system state at time t, control input $u_t \in \mathbb{R}^m$, and let the system dynamics evolve according to the equation

$$dx_t = \left(f\left(x_t\right) + G\left(x_t\right) u_t\right) dt + B\left(x_t\right) \mathcal{L}\, d\omega_t \tag{1.18}$$

on a compact domain Ω. The expressions $f(x)$, $G(x)$, $B(x)$ are assumed to be smoothly differentiable, but possibly nonlinear, functions, and ω_t is a Brownian motion (i.e., a stochastic process such that ω_t has independent increments with $\omega_t - \omega_s \sim \mathcal{N}(0, t-s)$, for $\mathcal{N}(\mu, \sigma^2)$ a normal distribution). The matrix $\mathcal{L}$ is constant. Assume that the system incurs cost r_t at time t according to

$$r(x_t, u_t) = q(x_t) + \frac{1}{2} u_t^T R u_t \tag{1.19}$$

where $q(x)$ is a smooth, state dependent cost. It is assumed that $q(x) \geq 0$ for all x in the problem domain. The goal is to minimize the expectation of the cost functional

$$J(x, u) = \phi_T(x_T) + \int_0^T r(x_t, u_t)\, dt \tag{1.20}$$

where ϕ_T represents a state-dependent terminal cost. The solution to this minimization is known as the *value function*, where, beginning from an initial point x_t at time t

$$V(x_t) = \min_{u_{[t,T]}} \mathbb{E}\left[J(x_t, u_t)\right] \tag{1.21}$$

where the shorthand $u_{[\tau,T]}$ is used to denote the trajectory of $u(t)$ over the time interval $t \in [\tau, T]$.

The associated Hamilton-Jacobi-Bellman equation, derived previously in Section 1.2.2 and [10], is

$$-\partial_t V = \min_{u_{[0,t]}} \left(r + (\nabla_x V)^T f + \frac{1}{2} Tr\left((\nabla_{xx} V) G \Sigma_\epsilon G^T \right) \right) \tag{1.22}$$

where $\Sigma_\epsilon \triangleq \mathcal{L}\mathcal{L}^T$. As the control effort enters quadratically into the cost function it is a simple matter to solve for it analytically by substituting (1.19) into (1.22) and finding the minimum, yielding:

$$u^* = -RG^T(\nabla_x V). \tag{1.23}$$

The minimal control, u^*, may then be substituted into (1.22) to yield the following nonlinear, second order partial differential equation

$$\begin{aligned} -\partial_t V = q + (\nabla_x V)^T f - \frac{1}{2} (\nabla_x V)^T G R^{-1} G^T (\nabla_x V) \\ + \frac{1}{2} Tr \left((\nabla_{xx} V) B \Sigma_\epsilon B^T \right). \end{aligned} \tag{1.24}$$

The difficulty of solving this PDE is what usually prevents the value function from being directly solved for. However, it has recently been found [62, 63, 70] that with the assumption that there exists a $\lambda \in \mathbb{R}$ and a control penalty cost $R \in \mathbb{R}^{n \times n}$ satisfying this equation

$$\lambda G(x) R^{-1} G(x)^T = B(x) \Sigma_\epsilon B(x)^T \triangleq \Sigma_t \tag{1.25}$$

and using the logarithmic transformation

$$V = -\lambda \log \Psi \tag{1.26}$$

it is possible, after substitution and simplification, to obtain the following linear PDE from Equation (1.24)

$$-\partial_t \Psi = -\frac{1}{\lambda} q \Psi + f^T (\nabla_x \Psi) + \frac{1}{2} Tr \left((\nabla_{xx} \Psi) \Sigma_t \right). \tag{1.27}$$

This transformation of the value function, which is termed the *desirability* [63], provides an additional, computationally appealing, method by which to calculate the value function.

The difference in computationally difficulty between nonlinear and linear PDEs is analogous to the difference between linear and nonlinear systems in control. Typically, nonlinear PDEs must be solved via iterative linearization, with uniqueness or existence of the solution not guaranteed. The reduction to linearity removes or alleviates many of these considerations, and allows for the novel computational techniques developed in this thesis.

Remark 5. *The condition* (1.25) *can roughly be interpreted as a controllability-type*

	Cost Functional	Desirability PDE
Finite	$\phi_T(x_T) + \int_0^T r(x_t, u_t)dt$	$\frac{1}{\lambda}q\Psi - \frac{\partial \Psi}{\partial t} = L(\Psi)$
First-Exit	$\phi_{T_*}(x_{T_*}) + \int_0^{T_*} r(x_t, u_t)dt$	$\frac{1}{\lambda}q\Psi = L(\Psi)$
Average	$\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\int_0^T r(x_t, u_t)dt\right]$	$\frac{1}{\lambda}q\Psi - c\Psi = L(\Psi)$

Table 1.1: Linear Desirability PDE for Various Stochastic Optimal Control Settings, from [63]. $L(\Psi) := f^T(\nabla_x \Psi) + \frac{1}{2}Tr\left((\nabla_{xx}\Psi)\Sigma_t\right)$

condition: the system controls must span (or counterbalance) the effects of input noise on the system dynamics. A degree of designer input is also given up, as the constraint restricts the design of the control penalty R, requiring that control effort be highly penalized in subspaces with little noise, and lightly penalized in those with high noise. Additional discussion may be found in [63].

The boundary conditions of (1.27) correspond to the exit conditions of the optimal control problem. This may correspond to colliding with an obstacle or goal region, and in the finite horizon problem there is the added boundary condition of the terminal cost at $t = T$. These final costs must then be transformed according to (1.26), producing added boundary conditions to (1.27).

Linearly solvable optimal control is not limited to the finite horizon setting. Similar analysis can be performed to obtain linear Hamilton Jacobi Bellman PDEs for infinite horizon average cost, and first-exit settings, with the corresponding cost functionals and PDEs shown in Table 1.1.

1.4.1 Path Integral Control

Previously, the greatest research effort into harnessing the linear Hamilton Jacobi Bellman equation was through a technique that has come to be called Path Integral Control. The method arises from the recognition that the PDE (1.27) is in fact a Chapman-Kolmogorov PDE [64]. Certain classes of linear PDEs may be connected to corresponding Stochastic Differential Equations (SDE) by the Feynman-Kac formula

[71, 72]. Applying the formula for this problem, (1.27) may be transformed to

$$\begin{aligned}\Psi_{t_i} &= \mathbb{E}_{\tau_i}\left[\Psi_{t_N} e^{-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt}\right] \\ &= \mathbb{E}_{\tau_i}\left[\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda}\int_{t_i}^{t_N} q_t dt\right)\right]\end{aligned}$$

where τ_i represents a sample, stochastic trajectory, and the expectation is taken over the set of sample trajectories [73]. Discretizing this in time yields

$$\Psi_{t_i} = \lim_{dt\to 0} \int p(\tau_i \mid x_i) \exp\left(-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=1}^{N-1} q_{t_j} dt\right)\right) d\tau_i \tag{1.28}$$

where $\tau_i = (x_{t_i}, \ldots, x_{t_N})$ is a sample path that evolves according to the natural stochastic dynamics of the system, and $p(\tau_i \mid x_i)$ is the probability of a trajectory segment τ_i under the uncontrolled dynamics of the system.

This sampling-based approached may then combined with Reinforcement Learning techniques, allowing for the problem to be solved when the dynamics are uncertain or unknown. The policies are further parameterized by Dynamic Movement Primitives (DMPs) [74], allowing for efficient representation and learning. The Path Integral approach has several appealing properties. Because it is Monte Carlo based, the method scales benignly with dimension. The result is the ability to control high dimensional systems, even including a four legged robot dog [75], with a dimensionality of twelve. The method has also been extended to plan through multiple sub-goals, allowing for trajectories that incorporate events such as contact to be constructed [66].

1.5 Markov Decision Processes

Currently, the most commonly used framework for solving stochastic optimal control problems is via state space discretization, transforming the problem into a Markov decision process (MDP). Assume the system may occupy one of a finite set of states $x \in \mathcal{X}$. At each state, an agent has a set of actions $u \in \mathcal{U}$ that may be chosen. When such an action is chosen, the system transition moves from state x to x' with probability $\mathcal{T}(x, u, x') = \mathbb{P}(x' \mid x, u)$. The agent also receives an award at each point in time dependent on its state and choice of action $R(x, a)$.

Like the derivations in Section 1.2.1, a value function $V(x)$ may be proposed, representing the cost of any particular (potentially suboptimal) policy beginning from an arbitrary state x. The central difference with the current derivation is that the state and action set are finite, and the transition function is no longer governed by smooth dynamics $f(x_k, u_k)$. Adopting the system parameters of Section 1.2.1, define a state dependent control law π such that $u = \pi(x)$, then the value of that control law, known as a policy in the MDP literature, is given as

$$V(x) = \mathbb{E}_{x_{[0,t_{T_f}}} \left[\sum_{t=0}^{T_f} \ell(x_t, \pi(x_t)) \right]$$

The optimal value function V^*, corresponding to a yet unknown optimal policy π^* arises when the accrued cost is minimized

$$V^*(x_\tau) = \min_u \ell(x, u) + \mathbb{E}_{x_{\tau+1} \sim p(\cdot|x_\tau, u)} \left[\sum_{t=\tau+1}^{T_f} \ell(x_t, \pi^*(x_t)) \right]$$

Invoking Bellman's principle of optimality results in the relation

$$V(x_\tau) = \min_u \ell(x, u) + \mathbb{E}_{x_{\tau+1} \sim p(\cdot|x_\tau, u)} \left[V(x_{\tau+1}) \right]$$

Expanding the expectation over the finite set of states and actions yields

$$V(s) = \min_u \sum_{x'} T(x, u, x') \left(R(s, u) + V(s') \right)$$

This is the *Bellman* equation, and gives rise to the most fundamental algorithm to solve the discrete-state stochastic optimal control problem.

Value Iteration is performed by assuming an under approximation for the value function at each point in the state space. Typically, it is initialized as the cost at each state $\ell(x, u)$ for some control u. The approximation may then be improved iteratively, yielding a sequence of improving approximation $V_0, \ldots, V_i$. The process, known as Value Iteration, is performed by sweeping through the state space, improving the choice of action at each state.

$$V_{i+1}(s) = \min_{u} \sum_{x'} T(x,u,x') \left(R(s,u) + V_i(s') \right)$$

The operator that represents this sweep is deemed the *Bellman operator* and may be shown to be contractive [8]. Value iteration may therefore be repeated until convergence to obtain the optimal value function.

There exist a number of variants upon value iteration. One possibility is to selectively improve the policy and value function independently. This gives rise to *policy iteration*, which may be more rapid in practice. Unfortunately, the Bellman operator is nonlinear, and convergence may require significant computational time.

Alternatives have been proposed that rely on Linear Programming [76]. The computational benefit isn't direct, but the Linear Programming variant allows for a different tack of analysis. The curse of dimensionality may be mitigated in this context by parameterizing the value function with a sparse set of basis, giving rise to Approximate Dynamic Programming [9]. Allowing the basis to change online results in Adaptive Dynamic Programming (ADP), also available in [9]. These techniques result in linear programming problems that have constraint sets which grow exponentially with dimensionality [77]. Nonetheless, these techniques are the most popular methods to deal with the curse of dimensionality and have even been used to surpass human capabilities on complex time dependent games via synthesis with modern machine learning techniques [78].

The examination of stochastic optimal control problems in a discrete state space has been the more popular approach in the literature. A number of reasons for this exist, including no need for partial differential equations, and the complexities and continuity considerations these entail, as well as the natural ability to incorporate switched behavior. There has also historically been little in the way of computational gain from considering system behavior in its natural continuous state space. Success has been significant, in topics ranging from networked systems [79, 80], robotics [81], among many others. The framework may also be extended to deal with uncertain observations of the current state, giving rise to Partially Observable Markov Decision Processes (POMDP) [82].

1.5.1 Linear MDPs

The development of the linear Hamilton Jacobi Bellman equation has a parallel in the development of linear Markov decision processes (MDP) [63, 70]. Again, the presence of structural assumptions on the noise allows for a significant reduction in computational effort necessary to solve these problems. In [63] it is further shown that linear MDPs and the linear Hamilton Jacobi Bellman equation may in fact be derived from one another through discretization or continuous limit arguments.

This formulation has been extended by Todorov et. al. in a number of compelling directions. Beyond being able to solve MDP control design problems in a more computationally efficient manner, in the preliminary work a *z-learning* approach, based on Q-learning, was proposed [70, 63]. The method demonstrated superior convergence and was able to take advantage of the analytical results of the linear MDPs.

Beyond learning, in [83] the linearity of these stochastic optimal control problems was harnessed to compose solutions to new problems out of existing solutions to different problems. This allowed for solutions to LQG problems to be composed to solve non-LQG problems. The approach also allowed for a principled method of compression to be performed via the Singular Value Decomposition, paving the way for improved scalability of stochastic optimal control. The notion of composing solutions at essentially zero cost was adopted and extended in [6] to rapidly generate solutions to temporal problems efficiently, corresponding to Chapter 5 of this thesis.

Efficient computational techniques were also developed in [84, 85], where an adaptive set of basis were used for eigenfunction approximation in the average-cost setting via collocation methods. A review of the many directions taken by Todorov et. al. is available at [67, 63].

Chapter 2

Optimization Based Numerical Methods

2.1 Semidefinite Programming for Optimal Control

This chapter presents a novel method to solve stochastic optimal control problems using polynomial optimization and semidefinite programming. *Sum of squares* (SOS) techniques [86] are used to construct sub- and super-solutions to the value functions of linearly solvable Hamilton Jacobi Bellman equations. This approach allows for optimal control solutions to be computed quickly, with globally optimal guarantees. In contrast to dynamic programming approaches, no discretization is required, postponing the curse of dimensionality and eliminating a potential source of approximation error. Moreover, the formulation leads directly to gap theorems, or bounds, on the approximation error.

Building on this technique, a domain decomposition augmentation is proposed in Section 2.2, allowing for the state space to be split into disjoint problems that are linked along shared boundaries. The power of the approach lies in the use of lower order polynomials in disjoint areas of the state space, rather than one high degree polynomial over the entire domain, improving the scalability of the method and allowing for local phenomena to be accurately captured.

The underlying methodology proposed in this chapter is shown to be useful beyond the solution to Hamilton Jacobi Bellman equations, and is applicable more broadly to the analysis of elliptic and parabolic PDEs. Via the *Positivstellensatz*, uncertainty in the problem data is incorporated in to the optimization problem. The result is an algorithm to produce upper and lower bounds over the solutions to PDEs. The generality of the technique has implications in the field of Uncertainty Quantification (UQ) of systems governed by partial differential constraints.

Although sharing the same set of tools and with a similar goal in mind, the method developed in this chapter contrasts with those of Lasserre et al., detailed in Section 1.3, in that candidate approximate solutions of the value function itself are proposed, and thus avoiding the need to include the control signal in the polynomial basis. This property lessens the computational burden in the resulting optimization problem, and takes advantage of the specific structure of this class of stochastic systems. The method also allows for both upper and lower bounds to the value function to be calculated, whereas only lower bounds were previously possible.

2.1.1 Sum of Squares Programming

To set the stage for the contributions of this chapter, Sum of Squares (SOS) programming, originally proposed in the thesis of Parrilo [87] is reviewed. These tools will be key in the development of approximate solutions to the Hamilton Jacobi Bellman equation (1.27), which specifies a set of *partial differential* equality constraints that the optimal solution must satisfy. Instead of satisfying these constraints exactly, as in Galerkin or collocation techniques, the equality constraints are relaxed directly. The optimization problem is then to find the best approximate solution that lies in the set of polynomials that satisfy these inequality constraints. This is done by reducing these inequalities to a *semialgebraic set*, allowing for the tools of algebraic geometry to be employed. Specifically, SOS programming provides a method to perform optimization over such a set.

Definition 6. *Let $x = (x_1, \ldots, x_n)$, $x \in \mathbb{R}^n$ and $\alpha = (\alpha_1, \ldots, \alpha_n)$, $\alpha \in \mathbb{N}^n$. The function $z_\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ is a monomial in $(x_1, \ldots, x_n)$ of degree $|\alpha| = \sum_{i=1}^n \alpha_i$. A polynomial p in x with coefficients in $\mathbb{R}$ is a linear combination of a finite set of* *monomials*

$$p(x) = \sum_{\alpha} c_\alpha x^\alpha = \sum_{\alpha} c_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}, \quad c_\alpha \in \mathbb{R}. \tag{2.1}$$

For brevity of notation, define the ring of polynomials in $(x_1, \dots, x_n)$ with real coefficients as $\mathbb{R}[x] \triangleq \mathbb{R}[x_1, \dots, x_n]$. A *semialgebraic set* is a subset of $\mathbb{R}^n$ that is specified by a finite number of polynomial equations and inequalities.

$$S = \{x \in \mathbb{R}^n \mid p_i(x) \geq 0, i = 1, \dots, n\}$$

An example is

$$S = \left\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1, x_1^3 - x_2 \leq 0\right\}.$$

Such a set is not necessarily convex, and testing membership in the set is NP-Hard in general [88, 87]. However, there exists a class of semialgebraic sets that are in fact semidefinite-representable. Key to this development will be the ability to test for non-negativity of a polynomial.

A multivariate polynomial $p(x)$ is a *sum of squares* (SOS) if there exist polynomials $p_1(x), \dots, p_m(x)$ such that

$$p(x) = \sum_{i=1}^{m} p_i^2(x).$$

A seemingly unremarkable observation is that a sum of squares is always positive. Thus, a sufficient condition for non-negativity of a polynomial is that the polynomial is SOS. Perhaps less obvious is that membership in the set of SOS polynomials may be tested as a convex problem and therefore polynomial time-solvable. Denote the function $p(x)$ being SOS as $p(x) \in \Sigma(x)$, where $\Sigma(x)$ is the set of all SOS polynomials. The key to this reduction in complexity is the following result.

Theorem 7. *([89]) A polynomial $p(x)$ of degree $2d$ is a sum of squares if and only if there exists a positive semidefinite matrix Q and a vector of monomials $Z(x)$ con-*

taining monomials in x of degree less than or equal to d such that

$$p = Z(x)^T Q Z(x). \tag{2.2}$$

The Q matrix in (2.2) is referred to as the *Gram Matrix.* The monomials of $Z(x)$ are not in general algebraicaly independent, meaning that if the equation is expanded and coefficients are matched, there will be free parameters in Q. The result is that optimizations may take place over Q with the constraint $Q \succeq 0$, i.e. Q is positive semidefinite, placing the problem of SOS non-negativity in the realm of semidefinite programming.

Theorem 8. *([87]) Given a finite set of polynomials $\{p_i\}_{i=0}^m \in \mathbb{R}^n$ the existence of $a_i \in \mathbb{R}$ for $i = 1, \ldots, m$ such that*

$$p_0 + \sum_{i=1}^{m} a_i p_i \in \Sigma(x)$$

is a semidefinite programming feasibility problem.

Thus, while the problem of testing non-negativity of a polynomial is intractable in general, by constraining the feasible set to a SOS, the problem becomes tractable. The converse question, is a non-negative polynomial necessarily a sum of squares is, unfortunately, false. This indicates that this test is conservative [87]. Nonetheless, SOS feasibility will be sufficiently powerful for the purposes of this work. Details of how SOS feasibility are reducible to semidefinite programs are given in [36], and have become well known in the control community.

2.1.1.1 The Positivstellensatz

Using SOS theory, it is possible to determine whether a particular polynomial, possibly parameterized, is a sum of squares. The next step is to determine how to combine multiple polynomial inequalities, i.e. semialgebraic sets of the form

$$P = \{x \in \mathbb{R}^n \mid p_i(x) \geq 0 \text{ for all } i = 1, \ldots, m\}$$

for polynomial functions $p_i(x)$ where $x \in \mathbb{R}^n$. The answer is given by Stengle's *Positivstellensatz.*

Theorem 9. *(Positivstellensatz [90]) The set*

$$X = \{x \mid p_i(x) \geq 0, h_j(x) = 0 \text{ for all } i = 1, \dots m,\, j = 1, \dots p\} \tag{2.3}$$

is empty if and only if there exists $t_i \in \mathbb{R}[x]$, $s_i, r_{ij}, \dots \in \Sigma(x)$ *such that*

$$-1 = s_0 + \sum_i h_i t_i + \sum_i s_i p_i + \sum_{i \neq j} r_{ij} p_i p_j + \cdots \tag{2.4}$$

Although this theorem is presented in terms of feasibility, it is easily adapted for the purposes of optimization. Given the problem

$$\begin{aligned} &\min\, p_0(x) \\ &s.t.\ p_i(x) \leq 0 \quad \forall i \in 1, \dots, k \end{aligned}$$

a slack factor γ may be introduced to frame the equivalent infeasibility problem

$$\begin{array}{ll} \max & \gamma \\ s.t. & \left.\begin{array}{l} p_0(x) \leq \gamma \\ p_i(x) \leq 0 \ \ \forall i \in 1, \dots, k \end{array}\right\} \textit{infeasible} \end{array}$$

which is in a form directly applicable to the *Positivestellensatz.*

By setting some of the *Positivestellensatz* multipliers, such as r_{ij} or r_{ijk}, to zero, a sufficient condition for infeasibility may be created. Alternatively, it is possible to limit the degree of the multipliers h_i, s_i, r_{ij}. The search for infeasibility may therefore begin with a limited polynomial degree, increasing the degree if additional precision is required. This creates a *hierarchy* of semidefinite relaxations of increasing complexity but also with a decrease in the suboptimality of the solution. This construction is known more broadly as a Lasserre Hierarchy [46], or Theta Body relaxation [91].

2.1.2 Sum-of-Squares Relaxation of the linear HJB PDE

Sum of squares programming has found many uses in combinatorial optimization [48], control theory [36], as well as other applications [92]. Its use is now expanded to include finding approximate solutions to the value function of the stochastic optimal control problem.

Obtaining solutions to linear PDEs is far from trivial, with the multitude of numerical methods a testament to the many issues that arise. Control theoretic techniques typically avoid many of these issues by not considering the partial differential nature of the problem directly. The synthesis of these two disciplines will require an understanding of the issues unique to both.

A candidate polynomial is proposed as the solution to the linear desirability Hamilton Jacobi Bellman PDE. While the value function may in fact be discontinuous, the modeling assumption is made that it may be approximated to a sufficiently high accuracy given a polynomial of sufficient degree. Furthermore, although the solution to the Hamilton Jacobi Bellman equation is discontinuous in some locations, in many applications, such as many robotics and control problems of interest, it will remain continuous over large portions of the domain. This assumption of underlying smoothness may be seen as seeking viscosity solutions [10] to the problem (1.27), in effect placing a smoothness requirement on the solution. Furthermore, the diffusivity that enters the equation due to the presence of noise has the effect of enforcing this assumption, as diffusitivity terms smooth the solutions directly.

The following discussion proceeds with the finite horizon problem (see Section 1.2.2), but similar steps apply to all the problems listed in Table 1.1. The assumption is made that the control problem is defined over a compact domain $\mathbb{S}$ that is representable as a semialgebraic set, as is its boundary $\partial\mathbb{S}$.

The equality constraint of (1.27) may be relaxed, yielding the following constraints that are necessary for an over-approximation of the desirability function

$$\frac{1}{\lambda} q \Psi \leq \partial_t \Psi + f^T (\nabla_x \Psi) + \frac{1}{2} Tr \left((\nabla_{xx} \Psi) \Sigma_t \right). \tag{2.5}$$

Hereafter, solutions to the above inequality will be denoted as Ψ, and exact solutions to (1.27) denoted as Ψ^*, the optimal desirability function. For brevity, as in Table 1.1, define the differential operator

$$L(\Psi) := f^T (\nabla_x \Psi) + \frac{1}{2} Tr\left((\nabla_{xx} \Psi) \Sigma_t \right). \tag{2.6}$$

To obtain the best approximation Ψ for a given polynomial order, the pointwise error of the approximation may be minimized in the optimization problem

$$\begin{aligned} \min \quad & \gamma \\ s.t. \quad & \gamma - \left(-\frac{1}{\lambda} q \Psi + \partial_t \Psi + L(\Psi) \right) \geq 0 \end{aligned}$$

for $x \in \mathbb{S}$. The boundary conditions of (1.27) correspond to the exit conditions of the optimal control problem. In all problems these conditions may correspond to colliding with an obstacle or goal region, and in the finite horizon problem there is the added boundary condition of the terminal cost at $t = T$. These final costs are transformed according to Eq. (1.26), producing the added constraint

$$\Psi \mid_{\partial \mathbb{S}} = e^{-\frac{\phi_T(x_T)}{\lambda}}$$

where $\phi_T(x_T)$ is the terminal cost from Eq. (1.20). As shown below, this constraint may be also be relaxed as an inequality. The complete optimization problem is then

$$\begin{aligned} \min \quad & \gamma \\ s.t. \quad & \frac{1}{\lambda} q \Psi \leq \partial_t \Psi + L(\Psi) & x \in \mathbb{S} \\ & \gamma \geq -\frac{1}{\lambda} q \Psi + \partial_t \Psi + L(\Psi) & x \in \mathbb{S} \\ & \Psi \leq e^{-\frac{\phi_T(x)}{\lambda}} & x \in \partial \mathbb{S}. \end{aligned} \tag{2.7}$$

As the inequalities are defined over polynomials, this optimization is defined over a semialgebraic set. This formulation may be made tractable as follows.

Proposition 10. *The optimization problem* (2.7)*, where inequality constraints are relaxed to SOS membership, may be solved as a semidefinite optimization program.*

Proof. Let us propose a candidate solution to the optimization of Ψ, a polynomial of fixed degree n, denoted Ψ_n. Each of the inequality constraints are non-negativity constraints over a polynomial and are therefore a semialgebraic set. The full set of constraints is an intersection of semialgebraic sets and therefore also a semialgebraic set. When the inequalities in this set are relaxed as SOS constraints, membership in the constraint set may be tested as a semidefinite program by Theorem 8. The optimization over this set is then enabled by Theorem 9. □

Furthermore, one can in fact guarantee that the exact and polynomial approximate desirability functions have a bounded relationship.

Theorem 11. *Given a feasible solution pair* $\{\Psi, \gamma\}$ *to* (2.7)*, and if* Ψ^* *is the exact, optimal solution to* (1.27)*, then* $\Psi(x) \leq \Psi^*(x)$ *for all* $x \in \mathbb{S}$*.*

Proof. Consider the first-exit case for simplicity, and define the error between approximation Ψ and the optimal desirability, Ψ^*, as $e = \Psi - \Psi^*$. Then, as all operators are linear,

$$\begin{aligned}
\frac{1}{\lambda} q e &= \frac{1}{\lambda} q \left(\Psi - \Psi^*\right) \\
&= \frac{1}{\lambda} q \Psi - L(\Psi^*) \\
&\leq L(\Psi) - L(\Psi^*) \\
&\leq L(e)
\end{aligned}$$

since $\frac{1}{\lambda} q \Psi^* = L(\Psi^*)$, since $\frac{1}{\lambda} q \Psi \leq L(\Psi)$. Defining the augmented operator

$$P(e) := L(e) - \frac{1}{\lambda} q e$$

then P is an elliptic operator, since the Hamilton Jacobi Bellman equation is elliptic for the first-exit case, and by the weak maximum principle for elliptic operators [93]

$$\sup_{\mathbb{S}} e \leq \sup_{\partial \mathbb{S}} e^{+} \tag{2.8}$$

where $e^{+} = \max(e, 0)$ and e is non-positive on the boundary. Thus, the error remains less than zero everywhere, implying that $\Psi \leq \Psi^{*}$, and that Ψ is indeed a lower bound.

The weak maximum principle for parabolic operators can similarly be used in the case where the desirability PDE is parabolic, i.e., in the finite horizon case where the Hamilton Jacobi Bellman equation is time-dependent. For the finite horizon case, define

$$P(e) := L(e) - \partial_t - \frac{1}{\lambda} q e.$$

Note that the PDE (1.27) yields $L(e) + \partial_t - \frac{1}{\lambda} q e = 0$, apparently yielding an operator with the incorrect sign on the time-derivative. That is, the expression must be of the form $L(e) - \partial_t - \frac{1}{\lambda} q e = 0$ for the parabolic maximum principle to apply. The correct operator is recovered when it is remembered that the boundary condition along the time axis is assigned only at the terminal time $t = T$. Conventionally, the time boundary is assigned at the beginning of time, so the direction of time must be flipped for the Hamilton Jacobi Bellman equation. This aligns the Hamilton Jacobi Bellman equation with the convention for parabolic PDEs, with both proceeding forward in time, with the boundary conditions at $t = 0$.

By the same arguments as the elliptic case, the error $e = \Psi - \Psi^{*}$ is a subsolution of (1.27), i.e., $e \leq 0$. As $P(e)$ is a parabolic operator the weak maximum principle for parabolic operators dictates that the relation (2.8) is maintained [93], and once again the residual does not change signs, indicating that Ψ is indeed a lower bound. $\square$

This construction of a sum of squares program for lower bounds may be repeated for each of the objective functions found in Table 1.1. Furthermore, note that the proof underlying Theorem 10 may in fact be repeated with the relaxation inequality reversed in optimization (2.7), resulting in a superharmonic error function. In particular, the optimization

$$\begin{aligned}
\min \quad & \gamma \\
s.t. \quad & \frac{1}{\lambda} q\Psi \geq \partial_t \Psi + L(\Psi) && x \in \mathbb{S} \\
& \gamma \geq \frac{1}{\lambda} q\Psi - (\partial_t \Psi + L(\Psi)) && x \in \mathbb{S} \\
& \Psi \geq e^{-\frac{\phi_T(x)}{\lambda}} && x \in \partial\mathbb{S}.
\end{aligned} \tag{2.9}$$

yields a reverse, upper bound theorem.

Theorem 12. *Given a feasible solution pair* $\{\Psi, \gamma\}$ *to* (2.9)*, and if* Ψ^* *is the exact, optimal solution to* (1.27)*, then* $\Psi(x) \geq \Psi^*(x)$ *for all* $x \in \mathbb{S}$*.*

As both upper bound $\{\gamma^u, \Psi^u\}$ and lower bound $\{\gamma^l, \Psi^l\}$ solution pairs may be obtained from optimization (2.7), the pointwise distance of either of these approximate solutions from the optimal Ψ^* may trivially be bounded as

$$|\Psi(x) - \Psi^*(x)| \leq \gamma^u + \gamma^l \quad x \in \mathbb{S} \tag{2.10}$$

for $\Psi = \Psi^u$ or $\Psi = \Psi^l$.

