Incremental Sampling based Algorithms for State Estimation

by

Pratik Chaudhari

B.Tech., Indian Institute of Technology Bombay (2010)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Aeronautics and Astronautics
August 27, 2012

Certified by .....
Emilio Frazzoli
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chairman, Department Committee on Graduate Theses

Incremental Sampling based Algorithms for State Estimation

by

Pratik Chaudhari

Submitted to the Department of Aeronautics and Astronautics
on August 27, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Perception is a crucial aspect of the operation of autonomous vehicles. With a multitude of different sources of sensor data, it becomes important to have algorithms which can process the available information quickly and provide a timely solution. Also, an inherently continuous world is sensed by robot sensors and converted into discrete packets of information. Algorithms that can take advantage of this setup, i.e., which have a sound founding in continuous time formulations but which can effectively discretize the available information in an incremental manner according to different requirements can potentially outperform conventional perception frameworks. Inspired from recent results in motion planning algorithms, this thesis aims to address these two aspects of the problem of robot perception, through novel incremental and anytime algorithms.

The first part of the thesis deals with algorithms for different estimation problems, such as filtering, smoothing, and trajectory decoding. They share the basic idea that a general continuous-time system can be approximated by a sequence of discrete Markov chains that converge in a suitable sense to the original continuous time stochastic system. This discretization is obtained through intuitive rules motivated by physics and is very easy to implement in practice. Incremental algorithms for the above problems can then be formulated on these discrete systems whose solutions converge to the solution of the original problem.

A similar construction is used to explore control of partially observable processes in the latter part of the thesis. A general continuous time control problem in this case is approximates by a sequence of discrete partially observable Markov decision processes (POMDPs), in such a way that the trajectories of the POMDPs—i.e., the trajectories of beliefs—converge to the trajectories of the original continuous problem. Modern point-based solvers are used to approximate control policies for each of these discrete problems and it is shown that these control policies converge to the optimal control policy of the original problem in an appropriate space. This approach is promising because instead of solving a large POMDP problem from scratch, which is PSPACE-hard, approximate solutions of smaller problems can be used to guide the search for the optimal control policy.

Thesis Supervisor: Emilio Frazzoli
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

I would first like to thank my advisor, Emilio Frazzoli. His vision and ideas are really what made this thesis possible. His confidence and encouragement helped at times when I was ready to give up on my ideas. I would also like to thank David Hsu whose patient ear was instrumental in completing a significant portion of this thesis. I have been lucky to be taught by some truly legendary teachers here at MIT. Their style has been a great source of inspiration. I dream of becoming a researcher and a teacher one day and I couldn't have wished for better role models.

The Laboratory of Information and Decision Systems is probably the best graduate lab I can ever imagine. With a strong legacy of many years, it continues to solve ground breaking problems even today. My lab mates at LIDS have played an important part in shaping this thesis. I have learnt a lot by sharing my ideas with them and I am grateful for their inputs. A special thanks goes to Sertac Karaman with whom I have worked closely. Long discussions with him have had a significant impact in shaping my thinking.

I would like to thank all my friends, some of whom I have had the pleasure of knowing for almost a decade now. Graduate school so far away from home would not have been as interesting or entertaining without the countless trips to movie theatres and downtown Boston.

No words can do justice to the role played by my family in my education. They deserve this accomplishment much more than me.

Contents

List of Figures

Chapter 1

Introduction

> *Moreover a mathematical problem should be difficult in order to entice us, yet not completely inaccessible, lest it mock our efforts. It should be to us a guidepost on the mazy path to hidden truths, and ultimately a reminder of our pleasure in the successful solution ....*
>
> David Hilbert, Paris International Congress, 1900

Autonomous vehicles have become popular due to the wide scope for their applications. From avoiding traffic jams in busy cities to helping the disabled and the elderly on their daily commute, they promise to revolutionize transportation. The success of recent DARPA challenges have shown that tasks like autonomous driving in unknown terrain or performing complicated routing tasks while operating under normal traffic rules is indeed possible.

Let us look at MIT's DARPA Urban Challenge vehicle [LHT+08] in greater detail to motivate the problem. It travelled completely autonomously for over six hours in an urban environment to perform specific tasks like navigation through a set of checkpoints. In order to do so, these vehicles rely on an intricate system of sensors to give them information about the real world. Vast arrays of sensors such as lidars (laser-sensors), radars and cameras together provide information in the order of several megabytes per second. An example is shown in Figure 1-2. There are a number of problems that an autonomous vehicle has to solve using this data. The first can be called "perception" and it involves processing all this data from different sources like laser points, camera images, sonar readings to create a unified usable map of the surroundings. The second problem—let us call it "planning"—involves executing certain specified tasks in this world created by perception algorithms. The crucial aspect of this program is processing this enormous amount of information efficiently to quickly provide information about the location and orientation of the vehicle, location, speed and kind of obstacles near the vehicle, etc., which the planner typically uses.

As will be discussed in Section 1.1.1, the motion planning problem has found a satisfactory solution. This thesis aims to provide a similar solution for the perception problem. In particular, it proposes algorithms for state estimation and control of partially observed systems that have two major flavors.

- *Generality* : Sensors interact with the outside world that is inherently continuous. Data collection and analysis is however a discrete process. By formulating the original problem in continuous time, we ensure a faithful representation of the continuous information process. The algorithms proposed in this thesis provide a natural way of discretizing this while keeping the formulation as general as possible.

- *Anytime computation* : Algorithms that do not need to be tuned to operate on systems with varying parameters or computational capability are desirable. The notion of anytime computation in this context applies to algorithms which provide the answer (viz. an estimate or a control policy) quickly and improve upon the solution if given more computational resources or more information. In addition to this, the improved solution should be obtained only after an *incremental* effort i.e., incorporating a new set of observations or improving the control policy should utilize the previous solution, require a little amount of additional computation, instead of solving again from scratch.

(a) DARPA Urban Challenge [LHT+08] (b) Voice-controlled Forklift [TWA+10]

Figure 1-1: Autonomous vehicles present significant challenges to perception algorithms

1.1 Background

1.1.1 Motion Planning

In a different context, the curse of dimensionality has been shown to be inevitable in robot motion planning problems, i.e., the problem of finding an optimal dynamically feasible trajectory around obstacles so as to reach a goal region. Even setting aside optimality, it was shown early on that finding a feasible solution to this problem is PSPACE-hard [Rei79], which strongly suggests that algorithms that return a feasible solution if one exists and return failure otherwise are doomed to suffer from excessive computational costs. Yet, algorithms with probabilistic guarantees such as Probabilistic RoadMaps (PRM) [KL98] or the Rapidly-exploring Random Tree (RRT) [LKJ01] have been shown to work quite effectively in returning a feasible solution in high-dimensional configuration spaces by effectively

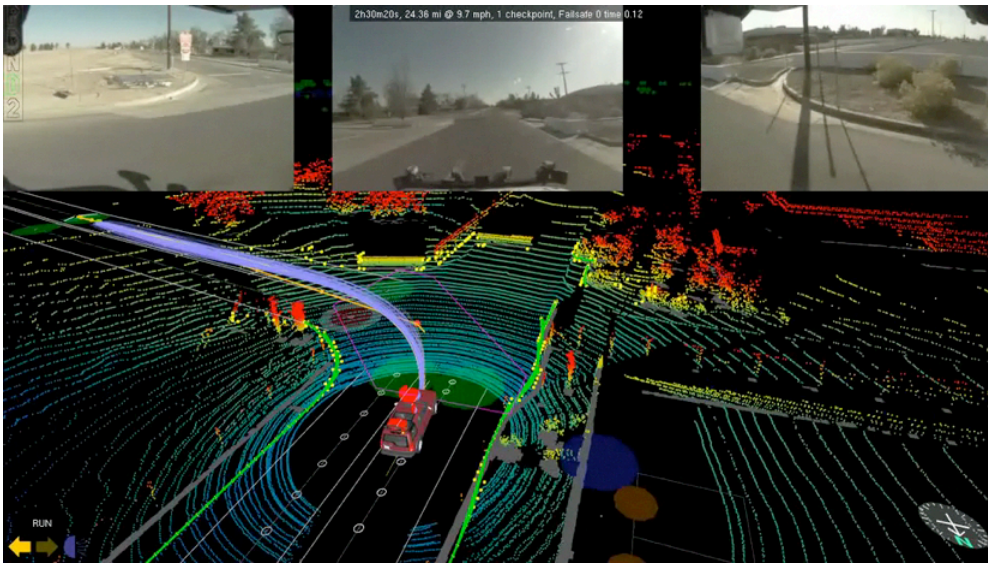


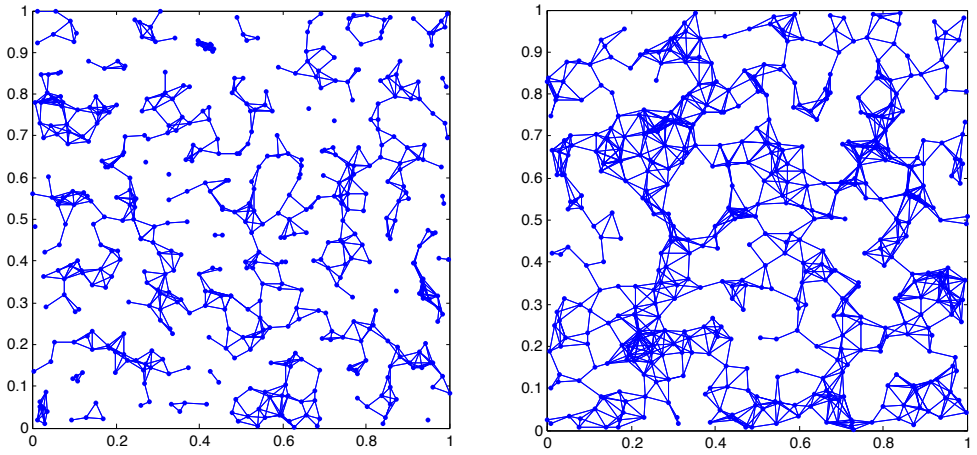
Figure 1-2: A picture showing sensor data gathered on MIT's DUC vehicle. Concentric circles are data from one 3D laser sensor (velodyne) giving more than 1 million points per second. With 5 cameras, 12 lidars, 16 radars and a velodyne, efficiently processing such large amounts of information becomes important.

discretizing the said space based on random sampling. In [KF11], two novel motion planning algorithms were proposed under the name of PRM* and RRT*, which are shown to be both computationally efficient in the sense that they return a solution quickly and converge to the optimal solution asymptotically if given more computation time. In particular, the RRT* algorithm has been successfully applied to many challenging motion planning problems involving high-dimensional configuration spaces [PKW$^+$11], complex dynamical systems [JKF11]. RRT* uses a classic result in random geometric graphs which says that if a randomly sampled set of points S has size n, connecting every point $z \in S$ to its $\mathcal{O}(\log n)$ neighbors ensures that the resulting graph has optimal shortest paths in the limit as the number of samples goes to infinity. This result is sharp in the sense that fewer connections than this is almost surely sub-optimal.

This argument is at the heart of all the algorithms proposed in this thesis. It provides an effective way of discretizing the state-space doing just enough work to ensure optimality. Algorithms for construction of Markov chain approximations in this thesis use this idea to discretize both the state-space and the time-axis at appropriate rates, thereby ensuring optimality of estimates while at the same time keeping the size of the graph to a minimum.

1.1.2 State Estimation

First originating in the works of Wiener and Kolmogorov, the filtering problem has a vast body of literature dedicated to it. Filters belonging to the Kalman filter family like the KF, EKF, UKF etc. are predominantly used in control systems and signal processing applications. On the other hand, newer problems such as localization of mobile robots in unknown environments have been tackled with Monte Carlo algorithms known as particle



(a) Sub-critical (b) Super-critical

Figure 1-3: Isolated regions in Figure 1-3a result from less than $\mathcal{O}(\log n)$ connections leading to sub-optimality. On the other hand, with the graph in Figure 1-3b is fully connected and leads to an optimal trajectory with $\mathcal{O}(\log n)$ connections.

filters. The Kalman filter is an optimal filter for linear systems with additive Gaussian noise in the sense that it provides the estimate which minimizes the mean square error. It turns out that it is possible to get a finite dimensional optimal filter only in the case of linear systems with Gaussian noise corrupting the observations. The general filtering problem, e.g., for continuous time nonlinear systems or non-Gaussian noise, is difficult because even parametrizing the solution is non-trivial. The Kalman filter has to propagate a finite state vector and a covariance matrix to keep track of the optimal estimate whereas particle filters have to propagate a large set of particles with their probabilities that represent the whole conditional distribution of the state variable based on observations till then. While the former family of filters is applicable to only a restricted class of systems, the latter approaches although very general are not robust [PS99] and require a number of techniques like adaptive sampling [Fox03] and particle resampling [DC05] for application on real-world problems.

This thesis takes a different approach to the filtering problem. It focuses on creating a simple discrete approximation of the underlying continuous time system using Markov chain approximations. Propagating the filtering estimates on this discrete Markov chain is then as easy as an application of Bayes' law. To do so, it uses a discretization created by random sampling inspired from asymptotically optimal motion planning algorithms. In addition to this, it also utilizes new ideas such as "anytime" computation to propose a filtering algorithm that provides a quick inaccurate state estimate but can improve it if given more computation time.

1.1.3 POMDPs

The second part of the thesis deals with control of partially observable processes modeled as Markov decision processes (POMDPs). POMDPs are a principled approach for deci-

sion making under uncertainty. A number of different approaches are available for solving this problem such as value function approximation, belief space planners, stochastic control methods and more recent point based methods. The generality of this formulation however comes at a cost. Most real-world problems beyond toy examples are computationally intractable. The number of future states of the system grows exponentially with the size of the system. The problem becomes even harder as the length of the time horizon over which future actions are predicted increases. These two aspects are coupled in real problems and that makes achieving a tractable solution much harder.

This thesis proposes a way to solve a larger POMDP by breaking it into smaller parts which can be solved more efficiently. Roughly, by creating a sequence of smaller problems that are close to the original problem, we can use the solutions of smaller problems to get an approximation of the solution to the original problem. In fact the formulation given here is general enough to encompass the complete continuous-time stochastic problem. The program proposed by this thesis then reduces this continuous time problem into a sequence of discrete POMDPs in such a way that trajectories of beliefs of the discrete problems converge to the trajectories of beliefs of the continuous time problem. This is then used to prove that the cost function obtained by solving the discrete problem using, say, a point-based POMDP solver converges to the cost function of the original continuous-time problem. The final output of this offline approach also gives a control policy which can be shown to converge to the control policy of the original problem in an appropriate relaxed sense.

1.2 Contributions

The contributions of this thesis are as follows. The major idea pursued all through this thesis is that a continuous time system can be approximated by a sequence of completely discrete finite systems. These approximations are derived from very intuitive rules and are easy to implement in practice. Algorithms proposed in Chapter 2 enable one to generate a sequence of approximate solutions in an incremental fashion. Ideas inspired from random sampling algorithms in the motion planning literature are used to enable approximations of continuous systems with large state-spaces.

Modifications of the Markov chain approximation method are used to solve the continuous-time optimal nonlinear filtering problem in Chapter 3. Random sampling introduced in Chapter 2 enables us to provide an anytime solution to this. In the same spirit, related problems such as smoothing and trajectory decoding are formulated and solved using Markov chain approximation in an incremental fashion. The nature of these algorithms, on one hand, allows us to formulate the continuous time problem in its full generality; on the other hand, since they rely at each step on completely discrete structures, traditional algorithms for inference for finite problems still apply.

The second part of the thesis deals with control of partially observable processes. A continuous time version of the problem is formulated and it is shown that it is equivalent to a number of other popular formulations which originated in vastly different fields like control of distributed systems or separation principle style results. The discrete version of this problem is popular in computer science literature solves problems like Markov decision process (MDPs) and partially observable Markov decision processes (POMDPs). Sequences of discrete POMDPs can then be constructed to converge to the original continuous time

problem, algorithms for which this thesis proposes. The notion of convergence is usually weak but can still be used to prove convergence of corresponding cost functions and control policies. The program then is to solve these discrete POMDPs using state-of-the-art point based solvers and use their solutions to converge to the solution of the continuous time problem. An important offshoot of this work is that the discretization of the state space and of the time horizon gives a natural way to work within the limits of both the curse of dimensionality and the curse of history simultaneously.

Parts of this thesis have been published in [CKF12].

1.3 Organization

There are three major parts to this thesis. Chapter 2 introduces the main idea of the thesis which is used throughout the other chapters. It gives a background on Markov chain approximations and proposes two algorithms to construct these both utilizing random sampling techniques. Chapter 3 takes a look at one of the two major problems discussed in this thesis, i.e., filtering. It proposes an algorithm for optimal filtering which is valid for a large class of general dynamical systems. Chapter 3 also formulates and discusses two related problems of state estimation, i.e., smoothing and trajectory decoding, and proposes an incremental and anytime solution to these state estimation problems.

Control of partially observable continuous-time processes forms the second part of this thesis and is discussed in Chapter 4. A general continuous-time partially observable control problem is formulated and then approximated by a sequence of discrete time, discrete state POMDP problems in such a way that sequences of discrete problems converge to the original continuous problem in an appropriate sense. Optimal solutions of discrete problems can be shown to converge to the optimal solution of the continuous time problem in terms of the cost function and the control policy obtained.

Chapter 5 takes a holistic view of the problems addressed in this thesis and identifies future directions for research.

Chapter 2

Markov Chain Approximation Method

This chapter studies diffusion processes of the form

$$dx(t) = f(x(t), t)\ dt + F(x(t))\ dw. \tag{2.1}$$

where $x(t) \in \mathbb{R}^d$, $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is the drift vector, $F(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d\times k}$ is the diffusion matrix and $w(t)$ is the standard k-dimensional Wiener process. The aim is to approximate the cost of the form

$$W(x) = \mathbb{E}\left[\int_0^\tau\ l(x(t))\ dt + L(x(\tau))\right], \tag{2.2}$$

until some target set ∂G is reached. The time τ is defined as the exit time from some compact set $G \subset \mathbb{R}^d$, i.e., $\tau = \inf\{t : x(t) \notin G^o\}$ where G^o denotes the interior of G. The problem thus stops when the system hits the boundary of set G. Denote the differential operator of Equation (2.1) by $\mathcal{L}$ using Dynkin's formula as

$$\mathcal{L} = f(x)\frac{d}{dx} + \frac{1}{2}F(x)\frac{d^2}{dx^2}. \tag{2.3}$$

It can then be shown that formally, the cost function $W(x)$, i.e., the cost incurred if the process starts from $x(0) = x$ satisfies the equation

$$\mathcal{L}W(x) + l(x) = 0 \qquad x \in G^0, \tag{2.4}$$

with the boundary condition $W(\tau) = L(x(\tau))$. It is possible to obtain an approximate solution of the partial differential equation (2.4) using a finite-difference method. However, these equations are only formal and even when it can be proved that the solution exists and is unique, the convergence of finite-difference solutions is difficult to prove without additional regularity assumptions. On the other hand, equations of the kind (2.1) are often obtained from physical models and it is possible to use the intuition gained from there to solve problems of stochastic control. An added advantage is that such approaches neither require an understanding of analytical properties of the diffusion process nor guarantees on the nature of solutions of the Bellman equation (2.4). The methods discussed in this chapter

are very similar to finite difference methods even though their analysis is quite different the former lends to completely probabilistic algorithms.

This chapter pursues the basic idea that a diffusion process can be approximated by a sequence of computationally tractable finite stochastic process along with a corresponding simplification of the cost function. In particular, the simpler processes will be Markov chains in the case of uncontrolled diffusions, while they will be controlled Markov chains (Markov decision processes) when the original diffusion problem has control embedded in it. These Markov chains will be derived from the original system by a series of intuitive rules which will be demonstrated on the canonical problem described above. Essentially, it will be ensured that the Markov chain and the diffusion process are equivalent in a certain "local" sense. There are a number of ways of achieving this end and one usually chooses the most computationally feasible one.

The approximating Markov chains will have a state space which is a finite subset of the state space of the original problem. They will be parametrized by the size of this finite set. This is akin to the resolution parameter h in finite-difference algorithms. Roughly, as the number of states in the Markov chain goes to infinity, it will increasingly resemble the diffusion process in its local properties, i.e., the expected change of state per step (drift) and the expected mean square change of state per step (diffusion). The approach does not need regularity assumptions on the solutions of the Bellman equation and can take advantage of purely intuitive notions of the physical process. For example, the expected change in state per transition is equal to the drift whereas the covariance of the change in state is the diffusion. It should be noted that the nature of convergence is similar to those of finite difference schemes.

Virtues of the said approximation being many, it still turns out that it is plagued by the same problems faced by finite difference schemes. It is essential to find a good way to choose the resolution parameter to create a consistent and computationally feasible Markov chain approximation. The later part of the chapter employs methods from random sampling algorithms to create consistent Markov chains. This will enable us to approximate high-dimensional stochastic systems. The major advantage of using random sampling methods to create Markov chains is that they can be made "incremental" very easily. Given a Markov chain with n states that approximates the continuous-time system, creating a refinement of the Markov chain to contain $n+1$ states say is an $\mathcal{O}(\log n)$ operation. This approximation also has the "anytime" property, i.e., for real-time implementations, the algorithm finds a crude solution to the state-estimation problem very quickly and converges to optimal solution (in an appropriate sense) if given more computation time. This contrasts conventional algorithms which need a fixed amount of time to come up with the solution.

The organization of this chapter is as follows. Section 2.1 will provide introductory concepts on Markov chains and background of the approximation method. The next section, Section 2.2, details the construction of the Markov chain. It provides two methods: one utilizes fixed grids, while the other creates an approximation using a random sampling of the state space. This is followed by Section 2.5 which discusses convergence properties of these approximations. The chapter concludes with experiments verifying the convergence for an example stochastic system in Section 2.6.

2.1 Preliminaries

This section is devoted to some preliminary definitions and results.

2.1.1 Discrete Markov Chains

A *Markov chain* is denoted by the tuple $M = (S, P)$, where $S \subset \mathcal{S} \subset \mathbb{R}^d$ is a finite set of states and $P(\cdot \,|\, \cdot) : S \times S \to \mathbb{R}_{\geq 0}$ is a function that denotes the transition probabilities, i.e., the function $P(z \,|\, z')$ is the probability that the next state is z given that the current state is z'. As a conditional probability mass function, P satisfies $\sum_{z \in S} P(z \,|\, z') = 1$ for all $z' \in S$. The trajectory of a Markov chain starting from an initial state z_0 is denoted by the sequence $\{\xi_i : \ i \in \mathbb{N}\}$ of random variables that satisfy (i) $\xi_0 = z_0$ and (i) $\mathbb{P}(\xi_{i+1} = z \,|\, \xi_i = z') = \mathbb{P}(z \,|\, z')$ for all $z, z' \in S$ and all $n \in \mathbb{N}$. The initial state z_0 can also be drawn from some distribution π.

2.1.2 Continuous time interpolations

Markov chains as described in Section 2.1.1 are primarily discrete objects. The trajectories as described there can however be interpolated to give a continuous-time trajectory. To this end, define $M = (S, P, T)$ to be a Markov chain where T is the set of functions $\Delta t(z) : S \to \mathbb{R}_{>0}$ that associate a time interval to each state in S. The function Δt is called the function of interpolating times, or *holding time* for short. Roughly, $\Delta t(z)$ is the time that the chain spends at state z before making another transition. It is made precise in the sequel with an explicit formula.

Given an initial state $z_0 \in S$, let $\{\xi_i : \ i \in \mathbb{N}\}$ denote the trajectory of the Markov chain M starting from z_0. $\xi(\cdot)$ is the *continuous-time interpolation* of such trajectories under holding times Δt, i.e.,

$$\xi(s) = \xi_i \quad \text{for all } s \in [t_i, t_{i+1}),$$

where $t_i = \sum_{j=1}^{i} \Delta t(\xi_j)$. For any realization $\xi(t, \omega)$ of the stochastic process $\{\xi(t); t \in \mathbb{R}_{\geq 0}\}$, the function $\xi(\cdot, \omega)$ is continuous from the right and has limits from the left, i.e., $\xi(t, \omega) \in D^d[0, \infty)$. Hence, ξ can be thought of as a random mapping that takes values in the function space $D^d[0, \infty)$. Figure 2-1 shows the interpolation of the trajectories of the discrete Markov chain. If $\Delta t(z) \to 0$ as the number of states in the Markov chain $n \to \infty$, the bold trajectory converges in distribution to the blue trajectory of the original stochastic system.

The interpolation interval $\Delta t(z)$ can in fact be taken to be a constant for a finite Markov chain if we prefer but this is restrictive. If the drift vector $|f(z)|$ is large for some parts of the state-space, we might find it beneficial to reduce the interpolating interval there. It is also because of this that numerical algorithms approximating the cost function converge faster if we have a roughly uniformly dense Markov chain which is ensured if the interpolating time is a function of the drift and diffusion terms. As we will see later, the construction of the Markov chain gives the form of both the transition probabilities and the holding times as a by product.

Given a continuous time interpolation, we can come up with conditions under which the trajectories of a sequence of Markov chains converge to the trajectories of the original

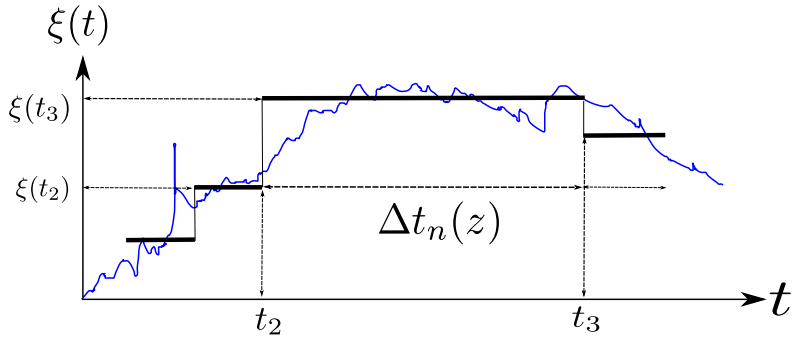


Figure 2-1: Blue : Stochastic trajectory, Bold black : interpolated trajectory

process described by Equation (2.1). Let $\{M_n; n \in \mathbb{N}\}$, where $M_n = (S_n, P_n, T_n)$, denote a sequence of Markov chains. For each $n \in \mathbb{N}$, let $\{\xi_i^n; i \in \mathbb{N}\}$ be the trajectory of M_n with initial state distributed according to some distribution π_n. The sequence of Markov chains $\{M_n; n \in \mathbb{N}\}$ is said to be *locally consistent* [KD01] with the original system described by Equation (2.1) if the following criteria are satisfied for all $z \in S$.

$$\circ \quad \lim_{n\to\infty} \Delta t_n(z) = 0, \tag{2.5}$$

$$\circ \quad \lim_{n\to\infty} \frac{\mathbb{E}[\xi_{i+1}^n - \xi_i^n \,|\, \xi_i^n = z]}{\Delta t_n(z)} = f(z), \tag{2.6}$$

$$\circ \quad \lim_{n\to\infty} \frac{\mathrm{Cov}[\xi_{i+1}^n - \xi_i^n \,|\, \xi_i^n = z]}{\Delta t_n(z)} = F(z)F(z)^T. \tag{2.7}$$

where $\mathrm{Cov}(x) = \mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^T]$.

Note that the conditions above imitate local properties of the drift-diffusion as described in Equation (2.1), i.e., if the current state is x, the state after a small time $\delta > 0$ is given by $x(\delta)$ and the initial state is x, we have,

$$\mathbb{E}[x(\delta) - x] = f(x)\ \delta + o(\delta),$$
$$\mathrm{Cov}[x(\delta) - x] = F(x)F(x)^T\delta + \mathcal{O}(\delta^2).$$

The first condition, i.e., effectively, $\sup_z \Delta t(z) \to 0$, ensures that $\delta \to 0$ resulting in the local behavior of the Markov chain converging to that of Equation (2.1). Surprisingly, as stated in Theorem 2.2, under mild technical assumptions, local consistency implies the convergence of continuous-time interpolations of the trajectories of the Markov chain to the trajectories of the stochastic dynamical system described by Equation (2.1). An example to convince the reader is due at this point.

Example 2.1. Consider the system $dx = -x\ dt + \sigma\ dw$ on some bounded interval $\mathcal{S} \subset \mathbb{R}_{\geq 0}$ with a regular discretization of distance h. In this example, any state x in the Markov chain is only connected with its neighbors $x - h$ and $x + h$. The Markov chain transitions to the right using only diffusion whereas it uses both drift and diffusion to go left:

$$P(x + h \,|\, x) = \frac{\sigma^2/2}{c} \quad \text{and} \quad P(x - h \,|\, x) = \frac{\sigma^2/2 + hx}{c}.$$

These transition probabilities sum up to 1, i.e., $c = (\sigma^2 + hx)$. Finally, local consistency conditions are satisfied if we choose $\Delta t_h = c^{-1}h^2$. The following section provides an explanation of why the transition probabilities take the above form. It will also be seen that the exact form is quite immaterial to the convergence, all that is required is that the approximation satisfy the local consistency conditions.

2.2 Construction of the Markov chains

This section concentrates on constructions of approximations which can be readily implemented. They are obtained from intuitive techniques and are very versatile. The basic method to obtain consistent Markov chains is using a finite discretization of the partial differential equation shown in Equation (2.4). This is however only used as a guide to the process, the construction is probabilistic and gives the transition probabilities as well as the holding times. It is also possible to change these probabilities to get different versions for numerical benefits so long as the consistency equations are satisfied. This aspect of the construction will be exploited throughout this thesis to apply the method to different problems.

