Sequential Decision Making in General State Space models

A dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Nikolas Kantas
Corpus Christi College, February 2009

SIGNAL PROCESSING LABORATORY
DEPARTMENT OF ENGINEERING
UNIVERSITY OF CAMBRIDGE

To Antonis and Fenia

Sequential Decision Making in General State Space models

Nikolas Kantas, Corpus Christi College

Submitted to the University of Cambridge for the degree of Doctor of Philosophy

Summary

Many problems in control and signal processing can be formulated as sequential decision problems for general state space models. However, except for some simple models one cannot obtain analytical solutions and has to resort to approximation. In this thesis, we have investigated problems where Sequential Monte Carlo (SMC) methods can be combined with a gradient based search to provide solutions to online optimisation problems. We summarise the main contributions of the thesis as follows.

Chapter 4 *focuses on solving the sensor scheduling problem when cast as a controlled Hidden Markov Model. We consider the case in which the state, observation and action spaces are continuous. This general case is important as it is the natural framework for many applications. In sensor scheduling, our aim is to minimise the variance of the estimation error of the hidden state with respect to the action sequence. We present a novel SMC method that uses a stochastic gradient algorithm to find optimal actions. This is in contrast to existing works in the literature that only solve approximations to the original problem.*

In Chapter 5 *we presented how an SMC can be used to solve a risk sensitive control problem. We adopt the use of the Feynman-Kac representation of a controlled Markov chain flow and exploit the properties of the logarithmic Lyapunov exponent, which lead to a policy gradient solution for the parameterised problem. The resulting SMC algorithm follows a similar structure with the Recursive Maximum Likelihood (RML) algorithm for online parameter estimation.*

In Chapters $6, 7$ *and* 8*, dynamic Graphical models were combined with with state space models for the purpose of online decentralised inference. We have concentrated more on the distributed parameter estimation problem using two Maximum Likelihood techniques, namely Recursive Maximum Likelihood (RML) and Expectation Maximization (EM). The resulting algorithms can be interpreted as an extension of the Belief Propagation (BP) algorithm to compute likelihood gradients. In order to design an SMC algorithm, in Chapter* 8 *uses a nonparametric approximations for Belief propagation. The algorithms were successfully applied to solve the sensor localisation problem for sensor networks of small and medium size.*

Declaration

The research described in this dissertation was carried out between September 2003 and June 2008. This dissertation is the result of my own work and includes nothing which is the outcome of the work done in collaboration, except where specifically indicated in the text. The dissertation contains no more than 65,000 words and 70 figures.

Nikolas Kantas

Acknowledgements

I would like to thank my supervisors Prof. Arnaud Doucet and Dr. Sumeetpal S. Singh for their supervision and assistance. I have greatly benefitted from their ideas and insight. Their advice, encouragement and support have been invaluable. This Ph.D. was supported by DIF-DTC and a Cambridge European Trust Scholarship.

I would also like to thank Dr. Simon Hill for his detailed comments on how to improve the presentation of this thesis, Dr. George Poyiadjis and Dr. Frédéric Desobry for their daily advice on scientific and general matters, and all the members of the Signal Processing lab, especially Dr. Ali Taylan Cemgil, (soon-to-be Dr.) Nick Whiteley, Dr. Mark Briers, and last but not least Dr. Paris Kaimakis, for the very pleasant daily interaction and many stimulating discussions.

Also, many thanks to all my friends, especially long-time brothers-in-arms Stelios and George, and of course my partner Pepi. Without all of them this journey would not have been near as exciting and fruitful. Finally, I would like to thank my parents for their endless support and encouragement all these years. Without it, probably most of the effort put in this project would not have been possible.

Keywords

general state space models; hidden markov models; parameter estimation; stochastic control; graphical models; particle filter; sequential monte carlo; stochastic approximation; recursive maximum likelihood; partially observed markov decision process;

Publications

Journal Papers:

S. Singh, N. Kantas, B. Vo, A. Doucet and R. Evans, Simulation-Based Optimal Sensor Scheduling with Application to Observer Trajectory Planning,*Automatica*, vol. 43, no. 5, pp. 817-830, 2007

Conference Papers:

N. Kantas, S. S. Singh, A. Doucet, Distributed Online Self-Localization and Tracking in Sensor Networks, *Proceedings ISPA* 2007, Istanbul, Turkey

N. Kantas, S. S. Singh, A. Doucet, Distributed Self Localisation of Sensor Networks using Particle Methods, *Proceedings of NSSPW 2006*, Cambridge

N. Kantas, S. Singh, A. Doucet, A Distributed Recursive Maximum Likelihood Implementation for Sensor Registration, *Fusion* 2006, Florence.

S. Singh, N. Kantas, B. Vo, A. Doucet and R. Evans, Simulation-Based Optimal Sensor Scheduling with Application to Observer Trajectory Planning, *Proceedings of CDC-ECC*, Dec 2005, Sevilla.

Technical Reports:

S. Singh, N. Kantas, B. Vo, A. Doucet and R. Evans, On the convergence of a stochastic optimisation algorithm for optimal observer trajectory planning, Technical Report CUED/F-INFENG/TR 522, University of Cambridge, Department of Engineering, March 2005.

Contents

List of Figures

List of Tables

List of Acronyms

CV	**C**ontrol **V**ariates
EM	**E**xpectation **M**aximisation
GM	**G**raphical **M**odels
FK	**F**eynman **K**ac
HMM	**H**idden **M**arkov **M**odel
IS	**I**mportance **S**ampling
KL	**K**ullback **L**eibler
MAP	**M**aximum **a**-**P**osteriori
MC	**M**onte **C**arlo
MCMC	**M**arkov **C**hain **M**onte **C**arlo
MDP	**M**arkov **D**ecision **P**rocess
ML	**M**aximum **L**ikelihood
MLE	**M**aximum **L**ikelihood **E**stimator
MMSE	**M**inimum **M**ean **S**quare **E**rror
MSE	**M**ean **S**quare **E**rror
PF	**P**article **F**ilter
POMDP	**P**artially **O**bserved **M**arkov **D**ecision **P**rocess
RML	**R**ecursive **M**aximum **L**ikelihood
SA	**S**tochastic **A**pproximation
SIS	**S**equential **I**mportance **S**ampling
SISR	**S**equential **I**mportance **S**ampling **R**esampling
SMC	**S**equential **M**onte **C**arlo

1 Introduction

1.1 Motivation

Stochastic processes can be a very effective modeling tool for a number of real-world phenomena. Uncertainty in real-world phenomena can be well captured using probability theory and statistics. This is achieved using a rigorous mathematical framework, where particular model variables and parameters can be used to describe the process with adequate generality. This makes any problem formulation intuitive and a scientist can benefit from many algorithms that deal with the computational side of the problem. The advent of accessible computing power in the last two decades and the development of sophisticated simulation tools have revolutionised the field of statistical modeling and opened up new directions for statistical inference of many complex systems. In this thesis we aim to use such methods from computational statistics to solve difficult problems that arise in the field of Engineering.

In the context of dynamical models, the modeled process generally produces a sequence of observable outputs over time. The process can be thought to evolve in time either continuously

or discretely, but in this thesis we shall consider only the discrete time case. Dynamic models attempt to statistically describe and analyse processes based on the observed time series. Moreover, they can include decision variables or control inputs, where the decision variables are chosen to optimise a particular criterion. Such decision or control problems can describe a broad class of phenomena, where an external user can influence the evolution of the process to meet certain specifications. When a statistical framework of analysis is used these problems are generally referred to as stochastic control or decision making problems.

In many dynamical models a hidden state of interest evolves in a dynamic fashion, typically having a simple Markovian structure, i.e. the current state of the process is influenced only by the previous one. Such a setting arises often in statistics, engineering and other applied sciences. Recently, there has been a surge of interest in Sequential Monte Carlo (SMC) methods, also known as Particle Filtering methods, to perform sequential state estimation in non-linear non-Gaussian models [48, 53, 85, 103]. SMC methods are a set of simulation-based techniques that recursively generate and update a set of weighted samples that provide approximations to the posterior probability distributions of interest.

In this thesis we shall attempt to employ SMC to solve decision problems for dynamical models with general state and observation spaces. We should say that we have noted a significant absence of algorithms for general state spaces in the literature of stochastic control. SMC can provide a framework and many computational tools to deal with the difficulties of stochastic control in general state spaces. We emphasise that SMC methods need only very weak assumptions on the model used and these do not include any particular restrictions on the state or observation space.

We shall also consider some online parameter estimation problems. By the term online we mean that as more observations of the latent variable appear available, the parameter is estimated sequentially in time using a recursion with a fixed amount of computation and memory requirement. These problems are essentially sequential optimisation problems due to their recursive and online character. Under the assumption that the model parameters are known, numerous SMC algorithms have been proposed and successfully applied to many practical problems, over the last decade. In real-world applications however, the model parameters are usually unknown and need to be estimated from the observed data, before SMC methods can be applied for state estimation. This is a severe limitation in the practicality of standard SMC

algorithms, hence the development of parameter estimation methods is of significant importance. Online parameter estimation has been studied much by the statistical community but without significant success. One popular approach was to include the unknown parameter with the hidden state and cast the problem as a filtering one [56, 105, 154]. This proved to be inefficient due to a degeneracy problem inherent in the standard SMC algorithm. Recently, in [136], one can find a detailed study on developing advanced SMC algorithms for parameter estimation problems, which avoid limitations and problems of previous approaches. We shall build on the methodology and ideas used in [136] to extend them also for distributed environments, as well generalising them to fit a particular stochastic control framework.

1.2 Applications and Real Engineering Problems

In this section we shall briefly describe some of the applications, which shall be dealt with in this thesis. Most of them arise from the area of target tracking. While solving these problems we have to say that a consistent effort has been made to provide generic solutions that fit a general statistical framework and are not just useful to solve the particular application. This is important as our algorithms can be reused in other problems from the areas of robotics, computer vision, or finance, which fit the same general framework.

Consider the following typical scenario, which arises from target tracking. A manoeuvering target is to be tracked based on noise corrupted measurements of the target's state. These can be received by one or more moving observers. The quality of the target state observations can be improved by the appropriate positioning of the observers relative to the target during tracking. The question of optimal observer trajectory planning naturally arises, i.e. how should the observer manoeuvre relative to the target in order to optimise the tracking performance? This problem of optimal sensor manoeuvering is both a filtering and control problem. Another typical and closely related problem in target tracking is sensor scheduling. Here an array of static sensors is used to track the target. However sensors provide noise corrupted measurements and due to bandwidth limitations only one of them can be used at each time. Now the problem consists of choosing at each time instant one active sensor from which we receive a measurement with the aim of optimising the tracking performance.

Another problem which is solved in this thesis is the problem of distributed self localisation for sensor networks. Consider a network of sensors that are deployed for target tracking.

Distributed collaborative tracking can be achieved if each node is able to accurately determine the position of its neighboring nodes in its local frame of reference. Due to cost, size or power consideration say this has to be achieved without the need of a Global Positioning System (GPS) or direct measurements of the distance between neighboring nodes. Then the problem of recovering the relative coordinate of a neighbouring sensor can be cast as a parameter estimation problem in a distributed environment. Initially as nodes are not localized they behave as independent trackers. As the tracking task is performed on objects that traverse the field of view of the sensors, information can be shared between nodes in a way that allows them to self-localize. Even though the target's true trajectory is not known to the sensors, localization can be achieved in this manner because the same target is being simultaneously measured by the sensors.

1.3 Scope of this thesis

In this thesis we shall propose algorithms for solving stochastic optimisation problems appearing in the presented applications. In a sense, optimal sensor manoeuvering, sensor scheduling, and self-localisation, share a similar structure as in each case a certain performance criterion needs to be optimised. Moreover, depending on the application, the solutions proposed by an optimiser can be computed and implemented in real time. We shall refer to this approach as online optimisation in contrast to the offline case, where the solutions are computed before or sometimes after the actual process takes place. For example, in target tracking while collecting measurements of the position of the target, one should aim to perform sensor manoeuvering or sensor scheduling online as in any control problem. Offline computations done before initiating a real time experiment can assist this task, but as sensors receive measurements one should update their control inputs or actions so that the target tracking is kept optimal. Similarly, the self-localisation problem is a parameter estimation problem that can be solved both online while the sensor network receives measurements of the target, as well as offline after a set of measurements have been collected. In this thesis we will focus mostly on online, which are more challenging and they appear to be more useful for applications.

The scope of this thesis is to derive novel algorithms for online stochastic optimisation problems. By using a general mathematical framework to describe each problem of interest, our algorithms should be able to be used for any other problems that fit the specific framework.

Therefore, we have made an effort to present generic solutions and to keep the class of problems we are addressing as general as possible. The main statistical modeling tool will be that of *general state space models*, which is commonly used both for stochastic control as well as parameter estimation. As was mentioned in the previous paragraph for specific examples arising from the applications of interest, control and online parameter estimation problems follow a similar structure. This holds in general, as in both cases a similar framework as well as computational tools can be used to decide how a control or a parameter estimate should be updated sequentially in time. Therefore, in the title of this thesis it was preferred to use the term sequential decision making as a more general phrase that encompasses stochastic control or parameter estimation.

The methodology will involve combining two powerful approaches, gradient based methods and computer simulation. Under mild regularity assumptions, gradient based methods are guaranteed to converge to a local optimum of the performance criterion. This is sufficient for the applications we shall consider. In addition, they can be combined with recent simulation based methods, to produce a new type of stochastic gradient methods. This is done without losing any theoretical guarantees, or needing any additional strong assumptions, since the simulation based methods we shall be using rely on very mild conditions. This can allow us to address nonlinear non-Gaussian sequential decision problems, which is a very challenging task overlooked by most of the control literature. We shall be using predominantly Sequential Monte Carlo (SMC) as the tool for simulation. SMC methods can naturally handle general state space models and there are many theoretical results available on their asymptotic behaviour, providing thus valuable insight for the design of particular algorithms.

1.4 Thesis Organisation

Besides this introductory chapter, this thesis is comprised of 7 chapters and a concluding chapter. Thematically, the material can be partitioned into three parts:

Part I: Review of Monte Carlo methods for inference in general state spaces (Chapters $2, 3$).

Part II: Control using policy gradient and SMC (Chapters $4, 5$).

Part III: Online distributed parameter estimation for Graphical Models (Chapters $6, 7, 8$)

The individual chapters are structured as follows:

Chapter 2: Monte Carlo Methods

This chapter introduces Bayesian inference, Monte Carlo methods and some standard sampling techniques, such as Importance Sampling (IS) and Markov Chain Monte Carlo (MCMC). Sequential Monte Carlo (SMC) methods are investigated in detail and Feynman-Kac models are introduced.

Chapter 3: General State Space Models

In this chapter we will present general state space models, which can be used for state and parameter inference as well as for control. These models are also known as Hidden Markov Models (HMM) and encompass a broad class of dynamic models. Graphical Models are also introduced so that an extension of general state space models for problems regarding distributed systems is proposed.

Chapter 4: Simulation-Based Optimal Sensor Scheduling with Application to Observer Trajectory Planning

In this chapter, we address the sensor scheduling problem when cast as a controlled Hidden Markov Model. We consider the case in which the state, observation and action spaces are continuous. This general case is important as it is the natural framework for many applications. In sensor scheduling, our aim is to minimise the variance of the estimation error of the hidden state with respect to the action sequence. We present a novel simulation-based method that uses a stochastic gradient algorithm to find optimal actions. This is in contrast to existing works in the literature that only solve approximations to the original problem.

Chapter 5: Risk Sensitive Control using Policy Gradient

In Chapter 5 we present how an SMC can be used to solve a risk sensitive control problem. We adopt the use of the Feynman-Kac representation of a Markov chain flow and exploit the properties of the logarithmic Lyapunov exponent, which leads to a policy gradient solution for the parameterised problem. The resulting SMC algorithm follows a similar structure with the Recursive Maximum Likelihood (RML) algorithm used for online parameter estimation.

Chapter 6: Self Localisation of Sensor Networks for Target Tracking

Chapter 6 presents an overview of the self localisation problem for sensor networks deployed for target tracking. Graphical models are integrated with general state space models for the purpose of decentralised inference. We concentrate on the distributed parameter estimation problem and describe a decentralised version of Recursive Maximum Likelihood (RML), which can be implemented in dynamic graphical models through the propagation of suitable messages that are exchanged between neighboring nodes of the graph. The resulting algorithm can be interpreted as an extension of the celebrated Belief Propagation algorithm to compute the likelihood gradient. This algorithm is applied to solve the sensor localisation problem for sensor networks.

Chapter 7: Distributed Localisation and Tracking For Linear Gaussian Sensor Networks

In Chapter 7 we continue the work of Chapter 6, and emphasise on the linear Gaussian case. We also illustrate how the distributed framework established in Chapter 5 can incorporate Expectation Maximization (EM), which is another popular algorithm for estimating unknown static parameters. The underlying methodology is generic and we show how it can be applied to solve the sensor localisation problem for sensor networks of different sizes.

Chapter 8: Distributed Localisation and Tracking For Nonlinear Non-Gaussian Sensor Networks

In Chapter 8, we consider the problem of estimating static parameters distributed in a graph for the general nonlinear non-Gaussian case. For the nonlinear Gaussian case, a distributed Extended Kalman filter (EKF) based on the material of Chapter 7 is presented. For the nonlinear non-Gaussian case, we extend previous work already developed for problems involving dynamic filtering for distributed environments. We consider the decentralised static parameter inference problem using distributed RML. We propose a Sequential Monte Carlo algorithm combined with Nonparametric Belief Propagation (NBP), which is used to approximate the messages passed between adjacent nodes. The resulting algorithm can be thought as as an extension of NBP to compute likelihood gradients and was applied to solve the sensor localisation problem for sensor networks.

Chapter 9: Conclusions and Future directions

This last chapter concludes the thesis with a review of the key components and contributions of the research described. Also, potential areas for further research are discussed.

Part I

Review of Monte Carlo Inference for General State Spaces

2

Monte Carlo Methods

Summary. *In this chapter a brief review of Monte Carlo methods is presented. The chapter starts with an overview of Bayesian Inference followed by an introduction on the basic principles of Monte Carlo and discrete time Markov Chains. Other popular Monte Carlo methodologies are also presented, such as Importance Sampling (IS) and Markov Chain Monte Carlo (MCMC). Finally, this chapter describes Sequential Monte Carlo methods (SMC) and Feynman-Kac Models.*

2.1 Introduction

Statistical modelling has become increasingly important in the areas of engineering, finance, and computer science. One can use probability distributions and statistical methods to deal with the inherent uncertainty in real world problems. Bayesian statistics has been an extremely popular framework in which to perform statistical inference. From a Bayesian perspective there is no fundamental distinction between the observed and unobserved variables and the parameters of a statistical model. All are treated as random variables. In addition, all expert knowledge or prior information on the problem is formally incorporated in the analysis using prior distributions. Note that this does not necessary imply that this prior information has to be precise, but rather adds another degree of freedom to the investigation.

For many years, the applicability of Bayesian analysis was limited by the fact that the high-dimensional integrations or maximisations involved rarely admit tractable forms. This pushed researchers towards devising systematic means for designing suitable approximations, so that resorting to simplified models or ad hoc methods could be avoided. More rigorous Bayesian implementations were achieved when computer simulation was employed, following the dramatic increase of computational power of computers starting during the period of the late 80's. This increase in computational power made the use of Monte Carlo methods for integration and maximisation over complex multidimensional functions a practical approach. This further encouraged the development of numerical Bayesian methods. Monte Carlo methods characterise the probability distributions of interest by using a large number of samples from them. In most cases, the distributions are complex and multivariate and are usually known only up to a normalising constant. As a result direct sampling is not possible in general and more sophisticated techniques need to be used, such as Importance Sampling [103, 140], Markov Chain Monte Carlo (MCMC) [63, 73, 115, 140] and Sequential Monte Carlo (SMC) methods [34, 47, 48, 103].

The organisation of this chapter is structured as follows. In Section 2.2 we present briefly the key ideas behind Bayesian inference. Then in Sections 2.3, 2.4, and 2.6 we introduce traditional Monte Carlo methodologies such as Importance sampling and Markov Chain Monte Carlo (MCMC). Section 2.5 also contains a brief review on Markov chains theory defined on general state spaces. In Section 2.7, we review Sequential Monte Carlo methods, which is the basis of all computation in this thesis. In Section 2.8 we introduce the Feyman-Kac model [34] and show how interacting particle approximations can be derived for probability measures propagated

according to this model.

2.2 Bayesian inference

In the Bayesian framework, all the unknown parameters of a problem are assumed to be random variables and admit a prior distribution. Unknown parameters can include unobserved model components, actual model parameters and/or missing data. Under a Bayesian framework, the prior distribution of these unknown parameters is updated by means of Bayes' theorem, which can be informally put as

$$\text{posterior} \propto \text{likelihood} \times \text{prior},$$

where the likelihood is the probability distribution of the observed data, y, conditioned on the unknown parameters, x, and the particular model $\mathcal{M}$ used. This results in a posterior distribution of x conditioned on the observed data, $y \in \mathcal{Y}$, and model $\mathcal{M}$. Let x denote the vector of the unknown parameters taking value on some measurable space $(\mathcal{X}, \mathcal{E})$, the posterior distribution will be given by Bayes' law as

$$p(x|y, \mathcal{M}) = \frac{p(y|x, \mathcal{M})p(x|\mathcal{M})}{p(y|\mathcal{M})}, \tag{2.1}$$

where $p(y|x, \mathcal{M})$ is the likelihood of the observed data, $p(x|\mathcal{M})$ is the prior distribution of the unknown parameters and $p(y|\mathcal{M})$ is the probability distribution of the data, which is also referred as the evidence or the normalisation constant.

Although this approach makes the problem formulation quite elegant, in general it suffers from the fact that the normalisation constant, given as

$$p(y|\mathcal{M}) = \int_{x \in \mathcal{X}} p(y|x, \mathcal{M})p(x|\mathcal{M})dx,$$

cannot be calculated analytically in most cases of interest and also may involve high dimensional integration. This makes it necessary to resort to approximations. Most applications of Monte Carlo methods are found in the evaluation of high dimensional integrals for Bayesian inference.

Another related problem to calculating the normalising constant is that of marginalisation. A useful feature of the Bayesian framework is that parameters of no interest, commonly referred to as *nuisance* parameters, can be removed by marginalisation. If for example $x = (x_1, x_2)$

and x_2 is a nuisance parameter, the marginal posterior density of x_1 can be obtained from the joint posterior as

$$p(x_1|y,\mathcal{M}) = \int p(x_1,x_2|y,\mathcal{M})\,dx_2.$$

Again, this integral may not be available analytically, so one has to resort to approximation.

2.2.1 Bayesian estimation

Bayesian methods allow the construction of appropriate posterior probability distributions of the unknown parameter x after setting a prior. However, often particular statistics, such as modes, moments and quantiles, need to be estimated so that one can use them in applications. This means we might want to summarise the distribution $p(x|y,\mathcal{M})$ through a *point estimate* for the value of the unknown parameter vector x. This point estimate is itself a random variable and can be estimated from $p(x|y,\mathcal{M})$ in a number of ways. Two simple and commonly used methods for estimating x are the *Maximum Likelihood* (ML) and *Maximum A-Posteriori* (MAP) estimation methods.

Maximum likelihood requires the computation of the mode of the likelihood $p(y|x,\mathcal{M})$. The resulting *maximum likelihood estimator* (MLE) $\widehat{x}_{\mathrm{ML}}$ can be written as

$$\widehat{x}_{\mathrm{ML}} = \arg\max_x\, p(y|x,\mathcal{M}).$$

Maximum A-Posteriori estimation requires the computation of the mode of the posterior distribution $p(x|y,\mathcal{M})$. The underlying maximisation results to the *maximum a-posteriori estimator*, $\widehat{x}_{\mathrm{MAP}}$, which can be written as

$$\widehat{x}_{\mathrm{MAP}} = \arg\max_x\, p(x|y,\mathcal{M}).$$

If no prior information about x is available, and the prior is vague, e.g. a uniform distribution, the MAP estimator reduces to an MLE.

Suppose that one would like to obtain an estimator of or some function $h(x)$. A more general approach would be to follow a decision theoretic approach and use *loss or risk functions*, [140]. We specify a certain loss function $L(\delta,x)$, which represents the loss or risk incurred by using δ as an estimator of $h(x)$. From a Bayesian perspective we would like to minimise the expected loss or *Bayes risk*

$$\int_{\mathcal{X}}\int_{\mathcal{Y}} L(\delta,x)p(y|x,\mathcal{M})p(x|\mathcal{M})dydx.$$

The resulting estimator $\widehat{x}$ would be

$$\widehat{x} = \arg\min_{\delta} \int_{\mathcal{X}} L(\delta, x) p(x|y, \mathcal{M}) dx.$$

A popular choice of loss function is a quadratic

$$L(\delta, x) = (h(x) - \delta)^T \Sigma (h(x) - \delta),$$

where Σ is a positive definite matrix. Let $\widehat{x}$ be the resulting estimator. Then,

$$\widehat{x} = \arg\ \min_{\delta} \int_{\mathcal{X}} (h(x) - \delta)^T \Sigma (h(x) - \delta) p\left(x|y, \mathcal{M}\right) dx,$$

This is equivalent to

$$\widehat{x} = \int_{\mathcal{X}} h(x) p(x|y, \mathcal{M}) \, dx.$$

If $h(x) = x$, then the estimation is also known as *Minimum Mean Square Error* (MMSE) or Minimum Variance estimation. The estimator is simply the the posterior mean.

2.2.2 Bayesian Model Selection

Bayesian inference assumes the use of a specific probability model $\mathcal{M}$. If the particular model is not appropriate for the problem, the inference mechanism will not lead to valid results. Hence an appropriate model selection procedure is crucial. Bayesian inference can be used to select the most plausible model between a set of different candidate models, $\{\mathcal{M}_k\}_{k=1}^K$, where K is a finite and positive integer. The best model in the Bayesian sense, $\mathcal{M}^*$, is the one that maximises the posterior density of the model given the observed data,

$$\mathcal{M}^* = \arg\ \max_{\mathcal{M}_k} p(\mathcal{M}_k|y),$$

where by Bayes' theorem

$$p(\mathcal{M}_k|y) \propto p(y|\mathcal{M}_k) p(\mathcal{M}_k).$$

As it can be seen, besides a subjective model prior $p(\mathcal{M}_k)$, model selection depends on the normalisation constant $p(y|\mathcal{M}_k)$. This gives the intuition behind using the term *model evidence* for the normalisation constant.

As presented in [6], a common way of comparing pairwise models, say i and j, is to use the posterior odds ratio

$$B_{ij} = \frac{p(y|\mathcal{M}_i) p(\mathcal{M}_i)}{p(y|\mathcal{M}_j) p(\mathcal{M}_j)}.$$

Clearly if $B_{ij} > 1$ model $\mathcal{M}_i$ is favored, otherwise if $B_{ij} < 1$ model $\mathcal{M}_j$ is favored.

2.2.3 Motivation for Monte Carlo methods

Bayesian methods are very useful since they can incorporate any previous knowledge in the prior distribution. On the other hand, the posterior may not be analytically tractable. Similarly, when performing estimation it might be hard to handle loss functions other then a quadratic or some other convinient convex function that allows explicit analytical computations. For long time this has lead to favoured types of priors, which allowed analytical computations. Such an example are *conjugate priors*, for which the corresponding posterior distributions are themselves members of the original prior family. This would allow the Bayesian update to be performed by appropriately updating the parameters of the distributions.

When analytical computation is not an option one has to rely on approximations. Deterministic techniques suffer in terms of computational complexity as it increases substantially with the dimension of the parameter space. For example, standard numerical integration methods suffer from a computational complexity that grows exponentially with the dimension of the integration region. Apart from some simple situations and low dimensions, these numerical approximations are inadequate and inefficient. The only alternative is to use methods based on simulation. During the last few decades, various simulation techniques based on the Monte Carlo principle have been developed to efficiently address Bayesian inference problems (and not only) whose analytical computations cannot be performed.