It is straightforward to demonstrate that these bounds on the desirability also correspond to bounds on the value function.

Proposition 13. *Given a pointwise upper (lower) bound* $\Psi = \Psi^u$ *(or* $\Psi = \Psi^l$*) to a solution* Ψ^* *of* (1.27)*, then* $V = -\lambda \log \Psi$ *is a lower (upper) bound of* V^**, the solution to* (1.24)*.*

Proof. For $\Psi \geq \Psi^*$

$$\begin{aligned}
V &= -\lambda \log \Psi \\
&\leq -\lambda \log \Psi^* \\
&= V^*
\end{aligned}$$

since λ is always positive and logarithms are monotonic. Similar reasoning applies to the lower bound. $\square$

Remark 14. *Due to the nature of the log transformation* (1.26)*,* Ψ *is necessarily positive on the domain* $\mathbb{S}$*. This requirement may be included as an additional constraint* $\Psi \geq 0$ *in* (2.7)*. However, in this case the optimization for the lower bound of* Ψ^* *may not converge. It is possible to instead neglect this constraint. If the solution does at any point fall below zero, it will not be possible to transform it to the value function, and is therefore inappropriate as an approximate value function. The solution may nonetheless be informative.*

2.1.3 Analysis of SOS Relaxation

Some preliminary analysis of this approach demonstrates several appealing qualities. The first of these is that the convergence of the algorithm to calculate the upper bound is guaranteed.

Proposition 15. *There exists a constant* c *such that the SOS optimization problem to calculate the upper bound* Ψ^u *from (2.7) has a solution for all* $\gamma \geq c$

Proof. For the PDEs in Table 1.1 that are elliptic, all problem data are polynomial and therefore infinitely differentiable. By the elliptic regularity theorem [94], the solution Ψ^* is infinitely differentiable and therefore continuous. As the differential operator $L(\Psi)$ is a linear operator on a compact set $\mathbb{S}$, it is continuous if and only if it is bounded. Therefore there exists some constant $c \geq \Psi^*$ on the domain $\mathbb{S}$. Similarly for the parabolic case the solution $\Psi^*(x,t)$ has an upper bound for each point in time t, and integration of these finite quantities over a bounded time period also produces bounded solutions.

A candidate solution $\Psi(x,t)$ may be taken to be the plane with $\Psi(x,t) := b$ for b a constant greater than all boundary conditions, i.e. $b \geq e^{-\frac{\phi_T(x)}{\lambda}}$ for all $x \in \partial\mathbb{S}$. Plugging in this solution into the optimization (2.7) it is seen to satisfy the constraint set by construction. As this is a polynomial of degree zero, it is in the set of feasible

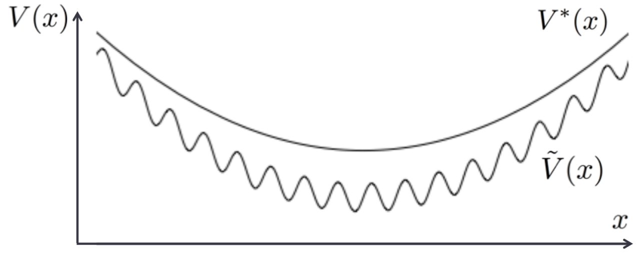


Figure 2.1: Illustration of potential misalignment between the optimal and approximate value function, $V(x)$ and $\tilde{V}(x)$, gradients despite their proximity. The x-axis here denotes state space domain, while the y-axis denotes the cost-to-go at a particular state.

solutions to (2.7). Since this is a convex problem, the existence of a feasible solution $p(x,t)$ is sufficient for the algorithm to converge. $\square$

Intuitively, the previous result states that there must exist constant values that upper and lower bound the solution to the desirability, which are of course polynomial representable. Clearly such bounds may be quite poor in practice. However, placing this problem within a hierarchy of optimization problems, namely in a Lasserre Hierarchy [46], with increasing polynomial degree yields the following result.

Proposition 16. *Let Ψ_n be a polynomial approximation of the desirability function with maximum polynomial degree n that is a solution to* (2.7) *(or* (2.9)*). The hierarchy of SOS problems consisting of solutions to* (2.7) *(or* (2.7)*) with increasing polynomial degree, or increasing degree of Positivstellensatz multipliers in* (2.3)*, produce a sequence of solutions $\{\Psi_i, \gamma_i\}_i$ with $\gamma_i \geq \gamma_{i+1}$.*

Proof. Given a solution Ψ_n to (2.7), and an additional solution Ψ_{n+1} of higher degree, each with solutions γ_n, γ_{n+1} respectively, $\gamma_{n+1} \leq \gamma_n$ as Ψ_{n+1} may achieve error γ_n by setting its additional degrees of freedom to zero, so the solution sequence γ_n is non-increasing. Clearly, $\gamma \geq 0$, indicating the sequence has a limit. $\square$

Note that no guarantee are available as to the divergence of the cost when executing the approximate value function from the true value function. The only guarantee provided is that the value function is an over-approximation at a particular state. A consequence is illustrated in Figure 2.1. By following the gradient, the system may diverge significantly from the optimal path, further undermining the accuracy of the approximate value function. This is an issue common to many approximate dynamic programming schemes [95, 76, 96]. A commonly employed technique is to simply use Monte Carlo simulation of the policy resulting from the approximate value solution, providing an upper bound J^{ub} on the realizable cost. If the resulting sampled upper bound J^{ub} is near this lower bound, then the policy may be said to be empirically near-optimal.

2.1.4 Examples

A scalar and a two-dimensional pair of examples reveal the computational characteristics of the method. In the following problems the optimization parser YALMIP [97] was used in conjunction with the semidefinite optimization package SDPT3 [98].

2.1.4.1 Scalar System Example

A nonlinear, unstable system with the following dynamics is considered

$$dx = \left(x^3 + 5x^2 + x + u\right) dt + d\omega \tag{2.11}$$

on the domain $x \in \mathbb{S} = [-1, 1]$. The problem chosen is a first-exit problem, with $\phi(-1) = 10$, and $\phi(1) = 0$. For this instance, $\mathcal{L} = 1$, $G = 1$, $B = 1$, and the cost parameters $q = 1$, $R = 1$ are assigned. Optimal solutions to (2.7) of the desirability for varying polynomial degree $deg(\Psi)$ are shown in Figure 2.2 along with its transformed cost-to-go. The pointwise error in the desirability for increasing polynomial degree on the solution and the *Positivstellensat* multipliers (of (2.3) is shown in Table 2.1.4.1.

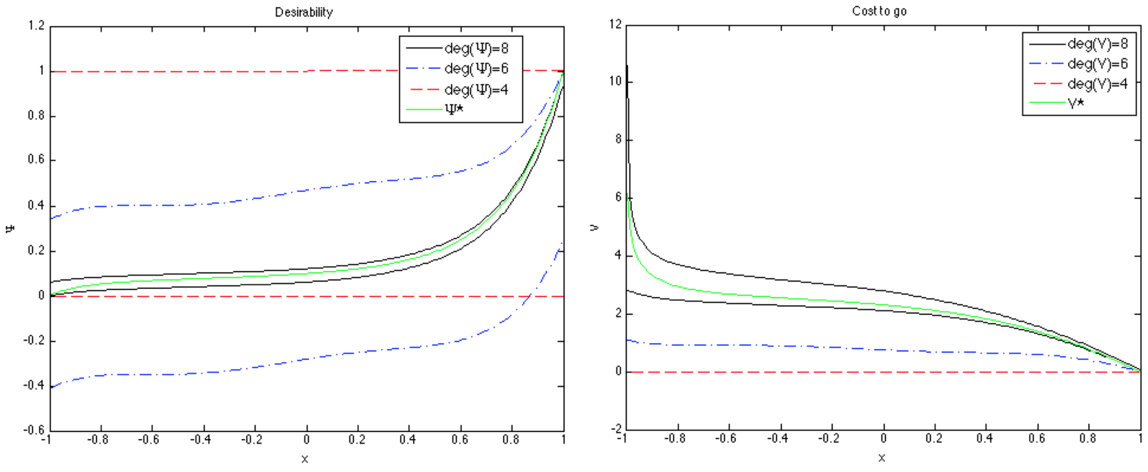


Figure 2.2: Plots of approximate and exact desirability and cost-to-go solutions for the scalar system (2.11) versus state x in the interval $x \in [-1, 1]$. The dashed red, dashed blue, and solid black lines represent the $deg(\Psi) = 4$, $deg(\Psi) = 6$, and $deg(\Psi) = 8$ approximations respectively. The *Positivstellensatz* polynomial multipliers in (2.3) were set to have matching degree, i.e. $deg(s_i) = deg(\Psi)$.

$deg(\Psi) \setminus deg(s_i)$	2	4	6	8	10
2	1.0	1.0	1.0	1.0	0.9994
4	1.0	1.0	1.0	0.9999	0.9947
6	1.0	1.0	0.7508	0.7498	0.7406
8	1.0	1.0	0.2834	0.0592	0.0592
10	1.0	1.0	0.2834	0.0590	0.0487

Table 2.1: Solution quality γ of the desirability lower bound for the scalar system (2.11) with varying polynomial degree of solution Ψ and Positivestellensatz multipliers s_i of (2.3).

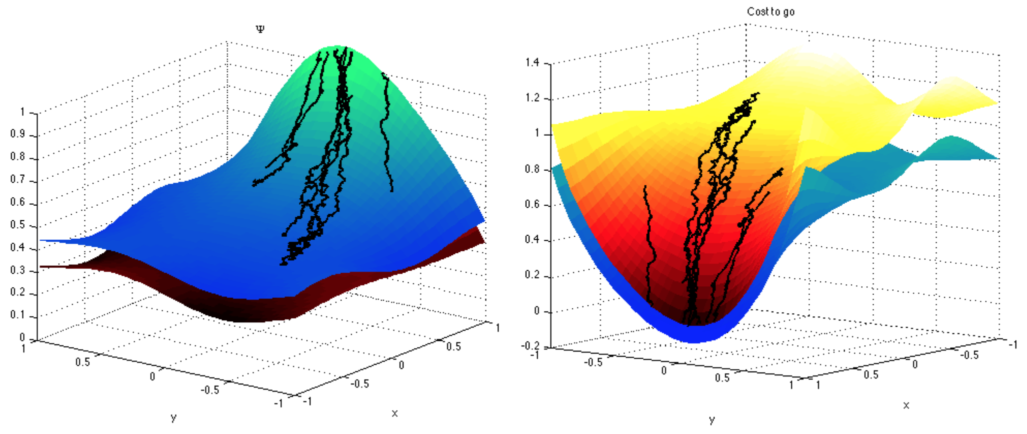


Figure 2.3: Approximate Desirability and Value solutions for the two dimensional example (2.12). The problem is solved with $deg(\Psi) = deg(s_i) = 14$. The upper bound had gap $\gamma_{up} = 0.0979$, and lower bound $\gamma_{lw} = 0.1049$. Ten simulated trajectories of the closed loop system, randomly sampled from $x, y \in [-.75, .75]^2$, are shown in black.

2.1.4.2 Two Dimensional Example

Next, a nonlinear 2-dimensional problem example adapted from [99] was solved as a first-exit problem. The dynamics are set as

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \left(\begin{bmatrix} -2x - x^3 - 5y - y^3 \\ 6x + x^3 - 3y - y^3 \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) dt + \begin{bmatrix} d\omega_1 \\ d\omega_2 \end{bmatrix} \tag{2.12}$$

The system was given the task of reaching a boundary of the domain $\mathbb{S} = [-1, 1]^2$, and once there would fulfill its task with no additional cost. The control penalty was set to $R = I_{2\times 2}$, and state cost as $q(x) = 0.1$. The boundary conditions for the sides $x = -1, y = 1, y = -1$ were set to have a penalty of $\phi(x, y) = 1$, while for the remaining boundary $x = 1$ the boundary has quadratic cost $\phi(x, y) = 1 - (y - 1)^2$. The results are shown in Figure 2.8.

Discussion

Sum of squares and semidefinite programming have been used to construct a global solution without recourse to value iteration or other forms of dynamic programming. The method produces a-priori bounds on the solutions' pointwise error from the optimal Hamilton Jacobi Bellman solution. Unfortunately, a priori error bounds on the cost of the trajectories resulting from policies which follow the approximate solution cannot be directly obtained. Such bounds are the subject of further investigation. As it stands, there is no guarantee that a specific objective will be obtained, e.g., to reach a goal region or provide stabilization. Indeed, the mis-alignment of true and approximate value functions has surfaced in the controls community [43] as well as in the broader literature on approximate dynamic programming [76].

The algorithms presented in this thesis differ from the simple process of applying approximate dynamic programming with polynomial basis functions. Key in this work is the development in the continuous state space of the problem. Although approximate dynamic programming can use a polynomial basis for the value function similar to that in this work, it nonetheless begins from a discrete state space. The result is that the number of constraints in the corresponding dynamic program depends on the size of the discrete state space [76]. While in practice many of these constraints may be inactive, it isn't possible to determine a-priori the inactive ones. Furthermore, as has been shown, the SOS framework gives strong guarantees on the pointwise distance between the approximate and exact value functions. However, in contrast with approximate dynamic programming, in the work presented in this thesis, the solution to an Semidefinite Programming problem is required, as compared to that of an Linear Program.

As mentioned, this method is proposed as an alternative to sampling based methods that utilize the Feynman-Kac lemma. A distinct advantage of the Feynman-Kac based approach is that the required sampling scales well with increasing dimension of the state space. It is an interesting question as to how the method proposed here can be extended to high dimensional state spaces. A method to do so follows in Section 6.2.

2.2 Domain Partitioning

Building on the Sum of Squares-based algorithm of Section 2.1, this section proposes an extension in which the domain is split into distinct partitions, each of which has its own local approximating polynomial. The value function may vary significantly over the domain, and thus may require an impractically high degree polynomial if approximated over the domain's entirety. By using a sufficiently local approximation, the same quality of approximation may be achieved with a smaller degree polynomial. An efficient choice of partitioning, presented subsequently, may lead to a decoupling in the optimal control problems on each partition, allowing for an almost unlimited degree of parallelization. The Alternating Direction Method of Multipliers (ADMM) [100, 101] provides a principled method for parallelization of convex problems. It is adopted here to provide general guarantees on the convergence of the design process. Other decomposition schemes are possible, see [102, 103] for a survey, although these alternatives are not investigated here.

Domain partitioning has long been used in traditional numerical methods for PDEs, from the local analysis behind the Finite Element Method to multiscale decomposition techniques [104]. In control, these techniques have also been used to improve local approximation to Lyapunov functions [105], and are complimentary to approaches that approximate nonlinear systems as piecewise-affine (PWA) [106]. This thesis extends these techniques not only to the study of stability, as is the case for Lyapunov functions, but to control design as well. Furthermore, the ability to generate solutions to Hamilton Jacobi Bellman equations has implications in regards to Control Lyapunov Functions [42], allowing for stabilization to be shown alongside near-optimality. The method has the distinct advantage over PWA approximations in that the system itself is not approximated in the approach, and the full nonlinear dynamics are incorporated into the solution.

2.2.1 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) [100] will serve as the basis for enforcing continuity and differentiability of the desirability function $\Psi(x)$ of Eq. (1.27) on the boundaries of the decomposed regions. ADMM is a "meta"-optimization scheme, where each step is carried out by solving a convex optimization problem. Consider the optimization

$$\begin{array}{ll} \text{minimize}_{x,z} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array} \tag{2.13}$$

over real vector variables x and z and convex functions f and g. Define an augmented Lagrangian

$$L_\rho = f(x) + g(z) + y^T \left(Ax + Bz - c\right) + \frac{\rho}{2} \left\|Ax + Bz - c\right\|_2^2, \tag{2.14}$$

where scalar $\rho > 0$ is an algorithm parameter, and y is the dual variable associated with the equality constraint. The constrained optimization is solved through alternately minimizing the augmented Lagrangian over the primal variables x, z, and updating the dual variable y,

$$\begin{aligned} x^{k+1} &:= \text{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \text{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right). \end{aligned} \tag{2.15}$$

The sum of squares formalism allows a general polynomial optimization problem to be converted to a sequence of SDPs, as detailed in Section 2.1, where the variables are the polynomial coefficients. ADMM extends readily to SDPs. Consider

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \\ & x \in \mathcal{C}_1, \quad z \in \mathcal{C}_2, \end{array}$$

where $x, z \in \mathbb{R}^n$ are the variables and $\mathcal{C}_1, \mathcal{C}_2$ are Semidefinite-representable sets, i.e., feasible sets of Linear Matrix Inequalities. With the same form of L_ρ as in (2.14), the ADMM iterations are quadratically penalized SDPs,

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_{x \in \mathcal{C}_1} L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \operatorname{argmin}_{z \in \mathcal{C}_2} L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right). \end{aligned} \tag{2.16}$$

The only difference is the primal variables are now constrained to lie in the spectrahedra, i.e., the convex set of semidefinite constraints [107], $\mathcal{C}_1$ and $\mathcal{C}_2$.

The value in this decomposition lies in the convergence guarantees which can be obtained with ADMM. In particular, if it can be demonstrated the proposed domain decomposition technique obeys the following two assumptions:

Assumption 17. *The (extended real valued) functions $f : \mathbb{R}^n \to \mathbb{R} \cup +\infty$ and $g : \mathbb{R}^m \to \mathbb{R} \cup +\infty$ are closed, proper, and convex.*

Assumption 18. *The unaugmented Lagrangian has a saddle point.*

then the following theorem holds:

Theorem 19. *(See [100]) Given Assumptions 17, 18 then the ADMM iterates* (2.16) *satisfy the following:*

- ***Residual convergence**: $r^k \to 0$ as $k \to \infty$, i.e., the iterates approach feasibility.*
- ***Objective convergence**: $f(x^k) + g(z^k) \to p^*$ as $k \to \infty$, i.e., the objective function of the iterates approaches the optimal value.*
- ***Dual variable convergence**: $y^k \to y^*$ as $k \to \infty$, where y^* is a dual optimal point.*

where the residual is defined as $r^k := Ax^k + Bz^k - c$, and the optimal objective value $p^* = \inf\{f(x) + g(z) \mid Ax + Bz = c\}$.

2.2.2 Decomposition of Stochastic Optimal Control

As the optimal control problem is assumed to take place over a compact state space, the domain of (1.27) may partitioned into finitely many non-overlapping regions $\mathcal{R}_j \subseteq \mathbb{R}^n$, $j = 1, \ldots, n_R$, where $\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_{n_R} = \mathbb{S}$. For example, the regions $\mathcal{R}_j$ might be adjacent squares or hypercubes. Assuming the pairwise boundary between the regions may be described in terms of a semialgebraic set, a straightforward consequence of the Positivstellensatz (see [99] for details) is the following result

Theorem 20. *Given desirability function $\Psi_i(x)$ valid on region $\mathcal{R}_i$, $\Psi_j(x)$ valid on region $\mathcal{R}_j$, and shared boundary $\xi = \{x \mid h(x) = 0\}$ between $\mathcal{R}_i$ and $\mathcal{R}_j$, $\Psi_i(x) = \Psi_j(x)$ on ξ if there exists c(x) such that*

$$\Psi_i(x) - \Psi_j(x) + c(x)h(x) = 0 \tag{2.17}$$

In the following analysis, this result is used to bind together optimization problems over a decomposed domain. The combined policy will be required to be $\mathcal{C}^1$ continuous, requiring equality constraints on the solution and its gradient over shared boundaries. Of course, Theorem 20 can also be used to enforce $\mathcal{C}^n$ continuity for finite n by taking successive derivatives of Ψ_i normal to each boundary $h(x)$. A discussion of continuity considerations is given subsequently in Section 2.2.4.

Fix a pair of bordering partitions Ψ_1 and Ψ_2, with shared boundary $h(x)$=0. The polynomials approximating the desirability function are assumed to be of bounded degrees, with $\Psi_i(x)$ bounded by d and $c_i(x)$ by $d-k$, for all i, j. In this case,

$$\begin{aligned}
\Psi_1(x) &= \alpha_0 + \alpha_1 x + \cdots + \alpha_d x^d \\
\Psi_2(x) &= \beta_0 + \beta_1 x + \cdots + \beta_d x^d \\
c_1(x) &= \theta_0 + \theta_1 x + \cdots + \theta_{d-k} x^{d-k} \\
c_2(x) &= \mu_0 + \mu_1 x + \cdots + \mu_{d-k} x^{d-k},
\end{aligned}$$

where $h(x) = \rho_0 + \rho_1 x + \cdots + \rho_k x^k$ defines the shared boundary between partitions

Ψ_1 and Ψ_2 for some set of coefficients ρ_i. The continuity constraint

$$\Psi_1(x) - \Psi_2(x) + c_1(x)h(x) = 0$$

is equivalent to the coefficient matching constraints

$$\begin{aligned}
0 &= \alpha_0 - \beta_0 + (\theta_0\rho_0) \\
0 &= \alpha_1 - \beta_1 + (\theta_0\rho_1 + \theta_1\rho_0) \\
0 &= \alpha_2 - \beta_2 + (\theta_0\rho_2 + \theta_1\rho_1 + \theta_2\rho_0) \\
&\vdots \\
0 &= \alpha_d - \beta_d + (\theta_{d-k}\rho_k).
\end{aligned}$$

Note that the coefficient matching constraints are affine in the decision variables α_i, β_i, $i = 1, \ldots, d$, and θ_j, μ_j, $j = 1, \ldots, d-k$. The derivative constraint at the boundary appends additional coefficient matching constraints,

$$\begin{aligned}
0 &= \alpha_1 - \beta_1 + (\mu_0\rho_0) \\
0 &= 2\alpha_2 - 2\beta_2 + (\mu_0\rho_1 + \mu_1\rho_0) \\
0 &= 3\alpha_2 - 3\beta_2 + (\mu_0\rho_2 + \mu_1\rho_1 + \mu_2\rho_0) \\
&\vdots \\
0 &= d\alpha_d - d\beta_d + (\mu_{d-k}\rho_k).
\end{aligned}$$

The continuity and derivative coefficient matching constraints, together with the approximation error constraint (2.25), can be aggregated into matrix form,

$$A^{(1)}z_1 + A^{(2)}z_2 = 0,$$

where $z_1 = (\alpha_0, \ldots, \theta_{d-k}, \gamma_1)$ are the coefficients associated with $\mathcal{R}_1$, and $z_2 = (\beta_0, \ldots, \mu_{d-k}, \gamma_2)$ are the coefficients associated with $\mathcal{R}_2$. It is now straightforward to incorporate the affine matrix constraint into a dual decomposition scheme. The decomposed variant of optimization (2.7) is

$$\begin{aligned}
&\text{min. } \gamma_1 && (2.18)\\
&\text{s.t. } \frac{1}{\lambda} q\Psi_1 \geq \partial_t \Psi_1 + L(\Psi_1), \quad x \in \mathcal{R}_1 && (2.19)\\
&\frac{1}{\lambda} q\Psi_2 \geq \partial_t \Psi_2 + L(\Psi_2), \quad x \in \mathcal{R}_2 && (2.20)\\
&\gamma_1 - \left(\frac{1}{\lambda} q\Psi_1 - L(\Psi_1) - \partial_t \Psi_1\right) \geq 0, \quad x \in \mathcal{R}_1 && (2.21)\\
&\gamma_2 - \left(\frac{1}{\lambda} q\Psi_2 - L(\Psi_2) - \partial_t \Psi_2\right) \geq 0, \quad x \in \mathcal{R}_2 && (2.22)\\
&\Psi_1(x) - \Psi_2(x) + c_1(x)h(x) = 0 && (2.23)\\
&\frac{\partial \Psi_1}{\partial x}(x) - \frac{\partial \Psi_2}{\partial x}(x) + c_2(x)h(x) = 0 && (2.24)\\
&\gamma_1 = \gamma_2 && (2.25)
\end{aligned}$$

where the Positivstellensatz is used to enforce the domain restrictions (see Section 2.1.2 and [2] for details). The coupling constraints (2.23) and (2.24) prevent decomposition into two parallel optimizations. In addition, the objective is coupled through the equality constraint (2.25), which ensures that the maximum pointwise approximation error over any region is no more than $\gamma = \gamma_1 = \gamma_2$.

To wit, define the quadratically penalized Lagrangian

$$\begin{aligned}
L_\rho(\gamma_1, z_1, \gamma_2, z_2, \lambda) = \gamma_1 + \gamma_2 + \mathcal{I}_{\mathcal{C}_1}(z_1) + \mathcal{I}_{\mathcal{C}_2}(z_2) + \\
+ \lambda^T (A^{(1)} z_1 + A^{(2)} z_2) + \frac{\rho}{2} \left\| A^{(1)} z_1 + A^{(2)} z_2 \right\|_2^2,
\end{aligned}$$

where $\mathcal{I}_{\mathcal{C}_i}(z_i)$ is the indicator function of the optimization problem over each individual partition

$$\mathcal{I}_{\mathcal{C}}(x) = \begin{cases} 0 & x \in \mathcal{C} \\ \infty & x \notin \mathcal{C}. \end{cases} \tag{2.26}$$

The convex sets $\mathcal{C}_i$ are obtained by reduction of (2.7) to semidefinite program form

[36]. The alternating direction iteration may then be performed as

$$(\gamma_1^{k+1}, z_1^{k+1}) := \arg\min_{\gamma_1, z_1} L_\rho(\gamma_1, z_1, \gamma_2^k, z_2^k, \lambda^k) \tag{2.27}$$

$$(\gamma_2^{k+1}, z_2^{k+1}) := \arg\min_{\gamma_2, z_2} L_\rho(\gamma_1^{k+1}, z_1^{k+1}, \gamma_2, z_2, \lambda^k) \tag{2.28}$$

$$\lambda^{k+1} := \lambda^k + \rho(A^{(1)} z_1^{k+1} + A^{(2)} z_2^{k+1}). \tag{2.29}$$

Each minimization, a semidefinite program, is taken over only those constraints associated with the specified region. This achieves a degree of decoupling, limiting the size of the polynomial optimization problem, and thus the semidefinite program, for each individual partition. Indeed, even the use of domain partitioning on its own is insufficient to make these problems tractable. If alternating directions are not taken, the optimization problem over all partitions, with all low order polynomials, would necessarily be solved simultaneously. Although there exists an extreme degree of sparsity, specialized solvers, such as [47], would be required and standard ones such as SDPT3 [98] will fail.

2.2.3 Parallelization

A further decoupling may be achieved through a judicious choice of domain partitions. This will allow for necessary computations to be parallelized. This idea is well known in the partial differential equation community [104]. Suppose partition $\mathcal{R}_i$ and $\mathcal{R}_j$ share no common border $h_{i,j}(x)$. As the variables from disjoint partitions are only shared through the common boundary constraints (2.23), (2.24), it is straightforward to see that z_i^{k+1} and z_j^{k+1} are independent of one another. This decomposition allows for the optimization (2.27) over each region $\mathcal{R}_i$ to be performed in parallel with no affect on the performance of the ADMM algorithm.

In particular, one valid partition, developed by way of example, is to decompose the domain into a checkerboard pattern, separating the domain into white and black tiles. As white tiles share no optimization variables with one another, they may be optimized in parallel, and similar with the black. By alternating between white and

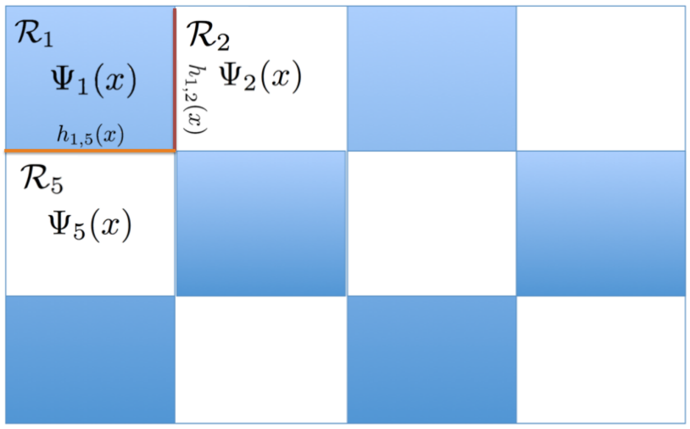


Figure 2.4: Example of a particular grid domain decomposition, the checkerboard pattern, with the partitions grouped into white and black sets. As the sets of the same color require no consensus over their local variables, it is possible to perform the optimization over each set in parallel while maintaining the convergence properties of ADMM.

black, the alternate directions continue to be taken and, as will be shown in Section 2.2.5, guaranteeing convergence. See [108] for a detailed discussion of parallelization ideas, and Fig. 2.4 for an illustration of the decomposition pattern that is examined in particular, the checkerboard pattern.

The domain is therefore partitioned into hypercubes $\mathcal{R}_k$. Divide the partitions into two regions, *white* and *black*, as $\{\mathcal{R}_k\}_{k=1,\ldots,n_R} = \{\mathcal{R}_i^w\}_{i=1,\ldots,\mathcal{W}} \cup \{\mathcal{R}_j^b\}_{b=1,\ldots,\mathcal{B}}$ such that $\mathcal{R}_i^w \cap \mathcal{R}_j^w = \emptyset$ and $\mathcal{R}_i^b \cap \mathcal{R}_j^b = \emptyset$, which is possible by the construction detailed above.

The optimization problem that results is

$$\begin{aligned} x_i^{k+1} &:= \operatorname{argmin}_{x\in\mathcal{C}_1} L_\rho(x, z_{\bar{i}}^k, y^k), && i = 1,\ldots,\mathcal{W} \\ z_j^{k+1} &:= \operatorname{argmin}_{z\in\mathcal{C}_2} L_\rho(x_{\bar{j}}^{k+1}, z, y^k), && j = 1,\ldots,\mathcal{B} \\ y^{k+1} &:= y^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right). \end{aligned} \tag{2.30}$$

where the regions that are white are labeled as x_i while those that are black are z_j, and the set of neighbors to a region i are denoted as $\bar{i}$.