2.3 Grid based methods

The set of states of the Markov chain M is a subset of the original state space. Let $x \in \mathbb{R}$, i.e., consider a $1-$dimension drift-diffusion process in Equation (2.1) to be amenable to a rough derivation. Given a compact set $G = [0, B]$, we can create a uniform grid of discretization h and call it S. Writing out Equation (2.4) in its explicit form, we get that the cost function $W(x)$ satisfies the PDE

$$W_x f(x) + W_{xx} F(x) + l(x) = 0. \tag{2.8}$$

Write the finite difference approximations as

$$W_x \sim \begin{cases} \dfrac{W(x+h) - W(x)}{h} & \text{if } f(x) \geq 0, \\ \dfrac{W(x) - W(x-h)}{h} & \text{if } f(x) < 0; \end{cases}$$

$$W_{xx} \sim \frac{W(x+h) - 2W(x) + W(x-h)}{h^2}.$$

For a general function $f(x)$, decompose it into positive and negative parts as $f(x) = f^+(x) - f^-(x)$, i.e., $f^+(x) = \max(f(x), 0)$ and $f^-(x) = -\min(f(x), 0)$ which also gives $|f(x)| = f^+(x) + f^-(x)$. Substituting these in Equation (2.8), after multiplying and dividing by $F^2(x) + h\,|f(x)|$, we get,

$$W(x) = \frac{F^2(x)/2 + hf^+(x)}{F^2(x) + h\,|f(x)|}\, W(x+h) + \frac{F^2(x)/2 + hf^-(x)}{F^2(x) + h\,|f(x)|}\, W(x-h) + \frac{h^2}{F^2(x) + h\,|f(x)|}\, l(x)$$

Write this as

$$W(x) = \mathbb{P}(x+h \,|\, x)\ W(x+h) + \mathbb{P}(x-h \,|\, x)\ W(x-h) + \Delta t(x)\ l(x)$$

with $\mathbb{P}(x+h \,|\, x), \mathbb{P}(x-h \,|\, x)$ and $\Delta t(x)$ being the coefficients of the respective terms above. Let $\mathbb{P}(x \,|\, y) = 0$ if $y \neq x \pm h$. These probabilities sum up to one (by construction in fact) and thus can be considered a legitimate Markov transition kernel. A quick check for local consistency gives,

$$\mathbb{E}[\Delta\xi(x)] = \frac{F^2(x)/2 + hf^+(x)}{F^2(x) + h\,|f(x)|}\ h + \frac{F^2(x)/2 + hf^-(x)}{F^2(x) + h\,|f(x)|}\ (-h) = f(x)\ \Delta t(x)$$

and similarly

$$\begin{aligned}\mathbb{E}[(\Delta\xi(x))^2] = h^2 \left(\frac{F^2(x)/2 + hf^+(x)}{F^2(x) + h\,|f(x)|} + \frac{F^2(x)/2 + hf^-(x)}{F^2(x) + h\,|f(x)|} \right) &= F^2(x)\ \Delta t(x) + \Delta t(x) h\,|f(x)| \\ &= F^2(x)\ \Delta t(x) + \Delta t(x)\mathcal{O}(h).\end{aligned}$$

Note that $\mathbb{E}[(\Delta\xi - \mathbb{E}[\Delta\xi])^2] = F^2(x)\ \Delta t(x) + \Delta t(x) O(h)$ as well and this assignment of probabilities thus satisfies the local consistency conditions in Section 2.1.2. Of course, the condition for this assignment to be valid is that,

$$\inf(F^2(x) + h\,|f(x)|) > 0 \qquad \text{for } x \in S.$$

For a number of other considerations and a more elaborate discussion refer Chapter 5 of [KD01]. In the general case, transition probabilities to neighboring states depend upon both the direction of the drift vector and the diffusion vector. For example, the state in the top-right corner will have the highest probability of being the next state in a Markov trajectory starting from state z. Connections of this form thus implement the intuitive notion that a stochastic differential equation is a superposition of a drift term and a diffusion term.

2.4 Random sampling based methods

The previous section gave a method of constructing Markov chains for *a priori* discretization level h. It is worthwhile to explore methods which can create successively refined approximations of the same stochastic dynamics. There are a number of ways of doing so using grid based constructions. Consider a simple example, that of laying a grid in a unit hypercube. In d-dimensions, using a regular grid of distance h this takes $\mathcal{O}(h^{-d})$. The number of samples grows exponentially in the number of dimension. This is otherwise known as the *curse of dimensionality* which we will encounter in later chapters as well. Let us look at this from two different perspectives.

- **Incrementality :** Suppose we want to refine the grid, i.e., reduce the distance between nodes by half. An extra $\mathcal{O}((2^d - 1)\ h^{-d})$ nodes have to be added to achieve this. More generally, given a grid of $\mathcal{O}(n)$ nodes, refining the grid so as to halve the distance between the nodes takes an additional $\mathcal{O}((2d-1)\ n)$ nodes. The transition

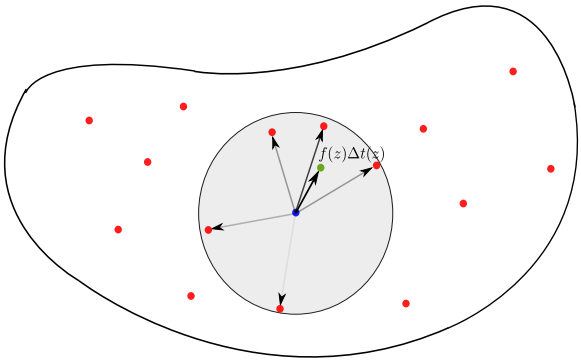


Figure 2-2: Transition probabilities using random sampling of states. Note that the directions where the drift vector has low magnitude result in low transition probabilities (boldness of arrows) while the probability of transition in direction of the drift vector is very high.

probabilities explicitly depend upon the discretization level h and thus we have to find the transition probabilities of all the nodes, i.e., $\mathcal{O}(2d\ n)$ operations in total. In other words, we have to run the algorithm *afresh* on the new grid instead of utilizing the transition probabilities of the earlier construction. It is possible to increase the number of nodes in such a way that the grid is refined incrementally by adding nodes to only certain regions of the state-space. However, doing so is complicated and it is important to ensure that the *dispersion*, i.e., the maximum distance between nodes in the set S goes to zero. In other words, it is necessary to artificially introduce a "bias" to reduce the dispersion in regions with sparse nodes. Random sampling methods are based on probabilistic techniques and this bias is incorporated in the method itself. We can thus easily construct the new transition probabilities in an incremental fashion. This typically results in vast benefits for successive refinement algorithms.

- **Anytime completeness :** As mentioned above, grid based methods start with an *a priori* discretization of the state space. It is necessary to perform an $\mathcal{O}(n)$ operations to construct the transition probabilities of all the nodes to get the Markov chain. On the other hand, random sampling methods construct the grid probabilistically in such a way that the transition probabilities are constructed along with the sampling. This makes the algorithms amenable to *anytime completeness* which means that a poorly refined Markov chain is obtained quickly without having to wait for completion of the whole construction.

2.4.1 Primitive procedures

Sampling : Let $x \in \mathcal{S} \subset \mathbb{R}^d$. We will primarily use uniform random sampling procedure to populate the set of states of the Markov chain S. The `Sample` procedure returns states sampled independently and uniformly from the bounded set $\mathcal{S}$.

Neighboring states : Given $z \in \mathcal{S}$ and a finite set $S \subset \mathcal{S}$ of states, the procedure `Near`(z, S) returns the set of all states that are within a distance of $r = \gamma\ (\log n/n)^{1/d}$ from

z, i.e.,

$$Z_{near}(z) = \left\{ z_k \in S,\ z_k \neq z : \|z_k - z\|_2 \leq \gamma \left(\frac{\log n}{n}\right)^{1/d} \right\},$$

where $n = |S|$, $d = \dim(S)$ and $\gamma > 0$ is a constant that will be specified in Section 2.5. Given a state $z \in S$, let Z_{near} be the set of states returned by $\texttt{Near}(z, S)$.

Time Intervals : Given a state $z \in S$, the procedure $\texttt{ComputeHoldingTime}(z, S)$ returns a holding time given by the formula

$$\Delta t(z) = \frac{r^2}{\|F(z)F^T(z)\|_2 + r\|f(z)\|_2},$$

where r is as given in the procedure $\texttt{Near}(z, S)$. A rough explanation of the nature of the above formula is as follows. For a deterministic system without the diffusion term, the time taken to travel a distance r is $\Delta t_1 = \frac{r}{\|f(z)\|_2} + o(r)$. On the other hand, for a system with no drift and only diffusion the expected time taken to travel the same distance is $\mathcal{O}\left(\frac{r^2}{\|F(z)F^T(z)\|_2}\right)$ (using Dynkin's formula). Thus the expression of $\Delta t(z)$ is motivated by $\Delta t = \frac{\texttt{distance}}{\texttt{average velocity}}$. This formula satisfies the condition given in Equation (2.5) as $r \to 0$, i.e., $n \to \infty$. From the derivation in Section 2.3, we can see that the exact expression for the transition probabilities and holding time is flexible so long as it satisfies the local consistency conditions. While applying the method to different problems we will again leverage this flexibility to scale the holding times to achieve computationally efficient algorithms.

Transition Probabilities Correctly choosing the transition probabilities so as to satisfy local consistency conditions is key to success of the Markov chain approximation method. There are a number of ways of satisfying these conditions, two of which will be described below. Once the basic methodology of the Markov chain approximation is understood, we can appropriately change the transition probabilities and the holding times to still get consistent approximations.

Local consistency conditions translate into a linear program for finding the transition probabilities which is given as follows. Given a set Z_{near} find $K = |Z_{near}|$ transition probabilities say $p_1, p_2, \ldots, p_K$ such that $p_1 = \mathbb{P}(z_k \,|\, z)$ where $z_k \in Z_{near}$. The local consistency in Equations (2.5), (2.6) and (2.7) conditions can be written as a set of linear constraints. Roughly, these conditions essentially approximate the small time behavior the stochastic system. It is because of this that we can also use a local Gaussian to get the probabilities as follows. Given a state $z \in \mathcal{S}$ and a finite set $Z_{\text{near}} \subset \mathcal{S}$, the procedure $\texttt{ComputeTransProb}(z, Z_{\text{near}}, \Delta t(z))$ returns a function $p(\cdot \,|\, z)$ which is computed as follows. Let $\mathcal{N}_{\mu,\Sigma}(\cdot)$ denote the density of the (possibly multivariate) Gaussian distribution with mean μ and variance Σ; then,

$$p(z_k \,|\, z) = \eta\, \mathcal{N}_{\mu,\Sigma}(z_k).$$

Here $\mu = z + f(z)\Delta t(z)$ and $\Sigma = F(z)F(z)^T\Delta t(z)$ and the constant η ensures $\sum_{k=1}^{K} p(z_k \,|\, z) = 1$. Lemma 2.3 in Section 2.5 proves that this satisfies local consistency conditions in the

limit. Note that $\mathbb{P}(z' \mid z) = 0$ if $z' \notin Z_{near}$. Although it is certainly possible to make more densely connected Markov chains, it will turn out that just connecting to $\mathcal{O}(|Z_{near}|)$ nearest vertices is enough. This is seen in another place as follows. For grid based methods, every vertex is connected to $2d$ neighboring vertices and this still gives an irreducible Markov chain. For randomly sampled nodes, connecting to mere nearest neighbors is not enough. We need to connect to at least $\mathcal{O}(\log n)$ neighbors to ensure that the resulting Markov chain is irreducible. This is easily seen because of the following. An irreducible Markov chain is defined as a Markov chain where all the states of the chain form a single communicating class, i.e., there exists some integer $m > 0$ such that $\mathbb{P}(\xi(m) = z_2 \mid \xi(0) = z_1) > 0$ for any $z_1, z_2 \in S_n$. This is proved in Theorem 2.4 by proving that (i) every state is connected to its neighbors, (ii) all transition probabilities to neighbors are positive. These two thus together make the Markov chain irreducible. The value of the constant γ is set precisely such that this happens.

Connect State : The procedure `ConnectState` acts on a state $z \in S$. It computes the holding time for z and also the transition probabilities to all the states in the set Z_{near}.

2.4.2 Batch construction of the Markov chain

Sampling n states for the Markov chain to populate the set S can be done in a "batch" fashion where the transition probabilities and holding times are calculated on a pre-sampled set S. Algorithm 2.1 takes a set of sampled states S_n as the input (lines 1-5). It runs the procedure `ConnectState` on all $z \in S_n$ to calculate the transition probabilities and the holding times at each state. This creates the Markov chain M_n. The `Near` procedure takes worst case $\mathcal{O}(\log n)$ time using approximate nearest neighbor algorithms [Sam95]. Note that the expected number of samples in a ball of radius $\gamma \left(\frac{\log n}{n}\right)^{1/d}$ is $\mathcal{O}(\log n)$ [KF11]. Hence, the `ComputeTransProb` procedure takes $\mathcal{O}(\log n)$ time. The complexity of `ConnectState` is thus $\mathcal{O}(\log n)$. Algorithm 2.1 thus runs in $\mathcal{O}(n \log n)$ time to create a Markov chain M_n.

Algorithm 2.1: Batch Markov chain construction

```
1 n ← 0;
2 while n < N do
3     z ← Sample();
4     S ← S ∪ {z};
5     n ← n + 1;
6 for z ∈ S_N do
7     (S_N, P_N, T_N) ← ConnectState(z, (S_N, P_N, T_N));
8 return (S_N, P_N, T_N);
```

Algorithm 2.2: `ConnectState`$(z, (S, P, T))$

1 $\Delta t(z) \leftarrow$ `ComputeHoldingTime`(z, S);
2 $Z_{\text{near}} \leftarrow$ `Near`(z, S);
3 $P(\cdot \mid z) \leftarrow$ `ComputeTransProb`$(z, Z_{\text{near}}, \Delta t(z))$;
4 $T(z) \leftarrow \Delta t(z)$;
5 **return** (S, P, T);

2.4.3 Incremental construction of the Markov chain

For algorithms discussed in the following chapters, it is of interest to create the Markov chains incrementally. Given a Markov chain M_n, we want to obtain a more refined Markov chain M_{n+1} without calculating all the transition probabilities of $n+1$ nodes. Using random sampling to create the set S_n ensures that it is very easy to make the construction incremental. Algorithm 2.3 uses the procedures introduced in the previous section to generate an incremental Markov chain. Note that we want to create a Markov chain which satisfies local consistency conditions given in Equations (2.5)- (2.7). The last two conditions are trivially satisfied by construction, this is proved in Theorem 2.3. In an incremental construction, we have to ensure that the first condition on holding times is also satisfied in the limit, i.e., as the number of samples in the Markov chain goes to infinity, we need to ensure that the holding time for *all* states goes to zero. This would be satisfied easily if we construct the chain from scratch for every new n which we want to avoid doing. Roughly, given S_n there are some states $z \in S_n$ with large holding times. As n increases we need to recalculate the transition probabilities of these states to ensure that the new holding times (which now depend on a larger value of n, see Section 2.4.1) keep decreasing. We do this by the following idea in Algorithm 2.3. Recalculating the probabilities for all the neighbors in the set Z_{near} for every newly sampled state z_{n+1} ensures that the holding times for *all* states are reduced asymptotically as proved in Theorem 2.5. As shown in Figure 2-3, transition probabilities for an old state z which lies within distance $r_2 = \gamma \left(\log(n+1)/(n+1)\right)^{1/d}$ are recalculated with a new state z_{n+1} is added to the Markov chain. Since $|Z_{near}| = \mathcal{O}(\log n)$, the complexity of recalculating probabilities (lines 7-8 in Algorithm 2.3) is $\mathcal{O}((\log n)^2)$. The incremental algorithm is thus only a factor $\mathcal{O}(\log n)$ worse than the batch construction.

Algorithm 2.3: Incremental construction

1 $n \leftarrow 0$;
2 **while** $n < N$ **do**
3 $\quad z \leftarrow$ `Sample`();
4 $\quad S_n \leftarrow S_{n-1} \cup \{z\}$;
5 $\quad (S_n, P_n, T_n) \leftarrow$ `ConnectState`$(z, (S_n, P_{n-1}, T_{n-1}))$;
6 $\quad Z_{\text{near}} \leftarrow$ `Near`(z, S_n);
7 $\quad$ **for** $z_{\text{near}} \in Z_{\text{near}}$ **do**
8 $\quad\quad (S_n, P_n, T_n) \leftarrow$ `ConnectState`$(z_{\text{near}}, (S_n, P_n, T_n))$;
9 $\quad n \leftarrow n + 1$;
10 **return** (S_N, P_N, T_N);

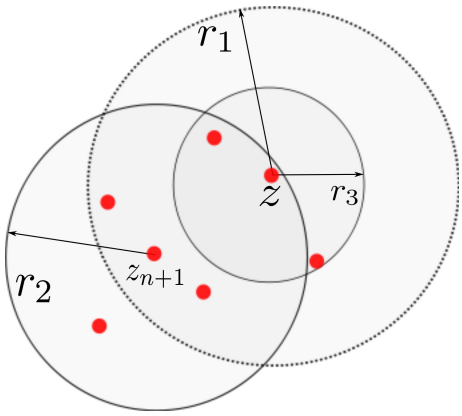


Figure 2-3: Incremental construction

2.5 Analysis

Theorem 2.2 (Theorem 10.4.1 in [KD01]). *Assume that $f(\cdot)$ and $F(\cdot)$ are bounded and continuous. Let $\{M_n;\ n \in \mathbb{N}\}$ be a sequence of Markov chains that are locally consistent with stochastic dynamical system described by Equation (2.1). For each $n \in \mathbb{N}$, let $\{\xi_n(t); t \in \mathbb{R}_{\geq 0}\}$ denote the continuous-time interpolation to the trajectory of M_n. Then, $(\xi_n(\cdot))$ has a subsequence that converges in distribution to $(x(\cdot))$ satisfying*

$$x(t) = x_0 + \int_0^t f(x(s))ds + \int_0^t F(x(s))\ dw(s),$$

where x_0 is distributed according to $\lim_{n\to\infty} \pi_n$, π_n being the prior distribution of the initial state on M_n.

Proof. We will only give a short sketch of the proof here. Refer Section 9.4 of [KD01] for complete proof. Let us first prove that the collection of sequences $\{\xi_n(\cdot);\ n \in \mathbb{N}\}$ is tight. Let $\xi_n(t_k) = \xi_n^k$. Suppress the subscript ξ_n for the purposes of the following derivation. Note that by construction,

$$\begin{aligned}\mathbb{E}[\Delta\xi^k] &= f(\xi^k)\ \Delta t(\xi^k)\\ \mathbb{E}[(\Delta\xi^k - \mathbb{E}[\Delta\xi^k])(\Delta\xi^k - \mathbb{E}[\Delta\xi^k])^T] &= F(\xi^k)\ F^T(\xi^k)\ \Delta t(\xi^k).\end{aligned}$$

Let $M = \max\{n : \sum_{k=1}^n \Delta t(\xi^k) \leq t\}$. Then,

$$\begin{aligned}\mathbb{E}[\xi_n(t)^2] &= \mathbb{E}\left[\left|\sum_{k=0}^{M} \mathbb{E}[\Delta\xi^k] + (\Delta\xi^k - \mathbb{E}[\Delta\xi^k])\right|^2\right]\\ &\leq 2\mathbb{E}\left[\left|\sum_{k=0}^{M} f(\xi^k)\ \Delta t(\xi^k)\right|^2\right] + 2\mathbb{E}\left[\sum_{k=0}^{M} F(\xi^k)F^T(\xi^k)\ \Delta t(\xi^k)\right]\\ &\leq 2K^2t^2 + 2Kt,\end{aligned}$$

where K is the common bound for $f(\cdot)$ and $F(\cdot)$. Finally the probability

$$\mathbb{P}(|\xi_n(t)| \geq a) \leq \frac{\mathbb{E}[\xi_n(t)^2]}{a^2} \leq \frac{2K^2t^2 + 2Kt}{a^2}$$

can be made arbitrarily small for any *fixed* value of t by varying a to create a compact set of the form $[-a, a]$. Thus we have proved that sequences $\xi_n(\cdot)$ are tight. Using Prokhorov's theorem we can now extract a subsequence of $\xi_n(\cdot)$ such that it converges to some $x(\cdot)$ say, i.e. $\xi_n(\cdot) \Rightarrow x(\cdot)$. Now construct a process $w_n(\cdot)$ as

$$w_n(t) = \sum_{k=0}^{M-1} \frac{\Delta\xi^k - \mathbb{E}[\Delta\xi^k]}{F(\xi^k)}$$

which is essentially an approximation of the Wiener process constructed by $\xi_n(\cdot)$. It can be checked that this $w_n(\cdot)$ converges to the actual Wiener process. Finally we claim that the solution of the stochastic differential equation is unique in the weak sense and it is constructed by the same Wiener process $w(\cdot)$. Thus the subsequence that we extracted above converges weakly (see Section A in Appendix A) to the solution of Equation (2.1). ■

Lemma 2.3. *The Gaussian approximation presented in the* `ComputeTransProb` *procedure satisfies the local consistency conditions given in equations* (2.6) *and* (2.7).

Proof. Let $\phi(x(t))$ denote the probability density without observations of the stochastic dynamics given in Equation (2.1). The Fokker-Plank equation can then be used to compute this density as

$$\frac{\partial}{\partial t}\phi(x(t)) = \left[-\frac{\partial}{\partial x}f(x) + \frac{1}{2}\,\frac{\partial}{\partial x}F(x)F^T(x)\right]\phi(x(t)).$$

The solution for small times Δt can then be written to obtain [Ris96],

$$P(x', t + \Delta t \,|\, x, t) = \frac{1}{\sqrt{2\pi F(x)F^T(x)\Delta t}}\ \exp\left(-\frac{1}{2}\frac{[x' - x - f(x,t)\Delta t]^2}{F(x)F^T(x)\Delta t}\right)$$

The Gaussian for the transition probabilities in the `ComputeTransProb` procedure is thus the small-time solution of the Fokker-Planck equation. Also, it can be proved that the number of samples in the neighborhood of every sample (in every grid cell $G_n(i)$ of Theorem 2.4 to be precise) is increasing [KF11], i.e., the small-time solution of the Fokker-Plank equation becomes exact as $n \to \infty$. ■

Theorem 2.4. *The Markov chain* (S_n, P_n, T_n) *returned by Algorithm 2.1 is locally consistent with the stochastic dynamical system described by Equation* (2.1)*, with probability one.*

Proof. The transition probabilities are consistent by Lemma 2.3. Hence, we only need to prove that using Algorithm 2.1, every state has non-zero probability of transition to another state, i.e. every state is connected to at least one other state or that the Markov chain is irreducible. The analysis here is similar to the analysis in [KF11, HKF12].

For each $n \in \mathbb{N}$, divide the state space $\mathcal{S}$ into grid cells with side length $\frac{\gamma}{2}(\log n/n)^{1/d}$ as follows. Define the grid cell $G_n(i)$ for $i \in \mathbb{Z}^d$ as

$$i \left(\frac{\gamma}{2} \frac{\log n}{n}\right)^{1/d} + \left[-\frac{1}{4}\gamma \left(\frac{\log n}{n}\right)^{1/d}, \frac{1}{4}\gamma \left(\frac{\log n}{n}\right)^{1/d}\right]^d,$$

where $[-a, a]^d$ denotes the d-dimensional cube with side length $2\,a$ centered at the origin. Hence, the expression above translates the d-dimensional cube with side length $\frac{\gamma}{2}(\log n/n)^{1/d}$ to the point with coordinates $i\,\frac{\gamma}{2}(\log n/n)^{1/d}$. Let K_n denote the indices of set of all cells that lie completely inside the state space $\mathcal{S}$, i.e., $K_n = \{i \in \mathbb{Z}^d : G_n(i) \subseteq \mathcal{S}\}$.

We claim that for all large n, all grid cells in K_n contain at least one vertex of S_n. Given an event A, let A^c denote its complement. Let $A_{n,k}$ denote the event that the cell $G_n(k)$ contains a vertex from S_n. Then, for all $k \in K_n$,

$$\begin{aligned}
\mathbb{P}\left(A_{n,k}^c\right) &= \left(1 - \frac{(\frac{\gamma}{2})^{-d}}{\mu(\mathcal{S})} \frac{\log n}{n}\right)^n \\
&\leq \exp\left(-((\frac{\gamma}{2})^d/\mu(\mathcal{S}))\, \log n\right) \\
&= n^{-(\frac{\gamma}{2})^d/\mu(\mathcal{S})},
\end{aligned}$$

where $\mu(\mathcal{S})$ denotes the Lebesgue measure assigned to $\mathcal{S}$. Let A_n denote the event that all cells $G_n(i)$ contain at least one vertex of S_n. Then,

$$\begin{aligned}
\mathbb{P}(A_n^c) = \mathbb{P}\left(\left(\bigcap\nolimits_{k\in K_n} A_{n,k}\right)^c\right) &= \mathbb{P}\left(\bigcup\nolimits_{k\in K_n} A_{n,k}^c\right) \\
\leq \sum\nolimits_{k\in K_n} \mathbb{P}\left(A_{n,k}^c\right) &= |K_n|\, n^{-(\frac{\gamma}{2})^d/\mu(\mathcal{S})},
\end{aligned}$$

where the first inequality follows from the union bound and $|K_n|$ denotes the cardinality of the set K_n. Merely calculating the maximum number of cubes that can fit into $\mathcal{S}$, the latter quantity can be bounded by

$$|K_n| \leq \frac{\mu(\mathcal{S})}{(\frac{\gamma}{2})^d \frac{\log n}{n}} = \frac{\mu(\mathcal{S})}{(\frac{\gamma}{2})^d} \frac{n}{\log n}.$$

Hence,

$$\begin{aligned}
\mathbb{P}\left(A_n^c\right) &\leq \frac{\mu(\mathcal{S})}{(\frac{\gamma}{2})^d} \frac{n}{\log n}\, n^{-(\frac{\gamma}{2})^d/\mu(\mathcal{S})} \\
&\leq \frac{\mu(\mathcal{S})}{(\frac{\gamma}{2})^d}\, n^{1-(\frac{\gamma}{2})^d/\mu(\mathcal{S})},
\end{aligned}$$

which is summable for all $\gamma > 2\ (2\,\mu(\mathcal{S}))^{1/d}$. Hence, by the Borel-Cantelli lemma, the probability that A_n^c occurs infinitely often is zero, which implies that the probability that A_n occurs for all large n is one. Since the radius of the ball in the procedure `Near` is $\gamma(\frac{\log n}{n})^{1/d}$, every state z is connected to at least one other state. Finally, since, $\Delta t(z) \to 0$

as $n \to \infty$ we have proved that Algorithm 2.1 is locally consistent. ■

Theorem 2.5. *Incremental construction of the approximating chain using Algorithm 2.3 is also locally consistent for large n, with probability one.*

Proof. The proof of connectivity of the Markov chain is the same as the proof of Theorem 2.4 whereas Equations (2.6)-(2.7) are satisfied for any state $z \in \cup_{i\in\mathbb{N}} S_i$ by Lemma 2.3. We only to show that Equation (2.5) is satisfied, i.e, $\Delta t(z)$ for any state z that is added to the Markov chain at some iteration, say i, goes to zero. Note that, calling `ConnectState` on an existing state always results in reduction of $\Delta t(z)$ because n is increasing. We thus essentially prove that z is reconnected to its neighbors (which are changing with n) infinitely often.

Fix some iteration i and some state $z \in S_i$. Let A_n, defined for all $n > i$, denote the event that the state z belongs to $\texttt{Near}(z_n, S_n)$ of the newly node z_n at iteration n. It is thus inside the ball of volume $\gamma^d(\frac{\log n}{n})$ centered at z_n. Hence, $\mathbb{P}(A_n) = \frac{\gamma^d}{\mu(\mathcal{S})}(\frac{\log n}{n})$. Since $\sum \mathbb{P}(A_n) = \infty$ and the event A_n is independent from A_i for all $i \neq n$, Borel-Cantelli lemma implies that $\mathbb{P}(\limsup_{n\to\infty} A_n) = 1$. Hence, any state z is reconnected infinitely often, with probability 1. ■

Theorems 2.2, 2.4, and 2.5 imply that the trajectories of the successive Markov chains (S_n, P_n, T_n) converge in distribution to the trajectories of the system described by Equation (2.1).

2.6 Experiments

This section is devoted to some experiments that empirically verify the convergence of Markov chain approximations. We test the construction on an example linear system and show that the distribution of states at any time t converges to the actual distribution for the original system (which can be calculated in closed form for a linear system).

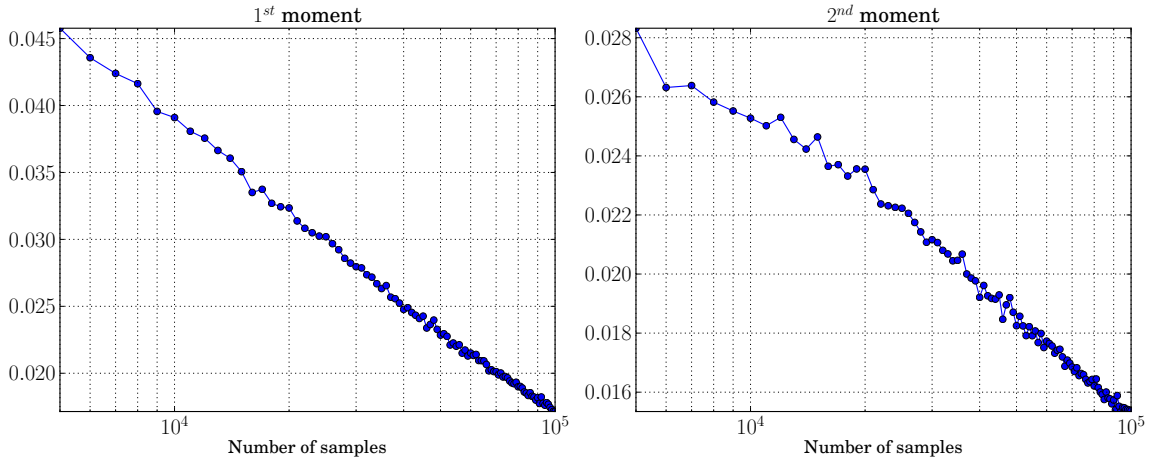
We can numerically verify the results of Theorems 2.4 and 2.5 by a Monte-Carlo simulation. Since the distributions of trajectories converge, the distribution of states at any fixed time t also converges. Also, by definition, the moments of the distributions of states at any time t converge, which we will verify. Consider a 2-dimensional single integrator with drift but no observations,

$$\begin{aligned} dx_1 &= -\frac{x_1}{2}\, dt + 0.03\, dw_1 \\ dx_2 &= -x_2\, dt + 0.03\, dw_2 \end{aligned} \tag{2.9}$$

Figure 2-5 simulates 50,000 trajectories of the Markov chain and the actual system dynamics until time $T = 2$ and looks at the distribution of states at five specific time instants. The scatter plots in Figure 2-5 show the distribution of states (x_1, x_2) of the Markov chain at five specific time instants $t \in \{0, 0.3, 0.5, 1.0, 2.0\}$ secs. Translucent ellipses are 3σ ellipses from the simulation of the stochastic system as given in Equation (2.9). The dotted blue and red lines show the mean of the actual and Markov trajectories respectively for $t \in [0, 2]$ secs. The mean trajectories converge, i.e. the first moment of the distribution converges

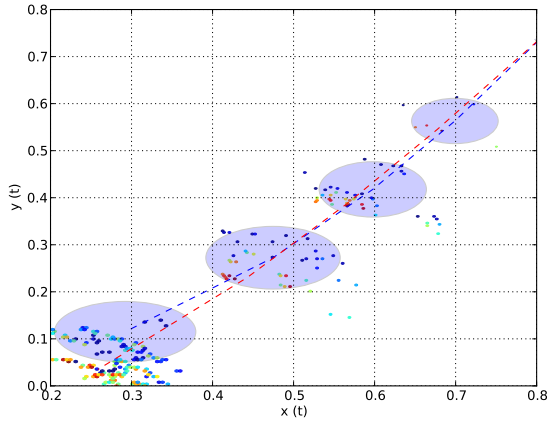
as more samples are added. The variance shown as a scatter plot also converges. Both the Markov chain and the original system are started from the nearest state to $(0.8, 0.8)$ in S_n.