2.3 Monte Carlo Integration

Monte Carlo methods are powerful numerical techniques that deal with the integration and optimisation of complex, multi-dimensional functions. Here we will consider Monte Carlo integration, which approximates any complex integral, such as an expectation, with the aid of simulation. We are interested in approximating the following integral with respect to a target density $\pi(x)$ defined on a measurable space $(\mathcal{X}, \mathcal{E})$,

$$\pi(\varphi) = \mathbb{E}_\pi[\varphi(X)] = \int_{\mathcal{X}} \varphi(x)\pi(dx)\,, \tag{2.2}$$

where φ is some measureable function, such that $\varphi : \mathcal{X} \longrightarrow \mathbb{R}^{n_x}$.

2.3.1 Perfect Monte Carlo

Assume a set of N independent identically distributed (iid) random samples $\{X^{(i)}\}_{i=1}^{N}$ can be drawn from $\pi(x)$. Then, an empirical estimate of π based on *perfect Monte Carlo* sampling can be given by

$$\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^{(i)}}(dx),$$

where δ denotes the Dirac delta function. The integral in (2.2) can be approximated by the sample mean

$$\widehat{\pi}(\varphi) = \int_{\mathcal{X}} \varphi(x)\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N} \varphi(X^{(i)}). \tag{2.3}$$

This implies that a continuous distribution is approximated by a discrete one with random support. The empirical density $\widehat{\pi}(x)$ will form an increasingly better approximation of the continuous density $\pi(x)$ as $N \to \infty$. For independent samples[1] the following convergence holds,

$$\widehat{\pi}(\varphi) \xrightarrow[N\to\infty]{a.s.} \pi(\varphi),$$

by the Law of Large Numbers, where $\xrightarrow{a.s.}$ denotes almost sure convergence. $\widehat{\pi}(\varphi)$ is also an unbiased estimator of $\pi(\varphi)$.

Perfect Monte Carlo Variance The variance of $\widehat{\pi}(\varphi)$ is given by

$$\begin{aligned} var(\widehat{\pi}(\varphi)) &= \frac{var_{\pi}[\varphi(x)]}{N} \\ &= \frac{1}{N}\int_{\mathcal{X}} [\varphi(x) - \pi(\varphi)]^2 \pi(dx) \end{aligned} \tag{2.4}$$

Thus accuracy improves as more samples are taken and if $var_{\pi}[\varphi(x)] < \infty$ the following central limit theorem holds,

$$\sqrt{N}(\widehat{\pi}(\varphi) - \pi(\varphi)) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, var_{\pi}[\varphi(x)]),$$

where $\Longrightarrow$ denotes convergence in distribution.

Intuitively, the advantage of Monte Carlo integration comes from the fact that the samples $\{X^{(i)}\}_{i=1}^{N}$ are automatically chosen to be in the important regions of the state space. The rate of

[1] Even in cases where the samples are dependent, it is still possible to obtain convergence under some weak assumptions. This applies, for example, when the dependent samples are drawn from a suitable Markov Chain.

convergence of $\frac{1}{\sqrt{N}}$ is independent of the dimensions of x as opposed to typical deterministic numerical integration methods, [140]. Unfortunately, it might not be possible to directly sample from π, which brings up the issue of devising means of obtaining samples of π using the aid of another density, say q, we can sample from.

2.3.2 Monte Carlo Sampling

The requirement of perfect Monte Carlo is the ability to draw iid samples from the target distribution $\pi(x)$. For most practical models this is not possible. In such cases to use Monte Carlo for integration we should use more sophisticated sampling techniques, such as Rejection Sampling [103], Importance Sampling [103, 140] and Markov Chain Monte Carlo (MCMC) methods [63, 73, 115].

Rejection sampling

Rejection sampling is a simple idea that allows one to sample from a distribution $\pi(x)$ known up to a proportionality constant. Let $\pi(x) = \frac{l(x)}{Z}$ hold, where Z is the normalisation constant. First, we aim to find a different distribution $q(x)$, which is easy to sample from and whose support that includes that of $\pi(x)$. Also, $q(x)$ should be such that $l(x) \leq Mq(x)$ is satisfied for all x and some constant $M < \infty$. We can then use $q(x)$ as a proposal distribution to obtain samples $X^{(i)}$ and then use an auxiliary variable sampled from a uniform distribution $\mathcal{U}(0,1)$ to determine whether $X^{(i)}$ is a sample of $\pi(x)$. Rejection sampling can be summarised by the following algorithm.

Algorithm 2.1 ***Rejection Sampling algorithm:***

1. *Sampling Step*

- *Sample* $X^{(i)} \sim q$
- *Sample* $U^{(i)} \sim \mathcal{U}(0,1)$

2. *Accept-Reject Step*

- *If* $U^{(i)} \leq l\left(X^{(i)}\right) / Mq\left(X^{(i)}\right)$ *accept* $X^{(i)}$ *as a sample of* π
- *Else, reject* $X^{(i)}$ *and go back to step 1*

Repeat

It can verified by considering the distribution of the accepted samples that the set of accepted samples will be distributed according to $\pi(x)$. Also, one can show through a simple manipulation that the probability of the accepted samples equals to $\frac{1}{M}$. As one would expect, the size of M shows how efficiently our samples are used and resembles how close q imitates π. Unfortunately, in practice a rejection sampling approach is usually viable only for low-dimensional problems as it tends to waste sampling effort when rejection levels are high. In the literature, one can find improved rejection sampling approaches, which use either cheap to evaluate *squeezing functions* to tightly bound $l(x)$ on both sides or an adaptive mechanism to obtain bounds from the samples themselves, leading to *adaptive rejection sampling*. For more details see [140].

2.4 Importance sampling (IS)

In contrast to rejection sampling, we would like to use every sample, rather than discarding it. Importance sampling techniques achieve this by weighting each sample according to how "well" it resembles the target distribution. Although this is a rather informal statement, it should made clearer later in this section. As before we can introduce an instrumental distribution q whose support includes that of π, i.e. $\pi << q$, and such that the Radon-Nikodym derivative $\frac{d\pi}{dq}$ is well defined and bounded. In [140] the authors refer to the importance sampling fundamental identity, as

$$\pi(\varphi) = q(\frac{d\pi}{dq}\varphi).$$

Let ν be some σ-finite measure, such that $\pi << \nu$ and $q << \nu$. Then $\frac{d\pi}{dq} = \frac{d\pi}{d\nu}/\frac{dq}{d\nu}$.To ensure $\pi << q$ we have to select q so that $\frac{dq}{d\nu} > 0$ for $\frac{d\pi}{d\nu} > 0$. This formalises the well known rule of thumb found in [61], that ones has to make sure that the tails of the importance function q are heavier than the tails of the target distribution π. Let also $q(dx) = q(x)\,\nu(dx)$ and $\pi(dx) = \pi(x)\,\nu(dx)$, so that the reformulating the integral of interest in (2.2) is

$$\pi(\varphi) = \int_{\mathcal{X}} \varphi(x)\frac{\pi(x)}{q(x)} q(x)\,\nu(dx) = \int_{\mathcal{X}} \varphi(x)w(x)q(x)\nu(dx), \tag{2.5}$$

where

$$w(x) = \frac{\pi(x)}{q(x)}$$

is known as the *importance weight*. Using a set of iid samples $\{X^{(i)}\}_{i=1}^{N}$ sampled from $q\,(dx)$ leads to the Monte Carlo estimate

$$\widehat{\pi}(\varphi) = \int_{\mathcal{X}} w\,(x)\,\varphi(x)\widehat{q}(x)\,dx = \frac{1}{N}\sum_{i=1}^{N} w\left(X^{(i)}\right)\varphi\left(X^{(i)}\right), \tag{2.6}$$

which is an unbiased estimator that converges to (2.5) for the same reason as the perfect Monte Carlo estimator. In this case the empirical distribution estimate for q and π are now given by the weighted set of samples

$$\widehat{q}(dx) = \frac{1}{N}\sum_{i=1}^{N}\delta_{X^{(i)}}\,(dx)\,,$$
$$\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N} w\left(X^{(i)}\right)\delta_{X^{(i)}}\,(dx)\,.$$

This importance sampling approach also works if π is only known up to a proportionality constant. In this case we can use the approximation.

$$\widehat{\pi}(\varphi) = \frac{\frac{1}{N}\sum_{i=1}^{N} w\left(X^{(i)}\right)\varphi(X^{(i)})}{\frac{1}{N}\sum_{j=1}^{N} w\left(X^{(j)}\right)} = \sum_{i=1}^{N}\widetilde{w}\left(X^{(i)}\right)\varphi\left(X^{(i)}\right), \tag{2.7}$$

where $\widetilde{w}\left(X^{(i)}\right) = w\left(X^{(i)}\right)/\sum_{j=1}^{N} w\left(X^{(j)}\right)$ is the normalised importance weight. The result in (2.7) is based on the fact that $\frac{1}{N}\sum_{j=1}^{N} w\left(X^{(j)}\right) \xrightarrow[N\to\infty]{a.s.} 1$. This strategy makes use of the ratio of two unbiased estimator, a bias is introduced for the finite number of samples case. However, the estimator is asymptotically consistent and can provide lower variance estimates than the standard Importance sampling estimator.

Controlling the variance of Importance sampling The variance of $\widehat{\pi}(\varphi)$ in (2.6) is given by

$$\begin{aligned} var_q\left(\widehat{\pi}(\varphi)\right) &= \frac{1}{N} var_q\left[w\,(x)\,\varphi(x)\right] \\ &= \frac{1}{N}\left[\int_{\mathcal{X}} w\,(x)\,\varphi^2(x)\pi\,(x)\,dx - \mathbb{E}_{\pi}^{2}\left(\varphi\,(x)\right)\right]. \end{aligned} \tag{2.8}$$

Clearly, the variance of the estimate in (2.6) will depend on the specific choice of $q(x)$.

Proposition 2.4.1 *Assuming* $supp\,(q) \supset supp\,(\varphi\pi)$ *holds, an optimal importance density that minimises the variance of the estimator in (2.6) and is given by*

$$q(x) = \frac{|\varphi\,(x)|\,\pi\,(x)}{\int_{\mathcal{X}} |\varphi\,(x)|\,\pi\,(x)\,dx}.$$

Proof. From [140] : The variance of $w(x)\varphi(x)$ with respect to $q\left(x\right)$ is given as

$$Var_q\left[w\left(x\right)\varphi(x)\right] = \mathbb{E}_q\left[w^2(x)\varphi^2(x)\right] - \mathbb{E}_q^2\left[w(x)\varphi(x)\right],$$

where from Jensen's inequality the first term gives

$$\mathbb{E}_q\left(\frac{\pi^2(x)\varphi^2(x)}{q^2(x)}\right) \geq \mathbb{E}_q^2\left(\frac{\pi(x)\left|\varphi\left(x\right)\right|}{q(x)}\right) = \mathbb{E}_\pi^2\left(\left|\varphi\left(x\right)\right|\right)$$

and the for the second term,

$$\mathbb{E}_q^2\left[w(x)\varphi(x)\right] = \mathbb{E}_\pi^2\left[\varphi(x)\right].$$

The lower bound is attained at

$$q(x) = \frac{\left|\varphi\left(x\right)\right|\pi\left(x\right)}{\int\limits_{\mathcal{X}}\left|\varphi\left(x\right)\right|\pi\left(x\right)dx},$$

where the variance becomes zero. ■

Note that importance sampling can in principle achieve a lower variance than the variance of the perfect Monte Carlo sampling estimator in (2.4). In [91] the variance of the IS estimator, $var_q\left[\widehat{\pi}(\varphi)\right]$, was approximated by

$$var_q\left[\widehat{\pi}(\varphi)\right] \approx \frac{var_\pi\left[\varphi\left(x\right)\right]\left\{1 + var_q\left[w\left(x\right)\right]\right\}}{N}.$$

The above approximation has been suggested in to provide an easy way of monitoring the efficiency of the importance sampling method. To see this, we shall compare the variance with that of using perfect Monte Carlo, when N' samples from the target density $\pi(x)$ are available. Using (2.4) and taking the ratio of the variances with and without importance sampling gives

$$\frac{var_q\left[\widehat{\pi}(\varphi)\right]}{var\left[\widehat{\pi}(\varphi)\right]} \approx \frac{N'}{N}\left(1 + var_q\left[w\left(x\right)\right]\right).$$

In [102], the value of N' at which the two variances become equal is defined as *effective sample size*, and is given by

$$N_{eff} = \frac{N}{1 + var_q\left[w\left(x\right)\right]}. \tag{2.9}$$

2.5 Markov Chains

Markov chains theory is very important for all the topics included in this thesis, so we shall continue with a brief introduction on discrete time Markov chains defined on general state spaces.

The concepts and ideas in this introduction will also prove useful later in this thesis, for example when we consider *Hidden Markov models* for general state spaces. As indicated by the title of this thesis, we are primarily interested in stochastic processes taking values on an arbitrary set equipped with a σ-field. Thus, the framework is kept as general as possible by using definitions on arbitrary measurable spaces in contrast to using discrete state spaces, which has been very popular especially in the literature of Hidden Markov Models (for example see [138]). In this section we shall list some of the main definitions and properties of discrete time Markov processes defined on general state spaces. This section is not intended to be a detailed review and for a more concise treatment we refer the interested reader to [114, 145, 162].

A *Markov chain* is a sequence of random variables $(X_n)_{n\in\mathbb{N}}$, which takes its values on a sequence of measurable spaces $(\mathcal{X}_n, \mathcal{E}_n)_{n\in\mathbb{N}}$ and has an initial distribution η_0. Each X_n is referred as the state at time n and obeys elementary transitions given by a sequence of Markov kernels $(M_n)_{n\in\mathbb{N}}$ with each $M_n : \mathcal{X}_{n-1} \to \mathcal{P}(\mathcal{X}_n)$, where $\mathcal{P}(\mathcal{X}_n)$ denotes the set of probability measures on space $\mathcal{X}_n$. For each transition kernel M_n we require that

- $M_n(\cdot, A)$ is a nonnegative measurable function on $\mathcal{X}_{n-1}$, for each set $A \in \mathcal{E}_n$,
- $M_n(x, \cdot)$ is a probability measure on $\mathcal{E}_n$, for each $x \in \mathcal{X}_{n-1}$.

If that holds, then for any $n \in \mathbb{N}$, any initial distribution η_0 and any sequence of transition kernels, $(M_n)_{n\in\mathbb{N}}$, there exists a stochastic process $(X_n)_{n\geq 0}$ on $\Omega_n = \mathcal{X}_0 \times ... \times \mathcal{X}_n = \mathcal{X}_{0:n}$, measurable with respect to the product σ-field $\mathcal{F}_n = \otimes_{p=0}^{n} \mathcal{E}_p$, and following a probability law $\mathbb{P}_{\eta_0}$ on $\mathcal{F}_n$ such that

$$\mathbb{P}_{\eta_0}(X_0 \in A_0, X_1 \in A_1, ..., X_n \in A_n) = \int_{A_0}\int_{A_1}\cdots\int_{A_{n-1}} \eta_0(dy_0)M_1(y_0, dy_1)\cdots M_n(y_{n-1}, A_n),$$

where $A_i \subseteq \mathcal{X}_i$ for any $i = 0, ..., n$.

The defining property of a Markov chain is that the current state of the chain at time n depends only on the previous state at time $n-1$. Then given a set $A \subseteq \mathcal{X}_n$, we have for any n

$$\mathbb{P}_{\eta_0}(X_n \in A | X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1}) = \mathbb{P}_{\eta_0}(X_n \in A | X_{n-1} = x_{n-1}), \tag{2.10}$$

which using kernel M_n is given as

$$\mathbb{P}_{\eta_0}(X_n \in A | X_{n-1} = x_{n-1}) = \int_A M_n(x_{n-1}, dx_n).$$

For the remaining of this section we shall focus only on *time homogeneous* Markov chains, for which $\mathcal{X}_i = \mathcal{X}, \mathcal{E}_n = \mathcal{E}$, and $M_i = M$ at all times i. The kernel for n transitions, $M^n(x, A)$,can be written recursively as

$$M^n(x, A) = \int_{\mathcal{X}} M(x, dy) M^{n-1}(y, A),$$

where we use $M^1(x, A) = M(x, A)$. Alternatively, the nth step transition kernel can be given by the *Chapman-Kolmogorov* equations, which state that for every m with $0 \leq m \leq n$

$$M^n(x, A) = \int_{\mathcal{X}} M^m(x, dy) M^{n-m}(y, A). \tag{2.11}$$

Markov Properties

Equation (2.10) can be generalised to the so called weak Markov property, which is given by the following proposition.

Proposition 2.5.1 ***Weak Markov Property***. *For every initial distribution η_0, every $n + 1$ sample $(X_0, ..., X_n)$, and every bounded measurable test function $\varphi : \mathcal{X} \to \mathbb{R}$, we have*

$$\mathbb{E}_{\eta_0}[\varphi(X_{n+1}, X_{n+2}, ...)|X_0 = x_0, X_1 = x_1, ..., X_n = x_n] = \mathbb{E}_{x_n}[\varphi(X_1, X_2, ...)].$$

When φ is simply the indicator function $\mathbb{I}_A$, Definition 2.5.1 coincides with equation (2.10).

We shall now introduce certain random times in the evolution of (X_n), which are used to analyse the behaviour of the chain.

Definition 2.5.1 *For any set $A \in \mathcal{E}$, we define*

- *the occupation time or number of passages of X_n in A, ν_A, as the number of visits by $(X_n)_{n\geq 1}$ to A after time 0, given by*

$$\nu_A = \sum_{n=1}^{\infty} \mathbb{I}_A(X_n)$$

- *the stopping time or first return times on A as*

$$\tau_A = \inf\{n \geq 1 : X_n \in A\}$$

We shall be particulary interested in quantities such as the *average number of passages in A*, $\mathbb{E}_{\eta_0}[\nu_A]$, and the *probability of return to A in a finite number of steps*, $\mathbb{P}_{\eta_0}(\tau_A < \infty)$. If we include the notion of stopping time in Proposition 2.5.1 we obtain the *Strong Markov property*.

Proposition 2.5.2 ***Strong Markov Property***. *For every initial distribution η_0, every bounded measurable test function $\varphi : \mathcal{X} \to \mathbb{R}$, and every almost surely finite stopping time ζ, we have*

$$\mathbb{E}_{\eta_0}(\varphi(X_{\zeta+1}, X_{\zeta+2}, ...)|X_0 = x_0, X_1 = x_1, ..., X_\zeta = x_\zeta) = \mathbb{E}_{x_\zeta}(\varphi(X_1, X_2, ...)).$$

Irreducibility

Irreducibility is a property of certain Markov chains, which generalises the concept of all states being able to communicate found in discrete space Markov chains. Using an auxiliary measure ϕ we are interested in checking if the chain is able to visit the entire state space with a positive probability, regardless of the starting point of the chain.

Property 2.5.1 *Given a measure ϕ, a Markov chain is ϕ-irreducible, if for every $A \in \mathcal{F}$ with $\phi(A) > 0$, there exists a time n such that $M^n(x, A) > 0$ for all $x \in \mathcal{X}$ or equivalently $\mathbb{P}_x(\tau_A < \infty) > 0$. The chain is also strongly ϕ-irreducible if $n = 1$ for all measurable A.*

Aperiodicity

Aperiodicity is a property, which loosely speaking restricts the chain from moving in the state space in a periodic manner, i.e. getting trapped in cycles. In discrete state spaces a cycle is defined as the greatest common denominator of the lengths of all paths with positive probability between two given states. If there exists no cycle greater than one for any pair of states then the chain is *aperiodic*. In order to extend this concept to general state spaces, we introduce the notion of *small sets*, for which the *minorisation condition* holds. The minorisation condition is that there exists a set $C \in \mathcal{E}$, $\epsilon > 0$ and a probability measure ν such that $M(x, A) \geq \epsilon\nu(A)$, for every $x \in C, A \in \mathcal{E}$. Then, ν appears a constant component of the transition kernel on C and we can proceed to the following definition:

Definition 2.5.2 *A set C is small if there exist $m \in \mathbb{N}^*$ and a nonzero measure ν_m such that*

$$M^m(x, A) \geq \nu_m(A), \quad \forall x \in C, \forall A \in \mathcal{E}.$$

If there exist a small set C, an associated integer m' and a probability measure $\nu_{m'}$, the cycle of a ϕ-irreducible chain is defined as the greatest common denominator of

$$\{m \geq 1; \exists \delta_m > 0 : \quad C \text{ is small for } \nu_m \geq \delta_m \nu_{m'}\}$$

If the longest cycle is of length one, then the chain is aperiodic.

Recurrence

In order to examine how often every set A will be visited by the Markov chain, we introduce the notion of *recurrence*. Any set A is defined as recurrent if $\mathbb{E}_{\eta_0}[\nu_A] = +\infty$. Otherwise the set is called *uniformly transient*. We can extend the notion of recurrence, as a property of ϕ-irreducible chains.

Definition 2.5.3 *A ϕ-irreducible Markov chain $(X_n)_{n\geq 0}$ is recurrent, if for every $A \in \mathcal{E}$ such that $\phi(A) > 0$ then $\mathbb{E}_x[\nu_A] = +\infty$ for every $x \in A$.*

If this does not hold for a ϕ-irreducible Markov chain, then the chain is transient. A popular criterion for establishing recurrence is to show that there exists a small set C with $\phi(C) > 0$ such that $\mathbb{P}_x(\tau_A < \infty) = 1$ for every $x \in C$.

We can further strengthen recurrence by requiring an infinite number of visits for every path of the Markov chain. This introduces the property of *Harris recurrence*.

Definition 2.5.4 *A set A is Harris recurrent if $\mathbb{P}_x(\nu_A = \infty) = 1$ for all $x \in A$. A ϕ-irreducible Markov chain is Harris recurrent if every set with $\phi(A) > 0$ is Harris recurrent.*

One can show Harris recurrence for a ϕ-irreducible chain, if there exists a small set C with $\phi(C) > 0$ such that $\mathbb{P}_x(\tau_A < \infty) = 1$ for every $x \in \mathcal{X}$. This form of recurrence was shown to be sufficient to guarantee the existence of a unique invariant distribution for the chain [114].

Invariant Measures

A stronger form of stability for the chain $(X_n)_{n\geq 0}$ is attained if the marginal distribution of X_n is independent of n. More formally, this requires the existence of a probability measure π such that if $X_n \sim \pi$ then $X_{n+1} \sim \pi$. MCMC methods are based on this requirement which defines a particular kind of recurrence called positive recurrence.

Definition 2.5.5 *A σ-finite measure π is invariant for the transition kernel $M(\cdot, \cdot)$ and the associated Markov chain if*

$$\pi(A) = \int_{\mathcal{X}} \pi(dy) M(y, A), \quad \forall A \in \mathcal{E}.$$

If the invariant measure π is a probability measure then the invariant distribution is referred as *stationary*. A ϕ-irreducible chain is called *positive*, if it admits an invariant probability measure. If the chain does not allow for a σ-finite invariant measure it is called null. If a chain is positive

then it should be also recurrent. In addition, if a chain is Harris recurrent and positive, it is called Harris positive.

Reversibility

The property of reversibility is inherent to stationary Markov chains. A Markov chain $(X_n)_{n\geq 0}$ is *reversible* if the distribution of $X_{n+1}|X_{n+2} = x$ is the same as the distribution of $X_{n+1}|X_n = x$. Moreover, we define the *detailed balance condition* as the existence of function f satisfying

$$f(x)M(x,y) = f(y)M(y,x).$$

Proposition 2.5.3 *If the detailed balance condition is satisfied for kernel M with the associated function being a probability density function $\pi^*(dx)$, then the chain is reversible and π^* is the stationary invariant density of the chain.*

This sufficient condition proves extremely useful for designing MCMC algorithms, where we are interested in simulating an invariant stationary Markov chain. The detailed balance condition provides a useful design rule for the proposal kernel of the MCMC chain, which is also easy to check.

Ergodicity

For an invariant chain, we would like to be able to show that the chain will gradually "forget" the initial state and eventually converge to a unique stationary distribution π that is independent of both, X_0 and n. Ergodic theorems provide results, which give conditions under which a strong law of large numbers holds and the probability density of the nth iterate of the Markov chain converges to its unique, invariant density.

Theorem 2.5.1 *Suppose $(X_n)_{n\geq 0}$ is a Harris recurrent Markov chain with transition kernel $M(\cdot,\cdot)$ and invariant distribution π, then for all π-integrable functions φ,*

$$\frac{1}{n}\sum_{i=1}^{n}\varphi(X_i) \xrightarrow[n\to\infty]{a.s.} \pi(\varphi).$$

Theorem 2.5.2 *Suppose $(X_n)_{n\geq 0}$ is a aperiodic, Harris recurrent Markov chain with transition kernel $M(\cdot,\cdot)$ and invariant distribution π. Then for π-almost every $x \in \mathcal{X}$, and all sets $A \in \mathcal{E}$*

$$\| M^n(x,A) - \pi(A) \|_{TV} \xrightarrow[n\to\infty]{a.s.} 0$$

where $\|\cdot\|_{TV}$ *denotes the total variation norm*[2]*.*

A further strengthening of the conditions is required to obtain a central limit theorem for sample-path averages. In this case it is required that $(X_n)_{n\geq 0}$ is an *ergodic chain*.

Definition 2.5.6 *An ergodic Markov chain is defined as a* ϕ*-irreducible chain, which is also aperiodic and positive Harris recurrent.*

In addition, one needs the notion of geometric ergodicity. An ergodic Markov chain with invariant distribution π is *geometrically ergodic* if there exists a non-negative function $\zeta : \mathcal{X} \to \mathbb{R}_+$ and a positive constant $r > 1$ such that

$$\| M^n(x, A) - \pi(A) \|_{TV} \leq \zeta(x) r^{-n} \quad \forall x \in \mathcal{X}, \forall n \geq 0, \forall A \in \mathcal{E}$$

If the Markov chain is geometrically ergodic with invariant distribution π, then for all measurable functions φ, and any initial distribution η_0, the distribution we obtain obeys the following Central Limit Theorem (CLT):

$$\sqrt{n}(\frac{1}{n}\sum_{i=1}^{n}\varphi(X_i) - \pi(\varphi)) \underset{N\to\infty}{\Longrightarrow} \mathcal{N}(0, \Sigma_\varphi),$$

where $\Sigma_\varphi = \ var[\varphi(X_0)] + 2\sum_{i=1}^{\infty} \ cov\left[\{\varphi(X_0), \varphi(X_i)\}\right]$.

One can further strengthen the notion of geometric ergodicity, so that the rate of geometric convergence must be uniform over the whole space $\mathcal{X}$, i.e. ζ is a constant function. This property is referred as *uniform ergodicity* and the CLT above holds for all L^2 measurable functions . It can be shown that for a uniformly ergodic chain, aperiodicity holds and $\mathcal{X}$ is a small set.

2.6 Markov Chain Monte Carlo methods

In many situations it is difficult to obtain large number of iid samples from the distribution of interest π. Markov Chain Monte Carlo (MCMC) methods [63, 73, 115, 140] are based on generating samples from a suitable ergodic Markov chain, which has π as its stationary distribution.

2.6.1 Metropolis-Hastings and the Gibbs sampler

MCMC methods run an ergodic Markov chain long enough, so that it converges to the stationary distribution π, which is chosen to be precisely the target distribution of interest. The most

[2]The total variation norm between measure μ_1 and μ_2 is defined as $\| \mu_1 - \mu_2 \|_{TV} = \sup_A | \mu_1(A) - \mu_2(A) |$

widely used algorithm up to date is the Metropolis-Hastings algorithm as presented below.