The following property is trivial by inspection of the constructed problem

Proposition 21. *The set of optimization variables x_i, x_j in* (2.30)*, each representing the optimization* (2.7) *over region $\mathcal{R}_i$, $\mathcal{R}_j$ share no common optimization variables.*

It is seen that the optimization of all x_i and all z_i may be done in parallel, as they share no optimization variables. The result is that if the number of regions in the partition set is M, then up to $\frac{M}{2}$ of these optimizations may be computed simultaneously.

2.2.4 Decomposition Issues

When solving the Hamilton Jacobi Bellman equation, continuously differentiable desirability functions $\Psi(x)$ are required. This imposes not only a continuity constraint, but also a derivative constraint on each boundary between two adjacent decomposed regions. The solution behavior when these constraints are relaxed is now investigated.

Consider the setting of Fig. 2.5. A degree five polynomial (gray) in two variables is approximated over the box $[-1, 1]^2$, with regions given by the four quadrants

$$\begin{aligned}
\mathcal{R}_1 &= \{(x, y) \mid 0 \leq x \leq 1,\ 0 \leq y \leq 1\} \\
\mathcal{R}_2 &= \{(x, y) \mid -1 \leq x \leq 0,\ 0 \leq y \leq 1\} \\
\mathcal{R}_3 &= \{(x, y) \mid -1 \leq x \leq 0,\ -1 \leq y \leq 0\} \\
\mathcal{R}_4 &= \{(x, y) \mid 0 \leq x \leq 1,\ -1 \leq y \leq 0\}.
\end{aligned}$$

Several points merit discussion. First, by giving up continuity along the region boundaries, it is possible to approximate the polynomial with a lower degree polynomial, as evidenced by smaller approximation error γ in the left column.

Next, depending upon relative degree and problem dimension, simply enforcing continuity along the boundaries may be enough to enforce differentiability, as evidenced by the smoothness of the solutions in the right column. This bodes well for reducing the problem size by throwing away superfluous differentiability constraints, a possibility that hints at future research.

Finally, an increase in approximation degree, when combined with continuity and/or differentiability along the region boundaries does not necessarily result in a better approximation accuracy, as evidenced by a lack of improvement from $d = 1$ to $d = 2$ and from $d = 3$ to $d = 4$ in the right column.

All of these points are relevant in employing problem structure to decrease the computational burden associated with the optimization in each region. Careful consideration of the degree bounds may result in smaller SDPs with fewer variables.

2.2.5 Decomposition Analysis

The following results demonstrate that the domain decomposition presented in this work is well motivated theoretically.

Theorem 22. *The domain partitioned sum of squares ADMM program iterates for two regions* (2.27) *satisfy the **Residual convergence**, **Objective convergence**, and **Dual variable convergence** properties of Theorem 19.*

Proof. The problem for two regions is the following

$$\begin{aligned} min \quad & f_1(x) + f_2(z) \\ s.t. \quad & Ax + Bz = c \\ & x \in \mathcal{C}_1 \\ & z \in \mathcal{C}_2 \end{aligned}$$

where $\mathcal{C}_1, \mathcal{C}_2$ are the spectrahedra generated by the SOS constraints for each partition of the decomposed domain. As these sets are semidefinite representable [36], they are convex. These partitions are included in the objective as

$$\begin{aligned} min \quad & f_1(x) + f_2(z) + \mathcal{I}_{\mathcal{C}_1}(x) + \mathcal{I}_{\mathcal{C}_2}(z) \\ s.t. \quad & Ax + Bz = c. \end{aligned}$$

New functions $h_i(x) = f_i(x) + \mathcal{I}_{\mathcal{C}_i}(x)$ may be defined to obtain exactly the ADMM form (2.13). It is clear that h_i are closed, proper, and convex, satisfying Assumption 17.

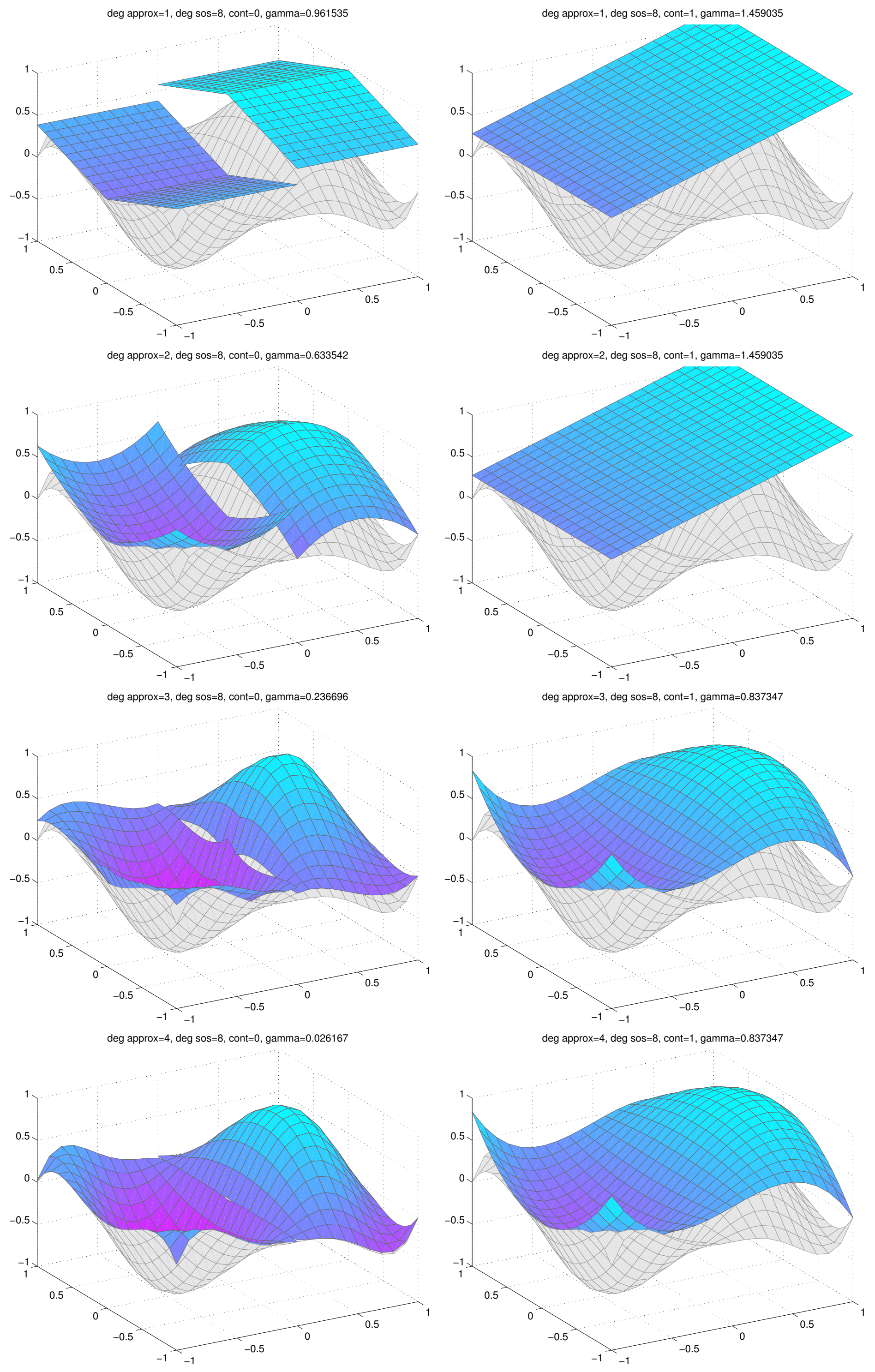


Figure 2.5: Approximations of a degree five polynomial $f(x, y) = 2x^5 + x^3(3y^2 - 5) + x(3 - 4y^2 + y^4)$ over four quadrants of the box $[-1, 1]^2$ without (left column) and with (right column) enforced continuity at the quadrant boundaries. Each row imposes a different degree bound d on the approximating polynomial and gives the approximation error. The degree of the Positivestellensatz multipliers are denoted as sos, and the optimization gap as gamma.

Further, because the optimization problem is convex to begin with, the augmented Lagrangian has a saddle point, satisfying Assumption 18. By the general ADMM convergence Theorem 19, the desired convergence properties are obtained. □

Note that the above theorem holds true only when the domain decomposition has taken place between a total of two regions. It has been shown that the naive extension of ADMM to the minimization of more than two separated functions faces more severe restrictions for convergence to be guaranteed [109, 110]. Instead of relying on these more restrictive results, a decomposition of the domain into only two regions is used.

Corollary 23. *The domain partitioned sum of squares ADMM program iterates for the system* (2.27) *with domain partitioned according to the checkerboard pattern (see Figure 2.4) satisfy the* ***Residual convergence****,* ***Objective convergence****, and* ***Dual variable convergence*** *properties of Theorem 19.*

Proof. This is a simple result of Theorem 22 since the checkerboard pattern simply splits the domain into two regions. The lack of connectivity between these partitioned regions play no role with respect to the theorem. □

As pointed out in the parallelization discussion, each individual optimization for each partition is completely disjoint from all others. Thus, even though each local solution is calculated independently, they generate the same result as if they were solved simultaneously as a common optimization problem. Thus, the solution is identical to the two-way partitioned ADMM problem of Corollary 23, with the associated convergence results inherited.

In addition to the parallelization of computation, the desirable properties of the sum of squares solutions developed in Section 2.1 are maintained. A benefit of the sum of squares-relaxation approach is that the solutions produced are guaranteed to be upper and lower bounds (depending on the direction of the relaxations (2.5)) when performed over a single partition [2].

Theorem 24. *Given a solution set $\{\Psi_i, \gamma_i\}$, $i = 1, \dots, n_R$ to the converged optimization problem* (2.27) *with $\mathcal{C}^2$ continuity enforced along the partition boundaries, and if*

Ψ^* *is the solution to* (1.27), *then* $\Psi(x) \geq \Psi^*(x)$ *for all* $x \in \mathcal{R}_i$.

Proof. The derivation follows the proof of Theorem 11 in Section 2.1 with but one modification. The only modification arises from the fact that the elliptic and parabolic maximum principles rely on $\mathcal{C}^2$ continuity of the super-solution. As the solution is polynomial on the interior of each boundary, and therefore infinite differentiable, this requirement needs only be enforced explicitly along the partition boundaries. □

Once again the inequalities of the optimization can be reversed to produce a lower bound to the optimal solution as well. See [2] for details.

2.2.6 Scalar Example

In the following examples, the SDP optimization on each region was carried out using YALMIP with its Sum of Squares module [111] and SDPT3 for the interior point solver [98].

The optimization is constructed for a simple scalar example for illustrative purposes. Consider the one dimensional system

$$dx = (x^2 + u)\, dt + d\omega \tag{2.31}$$

on the domain $x \in [-1, 1]$ with state cost $q(x) = 1$, control cost $R = 1$, and parameter $\lambda = 1$. The domain is split into regions $\mathcal{R}_1 = \{x \mid x \in [-1, 0]\}$, $\mathcal{R}_2 = \{x \mid x \in [0, 1]\}$, creating $h(x) = x$. For each of these problems the optimization (2.7) is formed on $\mathcal{R}_1$, $\mathcal{R}_2$ independently. To enforce equality of both the solution and its derivative at the shared point $x = 0$, the coupling constraints are

$$\Psi_1(x) - \Psi_2(x) + c_1(x)x = 0 \tag{2.32}$$

$$\frac{\partial \Psi_1}{\partial x}(x) - \frac{\partial \Psi_2}{\partial x}(x) + c_2(x)x = 0, \tag{2.33}$$

To enforce the continuity and derivative constraints (2.32), (2.33) for the point boundary at the origin, it suffices to match the constant coefficients of Ψ_1 and Ψ_2, i.e., re-

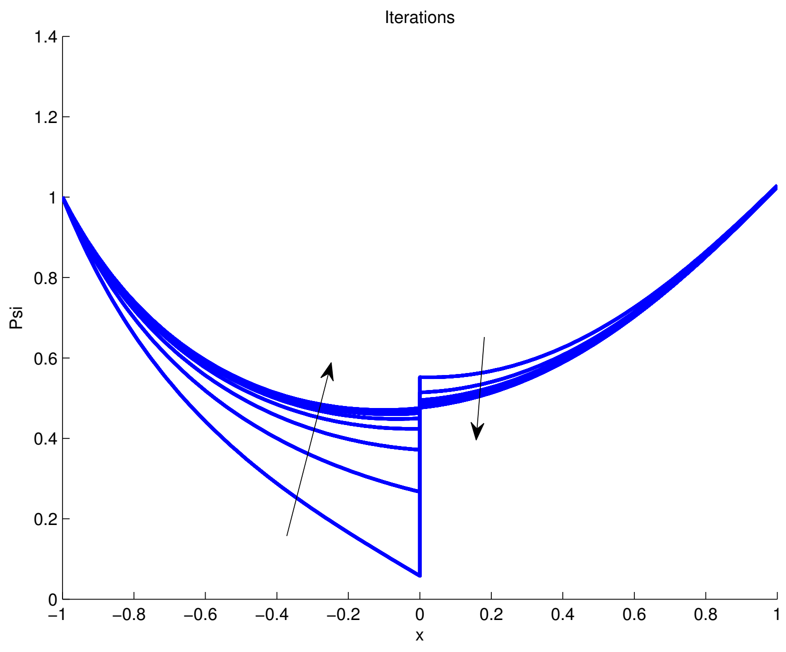


Figure 2.6: Evolution of the alternative value function over 10 alternating direction steps for the scalar system (2.31). Arrows show direction of evolution with each step of ADMM.

quire $\Psi_1(0) = \Psi_2(0)$. These are affine constraints when the polynomial optimization is passed to an SDP.

Numerical results for the one dimensional example are shown in Fig. 2.6 and Fig. 2.7. For simplicity, the conditioning parameter was set to $\rho = 1$, and the polynomial degree bound to $deg(\Psi_i) = 6$ for each region. Fig. 2.6 shows that within about ten iterations of the ADMM procedures, continuous differentiability at the boundary region $x = 0$ is achieved. Fig. 2.7 shows the evolution of the dual variables, as well as the maximum approximation gap versus iteration number.

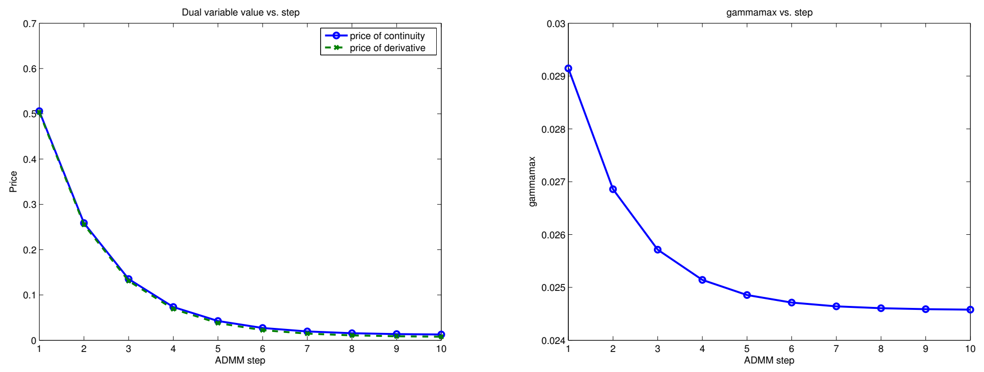


Figure 2.7: Values of the dual variables (top) and maximum approximation gap (bottom) with iteration number for the scalar system (2.31).

2.2.7 Two Dimensional System

To demonstrate the versatility of the method, a nonlinear, multidimensional problem, the same as from Section 2.1.4.2 was solved with the following dynamics.

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \left(\begin{bmatrix} -2x - x^3 - 5y - y^3 \\ 6x + x^3 - 3y - y^3 \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) dt + \begin{bmatrix} d\omega_1 \\ d\omega_2 \end{bmatrix}. \tag{2.34}$$

The problem is framed as a first exit problem, with the three sides of a square domain $\mathbb{S} = [-1, 1^2]$ given a unit penalty $\phi(x, y) = 1$, while on the remaining edge at $x = 1$ a reward was given for achieving the center of the edge with $\phi(x, y) = 1-(y-1)^2$. The results of applying ADMM are shown in Figure 2.8, where $deg(\Psi) = 8$ and achieves a gap of $\gamma = 0.6321$. Note that an eighth order polynomial was entirely insufficient to model the solution to this problem when only a single domain was used. Similar results to those demonstrated in the single domain example of Section 2.1.4.2 only became possible with polynomials of $deg(\Psi) \geq 12$.

2.2.8 Discussion

A method to perform domain decomposition on stochastic optimal control problems has been developed, allowing for local polynomial approximations to the Hamilton Jacobi Bellman equation to be generated in parallel. Of importance is the fact that the

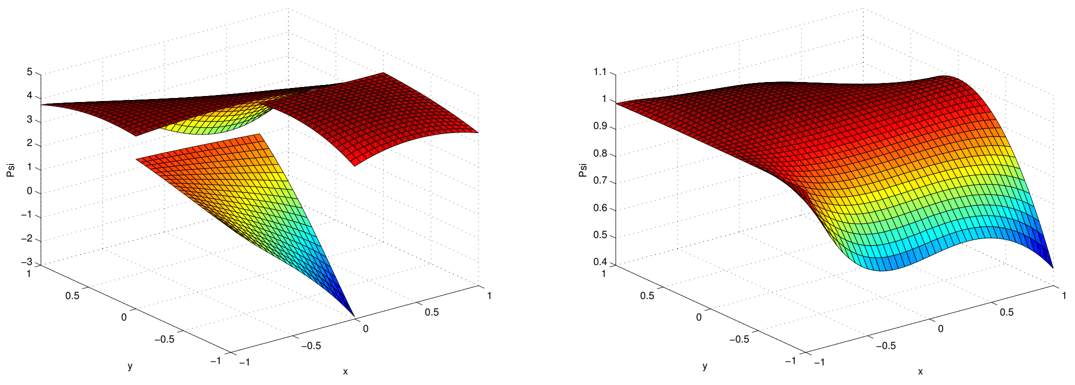


Figure 2.8: Results of multidimensional, nonlinear example system (2.34). On the left is the optimization after two iterations of ADMM, while the one on the right is the converged results, when using $deg(\Psi) = deg(s_i) = 8$, where s_i are the Positivstellensatz multiplers. At convergence the upper bound has distance $\gamma = 0.6321$.

Sum of Squares relaxation does not fundamentally rely on the particular structure of the Hamilton Jacobi Bellman PDE. In fact, [2] demonstrates that the technique may be readily applied to any linear parabolic or elliptic PDE to obtain guaranteed upper and lower bounds over the domain. The domain splitting introduced in this chapter can be extended, allowing for local upper and lower bounds to broader classes of linear PDEs to be generated via optimization. While having different characteristics and computational burden than existing numerical techniques such as the Finite Element method, these techniques have guarantees that do not require an asymptotic limit in the discretization mesh.

A more direct implication lies in the generation of stabilizing controllers for nonlinear systems. Until now, there has not existed a method to generate near-optimal Control Lyapunov Functions for arbitrary nonlinear, stochastic systems [42]. These domain decomposition techniques improve the ability for optimal control policies to respond to system dynamics, enlarging the class of systems that can be handled.

2.3 Uncertainty Quantification of Partial Differential Systems

In the study of control systems and the Hamilton Jacobi Bellman equation, a partial differential equation-centric perspective has been adopted. The key result of the previous two sections, Theorem 24, borrowed from the partial differential equation literature, with the proof using the elliptic and parabolic maximum principle to guarantee upper and lower bounds to these solutions. This result is in fact not limited to the Hamilton Jacobi Bellman equation, and applies to all elliptic and parabolic equations. This suggests a novel method of solving for sub- and super-solutions to these broad classes of partial differential equations.

The discipline of Uncertainty Quantification (UQ) [112] is concerned with a similar issue, the generation of optimal bounds on system behavior given a certain set of assumptions. With the techniques developed in this thesis, it is possible to bound the solution to a system governed by differential equations. As will be seen, the method can be further generalized to incorporate additional assumptions, including uncertainty on parameter values. Of course, as the Hamilton Jacobi Bellman equation is but one class of these systems, all the methods and generalizations developed here apply to that domain as well. The method also has ready application in filtering equations, such as the Fokker-Planck [113] and Zakai equations [114].

The study of partial differential equations with uncertainty, both with respect to stochastic forcing and parametric uncertainty, has been studied under the guise of Stochastic Partial Differential Equations (SPDE). Such problems are prevalent in manufacturing, construction, finance, remote sensing, and geographic exploration [115]. There are two forms of uncertainty in these problems. In the first, a stochastic forcing term may be present. A simple example is the stochastic heat equation

$$\partial_t u = \Delta u + \xi$$

where ξ is space-time white noise and Δ is the Laplacian. These problems are typically considered by discretizing the white noise and sampling from the problem space [115]. This class of problems is not considered here.

The second, popular class of SPDEs are those where the coefficients in the PDE are random and may lie in some uncertainty set. Prior solutions to such problems have used either Monte Carlo-based techniques [116], or those based on Polynomial Chaos [117]. In the former, realizations may be sampled, solutions calculated, and distributions over quantities of interest estimated in a natural way. In the latter, an orthonormal polynomial basis is used to represent the distribution of the random coefficient. The PDE system is then projected onto this basis, creating a linear system of equations that may be solved. This method has recently been the focus of research effort, but faces the obstacle that the number of polynomial basis functions grows exponentially with dimensionality and the number of random coefficients, giving rise to the curse of dimensionality once again.

The Uncertainty Quantification problem differs from traditional studies of partial differential equations in that it is not the exact solution of any realization of the problem that is sought, but instead the focus is on the characteristics of the solution set. In this spirit, this section develops a novel method to bound the feasible solutions to a PDE given known uncertainty sets for each unknown coefficient.

2.3.1 Sum of Squares-Based Solution Bounds

The technique builds upon the approach of Section 2.1. Given a partial differential operator L, the task is to find pointwise upper and lower bounds to the solution $u(x)$ over a compact domain Ω such that $u(x)$ satisfies the PDE

$$Lu(x) = f(x), \quad x \in \Omega \tag{2.35}$$

$$\tilde{L}u(x) = g(x), \quad x \in \partial\Omega \tag{2.36}$$

where (2.36) denotes the boundary conditions of the problem. The assumption is made that all problem data $L, \tilde{L}, g, f, \partial\Omega$ are encoded as polynomial functions of the domain $x = (x_1, \ldots, x_d) \in \Omega \subset \mathbb{R}^d$.

As explored in Section 2.1, since the set of polynomials of degree m are closed under differentiation, a polynomial optimization problem may be formed by relaxing (2.35) and (2.36) to the inequalitiies

$$\begin{aligned} Lu(x) &\geq f(x), \quad x \in \Omega && (2.37)\\ \tilde{L}u(x) &\geq g(x), \quad x \in \partial\Omega. && (2.38)\end{aligned}$$

Once again, in all optimization problems that follow, the inequalities may be reversed to produce complementary upper bounds. The relaxation leads to the optimization

$$\begin{aligned} \min \quad & \gamma && (2.39)\\ s.t. \quad & f(x) \leq L(\Psi), \qquad x \in \mathbb{S} \\ & L(\Psi) - f(x) \leq \gamma, \quad x \in \mathbb{S} \end{aligned}$$

where the boundary conditions are similar and are suppressed. By Theorem 24, the resulting solution is an upper or lower bound to the PDE, with the two bounds approaching one another as the degrees of the polynomials are increased.

Suppose that the operator L has a set of free parameters $a = (a_1, \ldots, a_k)$, where each of these parameters are known to have support limited to some set $a_i \in \mathcal{A}_i$ where $\mathcal{A}_i$ is compact. This yields the augmented optimization problem

$$\begin{aligned} \min \quad & \gamma && (2.40)\\ s.t. \quad & f(x,a) \leq L(\Psi,a), \qquad x \in \mathbb{S} \\ & L(\Psi,a) - f(x,a) \leq \gamma, \quad x \in \mathbb{S} \\ & a_i \in \mathcal{A}_i \end{aligned}$$

The addition of the variables a_i increases the complexity of the optimization problem as these additions increase the size of the monomial basis. Each a_i is treated as an additional domain variable, in exactly the same manner as the state x. The limitation of the support for each a_i is handled via the Positivstellensatz (see Section 2.1.1) in the same manner as the domain restriction on x.

Although only a minor conceptual addition to the framework, this line of reasoning opens up the possibility of rapidly generating guaranteed upper and lower bounds to a wide class of uncertain problems. Although the Positivstellensatz has been used previously in robust optimization [89], which is the name given to optimization problems with uncertainty over the problem data (as is the case here), its use in optimizing candidate solutions to linear partial differential equations is novel. As the method is valid for even low degree polynomials, loose bounds are readily computable, with increasing accuracy obtained when additional computational resources are harnessed.

Of note, this approach lies in the broader set of tools presented in this thesis. The techniques developed subsequently to increase the accuracy and tractability of the Hamilton Jacobi Bellman equation are also applicable to these UQ problems. For instance, methods to numerically solve problems of high dimensions developed in Section 3.4 are readily applicable. This opens the possibility of bounding solution sets to even high dimensional PDE problems.

2.3.2 Moment-Based Bounds

The problem of bounding the solutions to linear PDEs via optimization techniques has been considered before by Bertsimas and Caramanis [118]. In their work, they proposed to integrate the partial differential equation against a set of monomial test functions, and by encoding the resulting constraints on the coefficients of these test functions were able to generate semidefinite relaxations that solved the PDE as the number of moments approached infinity. Fundamental to this work is the ability to truncate the moment list, creating upper and lower bounds on functions of the moments, which in turn were defined over the entire domain and not any finite discretization of the domain. This related work is reviewed before demonstrating its connection to the sum of squares relaxation proposed above. Given partial differential operators L and G the goal is to calculate

$$\int Gu(x)dx$$

on some domain Ω. The solution u must satisfy the PDE given by

$$Lu(x) = f(x).$$

The constraints of the PDE are enforced by integrating against test functions chosen as monomials

$$\int Lu(x)x^\alpha dx = \int f(x)x^\alpha dx. \tag{2.41}$$

In order to frame the problem only in terms of generalized moments of $u(x)$, the adjoint is taken

$$\int u(x)L^* x^\alpha dx = \int f(x)x^\alpha dx \tag{2.42}$$

through the use of integration by parts. The adjoint of L^* may therefore be calculated a-priori and applied to a given monomial x^α. The result is a set of linear constraints over variables which have the form

$$m_\alpha = \int_\Omega x^\alpha u(x)dx = \int_\Omega x_1^{i_1} \cdots x_d^{i_d} u(x)dx_1 \dots dx_d. \tag{2.43}$$

These moment variables must be constrained such that the moment sequence $\mathbf{M} = \{m_\alpha\}$ is in fact a valid moment sequence. This may be guaranteed as follows: given a sequence of numbers $\{m_i\}$, this set of constraints is the set of moments of some nonnegative function $u(x)$ if and only if the matrix

$$M_{2n} = \begin{pmatrix} m_0 & m_1 & \cdots & m_n \\ m_1 & m_2 & & m_{n+1} \\ \vdots & & \ddots & \vdots \\ m_n & & \cdots & m_{2n} \end{pmatrix} \tag{2.44}$$

is positive semidefinite for every n. A relaxation may be obtained by simply truncating this requirement to fixed size n. For a vector of moment variables z, i.e., $z_i = \int_{-\infty}^{+\infty} u(x)dx$, denote the corresponding moment matrix of (2.44) $M(z)$.

The geometry of the domain Ω is accounted for through the use of the Positivstellensatz. This creates an additional semidefinite constraint on the moment variables for every semialgebraic segment of the domain boundary. The details are given in [118] and are known as *localizing* constraints. These constraints are denoted by $M_\ell^i(z)$ for the constraints generated by boundary segment i.

The result is a set of constraints on the moments of the solution to the PDE. An objective may then be formed in terms of these moments, yielding for appropriate polynomial G,

$$\begin{aligned} \max/\min \quad & \int Gu(x)dx \\ \text{s.t.} \quad & \int u(x)L^* x^\alpha dx = \int f(x)x^\alpha dx \\ & h_i(M) \succeq 0 \end{aligned} \tag{2.45}$$

where $\Omega = \{x \mid h_i(x) \geq 0\}$ describes the domain of the PDE. The result of this optimization is either a lower or upper bound on the objective. By incorporating higher order monomial test functions, these two bounds on the objective may be shown to converge [118]. This allows for moment data of the solution to a PDE to be collected without solving the PDE itself.

There are several connections between the work of this thesis and the prior work of [118]. It is first demonstrated that the moment-based approach is related to the current work by examining the dual to the optimization problem proposed in Section 2.1. In addition, two modest extensions of this framework are proposed, in which non-polynomial test functions are used to improve the accuracy of the result locally, and also examine the use of domain decomposition in this context. These methods are designed with the requirements of Stochastic Optimal Control as a particular focus. Namely, such systems typically only inhabit a limited fraction of the state space for any significant amount of time. Thus, only point- or region-specific solutions are desired.

2.3.2.1 Duality between Moment and Sum of Squares

In the following analysis, it is shown that these two approaches to generating bounds to linear PDEs, each of which began from distinct perspectives, are in fact related via convex duality to one another. Each approach has its merits individually, but as many convex solvers simultaneously solve the primal and dual problems, it is therefore worthwhile that the connection be elucidated. In particular for the methods presented here, it is shown that each method is not the dual of the other due to varying objectives, but the methods do share equivalent constraint sets.