We can also compare the moments of the distribution of the states $x(T)$ and $x_{\text{Markov}}(T)$ for different number of states in the Markov chain. Figure 2-4 shows the convergence of the error in the first two moments calculated over $50,000$ trajectories with increasing number of states in the Markov chain ranging from $1,000$ to $100,000$.

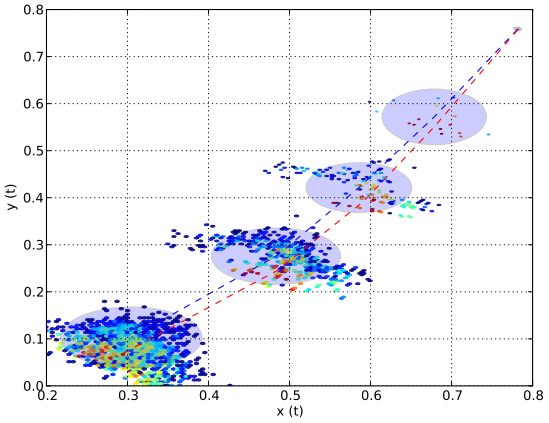


(a) (b)

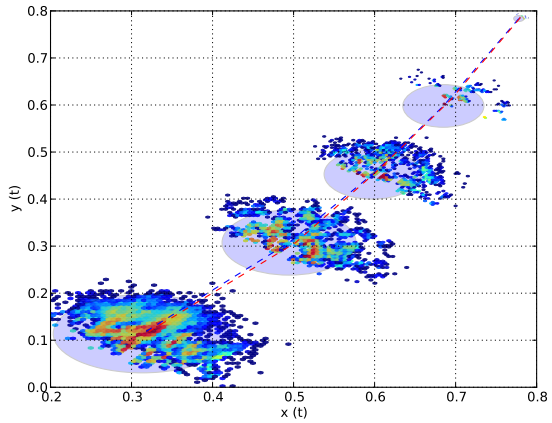
Figure 2-4: Figure (a) shows $|\mathbb{E}[\xi_n(T)] - \mathbb{E}[x(T)]|$ versus the number of samples n while Figure (b) shows a similar plot for the 2^{nd} moment, i.e.$\|\mathbb{E}[\xi_n(T)\xi_n^T(T) - x(T)x^T(T)]\|_2$.



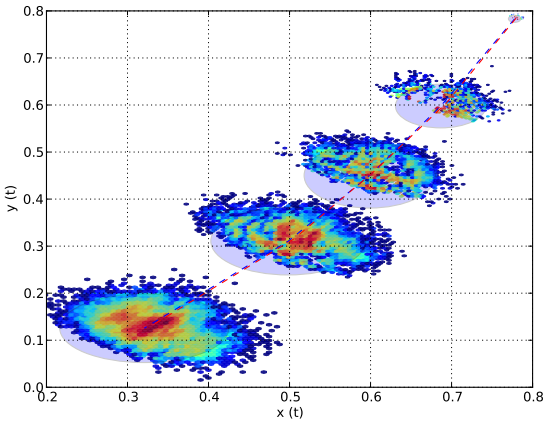
(a) 500 samples



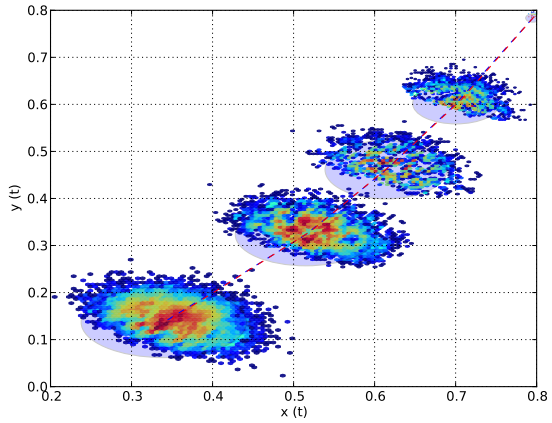
(b) 1000 samples



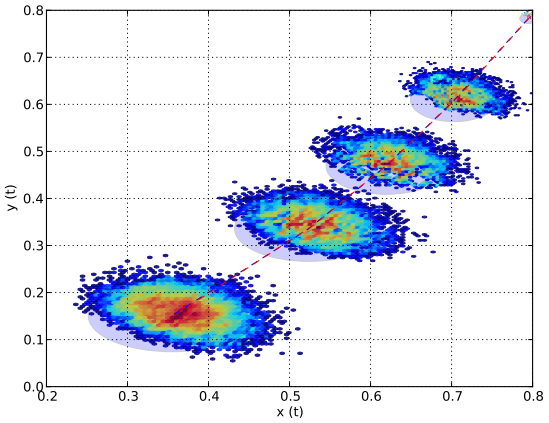
(c) 5,000 samples



(d) 10,000 samples



(e) 20,000 samples



(f) 40,000 samples

Figure 2-5: Convergence in distribution of the trajectories for a 2-dimensional single integrator.

Chapter 3

Filtering

After the problem of estimating the state of a system on the basis of noisy observations was first introduced in the works of Kolmogorov and Wiener, filtering has become one of the most fundamental problems in stochastic control and communications literature. The underlying system is assumed to be continuous-time and continuous state-in some cases whereas it is discrete in some others. The primary motivation in this chapter is to create a filtering algorithm for a general class of systems, in particular, systems with continuous time and continuous state space. We are interested in algorithms that have essentially no parameters that need to be tuned to achieve satisfactory performance, i.e., we want state estimation algorithms that are both *anytime* and *incremental* in nature. This means that instead of a practitioner having to tune the algorithm to work on platforms with different computational capacity, the algorithm gives the best solution it can in the computational capacity it is given. The state estimate is improved upon if more computational resources are available. It utilizes the idea introduced in Chapter 2 that continuous time stochastic systems can be approximated by sequences of discrete-time finite Markov chains.

This chapter is organized as follows. The introductory section 3.1 describes existing literature and motivates the solution approach used for the filtering problem. The continuous time filtering problem is formulated in Section 3.2. Section 3.3 discusses a modification of the Markov chain approximations constructed in Chapter 2 and uses them to give an incremental algorithm for filtering. The resulting solution is tested on a number of examples in Section 3.4 and compared to other state-of-the-art algorithms.

3.1 Previous approaches

This section discusses various different approaches that solve the filtering problem. We concentrate on the many different fields that have given rise to state of the art algorithms. In the end, we will take a holistic view of all these methods which will motivate the essentials of a new class of algorithms for filtering.

Kalman filter Arguably, the Kalman filter dominates most real world applications in signal processing and optimal state estimation. It is a provably optimal filter for linear systems with additive Gaussian noise. As we will see in the following paragraphs, the

filtering problem does not always lend itself to a nice computable finite dimensional solution. Kalman filter is a special case in which it is possible to do so. The Extended Kalman filter linearizes the dynamics and observation equations at every step and approximates the state-estimate of the non-linear system using the Kalman filter. What it gains in generality, the EKF loses in optimality. The linearization procedure results in the EKF no longer being optimal for non-linear systems. Also, although EKF accurately propagates the mean of the conditional distribution of state given observations, it was shown by Julier and Uhlmann in [JU97] that linearization results in very poor approximation of the posterior covariance. They introduced the Unscented Transform to approximate the conditional distribution by a small set of *deterministically chosen "sigma points"* and propagate these points to give the Unscented Kalman filter (UKF). Different methods of choosing sigma points can preserve arbitrary number of moments of the conditional distribution and thus UKF is very accurate in practice. It effectively exploits the symmetry of the conditional distribution (in cases where it is Gaussian say) to vastly reduce the computational complexity of covariance propagation.

PDE based approaches Consider the drift-diffusion Equation (2.1) in Chapter 2,

$$dx(t) = f(x(t))\ dt + F(x(t))\ dw. \tag{3.1}$$

If observations of this system are obtained as

$$dy(t) = g(x(t))\ dt + G(x(t))\ dv, \tag{3.2}$$

where $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^k$, $G(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d\times n}$ and $v(t)$ is the standard n-dimensional Wiener process, the propagation of the conditional density of the state given the observations, i.e. $\pi(t) = \mathbb{P}(x(t) \,|\, \mathcal{Y}_t)$ where $\mathcal{Y}_t = \sigma(y(s);\ s \le t)$ is the σ-field generated by the observation process, is given by a stochastic partial differential equation of the form

$$d\pi = \mathcal{L}_f(\pi(t))\ dt + g(x(t))\pi(t)dy(t).$$

In the above equation, $\mathcal{L}_f$ is the forward Fokker Plank operator defined in Equation (2.3), and $\pi(t)$ is the unnormalized conditional density in this equation also known as the Zakai equation. One way of then solving the non-linear filtering problem is to search for numerical solutions for this PDE. Grid based methods discussed in [Kus77] are a precursor to a variety of other methods such as separation of variables, adaptive local grids, quadrature methods in this area. However, it is generally seen that these algorithms are neither very computational efficient nor are they recursive. This hampers their usage in most real-world applications where filtering can be made recursive so as to enable quick computation.

Exact filters It turns out that there exist some systems for which we do not have to explicitly solve the Zakai equation. Exact filters are a class of filters where it is possible to obtain an *optimal finite-dimensional filter* even for non-linear systems. Kalman filter also belongs to this class with a dimension of $d + d(d+1)/2$ because it propagates only the mean and the symmetric covariance matrix. Exact filters are built upon a result by Fisher which says that the only distributions with finite-dimensional sufficient statistics are

from the exponential family. Benes in [Ben81] showed that if the drift term $f(x)$ can be written as a gradient, or equivalently if $\frac{\partial f}{\partial x} = \left(\frac{\partial f}{\partial x}\right)^T$, the propagation equations of the filter are finite dimensional. This case however does not cover the Kalman filter completely and there are extensions to cases where the drift is the solution of a Riccati equation for example. See [Dau05] for a thorough exposition on exact filters as well as a general overview of filtering literature.

Monte Carlo methods In many real-world problems, nonlinearities and non-Gaussian noise prevent us from getting closed-form expressions for the optimal filter. The seminal paper [GSS93] introduced the bootstrap filter which forms the basis for a class of general filters based on sequential Monte-Carlo methods [DDFG01] known as particle filters. They utilize a large number of random samples (called particles) to represent arbitrary posterior distributions and are propagated in time using importance sampling techniques. The crucial aspect of particle filtering is estimating a good posterior distribution to sample from. These filters are very easy to implement and have good convergence properties inherited from Monte Carlo algorithms. On the other hand, it is necessary to tune the filtering algorithm to the given problem for robust performance [PS99]. Sparse particles in systems with large nonlinearities can result in divergence of the estimated posterior. Techniques like resampling posterior [DC05] to reduce variance of particles and adaptive sampling [Fox03] result in the algorithm being flexible and applicable to a wide class of nonlinear and non-Gaussian models. Altogether, it is essential to tune these algorithms to work on different systems and given the huge number of different techniques to do so, it is difficult to quantify which technique is the best for a particular problem.

Continuous time filtering Continuous-time filtering algorithms have also received wide attention in literature, starting from the Kalman-Bucy filter for continuous-time linear systems with Gaussian additive noise. More recent results on continuous time particle filters are inspired by weak approximations of solutions of stochastic differential equations (SDEs) and come with explicit rates of convergence [Cri06]. Elsewhere, branching and interacting particle systems in continuous time have also been applied to the nonlinear filtering problem [DMM00]. Numerical solutions to the partial differential equations arising from the Zakai equation and the Kushner-Stratonovich equation have been used to perform continuous time nonlinear filtering [Kus77]. These applications of these algorithms are however limited due to computational intractability or non-recursive nature.

Discussion State-of-the-art filtering algorithms like particle filters and UKFs are notoriously difficult to implement on real systems. A part of this complexity comes from the fact that they contain a number of tunable parameters like (i) number of particles, (ii) time discretization of integration or (iii) assumptions about the conditional density (e.g., Gaussian). It is seldom clear beforehand how many particles are necessary for localization of a robot using range sensors or how small should the integration step of the dynamics should be to ensure that the covariance is propagated accurately. These simple issues often lead to divergence of filtering algorithms in real-world scenarios. In order to reduce the instances of divergence, we often add a larger number of particles which results in an increase

in computational complexity which in turn results in a larger integration time step if the computational resource is kept constant. This conundrum does not always have a precise solution although techniques like adaptive sampling [Fox03] alleviate it to some extent.

It is the primary aim of this chapter to find a satisfactory solution to this multiplicative effect. In particular, it will be our aim to design an *incremental* and *anytime* state estimation algorithm. We leverage recent results in asymptotically optimal motion planning algorithms and apply them to construct incrementally refined Markov chain approximations of stochastic systems. The key idea behind this construction is to discretize time and state space at a rate similar to that of PRM* and the RRT* algorithms, and get the transition probabilities of the chain using local consistency ideas of the Markov chain approximation method. We prove that the trajectories of these successive approximations converge in distribution to the trajectories of the original stochastic dynamical system. We propose an algorithm to solve the nonlinear optimal filtering problem using these approximations. The resulting algorithms draw their features from both motion planning and Markov chain approximation method and are, (i) fairly general, i.e., designed for a large class of stochastic dynamical systems, (ii) easy to implement even for complex dynamical systems and (iii) do not need to be explicitly tuned for different problems or for platforms with different computational capabilities due to their incremental nature.

3.2 Problem Definition

Let $\mathbb{R}$ denote the set of real numbers and $\mathbb{R}^{n\times k}$ denote the set of all $n \times k$ real valued matrices. Consider a stochastic differential equation of the form

$$dx(t) = f(x(t))\,dt + F(x(t))\,dw(t), \quad x(0) = x_0, \tag{3.3}$$

where (i) $x(t) \in \mathbb{R}^d$ for all $t \geq 0$, (ii) $f : \mathbb{R}^d \to \mathbb{R}^d$, $F : \mathbb{R}^d \to \mathbb{R}^{d\times k}$ are Borel-measurable functions, (iii) the stochastic process $\{w(t) : t \geq 0\}$ is the standard k-dimensional Brownian motion, and the random variable x_0 is bounded with probability one. A solution to the differential form presented in Equation (3.3) is a stochastic process $\{x(t) : \ t \geq 0\}$ that constitutes a solution to the following integral equation:

$$x(t) = x_0 + \int_0^t f(x(\tau))d\tau + \int_0^t F(x(\tau))\ dw(\tau),\ \forall\ t \geq 0,$$

where the second term on the right hand side is the usual Itô integral [Øks03]. We tacitly assume throughout the paper that the functions $f(\cdot)$ and $F(\cdot)$ are bounded and continuous functions to guarantee weak existence and weak uniqueness for the solutions of Equation (3.3).

In the standard nonlinear filtering problem [KD01, Øks03] one attempts to estimate the process $\{x(t); t \geq 0\}$ using data available till time t, defined by $\mathcal{Y}_t := \{y(s) : s \leq t\}$, where $\{y(t) : t \geq 0\}$ is a solution to the stochastic differential equation of the form

$$dy(t) = g(x(t))\,dt + G(x(t))dv(t), \tag{3.4}$$

where $g : \mathbb{R}^d \to \mathbb{R}^m$ and $G : \mathbb{R}^d \to \mathbb{R}^{m\times l}$ are Borel-measurable functions, and $\{v(t) : t \geq$

$0\}$ is a l-dimensional Brownian motion independent of the stochastic process $\{x(t) : t \geq 0\}$. Similarly, we assume that the functions $g(\cdot)$ and $G(\cdot)$ are bounded and continuous to guarantee weak existence and weak uniqueness of solutions to Equation (3.4). As in [KD01], we formulate the problem such that the system evolves inside a compact subset, denoted by $\mathcal{S}$, of $\mathbb{R}^d$. The process is stopped if it hits the boundary of $\mathcal{S}$. That is, define

$$\tau := \inf\{s : x(s) \notin \mathcal{S}^o\},$$

where, $\mathcal{S}^o$ denotes the interior of $\mathcal{S}$. Then, the definition of the problem is given as follows.

Problem 3.1. *Given a set $\mathcal{Y}_t := \{y(s) : s \leq t\}$ of observations generated by process (3.4), find an estimate $\hat{x}(t)$ such that (i) $\mathbb{E}[\|x(t)-\hat{x}(t)\|^2]$ is minimized and (ii) the random variable $\hat{x}(t)$ is square integrable and H_t-measurable, where H_t is the σ-algebra generated by $\mathcal{Y}_t$.*

It is well known that the error-minimizing state estimate $\hat{x}(t)$ based on observations $\mathcal{Y}_t$ is

$$\hat{x}(t) = \mathbb{E}[x(t) \,|\, \mathcal{Y}_t].$$

In fact, this equation forms the basis of the Fujisaki-Kallianpur-Kunita equation of filtering theory [Øks03]. In some references the filtering problem is posed as the estimation of the distribution of the random variable $\mathbb{E}[x(t) \,|\, H_t]$ (see, e.g., [KD01]), which is equivalent to our formulation of the problem. Let us also note that the solution of the filtering problem can be given as follows [Kus67]. If $\tilde{x}(t)$ is a process with the same distribution as that of $x(t)$ but independent of $(x(t), y(t))$, and $\phi(\cdot)$ is any continuous real-valued function, then the solution of the filtering problem is,

$$\mathbb{E}[\phi(x(t)) \,|\, \mathcal{Y}_t] = \frac{\mathbb{E}[R(t)\phi(\tilde{x}(t)) \,|\, \mathcal{Y}_t]}{\mathbb{E}[R(t) \,|\, \mathcal{Y}_t]}, \tag{3.5}$$

where

$$R(t) = \exp\left[\int_0^t g(\tilde{x}(s))^T dy(s) - \frac{1}{2}\int_0^t |g(\tilde{x}(s))|^2 \, ds\right].$$

Our approach in this chapter uses the Markov chain approximation method to generate a process $\tilde{x}(t)$ which has the same law as the original process $x(t)$. Equation (3.5) which is really the limiting version of Bayes' rule then gives the optimal filter.

Example 3.2 (Optimal filter on Markov chain). The optimal filtering problem for a discrete Markov chain is very similar to Problem 3.1. Given a Markov chain $\mathcal{M} = (S, P)$ and a set $Y_k = \{y_i : i = 1, 2, \ldots, k\}$ of discrete-time observations coming from an equation of the form $y_k = g(\xi_k) + G(\xi_k)\tilde{v}$, where, $\tilde{v}$ is unit-variance white Gaussian noise, we can calculate the conditional distribution $\phi_k(z) = \mathbb{P}(\xi_k = z \,|\, Y_k)$ to be

$$\phi_n(z) = \sum_{z' \in S} \mathbb{P}(\xi_n = z, \xi_{n-1} = z' \,|\, Y_n),$$

which can be written using recursive Bayes' rule as

$$\phi_n(z) = \eta \sum_{z' \in S} \mathbb{P}(y_n \,|\, \xi_n = z, \xi_{n-1} = z') \; \mathbb{P}(\xi_n = z \,|\, \xi_{n-1} = z') \; \phi_{n-1}(z'), \tag{3.6}$$

where η is a normalization constant and $\phi_0(z)$ is the initial distribution of states. Note that the probability $\mathbb{P}(y_n \,|\, \xi_n = z, \xi_{n-1} = z')$ becomes $\mathbb{P}(y_n \,|\, \xi_n = z)$ under our observation equation. This formulation is similar to estimation on Hidden Markov Models except for the fact that observations come from an observation space instead of a finite set. So as long as we can calculate $\mathbb{P}(y_n \,|\, \xi_n = z)$, the same formulae hold. Also note that the observations in Equation (3.4) are often approximated [Kus08] by discretizing them at times $k\delta$ as

$$y_k = g(x(k\delta))\delta + G(x(k\delta))[v(k\delta) - v(k\delta - \delta)],$$

which is the conventional discrete observation model

$$y_k = \tilde{g}(x_k) + G(x_k)\ \tilde{v}_k,$$

with $\tilde{g}(x_k) = g(x(k\delta))\ \delta$ and $\tilde{v}_k = v(k\delta) - v(k\delta - \delta)$ being the white Gaussian noise constructed from the continuous-time Brownian motion $v(t)$. Note that $\tilde{v}_k$ need not be Gaussian noise in the above equation; $v(t)$, however, needs to be Brownian motion to formally guarantee existence and uniqueness of solution to Equation (3.4).

3.3 Filtering on Markov chain approximations

3.3.1 Modified Markov chain for filtering

We use Equation (3.6) to propagate estimates on the Markov chain constructed in Section 2.2. This requires that we know the single step transition probabilities, i.e., roughly the holding times of all states in the discrete Markov chains need to be the same. Transition probabilities and holding times that result in a locally consistent chain can be scaled appropriately to adapt the construction in Chapter 2 to the filtering problem. In Chapter 4, we will see a different type of scaling. The constructions given below are inspired from traditional finite-difference methods of solving partial differential equations. We will discuss two methods, which we call explicit and implicit, respectively.

Explicit construction The only requirement is that the set of probabilities and holding times satisfy local consistency conditions, i.e., Equations (2.6)- (2.7). Denote the new transition probabilities by $p^\delta(\cdot)$ where δ is the time interval to which we want to equalize all the holding times. It can be shown using the example of 2.3 that (i) if the terminal time is finite, the cost function $W(x,t)$ satisfies the partial differential equation

$$\frac{\partial}{\partial t}W(x,t) + \mathcal{L}_f W(x,t) + l(x) = 0, \tag{3.7}$$

and (ii) a similar analysis as that of 2.3 shows that the new probabilities are scaled versions of old probabilities, i.e.,

$$\frac{p^\delta(z' \,|\, z)}{1 - p^\delta(z \,|\, z)} = p(z' \,|\, z) \tag{3.8}$$

This is roughly explained by the fact that the only difference between Equation 2.8 and Equation 3.7 above is the presence of the term $\frac{\partial}{\partial t}W(x,t)$ which results in a self-transition

probability of the form $p^\delta(z \mid z)$ which gives the scaling factor. $(p(\cdot), \Delta t(z))$ and$(p^\delta(\cdot), \delta)$ then satisfy the same consistency conditions. Equation (2.6) then gives

$$1 - p^\delta(z \mid z) = \frac{\delta}{\Delta t(z)} \tag{3.9}$$

Additionally, we require the following condition to ensure that $p^\delta(z \mid z) \leq 1$ for all $z \in S$.

$$\delta \leq \min_{z \in S} \Delta t(z).$$

Equations 3.8 and 3.9 together give the modified probabilities provided the above condition is satisfied. The new holding time is of course equal to δ. Figure 3-1a shows how the explicit chain is constructed. If the X-axis depicts time, a state z connects to its neighbors after taking a step δ. Given a state $z \in S_n$ and a finite set $Z_{\text{near}} \subset S_n$, the `ComputeTransProb` procedure is replaced by a new `ComputeTransProbTime`$(z, Z_{\text{near}}, \Delta t, \delta)$ that returns a probability density function over $T_\delta \times Z_{\text{near}}$, where $T_\delta = \{0, \delta, 2\delta, \dots, \}$. This probability density is denoted by $p^\delta(\cdot \mid k\delta, z)$ defined for $z \in S$ and $k \in \{1, 2, \dots\}$ for time-inhomogeneous Markov chains. We will continue to use the notation $p(\cdot \mid z)$ in the later sections. Having thus ensured that the holding time from every state to every other state is the same, we can use Bayes' rule given in Equation (3.6) to update the conditional distribution of the filter given a new observation.

Implicit construction The difference between explicit and implicit method is that the explicit method treats the time variable explicitly. In other words, time increases by a fixed amount δ at every step which transitions between states take place with the corresponding transition probabilities. The generality of local consistency conditions indicates that this is not the only way of satisfying them. Roughly, we ensure that the spatial component i.e. $\xi(\cdot)$ satisfies local consistency conditions while the time component i.e. $\Delta t(z)$ changes probabilistically.

The procedure starts with an equation similar to Equation 3.8,

$$\frac{p^\delta(z'; k\delta \mid z; k\delta)}{1 - p^\delta(z; k\delta + \delta \mid z; k\delta)} = p(z' \mid z) \tag{3.10}$$

where $p^\delta(z'; k\delta \mid z; k\delta)$ denotes the probability of transitioning to another state with the time component of the chain advancing. In the same way, the conditional increment of the time component with the state being same is actually

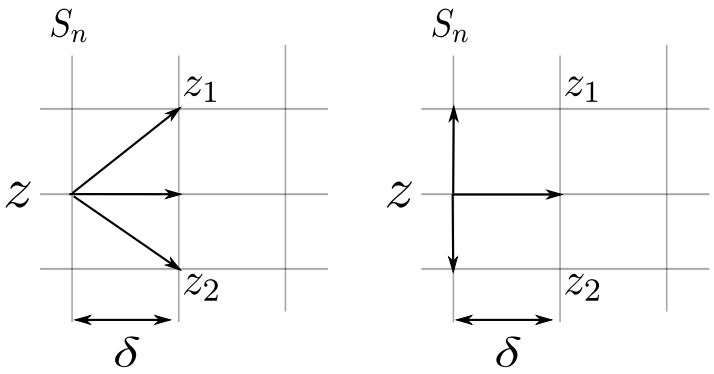
$$\Delta t^\delta(z) = p^\delta(z; k\delta + \delta \mid z; k\delta)\ \delta.$$

We will not actually "equalize" the holding times using this method (refer [Kus08] for that) but suffice here by providing modified probabilities for the appended state-space for later use. Thereby, the above equation actually *defines* the new holding time $\Delta t^\delta(z)$. Substitute

in local consistency conditions to get

$$p^{\delta}(z; k\delta + \delta \,|\, z; k\delta) = \frac{\Delta t(z)}{\Delta t(z) + \delta}$$
$$\Delta t^{\delta}(z) = \frac{\delta\ \Delta t(z)}{\Delta t(z) + \delta}. \tag{3.11}$$

Equation (3.11) along with Equation (3.10) defines the new transition probabilities and interpolation intervals.



(a) Explicit chain (b) Implicit chain

Figure 3-1: The Markov chain for filtering can be built in two ways. The first one incorporates the time part explicitly whereas the second one called the implicit method calculates the transition probabilities implicitly in the sense that chain is built on a space $\mathcal{S} \times [0, T]$. The time component of the state also transitions probabilistically in the implicit construction.

3.3.2 Incremental construction

Equation (3.9) suggests that $\delta \leq \min_{z \in S_n} \Delta t(z)$. If the Markov chain is obtained from Algorithm 2.1, we fix a $\delta = \min_{z \in S_n} \Delta t(z)$ and modify the transition probabilities of every state $z \in S_n$ using Equations (3.8) and (3.9). On the other hand, if the Markov chain is being constructed incrementally using Algorithm 2.3, we cannot fix such a δ because $\Delta t(z)$ is decreasing as $n \to \infty$. Instead, we incrementally reduce the time discretization as $\delta_{\text{new}} = \delta_{\text{current}}/2$ and recalculate probabilities for all states in S_n every time we add a new state z_{n+1} that has $\Delta t(z_{n+1}) \leq \delta_{\text{current}}$. Since $\delta \sim \Delta t(z) = \mathcal{O}((\frac{\log n}{n})^{2/d})$ from Section 2.4.1, two successive values of n, n_1, and n_2 when we have to recalculate the transition probabilities are such that $n_2 \sim n_1 2^{d/2}$ which gives an amortized complexity of $\mathcal{O}(n(\log n)^2)$. This technique is called *exponential backup*.

Theorem 3.3 (see Theorem 4.1 in [Kus08]). *For any continuous real-valued function $\phi(\cdot)$, for any $T < \infty$, if $\xi_n(\cdot)$ is a sequence which converges in distribution to the solution to Equation (3.3) i.e., $x(t)$, as $n \to \infty$ and is independent of $(x(t), y(t))$, then,*

$$\lim_{n \to \infty} \sup_{t \leq T} \left| \mathbb{E}[\phi(\xi_n(t)) \,|\, \mathcal{Y}_t] - \mathbb{E}[\phi(x(t)) \,|\, \mathcal{Y}_t] \right| = 0.$$

The above theorem coupled with the formula given in Equation (3.5) proves that the filtered density calculated on the Markov chain M_n converges to the optimal nonlinear filtering density as $n \to \infty$ and $\delta \to 0$.

3.3.3 Heuristics

The algorithms proposed in Chapter 2 are general and can be used to get a discrete approximation of a large class of stochastic systems. This chapter provides a way to apply them to the problem of nonlinear filtering. There are a number of heuristics specific to the filtering problem applied to our Markov chain construction that can vastly improve the computational complexity in practice. As proposed in Algorithms 2.3 and 2.1, we sample the bounded state-space uniformly. Roughly, this results in the convergence rate depending upon the size of state-space. In order to avoid this, we can concentrate the samples around the estimated posterior to create the Markov chain. This estimated posterior can come for example from an EKF prediction. It is not necessary that it be accurate since the exploring properties of the Markov chain construction will result in a dense sampling of the state-space around the approximated posterior. In the examples given in the following section, the mean and variance of the prior (assumed to be Gaussian) are propagated for a time δ to get the posterior which is used to concentrate samples. It is seen that this vastly improves the computational complexity of the filtering algorithm.

Creating Markov chain approximation using Algorithm 2.1 and 2.3 is $\mathcal{O}(n \log n)$ and $\mathcal{O}(n(\log n)^2)$ respectively as discussed in Chapter 2. A particle filter for n particles is easily seen to be $\mathcal{O}(n)$ [**?**]. Thus the filtering algorithms proposed in this chapter are a factor $\mathcal{O}(\log n)$ or $\mathcal{O}((\log n)^2)$ worse than the particle filter for the same number of particles n. It should be noted however that they not only provide a state estimate but explicitly construct a discrete approximation of the continuous time system as well. In addition to this, heuristics discussed above vastly improve the computational complexity in practice while providing the same theoretical guarantees. Experiments in the following section show that the proposed algorithms result in lower average state-estimate error for the same number of particles.

3.4 Experiments

We compare the proposed filter with other filtering algorithms like EKF and particle filter on a number of examples in this section.