Algorithm 2.2 ***Metropolis-Hastings (MH) algorithm:***

1. *Sampling Step*

• *Sample* $\tilde{X}_n \sim K(X_{n-1}, \cdot)$

•*Calculate the acceptance probability* $\alpha = \min(1, \frac{\pi(\tilde{X}_n)K(\tilde{X}_n, X_{n-1})}{\pi(X_{n-1})K(X_{n-1}, \tilde{X}_n)})$

•*Sample* $U^{(i)} \sim \mathcal{U}(0, 1)$

2. *Accept-Reject Step*

• *If* $U \leq \alpha$ *accept* $\tilde{X}_n$ *as a sample of* π*, and set* $X_{n+1} = \tilde{X}_n$

• *Else, reject* $\tilde{X}_n$*, and set* $X_{n+1} = X_n$ *and go back to step 1*

Historically, the first MCMC method was proposed by Metropolis et al. in 1953, [115] and was later generalised by Hastings, [73]. In Algorithm 2.2 the transition kernel M of the underlying Markov chain (X_n) can be written as

$$M(x, dy) = \alpha(x, y)K(x, dy) + (1 - \alpha(x, y))\delta_x(dy)$$

A simple substitution can show that M satisfies the detailed balance equation for π, and therefore π is the stationary distribution of the chain.

A popular special case of the general Metropolis-Hastings algorithm is the Gibbs sampler. Let the variable of interest X be distributed according to π and have a cardinality of size d. For simplicity, we denote x_{-i} the vector $x_{1:i-1,i+1:d}$ in case $i \neq 1, d$, and $x_{-1} = x_{2:d}$, $x_{-d} = x_{1:d-1}$ otherwise. If the full conditionals $\pi_i(x_i|X_{-i} = x_{-i})$ are available to sample from, one can obtain a sample from π by iteratively sampling from the full conditionals as shown below.

Algorithm 2.3 ***Gibbs sampler algorithm:*** *Given* $x_n = (x_{1_n}, x_{2_n}, ..., x_{d_n})$*, generate for* $i = 1, ..., d$

$$X_{i_n} \sim \pi_i(x_i|X_{-i_n} = x_{-i_n})$$

Repeat

Although this algorithm is computationally very attractive, easy to implement and has thus

been used widely, the requirement of the conditionals of the target distributions is quite restrictive.

MCMC is itself an interesting and complicated topic. We shall not attempt to present details or a further discussion, as it is beyond the scope of this chapter. Further details can be found in [63] and [140]. Despite their versatility and wide success, it might be impractical to apply classical MCMC algorithms to sequential inference problems, which require simulating time varying distributions. The next section discusses a different type of Monte Carlo techniques, known as Sequential Monte Carlo methods that can provide with approximation tools for target probability distributions, which vary sequentially in time.

2.7 Sequential Monte Carlo Methods

Motivated by many real-world applications, it is often essential that inference is performed sequentially. There are many examples where this is particularly useful. For example, when handling high dimensional spaces it might be hard to design good proposal distributions for standard Importance Sampling. A strategy to circumvent this would be to build up the instrumental density sequentially as a product of conditional densities of each dimension given the ones previously examined. Also, in problems involving time series, the observations of a latent variable arrive sequentially and one would like to compute the Bayesian posterior recursively as the observations become available. This relates to the problem of optimal Bayesian filtering, which will be discussed in more detail in the next section of this chapter. Sequential Monte Carlo (SMC) methods are a set of simulation based methods, which provide a convenient approach to compute posterior distributions. They are very flexible, easy to implement, parallelisable and applicable in very general settings. Hence, they have been applied successfully to a wide range of sequential inference applications, such as target tracking [71], navigation [19], econometrics [132] and telecommunications [7].

With reference to the notation of Section 2.4, suppose the variable of interest $X \sim \pi$ is partitioned as a $d+1$-dimensional vector[3]. Each element in that vector can be another subvector of arbitrary size. We index each element of the partition by index n with $n = \{0, ..., d\}$.

[3]We alert the reader that the first element of the $d+1$-dimensional vector is indexed by 0. This was done so that the indexing in the notation remains consistent with the rest of the chapter.

Let then X_n take values at some measurable space $(\mathcal{X}_n, \mathcal{E}_n)$[4]. At any n, we denote the joint variable of all the partitions until n as $X_{0:n}$, which takes values on the space of the complete path until time n, $\mathcal{X}_{0:n} = \mathcal{X}_0 \times ... \times \mathcal{X}_n$, and $dx_{0:n}$ is the infinitesimal neighbourhood of the path point $X_{0:n}$. Then one could decompose the instrumental density as

$$q(x_{0:d}) = q_0(x_0)q_1(x_1|x_0)q_2(x_2|x_{0:1}) \cdots q_d(x_d|x_{0:d-1}).$$

Then one can write a similar decomposition for the target density as

$$\pi(x_{0:d}) = \pi(x_0)\pi(x_1|x_0)\pi(x_2|x_{0:1}) \cdots \pi(x_d|x_{0:d-1}).$$

and then obtain a recursive expression for the IS weight as

$$w(x_{0:n}) = w(x_{0:n-1})\frac{\pi(x_n|x_{0:n-1})}{q_n(x_n|x_{0:n-1})}.$$

In practice, for the most general setting it might be hard to obtain analytical expressions for each $\pi(x_n|x_{0:n-1})$ as this would require being able to compute the marginalisation

$$\pi(x_{0:n}) = \int \pi(x_{0:d})dx_{n+1:d}.$$

For the purpose of maintaining the presentation general we shall assume that there is available a sequence of distributions $\{\pi_n\}_{n\geq 0}$, where each π_n can reasonably approximate $\pi(x_n|x_{0:n-1})$ and $\pi_d = \pi$. Note that each π_n need to be known only up to a normalisation constant and they only serve as intermediate auxiliary guides to obtain a final sample of X. In this section we shall consider how to sample from such a sequence of distributions $\{\pi_n\}_{n\geq 0}$, such that each π_n is defined on some measurable space $(\Omega_n, \mathcal{F}_n)$, where $\Omega_n = \mathcal{X}_{0:n}$ and is equipped with the product σ-field $\mathcal{F}_n = \otimes_{p=0}^{n}\mathcal{E}_p$. We are interested in approximating the following integral with respect to a target density $\pi_n(x_{0:n})$,

$$\pi_n(\varphi_n) = \mathbb{E}_{\pi_n}[\varphi_n(x_{0:n})] = \int_{\mathcal{X}_{0:n}} \varphi_n(x_{0:n})\pi_n(dx_{0:n}), \tag{2.12}$$

where φ_n is some measureable function, such that $\varphi_n : \mathcal{X}_{0:n} \longrightarrow \mathbb{R}^{n_\varphi}$.

2.7.1 Sequential Importance Sampling (SIS)

As already described, Importance Sampling simulates samples from an easy-to-sample importance distribution and corrects for the bias by introducing an importance weight. One can

[4]Since the dimension of X_n can be arbitrary we shall be using a varying measurable space $(\mathcal{X}_n, \mathcal{E}_n)$.

therefore employ this approach on (2.12) using a sequence of importance densities $\{q_n\}_{n\geq 0}$ defined on $(\Omega_n, \mathcal{F}_n)$. As stated earlier when introducing Importance sampling, each q_n should be chosen such that if $\pi_n > 0$ then $q_n > 0$, so that the importance ratio

$$w_n(x_{0:n}) = \frac{\pi_n(x_{0:n})}{q_n(x_{0:n})},$$

is defined. This leads to the expression

$$\begin{aligned}\pi_n(\varphi_n) &= \int_{\mathcal{X}_{0:n}} \varphi_n(x_{0:n}) w_n(x_{0:n}) q_n(dx_{0:n}) \\ &= \mathbb{E}_{q_n}\left[\varphi_n(x_{0:n}) w_n(x_{0:n})\right] \\ &= \frac{\mathbb{E}_{q_n}\left[\varphi_n(x_{0:n}) w_n(x_{0:n})\right]}{\mathbb{E}_{q_n}\left[w_n(x_{0:n})\right]},\end{aligned} \tag{2.13}$$

Note that we have adopted an importance sampling approach similar to the one in (2.7) for the more general case when the normalisation constants are not available.

As in the simple Importance Sampling case, the efficiency of the scheme here will depend on the choice of the importance density. A good candidate for q_n would be one that is close in shape to $|\varphi_n(x_{0:n})|\,\pi_n(x_{0:n})$. In such a case, the variance of the weights will be roughly constant and importance sampling will generate fairly precise estimates. However, if the importance weights vary substantially, such a method should not be used. This would be for example the case if $|\varphi_n(x_{0:n})|\,\pi_n(x_{0:n})$ has thicker tails than $q_n(x_{0:n})$ [61]. A naive implementation of q_n would be to design it for the complete path from time 0 to n, but one can see this would be extremely inefficient due to the increasing dimensional spaces. Instead, we can keep the the realisation of the path up to time $n-1$, $X_{0:n-1}$, and produce samples at time n conditional on the existing path. This sequential setting can be obtained if the importance distribution is chosen to have the form

$$q_n\left(x_{0:n}\right) = q_0\left(x_0\right) \prod_{k=1}^{n} q_k\left(x_k | x_{0:k-1}\right). \tag{2.14}$$

Then, we can use

$$\frac{\pi_n(x_{0:n})}{q_n(x_{0:n})} = \frac{\pi_{n-1}(x_{0:n-1})}{q_{n-1}(x_{0:n-1})} \frac{\pi_n(x_{0:n})}{\pi_{n-1}(x_{0:n-1}) q_n\left(x_n | x_{0:n-1}\right)}$$

to obtain a sequential importance weight update as

$$w_n(x_{0:n}) = w_{n-1}(x_{0:n-1}) \frac{\pi_n(x_{0:n})}{\pi_{n-1}(x_{0:n-1}) q_n\left(x_n | x_{0:n-1}\right)}. \tag{2.15}$$

The SMC approximation is based on a weighted empirical distribution of a set of $N \gg 1$ samples, termed as *particles*. These provide a flexible way to approximate posterior distributions using point masses. Assume that at time $n-1$, an empirical approximation to π_{n-1}, $\widehat{\pi}_{n-1}$, is available, consisting of a set of particles $X_{0:n-1}^{(1:N)} \triangleq \left[X_{0:n-1}^{(1)}, \ldots, X_{0:n-1}^{(N)}\right]$ with corresponding weights $\widetilde{w}_{n-1}^{(1:N)} \triangleq \left[\widetilde{w}_{n-1}^{(1)}, \ldots, \widetilde{w}_{n-1}^{(N)}\right]$. We can write $\widehat{\pi}_{n-1}$ as

$$\widehat{\pi}_{n-1}(dx_{0:n-1}) = \sum_{i=1}^{N} \widetilde{w}_{n-1}^{(i)} \delta_{X_{0:n-1}^{(i)}} dx_{0:n-1}$$

If a set of N independent samples from q_n are available, we can augment the path of the state as $X_{0:n}^{(1:N)} = [X_{0:n-1}^{(1:N)}, X_n^{(1:N)}]$. A Monte Carlo estimate of (2.13), $\widehat{\pi}_n$, will be given by

$$\begin{aligned}\widehat{\pi}_n(\varphi_n) &= \frac{\frac{1}{N}\sum_{i=1}^{N} \varphi_n(X_{0:n}^{(i)})\, w_n(X_{0:n}^{(i)})}{\frac{1}{N}\sum_{j=1}^{N} w_n(X_{0:n}^{(j)})} \\ &= \sum_{i=1}^{N} \varphi_n(X_{0:n}^{(i)})\, \widetilde{w}_n^{(i)}, \end{aligned} \tag{2.16}$$

where the normalised importance weights are given by

$$\widetilde{w}_n^{(i)} = \frac{w_n(X_{0:n}^{(i)})}{\sum_{j=1}^{N} w_n\left(X_{0:n}^{(j)}\right)}$$

In the limit,

$$\widehat{\pi}_n(\varphi_n) \xrightarrow[N\to\infty]{a.s.} \pi_n(\varphi_n)$$

will hold. Furthermore, it is possible to obtain a central limit theorem for $\widehat{\pi_n}(\varphi_n)$ [46, 62]. We can summarize SIS as the following algorithm.

Algorithm 2.4 ***Sequential Importance Sampling algorithm:*** *At each $n \geq 0$ we have available particle approximation $\{X_{0:n-1}^{(i)}, \widetilde{w}_{n-1}^{(i)}\}_{i=1}^{N}$. The recursion is proceeded in a two step procedure as follows:*

1. *Sampling Step*
   - *For $i = 1, \ldots, N$, sample particles as $X_n^{(i)} \sim q_n\left(\cdot | X_{0:n-1}^{(i)}\right)$,*
   - *Augment the path of the state as $X_{0:n}^{(1:N)} = [X_{0:n-1}^{(1:N)}, X_n^{(1:N)}]$.*
2. *Weight Calculation*
   - *Compute $w_n^{(i)} = w_{n-1}^{(i)} \frac{\pi_n(X_{0:n}^{(i)})}{\pi_{n-1}(X_{0:n-1}^{(i)}) q_n\left(X_n^{(i)} | X_{0:n-1}^{(i)}\right)}$,*
   - *Normalise weights $\widetilde{w}_n^{(i)} = \frac{w_n^{(i)}}{\sum_{j=1}^{N} w_n^{(j)}}$.*

2.7.2 Sequential Importance Sampling Resampling (SISR)

SIS is an attractive method, but one has to bear in mind that it is still a constrained version of IS. If n grows very large then it might fail to adequately represent the high dimensional posteriors. As n increases, the variance of the importance weights will increase with time, see [48]. Then the distribution of the importance weights becomes more and more skewed. As a result, after a few iterations all but one of the normalised importance weights will be very close to zero. To make this clearer we shall use a simple argument borrowed from [140]. For any particle (i) the weight is given by an expression of the form $w_n^{(i)} \propto w_{n-1}^{(i)} \varrho_n^{(i)}$. Consider the case of using standard IS with q as the proposal and π as the target distribution, in order to obtain n successive independent samples π. We would like to use successive independent IS draws to sample from the joint distribution of $X_{0:n}$. Then for the weight we have

$$w_n^{(i)} \propto \exp(\sum_{k=1}^{n} \log(\frac{\pi(X_k^{(i)})}{q(X_k^{(i)})})),$$

and using the Law of Large Numbers as $n \to \infty$ for the sum within the exponential we get

$$w_n^{(i)} \propto \exp(-n\mathbb{E}_q[\log \frac{q(X)}{\pi(X)}]),$$

where it is easy to show that $\mathbb{E}_q[\log \frac{q(X)}{\pi(X)}]$ is positive. It is clear that the weights have tendency to degenerate towards zero. Of course, when using SIS the normalisation step will eventually ensure that one particle will be set to one. Also, in SIS the independence structure does not hold but it is still clear that weights tend to decay, when targeting a distribution of increasing dimension. This problem is commonly referred in the literature as the *degeneracy* of the SIS algorithm.

To counter the degeneracy problem, a resampling step is introduced. This effectively selects (multiplies/discards) the previous particle paths $X_{0:n-1}^{(i)}$ according to their (large/small) weights $\widetilde{w}_n^{(i)}$. This is achieved by generating a number of copies for each path by sampling the N available paths according to $\Pr\left(\widetilde{X}_{0:n}^{(i)} = X_{0:n}^{(k)}\right) = \widetilde{w}_n^{(k)}$. This gives the new set of paths

$$\widetilde{X}_{0:n-1}^{(1:N)} = [\underbrace{X_{0:n-1}^{(1)}, \ldots, X_{0:n-1}^{(1)}}_{I_{n-1}^{(1)} \text{ times}}, \ldots, \underbrace{X_{0:n-1}^{(N)}, \ldots, X_{0:n-1}^{(N)}}_{I_{n-1}^{(N)} \text{ times}}],$$

where the number of copies of the i^{th} path is denoted by $I_{n-1}^{(i)}$ and is such that $\sum_{i=1}^{N} I_{n-1}^{(i)} = N$. A number of resampling methods have been proposed in the literature that satisfy $\mathbb{E}\left(I_{n-1}^{(i)}\right) = N$

$\widetilde{w}_n^{(i)}$, but have different $var\left(I_{n-1}^{(i)}\right)$. Standard resampling schemes include multinomial resampling [71], residual resampling [103] and stratified resampling [85]. We refer the reader to [44] for a comparison. The latter scheme has the least variance and it is the one adopted in this thesis. In general, the resampling procedure introduces undesirable Monte Carlo variance into the algorithm.

Unfortunately, resampling also leads to impoverishment of the particles, since at every time step some of the distinct particles are dropped in favour of more copies of highly-weighted particles. One remedy for this is to decrease the resampling frequency and use resampling only when it is necessary. A simple measure of degeneracy is the *effective sample size*, seen in equation (2.9), which can be written for SIS as

$$N_{eff} = \frac{N}{1 + var_{q_n}[w(x_{0:n})]} \tag{2.17}$$

and estimated in practice using

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}\left[\widetilde{w}_n^{(i)}\right]^2}.$$

When $\widehat{N}_{eff}$ is below some threshold value, say $\frac{N}{2}$, a resampling procedure is applied to the particles [48]. We can summarize SISR as the following algorithm.

Algorithm 2.5 ***Sequential Importance Sampling Resampling algorithm:*** *At each $n \geq 0$ we have available particle approximation $\{\widetilde{X}_{0:n-1}^{(i)}, \widetilde{w}_{n-1}^{(i)}\}_{i=1}^{N}$. The recursion is proceeded in a two step procedure as follows:*

1. *Sampling Step*
   - *For $i = 1, \ldots, N$, sample particles as $X_n^{(i)} \sim q_n\left(\cdot|\widetilde{X}_{0:n-1}^{(i)}\right)$,*
   - *Augment the path of the state as $X_{0:n}^{(1:N)} = [X_{0:n-1}^{(1:N)}, X_n^{(1:N)}]$.*
2. *Weight Calculation*
   - *Compute $w_n^{(i)} = w_{n-1}^{(i)} \frac{\pi_n(X_{0:n}^{(i)})}{\pi_{n-1}(X_{0:n-1}^{(i)}) q\left(X_n^{(i)}|X_{0:n-1}^{(i)}\right)}$,*
   - *Normalise weights $\widetilde{w}_n^{(i)} = \frac{w_n^{(i)}}{\sum_{j=1}^{N} w_n^{(j)}}$.*
3. *Resampling*
   - *If $\widehat{N}_{eff} < \frac{N}{2}$, resample $\widetilde{X}_{0:n}$ according to $\Pr\left(\widetilde{X}_{0:n}^{(i)} = X_{0:n}^{(k)}\right) = \widetilde{w}_n^{(k)}$, and assign new resampled particle indexed with i, the new importance weight $\widetilde{w}_n^{(i)} = \frac{1}{N}$.*

As in the non-sequential importance sampling case, the choice of the importance densities $\{q_n\}_{n\geq 0}$ is critical for controlling the variance. Equation (2.17) would still hold if not for the re-sampling stage, which introduces correlations between the particles leading to a more complex expression. However, it still remains sensible to try to minimise the the variance of the unnormalised weights appearing in SISR algorithm. In [46], we can find the following proposition.

Proposition 2.7.1 *The optimal proposal distribution that minimises the variance of the incremental importance weight is given by*

$$q_n^{opt}(x_n|x_{0:n-1}) = \pi_n(x_n|x_{0:n-1}).$$

Using this optimal importance distribution, the incremental importance weight is given by

$$w_n^{opt}(x_{0:n}) = \frac{\pi_n(x_{0:n-1})}{\pi_{n-1}(x_{0:n-1})}.$$

Proof. See [46]. ■

In practice it might be hard to directly sample from $\pi_n(x_n|x_{0:n-1})$, or compute

$$\pi_n(x_{0:n-1}) = \int_{\mathcal{X}_n} \pi_n(x_{0:n-1})dx_n.$$

Therefore, approximations should be considered, such as local linearisation techniques. Other possibilities include rejection sampling approaches [46, 103] and MCMC methods [24, 103], however these usually involve a significant computational overhead.

2.7.3 Auxiliary SMC filter

A well known drawback of SISR is the fact that if π_n varies significantly compared to π_{n-1}, the variance of the weights can be quite high and the algorithm ineffective, as a large number of particles might be required. One way to deal with these problems is to focus on carefully designing good proposal distributions which bridge the difference between π_n and π_{n-1}. In Proposition 2.7.1 we see that the optimal weights with respect the variance are equal to the ratio of $\frac{\pi_n(x_{0:n-1})}{\pi_{n-1}(x_{0:n-1})}$. Assume we can construct $\widetilde{\pi}_n(x_{0:n-1})$, which is some reasonable approximation of $\pi_n(x_{0:n-1})$. We could then use the ratio $\frac{\widetilde{\pi}_n(x_{0:n-1})}{\pi_{n-1}(x_{0:n-1})}$ at time $n-1$ to modify the weights prior to the resampling and sampling a new particle steps, so that the empirical mass is strengthened at more promising regions of the state space regarding the next time step. In a IS context, we

could use $\widehat{\pi}_{n-1}$ to construct an importance distribution $q_n(x_{0:n})$ for the complete path, which is proportional to

$$q_n(x_n|x_{0:n-1})\frac{\widetilde{\pi}_n(x_{0:n-1})}{\pi_n(x_{0:n-1})}\widehat{\pi}_{n-1}(x_{0:n-1}) = \sum_{i=1}^{N} q_n(x_n|X_{0:n-1}^{(i)})\frac{\widetilde{\pi}_n(X_{0:n-1}^{(i)})}{\pi_n(X_{0:n-1}^{(i)})}w_n^{(i)}\delta_{X_{0:n-1}^{(i)}}(x_{0:n-1})$$

Instead of using this expression directly, we will use an auxiliary variable i to track the index of the i-th particle $X_{0:n-1}^{(i)}$ and consider using a procedure similar to SIS based on equation (2.14). The difference will be that this time at each proposal step, we will propose i and x_n jointly from

$$q_n(i, x_n|x_{0:n-1}) = q_n(x_n|i, x_{0:n-1})q_n(i|x_{0:n-1})$$

where

$$q_n(i|X_{0:n-1}^{(i)}) = w_{n-1}^{(i)}\frac{\widetilde{\pi}_n(X_{0:n-1}^{(i)})}{\pi_n(X_{0:n-1}^{(i)})}.$$

Sampling the particle-index pair from the joint distribution can be split into a two steps. We can sample first the auxiliary variable using any standard resampling scheme. Let $\kappa(i)$ be the index obtained from the resampling mechanism. Then, we proceed by extending the path of $X_{0:n-1}$, by sampling $X_n^{(i)}$ from $q_n(\cdot|X_{0:n-1}^{(\kappa(i))})$. The elegance of the approach is also improved from the fact that the resampling step is embedded in the sampling procedure. Finally, as in SIS we weight the new samples using a direct substitution to $\frac{\pi_n(x_n|x_{0:n-1})}{q_n(i,x_n|x_{0:n-1})}$. We can summarize this approach as the following algorithm.

Algorithm 2.6 ***Auxiliary SMC filter:*** *At each $n \geq 0$ we have available particle approximation $\{X_{0:n-1}^{(i)}, w_{n-1}^{(i)}\}_{i=1}^{N}$. The recursion is proceeded in a three step procedure as follows:*

1. *Calculate auxiliary weight:For $i = 1, \ldots, N$*

• *Compute $\widehat{w}_n^{(i)} = w_{n-1}^{(i)}\frac{\widetilde{\pi}_n(X_{0:n-1}^{(i)})}{\pi_{n-1}(X_{0:n-1}^{(i)})}$.*

2. *Sample new index/particles: For $i = 1, \ldots, N$,*

• *If $\widehat{N}_{eff}(\widehat{w}_n^{(1:N)}) < \frac{N}{2}$, sample index i, according to $\Pr(i = \kappa(i)) = \widehat{w}_n^{(\kappa(i))}$.*

• *Sample $X_n^{(i)} \sim q_n\left(\cdot|X_{0:n-1}^{(\kappa(i))}\right)$ and augment the path of the state as $X_{0:n}^{(i)} = [X_{0:n-1}^{(\kappa(i))}, X_n^{(i)}]$.*

3. *Weight Calculation*

•*For $i = 1, \ldots, N$, compute $\widetilde{w}_n^{(i)} = \frac{\pi_n(X_{0:n}^{(i)})}{\pi_{n-1}(X_{0:n-1}^{(i)})q_n\left(X_n^{(i)}|X_{0:n-1}^{(\kappa(i))}\right)\widehat{w}_n^{(\kappa(i))}}$,*

• *Normalise weights $w_n^{(i)} = \frac{\widetilde{w}_n^{(i)}}{\sum_{j=1}^{N}\widetilde{w}_n^{(j)}}$.*

This approach was first introduced with the name *Auxiliary particle filter* in [132] for the context of optimal Bayesian filtering and remains a very popular and efficient SMC method. In [132] the authors also show that the auxiliary SMC approach improved the performance of SISR, in cases where π_n contains an outlier, which causes the weights to be very unevenly distributed in the SISR case. Also, they present how approximating the tails of a distribution can be improved compared to SISR, which tends to concentrate the the mass of the empirical approximation in regions of higher density.

2.8 Feynman-Kac Models

In this section we attempt a small scale literature review for the Feynman-Kac representations for a discrete time Markov chain $\{X_n\}_{n\geq 0}$ defined on some sequence of of some measurable spaces. We aim to give some introductory background on these representations and their properties. The main reference is [34], where a more detailed and rigorous discussion with plenty of examples on the topic can be found, and some of the material is taken from [35–37].

We conclude this part of this section with some definitions and notation. For any measure μ on E, let $\mu(f)$ or $\langle \mu, f\rangle$ denote $\int_E \mu(dy)f(y)$. For measurable sets $A \subseteq E$, let $\mu K(A) = \int_E \mu(dy)K(y, A)$, where K is an appropriate transition kernel.

2.8.1 Description of the Models

Let $\{X_n\}_{n\geq 0}$ be an inhomogeneous Markov process taking values in some sequence of measurable spaces $(E_n, \mathcal{E}_n)_{n\geq 0}$ with initial distribution ν and a family of transition kernels $\{M_n\}_{n\geq 0}$ such that

$$\mathbb{P}(X_n \in dx_n | X_{0:n-1} = x_{0:n-1}) = M_n(x_{n-1}, dx_n).$$

At any time n we denote the complete path of the chain until time n as $X_{0:n} \in E_0 \times ... \times E_n$ and $dx_{0:n}$ as the infinitesimal neighbourhood of the path point $x_{0:n}$. We can write the distribution of the canonical path on $\Omega_n = \prod_{p=0}^{n} E_p$ equipped with the product σ-field $\mathcal{F}_n = \otimes_{p=0}^{n}\mathcal{E}_p$ as

$$\begin{aligned}\mathbb{P}_\nu(X_{0:n} \in dx_{0:n}) &= M_n(x_{n-1}, dx_n)\mathbb{P}_\nu(X_{0:n-1} \in dx_{0:n-1}) \\ &= \nu(dx_0)M_1(x_0, dx_1)\cdots M_n(x_{n-1}, dx_n).\end{aligned}$$

The subscript on the probability measure $\mathbb{P}_\nu$ is there to emphasise the dependance on the distribution ν of the initial state X_0.

Moreover, let $\{G_n\}_{n\geq 0}$ be a family of non negative, bounded measurable functions, which will be referred as *potential functions*, such that

$$\mathbb{E}_\nu[\prod_{p=0}^{n} G_p(X_p)] > 0, \forall n \geq 0. \tag{2.18}$$

where $\mathbb{E}_\nu$ denotes the expectation operator with respect to the probability measure $\mathbb{P}_\nu$ of the path $X_{0:n}$. We shall proceed with a few definitions to introduce the Feynman-Kac model.

Definition 2.8.1 *[34, Def. 2.3.1] The Feynman-Kac prediction and updated path models, associated with the pair (G_n, M_n) and an initial distribution ν, are the sequence of measures on the path space defined respectively, for any $n \geq 0$, by*

$$\mathbb{Q}_\nu(X_{0:n} \in dx_{0:n}) = \frac{1}{Z_{n-1}}[\prod_{p=0}^{n-1} G_p(X_p)]\mathbb{P}_\nu(X_{0:n} \in dx_{0:n}),$$

$$\mathbb{Q}'_\nu(X_{0:n} \in dx_{0:n}) = \frac{1}{Z_n}[\prod_{p=0}^{n} G_p(X_p)]\mathbb{P}_\nu(X_{0:n} \in dx_{0:n})$$

where $(Z_n)_{n\geq 0}$ are the normalising constants (also known as partition functions) given by

$$Z_n = \mathbb{E}_\nu[\prod_{p=0}^{n} G_p(X_p)]$$

To describe the dynamic evolution of the models we introduce the definition of the flows of its time marginals.