This result on the equivalence of the constraints between the two optimization problems follows from existing work in [119] demonstrating the sum of squares and moment duality for optimal control problems. Define the moments $m_\alpha = \int x^\alpha d\omega$, and the vector of moment variables $m = (m_1, \ldots, m_r)$ for maximal degree r of the moments considered. The optimization problem in (2.45) then becomes

$$\begin{aligned} \min \quad & G_M m \\ s.t. \quad & L_M m + B_M = f_M m \\ & M(m) \succeq 0 \\ & M_\ell^i(m) \succeq 0 \end{aligned} \tag{2.46}$$

where M denotes the moment matrix mapping, and M_ℓ denotes the localization matrix mapping, L_M and f_M are the linear coefficient matrices garnered from the partial differential equations, B_M is a vector arising due to boundary conditions during the integration by parts needed to obtain the adjoint, and G_M is the coefficient vector from the objective. The SDP dual of (2.46) is

$$\begin{aligned} \max_{c,S,S_j} \quad & (B_M)^T c \\ s.t. \quad & (f_M)^T c - (L_M)^T c + M^*(S) + \sum M_\ell^i (S_j) = G_M \end{aligned} \tag{2.47}$$

where S is the dual variable for the moment matrix constraint, S_j are those for the localizing matrix constraints, and * denotes the adjoint. If the mapping $\varphi(x) = c \cdot m(x)$ is defined, then by the derivation of the SOS dual derivation in [119], (2.47) is equivalent to the problem,

$$\begin{aligned} \max_{\varphi,s,s_j} \quad & B(\varphi) \\ s.t. \quad & f - L(\varphi) + G(\varphi) = s + \sum_{j=1} s_j g_j \end{aligned} \tag{2.48}$$

where the dependence of each function on the domain variables x has been suppressed. $B(\varphi)$ can be seen to encode the boundary data of the approximate solution represented by φ. The constraint of (2.48) is quickly seen to be exactly the Positivstellensatz enforcement of the compact domain restriction. However, unlike the other optimization (2.39), the objective is no longer a maximization pointwise over the domain. Therefore, the two problems are equivalent in their construction of the partial differential constraints, i.e., what becomes the constraints of the optimization problem, but differ only in their objectives. This gives a straightforward method to derive duals of either problem, the moment data or pointwise error variant. The two methods are therefore related, but as framed in [118] not quite the same.

2.3.2.2 Non-Polynomial Test Functions

In seeking data about an individual point in the solution of the Hamilton Jacobi Bellman equation, note that

$$\int \delta(x, x_0) u(x) dx = u(x_0) \tag{2.49}$$

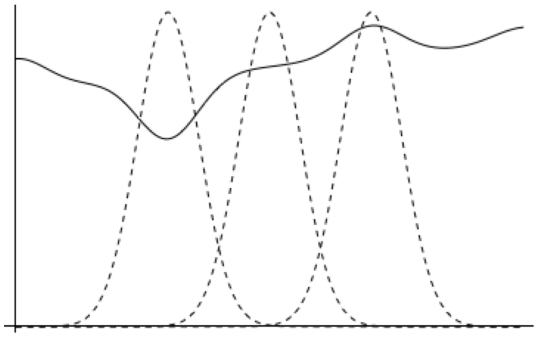


Figure 2.9: Illustration of convolution of the unknown solution with Gaussian moments.

By integrating this Dirac function against the solution, it would be possible to obtain the exact solution at a point. Unfortunately, such an integral is unavailable, as the Dirac function is of course not polynomial representable or differentiable. Instead, the Dirac function is approximated through the use of an exponential. The results of [118] are extended to integrals of exponentials to accommodate this approach.

The goal is to obtain the solution $u(x)$, and its gradient $du(x)$, satisfying (1.27) at a particular point x_0. One approach would be to approximate the Dirac function δ with a polynomial $h(x, x_0)$.

$$\int Gu(x) = \int h(x, x_0)u(x) \approx \delta(x, x_0)u(x) = u(x_0). \tag{2.50}$$

Unfortunately, it is quite hard to approximate a delta function well, as even high order polynomials are poor approximations. An alternative is to use Gaussians kernels as the Dirac approximation

$$u(x_0) = \int u(x)\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-x_0)^2}{2\sigma^2}}dx, \quad \sigma \to 0 \tag{2.51}$$

The Gaussian approximation has several properties that make it appropriate for this application. First, it is uniformly positive, and thus can lead to its analogue of moment matrices. Further, the space of polynomial-exponentials

$$\Delta = \left\{e^{f(x)}, xe^{f(x)}, x^2e^{f(x)}, \ldots\right\} \tag{2.52}$$

Algorithm 1 Initialization for Localized Method of Moments.

```
Given domain $\Omega$, region of interest $\omega$, and adjoint operator $L^*$:
  1. Partition $\Omega$ into a set $\{R_i\}_{i \in \mathcal{I}}$ such that $\omega$ is preserved, i.e., $\omega = R_i$ for some $i$
  2. For each $R_i \neq \omega$
     (a) For each monomial test function $\phi_j$, $1 \leq j \leq n$
         i. Generate the $j^{th}$ equality constraint. Label each unknown boundary
            condition as $b_i^k$
                $$\int u\,(L^* \phi_i) = \int f \phi_i$$
         ii. Generate the semidefinite moment constraint $M \succeq 0$
         iii. For each boundary $h_l$ of partition $R_i$
              A. Generate the boundary moment constraint $h_l(M) \succeq 0$
              B. If there exists boundaries $h_l = h_k$, shared with region $R_j$, set
                 $b_i^k = b_j^l$
  3. Set object as max/min $b_i^k$ for some $i, k$.
```

is closed under differentiation. This is a necessity as the adjoint operator (2.42) will result in the differentiation of the test functions. This allows us to maintain the framework, and substitute variables for the exponential moments.

The method's outline is to construct a series of Gaussians with varying mean over the domain of the PDE with minimal variance. In the limit of zero variance, the exact solution would be obtained, but numerical issues arise quickly. By "sampling" the solution at various locations, it is instead possible to obtain the solution to the problem convolved with the Gaussian. It is then possible to perform a deconvolution operation to obtain an estimate of the original solution. Additional sample points may improve the deconvolved solution, creating a tradeoff between computational time and quality of the bound.

2.3.2.3 Domain Decomposition for Explicit Interior Boundary Variables

The second approach is to partition the domain of the problem in order that the solutions along boundaries of the partition become explicit as optimization variables.

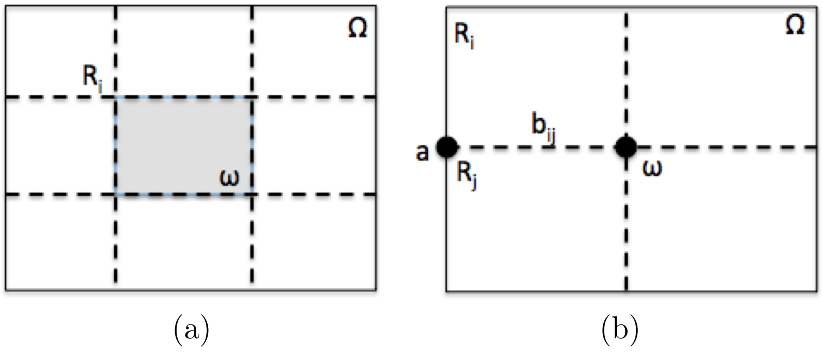


Figure 2.10: Example decomposition of planar problem domain with the resulting solution regions appearing explicitly as optimization variables. (a) On the left, the optimization problem (2.45) may be solved over the region ω with only the boundary data on Ω given. (b) On the right, the point solution ω or the interior boundaries b_{ij} appear explicitly as optimization variables and may also be bounded.

The result is that solutions along these boundary variables may be bounded. These boundaries may be structured to be any lower dimensional surface in the domain, be it a point or curve in two dimensions, or a plane in three dimensions, etc. Furthermore, due to the general construction of [118], the method may also be applied to the solutions gradient. This allows for the regional solution estimates to be obtained without the need to solve the PDE globally. Finally, the inclusion of additional boundary conditions and moments both improve the solution, but only a minimal number of these conditions are required.

The method is first outlined on a simple two dimensional example. Given a square domain $\Omega = [0, a]^2$, suppose the goal is to bound the PDE solution on an inner square $\omega = [x_1, x_2] \times [y_1, y_2]$. The domain may be partitioned into nine regions, one of which is ω. Over each partition, the construction of (2.45) is performed, but there is now coupling between the shared boundaries of these domains. Unknown are the moment data within each partition, as well as these shared boundary conditions. Bounds on any of these variables, or any convex function thereof, may then be obtained. The partition is illustrated in Figure 2.10a.

Additionally, it is possible to use integration by parts to not only obtain the boundaries of these domains, but also lower dimensional components of the boundaries. For instance, a partition of the example in Figure 2.10a consists of line segments, and the point boundaries of these segments may be made as explicit optimization variables. Consider the two dimensional square and examine the partition boundary lying on the y-axis. Along this boundary, the integral is

$$\int_a^b u(x, y = 0)x^i dx = uy^{i+1} \mid_a^b - \int \frac{du}{dx} y^{i+1}$$

so the variable $\alpha = u(b)b_y^{i+1}$ appears explicitly. This is illustrated in Figure 2.10b. More generally, the approach is outlined in the steps of Algorithm 1.

2.4 Discussion

In this chapter, analytic and computational tools have been developed for solving the linear Hamilton Jacobi Bellman equation. Key was the notion of a relaxed solution to the PDE, and it was demonstrated that the relaxation of a PDE in this context to a partial differential inequality in fact produced pointwise upper and lower bounds to the true solution. The linearity has proved to be crucial to these techniques, as it allows for candidate solutions to appear linearly in the relevant optimization problems. It was also shown how these techniques could be applied to broader classes of elliptic and parabolic PDEs. A domain decomposition technique, with the capability for parallelization, improves both the scalability and computational difficulty of the method. Finally, a discussion of the convex dual to the polynomial optimization problem was found to correspond to moment based methods in the literature, and several variants of these techniques were proposed.

Chapter 3

High Dimensional Optimal Control

The Hamilton Jacobi Bellman Equation provides the globally optimal solution to large classes of control problems. Unfortunately, this generality comes at a price, the calculation of such solutions is typically intractible for systems with more than moderate state space size (five or six are typical in current practice) due to the *curse of dimensionality*. This chapter combines recent results in the structure of the Hamilton Jacobi Bellman equation, and its reduction to a linear partial Differential Equation, introduced in Section 1.4, with methods based on low rank tensor representations, known as a separated representations, to address the curse of dimensionality. The result is an algorithm to solve optimal control problems which scales linearly with the number of states in a system assuming the existence of a particular form of internal structure for the solution, namely low *separation rank*. Key to this work is the notion that problem data, and the resulting solution, may have an underlying *low-rank* structure. The method is applicable to systems that are nonlinear with stochastic forcing in finite-horizon, average cost, and first-exit settings. The method is demonstrated on inverted pendulum, VTOL aircraft, and quadcopter models, with system dimension two, six, and twelve respectively.

The method relies on recent work in Separated Representations (SR) [120], which have recently emerged as a method to solve a number of problems in machine learning and the numerical solution of PDEs with complexity that scales *linearly* with dimension, bypassing the curse of dimensionality. The central idea of this paper is to approximate the solution, and its associated operators, by a low rank tensor. If the problem's components can be adequately modeled in this regime, then the complexity grows with the rank of the approximation, rather than the dimensionality. For many problems of interest this proves to be a valid modeling assumption.

As discussed in Section 1.2.4, researchers have previously attacked the intractability of the Hamilton Jacobi Bellman equation through discretization into a MDP. The curse of dimensionality is mitigated in this context by parameterizing the value function with a sparse set of bases, giving rise to Approximate Dynamic Programming, or Adaptive Dynamic Programming (ADP) when the basis may change online [9]. These techniques have constraint sets that formally grow exponentially with dimensionality [76]. Furthermore, examples of the basis functions chosen, such as radial basis functions, typically fall prey to the curse of dimensionality. Nonetheless, these techniques are the most popular method to deal with the curse of dimensionality and have even been used to surpass human capabilities on complex time dependent games via synthesis with modern machine learning techniques [78]. These methods are closest to ours in spirit, and the method developed in this chapter could be seen as generating a sparse basis, as is desired in ADP, albeit ours is performed without recourse to an MDP, with the attendant constraints.

3.1 High Dimensional Tensor Background

Tensor approximations have historically been developed with the goal of approximating high dimensional data, yielding rise to the framework used here and previously called CANDECOMP/PARAFAC (C/P) [121, 122]. However, Beylkin & Mohlenkamp in [120] demonstrated that these approximation techniques were applicable to the linear systems describing discretized PDEs as well. This C/P technique has been applied in several domains, including computational chemistry and quantum physics, among others [123]. In particular, [124] examines the use of C/P in the context of stationary Fokker-Planck equations. There are interesting connections between the fundamental goal of these techniques, approximating a tensor with one of lower rank, and convex relaxation based methods [125, 126]. Unfortunately, low rank tensor approximation is NP-hard in general, and an optimal solution cannot be expected [127]. Nonetheless, suboptimal solutions appear to be excellent in practice.

3.2 Separated Representations of Tensors

Traditional numerical techniques to solve PDEs rely on a complete discretization of the problem domain [128]. However, in these schemes the degrees of freedom in the problem grows exponentially with the number of dimensions, as the complexity is proportional to the discretization process. While tractable when the number of dimensions is small, in higher dimensions these problems become computationally prohibitive. In [120], Beylkin and Mohlenkamp proposed to model the solutions to such problems via so-called separated representations, which may be viewed as an adaptation of the separation of variables technique. Problem data, and the solution, are modeled as a sum of terms, each of which is dependent on individual dimensional variables. Specifically, a function operating on a space $\Omega = (x_1, \ldots, x_d)$ is approximately modeled as

$$f(x_1, \ldots, x_d) \approx \sum_{l=1}^{r} s_l \phi_1^l(x_1) \cdots \phi_d^l(x_d). \tag{3.1}$$

The key is that such a representation separates the dependence of the solution into each state-space component dimension. By then framing operations to act on single dimensions, it is possible to create algorithms that need only operate along each dimension independently and thus scale linearly with dimension d. However, the complexity of the problem now grows with r, termed the *separation rank*. Thus, maintaining a low separation rank becomes paramount for any practical algorithm. Unfortunately, many operations needed to compute a solution inherently increase the separation rank, including vector addition and matrix-vector multiplication.

This unbounded growth in separation rank can be mitigated by reducing the separation rank at each step of an algorithm in an attempt to continually maintain low rank approximations. Unfortunately, there are often no guarantees that a given function, or solution to a PDE, will have low separation rank and situations may arise where it is impossible to lower the rank while maintaining a desired accuracy.

An introduction to the separated representation follows, with a complete treatment given in [120]. Using the notation in [120] a vector F in dimension d is a discrete representation of a function f on a rectangular domain, $\boldsymbol{F} = F(j_1, \ldots, j_d)$ where $j_i = 1, \ldots, M_i$ are the indices into the vector $\boldsymbol{F}$ and M_i is the size of the vector in dimension i. A linear operator $\boldsymbol{\mathcal{A}}$ in dimension d is a linear map $\boldsymbol{\mathcal{A}} : S \to S$ where S is the space of functions in dimension d. A matrix $\mathbb{A}$ in dimension d is a discrete representation of a linear operator in dimension d.

Definition 25. *For a given $\epsilon > 0$, represent a vector $\boldsymbol{F} = F(j_1, j_2, \ldots, j_d)$ in dimension d as*

$$\boldsymbol{F} \approx \sum_{l=1}^{r_{\boldsymbol{F}}} s_l \bigotimes_{i=1}^{d} \boldsymbol{F}_i^l \tag{3.2}$$

where $\bigotimes$ denotes the tensor product and $\boldsymbol{F}_i^l$ are (traditional) vectors in $\mathbb{R}^{M_i}$ with entries $F_i^l(j_i)$ and unit norm. For this to be an ϵ-accurate representation it is required that

$$\left\| \boldsymbol{F} - \sum_{l=1}^{r_{\boldsymbol{F}}} s_l \bigotimes_{i=1}^{d} \boldsymbol{F}_i^l \right\| \leq \epsilon. \tag{3.3}$$

The integer r is known as the ***separation rank****.*

The matrix definition is analogous, with the matrices $\mathbb{A}_i^l \in \mathbb{R}^{M_i \times M_i}$ in lieu of $\boldsymbol{F}_i^l = F_i^l(j_i)$. Matrix multiplication is then performed as

$$\mathbb{A}F = \sum_{m=1}^{r_{\mathbb{A}}} \sum_{l=1}^{r_F} s_m^{\mathbb{A}} s_l^F \left(\mathbb{A}_1^m F_1^l\right) \otimes \cdots \otimes \mathbb{A}_d^m F_d^l.$$

Since matrix operations in this formulation reduce to individual operations along each dimension, as the dimensionality of the problem increases the complexity of these operations scales linearly, *e.g.*, if $M_i = M$ for all i a matrix vector multiplication costs $O(r_A r_F d M^2)$. Assuming that a low separation rank may be maintained, iterative methods may provide the best option for solving systems in this framework or computing quantities of interest such as the largest eigenvector. A number of such schemes are given in [120].

3.3 Alternating Least Squares

As discussed in the previous subsection, and as demonstrated by the computational cost of the matrix vector multiplication, any scheme that uses these separated representations will become computationally prohibitive if the separation ranks are allowed to grow too much. For example, in the operation of matrix vector multiplication, the separation rank of the output is $r_A r_F$, so even performing the most basic of operations may have a large impact on the separation rank, and in an iterative method where, say, each iteration requires a matrix vector multiplication, the growing separation rank would quickly make the problem intractable. Therefore, an algorithm is required that allows for a reduction in separation rank. If the assumption is that the discrete versions of the functions being represented have low separation rank, then any increase in the separation rank may be an artifact of the way that operations are performed in these tensor representations and not indicative of a fundamental change in the underlying separation rank. Therefore, it is expected that after performing an operation that increases the separation rank, it is possible to produce an accurate representation of the resultant tensor that has much lower separation rank.

A high level overview of the alternating least squares (ALS) algorithm, as used to achieve a reduction in separation rank, is provided, with the reader directed to [120] for details. A recently proposed variant relying on a randomized interpolative decomposition has also been proposed in [129] and may be used as a precursor to ALS.

As the separation rank grows after each operation, for tractability it is necessary to periodically approximate intermediate tensors with low-rank approximations, *i.e.*, given a separated representation $\mathbf{F}$, find an approximate representation $\mathbf{G}$ with a smaller separation rank than $\mathbf{F}$ such that $||\mathbf{F} - \mathbf{G}||$ is minimized. If the separation rank of $\mathbf{G}$ is fixed, then this is a nonlinear least-squares problem. ALS attempts to solve such a problem, though there are often few theoretical guarantees on the resulting $\mathbf{G}$ when using this algorithm.

At an individual step in this iterative algorithm, all dimensions of the tensor $\mathbf{G}$ are held constant save one dimension k, in which case the least-squares problem becomes linear in $\mathbf{G}_k^l$ for $l = 1, \ldots, r_G$ and may thus be solved. This is done by forming the normal equations relative to those degree of freedom in the dimension being optimized, with the construction of these normal equations available in [120].

The algorithm sweeps through the coordinate directions, effectively performing block-gradient descent. For a fixed separation rank of $\mathbf{G}$ this process may be repeated until the algorithm has either achieved the desired accuracy, or has stagnated. If the algorithm has stagnated, and the representation error is not small enough, *e.g.*, $\|\mathbf{G} - \mathbf{F}\| \geq \epsilon$, a random rank-one tensor is added to $\mathbf{G}$, and the algorithm is allowed to continue until it either achieves the desired accuracy or stagnates once again.

This algorithm continues until the desirable tolerance is reached or adding a random vector would result in separation rank equal to the starting rank, in which case it is assumed that the separation rank cannot be reduced. This algorithm may be used on operators as well by simply vectorizing each component matrix.

The procedure described above may also be be applied to construct a low rank solution to a linear system of equations by minimizing $\|\mathbb{A}\mathbf{F} - \mathbf{G}\|$. The resulting normal equations for block-coordinate descent become increasingly coupled, see, *e.g.*, [120, 124] for details, raising the complexity of the algorithm. While the core ALS algorithm costs $\mathcal{O}\left(dM + dr_{\mathbf{F}}^3\right)$ per iteration, its use to solve a linear system costs $\mathcal{O}\left(dM^3 + r_{\mathbb{A}}^3 M^3\right)$ per iteration, where d is the underlying dimensionality of the system, M is the maximal number of mesh nodes along each dimension, and $r_{\mathbb{A}}, (r_{\mathbf{F}})$ is the rank of the operator $\mathbb{A}$ (vector $\mathbf{F}$). See [120] for a more comprehensive list of algorithms that may be used with operators and vectors in separated representations.

3.4 Separated Solution to the HJB

The modeling assumption is made that the problem data of Equation (1.18) can be accurately represented, or approximated, with a low rank separated representation.

$$f_i(t,x) = \sum_{l=1}^{r_{f_i}} \bigotimes_{k=1}^{d} (f_i)_d^l \tag{3.4}$$

where r_{f_i} is assumed to be small.

There is then the need to approximate the relevant operators present in (1.18), specifically the gradient and Hessian, in a low rank representation. A number of options exist, with varying levels of complexity in the analysis and accuracy, ranging from simple finite difference schemes to spectral differentiation techniques [130]. Specifically, the gradient along dimension k can be simply represented as

$$\nabla_k = I_1 \otimes \cdots \otimes \nabla \otimes \cdots \otimes I_d$$

while the Hessian has entries $\nabla_{k,j} = \nabla_k \cdot \nabla_j$, and the estimates of the derivative along an individual coordinate are simply a suitably high order finite difference scheme in one dimension. For instance, the first and second order central finite difference matrices are given by the tri-diagonal matrices

$$\nabla_x = \frac{1}{2h}\begin{bmatrix} \ddots & & & \\ & -1 \quad 0 \quad 1 & \\ & & & \ddots \end{bmatrix}, \quad \nabla_{xx} = \frac{1}{h^2}\begin{bmatrix} \ddots & & & \\ & 1 \quad -2 \quad 1 & \\ & & & \ddots \end{bmatrix}$$

Thus, the directional gradient and second order terms may simply be constructed out of rank one representations. For example, using of sums of these rank one terms yields a representation for the Laplacian that has separation rank d. However, such a representation may be not have minimal separation rank for a given accuracy. Other constructions specifically targeting the separated representation exist [120], for example a Laplacian approximation may be made with separation rank two, rather than requiring a full rank-d sum of second order terms.

3.4.1 Separation Rank of the HJB

Determining the separation rank of the Hamilton Jacobi Bellman operator is straightforward. Denote the separation rank of a vector or operator X as r_X, i.e., r_X are the number of additive terms in X. Recalling (1.27) and neglecting the time dependent component for the first-exit case, the operator consists of three additive terms.

$$\frac{1}{\lambda} q\Psi = f^T \left(\nabla_x \Psi\right) + \frac{1}{2} Tr \left(\left(\nabla_{xx} \Psi\right) \Sigma_t\right).$$

The state-cost term $q\Psi$ is a diagonal operator along each dimension, and thus contributes r_q. The second, advection term is an inner product between the dynamics f and the gradient of the desirability, resulting in the multiplication of each element f_i by a rank one operator, and then their summation. The contribution from this component results in separation rank $\sum_{k=1}^{d} r_{f_i}$ where r_{f_i} is the separation rank of f_i. Finally, the second-order term requires the construction of Σ_t in (1.25). Here the growth in the separation rank may be significant, due to the multiplicative contribution of G. However, given diagonal cost matrix R or noise covariance Σ_ϵ the number of terms may collapse significantly. The separation rank of the Hamilton Jacobi Bellman operator is simply the sum of these three terms' rank.

The result is that the separation rank for individual problems may vary over a wide range, depending on the problem data. However, in many problems of interest it remains low. For even apparently complex systems, complexity typically manifests as nonlinear multiplicative terms in the dynamics. This form of complexity, specifically the presence of nonlinearities, effectively adds no cost in terms of separation rank, and it is instead the number of additive terms that are of concern, which is typically small. Furthermore, in many applications the control or noise matrix Σ_t typically contain constant terms, corresponding to tensors of separation rank one. Finally, for systems where a high separation rank accumulates, it remains possible to search for low rank structure by performing ALS on the operator before attempting to solve the linear system.

Algorithm 2 Assignment of boundary conditions in an operator $\mathbb{A}$ with separated representation for hypercube boundary $\mathcal{R}$.

```
Inputs: Operator $\mathbb{A}$, hypercube $\mathcal{R} = \{x \in \mathbb{R}^d \mid x_i \in [a_i, b_i]\}$, boundary value tensor $T$, grid points $\{x_{i,j}\}$ representing the domain of $\mathbb{A}$.
Output: Modified operator $\bar{\mathbb{A}}$.

  1. Define $\tilde{\mathbb{A}} := \mathbb{A}$
  2. Let $\tilde{\mathbb{I}}$ be the identity operator tensor
  3. For $k = 1 \dots d$
      (a) Set $\tilde{\mathbb{A}}^l_k(i,j) = 0$ for all $x_{i,j} \notin [a_i, b_i]$
      (b) Set $\tilde{\mathbb{I}}^l_k(i,j) = 0$ for all $x_{i,j} \notin [a_i, b_i]$
  4. Set $\bar{\mathbb{A}} = \mathbb{A} - \tilde{\mathbb{A}} + \tilde{\mathbb{I}}$
```

3.4.2 Representation of Interior Boundary Conditions

Optimal control applications impose irregular boundary conditions on many problems of interest. For example, stabilization to the origin corresponds to a zero-cost point-boundary at the origin. Obstacles or unsafe regions are boundary conditions as well, and typically have value according to some penalty. In temporal problems, dynamic programming can be used to show that the cost of achieving sub-goals in the future relative to the current task manifest as boundary conditions along exit points of the current task. The value along these boundaries equals to the cost-to-go [131], computed from value functions in subsequent tasks, i.e., a set of Dirichlet boundary conditions within the domain.

Essential boundary conditions are imposed by setting the value of grid points to some desired value via linear equalities within the domain. Although in other settings it is desirable to remove the degrees of freedom from within the boundaries to save computational effort, in the context of this work maintaining the symmetry of the discretization grid is a far greater concern. Specifically, Dirichlet boundary conditions are imposed only on regions composed of hypercubes in the domain, allowing us to modify the domain with only a modest increase in the separation rank of the operator. The operator's initial effect on the hypercube is first extracted, then subtracted,

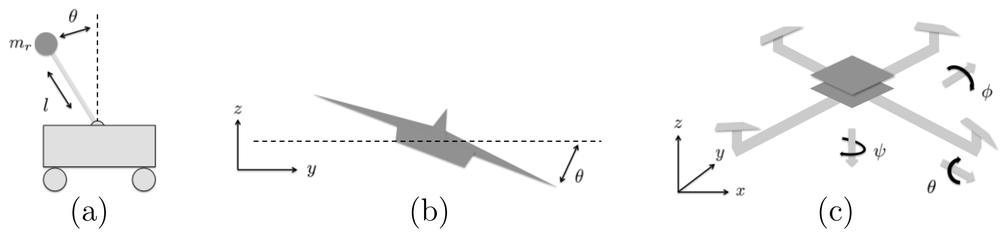


Figure 3.1: Illustrations of an inverted pendulum, a VTOL aircraft, and a quadcopter.

leaving a “hole” which can then be filled with the identity and allowing for Dirichlet conditions to be imposed. Specifically, Given an operator $\mathbb{A}$, first extract the operator's current effect upon grid points in the hypercube $\mathcal{R} = \{x \in \mathbb{R}^d \mid x_i \in [a_i, b_i]\}$, giving the operator within the domain $\tilde{\mathbb{A}}$, with separation rank $r_{\tilde{\mathbb{A}}} \leq r_{\mathbb{A}}$. As Dirichlet boundary conditions are assigned, the values on the discretization grid within $\mathcal{R}$ are set to the identity, yielding the operator $\tilde{\mathbb{I}}$, which has $r_{\tilde{\mathbb{I}}} \leq d$. The desired operator is then formed by subtracting the effects of the original operator within the region and then adding the identity, yielding an operator with bounded separation rank $r_{\bar{\mathbb{A}}} \leq d + 2r_{\mathbb{A}}$

$$\bar{\mathbb{A}} = \mathbb{A} - \tilde{\mathbb{A}} + \tilde{\mathbb{I}}. \tag{3.5}$$

3.5 Implementation Details and Examples

In the following examples, first and second order derivatives are approximated using eighth order finite differences, with the number of mesh points along each dimension varying between $M_i = 100$ and $M_i = 201$. The result are tensors that would typically not fit in the memory of even the largest modern super computers if expressed naively without the use of the separated representation, e.g., the quadcopter has a twelve dimensional space space, which would require 10^{24} float values if each dimension were discretized into $M_i = 100$ points. In each case the problem is modeled as first-exit (see Table 1.1). In all cases the noise was assumed to enter the dynamics in the same manner as the control, with $G(x) \triangleq B(x)$ in (1.18).

The operator is constructed as described in Section 3.4. The operator and boundary conditions are compressed independently using Alternating Least Squares with the linear system $\mathbb{A}$ set to identity. With this low-rank representation, the problem is then solved using Alternating Least Squares for the Hamilton Jacobi Bellman system. The Matlab Tensor Toolbox [132, 133] was employed for storage and manipulation of tensor objects.

The problems were solved on a quad-core 2.3GHz Intel i7 cpu with 16GB of memory. Denote $\bar{u}$, $\bar{x}$ as the vector of system control inputs and states for each example. Figure 3.1 illustrates the three systems considered.

Remark 26. *In several examples the sparse basis that ALS is able to construct are presented. Note that the separated components are normalized, and thus units on the axes are omitted in these plots. The weights of the separated components in the final solution are indicated in distinct figures, where relevant.*

3.5.1 Inverted Pendulum

In [134] the inverted pendulum on a cart was investigated in detail, yielding an interesting analysis on the geometry of optimal control. In particular, Osinga & Hauser produce the value function for the inverted pendulum when actuated directly at the base

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{\frac{g}{l}\sin(x_1) - \frac{1}{2}m_r x_2^2 \sin(2x_1) - \frac{m_r}{ml}\cos(x_1)u}{\frac{4}{3} - m_r\cos^2(x_1)} \end{aligned} \tag{3.6}$$

where x_1 is the angular position and x_2 the angular velocity. The cost function is $q(x) = 0.1x_1^2 + 0.05x_2^2 + 0.01u^2$. This problem has periodic boundary conditions along the x_1 dimension, and a Dirichlet boundary condition of $\phi(x_1, \pm 11) = 10$ were imposed, i.e., a high penalty for exceeding the maximal angular velocity of $\dot{\theta} > 11$ rad/s. An exit interior boundary was placed at the origin, with Dirichlet boundary conditions corresponding to unity desirability. $M_i = 201$ discretization points were used in each dimension.