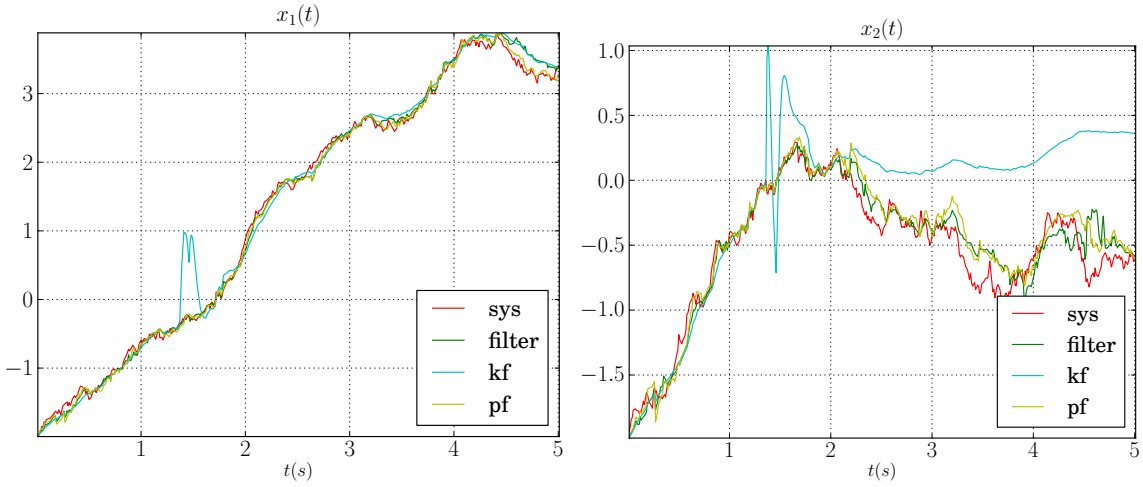
3.4.1 Drifting ship

Consider a drifting ship [KB00] confined to move within a disc of radius 9 units. A large force $f_i(x(t))$ acts on the ship to make it move inwards if it is moving outwards when it goes out of this disc as shown in Equation (3.12). The ship is like a 2-dimensional double integrator with forces $f_1(x)$, $f_2(x)$ with observations being range and heading as given in Equation (3.13). Figures 3-2 shows that the tracking error is similar to that of the particle filter.

$$f_i(x(t)) = \frac{-50x_i}{\sqrt{x_1^2 + x_2^2}} \mathcal{I}_{\{\sqrt{x_1^2+x_2^2} \geq 9\}} \mathcal{I}_{\{x_1 x_3 + x_2 x_4 \geq 0\}}$$

$$\begin{aligned} dx_1 &= x_3\ dt + e\ dw_1 \\ dx_2 &= x_4\ dt + e\ dw_2 \\ dx_3 &= f_1(x)\ dt + e\ dw_3 \\ dx_4 &= f_2(x)\ dt + e\ dw_4 \end{aligned} \tag{3.12}$$

$$\begin{aligned} dy_1 &= [x_1^2 + x_2^2]^{1/2} dt + e_1\ dv_1 \\ dy_2 &= \tan^{-1}(x_2/x_1) dt + e_2\ dv_2 \end{aligned} \tag{3.13}$$



(a) $x_1(t)$ (b) $x_2(t)$

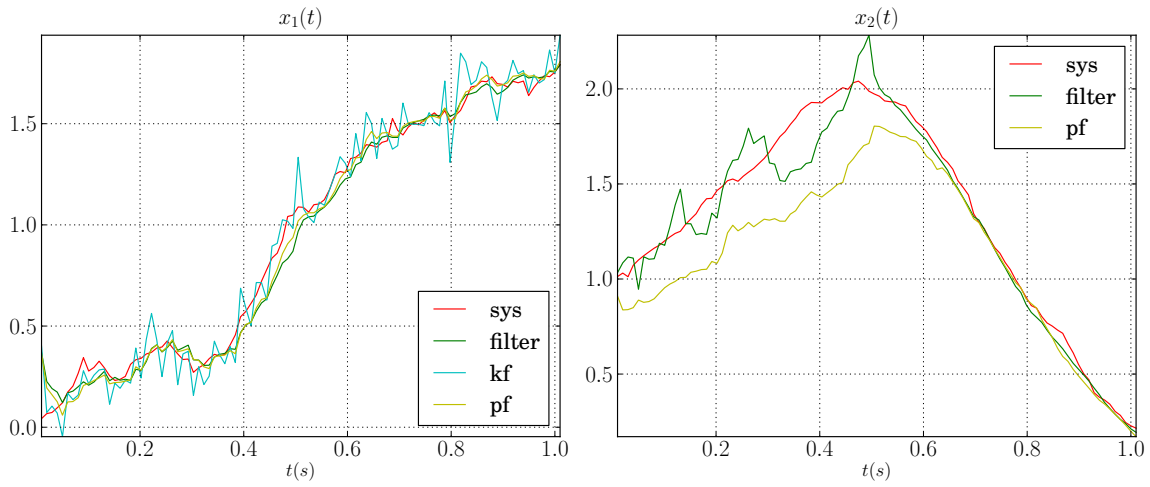
Figure 3-2: Filter estimate for the drifting ship in Equation (3.12) with $e = 0.3$, $e_1 = 0.03$ and $e_2 = 0.03$. The EKF error is very large near $(0, 0)$, when the observation nonlinearity is large. The average estimated state error i.e., $\mathbb{E}[\frac{1}{T}\int_0^T \|x - \hat{x}\| dt]$ is 5.02×10^{-3} for the HMM filter, 5.2×10^{-3} for the particle filter both with 100 particles and 1.36×10^{-2} for the EKF.

3.4.2 Van der Pol oscillator

Next, we consider a noisy Van der Pol oscillator given by Equation (3.14). This system is highly non-linear with a stable limit cycle for $\mu > 0$.

$$\begin{aligned} dx_1 &= x_2\ dt + e_1\ dw_1 \\ dx_2 &= [-x_1 + \mu\ x_1\ (1 - {x_1}^2)]\ dt + e_2\ dw_2 \\ dy &= x_1\ dt + e_3\ dv \end{aligned} \tag{3.14}$$

The last equation is the scalar observation equation and $\mu = 2$. Figure 3-3 shows the performance of the sampling filter on this system. Note that this system is typically hard for the EKF which accumulates linearization error due to varying time scales and, predictably, the EKF estimate of $x_2(t)$ completely diverges. The proposed filter took 0.2 secs to execute while the PF took 0.013 secs for 100 samples with similar average error. This example



(a) (b)

Figure 3-3: $x_1(t)$ and $x_2(t)$ for a Van der Pol oscillator. Mean error of the estimate averaged over 100 runs was 0.1816 for the proposed filter with 100 particles and 0.1849 for the particle filter with 100 particles. The EKF estimate for $x_2(t)$ completely diverges.

shows that the proposed filter performs as well as other filters both in terms of estimate and is also computationally tractable.

3.4.3 Parameter estimation

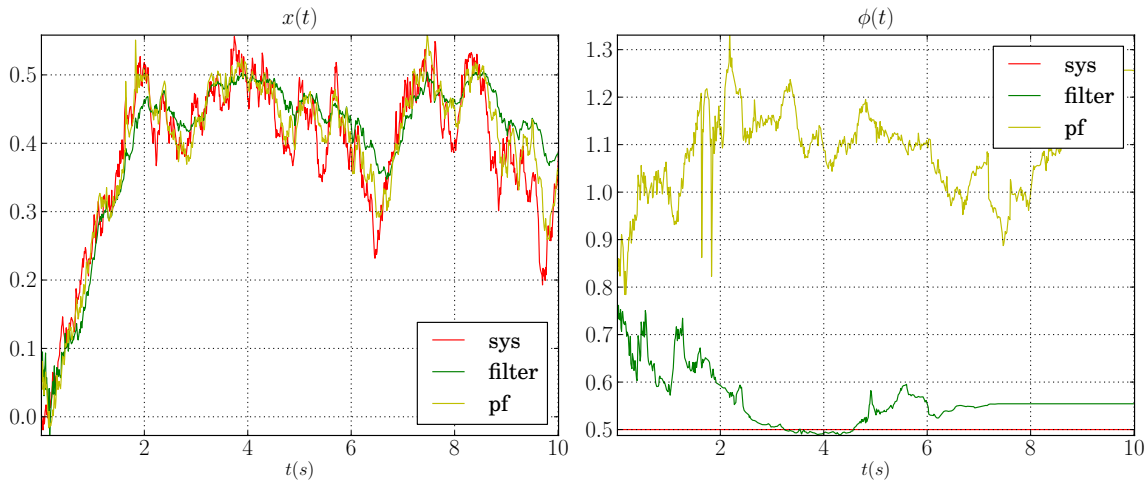
Next we compare these filters on a modified version of a parameter estimation problem from [DT03] as given in Equation (3.15). The parameter we are estimating is $\phi = 0.5$.

$$\begin{aligned} dx &= x\cos(2\pi\phi x)\ dt + \sigma_x\ dw_1 \\ d\phi &= 0\ dt + \sigma_\phi\ dw_2 \\ dy &= x\ dt + \sigma_v\ dv. \end{aligned} \tag{3.15}$$

To begin with, it is only known that $\phi \sim \mathcal{N}(0.8, 1)$. Append the state-space with ϕ and inject small noise dw_2 into its dynamics for estimation. This is a hard problem for a particle filter because the conditional density of ϕ given data is not in the exponential family [DT03] which makes resampling difficult. Figure 3-4 shows an example run with the particle filter using multinomial sampling. The proposed filter consistently ends up with lower estimation error.

3.5 Smoothing

Filters are generally used to provide an estimate of the state of the system in real time. This estimate is then used for various purposes like control or planning. In this section, we are interested in estimating the states of the system off-line based over some given observation interval. The crucial difference between these two situations is that in the current case



(a) (b)

Figure 3-4: $x(t)$ and $\phi(t)$ for the parameter estimation problem with $\sigma_x = 0.1$, $\sigma_v = 0.1$ and $\sigma_\phi = 0.1$. Average state error over 100 Monte-Carlo runs was 1.44 for the proposed filter whereas it was 1.878 for the particle filter with 100 samples for both.

we have access to all the observations collected during the time interval as opposed to only the observations until the current instant like the filtering problem. An example of a situation where we might be interested in something more than filtering is the SLAM (Simultaneous Localization And Mapping) problem. This problem is as follows. A robot is placed in an unknown environment and its job is to build a map of the environment. Since the surroundings are completely unknown it has to simultaneously build a map, estimate its position and evaluate the next position to go to. To do so, it typically uses a filtering algorithm to estimate its position and the map is built based on these filter estimates. This results in a coupling between errors of the map and errors of the state estimate and arguably a much accurate map can be built using an accurate estimate. The procedure is then to use all the observations received by the robot together to calculate the corrected state estimate and then improve the map based on this. This is akin to something like least squares problems where we have access to a lot of data and are tasked with estimating the state at one particular instance.

These two sections look at *maximum a posteriori* estimation problems, in particular trajectory smoothing and trajectory decoding. The smoothing problem can be formally stated as below.

Problem 3.4. *For a system defined by Equations (3.3) and (3.4), given a set of observations* $\mathcal{Y}_t := \{y(s) : s \leq t\}$, *find an estimate* $\{\hat{x}(s) : s \leq t\}$ *such that (i)* $\mathbb{E}[\|x(s) - \hat{x}(s)\|^2]$ *is minimized and (ii) the random variable* $\hat{x}(s)$ *is square integrable and* $\mathcal{Y}_t$*-measurable.*

This is popularly known as the fixed-interval smoothing problem and the solution to this is [AE86],

$$\hat{x}(s) = \mathbb{E}[x(s) \,|\, \mathcal{Y}_t] \quad \forall\ s \leq t.$$

3.5.1 Forward-Backward algorithm

We will first give a quick derivation of the continuous-time smoothing algorithm and see how it relates to filtering estimates. Let the smoothing density be given as $\psi(\mu, t)$. Also, denote the optimal filtering density at time t by $\phi(\mu, t)$. The corresponding distribution for smoothing is written as $P_{x(s)}(\mu \,|\, \mathcal{Y}_t) = \mathbb{P}(x(s) = \mu \,|\, \mathcal{Y}_t)$ and similarly for filtering. Formally,

$$P_{x(s)}(\mu \,|\, \mathcal{Y}_t) = \int_{-\infty}^{\infty} P_{x(s)}(\mu \,|\, x(s+ds) = \nu, \mathcal{Y}_t)\ P_{x(s+ds)}(\nu \,|\, \mathcal{Y}_t)\ d\nu \tag{3.16}$$

The process $y(t)$ is assumed to be separable [**?**] as is done in [**?**]. This means that limits and bounds over dense subsets for example $\bar{y}(t) = \{y(t) :\ t \leq T;\ t \in \mathbb{Q}\}$ are equal to the limits and bounds over the continuous space $\{y(t) :\ t \leq T;\ t \in \mathbb{R}\}$. We can replace $y(t)$ by $\bar{y}(t)$ which is the discrete equivalent of $y(t)$. In particular, this process is equal to $y(t)$ if $t \in \mathbb{Q}$, otherwise zero. This assumption is not restrictive because any process $y(t)$ which is not separable has a corresponding separable process $y'(t)$ such that they are equal with probability one, i.e., $\mathbb{P}(y'(t) = y(t)) = 1$. Since rational numbers are countable, replace $y(s)$ by $\bar{y}(i)$ for some i and rewrite Equation 3.16 as,

$$P_{x(i)}(\mu \,|\, \mathcal{Y}_t) = \int_{-\infty}^{\infty} P_{x(i)}(\mu \,|\, x(i+1) = \nu, \mathcal{Y}_t)\ P_{x(i+1)}(\nu \,|\, \mathcal{Y}_t)\ d\nu.$$

But, $P_{x(i)}(\mu \,|\, x(i+1) = \nu, \mathcal{Y}_t)$ is equal to $P_{x(i)}(\mu \,|\, x(i+1) = \nu, \mathcal{Y}_i)$ using the Markov property. Also,

$$P_{x(i)}(\mu \,|\, x(i+1) = \nu, \mathcal{Y}_i) = \frac{P_{x(i+1)}(\nu \,|\, x(i) = \mu, \mathcal{Y}_i)\ P_{x(i)}(\mu \,|\, \mathcal{Y}_i)}{P_{x(i+1)}(\nu \,|\, \mathcal{Y}_i)}.$$

Thus,

$$P_{x(i)}(\mu \,|\, \mathcal{Y}_t) = \int_{-\infty}^{\infty} \frac{P_{x(i+1)}(\nu \,|\, x(i) = \mu, \mathcal{Y}_i)\ P_{x(i)}(\mu \,|\, \mathcal{Y}_i)}{P_{x(i+1)}(\nu \,|\, \mathcal{Y}_i)}\ P_{x(i+1)}(\nu \,|\, \mathcal{Y}_t)\ d\nu$$

Using the separability assumption, replace i by s and $i+1$ by $s+ds$ to get

$$\psi(\mu, s) = \phi(\mu, s) \int_{-\infty}^{\infty} \frac{P_{x(s+ds)}(\nu \,|\, x(s) = \mu)}{P_{x(s+ds)}(\nu \,|\, \mathcal{Y}_t)}\ \psi(\nu, s+ds)\ d\nu. \tag{3.17}$$

Note that the denominator can also be written as $\displaystyle\int_{-\infty}^{\infty} P_{x(s+ds)}(\nu \,|\, x(s) = \mu)\ \phi(\mu, s)\ d\mu$. Let us discuss some of the terms in the formula for $\psi(\mu, s)$. The first term is the filtering density $\phi(\mu, s)$. The numerator of the integrand is the effect of dynamics of the system while the denominator is the next predicted state estimate given the filtering estimate. This is weighed by the smoothing density at a later time $s+ds$. Translating this formula to the discrete-state, discrete-time case, if we define, $\alpha(\mu, i) = P(x(i) = \mu \,|\, \mathcal{Y}_i)$ and $\beta(\mu, i) = P(\mathcal{Y}_{i+1:T} \,|\, x(i) = \mu)$, the above expression simplifies to

$$\psi(\mu, i) = \frac{\alpha(\mu, i)\ \beta(\mu, i)}{P(\mathcal{Y}_T)}. \tag{3.18}$$

This is the motivation for the classic Forward-Backward algorithm for smoothing which we will use. In addition, the definition of α and β yields itself to an iterative procedure using Bayes' rule given in Equation (3.20).

Smoothing on approximate Markov chains The modified Markov chain, call it M_n^δ, for the time axis discretized by δ that was created for filtering in Section 3.3.1 will be used here for obtaining smoothing estimates. The proposed solution to the fixed-interval smoothing problem uses the forward-backward algorithm [Rab89] as motivated above. In particular, let

$$\begin{aligned} \alpha_t(z) &= \mathbb{P}(\mathcal{Y}_{0:t},\ \xi(t) = z), \\ \beta_t(z) &= \mathbb{P}(\mathcal{Y}_{t:T} \,|\, \xi(t) = z), \\ \mathbb{P}(\xi(t) = z \,|\, \mathcal{Y}_T) &= \eta\ \alpha_t(z)\ \beta_t(z), \end{aligned} \tag{3.19}$$

where η is a normalization factor dependent only upon $\mathcal{Y}_T$.

We are interested in obtaining the smoothing estimate on a discrete Markov chain whereby Theorem 3.3 will be used to prove that it converges to the optimal continuous time smoothing estimate. Given $\mathcal{Y}_t$, partition the interval $[0, T]$ into m sub-intervals of length δ at times t_k, $k \in 1, \ldots, m$. Using the definition of $\alpha_t(.)$ and $\beta_t(.)$ in Equation (3.19), for this discrete-system, the iterative procedures can be given as,

$$\begin{aligned} \alpha_{k+1}(z) &= \mathbb{P}(y_{k+1} \,|\, z) \sum_{z' \in S} \alpha_k(z) \mathbb{P}(z \,|\, z') \\ \beta_k(z) &= \sum_{z' \in S} \beta_{k+1}(z')\ \mathbb{P}(z' \,|\, z)\ \mathbb{P}(y_{k+1} \,|\, z') \end{aligned} \tag{3.20}$$

Notice that $\alpha_t(z)$ is just the usual filtering estimate as calculated in Section 3.3. $\beta_t(z)$ is the backward filtering (since it runs backward in time) with the uniform density at time T as the prior. Figure 3-6 describes the iterative update of $\alpha(\cdot)$ and $\beta(\cdot)$. Run the filtering

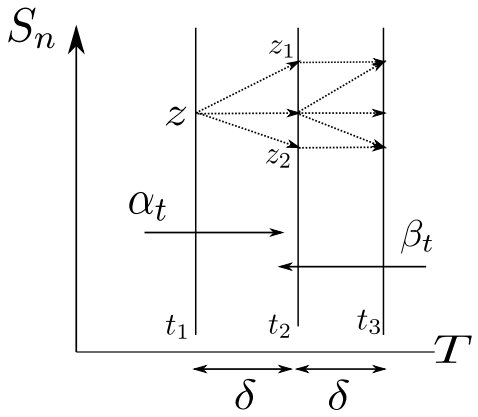


Figure 3-5: Forward-Backward algorithm for trajectory smoothing

algorithm on M_n^δ in the forward direction to get $\alpha_{t_k}(z)$ for all t_ks. A backward filter with

uniform density as the prior then gives $\beta_{t_k}(.)$. Let the smoothing density be

$$\psi^{\delta}(z,t) = \mathbb{P}(\xi(t) = z \,|\, \mathcal{Y}_T).$$

3.5.2 Smoothing on approximate Markov chains

We can now give the algorithm for smoothing. The inputs required are the Markov chain $M_n^{\delta_k}$ constructed for some n using Algorithm 2.3 and a δ_k as defined in Section 3.3. A trajectory of the output $y(t)$ $0 \leq t \leq T$, is partitioned into m intervals thereby giving the set $\{t_k;\ k \leq m\}$ where $0 = t_0 < t_1 < \ldots < t_m = T$. We find the unnormalized smoothed density only at these m instants. Given these two numbers m and n, the batch smoothing algorithm is described as follows. A Markov chain $M_n^{\delta_k}$ where $\delta_k = t_{k+1} - t_k$ shown as the vertical line in Figure 3-6 is such that the transition probabilities are calculated for the time discretization δ_k i.e. to go from time t_k to t_{k+1}. There is one such Markov chain with the same set of states but different transition probabilities calculated using Equations (3.8) and (3.9) for every instant t_k. These Markov chains are used to calculate the solution of the forward filtering problem to give $\alpha_{t_k}(z)$ while a backward filtering problem gives the values of $\beta_{t_k}(z)$ for all $z \in S_n$. The optimal smoothing density is just the product of these two densities after appropriate normalization at every t_k.

The incremental solution to this problem is not as straightforward. We want to increase m and n in such a way that the smoothing density calculated on these discrete Markov chains approximates the optimal smoothing density. The algorithm for this is given in Algorithm 3.1 while some preliminary procedures are discussed below. We refine the Markov chain approximation by the incremental algorithm given in Chapter 2 while simultaneously breaking the interval $[0, T]$ into finer refinements to calculate the smoothing density at those intervals.

- Partition the interval $[0, T]$ such that t_k such that $0 < t_1 < t_2 \ldots < t_m < T$ and $\max \delta_k = \delta$. Create Markov chain approximations $M_n^{\delta_k}$ each with the same n states. The procedure `Forward` takes in the current time instant t_k and Markov chain $M_n^{\delta_k}$ and calculates α_{t_k} from $\alpha_{t_{k-1}}$ using Equation 3.20. Similarly, the procedure `Backward` takes in the current time instant t_k and Markov chain $M_n^{\delta_k}$ and calculates β_{t_k} from $\beta_{t_{k+1}}$ using Equation 3.20. The procedure `Forward` − `Backward` does both of these operations.

- Every iteration adds a new sample z_{n+1} to create a new Markov chain $M_{n+1}^{\delta_k}$. The smoothing density $\psi(z_{n+1}, t_k)$ for all $k \leq m$ has to be recalculated. This probability is dependent upon all the observations that create $\mathcal{Y}_{0:T}$, but we can use a local approximation by assuming that the density is smooth, i.e.,

$$\psi_{t_k}(z) = \frac{1}{|Z_{near}|} \sum_{z' \in Z_{near}(z)} \psi_{t_k}(z')$$

  for all t_k with $k \leq m$. Since $|Z_{near}| = \mathcal{O}(\log n)$, this procedure takes $\mathcal{O}(m \log n)$ time to average and $\mathcal{O}((\log n)^2)$ time to create the new Markov chain $M_{n+1}^{\delta_k}$. Call this averaging procedure `ApproximateDensity`.

- Every iteration of the smoothing algorithm thus adds a new state to the Markov chains. To approximate the smoothing density at all instants, we have to successively refine the time partition also. Every $\mathcal{O}(\log n)$ iterations, uniformly randomly sample a new time instant say t' from the interval $[0, T]$ and create the corresponding Markov chain $M_n^{\delta'}$. To obtain the $\alpha'_t(z)$ and $\beta'_t(z)$, use the iterative procedure given in Equation (3.20). This however affects the forward density of all $t_k > t'$ and the backward density of all instants $t_k < t'$. We thus have to re-propagate these densities forward or backward m times for every new time instant added to the partition. This procedure is an $\mathcal{O}(m\ n \log n)$ operation. Since a new time instant is added only if $c\lfloor \log(n+1) - \log n \rfloor > 1$ we have $m = \mathcal{O}(\log n)$. This results in the asymptotic amortized complexity of the propagation operation being $\mathcal{O}(n \log n)$.

Algorithm 3.1: Incremental Smoothing

$n \leftarrow n_1$;
$m \leftarrow m_1$;
for $k \leq m$ **do**
    Create Markov chain $M_n^{\delta_k}$;
    $\psi_{t_k} = \texttt{Forward} - \texttt{Backward}(M_n^{\delta_k}, \mathcal{Y}_T)$;
while $n < N$ **do**
    for $k \leq m$ **do**
        Create $M_{n+1}^{\delta_k}$ from $M_n^{\delta_k}$;
        $\psi_{t_k}(z_{n+1}) \leftarrow \texttt{ApproximateDensity}(\psi_{t_k})$;
    if $c\ \lfloor \log n - \log(n-1) \rfloor > 1$ **then**
        $t' \leftarrow \texttt{SampleTime}$;
        Create $M_{n+1}^{\delta'}$;
        $\psi_{t_k} \leftarrow \texttt{Forward} - \texttt{Backward}(M_{n+1}^{\delta'}, \mathcal{Y}_T)$;
        for $t_k > t'$ **do**
            $\psi_{t_k} \leftarrow \texttt{Forward}(M_{n+1}^{\delta'}, \mathcal{Y}_T)$;
        for $t_k < t'$ **do**
            $\psi_{t_k} \leftarrow \texttt{Backward}(M_{n+1}^{\delta'}, \mathcal{Y}_T)$;
    $n \leftarrow n + 1$;
return $(M_N^{\delta_k}, \psi_{t_k})$;

Theorem 3.5. *The smoothing density calculated by Algorithm 3.1 converges to the optimal smoothing density given by Equation (3.17) as $n \to \infty$ i.e. for any continuous bounded function $\phi(\cdot)$,*

$$\lim_{n\to\infty} \sup_{t \leq T} \left| \mathbb{E}[\phi(\xi_n(t)) \,|\, \mathcal{Y}_T] - \mathbb{E}[\phi(x(t)) \,|\, \mathcal{Y}_T] \right| = 0.$$

Remark 3.6. The optimal smoothing density is as given in Equation (3.18). Both the forward and backward filters converge to the optimal foward and backward filtering densities

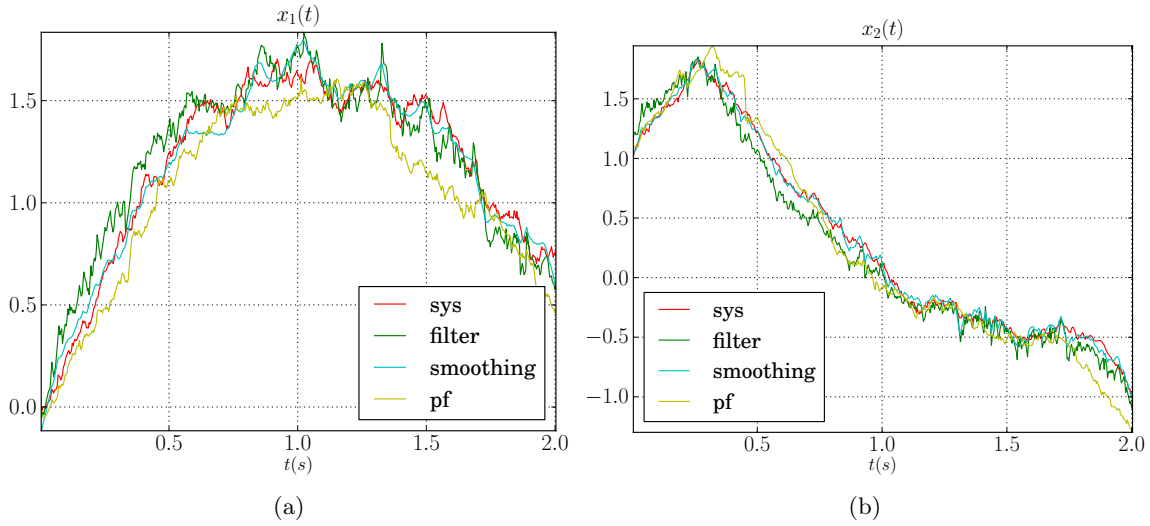


Figure 3-6: $x_1(t)$ and $x_2(t)$ for Vander pol oscillator. Average state error over 100 runs was 0.051 for the filtering algorithm in Section 3.3 (without heuristics), 0.0164 for the smoothing algorithm described in this section using 5000 samples and 0.04233 for the particle filter using 1000 samples.

respectively by Theorem 3.3 and hence the smoothing density given by Equation (3.18) also converges to the optimal smoothing density which is the solution of Problem 3.4.

3.5.3 Examples

Figure 3-6 shows an example run on the Van der Pol oscillator as described in Section 3.4. Note that the jittery nature of the filtered output for the Markov chain filter as well as the particle filter is not seen in the smoothing output which leads to a much improved state estimate.

3.6 Decoding

We can also consider the *maximum a posteriori* (MAP) trajectory estimator also known as decoding. It aims to find the most likely *trajectory* that explains all the observations until a fixed terminal time T. Consider the discrete time case to motivate the continuous time solution. The problem here is to find a trajectory $\hat{x}(t_k)$ such that

$$\hat{x} = \arg \max_{x(t_k) \in \mathbb{R}^d;\ \forall k} \mathbb{P}(x(\cdot) \,|\, \mathcal{Y}_T).$$

Let us briefly go through Viterbi's algorithm which obtains the solution to the above problem using dynamic programming. Given a Markov chain M_n and a discrete observation sequence y_k at time instants t_k, the Viterbi algorithm [FJ73] finds the most likely trajectory, i.e., $\hat{x}(t_k)$ for all k. Let $\gamma_{t_k}(z)$ be the probability of the most likely path that ends at state z and time

t_k generating observations $y(t_i);\ i \le k$

$$\gamma_{t_k}(z) = \max_{\xi(t_i);\ i<k} \mathbb{P}\left(\xi(t_i);\ i < k,\ \xi(t_k) = z,\ y(t_i);\ i \le k\right).$$

This can be written in an iterative form as follows,

$$\gamma_{t_{k+1}}(z) = \max_{z' \in S_n} \gamma_{t_k}(z')\ \mathbb{P}(z \,|\, z')\ \mathbb{P}(y_{t_{k+1}} \,|\, z). \tag{3.21}$$

The Viterbi algorithm iterates over all time steps t_k and all states $z \in S_n$ to find the value of $\gamma_T(z)$. It also maintains the maximizer of the $\gamma_{t_k}(z)$ at each step. At time T, the state with the highest value of $\gamma_T(z)$ is the end-point of the most likely trajectory which can be followed back using the maximizers maintained by the algorithm to yield the complete trajectory.

In the general continuous time case the solution to the problem of finding the most probable trajectory is infinite-dimensional. We will again consider the system in the drift-diffusion form as given in Equation (3.3) and observations as given by Equation (3.4). Formally, the objective then is to find the most probable trajectory as follows.

Problem 3.7. *Given a stochastic differential equation like Equation (3.3) with observations of the form given in Equation (3.4), the decoding problem finds the most probable trajectory such that*

$$\theta(t) = \arg\max_{\theta \in C^1} \mathbb{P}\Big(x(t) : \|x(t) - \theta(t)\| < \epsilon \ \Big|\, \mathcal{Y}_T\Big)$$

Remark 3.8. The maximization in the above problem looks peculiar for the following reason. It is easy to define the most probable state problem as $\hat{x}_s = \arg\max_{z(s)\in\mathbb{R}^d} \mathbb{P}(z(s) \,|\, \mathcal{Y}_T)$ where $\hat{x}_s$ is the most likely state at time s. But this problem is not very meaningful for the case of the space of trajectories because there is no legitimate prior on this space. We however can consider a relaxation of the problem as given by Zeitouni in [ZD87] and take an ϵ-neighborhood around the most probable trajectory. Given this form of the problem, it is worthwhile to check whether the solution to this problem always exists. It turns out that a solution to the above problem exists almost surely with respect to the observation trajectory $\mathcal{Y}_T$ in an appropriate space [ZD88]. An interesting aside that comes from this analysis is that a finite dimensional MAP trajectory estimator exists only in cases when a finite dimensional optimal filter exists. Thus we can find a finite dimensional estimator to calculate the MAP trajectory for situations when the Kalman-Bucy filter and the Benes' filter work. In fact, if the observation equation is linear in the state, i.e., $g(x(t)) = cx(t)$, the solution of the MAP smoothing problem is the same as that of the optimal smoothing algorithm given in Section 3.5.