Definition 2.8.2 *[34, Def. 2.3.2] For any bounded measurable function $f \in B_b(E_n)$, where $B_b(E_n)$ is the set of bounded continuous functions from $E_n \to \mathbb{R}$, we define the sequence of distributions $\{\eta_n\}$ and $\{\mu_n\}$ on E as the normalised prediction and updated Feynman-Kac model or flow associated with the pair (G_n, M_n) respectively. They are given by*

$$\eta_n(f) = \frac{\gamma_n(f)}{\gamma_n(1)},$$

$$\mu_n(f) = \frac{\lambda_n(f)}{\lambda_n(1)},$$

with

$$\gamma_n(f) = \mathbb{E}_\nu[f(X_n)\prod_{p=0}^{n-1} G_p(X_p)], \tag{2.19}$$

$$\lambda_n(f) = \mathbb{E}_\nu[f(X_n)\prod_{p=0}^{n} G_p(X_p)], \tag{2.20}$$

where $\{\gamma_n\}$ and $\{\lambda_n\}$ are called respectively the unnormalised prediction and updated Feynman-Kac model or flow associated with the pair (G_n, M_n).

We easily check that

$$\gamma_n(fG_n) = \lambda_n(f). \tag{2.21}$$

The unnormalised model can be related to the prediction distribution flow marginal as $\{\eta_n\}_{n\geq 0}$ as follows,

$$\gamma_n(f) = \eta_n(f) \prod_{p=0}^{n-1} \eta_p(G_p), \tag{2.22}$$

$$\lambda_n(f) = \mu_n(f) \prod_{p=0}^{n} \eta_p(G_p). \tag{2.23}$$

If $\eta_n(G_n) > 0$ and $\gamma_n(G_n) > 0$, then we observe the following property of the updated flow distribution,

$$\mu_n(f) = \frac{\gamma_n(fG_n)}{\gamma_n(G_n)} = \frac{\eta_n(fG_n)}{\eta_n(G_n)} \tag{2.24}$$

Motivated by (2.24) we proceed to define the following transformation.

Definition 2.8.3 *(Boltzmann-Gibbs transformation [34, Def. 2.3.3]) Define the subset of probability measures $\mathcal{P}_n(E_n) = \{\eta \in \mathcal{P}(E_n) : \eta(G_n) > 0\}$, where $\mathcal{P}(E_n)$ is the set of all probability measures on E_n. The Boltzmann-Gibbs transformation $\Psi_n(\cdot) : \mathcal{P}_n(E_n) \to \mathcal{P}_n(E_n)$, associated with the potential function G_n on E is defined by*

$$\Psi_n(\eta)(dx_n) = \frac{G_n(x_n)\eta(dx_n)}{\eta(G_n)}.$$

Using this definition, we can observe recursive nature of the Feynman-Kac model given by a prediction and update operation described by:

$$\eta_n = \mu_{n-1} M_n, \tag{2.25}$$

$$\mu_n = \Psi_n(\eta_n), \tag{2.26}$$

since we can check that $\gamma_n(f) = \lambda_{n-1}(M_n(f))$. This of course requires that the Boltzmann-Gibbs transformation is well defined and that $\eta_n(G_n) > 0$ always.

2.8.2 Properties of the Models

The Feynman-Kac models $(\eta_n, \mu_n)_{n\geq 0}$ can be viewed recursive measure valued processes obeying operators

$$\eta_n = \Phi_n(\eta_{n-1}), \tag{2.27}$$

$$\mu_n = \Upsilon_n(\mu_{n-1}), \tag{2.28}$$

with the operators defined as

$$\Phi_n(\eta) = \Psi_{n-1}(\eta)M_n,$$
$$\Upsilon_n(\mu) = \Psi_n(\mu M_n).$$

The importance of the regularity condition (2.18) on G_n becomes more clear since the operators of (2.27)-(2.28) require that the denominator of the Boltzmann-Gibbs transformation is well defined. It might occur for example that $\eta_0(G_0) = 0$ or $\mu_0(G_0) = 0$ or that if G_n is unbounded then Ψ_n can be defined only for the set of measures $\eta \in \mathcal{P}(E)$ such that $\eta_0(G_n) \in (0, \infty)$. This brings up the need to impose some restrictions on the pair (G_n, M_n) and further account for what happens in case these are violated. Let us impose the following condition.

Assumption 2.8.1 ***(A1)****, [37, p.19]. For any $x_n \in E_n$, the pairs (G_n, M_n) satisfy*

$$M_{n+1}(G_{n+1}) > 0,$$
$$\sup_{x_n \in E_n} |M_{n+1}(G_{n+1})(x_n)| < \infty.$$

If this assumption is fulfilled, the following operators can be defined

$$G'_n = M_n(G_n), \tag{2.29}$$

$$M'_n(x_{n-1}, dx_n) = \frac{M_n(x_{n-1}, dx_n)G_n(x_n)}{M_n(G_n)}. \tag{2.30}$$

Then we can write the transition operator for the flow μ_n as

$$\Phi_n(\mu) = \Psi'_{n-1}(\mu)M',$$

where $\Psi'_{n-1}(\mu)$ is the Boltzmann-Gibbs transformation for the associated Feynman Kac pair (G'_n, M'_n). It is more apparent now that the flow starting from η_0 can be studied using both pairs (G_n, M_n) and (G'_n, M'_n) using an initial measure η'_0. The updated models associated with

(G_n, M_n) and initial measure η_0 coincide with the prediction models associated with (G'_n, M'_n) and initial measure $\eta'_0 = \Psi_0(\eta_0)$.

We will use this observation in order to relax assumption (A1) and examine the case where G_n can take some null values. We can consider a subset of E_n in which the null values of G_n have been excluded, so define the set S_n as $S_n = G_n^{-1}((0,\infty))$. In general $S_n \subseteq E_n$, but it is convenient to restrict the study of the flows distributions $\{\eta_n, \mu_n\}$ on S_n. This can bypass the case where $M_n(x_{n-1}, S_n) = 0$, for some $x_{n-1} \in E_{n-1}$, which will result $M_n(G_n)(x_{n-1}) = 0$ and (A1) not to be met. Let us pose the following accesibility assumption on S_n.

Assumption 2.8.2 ***(A2)****, [34, p.67]. We have $\eta_0(S_0) > 0$ and for each $n \geq 1$ and $x_n \in S_n$, we have that $M_{n+1}(x_n, S_{n+1}) > 0$.*

Given this assumption we ensure that S_n is M_n accessible from any point in S_{n-1}. Assuming (A2) is met then the conditions of (A1) are only met for any $x_n \in S_n$, and the operators M'_n, which are already defined for any $x_{n-1} \in S_{n-1}$, are well defined Markov kernels from S_{n-1} to S_n. In addition, for any $\eta_0(S_0) > 0$ the updated measure $\eta'_0 = \Psi_0(\eta_0)$ is such that $\eta'_0(S_n) = 1$. As soon as (A2) is met then we can use the following interpretation for the updated Feynman-Kac measures μ_n and λ_n,

$$\mu_n = \eta'_n, \tag{2.31}$$

$$\lambda_n(f) = \eta_0(G_0)\gamma'_n(f). \tag{2.32}$$

Using (2.32) we can show that for any $n \geq 0$ we always have $\eta_n(G_n) > 0$. Finally in the case where at some point in time τ, the accesibility condition (A2) fails for $M_{\tau+1}$ the flow η_n is well defined up to time τ, but η_τ cannot be updated anymore and terminates, and $\lambda_\tau(1) = 0$.

2.8.3 Interacting Particle Algorithm

In this section we shall present an interacting particle method to approximate the flows. This is based on an interacting process physical interpretation of the Feynman-Kac models as a non-linear measure valued process. From now on we shall assume G_n satisfies $0 < G_n \leq 1$. This is not restrictive since for bounded G_n, if we use $\frac{G_n(x_n)}{\sup_{x_n}|G_n(x_n)|}$ as the potential instead, the definition of the normalised measures μ_n, η_n remains unaltered. Moreover we have already shown how the flow can be analysed via a transformation when zero values of G_n occur. Therefore this simplification does not cause any loss in generality.

We recall equation (2.27),

$$\eta_n = \Phi_n(\eta_{n-1}),$$

where the prediction flow η_n is viewed as recursive measure valued processes. Consider a family of kernels $\{K_{n+1,\eta_n}\}_{n\geq 0}$ such that,

$$\eta_{n+1} = \eta_n K_{n+1,\eta_n}.$$

Definition 2.8.4 *[34, Def. 2.5.4] Any realisation of $\eta \in \mathcal{P}(E_n)$ that satisfies*

$$\Phi_{n+1}(\eta) = \eta K_{n+1,\eta} \tag{2.33}$$

is called a McKean interpretation of the flow η_n.

Although this choice is not unique, we shall use it as in [34, Def 2.5.4] with K_{n+1,η_n} being a Markov kernel given by

$$K_{n+1,\eta_n}(x, dz) = S_{n,\eta_n} M_{n+1}(x, dz). \tag{2.34}$$

The kernel $S_{n,\eta_n}(x, dy)$ is an interacting selection transition given by

$$S_{n,\eta_n}(x, dy) = G_n(x)\delta_x(dy) + (1 - G_n(x))\Psi_n(\eta_n)(dy). \tag{2.35}$$

The Markov evolution of the flow then can be defined as a two step transition:

$$X_n \overset{\text{interacting jump}}{\longrightarrow} \widehat{X}_n \sim S_{n,\eta_n}(X_n, \cdot) \overset{\text{prediction}}{\longrightarrow} X_{n+1} \sim M_{n+1}(\widehat{X}_n, \cdot).$$

In order to force X_n in areas of high potential directed by G_n, X_n either remains at its location with probability G_n or otherwise jumps into a new location randomly chosen from the Boltzmann Gibbs distribution [34, p. 75],

$$\Psi_n(\eta_n)(dx_n) = \frac{G_n(x_n)\eta_n(dx_n)}{\eta_n(G_n)}.$$

This is followed by a prediction step according to the Markov transition density.

2.8.4 Particle Approximation

We shall proceed by presenting an interacting particle system that approximates the flows η_n and μ_n. The particle system is initialised by sampling N independent samples or particles, from common initial density ν,

$$\xi_0^i \sim \nu(\cdot),$$

where i is the sample's index and $1 \leq i \leq N$. At each time n we have a set of N independent, uniformly weighted, particles ξ_n^i. Define the discrete collection of particles as $\xi_n = (\xi_n^1, ..., \xi_n^N)$, where $\xi_n \in E_n^N$.

The prediction flow η_n is approximated by $\widehat{\eta}_n$, which is given as follows,

$$\widehat{\eta}_n(dx_n) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\xi_n^i}(dx_n).$$

The particle system aims to propagate the particles ξ_n in the same spirit as (2.33), using the following Markov transition

$$\mathbb{P}_\nu(\xi_{n+1} \in dx_n^{1:N} | \xi_n) = \mathcal{K}_{n+1,\widehat{\eta}_n}(\xi_n, dx_{n+1}^{1:N}).$$

We have introduced a transition kernel $\mathcal{K}_{n+1,\widehat{\eta}_n}$ for the Markov chain $\{\xi_n\}_{n\geq 0}$ defined on the product space E_n^N. In the same spirit as (2.34) we decompose $\mathcal{K}_{n+1,\widehat{\eta}_n}$ to an update and prediction step resembled by $\mathcal{S}_{n,\widehat{\eta}_n}$ and $\mathcal{M}_{n+1}$ respectively

$$\mathcal{K}_{n+1,\widehat{\eta}_n}(\xi_n, dx_{n+1}^{1:N}) = \int \mathcal{S}_{n,\widehat{\eta}_n}(\xi_n, dx_n^{1:N}) \mathcal{M}_{n+1}(x_n^{1:N}, dx_{n+1}^{1:N}),$$

where $\mathcal{S}_{n,\widehat{\eta}_n}$ and $\mathcal{M}_{n+1}$ act also on the product space E_n^N and are given by

$$\mathcal{S}_{n,\widehat{\eta}_n}(\xi_n, dx_n^{1:N}) = \prod_{i=1}^{N} S_{n,\widehat{\eta}_n}(\xi_n^i, dx_n^i),$$
$$\mathcal{M}_{n+1}(x_n^{1:N}, dx_{n+1}^{1:N}) = \prod_{i=1}^{N} M_{n+1}(x_n^i, dx_{n+1}^i).$$

Compared to (2.35), we slightly modify $S_{n,\eta_n}(x, dy)$ to include a non negative parameter ε_n as in [34, p.98],

$$S_{n,\eta_n}(x, dy) = \varepsilon_n G_n(x_n)\delta_x(dy) + (1 - \varepsilon_n G_n(x_n))\Psi_n(\eta_n)(dy), \tag{2.36}$$

where $\varepsilon_n G_n \leq 1$. Note that apart from that there is no restriction on ε_n and it may depend on η_n as well. Once we substitude in this expression $\widehat{\eta}_n$ for η_n, one can obtain the particle version, $S_{n,\widehat{\eta}_n}$:

$$S_{n,\widehat{\eta}_n}(\xi_n^i, dx_n^i) = \varepsilon_n G(\xi_n^i)\delta_{\xi_n^i}(dx_n^i) + (1 - \varepsilon_n G(\xi_n^i)) \sum_{j=1}^{N} \frac{G(\xi_n^j)}{\sum_{k=1}^{N} G(\xi_n^k)} \delta_{\xi_n^j}(dx_n^j). \tag{2.37}$$

Moreover $M_{n+1}(x_n^i, dx_{n+1}^i)$ is basically the standard Markov transition step of the chain that the particle is used to approximate.

To sum up, using less formalism we can say that the nonlinear flow is approximated by a set of a few particles each of which undertakes a selection and mutation step as follows:

$$\xi_n^i \overset{\text{selection}}{\longrightarrow} \widehat{\xi}_n^i \sim S_{n,\widehat{\eta}_n}(\xi_n^i, \cdot) \overset{\text{mutation}}{\longrightarrow} \xi_{n+1}^i \sim M_{n+1}(\widehat{\xi}_n^i, \cdot),$$

[34, p.104]. During the selection step each particle $\widehat{\xi}_n^i$ remains at its previous location given by ξ_n^i with probability $\varepsilon_n G(\xi_n^i)$, otherwise a new particle $\widetilde{\xi}_n$ is sampled instead from the discrete Boltzmann-Gibbs distribution $\Psi_n(\widehat{\eta}_n)(dx_n^{1:N}) = \sum_{j=1}^{N} \frac{G(\xi_n^j)}{\sum_{k=1}^{N} G(\xi_n^k)} \delta_{\xi_n^j}(dx_n^j)$ and we set $\widehat{\xi}_n^i = \widetilde{\xi}_n$. During the mutation step each sample is propagated independently following M_{n+1}.

We can then use the particle sets $\widehat{\xi}_n$ and ξ_{n+1} to approximate μ_n and η_{n+1} respectively:

$$\widehat{\mu}_n(dx_n) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\widehat{\xi}_n^i}(dx_n),$$

$$\widehat{\eta}_{n+1}(dx_{n+1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\xi_{n+1}^i}(dx_{n+1}).$$

In [34], asymptotic convergence results can be found. We do not expand this discussion further as it is not within our scope to analyse particle methods in more detail.

3

General State Space Models

Summary. *In this chapter we will present general state space models. These models are also known as Hidden Markov Models (HMM) and encompass a broad class of dynamic models, which do not require restrictive assumptions, such as discreteness of the state space, linearity of the dynamics or Gaussian noise. We shall touch on the topic of parameter estimation for these models and also see how these models can be extended to describe decision or control problems. We will then provide a brief review of Graphical Models and show how these can be used in the context of general state space models, so that their formulation can be extended for problems regarding distributed systems.*

3.1 Introduction

The main purpose of this chapter is to introduce general state space models, which are also known as Hidden Markov Models (HMM), as a powerful modelling framework for a variety of problems. We will present formulations that are appropriate for problems in the areas of optimal filtering [2], control [20] and parameter estimation [107]. The formulation of the models is taken from [5,8,50–52,157] and has also appeared in the same or similar context in [30,111,114]. In order to address problems regarding distributed systems, we will attempt to extend these models so that they can be used together with Graphical Models (GM) and Belief Propagation (BP) [82, 95, 130, 171]. This has already been initiated in [29, Ch.5] for optimal filtering in dynamic Graphical Models and in this thesis we aim to extend that work so that it can be used for parameter estimation.

The main computational tool used in this thesis is Sequential Monte Carlo (SMC) methods, also known in the context of optimal Bayesian filtering as Particle Filters (PF). These have been presented in its general form in Section 2.7. SMC consists of a set of powerful simulation-based techniques that combine Importance Sampling and resampling methods and allow one to sample sequentially in time from a sequence of complex distributions. Particle filters have become increasingly popular during the last decade for performing optimal Bayesian filtering for a general state space models [47, 48, 103]. In this thesis we shall also formulate problems in the areas of control or parameter estimation using this type of models, and show how different SMC approximations can be used for solving these problems.

The organisation of this chapter is as follows: in Section 3.2 we define general state space models and in Section 3.3 we present a formulation suited to parameter estimation problems. In Section 3.4 we introduce optimal Bayesian filtering and particle filters. Moreover, in Section 3.5 one can find an overview of popular Maximum Likelihood Estimation (MLE) methods for static parameters in general state space models. We also show how to obtain SMC approximations for MLE. In Section 3.6 we formulate general state space models for control and discuss various SMC approximations for control problems. Finally in Section 3.7, we present a brief review of Graphical models and Belief Propagation and present a distributed formulation of Hidden Markov Models, which can be used as a framework for distributed filtering and parameter estimation. A Recursive Maximum Likelihood implementation is also derived.

3.2 Description of the Models

Consider a homogeneous Markov chain $\{X_n\}_{n\geq 0}$ defined on $(\mathcal{X}, \mathcal{E}_\mathcal{X})$ with initial density μ and Markov transition density $M(x_n, dx_{n+1})$. Then suppose that one can obtain indirect observations of the sequence $\{X_n\}_{n\geq 0}$ via some sequence of observations $\{Y_n\}_{n\geq 0}$ defined on $(\mathcal{Y}, \mathcal{E}_\mathcal{Y})$, which is independent conditional on $\{X_n\}_{n\geq 0}$ and at time n the conditional distribution of Y_n depends only on X_n. The bivariate process $\{(X_n, Y_n)\}_{n\geq 0}$ is called a *Hidden Markov Model* (HMM). We proceed with a more formal definition taken from [157].

Definition 3.2.1 *Let $\{X_n\}_{n\geq 0}$ be a Markov chain with initial density μ defined on $(\mathcal{X}, \mathcal{E}_\mathcal{X}, \mathbb{P})$ and $\{Y_n\}_{n\geq 0}$ on $(\mathcal{Y}, \mathcal{E}_\mathcal{Y}, \mathbb{P})$ and M and G denote, respectively, a Markov transition kernel from $(\mathcal{X}, \mathcal{E}_\mathcal{X})$ to $(\mathcal{X}, \mathcal{E}_\mathcal{X})$ and a transition kernel from $(\mathcal{X}, \mathcal{E}_\mathcal{X})$ to $(\mathcal{Y}, \mathcal{E}_\mathcal{Y})$. The bivariate process $\{(X_n, Y_n)\}_{n\geq 0}$ is called a Hidden Markov Model (HMM) with state X_n and observation Y_n, if for any sets $B_X \in \mathcal{E}_\mathcal{X}$ and $B_Y \in \mathcal{E}_\mathcal{Y}$, we have for any n*

$$\mathbb{P}(X_n \in B_X | X_{0:n-1} = x_{0:n-1}, Y_{0:n-1} = y_{0:n-1}) = \mathbb{P}(X_n \in B_X | X_{n-1} = x_{n-1}) = \int_{B_X} M(x_{n-1}, dx_n),$$

$$\mathbb{P}(Y_n \in B_Y | X_{0:n-1} = x_{0:n-1}, Y_{0:n-1} = y_{0:n-1}) = \mathbb{P}(Y_n \in B_Y | X_n = x_n) = \int_{B_Y} G(x_n, dy_n).$$

The HMM is itself a Markov chain and using (2.10), for the joint process we can derive the joint transition kernel on the product space $(\mathcal{X} \times \mathcal{Y}, \mathcal{E}_\mathcal{X} \otimes \mathcal{E}_\mathcal{Y})$ as

$$\mathbb{P}((X_n, Y_n) \in B_X \times B_Y | X_{0:n-1} = x_{n-1}, Y_{0:n-1} = y_{0:n-1}) = \iint_{B_X \times B_Y} M(x_{n-1}, dx_n) G(x_n, dy_n).$$

We will relax this formal definition by considering in the remainder of this section the only the case of *fully dominated HMM*s as defined in [30].

Definition 3.2.2 *From [30]. Let there exist a dominating probability measure ρ on $(\mathcal{Y}, \mathcal{E}_\mathcal{Y})$ such that for all $x \in \mathcal{X}$, $G(x, \cdot)$ is absolutely continuous with respect to ρ, $G(x, \cdot) \ll \rho(\cdot)$, with the transition density function being $g(\cdot|x) = \frac{dG(x,\cdot)}{d\rho}$. Also, let there exist a dominating probability measure λ on $(\mathcal{X}, \mathcal{E}_\mathcal{X})$ such that for all $x \in \mathcal{X}$, $\mu(\cdot)$ and $M(x, \cdot)$ are absolutely continuous with respect to λ, $\mu(\cdot) \ll \lambda(\cdot)$ and $M(x, \cdot) \ll \lambda(\cdot)$, with the transition density function being $f(\cdot|x) = \frac{dM(x,\cdot)}{d\lambda}$. The Hidden Markov Model $\{(X_n, Y_n)\}_{n\geq 0}$ is then called fully dominated and the joint Markov transition kernel $M(x', x)G(x, y)$ is dominated by the product measure $\lambda \otimes \rho$ and admits the transition density $f(x|x')g(y|x)$.*

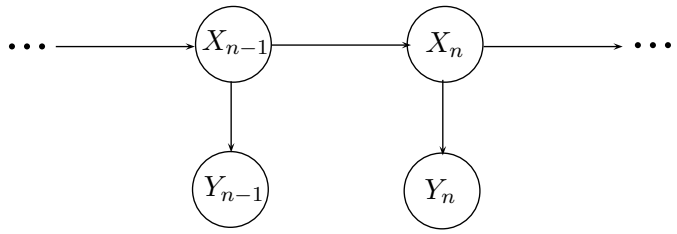


Figure 3.1: *A Hidden Markov Model.*

A Hidden Markov Model is illustrated in Figure 3.1 by means of a Graphical model. Loosely speaking, a HMM covers a wide range of problems that require the estimation of an unobserved, time-varying states of a Markov chain using a sequence of noisy observations. This class of models includes many nonlinear and non-Gaussian time series models such as

$$X_{n+1} = \psi\left(X_n, V_{n+1}\right), \tag{3.1}$$

$$Y_n = \phi\left(X_n, W_n\right), \tag{3.2}$$

where $\{V_n\}_{n\geq 1}$ and $\{W_n\}_{n\geq 0}$ are mutually independent sequences of independent random variables and ψ, ϕ are nonlinear measurable functions that determine the evolution of the state and observation processes. This description of HMMs has appeared vastly in many areas of research also under the name of nonlinear or *general state space models* [114].

Unfortunately, in the time series literature the term HMM has beed widely associated with the case of $\mathcal{X}$ being finite [138]. We want to stress at this point that by using the term HMM we do not limit ourselves to models with finite state spaces, but also include models with continuous state spaces. Such models are often referred to as state-space models in the literature. Again, in some of the control literature the use of the term "state space models" refers to the case of linear Gaussian systems [2]. We emphasise that in this thesis we will avoid making any restrictive assumptions on the dynamics such as linearity of ψ, ϕ or any assumptions on the distributions of V_n, W_n. We shall keep the framework as general as possible and consider the general case of measurable spaces. Also, we clarify that in contrast to previous restrictive use of terminology, we will use both terms HMM and general state space models to describe exactly the same thing as defined by Definitions 3.2.1 and 3.2.2.

In this chapter we aim to show how SMC methods can be used in the context of general state space models to perform optimal Bayesian filtering. Optimal filtering for nonlinear non-Gaussian state space models has numerous applications in signal processing and related areas

such as finance [132], robotics [19], telecommunications [7] etc. However, except for simple models such as linear Gaussian state space models, optimal filters do not typically admit a closed-form expression. Standard approximation schemes can be unreliable [71], e.g. Extended Kalman filter, or difficult to implement, such as deterministic integration methods. Sequential Monte Carlo (SMC) methods, also known as particle methods, are simulation-based approximations of the posterior distributions of interest that are both easy to implement and have been demonstrated in numerous settings to yield more accurate estimates than the previous two methods mentioned [48, 53, 85, 103]. Based on optimal Bayesian filtering, we should introduce how SMC can be used for parameter estimation and control problems in the context of general state space models. The content of the remainder of this chapter is based on the material found in the review [8], the papers in [47] and in the recent thesis [136].

3.3 General State Space Models for Parameter Estimation

Let $\{X_n\}_{n\geq 0}$ and $\{Y_n\}_{n\geq 0}$ be $\mathcal{X}$ and $\mathcal{Y}$-valued stochastic processes defined on a measurable space $(\Omega, \mathcal{F})$ and suppose that $\theta \in \Theta$ is the parameter vector where Θ is an open subset of $\mathbb{R}^{n_\theta}$. A general discrete-time state space model represents the unobserved state $\{X_n\}_{n\geq 0}$ as a Markov process of initial density $X_0 \sim \mu$ and Markov transition density $f_\theta(x'|x)$. The process $\{X_n\}_{n\geq 0}$ is not itself observed, but instead indirect measurements of the hidden states are available through an observation sequence $\{Y_n\}_{n\geq 0}$. The observations $\{Y_n\}_{n\geq 0}$ are assumed conditionally independent given $\{X_n\}_{n\geq 0}$ and are characterised by a conditional marginal density $g_\theta(y|x)$. The model is summarised as follows

$$\begin{aligned} X_n|X_{n-1} = x_{n-1} &\sim f_\theta(\,.\,|x_{n-1}), \\ Y_n|X_n = x_n &\sim g_\theta(\,.\,|x_n). \end{aligned} \tag{3.3}$$

All densities are taken with respect to appropriate dominating measures, e.g. the Lebesgue measure. Here θ denotes a static parameter, e.g. the dynamic noise variance. This parameter is either known or unknown dependent on the application under study. For the time being, we shall assume that θ is known or has already been estimated.

With refence to Section 2.7, in the general state space model the sequence of distributions $\{\pi_n\}_{n\geq 0}$ of interest are the sequence of posterior densities $\{p(x_{0:n}|y_{0:n})\}_{n\geq 0}$. So for the remain-

der of this chapter we can say for distributions of interest that

$$\pi_n(x_{0:n}) = p(x_{0:n}|Y_{0:n}).$$

This posterior density constitutes a complete solution to the state inference problem as it summarises all that is known about the hidden states given the observations. In this section we shall focus on presenting how the general SMC algorithms of the previous section can be used for approximating expectations of the form $\pi_n(\varphi_n)$, where φ_n is some measurable function, such that $\varphi_n : \mathcal{X}^n \longrightarrow \mathbb{R}^{n_\varphi}$.

Using the properties of the state space model and Bayes' theorem, the following recursion holds

$$p(x_{0:n}|Y_{0:n}) = \frac{g_\theta(Y_n|x_n) f_\theta(x_n|x_{n-1})}{p(Y_n|Y_{0:n-1})} p(x_{0:n-1}|Y_{0:n-1}), \tag{3.4}$$

where the normalising constant is given by

$$p(Y_n|Y_{0:n-1}) = \int g_\theta(Y_n|x_n) p(x_n|Y_{0:n-1}) dx_n. \tag{3.5}$$

Note that the probability distributions $p(x_{0:n}|Y_{0:n})$, $p(Y_n|Y_{0:n-1})$, etc. are inherently depending on the static parameter θ. In terms of notation, we could have illustrated this inherent dependance by means of some subscript on p. However as all distributions are θ dependent we omit this for simplicity.