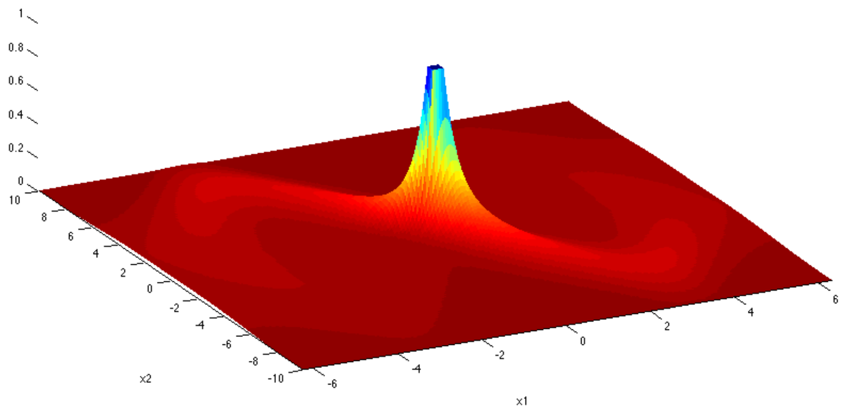


Figure 3.2: Desirability function for the inverted pendulum.

The solution Ψ is shown in Figure 3.2 while the value function obtained by inverting the transformation (1.26) is shown in Figure 3.3. The process took approximately ten minutes, achieving residual error $e = 5.22 \cdot 10^{-5}$ with a basis of $r_\Psi = 20$ rank one tensors. The five principal separated components along each dimension are shown in Figure 3.4.

3.5.2 VTOL Aircraft

Next, consider a Vertical Takeoff and Landing aircraft (also known as the Harrier Jet). A planar cross subsection of the translational state is examined, i.e., the jet's (x, y) location where y is in the vertical direction. The system is characterized by second order dynamics with gravitational drift and trigonometric inputs, giving rise to a six dimensional nonlinear system. Specifically, the equations governing the system are given in [135] as

$$\begin{aligned} \ddot{x} &= -u\sin(\theta) + \epsilon\,\tau\cos(\theta) \\ \ddot{y} &= u\cos(\theta) + \epsilon\,\tau\sin(\theta) - g \\ \ddot{\theta} &= \tau, \end{aligned}$$

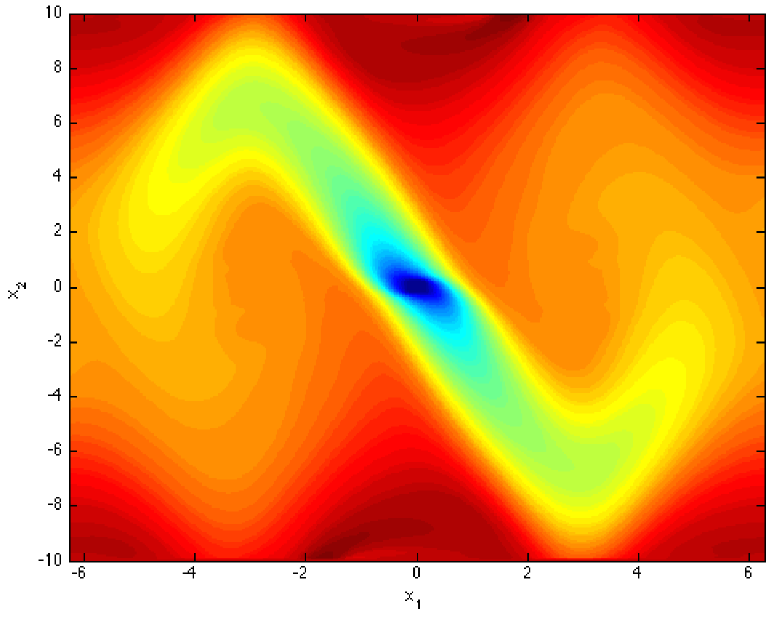


Figure 3.3: Cost to go for the inverted pendulum. The effects of the noise may be seen in the smoothing of the value function in comparison to the deterministic case seen in [134].

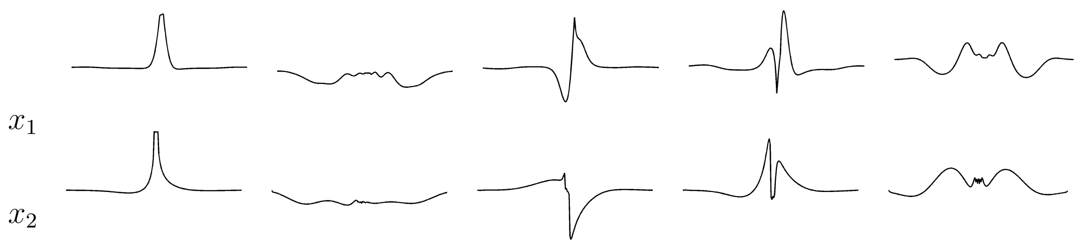


Figure 3.4: Five principal separated components for the inverted pendulum desirability solution along the x_1, x_2 dimensions.

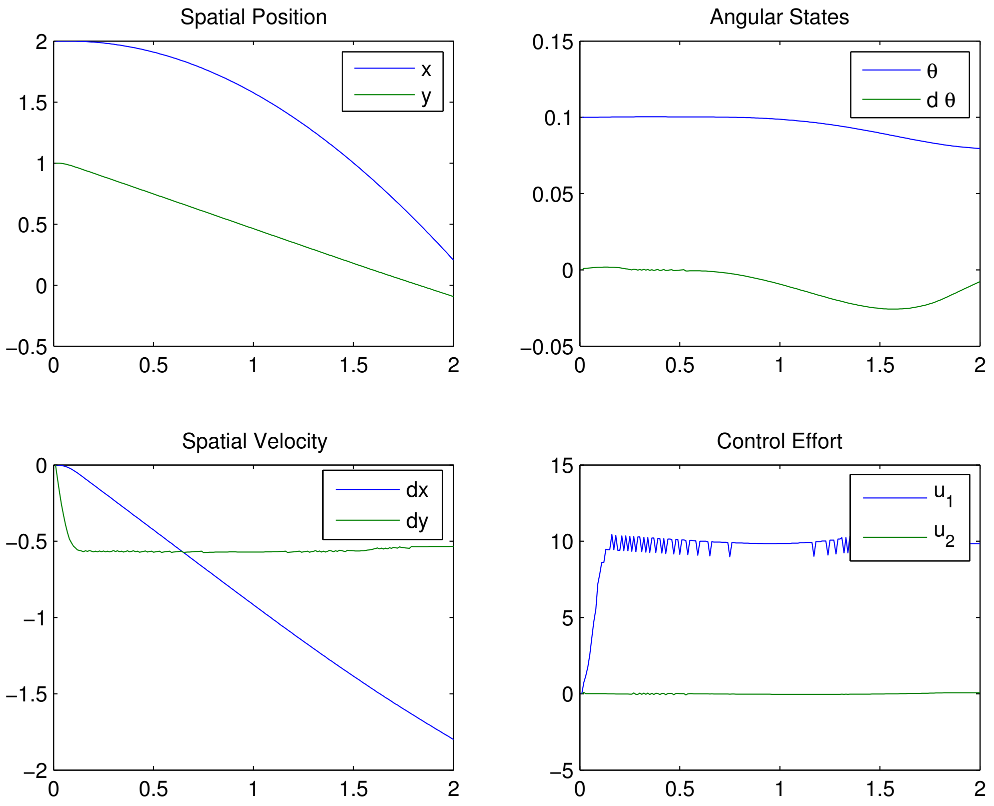
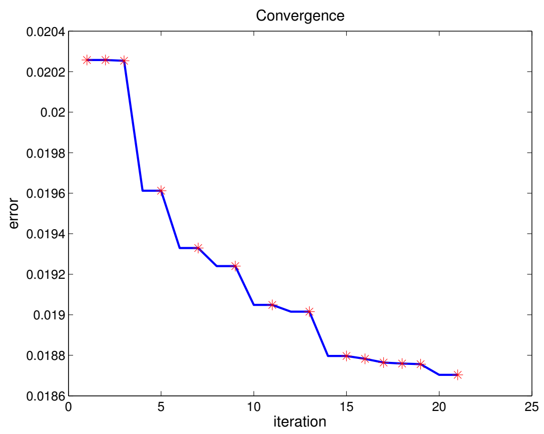


Figure 3.5: Sample trajectory when executing desirability for the VTOL aircraft.

where $\epsilon = 0.01$ for this example. The cost function chosen was $r = u^2$, and $q(x, y, \theta, \ldots) = 1.0$ on the domain $x \in [-4, 4]$, $y \in [0, 2]$, $\dot{x} \in [-8, 8]$, $\dot{y} \in [-1, 1]$, $\dot{\theta} \in [-5, 5]$, with θ periodic on $[-\pi, \pi]$. Boundary conditions were set as $\Psi\mid_{\partial\Omega} = 0$ (indicative of an infinite penalty for exiting the domain), save $y = 0$, which had condition $\Psi \mid \partial\Omega = 1 - s^2$ for each coordinate direction s, placing a target of landing with zero velocities. Discretization $M_i = 100$ were used along each dimension. The solver was limited to twenty iterations, which required approximately five minutes. A sample trajectory when executing the policy in closed loop is shown in Figure 3.5. The error and basis function weighting are shown in Figure 3.6.

3.5.3 Quadcopter

The next example is in the stabilization of a quadcopter. The derivation of the dynamics may be found in [136], and results in a system of order twelve with highly



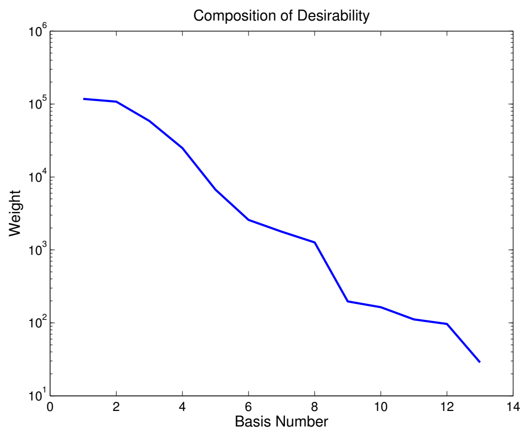


Figure 3.6: Convergence and weighting for the VTOL solution. The red markers indicate at which iterations the ALS algorithm enriched the solution by adding a basis element. The weights correspond to the variables s_l in (3.1)

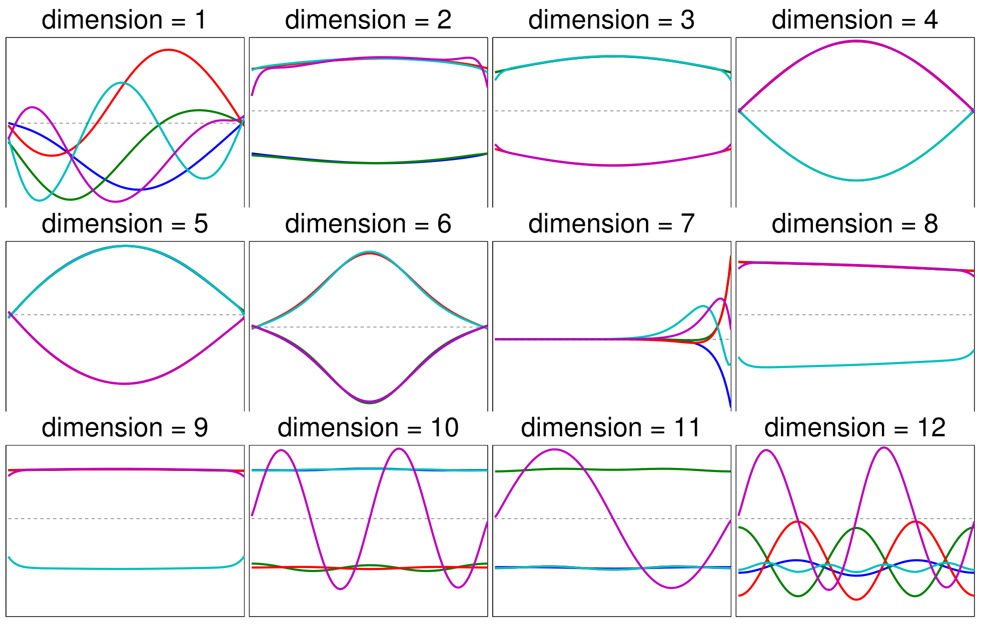


Figure 3.7: Complete basis function set for Quadcopter policy.

nonlinear dynamics.

$$\begin{aligned} m\ddot{x} &= u\left(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta\right) \\ m\ddot{y} &= u\left(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi\right) \\ m\ddot{z} &= u\cos\theta\cos\phi - mg \\ \ddot{\psi} &= \tilde{\tau}_\psi \\ \ddot{\theta} &= \tilde{\tau}_\theta \\ \ddot{\phi} &= \tilde{\tau}_\phi \end{aligned}$$

where $\eta = (x, y, z)$ are in the horizontal and vertical plane, respectively, while $\tilde{\tau} = (\tilde{\tau}_\psi, \tilde{\tau}_\theta, \tilde{\tau}_\phi)$ are the yaw, pitch, and roll moments. For simplicity, assume direct actuation control over $\tilde{\tau}$ is provided. The problem is solved with $r = \|\bar{u}\|$ and $q(\bar{x}) = 2$. Similar to the VTOL example, all boundaries are penalized, save $x = 1$, where a quadratic along the boundary in each dimension induces the system to exit with small velocity in all dimensions. This corresponds to encoding the goal that the quadcopter translate by one unit in the x-direction, and reach that location with minimal velocity. Discretization $M_i = 100$ was again used along each dimension.

In this instance the dynamics $f(x) \equiv 0$ for all but the $z-$acceleration due to gravity, whocse constant values implies a separation rank of one, and $G(x)$ has separation rank two for only the first three coordinate dimensions. The formation of the partial differential operator requires $r_{\mathbb{A}} = 56$, but the ALS algorithm is able to compress this to $r_{\tilde{\mathbb{A}}} = 24$ with a relative error of 10^{-4} in approximately two minutes, indicating there exist a great deal of underlying structure that the optimizer is able to exploit.

Only five basis functions were computed, with the results shown in Figure 3.7 demonstrating the expressive power of even particularly low rank solutions. The time for each ALS iteration is shown in Figure 3.8, along with the weighting upon each basis function. The total computation time was approximately ten minutes. Finally, Figure 3.9 shows a trajectory of the closed loop system.

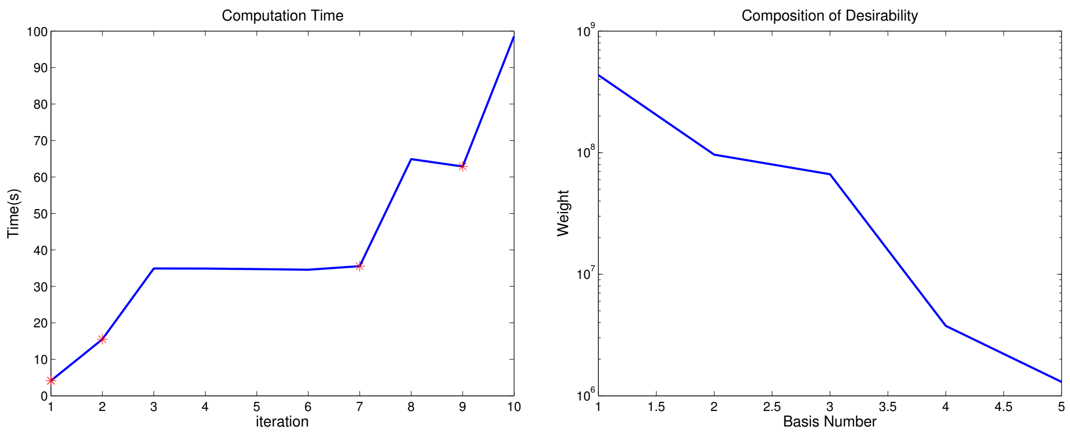


Figure 3.8: Convergence and weighting for the quadcopter solution.

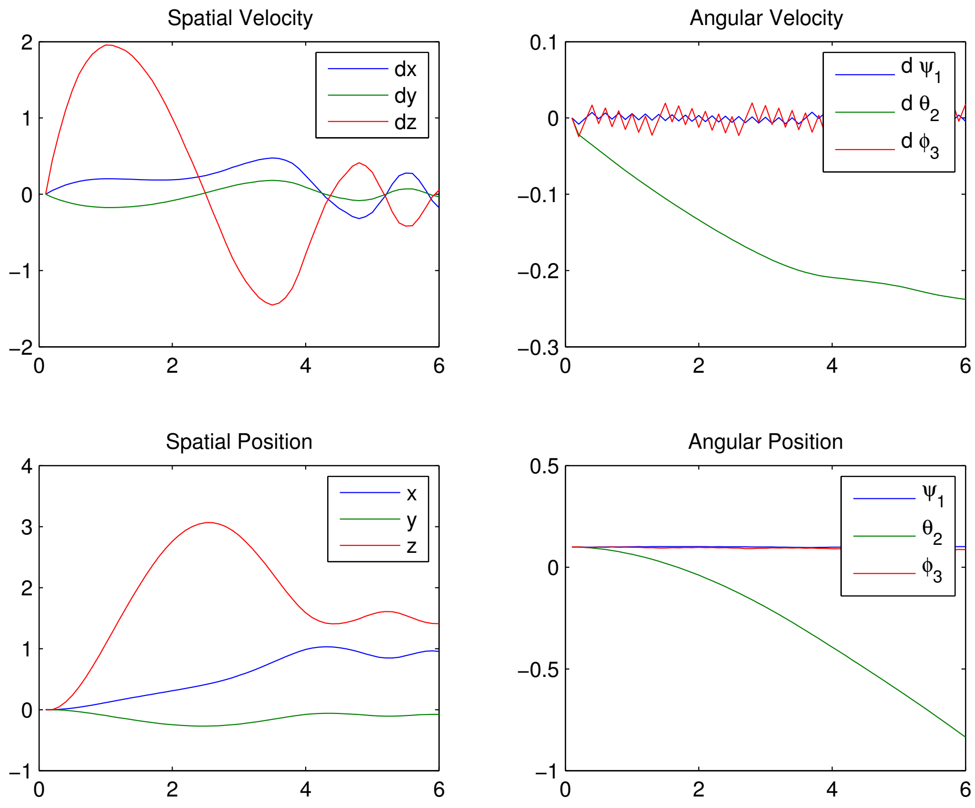


Figure 3.9: Simulation of the closed loop quadcopter system.

3.6 Discussion

There are a number of immediate implications of the work in this chapter. The first is in the control of nonlinear distributed systems. In these problems, multiple interacting systems manifest as additional dimensions for the PDE. Formally, the complexity therefore grows linearly with the number of subsystems. As well, if the coupling between such subsystems is sparse, it is expected that this interconnection could be simply described, leading to low separation rank necessary to describe the coupled dynamics.

The techniques that have been developed which rely on Sums of Squares programming [2] have been limited in degree and dimensionality due to the factorial growth in monomial basis. However, returning to the development of the separated representation, each rank-1 term corresponds to a single monomial of (3.1). By limiting the basis to those with high representative power, such problems may be scaled to arbitrarily high degree and dimensionality.

A key limitation of this work is that it requires the structural assumptions of (1.25) to obtain a linear set of equations for which ALS may be applied. The general nonlinear value function may not be directly solved. However, it has been shown that iterative linearization of the nonlinear equations may be constructed in such a manner as to solve the more general Hamilton Jacobi Bellman problem without the structural assumptions of (1.25) [137].

As detailed in Section 1.5.1 to in the introduction, these linear PDEs have a discrete counterpart in linearly solvable MDPs [63, 70]. In general, MDPs must be solved through an iterative maximization process known as value or policy iteration. However, by assuming a similar restriction on the noise of the system, specifically that it enters into the system along the same transitions actuated by the control input, Todorov has demonstrated in these works that average cost, first exit, and finite horizon optimal control problems may be solved through a set of linear equations, which may also be approached in the separated representation setting.

Chapter 4

Navigation Functions as Optimal Controllers

As discussed in Section 1.3, Navigation Functions are one of several methodologies that researchers have used for robotic path planning, with a review given in Section 1.3.5. The focus on Navigation Function design has historically not been on optimality of robot motion, but instead on the ability to rapidly compute motion plans from any location in the state space. This chapter demonstrates how to incorporate optimality criteria into Navigation Function construction that can be modeled as the sum of a (possibly) nonlinear state dependent cost and a quadratic control cost. This contrasts with the prior work on navigation, which implicitly defines a decomposition of the problem into a trajectory generation method (the solution of the navigation function) and a local feedback-based trajectory following control method. This classical two-step decomposition may lead to suboptimality in the path planning or control law, or even instability [55]. By formulating the problem starting from the Hamilton Jacobi Bellman problem, this chapter introduces a method which allows the impact of the dynamic model to be directly incorporated into the navigation function, if desired.

This chapter relates the Navigation Function approach to an optimal control problem which optionally includes the presence or absence of both dynamics and a state dependent cost function. It is found that when dynamics and control cost are neglected, the resulting solution is similar to those previously used to generate navigation functions in the literature [138, 139]. The result is that a spectrum of problems, ranging from the full Hamilton Jacobi Bellman equation to the classical potential-based navigation function, are made explicit and the tradeoffs in modeling complexity becomes visible. The analysis presented in this chapter makes it apparent how dynamics may then be incorporated to a navigation function if desired. To the author's knowledge, this chapter also represents the first attempt to formally include stochastic uncertainty into the construction of the navigation function. The central result is that techniques that have been used to create navigation functions historically can be interpreted as solving a particular optimal control problem subject to a specific form of stochastic forcing.

Let $\mathcal{CS}$ denote the robot's *configuration space* (or *c-space*)– the possible configurations that a robot can occupy. As is standard, let the subset of $\mathcal{CS}$ where the robot collides with an obstacle define the set of *configuration-space obstacles*, $\mathcal{CO}$, while the *free configuration space*, $\mathcal{F} \subset \mathcal{CS}$, is the complement of $\mathcal{CO}$ in $\mathcal{CS}$. Under the assumption of perfect sensory information, the motion planing task is to move the robot from its starting configuration, $q_{init} \in \mathcal{F}$ to a desired goal position $q_d \in \mathcal{F}$. One approach to solve this problem is to construct a *navigation function* (also introduced in Section 1.3.5):

Definition 27. *(From [50]) Let q_d be a goal configuration in $\mathcal{F}$, the free c-space. A map $\varphi : \mathcal{F} \to [0, 1]$ is a* navigation function *if it is*

1. *smooth on $\mathcal{F}$ (at least a $C^{(2)}$ function);*
2. *polar at q_d, i.e., has a unique minimum at q_d on the path-connected component of $\mathcal{F}$ containing q_d;*
3. *admissable on $\mathcal{F}$, i.e., uniformly maximal on the boundary of $\mathcal{F}$;*
4. *a Morse function, i.e., the Hessian at critical points is nonsingular.*

Given a navigation function, $\varphi(q)$, the robot's path to the goal from any staring configuration in $\mathcal{F}$ can be realized by following the gradient $\nabla\varphi(q)$ at each q. The definition assures that the robot will achieve the goal while remaining in $\mathcal{F}$, and not become trapped in a local minima of $\varphi(q)$.

Navigation functions may be constructed in several forms. In the classical approach of Koditschek & Rimon [51], a navigation function may be calculated analytically when the when the bounded problem domain, the obstacle shapes, and the goal region are all diffeomorphic to spheres. Similarly, if the boundary, obstacles, and goal region are star-shaped sets (which are homeomorphic to spheres), then one can compute the navigation function by transforming the problem to a sphereworld, find the sphereworld navigation function, and transform the function back to the original problem domain.

4.1 Navigation Functions Constructed from Optimal Control

The Stochastic Optimal Control problem introduced in Section 1.4 is first reduced to the standard setting of navigation functions by sequentially incorporating the assumptions which hold in the classical navigation function setting. These successive eliminations of terms will then illuminate some connections between the approach presented in this chapter and classical navigation function approaches. Finally, the approach will allow the formulation of approximate minimum time solutions.

For the remainder of the chapter, it is assumed that the system has full state controllability, an assumption common in the Navigation Function literature. It is also assumed that the system obeys the assumption (1.25), allowing for the linear Hamilton Jacobi Bellman equation to be formed. Recall the Hamilton Jacobi Bellman equation of (1.27)

$$-\partial_t \Psi = -\frac{1}{\lambda} q \Psi + f^T \left(\nabla_x \Psi \right) + \frac{1}{2} Tr \left(\left(\nabla_{xx} \Psi \right) \Sigma_t \right)$$

Dynamics. Since the classical navigation function approach implicitly decouples the trajectory generation problem from the trajectory following control design, the dynamics of the specific mechanical system to be guided are ignored. The Navigation Hamilton Jacobi Bellman equation, is defined as the HJB PDE with the dynamic dropped, i.e., $f := 0$. This simplification results in the *Navigation PDE*:

$$0 = -\frac{1}{\lambda} q \Psi + \frac{1}{2} Tr \left(\left(\nabla_{xx} \Psi \right) \Sigma_\epsilon \right).$$

Similarly, the classical navigation function setting does not consider spatially dependent costs. Thus, the state-dependent term in the cost function, $q(x)$, may be simplified to a free scalar parameter $q := \alpha$, producing the PDE

$$0 = -\frac{\alpha}{\lambda} \Psi + \frac{1}{2} Tr \left(\left(\nabla_{xx} \Psi \right) \Sigma_\epsilon \right) \tag{4.1}$$

This PDE is termed the *Augmented Navigation PDE*, as it incorporates additional cost information as compared to traditional navigation functions, but does not include the effects of system dynamics. If one wishes to include the robot mechanism's dynamics, their presence in the function $f(x)$ will require the addition of these states as dimensions in the Hamilton Jacobi Bellman PDE.

Interestingly, the PDE (4.1) is well known as the homogeneous *Screened Poisson Equation* and has previously found applications in image processing [140]. Most importantly, this is a second order PDE with isotropic diffusion and mass terms, a situation which has been well studied [128].

Boundary Costs. The boundary conditions for the PDE (4.1) correspond to the penalty accrued as the robot exits the configuration domain and collides with an obstacle or reaches the goal state. In Equation (1.20) this effect is represented as the terminal cost ϕ. Recall that according to Equation (1.26), this terminal cost must be transformed, along with the value function, to the desirability domain. Thus, the boundary condition can be stated as

$$\Psi \mid_{\partial\Omega} = e^{-\frac{\phi}{\lambda}} \tag{4.2}$$

where $\partial\Omega$ is the boundary of the operating domain, Ω. Classically, the cost assigned to a collision has been modeled as uniform over all obstacles, and thus the boundary condition is $\phi(x_T) = c$ for an arbitrary constant c, in accordance with Property 3 of

Definition 27. Other choices are certainly possible, allowing for varying weights to be placed on different boundary types.

The free variables $q(x)$ and R define a notion of cost, and therefore a notion of optimality. The inclusion of these variables allows us to compare navigation functions according to their perceived cost, and furthermore to declare navigation functions optimal with respect to a choice of criteria. Such criteria has traditionally been eschewed in favor of simplicity in construction of the navigation function, and hence this framework may be said to be a slight generalization, bringing notions of optimality into consideration.

Control-dependent costs. Recall that the initial definition of cost (1.19) includes a control dependent term. Navigation functions have traditionally been unconcerned with the control effort. Recall that the assumption on control effort and noise (1.25) needed to realize a linearly solvable Hamilton Jacobi Bellman PDE is:

$$\lambda G(x) R^{-1} G(x)^T = \Sigma_t \tag{4.3}$$

where Σ_t is fixed as a function of the known control vector field matrix, $G(x)$, and noise characteristics, $B(x)$ and Σ_ϵ. The control effort penalty R cannot be brought to zero naively without violating this assumption. It is possible to compensate for this limitation by using the free parameter λ to maintain the underlying relation in this assumption. That is, set $\lambda = \beta$ and define $R = \beta\tilde{R}$, yielding expressions

$$\begin{aligned} \lambda G(x) \left(\beta \tilde{R}\right)^{-1} G(x)^T &= \Sigma_t \\ G(x) \tilde{R}^{-1} G(x)^T &= \Sigma_t \end{aligned} \tag{4.4}$$

which is independent of β, allowing the control penalty cost to be reduced to zero. The difficulty is that as $\lambda \to 0$, (4.1) becomes nonsensical in the limit. Fortunately, no cost has ben assumed over the states, and thus it is possible to set $\alpha = 0$ to produce the *Navigation PDE*

$$0 = Tr\left(\left(\nabla_{xx}\Psi\right)\Sigma_t\right) \tag{4.5}$$

which is recognized to be Laplace's equation scaled according to the system noise characteristics. The practical cost incurred by this reduction of the complete SOC HJB, Equation (1.27), to Equation (4.5) is that consideration of control effort and state dependent penalties have been neglected, which is often natural in the robotics setting. Interestingly, Laplace's equation has been used previously in the generation of navigation functions [141, 138, 139]. In this prior work, the authors suggested the use of Laplace's equation, with the motivation that solutions to Laplace's equations can be shown to have no local minima over their domain. The following theorem justifies this from an optimality perspective, albeit through the transformation (1.26).

Theorem 28. *The optimal robust desirability function absent costs over state is given by $V = -\lambda \log \Psi$ where λ is defined according to* (4.3)*, and Ψ is the solution to the following Laplace equation over the domain Ω:*

$$\begin{aligned} 0 &= Tr\left(\left(\nabla_{xx}\Psi\right)\Sigma_t\right) && (4.6) \\ \Psi\mid_{\partial\Omega} &= e^{-\frac{\phi}{\lambda}} && (4.7) \end{aligned}$$

There is an interesting trade-off resulting from Eq. (4.3). Define $\tilde{\Sigma}_t \triangleq \gamma\Sigma_t$ in order that the system noise may be scaled by γ. Define $\lambda = \beta$, as in Equation (4.4) in order to scale the control penalty. Then $\lambda = a\beta\gamma$ for some fixed constant a. The result is that a scaling of the control effort has the same effect on the solution as a scaling of the noise, and this scaling manifests only through the transformation (1.26) and the boundary conditions, and surprisingly not through the differential constraints on Ψ, as γ is simply cancelled in Equation (4.5). What does have an effect, however, is the directional influence of Σ_t, i.e., the solution will incorporate paths that have beneficial drift.

Due to the exponential dependence of Eq. (1.26), one cannot realize the limit $\beta \to 0$, as the transformation simply becomes nonsensical in the limit. Instead, β is chosen based on the magnitude of the noise, or the level of control penalty, depending on the perspective.

