

ABSTRACT

Title of dissertation: **PARTICLE FILTERING FOR
STOCHASTIC CONTROL AND
GLOBAL OPTIMIZATION**

Enlu Zhou, Doctor of Philosophy, 2009

Dissertation directed by: **Professor Steven I. Marcus**
Department of Electrical and Computer Engineering

Professor Michael C. Fu
Department of Decision, Operations,
and Information Technologies

This thesis explores new algorithms and results in stochastic control and global optimization through the use of particle filtering. Stochastic control and global optimization are two areas that have many applications but are often difficult to solve.

In stochastic control, an important class of problems, namely, partially observable Markov decision processes (POMDPs), provides an ideal paradigm to model discrete-time sequential decision making under uncertainty and partial observation. However, POMDPs usually do not admit analytical solutions, and are computationally very expensive to solve most of the time. While many efficient numerical algorithms have been developed for finite-state POMDPs, there are only a few proposed for continuous-state POMDPs, and even more sparse are relevant analytical results regarding convergence and error bounds. From the modeling viewpoint, many ap-

plication problems are modeled more naturally by continuous-state POMDPs rather than finite-state POMDPs. Therefore, one part of the thesis is devoted to developing a new efficient algorithm for continuous-state POMDPs and studying the performance of the algorithm both analytically and numerically. Based on the idea of density projection with particle filtering, the proposed algorithm reduces the infinite-dimensional problem to a finite-low-dimensional one, and also has the flexibility and scalability for better approximation if given more computational power. Error bounds are proved for the algorithm, and numerical experiments are carried out on an inventory control problem.

In global optimization, many problems are very difficult to solve due to the presence of multiple local optima or badly scaled objective functions. Many approximate solutions methods have been developed and studied. Among them, a recent class of simulation-based methods share the common characteristic of repeatedly drawing candidate solutions from an intermediate probability distribution and then updating the distribution using these candidate solutions, until the probability distribution becomes concentrated on the optimal solution. The efficiency and accuracy of these algorithms depend very much on the choice of the intermediate probability distributions and the updating schemes. Using a novel interpretation of particle filtering, these algorithms are unified under one framework, and hence, many new insights are revealed. By better understanding these existing algorithms, the framework also holds the promise for developing new improved algorithms. Some directions for new improved algorithms are proposed, and numerical experiments are carried out on a few benchmark problems.

PARTICLE FILTERING FOR
STOCHASTIC CONTROL AND GLOBAL OPTIMIZATION

by

Enlu Zhou

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Advisory Committee:

Professor Steven I. Marcus, Chair/Advisor

Professor Michael C. Fu, Co-Chair/Co-Advisor

Professor P. S. Krishnaprasad

Professor André L. Tits

Professor S. Raghavan

© Copyright by
Enlu Zhou
2009

Dedication

In memory of my grandfather Mr. Mengquan Yu.

Acknowledgments

First of all, I would like to express my sincerest gratitude to my advisors Professor Steven I. Marcus and Professor Michael C. Fu. They have guided me in research and in career, provided me with tremendous support, and always had faith in me. I regard them as my role models for my career life, because of their dedication to the quality and integrity of research, love and patience for students, and being wonderful human beings. I have become a better person because of the five years of interaction with them. I will always cherish my experience as a graduate student under their supervision.

I would also like to thank Professor P. S. Krishnaprasad. I took five graduate courses taught by him, and often went to ask him questions or discuss problems. He is also the first person that introduced me to the field of particle filtering. His lectures are insightful, inspiring, and challenging. I have been greatly influenced by his great passion in research and pursuit of new knowledge.

I also want to thank Professor André L. Tits. He is not only a great teacher but also a good friend. I have learned a lot from him in class, and especially benefited from his technical rigorousness. I also had a lot fun with him in skiing, playing ping pong, and chatting at coffee hours.

I cannot express enough thanks to my mother Zhilin Yu and father Huaishen Zhou, who always give me unconditional love and support, and are the source of my courage in hard times. I also want to thank Peng Qiu, for the continuous happy time spent together and the discrete times listening to my complaints.

Table of Contents

List of Tables	vi
List of Figures	vii
List of Abbreviations	viii
1 Introduction	1
1.1 Particle Filtering	1
1.2 Partially Observable Markov Decision Processes	4
1.3 Contributions	7
1.4 Outline	10
2 Nonlinear Filtering	12
2.1 Extended Kalman Filter	13
2.2 Weighted Extended Kalman Filter	15
2.3 Particle Filtering	18
2.4 Fading Memory Particle Filter	23
3 Stochastic Control	32
3.1 Markov Decision Processes	32
3.1.1 Value Iteration	34
3.1.2 Policy Iteration	36
3.1.3 Simulation-Based Methods	38
3.2 Partially Observable Markov Decision Processes	38
3.2.1 Belief MDPs	40
3.2.2 Finite-State vs. Continuous-State	42
4 Solving Continuous-State POMDPs	44
4.1 Related Work and Motivation	44
4.2 Dimension Reduction	47
4.2.1 Density Projection	47
4.2.2 Projected Belief MDP	49
4.3 Projection Particle Filtering	51
4.4 Analysis of Projected Belief MDP	55
4.4.1 Main Idea	55
4.4.2 Error Bound	56
4.5 Analysis of Projection Particle Filtering	61
4.5.1 Notations	62
4.5.2 Main Idea	63
4.5.3 Error Bound	65
4.6 Validation of Assumptions	74
4.7 Numerical Experiments	75
4.7.1 Scalability and Computational Issues	75

4.7.2	Simulation Results	77
4.8	Conclusions	83
5	Particle Filtering Framework for Optimization	88
5.1	Related Work and Motivation	88
5.2	Filtering for Optimization	91
5.3	Particle Filtering Framework for Optimization	94
5.4	Interpretation of EDAS, CE, MRAS	97
5.4.1	Estimation of Distribution Algorithms	98
5.4.2	Cross Entropy Method	100
5.4.3	Model Reference Adaptive Search	103
5.5	Implication for New Algorithms	106
5.5.1	Balancing Exploration and Exploitation	107
5.5.2	Combining Global Search with Local Search	107
5.5.3	Overcoming Premature Convergence	108
5.6	Numerical Experiments	109
5.7	Conclusions and Future Research	115
6	Conclusions and Future Research	116
6.1	Conclusions	116
6.2	Future Research	118
	Bibliography	121

List of Tables

2.1	Performances under different real system parameters ρ_r , with model system parameter $\rho_m = 0.5$, and system noise $u_k \sim \Gamma(3, 1)$. Each entry shows the mean and standard error (in parentheses) of the MSEs based on 100 independent runs.	29
2.2	Performances under different system noises, with system parameter $\rho_r = \rho_m = 0.5$. Each entry shows the mean and standard error (in parentheses) of the MSEs based on 100 independent runs.	29
4.1	Notations of Different Conditional Distributions	65
4.2	Optimal average cost estimates for the inventory control problem using different methods. Each entry represents the average cost of a run of horizon 10^5	85
4.3	Optimal discounted cost estimate for the inventory control problem using different methods. Each entry represents the discounted cost (mean, standard error in parentheses) based on 1000 independent runs of horizon 40.	86
4.4	Continue Table 4.3	87
5.1	Average performance of CEA and CE on some benchmark problems. Each entry presents the mean of $H(x^*)$ with standard error in parentheses, based on 100 independent runs.	113
5.2	Average performance of CEA with different parameter values of α and β on the Rosenbrock function. Each entry presents the mean of $H(x^*)$ with standard error in parentheses, based on 100 independent runs.	113

List of Figures

2.1	A graphic illustration of Table 2.1: Performances (mean and standard error of MSEs based on 100 independent runs) under different values of the real system parameter ρ_r	30
2.2	A graphic illustration of Table 2.2: Performances (mean and standard error of MSEs based on 100 independent runs) under different system noises.	30
2.3	A typical run of the estimated states tracking the true state, when the system noise $u_k \sim \Gamma(3, 2)$, and the system parameter $\rho_r = \rho_m = 0.5$	31
2.4	A typical run of the estimated states tracking the true state when the system model is inaccurate. The real system parameter $\rho_r = 0.6$, the model system parameter $\rho_m = 0.5$, and the system noise $u_k \sim \Gamma(3, 2)$	31
4.1	Our algorithm: actions taken for different inventory levels under different observation noise variances.	82
5.1	At the first three iterations, the conditional density b_k becomes more and more concentrated on the optimal solution.	94
5.2	Some benchmark problems in two dimensions.	112
5.3	Average performance of CEA and CE on some benchmark problems.	114

List of Abbreviations

CE	Cross Entropy Method
DP	Dynamic Programming
EDA	Estimation of Distribution Algorithm
EKF	Extended Kalman Filter
MDP	Markov Decision Process
MRAS	Model Reference Adaptive Search
KF	Kalman Filter
KL	Kullback-Leibler
PF	Particle Filter
POMDP	Partially Observable Markov Decision Process

Chapter 1

Introduction

Stochastic control and global optimization are fields that share two characteristics: (i) their models can be used to formulate problems in many applications areas; (ii) algorithms for solving problems in these fields are usually intractable and difficult to analyze. To attack difficult problems of a size that are found in most applications requires significant new methodologies. This dissertation attempts to solve problems in stochastic control and global optimization through the use of particle filtering. This chapter gives a brief introduction to particle filtering and an important class of problems in stochastic control, namely, the partially observable Markov decision processes (POMDPs).

1.1 Particle Filtering

The goal of filtering is to estimate the unobserved states of a dynamic system given a noisy observation process of the states. The classic problem is to estimate the conditional density of the current state given the history of observations [24]. The conditional density usually evolves according to some equation, for instance, in the case of a diffusion process the normalized Kushner-Stratonovich equation [52] [84] or the unnormalized Zakai equation [92]. However, these equations usually do not admit an analytical form of the solution except in some special cases, such as

linear Gaussian systems and finite state space hidden Markov models. For more general systems, many approximation filtering methods have been developed.

One approach of approximation is to modify the system in such a way that exact filters can be applied to the modified system. For example, a well-known method is the Extended Kalman filter (EKF) (pp. 182-190, [31]) for nonlinear/non-Gaussian systems. EKF linearizes the system equation and observation equation, and approximates the system noise and observation noise by their first two moments, i.e., Gaussian random variables. Hence, the approximated linear Gaussian system can be filtered using the standard Kalman filter [46] [47]. EKF is computationally efficient, but the convergence of the filter is not guaranteed and the performance could be poor in many situations. Another method is to discretize the state space and transform the system to a finite state hidden Markov model. The disadvantage of this method is that the number of grids has to be sufficiently large to obtain a good approximation and the number of grids increases dramatically as the dimension of the state space increases.

A recent and powerful method for nonlinear filtering is particle filtering. It is a class of Monte Carlo simulation-based filtering methods for nonlinear/non-Gaussian systems, a setting where traditional methods often lead to computational intractability. Particle filtering was first introduced by Gordon et al. [34], and is also called bootstrap filtering [34], sampling importance resampling [4], the condensation algorithm [42], and sequential Monte-Carlo method [4]. A good tutorial can be found in [4], a good survey of the field and recent trends can be found in [21], and more details can be found in the book [29].

The idea of particle filtering is to approximate the conditional density by a finite number of particles/samples and let these particles propagate in a certain way to mimic the evolution of the conditional density. The approximated conditional density converges to the true conditional density in a certain sense as the number of particles increases to infinity [23], [54].

The plain particle filter consists of three steps at each time/iteration: importance sampling, weighting, and resampling. In the importance sampling step, i.i.d. particles are sampled from a importance density. In the weighting step, particles are weighted according to the ratio of the conditional density to the importance density and also according to the Bayes' rule using the new observation of the current state. These weighted particles represent a discrete distribution with support points equal to the locations of the particles and the associated probabilities equal to the weights of the particles. This discrete distribution is an approximation to the true conditional distribution. In the last step, new particles are resampled from the old particles according to their weights. These new particles go through the same steps at the next time/iteration.

The importance distribution is often chosen as the distribution of the current state given the previous state. The benefit of this choice is twofold. First, this importance distribution is easy to sample, since it is equivalent to propagating the particles through the system equation. Thus, the importance sampling step is also called propagation or prediction step in this case. Second, the weight of each particle is proportional to the likelihood of the current observation given that particle, and hence is very easy to calculate. However, this is not the optimal choice of importance

distribution. The simulation can be very inefficient if the importance distribution is very different from the target distribution. For instance, if the conditional distribution is peaky and the importance distribution is flat, it is very likely that most of the particles sampled from the importance distribution lie in an area of probability close to zero according to the conditional distribution. Therefore, many improved particle filters have been proposed based on a better choice of the importance distribution, such as the extended Kalman particle filter [25], the auxiliary particle filter [68], and the unscented particle filter [88].

The plain particle filter often suffers from the problem of sample impoverishment or loss of diversity, meaning that all particles collapse to a small number of particles within a few iterations. Since new particles are resampled from the old particles according to their weights/probabilities, old particles with large weights/probabilities will have more copies in the next iteration. The problem of sample impoverishment is especially severe when the system noise is small, because the copies of the same particle cannot be dispersed far away enough through propagation. Therefore, improving the resampling step is also a goal of many improved particle filters, such as the regularized particle filter [65], and the particle filter with MCMC steps [3].

1.2 Partially Observable Markov Decision Processes

POMDPs provide a powerful paradigm for modeling discrete-time optimal decision making under uncertainty and partial observation. It has been used to

model application problems arising in the areas such as manufacturing systems, artificial intelligence, financial engineering, and operations research. A POMDP model consists of a time-indexed state equation that models the system dynamics, a time-indexed observation equation that models the observation or measurement of the state, and an objective function that models the cost (or reward) that needs to be minimized (or maximized). A decision maker interacts with the environment over a finite or infinite time horizon that is divided into stages or periods. At each stage, the decision maker receives a partial observation of the current state, takes an action based on this piece of information along with the history of observations and actions, and then transitions to a new state probabilistically at the next stage. The action taken incurs a one-step cost (or reward). The objective is to minimize (or maximize) a cost (or reward) function, where the one-step costs (or rewards) are accrued in each stage.

The difference between a POMDP and a Markov Decision Process (MDP) is in the observation of the state. The state can be fully recovered from the observation in an MDP, whereas the observation only provides partial information for the state in a POMDP. A POMDP can be transformed to a so-called belief MDP, which is an MDP whose states are conditional probability distributions of the true states of the POMDP and are called belief states. This transformation allows us to utilize the existing various techniques for solving MDPs. However, it also creates new difficulties. The first difficulty is how to accurately and efficiently estimate the distribution of the states, which is the goal of a filtering problem. The second difficulty is that although it is an MDP problem, the state space is generally continuous and may

have infinite dimensionality. The reason is explained as follows.

Consider a POMDP problem with a finite state space. The estimation of the state is a discrete probability distribution, which sits in a finite-dimensional probability simplex. Hence, the corresponding belief MDP has a finite-dimensional continuous state space, which suffers from the curse of dimensionality of the continuous state space of the MDP. Now consider a POMDP problem that has a continuous state space. The probability distribution of the state sits in an infinite-dimensional space of continuous probability distributions. Hence, the corresponding belief MDP has a infinite-dimensional continuous state space that makes the problem even more difficult to solve. Therefore, efficient numerical solution of POMDPs with large or continuous state spaces is a challenging research area.

The past research on numerical solutions of POMDPs is mostly focused on finite-state problems. For finite-state POMDPs, it is proved that the value function is a piecewise linear convex function after a finite number of iterations, provided that the one-step cost function is piecewise linear and convex [81]. This feature has been exploited in various exact and approximate algorithms such as those found in [82], [81], [83], [57], and [36]. The algorithm in [81] and the Witness algorithm in [57] carry out exact value iterations by finding and updating the minimum set of linear functions that determine the value function at each iteration for a finite-horizon problem. Because the number of such linear functions grow exponentially with the number of iterations, these algorithms are computationally very expensive and are limited to very simple problems in practice. Howard [39] and Sondik [83] use policy iteration to solve the infinite-horizon discounted-cost problems. Hauskrecht

[36] summarizes several value function approximation methods in a nice framework of modifying the value iteration equation by changing the order of summations and maximization, including approximation with fully observable MDP, approximation with Q-functions, fast informed bound method, and approximation with unobservable MDP. [36] also summarizes many grid-based methods with various techniques of choosing the set of grids and approximating the value function.

Despite the abundance of algorithms for finite-state POMDPs, the aforementioned infinite-dimensionality of continuous-state POMDPs suggests that simple generalizations of many of the discrete-state algorithms to continuous-state models are not appropriate or applicable. Therefore, some researchers have been motivated to look for efficient algorithms for continuous-state POMDPs [87] [69] [75] [17]. However, compared to finite-state POMDPs, the research on continuous-state POMDPs is more recent and still sparse. Therefore, it is a field worth exploring.

1.3 Contributions

My doctoral research has yielded new solution methods to partially observable Markov decision processes and global optimization through the use of particle filtering. To the best of our knowledge, application of particle filtering to POMDPs is relatively sparse, and it has never before been applied to the field of global optimization.

The first line of our research aims to establish a framework for solving large or continuous-state POMDPs. Most of the existing approximate solutions for continuous-

state POMDPs involve some kind of dimension reduction of the belief space, i.e., reducing the infinite dimensionality to a finite (low) dimensionality. Dimension reduction is a very broad idea, and it is worth studying how to do dimension reduction in an efficient and meaningful way. On the one hand, we want to reduce the dimensionality as much as possible so that the computational demand can be lowered. On the other hand, we do not want to lose too much information during dimension reduction so that the approximation is useful. Moreover, we would like to have a mechanism that allows us to control the tradeoff between complexity and accuracy: we can increase the accuracy of the solution by increasing the computational power, and vice versa. These considerations have motivated our research to investigate a new dimension reduction technique to develop efficient numerical solutions for large/continuous-state POMDPs. Based on the idea of density projection with particle filtering, we have developed a theoretically sound method that effectively reduces the dimension of the belief state and has the flexibility to represent arbitrary belief states, such as multimodal or heavy-tailed distributions. We have proved rigorous convergence results and error bounds of the algorithm. We have also applied the approach to and obtained good numerical results on an inventory control problem and portfolio optimization problems in financial engineering. The development and analysis of the approach appear in our papers [95] [96]. Applications of the approach to financial engineering are to appear in our working paper [94].

As a second line of our research, we have proposed a filtering approach to optimization, and in particular, have developed a framework based on particle filtering.

Global optimization problems can be extremely difficult to solve, due to the presence of multiple local optimal solutions in the general setting. A class of simulation-based methods for global optimization has been introduced recently, which includes the estimation of distribution algorithms (EDAs), the cross-entropy (CE) method, and model reference adaptive search (MRAS). In these algorithms, new solutions are generated from an intermediate probability distribution that is updated or induced from the previously generated solutions. Algorithms that fall in this category differ in the choice and updating of the intermediate probability distribution, which plays a key role in determining the effectiveness of the algorithm. This has motivated us to look for a unifying and systematic approach to such simulation-based methods for optimization. We have introduced a framework based on particle filtering that unifies EDAs, the CE method and MRAS, as well as combining the simulation-based global search with the gradient-based local search in a nice way. This flexible framework holds the promise of generating new improved algorithms by incorporating many of the vast array of techniques that have been developed to improve particle filters, and other recent results in nonlinear filtering. This framework unifies many recent simulation-based algorithms, and combines simulation-based global search with gradient-based local search in a nice way. More importantly, the framework holds the promise of generating new improved algorithms by incorporating many of the vast array of techniques that have been developed to improve particle filters, and other recent results in nonlinear filtering. We are currently developing and testing new algorithms under this framework, and analyzing convergence properties and error bounds of the framework. Preliminary results of this work appear in our

paper [97]. A more complete and developed work appears in our paper [96].

Besides the two main lines of research, we have also developed a fading memory particle filter. It is an improved particle filter in situations where the system models are inaccurate and simulation is insufficient.

1.4 Outline

The rest of the dissertation is organized as follows.

Chapter 2 provides the necessary background and literature review on nonlinear filtering, and proposes a fading memory particle filter. We describe the problem formulation of nonlinear filtering, and describe in details some approximate nonlinear filters, including the extended Kalman filter, the weighted extended Kalman filter, and particle filtering in general. In the end of the chapter, we present the derivation of a fading memory particle filter and results from numerical experiments.

Chapter 3 provides the necessary background and literature review on stochastic control. In particular, we focus on Markov decision processes and partially observable Markov decision processes, and describe the problem formulation of each. For MDPs, we describe in details the value iteration method and policy iteration method, and briefly describe simulation-based methods. For POMDPs, we describe the transformation to belief MDPs, and discuss numerical complexity of finite-state vs. continuous-state POMDPs.

Chapter 4 presents our work on a new efficient numerical method for solving continuous-state POMDPs. Section 4.1 reviews related work and explains our moti-

vation for this work; section 4.2 describes the density projection technique, and uses it to develop the projected belief MDP; section 4.3 develops the projection particle filter; section 4.4 computes error bounds for the projected belief MDP; section 4.5 computes an error bound for the projection particle filter; section 4.6 presents a validation of the assumptions used in the proof of error bounds; section 4.7 discusses scalability and computational issues of the method, and applies the method to a simulation example of an inventory control problem; section 4.8 concludes and discusses the work.

Chapter 5 presents a particle filtering framework for simulation-based optimization algorithms. Section 5.1 reviews related work and explains our motivation for this work; section 5.2 transforms a global optimization problem to a filtering problem; section 5.3 applies particle filtering to the transformed filtering problem and develops a general framework for simulation-based optimization algorithms; section 5.4 uses the framework to interpret some existing algorithms and reveals some new insights; section 5.5 discusses the directions for developing new improved algorithms under this framework; section 5.6 presents numerical results of a new improved algorithm in comparison with the cross-entropy method on some benchmark problems; section 5.7 concludes our current work and discusses some future research direction.

Chapter 6 concludes the dissertation and outlines some future research.

Chapter 2

Nonlinear Filtering

The nonlinear filtering problem [24] [61] [45] involves the estimation of a stochastic process x (called the state process) which cannot be observed directly. Information concerning x is obtained from observations of a related process y (the observation process). The objective is the computation, for each t , of least-square estimates of functions of the state x_t given the “past” observations $\{y_s, 0 \leq s \leq t\}$, i.e., to compute quantities of the form $E[\phi(x_t)|y_s, 0 \leq s \leq t]$, or perhaps to compute the entire conditional distribution of the state x_t given $\{y_s, 0 \leq s \leq t\}$. When the observations are being received sequentially, it is also desired that this computation be done recursively. Except in some special cases such as the linear Gaussian case, there do not exist statistics of the conditional distribution that can be computed recursively with finite-dimensional filters [16] [63] [62] [40] [90]. Hence, there have been many attempts to develop recursive finite dimensional suboptimal filters, and significant effort has been dedicated to developing numerical methods for solving nonlinear filtering problems (see [19] for a recent survey).

In this thesis, we focus on a discrete-time model

$$\begin{aligned}x_k &= f(x_{k-1}, u_{k-1}), \quad k = 1, 2, \dots, \\y_k &= h(x_k, v_k), \quad k = 0, 1, \dots,\end{aligned}\tag{2.1}$$

where for all k , $x_k \in R^{n_x}$ is the state, $y_k \in R^{n_y}$ is the observation, the random

disturbances $\{u_k\} \in R^{n_u}$ and $\{v_k\} \in R^{n_v}$ are sequences of i.i.d. continuous random vectors, and n_x , n_y , n_u , and n_v are the dimensions of x_k , y_k , u_k , and v_k , respectively. Assume that $\{u_k\}$ and $\{v_k\}$ are independent of each other, and independent of x_0 , which follows a distribution p_0 . The goal of filtering is to estimate the conditional density

$$p(x_k|y_{0:k}), \quad k = 0, 1, \dots, \quad (2.2)$$

where $y_{0:k} = \{y_0, \dots, y_k\}$, all the observations from time 0 to k .

2.1 Extended Kalman Filter

In (2.1), if f and g are linear functions in x , u and v , u and v are Gaussian noises, and the initial condition x_0 is Gaussian, then the conditional density $p(x_k|y_{0:k})$ is Gaussian for all time k , and there exists a finite-dimensional optimal filter, namely, the Kalman filter (KF) [46] [47].

If (2.1) takes the form as

$$\begin{aligned} x_k &= f(x_{k-1}) + u_{k-1}, \quad k = 1, 2, \dots, \\ y_k &= h(x_k) + v_k, \quad k = 0, 1, \dots, \end{aligned} \quad (2.3)$$

where f and h are nonlinear functions, then the Kalman filter can be applied to (2.3) by linearizing the system equations, and approximating the system noise and observation noise as Gaussian noises. This approach is referred to as the extended Kalman filter (EKF) [31] [1] [2]. More specifically, the true conditional density $p(x_k|y_{0:k})$ is approximated by a Gaussian density with mean $\bar{x}_{k|k}$ and covariance

$P_{k|k}$ that are defined as

$$\bar{x}_{k|k} \triangleq E[x_k|y_{0:k}], \quad P_{k|k} \triangleq E[(x_k - \bar{x}_{k|k})(x_k - \bar{x}_{k|k})^T|y_{0:k}].$$

The estimates $\bar{x}_{k|k}$ and $P_{k|k}$ are often called the posterior mean and covariance of the state x_k . Similarly, the predicted mean and covariance at time k are defined as

$$\bar{x}_{k|k-1} \triangleq E[x_k|y_{0:k-1}], \quad P_{k|k-1} \triangleq E[(x_k - \bar{x}_{k|k-1})(x_k - \bar{x}_{k|k-1})^T|y_{0:k-1}].$$

To linearize the system (2.3), define

$$F_k = \left. \frac{\partial f(x)}{\partial x} \right|_{x=\bar{x}_{k|k}}, \quad H_k = \left. \frac{\partial h(x)}{\partial x} \right|_{x=\bar{x}_{k|k-1}}.$$

Without loss of generality, we assume that u_k and v_k are zero mean. Let Q_k and R_k denote the covariance matrices of u_k and v_k , respectively. Then the linearized and Gaussianized system of (2.3) is

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + u_{k-1}, \\ y_k &= H_k x_k + v_k, \end{aligned} \tag{2.4}$$

where $u_{k-1} \sim N(0, Q_{k-1})$ and $v_k \sim N(0, R_k)$. For (2.4), if the initial condition x_0 follows a Gaussian distribution $N(\bar{x}_0, P_0)$, the conditional density is Gaussian for every time k , and the optimal filter is the Kalman filter, which recursively propagates the predicted mean $\bar{x}_{k|k-1}$ and the predicted covariance matrix $P_{k|k-1}$, and updates the posterior mean $\bar{x}_{k|k}$ and the posterior covariance matrix $P_{k|k}$ using the new observation y_k . Mathematically, the Kalman filter consists of the following recursive

equations

$$\begin{aligned}
\bar{x}_{k|k-1} &= F_{k-1}\bar{x}_{k-1|k-1}, \\
P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1}, \\
K_k &= P_{k|k-1}H_k^T(R_k + H_kP_{k|k-1}H_k^T)^{-1}, \\
\bar{x}_{k|k} &= \bar{x}_{k|k-1} + K_k(y_k - H_k\bar{x}_{k|k-1}), \\
P_{k|k} &= P_{k|k-1} - K_kH_kP_{k|k-1},
\end{aligned} \tag{2.5}$$

with initialization $\bar{x}_{0|-1} = \bar{x}_0$, $P_{0|-1} = P_0$. The EKF extends the KF to the nonlinear system (2.3) as follows:

$$\begin{aligned}
\bar{x}_{k|k-1} &= f(\bar{x}_{k-1|k-1}), \\
P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1}, \\
K_k &= P_{k|k-1}H_k^T(R_k + H_kP_{k|k-1}H_k^T)^{-1}, \\
\bar{x}_{k|k} &= \bar{x}_{k|k-1} + K_k(y_k - h(\bar{x}_{k|k-1})), \\
P_{k|k} &= P_{k|k-1} - K_kH_kP_{k|k-1},
\end{aligned} \tag{2.6}$$

with initialization $\bar{x}_{0|-1} = \bar{x}_0$, $P_{0|-1} = P_0$. The EKF (2.6) looks almost the same as the KF (2.5), except that $f(\bar{x}_{k-1|k-1})$ replaces $F_{k-1}\bar{x}_{k-1|k-1}$ and $h(\bar{x}_{k|k-1})$ replaces $H_k\bar{x}_{k|k-1}$. The EKF is a suboptimal filter, and its convergence is not guaranteed.