The interest in the posterior of $p(x_{0:n}|Y_{0:n})$ might be for direct inference on the hidden state $\{X_n\}_{n\geq 0}$ or for making predictions on $\{Y_n\}_{n\geq 0}$. In general, the marginal posterior density $p(x_k|Y_{0:n})$ is called a *smoothing*, *filtering* or *prediction density* if $k < n$, $k = n$ and $k > n$, respectively. Here we will only consider the filtering case. The filtering density is usually obtained recursively in two stages, prediction and update. These are given as

$$\textbf{Prediction } p(x_n|Y_{0:n-1}) = \int f_\theta(x_n|x_{n-1}) p(x_{n-1}|Y_{0:n-1}) dx_{n-1}, \tag{3.6}$$

$$\textbf{Update } \quad p(x_n|Y_{0:n}) = \frac{g_\theta(Y_n|x_n) p(x_n|Y_{0:n-1})}{p(Y_n|Y_{0:n-1})}, \tag{3.7}$$

In some special cases, the above integrals can be computed exactly. For instance, if the model in (3.3) is a linear Gaussian state space model, the filtering problem formulated by (3.6) and (3.7) has an optimal solution in the Minimum Mean Square Error (MMSE) sense, given by the Kalman filter [2]. An exact solution is also available for the case of a finite state Hidden Markov Model (HMM) [138]. In general however, none of the above integrals admit a closed-form expression and one typically resorts to numerical approximations. Past approaches to

the problem used approximations such as the Extended Kalman filter, the Gaussian sum filter and approximate grid-based methods [2]. The Extended Kalman filter (EKF) is a common approach that approximates the nonlinear equations by local linearisation, using the first term of their Taylor series expansion. After the linearisation is performed, standard Kalman filtering can be employed, under the assumption that the model is Gaussian. As one would expect, this method fails if the model has substantial nonlinearities or if the model noises are far from Gaussian (e.g. multi-modal or heavily skewed). This last drawback was addressed in [83], where the authors proposed the so called Unscented Kalman Filter, which yielded considerable improvement compares to the standard EKF. Finally, another approximation that has been used assumes that the continuous state space can be represented by a finite number of values. This allows an approximate grid-based method to be employed. The approximation improves as the discretisation gets more dense, however the computational cost increases dramatically as the dimensions increase.

3.4 Optimal Bayesian filtering

SMC methods can provide an efficient solution to the optimal filtering problem. As already mentioned, when one applies Monte Carlo methods to non trivial problems, it is generally impossible to sample from the target distribution. A way to circumvent this problem in a recursive setting is to employ a Sequential Importance Sampling Resampling (SISR) approach [46, 85, 103].

One can rewrite (3.4) as

$$p\left(x_{0:n}|\, Y_{0:n}\right) = \frac{\alpha_n\left(x_{n-1:n}, Y_n\right) q\left(x_n|Y_n, x_{0:n-1}\right) p\left(x_{0:n-1}|\, Y_{0:n-1}\right)}{p\left(Y_n|\, Y_{0:n-1}\right)} \tag{3.8}$$

where $q\left(x_n|Y_n, x_{0:n-1}\right)$ is the importance distribution and the corresponding weight is given by

$$\alpha_n\left(x_{n-1:n}, Y_n\right) = \frac{g_\theta\left(Y_n|\, x_n\right) f_\theta\left(x_n|\, x_{n-1}\right)}{q\left(x_n|Y_n, x_{0:n-1}\right)}. \tag{3.9}$$

One could actually sample from a proposal conditioned to all the sequence of observations up to time n, $q\left(\cdot|\, Y_{0:n}, x_{0:n-1}\right)$, but this generalization is not useful for the class of models considered. Similar to Section 2.7 and particulary (2.14) one can write,

$$q(x_{0:n}|Y_{0:n-1}) = q\left(x_0|Y_0\right) \prod_{k=1}^{n} q\left(x_k|Y_{k,}x_{0:k-1}\right). \tag{3.10}$$

The importance ratio for the complete path becomes

$$\frac{p(x_{0:n}|Y_{0:n})}{q\left(x_{0:n}|Y_{0:n}\right)} \propto \prod_{k=1}^{n} \alpha_k(x_{k-1:k,Y_k}).$$

With reference to general SIS, the sequential importance weight update of equation (2.15) will be

$$w_n(x_{0:n}, Y_{0:n}) \propto w_{n-1}(x_{0:n-1}, Y_{0:n-1})\alpha_n(x_{n-1:n}, Y_n),$$

where $Y_{0:n}$ was added to the argument of w_n to account for the dependence on the observation sequence.

Using Proposition 2.7.1, the optimal choice of the importance density in the sense that it minimises the variance of the weights conditional on Y_n and x_{n-1} [46], can be readily shown to be

$$\begin{aligned} q\left(x_n|Y_n, x_{0:n-1}\right) &= p\left(x_n|Y_n, x_{n-1}\right) \\ &= \frac{g_\theta\left(Y_n|x_n\right) f_\theta\left(x_n|x_{n-1}\right)}{p(Y_n|x_{n-1})}, \end{aligned} \tag{3.11}$$

where

$$p(Y_n|x_{n-1}) = \int g_\theta\left(Y_n|x_n\right) f_\theta\left(x_n|x_{n-1}\right) dx_n. \tag{3.12}$$

This optimal approach requires the ability to sample from (3.11) and compute (3.12) analytically. This is possible only in some rare cases, therefore in the general case various approximations should be used.

3.4.1 Particle Filters

Particle filters are a set of SMC methods that are widely used to numerically approximate the filtering recursion in (3.4). We sketch here briefly a SISR method to approximate the optimal filter based on the sampling resampling approach. More elaborate algorithms are reviewed in [47] . Assume at time $n-1$, one has a collection of N particles $\left\{\widetilde{X}_{0:n-1}^{(i)}\right\}_{i=1}^{N}$ distributed approximately according to $p\left(x_{0:n-1}|\, Y_{0:n-1}\right)$. At time n, one wants to obtain N particles distributed approximately according to $p\left(x_{0:n}|\, Y_{0:n}\right)$. To achieve this, one samples $\widehat{X}_n^{(i)} \sim q\left(\cdot|\, Y_n, \widetilde{X}_{n-1}^{(i)}\right)$. It follows that the empirical distribution of the particles $\left\{\widehat{X}_{0:n}^{(i)}\right\}$ approximates the joint density $p\left(x_{0:n-1}|\, Y_{0:n-1}\right) q\left(x_n|\, Y_n, x_{n-1}\right)$. By plugging this empirical distribution in (3.8), one obtains

the following approximation of $p(x_{0:n}|Y_{0:n})$

$$\widehat{p}(x_{0:n}|Y_{0:n}) = \sum_{i=1}^{N} w_n^{(i)} \delta_{\widehat{X}_{0:n}^{(i)}}(x_{0:n}), \tag{3.13}$$

i.e. each particle $\widehat{X}_{0:n}^{(i)}$ has now a weight $w_n^{(i)}$ where

$$w_n^{(i)} \propto \alpha_n\left(\widehat{X}_{n-1:n}^{(i)}, Y_n\right), \ \sum_{i=1}^{N} w_n^{(i)} = 1.$$

Then a resampling step is applied to the particles to get a new set of particles $\left\{\widetilde{X}_{0:n}^{(i)}\right\}_{i=1}^{N}$.

The computational complexity of this algorithm is in $O(N)$. The memory requirements are in $O(N(n+1))$ if one stores the whole paths, but if one is only interested in estimating the marginal density $p(x_n|Y_{0:n})$ then the only memory requirements to update the algorithm are in $O(2N)$ to store $\left\{w_n^{(i)}, \widetilde{X}_n^{(i)}\right\}$. Various other extensions to particle filter algorithms have been proposed in the literature. These include continuous approximations of the posterior density [120], MCMC moves for the particles [24, 69] and different schemes for obtaining the importance densities [164].

3.5 Maximum likelihood Estimation

So far we have considered optimal Bayesian filtering for the case where θ is known or has been somehow estimated. Of course, this is not always the case and in many applications θ has to be estimated. In this section we shall introduce briefly Maximum Likelihood (ML) estimation, which will be a an important topic for the problems to be seen later in this thesis.

For the remainder of this thesis we will always assume that $\{Y_n\}_{n\geq 0}$ is generated from the true value θ^*, so that $Y_n|X_n = x_n \sim g_{\theta^*}(\ .\ |x_n)$. The likelihood of $Y_{0:n}$ with respect to the unknown parameter, $\theta \in \Theta$, is given by

$$p_\theta(Y_{0:n}) = \int \cdots \int \mu(x_0) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1}) \prod_{k=0}^{n} g_\theta(Y_k|x_k)\, dx_{0:n}. \tag{3.14}$$

It also admits the following recursive form

$$p_\theta(Y_{0:n}) = \prod_{k=0}^{n} p_\theta(Y_k|Y_{0:k-1}) \tag{3.15}$$

with $p(Y_0|Y_{-1}) \triangleq \int g_\theta(Y_0|x_0)\,\mu(x_0)\,dx_0$. Note that now we have used θ in the subscript of the distributions, as it is more important to emphasize that we are searching for optimal θ and that different estimates of θ, will result in different distributions.

The traditional approach of ML would be to find the maximiser of $p_\theta\left(Y_{0:n}\right)$ with respect to θ. In practice, one uses the log-likelihood, which is numerically better behaved and satisfies

$$l_\theta\left(Y_{0:n}\right) = \log p_\theta\left(Y_{0:n}\right) = \sum_{k=0}^{n} \log p_\theta\left(Y_k \middle| Y_{0:k-1}\right). \tag{3.16}$$

Maximum likelihood estimation can be then defined as finding an estimate of θ^* by computing

$$\theta_{ML} = \arg\max_\theta l_\theta\left(Y_{0:n}\right).$$

Except in a few simple cases, it is impossible to compute the optimal filter and the log-likelihood/ likelihood in closed-form and one requires numerical approximation schemes.

3.5.1 Particle Approximations for the Likelihood

Based on the particle approximations of the filtering distributions in Section 3.4.1, it is possible to come up with an approximation of the likelihood function. Using (3.5) and (3.9), one clearly has

$$p(Y_n|Y_{0:n-1}) = \int\int \alpha_n(x_{n-1:n}, Y_n) q_\theta\left(x_n|Y_n, x_{0:n-1}\right) p(x_{n-1}|Y_{0:n-1}) dx_{n-1:n} \tag{3.17}$$

which can clearly lead to the following particle approximation $\widehat{p}_\theta\left(Y_n \middle| Y_{0:n-1}\right)$

$$\widehat{p}_\theta\left(Y_n \middle| Y_{0:n-1}\right) = \frac{1}{N}\sum_{i=1}^{N} \alpha_n\left(\widehat{X}_{n:n-1}^{(i)}, Y_n\right), \tag{3.18}$$

where the notation follows from Section 3.4.1. Note that if the importance density is chosen equal to the prior f_θ, then (3.18) becomes

$$\widehat{p}_\theta\left(Y_n \middle| Y_{0:n-1}\right) = \frac{1}{N}\sum_{i=1}^{N} g_\theta\left(Y_n \middle| \widetilde{X}_n^{(i)}\right).$$

If the importance density is optimal in terms of minimization of $var_{q_\theta}\left[w_\theta\left(x_{n-1:n}, Y_n\right) \middle| x_{n-1}\right]$, that is

$$q\left(x_n \middle| Y_n, x_{n-1}\right) = \frac{g_\theta\left(Y_n \middle| x_n\right) f_\theta\left(x_n \middle| x_{n-1}\right)}{\int g_\theta\left(Y_n \middle| x_n\right) f_\theta\left(x_n \middle| x_{n-1}\right) dx_n},$$

then (3.18) becomes

$$\widehat{p}_\theta\left(Y_n \middle| Y_{0:n-1}\right) = \frac{1}{N}\sum_{i=1}^{N} \int g_\theta\left(Y_n \middle| x_n\right) f_\theta\left(x_n \middle| \widetilde{X}_{n-1}^{(i)}\right) dx_n.$$

If the resampling scheme is unbiased, i.e. the expected number of times a particle is copied in the resampling scheme is equal to its normalized weight, one can show that (3.18) is an unbiased estimate of $p_\theta\left(Y_n|\, Y_{0:n-1}\right)$.

However if we substitute any of these approximations for $p(Y_n|Y_{0:n-1})$ to get an estimate of the log-likelihood

$$\widehat{l}_\theta\left(Y_{0:n}\right) = \sum_{k=0}^{n} \log \widehat{p}_\theta\left(Y_k|\, Y_{0:k-1}\right), \tag{3.19}$$

it will be obviously biased. According to [8], this bias can be reduced by using the following standard correction technique based on an first order Taylor expansion. As $N \to \infty$, one typically has

$$\mathbb{E}\left[\widehat{p}_\theta\left(Y_{0:n}\right)\right] = p_\theta\left(Y_{0:n}\right)$$

as well as

$$var\left[\widehat{p}_\theta\left(Y_{0:n}\right)\right] = \frac{\sigma^2}{N},$$

where σ^2 can be easily estimated (call the estimate $\widehat{\sigma}^2$) using the particles. One then gets for a second order Taylor expansion

$$\begin{aligned}\mathbb{E}\left[\widehat{l}_\theta\left(Y_{0:n}\right)\right] &= \mathbb{E}\left[\log\left(\widehat{p}_\theta\left(Y_{0:n}\right)\right)\right] \\ &\simeq \log p_\theta\left(Y_{0:n}\right) - \frac{1}{2}\frac{\sigma^2}{N(p_\theta\left(Y_{0:n}\right))^2}\end{aligned}$$

so we can get the following bias corrected estimate of the log likelihood as

$$\widehat{\log\left(p_\theta\left(Y_{0:n}\right)\right)} \simeq \widehat{l}_\theta\left(Y_{0:n}\right) + \frac{1}{2}\frac{\widehat{\sigma}^2}{N \exp 2\widehat{l}_\theta\left(Y_{0:n}\right)}. \tag{3.20}$$

If Θ is a discrete space or some discretized version of a continuous space, given these approximations the search for the maximiser of the log likelihood in the domain of θ would be a rather simple task. Unfortunately, this is not the case and even in discrete or discretized spaces such an approach would not be able to handle a θ of large dimension. Therefore we feel one should also consider using a stochastic gradient type search.

3.5.2 Log Likelihood gradient

To obtain the log likelihood gradient, we can using (3.16) to obtain the so-called *score* function, see [131],

$$\nabla_\theta l_\theta\left(Y_{0:n}\right) = \sum_{k=0}^{n} \frac{\nabla_\theta p_\theta\left(Y_k|\, Y_{0:k-1}\right)}{p_\theta\left(Y_k|\, Y_{0:k-1}\right)} \tag{3.21}$$

where using (3.17) we get

$$\nabla_\theta p_\theta\left(Y_k|\, Y_{0:k-1}\right) = \int\int \nabla_\theta \alpha_k\left(x_{k-1:k}, Y_k\right).q_\theta\left(x_k|\, Y_k, x_{k-1}\right) p_\theta\left(x_{k-1}|\, Y_{0:k-1}\right) dx_{k-1:k} \quad (3.22)$$
$$+ \int\int \alpha_k\left(x_{k-1:k}, Y_k\right).\nabla_\theta\left(q_\theta\left(x_k|\, Y_k, x_{k-1}\right) p_\theta\left(x_{k-1}|\, Y_{0:k-1}\right)\right) dx_{k-1:k}.$$

From now on and for the remainder of this thesis, we shall assume that all functions are regular enough so that we are permitted to exchange the integral and differentiation operator. Except in simple cases, it is impossible to compute the gradients of the optimal filter and of the log-likelihood function in closed-form and one requires numerical approximation schemes. As this topic will be studied in more depth later during this thesis, we shall not give any further details on how to use particle methods to compute such integrals at the moment, but instead refer the interested reader to [136].

3.5.3 Recursive Maximum Likelihood

Recursive Maximum Likelihood is is a gradient algorithm that maximises the *average log likelihood* $l(\theta)$,

$$l(\theta) = \lim_{n\to\infty} \frac{1}{n} l_\theta(Y_{1:n}) = \int_{\mathcal{Y}\times\mathcal{P}(\mathcal{X})} \log\left(\int g_\theta(y|x)\mu(x)dx\right) \lambda_{\theta,\theta^*}(dy, d\mu)$$

Under regularity assumptions including the stationarity of the state space model, then where $\mathcal{P}(\mathcal{X})$ is the space of probability distributions on $\mathcal{X}$, and $\lambda_{\theta,\theta^*}(dy, d\mu)$ is the joint invariant distribution of the measurement and the prediction density, $(Y_n, p_\theta(x_n|Y_{1:n-1}))$ [157].

RML solves for $\theta^* = \arg\max l(\theta)$ recursively using the sequence of observations $\{Y_n\}_{n\geq 1}$ and learns the static parameter θ^* by using a stochastic gradient algorithm where at time n the parameter estimate θ_n is given by

$$\begin{aligned}\theta_{n+1} &= \theta_n + \gamma_n \nabla_\theta \log(p_\theta(Y_n|Y_{1:n-1}))|_{\theta=\theta_n} \\ &= \theta_n + \gamma_n \nabla_{\theta_n} \log(\int g_{\theta_n}(Y_n|x_n) p_{\theta_{1:n}}(x_n|Y_{1:n-1}) dx_n),\end{aligned} \quad (3.23)$$

where $\{\gamma_n\}_{n\geq 1}$ is a sequence of positive real step-sizes, such that $\sum_n \gamma_n = \infty$ and $\sum_n {\gamma_n}^2 < \infty$ (e.g. $\gamma_n = n^{-2/3}$)[1]. Upon receiving Y_n, θ_n is updated in the direction of ascent of the conditional likelihood $\log p_\theta(Y_n|Y_1 : n-1)$.

[1]Note that for the RML, in the prediction density the subscript is now $\theta_{1:n}$ to explicitly acount for θ being updated sequentially and in parallel to the filter. Also it is apparent from the expression of $l(\theta)$ that RML is a Stochastic Approximation (SA) algorithm [38, 39].

The algorithm in the present form is not suitable for online implementation due to the need to evaluate the gradient of $\log p_\theta(Y_n|Y_{1:n-1})$ at $\theta = \theta_n$. Doing so would require browsing through the entire history of observations. This limitation is removed by defining certain intermediate quantities that facilitate the online evaluation of this gradient [98]. In particular, let

$$\dot{g}_{\theta_n}(Y_n|x_n) \equiv \nabla_\theta g_\theta(Y_n|x_n)|_{\theta=\theta_n}\,, \tag{3.24}$$

$$\dot{f}_{\theta_n}(x_{n+1}|x_n) \equiv \nabla_\theta f_\theta(x_{n+1}|x_n)|_{\theta=\theta_n}\,, \tag{3.25}$$

$$\dot{p}_{\theta_{n-1}}(x_n|Y_{1:n-1}) \equiv \nabla_\theta p_\theta(x_n|Y_{1:n-1})|_{\theta=\theta_{n-1}}\,, \tag{3.26}$$

$$\dot{p}_{\theta_n}(Y_n|Y_{1:n-1}) \equiv \nabla_\theta p_\theta(Y_n|Y_{1:n-1})|_{\theta=\theta_n}\,. \tag{3.27}$$

Furthermore, assume $p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})$ and $\dot{p}_{\theta_{n-1}}(x_n|Y_{1:n-1})$ has been computed from the previous iteration of RML (i.e. iteration $n-1$, $n \geq 2$. RML is initialized with an arbitrary value in Θ for θ_1.) Then, given the new observation Y_n the online version of RML computes:

$$p_{\theta_n}(Y_n|Y_{1:n-1}) = \int g_\theta(Y_n|x_n)p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})dx_n, \tag{3.28}$$

$$\dot{p}_{\theta_n}(Y_n|Y_{1:n-1}) = \int \dot{g}_{\theta_n}(Y_n|x_n)p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})dx_n + \int g_\theta(Y_n|x_n)\dot{p}_{\theta_{n-1}}(x_n|Y_{1:n-1})dx_n, \tag{3.29}$$

$$p_{\theta_{1:n}}(x_{n+1}|Y_{1:n}) = p_{\theta_n}(Y_n|Y_{1:n-1})^{-1}\int f_{\theta_n}(x_{n+1}|x_n)g_{\theta_n}(Y_n|x_n)p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})dx_n, \tag{3.30}$$

$$\begin{aligned}\dot{p}_{\theta_n}(x_{n+1}|Y_{1:n}) &= -p_{\theta_n}(Y_n|Y_{1:n-1})^{-1}\dot{p}_{\theta_n}(Y_n|Y_{1:n-1})p_{\theta_{1:n}}(x_{n+1}|Y_{1:n}) \\ &\quad + p_{\theta_n}(Y_n|Y_{1:n-1})^{-1}\int f_{\theta_n}(x_{n+1}|x_n)\dot{g}_{\theta_n}(Y_n|x_n)p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})dx_n \\ &\quad + p_{\theta_n}(Y_n|Y_{1:n-1})^{-1}\int f_{\theta_n}(x_{n+1}|x_n)g_{\theta_n}(Y_n|x_n)\dot{p}_{\theta_{n-1}}(x_n|Y_{1:n-1})dx_n. \\ &\quad + p_{\theta_n}(Y_n|Y_{1:n-1})^{-1}\int \dot{f}_{\theta_n}(x_{n+1}|x_n)g_{\theta_n}(Y_n|x_n)p_{\theta_{1:n-1}}(x_n|Y_{1:n-1})dx_n\end{aligned} \tag{3.31}$$

The parameter is now updated as follows:

$$\theta_{n+1} = \theta_n + \gamma_{n+1}\frac{\dot{p}_{\theta_n}(Y_n|Y_{1:n-1})}{p_{\theta_n}(Y_n|Y_{1:n-1})}.$$

Almost sure convergence of RML to θ^* together with a central limit theorem is established in [**?**] when the hidden process is discrete valued under certain regularity assumptions on the HMM. The intermediate quantities (3.28)-(3.31) can only be computed exactly for HMMs with a discrete valued hidden process or for linear Gaussian state-space models. For a general space models these quantities may be approximated using SMC [134].

3.5.4 Expectation Maximisation

For a given sequence of T observations, the Expectation Maximisation (EM) algorithm for learning θ^* is a two step procedure, [40]. The first step, the expectation or E-step, computes

$$Q(\theta_k, \theta) = \int \log p_\theta(x_{1:T}, Y_{1:T}) p_{\theta_k}(x_{1:T}|Y_{1:T}) dx_{1:T}.$$

The second step is the maximization or M-step that updates the parameter θ_k,

$$\theta_{k+1} = \arg\max_\theta Q(\theta_k, \theta)$$

Upon the completion of an E and M step, the likelihood surface is ascended, i.e. $p_{\theta_{k+1}}(Y_{1:T}) \geq p_{\theta_k}(Y_{1:T})$ [40]. For linear Gaussian state-space models this procedure can be implemented exactly. In the nonlinear non-Gaussian setting SMC methods may be applied [9, 30].

3.6 General State Space Models for Control

So far there has not been any mention of control. This would involve dynamical models for which a specified user or controller or decision maker influences the evolution of the hidden process and the nature of its observation by means of an action or control input A_n at each time n. It is of the interest to the decision maker to choose the sequence $\{A_n\}_{n\geq 0}$, so that it optimises some user specified criterion $J(A_{0:n})$. In this section, we shall consider how such stochastic processes can be described by general state space models and see how SMC can be a powerful computational tool for solving sequential decision or control problems.

We consider here nonlinear non-Gaussian state space models on which it is possible to apply an $\mathcal{A}-$valued control term A_n at time n. More precisely conditional upon $\{A_n\}_{n\geq 0}$, the process $\{X_n\}_{n\geq 0}$ is a Markov process such that $X_0 \sim \mu$ and Markov transition density $f(x'|x, a)$; i.e.

$$X_{n+1}|(X_n = x, A_{n+1} = a) \sim f(\cdot|x, a) \tag{3.32}$$

where the observations $\{Y_n\}_{n\geq 0}$ are conditionally independent of marginal density $g(y|x, a)$; i.e.

$$Y_n|(X_n = x, A_n = a) \sim g(\cdot|x, a). \tag{3.33}$$

In the most general case, the control A_{n+1} at time $n+1$ is a function of all the available information at time n which can be summarized by the optimal filter $p(x_n|Y_{0:n}, A_{0:n})$. We are

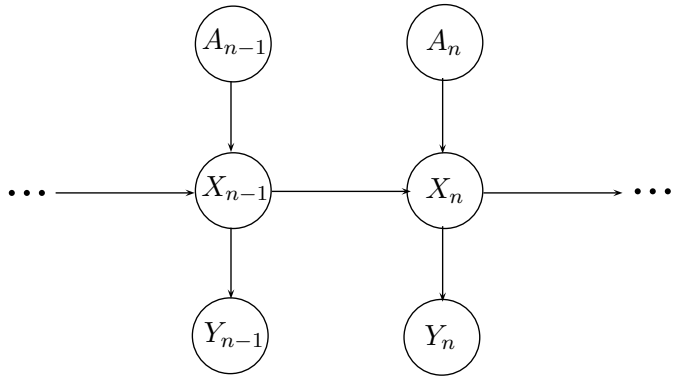


Figure 3.2: *A Partially Observed Markov Decision Process.*

here interested in finite horizon and infinite horizon control problems whose detailed descriptions will be given in the forthcoming subsections. These models are also referred as controlled Hidden Markov models or Partially Observed Markov Decision Processes (POMDP) and are illustrated in Figure 3.2 by means of a simple graphical model.

This class of models includes many nonlinear and non-Gaussian time series models such as

$$X_{n+1} = \psi\left(X_n, A_n, V_{n+1}\right),$$
$$Y_n = \phi\left(X_n, A_n, W_n\right),$$

where $\{V_n\}_{n\geq 1}$ and $\{W_n\}_{n\geq 0}$ are mutually independent sequences of independent random variables and ψ, ϕ are nonlinear functions that determine the evolution of the state and observation processes.

Solving optimal control problems for general state space models with nonlinear non- Gaussian dynamics is a formidable task. Their solutions are given by solving Dynamic Programming or Bellman equations. See [20] for more details. Except in very specific cases, e.g. linear Gaussian state space models and a quadratic cost function, there is no analytical solution to this equation. In the nonlinear non-Gaussian state space models, such a solution admits as argument a probability distribution and it seems extremely difficult to come up with any sensible approximation to it. This is why, despite its numerous applications, the literature on applications of particle methods for control of non linear non Gaussian models is extremely limited.

3.6.1 Finite Horizon Control Problems

Let us introduce a measurable cost function $h : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^+$. In [111] and [142], the authors propose to address the following additive cost decision problem. At time $k-1$, the sequence $A_{0:k-1}$ has been selected and one wants to minimize the function defined as

$$J\left(A_{k:k+H-1}\right) = \mathbb{E}_\mu \left[\sum_{j=k}^{k+H-1} h\left(X_j, A_j\right) \right] \tag{3.34}$$

where the expectations are with respect to the joint distribution of both the states and the observations, i.e.

$$\begin{aligned} &\mathbb{E}_\mu \left[h\left(X_k, A_k\right)\right] \\ &= \int h\left(x_j, A_j\right) \prod_{l=k}^{j} g\left(y_l \middle| x_l, A_l\right) f\left(x_l \middle| x_{l-1}, A_l\right) p\left(x_{k-1} \middle| Y_{1:k-1}, A_{0:k-1}\right) dx_{k-1:l} dy_{k:l} \end{aligned}$$

If the control input takes its values in a finite set $\mathcal{A}$ of cardinality K, It is possible to approximate numerically this cost using particle methods for the K^H possible values of $A_{k:k+H-1}$ and then select the optimal value

$$A^*_{k:k+H-1} = \arg\max J\left(A_{k:k+H-1}\right).$$

To get a particle approximation one can obtain samples from $p\left(x_j \middle| Y_{1:k}, A_{0:j}\right)$ for $j \geq k$, which can be achieved by using the particle approximation (3.13) and then sampling particles $\widetilde{X}_j^{(i)} \sim f\left(\cdot \middle| \widetilde{X}_{j-1}^{(i)}, A_j\right)$ for $j > k$. Then one has the following approximation of (3.34)

$$\widehat{J}\left(A_{k:k+H}\right) = \sum_{j=k}^{k+H-1} \sum_{i=1}^{N} w_n^{(i)} h\left(\widetilde{X}_j^{(i)}, A_j\right).$$

This approximation can be computed for all values of $A_{k:k+H}$ to get $A^*_{k:k+H}$. Of course, in practice this approach cannot handle a large value of H and K.