Remark 29. *Laplace's equation has been justified in the presence of noise here,* *whereas it was previously justified due to its lack of local minima.*

Two PDEs (4.1), (4.5) have been produced in this initial analysis. The first of these allows one to naturally incorporate several optimality criteria into the concept of a navigation function, while the second is especially simple and creates a connection to previously inspired navigation function formulae.

4.2 Approximate Time-Optimal Navigation Functions

The design freedom afforded by the existence of parameters in the cost function allows for the solution to be biased towards time-optimal navigation functions by penalizing time spent away from the goal. This is accomplished by setting $q(x) = c$ for some constant c. Control effort may also be adjusted through the free parameter λ, and its cost can be decreased relative to the state cost. The robustness of the navigation function to noise may also be controlled through the noise characteristics defined by Σ_ϵ, and again reduced. As the value of the constant c is increased while parameters λ and Σ_ϵ are increased, cost is accrued only when the system remains outside the goal region. The optimal action during this time is to take the quickest path to the goal, ignoring the amount of control effort used.

4.2.1 Analogy with electro-statics

It is interesting that early researchers on potential field navigation methods were naturally drawn towards analogies with electro-statics. Khatib's seminal work [52] conceptually frames the collision avoidance problem as a process of adding potential fields that would repulse or attract the point robot mass in much the same way as electrostatic fields might. This intuitive notion of attractive and repulsive forces therefore can also be grounded in notions of optimality. Indeed, for the Navigation PDE of Eq. (4.5), the analogy is exact in desirability-space, with the representation of obstacles and goals manifesting identically to static charges on their surfaces. However, a logarithmic transform improves the solution from an intuitive one to one which is optimal.

4.2.2 Convergence of the solution trajectories

Due to the presence of the control cost in the solution to the Hamilton Jacobi Bellman equation, it isn't possible to directly determine the probability that the controlled system successfully reaches a goal region. As mentioned in Remark 4.1, entirely removing the affect of this cost component isn't possible as the necessary formulae break down in the limit. However, it is possible to choose a small value for the control cost R. The cost-to-go is then predominantly governed by the cost of colliding with an inadmissible boundary. By setting the boundary conditions for obstacles to $\phi = 1$, the cost-to-go then becomes a *conservative* approximation of probability of success in reaching the goal, i.e., the cost-to-go will overstate the probability of failure by also including the cost of future trajectories' control effort in its value. For even moderately small values of R, this approximation may not be overly conservative.

4.3 Navigation Examples

The approach introduced in this chapter is now illustrated by numerical examples. Each of these examples is solved with a discretization mesh with grid size $h = 0.1$. The derivatives of the PDEs are solved via the second order central Finite Difference Method [130].

4.3.1 Problems with Analytical Solutions

In simple domains it is possible to find an analytical or simply computed solution to (4.1), (4.5). For instance, suppose a point robot is commanded to move to a goal location located at the origin of a two-dimensional configuration space with no obstacles, while it is perturbed with noise whose characteristics are uniform across the configuration space. The solution to the associated Navigation PDE (4.5) corresponds to the solution of Laplace's equation for a point potential, i.e., the fundamental solution, which is

$$\Psi = -\frac{\log(r)}{2\pi}$$

where r is the distance from the robot's current configuration to the origin [94]. The solution to the Augmented Navigation PDE (4.1) for this problem may be found as follows. Taking the Fourier Transform (4.1) yields

$$\left(k^2 + \frac{\alpha}{\lambda}\right)\tilde{\Psi} = 1.$$

The fundamental solution is then found by solving for $\tilde{\Psi}$ and then finding the inverse Fourier Transform, which yields

$$\Psi = \frac{1}{2\pi} K_0\left(\sqrt{\frac{\alpha}{r}}\lambda\right)$$

with K_0 the modified Bessel function of the second kind. These kinds of analytical solutions suggest that optimal or near optimal solutions may be used to create navigation functions quickly through composition, a topic for further study.

4.3.2 Effect of Noise on Corridor Navigation

Next, the method is demonstrated on a two dimensional robot whose task is to reach the top right corner of a square configuration space. The domain has two obstacles, creating a pair of corridors that the robot must traverse if it begins in lower portion of the configuration space. For this example $\Sigma_t = 2I_{2\times 2}$, $\lambda = 1$, and the width of the thinner corridor is set to two different values of 1.5 and 2 distance units for comparison. The resulting solutions are shown in Fig. 4.1. This example shows why it can be important to include noise in the construction of a navigation function. Consider the situation when the robot starts near the bottom of the figure. In both environments, robot can travel through two different corridors to reach the goal. In both environments, the navigation functions lack local minima, as expected. In the left-hand environment, the robot can potentially choose between a wide corridor, and a medium width corridor. The choice of the corridor will depend upon the robot's

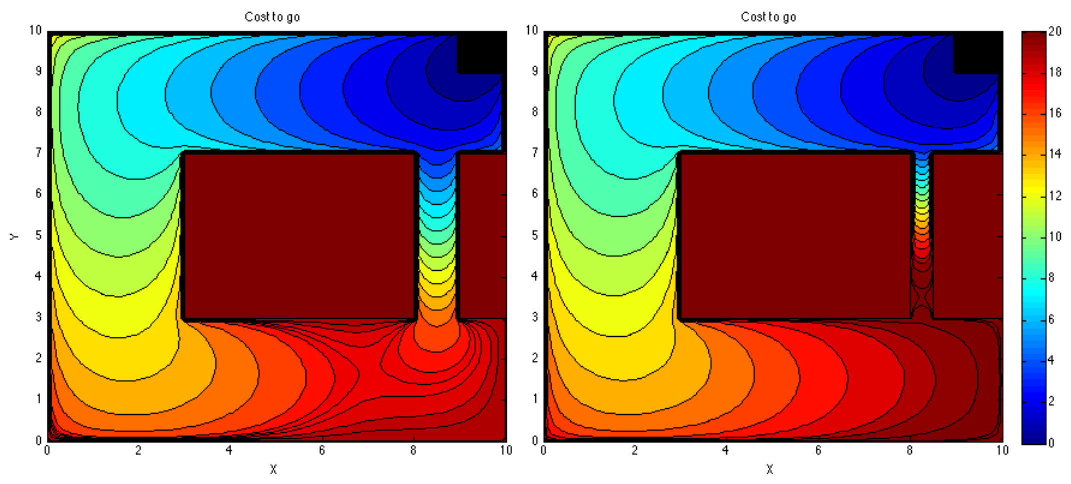


Figure 4.1: Navigation function for a two dimensional point-mass robot calculated according to Eq. (4.5) with varying corridor widths. The goal location is the black square in the top right corner.

specific starting configuration, as it is safe to traverse both corridors. In the right hand figure, the robot can potentially choose between the same large corridor, or a very narrow corridor (whose width is $\frac{3}{4}$ the magnitude of the noise variance). For almost all starting positions, the navigation function guides the robot away from the potentially dangerous narrow corridor, unless the robot happens to start positioned well into the narrow corridor. This intuitively logical result occurs because the potential for collision in the narrow corridor, given the uncertainty on the stochastic forcing on the robot places too high a cost on that potential path.

Remark 30. *The solution to Eqs.* (4.1)*,* (4.5) *take place in exponentiated coordinates, and for many examples tend to be close to zero for large regions of the state space. It is therefore usually more useful to consult the value function directly rather than examining the desirability.*

4.3.3 Maze

The second example shows that complicated environments can be well handled by this method, and also highlights the effects of including additional cost criteria into the navigation function. The same robot dynamics and same noise distribution of the previous example are used, however the obstacles are placed in a more complicated maze-like pattern. The Augmented Navigation PDE of Eq. (4.1) is used in order to incorporate the additional optimality criteria. The resulting navigation functions are shown in Figure 4.2 with several noise and cost configurations..

The results shown in the figures compare the use of (4.5) with additional cost criteria. It is seen that the solutions of (4.5) and (4.1) are qualitatively similar. As λ is decreased in the general case, this has the interpretation of either increasing the control cost weighting, R, or increasing the noise covariance. The solutions do not change qualitatively, only the magnitude of the cost-to-go. In contrast, the approximated minimal-time solution is characterized by shortest-path level sets. These level sets are characterized by straight lines near corners, in contrast to the circular level sets of other examples, as would be expected for a shortest path solution amongst piecewise-linear obstacles.

Remark 31. *For some choices of costs q, R the navigation function* (4.1) *may bring the robot directly into an obstacle. This is no contradiction, as framing the problem through the lens of optimal control allows for freedom on the placement of boundary conditions. The penalty for hitting an obstacle, if chosen improperly, may be less than the cost to traverse the domain and enter the goal region, and thus the most economical choice is for the robot to simply collide with the boundary. This isn't a problem when using* (4.5).

4.3.4 Grasping

A final example is of a simple planar grasping task wherein a gripper must be positioned in the plane so that it may close and grasp a small nut, an illustration of which is shown in Figure 4.3. The goal of the problem is to move the end effector to a desirable location surrounding the object in such a way that the gripper's orientation places the jaws around the nut, whereupon a simple closing action will reliably grasp the nut. The problem is transformed into the system's configuration space, and then solved in the Optimal Navigation framework, with the results shown in Figure 4.3a. As the method treats the goal states as a set of boundary conditions, the relatively

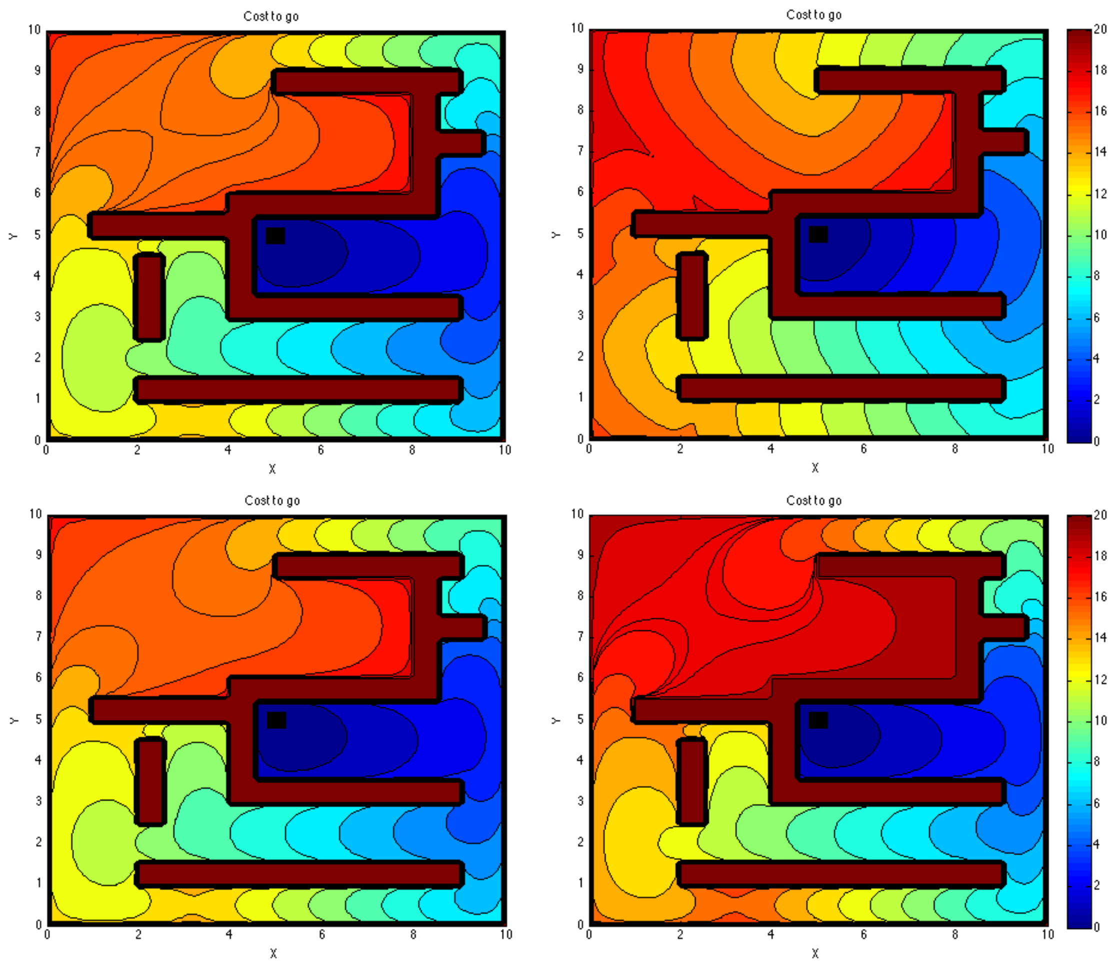
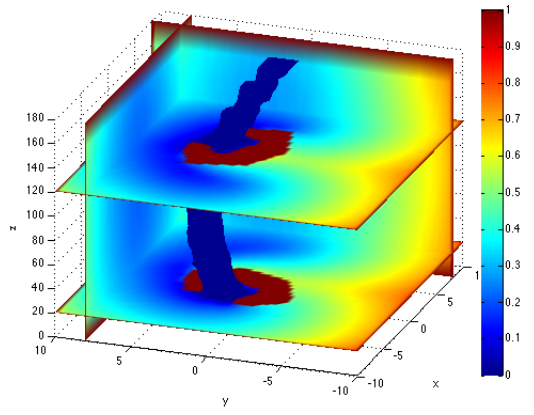
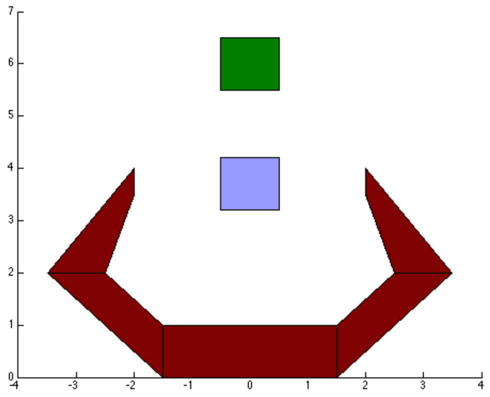


Figure 4.2: Navigation function for a two dimensional point-mass robot in a maze-like environment with equi-dimensional noise. On the top-left is the standard navigation function calculated according to (4.5). On the top-right, the minimum time-criteria is approximated by taking $\alpha = 100$, $\lambda = .04$ in (4.1). The bottom-left and -right have $\Sigma_t = 4I_2, 4.6I_2$, and $\lambda = .4, .46$ respectively. The obstacles and boundary are chosen to have penalty of 20 units, while the goal region has a penalty of 0 units. The goal is located at the center of the domain at $(x, y) = (5, 5)$ and is illustrated as a black rectangle.



(a)



(b)

Figure 4.3: An illustration of a nut grasping task. In (b) the square nut is shown in green while the gripper is shown in red. The blue region denotes a range of acceptable goal nut locations relative to the gripper. On the left in (a), cross sections of navigation function for the nut grasping task illustrated in Figure 4.3. Parameters used are $\alpha = 0.02, R = .02I_{3\times3}, \Sigma_t = 5I_{3\times3}$ and the boundary costs are set to $\phi = 1$. Spatial discretization in the $x-$ and $y-$coordinates are $h_x = h_y = 0.25$, and in the angular direction $h_\theta = 20°$. The goal region isosurface is displayed in dark blue.

large goal region is handled easily.

The example illustrates the approach for a non-point mass robot, and illustrates the smoothness of the navigation function over the domain in a typical manipulation task. It is easy to see the optimal path of the gripper, wherein it smoothly rotates, following the basin of low cost in blue, until the nut is captured.

4.4 Discussion

This chapter introduced a generalization of the classical navigation function framework used in robotics to include system noise, dynamics, and cost criteria. Philosophically, this chapter links the classical robotics subject of navigation functions with recent advances in Stochastic Optimal Control. Previous results that were developed on a somewhat ad-hoc basis have been shown to be related to optimality considerations, and the intuition which led to their development well placed. Remarkably, many existing results using harmonic potentials can be shown to be optimal for a particular configuration of noise and cost models when the solution is simply adjusted by a logarithmic transformation.

The use of navigation functions to approximately incorporate minimum time task requirements was also demonstrated. From a practical point of view, the methods developed here to construct the navigation functions can be applied to environments, obstacles, and robots with arbitrary smooth geometries. Furthermore, it allows for the results of existing methods (i.e. [142]) to be compared against the underlying *optimal* solution to the problem, if the system's noise characteristics are captured by the model. The ability to solve for the navigation function in terms of a linear PDE also expands the space of algorithms for calculating solutions.

Numerical experiments show that solutions in configurations space having dimension up to 5 can be computed on a desktop PC computer using Finite Difference. The methods developed in Chapter 3 are applicable and allow for Navigation functions to be developed for systems of far higher dimensionality. Additionally, it may be possible to use the results presented here to inform the choice of artificial potential fields, yielding navigation functions that may be assembled quickly, but that better approximate the optimal navigation function for the problem.

Chapter 5

Temporal Task Planning

Up to this point, this thesis has largely been concerned with solving problems with individual goal-guided tasks, encoded as the minimization of a general cost function. However, many problems in the design of autonomous control systems require the ability to reason over multiple sub-tasks that must be executed according to some temporal and logical rules. In this chapter, as a result of collaboration with Eric Wolff and Richard Murray [6], a method for synthesizing control policies for continuous-time stochastic nonlinear systems with temporal logic task specifications is presented. These problems are motivated by safety-critical robotics applications involving autonomous ground and air vehicles executing complex tasks. In such applications, it is desirable to automatically synthesize a control policy that provably implements specified system behavior, despite nonlinearities and disturbances.

Linear temporal logic (LTL) is a task specification language that has been widely used for specifying properties of hybrid systems, robots, and software. Syntactically co-safe LTL, an expressive finite-time fragment of LTL, is used to specify a wide range of properties relevant to autonomous systems. These properties include safety, response to the environment, and goal visitation. Such properties generalize classical motion planning [143].

Common approaches for control policy synthesis for stochastic systems with LTL specifications first abstract the dynamical system as a finite Markov decision process (MDP) [144, 145]. Each state in this MDP corresponds to a subset of the system state space, and transition probabilities between states in the MDP encode possible system behaviors. Given such an MDP and an LTL specification, control policies can be automatically constructed using an automata-based approach [146, 145]. This approach extends work in the formal verification community [147, 148] to hybrid systems via the use of finite abstractions. The main drawback of this approach is that it is computationally expensive to create a finite abstraction [144, 145].

This chapter introduces a method wherein the expensive computation of an MDP abstraction is avoided, and instead the solution is directly computed on the state space of the system using techniques from stochastic optimal control. An automaton representing the temporal logic specification guides the computation of a control policy that maximizes the probability that the system satisfies the specification. This automaton is treated as an MDP, where states encode progress towards task completion and each action corresponds to a control policy for the continuous system. Value iteration is used to maximize the probability that the system satisfies the specification from its initial state. A feedback control policy is returned which selects the current action based on the system's continuous state and its mode.

The approach avoids computing a discrete abstraction of the system, and lets one take advantage of recent advances in computing constrained reachability relations for nonlinear stochastic systems. In particular, the Hamilton Jacobi Bellman equation takes the form of a linear PDE. The solutions to linear PDEs obey the principle of superposition, a characteristic first exploited in [83]. This chapter expands upon that idea, leveraging superposition to quickly solve problems defined with temporal specifications. Indeed, the computation of solutions by superposition is appealing in situations where many control problems must be solved over a common domain, as in many temporal logic planning problems. For such problems, the specification creates a large number of reach-avoid subproblems, where the property of superposition is leveraged to efficiently compose the subproblems.

The main contribution of this work is a framework for control policy synthesis for stochastic nonlinear systems for syntactically co-safe LTL. The framework is general in that it can utilize any technique that computes solutions to a stochastic constrained reachability problem. The case where the Hamilton Jacobi Bellman equation is linear is leveraged to increase the efficiency of the approach. This is done by exploiting the fact that solutions constructed for individual specifications may be superimposed to satisfy richer specifications.

The work is closely related to recent automata-guided approaches for control policy synthesis for discrete-time deterministic [149] and stochastic [150] systems subject to temporal logic specifications. The work is also related to [151], which also encodes the relationship between sequential tasks via the boundary conditions to Hamilton Jacobi Bellman equations. These approaches directly compute over the state space of the system. The work in this chapter differs in that continuous-time dynamics are considered, and a compositional approach based on superposition of solutions to linear PDEs is used. The compositional approach allows solutions to be quickly computed, at the expense of additional conservatism (see Section 5.3).

The system model and specification language is introduced in Section 5.1, and the main problem is stated in Section 5.1.3. It is shown how to compose solutions in Section 5.3, and demonstrate how this composition provides a critical benefit in the context of temporal logic planning problems. Numerical experiments are presented in Section 5.4.

5.1 Preliminaries

Notation: An *atomic proposition* is a statement that is either $True$ or $False$. The expectation of a random variable is denoted by $\mathbb{E}[\cdot]$, and the probability of an event is denoted by $\mathbb{P}[\cdot]$.

5.1.1 System Model

Consider continuous-time stochastic nonlinear systems that evolve with dynamics

$$dx_t = (f(x_t) + G(x_t)u_t)\,dt + B(x_t)L\,d\omega_t, \tag{5.1}$$

with state $x_t \in X \subset \mathbb{R}^n$, control input $u_t \in \mathbb{R}^m$, Brownian motion $\omega_t \in \mathbb{R}^m$, and disturbance matrix $L \in \mathbb{R}^{n \times m}$. The functions $f(x_t)$, $G(x_t)$, and $B(x_t)$ are continuously differentiable, and the set X is compact.

Let AP be a finite set of atomic propositions. The labeling function $L : X \to 2^{AP}$ maps the state to the set of atomic propositions that are $True$. The set of states where atomic proposition $p \in AP$ holds is denoted by $[\![p]\!]$.

Definition 32. *A* memoryless control policy *for a system of the form* (5.1) *is a map* $\mu : X \to \mathbb{R}^m$. *A* finite-memory control policy *is a map* $\mu : X \times M \to \mathbb{R}^m \times M$ *where the finite set* M *is called the memory.*

The *trajectory* $\mathbf{x}(x_0, \mu, \boldsymbol{\omega}) = \mathbf{x} : R_{\geq 0} \to X$ represents a solution of (5.1) induced by an initial state $x_0 \in X$, a given control policy μ, and an instance of Brownian motion $\boldsymbol{\omega}$. A *word* of a trajectory $\mathbf{x}$ is an infinite sequence of labels $L(\mathbf{x}) = L(x_{t_0})L(x_{\tau_0})L(x_{t_1})L(x_{\tau_1})L(x_{t_2})\ldots$, such that $t_0 = 0$ and for all $i \geq 0$, $t_{i+1} \geq t_i$, $\tau_i \in [t_i, t_{i+1}]$, and $L(x_t) = L(x_{\tau_i})$ for all $t \in (t_i, t_{i+1})$. A word of trajectory $\mathbf{x}$ defines the behavior of $\mathbf{x}$ in terms of the label sequence. The assumption is made that during any finite time interval, the label changes a finite number of times.

The set of words of system (5.1) with initial state $x_0 \in X$ induced by a control policy μ is denoted by $\mathbf{x}(x_0, \mu)$. The Brownian motion induces a probability measure over the trajectories of the system $\mathbf{x}(x_0, \mu)$, and thus the words.

5.1.2 Specification Language

Syntactically co-safe linear temporal logic (sc-LTL) [152] is used to concisely and unambiguously specify desired system behavior over a finite horizon. In this work, system behavior is examined only over a finite horizon due to the unbounded disturbances in Equation (5.1). A brief introduction to the syntax and semantics of sc-LTL follows, and the reader is referred to [152] for details.

An sc-LTL formula is formed from: the Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and the temporal operators: $\mathcal{U}$ (until), and $\diamond$ (eventually). An sc-LTL formula is written in positive normal form (i.e., negations are only allowed in front of atomic propositions). The $\bigcirc$ (next) temporal operator is not considered, which is ill-defined in continuous-time.

Definition 33. *A syntactically co-safe LTL (sc-LTL) formula over a set of atomic propositions is inductively defined as follows:*

$$\varphi ::= p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U} \varphi_2 \mid \Diamond \varphi,$$

where $p \in AP$ is an atomic proposition.

The semantics of an sc-LTL formula is defined over infinite words $w = w_0 w_1 w_2 \dots$ where $w_i \in 2^{AP}$. Informally, $\varphi_1 \mathcal{U} \varphi_2$ means that φ_1 is $True$ until φ_2 is $True$ and $\Diamond\varphi$ means that φ eventually is $True$. More complex specifications can be defined by combining Boolean and temporal operators. The satisfaction of an sc-LTL formula is guaranteed in finite time [152].

There exists a close connection between sc-LTL formulae and deterministic finite automata.

Definition 34. *A* deterministic finite automaton *(DFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with a finite set of states Q, a finite alphabet Σ, a transition function $\delta : Q \times \Sigma \to Q$, an initial state $q_0 \in Q$, and a set of accepting states $F \subseteq 2^Q$.*

An accepting run *σ of an automaton $\mathcal{A}$ on a finite word $w = w_0 \dots w_k$ over $\Sigma = 2^{AP}$ is a sequence of states $\sigma = q_0 \dots q_{k+1}$ such that q_0 is the initial state, $q_{k+1} \in F$, and $\delta(q_i, w_i) = q_{i+1}$ for all $i = 0, \dots, k$.*

For any sc-LTL formula φ, there exists a deterministic finite automaton $\mathcal{A}_\varphi$ that accepts exactly the prefixes of all satisfying words. Software exists for constructing such a deterministic finite automata from an sc-LTL formula [153].

5.1.3 Problem Statement

The problem is now stated formally. First, the probability of satisfaction of a specification by a stochastic system of the form (5.1) is defined.

The stochastic system (5.1) may have an infinite set of words $\mathbf{x}(x_0, \mu)$ for a given initial state x_0 and control policy μ. There is a well-defined probability measure over this infinite set of words [154], which gives rise to the notion of the expected satisfaction probability of a specification.

Definition 35. *Let $\mathbf{x}(x_0, \mu)$ be a set of words of system* (5.1) *from initial state x_0 under control policy μ with an associated probability measure. Let φ be an sc-LTL formula over AP. Then, $\mathbb{P}(\mathbf{x}(x_0, \mu) \models \varphi)$ is the* expected satisfaction probability *of φ by system* (5.1) *under control policy μ.*

Problem 36. *Given a system of the form* (5.1) *and a syntactically co-safe LTL formula φ over AP, compute a control policy μ^* that maximizes the probability that φ is satisfied, i.e., $\mu^* \in \arg\max_\mu \mathbb{P}(\mathbf{x}(x_0, \mu) \models \varphi)$.*

A conservative solution to Problem 36 is developed in the remainder of this chapter. At a high level, Problem 36 is reduced to a series of reach-avoid problems, where the system attempts to reach a goal region while avoiding other regions, e.g., obstacles. Each of these individual reach-avoid problems, referred to formally as stochastic constrained reachability problems defined below, may be solved through the tools of optimal control. These solutions may be chained together, via a dynamic programming argument, to solve temporal tasks specified in the sc-LTL language. As the specification becomes more complex, many such reach-avoid problems must be solved, creating a growth in complexity that typically stymies existing techniques. By leveraging the principle of superposition on the underlying repetitive task of solving optimal control problems over a common domain, the methods presented below scale to complex tasks.

5.2 Solution

In this subsection a general method for (conservatively) solving Problem 36 is presented. This method uses an oracle for solving stochastic constrained reachability problems. A computationally efficient methods for solving stochastic constrained reachability problems is given in subsection 5.2.2.

The fact that every syntactically co-safe LTL formula can be represented by a deterministic finite automaton is exploited. A word, i.e., a labeled system trajectory, satisfies an sc-LTL formula if and only if the acceptance condition of the automaton holds. This reduces Problem 36 to computing a control policy that maximizes the probability that the system reaches an accepting state in the automaton. The deterministic finite automaton is modified by including stochastic transitions between modes, which will account for uncertainty due to the stochastic system dynamics.

5.2.1 Dynamic programming

A deterministic finite automaton (DFA) corresponding to the sc-LTL specification is used to guide the computation of a control policy that (conservatively) solves Problem 36. Informally, the modes (i.e., states) in the DFA represent progress towards the completion of the task, and the goal is to compute a control policy that maximizes the probability that the system will reach an accepting mode of the automation from its initial state. However, transitions between modes in the automaton are not deterministic due to the system's stochastic noise. Thus, an MDP corresponding to the DFA is constructed, where the actions correspond to memoryless control policies that are executed by the continuous system. A control policy selects the appropriate action (i.e., memoryless control policy) at each mode in the automaton.

The product MDP $\mathcal{M}$ restricts behaviors to those that satisfy both the system dynamics (5.1) and the deterministic finite automaton $\mathcal{A}_\varphi$ representing the sc-LTL formula φ.

Definition 37. *For a system of the form* (5.1) *with state space X, and a deterministic finite automaton $\mathcal{A}_\varphi = (Q, 2^{AP}, \delta, q_0, F)$, the **product MDP** is given by $\mathcal{M} = (S, A, P, s_0, \mathcal{F})$ with*

- *an infinite set of states $S = X \times Q$,*
- *an infinite set of actions A, and*

- *initial state $s_0 = (x_0, q)$ such that $q = \delta(q_0, L(x_0))$.*

The compact action set A corresponds to all possible memoryless control policies for the original system. Each control policy $a \in A$ induces a transition probability between states in $\mathcal{M}$ as

$$P((x,q),a,(x',q')) = \begin{cases} P(x,a,x') & \text{if } q' = \delta(q, L(x')) \\ 0 & \text{otherwise,} \end{cases}$$

where $P(x,a,x')$ is the probability density function of system (5.1) transitioning from state $x \in X$ to state $x' \in X$ under the memoryless control policy $a \in A$.

The accepting product states $\mathcal{F}$ are lifted directly from F, i.e., state $(x,q) \in \mathcal{F}$ if and only if $q \in F$.

It is shown in [150] that Problem 36 (in discrete time) can be solved by performing dynamic programming on $\mathcal{M}$. Here, the *value function* $V : S \to \mathbb{R}$ maps each product MDP state to a scalar. All accepting states $s \in \mathcal{F}$ are initialized to $V(s) = 1$. All states $s \in \mathcal{F}_0$, where $\mathcal{F}_0$ is the set of all states that cannot reach $\mathcal{F}$, are initialized to $V(s) = 0$. The value function for the remaining states $x \in S \backslash (\mathcal{F} \cup \mathcal{F}_0)$ can then be computed by dynamic programming [150].