2.2 Weighted Extended Kalman Filter

Sometimes the system model is not an accurate model of the actual system. The model inaccuracy degrades the value of past information [44]. Therefore, we

sometimes want to discount old observations, or equivalently, to put more weight on the more recent observations. Fading memory of old observations can compensate the model inaccuracy, and the physical justification is that “old observations, when predicted over long time arcs through an erroneous system, can be valueless” (pp. 305, [44]). This idea of fading memory of old observations has been proposed and studied theoretically and empirically for several decades for the Kalman filter [1][56] [31]. One approach often used is exponential data weighting, or in other words, exponentially discounting the old observations.

Anderson and Moore derived the *Kalman filter with exponential data weighting* (KF-EDW) (pp. 135-138, [1]), by observing that the KF estimate $\bar{x}_{k|k-1}$ is equal to x_k^* , which is the last component of the solution (x_0^*, \dots, x_k^*) to the minimization problem

$$\begin{aligned}
\min_{x_0, \dots, x_k} J_k &= \frac{1}{2}(x_0 - \bar{x}_0)^T P_0^\dagger (x_0 - \bar{x}_0) \\
&+ \frac{1}{2} \sum_{i=0}^{k-1} (y_i - H_i^T x_i) R_i^\dagger (y_i - H_i^T x_i) \\
&+ \frac{1}{2} \sum_{i=0}^{k-1} u_i^T Q_i^\dagger u_i, \tag{2.7} \\
s.t. \quad x_i &= F_{i-1} x_{i-1} + u_{i-1}, \quad i = 1, \dots, k.
\end{aligned}$$

where the superscript \dagger denotes pseudo inverse. The function (2.7) can be viewed as a total cost function on the estimation errors from time 0 to $k - 1$. With this interpretation, placing more emphasis on the more recent observations is equivalent to penalizing the more recent estimation errors more heavily. This suggests increasing the weighting matrices R_i^\dagger and Q_i^\dagger in (2.7) for larger values of i , such as replacing

R_i and Q_i by $\alpha^{-2i}R_i$ and $\alpha^{-2(i+1)}Q_i$ in (2.7), yielding

$$\begin{aligned}
J_k &= \frac{1}{2}(x_0 - \bar{x}_0)^T P_0^\dagger (x_0 - \bar{x}_0) \\
&\quad + \frac{1}{2} \sum_{i=0}^{k-1} (y_i - H_k^T x_i) \alpha^{2i} R_i^\dagger (y_i - H_i^T x_i) \\
&\quad + \frac{1}{2} \sum_{i=0}^{k-1} u_i^T \alpha^{2(i+1)} Q_i^\dagger u_i,
\end{aligned} \tag{2.8}$$

where α is some constant greater than 1. In view of the relationship between (2.7) and the Kalman filter, there could exist a similar relationship between (2.8) and a designed filter. More specifically, we can replace the actual covariance matrices R_k and Q_{k-1} in (2.5) by the design values $\alpha^{-2k}R_k$ and $\alpha^{-2k}Q_{k-1}$, to obtain the designed filter that achieves the minimum of (2.8). The resultant designed filter is the KF-EDW:

$$\begin{aligned}
\bar{x}_{k|k-1} &= F_{k-1} \bar{x}_{k-1|k-1}, \\
P_{k|k-1}^\alpha &= \alpha^2 F_{k-1} P_{k-1|k-1}^\alpha F_{k-1}^T + Q_{k-1}, \\
K_k &= P_{k|k-1}^\alpha H_k^T (R_k + H_k P_{k|k-1}^\alpha H_k^T)^{-1}, \\
\bar{x}_{k|k} &= \bar{x}_{k|k-1} + K_k (y_k - H_k \bar{x}_{k|k-1}), \\
P_{k|k}^\alpha &= P_{k|k-1}^\alpha - K_k H_k P_{k|k-1}^\alpha,
\end{aligned} \tag{2.9}$$

where

$$P_{k|k-1}^\alpha = \alpha^{2k} P_{k|k-1}, \quad P_{k|k}^\alpha = \alpha^{2k} P_{k|k},$$

with initialization $\bar{x}_{0|-1} = \bar{x}_0$, $P_{0|-1}^\alpha = P_0$. The parameter α is called the forgetting factor, because it indicates how fast the observation is forgotten. Notice the main difference between (2.9) and (2.5) is the coefficient α^2 in the second equation. Also

notice that while $P_{k|k}$ in the KF is the error covariance $E[(x_k - \bar{x}_{k|k})(x_k - \bar{x}_{k|k})^T | y_{0:k}]$, the estimate $P_{k|k}^\alpha$ in the KF-EDW is not. Although the KF-EDW is not optimal, the exponential data weighting promotes exponential stability of the filter.

In a similar spirit to how the EKF extends the KF, we extend the KF-EDW to the nonlinear system (2.3) as follows to obtain the *weighted extended Kalman filter* (WEKF):

$$\begin{aligned}
\bar{x}_{k|k-1} &= f(\bar{x}_{k-1|k-1}), \\
P_{k|k-1}^\alpha &= \alpha^2 F_{k-1} P_{k-1|k-1}^\alpha F_{k-1}^T + Q_{k-1}, \\
K_k &= P_{k|k-1}^\alpha H_k^T (R_k + H_k P_{k|k-1}^\alpha H_k^T)^{-1}, \\
\bar{x}_{k|k} &= \bar{x}_{k|k-1} + K_k (y_k - h(\bar{x}_{k|k-1})), \\
P_{k|k}^\alpha &= P_{k|k-1}^\alpha - K_k H_k P_{k|k-1}^\alpha,
\end{aligned} \tag{2.10}$$

with initialization $\bar{x}_{0|-1} = \bar{x}_0$, $P_{0|-1}^\alpha = P_0$. Unlike the optimality difference between the KF and the KF-EDW, the EKF and the WEKF are both suboptimal, and none of the estimates $P_{k|k}$ and $P_{k|k}^\alpha$ are the error covariance. Like the EKF, the convergence of the WEKF is not guaranteed either.

2.3 Particle Filtering

Particle filtering is a class of filters that utilize Monte Carlo simulation and importance sampling techniques to estimate the conditional density of the state given the past observations [4] [29] [21]. It is also called bootstrap filtering [34], sampling importance resampling [4], the condensation algorithm [42], and sequential Monte-Carlo method [21]. Particle filtering approximates the conditional density

$p(x_k|y_{0:k})$ using a finite number of particles and mimicking the evolution of the conditional density through the propagation of these particles. More specifically, the particle filter approximates $p(x_k|y_{0:k})$ by a probability mass function

$$\hat{p}(x_k|y_{0:k}) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i), \quad (2.11)$$

where δ denotes the Dirac delta function, $\{x_k^i, i = 1, \dots, N\}$ are the random support points, and $\{w_k^i, i = 1, \dots, N\}$ are the associated weights satisfying $\{w_k^i \geq 0, i = 1, \dots, N, \sum_{i=1}^N w_k^i = 1\}$.

Since $p(x_k|y_{0:k})$ is unknown, we opt to generate the particles by sampling from another known density $q(x_k|y_{0:k})$, and adjust the weights of the samples to get an estimate of $p(x_k|y_{0:k})$. This approach is known as the *importance sampling*, and the density $q(x_k|y_{0:k})$ is referred to as the *importance density*. The rationale of importance sampling can be observed from the following transformation:

$$E_p[\phi(x)] = \int \phi(x)p(x)dx = \int \phi(x)\frac{p(x)}{q(x)}q(x)dx = E_q[\phi(x)\frac{p(x)}{q(x)}], \quad (2.12)$$

where ϕ is an arbitrary integrable function, p is the target density, and q is the importance density. Hence, from (2.12) it is easy to see that in order to approximate $p(x_k|y_{0:k})$, for samples $\{x_k^i, i = 1, \dots, N\}$ drawn i.i.d. from $q(x_k|y_{0:k})$, their weights should be

$$w_k^i \propto \frac{p(x_k^i|y_{0:k})}{q(x_k^i|y_{0:k})}, \quad (2.13)$$

where \propto means “proportional to”, and the weights should be normalized.

To carry out the estimation recursively, we use the Bayes’ rule to derive the

following recursive equation for the conditional density:

$$\begin{aligned}
p(x_k|y_{0:k}) &= \frac{p(x_k, y_k|y_{0:k-1})}{p(y_k|y_{0:k-1})} \\
&= \frac{p(y_k|y_{0:k-1}, x_k)p(x_k|y_{0:k-1})}{p(y_k|y_{0:k-1})} \\
&= \frac{p(y_k|x_k) \int p(x_k|y_{0:k-1}, x_{k-1})p(x_{k-1}|y_{0:k-1})dx_{k-1}}{p(y_k|y_{0:k-1})} \\
&\propto p(y_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|y_{0:k-1})dx_{k-1}, \tag{2.14}
\end{aligned}$$

where $p(y_k|y_{0:k-1}, x_k) = p(y_k|x_k)$ and $p(x_k|y_{0:k-1}, x_{k-1}) = p(x_k|x_{k-1})$ both follow from the Markovian property of model (2.1), the denominator $p(y_k|y_{0:k-1})$ does not explicitly depend on x_k and k , and \propto means $p(x_k|y_{0:k})$ is the normalized version of the right-hand side. The state transition density $p(x_k|x_{k-1})$ is induced from the state equation in (2.1) and the distribution of the system noise u_{k-1} , and the likelihood $p(y_k|x_k)$ is induced from the observation equation in (2.1) and the distribution of the observation noise v_k . Substituting (2.14) into (2.13), we get

$$w_k^i \propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|y_{0:k})}p(x_{k-1}^i|y_{0:k-1}). \tag{2.15}$$

If the importance density $q(x_k|y_{0:k})$ is chosen to be factored as

$$q(x_k|y_{0:k}) = q(x_k|x_{k-1}, y_k)q(x_{k-1}|y_{0:k-1}), \tag{2.16}$$

then

$$\begin{aligned}
w_k^i &\propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, y_k)} \frac{p(x_{k-1}^i|y_{0:k-1})}{q(x_{k-1}^i|y_{0:k-1})} \\
&\propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, y_k)} w_{k-1}^i. \tag{2.17}
\end{aligned}$$

Moreover, to avoid sample degeneracy, new samples are resampled i.i.d. from the approximate conditional density $\hat{p}(x_k|y_{0:k})$ at each step, hence the weights are

reset to $w_{k-1}^i = 1/N$, and (2.17) is reduced to

$$w_k^i \propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, y_k)}, i = 1, \dots, N. \quad (2.18)$$

In the plain particle filter, the importance density $q(x_k|x_{k-1}^i, y_k)$ is chosen to be the state transition density $p(x_k|x_{k-1}^i)$, which is independent of the current observation y_k , yielding

$$w_k^i \propto p(y_k|x_k^i), i = 1, \dots, N. \quad (2.19)$$

The plain particle filter recursively propagates the support points and updates the associated weights. The algorithm is as follows:

Algorithm 2.1 *Plain Particle Filter*

1. *Initialization: Sample x_0^1, \dots, x_0^N i.i.d. from an initial p.d.f./p.m.f. p_0 . Set $k = 1$.*
2. *Importance Sampling/Propagation: Sample x_k^i from $p(x_k|x_{k-1}^i)$, $i = 1, \dots, N$.*
3. *Bayes' Updating: Receive new observation y_k . The conditional density is approximated by $\hat{p}(x_k|y_{0:k}) = \sum_{i=1}^N w_k^i \delta(x - x_k^i)$, where w_k^i is computed according to (2.19) and normalized.*
4. *Resampling: Sample x_k^1, \dots, x_k^N i.i.d. from $\hat{p}(x_k|y_{0:k})$.*
5. *$k \leftarrow k + 1$ and go to step 2.*

For the resampling step, we can similarly use the importance sampling technique to resample from an importance density $g_{k-1}(x_{k-1}|y_{0:k-1})$, which we will refer

to as the *resampling importance density* to distinguish from $q_k(x_k|x_{k-1}, y_k)$. Hence, in general, the weights are updated according to

$$w_k^i \propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)p(x_{k-1}^i|y_{0:k-1})}{q_k(x_k^i|x_{k-1}^i, y_k)g_{k-1}(x_{k-1}^i|y_{0:k-1})}, \quad (2.20)$$

The plain particle filter (Algorithm 2.2) is a special case of the *general particle filter*, with the particular choice of the importance density $q_k(x_k|x_{k-1}, y_k)$ and the resampling importance density $g_k(x_k|y_{0:k})$.

Algorithm 2.2 *General Particle Filter*

1. *Initialization:* Sample x_0^i, \dots, x_0^N *i.i.d.* from an initial *p.d.f./p.m.f.* p_0 . Set $k = 1$.
2. *Importance Sampling:* Sample x_k^i from $q_k(x_k|x_{k-1}^i, y_k)$, $i = 1, \dots, N$.
3. *Bayes' Updating:* Receive new observation y_k . The conditional density is approximated by $\hat{p}(x_k|y_{0:k}) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i)$, where w_k^i is computed according to (2.20) and normalized.
4. *Importance Resampling:* Sample x_k^1, \dots, x_k^N *i.i.d.* from $g_k(x_k|y_{0:k})$.
5. $k \leftarrow k + 1$ and go to step 2.

It has been proved that $\hat{p}(x_k|y_{0:k})$ converges to $p(x_k|y_{0:k})$ as the sample size N increases to infinity [23] [54]. However, uniform convergence in time has only been proved for the special case where the system dynamics has a mixing kernel that ensures that any error is forgotten (exponentially) in time. Usually, the particle filter needs an increasing number of samples as time k increases to ensure a given precision of the approximation $\hat{p}(x_k|y_{0:k})$ for all k .

2.4 Fading Memory Particle Filter

Particle filtering weights equally all the observations when estimating the current state. However, if, for example, the system model is inaccurate, the prediction based on all the past observations and the system model may not be a good one, which we have already mentioned in Section 2.2. Bad predictions can be also caused by the simulation in the particle filtering, when the system noise is large and the number of samples is not sufficient. This can also be viewed as model inaccuracy, since the system noise is inaccurately represented by the samples due to simulation. The model inaccuracy degrades the value of past information [44]. Therefore, we sometimes want to discount old observations, or equivalently, to put more weight on the more recent observations. This idea of fading memory of old observations has been proposed and studied theoretically and empirically for several decades for the Kalman filter but rarely in particle filtering. Related work on particle filtering with fading memory [25] [28] [66] addresses the issue by incorporating the fading memory into the model not the filter, and the models are tailored for special cases and do not address the general setting.

Our approach is to incorporate the weighted extended Kalman filter to generate the importance density in the particle filter. The weighted extended Kalman filter is based on the Kalman filter with exponential data weighting. The exponential data weighting weights more heavily the more recent observations, and it promotes exponential stability of the Kalman filter. Since the fading memory only affects the locations of the generated particles, the convergence property of the par-

ticle filter is preserved. Moreover, we expect that the fading memory can enhance the convergence of the particle filter, as it does for the Kalman filter.

Like many improved particle filters, such as the auxiliary particle filter [68], the extended Kalman particle filter [25], and the unscented particle filter [88], the fading memory particle filter also incorporates the current observation into generating the importance density and thus holds the promise for generating better importance densities than the plain particle filter which takes $p(x_k|x_{k-1})$ as the importance density. The fading memory particle filter looks similar to the extended Kalman particle filter [25] [88]; however, the fading memory particle filter uses the weighted extended Kalman filter to generate the importance density instead of the unweighted one. With the simple addition of a forgetting factor, the weighted extended Kalman filter has the advantages in giving the filter a prescribed degree of stability and curing many error problems.

Retaining the convergence property of the particle filter, we incorporate the WEKF into the particle filter through the importance density. At time k , the WEKF generates a Gaussian approximation, denoted as $\mathcal{N}(\bar{x}_{k|k}, P_{k|k}^\alpha)$. Let the importance density be

$$q(x_k|x_{k-1}^i, y_k) = N(\bar{x}_{k|k}, P_{k|k}^\alpha), i = 1, \dots, N, \quad (2.21)$$

from which the random support points $\{x_k^i\}_{i=1}^N$ are drawn. Substituting (2.21) into the weight updating equation (2.20), and assuming resampling is applied at time $k - 1$, we obtain

$$w_k^i \propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\mathcal{N}(x_k^i|\bar{x}_{k|k}, P_{k|k}^\alpha)}, i = 1, 2, \dots, N. \quad (2.22)$$

Now that an approximated conditional density $\hat{p}(x_k|y_{0:k}) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i)$ is obtained, it can be used to update the WEKF mean $\bar{x}_{k|k}$ and variance $P_{k|k}^\alpha$ according to

$$\bar{x}_{k|k} = E_{\hat{p}_k}[x], \quad P_{k|k}^\alpha = \text{Var}_{\hat{p}_k}(x),$$

where \hat{p}_k is short for $\hat{p}(x_k|y_{0:k})$. The algorithm is as follows:

Algorithm 2.3 *Fading Memory Particle Filter (FMPPF)*

1. *Initialization: Set forgetting factor $\alpha > 1$. Sample x_0^1, \dots, x_0^N i.i.d. from the initial density p_0 . Set $\bar{x}_0 = E_{p_0}[x]$, $P_0^\alpha = \text{Var}_{p_0}(x)$. Set $k = 1$.*
2. *Importance Sampling:*
 - *Generate the importance density. Compute $\bar{x}_{k|k}$ and $P_{k|k}^\alpha$ by the WEKF (2.10), using $\bar{x}_{k-1|k-1}$, $\bar{P}_{k-1|k-1}^\alpha$ and y_k .*
 - *Sample x_k^1, \dots, x_k^N i.i.d. from $N(\bar{x}_{k|k}, P_{k|k}^\alpha)$.*
3. *Bayes' Updating: Receive new observation y_k . The conditional density is approximated by $\hat{p}(x_k|y_{0:k}) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i)$, where w_k^i is computed according to (2.22) and normalized.*
4. *Resampling. Sample x_k^1, \dots, x_k^N i.i.d. from $\hat{p}(x_k|y_{0:k})$.*
5. *Parameter Updating: $\bar{x}_{k|k} = E_{\hat{p}_k}[x]$, $P_{k|k}^\alpha = \text{Var}_{\hat{p}_k}(x)$.*
6. *$k \leftarrow k + 1$ and go to step 2.*

The FMPPF can be viewed from two viewpoints. The obvious viewpoint is that the WEKF is incorporated to improve the PF, since it is used to generate a better

importance density for the PF. From another viewpoint, the PF is incorporated to improve the WEKF, since particles sampled from the WEKF approximated density $N(\bar{x}_{k|k}, P_{k|k}^\alpha)$ are updated to yield an empirical density $\hat{p}(x_k|y_{0:k})$, which is used to tune the WEKF parameters $\bar{x}_{k|k}$ and $P_{k|k}^\alpha$. Therefore, we expect that the FMKF retains the advantages of both PF and WEKF, namely the asymptotic convergence of the PF and the stability of the WEKF.

Fading memory is not only a technique of weighting heavily the more recent data, but also a way to prevent the divergence of the filter (see pp. 137-138, [1] for details). As the exponential data weighting promotes exponential stability of the Kalman filter, our hope is that fading memory also enhances uniform convergence of the particle filter.

Note that the FMPF only needs to compute one importance density $N(\bar{x}_{k|k}, P_{k|k}^\alpha)$ at time k , whereas the *extended Kalman particle filter* (EKPF) needs to compute a different importance density $N(\bar{x}_{k|k}^i, P_{k|k}^i)$ in order to draw each particle x_k^i (see [88], [4] and [25] for details on EKPF). Hence, the EKPF is much more computationally expensive than the FMPF, especially when the state is multi-dimensional, since the computation of the EKF involves matrix inversions.

We test the FMPF, EKPF, and PPF numerically on the example in [88]

$$\begin{aligned} x_{k+1} &= 1 + \sin(0.04\pi k) + \rho x_k + u_k, x_0 = 0, \\ y_k &= \begin{cases} 0.2x_k^2 + v_k & k \leq 30 \\ 0.5x_k - 2 + v_k & k > 30, \end{cases} \end{aligned}$$

where ρ is a scalar system parameter. The system noise u_k follows a Gamma distribution $\Gamma(3, \theta)$, where θ is the scale parameter. The observation noise v_k follows a

Gaussian distribution $N(0, 1e-5)$. A larger θ means a more spread distribution and hence implies a larger system noise.

We compare the performance of the PPF, EKPF, and FMPF in terms of tracking the true state x_k by the filtered state mean $E[x_k|y_{0:k}]$ in two cases:

- Inaccurate models of system dynamics, i.e., the real system parameter ρ_r is not equal to the model system parameter ρ_m .
- Different values of system noise parameter θ , with other parameters fixed and $\rho_r = \rho_m$.

For each case, we simulate 100 independent runs of length 60 with random initializations, calculate the mean square error (MSE) of the estimated states for each run, and then calculate the means and standard errors of the MSEs of the 100 runs. All three filters use 200 particles and stratified resampling [74]. The forgetting factor in the FMPF is chosen as $\alpha = 1.2$.

Table 2.1 lists the means and standard errors of the MSEs under different real system parameters ρ_r , when the model system parameter is $\rho_m = 0.5$, the system noise is $u_k \sim \Gamma(3, 1)$ (see Fig. 2.1 for a graphical illustration). As we can see, when the model is accurate, i.e., $\rho_r = 0.5 = \rho_m$, the MSEs are smallest for all three filters, which is consistent with our intuition. The PPF is very sensitive to the inaccuracy of the model parameter, while the EKPF and FMPF are much more robust. The FMPF performs best among all three.

Table 2.2 lists the means and standard errors of the MSEs for different values of system noise, with the system parameter $\rho_r = \rho_m = 0.5$ (see Fig. 2.2 for a graphical

illustration). As the system noise increases, the PPF and EKPF deteriorate quickly, and their performances at $\theta = 2$ are probably unacceptable, whereas the FMPF performs well under all cases with slightly increasing MSEs.

Fig. 2.3 shows a typical run of the estimated state generated by the three filters tracking the true state, with $\theta = 2$, and $\rho_r = \rho_m = 0.5$. Fig. 2.4 shows a typical run of the estimated states tracking the true state when the system model is inaccurate, where the real system parameter is $\rho_r = 0.6$, and other parameters are the same as Fig. 2.3. The PPF apparently misses many points before time $k = 30$ in Fig. 2.3, and even beyond $k = 30$ in Fig. 2.4. The EKPF tends to overshoot when there is a big upward change in the trajectory, such as at times $k = 2, 19, 22, 29$ in Fig. 2.3, and $k = 4, 9, 11, 13, 17, 22, 26, 30$ in Fig. 2.4. In contrast, the FMPF tracks the true states very closely in both cases, and shows its robustness with respect to the inaccuracy in the model system parameter. All three filters perform better after time $k = 30$ than before $k = 30$, because the observation function is not invertible in a unique way before $k = 30$.

Table 2.1: Performances under different real system parameters ρ_r , with model system parameter $\rho_m = 0.5$, and system noise $u_k \sim \Gamma(3, 1)$. Each entry shows the mean and standard error (in parentheses) of the MSEs based on 100 independent runs.

ρ_r	$\rho_r = 0.3$	$\rho_r = 0.4$	$\rho_r = 0.5$	$\rho_r = 0.6$
PPF	3.77 (0.111)	1.63 (0.0801)	1.48 (0.102)	3.62 (0.207)
EKPF	0.323 (0.0305)	0.290 (0.0294)	0.290 (0.0392)	0.344 (0.0219)
FMPF	0.112 (0.0092)	0.0696 (0.0069)	0.0630 (0.0124)	0.0854 (0.0092)

Table 2.2: Performances under different system noises, with system parameter $\rho_r = \rho_m = 0.5$. Each entry shows the mean and standard error (in parentheses) of the MSEs based on 100 independent runs.

$Gamma(3, \theta)$	$\theta = 0.5$	$\theta = 1$	$\theta = 1.5$	$\theta = 2$
PPF	0.226 (0.025)	1.48 (0.102)	4.10 (0.271)	8.71 (0.412)
EKPF	0.0087 (0.0010)	0.290 (0.0392)	1.38 (0.123)	5.18 (0.440)
FMPF	0.0086 (0.0010)	0.0630 (0.0124)	0.192 (0.0265)	0.421 (0.0614)

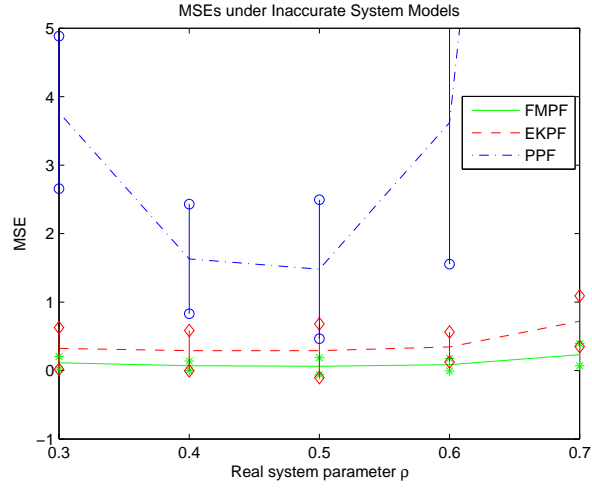


Figure 2.1: A graphic illustration of Table 2.1: Performances (mean and standard error of MSEs based on 100 independent runs) under different values of the real system parameter ρ_r .