Note that $\mathbb{P}(\|x-\theta\| < \epsilon \,|\, \mathcal{Y}_T) \to 0$ as $\epsilon \to 0$. So the cost function for a decoded trajectory θ of the MAP estimator problem needs to be relaxed via normalizing as follows,

$$J(\theta) = \lim_{\epsilon \to 0} \frac{\mathbb{P}(x : \|x - \theta\| < \epsilon \,|\, \mathcal{Y}_T)\ K_y}{\mathbb{P}(w : \|w\| < \epsilon)} \tag{3.22}$$

where

$$K_y = \mathbb{E}\left[\exp\left(\int_0^1 g^*(w_s)dy_s + \int_0^T f^*(w_s)dw_s - \frac{1}{2}\int_0^T (|g(w_s)|^2 + |g(w_s)|^2)ds\right) \mid \mathcal{Y}_T\right]$$

with $(\cdot)_s = (\cdot)(s)$. As proved in [ZD88], under some technical conditions, this converges to the cost function

$$J(\theta) = \exp\Big[-\frac{1}{2}\int_0^T \left|\dot{\theta}(s) - f(\theta(s))\right|^2 ds - \frac{1}{2}\int_0^T \operatorname{div} f(\theta(s))\, ds \tag{3.23}$$

$$+ y^*(T)g(\theta(T)) - \int_0^T y^*(s)\nabla g(\theta(s))\, \dot{\theta}(s)\, ds\Big] := A_y(\theta). \tag{3.24}$$

Using variational calculus, it can be shown [ZD87] that the trajectory θ that maximizes the above functional is the solution of the stochastic differential equation

$$\begin{aligned} d\psi_s &= (G^*(\theta_s) - G(\theta_s))\, \psi_s ds + K(\theta_s, y_s)ds + M(\theta_s, y_s)\, dy_s \\ d\theta_s &= \psi_s ds. \end{aligned} \tag{3.25}$$

where if $G_{ij} = \frac{\partial f_j}{\partial x_j}$, $H_{ij} = \frac{\partial g_j}{\partial x_i}$ and $Z_i = \frac{\partial}{\partial x_i}$ trace G, $M_{ij} = H_{ij}$ and $K_i = (Gf)_i + (Hg)_i + 1/2Z_i$ depend only upon the observations y_s and the system defined by Equation (3.3) and (3.4). In other words,

$$\theta = \arg\max_{f \in C^1}\ J(f).$$

Solving for the MAP trajectory then is finding the trajectory that minimizes this cost function. The above equation has a stochastic term dy but the right way to think of a solution θ is that it is a strong-sense unique solution of Equation (3.25) (see Appendix A) because the observation trajectory $y(s)$ is given beforehand and the solution is calculated with respect to the filtration $\mathcal{Y}_s$. Let us only consider a simple case of boundary conditions when the initial state of the system is known perfectly $\theta_0 = x_0$. The boundary conditions can be given as,

$$\begin{aligned} \dot{\theta}_T &= f(\theta_T) \\ \theta_0 &= x_0. \end{aligned}$$

Note that the boundary conditions are however split. This is similar to optimal control theory where the differential equations of states and co-states have split boundary conditions. The problem is then to find a trajectory which is the solution of the above differential equation with these boundary conditions. This is a general two-point boundary value problem and can be solved in a number of ways given access to the observation sequence $y(s);\ 0 \le s \le 1$. We can however use the knowledge that $\theta(s)$ minimizes the above cost function (which can be calculated for any state) to create a Rapidly Expanding Random Graph (RRG) [KF11] with the split boundary conditions. The shortest cost path in this graph is then the solution Equation (3.25). Note that $\theta(t, x)$ is the probability of the most likely path reaching $x \in \mathcal{S}$ at time t while generating observations $y(s);\ s \le t$ along the way [ZD87], it is the continuous time counterpart of $\gamma_k(z)$. Thus, solving Equation (3.25)

for the MAP trajectory is equivalent to solving the Viterbi algorithm on a discrete Markov chain. It is this equivalence which will be useful to solve the decoding problem using Markov chain approximations.

3.6.1 Decoding on approximate Markov chains

In this section, we will give an algorithm which use the Viterbi algorithm on discrete Markov chains approximations to recover the solution to the continuous time MAP problem described in Problem 3.7. We will follow a similar procedure as that of smoothing. Given continuous-time observations of $y([0,T])$, partition the interval $[0,T]$ into m subintervals and create a Markov chain approximation $M_n^{\delta_k}$ at every time instant. The Viterbi algorithm can be then be run to find the most likely trajectory, i.e., the trajectory with the least cost as defined by Equation (3.22) in a batch fashion. In addition to this, we can refine the approximate Markov chains and also create successive refinements of the time axis to get an incremental solution to the decoding problem.

The decoding problem is however simpler than smoothing in the following sense. Since we are only interested in the most likely trajectory, we need not maintain the whole smoothing density at every time instant. The remainder of the section gives an incremental solution to the decoding problem by constructing a Markov chain directly in the $\mathcal{S} \times [0,T]$ space. This space can be discretized in a different way than the smoothing in Algorithm 3.1. Append the state space with time as $z = (t,x)$ along with the trivial dynamics for the time part, i.e., $\dot{t} = 1$ and create a locally consistent Markov chain using Algorithm 2.3 to get M_n. Note that $\Delta t(z)$ as defined in Section 2.4.1 is the expected increment in time of any Markov trajectory $\xi(t)$ after a transition at state z. i.e.,

$$\mathbb{E}[\Delta\xi^0(t_{n+1}) \,|\, \xi(t_n) = z] = \Delta t(z),$$

where $\xi^0(.)$ denotes the time component of the state. This condition along with the other three conditions given in Equations (2.5)-(2.7) then ensure a locally consistent chain in the new state space. Figure 3-7 shows the Markov chain constructed in this appended state-space. Arguably, this construction could have been used in Algorithm 3.1 also. However, the smoothing problem calculates the whole conditional density of the state at every time instant $s \leq t$. With probability 1, no two samples in S_n as constructed here even have the same time components whereas we need the conditional density as time t_k to propagate it to time t_{k+1} using Equation 3.6. The decoding problem only finds the most probable trajectory and does not need to maintain the whole conditional distribution.

3.6.2 Algorithm

The above discussion and the fact that the solution to the decoding problem is the solution to some optimal control problem will help us formulate an algorithm for the MAP trajectory. This relies on a direct construction of a Rapidly Exploring Random Graph (RRG) with appropriate changes to its elementary routines as discussed below. Roughly, an appended state-space $[0,T] \times \mathcal{S}$ as shown in Figure 3-7 is sampled to create a Markov chain incrementally after which the Viterbi algorithm can be run on it to obtain the solution. The preliminary procedures described in Section 2.4.1 almost remain the same with a few

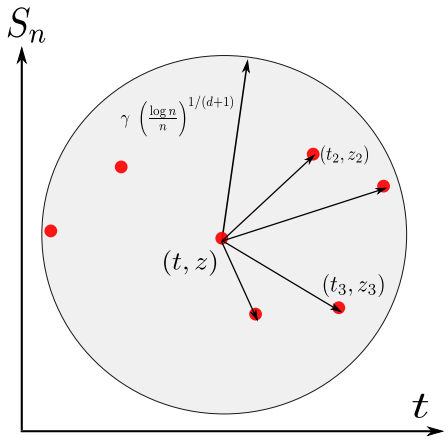


Figure 3-7: Decoding Markov chain

key differences are discussed here. `Sample` procedure samples a point $z = (t, x)$ uniformly randomly from the appended state space $[0, T] \times \mathcal{S}$. Let $t(z)$ denote the time component of the state z of the Markov chain. `Near` returns a set Z_{near} of all states $z_k \in S_n$ such that

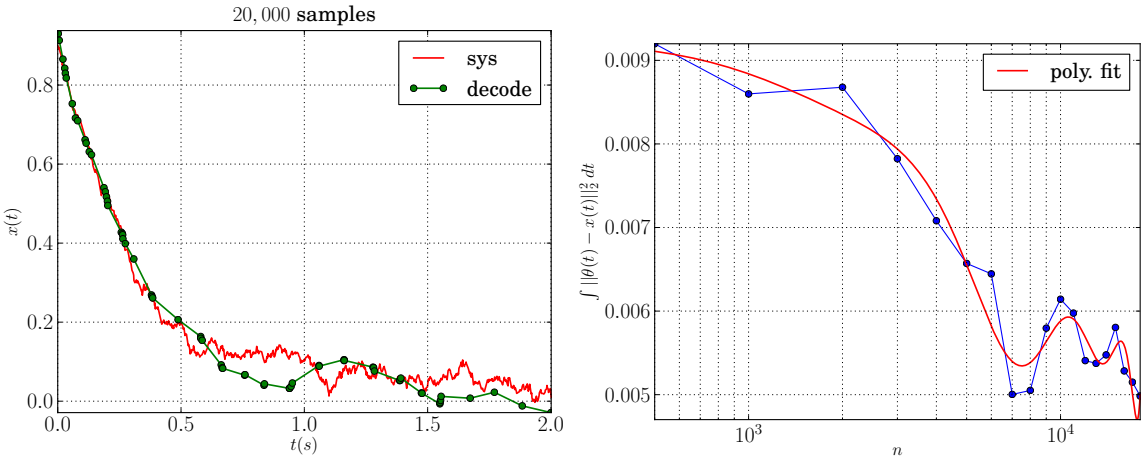
$$\|z_k - z\|_2 \leq \gamma \left(\frac{\log n}{n}\right)^{1/(d+1)}$$

where γ is the same as described in Theorem 2.4. The holding time is trivial in the case of decoding and is the *expected time* that the chain takes to go from state z to the next. The only difference in this construction is that instead of an explicit formula for the holding time, given states in $z_k \in Z_{near}$ we calculate it as

$$\Delta t(z) = \mathbb{E}[\Delta \xi_k^0(z)]$$

where $\Delta \xi^0$ denotes the increment of the time component i.e. $\Delta \xi_k^0(z) = t_{z_k} - t_z$. Transition probabilities are calculated similarly to satisfy the local consistency conditions which can be done using the linear program in Section 2.4.1. If we wish to use the Gaussian approximation which requires a full rank $F(x)$, we can consider a "stochastic" time dynamics given by $\dot{t} = 1 + \sigma_t \tilde{w}$ where $\tilde{w}$ is zero mean standard white Gaussian noise with σ_t being very small. The procedure `UpdateGamma`(z) solves Equation (3.21) to obtain $\gamma(z)$ using states $z_k \in Z_{near}$ with $t(z_k) < t(z)$. Refer to [KF11] for a detailed description of RRG. Algorithm 3.2 uses modified preliminary procedures as described above to build a Markov chain M_n on the appended state-space using Algorithm 2.3 and run the Viterbi algorithm on it to calculate $\gamma_k(z)$ for all $z \in S_n$.

Remark 3.9. The Viterbi algorithm iterates backwards from the final time to the starting time to get the most likely trajectory. In order to converge to the optimal decoded output, we need to ensure that the set $\partial\mathcal{S} = \{T\} \times \mathcal{S}$ is densely sampled while constructing M_n. Just sampling $[0, T) \times \mathcal{S}$ is not enough because the decoded trajectory is constructed explicitly by starting at the final time. A trivial counterexample is a 1-dimensional process $x(\cdot)$ with $x(T) > \epsilon$. If there is no sample in the set $\{T\} \times [0, \epsilon]$, the decoded trajectory will be such



(a) Example decoded trajectory

(b) Error convergence

Figure 3-8: Decoded trajectories with $\sigma = 0.1$ and $\gamma = 0.1$. Total error between the actual and estimated trajectory, calculated as $\int_0^T \|x(t) - \hat{x}(t)\|_2^2 \, dt$, is 4.9×10^{-3} with 20,000 samples in Figure (a). The convergence of decoding error averaged over 100 trajectories is shown on a log-log plot in Figure (b). The red curve shows a 10^{th} order polynomial fit for the error data.

that $|\theta(T) - x(T)| > \epsilon$. In other words, the set $\partial\mathcal{S}$ is a boundary set of the state-space for this problem. The `Sample` procedure is thus modified to sample two points one inside the set $\mathcal{S}$ and one on the boundary $\partial\mathcal{S}$.

Having constructed the Markov chain M_n, the Viterbi algorithm gives the most likely trajectory in $\mathcal{O}(n \log n)$ time (since every state z is connected to $\mathcal{O}(\log n)$ states in our construction of the approximate Markov chain.

Algorithm 3.2: Decoding on Markov chain

```
1 Create Markov chain M_n;
2 S'_n ← Sort z ∈ S_n according to t(z);
3 for z ∈ S'_n do
4     γ(z) ← UpdateGamma(z);
5 z_end ← z ∈ S_n ∩ ∂S with largest γ(z);
6 return (M_N, γ(·), z_end);
```

Example Let us consider a simple example of a 2-dimensional linear system and calculate the maximum a posteriori trajectory for it. The dynamics is given by the equation $dx_1 = -3\, x_1\, dt + \sigma\, dw_1$ while observations are obtained through a process $y(t) = [y_1(t), y_2(t)]^T$ with $dy_1 = x_1\, dt + \gamma\, dv_1$. The corresponding equations for x_2 and y_2 are similar. The system is propagated for $T = 2$ secs to generate the process $y(t)$ which is then used in Algorithm 3.2 to run the Viterbi algorithm. Figure 3-8 shows an example decoded trajectory and the decoding error as a function of the number of states n in the approximate Markov chain.

Chapter 4

Control of Partially Observable Processes

After considering state estimation algorithms in the previous chapter, we move to a related problem, that of decision making in dynamic environments. It is a problem that we come across in many different branches like control systems, portfolio management, supply chain management in operations research to name a few. These problems typically involve a situation where we have to choose a feasible set of actions to satisfy numerous possible future outcomes and do so while preserving some notion of optimality. These are primarily modeled as *Markov decision processes* (MDPs) wherein the system is modeled as a Markov chain that transitions based upon the action chosen. This action is chosen in such a way that it minimizes some form of a state and action dependent cost. In certain situations, instead of fully observing state variables we might instead be able to observe other quantities which give partial information about the state of the underlying system. These observations may either be exact or they may be corrupted by noise. This gives rise to *partially observed Markov decision processes* (POMDPs). POMDPs are thus a very general and principled approach to model problems of decision making under uncertainty.

The solution to real-life POMDP problems is marred by computational intractability. In fact until a few years ago, even problems with more than 10 states were beyond the reach of computers. This is primarily because of two aspects. As the number of states, actions and observations gets larger, the number of possible future states of the system grows exponentially. This space, as we will later parametrize it, is known as the belief space. The task is then to search for optimal policies that take into account all different future states. The other equally hard aspect of the problem is that as the number of time steps considered in the future grows larger, this search tree grows exponentially. The problem is thus doubly exponential in the time horizon and the size of the system. Modern approaches to this make extensive use of heuristics to search this large space efficiently. This chapter proposes the idea that instead of solving a larger POMDP from scratch, we can successively approximate the solution by solving smaller POMDPs which are much easier to solve due to their size. There are two major things to be considered in such a program. Firstly, given a larger problem we need to be able to construct smaller approximations of it. In particular, this chapter looks at general continuous time partially observable problems and

creates sequences of "smaller" problems which converge to it in the limit. Secondly, the solution obtained for the larger problem should not only be optimal but also be calculated in an *incremental* fashion i.e. the control policy of smaller POMDPs should approximate the solution of the continuous time POMDP in an appropriate sense. This program thus enables us to solve general partially observable problems if we can solve the smaller discrete POMDPs efficiently.

The structure of this chapter is as follows. Section 4.1 provides some background on discrete time POMDPs. With a large history of almost 40 years, there have been numerous different approaches to obtain a computationally tractable solution. In particular, Section 4.2 summarizes four major areas that these methods fall into. This section also discusses why the POMDP problem is so computationally challenging by analyzing different existing algorithms and complexity results. This motivates a new solution to the POMDP problem which is the primary content of this chapter. Section 4.4 concretely defines the continuous time stochastic control problem with noisy observations. This formulation is general enough to encompass a number of other equivalent problems. Section 4.5 discusses the the construction and details of the proposed solution. In particular, it defines the construction of sequences of discrete POMDPs which approximate the continuous time problem that we are looking at along with a way to find the solution incrementally. Section 4.6 is the core of the technical part of the chapter and proves a number of results that show that the approximate constructions converge to the original problem in terms of the cost function and the control policies on these discrete POMDPs. Simulation experiments in Section 4.7 are validate the performance of the proposed algorithms on a number of classic problems.

4.1 Preliminaries

Markov decision processes

An MDP is a tuple consisting of $M = (S, A, P, s_0)$ where S is a finite set of states such that the intial state is $s_0 \in S$ and A is a finite set of actions. The process is Markov in the sense that taking an action $a \in A$ from a state $s \in S$ leads to a new state $s' \in S$ with a probability $P(s' \mid s, a)$. If the process is not time homogeneous, we write this *transition probability* as $\mathbb{P}(s' \mid s, a, t)$. Let us consider the *finite horizon* problem first. Given a cost function of the form

$$J = \mathbb{E}\left[\sum_{k=1}^{T} l(s, a, k) + L(s(T))\right]$$

$l(s, a, t)$ is the cost incurred after taking an action a at state s. while $L(s(T))$ is a terminal state cost. A policy $\pi : S \to A$ is a mapping from the set of states S to the set of actions A. An optimal policy π^* is a mapping such that

$$J_{\pi^*} = \inf_{\pi \in \Pi} J_\pi$$

where $J_\pi = \mathbb{E}\left[\sum_{k=1}^{T} l(s, \pi(s), t) + L(s(T))\right]$ and Π is the set of all feasible policies which is the set of all Markov policies. Discounted cost, as shown below, ensures that the cost function remains finite even for infinite horizon control problems by considering a cost

function like

$$J = \mathbb{E}\left[\sum_{k=1}^{T} \gamma^k \; l(s, a, k) + L(s(T))\right]$$

where $\gamma \in (0, 1]$ is a discount factor.

Partially observable Markov decision processes

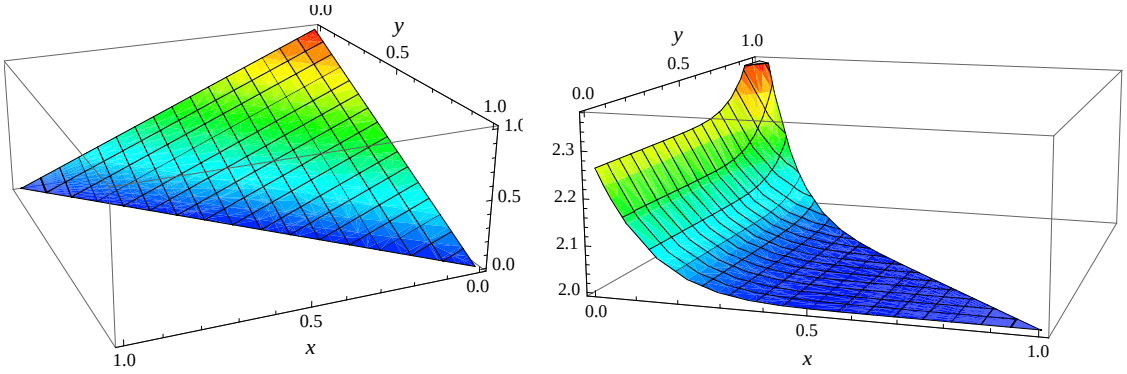
A POMDP is a tuple $M = (S, A, O, P, Q, b_0)$ such that S is a finite set of states, A is a finite set of actions and O is a finite set of possible observations received at those states. The variable b_0 is called the initial belief which is a probability mass function over the set S denoting the probability $b_0(s) = \mathbb{P}(s(0) = s)$. P denotes the transition probabilities and is same as that of the Markov decision process. M is a probability mass function which gives the probability $Q(o \,|\, s) = \mathbb{P}(o \,|\, s)$ for all $o \in O$. Similar to the MDP case, we will consider cost functions of the form

$$J = \mathbb{E}\left[\sum_{k=1}^{T} l(s, a, k) + L(s(T))\right].$$

A POMDP has primarily two sources of uncertainty, one is the stochastic nature of the inherent system while the other is the uncertainty arising from noisy observations of its states. It turns out that these two sources together lead to an explosion of possible scenarios which makes the solution much harder than either of its component problems viz. MDPs or Hidden Markov Models (HMMs). A simple example is the following problem.

Problem 4.1. *Consider a system with 3 states s_1, s_2, s_3. Every state s_i has 2 transition matrices, one for each action a_{ij} $(j \neq i)$ of probabilities $\mathbb{P}(s_j \,|\, s_i, a_{ij})$. The observation consists of a matrix of probabilities of the form $\mathbb{P}(s_j \,|\, s_i)$ if s_i is the true state. The system gets a (state dependent) reward after it takes an action based upon one observation.*

A distribution on these 3 states is a normalized tuple of probabilities $\pi = (p_1, p_2, p_3)$ also called *belief* shown as a plane in a 3-dimensional space in Figure 4-1a. Every action and observation sequence results in a different tuple in this space. The objective then is to find some sequence of actions such that each action $a \in \{a_{ij};\ i, j \leq 3\}$ starting from some initial tuple π_0 to get the maximum reward. For a finite horizon problem, the set of beliefs is finite, although exponentially large in both the number of states, actions and observations and the time horizon. After searching on a tree with 6 actions and 3 observations i.e. 6×3 possible beliefs at every step, we get a value function given in Figure 4-1b parametrized by p_1, p_2 due to normalization. This shows the explosion of the computation required for 1 step, 6 actions and 3 possible observations. As the number of steps, actions and observations gets larger, the computation required grows exponentially and the problem very quickly becomes intractable. In fact after 30 years of research since the first results in the area, problems with more than tens of states were still intractable.



(a) Belief space (b) Value function

Figure 4-1: Example 3-dimensional belief space with its corresponding value function.

4.2 Previous approaches

This section discusses some solutions to the POMDP problem. It focusses on the widely different areas that have generated different approaches to this problem.

Stochastic control. The results in this area can be broken down into two subparts. First is the existence and nature of controls for the general non-linear continuous time optimal control problem. The second section focusses on finite dimensional filtering problems which enable a solution to the partially observed stochastic control problem using an equivalent separation theorem.

Consider a stochastic system in the drift-diffusion form given by

$$dx(t) = f(x,u)\ dt + F(x,u)\ dw(t). \tag{4.1}$$

The unique weak-sense solution to the above equation written as

$$x(t) = x(0) + \int_0^t f(x(s),u(s))\ ds + \int_0^t F(x(s),u(s))\ dw(s)$$

where $x(0)$ is a random variable drawn from some initial distribution $x(0) \sim b(0)$. Formally under assumptions of continuity and smoothness, the Fokker-Planck equation describes the evolution of this initial distribution under the dynamics given by Equation (4.1). If $b(x,t)$ be the density of $x(t)$ at a time t we have

$$\frac{\partial b}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}\ (f_i b) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}\ (FF_{ij}^T\ b). \tag{4.2}$$

This can also be written in short as

$$b_t = \mathcal{L}_f\ b$$

where $\mathcal{L}_f$ is the forward Fokker-Plank operator defined as

$$\mathcal{L}_f() = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \; f_i(\cdot) + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \; F F_{ij}^T(\cdot).$$

If in addition to the dynamics we have stochastic observations of the form

$$dy(t) = g(x) \; dt + G(x) \; dv(t) \tag{4.3}$$

with $v(t)$ being uncorrelated to $w(t)$, the evolution of the unnormalized conditional distribution $\mathbb{P}(\xi(t) = x \,|\, b(0), \mathcal{Y}_t)$ is given by the Zakai equation. The density $q(x,t)$ evolves as follows.

$$dq = \mathcal{L}_f(q) \; dt + q \; g^T dy \tag{4.4}$$

$\mathcal{L}_f$ is the Fokker-Planck operator as defined above. As seen from this formula, the Zakai equation is a stochastic partial differential equation dependent upon the random process $y(t)$. The solution of this equation entails the solution of the optimal filtering problem.

The aim of the stochastic control is then to influence the distribution given by Equations (4.2) and (4.4) from an initial distribution to some terminal distribution so as to minimize some cost criterion of the form

$$J = \mathbb{E}\left[\int_0^T l(b,u,t) \; dt + L(b(T))\right]$$

Complications arise on this route because the resulting dynamics is an infinite dimensional system and the controlled object i.e. the state $x(t)$ no longer enters the control problem. However, the theory of optimal control of distributed parameter systems is sufficiently developed [ABC+08], this problem is a special case of that general theory. A number of results in this area are known. For example the Hamilton-Jacobi-Bellman equation for this problem is given by Mortensen in [Mor66] or in a different form by Kushner in [Kus65]. The controller in such situations depends upon the whole history of the processes $x(t), y(t)$ or equivalently, the control is a Markov function of the conditional state distribution at time t. Explicit solutions to the HJB equation are difficult to obtain because it is an HJB equation in function space. However, special cases like linear dependence of the drift $(f(x,u))$ and observation $(h(x))$ on the state x give back the Linear-Quadratic-Gaussian (LQG) problem.

Speaking of the LQG problem, with linear dynamics and observations, we have access to a fundamental result called the *separation principle* which allows us to reformulate the partially observed problem as a completely observed one with the state as the conditional mean given by the Kalman filter. It can be shown that optimal LQR controller placed in a loop with a Kalman filter is optimal. It is tempting to look for a separation result for non-linear systems but it eludes effort since optimal finite dimensional filters for the general non-linear problem are unavailable. A major result in this area is however given by Mortensen [Mor66] where he shows that the separation principle is valid with the conditional distribution of the state based on observation history taken as a sufficient statistic on which control actions are based.

Theorem 4.2 ([Mor66]). *Conditional distribution of the state given observation history*

is a sufficient statistic for POMDP

Any success towards a separation theorem for non-linear systems thus depends upon solving for the optimal conditional density. Charalambous in [CE97] gives a number of cases where such a solution is possible viz. finite dimensional estimation algebra [BC80], drift and observations being in gradient form(Benes filter [Ben81]), rational or exponential non-linearities. We will explore these methods further in following sections.

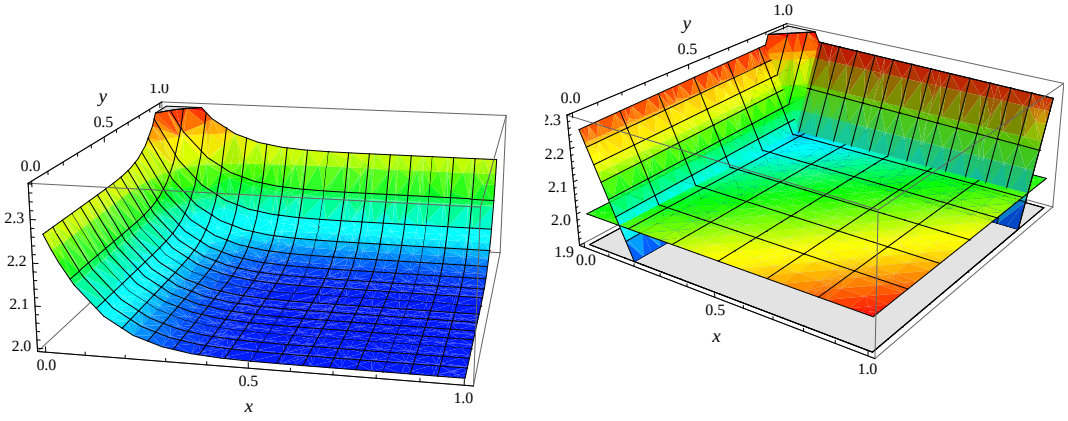
Value function approximation. Arguably, the most important result that makes POMDPs practically tractable is due to Smallwood and Sondik [SS73] which shows that optimal cost function is a convex and piecewise-linear function of the current state of the underlying Markov process. Let us quickly discuss this result. The Bellman equation for the optimal cost in the discounted cost problem starting at a belief b, $J(b)$ can be written as

$$J(b) = \min_{a \in A} \left(l(x,a) + \gamma \sum_{o \in O} \mathbb{P}(o \,|\, b, a)\ J(b') \right) \tag{4.5}$$

where $l(\cdot,\cdot)$ is the running cost, γ is a constant discount factor and $b'(s) = \mathbb{P}(s \,|\, b, a, o)$ is the new belief state after taking an action a at state b_n. The optimal cost function can however be expressed as

$$J(b) = \min_{\alpha \in \mathcal{A}} (b \,.\, \alpha)$$

where $\mathcal{A}$ is a finite set of $\alpha-$vectors which are supported on the set of states S. Each of the $\alpha-$ vector is associated with an action $a(\alpha)$ which is precisely the minimizing control of the Bellman update. Thus these α-vectors completely represent the value function as well as the policy in a POMDP problem. Instead of explicitly calculating and updating the value function at every belief, fast methods just maintain a set of α-vectors and update them.



(a) Value function (b) α-vectors

Figure 4-2: Approximation of the value function using α-vectors which are tangent planes to the value function at the sampled beliefs. The value function is mirrored about the line $x + y = 1$ only for visualization purposes.

Owing to this result, a significantly large portion of literature is thus dedicated to efficient computations and representations of the value function like Sondik's *one-pass algorithm* which uses dynamic programming and the above mentioned structure of the value function to approximate the value function [SS73]. Dynamic programming has been used in a number of other algorithms such as [Mon82, LCK96]. It is evident that some form of relaxation is necessary to make the problem tractable. Exploiting the underlying structure is in fact crucial for practical solutions to this problem. Structured representations of the policy have been explored in [BP96, ZL96]. Pruning methods for dynamic programming updates enable much faster algorithms in practice as is shown in [CLZ97].

Motion planning. Another major area of modern results is inspired from motion planning methods. Given the vast success of randomized sampling based methods for path-planning problems even in high-dimensional configuration spaces like Probabilistic Roadmaps [KL98] and Rapidly Exploring random trees [KJL00, LaV98], it is tempting to apply them to search the large belief spaces of the POMDP problem. Though these methods are primarily for deterministic problems, efforts have been made to incorporate stochastic actions by using probabilistic roadmaps. Incorporating observations has generally not been feasible. However, if we can identify a certain structure to the belief space such as Gaussian with factored covariance [PR09] or linearity of the underlying system [BR11, PPK+12] we can create tractable algorithms easily. The crucial observation in such cases is that the evolution equations for the mean of the conditional distribution are controllable and in cases where we can efficiently keep track of the variance of the distribution after different actions, the problem of an exponentially growing action-observation tree is tractable. The strong point of this approach is that it leverages sampling strategies of motion planning algorithms which have been empirically very successful to get solutions quickly. In addition to this, the algorithms are applicable to continuous-time, continuous-state systems and stand out as compared to the vast discrete-time, discrete-state literature. The crucial assumption necessary for this approach, which is that the belief space has a particular structure (e.g., Gaussian) is however vastly untrue and even simple real world examples like a robot traveling inside a room significantly defy this. The algorithms proposed in this Chapter

Approximation methods. Evidently, the scalability of all solutions to POMDP value iteration problems is affected by two main reasons.