If $A_{k:k+H}$ takes values in a continuous space $\mathcal{A}^{H+1}$ and $J\left(A_{k:k+H}\right)$ is differentiable with respect to $A_{k:k+H}$, one can still resort to a gradient search in $\mathcal{A}^{H+1}$. An estimate of the gradient of $J\left(A_{k:k+H}\right)$ can be estimated and a stochastic gradient algorithm can be used. This procedure requires a method to be developed for the gradient of the optimal filter. Such problems will be considered in more detail in Chapter 4.

3.6.2 Infinite Horizon Control Problems

In the infinite horizon case, we are interested in selecting a control sequence $\{A_n\}_{n\geq 0}$ minimising an infinite horizon discounted cost

$$J\left(\mu, \{A_n\}_{n\geq 0}\right) = \mathbb{E}_\mu\left[\sum_{k=0}^{\infty} \gamma^k h\left(X_k\right)\right] \tag{3.35}$$

where $0 < \gamma < 1$ is the discount factor and the expectation is with respect to the joint distribution of both the states and the observations.We shall also remark that all expectations are also taken with respect to μ, where $X_0 \sim \mu$, and therefore are a function of μ as well. In [160], a method using particle methods based on Q-learning is proposed to solve (3.38) when $\mathcal{A}$ is a finite set. This method is complex as the Q-factors are functions admitting as arguments probability distributions over $\mathcal{X}$. It is thus necessary to perform further approximations. In [160] a nearest-neighbour type method is proposed to perform quantization in this space.

In the infinite horizon average cost we are interested in minimising

$$J\left(\mu, \{A_n\}_{n\geq 0}\right) \triangleq \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \mathbb{E}_\mu\left[h\left(X_k\right)\right] \tag{3.36}$$

where the expectation is with respect to the joint distribution of both the states and the observations.

We review here two algorithms proposed recently by [52] to solve problem (3.36). Let $\mathcal{P}\left(\mathcal{X}\right)$ is the set of probability distributions on a set $\mathcal{X}$. We consider a *randomized stationary policy* $\nu_\theta : \mathcal{P}\left(\mathcal{X}\right) \to \mathcal{P}\left(\mathcal{A}\right)$ to be such that

$$A_k \sim \nu_\theta\left(p\left(x_{k-1}|\, Y_{0:k-1}, A_{0:k-1}\right)\right) \tag{3.37}$$

or a *deterministic stationary policy* $\Pi_\theta : \mathcal{P}\left(\mathcal{X}\right) \to \mathcal{A}$ such that

$$A_k = \Pi_\theta\left(p\left(x_{k-1}|\, Y_{0:k-1}, A_{0:k-1}\right)\right) \tag{3.38}$$

where $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$ is a parameter to determine. The methods considered in [52] are called *policy gradient* methods. This means that gradient methods are used to compute the parameters of the optimal policy.

Taking this into acount one can rewrite $J\left(\mu, \{A_n\}_{n\geq 0}\right) = J\left(\mu, \theta\right)$. Under some regularity assumptions [52, 157], the joint process $\{A_n, X_n, Y_n, p\left(x_n|\, Y_{0:n}, A_{0:n}\right)\}_{n\geq 1}$ is an ergodic

Markov chain $\theta \in \Theta$ with invariant distribution $\lambda_\theta(da, dx, dy, dp)$. Using this ergodic assumption $J(\mu, \theta) = J(\theta)$; i.e. the cost is independent of the initial distribution μ and

$$J(\theta) = \lim_{n\to\infty} J_n(\theta) \text{ where } J_n(\theta) \triangleq \mathbb{E}_{\lambda_\theta}[h(X_n)].$$

See [52, 157] for more details. Under additional regularity assumptions [156, 157], one also has

$$\lim_{n\to\infty} \nabla_\theta J_n(\theta) = \nabla_\theta J(\theta).$$

We propose to compute

$$\theta^* = \arg\min_{\theta\in\Theta} J(\theta)$$

using a stochastic gradient algorithm

$$\theta_{n+1} = \theta_n - \gamma_n \widehat{\nabla J_n}(\theta),$$

where $\widehat{\nabla J_n}(\theta)$ is an unbiased estimate of the gradient and γ_n is a step size as in RML.

For a randomized policy (3.37), one can easily check that an unbiased estimate of the gradient is given by

$$\widehat{\nabla J_n}(\theta) = \left(\int h(x_n)\, p(x_n|Y_{0:n}, A_{0:n})\, dx_n\right)\left(\sum_{k=1}^{n} \frac{\nabla\nu_\theta(A_k|p(x_{k-1}|Y_{0:k-1}, A_{0:k-1}))}{\nu_\theta(A_k|p(x_{k-1}|Y_{0:k-1}, A_{0:k-1}))}\right). \quad (3.39)$$

In the nonlinear non-Gaussian state space models, we compute the first term using the particle approximation (3.13). Note that as in [16], we need to add a discount factor if we update recursively the second term on the right hand side of (3.39) to prevent the variance of our gradient estimate going to infinity at the cost of adding a bias. A stochastic gradient algorithm to minimize $J(\theta)$ follows directly. We refer the interested reader to [52] for details.

For a deterministic policy (3.38), the notation is somehow imprecise as one should make explicit the dependency of the filter on θ. In this case, it can be seen that an unbiased estimate of the gradient is given by

$$\widehat{\nabla J_n}(\theta) = \int h(x_n)\, \nabla_\theta p_\theta(x_n|Y_{0:n}, A_{0:n})\, dx_n + \left(\int h(x_n)\, p_\theta(x_n|Y_{0:n}, A_{0:n})\, dx_n\right)\nabla_\theta l_\theta(Y_{0:n}).$$

In the non linear non Gaussian state space models, one can easily approximate all these terms, but as it occurred for the finite horizon case this requires a method to be developed for the gradient of the optimal filter. We shall investigate how this can be done in the chapters to follow. Moreover, as in the randomised policy case, a discount factor might have to be added when

we update the score term of (3.21) recursively, so as to prevent the variance of our gradient estimate going to infinity at the cost of adding a small bias [16]. Finally, convergence analysis of SMC algorithms for infinite horizon control requires non-standard stochastic approximation results developed in [156] and geometric ergodicity results developed in [157].

3.6.3 Motivation for Gradient Methods

We will now consider how discretising the state space affects the dynamics of the state space model, the computational efficiency of solving it, and some sub-optimality issues. This will then justify the use of policy gradient and stochastic approximation. Because of the absence of a closed-form expression for complex high dimensional integrals of Dynamic Programming or Bellman's equation, initial approaches to solving control problems with a general state space involved standard state-space discretisation for numerical implementation. In [78] there is a study of discretisation methods for POMDPs. In discretisation, the target, observation and control state-space is discretised to obtain a finite state POMDP. This was the approach adopted in [163].

Consider a state comprised of a continuous and discrete component, $(x, \vartheta) \in \mathbb{R}^d \times \Theta$. If we discretise each component of x to L values then, the discretised state-space has $L^d|\Theta|$ elements! Note that for example when considering the dynamics of manoeuvering targets, it appears that one must not lump the states in Θ together to yield a smaller discretised state-space as each mode $\vartheta \in \Theta$ corresponds to a manoeuver and hence possibly vastly different dynamics. Note also that the observation and control spaces need to be discretised as well. The problem is compounded even further by the action path constraints, such as when excessive maneuvering in when controlling a the motion of an object. In this case, the previous action must be augmented to the state descriptor to yield states $(x_n, \vartheta_n, A_{n-1})$, see [54].

Although the solution for a discrete state spaces can be obtained exactly, it is often too computationally intensive. It is well known that the controller (or policy) of a finite horizon finite controlled Hidden Markov Model is characterised by a finite set of vectors [151]. However, the number of vectors needed to characterise the optimal policy grows exponentially with the horizon and computing these vectors may be computationally infeasible even for very small prediction horizons, e.g. $H = 5$ or greater. In practise, various pruning methods are employed, such as in [110], which effectively amounts to another level of discretisation.

The optimisation problem cannot be solved exactly because a closed-form expression for the criterion is not available. Only an approximate solution is possible via discretisation (as described above) and Dynamic Programming. An iterative gradient method appears to be the only way to solve it exactly. Apart from this the satisfaction of any constraints can be ensured by the use of a projection of the computed gradient. In this thesis, we shall focus on using SMC to develop gradient algorithms to compute the optimal policy. Therefore, we shall address finite horizon control problems or infinite horizon ones, where policy gradient can be used.

3.7 Distributed General State Space Models

In many applications such as machine learning [82], statistical physics [171] or sensor networks [1], it seems an advantage to be able to distribute the computation required. One obvious reason for this can be the decentralised architecture of the problem, e.g. sensor fusion for sensor networks [123,128]. Also, sometimes if a problem dealing with large dimensions admits a centralised formulation, it might be very inefficient to use standard computational methods without exploiting the structure of interactions or dependencies between variables [81,86]. For example, in computer vision many images of a particular scene are obtained simuntaneously from a large number of cameras positioned at different locations and recording from different agles [59]. Clustering the images together and filtering using traditional centralised techniques might result in exhaustive amounts of computation [59]. Finally, in large scale systems, a decentralised approach might be convinient for performing model reduction by eliminating weak dependancies between different random variables [59,95,128].

Consider the case of using different HMMs in parallel for modelling a distributed process. Assuming we use i to index each separate HMM, where i belongs to some finite set $\mathcal{V}$. Let at time $n,$ the state of the i-th HMM be denoted as X_n^i and obey a Markov transition as follows:

$$X_{n+1}^i \big| \, X_n^i = x^i \sim f^i \left(\cdot \big| \, x^i \right).$$

Similarly denote the i-th HMM's observation of the hidden state as Y_n^v and assume it is generated as follows:

$$Y_n^v \big| \, X_n^v = x^v \sim g^v \left(\cdot \big| \, x^v \right).$$

In a completely decoupled setting, each separate HMM will maintain its own local filtering distribution $p(x_n^i | Y_{0:n}^i)$, where it will only process its own observations $Y_1^i, \ldots, Y_n^i$. In case we

have some information on how the different models' states interact, is possible to utilise all the observations from all the models to enhance the performance of filtering. This interaction between the states might be some form of statistical dependancy or some deterministic coupling. Assuming all the states are fully coupled together, a collaborative filtering approach might be possible and they can process all the received observations. Therefore the filtering distribution for each state x^i will become $p(x_n^i|Y_{0:n})$, where Y_n denotes the vector of stacked observations $[Y_n^v]_{v\in\mathcal{V}}$. It is clear that a fully coupled implementation is advantageous since hidden states with poor local observations can benefit from other nodes with better quality observations.

Graphical Models (GM) provide a natural framework for coupling together different simple models that interact together to form a more complex one. In the remainder of this section we shall introduce Graphical Models as a modelling framework and then see how they can be combined with HMMs to represent distributed general state space models.

3.7.1 Background on Graphical Models

The aim of this section is to introduce the basic concepts of Graphical models and Belief Propagation so that it is easier for the reader to understand the purpose of our work in the third part of this thesis. We do not intend to give a complete review of the topic and we shall borrow some of the presented material in this section from two main references [95, 130], as well the following review papers [82, 171]. We believe that a variety of examples are probably the best way to understand the ideas of this section and therefore we shall refer the interested reader to the previous references for further details and more examples.

Graphical models are a successful marriage of probability and graph theory. They can be viewed as a natural framework to deal with uncertainty and complexity in order to solve inference, decision or learning problems. Graphical models essentially utilise a systematic representation of the interactions between random variables, so as to build complex systems or representations from simpler parts, such as links of variables. The aim of this is to exploit the specific system structure in order to promote more efficient algorithms for inference, decision or learning.

We shall start by introducing some basic notation and concepts related to graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes, or vertices, $\mathcal{V} = \{v_i\}$, and a set of edges, $\mathcal{E} = \{(v_i, v_j)\}$, where $i \neq j$. We shall not consider the case where there are self edges or multiple edges between

a pair of vertices, and shall restrict our presentation to simple graphs, in which $(v_i, v_i) \notin \mathcal{E}$ and each edge in the set $\mathcal{E}$ is distinct[2].

The main types of graphs one can consider are *undirected* graphs and *directed* graphs. For an undirected graph, if $(v_i, v_j) \in \mathcal{E}$ then $(v_j, v_i) \in \mathcal{E}$. Nodes i and j in this case, are said to be neighbours. The set of neighbours of a vertex v_i is denoted $ne(i)$. For a directed graph, node i is said to be the parent of node j if $(v_i, v_j) \in \mathcal{E}$. The set of parents of node j is denoted as $pa(v_j)$. Conversely node j is the child of node i and the set of children of node i denoted as $ch(v_i)$. A *walk* is a sequence $v_1, ..., v_n$ of vertices within a graph such that $(v_{i-1}, v_i) \in \mathcal{E}$, where $i = 2, ..., n$. A *path* is a non-intersecting walk (i.e. the vertices within the sequence $v_1, ..., v_n$ are distinct). If one can reach every node in the graph from every other node through a path, the graph is said to be complete (i.e. all vertices are joined by an arrow or a line). A *cycle* is denoted to be a path with the same start and end nodes. If there are no cycles within the graph, then the graph is said to be acyclic.

A *clique* of a graph is a subset of a graph that is fully connected. Each node within this subset has an edge connecting it to all other nodes within the subset. We let $\mathcal{C}$ be denoted the set of all cliques of a graph. An important structure of graphs is that of a tree. A tree is a complete undirected graph that does not contain cycles. Moreover a rooted tree is the directed acyclic graph obtained from a tree by choosing a vertex as the root and directing all edges away from this root.

3.7.2 Graphical Models

Graphical models are a modelling tool to represent random variables according to an existing structure that can be posed by a graph. We assign a random variable $X_i \in \mathcal{X}_i$ to each vertex i of a graph and represent all the statistical dependencies by the edges of a specific graph. Let also $X_{\mathcal{V}}$ be the joint collection of all the variables X_i, for every $i \in \mathcal{V}$. Using the specific graph structure, the statistical properties of $X_{\mathcal{V}} \in \mathcal{X}_{\mathcal{V}}$ have been encoded in an intuitive way. We can now utilise the factorisation of the joint probability distribution $p(x_{\mathcal{V}})$ to solve any inference problem and develop efficient algorithms. So we can define a Graphical model as a collection of probability distributions that factorise according to the structure of an underlying graph.

[2]Note that in this chapter symbol $\mathcal{E}$ does not denote some σ-field but a set of edges instead. This notation will be used also in chapters 5 and 6

Directed Graphical Models

A directed Graphical model consists of a collection of probability distributions that factorise in the following way

$$p(x_{\mathcal{V}}) = \prod_{i \in \mathcal{V}} p(x_i | x_{pa(i)}).$$

This use of notation is consistent, given that the conditional distribution $p(x_i|x_{pa(i)})$ is well defined. If this is true then the joint or global distribution is also properly defined $p(x_{\mathcal{V}})$.

Informally, each directed edge is an arc or an arrow and can be attributed to an indication that X_i causes X_j, if node i is a parent of j. This causality can be more formally viewed as a conditional dependance between X_j and X_i. Each random variable of a node is independent of its ancestors given its parents. Directed Graphical models are also referred as *Bayesian networks*. As we shall mainly be adopting the use of undirected Graphical models, we refer the interested reader to [130] for more details and examples.

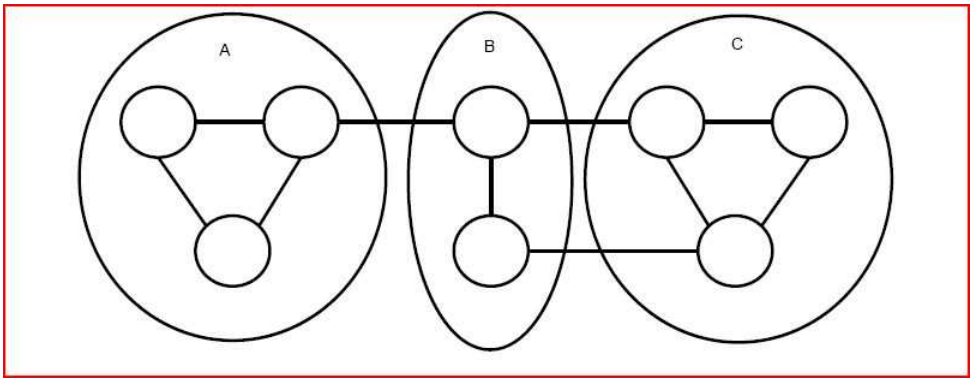


Figure 3.3: *Example of undirected graph, taken from [29].*

Undirected Graphical Models

First of all, we recall the definition of cliques as a subset of a graph that is fully connected. We proceed by defining for each clique C a compatibility function $\psi_C : \mathcal{X}_{\mathcal{C}} \rightarrow \mathbb{R}^+$, that depends only on the joint variable x_C, which is composed of all the variables of the nodes in clique C. Using this we define an undirected Graphical model as the collection of distributions that factorise in the following way,

$$p(x_{\mathcal{V}}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{3.40}$$

where the product is taken over all the cliques in the graph, $\mathcal{C}$ is the set of all cliques and Z is the normalisation constant. In contrast to the directed case the functions $\psi_C(x_C)$ need not have any obvious or direct relations to local marginal distribution. In general, we refer the term local to be used when we talk about local variables, marginal or conditional distributions at each node, and use the term global for the variables and distributions over the whole graph. Moreover, undirected Graphical models are also referred in the literature as *Markov random fields* [95].

Consider the undirected graph of Figure 3.3, which we have borrowed from [29, Ch.5]. The graph has been decomposed to three cliques, A, B, and C. Clique B separates, A from C, which can be stated as every path or walk in the graph from any element of A in order to reach any element of C has to pass through B. This is the graph theoretic concept of reachability and separation. This can be extended for each node in the graph, by simply stating that nodes i and j are separated by node k or clique C (given i, j are not members of C) if every path connecting i and j passes through node k or clique C respectively.

We shall aim to link this reachability concept with the algebraic concept of factorisation, so that this can be used effectively to examine conditional independence on Graphical models. More specifically, returning to the graph of Figure 3.3, we can stipulate that x_A is independent from x_C given x_B. This is the global Markov property and it can be shown that the set of distributions satisfying this assertion is exactly the set of distributions defined by (3.40). This global Markov property implies also the pairwise or local Markov property, which states that non adjacent variables x_i, x_j, where $(i, j) \notin \mathcal{E}$ and $i \neq j$, are independent given all the vertices in the graph excluding i, j. A similar result holds also for the case of directed graphs, only with a different notion of reachability. For more details see [95].

3.7.3 Exact Inference in Graphical Models

In this section we shall aim to show how can Graphical models be used to solve inference problems. Let $p(\cdot)$ a distribution defined by a Graphical model. In [166] we find some examples of inference problems, where Graphical models are used to compute the following cases:

- (a) the likelihood of data
- (b) the marginal distribution $p(x_A)$
- (c) the conditional distribution $p(x_A|x_C)$, where A, C are disjoint subsets of $\mathcal{V}$,

- (d) the mode of a density, such as $\arg\max_{x_A \in \mathcal{X}_A} p(x_A)$.

Cases (a) and (b) are essentially equivalent within the Bayesian framework, since it involves summing or integrating with respect to a subset of random variables. Case (c) is also connected (a) and (b), since it also involves a marginalisation step to get $p(x_A, x_C)$ and a further one for $p(x_C)$. Finally, case (d) is fundamentally different from the rest due to the maximisation involved, but it is possible to extend the methodology developed for cases (a)-(c) to deal with problems involving modes.

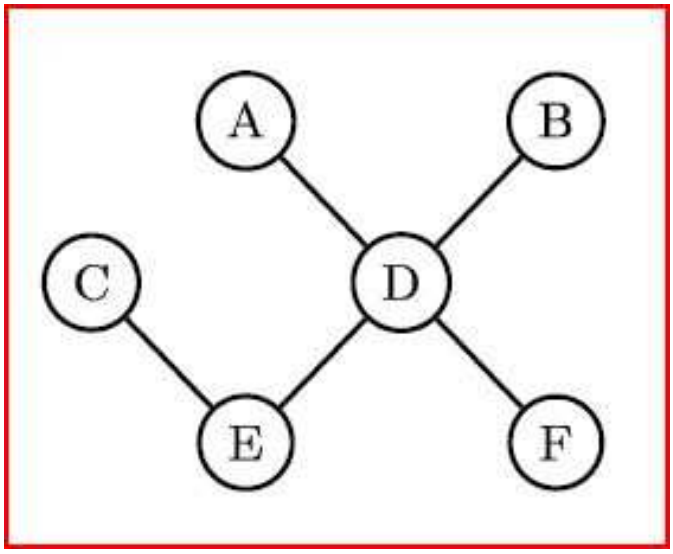


Figure 3.4: *Example of undirected graph with tree structure.*

When the graph admits a tree topology, one can compute the problems in cases (a)-(c) exactly. We will define a recursive message passing algorithm that scales linearly in the number of nodes to be used. For problems involving computing marginals it is usually referred to as the sum-product algorithm, whereas its extension for problems involving modes is called max-product algorithm. This method is also known as *Belief Propagation* (BP) [130].

Before presenting the message passing algorithm for Belief Propagation, we shall introduce an elimination principle for marginalisation based on a sequential ordering of the variables. This will lead to deriving the sum product algorithm. We shall present the elimination only for the undirected Graphical models case. Note that this is not restrictive, as in general any directed graph can be transformed to an equivalent undirected one via a transformation known as *moralisation*, which is described with more detail in [95, 171]. From now we shall be consid-

ering only undirected graphs, because these suit better the problems we shall be investigating later on.

Consider the case for the graph of Figure 3.4. The factorisation of the joint distribution will admit a pairwise Markov structure, and we can write

$$p(x_a, x_b, x_c, x_d, x_e, x_f) \propto \psi_{ad}(x_a, x_d)\psi_{bd}(x_b, x_d)\psi_{ce}(x_c, x_e)\psi_{de}(x_d, x_e)\psi_{df}(x_d, x_f).$$

Let us say we are interested in computing the marginal $p(x_b)$. We write $p(x_b)$ as follows

$$\begin{aligned} p(x_b) &= \int_{\mathcal{X}_F}\int_{\mathcal{X}_E}\int_{\mathcal{X}_D}\int_{\mathcal{X}_C}\int_{\mathcal{X}_A} p(x_a, x_b, x_c, x_d, x_e, x_f)dx_a dx_c dx_d dx_e dx_f, \\ &\propto \int_{\mathcal{X}_F}\int_{\mathcal{X}_E}\int_{\mathcal{X}_D}\int_{\mathcal{X}_C}\int_{\mathcal{X}_A} \psi_{ad}(x_a, x_d)\psi_{bd}(x_b, x_d)\psi_{ce}(x_c, x_e)\psi_{de}(x_d, x_e)\psi_{df}(x_d, x_f)dx_a dx_c dx_d dx_e dx_f, \end{aligned}$$

where $\mathcal{X}_A$ is the state space of x_a and similarly for the rest. We shall assume we can apply Fubini's theorem and are able to interchange the order of the integrals. To do this we assume $\int_{\mathcal{X}_E}\int_{\mathcal{X}_D} |\psi_{de}(x_d, x_e)|\, dx_d dx_e < \infty$ and similarly for the rest of the potentials. We reorder the sequence of elimination in order to distribute the computation along the graph and use local computations at each node to get the final marginal. So we write

$$\begin{aligned} p(x_b) &\propto \int_{\mathcal{X}_F}\int_{\mathcal{X}_E}\int_{\mathcal{X}_D}\int_{\mathcal{X}_C}\int_{\mathcal{X}_A} \psi_{ad}(x_a, x_d)\psi_{bd}(x_b, x_d)\psi_{ce}(x_c, x_e)\psi_{de}(x_d, x_e)\psi_{df}(x_d, x_f)dx_a dx_c dx_d dx_e dx_f \\ &= \int_{\mathcal{X}_D}\int_{\mathcal{X}_F}\int_{\mathcal{X}_A} \underbrace{\left(\int_{\mathcal{X}_E} \underbrace{\left(\int_{\mathcal{X}_C} \psi_{ce}(x_c, x_e)dx_c \right)}_{\rho_{ce}(x_e)} \psi_{de}(x_d, x_e)dx_e \right)}_{\rho_{ed}(x_d)} \psi_{ad}(x_a, x_d)\psi_{bd}(x_b, x_d)\psi_{df}(x_d, x_f) \\ &\quad \times dx_a dx_f dx_d \end{aligned} \tag{3.41}$$

It is already apparent that we start from the most remote node and eliminate the variables in order. The order of elimination can be seen from the order of the integrals moving from inwards to outwards. This sequence has been derived according to the path from each node to node b. In equation (3.41) we identify $\rho_{ce}(x_e)$ and $\rho_{ed}(x_d)$. The notation is chosen in such a way so that the subscript and the argument of function ρ reveal the sequence of the elimination, i.e. at node e we eliminate variable x_c along the edge (c, e). The remaining function is local to node e and is depending only on x_e. We choose this notation deliberately so that later on it is more apparent

how these functions can play the role of messages between neighbouring nodes. Moving on with the elimination, we have that

$$
\begin{aligned}
p(x_b) &\propto \int_{\mathcal{X}_D} \underbrace{\left(\int_{\mathcal{X}_F} \underbrace{\left(\int_{\mathcal{X}_A} \rho_{ed}(x_d)\psi_{ad}(x_a, x_d)dx_a \right)}_{\rho_{ad}(x_d)} \psi_{df}(x_d, x_f)dx_f \right)}_{\rho_{fd}(x_d)} \psi_{bd}(x_b, x_d)dx_d \\
&= \int_{\mathcal{X}_D} \rho_{fd}(x_d)\psi_{bd}(x_b, x_d)dx_d \\
&= \rho_{db}(x_b)
\end{aligned}
$$

Note that eliminating x_a and then x_f is equivalent with performing the elimination with the opposite order. This can be attributed to the symmetry of the graph along the walk from c to b. In general determining the order of integration for an arbitrary graph, in order to optimally exploit the redundancy in the structure and minimise computations is an NP-hard problem. Fortunately for tree graphs using the same principles as in elimination, we can compute in parallel each marginal density at each node, using the Belief Propagation (BP) algorithm [130].

3.7.4 Belief Propagation

Consider an undirected Graphical model defined by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and admitting a tree structure. We restrict ourselves to pairwise Markov potentials $\psi_{ij}(x_i, x_j)$, where $(i, j) \in \mathcal{E}$ and we shall assume integrability given by

$$
\int_{\mathcal{X}_i} |\psi_{ij}(x_i, x_j)|\, dx_i < \infty, \int_{\mathcal{X}_j} |\psi_{ij}(x_i, x_j)|\, dx_j < \infty,
$$

where $x_i \in \mathcal{X}_i$ for each $i \in \mathcal{V}$. These pairwise potentials can be combined with a local potential $\phi_i(x_i)$ at each node $i \in \mathcal{V}$. Then equation (3.40) can be equivalently written as

$$
p(x_{\mathcal{V}}) \propto \prod_{i \in \mathcal{V}} \phi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j).
$$

Note that this formulation is general for undirected graphs and is not just intended for graphs with tree topology. From now on we shall use this equation to describe every undirected Graphical model.