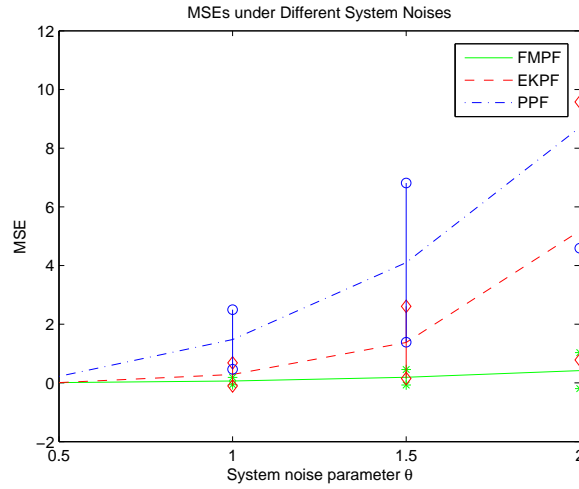


Figure 2.2: A graphic illustration of Table 2.2: Performances (mean and standard error of MSEs based on 100 independent runs) under different system noises.

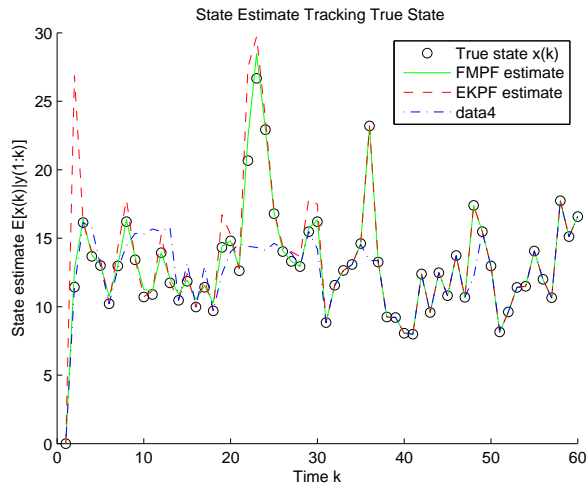


Figure 2.3: A typical run of the estimated states tracking the true state, when the system noise $u_k \sim \Gamma(3, 2)$, and the system parameter $\rho_r = \rho_m = 0.5$.

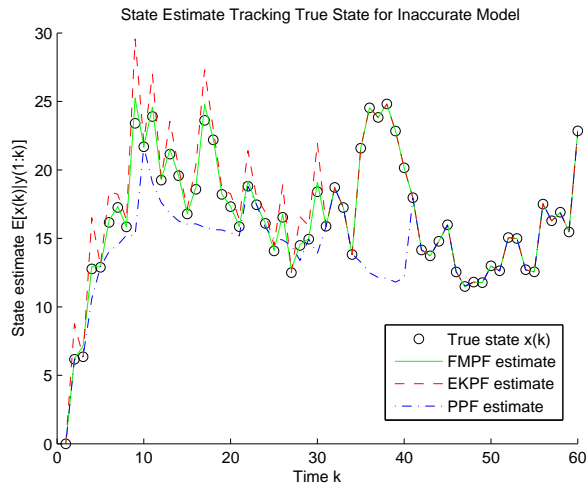


Figure 2.4: A typical run of the estimated states tracking the true state when the system model is inaccurate. The real system parameter $\rho_r = 0.6$, the model system parameter $\rho_m = 0.5$, and the system noise $u_k \sim \Gamma(3, 2)$.

Chapter 3

Stochastic Control

The stochastic control problem involves designing a controller for a stochastic process in order to minimize (or maximize) some cost (or reward) function. In this thesis, we focus on discrete-time problems, namely, Markov decision processes and partially observable Markov decision processes.

3.1 Markov Decision Processes

Consider a stationary discrete-time system model:

$$x_k = f(x_{k-1}, a_{k-1}, u_{k-1}), \quad k = 1, 2, \dots, \quad (3.1)$$

where for all k , the state x_k is in a state space $S \subseteq \mathbb{R}^{n_x}$, the action a_k is in an action space $A \subseteq \mathbb{R}^{n_a}$, and the random disturbance $u_k \in D \subseteq \mathbb{R}^{n_u}$ is a sequence of i.i.d. random vectors with known distributions. Assume that $\{u_k\}$ is independent of x_0 , which follows a distribution p_0 . For simplicity, we assume that all actions in A are admissible to each state $x \in S$, and that D is a countable set to avoid mathematical complications.

Given an initial state x_0 , the objective is to find a policy π that consists of a sequence of functions $\pi = \{\bar{\mu}_0, \bar{\mu}_1, \dots\}$, where $\bar{\mu}_k : S \rightarrow A$, such that π minimizes

the infinite-horizon discounted cost function

$$J_\pi(x_0) = \lim_{H \rightarrow \infty} E_{\{u_k\}_{k=0}^H} \left\{ \sum_{k=0}^H \gamma^k g(x_k, \bar{\mu}_k(x_k), u_k) \right\},$$

subject to the system equation (3.1). The *one-step cost function* $g : S \times A \times D \rightarrow \mathbb{R}$ is given, $\gamma \in (0, 1)$ is the *discount factor*, and $E_{\{u_k\}_{k=0}^H}$ denotes the expectation with respect to the joint distribution of u_0, \dots, u_H . For simplicity, we assume that the above limit defining $J_\pi(x_0)$ exists.

Denote the set of all admissible policies by Π . The *optimal value function* is defined by

$$J_*(x) = \min_{\pi \in \Pi} J_\pi(x), \quad \forall x \in S$$

An *optimal policy*, denoted by π^* , is an admissible policy that achieves J_* . A *stationary policy* is an admissible policy of the form $\pi = \{\mu, \mu, \dots\}$ (i.e., $\bar{\mu}_k$ is independent of k), referred to as the stationary policy μ for brevity, and its corresponding value function is denoted by J_μ .

For any function $J : S \rightarrow \mathbb{R}$, the dynamic programming (DP) mapping applied to J is defined as

$$(TJ)(x) = \min_{a \in A} E_u[g(x, a, u) + \gamma J(f(x, a, u))], \quad x \in S.$$

Hence, the mapping T transforms the function J on S into the function TJ on S .

For a given stationary policy μ , the mapping T_μ is defined as

$$(T_\mu J)(x) = E_u[g(x, \mu(x), u) + \gamma J(f(x, \mu(x), u))], \quad x \in S.$$

Throughout this section, we assume the following:

Assumption 3.1 *The one-step cost function g satisfies*

$$|g(x, a, u)| \leq M < \infty, \quad \forall (x, a, u) \in S \times A \times D.$$

Please note this is the simplest type of infinite-horizon discounted cost MDPs. Nonetheless, the boundedness of the one-step cost function is not as restrictive as it seems to be. The boundedness is satisfied if S , A , and D are of finite cardinality. Even if they are not, the boundedness is satisfied in many computation methods because they are approximated by finite sets. It needs more complicated technical treatment for MDPs involving unbounded one-step cost functions, which we will not delve into.

The following proposition (cf. prop. 1.2.2 [10]) shows that the value function J_* is the unique solution of Bellman's equation.

Proposition 1 *The optimal value function J_* satisfies*

$$J_*(x) = \min_a E_u[g(x, a, u) + \gamma J_*(f(x, a, u))], \quad x \in S,$$

or equivalently,

$$J_* = TJ_*.$$

Furthermore, J_ is the unique solution of this equation within the class of bounded functions.*

3.1.1 Value Iteration

Value iteration is an iterative method to compute the value function J_* . It is basically the dynamic programming algorithm, and its performance is based on the

convergence of dynamic programming, as shown in the next proposition (cf. prop. 1.2.1 in [10]).

Proposition 2 *For any bounded function $J : S \rightarrow \mathbb{R}$, the value function satisfies*

$$J_*(x) = \lim_{k \rightarrow \infty} (T^k J)(x), \quad \forall x \in S.$$

Value iteration starts with an arbitrary bounded function $J_0(x), \forall x \in S$, and then at each iteration k applies the DP mapping T to the old function $J_k(x)$ to get a new function $J_{k+1}(x)$ for all $x \in S$. More specifically, at each iteration k ,

$$J_{k+1}(x) = (TJ_k)(x) = \min_{a \in A} E_u[g(x, a, u) + \gamma J_k(f(x, a, u))], \quad \forall x \in S.$$

In view of Proposition 2, the value iteration method usually needs an infinite number of iterations to achieve convergence. Hence, it is desirable to have an estimate on the convergence rate. The following convergence rate [10] [73] has been established for any bounded function J :

$$\max_{x \in S} |(T^k J)(x) - J_*(x)| \leq \gamma^k \max_{x \in S} |J(x) - J_*(x)|, \quad k = 0, 1, \dots$$

The value iteration method may be implementable only approximately, if the state space is continuous or has a finite but large number of states. Instead of computing $(TJ)(x)$ for all states $x \in S$, $(TJ)(x)$ can be computed for only some of the states and estimated for the other states. The expectation in (3.1.1) may also be computed through approximation or simulation.

3.1.2 Policy Iteration

As an alternative to value iteration, the policy iteration method directly works with policies to find an optimal stationary policy. It starts with an arbitrary stationary policy μ_0 , at each iteration, evaluates the current policy μ_k to obtain the associated value function J_{μ_k} , and then computes a new policy μ_{k+1} based on J_{μ_k} . The evaluation of the policy μ_k is based on the following corollary (cf. cor. 1.2.2.1 in [10]) that is derived from Proposition 2.

Corollary 1 *For every stationary policy μ , the associated value functions satisfies*

$$J_\mu(x) = E_u[g(x, \mu(x), u) + \gamma J_\mu(f(x, \mu(x), u))], \quad \forall x \in S,$$

or, equivalently,

$$J_\mu = T_\mu J_\mu.$$

Furthermore, J_μ is the unique solution of this equation within the class of bounded functions.

Therefore, the detailed algorithm for the policy iteration method is as follows.

Algorithm 3.1 Policy Iteration Algorithm

1. *Initialization: Guess an initial stationary policy μ_0 .*
2. *Policy Evaluation: Evaluate the value function J_{μ_k} associated with the current stationary policy μ_k by solving*

$$J_{\mu_k} = T_{\mu_k} J_{\mu_k}.$$

3. *Policy Improvement: Compute a new stationary policy μ_{k+1} by applying the DP mapping T to the value function J_{μ_k} , i.e.,*

$$T_{\mu_{k+1}} J_{\mu_k} = T J_{\mu_k}.$$

4. *If $J_{\mu_k} = T J_{\mu_k}$, stop; otherwise return to the policy evaluation step and repeat the process.*

It can be proved that $J_{\mu_{k+1}} \leq J_{\mu_k}$, and the strict inequality $J_{\mu_{k+1}}(x) < J_{\mu_k}(x)$ holds for at least one $x \in S$, if μ_k is not optimal. Hence, μ_{k+1} is strictly better than μ_k , if μ_k is not optimal. For finite state and action spaces, the number of admissible policies is finite, and hence, the policy iteration algorithm terminates after a finite number of iterations to yield an optimal policy.

Similar to the value iteration method, the policy iteration method is implementable only approximately, if the state space is continuous or has a finite but large number of states. The policy evaluation step may be approximated using a finite number of value iterations or linear programming. The policy improvement step may be carried out for only some of the states, and approximated for the remaining states.

In addition to variations of the value iteration and policy iteration methods, there are other approximation methods for solving large-state MDPs, such as the state aggregation approach [11], approximation using basis functions and linear programming [80], and approximation using post-decision state variable [72].

3.1.3 Simulation-Based Methods

The above methods are applicable only when the system model and cost function are available. However, sometimes the system model is not available, but instead, the system and cost structure can be simulated. The above methods can still be applied to this case, with the help of using Monte Carlo simulation to calculate approximately the transition probabilities and cost functions. Another branch of simulation-based methods is the so-called reinforcement learning [86] or neurodynamic programming, such as temporal difference learning [85], and Q-learning [89]. The reinforcement learning methods learn the system model (and sometimes a policy) through repeatedly simulating the system using the current policy and improving the current policy. Some of the recent simulation-based methods concern efficient allocation of the simulation budget by focusing on the more promising actions or policies, such as the multi-stage adaptive sampling algorithm, and the population-based evolutionary approaches, both described in [22].

3.2 Partially Observable Markov Decision Processes

Consider a stationary discrete-time continuous-state system model:

$$x_k = f(x_{k-1}, a_{k-1}, u_{k-1}), k = 1, 2, \dots, \quad (3.2)$$

$$y_k = h(x_k, a_{k-1}, v_k), k = 1, 2, \dots, \quad y_0 = h_0(x_0, v_0), \quad (3.3)$$

where for all k , the state x_k is in a continuous state space $S \subseteq \mathbb{R}^{n_x}$, the action a_k is in an action space $A \subset \mathbb{R}^{n_a}$, the observation y_k is in a continuous observation space $O \subseteq \mathbb{R}^{n_y}$, and the random disturbances $u_k \in \mathbb{R}^{n_u}$ and $v_k \in \mathbb{R}^{n_v}$ are sequences

of i.i.d. random vectors with known distributions. Assume that $\{u_k\}$ and $\{v_k\}$ are independent of each other, and independent of x_0 , which follows a distribution p_0 . Also assume that $f(x, a, u)$ is continuous in x for every $a \in A$ and $u \in \mathbb{R}^{n_u}$, $h(x, a, v)$ is continuous in x for every $a \in A$ and $v \in \mathbb{R}^{n_v}$, and $h_0(x, v)$ is continuous in x for every $v \in \mathbb{R}^{n_v}$. Eq. (3.2) is often called the state equation, and (3.3) the observation equation.

All the information available to the decision maker at time k can be summarized by means of an *information vector* I_k , which is defined as

$$\begin{aligned} I_k &= (y_0, y_1, \dots, y_k, a_0, a_1, \dots, a_{k-1}), k = 1, 2, \dots, \\ I_0 &= y_0. \end{aligned}$$

The objective is to find a policy π consisting of a sequence of functions $\pi = \{\bar{\mu}_0, \bar{\mu}_1, \dots\}$, where each function μ_k maps the information vector I_k onto the action space A , that minimizes the *cost function*

$$J_\pi = \lim_{H \rightarrow \infty} E_{x_0, \{u_k\}_{k=0}^{H-1}, \{v_k\}_{k=0}^H} \left\{ \sum_{k=0}^H \gamma^k g(x_k, \mu_k(I_k)) \right\},$$

where $g : S \times A \rightarrow \mathbb{R}$ is the one-step cost function, $\gamma \in (0, 1)$ is the discount factor, and $E_{x_0, \{u_k\}_{k=0}^{H-1}, \{v_k\}_{k=0}^H}$ denotes the expectation with respect to the joint distribution of $x_0, u_0, \dots, u_{H-1}, v_0, \dots, v_H$. For simplicity, we assume that the above limit defining J_π exists. The value function is defined by

$$J_* = \min_{\pi \in \Pi} J_\pi,$$

where Π is the set of all admissible policies. An *optimal policy*, denoted by π^* , is an admissible policy that achieves J_* . A stationary policy is an admissible policy of

the form $\pi = \{\mu, \mu, \dots\}$, referred to as the stationary policy μ for brevity, and its corresponding value function is denoted by J_μ .

3.2.1 Belief MDPs

The information vector I_k grows as the history expands. The standard approach to encode historical information is the use of the *belief state*, which is the conditional probability density of the current state x_k given the past history, i.e., $b_k : S \rightarrow [0, \infty)$:

$$b_k(x) \triangleq p(x_k = x | I_k).$$

Given our assumptions on (3.2) and (3.3), b_k exists, and can be computed recursively via Bayes' rule:

$$\begin{aligned} b_k(x) &= p(x_k = x | I_{k-1}, a_{k-1}, y_k) \\ &\propto p(y_k | x_k = x, a_{k-1}) \int_S p(x_k = x | I_{k-1}, a_{k-1}, x_{k-1}) \dots \\ &\quad p(x_{k-1} | I_{k-1}, a_{k-1}) dx_{k-1} \\ &\propto p(y_k | x_k = x, a_{k-1}) \int_S p(x_k = x | a_{k-1}, x_{k-1}) \dots \\ &\quad b_{k-1}(x_{k-1}) dx_{k-1}, \end{aligned} \tag{3.4}$$

where the second line follows from the Markovian property; \propto means “proportional to” because the denominator $p(y_k | I_{k-1}, a_{k-1})$ does not explicitly depend on x_k or k ; and the third line follows from the Markovian property of $\{x_k\}$ and the fact that a_{k-1} is a function of I_{k-1} given a policy. The righthand side of (3.4) can be expressed

in terms of b_{k-1} , a_{k-1} and y_k . Hence,

$$b_k = \psi(b_{k-1}, a_{k-1}, y_k), \quad (3.5)$$

where y_k is characterized by the time-homogeneous conditional distribution $P_Y(y_k|b_{k-1})$ that is induced by (3.2) and (3.3), and does not depend on $\{y_0, \dots, y_{k-1}\}$.

A POMDP can be converted to an MDP by conditioning on the information vectors, and the converted MDP is called the *belief MDP* (Chapter 5, [10]). The states of the belief MDP are the belief states, which follow the system dynamics (3.5), where y_k can be seen as the system noise with the distribution P_Y . The state space of the belief MDP is the *belief space*, denoted by B , which is the set of all belief states, i.e., a set of probability densities. A policy π is a sequence of functions $\pi = \{\mu_0, \mu_1, \dots\}$, where each function μ_k maps the belief state b_k onto the action space A . Notice that

$$E_{x_0, \{u_i\}_{i=0}^{k-1}, \{v_i\}_{i=0}^k} \{g(x_k, a_k)\} = E \{E_{x_k} \{g(x_k, a_k) | I_k\}\},$$

thus the one-step cost function can be written in terms of the belief state as the *belief one-step cost function*

$$\begin{aligned} \tilde{g}(b_k, a_k) &\triangleq E_{x_k} \{g(x_k, a_k) | I_k\} \\ &= \int_{x \in S} g(x, a_k) b_k(x) dx \\ &\triangleq \langle g(\cdot, a), b \rangle. \end{aligned}$$

Assuming there exists a stationary optimal policy, the optimal value function is given by

$$J_*(b) = \lim_{k \rightarrow \infty} T^k J(b), \quad \forall b \in B,$$

where T is the *dynamic programming (DP) mapping* operated on any bounded function $J : S \rightarrow \mathbb{R}$ according to

$$TJ(b) = \min_{a \in A} [\langle g(\cdot, a), b \rangle + \gamma E_Y \{J(\psi(b, a, Y))\}], \quad (3.6)$$

where E_Y denotes the expectation with respect to the distribution P_Y .

3.2.2 Finite-State vs. Continuous-State

For finite-state POMDPs, the belief state b is a vector with each entry being the probability of being at one of the states, and hence, the belief space B is a finite-dimensional probability simplex. Past research on numerical solutions of POMDPs is mostly focused on finite-state problems. For finite-state POMDPs, it is proved that the value function is a piecewise linear convex function after a finite number of iterations, provided that the one-step cost function is piecewise linear and convex [81]. This feature has been exploited in various exact and approximate algorithms such as those found in [82], [81], [83], [57], and [36]. The algorithm in [81] and the Witness algorithm in [57] carry out exact value iterations by finding and updating the minimum set of linear functions that determine the value function at each iteration for a finite-horizon problem. Because the number of such linear functions grow exponentially with the number of iterations, these algorithms are computationally very expensive and are limited to very simple problems in practice. Howard [39] and Sondik [83] use policy iteration to solve the infinite-horizon discounted-cost problems. Hauskrecht [36] summarizes several value function approximation methods in a nice framework of modifying the value iteration equation by changing the order

of summations and maximization, including approximation with fully observable MDP, approximation with Q-functions, fast informed bound method, and approximation with unobservable MDP. [36] also summarizes many grid-based methods with various techniques of choosing the set of grids and approximating the value function.

For continuous-state POMDPs, the belief state b is a continuous density, and thus, the belief space B is an infinite-dimensional space that contains all sorts of continuous densities. For continuous-state POMDPs, the value function preserves convexity [91], but value iteration algorithms are not directly applicable because the belief space is infinite dimensional. The infinite-dimensionality of the belief space also creates difficulties in applying the approximate algorithms that were developed for finite-state POMDPs. For example, one straightforward and commonly used approach is to approximate a continuous-state POMDP by a finite-state one via discretization of the state space. In practice, this could lead to computational difficulties, either resulting in a belief space that is of huge dimension or in a solution that is not accurate enough. In addition, note that even for a relatively nice prior distribution b_k (e.g., a Gaussian distribution), the exact evaluation of the posterior distribution b_{k+1} is computationally intractable; moreover, the update b_{k+1} may not have any structure, and therefore can be very difficult to handle. Past research is relatively sparse on numerically solving continuous-state POMDPs.

Chapter 4

Solving Continuous-State POMDPs

4.1 Related Work and Motivation

As described at the end of last chapter, a POMDP can be converted to a continuous-state Markov decision process (MDP) by introducing the notion of the belief state [10], which is the conditional distribution of the current state given the history of observations and actions. For a finite-state POMDP, the belief space is finite dimensional (i.e., a probability simplex), whereas for a continuous-state POMDP, the belief space is an infinite-dimensional space of continuous probability distributions. This difference suggests that simple generalizations of many of the finite-state algorithms to continuous-state models are not appropriate or applicable. For example, discretization of the continuous-state space may result in a finite-state POMDP of dimension either too large to solve computationally or not sufficiently precise. Taking another example, many algorithms for solving finite-state POMDPs (see [36] for a survey) are based on discretization of the finite-dimensional probability simplex; however, it is usually not feasible to discretize an infinite-dimensional probability distribution space. Throughout this chapter, when we use the word “dimension” or “dimensional”, we refer to the dimension of the belief space/state.

Despite the abundance of algorithms for finite-state POMDPs, the aforementioned difficulty has motivated some researchers to look for efficient algorithms for

continuous-state POMDPs [69] [70] [87] [75] [17] [18]. Assuming discrete observation and action spaces, Porta et al. [69] showed that the optimal finite-horizon value function is defined by a finite set of “ α -functions”, and model all functions of interest by Gaussian mixtures. In a later work [70], they extended their result and method to continuous observation and action spaces using sampling strategies. However, the number of Gaussian mixtures in representing belief states and α -functions grows exponentially in value iteration as the number of iterations increases. Thrun [87] addressed continuous-state POMDPs using particle filtering to simulate the propagation of belief states and represent the belief states by a finite number of samples. The number of samples determines the dimension of the belief space, and the dimension could be very high in order to approximate the belief states closely.

Roy [75] and Brooks et al. [17] used sufficient statistics to reduce the dimension of the belief space, which is often referred to as belief compression in the Artificial Intelligence literature. Roy [75] proposed an augmented MDP (AMDP), using maximum likelihood state and entropy to characterize belief states, which are usually not sufficient statistics except for a linear Gaussian model. As shown by Roy himself, the algorithm fails in a simple robot navigation problem, since the two statistics are not sufficient for distinguishing between a unimodal distribution and a bimodal distribution. Brooks et al. [17] proposed a parametric POMDP, representing the belief state as a Gaussian distribution so as to convert the POMDP to a problem of computing the value function over a two-dimensional continuous space, and using the extended Kalman filter to estimate the transition of the approximated belief state. The restriction to the Gaussian representation has the same

problem as the AMDP. The algorithm recently proposed in Brooks and Williams [18] is similar to ours, in that they also approximate the belief state by a parameterized density and solve the approximate belief MDP on the parameter space using Monte Carlo simulation-based methods. However, they did not specify how to calculate the parameters except for Gaussian densities, whereas we explicitly provide an analytical way to calculate the parameters for exponential families of densities. Moreover, we develop rigorous theoretical error bounds for our algorithm. There are some other belief compression algorithms designed for finite-state POMDP, such as value-directed compression [71] and the exponential family principle components analysis (E-PCA) belief compression [76]. They are not suitable for generalization to continuous-state models, since they are based on a fixed set of support points.

Motivated by the work of [87] [75] and [17], we develop a computationally tractable algorithm that effectively reduces the dimension of the belief state and has the flexibility to represent arbitrary belief states, such as multimodal or heavy tail distributions. The idea is to project the original high/infinite-dimensional belief space to a low-dimensional family of parameterized distributions by minimizing the Kullback-Leibler (KL) divergence between the belief state and its projection on that family of distributions. For an exponential family, the minimization of KL divergence can be carried out in analytical form, making the method very easy to implement. The projected belief MDP can then be solved on the parameter space by using simulation-based algorithms, or can be further approximated by a finite-state MDP via a suitable discretization of the parameter space and thus solved by using standard solution techniques such as value iteration and policy iteration. Our

method can be viewed as a generalization of the AMDP in [75] and the parametric POMDP in [17], which considers only the family of Gaussian distributions. In addition, we provide theoretical results on the error bound of the value function and the performance of the near-optimal policy generated by our method.

We also develop a projection particle filter for online filtering and decision making, by incorporating the density projection technique into particle filtering. The projection particle filter we propose here is a modification of the projection particle filter in [5]. Unlike in [5] where the *predicted* conditional density is projected, we project the *updated* conditional density, so as to ensure the projected belief state remains in the given family of densities. Although seemingly a small modification in the algorithm, we prove under much less restrictive assumptions a similar bound on the error between our projection particle filter and the exact filter.

4.2 Dimension Reduction

4.2.1 Density Projection

Density projection is a useful idea to approximate an arbitrary (most likely, infinite-dimensional) density as accurately as possible by a density in a chosen family that is characterized by only a few parameters. A *projection mapping* from the belief space B to a family of parameterized densities Ω , denoted as $Proj_{\Omega} : B \rightarrow \Omega$, is defined by

$$Proj_{\Omega}(b) \triangleq \arg \min_{f \in \Omega} D_{KL}(b||f), \quad b \in B, \quad (4.1)$$

where $D_{KL}(b||f)$ denotes the *Kullback-Leibler (KL) divergence* (or *relative entropy*) between b and f , which is

$$D_{KL}(b||f) \triangleq \int b(x) \log \frac{b(x)}{f(x)} dx. \quad (4.2)$$

Hence, the projection of b on Ω has the minimum KL divergence from b among all the densities in Ω .