Remark 38. *Although an MDP is used to model the automaton, an expensive abstraction of the system itself as an MDP is not performed.*

5.2.2 Stochastic constrained reachability

The *stochastic constrained reachability problem* is now defined, which corresponds to an action in the MDP in Section 5.2.1. A solution to this problem is a control policy that maximizes the probability that the system reaches the boundary of set X_2 before reaching the boundary of set X_1. This stochastic reachability problem always has an optimal memoryless policy [155].

Problem 39 (Stochastic constrained reachability)**.** *Given a system of the form* (5.1) *and sets $X_1, X_2 \subseteq X$, compute a control policy μ^* that maximizes the probability that the system reaches the boundary of set X_2 before reaching the boundary of set X_1.*

Solving a stochastic constrained reachability problem is generally undecidable [156]. However, there exist numerous sound algorithms that compute solutions to constrained reachability problems using PDE-based methods [2, 157, 14, 158].

The standing assumption is made that there exists an oracle for computing a solution to a stochastic constrained reachability problem that under-approximates the probability of reaching the goal set. This method is denoted by CstReach(X_1, X_2), with constraint set X_1 and reach set X_2. For a given query, CstReach returns a memoryless control policy μ^*. A method to solve the CstReach problem is now given with an under-approximation for the case when the stochastic reachability problem has a particularly simple form.

5.3 Composition of Solutions

The superposition principle can be used to construct solutions to arbitrary specifications from the solutions of individual specifications over labeled regions. The method consists of two parts, the calculation of individual solutions for individual propositions in Algorithm 3, and the generation of solutions to a complete reachability problem in Algorithm 4.

The development of Algorithm 3 begins with the observation that the transformed Hamilton-Jacobi-Bellman PDE (1.27) is linear. The importance of linearity in developing solutions to PDEs is well known, and in particular the following theorem will prove useful.

Theorem 40. *(Superposition Principle [94]) Given a pair of partial differential boundary value problems $L(\Psi_i) = 0$, $i = 1, 2$ on Ω, where L is an arbitrary linear differential operator, with boundary conditions $\Psi_1 = f$, $\Psi_2 = g$ on $\partial\Omega$, then $\Psi^* = \Psi_1 + \Psi_2$ is the solution to the boundary value problem with $\Psi^* = f + g$ on $\partial\Omega$.*

Algorithm 3 Pre-processing of constituent solutions

```
Given compact domain $\Omega$, and labeled regions $\mathcal{R} = \{R_i\}$:
  1. **Set** $\partial\Phi = \{\partial\Omega, \{\partial R_i\}\}$
  2. **Set** $\psi_0 |_{\partial\Omega} = C$, null elsewhere
  3. **For each** $R_i \in \mathcal{R}$
      (a) Set $\psi_0 |_{\partial R_i} = 0$
  4. **Solve** for $\Psi_d$ with PDE constraints (1.27) and boundary conditions $\Psi_d |_{\partial\Phi} = e^{-\psi_0}$
  5. **For each** $R_i \in \mathcal{R}$
      (a) **Solve** for $\Psi_p^i$ according to (1.27) with boundary conditions $\Psi_p^i = 1$ on $\partial R_i$, $\Psi_p^i = 0$ on $\partial\Phi \backslash \partial R_i$
  6. **Return** $\left\{\Psi_d, \left\{\Psi_p^i\right\}\right\}$
```

Algorithm 4 Generation of Value function

```
Given solution to Algorithm 3 $\left\{\Psi_d, \left\{\Psi_p^i\right\}\right\}$, and a set of penalties $C_i$ for each labeled region in $\mathcal{R}$:
  1. **Initialize** Value function $\Psi^* := \Psi_d$
  2. **For each** region $R_i \in \mathcal{R}$
      (a) **Set** $\Psi^* := \Psi^* + C_i \cdot \Psi_g^i$
  3. **Return** $\Psi^*$
```

The proposed method is to solve the problem separately for each labeled region. The activation of these regions as either goal regions or obstacles as part of a specification is then accomplished through superposition of the solutions of each individual activated region.

To begin, suppose all labeled partitions as well as the domain are given. First, construct the *default solution* Ψ_d as that for which the boundary of the region is taken into account. The result is the simple boundary value problem

$$\begin{aligned} 0 &= -\frac{1}{\lambda} q \Psi_d + f^T (\nabla_x \Psi_d) + \frac{1}{2} Tr \left((\nabla_{xx} \Psi_d) \Sigma \right), \\ \Psi_d \mid_{\partial \Omega} &= 1, \\ \Psi_d \mid_{\partial \Phi \setminus \partial \Omega} &= 0. \end{aligned} \tag{5.2}$$

The next step is the calculation of a solution primitive Ψ for an individual labeled region R. Recall that such solutions will be added, and it is therefore necessary that the solution not alter the boundary values at other labeled regions. Thus, the boundary conditions for this problem are set to

$$\Psi \mid_{\partial \Phi \setminus \partial R} = 0.$$

Note that these conditions are also set for the domain boundary $\partial \Omega \subset \partial \Phi$. When solutions constructed in this manner are superimposed, the resultant boundary conditions then have the correct values. The approach is illustrated graphically in Figure 5.1.

Remark 41. *Note that composing solution in this manner is not a form of averaging the solution primitives. The principle of superposition dictates that the solution will match exactly the composite problem. Although the composite solution is constructed by simple addition, the transform* (1.26) *induces sophisticated behavior.*

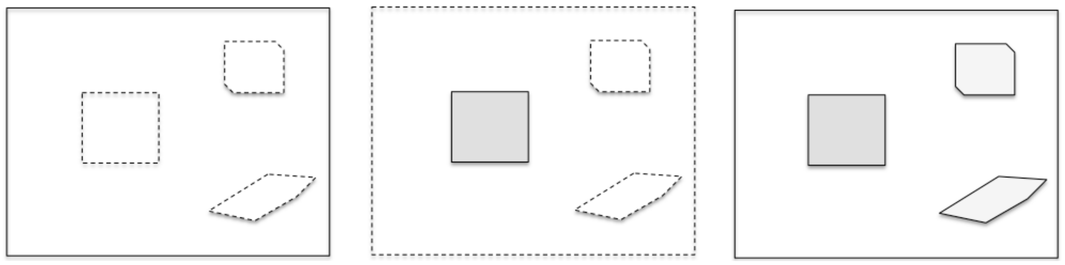


Figure 5.1: Illustration of the construction of a solution primitive. On the left the solution is created for the boundary-only problem with the boundaries of the labeled regions set to zero, indicated by dashed lines. In the middle, the PDE is solved for one region activated while all other regions and boundary set to zero. In the last, a complete solution is shown with the solutions added.

5.3.1 Automata-Guided Task

At each node of the automaton, the specification is reduced to a set of reach-avoid tasks over the set of regions, with their reach or avoid nature indicated by the relevant propositions. Given the output $\mathcal{O}$ of Algorithm 3, Algorithm 4 generates the Value function for the current task specified by the node. The reach-avoid regions are selected, and appropriately scaled by some constant C_i. For each region these scalings may be collected into a vector of coefficients $\mathcal{E} = \{C_i\}_{i=1,\ldots,|\mathcal{R}|}$. The solutions $\Psi = \{\Psi_i\}$ to these individual solutions are then added and scaled appropriately to produce the desired solution

$$\Psi^* = \mathcal{E}^T \cdot \Psi$$

Once the goal region is reached, the current node on the graph is transitioned according to which goal was achieved, and the process repeats.

5.3.2 Dynamic Programming

As the completion of the specification of Problem 36 involves multiple individual steps, each corresponding to Problem 39, future goals must be weighted according to their future reward when planning in the current time step. By Bellman's Principle of Optimality, this suggests that at stage i the goals should be weighted according to their cost-to-go when beginning stage $i+1$. By proceeding from the accepting state, it is therefore possible to perform dynamic programming for this problem by setting the boundary conditions, i.e., the vector $\mathcal{E}$, for precedent problems according to their subsequent cost-to-go.

Dynamic programming in this setting therefore proceeds as follows. The Deterministic Finite Automaton representing the temporal specification is constructed, and the final accepting states are selected. Each final-stage stochastic reachability problem is then solved using Algorithm 4, with only the final goal regions active. The cost-to-go at the other labeled regions are then used as boundary conditions at the previous stage. The process is then repeated until the initial conditions are reached.

5.3.3 Temporal Boundary Conservatism

Several simplifying assumptions are necessary in the creation of the superposition framework that introduce a degree of conservatism. The first of these is that the boundary conditions remain consistent for all reachability problems, requiring that boundary conditions for all regions be prescribed. Unfortunately, this prevents the elimination of some regions, for example if an obstacle is not necessarily to be avoided at some stages of the task. A straightforward method to fix this issue is to penalize the boundary conditions until their desirability achieves a lower value than that of the surrounding domain, i.e., such that $\frac{\partial \Psi}{\partial n} |_{O_i} \geq 0$ for n a normal vector on the surface of O_i. The effect is that the desirability of reaching this deactivated region is lower than it would otherwise be. Although this will result in an approximation error, the approach only overstates the expected cost of reaching the goal, resulting in a unnecessarily conservative but sound policy.

A second caveat is that the efficient construction of value functions also requires that the boundary of each region be weighted according to a constant value. To facilitate this approach, weight each region $\mathcal{R}$ according to its *worst-case* cost-to-go, that is, the lowest desirability value along the region boundary. This again results in a conservative but sound policy, with the solution treating some boundary values with a higher cost than is truly the case.

A last difficulty arises in that it is impossible to calculate a priori the exact cost-to-go from the boundary of the previous stage region. This is prevented by Steps 2 and 3 of Algorithm 4, which prescribe boundary conditions along all labeled regions, preventing us from calculating what the cost-to-go would be if that region was not present. It is possible to raise the boundary condition from the region whose cost-to-go is to be approximated until it is greater than all neighboring states, i.e., such that the gradient away from the region is negative. This implies that the cost to go from the boundary of the region is greater than it would have been otherwise, and provides, again, an under approximation of the satisfaction probability.

5.3.4 Introduction of New Propositions

Once Algorithm 3 has been completed, it is still possible to add a newly created region to the problem. The need could arise from varying requirements over the course of execution, due to perhaps the introduction of a previously unseen obstacle, or if it desirable to completely eliminate a region from further consideration.

Given an existing specification φ with solution Ψ, there may be a need to add a proposition a to an unlabeled region A in the domain (A may have no intersection with existing labeled reigons). The value of the solution must be adjusted to match $\phi(x) \mid_{\partial A}= C$. The existing desirability is captured from the boundary ∂A, and the boundary value problem is solved with boundary conditions

$$\begin{aligned} \phi \mid_{\partial A} &= \Phi(x), \\ \phi \mid_{\partial\Phi\setminus\partial A} &= 0, \end{aligned}$$

with solution denoted Ψ_e. This captures the effects of the existing solution along ∂A upon the rest of the solution. The process is repeated with the new boundary conditions

$$\begin{aligned} \phi \mid_{\partial A} &= 1, \\ \phi \mid_{\partial\Phi\setminus\partial A} &= 0, \end{aligned}$$

Algorithm 5 Patching of value function with the introduction of a new region.

```
Given primitive set $\{\Psi_d, \{\Psi_p^i\}\}$ on domain $\Phi$ with boundaries $\partial\Phi$, task $\beta$, and new region $R_n$:
  1. Compose solution $\Psi_o$ to task $\beta$ without region $R_a$ using Algorithm 4
  2. Set $\psi_s := \Psi_o \mid_{\partial R_a}$
  3. Solve (1.27) for solution $\Psi_e$ with boundary conditions $\Psi_e \mid_{\partial R_a} = \psi_s$, $\Psi_e \mid_{\partial\Phi \backslash R_a} = 0$
  4. Solve (1.27) for solution $\Psi_n$ with boundary conditions $\Psi_n \mid_{\partial R_a} = 1$, $\Psi_n \mid_{\partial\Phi \backslash R_a} = 0$
  5. Construct new solution $\Psi^* = \Psi + (\gamma\Psi_n - \Psi_e)$ where $\gamma = C$
  6. Return $\Psi^*$
```

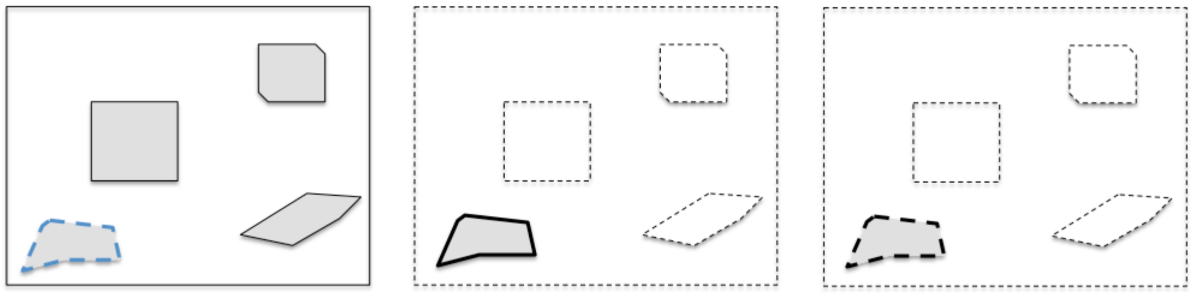


Figure 5.2: Visualization of Algorithm 5. On the left, the solution for a given task is extracted along the boundary of the new region. In the middle, the solution is generated for the desired boundary conditions along the new region while all others are set to zero. In the last, the boundary is set to its desired value while all other boundaries are set to zero.

producing solution Ψ_n. The new solution with the added boundary conditions may then be constructed. Note however that this new solution is particular to the boundary conditions prescribed on $\partial\Phi$, and cannot be scaled and used as part of the superposition framework.

5.3.5 Complexity

The primary cost in this framework is the calculation of the solution to the PDE (1.27), which must be computed once for each element of $\mathcal{R}$, and once more for the boundaries of all elements $\mathcal{R}$ set to zero. The Finite Difference Method has complexity $\mathcal{O}\left(\left(\frac{1}{h}\right)^d\right)$ for discretization length h and state space dimension d. Methods based on

Monte Carlo sampling, such as Feynman-Kac, are known to have accuracy that scale independently of state space dimension at $\mathcal{O}(n^{\frac{1}{2}})$ where n are the number of samples used in the estimation process. More recently, methods to directly solve linear PDEs that scale linearly with dimension have also appeared [120].

The computation of the deterministic finite automaton for the specification is worst-case doubly exponential in the size of the specification [159]. The convergence of the value iteration for the specification MDP is guaranteed as the continuum of actions exists in a compact set, and the costs of the actions are non-negative [155]. Investigation of the specific convergence rate in this context is the subject of future research.

The main advantage of the framework presented lies in the composition of solutions. At each step in the automaton it is necessary to calculate the solution to a reach-avoid problem. This is done through simple vector addition of the primitives in Algorithm 4. As all primitives are used, this is an $\mathcal{O}\left(|\mathcal{R}|\right)$ operation, but with a quite small leading constant as vector addition is computationally negligible. Denoting the method of calculating an individual PDE solution as having complexity $\mathcal{O}(p)$, Algorithm 3 requires $\mathcal{O}\left(|\mathcal{R}|\, p\right)$ time.

5.4 Temporal Examples

The approach is illustrated on two examples. The first example illustrates the method, and the second example shows how the compositional approach scales with task complexity. For simplicity, the finite difference method for solving the HJB PDE is used. The standard approach to such problems relies on a discrete abstraction with transition probabilities gained from Monte Carlo simulation [145]. These simulations create a high computational burden, taking as much as several hours, as well as approximation error in the model. These issues are avoided in the approach, with computation time in the tens of seconds. In both of these examples a nonlinear two dimensional example is used, as it facilitates visualization, but also demonstrates the generality of the method.

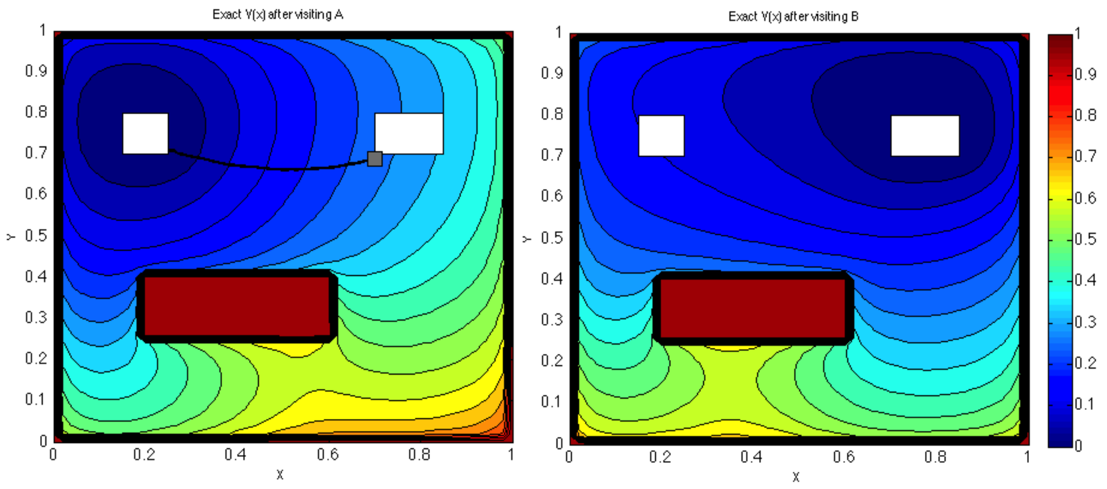


Figure 5.3: Results for last stage cost-to-go of Example 1. The domain is $x, y \in [0,1]^2 \backslash C$ for $C = \{x \in [0.2, 0.6], y \in [0.25, 0.4]\}$. The goals are $A = \{x \in [0.7, 0.85], y \in [0.7, 0.8]\}$, $B = \{x \in [0.7, 0.8], y \in [0.15, 0.25]\}$. A trajectory of the closed loop system begins at the grey square.

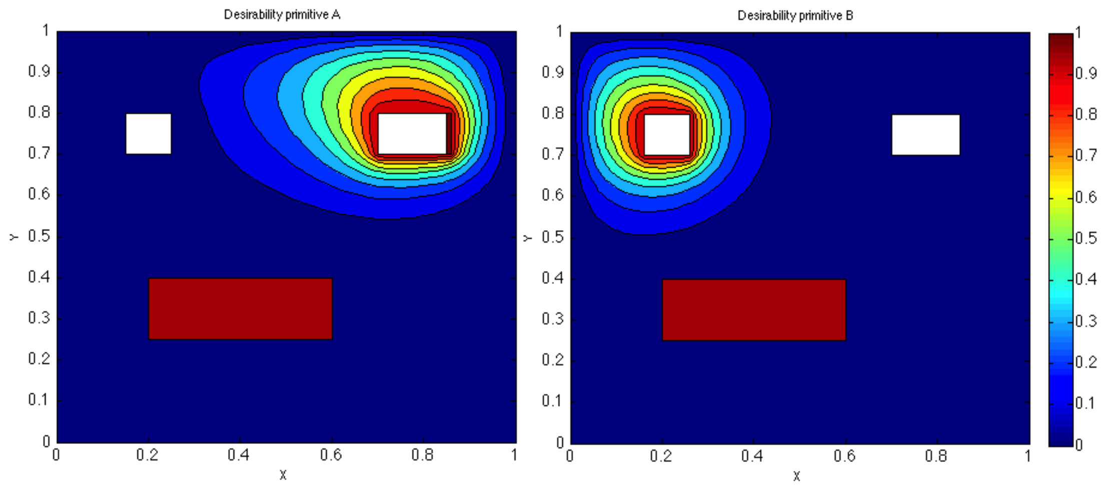


Figure 5.4: Individual desirability primitives for regions A, B on the left and right respectively.

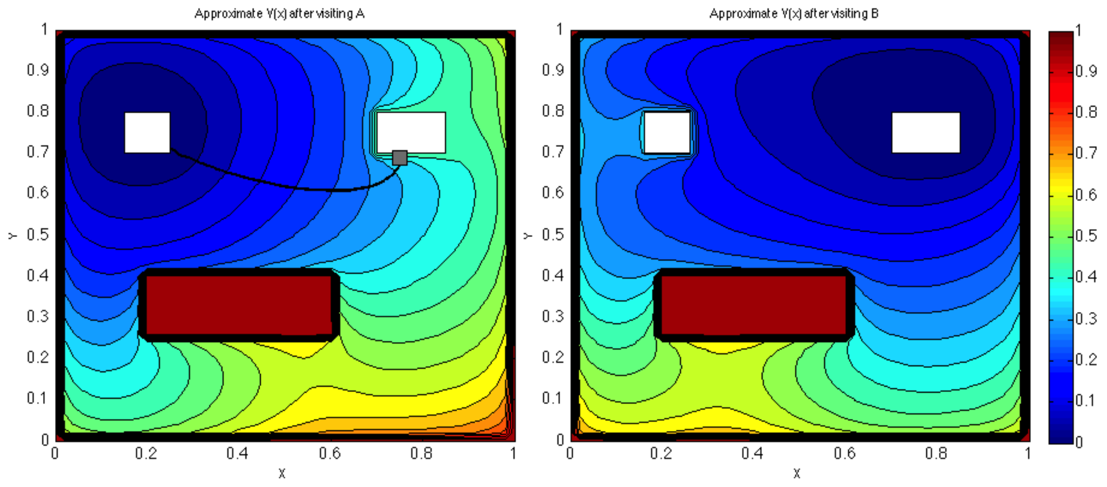


Figure 5.5: Results of using composition method on the two region visit task of Example 1.

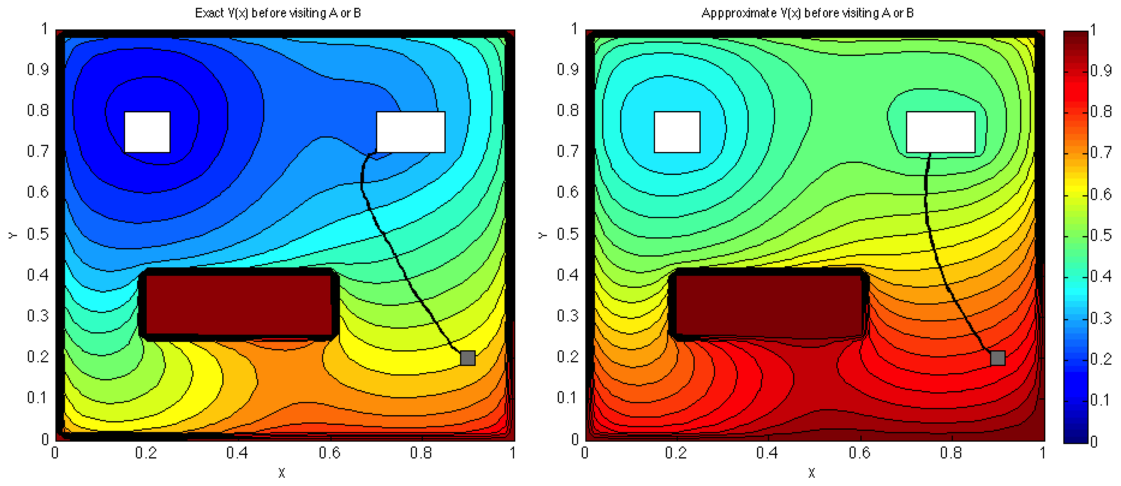


Figure 5.6: Exact and approximate cost-to-go before either region is visited on the left and right respectively. A trajectory, beginning from the grey square, is shown in black when following the induced policies. After visiting A the trajectories are continued in Fig. 5.3, 5.5.

5.4.1 Two Goal Temporal Example

In the first example, a mobile agent is tasked to visit two goal regions A and B while remaining in an obstacle-free bounded domain S. The formal specification is

$$\varphi = \Diamond A \wedge \Diamond B \wedge \Box S$$

and the system dynamics, taken from [160] are given by

$$\begin{aligned} \begin{bmatrix} dx \\ dy \end{bmatrix} &= \left(\begin{bmatrix} -2x - x^3 - 5y - y^3 \\ 6x + x^3 - 3y - y^3 \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) dt \\ &+ \begin{bmatrix} d\omega_1 \\ d\omega_2 \end{bmatrix} \end{aligned}$$

The state cost is set to $q = 0.4$ with control penalty is $R = 0.05 I_{2\times 2}$. In Figure 5.3 the geometry is shown along with the cost-to-go in the last stage of the automaton after having visited one of the goals. The value function for the first stage is calculated and shown in Figure 5.6a.

The problem is then repeated using the compositional approach. The primitives are shown in Figure 5.4. These are superimposed to produce the solution from the last stage, shown in Figure 5.5, where the worst-case cost-to-go is applied to the last visited region. These worst-case values are then used as the boundary conditions for the first-stage value function, shown in Figure 5.6b.

5.4.2 $n-$Goal Temporal Example

The previous example is expanded upon, now scaling the number of goal regions n up to ten in a larger domain $\Omega = [0, 130]^2$ with discretization size $h = 1.0$. The goals correspond to labeled regions R_i with width 4, and are equally spaced in the domain Ω. Specifically, the specification is

$$\varphi = \Diamond A_1 \wedge \Diamond A_2 \wedge \ldots \wedge \Diamond A_n \wedge \Box S.$$

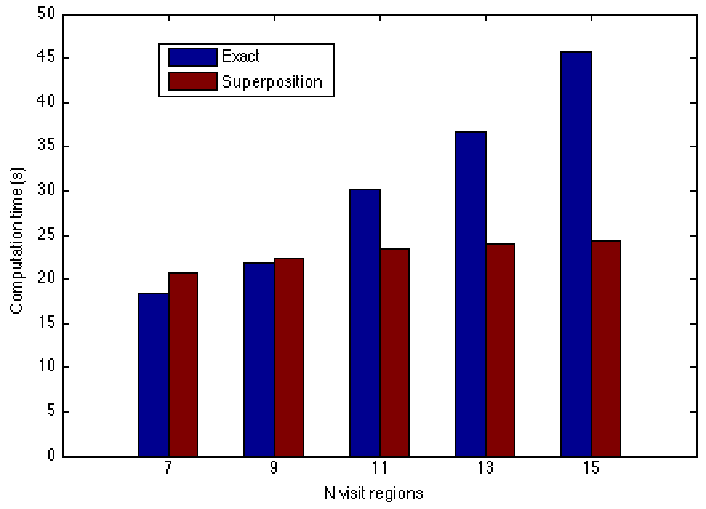


Figure 5.7: Calculation times using exact and superposition method for visitation problem with n regions to visit.

The problem can be solved exactly using the method of superposition as before. As the exact approach requires the solution of a PDE for each edge of the automaton, the computation also scales with the size of the automaton. In contrast, using superposition the policy at each automaton node requires only vector additions and a maximization operation over a vector that describes the region boundaries. Calculation times for the two are shown in Figure 5.7.

5.5 Conclusions

A method for efficiently synthesizing control policies for stochastic nonlinear systems with syntactically co-safe LTL specifications was introduced. The method was developed for generic systems, but when a certain structural assumption was made on the noise entering the system an algorithm with significant computational gains was introduced. The method relies on pre-computed primitives that relate the solution to a stochastic optimal control problem to the reachability of an individual region. Solutions to individual reach-avoid tasks are cheaply constructed by simple vector addition of these pre-computed solutions, allowing for the individual stages of a task specified by an automaton to be solved quickly. The method relies on the state space of the original system and requires no a-priori discretization or cellular decomposition.

The drawbacks of this method are that all boundary conditions for the associated PDE must be specified, making it difficult to treat a labeled region as neither a goal or an obstacle. A region may be removed exactly, but this requires the solution of an additional PDE boundary value problem, and is not applicable when any of the other boundary conditions are changed. Instead, it is possible to construct a sound but conservative solution with mild penalty on the inactive regions. While inexact, the method has many benefits, among which is the ability to rapidly adapt to new specifications over the existing labeled regions in real time.

Chapter 6

Implications and Future Work

The methods and results presented in this thesis can be potentially applied to problems beyond those directly addressed in the previous chapters. Indeed, the ability to compute solutions to high dimensional problems has far-reaching implications. Chief among these is the ability to solve general path planning controllers, along with corresponding optimal feedback controllers, for complex, realistic problems. However, other future directions of possible interest are addressed here before concluding.

A common task in control theory isn't the generation of trajectories, but instead the stabilization of a system around an equilibrium point. In these problems it is desirable to not only design these controllers, but also to *prove* their stabilization properties. This is the emphasis of Lyapunov theory, and extensions to this framework are discussed that can make such analysis possible. Other variants of these problems are discussed, including Markov decision processes as well as the Hamilton Jacobi Bellman equation when the structural assumptions for linearity do not hold.

6.1 Control Lyapunov Functions

Recall the review of Control Lyapunov Functions from Section 1.3.3, wherein the definition of a Lyapunov function (1.15) is augmented to allow for energy dissipation with the assistance of a control signal. For a system defined as in Section 1.4

$$dx_t = \left(f\left(x_t\right) + G\left(x_t\right)u_t\right)dt + B\left(x_t\right)Ld\omega_t \tag{6.1}$$

the requirement for a function $V(x)$ to be a Control Lyapunov Function (CLF) is

$$\inf_u \left[\nabla_x V \cdot f(x) + \nabla_x V \cdot G(x)u\right] < 0. \tag{6.2}$$

If it is assumed that the Lyapunov function candidate is a solution to the Hamilton Jacobi Bellman equation, then

$$\begin{aligned}
\nabla_x V\left[f + Gu^*\right] &= \left(\nabla_x V\right)^T f + \left(\nabla_x V\right)^T G R^{-1} G^T \left(\nabla_x V\right) && (6.3)\\
&= -q - \frac{1}{2}\left(\nabla_x V\right)^T G R^{-1} G^T \left(\nabla_x V\right) && (6.4)\\
&\quad -\frac{1}{2} Tr\left(\left(\nabla_{xx} V\right) B \Sigma_\epsilon B^T\right).
\end{aligned}$$

It is immediately apparent that the Hamilton Jacobi Bellman solution is in fact a CLF. In particular, if stabilization is the goal, then the control problem domain, either compact or unbounded, has an internal boundary condition at the equilibrium point, typically taken to be the origin by change of coordinates.