Pearl in [130] was one of the first to use a message passing approach so that each node i can compute locally its marginal $p(x_i)$ using messages from its neighbours and local variables. This effectively parallelises the the elimination procedure described above and distributes the computation along the nodes of the graph. The messages along the graph can be defined by defining each message m^{ij} from node i to node j, $(i,j) \in \mathcal{E}$, as

$$m^{ij}(x_j) = \int_{\mathcal{X}_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{p \in \text{ne}(i)\setminus\{j\}} m^{pi}(x_i)dx_i, \tag{3.42}$$

where $ne(i) \setminus \{j\}$ denotes the set of neighbours of i except j. Similarly to $m^{ij}(x_j)$, $m^{pi}(x_i)$ is the message from node p to node i. For a tree it can be informally said that the messages are initiated at the outer leafs and are propagated towards the root of the tree, thus moving "upwards", and once they reach the root of the tree then are propagated "downwards" or backward so that they terminate again at the leaves.

Once all messages are received at each node i, it can compute its marginal by

$$p(x_i) \propto \phi_i(x_i) \prod_{k \in \text{ne}(i)} m^{ki}(x_i) \tag{3.43}$$

This procedure leads to exact computations of marginal distributions on tree Graphical Models. In the case when the graph contains loops, the same algorithm can be applied, only this time it will be an approximation. This is commonly known as *Loopy Belief Propagation* (LBP).

3.7.4.1 Loopy Belief Propagation

Although it is uncertain in general whether Loopy Belief Propagation (LBP) will lead to good approximations, it has been studied in many examples. A good survey with empirical results can be found in [119]. In some cases, such as Gaussian Markov fields, it can lead to very good approximations [167, 168]. LBP is essentially a recursive implementation of BP in graphs with loops. After initiallising all the messages, we iterate the marginal distribution as

$$p_n(x_i) \propto \phi_i(x_i) \prod_{k \in \text{ne}(i)} m_n^{ki}(x_i),$$

where the superscript n denotes the iteration number n. The messages from node i to node j, $(i,j) \in \mathcal{E}$, are iterated as follows:

$$\begin{aligned} m_n^{ij}(x_j) &\propto \int_{\mathcal{X}_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{p \in \mathsf{ne}(i)\setminus\{j\}} m_{n-1}^{pi}(x_i)dx_i \\ &= \int_{\mathcal{X}_i} \psi_{ij}(x_i, x_j) \frac{p_{n-1}(x_i)}{m_{n-1}^{ji}(x_i)} dx_i. \end{aligned}$$

Unfortunately, as it was mentioned earlier this method is not always guaranteed to converge to fixed point solutions. For more details see [158]. Finally in the variational inference literature, where Graphical models are used extensively, alternatives of LBP can be found, when dealing with loops. These minimise the Bethe free energy, a concept arising from statistical physics. For more details see [171].

3.7.5 Distributed State Space Models using Graphical Models

Consider an simple undirected tree $(\mathcal{V}, \mathcal{E})$, where for every $i \in \mathcal{V}$ we have the following HMM

$$\begin{aligned} X_n^i | X_{n-1}^i = x_{n-1}^i &\sim f_{\theta^i}^i(\,.\,|x_{n-1}^i), \\ Y_n^i | X_n^i = x_n^i &\sim g_{\theta^i}^i(\,.\,|x_n^i). \end{aligned}$$

This representation is not yet very useful for realistic problems in a distributed setting. For example, in problems involving sensor networks, one might have to account for uncertain communication between nodes, cross sensor interference between readings, and power loss due to intense communications. These issues indicate that our modelling should account for a interaction between x^i and x^j. The distributed state space model has to be cast as a dynamic undirected Graphical model. To address this we introduce time varying pairwise integrable Markov potentials $\psi_{\vartheta^e}^e(x_i, x_j)$, where $e \in \mathcal{E}$. For example, the potential can represent the probability of node i managing to pass a message successfully to node j, or the probability that node i detects node j as its neighbour. We alert the reader that in this section the notation is slightly different than in the previous one. Node and edge indeces are now moved from subscripts to superscripts and in the subscripts we denote time or parameter dependance. Note for example, that for each edge e, the potential's dependence on a static parameter ϑ^e is incorporated by adding the parameter in the subscript. Let also ϑ, θ and $x_n^{\mathcal{V}}$ denote the joint vectors of stacked

parameters $[\vartheta^e]_{e\in\mathcal{E}}$, $[\theta^i]_{e\in\mathcal{V}}$ and $[x_n^i]_{i\in\mathcal{V}}$ respectively. Then[3],

$$p(x_n^{\mathcal{V}}|Y_n) \propto \prod_{i\in\mathcal{V}} \phi_n^i(x_n^i) \prod_{e\in\mathcal{E}} \psi_{\vartheta^e}^e(x_n^i, x_n^j),$$

where we define the local potential ϕ_n^i as the product of the likelihood and the prediction density

$$\phi_n^i(x_n^i) = g_{\theta^i}^i(Y_n^i|x_n^i)p(x_n^i|Y_{1:n-1}).$$

The underlying joint process $\{X_n^{\mathcal{V}}\}_{n\geq 0}$ admits a transition density given by

$$f(x_n^{\mathcal{V}}|x_{n-1}^{\mathcal{V}}) = \prod_{i\in\mathcal{V}} f_{\theta^i}^i(x_n^i|x_{n-1}^i),$$

The transition step is independent for each node. Thus, assuming $p(x_{n-1}^i|Y_{1:n-1})$ is available to node i at time n, the propagation of the prediction density at each node $p(x_n^i|Y_{1:n-1})$ can be computed locally at node i and independently of other nodes as given by (3.6). Furthermore, each node can obtain its filtering density $p(x_n^i|Y_{1:n})$ by marginalisation as

$$p(x_n^i|Y_{1:n}) = \int p(x_n^{\mathcal{V}}|Y_{1:n}) dx_n^{\mathcal{V}\setminus i},$$

where $x_n^{\mathcal{V}\setminus i}$ denotes the joint vector containing every x_n^i, for all $i \in \mathcal{V}$ except i . Hence, the filtering update can be implemented in a distributed fashion using Belief Propagation and message passing as in equations (3.42)-(3.43). In this case, we have

$$m_n^{ij}(x_n^j) = \int \psi_{\vartheta^{ij}}^{ij}(x_n^i, x_n^j)\phi_n^i(x_n^i) \prod_{p\in \text{ne}(i)\setminus\{j\}} m_n^{pi}(x_n^i) dx_n^i, \tag{3.44}$$

$$p(x_n^i|Y_{1:n}) \propto \phi_n^i(x_n^i) \prod_{k\in\text{ne}(i)} m_n^{ki}(x_n^i). \tag{3.45}$$

This sequential filtering approach of using a Graphical model and Belief Propagation has already appeared in [29, Ch.5]. Similarly to standard Graphical models, performing sequential Bayesian inference for the state will be exact for tree graphs only. In [29, Ch.5] the author proposed to use LBP to deal with graphs including loops. An alternative approach would be to incorporate the ideas from [59,95,128] and generalise this formulation for junction trees. Using such an approach each node i in our formulation would represent a clique instead of a simple

[3]Note that this factorisation allows the graph structure to be time evolving and be denoted as $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$. Similarly the potential can be also time varying and denoted as $\psi_{n,\vartheta^e}^e(x_i, x_j)$, although we shall not consider this any further.

node, $\mathcal{V}$ would be replaced by $\mathcal{C}$, and $\mathcal{E}$ would refer to the edges of the junction tree. Otherwise the present formulation remains unchanged and each clique can now naturally include loops. It is clear that this formulation allows both the use of LBP and junction tree filtering. What has not been investigated though is the possibility of performing parameter estimation on such models.

3.7.6 Distributed Parameter Estimation using RML

So far it should be clear how Graphical models and Belief Propagation can be used to perform collaborative filtering in general state space models. What we have not addressed yet, is the problem of inferring the static parameters (θ, ϑ). As far as the local parameters in θ are concerned, each θ^i can be estimated locally using filtering (e.g. for the RML) or smoothing (e.g. for EM) given the local observations $Y^i_{1:n}$. Any ML method presented so far in this chapter is suited. To avoid confusion and simplify the exposition in the remainder of this chapter the subscripts of θ^i will be dropped. As regard to ϑ, we will now briefly show how RML can be used for a distributed implementation, as this will be the main focus in the third part of this thesis. The motivation for emphasising on RML is that it allows performing parameter estimation both on line and in parallel to collaborative filtering.

We shall use Graphical models and extend Belief Propagation so that a distributed RML recursion can be derived. Let $e = (r, q)$ be the edge of interest and r be the node at which

$$\vartheta^{e^*} = \arg\max_{\vartheta^e} l^r(\vartheta^e)$$

is estimated, where $l^r(\vartheta^e) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} \log p_{\vartheta^e}(Y_k | Y_{0:k-1})$. The superscript r in the average log-likelihood and the subscript ϑ^e in the recursive likelihood are there to emphasise that we are interested on estimating ϑ^e at node r.

When using RML we are interested in expanding $\nabla_{\theta^e} \log p(Y_n|Y_{1:n-1})$ as follows:

$$\nabla_{\vartheta^e} \log p(Y_n|Y_{1:n-1}) = \nabla_{\vartheta^e} \log \left(\int \phi^r_n(x^r_n) \prod_{k\in \text{ne}(r)} m^{kr}_n(x^r_n) dx^r_n \right). \tag{3.46}$$

Note only $\phi^r_n(x^r_n)$, m^{qr}_n and $\psi^e_{\vartheta^e}$ are a function of ϑ^e. The dependance of $\phi^r_n(x^r_n)$ on ϑ^e comes

from using message passing at time $n-1$ to obtain the filtering density. This would lead to

$$\begin{aligned}&\nabla_{\vartheta^e}\log(p(Y_n|Y_{1:n-1}))\\&=\left(\int\phi_n^r(x_n^r)\prod_{k\in\text{ne}(r)}m_n^{kr}(x_n^r)dx_n^r\right)^{-1}\left(\begin{array}{c}\int\nabla_{\vartheta^e}m_n^{qr}(x_n^r)\phi_n^r(x_n^r)\prod\limits_{k\in\text{ne}(r)\setminus q}m_n^{kr}(x_n^r)dx_n^r\\+\int\nabla_{\vartheta^e}\phi_n^r(x_n^r)\prod\limits_{k\in\text{ne}(r)}m_n^{kr}(x_n^r)dx_n^r\end{array}\right)\end{aligned}\tag{3.47}$$

Propagating recursively $\nabla_{\vartheta^e}\phi_n^r(x_n^r)$ in the spirit of (3.28)-(3.31) is a straightforward task. Assuming that at time n we have $\nabla_{\vartheta^e}p(x_n^r|Y_{1:n-1})$ available from the previous iteration at time $n-1$, consider the following recursion for the gradients:

$$\nabla_{\vartheta^e}p(x_n^r|Y_{1:n-1})=\int f^r(x_n^r|x_{n-1}^r)\nabla_{\vartheta^e}p(x_{n-1}^r|Y_{1:n-1})dx_{n-1}^r,\tag{3.48}$$

$$\nabla_{\vartheta^e}\phi_n^r(x_n^r)=g^r(Y_n^r|x_n^r)\nabla_{\vartheta^e}p(x_n^r|Y_{1:n-1}),\tag{3.49}$$

$$\nabla_{\vartheta^e}p(x_n^r|Y_{1:n})=p(x_n^r|Y_{1:n})\left(\begin{array}{c}\frac{\nabla_{\vartheta^e}\phi_n^r(x_n^r)}{\phi_n^r(x_n^r)}+\frac{\nabla_{\vartheta^e}m_n^{qr}(x_n^r)}{m_n^{qr}(x_n^r)}\\-\int\left(\frac{\nabla_{\vartheta^e}\phi_n^r(x_n^r)}{\phi_n^r(x_n^r)}+\frac{\nabla_{\vartheta^e}m_n^{qr}(x_n^r)}{m_n^{qr}(x_n^r)}\right)p(x_n^r|Y_{1:n})dx_n^r\end{array}\right).\tag{3.50}$$

Such a recursion can be implemented in parallel to the filtering recursion for each node given by (3.6) and (3.45).

An essential requirement that is overlooked so far is that $\nabla_{\vartheta^e}m_n^{qr}(x_n^r)$ has to be available at node r in addition to $m_n^{qr}(x_n^r)$, in order to compute (3.47) and (3.50). This can be made possible by defining an additional message passing algorithm than the one in (3.44), which aims to propagate and compute the gradients of the messages at each node in the graph. For every $(i,j)\in\mathcal{E}$ this can be given by

$$\nabla_{\vartheta^{ij}}m_n^{ij}(x_n^j)=\int\nabla_{\vartheta^{ij}}\psi^{ij}(x_n^i,x_n^j)\phi_n^i(x_n^i)\prod_{p\in\text{ne}(i)\setminus\{j\}}m_n^{pi}(x_n^i)dx_n^i.\tag{3.51}$$

In the resulting message passing algorithm for RML we have added an additional message in parallel to the standard BP messages of (3.44). The messages in (3.44) and (3.51) are scalable with the size of the network and also ensure that the particular choice of r and q is arbitrary given (r,q) is an edge. Therefore, every node of the graph can estimate the components of ϑ related with its edges. For any choice of $e=(r,q)\in\mathcal{E}$, the following RML parameter update can be used at node r :

$$\vartheta_{n+1}^e=\vartheta_n^e+\gamma_{n+1}^r\nabla_{\vartheta^e}\log p(Y_n|Y_{1:n-1})|_{\vartheta^e=\vartheta_n^e},$$

where γ_{n+1}^r is a RML step size. This update be done either in parallel for every node or in a cyclic manner. This concept of gradient based parameter estimation on Graphical models and generelising the BP algorithm to include a message used to compute gradients is to the best of our knowledge novel and is one of the contributions of Chapters 6, 7 and 8.

It is clear that in general all the integrals in this section may not have closed form expressions, which means we have to rely on approximations. Developing methods using appropriate approximations will be a topic discussed in more detail in the next chapters of this thesis. For RML the SMC approximations appeared in [133] can be extended for distributed state space models defined by Graphical models. This will be presented in Chapters 7 and 8.

Part II

Optimal Control using Policy Gradient and Sequential Monte Carlo

4

Simulation-Based Optimal Sensor Scheduling with Application to Observer Trajectory Planning

Summary. *Sensor scheduling has been a topic of interest to the target tracking community for some years now. Recently, research into it has enjoyed fresh impetus with the current importance and popularity of applications in Sensor Networks and Robotics. The sensor scheduling problem can be formulated as a controlled Hidden Markov Model. In this chapter, we address precisely this problem and consider the case in which the state, observation and action spaces are continuous. This general case is important as it is the natural framework for many applications. In sensor scheduling, our aim is to minimise the variance of the estimation error of the hidden state with respect to the action sequence. We present a novel simulation-based method that uses a stochastic gradient algorithm to find optimal actions. This is in contrast to existing works in the literature that only solve approximations to the original problem.*

4.1 Introduction

Consider the following continuous state Hidden Markov Model (HMM),

$$X_{n+1} = f(X_n, A_{n+1}, W_n), \quad Y_n = g(X_n, A_n, V_n), \tag{4.1}$$

where $X_n \in \mathbb{R}^{d_x}$ is the hidden system state, $Y_n \in \mathbb{R}^{d_y}$ the observation of the state, and W_n and V_n are i.i.d. noise terms.[1] Unlike the classical HMM model, the evolution of the state and observation processes depends on an input parameter $A_n \in \mathbb{R}^{d_a}$, which is the control or action. In HMM models, one is primarily concerned with the problem of estimating the hidden state, which is achieved by propagating the posterior distribution (or filtering density) $\pi_n(x)dx = \mathbf{P}(X_n \in dx | A_{1:n}, Y_{1:n})$. By a judicious choice of action sequence $\{A_n\}$, the evolution of the state and observation processes can be 'steered' in order to yield filtering densities that give more accurate estimates of the state process. This problem is also known in the literature as the sensor scheduling problem.[2]

Sensor scheduling has been a topic of interest to the target tracking community for the some years now [55,76,77,84,109,112,148,163]. The classical setting is the problem of tracking a maneuvering target over N epochs. Here X_n denotes the state of the target at epoch n, Y_n the observation provided by the sensor, and A_n some parameter of the sensor that may be adjusted to improve the "quality" of the observation. For example, consider a non-moving platform with a finite number of sensors, where each has different characteristics. In this case A_n denotes the choice of sensor to be used at epoch n [55, 96, 149]. Alternatively, there may be only one sensor and A_n could denote some tunable parameter of the sensor, as in the waveform selection problem [84], or in the case of directing an electronically scanned aperture (ESA) radar [25]. In contrast, consider the scenario in which a moving platform (or observer) is to be adaptively maneuvered to optimise the tracking performance of a maneuvering target. In this setting, A_n denotes the position of the observer at epoch n and the problem is termed the optimal *observer trajectory planning* (OTP) problem [76, 77, 109, 163]. In all these sensor scheduling problems, a measure of tracking performance is the mean squared tracking error over the N epochs,

$$\mathbf{E}\left\{\sum_{n=1}^{N} \left(\psi(X_n) - \langle \pi_n, \psi \rangle\right)^2\right\}, \tag{4.2}$$

[1] A more general model is described in the problem formulation in Section 4.2.

[2] Henceforth, we refer to any controlled HMM where the aim is to optimise estimation accuracy generically as a sensor scheduling problem.

where $\psi : \mathbb{R}^{d_x} \to \mathbb{R}$ is a suitable *test* function that emphasises the component (or components) of interest of the state vector we wish to track. The aim is to minimise (4.2) with respect to the choice of actions $\{A_1, \ldots, A_N\}$.

The current importance and popularity of applications in Sensor Networks and Robotics has given fresh impetus to sensor scheduling. Distributed tracking in ad hoc sensor networks employs sensor scheduling to determine which sensors should collaborate in the tracking exercise [106, 172]. Additionally, the problem of terrain-aided navigation and path planning for localization and mapping can be formulated as sensor scheduling problems [127].

When the dynamics of the state and the observation processes are both linear and Gaussian then, the optimal solution to the sensor scheduling problem (4.2) (when ψ gives a quadratic cost) can be computed off-line [112]; this is not surprising given that the Kalman filter covariance is also independent of the actual realisation of observations. In the general setting studied in this chapter, the dynamics can be both non-linear and non-Gaussian, which means that the filtering density π_n, and integration with respect to it, cannot be evaluated in closed-form. Hence, the tracking error performance criterion itself does not admit a closed-form expression. To further complicate matters, the actions sought are continuous valued, i.e., vectors in $\mathbb{R}^{d_a}$.

To address the complications to do with the non-linear and non-Gaussian dynamics, one could linearise the state and observation model (as in [109]), i.e. using the Extended Kalman Filter to propagate the filtering density π_n. However, dealing with the tracking error performance criterion directly is the exception rather than the rule. The majority of works [76, 77, 127, 163] (and references therein), while aiming to minimise mean squared tracking error, do so indirectly by defining a lower bound to the tracking error criterion and minimising the lower bound instead. The bound in question is the Posterior Cramer-Rao Lower Bound (PCRLB), which is the inverse of the Fisher Information Matrix (FIM). This approach hinges on the ability to propagate recursively the FIM in closed form by a Ricatti-type equation for the non-linear and non-Gaussian filtering problem. Unfortunately, the recursion for the FIM involves evaluating the expectation of certain derivatives of the transition probability density of the state dynamics, as well as the expectation of certain derivatives of the observation likelihood (see (4.3) and (4.4) below). As these quantities cannot be evaluated in general except for the linear and Gaussian case, this assumption is either invoked or the authors resort to simulation-based approximations. In addition, the PCRLB bound is not always tight [19].

As for the complications due to continuous valued actions, the approach in the literature is to discretise $\mathbb{R}^{d_a}$ to a grid. There have also been studies where the continuous state HMM (4.1) is approximated by a discrete state HMM, and the latter solved using Dynamic Programming [163].

The aim of this chapter is to solve the sensor scheduling problem with continuous action space directly, and not a surrogate problem defined through the PCRLB or otherwise. We make no assumptions of linearity or Gaussianity for analytic convenience, nor do we discretise the state, observation, or action space. We avoid these restrictive modelling assumptions on the continuous state HMM by recourse to methods based on computer simulation (simulation for short). As the action policy derived will be a function of the filtering density, we will employ simulation to approximate the posterior density by a finite sum of weighted point-mass distributions [47]. The main advantage of simulation over other numerical integration methods is that it is typically very easy to implement. Furthermore, it follows the Strong Law of Large Numbers [34] and there is much literature on its efficient implementation for approximating posterior densities [47].

In order to solve for the optimal sequence of continuous valued actions, we will use an iterative stochastic gradient algorithm. We derive the gradient of the performance criterion with respect to the action trajectory and demonstrate how low variance estimates of it may be obtained using *control variate* [68] techniques. One major advantage of a stochastic gradient based method is that theoretical guarantees are easily obtained. Under suitable regularity assumptions, one can guarantee convergence to a local optimum of the performance criterion, while it is difficult to make similar assertions about the quality of the solutions obtained by other approximate methods proposed in the literature for sensor scheduling.

As an instance of the sensor scheduling problem, we study the observer trajectory planning problem for a bearings-only application. We state theoretical results concerning the convergence of the observer trajectory identified by our simulation-based algorithm. Handling multiple observers simultaneously is easy in our proposed framework, and numerical results are presented for cooperating observers tracking a maneuvering target.

The organisation of this chapter is as follows. In Section 4.2, we formulate the optimal sensor scheduling problem. We also summarise some key points concerning several methods in the literature that may be used to solve this problem. In Section 4.3, we derive the gradient of

the performance criterion being optimised, and detail the use of simulation and variance reduction techniques for its estimation. In Section 4.4.2, we present the main algorithm of the chapter, which is a two time-scale stochastic gradient algorithm for solving the sensor scheduling problem. General convergence results for this algorithm are presented in [148]. In Section 4.5, we formulate the observer trajectory planning problem as an instance of the sensor scheduling problem and apply the convergence results to this application. Numerical examples are presented in Section 4.6, and concluding remarks are presented in Section 4.7.

Notation: The notation that is used in the chapter is now outlined. The norm of a scalar, vector or matrix is denoted by $|\cdot|$. For a vector b, $|b|$ denotes the vector 2-norm $\sqrt{\sum_i |b(i)|^2}$. For a matrix A, $|A|$ denotes the matrix 2-norm, $\max_{b:|b|\neq 0} \frac{|Ab|}{|b|}$. For convenience, we also denote a vector $b \in \mathbb{R}^n$ by $b = [b(i)]_{i=1,\ldots,n}$, or the i-th component of a vector by $[b]_i$. For scalars $a_{j,i}$, $j = 1,\ldots,m$, $i = 1,\ldots,n$, let $\left[[a_{j,i}]_{j=1,\ldots,m}\right]_{i=1,\ldots,n}$ denote the stacked vector $[a_{1,1},\ldots,a_{m,1},\ldots,a_{1,n},\ldots,a_{m,n}]^T$. For a vector b, let diag(b) denote the diagonal matrix formed from b.

For a function $f : \mathbb{R}^n \to \mathbb{R}$ with arguments $z \in \mathbb{R}^n$, we denote $(\partial f/\partial z(i))(z)$ by $\nabla_{z(i)} f(z)$ and $\nabla f(z) = [\nabla_{z(1)} f(z), \cdots, \nabla_{z(n)} f(z)]^T$. For the vector valued function $F = [F_1, \cdots, F_n]^T : \mathbb{R}^n \to \mathbb{R}^n$, let ∇F denote the matrix $[\nabla F_1, \cdots, \nabla F_n]$. For real-valued integrable functions f and g, let $\langle f, g\rangle$ denote $\int f(x)g(x)dx$.

4.2 Problem Formulation

At time n, let X_n and Y_n be random vectors that model the d_x-dimensional state and its d_y-dimensional observation respectively. Suppose that an action $A_n \in \mathbb{R}^{d_a}$ is applied at time n. The state $\{X_n\}_{n\geq 0}$ is an unobserved Markov process with initial distribution and transition law given by

$$X_0 \sim \pi_0, \quad X_{n+1} \sim p\left(\cdot|X_n, A_{n+1}\right). \tag{4.3}$$

The observation process $\{Y_n\}_{n\geq 1}$ is generated according to the state and action dependent probability density

$$Y_n \sim q\left(\cdot|X_n, A_n\right). \tag{4.4}$$

Given the sequence of actions $a_{1:n} := \{a_1, ..., a_n\}$ and measurements $y_{1:n} := \{y_1, ..., y_n\}$, the *filtering density* at time n is denoted by π_n, (or $\pi_n^{(y_{1:n}, a_{1:n})}$ to emphasise the dependence on $y_{1:n}$

and $a_{1:n}$,) and satisfies the *Bayes* recursion

$$\pi_n(x) = \frac{q\left(y_n|x,a_n\right)\int p(x|x',a_n)\pi_{n-1}(x')dx'}{\int\int q\left(y_n|x,a_n\right)p(x|x',a_n)\pi_{n-1}(x')dx'dx}. \tag{4.5}$$

Consider a suitable *test* function $\psi : \mathbb{R}^{d_x} \to \mathbb{R}$ where, for example, ψ could pick out a component of interest of the state vector we wish to estimate. The optimal sensor scheduling problem is to solve

$$\min_{A_{1:N}\in\Theta_A} J(A_{1:N}) = \mathbf{E}_{(\pi_0,A_{1:N})}\left\{\sum_{n=1}^{N}\lambda^{N-n}\left(\psi(X_n) - \langle\pi_n,\psi\rangle\right)^2\right\}, \tag{4.6}$$

where $\mathbf{E}_{(\pi_0,A_{1:N})}$ denotes expectation with respect to the random variables $(X_{0:N}, Y_{1:N})$ which are distributed according to the law specified by $(\pi_0, A_{1:N})$, i.e., for any $1 \leq n \leq N$ and integrable function $h : \left(\mathbb{R}^{d_x}\right)^n \times \left(\mathbb{R}^{d_a}\right)^n \times \left(\mathbb{R}^{d_y}\right)^n \to \mathbb{R}$,

$$\begin{aligned}\mathbf{E}_{(\pi_0,A_{1:n})}\left\{h(X_{1:n},A_{1:n},Y_{1:n})\right\} :=& \\ \int h(x_{1:n},A_{1:n},y_{1:n}) & \underbrace{\Pi_{i=1}^{n} q\left(y_i|x_i,A_i\right)p\left(x_i|x_{i-1},A_i\right)\pi_0(x_0)dx_{0:n}dy_{1:n}}_{\mathbf{P}_{(\pi_0,A_{1:N})}}.\end{aligned} \tag{4.7}$$

The set of actions $\Theta_A \subset \left(\mathbb{R}^{d_a}\right)^N$ is open and $\lambda \in [0,1]$ is a discount factor to favour better tracking performance in the later epochs if so desired.

Feedback control: The sensor scheduling problem stated in (4.6) is an open-loop stochastic control problem. In order to utilise feedback in a closed-loop implementation, we will use the *open-loop feedback control* (OLFC) approach, which is described as follows (see [20] for a detailed account on open-loop feedback and other closed loop policies). Let $A^*_{1:N}$ be the solution to (4.6). The action applied at epoch 1 is then A^*_1, for which an observation Y_1 is received and the filtering density is updated to π_1. At epoch n, $1 < n \leq N$, let π_{n-1} be the filtering density corresponding to all actions taken and observations received up to (and including) epoch $n-1$, $\left\{A^*_1, Y_1, \ldots, A^*_{n-1}, Y_{n-1}\right\}$. Now solve problem (4.6) for the initial target distribution π_{n-1} and horizon $N-n$, and let the solution (with an abuse of notation) be denoted by $A^*_{n:N}$. The action for epoch n is just A^*_n. This procedure is repeated until epoch N.

Path constraints: For trajectory planning application, the sequence of actions $A^*_{1:N}$ determined by solving (4.6) above may not be realisable due to motion constraints on the observer. For example, the sequence of possible accelerations in a linear motion model (see Section 4.5) may not be able to realise the desired sequence of positions $A^*_{1:N}$. Alternatively, we may seek

a sequence of positions that belong to some parametric class, e.g., an observer doing a constant velocity turn where the turn rate is to be determined. In some cases, we may summarise observer motion constraints through a bounded mapping

$$F : \left(\mathbb{R}^{d_u}\right)^N \to \left(\mathbb{R}^{d_a}\right)^N, \tag{4.8}$$

where the actions $A_{1:N} = F(U_1, \ldots, U_N)$. For example, $U_{1:N}$ could describe the sequence of accelerations of an observer; see Section 4.5 for more details. We would then solve problem (4.6) subject to the equality constraint

$$A_{1:N} = F(U_{1:N}), \qquad U_{1:N} \in \left(\mathbb{R}^{d_u}\right)^N, \tag{4.9}$$

i.e. Θ_A now corresponds to the range of the function F.