- *Curse of dimensionality :* For a problem with n discrete states, the belief space consists of a $(n-1)$ dimensional simplex. Therefore, simple approaches like discretizing the belief space using grids scales exponentially in the number of states.
- *Curse of history :* Value iteration for solving POMDP is roughly like breadth-first search. Starting from a given initial belief $b(0)$, it explores all the reachable beliefs, i.e., beliefs obtained from all possible action-observation sequences by simulating the POMDP forward in time. The number of nodes in this tree scales exponentially with the number of forward time steps. Pruning strategies for searching this large tree improve performance only by constant factors.

The two problems are in fact related, the larger the number of states the higher is the number of possible histories that the algorithm has to maintain. It has been found however

that the *curse of history* is a stronger predictor of the running time of value iteration algorithms [**?**]. Approaches that effectively solve this aspect of the problem thus stand a strong chance to perform well in real-world scenarios.

Modern point-based value iteration methods leverage this observation to sample highly likely beliefs and iterate over them instead of treating every belief equally. Coupled with the observation that the value function is piecewise linear and convex, maintaining a set of α-vectors for these sampled beliefs ensures that even the unsampled beliefs are appropriately (although not precisely) represented in the value function. PBVI is one of the first methods which exploits this observation [PGT03]. To make this more precise,

Theorem 4.3 (PBVI error bound [PGT03]). *For a sampled belief set B and any horizon T, the error produced by the PBVI algorithm is*

$$\|J_n^B - J_n^*\|_\infty \leq c\ \epsilon_B$$

where $\epsilon_B = \max_{b' \in \Delta}\ \min_{b \in B} \|b - b'\|_1$ is the maximum distance of any valid belief b' in the belief simplex Δ from the sampled set B.

The denser the sampled set B is, the better the error bound is on the value function obtained. This result is important in the sense that it provides an easy way to explore the belief tree starting from a small set of beliefs B and still guarantees how far the value function obtained from such a process will be. It follows very easily from the piecewise linear nature of the value function. It is seen that PBVI performs extremely well in practice solving problems as large as *Tag* which has 870 states, 5 actions and 30 observations. Why does such a simple sampling strategy perform so well? The primary difference between PBVI and naive belief space sampling is that it explores only the *reachable* set of beliefs. It follows that this reachable set should be significantly smaller than the set of all the beliefs. Indeed the following result from [HLR07] shows that a good policy can be found efficiently if the reachable belief space is small in some sense.

Theorem 4.4. *For any $b_0 \in \mathcal{B}$, if $C(\delta)$ be the number of δ-sized balls needed to cover $\mathcal{R}(b_0)$, if $|l(x,u,t)| \leq l_{\max}$ for all x, u, t for any constant $\epsilon > 0$ an approximation $J(b_0)$ such that $|J(b_0) - J^*(b_0)| < \epsilon$ can be computed in time*

$$\mathcal{O}\left(C\left(\frac{(1-\gamma)^2\ \epsilon}{4\gamma\ l_{\max}}\right)^2 \log_\gamma \frac{(1-\gamma)\ \epsilon}{2\ l_{\max}}\right)$$

Successive Approximation of Reachable spaces under Optimal Policies : SARSOP The success of the point-based POMDP algorithms was attributed to the fact the reachable belief space starting from any initial belief b_0, $\mathcal{R}(b_0)$ is significantly smaller than the entire belief space $\mathcal{B}$. This allowed point-based algorithms to dense sample the smaller reachable set and give sufficiently good solutions to even very large problems. An extension of this idea further is made by Kurniawati, Hsu et. al. in [HLR07]. Sampling only beliefs which are optimally reachable from the starting belief b_0, denote this sampled set by $\mathcal{R}^*(b_0)$, should reduce the size of the sampling set even further. This is however an epistemological issue since knowledge of the optimally reachable sets is equivalent to the solution of the

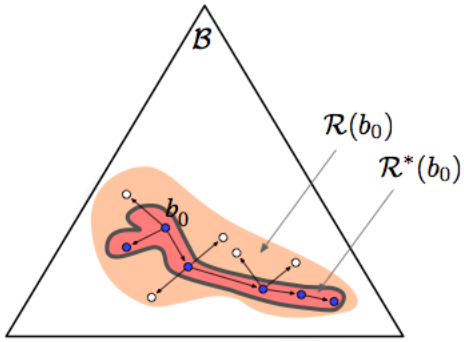


Figure 4-3: Belief space $\mathcal{B}$, reachable belief space $\mathcal{R}(b_0)$ and optimally reachable belief space $\mathcal{R}^*(b_0)$ [HLR07].

complete POMDP problem. Heuristic techniques can be used to approximate this optimally reachable set which results in the search algorithm avoiding large areas of the belief space that are unlikely to be optimal. Doing so results in significant pruning of the set of α-vectors also thus bringing more computational efficiency.

Algorithm 4.1: SARSOP

```
1 Initialize 𝒜 of α-vectors representing the lower
  bound on J*;
2 Initialize the upper bound on J*;
3 Start the belief tree rooted T_R at b_0;
4 while a do
5     Sample(T_R, 𝒜);
6     Choose a subset B ⊂ T_R ;
7     for b ∈ B do
8         Backup(T_R, 𝒜, b);
9     Prune(T_R, 𝒜);
```

For partially observed problems, the upper bound of the cost function is obtained easily by assuming complete information, i.e., by performing value iteration on the MDP. The lower bound can in fact be chosen to be the cost obtained by any simple policy such as the *fixed action policy*. These bounds are used to heuristically bias the sampling process towards beliefs that are optimally reachable and result in the best bound for the cost at the root belief, $J(b_0)$. As seen in Algorithm 4.1, SARSOP iterates on mainly three routines.

- **Sample :** Given a node in the belief tree T_R, a new node is obtained by choosing an action $a \in A$ and an observation $o \in O$. It is crucial to do this in an efficient so as not to sample the whole reachable set $\mathcal{R}(b_0)$. SARSOP traverses down the tree in a depth first manner and chooses action-observation sequences which create the largest change in the upper and lower bound of the cost function at b_0.
- **Backup :** This is the Bellman update as seen in Equation (4.5). However, instead of the value function, SARSOP propagates the gradient of the value function from

the children in the belief tree to their parents. Doing so not only updates the value function at the belief b but also updates the α-vectors over all the belief space.

- **Prune :** The **Backup** method results in an exponential increase in the number of α-vectors per operation. A lot of resulting vectors are such that they are dominated by other α-vectors over the entire belief space. It is thus prudent to prune these vectors because they are sub-optimal anyway. SARSOP again uses the upper and lower bounds of the value function at a belief $b \in T_R$ to prune away dominated α-vectors and all the children beliefs if they are sub-optimal.

This strategy of approximating and sampling the optimally reachable belief space enables very large POMDP problems with 1000s of states being tractable as shown in [HLR07].

4.3 Computational complexity of POMDPs

The seminal paper [PT87] argues that solving the POMDP problem is inherently hard. We will limit ourselves to a short exposition of their result. Let us give a very quick introduction to the theory of computational complexity to motivate the next result [PT87]. Let P denote the class of all problems that can be solved in a polynomial-time on a Turing machine. NP on the other hand is the class of problems that can be solved non-deterministically in polynomial time, i.e., a non-deterministic Turing machine essentially branches out at various stages of the decision problem and pursues every branch in parallel. It thus performs an exponentially large number of computations. Computer algorithms can also be compared with regards to the space required to solve them. The space consumed by an algorithm is the number of memory cells (disregarding the size of input), this could very well be logarithmic in the size of the input. PSPACE is the class of problems which need at most polynomial space. P $\subset$ PSPACE is easily seen because an algorithm running in polynomial time can at most consume polynomial amount of space. Now consider an arbitrary problem Y $\in$ NP. A solution to this can be converted to the solution of the 3-SAT problem in polynomial time. Since Y $\in$ NP, it can be solved in polynomial time by an algorithm which queries the solution of 3-SAT at most polynomial times. It is known that 3-SAT $\in$ PSPACE. Thus P and NP are both subsets of PSPACE. The notion of a problem being C-complete is defined as *A is C complete* if (a) A $\in$ C and (b) any member B of C can be reduced to A via a polynomial time algorithm.

Theorem 4.5 ([PT87]). *Finding optimal policy for a POMDP is PSPACE-hard.*

Remark 4.6. A simple example will illustrate the point of the above theorem. The partially observed problem can be converted to a fully observed problem by looking at a state b which is the conditional probability of the state being $b(s)$ for all $s \in S$. We have effectively made the state-space of the problem infinitely large by doing so [Ber95]. In spite of this it is known that the cost function has an easy minimization of the form

$$J = \inf_{y \in Y} \mathbb{E}\left[\sum_{k=1}^{T} l(b_k, u_k) + L(b(T))\right],$$

where Y is the set of all action-observation sequences of length T and b_k is the belief obtained starting from b_0 after k actions and k observations. It is thus a minimization over all possible action-observation sequences given a starting distribution of states. This set is in general exponentially large in the time horizon. This is precisely what makes it so hard to solve POMDP problems. The above theorem shows that this phenomenon does not arise due to our particular solution strategy (belief parametrization). It is in fact an inherent property of the problem we are considering. It is thus unlikely that we will have a finite-time algorithm to solve the infinite horizon POMDP.

Theorem 4.7 ([HLR07]). *Even if the optimally reachable belief space, $\mathcal{R}(b_0)$ can be covered by polynomially many δ-balls, computing an approximation of $J^*(b_0)$ such that $|J(b_0) - J^*(b_0)| \leq \epsilon \ |J^*(b_0)|$ is NP-hard.*

Theorem 4.8 ([HLR07]). *If δ-cover C of $\mathcal{R}(b_0)$ with $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma \ l_{\max}}$ is given, an approximation such that $|J(b_0) - J^*(b_0)| \leq \epsilon$ can be computed in time*

$$\mathcal{O}\left(|C|^2 + |C| \ \log_\gamma \frac{(1-\gamma)\epsilon}{2l_{\max}}\right)$$

The above two theorems effectively illustrate the advantage of having an approximation of the reachable space. Even if we can calculate an approximation of the optimally reachable space, and it is polynomial sized say, it is still NP-hard to compute the value function. On the other hand, if somebody gives the δ-covering of the reachable space, computing the value function is just $\mathcal{O}(|C|)$.

Remark 4.9. These two sections summarize the state-of-the-art approaches to solve POMDPs. Key features to keep in mind are as follows.

- Reformulating the problem of stochastic control as a completely observed MDP where the state of the system is the conditional distribution (belief) immediately results in the state-space being infinite dimensional. As such, any kind of dynamic programming method must suffer from the *curse of dimensionality*. Results of the form described in Theorems 4.5 and 4.7 effectively show that it is unlikely that computationally efficient and *complete* algorithms can be found.

- Since the space of beliefs is so large, any structure we can infer for a given problem at hand results in tremendous speed-up for search algorithms. There are few key differences in the way the fastest algorithms leverage this observation. While discrete POMDP solvers approximate the space of beliefs by sampling optimal sequences of actions and observations, motion planning methods impose a pre-defined structure on the nature of beliefs (e.g. Gaussian or finitely parametrizable). The former methods alleviate the *curse of dimensionality* by reducing the sampling space but still fall prey to the *curse of history* in the sense that they cannot solve long time-horizon problems. Motion planning methods impose a strict finitely parametrizable structure on the belief space so the *curse of dimensionality* does not affect them, but since they make use of the separation theorem for problems consisting of linear systems or Gaussian beliefs, they can evade the *curse of history* very well. Another significant

advantage of motion planning methods is that they are based on random sampling strategies and are able to quickly able to explore large high-dimensional state spaces.

4.4 Problem formulation

The major contribution of this work is that it attempts to combine the advantages of both discrete POMDP solvers and motion planning methods. The aims of this work are as follows,

- Discrete solvers can solve problems with very large state spaces efficiently because they explore only the reachable set of beliefs. Our algorithms improve upon this to create successive approximations of the reachable sets. Along the way, we use the power of discrete solvers to solve successively harder problems to eventually obtain a solution to the continuous time POMDP.

- Discrete solvers work on stochastic systems which are represented by discrete dynamics. Most real world problems are inherently continuous in nature. It is thus worthwhile to create efficient approximations of general continuous time systems to pass them on as inputs to discrete POMDP solvers. This problem can be solved in numerous ways, we use the Markov chain approximation method discussed in Chapter 2 to solve it. As an added advantage of using this method, our discrete approximations are

  - General and applicable to any continuous time system. In particular, we do not have to worry about the shape of the state space or boundary effects while picking a particular discretization
  - Very easy to compute
  - Can be shown to converge with proven convergence rates

  The crucial advantage however is that successive approximations of the POMDP problem will require finer discretizations. Instead of obtaining them from scratch, the Markov chain approximation lends itself to a very easy *incremental* implementation i.e. a finer discretization is obtained from a coarse discretization very efficiently.

- Random sampling strategies of motion planning algorithms are able to explore vast state-spaces quickly. It is a direct consequence of the fact that they do not depend upon any *a priori* discretization of the state space and thus the Voronoi bias is incorporated in the method itself. These concepts can be utilized to create computationally efficient approximations of large POMDP problems.

4.4.1 Problem definition

We formulate the continuous time, continuous-state partially observed stochastic control problem in this section. The methods discussed in Chapter 2 will be used in the following sections to provide a solution to this problem.

Problem 4.10. *Let the dynamics of a system be given as*

$$dx(t) = f(x(t), u(t))\ dt + F(x(t))\ dw$$

with $x(t) \in \mathbb{R}^d$, the control belongs to some compact set $u \in \mathcal{U} \subset \mathbb{R}^l$ and $w(t)$ is the standard m-dimensional Wiener process. The functions $f(x,u) : \mathbb{R}^d \times \mathbb{R}^l \to \mathbb{R}^d$ and $F(x) : \mathbb{R}^d \to \mathbb{R}^{d\times l}$ called "drift" and "diffusion" terms respectively are bounded and continuous. Observations for this stochastic system are given by a process $H(t)$ as

$$H(t) = g'(x(t)) + G'(x(t))\ v'$$

where $H \in \mathbb{R}^k$ and v' is standard n-dimensional white Gaussian noise. $g'(x) : \mathbb{R}^d \to \mathbb{R}^k$ and $G'(x) : \mathbb{R}^d \to \mathbb{R}^{k\times n}$ are bounded and continuous. Find a control policy $\pi(t) \in \mathcal{U}$ which is measurable with respect to the σ-algebra produced by the processes $x(t)$ and $y(t)$ i.e. $\pi(t) \in \mathcal{F}_t^x \cup \mathcal{F}_t^y$ such that it minimizes the cost function

$$J = \mathbb{E}\left[\int_0^T l(x,u,t)dt + L(x(T))\ \right]$$

with $x(0)$ drawn from some distribution $b(0)$. Again $l(x,u,t)$ and $L(x(T))$ are bounded and continuous. The time T can be finite or be be defined as the exit time from some compact set G.

Note that the boundedness assumptions in the problem definition exist for technical reasons such as using convergence theorems in function spaces (Dominated Convergence Theorem / Bounded Convergence Theorem). It is ironical that even the simplest systems such as a linearly dependent drift and observation flout these assumptions. However it is to be noted that we have solutions for the linear case in the form of Linear Quadratic Gaussian (LQG) case and hence need not worry about this. Note that if we desire to steer the system to some terminal probability distribution say b_f, we can write the terminal cost as a penalty function in the form $L(b(T), b_f)$.

Lemma 4.11. *Problem 4.10 is equivalent to minimizing the cost function*

$$J = \mathbb{E}\left[\int_0^T l'(b(t), u(t), t)\ dt + L'(b(T))\right]$$

for some functions $l'(\cdot)$ and $L'(\cdot)$ where $b(t)$ is the normalized conditional distribution of the state given all the past history of observations i.e. $b(t)$ is the normalized solution of the stochastic PDE

$$db(t) = \mathcal{L}_f(b(t))\ dt + g(x(t))^T b(t) dy(t)$$

Proof. Using the Law of Iterated Expectations and Fubini's Theorem, the cost function in

Problem 4.10 can be given as

$$\begin{aligned}\mathbb{E}\left[\int_0^T l(x,u,t)dt + L(x(T))\right] &= \mathbb{E}\left[\int_0^T l(x,u,t)dt\right] + \mathbb{E}\left[L(x(T))\right] \\ &= \mathbb{E}\left[\int_0^T \mathbb{E}[l(x,u,t)\,|\,\mathcal{Y}_t]\;dt\right] + \mathbb{E}[\;\mathbb{E}[L(x(T)\,|\,\mathcal{Y}_T]\;]\end{aligned}$$

This is equivalent to the above cost function with

$$l'(b(t),u(t),t) = \mathbb{E}[l(x,u,t)\,|\,\mathcal{Y}_t] = \int_{\mathbb{R}^d} l(x,u,t)\;b(x(t))\;dx(t)$$

and

$$L'(b(t)) = \mathbb{E}[L(x(T))\,|\,\mathcal{Y}_T] = \int_{\mathbb{R}^d} L(x(T))\;b(x(T))\;dx(T)$$

■

The above lemma is a version of Theorem 4.2. Thanks to it, we will freely use either of the two representations of the cost function in the upcoming parts of the thesis. For the purposes of proving theorems, we will be interested in an observation model which is itself given by a stochastic differential equation. That the two observation models are equivalent is proved in the following lemma.

Lemma 4.12. *The observation model in Problem 4.10 is equivalent to the observation model*

$$dy(t) = g(x(t))\;dt + G(x(t))\;dv$$

for some functions $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^k$ and $G(\cdot) : \mathbb{R}^{d\times n}$ and where $v(t)$ is standard n-dimensional Wiener process.

Proof. Let

$$y(t) = \int_0^t H(s)\;ds.$$

We then have

$$dy(t) = g'(x(t))\;dt + G'(x(t))\;dv$$

where $v(t)$ is standard n-dimensional Wiener process by the definition of the Ito integral. Thus in the above formulation of the lemma, we also have $y = \int H$, $g' = g$ and $G' = G$.

Finally, note that by performing linear operations like integration we are not changing the amount of information contained in the signal $H(t)$. Thus the two observation models are equivalent as far as state estimation is concerned. In other words, if $H(s)$ is known for $s \le t$, we also know $y(s)$ for $s \le t$. ■

The notion that observation at time t is given by a random variable which is normally distributed with mean $g(x(t))$ and variance $G\;G^T$ will be useful in constructing the algorithms. On the other hand, the fact that equivalent observations can also be given by some Ito process will be useful to prove that the conditional expectation calculated along the discrete Markov chain and the original process converge.

4.5 Construction of discrete POMDPs

As described in Section 4.2, value iteration on the large belief spaces of POMDPs grows computationally intractable very quickly and is impractical for problems beyond a few states. We instead combine the advantages of random sampling algorithms along with the Markov chain approximation method to create discrete POMDPs. These problems are constructed in such a way that they converge in the limit to the original problem given in Problem 4.10. This is the major result of the following sections. In addition to doing so, we will leverage the fact that powerful discrete POMDP solvers like SARSOP are available. The procedure then is to solve each of the discrete elements of our sequence M_n using a discrete solver and ultimately solve the complete continuous time problem. Doing so appears futile if we solve each new element of the sequence freshly from scratch. Hence we will look for methods which utilize the solution of the previous POMDP of the sequence to arrive at the solution of the next one. Roughly, the cost function approximation of M_n will be a lower bound on the value function of M_{n+1}. Successive iterations on smaller elements M_k to obtain a solution will be shown to be easier than directly attempting to solve M_n. This is because approximating the optimally reachable set $\mathcal{R}^*_n(b_0)$ is the crux of solving POMDPs efficiently. Larger the dimension of the belief space, harder this gets. The belief spaces of $M_k;\ k < n$ are effectively lower dimensional cuts of the larger belief space and thus approximations of those smaller belief spaces can be used to obtain an approximation of the larger space.

4.5.1 Primitive procedures

Most of the steps of the algorithm are similar to the construction of the Markov chain as discussed in Section 2.4.1. There are however some differences because we are constructing MDPs instead of Markov chains for the POMDP. Recall that a discrete POMDP is a tuple $M = (S, U, O, T, P, Q, b_0)$ where S is a finite set of states, U is a finite set of controls, O is a finite set of observations, T is the set of holding times, $P(u);\ u \in U$ is the state transition matrix while Q is an observation matrix which gives the probability of observations of the form $M(o, z) = \mathbb{P}(o \,|\, z)$.

Sampling : Let $x \in \mathcal{S} \subset \mathbb{R}^d$. We will primarily use uniform random sampling procedure to populate the set of states of the Markov chain S. The `Sample` procedure returns states sampled independently and uniformly from $\mathcal{S}$. In the event that there are state constraints, let the set of constrained states be denoted by $\mathcal{S}_{obs}$ while the free space is denoted by $\mathcal{S}_{free}$. The `Sample` procedure then returns a state $z \in \mathcal{S}$ such that $z \in \mathcal{S}_{free}$. The procedure `SampleControl` samples one value uniformly randomly from the set $\mathcal{U}$ of admissible controls.

Neighboring states : This procedure remains exactly the same. Given $z \in \mathcal{S}$ and a finite set $S \subseteq \mathcal{S}$ of states, the procedure $\texttt{Near}(z, S)$ returns the set of all states that are within a distance of $r = \gamma\ (\log n/n)^{1/d}$ from z, i.e.,

$$\texttt{Near}(z, S) = \left\{ z_k \in S,\ z_k \neq z : \|z_k - z\|_2 \leq \gamma \left(\frac{\log n}{n}\right)^{1/d} \right\}$$

where $n = |S|$, $d = \dim(S)$ and $\gamma > 0$ is a constant.

Transition Probabilities The transition matrix P for an MDP is a function of a particular chosen control. Let U be the set of sampled controls with $|U| = m$ i.e. $u_1, \dots u_m \in U \subset \mathcal{U}$. For every one of these controls, execute the `ComputeTransProb` procedure in Section 2.4.1 once i.e. effectively the local consistency conditions are changed as follows,

- $\Delta t(z,u) \to 0$ for all $z \in S,\ u \in U$
- $\dfrac{\mathbb{E}[\Delta\xi(z,u)]}{\Delta t(z,u)} \to f(z,u)$
- $\dfrac{\mathbb{E}[(\Delta\xi(z,u) - \mathbb{E}[\Delta\xi(z,u)])^2]}{\Delta t(z,u)} \to F(z)F^T(z)$

The transition probabilities obtained are a function of both the state $z \in S$ and the control $u \in U$. In the successive sections, we will give explicit dependence of the number of sampled controls m on the number of states in the MDP n. For now, it will suffice to assume that m controls are sampled uniformly from $\mathcal{U}$ and m is a function of n.

Time Intervals : The continuous time system travels for different distances depending upon which control is chosen. Thus the holding times are a function of the controls in the MDP case. Given a state $z \in S$ and a control $u \in U$, the procedure `ComputeHoldingTime`(z, S) returns

$$\Delta t(z,u) = \frac{r^2}{\|F(z)F^T(z)\|_2 + r\|f(z,u)\|_2},$$

where r is as given in the procedure `Near`(z, S).

Observation Probabilities : Given a state $z \in S$, the procedure `ComputeObsProb`(`o`) returns a function $\mathbb{P}(\cdot \mid o = g(z)\ )$ which is

$$p(g(z') \,|\, o) = \eta\ N(g(z'), g(z), G(z)\ G^T(z))$$

where $z' \in Z_{near}$, $N(x, \mu, \Sigma)$ means the probability density of a normal random variable with mean μ and variance Σ calculated at x and η is a normalizing constant.

Connect State : The procedure `ConnectState` acts on a state $z \in S$. It computes the holding time for z, transition and observation probabilities for that particular state.

With these procedures in tow, we are ready to give the full algorithm to construct a discrete POMDP approximation from a continuous system.

CreatePOMDP : This is essentially a wrapper procedure around Algorithm 4.4 and given a POMDP with n states denoted as M_n, it outputs a more refined POMDP M_{n+1} by running Algorithm 4.4. This creates the modified Markov chain construction described in Section 3.3 because the as discussed before, the holding times of all the states in the MDP have to be the same for running a filtering algorithm on it. The difference however

is that the holding times in the POMDP case are a function of the sampled controls so the minimization is over all the states and all the controls i.e.,

$$\Delta t_n = \min_{z \in S_n,\ u \in U_n} \Delta t(z, u)$$

4.5.2 Algorithm

We can get two algorithms for the construction of the approximating POMDP. In the first version called as "batch" we only look at creating the n^{th} element of the sequence M_n. It draws upon the same concepts used in creating approximating Markov chains in Chapter 2.

Batch construction

Algorithm 4.2: Batch POMDP

```
for k ≤ N do
    S, O ← Sample;
for k ≤ m do
    U ← SampleControl;
for z ∈ S do
    (S, P, T, Q) ← ConnectState(z, S, P, T, Q, U);
return (S, P, T, Q, U);
```

Algorithm 4.3: `ConnectState`(z, S, P, T, Q, U)

```
for u ∈ U do
    Δt ← ComputeHoldingTime(z, u, |S|);
    Z_near ← Near(z, S);
    P(· | z, u) ← ComputeTransProb(z, Z_near, u, Δt);
    Q(· | g(z)) ← ComputeObsProb(z);
    T ← T ∪ (Δt);
return (S, P, T, Q);
```

This construction of the POMDP is very similar to that of Markov chains in Section 2.4.2. There is one key difference however which we will discuss here.

- **Sampling controls** The procedure `ComputeTransProb` assumes m sampled controls which form an approximate set $U_m \subset \mathcal{U}$. For the batch algorithm we can fix this number of controls to be any reasonable number. They can be sampled for example uniformly randomly from the compact set $\mathcal{U}$. The crucial part of sampling controls is to ensure that the number of sampled controls grows as we add more states to S_n. It should grow in such a way that the set U_m is a dense subset of $\mathcal{U}$ in the limit. In particular, an $\mathcal{O}(\log n)$ uniformly randomly sampled controls will ensure that U_m is dense.

The following section proves that the cost is approximated arbitrarily well in spite of working only with a subset of $\mathcal{U}$. To put it more precisely, any control signal of the form $u(t)$ for $t \in [0, T]$ can be approximated arbitrarily well by a piecewise constant control on a dense subset.

Incremental construction

Algorithm 4.4: Incremental construction of POMDP

```
1  n ← 0;
2  U_0 ← ∅;
3  for k < a do
4      u ← SampleControl;
5      U_k ← U_{k−1} ∪ {u};
6  m ← a;
7  while n < N do
8      S_n, O_n ← Sample(S_{n−1}, O_{n−1});
9      (S_n, P_n, T_n, Q_n) ←
       ConnectState(z, S_n, P_{n−1}, T_{n−1}, Q_{n−1}, U_{m−1});
10     Z_near ← Near(z, S_n);
11     U ← U_{m−1};
12     if ⌊c (log n − log(n − 1))⌋ > 1 then
13         u ← SampleControl;
14         U ← U_{m−1} ∪ {u};
15         m ← m − 1;
16     for z_near ∈ Z_near do
17         (S_n, P_n, T_n, Q_n) ← ConnectState(z_near, S_n, P_n, T_n, Q_n, U);
18     U_m ← U;
19     n ← n + 1;
20 return (S_N, P_N, T_N, Q_N, U_M);
```

The difference between the construction of an incremental POMDP and that of the Markov chain in Chapter 2 are lines 11-15. The set U_m has to grow as $n \to \infty$. At the same time, we need to ensure that every state contains transitions using the newly sampled control. Using similar intuition as the construction of incremental Markov chains, we refine / recalculate the transition probabilities for all states in the set Z_{near} of every newly sampled state z. The aim is to sample an $\mathcal{O}(\log n) = c \log n$ controls if the MDP has n states. Hence if $\lfloor c\ (\log n - \log(n-1)) \rfloor > 1$ where $\lfloor x \rfloor$ is the largest integer smaller than x, we sample a new control u and add it to the set of all controls U_m. It was proved in Theorem 2.5 that every state $z \in S_n$ is picked up for refinement infinitely often in the limit. This ensures that every state will have a new transition matrix using the newly sampled control after finitely many iterations of the incremtal algorithm.

Cost function approximation Given a discrete POMDP M_n created by Algorithms 4.2 or 4.4, we can use any point-based solver for example Algorithm 4.1 to obtain an approxi-

mation to the value function. In particular, with an input of M_n, the output of SARSOP is a belief tree $T_n^R(b_0)$ rooted at the initial belief b_0 (which is a probability mass function over S_n) and the set of α-vectors $\mathcal{A}_n$. A small discussion on the nature of discrete cost functions will be helpful at this point. Consider a discrete POMDP with the cost function given in the form

$$J = \sum_{k=1}^{T} \gamma^k \; l(x_k, u_k) + \gamma^T \; L(x_T)$$

where T is some hitting some of a goal set ∂G, $x_k = x(k) \in S$ and $a_k = u(k) \in U$. We will briefly discuss how a continuous time cost function can be translated into this form given above. For two discrete POMDPs created by Algorithm 4.4, say M_{n_1}, M_{n_2}, since they approximate the same continuous time cost function, the time that the optimal policy takes in both cases is the same. i.e. if the optimal policy takes k_{n_1} steps for M_{n_1} and k_{n_2} steps for M_{n_2} we have

$$k_{n_1} \; \Delta t_{n_1} = k_{n_2} \; \Delta t_{n_2} = c_1$$

For problems of the kind where $l(\cdot, \cdot) = 0$ we can easily find the discount factor γ_n for successive discrete POMDPs. In particular

$$\gamma_n^{c_1/\Delta t_n} = c_2 < 1 \implies \frac{\log \gamma_n}{\Delta t_n} = c' < 0 \implies \gamma_n = e^{-c \; \Delta t_n}$$

for some $c > 0$.

Successive value function approximation Let $\mathcal{A}_n$ the set of α-vectors obtained by solving the POMDP M_n having n states. Let J_n be the approximation of the cost function created by $\mathcal{A}_n$. In other words,

$$J_n(b) = \min_{a \in \mathcal{A}_n} \; (b \; . \; \alpha)$$

We are interested in obtaining a bound on J_{n+1} based on $\mathcal{A}_n$ which will be used by SARSOP to obtain $\mathcal{A}_{n+1}$. There are two steps in such a program (i) the belief tree maintained by SARSOP T_n^R consists of beliefs b_n which are probability mass functions supported on the set S_n (ii) set of α-vectors $a \in \mathcal{A}_n$ is a n-dimensional vector. We thus need to approximate both the belief tree T_{n+1}^R and the α-vectors $\mathcal{A}_{n+1}$.