When Ω is an exponential family of densities, the minimization (4.1) has an analytical solution and can be carried out easily. The exponential families include many common families of densities, such as Gaussian, binomial, Poisson, Gamma, etc. An *exponential family of densities* is defined as follows [6]:

Definition 4.1 Let $\{c_1(\cdot), \dots, c_m(\cdot)\}$ be affinely independent scalar functions defined on \mathbb{R}^n , i.e., for distinct points x_1, \dots, x_{m+1} , $\sum_{i=1}^{m+1} \lambda_i c(x_i) = 0$ and $\sum_{i=1}^{m+1} \lambda_i = 0$ implies $\lambda_1, \dots, \lambda_{m+1} = 0$, where $c(x) = [c_1(x), \dots, c_m(x)]^T$. Assuming that $\Theta_0 = \{\theta \in \mathbb{R}^m : \varphi(\theta) = \log \int \exp(\theta^T c(x)) dx < \infty\}$ is a convex set with a nonempty interior, then Ω defined by

$$\Omega = \{f(\cdot, \theta), \theta \in \Theta\},$$

$$f(x, \theta) = \exp[\theta^T c(x) - \varphi(\theta)],$$

where $\Theta \subseteq \Theta_0$ is open, is called an *exponential family of probability densities*, with θ its parameter and $c(x)$ its sufficient statistic.

Substituting $f(x) = f(x, \theta)$ into (4.2) and expressing it further as

$$D_{KL}(b||f(\cdot, \theta)) = \int b(x) \log b(x) dx - \int b(x) \log f(x, \theta) dx,$$

we can see that the first term does not depend on $f(\cdot, \theta)$, hence $\min D_{KL}(b||f(\cdot, \theta))$ is equivalent to

$$\max \int b(x) \log f(x, \theta) dx,$$

which by Definition 4.1 is the same as

$$\max \int (\theta^T c(x) - \varphi(\theta)) b(x) dx. \quad (4.3)$$

Recall the fact that the log-likelihood $l(\theta) = \theta^T c(x) - \varphi(\theta)$ is strictly concave in θ [55], and therefore, $\int (\theta^T c(x) - \varphi(\theta)) b(x) dx$ is also strictly concave in θ . Hence, (4.3) has a unique maximum and the maximum is achieved when the first-order optimality condition is satisfied, i.e.,

$$\int \left(c_j(x) - \frac{\int c_j(x) \exp(\theta^T c(x)) dx}{\int \exp(\theta^T c(x)) dx} \right) b(x) dx = 0.$$

Therefore, b and its projection $f(\cdot, \theta)$ is related by

$$E_b [c_j(X)] = E_\theta [c_j(X)], \quad j = 1, \dots, m, \quad (4.4)$$

where E_b and E_θ denote the expectations with respect to b and $f(\cdot, \theta)$, respectively.

4.2.2 Projected Belief MDP

Using the idea of density projection, we can transform the belief MDP to another MDP confined on a low-dimensional belief space, and then solve this MDP problem. We call such an MDP the *projected belief MDP*. Its state is the *projected belief state* $b_k^p \in \Omega$ that satisfies the system dynamics

$$\begin{aligned} b_0^p &= Proj_\Omega(b_0), \\ b_k^p &= \psi(b_{k-1}^p, a_{k-1}, y_k)^p, \quad k = 0, 1, \dots, \end{aligned}$$

where $\psi(b_{k-1}^p, a_{k-1}, y_k)^p = Proj_{\Omega}(\psi(b_{k-1}^p, a_{k-1}, y_k))$, and the dynamic programming mapping on the projected belief MDP is

$$T^p J(b^p) = \min_{a \in A} [\langle g(\cdot, a), b^p \rangle + \gamma E_Y \{J(\psi(b^p, a, Y)^p)\}]. \quad (4.5)$$

For the projected belief MDP, a policy is denoted as $\pi^p = \{\mu_0^p, \mu_1^p, \dots\}$, where each function μ_k^p maps the projected belief state b_k^p into the action space A . Similarly, a stationary policy is denoted as μ^p ; an optimal stationary policy is denoted as μ_*^p ; and the optimal value function is denoted as $J_*^p(b^p)$.

The projected belief MDP is in fact a low-dimensional continuous-state MDP, and can be solved in numerous ways. One common approach is to use value iteration or policy iteration by converting the projected belief MDP to a discrete-state MDP problem via a suitable discretization of the projected belief space (i.e., the parameter space) and then estimating the one-step cost function and transition probabilities on the discretized mesh. The effect of the discretization procedure on dynamic programming has been studied in [9]. We describe this approach in detail below.

Discretization of the projected belief space Ω is equivalent to discretization of the parameter space Θ , which yields a set of grid points, denoted by $G = \{\theta_i, i = 1, \dots, N\}$. Let $\tilde{g}(\theta_i, a)$ denote the one-step cost function associated with taking action a at the projected belief state $b^p = f(\cdot, \theta_i)$. Let $\tilde{P}(\theta_i, a)(\theta_j)$ denote the transition probability from the current projected belief state $b_k^p = f(\cdot, \theta_i)$ to the next projected belief state $b_{k+1}^p = f(\cdot, \theta_j)$ by taking action a . Estimation of $\tilde{P}(\theta_i, a)(\theta_j)$ is done using a variation of the projection particle filtering algorithm, to be described

in the next section. $\tilde{g}(\theta_i, a)$ can be estimated by its sample mean:

$$\tilde{g}(\theta_i, a) = \frac{1}{N} \sum_{j=1}^N g(x_j, a), \quad (4.6)$$

where x_1, \dots, x_N are sampled i.i.d. from $f(\cdot, \theta_i)$.

Remark 4.1 *The approach for solving the projected belief MDP described here is probably the most intuitive, but not necessarily the most computationally efficient. Other more efficient techniques for solving continuous-state MDPs can be used to solve the projected belief MDP, such as the linear programming approach [27], neuro-dynamic programming methods [12], and simulation-based methods [22].*

4.3 Projection Particle Filtering

Solving the projected belief MDP gives us a policy, which tells us what action to take at each projected belief state. In an online implementation, at each time k , the decision maker receives a new observation y_k , estimates the belief state b_k , and then chooses his action a_k according to b_k and that policy. Hence, to implement our approach requires addressing the problem of estimating the belief state. Estimation of b_k , or simply called *filtering*, does not have an analytical solution in most cases except linear Gaussian systems, but it can be solved using many approximation methods, such as the extended Kalman filter and particle filtering. Here we focus on particle filtering, because 1) it outperforms the extended Kalman filter in many nonlinear/non-Gaussian systems [4], and 2) we will develop a projection particle filter to be used in conjunction with the projected belief MDP.

To obtain a reasonable approximation of the belief state, particle filtering needs a large number of samples/particles. Since the number of samples/particles is the dimension of the approximate belief state \hat{b} , particle filtering is not very helpful in reducing the dimensionality of the belief space. Moreover, particle filtering does not give us an approximate belief state in the projected belief space Ω , hence the policy we obtained by solving the projected belief MDP is not immediately applicable.

We incorporate the idea of density projection into particle filtering, so as to approximate the belief state by a density in Ω . The projection particle filter we propose here is a modification of the one in [5]. Their projection particle filter projects the empirical *predicted* belief state, not the empirical *updated* belief state, onto a parametric family of densities, and hence, the approximate belief state might not be in that family after Bayes' updating. We will directly project the empirical *updated* belief state onto a parametric family. In addition, we will need much less restrictive assumptions than [5] to obtain similar error bounds. Since resampling is from a continuous distribution instead of an empirical (discrete) one, the proposed projection particle filter also overcomes the difficulty of sample impoverishment [4] that occurs in the bootstrap filter.

Applying the density projection technique we described in the last section, projecting the empirical belief state \hat{b}_k onto an exponential family Ω involves finding a $f(\cdot, \theta)$ with the parameter θ satisfying (4.4). Hence, letting $b = \hat{b}_k$ in (4.4) and plugging in (2.11), θ should satisfy

$$\sum_{i=1}^N w_i c_j(x_{k|k-1}^i) = E_{\theta} [c_j], \quad j = 1, \dots, m,$$

which constitutes the projection step in the projection particle filter.

Algorithm 4.1 (*Projection particle filtering for an exponential family of densities (PPF)*)

- *Input: a (stationary) policy μ^p on the projected belief MDP; a family of exponential densities $\Omega = \{f(\cdot, \theta), \theta \in \Theta\}$; a sequence of observations y_1, y_2, \dots arriving sequentially at time $k = 1, 2, \dots$*
- *Output: a sequence of approximate belief states $f(\cdot, \hat{\theta}_1), f(\cdot, \hat{\theta}_2), \dots$*
- *Step 1. Initialization: Sample x_0^1, \dots, x_0^N i.i.d. from the approximate initial belief state $f(\cdot, \hat{\theta}_0)$. Set $k = 1$.*
- *Step 2. Prediction: Compute $x_{k|k-1}^1, \dots, x_{k|k-1}^N$ by propagating $x_{k-1}^1, \dots, x_{k-1}^N$ according to the system dynamics (3.2) using the action $a_{k-1} = \mu^p(f(\cdot, \hat{\theta}_{k-1}))$ and randomly generated noise $\{u_{k-1}^i\}_{i=1}^N$, i.e., sample $x_{k|k-1}^i$ from $p(\cdot | x_{k-1}^i, a_{k-1})$, $i = 1, \dots, N$.*
- *Step 3. Bayes' updating: Receive a new observation y_k . Calculate weights as*

$$w_k^i = \frac{p(y_k | x_k^i, a_{k-1})}{\sum_{i=1}^N p(y_k | x_k^i, a_{k-1})}, \quad i = 1, \dots, N.$$

- *Step 4. Projection: The approximate belief state is $f(\cdot, \hat{\theta}_k)$, where $\hat{\theta}_k$ satisfies the equations*

$$\sum_{i=1}^N w_k^i c_j(x_{k|k-1}^i) = E_{\hat{\theta}_k}[c_j], \quad j = 1, \dots, m.$$

- *Step 5. Resampling: Sample x_k^1, \dots, x_k^N from $f(\cdot, \hat{\theta}_k)$.*

- *Step 6.* $k \leftarrow k + 1$ and go to Step 2.

In an online implementation, at each time k , the PPF approximates b_k by $f(\cdot, \hat{\theta}_k)$, and then decides an action a_k according to $a_k = \mu^p(f(\cdot, \hat{\theta}_k))$, where μ^p is an optimal policy solved for the projected belief MDP.

As mentioned in the last section, PPF can be varied slightly for estimating the transition probabilities of the discretized projected belief MDP, as follows:

Algorithm 4.2 (*Estimation of the transition probabilities*)

- *Input:* θ_i, a, N ;
- *Output:* $\tilde{P}(\theta_i, a)(\theta_j), j = 1, \dots, N$.
- *Step 1. Sampling:* Sample x_1, \dots, x_N from $f(\cdot, \theta_i)$.
- *Step 2. Prediction:* Compute $\tilde{x}_1, \dots, \tilde{x}_N$ by propagating x_1, \dots, x_N according to the system dynamics (3.2) using the action a and randomly generated noise $\{u_i\}_{i=1}^N$.
- *Step 3. Sampling observation:* Compute y_1, \dots, y_N from $\tilde{x}_1, \dots, \tilde{x}_N$ according to the observation equation (3.3) using randomly generated noise $\{v_i\}_{i=1}^N$.
- *Step 4. Bayes' updating:* For each $y_k, k = 1, \dots, N$, the updated belief state is

$$\tilde{b}_k = \sum_{i=1}^N w_i^k \delta(x - \tilde{x}_i),$$

where

$$w_i^k = \frac{p(y_k | \tilde{x}_i, a)}{\sum_{i=1}^N p(y_k | \tilde{x}_i, a)}, \quad i = 1, \dots, N.$$

- *Step 5. Projection:* For $k = 1, \dots, N$, project each \tilde{b}_k to the exponential family, i.e., finding $\tilde{\theta}_k$ that satisfies (4.4).
- *Step 6. Estimation:* For $k = 1, \dots, N$, find the nearest-neighbor of $\tilde{\theta}_k$ in G . For each $\theta_j \in G$, count the frequency $\tilde{P}(\theta_i, a)(\theta_j) = (\text{number of } \theta_j)/N$.

4.4 Analysis of Projected Belief MDP

4.4.1 Main Idea

Our method solves the projected belief MDP instead of the original belief MDP, and that raises two questions: How well does the optimal value function of the projected belief MDP approximate the optimal value function of the original belief MDP? How well does the optimal policy obtained by solving the projected belief MDP perform on the original belief MDP? To answer these questions, we first need to rephrase them mathematically.

Here we assume perfect computation of the belief states and the projected belief states. We also assume the existence of an optimal policy that is stationary, as stated below.

Assumption 4.1 *There exist a stationary optimal policy for the belief MDP, denoted by μ_* , and a stationary optimal policy for the projected belief MDP, denoted by μ_*^p .*

Assumption 4.1 holds under some mild conditions [10], [37]. Using the stationarity, and the dynamic programming mapping on the belief MDP and the projected

belief MDP given by (3.6) and (4.5), the optimal value function $J_*(b)$ for the belief MDP can be obtained by

$$J_*(b) \triangleq J_{\mu_*}(b) = \lim_{k \rightarrow \infty} T^k J_0(b),$$

and the optimal value function for the projected belief MDP obtained by

$$J_*^p(b^p) \triangleq J_{\mu_*^p}^p(b^p) = \lim_{k \rightarrow \infty} (T^p)^k J_0(b^p).$$

Therefore, the questions posed at the beginning of this section can be formulated mathematically as:

1. How well the optimal value function of the projected belief MDP approximates the true optimal value function can be measured by

$$|J_*(b) - J_*^p(b^p)|.$$

2. How well the optimal policy μ_*^p for the projected belief MDP performs on the original belief space can be measured by

$$|J_*(b) - J_{\bar{\mu}_*^p}(b)|,$$

where $\bar{\mu}_*^p(b) \triangleq \mu_*^p \circ Proj_\Omega(b) = \mu_*^p(b^p)$.

4.4.2 Error Bound

The next assumption bounds the difference between the belief state b and its projection b^p , and also the difference between their one-step evolutions $\psi(b, a, y)$ and $\psi(b^p, a, y)^p$. It is an assumption on the projection error.

Assumption 4.2 *There exist $\epsilon_1 > 0$ and $\delta_1 > 0$ such that for all $a \in A, y \in O$ and $b \in B$,*

$$|\langle g(\cdot, a), b - b^p \rangle| \leq \epsilon_1,$$

$$|\langle g(\cdot, a), \psi(b, a, y) - \psi(b^p, a, y)^p \rangle| \leq \delta_1.$$

The following assumption can be seen as a continuity property of the value function.

Assumption 4.3 *Given $\delta > 0$ that satisfies $|\langle g(\cdot, a), b - b' \rangle| \leq \delta$, there exists $\epsilon > 0$ such that $|J_k(b) - J_k(b')| \leq \epsilon, \forall b, b' \in B, \forall k$, and there exists a $\tilde{\epsilon} > 0$ such that $|J_\mu(b) - J_\mu(b')| \leq \tilde{\epsilon}, \forall b, b' \in B, \forall \mu \in \Pi$.*

Now we present our main result.

Theorem 4.1 *Under Assumptions 4.1, 4.2 and 4.3, for all $b \in B$,*

$$|J_*(b) - J_*^p(b^p)| \leq \frac{\epsilon_1 + \gamma\epsilon_2}{1 - \gamma}, \quad (4.7)$$

$$|J_*(b) - J_{\bar{\mu}^*}^p(b)| \leq \frac{2\epsilon_1 + \gamma(\epsilon_2 + \epsilon_3)}{1 - \gamma}, \quad (4.8)$$

where ϵ_1 is the constant in Assumption 4.2, and ϵ_2, ϵ_3 are the constants ϵ and $\tilde{\epsilon}$, respectively, in Assumption 4.3 corresponding to $\delta = \delta_1$.

Proof:

Denote $J_k(b) \triangleq T^k J_0(b), J_k^p(b^p) \triangleq (T^p)^k J_0(b^p), k = 0, 1, \dots$, and define

$$b_k(b, a) = \langle g(\cdot, a), b \rangle + \gamma E_Y \{J_{k-1}(\psi(b, a, Y))\},$$

$$\mu_k(b) = \arg \min_{a \in A} Q_k(b, a),$$

$$b_k^p(b, a) = \langle g(\cdot, a), b^p \rangle + \gamma E_Y \{J_{k-1}(\psi(b^p, a, Y)^p)\},$$

$$\mu_k^p(b) = \arg \min_{a \in A} Q_k^p(b, a).$$

Hence,

$$J_k(b) = \min_{a \in A} Q_k(b, a) = Q_k(b, \mu_k(b)),$$

$$J_k^p(b^p) = \min_{a \in A} Q_k^p(b, a) = Q_k^p(b, \mu_k^p(b)).$$

Denote $err_k \triangleq \max_{b \in B} |J_k(b) - J_k^p(b^p)|, k = 1, 2, \dots$

We consider the first iteration. Initialize with $J_0(b) = J_0^p(b^p) = 0$. By Assumption 4.2, $\forall a \in A$,

$$|Q_1(b, a) - Q_1^p(b, a)| = |\langle g(\cdot, a), b - b^p \rangle| \leq \epsilon_1, \quad \forall b \in B. \quad (4.9)$$

Hence, with $a = \mu_1^p(b)$, the above inequality yields $Q_1(b, \mu_1^p(b)) \leq J_1^p(b^p) + \epsilon_1$. Using $J_1(b) \leq Q_1(b, \mu_1^p(b))$, we get

$$J_1(b) \leq J_1^p(b^p) + \epsilon_1, \quad \forall b \in B. \quad (4.10)$$

With $a = \mu_1(b)$, (4.9) yields $Q_1^p(b, \mu_1(b)) - \epsilon_1 \leq J_1(b)$. Using $J_1^p(b^p) \leq Q_1^p(b, \mu_1(b))$, we get

$$J_1^p(b^p) - \epsilon_1 \leq J_1(b), \quad \forall b \in B. \quad (4.11)$$

From (4.10) and (4.11), we conclude

$$|J_1(b) - J_1^p(b^p)| \leq \epsilon_1, \quad \forall b \in B.$$

Taking the maximum over b on both sides of the above inequality yields

$$err_1 \leq \epsilon_1. \quad (4.12)$$

Now we consider the $(k+1)^{th}$ iteration. For a fixed $Y = y$, by Assumption 4.2,

$$|\langle g(\cdot, a), \psi(b, a, y) - \psi(b^p, a, y)^p \rangle| \leq \delta_1.$$

Let δ_1 be the δ in Assumption 4.3 and denote the corresponding ϵ by ϵ_2 . Then

$$|J_k(\psi(b, a, y)) - J_k(\psi(b^p, a, y)^p)| \leq \epsilon_2, \quad \forall b \in B. \quad (4.13)$$

Therefore, $\forall a \in A$,

$$\begin{aligned} & |Q_{k+1}(b, a) - Q_{k+1}^p(b, a)| \\ & \leq |\langle g(\cdot, a), b - b^p \rangle| + \gamma E_Y \{|J_k(\psi(b, a, Y)) - J_k^p(\psi(b^p, a, Y)^p)|\} \\ & \leq \epsilon_1 + \gamma E_Y \{|J_k(\psi(b, a, Y)) - J_k(\psi(b^p, a, Y)^p)| + |J_k(\psi(b^p, a, Y)^p) - J_k^p(\psi(b^p, a, Y)^p)|\} \\ & \leq \epsilon_1 + \gamma(\epsilon_2 + err_k), \quad \forall b \in B. \end{aligned}$$

The third inequality follows from (4.13) and the definition of err_k . Using an argument similar to that used to prove (4.12) from (4.9), we conclude that

$$err_{k+1} \leq \epsilon_1 + \gamma(\epsilon_2 + err_k). \quad (4.14)$$

Using induction on (4.14) with initial condition (4.12) and taking $k \rightarrow \infty$, we obtain

$$\begin{aligned} |J_*(b) - J_*^p(b^p)| & \leq \sum_{k=0}^{\infty} \gamma^k \epsilon_1 + \sum_{k=1}^{\infty} \gamma^k \epsilon_2 \\ & = \frac{\epsilon_1 + \gamma \epsilon_2}{1 - \gamma}. \end{aligned} \quad (4.15)$$

Therefore, (4.7) is proved.

Fixing a policy μ on the original belief MDP, define the mappings under policy μ on the belief MDP and the projected belief MDP as

$$T_\mu J(b) = \langle g(\cdot, \mu(b)), b \rangle + \gamma E_Y \{J(\psi(b, \mu(b), Y))\}, \quad (4.16)$$

$$T_\mu^p J(b) = \langle g(\cdot, \mu(b)), b^p \rangle + \gamma E_Y \{J(\psi(b^p, \mu(b), Y)^p)\}, \quad (4.17)$$

respectively. Since μ_*^p is a stationary policy for the projected belief MDP, $\bar{\mu}_*^p = \mu_*^p \circ Proj_\Omega$ is stationary for the original belief MDP. Hence,

$$J_*^p(b^p) = T_{\mu_*^p}^p J_*^p(b^p),$$

$$J_{\bar{\mu}_*^p}(b) = T_{\bar{\mu}_*^p} J_{\bar{\mu}_*^p}(b).$$

Subtracting both sides of the above two equations, and substituting in the definitions of T^p and T (i.e., (4.17) and (4.16)) for the righthand sides respectively, we get

$$\begin{aligned} J_*^p(b^p) - J_{\bar{\mu}_*^p}(b) &= \langle g(\cdot, \mu_*^p(b^p)), b^p - b \rangle \dots \\ &+ \gamma E_Y \{ J_*^p(\psi(b^p, \mu_*^p(b^p), Y)^p) - J_{\bar{\mu}_*^p}(\psi(b, \mu_*^p(b^p), Y)) \} \end{aligned} \quad (4.18)$$

For a fixed $Y = y$,

$$\begin{aligned} &| J_*^p(\psi(b^p, \mu_*^p(b^p), y)^p) - J_{\bar{\mu}_*^p}(\psi(b, \mu_*^p(b^p), y)) | \\ &\leq \left| J_*^p(\tilde{b}) - J_{\bar{\mu}_*^p}(\tilde{b}) \right| + \left| J_{\bar{\mu}_*^p}(\psi(b^p, \mu_*^p(b^p), y)^p) - J_{\bar{\mu}_*^p}(\psi(b, \mu_*^p(b^p), y)) \right|, \end{aligned}$$

where $\tilde{b} = \psi(b^p, \mu_*^p(b^p), y)^p \in B$. By Assumption 4.2,

$$|\langle g(\cdot, a), \psi(b^p, \mu_*^p(b^p), y)^p - \psi(b, \mu_*^p(b^p), y) \rangle| \leq \delta_1.$$

Letting $\delta = \delta_1$ in Assumption 4.3 and denoting the corresponding $\tilde{\epsilon}$ by ϵ_3 , we get the second term

$$\left| J_{\bar{\mu}_*^p}(\psi(b^p, \mu_*^p(b^p), y)^p) - J_{\bar{\mu}_*^p}(\psi(b, \mu_*^p(b^p), y)) \right| \leq \epsilon_3.$$

Denoting $err \triangleq \max_{b \in B} |J_*^p(b^p) - J_{\mu_*^p}(b)|$, we obtain

$$\left| J_*^p(\psi(b^p, \mu_*^p(b^p), y)^p) - J_{\bar{\mu}_*^p}(\psi(b, \mu_*^p(b^p), y)) \right| \leq err + \epsilon_3.$$

Therefore, (4.18) becomes

$$|J_*^p(b^p) - J_{\bar{\mu}_*^p}(b)| \leq \epsilon_1 + \gamma(err + \epsilon_3).$$

Taking the maximum over b on both sides of the above inequality yields

$$err \leq \epsilon_1 + \gamma(err + \epsilon_3).$$

Hence,

$$err \leq \frac{\epsilon_1 + \gamma\epsilon_3}{1 - \gamma}. \quad (4.19)$$

With (4.15) and (4.19), we obtain

$$\begin{aligned} |J_*(b) - J_{\bar{\mu}_*^p}(b)| &\leq |J_*(b) - J_*^p(b^p)| + |J_*^p(b^p) - J_{\bar{\mu}_*^p}(b)| \\ &\leq \frac{2\epsilon_1 + \gamma(\epsilon_2 + \epsilon_3)}{1 - \gamma}, \quad \forall b \in B. \end{aligned}$$

Therefore, (4.8) is proved. \blacksquare

Remark 4.2 *In (4.7) and (4.8), ϵ_1 is a projection error, and ϵ_2 and ϵ_3 decreases as the projection error δ_1 decreases. Therefore, as the projection error decreases, the optimal value function of the projected belief MDP $J_*^p(b^p)$ converges to the true optimal value function $J_*(b)$, and the corresponding policy $\bar{\mu}_*^p$ converges to the true optimal policy μ_* . Roughly speaking, the projection error decreases as the number of sufficient statistics in the chosen exponential family increases (for details, please see Section 4.6: Validation of the Assumptions).*

4.5 Analysis of Projection Particle Filtering

In the above analysis, we assumed perfect computation of the belief states and the projected belief states. In this section, we consider the filtering error, and

compute an error bound on the approximate belief state generated by the projection particle filter (PPF).

4.5.1 Notations

Let $C_b(\mathbb{R}^n)$ be the set of all continuous bounded functions on \mathbb{R}^n . Let $B(\mathbb{R}^n)$ be the set of all bounded measurable functions on \mathbb{R}^n . Let $\|\cdot\|$ denote the supremum norm on $B(\mathbb{R}^n)$, i.e., $\|\phi\| \triangleq \sup_{x \in \mathbb{R}^n} |\phi(x)|$, $\phi \in B(\mathbb{R}^n)$. Let $\mathcal{M}^+(\mathbb{R}^n)$ and $\mathcal{P}(\mathbb{R}^n)$ be the sets of nonnegative measures and probability measures on \mathbb{R}^n , respectively. If $\eta \in \mathcal{M}^+(\mathbb{R}^n)$ and $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is an integrable function with respect to η , then

$$\langle \eta, \phi \rangle \triangleq \int \phi d\eta.$$

Moreover, if $\eta \in \mathcal{P}(\mathbb{R}^n)$,

$$E_\eta[\phi] = \langle \eta, \phi \rangle,$$

$$\text{Var}_\eta(\phi) = \langle \eta, \phi^2 \rangle - \langle \eta, \phi \rangle^2.$$

We will use the representations on the two sides of the above equalities interchangeably in the sequel.