If the exact solution is unavailable, then it may be seen that a sufficient condition for an approximate value function be a CLF is for the Hessian term to have the constraint

$$\nabla_{xx} V \preceq 0. \tag{6.5}$$

In turn, returning to the logarithmic transformation of Equation (1.26), Eq. (6.5) takes the form:

$$\nabla_{xx} V = \frac{\lambda}{\Psi^2}\left(\nabla_x \Psi\right)\left(\nabla_x \Psi\right)^T - \frac{\lambda}{\Psi} \nabla_{xx} \Psi.$$

Implying that this sufficient condition may be enforced by requiring

$$\nabla_{xx} \Psi \succeq 0. \tag{6.6}$$

Thus, if a near-exact solution is available, from the high dimensional numerical technique presented here or elsewhere, then a CLF has also been calculated. As well, if an approximate solution is sufficient, then the polynomial optimization techniques are appropriate. In practice, it was found that internal boundary conditions became difficult to incorporate for high dimensions for the high dimensional technique. These situations are likely ameliorated with the use of different discretization techniques beyond the high order finite difference techniques used in this work. In particular, the use of methods that directly deal with this problem, such as the Finite Increment Calculus [161], Essentially Non-Oscillatory (ENO) [162], or Discontinuous Galerkin schemes promise to allow for the well-conditioned inclusion of interior boundaries, including that of an exit point located at an equilibrium point, a necessity for stabilization tasks and Control Lyapunov examples.

6.2 Sum of Squares Programming in High Dimensions

The two central approaches presented in this work each have distinct advantages. The sum of squares-based approach developed in Chapter 2 generates a degree of guarantee, certifying the pointwise distance from optimal for any suboptimal solution generated. In turn, the high dimensional numerical technique of Chapter 3 has computational advantages, but for any given discretization level, little is known about its suboptimality. The synthesis of these two techniques is therefore an exciting research direction.

Key to the development of the separated representation approach is the division of computation into operations involving single dimensions. Once complete, the solution is a high dimensional tensor composed of a summation of rank-1 tensors, each individually composed of vectors in single dimensions, i.e., from (3.1)

$$f\left(x_1, \ldots, x_d\right) \approx \sum_{l=1}^{r} s_l \phi_1^l(x_1) \cdots \phi_d^l(x_d). \tag{6.7}$$

An observation is that in the continuous domain, each rank-1 term is simply a monomial, and the separated representation has simply given a method to calculate a polynomial solution with few additive terms. Indeed, if the solution is smooth, it may be possible to return to the continuous domain by simply projecting the discretized solution onto a monomial basis. For solution

$$\boldsymbol{F} \approx \sum_{l=1}^{r_{\mathbf{F}}} s_l \bigotimes_{i=1}^{d} \boldsymbol{F}_i^l \tag{6.8}$$

each term may be approximated as the one that solves

$$\min_{\phi_i^l(x)} \|\phi_i^l(x) - \tilde{\boldsymbol{F}}_i^l\|_{\ell_\infty} \tag{6.9}$$

where $\tilde{\boldsymbol{F}}$ is the step-wise approximation to $\boldsymbol{F}$. Such a problem may be solved easily using sum of squares programming [163].

The resulting sparse solution basis may then be used in the sum of squares framework of Section 2.1, where now the parameterization of the candidate solution Ψ no longer requires a full monomial basis, which grows factorially with dimension. Instead, if a low separation rank is available, the monomial basis will also have low size, and the problem made tractable as sums of squares problems have complexity that grow with the number of monomial basis [36].

This approach to generating a sparse monomial basis for polynomial optimization problems differs from the central approach in the literature. Previously, existing methods have focused on simplifying the monomial basis a priori from problem data. These approaches have included a focus on sparsity in the problem data [164], a focus on Newton polytopes of the data, allowing for monomial basis guaranteed to be unused to be discarded [111, 163], or more generally through a technique that generated a simplified representation of a face of the SOS cone via linear programming [111]. Instead, the approach is *data-driven*, and uses a numerical technique to generate a sparse basis *a-posteriori* of solving a version of the problem. It is therefore a computationally intensive technique, but serves as the only current method to perform polynomial optimization in high dimensions.

6.3 Linear Markov Decision Processes in High Dimensions

The Hamilton Jacobi Bellman equation has an analogue in discrete state spaces, that of linear MDPs, detailed in Section 1.5.1. In the past these techniques have been favored for computation of the global solution due to their better numerical behavior in practice. In the MDP setting, linearity corresponds to linear sets of equations being sufficient to solve for the value function, without recourse to value or policy iteration. This linear MDP framework gives additional design degrees of freedom, allowing for continuity not to be required in the transition system, and for better behavior around internal boundary conditions. It is therefore appealing to extend the separated representation framework to the linear MDP setting.

6.4 General Hamilton Jacobi Bellman Equations

The focus of this work has been on the development of techniques that take advantage of the linearity. This is a key limitation of this work in that it requires the structural assumptions of (1.25) to obtain a linear set of equations for which ALS may be applied or for polynomial candidate solutions to be applied. The general nonlinear value function may not be directly solved. However, it has been shown that iterative linearization of the nonlinear equations may be constructed in such a manner as to solve the more general Hamilton Jacobi Bellman problem without these structural assumptions [137]. Thus, although the computational effort of the algorithms presented in this thesis would necessarily grow, this work may offer an avenue through which to tackle the general problem.

6.5 Software

The work presented in this thesis contained a number of computational algorithms. This substantial code base is currently under development and will be released soon under an academic license, in addition to a publication detailing its use. It is hoped that the ability to solve the Hamilton Jacobi Bellman equation in high dimensions will prove a valuable asset to researchers in the future. A software package which includes the algorithms developed in this thesis will be available at the authors webpage `http://www.matanyahorowitz.com/hd_soc.html`.

6.6 Conclusion

The spirit of this work has been the development of tractable, theoretically grounded computational techniques for the synthesis of solutions to the Hamilton Jacobi Bellman Equations. As the work draws to a close, the contributions of this thesis are summarized below.

Semidefinite Programming for Stochastic Optimal Control

In Chapter 2 the linear Hamilton Jacobi Bellman equation was relaxed to a set of linear differential inequalities. This relaxation led to a semidefinite programming problem to approximately solve optimal control problems for stochastic, nonlinear systems. Theoretical arguments supported the relaxation used, demonstrating that the relaxed solutions were in fact pointwise upper and lower bounds, and that a series of solutions with decreasing gap could be produced through a hierarchy of relaxations.

Section 2.2 built upon these results, improving the accuracy of the method via domain partitioning. Partitioning was shown to have significant computational benefits, allowing for the optimization problem to be significantly parallelized. These methods were also shown to be well motivated theoretically, producing improving upper and lower bounds. The method was shown to be widely applicable to elliptic and parabolic problems in Section 2.3, allowing for uncertainties in the problem data to be incorporated, crucial for Uncertainty Quantification Problems.

High Dimensional Optimal Control

Chapter 3 introduced the Separated Representation for the solution to the linear Hamilton Jacobi Bellman equation, and it was shown that this technique is particularly well suited for optimal control problems. Numerical methods were developed and applied to several problems that had been, until this point, impractically large to be solved exactly. A twelve dimensional quadcopter stabilization problem was solved on the order of minutes with the framework.

Navigation Functions

Chapter 4 applied the linear Hamilton Jacobi Bellman equation to the study of particular classes of problems in the robotics literature. The first of these were Navigation Functions. In light of the Hamilton Jacobi Bellman equation, it was shown how techniques developed previously could be related to the optimal control theory presented in this thesis. It was demonstrated that these techniques based on the Laplace Equation could in fact be shown to be optimal for a certain criteria if they were simply altered via a logarithmic transform.

Temporal Problems

Chapter 5 analyzed the problem of Linear Temporal Logic-based planning. There, a dynamic programming argument demonstrated that the optimal path through multiple temporal sub-tasks could be linked via boundary conditions in the Hamilton Jacobi Bellman equations of distinct optimal control problems. The linearity of the particular Hamilton Jacobi Bellman equation under study was then leveraged to precalculate solution *primitives* which could be combined at runtime at essentially zero cost. This led to a divorce between the size of the automaton generated by LTL specifications and computational effort.

Implications

Chapter 6 considered the possible significance of the results in this thesis beyond those presented. A number of related problems are significantly effected by the development presented. In particular, the promise of developing solutions to the general Hamilton Jacobi Bellman equation, detailed in Section 6.4, would eliminate the structural assumptions made throughout this work. Section 6.1 also detailed the impact of the methods throughout this work on the development of Control Lyapunov Functions in practice.

6.6.1 Final Thoughts

As emphasized in the introduction of the work, the key to solving the difficult problems in the realm of control theory will be the development of techniques that drastically alter the computational considerations of these problems. Techniques such as those presented here hint at the feasibility of rapidly computable planning for robotics, automated systems, and artificial intelligence.

Bibliography

[1] President's Council of Advisors on Science and Technology, "Report to the president and congress - designing a digital future: Federally funded research and development in networking and information technology," tech. rep., Executive Office of the President, 2010.

[2] M. B. Horowitz and J. W. Burdick, "Semidefinite relaxations for stochastic optimal control policies," *American Controls Conference (ACC)*, 2014, arXiv:1402.2763v1.

[3] M. B. Horowitz, I. Papusha, and J. W. Burdick, "Domain decomposition for stochastic optimal control," in *IEEE Conference on Decision and Control*, 2014 (Submitted).

[4] M. B. Horowitz, A. Damle, and J. W. Burdick, "Linear Hamilton Jacobi Bellman equations in high dimensions," *arXiv*, 2014, 1404.1089v1.

[5] M. B. Horowitz and J. W. Burdick, "Optimal Navigation Functions for Stochastic Systems," in *International Conference on Intelligent Robots and Systems (IROS)*, 2014 (Accepted).

[6] M. B. Horowitz, E. M. Wolff, and R. M. Murray, "A compositional approach to stochastic optimal control with temporal logic specifications," *International Conference on Intelligent Robots and Systems (IROS)*, 2014 (Submitted).

[7] R. Bellman, *Dynamic Programming.* Dover Books on Computer Science, 1957. Reprint 2003.

[8] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I. Athena Scientific, 3rd ed., 2005.

[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. II. Athena Scientific, 4 ed., Mar. 2012.

[10] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*, vol. 25. New York: Springer, July 2006.

[11] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *International Journal of Control*, vol. 73, no. 11, pp. 1001–1025, 2000.

[12] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[13] I. M. Mitchell and C. J. Tomlin, "Overapproximating Reachable Sets by Hamilton-Jacobi Projections," *Journal of Scientific Computing*, vol. 19, no. 1-3, pp. 323–346, 2003.

[14] C. J. Tomlin, I. M. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proc. IEEE*, vol. 91, pp. 986–1001, 2003.

[15] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *Automatic Control, IEEE Transactions on*, vol. 50, no. 7, pp. 947–957, 2005.

[16] E. Al'Brekht, "On the optimal stabilization of nonlinear systems," *Journal of Applied Mathematics and Mechanics*, vol. 25, no. 5, pp. 1254–1266, 1961.

[17] B. Spencer Jr, T. Timlin, M. Sain, and S. Dyke, "Series solution of a class of nonlinear optimal regulators," *Journal of Optimization Theory and Applications*, vol. 91, no. 2, pp. 321–345, 1996.

[18] C. Navasca and A. J. Krener, "Patchy solutions of Hamilton-Jacobi-Bellman partial differential equations," in *Modeling, estimation and control*, pp. 251–270, Springer, 2007.

[19] F. Ancona and A. Bressan, "Patchy vector fields and asymptotic stabilization," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 4, pp. 445–471, 1999.

[20] C. O. Aguilar and A. J. Krener, "High-order numerical solutions to Bellman's equation of optimal control," in *American Control Conference (ACC), 2012*, pp. 1832–1837, 2012.

[21] C. O. Aguilar and A. J. Krener, "Numerical Solutions to the Bellman Equation of Optimal Control," *Journal of optimization theory and applications*, Sept. 2013.

[22] W. M. McEneaney, *Max-plus methods for nonlinear control and estimation.* Springer, 2006.

[23] W. H. Fleming and W. M. McEneaney, "A max-plus-based algorithm for a Hamilton–Jacobi–Bellman equation of nonlinear filtering," *SIAM Journal on Control and Optimization*, vol. 38, no. 3, pp. 683–710, 2000.

[24] W. M. McEneaney, "A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs," *SIAM Journal on Control and Optimization*, vol. 46, no. 4, pp. 1239–1276, 2007.

[25] B. Anderson and J. B. Moore, *Optimal control: linear quadratic methods.* Prentice-Hall, Inc., 1990.

[26] H. K. Khalil and J. Grizzle, *Nonlinear systems*, vol. 3. Prentice Hall, Upper Saddle River, 2002.

[27] E. F. Camacho and C. B. Alba, *Model predictive control.* Springer, 2013.

[28] A. Jadbabaie, J. Yu, and J. Hauser, “Stabilizing receding horizon control of nonlinear systems: a control Lyapunov function approach,” in *American Control Conference, 1999. Proceedings of the 1999*, vol. 3, pp. 1535–1539, IEEE, 1999.

[29] W. B. Dunbar and R. M. Murray, “Distributed receding horizon control for multi-vehicle formation stabilization,” *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.

[30] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267–278, 2010.

[31] M. V. Kothare, V. Balakrishnan, and M. Morari, “Robust constrained model predictive control using linear matrix inequalities,” *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[32] T. M. Caldwell and T. D. Murphey, “Switching mode generation and optimal estimation with application to skid-steering,” *Automatica*, vol. 47, no. 1, pp. 50–64, 2011.

[33] M. B. Horowitz and J. W. Burdick, “Combined grasp and manipulation planning as a trajectory optimization problem,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 584–591, 2012.

[34] I. Mordatch, Z. Popović, and E. Todorov, “Contact-invariant optimization for hand manipulation,” in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 137–144, Eurographics Association, 2012.

[35] T. Erez and E. Todorov, “Trajectory optimization for domains with contacts using inverse dynamics,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4914–4919, IEEE, 2012.

[36] P. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology, 2000.

[37] A. Papachristodoulou and S. Prajna, "Analysis of non-polynomial systems using the sum of squares decomposition," in *Positive Polynomials in Control*, pp. 23–43, Springer, 2005.

[38] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design.* John Wiley & Sons, Inc., 1995.

[39] R. A. Freeman and J. A. Primbs, "Control lyapunov functions: new ideas from an old source," in *Proceedings of the 35th IEEE Conference on Decision and Control*, vol. 4, pp. 3926–3931, 1996.

[40] E. D. Sontag, "A universal construction of Artstein's theorem on nonlinear stabilization," *Systems & Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.

[41] R. Freeman and P. Kokotovic, "Optimal nonlinear controllers for feedback linearizable systems," in *American Control Conference (ACC)*, vol. 4, pp. 2722–2726, 1995.

[42] E. D. Sontag, "A Lyapunov-Like Characterization of Asymptotic Controllability," *SIAM Journal on Control and Optimization*, vol. 21, pp. 462–471, May 1983.

[43] J. A. Primbs, V. Nevistić, and J. C. Doyle, "Nonlinear optimal control: A control Lyapunov function and receding horizon perspective," *Asian Journal of Control*, vol. 1, no. 1, pp. 14–24, 1999.

[44] J. B. Lasserre, D. Henrion, C. Prieur, and E. Trélat, "Nonlinear optimal control via occupation measures and LMI-relaxations," *SIAM Journal on Control and Optimization*, vol. 47, no. 4, pp. 1643–1666, 2008.

[45] M. Claeys, D. Arzelier, D. Henrion, and J. B. Lasserre, "Measures and LMI for impulsive optimal control with applications to space rendezvous problems," in *American Control Conference (ACC)*, pp. 161–166, 2012.

[46] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817, 2001.

[47] H. Waki, S. Kim, M. Kojima, and M. Muramatsu, "Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 218–242, 2006.

[48] J. B. Lasserre, *Moments, Positive Polynomials and Their Applications*, vol. 1. World Scientific, 2009.

[49] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *IEEE International Conference on Robotics and Automation*, pp. 1–6, 1987.

[50] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, pp. 412–442, Dec. 1990.

[51] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[52] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, pp. 500–505, 1985.

[53] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems*, pp. 299–308, Springer, 2002.

[54] V. J. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 5, pp. 1058–1069, 1990.

[55] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *IEEE International Conference on Robotics and Automation*, pp. 1398–1404, 1991.

[56] S. M. Lavalle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, Citeseer, 2000.

[57] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[58] N. Hudson, J. Ma, P. Hebert, A. Jain, M. Bajracharya, T. Allen, R. Sharan, M. Horowitz, C. Kuo, T. Howard, L. Matthies, P. Backes, and J. Burdick, "Model-based autonomous system for performing dexterous, human-level manipulation tasks," *Autonomous Robots*, vol. 36, no. 1-2, pp. 31–49, 2014.

[59] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems (RSS)*, 2013.

[60] K. Shankar and J. W. Burdick, "Kinematics and methods for combined quasi-static stance/reach planning in multi-limbed robots," in *International Conference on Robotics and Automation (ICRA) (To appear)*, IEEE, 2014.

[61] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, pp. P11011–P11011, Nov. 2005.

[62] H. Kappen, "Linear Theory for Control of Nonlinear Stochastic Systems," *Physical Review Letters*, vol. 95, Nov. 2005.

[63] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 106, no. 28, pp. 11478–11483, 2009.

[64] E. Theodorou, *Iterative path integral stochastic optimal control: Theory and applications to motor control.* PhD thesis, University of Southern California, June 2011.

[65] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and KL control," in *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, pp. 1466–1473, 2012.

[66] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement Learning With Sequences of Motion Primitives for Robust Manipulation," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[67] K. Dvijotham and E. Todorov, "A unified theory of linearly solvable optimal control," *Artificial Intelligence (UAI)*, p. 1, 2011.

[68] B. van den Broek, W. Wiegerinck, and B. Kappen, "Graphical model inference in optimal control of stochastic multi-agent systems.," *Journal of Artificial Intelligence Research (JAIR)*, vol. 32, pp. 95–122, 2008.

[69] W. Wiegerinck, B. v. d. Broek, and H. Kappen, "Stochastic optimal control in continuous space-time multi-agent systems," *arXiv preprint arXiv:1206.6866*, 2012.

[70] E. Todorov, "Linearly-solvable Markov decision problems," *Advances in Neural Information Processing*, pp. 1369–1376, 2006.

[71] J. Yong, "Relations among ODEs, PDEs, FSDEs, BSDEs, and FBSDEs," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 3, pp. 2779–2784, 1997.

[72] X. Mao, *Stochastic differential equations and applications.* Elsevier, 2007.

[73] E. A. Theodorou, J. Buchli, and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[74] S. Schaal, “Dynamic movement primitives — a framework for motor control in humans and humanoid robotics,” in *Adaptive Motion of Animals and Machines*, pp. 261–280, Springer, 2006.

[75] E. Theodorou, J. Buchli, and S. Schaal, “Reinforcement learning of motor skills in high dimensions: A path integral approach,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2397–2403, IEEE, Mar. 2010.

[76] D. P. de Farias and B. Van Roy, “The linear programming approach to approximate dynamic programming,” *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[77] D. P. de Farias and B. Van Roy, “On Constraint Sampling in the Linear Programming Approach to Approximate Dynamic Programming,” *Mathematics of Operations Research*, vol. 29, pp. 462–478, Aug. 2004.

[78] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *arXiv.org*, Dec. 2013, 1312.5602v1.

[79] M. B. Horowitz, “On the structure of decentralized controllers in networked MDPs,” in *52nd IEEE Conference on Decision and Control*, 2013.

[80] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, “Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings,” in *International Joint Conferences on Artificial Intelligence*, pp. 705–711, 2003.

[81] M. Horowitz and J. Burdick, “Interactive non-prehensile manipulation for grasping via POMDPs,” in *International Conference on Robotics and Automation (ICRA)*, pp. 3257–3264, 2013.

[82] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[83] E. Todorov, "Compositionality of optimal control laws," in *Advances in Neural Information Processing Systems*, pp. 1856–1864, 2009.

[84] E. Todorov, "Eigenfunction approximation methods for linearly-solvable optimal control problems," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 161–168, 2009.

[85] M. Zhong and E. Todorov, "Moving least-squares approximations for linearly-solvable stochastic optimal control problems," *Journal of Control Theory and Applications*, vol. 9, pp. 451–463, July 2011.

[86] P. A. Parrilo and S. Lall, "Semidefinite programming relaxations and algebraic optimization in control," *European Journal of Control*, vol. 9, no. 2, pp. 307–321, 2003.

[87] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming*, vol. 96, pp. 293–320, May 2003.

[88] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical programming*, vol. 39, no. 2, pp. 117–129, 1987.

[89] A. Papachristodoulou, *Scalable analysis of nonlinear systems using convex optimization*. PhD thesis, California Institute of Technology, 2005.

[90] G. Stengle, "A Nullstellensatz and a Positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, pp. 87–97, June 1974.

[91] J. Gouveia, P. A. Parrilo, and R. R. Thomas, "Theta bodies for polynomial ideals," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 2097–2118, 2010.

[92] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, “Lqr-trees: Feedback motion planning via sums-of-squares verification,” *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[93] M. H. Protter and H. F. Weinberger, *Maximum Principles in Differential Equations.* Springer, 1984.

[94] L. Evans, *Partial Differential Equations.* American Mathematical Society, 2010.

[95] B. O’Donoghue, *Suboptimal Control Policies Via Convex Optimization.* PhD thesis, Stanford University, 2012.

[96] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, “Efficient solution algorithms for factored MDPs,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 399–468, 2003.

[97] J. Lofberg, “YALMIP : a toolbox for modeling and optimization in MATLAB,” in *IEEE International Symposium on Computer Aided Control Systems Design*, pp. 284–289, 2004.

[98] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, “SDPT3 – a MATLAB software package for semidefinite programming,” *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[99] S. Prajna and A. Papachristodoulou, “Analysis of switched and hybrid systems — beyond piecewise quadratic methods,” in *American Control Conference (ACC)*, vol. 4, pp. 2779–2784, 2003.

[100] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[101] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, Massachusetts Institute of Technology, 1989.

[102] D. P. Bertsekas, "Multiplier methods: a survey," *Automatica*, vol. 12, pp. 133–145, Mar. 1976.

[103] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods.* Athena Scientific, 1996.

[104] P. Bjorstad and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, 2004.

[105] M. Johansson and A. Rantzer, "Computation of piecewise quadratic Lyapunov functions for hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 555–559, 1998.

[106] P. Biswas, P. Grieder, J. Löfberg, and M. Morari, "A survey on stability analysis of discrete-time piecewise affine systems," in *Proceedings of the 16th IFAC World Congress*, 2005.

[107] S. P. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[108] N. Parikh and S. P. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2014.

[109] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *Journal of Optimization Theory and Applications*, vol. 155, no. 1, pp. 227–238, 2012.

[110] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Optimization Online*, 2013.

[111] J. Löfberg, "Pre- and post-processing sum-of-squares programs in practice," *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 1007–1011, 2009.

[112] H. Owhadi, C. Scovel, T. J. Sullivan, M. McKerns, and M. Ortiz, "Optimal uncertainty quantification," *arXiv preprint arXiv:1009.0679*, 2010.

[113] C. W. Gardiner, *Handbook of stochastic methods*, vol. 3. Springer Berlin, 1985.

[114] S.-T. Yau and S.-T. Yau, "Finite-dimensional filters with nonlinear drift. iii: Duncan-mortensen-zakai equation with arbitrary initial condition for the linear filtering system and the benes filtering system," *IEEE transactions on aerospace and electronic systems*, vol. 33, no. 4, pp. 1277–1294, 1997.

[115] D. J. Higham, "An algorithmic introduction to numerical simulation of stochastic differential equations," *SIAM review*, vol. 43, no. 3, pp. 525–546, 2001.

[116] A. Barth, C. Schwab, and N. Zollinger, "Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients," *Numerische Mathematik*, vol. 119, no. 1, pp. 123–161, 2011.

[117] D. Xiu and G. E. Karniadakis, "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2002.

[118] D. Bertsimas and C. Caramanis, "Bounds on linear PDEs via semidefinite optimization," *Mathematical programming*, vol. 108, no. 1, pp. 135–158, 2006.

[119] D. Henrion, J. B. Lasserre, and C. Savorgnan, "Nonlinear optimal control synthesis via occupation measures," in *IEEE Conference on Decision and Control*, pp. 4749–4754, 2008.

[120] G. Beylkin and M. J. Mohlenkamp, "Algorithms for numerical analysis in high dimensions," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 2133–2159, 2005.

[121] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis," *Working Papers in Phonetics*, 1970.

[122] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition," *Psychometrika*, vol. 35, pp. 283–319, Sept. 1970.

[123] B. N. Khoromskij, "Tensors-structured numerical methods in scientific computing: Survey on recent advances," *Chemometrics and Intelligent Laboratory Systems*, vol. 110, pp. 1–19, Jan. 2012.

[124] Y. Sun and M. Kumar, "A tensor decomposition approach to high dimensional stationary fokker-planck equations," in *American Controls Conference (ACC)*, 2014.

[125] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, "The Convex Geometry of Linear Inverse Problems," *Foundations of Computational Mathematics*, vol. 12, pp. 805–849, Oct. 2012.

[126] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, p. 025010, Jan. 2011.

[127] J. Håstad, "Tensor rank is NP-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[128] C. Pozrikidis, *Introduction to finite and spectral element methods using Matlab*. CRC Press, 2005.

[129] D. J. Biagioni, D. J. Beylkin, and G. Beylkin, "On Rank Reduction of Separated Representations," *arXiv.org*, June 2013, 1306.5013v1.

[130] L. N. Trefethen, *Spectral methods in MATLAB*, vol. 10. SIAM, 2000.

[131] P. M. Esfahani, D. Chatterjee, and J. Lygeros, "Motion Planning via Optimal Control for Stochastic Processes," *arXiv.org*, Nov. 2012, 1211.1138v1.

[132] B. W. Bader and T. G. Kolda, "Matlab tensor toolbox version 2.5." Available online: `http://www.sandia.gov/~tgkolda/TensorToolbox/`, January 2012.

[133] E. Acar, D. M. Dunlavy, and T. G. Kolda, “A scalable optimization approach for fitting canonical tensor decompositions,” *Journal of Chemometrics*, vol. 25, pp. 67–86, February 2011.

[134] H. M. Osinga and J. Hauser, “The Geometry of the Solution Set of Nonlinear Optimal Control Problems,” *Journal of Dynamics and Differential Equations*, vol. 18, pp. 881–900, Aug. 2006.

[135] J. Hauser, S. Sastry, and G. Meyer, “Nonlinear control design for slightly non-minimum phase systems: application to V/STOL aircraft,” *Automatica*, vol. 28, no. 4, pp. 665–679, 1992.

[136] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, “Modeling the quad-rotor mini-rotorcraft,” pp. 23–34, London: Springer London, 2013.

[137] R. Leake and R.-W. Liu, “Construction of suboptimal control sequences,” *SIAM Journal on Control*, vol. 5, no. 1, pp. 54–63, 1967.

[138] J. O. Kim and P. K. Khosla, “Real-time obstacle avoidance using harmonic potential functions,” *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 338–349, June 1992.

[139] S. G. Loizou, “Closed form navigation functions based on harmonic potentials,” in *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pp. 6361–6366, IEEE, 2011.

[140] P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick, “Fourier analysis of the 2d screened poisson equation for gradient domain problems,” in *European Conference on Computer Vision (ECCV)*, pp. 114–128, Springer, 2008.

[141] C. I. Connolly, J. B. Burns, and R. Weiss, “Path planning using Laplace’s equation,” in *IEEE International Conference on Robotics and Automation*, pp. 2102–2106, 1990.

[142] K. Konolige, "A gradient method for realtime robot control," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 639–646, 2000.

[143] S. M. LaValle, *Planning algorithms.* Cambridge press, 2006.

[144] A. D'Innocenzo, A. Abate, M. D. D. Benedetto, and S. Sastry., "Approximate abstractions of discrete-time controlled stochastic hybrid systems," in *Proc. of IEEE Conf. on Decision and Control*, pp. 221–226, 2008.

[145] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Trans. on Robotics*, vol. 28, pp. 396–409, 2012.

[146] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *Proc. of 18th IFAC World Congress*, 2011.

[147] L. de Alfaro, *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.

[148] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: verification of probabilistic real-time systems," in *Proceedings of 23rd International Conference on Computer Aided Verification*, 2011.

[149] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *Proc. of Int. Conf. on Intelligent Robots and Systems*, 2013.

[150] I. Tkachev, A. Mereacre, J.-P. Koatoen, and A. Abate, "Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems," in *Proc. of Hybrid Systems: Computation and Control*, 2013.

[151] P. M. Esfahani, D. Chatterjee, and J. Lygeros, "Motion planning via optimal control for stochastic processes," *arXiv preprint arXiv:1211.1138*, 2012.

[152] O. Kupferman and M. Y Vardi, "Model Checking of Safety Properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[153] T. Latvala, "Efficient model checking of safety properties," in *Model Checking Software*, pp. 74–88, Springer, 2003.

[154] C. Baier and J.-P. Katoen, *Principles of Model Checking.* MIT Press, 2008.

[155] D. P. Bertsekas and J. N. Tsitsiklis, "An analysis of stochastic shortest path problems," *Mathematics of Operations Research*, vol. 16, pp. 580–595, 1991.

[156] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.

[157] I. Mitchell, "The flexible, extensible and efficient toolbox of level set methods," *J. Sci. Comput.*, vol. 35, pp. 300–329, 2008.

[158] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[159] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based Motion Planning with Temporal Goals," *International Conference on Robotics and Automation (ICRA)*, 2010.

[160] S. Prajna and A. Papachristodoulou, "Analysis of switched and hybrid systems–beyond piecewise quadratic methods," in *American Control Conference (ACC)*, pp. 2779–2784, 2003.

[161] E. Oñate and M. Manzan, "Stabilization techniques for finite element analysis of convection-diffusion problems," *Developments in Heat Transfer*, vol. 7, pp. 71–118, 2000.

[162] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, no. 2, pp. 439–471, 1988.

[163] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: A general purpose sum of squares programming solver," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 1, pp. 741–746, 2002.

[164] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto, "Algorithm 883: Sparsepop—a sparse semidefinite programming relaxation of polynomial optimization problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 2, p. 15, 2008.