Approximating T_{n+1}^R : Every belief $b_n \in T_n^R$ is a probability mass function supported on the discrete set S_n. If b_{n+1} be the corresponding new belief supported on S_{n+1}, we can calculate it by equating the moments of the two distributions as follows. Let b_n^k be the probability mass at point $z_k \in S_n$. Given b_n^k for all $k \leq n$, we have to calculate b_{n+1}^k for

$k \leq (n+1)$ such that all the moments of the two distributions b_n and b_{n+1} are equal i.e.,

$$\begin{aligned} b_n^1 + \ldots b_n^n &= b_{n+1}^1 + \ldots b_{n+1}^{n+1} = 1 \\ b_n^1 \ z_1 + \ldots b_n^n \ z_n &= b_{n+1}^1 \ z_1 + \ldots b_{n+1}^{n+1} \ z_{n+1} \\ &\vdots \\ b_n^1 \ (z_1)^n + \ldots b_n^n \ (z_n)^n &= b_{n+1}^1 \ (z_1)^n + \ldots b_{n+1}^{n+1} \ (z_{n+1})^n \end{aligned}$$

These are $n+1$ linear equations for $n+1$ unknowns $b_{n+1}^1 \ldots b_{n+1}^{n+1}$ and can be solved easily. In the absence of this, the value b_{n+1}^{n+1} can also be obtained by local averaging as described in the procedure `ApproximateDensity` in Section 3.5.

Approximating $\mathcal{A}_{n+1}$: For each $\alpha \in \mathcal{A}_n$, let α^k denote its value at z_k i.e., $\alpha^k = J_n(\delta(z_k))$ if $\delta(z_k)$ is the Dirac measure at z_k and if α were the best α-vector at a belief $\delta(z_k)$. This observation can be used to calculate the value α^{n+1} for all $\alpha \in \mathcal{A}_n$ to obtain a new set of α-vectors which are vectors of length $n+1$ and constitute $\mathcal{A}_{n+1}$. This is done by starting with the belief $b(0) = \delta(z_{n+1})$ and taking the optimal action u associated with α to obtain a new belief say $b(1)$. The bounds on the value function at this new belief can be approximated from the previous paragraph, let $\underline{J_{n+1}}$ be the lower bound. Then

$$\alpha^{n+1} = R(z_{n+1}, u) \ + \underline{J_{n+1}}.$$

$\mathcal{A}_{n+1}$ thus obtained forms a lower bound for the cost function of the new POMDP M_{n+1} which can be used by SARSOP for biasing its search for the optimal policy. Note that this was not implemented in the examples discussed in the experiments section.

4.6 Analysis

In gist, this section proves that the optimal cost as calculated on the approximate POMDPs converges almost surely to the optimal cost as calculated on the original continuous time POMDP as given in Problem 4.10. A few offshoots of the proofs are the facts that not only does the cost converge, but the control policies also converge albeit in a suitably relaxed fashion. After this section we can claim that the policies approximated by SARSOP while solving successive POMDPs as generated in Section 4.5 result in a convergent policy which is optimal for the continuous time problem.

A short summary of the program used in this section will be helpful before we dig into the mathematics. This follows the scheme in [KD01] for the convergence of the cost function of approximate Markov chains (see Theorem 2.2). There are some technical differences due to the fact that we are working with conditional distributions and this section fills in these details.

- The first part of this section considers uncontrolled processes. We saw in Section 4.2 that firstly, belief is a sufficient statistic for the cost function i.e. the cost function as described in Problem 4.10 is equivalently also given as an integral of the belief. Since

we are creating approximations to the original problem, the space of beliefs in our approximations is different from the original space. An obvious striking difference is that the approximate MDP belief is supported on a finite set (the set of states of the Markov chain) whereas the actual belief is a continuous function. Our first task will be to prove that these two objects converge in distribution. Belief being a sufficient statistic will then imply that the cost functions for the uncontrolled problems converge with probability 1.

- It might happen in finite difference approximations of continuous problems that the space of approximate controls is not rich enough to approximate certain types of controls which are actually optimal for the original continuous problem. It is a common trick to establish a measure on the space of controls called *relaxed controls* to alleviate such issues. We will spend considerable effort on understanding these controls and it will turn out that they are not that far away from original controls as far as the signals or the cost functions are concerned.

- Finally, the above two concepts come together and use weak convergence properties of the trajectories $x^n \Rightarrow x$, beliefs $b^n \Rightarrow b$ and controls $m^n \Rightarrow m$ to prove that the cost function of the approximate POMDP converges to that of the original continuous problem.

4.6.1 Convergence of belief trajectories

Theorem 4.13 ([Gog94]). *For two complete separable metric spaces S_1, S_2, suppose that X_n, Y_n are S_1, S_2 valued random variables that converge in distribution to X, Y respectively (X, Y are S_1, S_2 valued respectively). Define the Radon-Nikodym derivative as $dP_n/dQ_n = L_n(X_n, Y_n)$. Assume that X_n and Y_n become independent under Q^n with marginal distributions μ^n and ν^n. If $\mu^n \Rightarrow \mu$ and $\nu^n \Rightarrow \nu$ with $\mu \times \nu$ being the distribution of (X, Y) under some measure Q and if $Q^n \Rightarrow Q$ weakly then the following hold.*

(i) $P << Q$ *on* $\sigma(X, Y)$ *and* $dP/dQ = L(X, Y)$

(ii) For every bounded continuous function $F : S_1 \to \mathbb{R}$, the conditional distributions of $F(X_n)$ and $F(X)$ converge in distribution i.e.,

$$\mathbb{E}_{P^n}[F(X_n) \,|\, Y_n] \Rightarrow \mathbb{E}_P[F(X) \,|\, Y]$$

Theorem 4.14. *If $x_n \Rightarrow x$, the conditional distributions calculated with respect to the filtration generated by the observations $\mathcal{Y}_t = \sigma(y(t))$, $b^n(t)$ and $b(t)$ also converge in distribution i.e.,*

$$b_n(t) \Rightarrow b(t)$$

Proof. Consider the pair of stochastic differential equations

$$dx(t) = f(x(t))\ dt + F(x(t))\ dw(t)$$
$$dy(t) = g(x(t))\ dt + G(x(t))\ dv(t).$$

We use a change of measure from P to Q under which X is independent of Y. Q is defined by

$$\frac{dP}{dQ} = \exp\left[\int_0^T g(x(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x(t))|^2\ dt\right] \tag{4.6}$$

Then $P << Q$ on $\mathcal{F}_T^x \cup \mathcal{F}_T^y$ and the conditional expectation $\mathbb{E}[F(x(t)) \,|\, \mathcal{Y}_t]$ can be written as

$$\mathbb{E}_P[F(x(t)) \,|\, \mathcal{Y}_t] = \frac{\phi_t(y, F)}{\phi_t(y, 1)}$$

with

$$\phi_t(y, F) = \mathbb{E}_Q\left[F(x(t))\ \exp\left(\int_0^T g(x(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x(t))|^2\ dt\right)\right]$$

Using Theorem 2.2 of Chapter 2, the process $x(t)$ is approximated by a Markov chain resulting in trajectories $x_n(t)$. The conditional expectation thus becomes

$$\mathbb{E}_{P_n}[F(x(t)) \,|\, \mathcal{Y}_t] = \frac{\phi_t^n(y, F)}{\phi_t^n(y, 1)}$$

with

$$\phi_t^n(y, F) = \mathbb{E}_Q\left[F(x_n(t))\ \exp\left(\int_0^T g(x_n(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x_n(t))|^2\ dt\right)\right]$$

with the probability measure P_n that generates $x_n(\cdot)$ given here by

$$\frac{dP_n}{dQ} = \exp\left[\int_0^T g(x_n(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x_n(t))|^2\ dt\right].$$

Note that $x(\cdot)$ and $y(\cdot)$ are random variables taking values in $D[0, \infty)$ equipped with the Skorohod metric. We can use Theorem 4.13 to conclude that $\mathbb{E}[x_n(t) \,|\, \mathcal{Y}_t] \Rightarrow \mathbb{E}[x(t) \,|\, \mathcal{Y}_t]$ by verifying the following.

1. $(x_n(t), y(t))$ converges weakly to $(x(t), y(t))$ under the measure P_n. This is true from theorems proved in Chapter 2.

2. Q distribution of $(x_n(\cdot), y(\cdot), L_n(x_n, y))$ converges weakly to Q distribution of $(x(\cdot), y(\cdot), L(x, y))$. The first component is true because $x_n(\cdot)$ has the same distribution under P_n, P and Q. The second component is trivially true. The convergence of $L_n(\cdot, \cdot)$ to $L(\cdot, \cdot)$ is seen from the fact that

$$\left[\int_0^T g(x_n(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x_n(t))|^2\ ds\right] \to \left[\int_0^T g(x(t))\ dy(t) - \frac{1}{2}\int_0^T |g(x(t))|^2\ dt\right]$$

using the Skorohod embedding theorem in which $\tilde{x}_n \to \tilde{x}$ almost surely. Having thus verified all the conditions of Theorem 4.13, we can claim that

$$\mathbb{E}_{P_n}[F(x_n(t)) \,|\, \mathcal{Y}_t] \Rightarrow \mathbb{E}_P[F(x(t)) \,|\, \mathcal{Y}_t]$$

for *every* bounded continuous function $F(\cdot)$. By the Pormanteau theorem (see Ap-

pendix A), this means that indeed the conditional distributions at each time t converge i.e.

$$b_n(t) \Rightarrow b(t)$$

This convergence takes place with respect to the usual norm on the Hilbert space $L^2(\mathbb{R}^d \times [0,T])$ which is defined as follows. $\psi_k \Rightarrow \psi$ where $\psi_k, \psi \in L^2(\mathbb{R}^d \times [0,T])$ if

$$\int_{\mathbb{R}^d \times [0,T]} \psi_k f \; d\lambda \to \int_{\mathbb{R}^d \times [0,T]} \psi f \; d\lambda$$

where λ is the Lebesgue measure for all functions $f \in L^2(\mathbb{R}^d \times [0,T])$ which is the weak convergence criterion.

■

Remark 4.15. We are interested in an offline solution of the POMDP problem in this thesis. As such, a discrete solver is used to calculate the cost function over future belief trajectories which are sampled via actions and observations sampled in the sets U_n and O_n. The above theorem proves that the belief calculated on the Markov chain using the same observations as the continuous time process converge in distribution. We however need something stronger i.e. the beliefs calculated on the Markov chain using *sampled observations* converge in distribution to the beliefs calculated on the continuous time process using continuous time observations. More precisely, the conditional distributions also converge with the processes x, y are approximated by processes x_n, y_n which converge in distribution. This is fortunately true when the observation process has Gaussian noise as proved using some additional machinery in [Gog92]. Henceforth, let $\mathcal{Y}_t^n$ denote the filtration generated by observations coming only from the set O_n upto a time t.

Theorem 4.16. *The sequence $b_n(\cdot)$ is tight.*

Proof. Given $b_n(\cdot)$ we will prove the conditions of the Prokhorov's theorem to claim tightness. We follow the first part of the proof of Theorem 2.1 in [Gog94]. Let $\lambda_n(A, \omega) = \mathbb{P}_n(x_n(t) \in A \,|\, \mathcal{Y}_t)$. We thus have to prove that the sequence $\lambda_n(\cdot)$ is tight. Since $x_n(\cdot)$ is tight using Theorem 2.2, we have

$$\mathbb{P}_n(x_n(t) \in K_\epsilon) > 1 - \epsilon$$

for all n. Thus,

$$\begin{aligned} 1 - \epsilon < \mathbb{P}_n(x_n(t) \in K_\epsilon) &= \mathbb{E}_n[\lambda(K_\epsilon)] \qquad \text{(Iterated expectations)} \\ &= \int \lambda_n(K_\epsilon) \mathbf{1}_{\{\lambda_n(K_\epsilon) > 1 - 1/k\}} \; d\mathbb{P}_n + \int \lambda_n(K_\epsilon) \mathbf{1}_{\{\lambda_n(K_\epsilon) \leq 1 - 1/k\}} \; d\mathbb{P}_n \\ &< \frac{1}{k} \, \mathbb{P}_n \left(\lambda_n(K_\epsilon) > 1 - \frac{1}{k} \right) + \left(1 - \frac{1}{k} \right). \end{aligned}$$

Use $\epsilon = \frac{\epsilon'}{k \; 2^k}$ to get

$$\mathbb{P}_n \left(\lambda_n(K_\epsilon) > 1 - \frac{1}{k} \right) \geq 1 - \frac{\epsilon'}{2^k},$$

which implies

$$\mathbb{P}_n\left(\lambda_n(K_\epsilon) > 1 - \frac{1}{k} \quad \forall k\right) \geq 1 - \epsilon',$$

which proves that the sequence of measures λ_n (and equivalently b_n) is tight by Prokhorov's theorem. ■

Theorem 4.17. *The sequence $b_n(\cdot)$ converges weakly to a solution $b(\cdot)$ which is the solution of the stochastic PDE in Problem 4.11.*

Proof. Using a slight abuse of notation, let $b_n(\cdot)$ itself denote a convergent subsequence of $b_n(\cdot)$. We can always extract such a subsequence because of tightness proved previously in Theorem 4.16. Theorem 4.13 proves that the conditional distribution of states $b_n(t) = \mathbb{P}(x_n(t) \,|\, \mathcal{Y}_t^n)$ converges weakly to $b(t) = \mathbb{P}(x(t) \,|\, \mathcal{Y}_t)$ which is the solution of the stochastic PDE in Problem 4.11 for *any* t. The claim of this theorem then follows using Caratheodory's Extension Theorem which says that a measure on a field can be uniquely extended to its σ-algebra [**?**]. ■

4.6.2 Relaxed Controls

Convergence properties of the Markov chain approximation were used in Theorem 4.19 to establish the convergence of the uncontrolled cost function. For the analogous problem with controlled processes, we need to consider sequences of controlled Markov chains and in these cases convergence of the control process is also necessary in addition to the solution of the differential equations. Roughly, the nature of the convergent solution of the Markov chain is guaranteed merely by the local consistency conditions. However, the optimal controls for the individual chains can be arbitrary and we cannot force them to take a predefined form (e.g. feedback controls). This is due to the fact that unless the control space is compact in a sense, the infimum of the cost function will not in general be attained in the control space.

Example 4.18 (Non-existence of optimal controls). Consider a deterministic control problem with $u \in \mathcal{U} = [-1, 1]$ and the system given by

$$\dot{x}(t) = b(x(t), u(t)) = u(t).$$

Let the cost function be

$$W(x, u) = \int_0^\infty \left[x^2(t) + (u^2(t) - 1)\right] \, dt.$$

We can see that controls of the form

$$u_n(t) = (-1)^k \text{ on } \left[\frac{k}{n}, \frac{k+1}{n}\right) \text{ for } k = 0, 1, \ldots$$

result in $W(0, u_n) \to 0$ as $n \to \infty$ i.e., at $x(0) = 0$, the optimal control wants to take the values ± 1 simultaneously and there is no unique optimal control as such.

It is for this reason that we need to work with controls with are compact whereby we are assured that we still get the same infimum while taking limits as $n \to \infty$. "Relaxed controls" is a theoretical framework for this precise purpose. Note that this will be used only in proofs and will not present itself in the algorithms.

Given a compact control space $\mathcal{U}$, let $\mathcal{B}(\mathcal{U})$ denote the $\sigma-$algebra of its subsets. A relaxed control is then a Borel measure $m(\cdot)$ such that $m(\mathcal{U} \times [0,T]) = t$ for all $t \geq 0$. The derivative $m_t(\cdot)$ is defined as

$$m_t(A) = \lim_{\delta \to 0} \frac{m(A \times [t-\delta, t])}{\delta}$$

The system in the above example can now be written as

$$\dot{x}(t) = \int_{\mathcal{U}} b(x(t), \alpha)\ m_t(d\alpha)$$

with the cost function being

$$W(x,u) = \int_0^\infty \int_{\mathcal{U}} \left[x^2(t) + (\alpha^2(t) - 1)\right]\ m(d\alpha\ dt)$$

i.e. $m(A \times [0,T])$ is just the total time that the control $u(t)$ spends in the set $A \subset \mathcal{U}$ during the time interval $[0,T]$. Let us apply this to the above example. Let $\alpha_k = (-1)^k \in \mathcal{U}$ for $k = 1, 2$ be the values of control and let $m_t(\cdot)$ be a measure which takes values α_1, α_2 with equal probability. Thus

$$\dot{x}(t) = \frac{1}{2}\left[b(x, \alpha_1) + b(x, \alpha_2)\right] = \int_{\mathcal{U}} b(x(t), \alpha)\ m_t(d\alpha).$$

Relaxed control essentially convexify the possible velocities and the cost-rate at every time. The "optimal" relaxed control for this problem is then the measure $m_t(\cdot)$ which takes the values ± 1 with probability $1/2$.

The following theorems now enable us to prove convergence of the relaxed controls and the cost function.

4.6.3 Convergence of cost function

Theorem 4.19. *The cost function J_n converges to J almost surely where J is given by*

$$J_n = \mathbb{E}\left[L(b_n(T), T) + \int_0^T l(b_n(t), t)\ dt\right]$$

and

$$J = \mathbb{E}\left[L(b(T), T) + \int_0^T l(b(t), t)\ dt\right]$$

Proof. We will assume that the functions $l(\cdot,\cdot)$ and $L(\cdot,\cdot)$ are bounded and Lipschitz continuous. Theorem (4.13) proved that $b_n(t) \Rightarrow b(t)$. We can thus use the Mapping Theorem

(see Appendix A) for weak convergent sequences to claim that

$$f_n = L(b_n(T), T) + \int_0^T l(b_n(t), t)\ dt \Rightarrow L(b(T), T) + \int_0^T l(b(t), t)\ dt = f.$$

We know that $P_n \Rightarrow P$ where P_n is the distribution of b_n and P is the distribution of b. By the Portmanteau Theorem (see Appendix A) for any bounded continuous function f, we have

$$\int f dP_n \to \int f dP \qquad \text{a.s..}$$

Apply the above criterion for weak convergence to the functions f_n and f to get

$$\mathbb{E}_{P_n}\left[L(b_n(T), T) + \int_0^T l(b_n(t), t)\ dt\right] \to \mathbb{E}_P\left[L(b(T), T) + \int_0^T l(b(t), t)\ dt\right] \qquad \text{a.s.}$$

The proof for a finite fixed terminal time T is very easy as seen above. More conditions need to be checked for cases where T is a random time such as

$$T = \inf\ \{t :\ b(t) \notin G\}$$

for some compact set G. These situations arise in problems where we desire to drive the system to some terminal distribution.

Denote by T_n the time that the discrete POMDP leaves the set G i.e., $T_n = \inf\ \{t : b^n(t) \notin G\}$. Kushner proves that the sequence $T_n \Rightarrow T$ on the compactified interval $[0, \infty]$ in [KD01]. The proof of the above theorem in cases there T is a random exit time require additional machinery as regards to the exit times being continuous under the measure induced by the processes b_n. We will not dwell on these proofs here, please refer the proof of Theorem 9.4.3 in [KD01] for it. All the conditions go through except for the fact that beliefs belong to the Hilbert space $b_n \in L^2(\mathbb{R}^d \times [0, \infty))$ instead of $x_n \in D^d[0, \infty)$. ■

Theorem 4.20 (Theorem 10.1.1 in [KD01]). *If*

$$x(t) = x(0) + \int_0^t \int_{\mathcal{U}} f(x(s), \alpha) m_s(d\alpha ds) + \int_0^t F(x(s))\ dw(s) \tag{4.7}$$

and $x_n(0) \Rightarrow x_0$, then any sequence $\{x_n(\cdot), m_n(\cdot), w_n(\cdot)\}$ is tight. Let $\{x(\cdot), m(\cdot), w(\cdot)\}$ be the limit of a converging subsequence. Define

$$\mathcal{F}_t = \mathcal{F}(x(s), m(s), w(s), s \leq t).$$

Then $w(\cdot)$ is a $\mathcal{F}_t-$Wiener process, $m(\cdot) \in \mathcal{F}_t$, $x(0) = x_0$ and $x(\cdot)$ satisfies Equation (4.7).

Remark 4.21. This theorem thus shows that the limit of a sequence of controlled diffusion processes is also a diffusion process under our notion of relaxed controls. The essence of the theorem is that the limiting controls are admissible. This provides the basis for approximation of the limiting relaxed control by a piecewise constant control applied to the Markov chain approximation in the next theorem. As it turns out, the cost functions obtained by such a process are close.

Theorem 4.22 ([KD01]). *For a given $(m(\cdot), w(\cdot))$, let the solution to Equation (4.7) exist and be unique in the weak sense. Let the relaxed control cost function be defined by*

$$W'(x,m) := \mathbb{E}\left[\int_0^T \int_{\mathcal{U}} l(x(s),\alpha,s)\; m(d\alpha\; ds) + L(x(T))\right]. \tag{4.8}$$

Then there exists a probability space

$$(x_\epsilon(\cdot), u_\epsilon(\cdot), w_\epsilon(\cdot))$$

where ($w_\epsilon(\cdot)$ is standard Wiener process) with u_ϵ being an admissible piecewise constant control that satisfies Equation (4.7) and

(i) $\mathbb{P}\left(\rho_T(x_\epsilon, x) > \epsilon\right) < \epsilon$

(ii) $|W'(x,m) - W'(x,u_\epsilon)| < \epsilon$

The above theorem proves that any relaxed control can be approximated by an admissible ordinary control $u_\epsilon \in \mathcal{U}$. Notice that we have made use of the equivalence of the cost functions proved in Theorem 4.11 to use this theorem for the function $W'(x,m)$ defined above. Since the cost functions are equivalent, the theorem as cited can be used directly for the cost function which features belief $W(b,m)$. The program that we need to prove the convergence of the cost function after this is as follows.

- Let b, m be a solution such that it is ϵ away from the optimal cost. We would like to ensure some continuity in the relaxed controlled solutions i.e., does there exist a relaxed control m_n such that

  $$|W(b,m_n) - W(b,m)| \leq \delta$$

  The proof of this is given in Theorem 10.3.1 of [KD01].

- If the answer to the above question is yes then we move on to prove that the sequences of processes (b_n, m_n) are tight. If these sequences are tight, there exist subsequences which converge weakly to some solution (b,m). In that case we can claim

  $$W(b_n,m_n) \to W(b,m)$$

- It then remains to prove that the minimas also converge. Let $V_m(b) = \inf_{m\in\mathcal{U}} W(b,m)$. Then

  $$V_m(b_n) \to V_m(b)$$

- If we take the infimum over the space of ordinary controls, $V(b) = \inf_{u\in\mathcal{U}} W(b,u)$ we finally have using Theorem 4.22 that

  $$V_m(b) = V(b)$$

Theorem 4.23 (Theorem 10.4.1 in [KD01]). *If $x_n \Rightarrow x$, for any sequence of controls $u_n(\cdot)$, let $m_n(\cdot)$ denote the relaxed controls, let T_n denote the sequence of exit times for the*

discrete approximations. Then the sequence

$$\{x_n, b_n, m_n, w_n, T_n\}$$

is tight. In particular, there exists a subsequence which converges to a limit say $\{x, b, m, w, T\}$ *such that it satisfies the equation*

$$x(t) = x(0) + \int_0^t \int_{\mathcal{U}} f(x(s), \alpha)\ m(d\alpha)\ ds + \int_0^t F(x(s))dw \tag{4.9}$$

Proof. This theorem adds additional quantity $b_n(\cdot)$ to the tuple proved in the reference. Hence we only need to prove that the sequence $b_n(\cdot)$ is tight which is proved in Theorem 4.16. ■

Theorem 4.24 (Theorem 10.5.1 in [KD01]). *If we have*

$$\{x_n, b_n, m_n, w_n\} \Rightarrow \{x, b, m, w\}$$

weakly, then

$$W(b_n, m_n) \to W(b, m) \geq V_m(b)$$

Also,

$$\liminf_n\ V_m(b_n) \geq V_m(b)$$

and

$$\limsup_n\ V_m(b_n) \leq V_m(b)$$

Proof. (Sketch) The first part of theorem can be seen by the fact that as proved above every sequence of the form

$$\{x_n(\cdot), b_n(\cdot), m_n(\cdot), T_n\}$$

has a weakly convergent subsequence whose limiting processes satisfy Equation 4.9. Denote the limit of this subsequence by $\{x(\cdot), b(\cdot), m(\cdot), T\}$. By the definition of weak convergence, we than have

$$\mathbb{E}_{\mathbb{P}_n}\left[\int_0^{T_n} \int_{\mathcal{U}} l(x_n(s), \alpha)\ m_n(d\alpha)ds + L(x_n(T_n))\right] = \mathbb{E}_{\mathbb{P}}\left[\int_0^{T} \int_{\mathcal{U}} l(x(s), \alpha)\ m(d\alpha)ds + L(x(T))\right]$$

There are some technicalities about the continuity of the exit times T_n which are described in detail in the reference. We thus have

$$W(b_n, m_n) \to W(b, m) \geq V_m(b)$$

where the last inequality is trivially true. The second inequality i.e.

$$\liminf_n\ V_m(b_n) \geq V_m(b)$$

is only one side of the proof of convergence of the cost function. The other side is given by the third inequality. In order to prove that, construct an almost optimal control using

Theorem 4.22 and adapt it to use on the Markov chain, i.e., evaluate $V_{m_n}(b_n)$. The optimal cost function $V_m(b)$ is minimal and hence

$$V_{m_n}(b_n) \geq V_m(b).$$

Now use weak convergence of the process $m_n(\cdot)$ to $m(\cdot)$ on the left hand side to claim the final inequality. ■

Remark 4.25. This theorem thus proves that the cost function approximation as calculated on the approximate Markov chain converges to the cost function of the original stochastic system. Since $m_n \Rightarrow m$ weakly, it also means that the relaxed controls converge. It is to be noted that the controls converge in this weak space of relaxed controls but barring boundary cases like the one constructed in Example 4.18 the controls converge converge in the original space $\mathcal{U}$ as well.

4.7 Experiments

This section discusses a few simulation experiments where a continuous time partially observable stochastic control problem is broken down into discrete POMDPs to be solved via a point based solver (e.g. SARSOP).

4.7.1 Linear Quadratic Gaussian

Let us consider the LQG problem. It involves control of a stochastic linear system with additive Gaussian noise. Denote the system as

$$\dot{x} = Ax + B_u u + \tilde{w}$$

with observations of the form

$$y = Cx + \tilde{v}$$

where $\tilde{w} \sim N(0, R_{ww})$ and $\tilde{v} \sim N(0, R_{vv})$. The objective them is to minimize a cost function of the form

$$J = \mathbb{E}\left[\int_0^T (x' R_{xx} x + u' R_{uu} u)\ dt\right].$$

For a finite time T, the minimum cost can be given by

$$J_{min} = \int_0^T \text{tr}\left[P B_w R_{ww} B_w'\right] + \text{tr}\left[Q K_{ss}' R_{vv} K_{ss}\right]\ dt$$

for $u = -K_{ss}\hat{x}$. $\hat{x}$ is the estimated state of the Kalman filter. P and Q are the solutions of the Riccati equation for control and estimation respectively. Also, if there is a penalizing factor of $e^{2\alpha t}$ to the cost, the equations remain the same after changing $A \to A + \alpha I$ in the Riccati equation. To convert the continuous time cost function into the discrete time cost function of the form $J = \sum_{k=0}^{T} \gamma^k l(x, u)$ that SARSOP requires, we calculate an approximate α. If $\gamma \sim 1$ then $\alpha \sim \frac{1-\gamma}{2\Delta t}$. Note that there is no terminal state or terminal cost in this example.

Now consider a linear system with dynamics,

$$dx(t) = -x(t)\ dt + u(t)\ dt + F\ dw$$

and observations

$$y(t) = x(t) + G\ \tilde{v}$$

where $\tilde{v}$ is unit-variance white Gaussian noise. For the purposes of constructing the Markov chains, the state-space is bounded with $x, y \in [-1, 1]$ and $u \in [-0.5, 0.5]$. Using Algorithm 4.4 to construct the Markov chain for the above dynamics with $(5 \log n)$ controls and observations uniformly sampled from the bounded state space where n is the number of states in the POMDP. Table 4-4 shows the cost obtained by SARSOP after solving a

States	$\Delta t(s)$	Cost (95%)
20	0.398	-0.406 $\pm$ 0.016
40	0.247	-0.391 $\pm$ 0.017
80	0.146	-0.355 $\pm$ 0.014
160	0.083	-0.341 $\pm$ 0.015

Figure 4-4: Convergence of the discrete LQG cost using SARSOP. The optimal cost for the continuous time problem is -0.308.

POMDP with different number of states. If $x(0) \sim N(0.8, 0.03)$ and $F, G = 0.1$, the LQG cost is -0.308. It can be seen that the cost approximated on the discrete POMDPs reduces as the number of states in the problem increases. With larger than 150 states, the discrete problem becomes too large for SARSOP to solve and hence we provide results only upto 160 states. Figure 4-5 shows an example simulated run using the discrete policy.

4.7.2 Light-dark domain

We test the proposed approach on a popular problem known as "light-dark domain" [PPK+12]. In this problem, a robot with certain specified dynamics has to localize its position before entering a pre-defined goal region to obtain a reward. However, the robot does not have access to accurate observations always. There are regions in the state-space with beacons (i.e. light regions) where highly accurate observations can be obtained. In all other parts of the state-space, the magnitude of observation noise is very large (i.e. dark regions). Depending upon where the robot starts from, it might need to move into the light region and localize itself before venturing into the dark region again to obtain a discounted reward. Consider a robot with dynamics

$$dx(t) = u(t)\ dt + dw$$

and observations given by

$$y(t) = x(t) + G(x(t))\ \tilde{v}$$

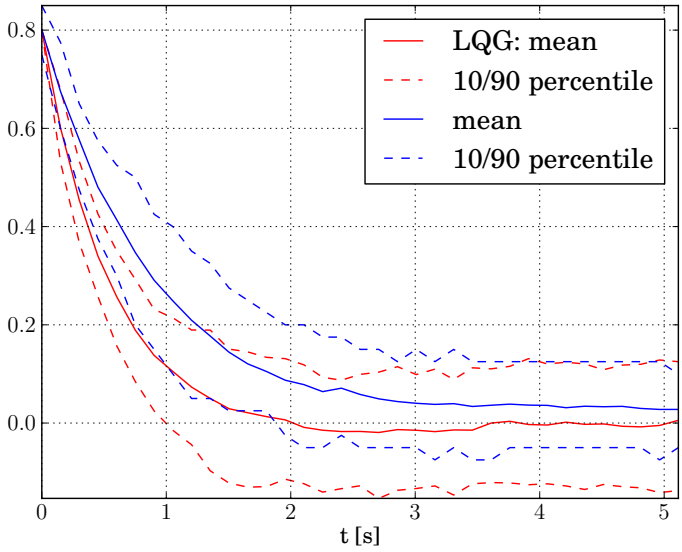


Figure 4-5: Simulated policy using 80 states for the LQG problem. This figure shows the mean and percentiles for state trajectories starting from $N(0.8, 0.03)$ for the SARSOP policy (blue) and the optimal LQG policy calculated using the Riccati equation (red).