The belief state and the projected belief state are probability densities; however, we will prove our results in terms of their corresponding probability measures, which we refer to as “conditional distributions” (belief states are conditional densities). The two representations are essentially the same once we assume the probability measures admit probability densities. Therefore, the notations used for probability densities before are used to denote their corresponding probability measures from now on. Namely, we use b to denote a probability measure on \mathbb{R}^{n_x} and

assume it admits a probability density with respect to Lebesgue measure, which is the belief state. Similarly, we use $f(\cdot, \theta)$ to denote a probability measure on \mathbb{R}^{n_x} and assume it admits a probability density with respect to Lebesgue measure in the chosen exponential family with parameter θ .

A probability transition kernel $K : \mathcal{P}(\mathbb{R}^{n_x}) \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is defined by

$$K\eta(E) \triangleq \int_{\mathbb{R}^{n_x}} \eta(dx) K(E, x),$$

where E is a set in the Borel σ -algebra on \mathbb{R}^{n_x} . For $\phi : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$, an integrable function with respect to $K(\cdot, x)$,

$$K\phi(x) \triangleq \int_{\mathbb{R}^{n_x}} \phi(x') K(dx', x).$$

Let $K_k(dx_k, x_{k-1})$ denote the probability transition kernel of the system (3.2) at time k , which satisfies

$$\begin{aligned} b_{k|k-1}(dx_k) &= K_k b_{k-1}(dx_{k|k-1}) \\ &= \int_{\mathbb{R}^{n_x}} b_{k-1}(dx_{k-1}) K_k(dx_{k|k-1}, x_{k-1}). \end{aligned}$$

We let Ψ_k denote the likelihood function associated with the observation equation (3.3) at time k , and assume that $\Psi_k \in C_b(\mathbb{R}^{n_x})$. Hence,

$$b_k = \frac{\Psi_k b_{k|k-1}}{\langle b_{k|k-1}, \Psi_k \rangle}.$$

4.5.2 Main Idea

The exact filter (EF) at time k can be described as

$$\begin{array}{ccc} b_{k-1} & \longrightarrow & b_{k|k-1} = K_k b_{k-1} & \longrightarrow & b_k = \frac{\Psi_k b_{k|k-1}}{\langle b_{k|k-1}, \Psi_k \rangle} \\ \textit{prediction} & & & & \textit{updating} \end{array}$$

The PPF at time k can be described as

$$\begin{aligned}
\hat{f}(\cdot, \hat{\theta}_{k-1}) &\longrightarrow \hat{b}_{k|k-1} = K_k f(\cdot, \hat{\theta}_{k-1}) \longrightarrow \dots \\
&\quad \textit{prediction} \qquad \qquad \qquad \textit{updating} \\
\hat{b}_k &= \frac{\Psi_k \hat{b}_{k|k-1}}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle} \longrightarrow f(\cdot, \hat{\theta}_k) \longrightarrow \hat{f}(\cdot, \hat{\theta}_k). \\
&\quad \textit{projection} \qquad \qquad \textit{resampling}
\end{aligned}$$

To facilitate our analysis, we introduce a conceptual filter (CF), which at each time k is reinitialized by $f(\cdot, \hat{\theta}_{k-1})$, performs exact prediction and updating to yield $b'_{k|k-1}$ and b'_k , respectively, and does projection to obtain $f(\cdot, \theta'_k)$. It can be described as

$$\begin{aligned}
f(\cdot, \hat{\theta}_{k-1}) &\longrightarrow b'_{k|k-1} = K_k f(\cdot, \hat{\theta}_{k-1}) \longrightarrow \dots \\
&\quad \textit{prediction} \qquad \qquad \qquad \textit{updating} \\
b'_k &= \frac{\Psi_k b'_{k|k-1}}{\langle b'_{k|k-1}, \Psi_k \rangle} \longrightarrow f(\cdot, \theta'_k). \\
&\quad \textit{projection}
\end{aligned}$$

The CF serves as an bridge to connect the EF and PPF, as we describe below.

We are interested in the difference between the true conditional distribution b_k and the PPF generated projected conditional distribution $f(\cdot, \hat{\theta}_k)$ for each time k . The total error between the two can be decomposed as follows:

$$b_k - f(\cdot, \hat{\theta}_k) = (b_k - b'_k) + (b'_k - f(\cdot, \theta'_k)) + (f(\cdot, \theta'_k) - f(\cdot, \hat{\theta}_k)). \quad (4.20)$$

The first term $(b_k - b'_k)$ is the error due to the inexact initial condition of the CF compared to EF, i.e., $(b_{k-1} - f(\cdot, \hat{\theta}_{k-1}))$, which is also the total error at time $k - 1$.

Table 4.1: Notations of Different Conditional Distributions

b_k	exact conditional distribution
\hat{b}_k	PPF conditional distribution before projection
$f(\cdot, \hat{\theta}_k)$	PPF projected conditional distribution
b'_k	CF conditional distribution before projection
$f(\cdot, \theta'_k)$	CF projected conditional distribution

The second term $(b'_k - f(\cdot, \theta'_k))$ evaluates the minimum deviation from the exponential family generated by one step of exact filtering, since $f(\cdot, \theta'_k)$ is the projection of b'_k . The third term $(f(\cdot, \theta'_k) - f(\cdot, \hat{\theta}_k))$ is purely due to Monte Carlo simulation, since $f(\cdot, \theta'_k)$ and $f(\cdot, \hat{\theta}_k)$ are obtained using the same steps from $f(\cdot, \hat{\theta}_{k-1})$ and its empirical version $\hat{f}(\cdot, \hat{\theta}_{k-1})$, respectively. We will find error bounds on each of the three terms, and finally find the total error at time k by induction.

4.5.3 Error Bound

We shall look at the the case in which the observation process has an arbitrary but fixed value $y_{0:k} = \{y_0, \dots, y_k\}$. Hence, all the expectations E in this section are with respect to the sampling in the algorithm only. We consider the test function

$\phi \in B(\mathbb{R}^{n_x})$. It can be seen that $K\phi \in B(\mathbb{R}^{n_x})$ and $\|K\phi\| \leq \|\phi\|$, since

$$\begin{aligned} |K\phi(x)| &= \left| \int_{\mathbb{R}^{n_x}} \phi(x')K(dx', x) \right| \\ &\leq \int_{\mathbb{R}^{n_x}} |\phi(x')K(dx', x)| \\ &\leq \|\phi\| \int_{\mathbb{R}^{n_x}} K(dx', x) = \|\phi\|. \end{aligned}$$

Since $\Psi \in C_b(\mathbb{R}^{n_x})$, we know that $\Psi \in B(\mathbb{R}^{n_x})$ and $\Psi\phi \in B(\mathbb{R}^{n_x})$.

We also need the following assumptions.

Assumption 4.4 *All the projected distributions are in a compact subset of the given exponential family. In other words, there exists a compact set $\Theta' \subseteq \Theta$ such that $\hat{\theta}_k \in \Theta'$, and $\theta'_k \in \Theta'$, $\forall k$.*

Assumption 4.5 *For all $k \in \mathbb{N}$,*

$$\begin{aligned} \langle b_{k|k-1}, \Psi_k \rangle &> 0, \\ \langle b'_{k|k-1}, \Psi_k \rangle &> 0, \quad w.p.1, \\ \langle \hat{b}_{k|k-1}, \Psi_k \rangle &> 0, \quad w.p.1. \end{aligned}$$

Assumption 4.5 guarantees that the normalizing constant in the Bayes' updating is nonzero, so that the conditional distribution is well defined. Under Assumption 4.4, the second inequality in Assumption 4.5 can be strengthened using the compactness of Θ' . Since $f(\cdot, a_k, u_k)$ in (3.2) is continuous in x , K_k is weakly continuous (pp. 175-177, [37]). Hence, $\langle b'_{k|k-1}, \Psi_k \rangle = \langle K_k f(\cdot, \hat{\theta}_{k-1}), \Psi_k \rangle = \langle f(\cdot, \hat{\theta}_{k-1}), K_k \Psi_k \rangle$ is continuous in $\hat{\theta}_{k-1}$, where $\hat{\theta}_{k-1} \in \Theta'$. Since Θ' is compact, there exists a constant $\delta > 0$ such that for each k

$$\langle b'_{k|k-1}, \Psi_k \rangle \geq \frac{1}{\delta}, \quad w.p.1. \quad (4.21)$$

The assumption below guarantees that the conditional distribution stays close to the given exponential family after one step of exact filtering if the initial distribution is in the exponential family. Recall that starting with initial distribution $f(\cdot, \hat{\theta}_{k-1})$, one step of exact filtering yields b'_k , which is then projected to yield $f(\cdot, \theta'_k)$, where $\hat{\theta}_{k-1} \in \Theta'$, $\theta'_k \in \Theta'$.

Assumption 4.6 *There exists a constant $\epsilon > 0$ such that for all $\phi \in B(\mathbb{R}^{n_x})$ and all $k \in \mathbb{N}$,*

$$E[|\langle b'_k, \phi \rangle - \langle f(\cdot, \theta'_k), \phi \rangle|] \leq \epsilon \|\phi\|.$$

Remark 4.3 *Assumption 4.6 is our main assumption, which essentially assumes an error bound on the projection error. Our assumptions are much less restrictive than the assumptions in [5], while our conclusion is similar to but slightly different from that in [5], which will be seen later. Although Assumption 4.6 appears similar to Assumption 3 in [5], it is essentially different. Assumption 3 in [5] says that the optimal conditional density stays close to the given exponential family for all time, whereas Assumption 4.6 only assumes that if the exact filter starts in the given exponential family, after one step the conditional distribution stays close to the family. Moreover, we do not need any assumption like the restrictive Assumption 4 in [5].*

Lemma 4.1 considers the bound on the first term in (4.20).

Lemma 4.1 *For each $k \in \mathbb{N}$, suppose $E[|\langle b_{k-1} - f(\cdot, \hat{\theta}_{k-1}), \phi \rangle|] \leq e_{k-1} \|\phi\|$, $\forall \phi \in B(\mathbb{R}^{n_x})$, where e_{k-1} is a positive constant. Then under Assumptions 4.4 and 4.5,*

for each $k \in \mathbb{N}$, there exists a constant $\zeta_k > 0$ such that for all $\phi \in B(\mathbb{R}^{n_x})$

$$E [|\langle b_k - b'_k, \phi \rangle|] \leq \zeta_k e_{k-1} \|\phi\|. \quad (4.22)$$

Proof:

$E \left[\left| \langle b_{k-1} - f(\cdot, \hat{\theta}_{k-1}), \phi \rangle \right| \right]$ is the error from time $k-1$, which is also the initial error for time k . Hence, the prediction step yields

$$\begin{aligned} & E \left[\left| \langle b_{k|k-1} - b'_{k|k-1}, \phi \rangle \right| \right] \\ &= E \left[\left| \langle K_k(b_{k-1} - f(\cdot, \hat{\theta}_{k-1})), \phi \rangle \right| \right] \\ &= E \left[\left| \langle b_{k-1} - f(\cdot, \hat{\theta}_{k-1}), K_k \phi \rangle \right| \right] \\ &\leq e_{k-1} \|K_k \phi\| \\ &\leq e_{k-1} \|\phi\|. \end{aligned} \quad (4.23)$$

Under Assumptions 4.4 and 4.5, we have showed (4.21). Using (4.21) and (4.23), the Bayes' updating step yields

$$\begin{aligned} & E [|\langle b_k - b'_k, \phi \rangle|] \\ &= E \left[\left| \frac{\langle b_{k|k-1}, \Psi_k \phi \rangle}{\langle b_{k|k-1}, \Psi_k \rangle} - \frac{\langle b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\ &= E \left[\left| \frac{\langle b_{k|k-1}, \Psi_k \phi \rangle}{\langle b_{k|k-1}, \Psi_k \rangle} - \frac{\langle b_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] + E \left[\left| \frac{\langle b_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} - \frac{\langle b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\ &\leq E \left[\left| \frac{\langle b_{k|k-1}, \Psi_k \phi \rangle \langle b_{k|k-1} - b'_{k|k-1}, \Psi_k \rangle}{\langle b_{k|k-1}, \Psi_k \rangle \langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] + E \left[\left| \frac{\langle b_{k|k-1} - b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\ &\leq \delta \frac{|\langle b_{k|k-1}, \Psi_k \phi \rangle|}{\langle b_{k|k-1}, \Psi_k \rangle} E [|\langle b_{k|k-1} - b'_{k|k-1}, \Psi_k \rangle|] + \delta E [|\langle b_{k|k-1} - b'_{k|k-1}, \Psi_k \phi \rangle|] \\ &\leq \delta e_{k-1} \|\phi\| \|\Psi_k\| + \delta e_{k-1} \|\Psi_k \phi\| \\ &\leq 2\delta e_{k-1} \|\Psi_k\| \|\phi\| = \zeta_k e_{k-1} \|\phi\|, \end{aligned}$$

where $\zeta_k = 2\delta\|\Psi_k\|$. \blacksquare

Lemma 4.2 considers the bound on the third term in (4.20) before projection.

Lemma 4.2 *Under Assumptions 4.4 and 4.5, for each $k \in \mathbb{N}$, there exists a constant $\tau_k > 0$ such that for all $\phi \in B(\mathbb{R}^{n_x})$*

$$E \left[\left| \langle \hat{b}_k - b'_k, \phi \rangle \right| \right] \leq \tau_k \frac{\|\phi\|}{\sqrt{N}}.$$

Proof:

This lemma uses essentially the same proof technique as Lemmas 3 and 4 in [23]. However, it is not quite obvious how these lemmas imply our lemma here. Therefore, we state the proof to make this chapter more accessible. After the re-sampling step, $\hat{f}(\cdot, \hat{\theta}_{k-1}) = \frac{1}{N} \sum_{i=1}^N \delta(x - x_{k-1}^i)$, where $x_{k-1}^i, i = 1, \dots, N$ are i.i.d. samples from $f(\cdot, \hat{\theta}_{k-1})$. Using the Cauchy-Schwartz inequality, we have

$$\begin{aligned} & \left(E \left[\langle \hat{f}(\cdot, \hat{\theta}_{k-1}) - f(\cdot, \hat{\theta}_{k-1}), \phi \rangle^2 \right] \right)^{1/2} \\ &= \left(E \left[\left(\frac{1}{N} \sum_{i=1}^N (\phi(x_{k-1}^i) - \langle f(\cdot, \hat{\theta}_{k-1}), \phi \rangle) \right)^2 \right] \right)^{1/2} \\ &= \frac{1}{\sqrt{N}} \left(E \left[\frac{1}{N} \sum_{i=1}^N (\phi(x_{k-1}^i) - \langle f(\cdot, \hat{\theta}_{k-1}), \phi \rangle)^2 \right] \right)^{1/2} \\ &= \frac{1}{\sqrt{N}} \left(\langle f(\cdot, \hat{\theta}_{k-1}), \phi^2 \rangle - \langle f(\cdot, \hat{\theta}_{k-1}), \phi \rangle^2 \right)^{1/2} \\ &\leq \frac{1}{\sqrt{N}} \langle f(\cdot, \hat{\theta}_{k-1}), \phi^2 \rangle^{1/2} \\ &\leq \frac{1}{\sqrt{N}} \|\phi\|. \end{aligned} \tag{4.24}$$

The Bayes' updating step yields

$$\begin{aligned}
& E \left[\left| \langle \hat{b}_k - b'_k, \phi \rangle \right| \right] \\
&= E \left[\left| \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle} - \frac{\langle b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\
&= E \left[\left| \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle} - \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] + E \left[\left| \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} - \frac{\langle b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\
&\leq E \left[\left| \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle \langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \rangle}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle \langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] + E \left[\left| \frac{\langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right].
\end{aligned}$$

Under Assumptions 4.4 and 4.5, we have showed (4.21). Using the Cauchy-Schwartz

inequality, (4.21) and (4.24), the first term can be simplified as

$$\begin{aligned}
& E \left[\left| \frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle \langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \rangle}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle \langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\
&\leq \delta \left(E \left[\frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle^2}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle^2} \right] \right)^{1/2} \left(E \left[\langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \rangle^2 \right] \right)^{1/2} \\
&= \delta \left(E \left[\frac{\langle \hat{b}_{k|k-1}, \Psi_k \phi \rangle^2}{\langle \hat{b}_{k|k-1}, \Psi_k \rangle^2} \right] \right)^{1/2} \left(E \left[\langle f(\cdot, \hat{\theta}'_{k-1}) - f(\cdot, \theta'_{k-1}), K_k \Psi_k \rangle^2 \right] \right)^{1/2} \\
&\leq \delta \|\phi\| \frac{1}{\sqrt{N}} \|\Psi_k\|,
\end{aligned}$$

and the second term can be simplified as

$$\begin{aligned}
& E \left[\left| \frac{\langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \phi \rangle}{\langle b'_{k|k-1}, \Psi_k \rangle} \right| \right] \\
&\leq \delta \left(E \left[\langle \hat{b}_{k|k-1} - b'_{k|k-1}, \Psi_k \phi \rangle^2 \right] \right)^{1/2} \\
&= \delta \left(E \left[\langle \hat{f}(\cdot, \hat{\theta}_{k-1}) - f(\cdot, \hat{\theta}_{k-1}), K_k \Psi_k \phi \rangle^2 \right] \right)^{1/2} \\
&\leq \delta \frac{1}{\sqrt{N}} \|\Psi_k \phi\| \\
&\leq \delta \frac{1}{\sqrt{N}} \|\Psi_k\| \|\phi\|.
\end{aligned}$$

Therefore, adding these two terms yields

$$E \left[\left| \langle \hat{b}_k - b'_k, \phi \rangle \right| \right] \leq 2\delta \|\Psi_k\| \frac{\|\phi\|}{\sqrt{N}} = \tau_k \frac{\|\phi\|}{\sqrt{N}},$$

where $\tau_k = 2\delta\|\Psi_k\|$. \blacksquare

Lemma 4.3 considers the bound on the third term in (4.20) based on the result of Lemma 4.2. The key idea of proof is to connect the errors before and after projection through (4.4), which we derived for the density projection that minimizes the KL divergence.

Lemma 4.3 *Let $c_j, j = 1, \dots, m$ be the sufficient statistics of the exponential family as defined in Definition 4.1, and assume $c_j \in B(\mathbb{R}^{n_x}), j = 1, \dots, m$. Then under Assumptions 4.4 and 4.5, for each $k \in \mathbb{N}$, there exists a constant $d_k > 0$ such that for all $\phi \in B(\mathbb{R}^{n_x})$*

$$E \left[\left| f(\cdot, \hat{\theta}_k) - \langle f(\cdot, \theta'_k), \phi \rangle \right| \right] \leq d_k \frac{\|\phi\|}{\sqrt{N}}. \quad (4.25)$$

Proof:

The key idea of the proof for Lemma 4 in [5] is used here. From (4.4), we know that $E_{\hat{\theta}_k}[c_j(X)] = E_{\hat{b}_k}[c_j(X)]$ and $E_{\theta'_k}[c_j(X)] = E_{b'_k}[c_j(X)]$. Hence, we obtain

$$E \left[\left| E_{\hat{\theta}_k}(c_j(X)) - E_{\theta'_k}(c_j(X)) \right| \right] = E \left[\left| \langle \hat{b}_k - b'_k, c_j \rangle \right| \right],$$

for $j = 1, \dots, m$. Taking summation over j , we obtain

$$E \left[\sum_{j=1}^m \left| E_{\hat{\theta}_k}(c_j(X)) - E_{\theta'_k}(c_j(X)) \right| \right] = \sum_{j=1}^m E \left[\left| \langle \hat{b}_k - b'_k, c_j \rangle \right| \right].$$

Since $c_j \in B(\mathbb{R}^{n_x})$, we apply Lemma 4.2 with $\phi = c_j$ and thus obtain

$$E \left[\left| \langle \hat{b}_k - b'_k, c_j \rangle \right| \right] \leq b_k \frac{\|c_j\|}{\sqrt{N}}, \quad j = 1, \dots, m.$$

Therefore,

$$E \left[\left\| E_{\hat{\theta}_k}(c(X)) - E_{\theta'_k}(c(X)) \right\|_1 \right] \leq \frac{\tilde{\tau}_k}{\sqrt{N}},$$

where $\|\cdot\|_1$ denotes the L_1 norm on \mathbb{R}^{n_x} , $c = [c_1, \dots, c_m]^T$, and $\tilde{\tau}_k = \sum_{j=1}^m \|c_j\|$.

Since Θ' is compact and the Fisher information matrix

$[E_\theta [c_i(X)c_j(X)] - E_\theta [c_i(X)] E_\theta [c_j(X)]]_{ij}$ is positive definite, we get (cf. Fact 2 in [5] for a detailed proof)

$$\left\| \hat{\theta}_k - \theta'_k \right\|_1 \leq \alpha \left\| E_{\hat{\theta}_k}(c(X)) - E_{\theta'_k}(c(X)) \right\|_1.$$

Taking the expectation on both sides yields

$$\begin{aligned} E \left[\left\| \hat{\theta}_k - \theta'_k \right\|_1 \right] &\leq \alpha E \left[\left\| E_{\hat{\theta}_k}(c(X)) - E_{\theta'_k}(c(X)) \right\|_1 \right] \\ &\leq \alpha \tilde{\tau}_k \frac{1}{\sqrt{N}}. \end{aligned}$$

On the other hand, taking derivative of $E_\theta[\phi(X)]$ with respect to θ_i yields

$$\begin{aligned} \left| \frac{d}{d\theta_i} E_\theta[\phi(X)] \right| &= |E_\theta[c_i(X)\phi(X)] - E_\theta[c_i(X)]E_\theta[\phi(X)]| \\ &\leq \sqrt{\text{Var}_\theta(c_i)\text{Var}_\theta(\phi)} \\ &\leq \sqrt{E_\theta(c_i^2)E_\theta(\phi^2)} \\ &\leq \|c_i\| \|\phi\|. \end{aligned}$$

Hence,

$$\left\| \frac{d}{d\theta} E_\theta[\phi(X)] \right\|_1 \leq \left(\sum_{i=1}^m \|c_i\| \right) \|\phi\|.$$

Since Θ' is compact, there exists a constant $\beta > 0$ such that $E_\theta[\phi(X)]$ is Lipschitz over $\theta \in \Theta'$ with Lipschitz constant $\beta\|\phi\|$ (cf. the proof of Fact 2 in [5]), i.e.,

$$|E_{\hat{\theta}_k}[\phi] - E_{\theta'_k}[\phi]| \leq \beta \|\phi\| \left\| \hat{\theta}_k - \theta'_k \right\|_1.$$

Taking expectation on both sides yields

$$\begin{aligned} E \left[\left| \langle f(\cdot, \hat{\theta}_k) - f(\cdot, \theta'_k), \phi \rangle \right| \right] &\leq \beta \|\phi\| E \left[\left\| \hat{\theta}_k - \theta'_k \right\|_1 \right] \\ &\leq \beta \|\phi\| \alpha \tilde{\tau}_k \frac{1}{\sqrt{N}} = d_k \frac{\|\phi\|}{\sqrt{N}}, \end{aligned}$$

where $d_k = \alpha \beta \tilde{\tau}_k$. \blacksquare

Now we present our main result on the error bound of the projection particle filter.

Theorem 4.2 *Suppose $E[\langle b_0 - f(\cdot, \hat{\theta}_0), \phi \rangle] \leq e_0 \|\phi\|$, $e_0 \geq 0$, $\forall \phi \in B(\mathbb{R}^{n_x})$. Under Assumptions 4.4, 4.5 and 4.6, and assuming that $c_j \in B(\mathbb{R}^{n_x})$, $j = 1, \dots, m$, there exist $\zeta_i > 0$, $d_i > 0$, $i = 1, \dots, k$ such that for all $\phi \in B(\mathbb{R}^{n_x})$,*

$$E \left[\left| \langle b_k - f(\cdot, \hat{\theta}_k), \phi \rangle \right| \right] \leq e_k \|\phi\|, \quad k = 1, 2, \dots,$$

where

$$e_k = \zeta_1^k e_0 + \left(\sum_{i=2}^k \zeta_i^k + 1 \right) \epsilon + \left(\sum_{i=2}^k \zeta_i^k d_{i-1} + d_k \right) \frac{1}{\sqrt{N}}, \quad (4.26)$$

$\zeta_i^k = \prod_{j=i}^k \zeta_j$ for $k \geq i$, and $\zeta_i^k = 0$ for $k < i$, ϵ is the constant in Assumption 4.6.

Proof:

Applying Lemma 4.1, Assumption 4.6, and Lemma 4.3, we have that for all $\phi \in B(\mathbb{R}^{n_x})$ and $k \in \mathbb{N}$, there exist $\zeta_i > 0$, $d_i > 0$, $i = 1, \dots, k$ such that

$$\begin{aligned} &E \left[\left| \langle b_k - f(\cdot, \hat{\theta}_k), \phi \rangle \right| \right] \\ &\leq E \left[\left| \langle b_k - b'_k, \phi \rangle \right| \right] + E \left[\left| \langle b'_k - f(\cdot, \theta'_k), \phi \rangle \right| \right] \dots \\ &\quad + E \left[\left| \langle f(\cdot, \theta'_k) - f(\cdot, \hat{\theta}_k), \phi \rangle \right| \right] \\ &\leq \left(\zeta_k e_{k-1} + \epsilon + d_k \frac{1}{\sqrt{N}} \right) \|\phi\| = e_k \|\phi\|. \end{aligned}$$

It is easy to deduce by induction that

$$e_k = \zeta_1^k e_0 + \left(\sum_{i=2}^k \zeta_i^k + 1 \right) \epsilon + \left(\sum_{i=2}^k \zeta_i^k d_{i-1} + d_k \right) \frac{1}{\sqrt{N}}.$$

■

Remark 4.4 *As we mentioned in Remark 4.2, the projection error e_0 and ϵ decrease as the number of sufficient statistics in the chosen exponential family, m , increases. The error e_k decreases at the rate of $\frac{1}{\sqrt{N}}$, as we increase the number of samples in the projection particle filter. However, notice that the coefficient in front of $\frac{1}{\sqrt{N}}$ grows as time, so we have to use an increasing number of samples as time goes on, in order to ensure a uniform error bound with respect to time.*

4.6 Validation of Assumptions

Assumptions 4.2 and 4.6 are the main assumptions of our analysis. They are assumptions on the projection error, assuming that density projection introduces a “small” error. We will show that in certain cases these assumptions hold, and the projection error converges to 0 as the number of sufficient statistics, m , goes to infinity. We will first state a convergence result from [7]. However, as their convergence result is in the sense of KL divergence, we will further show the convergence in the sense employed in our assumptions by using an intermediate result in [7].

Consider a probability density function b defined on a bounded interval, and approximate it by b^p , a density function in an exponential family, whose sufficient statistics consist of polynomials, splines or trigonometric series. The following theorem is proved in [7].

Theorem 4.3 *If $\log b$ has r square-integrable derivatives, i.e., $\int |D^r \log b|^2 < \infty$, then $D_{KL}(b||b^p)$ converges to 0 at rate m^{-2r} as $m \rightarrow \infty$.*

Theorem 4.3 says the projected density b^p converges to b in the sense of KL divergence, as m goes to infinity. An intermediate result (see (6.6) in [7]) is:

$\|\log b/b^p\| \leq \nu_m$, where ν_m is a constant that depends on m , and $\nu_m \rightarrow 0$ as $m \rightarrow \infty$.

Since b is bounded and $\log(\cdot)$ is a continuously differentiable function, there exists a constant ξ such that $\|b - b^p\| \leq \xi \|\log b - \log b^p\|$. Hence, with the intermediate result above,

$$\begin{aligned} |\langle \phi, b - b^p \rangle| &\leq \|\phi\| \int \|b - b^p\| dx \\ &\leq \|\phi\| \int \xi \|\log \frac{b}{b^p}\| dx \leq \|\phi\| \xi l \nu_m, \end{aligned}$$

where l is the length of the bounded interval that b is defined on. Since ν_m can be made arbitrarily small by taking large enough m , it is easy to see that Assumptions 4.2 and 4.6 hold in the cases that we consider.

4.7 Numerical Experiments

4.7.1 Scalability and Computational Issues

Estimation of the one-step cost function (4.6) and transition probabilities (Algorithm 4.2) are executed for every belief-action pair that is in the discretized mesh G and the action space A . Hence, the algorithms scale according to $O(|G||A|N)$ and $O(|G||A|N^2)$, respectively, where $|G|$ is the number of grid points, $|A|$ is the

number of actions, and N is the number of samples specified in the algorithms. In implementation, we found that most of the computation time is spent on executing Algorithm 4.2 over all belief-action pairs. However, estimation of cost functions and transition probabilities can be pre-computed and stored, and hence only needs to be done once.

The advantage of the algorithms is that the scalability is independent of the size of the actual state space, since G is a grid mesh on the parameter space of the projected belief space. That is exactly what is desired by employing density projection. However, to get a better approximation, more parameters in the exponential family should be used, and that will lead to a higher-dimensional parameter space to discretize. Increasing the number of parameters in the exponential family also makes sampling more difficult. Sampling from a general exponential family is usually not easy, and may require some advanced techniques, such as Markov chain Monte Carlo (MCMC) [33], and hence more computation time. This difficulty can be avoided by resampling from the discrete particles instead of the projected density, which is equivalent to using the plain particle filter and then doing projection outside the filter. This may lead to sample degeneracy however. The trade-off between a better approximation and less computation time is complicated and needs more research. We plan to study how to appropriately choose the exponential family and improve the simulation efficiency in the future.

4.7.2 Simulation Results

Since most of the benchmark POMDP problems in the literature assume a discrete state space, it is difficult to compare against the state of the art. Here we consider an inventory control problem by adding a partial observation equation to a fully observable inventory control problem. The fully observable problem has an optimal threshold policy [60], which allows us to verify our method in the limiting case by setting the observation noise very close to 0. In our inventory control problem, the inventory level is reviewed at discrete times, and the observations are noisy because of, e.g., inventory spoilage, misplacement, distributed storage. At each period, inventory is either replenished by an order of a fixed amount or not replenished. The customer demands arrive randomly with known distribution. The demand is filled if there is enough inventory remaining. Otherwise, in the case of a shortage, excess demand is not satisfied and a penalty is issued on the lost sales amount. We assume that the demand and the observation noise are both continuous random variables; hence the state, i.e., the inventory level, and the observation, are continuous random variables.

Let x_k denote the inventory level at period k , u_k the i.i.d. random demand at period k , a_k the replenish decision at period k (i.e., $a_k = 0$ or 1), Q the fixed order amount, y_k the observation of inventory level x_k , v_k the i.i.d. observation noise, h the per period per unit inventory holding cost, s the per period per unit inventory

shortage penalty cost. The system equations are as follows

$$\begin{aligned} x_{k+1} &= \max(x_k + a_k Q - u_k, 0), \quad k = 0, 1, \dots, \\ y_k &= x_k + v_k, \quad k = 0, 1, \dots \end{aligned}$$

The cost incurred in period k is

$$g_k(x_k, a_k, u_k) = h \max(x_k + a_k Q - u_k, 0) + s \max(u_k - x_k - a_k Q, 0).$$

We consider two objective functions: average cost per period and discounted total cost, given by

$$\limsup_{H \rightarrow \infty} \frac{E \left[\sum_{k=0}^H g_k \right]}{H}, \quad \lim_{H \rightarrow \infty} E \left[\sum_{k=0}^H \gamma^k g_k \right],$$

where $\gamma \in (0, 1)$ is the discount factor.

In the simulation, we first choose an exponential family and specify a grid mesh on its parameter space, then implement (4.6) and Algorithm 4.2 on the grid mesh, and use value iteration to solve for a policy. These are done offline. In an online run, Algorithm 4.1 (PPF) is used for filtering and making decisions with the policy obtained offline. We also consider a small variation of this method: instead of using PPF, we use Algorithm 2.1 (PF) and do density projection outside the filter each time. We compare our two methods (called ‘‘Ours 1’’ and ‘‘Ours 2’’, respectively) described above to four other algorithms: (1) Certainty equivalence using the mean estimate (CE-Mean); (2) Certainty equivalence using the maximum likelihood estimate (CE-MLE); (3) EKF-based Parametric POMDP (EKF-PPOMDP) in [17]; (4) Greedy policy. Certainty equivalence methods treat the state estimate as the true state in the solution to the full observation problem. We use the bootstrap filter to

obtain the mean estimate and the MLE of the states for the certainty equivalence method. EKF-PPOMDP approximates the belief state by a Gaussian distribution, and uses the extended Kalman filter to estimate the transition of the belief state. Similar to our method, it also solves a discretized MDP defined on the parameter space of the Gaussian density. The greedy policy chooses an action a_k that attains the minimum in the expression $\min_{a_k \in A} E_{x_k, u_k} [g_k(x_k, a_k, u_k) | I_k]$.

Numerical experiments are carried out in the following settings:

- *Problem parameters:* initial inventory level $x_0 = 5$, holding cost $h = 1$, shortage penalty cost $s = 10$, fixed order amount $b = 10$, random demand $u_k \sim \text{exp}(5)$, discount factor $\gamma = 0.9$, inventory observation noise $v_k \sim N(0, \sigma^2)$ with σ ranging from 0.1 to 3.3 in steps of 0.2.
- *Algorithm parameters:* The number of particles in both the usual particle filter and the projection particle filter is $N = 200$; the exponential family in the projection particle filter is chosen as the Gaussian family; the set of grids on the projected belief space is $G = \{ \text{mean} = [0 : 0.5 : 15], \text{standard deviation} = [0 : 0.2 : 5] \}$ for both our methods and EKF-PPOMDP; one run of horizon length $H = 10^5$ for each average cost criterion case, 1000 independent runs of horizon length $H = 40$ for each discounted total cost criterion case; nearest neighbor as the value function approximator in both our methods and EKF-PPOMDP.
- *Simulation issues:* We use common random variables among different policies and different σ 's.

In order to implement the certainty equivalence methods, we use Monte Carlo simulation to find the optimal threshold policy for the fully observed problem (i.e., $y_k = x_k$): if the inventory level is below the threshold L , the store/warehouse should order to replenish its inventory; otherwise, if the inventory level is above L , the store/warehouse should not order. That is,

$$a_k = \begin{cases} 0, & \text{if } x_k > L; \\ 1, & \text{if } x_k < L. \end{cases} \quad (4.27)$$

The simulation result indicates both the average and discounted cost functions are convex in the threshold and the minimum is achieved at $L = 7.7$ for both.

Table 4.2 and Table 4.3 list the simulated average costs and discounted total cost using different policies under different observation noises, respectively. Our methods generally outperforms all the other algorithms under all observation noise levels. also performs very well, and slightly outperforms CE-MLE. EKF-PPOMDP performs better in the average cost case than the discounted cost case. The greedy policy is much worse than all other algorithms. While our methods and the EKF-PPOMDP involve offline computation, the more critical online computation time of all the simulated methods is approximately the same.

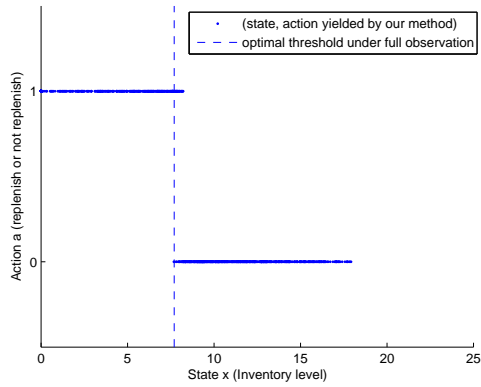
For all the algorithms, the average cost/discounted cost increases as the observation noise increases. That is consistent with the intuition that we cannot perform better with less information. Fig. 5.2 shows 1000 actions taken by our method versus the true inventory levels in the average cost case (the discounted total cost case is similar and is omitted here). The dotted vertical line is the optimal threshold under full observation L . Our algorithm yields a policy that picks actions very close

to those of the optimal threshold policy when the observation noise is small (cf. Fig. 5.2(a)), indicating that our algorithm is indeed finding the optimal policy. As the observation noise increases, more actions picked by our policy violate the optimal threshold, and that again shows the value of information in determining the actions.

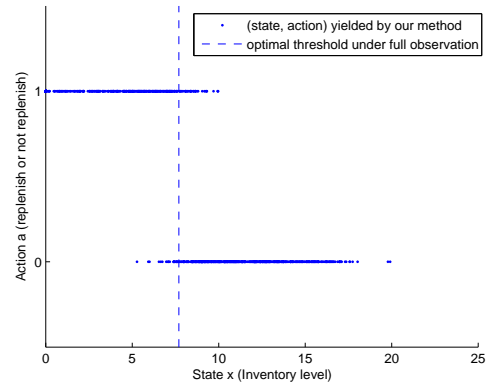
The performances of our two methods are very close, with one slightly better than the other. Solely for the purpose of filtering, doing projection outside the filter is easier to implement if we want to use a general exponential family, and also gives a better estimate of the belief state, since the projection error will not accumulate. However, for solving POMDPs, we conjecture that PPF would work better in conjunction with the policy solved from the projected belief MDP, since the projected belief MDP assumes that the transition of the belief state is also projected. However, we do not know which one is better.

Our method outperforms the EKF-PPOMDP, mainly because the projection particle filter used in our method is better than the extend Kalman filter used in the EKF-PPOMDP for estimating the belief transition probabilities. This agrees with the results in [18], which also observed that Monte Carlo simulation of the belief transitions is better than the EKF estimate.

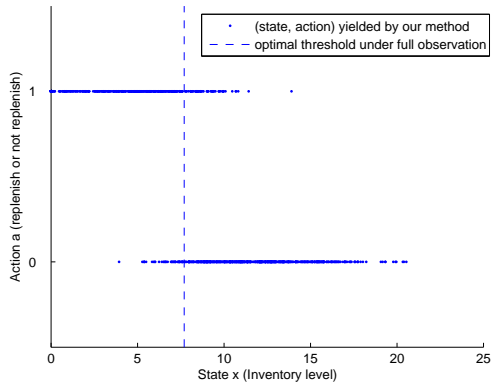
Although the performance of the certainty equivalence method is comparable to that of our methods for this particular example, certainty equivalence is generally a suboptimal policy except in some special cases (cf. section 6.1 in [10]), and it does *not* have a theoretical error bound. Moreover, to use certainty equivalence method requires solving the full observation problem, which is also very difficult in many



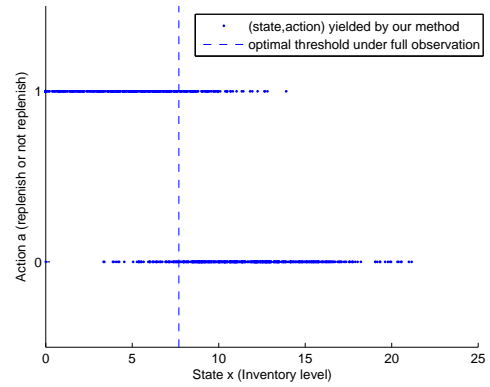
(a) observation noise $\sigma = 0.1$



(b) observation noise $\sigma = 1.1$



(c) observation noise $\sigma = 2.1$



(d) observation noise $\sigma = 3.1$

Figure 4.1: Our algorithm: actions taken for different inventory levels under different observation noise variances.

cases. In contrast, our method has a proven error bound on the performance, and works with the POMDP directly without having to solve the MDP problem under full observation.

4.8 Conclusions

We developed a method that effectively reduces the dimension of the belief space via the orthogonal projection of the belief states onto a parameterized family of probability densities. For an exponential family, the density projection has an analytical form and can be carried out efficiently. The exponential family is fully represented by a finite (small) number of parameters, hence the belief space is mapped to a low-dimensional parameter space and the resultant belief MDP is called the projected belief MDP. The projected belief MDP can then be solved in numerous ways, such as standard value iteration or policy iteration, to generate a policy. This policy is used in conjunction with the projection particle filter for online decision making.

We analyzed the performance of the policy generated by solving the projected belief MDP in terms of the difference between the value function associated with this policy and the optimal value function of the POMDP. We also provided a bound on the error between our projection particle filter and exact filtering.

We applied our method to an inventory control problem, and it generally outperformed other methods. When the observation noise is small, our algorithm yields a policy that picks the actions very closely to the optimal threshold policy for the fully observed problem. Although we only proved theoretical results for discounted cost problems, the simulation results indicate that our method also works well on average cost problems. We should point out that our method is also applicable to finite horizon problems, and is suitable for large-state POMDPs in addition to

continuous-state POMDPs.

Table 4.2: Optimal average cost estimates for the inventory control problem using different methods. Each entry represents the average cost of a run of horizon 10^5 .

σ	Ours 1	Ours 2	CE-Mean	CE-MLE	PPOMDP	Greedy
0.1	12.849	12.849	12.842	12.837	12.941	25.454
0.3	12.845	12.837	12.857	12.861	12.950	25.467
0.5	12.864	12.862	12.867	12.884	12.964	25.457
0.7	12.881	12.884	12.882	12.890	12.990	25.452
0.9	12.904	12.918	12.908	12.940	13.006	25.450
1.1	12.938	12.943	12.945	12.969	13.059	25.428
1.3	12.973	12.986	12.977	12.993	13.105	25.356
1.5	13.016	13.017	13.034	13.029	13.141	25.293
1.7	13.066	13.067	13.100	13.117	13.182	25.324
1.9	13.110	13.105	13.159	13.172	13.214	25.343
2.1	13.123	13.140	13.183	13.227	13.255	25.332
2.3	13.210	13.201	13.263	13.292	13.307	25.355
2.5	13.250	13.246	13.314	13.333	13.380	25.402
2.7	13.323	13.324	13.382	13.454	13.441	25.428
2.9	13.374	13.384	13.458	13.497	13.491	25.478
3.1	13.444	13.459	13.527	13.580	13.553	25.553

Table 4.3: Optimal discounted cost estimate for the inventory control problem using different methods. Each entry represents the discounted cost (mean, standard error in parentheses) based on 1000 independent runs of horizon 40.

σ	Ours 1	Ours 2	CE-Mean	CE-MLE	PPOMDP	Greedy
0.1	126.79 (1.64)	127.26 (1.63)	129.12 (1.81)	129.09 (1.81)	137.41 (1.65)	241.67 (2.99)
0.3	126.86 (1.63)	126.95 (1.63)	129.17 (1.78)	129.11 (1.78)	137.64 (1.62)	242.08 (2.98)
0.5	126.61 (1.63)	126.35 (1.62)	129.12 (1.77)	129.16 (1.78)	138.16 (1.60)	242.66 (2.98)
0.7	126.42 (1.62)	126.99 (1.61)	129.30 (1.77)	129.62 (1.79)	141.78 (1.55)	243.33 (2.98)
0.9	126.78 (1.63)	126.86 (1.63)	129.59 (1.76)	129.76 (1.78)	138.23 (1.60)	244.00 (2.97)
1.1	127.49 (1.64)	127.74 (1.63)	130.19 (1.77)	130.23 (1.75)	140.86 (1.57)	244.81 (2.97)
1.3	128.74 (1.65)	128.30 (1.64)	130.49 (1.76)	130.54 (1.72)	146.02 (1.52)	245.67 (2.96)
1.5	129.30 (1.68)	129.45 (1.66)	130.74 (1.75)	131.09 (1.77)	144.88 (1.52)	246.71 (2.95)

Table 4.4: Continue Table 4.3

σ	Ours 1	Ours 2	CE	CE-MLE	PPOMDP	Greedy
1.7	129.71 (1.67)	128.93 (1.67)	130.95 (1.76)	131.45 (1.77)	146.80 (1.52)	247.70 (2.96)
1.9	130.11 (1.69)	129.85 (1.67)	131.29 (1.75)	131.60 (1.73)	147.16 (1.56)	248.55 (2.93)
2.1	130.67 (1.69)	130.17 (1.67)	131.76 (1.74)	132.24 (1.79)	144.67 (1.54)	249.45 (2.95)
2.3	130.96 (1.68)	130.36 (1.67)	132.22 (1.75)	132.76 (1.78)	145.35 (1.55)	250.07 (2.97)
2.5	131.90 (1.68)	130.86 (1.68)	132.54 (1.76)	133.47 (1.78)	145.06 (1.58)	250.49 (2.96)
2.7	131.81 (1.68)	131.66 (1.68)	133.18 (1.75)	133.98 (1.78)	148.39 (1.54)	250.76 (2.96)
2.9	132.36 (1.68)	131.78 (1.68)	133.61 (1.75)	134.56 (1.83)	146.27 (1.57)	250.81 (2.96)
3.1	132.95 (1.70)	133.51 (1.70)	134.09 (1.76)	135.83 (1.79)	147.96 (1.54)	250.89 (2.95)
3.3	133.08 (1.69)	132.76 (1.69)	134.81 (1.76)	136.12 (1.84)	145.32 (1.60)	250.77 (2.94)

Chapter 5

Particle Filtering Framework for Optimization

5.1 Related Work and Motivation

We consider the global optimization problem:

$$x^* = \arg \max_{x \in \mathcal{X}} H(x),$$

and assume that it has a unique global optimal solution x^* .

Many of the simulation-based global optimization methods, such as the estimation of distribution algorithms (EDAs) [64] [59], the cross-entropy (CE) method [78] [79], and model reference adaptive search (MRAS) method [41], fall into the category of “model-based methods” as classified by [98]. They share the similarities of iteratively repeating the following two steps:

- Generate candidate solutions from an intermediate distribution over the solution space;
- Update the intermediate distribution using the candidate solutions.

This intermediate distribution is often referred to as a probabilistic model, since it often imposes a model on the relationship between the components that are needed to represent a solution. The choice and updating of the probabilistic model (or intermediate distribution) play a key role in determining the efficiency and accuracy of the algorithm.

EDAs were first proposed by [64] in the field of evolutionary computation, with the goal of eliminating the mutation and cross-over operations in genetic algorithms (GAs) in order to avoid partial solutions. EDAs generate offspring by sampling from a distribution over the solution space that is estimated from the candidate solutions of the previous iteration. The estimation of this distribution is often based on a probabilistic model that explicitly expresses the relationship between the underlying variables [53].

The cross-entropy (CE) method was originally introduced for estimating probabilities of rare events in complex stochastic networks [77], and later was modified slightly to be used for solving combinatorial and continuous optimization problems [78]. A key idea of the CE method is to minimize the Kullback-Leibler (KL) divergence between a desired density (the optimal importance sampling density) and a family of parameterized densities, in particular an exponential family, since an analytical solution can be calculated in this case.

The MRAS method was introduced in [41]. MRAS implicitly constructs a sequence of reference distributions and uses this sequence to facilitate and guide the parameter updating associated with a family of parameterized distributions. At each iteration, candidate solutions are sampled from the distribution (in the prescribed family) that has the minimum KL divergence with respect to the reference distribution of the previous iteration.

The aforementioned various ways of updating the probabilistic model motivate us to look for a unifying and systematic approach to the probabilistic model-based methods for optimization. Our main idea is to transform the optimization problem

into a filtering problem. The goal of filtering is to estimate the unobserved state in a dynamic system through a sequence of noisy observations of the state. The unobserved state corresponds to the optimal solution to be estimated; the noisy observations in filtering brings randomization into the optimization algorithm; and the conditional distribution of the unobserved state is a distribution over the solution space, which approaches a delta function concentrated on the optimal solution as the system evolves. Hence, the task of searching for the optimal solutions is carried out through the procedure of estimating the conditional density sequentially. This idea is only conceptual, since filtering can hardly be solved analytically, and some approximate filtering methods are needed. We apply particle filtering to solve the transformed filtering problem. Based on particle filtering, we propose a plain particle filtering framework and a general particle filtering framework for optimization. The former framework is a special case of the latter one and more intuitive, while the latter framework is a generalization and hence provides more opportunities for developing new algorithms. To the best of our knowledge, it is the first work on applying particle filtering to the field of optimization.

The particle filtering framework unifies EDAs, the CE method, and MRAS, and provides new insights into the three optimization methods from another viewpoint. More specifically, EDAs and the CE method fit in the plain particle filtering framework, and in particular, the CE method corresponds to the projection particle filtering described in section 4.3. EDAs and the CE method differ only in their ways of constructing an approximation for the conditional density based on the samples/particles. The MRAS method fits in the general particle filtering framework,

with a specific way to construct the resampling importance density.

The particle filtering framework also sheds light on developing new improved algorithms for global optimization. The possibilities of new algorithms come from the freedom in the particle filtering framework, as well as the vast arrays of techniques of improving nonlinear filtering and particle filtering. We focus on three promising directions: adjusting the trade-off between exploration and exploitation in the search by appropriately choosing the importance densities in particle filtering; incorporating gradient-based local search into simulation-based global search by including the gradient term of the objective function in the state-space model in the filtering problem; preventing premature convergence in simulation-based optimization methods by introducing the idea of “persistent excitation” to optimization.

5.2 Filtering for Optimization

We consider the global optimization problem:

$$x^* = \arg \max_{x \in \mathcal{X}} H(x), \tag{5.1}$$

where the solution space \mathcal{X} is a nonempty set in R^n , and $H(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$ is a real-valued function that is bounded, i.e., $\exists M_1 > -\infty, M_2 < \infty$ s.t. $M_1 \leq H(x) \leq M_2, \forall x \in \mathcal{X}$. We assume that (5.1) has a unique global optimal solution, i.e., $\exists x^* \in \mathcal{X}$ s.t. $H(x) < H(x^*), \forall x \neq x^*, x \in \mathcal{X}$.

The optimization problem (5.1) can be transformed into a filtering problem

by constructing an appropriate state-space model. Let the state-space model be

$$\begin{aligned}x_k &= x_{k-1}, \quad k = 1, 2, \dots, \\y_k &= H(x_k) - v_k, \quad k = 0, 1, \dots,\end{aligned}\tag{5.2}$$

where $x_k \in R^n$ is the unobserved state to be estimated, $y_k \in R$ is the observation, $v_k \in R$ is the observation noise that is an i.i.d. sequence, and the initial state is $x_0 = x^*$ which is unobserved. We assume that v_k has a p.d.f. $\varphi(\cdot)$ that is nondecreasing on its support.

As shown before, the conditional density $b_k(x_k) \triangleq p(x_k|y_{1:k})$ is updated recursively according to

$$b_k(x_k) \propto p(y_k|x_k) \int p(x_k|x_{k-1})b_{k-1}(x_{k-1})dx_{k-1}.\tag{5.3}$$

For the above state-space model (5.2), the transition density is

$$p(x_k|x_{k-1}) = \delta(x_k - x_{k-1}),\tag{5.4}$$

where δ denotes the Dirac delta function. The likelihood function is

$$p(y_k|x_k) = \varphi(H(x_k) - y_k).\tag{5.5}$$

Substituting (5.4) and (5.5) into (5.3), we obtain

$$b_k(x_k) = \frac{\varphi(H(x_k) - y_k)b_{k-1}(x_k)}{\int \varphi(H(x_k) - y_k)b_{k-1}(x_k)dx_k}.\tag{5.6}$$

The connection between the filtering problem (5.2) and the optimization problem (5.1) takes at several places. The underlying value of the unobserved state x_k is x^* , and hence, the goal of filtering (to estimate x_k) is the same as that of optimization (to find x^*). In filtering, at each time k , we estimate a conditional density b_k of

x_k . In optimization, b_k is a density over the solution space, interpreted as the our beliefs about the possible values that x^* might take, and serves as the intermediate distribution for drawing new candidate solutions at the next iteration . At each iteration k , b_k is updated according to (5.6) for an incoming new observation y_k , which reveals new information about x^* . The noise in y_k , or in other words, a nondegenerated p.d.f. φ , brings randomization into the optimization algorithm. We can choose φ to determine how the conditional density (i.e., b_{k-1}) is tuned by the performance of solutions to yield a new conditional density (i.e., b_k) at the next iteration. For example, if $\varphi(x)$ is an increasing function of x , the likelihood function (5.5) assigns more weight to the candidate solutions that have better performance; if $\varphi(x) = 0$ for $x < 0$, then (5.5) discards inferior candidate solutions whose performance is worse than y_k .

It should be expected that if y_k increases with k , the conditional density b_k will get closer to the density of x_k , i.e., a Dirac delta function concentrated on x^* . From the viewpoint of filtering, b_k is the posterior density of x_k that approaches the density of x_k . From the optimization viewpoint, b_k is a density defined on the solution space that becomes more and more concentrated on the optimal solution as k increases. Fig. 5.2 is an illustration of how b_k changes in the first three iterations: at $k = 0$, it has no prior knowledge of x^* (the true value of x_k), and hence, b_0 is uniform over the solution space; at $k = 1$, as observation y_0 provides new information about x^* , b_1 has more weight on the solution space centered around x^* ; at $k = 2$, b_2 becomes even more concentrated on x^* .