Similar to the previous problem, $x, y \in [-2, 2]$. $G(x)$ gives the location of beacons in this problem. Let $G(x)$ be defined as follows,

$$G(x) = \begin{cases} \epsilon & : \quad |x - b_1| < e_1 \\ 1/\epsilon & : \quad \text{otherwise} \end{cases}$$

Note that this is a slightly stronger condition than say a quadratic gradient in observation noise covariance. Roughly, the exploration in the belief tree can be biased towards searching for policies which reduce the robot's covariance by traveling into the light region. Nevertheless, the current approach works in either case. We will consider a 1-dimensional example with a single beacon first. The system gets a reward of the form,

$$R(x) = \begin{cases} 1000 & : \quad |x - g_1| < e_1 \\ -1000 & : \quad \text{otherwise} \end{cases}$$

The cost function considered is

$$J = \sum_{k=0}^{T} \gamma^k \; l(x_k, u_k) - \gamma^T R(x(T))$$

where the term $l(xk, u_k)$ is quadratic control cost. Reward $R(x)$ is obtained at time T. The terminal time T is decided by a terminal action u_{goal} which the system has to take in order to get the reward. The agent travels to the goal region and takes this terminal action when it is certain on being inside the goal region $|x - g_1| < e_1$. If this happens, it gets a reward of 1000 otherwise it gets a penalty of -1000 and the problem ends. An example policy calculated for the case when $b_1 = 0.9, g_1 = -0.9, e_1 = 0.1$ is shown in Figure 4-6.

Note that the belief is a probability mass function projected on the sampled states (which form a regular grid in this case). The control policy is then to travel first to the light region marked by green rectangles, localize itself and then proceed towards the goal marked by red rectangles to obtain the reward.

Single Beacon : The advantage of the proposed approach is evident in these example. This is a complete continuous time problem that has been conventionally solved using techniques like belief space planner [PPK+12] or convex optimization [JT12]. It needs additional assumptions of discrete / linear systems, Gaussian belief space. The same problem was solved using a completely discrete point based solver with just 20 sampled states. It should be noted that the output of such a solver is a set of α-vectors that give the expected reward at every belief point b as $J(b) = \min_{\alpha \in \mathcal{A}}(b \,.\, \alpha)$. The filtering algorithm as described in Chapter 3 gives a probability mass function projected on the same sampled set S_n and thus can be used with an offline solution in this fashion easily.

Two beacons : The next example looks at the same problem in two dimensions with two beacons placed at $b_1 = (1.4, 1.4)$ and $b_2 = (-1.4, -1.4)$, both of width $(1.2, 1.2)$. The initial position of the robot is at $(-1.5, -0.5)$ which means that it is closer to b_2 than b_1. Both the beacons have the same observation characteristics as described before. The problem is to enter a goal region located at $(1.5, -1.5)$ of width $(0.2, 0.2)$ to claim a discounted reward. The optimal trajectory is different depending upon the initial position of the robot. The following two examples show different policies obtained from the discrete solver for varying number of states in the POMDP or computational time allowed. It is seen that in each case, a rough policy that goes through one of the light regions is obtained before improving it to a policy which obtains a larger expected reward.

- **Incrementality of SARSOP** Figure 4-7 shows simulated belief trajectories for this problem. The X-Y plane represents the mean of the conditional distribution while the Z axis depicts the norm of the variance. The robot thus starts with some high variance and has to reach the goal region at (1.5, -1.5) with low variance to obtain a reward of 1000. The green squares show light regions. As seen in the figure, the solver quickly finds a policy which goes through the light region alright but later improves to go through the closer light region. This example shows the incrementality of the SARSOP algorithm i.e., the discrete POMDP used in both the figures is the same.

- **Construction of discrete POMDP** In addition to an incremental solver, Algorithm 4.4 constructs discrete POMDPs incrementally. We test this by running SARSOP on two different POMDPs one with 75 states and the other with 150 sampled states. Figure 4-8 shows simulated belief trajectories for this problem. Note that the expected reward for the sparse POMDP is lower because it reaches the goal with a larger variance. This is because the set of α-vectors calculated on this POMDP is not refined enough to ensure that all belief trajectories go through the light region. On the other hand, with larger number of states, most of the belief trajectories go through the light region.

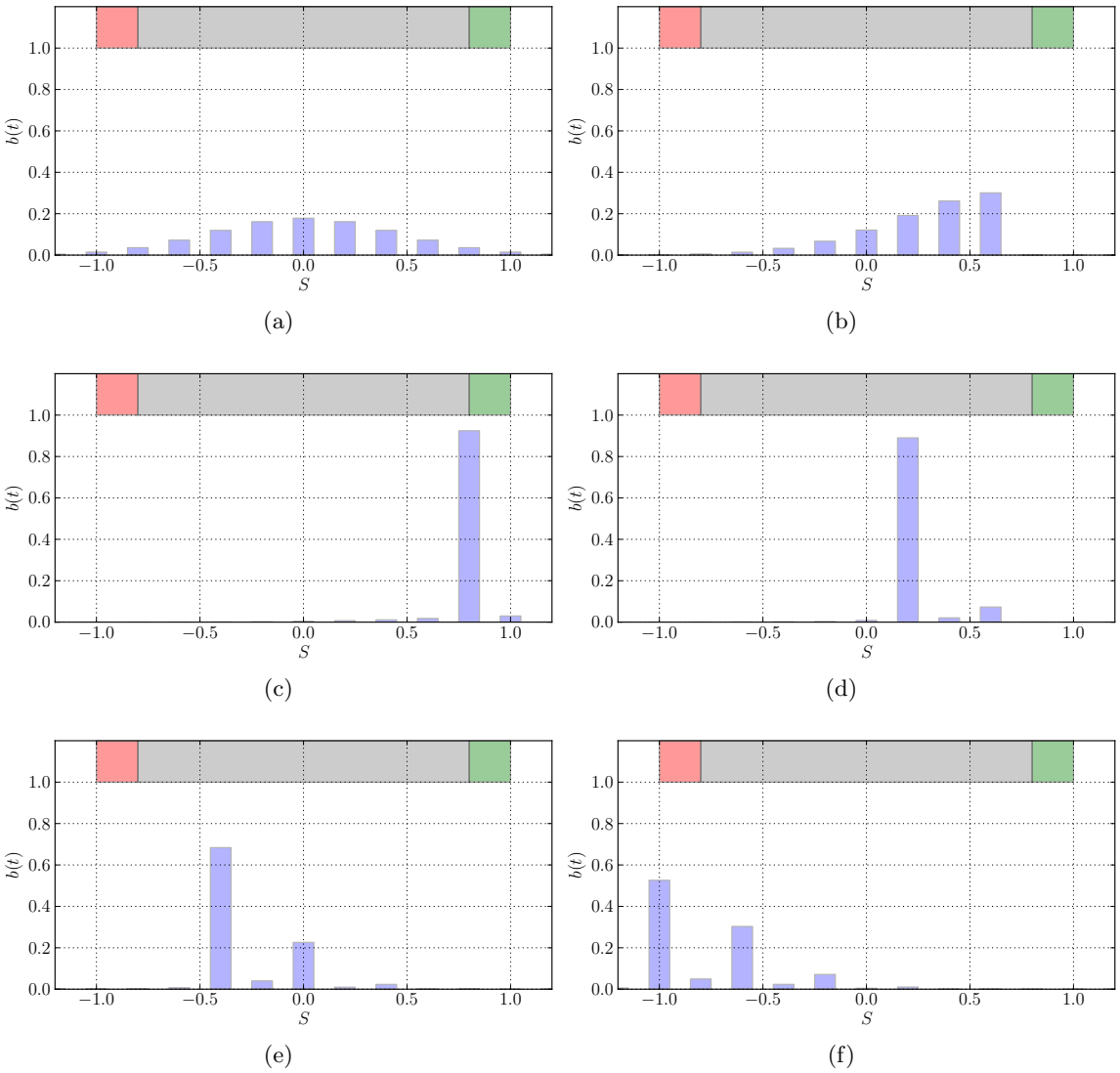
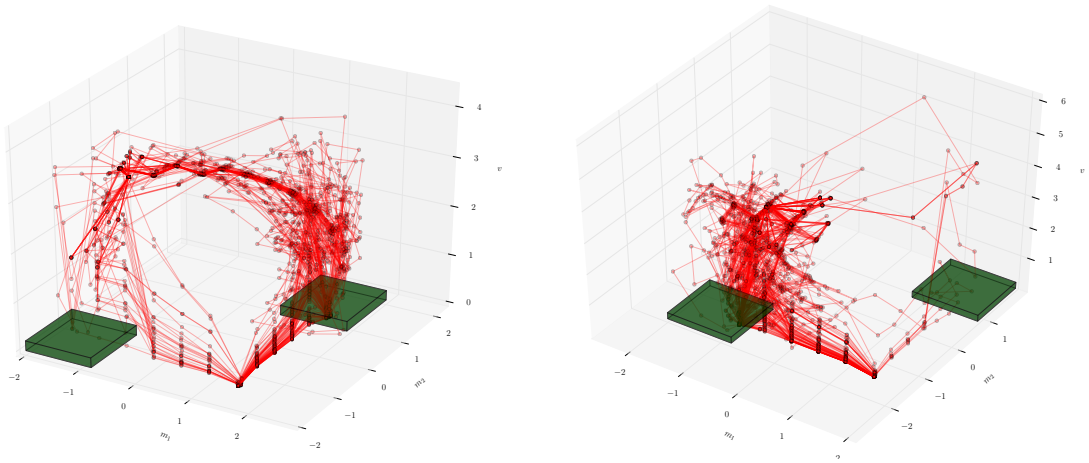
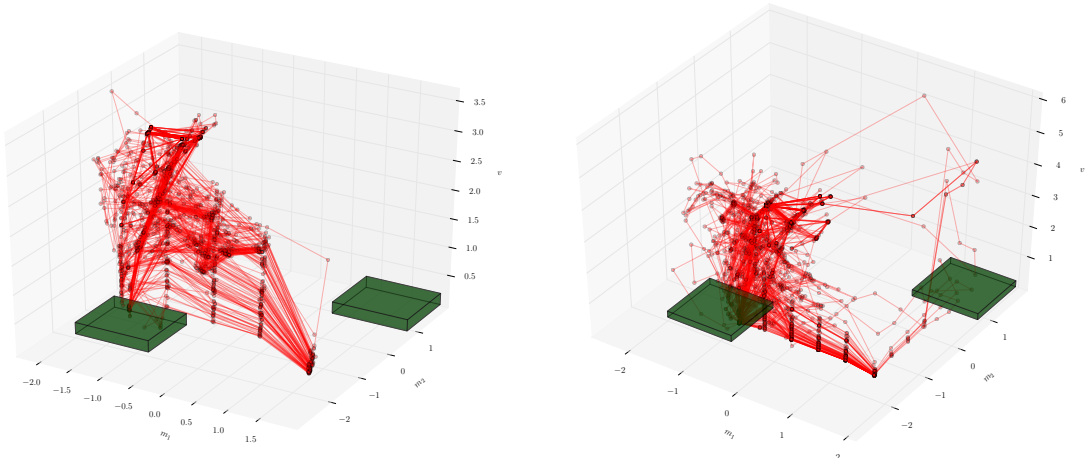


Figure 4-6: An example policy calculated on a POMDP of just 20 states for 1-dimensional Light-dark domain. Red denotes the goal region while the system has access to accurate observations in the green region. Blue rectangles denote the belief $b_n(t)$. These six figures show the belief state at 6 different instants of policy execution.



(a) After 100 secs (b) After 200 secs

Figure 4-7: Simulated belief trajectories for the light-dark domain with two beacons with the system starting from the same position for the same discrete POMDP having 150 states. Figure 4-7a is obtained after running the solver for 100 seconds giving an expected reward of 212.1 ± 99.7. If the incremental solver is run for 200 secs, the policy improves with the expected reward rising to 428.6 ± 55.2 as seen in Figure 4-7b.



(a) 75 states (b) 150 states

Figure 4-8: Incrementality of discrete POMDPs. Figure 4-8a shows simulated belief trajectories using a POMDP having 75 states with an expected reward of 377.9 ± 48.7. As more states are added to the POMDP, the expected reward rises to 428.6 ± 55.2 for 150 states. The belief trajectories for this are shown in Figure 4-8b.

Chapter 5

Conclusions

This thesis proposed algorithms for incremental estimation algorithms. The basic idea explored in this thesis is that, continuous-time stochastic estimation problems can be reduced into a number of discrete problems, each of which can be solved using simpler and computationally efficient algorithms. Classic results on Markov chain approximations were used to approximate the underlying stochastic systems. The primary features of this method are that we can derive approximate models using intuitive physical laws and do not need to depend upon hard-to-verify regularity assumptions on solutions of partial differential equations. The advantages of this are two-fold. Firstly, the resulting algorithms are very easy to implement in practice and secondly, these algorithms are completely probabilistic in nature. Even weak notions of convergence are thus enough to guarantee convergence of other quantities of interest such as conditional distributions, cost functions or control policies.

A major contribution of this thesis is that it uses ideas from asymptotically optimal motion planning algorithms to create Markov chain approximations via random sampling. This is a powerful approach because, as demonstrated in numerous algorithms like Probabilistic Road Maps (PRM) and Rapidly Expanding Random Trees (RRTs) or their optimal counterparts like PRM* and RRT*, random sampling algorithms can explore large state spaces very quickly to arrive at answers. There are two properties of these algorithms that the approaches in this thesis inherit, in particular, incrementality and anytime computability. Applications of this concept are particularly important in situations where we have a large amount of data, or if we have low computational resources. In this sense, the algorithms proposed in this thesis are such that they give an optimal answer with respect to available computational resources. They do not need to be tuned to work on platforms with different computational capacity, partly because of their anytime property and partly because of the fact that they have sound footing in inherently continuous formulations of the problem. The discretization of this general problem depends upon the amount of computation available *naturally*.

The first part of the thesis discusses state estimation problems. Markov chain approximations were used to construct Markov chains with a time component, i.e., roughly Markov chains on the space $[0, T] \times \mathcal{S}$ using both a grid based approach as well as random sampling. A simple Bayes' rule filter was then used to propagate the conditional density on this discrete Markov chain. Incremental refinement of the chain results in successively finer

discretizations of the time axis, whereby, the conditional density on this Markov chain converges to the conditional density of the optimal nonlinear filter. A similar approach was used to construct Markov chains for related problems like smoothing and trajectory decoding. It was shown that while the Markov chain approximation is necessary for the smoothing problem, a simpler version which uses only random sampling based methods can be used to obtain the solution of the decoding problem.

The second part of the thesis focused on control of partially observable processes. It follows the same basic idea as Chapter 2 of creating discrete approximations of a continuous time process. Roughly, it repeats the program of weak convergence of trajectories of Markov chains to solutions of stochastic differential equations. This case is technically more complicated because trajectories of a discrete POMDP are trajectories of conditional distributions (beliefs). These trajectories of beliefs converge to the corresponding trajectories of beliefs of the original continuous time stochastic control problem. However, correspondingly similar proofs for this case exist. We can prove that if the discrete POMDPs are created in a certain locally consistent fashion, the cost function as calculated along discrete trajectories, as well as the control policies calculated using them converges. This approach is applied to break down two example continuous time problems into conventional discrete POMDPs which are then solved using state-of-the-art point based solvers.

Future directions for perception algorithms mainly draw upon the idea that state-estimation is not the explicit objective of perception. The objective in most cases is control. This is an interesting idea because arguably, if we can isolate what elements of sensory information will affect the control actions that we choose, the computational complexity of all perception algorithms can be vastly improved. An example from vision will illustrate the point. The information obtained from a camera mounted on an autonomous vehicle is a stream of images, each of which is a vector, say, of length 640×480. The orientation or the distance of the camera from nearby objects can be calculated by analyzing this stream via stereo or optical flow algorithms. On the other hand, for a driver of a car turning along a road, just the estimate of his position with respect to the apex of the curve is enough information for control. The whole state estimate which is typically derived from complicated (and slow) algorithms is irrelevant to this particular task. In other words, we need a theory of observability for a particular task instead of reconstruction of the state-estimate. It is thus essential to (i) unify perception and control to create algorithms which grab only control relevant portions of information from a vast sensor array and (ii) understand what information is specific to control activities. Recent works like [YSDZ12] are exploring the former question while [Soa09] is a promising approach towards the latter problem.

Appendix A

A.1. Stochastic differential equations

Wiener Process : Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with $\mathcal{F}_t$ for $t \geq 0$ being the filtration defined on it. A process $w(t)$ is called a standard $\mathcal{F}_t$-Wiener process if it satisfies the following conditions.

1. $w(0) = 0$ with probability 1.
2. $w(t) \in \mathcal{F}_t$ and $\mathcal{F}(w(s) - w(t))$ for $s \geq t$ is independent of $\mathcal{F}_t$ for all $t \geq 0$.
3. $w(s) - w(t)$ is normal with mean zero and variance $s - t$.
4. Sample paths are continuous i.e., $w(t) \in C[0, \infty)$

A vector of mutually independent standard Wiener processes is called a vector-valued Wiener process. Note that $w(t)$ is a martingale and Markov with a quadratic variation $< w(t) >= t$.

Ito integral : For any $\mathcal{F}_t$ measurable function $f(t)$, we can define the Ito integral as

$$I_t(f) = \int_0^t f(s)dw(s) = \lim_{\Delta \to 0} \sum_{k=0}^{n-1} f(t_k)\ [w(t_{k+1}) - w(t_k)] + f(t_n)[w(t) - w(t_n)]$$

for any partition $0 \leq t_1 < t_2 < \ldots < t_n \leq t$ of $[0, t]$ with $\Delta = \sup_k |t_{k+1} - t_k|$. It has a number of nice properties like

- **Linearity :** $I_t(f_1 + f_2) = I_t(f_1) + I_t(f_2)$
- **Isometry :** $\mathbb{E}[(I_t(f))^2] = \int_0^t\ f(s)^2\ ds$

Ito's formula : Ito's formula is the analogous version of the chain-rule in calculus. Note that we use the notation

$$dx(s) = b(s)ds + \sigma(s)dw(s)$$

for the integral equation

$$x(t) = x(0) + \int_0^t b(s)ds + \int_0^t \sigma(s)dw(s) \tag{A.1}$$

where the first term is the ordinary Riemann integral and the second term is the Ito integral defined previously. For a continuous and differentiable function $f \in C^1(\mathbb{R})$ we define

$$df(x(s)) = \frac{\partial}{\partial x} f(x(s))\ dx(s) + \frac{1}{2}\ \frac{\partial^2}{\partial^2 x} f(x(s))\ (dx)^2$$

with $(dx)^2$ being calculated by rules of the form $ds^2 = 0$, $ds\ .\ dw = 0$ while quadratic variation of $w(t)$ gives $dw^2 = ds$. The vector version of this formula is obtained analogously.

Equation (A.1) denotes a stochastic differential equation. The solution of this equation is a process $x(t)$ which is $\mathcal{F}_t$ measurable and satisfies it. There are two ways in which the solutions of such equations can be said to exist and be unique.

- **Strong Existence :** Given a probabiltiy space $(\Omega, \mathcal{F}, \mathbb{P})$ and a filtration $\mathcal{F}_t$ along with an $\mathcal{F}_t$-Wiener process, if $x(0) \in \mathcal{F}_0$ is a measurable initial condition, then a $\mathcal{F}_t$-adapted process $x(t)$ exists that satisfies Equation (A.1) for all $t \geq 0$.
- **Weak Existence :** We say that Equation (A.1) has a weak sense unique solution if given any measure μ on $\mathbb{R}^k$, there exists a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a filtration $\mathcal{F}_t$ along with an $\mathcal{F}_t$-Wiener process and $\mathcal{F}_t$-adapted process $x(t)$ exists that satisfies it with $\mathbb{P}(x(0) \in \Gamma) = \mu(\Gamma)$.

The difference between these two notions is that strong existence requires that the filtration, Wiener process and the probabilty space be given in advance whereas weak solutions construct all the three elements along with the process $x(\cdot)$. Strong existence clearly implies weak existence.

- **Strong Uniqueness :** Two solutions $x_1(t)$ and $x_2(t)$ are strong sense unique if

$$\mathbb{P}(x_1(0) = x_2(0)) = 1 \implies \mathbb{P}(x_1(t) = x_2(t) \text{ for all } t \geq 0) = 1.$$

- **Weak Uniqueness :** Two weak sense solutions $(\Omega_k, \mathcal{F}_k, \mathbb{P}_k, \mathcal{F}_{k,t}, w_k(t), x_k(t))$ for $k = 1, 2$ are weak sense unique if the equality of distributions of $x_k(0) \in \mathbb{R}^k$ implies the equality of distributions of $x_k(t)$ on $C^k[0, \infty)$ under $\mathbb{P}_k$.

Strong uniqueness is called pathwise uniqueness whereas weak uniqueness is called uniquess of probability law. Proving convergence of numerical schemes related to stochastic differential equations only requires weak sense uniqueness of weak solutions.

A.2. Weak Convergence

Theorem A.1 (Portmanteau Theorem [Bil99]). *The following statements are equivalent.*

(i) $P^n \Rightarrow P$, *i.e.,* P^n *converges weakly to* P

(ii) $\int f dP^n \to \int f dP$ *for all bounded uniformly continuous functions* f.

(iii) $\limsup P^n(F) \leq P(F)$ *for any closed set* F.

(iv) $\liminf P^n(G) \geq P(G)$ *for any open set* G.

(v) $P^n(A) \to P(A)$ *for all sets* A *such that* $P(\partial A) = 0$.

Theorem A.2 (Mapping Theorem [Bil99]). *If* h *maps one metric space* S *into another* S' *and is a measurable function, if* D_h *is the set of all the points of discontinuity of* h *and if* $P(D_h) = 0$*, then* $P^n h^{-1} \Rightarrow P h^{-1}$.

Tightness of measures : A measure $\mathbb{P}$ is called tight if for any $\epsilon > 0$, there exists a compact set K such that $\mathbb{P}(K) > 1 - \epsilon$. It turns out that if the underlying space is complete and separable, then any probability measure is tight.
Topology of $C[0,1]$ **:** We will use the standard topology on the space of real-valued continuous functions with the metric defined as $\rho(x,y) = \sup_{t\in[0,1]} |x(t) - y(t)|$ Note that $\rho(x^n, x) \to 0$ only implies pointwise convergence. It can be shown that $C[0,1]$ with this metric is complete and separable [Rud76]. Thus any measure on $C[0,1]$ is tight.

Theorem A.3 (Arzela-Ascoli [Bil99]). *A relatively compact set of a metric space is a set* A *such that any sequence in* A *has a convergent subsequence. This celebrated theorem characterizes such sets i.e.,* A *is relatively compact subset of* $C[0,\infty)$ *if and only if*

- $\sup_{x\in A} |x(0)| < \infty$
- $\lim_{\delta\to 0} \sup_{x\in A} w_x(\delta) = 0$ *where* $w_x(\delta)$ *is the modulus of continuity given by* $w_x(\delta) = \sup_{|s-t|\leq\delta} |x(s) - x(t)|$ *for* $s, t \in [0,1]$.

Theorem A.4 (Prokhorov's Theorem [Bil99]). *A sequence of measures is called tight when there exists a compact set* K *such that* $P^n(K) > 1 - \epsilon$ *for all* n *for any* $\epsilon > 0$. *Prohorov's theorem claims that a sequence of measures* (P^n) *is tight, then it is relatively compact. The essense of the argument is that if every subsequence of* (P^n) *converges to some limit* P*, then the whole sequence in fact converges to* P*, i.e.,* $P^n \Rightarrow P$.

A converse of the above theorem is also true and it states that if the underlying space on which the measures are defined is complete and separable and if (P^n) *is relatively compact then it is tight.*

References

[ABC+08] F. Alabau-Boussouira, P. Cannarsa, et al. Control of Partial Differential Equations. *Encyclopedia of Complexity and Systems Science*, 2008. 59

[AE86] P. Antonelli and R. Elliott. The Zakai forms of the prediction and smoothing equations. *Information Theory, IEEE Transactions on*, 32(6):816–817, 1986. 44

[BC80] RW Brockett and JMC Clark. The geometry of the conditional density equation. *Analysis and optimization of stochastic systems*, pages 299–309, 1980. 60

[Ben81] VE Beneš. Exact finite-dimensional filters for certain diffusions with nonlinear drift. *Stochastics: An International Journal of Probability and Stochastic Processes*, 5(1-2):65–92, 1981. 35, 60

[Ber95] D.P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995. 64

[Bil99] Patrick Billingsley. *Convergence of probability measures*. Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York, second edition, 1999. A Wiley-Interscience Publication. 92, 93

[BP96] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1168–1175, 1996. 61

[BR11] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011. 61

[CE97] C.D. Charalambous and R.J. Elliott. Certain nonlinear partially observable stochastic optimal control problems with explicit control laws equivalent to LEQG/LQG problems. *Automatic Control, IEEE Transactions on*, 42(4):482–497, 1997. 60

[CKF12] P. Chaudhari, S. Karaman, and E. Frazzoli. Sampling-based algorithm for filtering using Makov chain approximations. In *IEEE Conference on Decision and Control (CDC) (submitted)*, 2012. 16

[CLZ97] A. Cassandra, M.L. Littman, and N.L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61. Morgan Kaufmann Publishers Inc., 1997. 61

[Cri06] D. Crisan. Particle filters in a continuous time framework. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 73–78, 2006. 35

[Dau05] F. Daum. Nonlinear filters: Beyond the Kalman filter. *Aerospace and Electronic Systems Magazine, IEEE*, 20(8):57–69, 2005. 35

[DC05] R. Doucet and O. Cappé. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE, 2005. 14, 35

[DDFG01] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001. 35

[DMM00] P. Del Moral and L. Miclo. Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non-linear filtering. *Seminaire de Probabilites XXXIV*, pages 1–145, 2000. 35

[DT03] A. Doucet and V.B. Tadić. Parameter estimation in general state-space models using particle methods. *Annals of the institute of Statistical Mathematics*, 55(2):409–422, 2003. 43

[FJ73] G.D. Forney Jr. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. 49

[Fox03] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research*, 22(12):985, 2003. 14, 35, 36

[Gog92] E.M. Goggin. Convergence of filters with applications to the Kalman-Bucy case. *Information Theory, IEEE Transactions on*, 38(3):1091–1100, 1992. 77

[Gog94] E.M. Goggin. Convergence in distribution of conditional expectations. *The Annals of Probability*, 22(2):1097–1114, 1994. 75, 77

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113. IET, 1993. 35

[HKF12] V. Huynh, S. Karaman, and E. Frazzoli. An incremental sampling-based algorithm for stochastic optimal control. In *IEEE Conference on Robotics and Automation*, 2012. 28

[HLR07] D. Hsu, W.S. Lee, and N. Rong. What makes some pomdp problems easy to approximate. *Advances in Neural Information Processing Systems (NIPS)*, page 28, 2007. 62, 63, 64, 65

[JKF11] J. Jeon, S. Karaman, and E. Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In *IEEE Conference on Decision and Control (CDC)*, 2011. 13

[JT12] Robert Platt Jr. and Russ Tedrake. Non-Gaussian belief space planning as a convex program. In *IEEE Conference on Robotics and Automation*, 2012. 86

[JU97] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, page 26. Citeseer, 1997. 34

[KB00] H. J. Kushner and A. S. Budhiraja. A Nonlinear Filtering Algorithm Based on an Approximation of the Conditional Distribution. *IEEE Transactions on Automatic Control*, 45(3):580–585, 2000. 41

[KD01] H. J. Kushner and P. Dupuis. *Numerical methods for stochastic control problems in continuous time*. Springer Verlag, 2001. 20, 22, 27, 36, 37, 74, 80, 81, 82

[KF11] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011. 13, 25, 28, 51, 53

[KJL00] J.J. Kuffner Jr and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000. 61

[KL98] L Kavraki and J C Latombe. Probabilistic Roadmaps for Robot Path Planning. 1998. 12, 61

[Kus65] H.J. Kushner. On the stochastic maximum principle: Fixed time of control. *Journal of Mathematical Analysis and Applications*, 11:78–92, 1965. 59

[Kus67] H. J. Kushner. Dynamical equations for optimal nonlinear filtering. *Journal of Differential Equations*, 3:179–190, 1967. 37

[Kus77] H. Kushner. Probability methods for approximations in stochastic control and for elliptic equations, acad. *Press, New York*, 1977. 34, 35

[Kus08] H.J. Kushner. Numerical approximations to optimal nonlinear filters. 2008. 38, 39, 40

[LaV98] S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. (98-11), 1998. 61

[LCK96] M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Efficient dynamic-programming updates in partially observable markov decision processes. 1996. 61

[LHT+08] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008. 11, 12

[LKJ01] S.M. LaValle and J.J. Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. 12

[LV04] T. Lyons and N. Victoir. Cubature on wiener space. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2041):169, 2004.

[Mon82] G.E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, pages 1–16, 1982. 61

[Mor66] RE Mortensen. Stochastic Optimal Control with Noisy Observations. *International Journal of Control*, 4(5):455–464, 1966. 59

[Øks03] Bernt Karsten Øksendal. *Stochastic differential equations*. an introduction with applications. Springer Verlag, 2003. 36, 37

[PGT03] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1025–1032. Citeseer, 2003. 62

[PKW+11] A. Perez, S. Karaman, M. Walter, A. Shkolnik, E. Frazzoli, and S. Teller. Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011. 13

[PPK+12] A. Perez, R. Platt, G.D. Konidaris, L.P. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2012. 61, 84, 86

[PR09] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009. 61

[PS99] M.K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, pages 590–599, 1999. 14, 35

[PT87] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, pages 441–450, 1987. 64

[Rab89] L.R. Rabiner. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 46

[Rei79] J.H. Reif. Complexity of the mover's problem and generalizations extended abstract. In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pages 421–427, 1979. 12

[Ris96] H. Risken. *The Fokker-Planck equation: Methods of solution and applications*, volume 18. Springer Verlag, 1996. 28

[Rud76] W. Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1976. 93

[Sam95] H. Samet. Spatial data structures. *Modern Database Systems, The Object Model, Interoperability and Beyond*, pages 361–385, 1995. 25

[Soa09] S. Soatto. Actionable information in vision. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2138–2145. IEEE, 2009. 90

[SS73] R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, pages 1071–1088, 1973. 60, 61

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[TWA+10] S. Teller, M.R. Walter, M. Antone, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J.P. How, A.S. Huang, et al. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 526–533. IEEE, 2010. 12

[YSDZ12] Zhao Yi, Stefano Soatto, Maneesh Dewan, and Yiqiang Zhan. Information forests. *CoRR*, abs/1202.1523, 2012. 90

[ZD87] O. Zeitouni and A. Dembo. A maximum a posteriori estimator for trajectories of diffusion processes. *Stochastics: An International Journal of Probability and Stochastic Processes*, 20(3):221–246, 1987. 50, 51

[ZD88] O. Zeitouni and A. Dembo. An existence theorem and some properties of maximum a posteriori estimators of trajectories of diffusions. *Stochastics: An International Journal of Probability and Stochastic Processes*, 23(2):197–218, 1988. 50, 51

[ZL96] N.L. Zhang and W. Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical report, Citeseer, 1996. 61