In summary, to solve the maximization problem (5.1) is equivalent to recur-

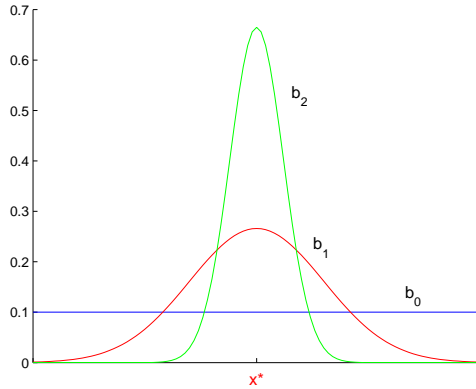


Figure 5.1: At the first three iterations, the conditional density b_k becomes more and more concentrated on the optimal solution.

sively estimating b_k of the model (5.2) while constructing an increasing sequence of observations $\{y_k\}$.

5.3 Particle Filtering Framework for Optimization

The filtering idea is only conceptual and the resulting problem is not analytically solvable. Hence, we need some approximation methods to solve the above filtering problem. We apply the plain particle filter and the general particle filter to the filtering problem, with just a little tweak to adapt to the optimization problem. The application of particle filtering turns out to be a framework for many simulation-based optimization algorithms. In the following, we present *the plain particle filter framework for optimization* (PPFO), and then *the general particle filter framework for optimization* (GPFO). The former framework is a special case of the latter and provides more intuition, while the latter framework is more general

and allows more variations in the development of new algorithms.

Algorithm 5.1 *Plain Particle Filter Framework for Optimization (PPFO)*

1. Initialization. Specify $\rho \in (0, 1]$, and an initial p.d.f./p.m.f. b_0 that is defined on \mathcal{X} . Sample $\{x_1^i\}_{i=1}^N$ i.i.d. from b_0 . Set $k = 1$.

2. Observation Construction. Let y_k be the sample $(1-\rho)$ -quantile of $\{H(x_k^i)\}_{i=1}^N$. If $k > 1$ and $y_k \leq y_{k-1}$, then set $y_k = y_{k-1}$.

3. Bayes' Updating. $\hat{b}_k(x_k) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i)$, where weights are calculated according to

$$w_k^i \propto \varphi(H(x_k^i) - y_k), i = 1, 2, \dots, N,$$

and normalized.

4. Resampling. Construct a continuous approximation $\tilde{b}_k(x_k)$ from $\hat{b}_k(x_k)$. Sample $\{x_{k+1}^i\}_{i=1}^N$ i.i.d. from $\tilde{b}_k(x_k)$.

5. Stopping. If a stopping criterion is satisfied, then stop; else, $k \leftarrow k + 1$ and go to step 2.

At initialization, the PPFO algorithm draws samples from an initial distribution b_0 that is defined on \mathcal{X} . A parameter ρ is specified to determine the $(1 - \rho)$ -quantile samples that will be used to construct a nondecreasing the observation sequence $\{y_k\}$. The requirement of nondecreasing is to ensure the observation sequence does not become worse as the algorithm goes on. Since the transition probability is 1, the importance sampling step is omitted with suitable change of the indices. The

Bayes' updating step assigns weights to the samples according to their performance. Slightly different from the plain particle filter, the resampling step here constructs a continuous density \tilde{b}_k first, since the discrete approximation \hat{b}_k does not provide any new samples. The new samples drawn from \tilde{b}_k are more concentrated in the promising areas than the old samples. To adapt to optimization, a stopping step is added to the algorithm; whereas in filtering, the algorithm keeps going on as the real system operates.

Similarly, by applying the general particle filter, we introduce the general particle filtering framework for optimization as follows:

Algorithm 5.2 *A General Particle Filtering Framework for Optimization (GPFO)*

1. Initialization. Specify $\rho \in (0, 1]$, a nonnegative nonincreasing sequence $\{\epsilon_k\}$, and an initial p.d.f./p.m.f. p_0 that is defined on \mathcal{X} . Sample $\{x_0^i\}_{i=1}^N$ i.i.d. from b_0 . Set $k = 1$.
2. Importance Sampling. Sample x_k^i from $q_k(x_k|x_{k-1}^i, y_k)$, $i = 1, \dots, N$.
3. Observation Construction. Let y_k be the sample $(1-\rho)$ -quantile of $\{H(x_k^i)\}_{i=1}^N$. If $k > 1$ and $y_k \leq y_{k-1}$, then set $y_k = y_{k-1}$.
4. Bayes' Updating. $\hat{b}_k(x_k) = \sum_{i=1}^N w_k^i \delta(x_k - x_k^i)$, where the normalized weights are calculated as

$$w_k^i \propto \frac{\varphi(H(x_k^i) - y_k) b_{k-1}(x_{k-1}^i)}{q_k(x_k^i|x_{k-1}^i, y_k) g_{k-1}(x_{k-1}^i|y_{0:k-1})}.$$

5. Importance Resampling. Sample $\{x_k^i\}_{i=1}^N$ i.i.d. from $g_k(x_k|y_{0:k})$.

6. Stopping. *If a stopping criterion is satisfied, then stop; else, $k \leftarrow k + 1$ and go to step 2.*

5.4 Interpretation of EDAS, CE, MRAS

In this section, we use the particle filtering framework to interpret some of the existing optimization algorithms: estimation of distribution algorithms (EDAs), the cross entropy (CE) method, and model reference adaptive search (MRAS). A class of EDAs can fit in the plain particle filtering framework. The CE method can be viewed as projection particle filtering, which also fits in the plain particle filtering framework with a specific way to construct a continuous approximation of the conditional density, namely the density projection approach. The MRAS method can fit in the general particle filtering framework where the resampling importance density is constructed via density projection. This interpretation also provides another insight into the relationships between the three methods in addition to the view in [41]. Specifically, the main difficulty in EDAs is to estimate a distribution from the samples, and this difficulty is solved in the CE method by projecting the empirical distribution of the samples to obtain an approximate continuous density. However, the density projection introduces an error, which is corrected in the MRAS method by taking the projected density as a resampling importance density and hence weighting the samples differently from the CE method.

In all three methods, the most common sample selection scheme is the so-called *truncated selection* [93], which selects the elite samples whose performance is

above a threshold. In the following, we will focus on the truncated selection scheme.

Recall that in the optimization problem (5.1), the objective function $H(x)$ is bounded by $M_1 \leq H(x) \leq M_2$. In the state-space model (5.2), let the observation noise v_k follow a uniform distribution $U(0, M_2 - M_1)$. Since $v_k = H(x_k) - y_k$, the likelihood function is

$$p(y_k|x_k) = \begin{cases} \frac{1}{M_2 - M_1}, & \text{if } 0 \leq H(x_k) - y_k \leq M_2 - M_1; \\ 0, & \text{otherwise.} \end{cases} \quad (5.7)$$

Since $y_k = H(x)$, $x \in \mathcal{X}$, the inequality $H(x_k) - y_k \leq M_2 - M_1$ always holds. Hence, (5.7) can be written in a more compact way as

$$p(y_k|x_k) = \frac{1}{M_2 - M_1} I_{\{H(x_k) \geq y_k\}}, \quad (5.8)$$

where $I_{\{\cdot\}}$ denotes the indicator function.

Substituting (5.8) into (5.6), we obtain

$$b_k(x_k) = \frac{I_{\{H(x_k) \geq y_k\}} b_{k-1}(x_k)}{\int I_{\{H(x_k) \geq y_k\}} b_{k-1}(x_k) dx_k}. \quad (5.9)$$

With i.i.d. samples $\{x_k^i\}_{i=1}^N$ drawn from b_{k-1} , $b_k(x_k)$ can be approximated by

$$\hat{b}_k(x_k) = \frac{\sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}} \delta(x_k - x_k^i)}{\sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}}}. \quad (5.10)$$

Thus, (5.9) is equivalent to selecting the elite solutions to tune the sampling distribution at the previous iteration, and (5.10) is the Monte Carlo version of (5.9).

These two equations are the cornerstone of the rest of this section.

5.4.1 Estimation of Distribution Algorithms

EDAs are a class of optimization algorithms based on the key idea of iteratively doing the two steps:

1. Select elite samples from a pool of samples that are generated from a probability distribution;
2. Estimate the probability distribution of selected samples and generate new samples from it.

With the truncation selection scheme, one class of EDAs can be viewed as an instantiation of the plain particle filtering framework as follows:

Algorithm 5.3 *Instantiation 1 of Plain Particle Filter Framework for Optimization (PPFO1)*

1. Initialization. Specify $\rho \in (0, 1]$, and an initial p.d.f./p.m.f. b_0 that is defined on \mathcal{X} . Sample $\{x_1^i\}_{i=1}^N$ i.i.d. from b_0 . Set $k = 1$.
2. Observation Construction. Let y_k be the sample $(1-\rho)$ -quantile of $\{H(x_k^i)\}_{i=1}^N$. If $k > 1$ and $y_k \leq y_{k-1}$, then set $y_k = y_{k-1}$.
3. Bayes' Updating. The discrete approximation $\hat{b}_k(x_k)$ is as (5.10).
4. Resampling. Estimate a continuous approximation $\tilde{b}_k(x_k)$ from $\hat{b}_k(x_k)$. Sample $\{x_{k+1}^i\}_{i=1}^N$ i.i.d. from $\tilde{b}_k(x_k)$.
5. Stopping. If a stopping criterion is satisfied, then stop; else, $k \leftarrow k + 1$ and go to step 2.

It is obvious that the observation construction and Bayes' updating steps essentially select the elite samples according to the truncated selection scheme, corresponding to step 1 in EDAs; and the resampling step corresponds to step 2 in EDAs.

The main difficulty in EDAs is to estimate the distribution of the selected samples. When doing so, EDAs often take into account the interaction between the underlying variables that represent a solution, and express the interaction explicitly through the use of different probabilistic models. One way is to use a dynamic Bayesian network (DBN) to represent such a probabilistic model and infer the underlying probability distribution [53]. Put in our context, the relationship between the components of the state vector x_k is expressed through the use of a DBN, and the estimation of the joint distribution of the components is $\tilde{b}_k(x_k)$. Interestingly, there is a particular particle filter designed especially for DBNs [49], which samples x_k according to the relationship between its components so that the sampling is more efficient. This particle filter can be adopted to improve EDAs that use the DBN representation.

5.4.2 Cross Entropy Method

The standard CE method (we use the word “standard” to distinguish it from the extended version of standard CE [26]) for the optimization problem (5.1) is as follows:

Algorithm 5.4 *Standard CE Algorithm for Optimization*

1. Choose an initial p.d.f./p.m.f. $f(\cdot, \theta_0)$, $\theta_0 \in \Theta$. Specify the parameter $\rho \in (0, 1]$, and set $k = 1$.
2. Generate samples $\{x_k^i\}_{i=1}^N$ from the density $f(\cdot, \theta_{k-1})$ and compute the sample $(1 - \rho)$ -quantile y_k of the performances $\{H(x_k^i)\}_{i=1}^N$.

3. Compute the new parameter according to

$$\theta_k = \arg \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}} \log f(x_k^i, \theta). \quad (5.11)$$

4. If a stopping criterion is satisfied, then terminate; else, set $k = k + 1$ and go to step 2.

Equation (5.11) comes from the density projection of the optimal importance sampling density onto a parameterized family of densities $\{f(\cdot, \theta), \theta \in \Theta\}$. Projection particle filtering [95] also uses the density projection technique, but for a very different reason. It projects the discrete approximation \hat{b}_k onto the parameterized family $\{f(\cdot, \theta), \theta \in \Theta\}$ in order to obtain a continuous approximation \tilde{b}_k that is characterized by only a few parameters, which is very useful in reducing the complexity of dynamic programming in a decision making problem. Specifically, projection particle filtering chooses a value of the parameter θ such that the Kullback-Leibler (KL) divergence between \hat{b}_k and $f(\cdot, \theta)$ is minimized. The KL divergence between \hat{b}_k and $f(\cdot, \theta)$ is:

$$\begin{aligned} D_{KL}(\hat{b}_k \| f(\cdot, \theta)) &= \int \hat{b}_k \log \frac{\hat{b}_k}{f(\cdot, \theta)} \\ &= \int \hat{b}_k \log \hat{b}_k - \int \hat{b}_k \log f(\cdot, \theta). \end{aligned}$$

Since the first term does not depend on $f(\cdot, \theta)$, minimizing the above equation is equivalent to solving the maximization problem

$$\max_{\theta \in \Theta} E_{\hat{b}_k}[\log f(\cdot, \theta)].$$

Since $\hat{b}_k(x_k)$ satisfies (5.10), the above maximization problem can be approximated by

$$\max_{\theta \in \Theta} \frac{\sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}} \log f(x_k^i, \theta)}{\sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}}},$$

which is equivalent to

$$\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}} \log f(x_k^i, \theta). \quad (5.12)$$

Therefore, the optimization algorithm adapted from projection particle filtering is as follows:

Algorithm 5.5 *Instantiation 2 of Plain Particle Filter Framework for Optimization (PPFO2)*

1. Initialization. Specify $\rho \in (0, 1]$, and an initial p.d.f./p.m.f. $f(x_0, \theta_0)$ that is defined on \mathcal{X} . Sample $\{x_1^i\}_{i=1}^N$ i.i.d. from $f(x_0, \theta_0)$. Set $k = 1$.
2. Observation Construction. Let y_k be the sample $(1 - \rho)$ -quantile of $\{H(x_k^i)\}_{i=1}^N$. If $k > 1$ and $y_k \leq y_{k-1}$, then set $y_k = y_{k-1}$.
3. Bayes' Updating. The discrete approximation $\hat{b}_k(x_k)$ is as 5.10.
4. Resampling. Construct a continuous approximation $\tilde{b}_k(x_k) = f(x_k, \theta_k)$, where

$$\theta_k = \arg \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N I_{\{H(x_k^i) \geq y_k\}} \log f(x_k^i, \theta). \quad (5.13)$$

Sample $\{x_{k+1}^i\}_{i=1}^N$ i.i.d. from $\tilde{b}_k(x_k)$.

5. Stopping. If a stopping criterion is satisfied, then stop; else, $k \leftarrow k + 1$ and go to step 2.

It is easy to see that this algorithm is essentially the same as the standard CE algorithm (Note that (5.11) and (5.13) are exactly the same). Compared with EDAs, the CE method avoids complicated estimation of the density b_k through the use of density projection without the need to specify the relationships among the components of x_k . However, from a filtering viewpoint, the projection particle filtering introduces a projection error that is accumulated over iterations. The reason can be seen by scrutinizing the one-step evolution of the approximate density. Since samples $\{x_k^i\}_{i=1}^N$ are sampled from $\tilde{b}_{k-1} = f(\cdot, \theta_{k-1})$, the density that the algorithm actually tries to approximate at iteration k is

$$b_k(x_k) = \frac{I_{\{H(x_k) \geq y_k\}} f(x_{k-1}, \theta_{k-1})}{\int I_{\{H(x_k) \geq y_k\}} f(x_{k-1}, \theta_{k-1}) dx_{k-1}}.$$

Compared with the original equation (5.9) for b_k , b_{k-1} is replaced by its approximation $f(\cdot, \theta_{k-1})$, which introduces a projection error that is accumulated to the next iteration. This projection error can be corrected by taking $f(\cdot, \theta_{k-1})$ as an importance density and hence taken care of by the weights of the samples. This leads to another instantiation of the particle filtering framework, which coincides with the instantiation of MRAS developed in [41].

5.4.3 Model Reference Adaptive Search

As in the CE method, at each iteration the MRAS method projects a desired density onto a family of parameterized densities to yield a density from which the candidate solutions are drawn. In the CE method the target distribution is a single optimal importance sampling density, whereas in the MRAS method, the parameter

updating is guided by an implicit sequence of distributions called the reference distributions. Here, we consider the Monte Carlo version of the MRAS₀ algorithm with truncated selection scheme presented in [41]:

Algorithm 5.6 *Model Reference Adaptive Search Method with Truncated Selection Scheme*

1. Choose an initial p.d.f./p.m.f. $f(\cdot, \theta_0)$, $\theta_0 \in \Theta$. Specify the parameter $\rho \in (0, 1]$, and set $k = 1$.
2. Generate samples $\{x_k^i\}_{i=1}^N$ from the density $f(\cdot, \theta_{k-1})$ and compute the sample $(1 - \rho)$ -quantile y_k of the performances $\{H(x_k^i)\}_{i=1}^N$.
3. Compute the new parameter according to

$$\theta_k = \arg \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \frac{I_{\{H(x_k^i) \geq y_k\}}}{f(x_k^i, \theta_{k-1})} \log f(x_k^i, \theta). \quad (5.14)$$

4. If a stopping criterion is satisfied, then terminate; else, set $k = k + 1$ and go to step 2.

Equation (5.14) comes from the projection of the implicit reference distribution onto the family of parameterized densities. In the particle filtering framework, we will see that the sequence of reference distributions is the sequence of the approximated conditional densities $\{\hat{b}_k\}$, which guide the design of the resampling importance densities $\{f(\cdot, \theta_k)\}$. Specifically, let the initial p.d.f./p.m.f. be

$$b_0(x_0) = \frac{I_{\{H(x_0) \geq y_0\}}}{\int I_{\{H(x_0) \geq y_0\}} dx_0}.$$

Since y_k is a nondecreasing sequence, using the recursive equation (5.9), it can be shown by induction that

$$b_k(x_k) = \frac{I_{\{H(x_k) \geq y_k\}}}{\int I_{\{H(x_k) \geq y_k\}} dx_k}. \quad (5.15)$$

Suppose the resampling importance density g_{k-1} is

$$g_{k-1}(x_{k-1}|y_{0:k-1}) = f(x_{k-1}, \theta_{k-1}),$$

from which the i.i.d. samples $\{x_k^i\}_{i=1}^N$ are drawn, then the weight of each x_k^i is

$$\begin{aligned} w_k^i &= \frac{b_k(x_k^i)}{g_{k-1}(x_{k-1}^i|y_{0:k-1})} \\ &\propto \frac{I_{\{H(x_k^i) \geq y_k\}}}{f(x_{k-1}^i, \theta_{k-1})}. \end{aligned} \quad (5.16)$$

As shown before, projection of $\hat{b}_k(x_k)$ onto the parameterized family of densities $\{f(\cdot, \theta), \theta \in \Theta\}$ is equivalent to the maximization problem

$$\max_{\theta \in \Theta} E_{\hat{b}_k}[\log f(\cdot, \theta)]. \quad (5.17)$$

Since $\hat{b}_k(x_k) = \sum_{i=1}^N w_k^i(x_k - x_k^i)$, where w_k^i satisfies (5.16), and $x_{k-1}^i = x_k^i$, (5.17) can be approximated by

$$\max_{\theta \in \Theta} \sum_{i=1}^N \frac{I_{\{H(x_k^i) \geq y_k\}}}{f(x_k^i, \theta_{k-1})} \log f(x_k^i, \theta).$$

Thus, the algorithm is as follows:

Algorithm 5.7 *Instantiation 1 of General Particle Filter Framework for Optimization (GPF01)*

1. Initialization. Specify $\rho \in (0, 1]$, and an initial p.d.f./p.m.f. $f(x_0, \theta_0)$ that is defined on \mathcal{X} . Sample $\{x_1^i\}_{i=1}^N$ i.i.d. from $f(x_0, \theta_0)$. Set $k = 1$.

2. Observation Construction. Let y_k be the sample $(1-\rho)$ -quantile of $\{H(x_k^i)\}_{i=1}^N$.

If $k > 1$ and $y_k \leq y_{k-1}$, then set $y_k = y_{k-1}$.

3. Bayes' Updating. $\hat{b}_k(x_k) = \sum_{i=1}^N w_k^i (x_k - x_k^i)$, where the weights are

$$w_k^i \propto \frac{I_{\{H(x_k^i) \geq y_k\}}}{f(x_k^i, \theta_{k-1})}.$$

4. Resampling. Construct a resampling importance density $g_k(x_k|y_{0:k}) = f(x_k, \theta_k)$,

where

$$\theta_k = \arg \max_{\theta} \frac{1}{N} \sum_{i=1}^N \frac{I_{\{H(x_k^i) \geq y_k\}}}{f(x_k^i, \theta_{k-1})} \log f(x_k^i, \theta). \quad (5.18)$$

Sample $\{x_{k+1}^i\}_{i=1}^N$ i.i.d. from $g_k(x_k|y_{0:k})$.

5. Stopping. If a specified stopping criterion is satisfied, then stop; else, $k \leftarrow k+1$

and go to step 2.

Note that the (5.18) is exactly the same as the updating equation (5.14) in MRAS, and Algorithm 5.7 and Algorithm 5.6 are essentially the same.

5.5 Implication for New Algorithms

The particle filtering framework for optimization not only provides a unifying framework for some of the existing algorithms, but also opens up the possibility for new algorithms. There is a considerable amount of freedom in the framework, such as the choices of the observation noise, the observation sequence, the sampling and resampling importance densities, each of which can lead to a different algorithm. In addition, many of the techniques that have been used to improve particle filtering

can also be adapted to optimization, such as the improvement of EDAs with the particle filter for DBNs as we mentioned in last section. In the rest of this section, we will focus on discussing three promising directions for developing new improved algorithms under this framework.

5.5.1 Balancing Exploration and Exploitation

Proper sampling and resampling importance densities can be chosen to adjust the trade-off between exploitation and exploration. Construction of the resampling importance density using the kernel method for density estimation [65], or approximation with Gaussian mixture [50] is very easy to implement, and the obtained continuous distributions are easy to sample from. They add more exploration on the solution space, compared to a single Gaussian density that is often used in the CE and MRAS. A Markov chain Monte Carlo (MCMC) step can be added after resampling [32] to further adjust the trade-off between exploitation and exploration, or add some local search.

5.5.2 Combining Global Search with Local Search

We can incorporate local/gradient search into global search systematically by including the gradient of the objective function into the state-space model. First, assume that the objective function $H(x)$ is differentiable with respect to x and its derivative is denoted as

$$G(x) \triangleq \nabla H(x). \quad (5.19)$$

Let the state-space model be

$$\begin{aligned}x_k &= x_{k-1} + \epsilon_k G(x_{k-1}), \quad k = 1, 2, \dots, \\y_k &= H(x_k) - v_k, \quad k = 0, 1, \dots,\end{aligned}\tag{5.20}$$

where $\{\epsilon_k\}$ is a sequence of step sizes that are properly chosen. The intuition of model (5.20) is that the unobserved state x_k has a stationary underlying value x^* if the initial condition $x_0 = x^*$. Since $G(x^*) = 0$, the state stays at x^* for all time k if it starts at x^* . For this model, the propagation step in the particle filtering framework moves each sample along its gradient. Therefore, the resulting algorithm incorporates a gradient-based local search into the simulation-based global search.

5.5.3 Overcoming Premature Convergence

Many of the simulation-based optimization algorithms suffer from the problem of premature convergence, i.e., they converge too fast such that they get stuck at a local optimum. This problem is especially severe if the objective function has many local optima. A similar phenomenon often happens in filtering for a static state, also called system identification: the algorithm converges too fast such that it converges to a wrong value. To avoid premature convergence, many system identification algorithms employ the idea of “persistent excitation”, which adds an artificial noise to the static system dynamics to keep a continuous exploration of the state space [48] [58]. The artificial noise is large at the beginning to prevent premature convergence to a wrong value, and gradually dies down as the estimate converges to the true

value. Hence, in the state-space model (5.2), the state equation becomes

$$x_k = x_{k-1} + \alpha_k \omega_k, \quad k = 1, 2, \dots, \quad (5.21)$$

where

$$\alpha_k = \alpha_1 \beta^{k-1}, \quad \beta \in (0, 1).$$

From the optimization viewpoint, the propagation step moves each candidate solution around randomly with a decreasing randomness at each iteration, which allows for more exploration at the beginning. This idea is very effective in preventing premature convergence in optimization algorithms, as shown in the next section.

5.6 Numerical Experiments

The standard CE method often suffers from the problem of premature convergence due to the quick convergence of the parameterized family of distributions $f(\cdot, \theta_k)$ to a degenerate measure (Dirac measure) [26]. A smoothed parameter updating procedure has been used to address this problem [26], i.e., a smoothed version of the distribution parameter θ_k is computed at each iteration k according to

$$\tilde{\theta}_k = \nu \theta_k + (1 - \nu) \tilde{\theta}_{k-1}, \quad \nu \in (0, 1),$$

where ν is the smoothing parameter, and $\tilde{\theta}_{k-1}$ the smoothed distribution parameter at iteration $k - 1$.

Since the standard CE method can be included in the particle filtering framework, it is straightforward to apply “persistent excitation” to the standard CE method. Hence, instead of smoothing the parameter, at each iteration we randomly

move the candidate solutions according to (5.21) before updating the parameterized density. In the following, we numerically compare our proposed method (abbreviated as CEA) and the CE method with smoothing on some benchmark problems, which have been previously studied in [67] [51] [41]. These test functions are:

(1) Rosenbrock function ($n = 20$)

$$H(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2,$$

where $x^* = (1, \dots, 1)^T$, $H(x^*) = 0$.

(2) Dejong's 5th function ($n = 2$)

$$H(x) = [0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{j,i})^6}]^{-1},$$

where $a_{j,1} = \{-32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32\}$, $a_{j,2} = \{-32, -32, -32, -32, -32, -16, -16, -16, -16, -16, 0, 0, 0, 0, 0, 16, 16, 16, 16, 16, 32, 32, 32, 32, 32\}$; with 24 local minima and one global minimum at $x^* = (-32, -32)^T$, $H(x^*) \approx 0.998$.

(3) Powel singular function ($n = 20$)

$$H(x) = \sum_{i=1}^{n-2} [(x_{i-1} + 10x_i)^2 + 5(x_{i+1} - x_{i+2})^2 + (x_i - 2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4],$$

where $x^* = (0, \dots, 0)^T$, $H(x^*) = 0$.

(4) Pintér's function ($n = 20$)

$$\begin{aligned} H(x) &= \sum_{i=1}^n ix_i^2 + \sum_{i=1}^n 20i \sin^2(x_{i-1} \sin x_i - x_i + \sin x_{i+1}) \\ &\quad + \sum_{i=1}^n i \log_{10}(1 + i(x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos x_i + 1))^2, \end{aligned}$$

where $x_0 = x_n$, $x_{n+1} = x_1$, $x^* = (0, \dots, 0)^T$, $H(x^*) = 0$.

(5) Shekel's function ($n = 4$)

$$H(x) = \sum_{i=1}^5 5((x - a_i)^T(x - a_i) + c_i)^{-1},$$

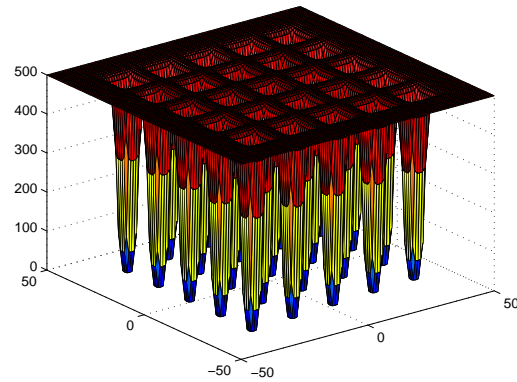
where $a_1 = (4, 4, 4, 4)^T$, $a_2 = (1, 1, 1, 1)^T$, $a_3 = (8, 8, 8, 8)^T$, $a_4 = (6, 6, 6, 6)^T$, $a_5 = (3, 7, 3, 7)^T$, and $c = (0.1, 0.2, 0.2, 0.4, 0.4)^T$, $x^* \approx (4, 4, 4, 4)^T$, $H(x^*) \approx -10.153$.

Dejong's 5th function and Shekel's function are relatively low dimensional and has only a few local optima, but the optima are separated by plateaus. Rosenbrock function and Powel singular function are 20-dimensional badly scaled problems. Pinter's function has many local optima, and the number of local optima increases exponentially as the number of dimension increases. Fig. 5.6 shows some of the functions in two dimensions.

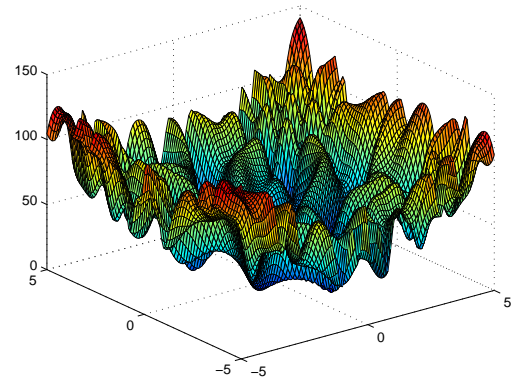
For the CE method, we use the parameter values suggested by [51], with smoothing parameter $\nu = 0.7$. We also found by trial and error that $\nu = 0.2$ works very well for the CE method with smoothing. For CEA, the noise ω_k is chosen to be a standard Gaussian $N(0, I_{n \times n})$, where n is the dimension of the solution space. In both methods, the parametrized family of distributions is chosen to be a multivariate Gaussian family. Table 5.1 and Fig. 5.3 show that CEA converges to better solutions than the CE method on these benchmark problems, and it converges faster than the CE method with $\nu = 0.2$.

Note that on the Rosenbrock function, CEA can approach the optimum very closely with appropriately chosen parameters; whereas for whatever smoothing parameter values α and β , the CE method with smoothing always gets stuck at somewhere far from the optimum. Table 5.2 shows on the Rosenbrock function how the

(a) Dejong's 5th function, $-50 \leq x_i \leq 50$, $i = 1, 2$



(b) Pinter's function, $-5 \leq x_i \leq 5$, $i = 1, 2$



(c) Rosenbrock function, $-5 \leq x_i \leq 5$, $i = 1, 2$

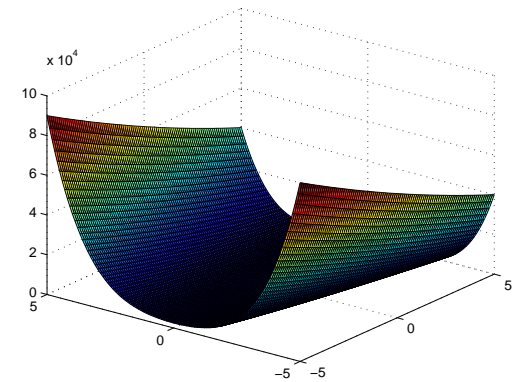


Figure 5.2: Some benchmark problems in two dimensions.

Table 5.1: Average performance of CEA and CE on some benchmark problems. Each entry presents the mean of $H(x^*)$ with standard error in parentheses, based on 100 independent runs.

Function	$H(x^*)$	CEA ($\alpha_1 = 10, \beta = 0.95$)	CE ($\nu = 0.7$)	CE ($\nu = 0.2$)
Rosenbrock	0	16.84 (0.971)	19.41 (1.11)	17.39 (0.009)
Dejong 5th	0.998	0.998 (1.8e-12)	1.01 (0.010)	0.998 (2.3e-15)
Powell	0	2.4e-5 (2.0e-6)	420.8 (266.8)	2.5e-4 (2.4e-4)
Pintér's	0	0.0068 (4.0e-4)	4.51 (0.15)	3.82 (0.055)
Shekel's	-10.153	-10.153 (8.5e-9)	-9.033 (0.268)	-9.929 (0.128)

value of β , i.e. the decreasing rate of the artificial noise, affects the performance of CEA: slower decrease of the noise yields better solutions, but not surprisingly takes more time to converge.

Table 5.2: Average performance of CEA with different parameter values of α and β on the Rosenbrock function. Each entry presents the mean of $H(x^*)$ with standard error in parentheses, based on 100 independent runs.

Function	$H(x^*)$	$\alpha_1 = 10, \beta = 0.95$	$\alpha_1 = 10, \beta = 0.98$	$\alpha_1 = 10, \beta = 0.995$
Rosenbrock	0	16.84 (0.971)	11.90 (0.023)	0.505 (0.010)

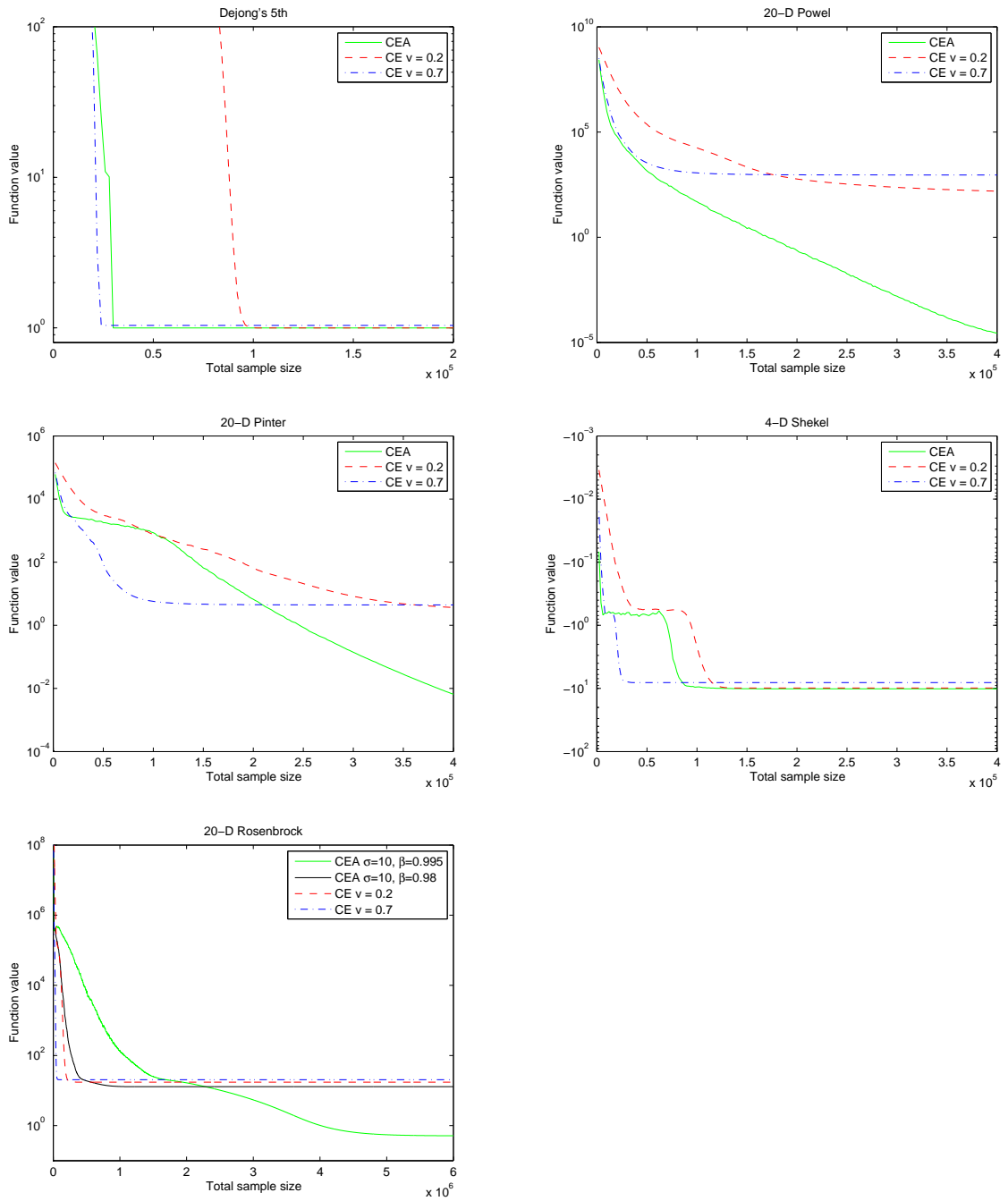


Figure 5.3: Average performance of CEA and CE on some benchmark problems.

5.7 Conclusions and Future Research

We transformed a global optimization problem to a filtering problem, and hence many ideas and results from filtering can be adapted to global optimization. Based on particle filtering, we proposed a framework for many simulation-based optimization algorithms. In particular, the framework unifies: EDAs, the CE method and MRAS, and provides new insight into the relationship between them. Moreover, the framework holds the promise for developing new optimization algorithms through the choice of observation noise, sampling and resampling importance densities, as well as a vast array of improving techniques for nonlinear filtering and particle filtering.

There are two important lines of future research that we will continue to pursue. First, we will further study how to develop new algorithms using the particle filtering framework and the performance of these new algorithms. Secondly, we plan to investigate the convergence property of the particle filtering framework for optimization. Although convergence has been proved for EDAs [93], the CE method [78], and MRAS [41] individually, we are interested in unifying convergence results under the particle filtering framework. The analysis will be based on some existing stability results for nonlinear filters, such as [20].

Chapter 6

Conclusions and Future Research

6.1 Conclusions

This dissertation has proposed and developed new methods and results for solving problems in partially observable Markov decision processes and global optimization.

The first part of the dissertation focuses on continuous-state POMDPs. Continuous-state POMDPs provide a more natural mathematical model than finite-state POMDPs for many application problems. While there are a good number of numerical methods for finite-state POMDPs, there are only a few for continuous-state ones and their convergence results are sparse. Existing algorithms (for finite-state POMDPs) are hard to extend to continuous-state POMDPs, mainly due to the infinite dimensional belief space in a continuous-state POMDP as opposed to a finite-dimensional belief space in a finite-state POMDP. Based on the idea of density projection with particle filtering, we have developed a numerical method for effective dimension reduction and scalable approximation for an arbitrary belief state, such as a multi-modal or a heavy-tail distribution. The idea of density projection orthogonally projects an arbitrary density onto a parameterized family of densities, namely, an exponential family of densities in our algorithm, to yield a finite low-dimensional representation for that density. Based on some existing results, we have shown that under certain

mild conditions, the approximate density approaches the true density as the number of parameters increases, and hence, shown the convergence of our algorithms. We have proved error bounds for our proposed POMDP solving algorithm and online filtering algorithm. We have applied our method to an inventory control problem for which there are some known analytical results for the purpose of comparison, and obtained good numerical results that show the promise of our method. Although we have only proved error bounds for the infinite-horizon discounted cost criterion, the numerical results also indicate our method works very well for the average cost criterion. In addition, with a little straightforward modification, our method can be easily applied to finite-horizon problems or (finite) large-state problems.

The second part of the dissertation is devoted to the understanding and improving of a class of simulation-based algorithms for global optimization. We have transformed a global optimization algorithm into a filtering problem, and hence, many results in filtering can be adapted to global optimization. In particular, we have used a novel interpretation of particle filtering to develop a unifying framework for many simulation-based optimization algorithms, such as the cross-entropy method, the estimation of distribution algorithms, and the model reference adaptive search. The framework reveals relationships between these algorithms and new interesting insights. By better understanding these algorithms, we have proposed several promising directions under the same framework for new improved algorithms, such as balancing the exploitation and exploration, combining simulation-based global search with gradient-based local search, and preventing premature convergence. We explored the last direction, and obtained a new improved algorithm which shows

better performance than the existing cross-entropy algorithm on some benchmark problems.

6.2 Future Research

Currently I am applying the method for continuous-state POMDPs described in Chapter 4 to some stochastic control problems in finance. Many financial problems, such as portfolio optimization or hedging, can be modeled as stochastic control problems, where the control is the investment strategy. In many financial models, such as the classical Black-Scholes model [15], the price of a risky asset is modeled as a stochastic process, where the volatility of the risky asset is treated as a constant. However, to better model the reality, more and more financial models assume that the volatility itself is also a stochastic process. While many analytical results have been derived for constant volatility models, stochastic volatility models generally do not admit analytical solutions except in some rare cases, such as the Heston model for option pricing [38]. Hence, problems involving stochastic volatility often need to be solved numerically. A POMDP is a natural model for a portfolio optimization or hedging problem involving assets with stochastic volatility, where the volatility is the hidden state, the price observed at discrete times is a partial observation of the volatility, and the utility function is the objective function. Since the asset price and capital are discrete at least down to the pennies, there are a huge number of states such that the algorithms for finite-state POMDPs are not applicable here. Our proposed method for continuous-state POMDPs provides an efficient numerical

tool for these financial problems to find the best investment strategy and study the effect of the amount of initial capital.

Another ongoing work is the study of the convergence of the particle filtering framework for global optimization. The hope is that the conditional density converges to a degenerated density concentrated on the optimal solution in spite of the initial conditional. That is closely related with the stability result of nonlinear filtering [20].

There are two lines of future research that we plan to pursue. The first line is to develop numerical methods for POMDPs with the state or/and observation equation being a jump diffusion process. The second line of future research is to extend the particle filtering framework to stochastic optimization, i.e., optimization problems where the objective functions cannot be evaluated exactly. ‘ For POMDPs, the state equation (3.2) can be viewed as either a discrete state difference equation, or a discretized diffusion process. If it is a diffusion process, the discretization introduces error in the simulation of the propagation of the state. Some recent research has studied the exact simulation (without discretization error) of a diffusion process [14] [13], to replace the usual Euler-discretization scheme of the state equation between observations. However, the exact simulation has not yet been extended to jump-diffusion processes, and recent research on the inference of a jump-diffusion process still uses an Euler scheme [43]. Therefore, we need to further investigate efficient filtering methods of jump-diffusion processes, and numerical solutions to the corresponding POMDPs. In financial engineering, many models with jump-diffusion processes have been proposed to better model the financial market than those mod-

els with only diffusion processes. For example, Bates [8] added a Poisson process to the asset price process based on Heston's stochastic volatility model. In recent years, much research has focused on jump-diffusion process models on the so-called ultra-high-frequency data [30] due to more frequent intra-day transactions. More examples of jump-diffusion processes in financial applications can be found in [35].

Bibliography

- [1] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice Hall, New Jersey, 1979.
- [2] B. D. O. Anderson, J. B. Moore, and M. Eslami. Optimal filtering. *IEEE Transactions on Systems, Man and Cybernetics*, 12(2):235–236, 1982.
- [3] C. Andrieu, J. F. G. deFreitas, and A. Doucet. Sequential MCMC for Bayesian model selection. *IEEE Higher Order Statistics Workshop*, 1999.
- [4] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.
- [5] B. Azimi-Sadjadi and P. S. Krishnaprasad. Approximate nonlinear filtering and its application in navigation. *Automatica*, 41(6):945–956, 2005.
- [6] O. E. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory*. Wiley, New York, 1978.
- [7] A. R. Barron and C. Sheu. Approximation of density functions by sequences of exponential family. *The Annals of Statistics*, 19(3):1347–1369, 1991.
- [8] D. S. Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutschemark option. *The Review of Financial Studies*, 9(1):69–107, 1996.
- [9] D. P. Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419, 1975.
- [10] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
- [11] D. P. Bertsekas and D. A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.
- [12] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Optimization and Neural Computation Series. Athena Scientific, 1st edition, 1996.
- [13] A. Beskos, O. Papaspiliopoulos, and G. Roberts. Computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion and reply from the authors). *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 68:1–29, 2006.
- [14] A. Beskos and G. Roberts. Exact simulation of diffusions. *Annals of Applied Probability*, 15:2422–2444, 2005.

- [15] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economics*, 81:637–659, 1973.
- [16] R. W. Brockett. Remarks on finite dimensional nonlinear estimation. In C. Lobry, editor, *Analyse des Systèmes*, 1978.
- [17] A. Brooks, A. Makarenko, S. Williamsa, and H. Durrant-Whytea. Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems*, 54(11):887–897, 2006.
- [18] A. Brooks and S. Williams. A monte carlo update for parametric POMDPs. *International Symposium of Robotics Research*, Nov. 2007.
- [19] A. Budhiraja, L. Chen, and C. Lee. A survey of numerical methods for nonlinear filtering problems. *Physica D: Nonlinear Phenomena*, 230:27–36, 2007.
- [20] A. Budhiraja and D. Ocone. Exponential stability of discrete-time filters for bounded observation noise. *System and Control Letters*, 30:185–193, 1997.
- [21] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.
- [22] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Communications and Control Engineering Series. Springer, New York, 1st edition, 2007.
- [23] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transaction on Signal Processing*, 50(3):736–746, 2002.
- [24] M. H. A. Davis and S. I. Marcus. An introduction to nonlinear filtering. In M. Hazewinkel and J. C. Willems, editors, *Stochastic Systems: The Mathematics of Filtering and Identification and Applications*, pages 53–75, Amsterdam, The Netherlands, 1981. Reidel.
- [25] J. F. G. de Freitas, M. Niranjan, and A. H. Gee. Hierarchical Bayesian Kalman models for regularisation and ARD in sequential learning. Technical Report CUED/F-INFENG/TR 307, Cambridge University Engineering Department, 1998.
- [26] P. T. DeBoer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67, 2005.
- [27] D. deFarias and B. VanRoy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6), 2003.
- [28] P. M. Djuric, J. H. Kotecha, F. Esteve, and E. Perret. Sequential parameter estimation of time-varying non-Gaussian autoregressive processes. *EURASIP Journal on Applied Signal Processing*, 8:865–875, 2002.

- [29] A. Doucet, J. F. G. deFreitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods In Practice*. Springer, New York, 2001.
- [30] R. Engle. The econometrics of ultra-high-frequency data. *Econometrica*, 68:1–22, 2000.
- [31] A. Gelb. *Applied Optimal Estimation*. The M.I.T. Press, 1974.
- [32] W. Gilks and C. Berzuini. Following a moving target - Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society*, 63(1):127–146, 2001.
- [33] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.
- [34] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113, 1993.
- [35] F.B. Hanson. *Applied Stochastic Processes and Control for Jump-Diffusions: Modeling, Analysis and Computation*, chapter Financial Engineering Applications. SIAM Books: Advances in Design and Control Series. Society For Industrial & Applied Mathematics,U.S., 2007.
- [36] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–95, 2000.
- [37] O. Hernandez-Lerma and J. B. Lasserre. *Discrete-Time Markov Control Processes Basic Optimality Criteria*. New York:Springer, 1996.
- [38] S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(327-343), 1993.
- [39] R. A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. press, Cambridge, 1960.
- [40] G.Q. Hu and S.S.-T. Yau. Finite-dimensional filters with nonlinear drift xv: New direct method for construction of universal finite-dimensional filter. *IEEE Transactions on Aerospace and Electronic Systems*, 38(1), 2002.
- [41] J. Hu, M. C. Fu, and S. I. Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55:549–568, 2007.
- [42] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of European Conference on Computer Vision*, pages 343–356, 1996.
- [43] A. Jasra, D. Stephens, A. Doucet, and T. Tsagaris. Inference for Levy driven stochastic volatility models via adaptive SMC. *Preprint*, 2008.

- [44] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. New York: Academic Press, 1970.
- [45] G. Kallianpur. *Stochastic Filtering Theory*. Springer-Verlag, New York, 1980.
- [46] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:33–45, 1960.
- [47] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83:95–108, 1961.
- [48] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, 33(443):1203–1215, 1998.
- [49] D. Koller and U. Lerner. *Sequential Monte Carlo Methods in Practice*, chapter 10: Sampling in factored dynamic systems, pages 445–464. Statistics for engineering and information science. Springer-Verlag, New York, 2001.
- [50] J. H. Kotecha and P. M. Djuric. Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, 51(10):2602–2612, 2003.
- [51] D. P. Kroese, S. Porotsky, and R. Y. Rubinstein. The cross-entropy method for continuous multiextremal optimization. *Methodology and Computing in Applied Probability*, 8:383–407, 2006.
- [52] H. J. Kushner. On the differential equations satisfied by conditional probability densities of Markov processes. *SIAM Journal of Control*, 2:106–119, 1964.
- [53] P. Larranaga, R. Etxeberria, J. A. Lozano, and J. M. Pena. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.
- [54] F. LeGland and N. Oudjane. Stability and uniform approximation of nonlinear filters using the Hilbert metric and application to particle filter. *The Annals of Applied Probability*, 14(1):144–187, 2004.
- [55] E. L. Lemann and G. Casella. *Theory of Point Estimation*. New York: Springer, 1998.
- [56] F. L. Lewis. *Optimal Estimation: With an Introduction to Stochastic Control Theory*. Wiley-Interscience, 1986.
- [57] M. L. Littman. The witness algorithm: Solving partially observable Markov decision processes. Tr cs-94-40, Department of Computer Science, Brown University, Providence, RI, 1994.
- [58] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In A. Doucet, J. F. G. deFreitas, and N. J. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, New York, 2001. Springer-Verlag.

- [59] J. A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer Verlag, New York, 2006.
- [60] P. Naor M. Resh. An inventory problem with discrete time review and replenishment by batches of fixed size. *Management Science*, 10(1):109–118, 1963.
- [61] S. I. Marcus. Nonlinear estimation. In *Systems Control Encyclopedia*, pages 3293 – 3304, Oxford, 1987. Pergamon Press.
- [62] S. I. Marcus. Low dimensional filters for a class of finite state estimation problems with Poisson observations. *System Control Letters*, 1:237–241, January 1982.
- [63] S. I. Marcus and A. S. Willsky. Algebraic structure and finite dimensional nonlinear estimation. *SIAM Journal on Mathematical Analysis*, 9:312–327, April 1978.
- [64] H. Muhlenbein and G. Paaß. From recombination of genes to the estimation of distributions: I. binary parameters. In H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature-PPSN IV*, pages 178–187, Berlin, Germany, 1996. Springer Verlag.
- [65] C. Musso, N. Oudjane, and F. LeGland. Improving regularized particle filters. In A. Doucet, J. F. G. deFreitas, and N. J. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, New York, 2001. Springer-Verlag.
- [66] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003.
- [67] J. D. Pintér. *Global Optimization in Action*. Kluwer Academic Publishers, Dordrecht, The Neitherlands, 1996.
- [68] M. K. Pitt and N. Shephard. Filtering via simulation: Auxilliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.
- [69] J. M. Porta, M. T. J. Spaan, and N. Vlassis. Robot planning in partially observable continuous domains. In *Robotics: Science and Systems*, Cambridge, MA, 2005. MIT.
- [70] J. M. Porta, N. Vlassis, and M. T.J. Spaan amd P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.
- [71] P. Poupart and C. Boutilier. Value-directed compression of POMDPs. *Advances in Neural Information Processing Systems*, 15:1547–1554, 2003.
- [72] W. B. Powell. *Approximate dynamic programming: Solving the curses of dimensionality*. Wiley, New York, 22007.

- [73] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley & Sons, New York, 1994.
- [74] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, 2004.
- [75] N. Roy. *Finding Approximate POMDP Solutions through Belief Compression*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburg, PA, 2003.
- [76] N. Roy and G. Gordon. Exponential family PCA for belief compression in POMDPs. *Advances in Neural Information Processing Systems*, 15(1635-1642), 2003.
- [77] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99:89–112, 1997.
- [78] R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 2:127–190, 1999.
- [79] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer-Verlag, New York, 2004.
- [80] P. J. Schweitzer and A. Seidman. Generalized polynomial approximations in markovian decision problems. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.
- [81] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.
- [82] E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, Palo Alto, CA, 1971.
- [83] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.
- [84] R. L. Stratonovich. *Conditional Markov Processes and Their Application to the Theory of Optimal Control*. Elsevier, New York, 1968.
- [85] R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
- [86] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

- [87] S. Thrun. Monte Carlo POMDPs. *Advances in Neural Information Processing Systems*, 12:1064–1070, 2000.
- [88] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The unscented particle filter. *Advances in Neural Information Processing Systems*, 13, 2001.
- [89] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.
- [90] S.S.-T. Yau and G.Q. Hu. Classification of finite dimensional estimation algebras of maximal rank with arbitrary state-space dimension and mitter conjecture. *International Journal of Control*, 78(10), 2005.
- [91] H. J. Yu. *Approximate Solution Methods for Partially Observable Markov and Semi-Markov Decision Processes*. PhD thesis, M.I.T., Cambridge, MA, 2006.
- [92] M. Zakai. On the optimal filtering of diffusion processes. *Probability Theory and Related Fields*, 11:230–243, 1969.
- [93] Q. Zhang and H. Muhlenbein. On the convergence of a class of estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8:127–136, 2004.
- [94] E. Zhou, M. C. Fu, and S. I. Marcus. A fading memory particle filter. In preparation for submission.
- [95] E. Zhou, M. C. Fu, and S. I. Marcus. Solving continuous-state POMDPs via density projection. *IEEE Transactions on Automatic Control*. Conditionally accepted, full paper.
- [96] E. Zhou, M. C. Fu, and S. I. Marcus. A density projection approach to dimension reduction for continuous-state pomdps. *Proceedings of 47th IEEE Conference on Decision and Control*, page 5576 C 5581, 2008.
- [97] E. Zhou, M. C. Fu, and S. I. Marcus. A particle filtering framework for randomized optimization algorithms. *Proceedings of the 2008 Winter Simulation Conference*, pages 647 – 654, 2008.
- [98] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131:373–395, 2004.