Simulation and gradient based methods: We do not have a closed-form expression for J because the filtering density π_n and integration with respect to it cannot be evaluated in closed-form in our general setting. To evaluate $J(A_{1:N})$, we could revert to state-space discretisation (see [78] for issues on discretising general state space HMMs). One could discretise $\mathbb{R}^{d_x}$, $\mathbb{R}^{d_y}$ and derive the corresponding state evolution and observation laws, i.e. (4.3) and (4.4), for the approximating discrete problem. We may then calculate the approximation to $J(A_{1:N})$ for any choice of actions. This approach has its drawbacks though. Firstly, assuming that $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$ are discretised to finite sets of cardinality L_x and L_y respectively then, the multiple integral in (4.6) (cf. (4.7)) is converted to a sum over $(L_y)^N \times (L_x)^{N+1}$ terms, which is computationally prohibitive. Thus, we would be limited to a coarse discretisation and a small horizon N at best. Secondly, it is not obvious how to choose the grid in $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$ since, for accuracy of the approximation, the grid should be finer in the regions where density in (4.7) has more mass. In [163], the HMM is discretised and a closed-loop formulation of problem (4.6) is solved. A closed-loop formulation of (4.6) is known as a Partially Observed Markov Decision Process (POMDP). However, solving a POMDP exactly is computationally very demanding, specifically PSPACE-hard, and various approximation schemes that trade-off accuracy and speed have been devised [74].

We propose to use simulation with Stochastic Approximation (SA) [17, 38, 39] to minimise $J(A_{1:N})$ when Θ_A is an open (i.e. continuous) set without resorting to discretising $\mathbb{R}^{d_x}$, $\mathbb{R}^{d_y}$ or Θ_A. SA is a recursive Robbins-Monro algorithm that can be used to find local minima or maxima of functions, which could not be computed otherwise [93]. We will use SA to perform a stochastic gradient descent algorithm that only requires noisy estimates of the cost function

gradient, i.e.,

$$A_{1:N,k+1} = A_{1:N,k} - \alpha_k \left(\nabla J(A_{1:N})|_{A_{1:N}=A_{1:N,k}} + \text{noise} \right), \tag{4.10}$$

where $k \geq 0$ and $\nabla J(A_{1:N})$ denotes the gradient of J w.r.t. $A_{1:N}$. The step-size α_k is a non-increasing positive sequence tending to zero. In Section 4.3, we derive the gradient ∇J. Once again, we do not have a closed-form expression for ∇J for the same reasons as in J; the filtering density π_n and integration with respect to it cannot be evaluated in closed-form in our general setting. We will show instead how one may obtain an estimate of ∇J, namely $\widehat{\nabla J}$. The noise in (4.10) arises precisely because we use $\widehat{\nabla J}$ instead of ∇J. The smaller the variance of the noise is, the more quickly (4.10) converges to a minimising action sequence. This motivates us to consider techniques such as *control variates* [68] to reduce the noise variance. We propose an *adaptive* control variate method to reduce the variance of $\widehat{\nabla J}$ by coupling with (4.10) a second SA iteration that estimates the optimal control variates for $\widehat{\nabla J}$. We then demonstrate in numerical examples that the variance of $\widehat{\nabla J}$ is reduced by several orders of magnitude and that the convergence of (4.10) to the minimising solution is accelerated. In [148], we address the convergence of the proposed method.

One major advantage of a gradient based method is that theoretical guarantees are easily obtained. Under suitable assumptions on the noise in (4.10), one can guarantee that $A_{1:N,k}$ eventually converges to a local minimiser of J, while it is difficult to make similar assertions about the quality of the solution obtained by other approximate methods proposed in the literature.

Real-time applications: The method we propose is not yet suitable for real-time applications where decisions need to be made in tens of seconds. Our main intention in this study is to propose a simulation based method for non-linear and non-Gaussian systems that is both easy to implement and provably convergent. We wish to avoid restrictive modelling assumptions so that the proposed method is generally applicable. The conventional approach in the literature is to linearize the non-linear and non-Gaussian dynamics which is not always straightforward to implement from a numerical stability point. Also, one cannot decrease the error in the approximation of the original problem - note that linearizing is an approximation. The same is true for methods based on the PCRLB. In the method we propose however, one merely increases the number of particles or samples L, which could not be simpler. However, there is a computational cost to pay. There is currently work being done on the fast implementation of

particle filters but we have not investigated this aspect of the problem.

Our final comment in this section concerns the length of the scheduling horizon N. If the model for the evolution of the target $p(\cdot|X_n, A_{n+1})$ is not accurate then a long planning horizon N would not be meaningful since there can be significant differences between the predicted evolution and true evolution of the target. The horizon N in this case should be just long enough to avoid the sub-optimality of short-term planning. We do not address this issue in this chapter.

4.3 The Cost Gradient and its Simulation-Based Approximation

In this section, we derive the gradient of the cost function (4.6) with respect to $A_{1:N}$. We then propose a suitable simulation-based approximation for optimising with SA. Because problem (4.6) is solved for a fixed initial state distribution π_0, henceforth, we omit reference to π_0 in the notation for $\mathbf{E}_{(\pi_0, A_{1:N})}$ and denote the probability with respect to which this expectation is taken by $\mathbf{P}_{A_{1:N}}$.

Keeping in mind that $(\psi(X_n) - \langle \pi_n, \psi \rangle)^2$ is a function of the form $h(X_{1:n}, A_{1:n}, Y_{1:n})$, (4.7) implies

$$\mathbf{E}_{A_{1:N}}\{(\psi(X_n) - \langle \pi_n, \psi \rangle)^2\} = \mathbf{E}_{A_{1:n}}\{(\psi(X_n) - \langle \pi_n, \psi \rangle)^2\}.$$

For $l > n$, $\nabla_{A_l} \mathbf{E}_{A_{1:N}} \left\{ (\psi(X_n) - \langle \pi_n, \psi \rangle)^2 \right\} = 0$. For $l \leq n$, using (4.7),

$$\begin{aligned}
&\nabla_{A_l} \int \left(\psi(x_n) - \left\langle \pi_n^{(y_{1:n}, A_{1:n})}, \psi \right\rangle \right)^2 \Pi_{i=1}^n q(y_i|x_i, A_i)\, p(x_i|x_{i-1}, A_i)\, \pi_0(x_0) dx_{0:n} dy_{1:n} \\
&= \int \left(\psi(x_n) - \left\langle \pi_n^{(y_{1:n}, A_{1:n})}, \psi \right\rangle \right)^2 \nabla_{A_l} \left[\Pi_{i=1}^n q(y_i|x_i, A_i)\, p(x_i|x_{i-1}, A_i) \right] \pi_0(x_0) dx_{0:n} dy_{1:n} \\
&+ \int \nabla_{A_l} \left[\left(\psi(x_n) - \left\langle \pi_n^{(y_{1:n}, A_{1:n})}, \psi \right\rangle \right)^2 \right] \Pi_{i=1}^n q(y_i|x_i, A_i)\, p(x_i|x_{i-1}, A_i)\, \pi_0(x_0) dx_{0:n} dy_{1:n}.
\end{aligned}$$

The first term of the gradient may be written as

$$\mathbf{E}_{A_{1:N}}\{(\psi(X_n) - \langle \pi_n, \psi \rangle)^2 [\frac{\nabla_{A_l} q(Y_l|X_l, A_l)}{q(Y_l|X_l, A_l)} + \frac{\nabla_{A_l} p(X_l|X_{l-1}, A_l)}{p(X_l|X_{l-1}, A_l)}]\}$$

and the second term is

$$\begin{aligned}
&\mathbf{E}_{A_{1:N}}\{\nabla_{A_l} [(\psi(X_n) - \langle \pi_n^{(Y_{1:n}, A_{1:n})}, \psi \rangle)^2]\} \\
&= -2\mathbf{E}_{A_{1:N}}\{(\psi(X_n) - \langle \pi_n^{(Y_{1:n}, A_{1:n})}, \psi \rangle) \nabla_{A_l} \langle \pi_n^{(Y_{1:n}, A_{1:n})}, \psi \rangle\} \\
&= 0,
\end{aligned}$$

where the final equality follows upon conditioning on $Y_{1:n}$. It follows from the above derivation that to obtain an unbiased estimator of $\nabla_{A_l} J(A_{1:N})$ for a given $A_{1:N}$, one samples a realisation of states and observations $(Y_{1:N}, X_{0:N}) \sim \mathbf{P}_{A_{1:N}}$ and forms the following estimate,

$$\widehat{\nabla_{A_l} J}(A_{1:N}) = \sum_{n=l}^{N} \lambda^{N-n} \mathbf{E}_{A_{1:N}} \{(\psi(X_n) - \langle \pi_n, \psi \rangle)^2 [\frac{\nabla_{A_l} q(Y_l|X_l, A_l)}{q(Y_l|X_l, A_l)} + \frac{\nabla_{A_l} p(X_l|X_{l-1}, A_l)}{p(X_l|X_{l-1}, A_l)}] | Y_{1:n}\} \tag{4.11}$$

where we have added the conditioning on $Y_{1:n}$ as it leads to a lower variance gradient estimate.[3] In sensor scheduling applications concerning target tracking, the state process X_n is the state of the target to be tracked and often evolves independently of the action. Henceforth, we assume this independence for simplicity in presentation, i.e. $p(X_n|X_{n-1})$, and remark that the work may also be extended to the more general case of state evolution and action dependence.[4] Define the vector valued function called the *score* [131],

$$S(y, x, a) := \frac{\nabla_a q(y|x, a)}{q(y|x, a)} \in \mathbb{R}^{d_a}. \tag{4.12}$$

We may also write $\widehat{\nabla_{A_l} J}(A_{1:N})$ in (4.11) as

$$\sum_{n=l}^{N} \lambda^{N-n} \{\langle \pi_{0:n}, \psi^2(\cdot) S(Y_l, \cdot, A_l) \rangle + \langle \pi_n, \psi \rangle^2 \langle \pi_{0:n}, S(Y_l, \cdot, A_l) \rangle - 2 \langle \pi_n, \psi \rangle \langle \pi_{0:n}, \psi(\cdot) S(Y_l, \cdot, A_l) \rangle\}. \tag{4.13}$$

To implement (4.11), we see that we require both the marginal π_n and the full posterior $\pi_{0:n}$ for all N epochs, i.e., for $1 \leq n \leq N$. We propose to approximate these densities using a mixture Dirac delta-masses,

$$\hat{\pi}_{0:n}(x_{0:n}) := \sum_{j=1}^{L} w_n^{(j)} \delta_{X_{0:n}^{(j)}}(x_{0:n}), \tag{4.14}$$

where $\delta_{X_{0:n}^{(j)}}$ denotes the Dirac delta-mass located at $X_{0:n}^{(j)}$ and the *importance weights* $\{w_n^{(j)}\}_{j=1}^{L}$ are non-negative scalars that sum to one. The approximation to π_n, namely $\hat{\pi}_n$, follows by marginalising $\hat{\pi}_{0:n}$, which is nothing more than dropping $X_{0:n-1}^{(j)}$ in (4.14). There are a number of ways to define such a point-mass approximation. For example, the simplest scheme would

[3] The variance is reduced since, for two jointly distributed random variables X and Y, $\mathbf{var}(\mathbf{E}(X|Y)) = \mathbf{var}(X) - \mathbf{E}(\mathbf{var}(X|Y))$, and $\mathbf{E}(\mathbf{var}(X|Y)) > 0$.

[4] In methods that use the PCRLB [76,77,127,163], even after assuming the state process evolves independently of the actions, one still needs to evaluate the expectation of derivatives of $\ln p(X_n|X_{n-1})$ w.r.t. X_n and X_{n-1}, while this is clearly not needed in (4.11).

be to sample L independent state trajectory realisations $\left\{X_{0:n}^{(j)}\right\}_{j=1}^{L}$ from $\left(\Pi_{i=1}^{n} p\left(x_i|x_{i-1}\right)\right)\pi_0(x_0)$. The importance weights would then be

$$w_n^{(j)} := \frac{\Pi_{i=1}^{n} q\left(Y_i|X_i^{(j)}, A_i\right)}{\sum_{j=1}^{L}\Pi_{i=1}^{n} q\left(Y_i|X_i^{(j)}, A_i\right)}. \tag{4.15}$$

For any integrable function h, $\int h(x_{0:n})\hat{\pi}_{0:n}(x_{0:n})dx_{0:n}$ converges to $\int h(x_{0:n})\pi_{0:n}(x_{0:n})dx_{0:n}$ as $L \to \infty$ (see [47, Ch. 2] for a precise statement of the mode of convergence). Practically though, we would prefer a small sample size L and this simple scheme of sampling from the state transition model can result in the majority of the importance weights $w_n^{(j)}$ being very small. There are number of remedies proposed for this in the Sequential Monte Carlo, also known as Particle Filtering (PF), literature [47, Ch. 1.3.2]. For example, the importance sampling step can be designed to minimise the conditional variance of the importance weights by sampling $\left\{X_{0:n}^{(j)}\right\}_{j=1}^{L}$ from a Markov transition density that takes the observations into account, i.e., $X_n^{(j)}|X_{n-1}^{(j)} \sim k(x_n|X_{n-1}^{(j)}, Y_n)$. We emphasise that standard techniques from the Sequential Monte Carlo literature can be adopted in constructing an approximation of the form (4.14) to the full posterior but we do not study this issue in detail here. A standard Particle Filter algorithm is presented earlier in this thesis in Section 3.4.1. We summarise the discussion thus far with the following algorithm.

Algorithm 4.1 ***Simulation-Based Sensor Scheduling Procedure:***

0. Initialisation: Choose $A_{1:N,0} \in \Theta_A$, step-size $\{\alpha_k\}_{k\geq 1}$, $\alpha_k \downarrow 0$, $\sum_k \alpha_k = \infty$, PF sample size L

For $k \geq 0$, iterate

1. Sample $(X_{0:N}, Y_{1:N}) \sim P_{A_{1:N,k}}$

2. Generate the Particle filter $\hat{\pi}_{0:N}$ according to (4.15) or a more sophisticated scheme

3. Substitute $(X_{0:N}, Y_{1:N})$, $A_{1:N,k}$ and $\hat{\pi}_{0:N}$ into (4.13) to obtain $\widehat{\nabla J}(A_{1:N,k})$:

$$\widehat{\nabla_{A_l} J}(A_{1:N,k}) = \sum_{n=l}^{N} \lambda^{N-n}\{\langle\hat{\pi}_{0:n}, \psi_n^2(\cdot)S(Y_l, \cdot, A_{l,k})\rangle + \langle\hat{\pi}_n, \psi_n\rangle^2\langle\hat{\pi}_{0:n}, S(Y_l, \cdot, A_{l,k})\rangle$$
$$-2\langle\hat{\pi}_n, \psi\rangle\langle\hat{\pi}_{0:n}, \psi_n(\cdot)S(Y_l, \cdot, A_{l,k})\rangle\}. \tag{4.16}$$

4. Update trajectory: $A_{1:N,k+1} = A_{1:N,k} - \alpha_k\widehat{\nabla J}(A_{1:N,k})$

5. Set $k = k + 1$ and repeat

One may use a constant step-size $\alpha_k = \alpha$ as was done in the numerical implementation; see Section 4.6. In implementation we found that the variance of the gradient estimate (4.16) was large. The reason is, for a large horizon N, we are approximating high dimensional integrals using simulation and moreover, with a moderate sample size L. We propose a remedy in Section 4.4.1.

Computation complexity: The particle filter $\hat{\pi}_{0:N}$ can be implemented at a cost of $O(LN)$ with or without resampling (as in (4.15)). This cost dominates the cost of sampling $(X_{0:N}, Y_{1:N})$ from $P_{A_{1:N},k}$. Thus the total cost per iteration k of the simulation-based sensor scheduling procedure is still order $O(LN)$.

4.4 A Verifiably Convergent Particle Implementation

Implementing the algorithm detailed in Section 4.3 with the gradient estimate (4.16) is straightforward. However to prove its convergence we would not be able to use standard SA results. Even though (4.16) is a noisy estimate of $\nabla_{A_l} J(A_{1:N})$, the noise is not zero-mean due to the bias of the simulation-based approximations to π_n and $\pi_{0:n}$. To assert convergence of (4.10) to a minima of J, we would have to gradually increase the number of samples L to remove the bias. (Similar conditions are required for convergence of SA driven by sample averages [14, 131].) While this is fine theoretically, it is infeasible in practice as the computational complexity of the SA recursion increases with each iteration. In this section we propose an alternate implementation whose convergence can be established for a finite number of particles L. Moreover in Section 4.5 we show that the proposed implementation applied to the standard observer trajectory planning problem with bearings-only observations converges.

To simplify the presentation, we will only focus on the simple scheme of sampling from the state transition model as in (4.15) and not the more sophisticated Particle Filter. To emphasise the dependence of $\hat{\pi}_{0:n}$ on the realisation of observations $Y_{1:n}$ and the sequence of actions $A_{1:n}$, we should use the notation $\hat{\pi}_{0:n}^{(Y_{1:n},A_{1:n})}$. However, we often do not do so in order to unclutter the expressions. The reader is reminded that $\hat{\pi}_{0:n}$ should always be regarded as a function of $(Y_{1:n}, A_{1:n})$. Henceforth, we fix the set of L state trajectory samples $\left\{X_{0:N}^{(j)}\right\}_{j=1}^{L}$, i.e., they are sampled once at the start and reused throughout to form $\hat{\pi}_{0:n}$.

By a conditioning argument, $J(A_{1:N})$ can be written as $\sum_{n=1}^{N} \lambda^{N-n} \mathbf{E}_{A_{1:N}} \left\{ \langle \pi_n, \psi^2 \rangle - \langle \pi_n, \psi \rangle^2 \right\}$

and we form the following approximation to J,

$$\hat{J}(A_{1:N}) = \sum_{n=1}^{N} \lambda^{N-n} \mathbf{E}_{A_{1:N}} \left\{ \langle \hat{\pi}_n, \psi^2 \rangle - \langle \hat{\pi}_n, \psi \rangle^2 \right\}. \tag{4.17}$$

Since the error in the approximation $\hat{\pi}_{0:n}$ diminishes as the sample size L increases, we would expect $\hat{J}$ to be a good approximation to J for sufficiently large L. We will then derive an unbiased estimate of the gradient of $\hat{J}$ in a similar manner to J above and minimise $\hat{J}$ via SA. This approach can be analysed and we show that, under suitable assumptions, SA converges to a local minima of $\hat{J}$ almost surely.

In the same way as the gradient of J was derived in (4.11) we have

$$\begin{aligned}
\nabla_{A_l} \hat{J}(A_{1:N}) &= \sum_{n=1}^{N} \lambda^{N-n} \nabla_{A_l} \mathbf{E}_{A_{1:N}} \{ \langle \hat{\pi}_n, \psi^2 \rangle - \langle \hat{\pi}_n, \psi \rangle^2 \} \\
&= \mathbf{E}_{A_{1:N}} \{ \sum_{n=l}^{N} \lambda^{N-n} (\langle \hat{\pi}_n, \psi^2 \rangle - \langle \hat{\pi}_n, \psi \rangle^2) S(Y_l, X_l, A_l) \} \qquad (4.18) \\
&+ \mathbf{E}_{A_{1:N}} \{ \sum_{n=l}^{N} \lambda^{N-n} (\nabla_{A_l} \langle \hat{\pi}_n, \psi^2 \rangle - 2 \langle \hat{\pi}_n, \psi \rangle \nabla_{A_l} \langle \hat{\pi}_n, \psi \rangle) \} \qquad (4.19)
\end{aligned}$$

where[5]

$$\nabla_{A_l} \langle \hat{\pi}_n, \psi \rangle = \langle \hat{\pi}_{0:n}, \psi(\cdot) S(Y_l, \cdot, A_l) \rangle - \langle \hat{\pi}_n, \psi \rangle \langle \hat{\pi}_{0:n}, S(Y_l, \cdot, A_l) \rangle. \tag{4.20}$$

It is now straightforward to obtain a simulation-based approximation of $\nabla \hat{J}(A_{1:N})$. For a given $A_{1:N}$, one samples a realisation of states and observations $(Y_{1:N}, X_{0:N}) \sim \mathbf{P}_{A_{1:N}}$ and forms the following unbiased estimate of $\nabla_{A_l} \hat{J}(A_{1:N})$, for $l = 1, \ldots, N$,

$$\begin{aligned}
& S(Y_l, X_l, A_l) \sum_{n=l}^{N} \lambda^{N-n} \left(\langle \hat{\pi}_n, \psi^2 \rangle - \langle \hat{\pi}_n, \psi \rangle^2 \right) \\
+ & \sum_{n=l}^{N} \lambda^{N-n} \left(\nabla_{A_l} \langle \hat{\pi}_n, \psi^2 \rangle - 2 \langle \hat{\pi}_n, \psi \rangle \nabla_{A_l} \langle \hat{\pi}_n, \psi \rangle \right). \qquad (4.21)
\end{aligned}$$

4.4.1 Variance Reduction by Control Variates

In implementation, we found that the variance of the gradient estimate (4.21) (or (4.16)) was quite large. This is because we are approximating high dimensional integrals using simulation and moreover, with a moderate sample size L. Naturally, it would be possible to reduce the

[5] It is possible to compute $\nabla_{A_l} \langle \hat{\pi}_n, \psi \rangle$ when resampling is employed to construct $\hat{\pi}_n$ as done in the standard Particle Filter Details of the gradient can be found in [135].

variance by simply increasing the number of samples. As we do not wish to do so, our aim is to extract the most accurate estimates of the quantities of interest for a given set of samples.

Control Variates are widely used to reduce the variance in simulation-based approximations [65, 68, 101]. The method involves collecting additional statistics from the samples and is very simple to implement. Recently, it has been shown that other popular variance reduction techniques such as conditional Monte Carlo, antithetics, rotation sampling, stratification can be viewed as various implementations of this method [68]. We now describe how control variates may be implemented for our problem.

For a random variable W, consider the problem of estimating $\mathbf{E}(W)$ when we have access to a zero-mean random variable Z correlated with W. Rather than using a realisation of W as an unbiased estimate, we use $W - bZ$ where b is a constant. The estimator $W - bZ$ is also unbiased. Furthermore, the function of b

$$\mathbf{var}(W - bZ) = \mathbf{var}(W) - 2b\mathbf{cov}(W, Z) + b^2\mathbf{var}(Z) \tag{4.22}$$

is convex and is minimised at $b^* = \mathbf{cov}(W, Z)/\mathbf{var}(Z)$, which implies the variance of the estimate $W - b^*Z$ of $\mathbf{E}(W)$ is less than the variance of the estimate W. The random variable Z is referred to as the control variate (CV) and we call b the CV constant [68].

In the context of the gradient estimate in (4.21), we found in implementation that reducing the variance of the estimate of (4.18) was sufficient. The score in (4.18) is zero-mean, i.e. $\mathbf{E}_{A_{1:N}}\{S(Y_l, X_l, A_l)\} = 0$, and we use it as the CV. Doing so yields the following unbiased estimator of $\nabla_{A_l}\hat{J}$ instead of (4.21),

$$\begin{aligned}\mathrm{diag}(S(Y_l, X_l, A_l))(-b_l + \sum_{n=l}^{N} \lambda^{N-n}(\langle\hat{\pi}_n, \psi^2\rangle - \langle\hat{\pi}_n, \psi\rangle^2)\mathbf{1}) \\ + \sum_{n=l}^{N} \lambda^{N-n}(\nabla_{A_l}\langle\hat{\pi}_n, \psi^2\rangle - 2\langle\hat{\pi}_n, \psi\rangle\nabla_{A_l}\langle\hat{\pi}_n, \psi\rangle),\end{aligned} \tag{4.23}$$

where $\mathbf{1} \in \mathbb{R}^{d_a}$ and the CV constant (vector) $b_l \in \mathbb{R}^{d_a}$ is to be determined in order to minimise the variance of the estimate. Noting that the optimal CV constant is a solution of the minimisation problem (4.22), we may employ the following SA algorithm to converge to it,

$$b_l \longleftarrow b_l - \beta\mathrm{diag}(S(Y_l, X_l, A_l))(\mathrm{diag}(S(Y_l, X_l, A_l))b_l - \sum_{n=l}^{N} \lambda^{N-n}(\langle\hat{\pi}_n, \psi^2\rangle - \langle\hat{\pi}_n, \psi\rangle^2)\mathbf{1}), \tag{4.24}$$

where β is the step-size. Under suitable assumptions we will have b_l converging to

$$\mathbf{E}_{A_{1:N}}\{\text{diag}(S(Y_l, X_l, A_l))^2\}^{-1}\mathbf{E}_{A_{1:N}}\{\text{diag}(S(Y_l, X_l, A_l))^2 \sum_{n=l}^{N} \lambda^{N-n}(\langle\hat{\pi}_n, \psi^2\rangle - \langle\hat{\pi}_n, \psi\rangle^2)\mathbf{1}\}. \tag{4.25}$$

The same approach applies when minimising the variance of the gradient estimate (4.16) with control variates.

4.4.2 The Main Algorithm

We now state the main algorithm of the chapter whose convergence we subsequently prove. It is a two time-scale SA algorithm to minimise $\hat{J}$ using the reduced variance estimate of $\nabla\hat{J}$ given by (4.23) and (4.24). We do so for the case with action path constraints as specified in (4.9). We can also derive a similar two time-scale version of the algorithm presented in Section 4.3; see [150] for details.

Solving problem (4.6) with (4.9) is equivalent to minimising $\hat{J} \circ F$ (which is the composite function $\hat{J}(F(\cdot))$) over $\left(\mathbb{R}^{d_u}\right)^N$. The appropriate modification to (4.10) for this case is

$$U_{1:N,k+1} = U_{1:N,k} - \alpha_k(\nabla\hat{J} \circ F(U_{1:N})|_{U_{1:N}=U_{1:N,k}} + \text{noise})$$

where $\nabla\hat{J} \circ F(U_{1:N}) = \nabla F(U_{1:N})\nabla\hat{J}(F(U_{1:N}))$.

We introduce the following functions to make the presentation of the main algorithm concise. For each $A_{1:N}$, define the functions $h_{i,A_{1:N}} : \left(\mathbb{R}^{d_x}\right)^{N+1} \times \left(\mathbb{R}^{d_y}\right)^N \to \left(\mathbb{R}^{d_a}\right)^N$, $i = 1, 2$, as follows:

$$h_{1,A_{1:N}}(X_{0:N}, Y_{1:N}) = [S(Y_l, X_l, A_l) \sum_{n=l}^{N} \lambda^{N-n}(\langle\hat{\pi}_n, \psi^2\rangle - \langle\hat{\pi}_n, \psi\rangle^2)]_{l=1,\ldots,N}, \tag{4.26}$$

$$h_{2,A_{1:N}}(X_{0:N}, Y_{1:N}) = [\sum_{n=l}^{N} \lambda^{N-n}(\nabla_{A_l}\langle\hat{\pi}_n, \psi^2\rangle - 2\langle\hat{\pi}_n, \psi\rangle\nabla_{A_l}\langle\hat{\pi}_n, \psi\rangle)]_{l=1,\ldots,N}. \tag{4.27}$$

Note that

$$\nabla\hat{J}(A_{1:N}) = \mathbf{E}_{A_{1:N}}\{h_{1,A_{1:N}}(X_{0:N}, Y_{1:N}) + h_{2,A_{1:N}}(X_{0:N}, Y_{1:N})\}, \in (\mathbb{R}^{d_a})^N.$$

For technical reasons concerning the convergence of the two time-scale SA algorithm below [148], we introduce the positive scalar valued function $\Gamma : \left(\mathbb{R}^{d_a}\right)^N \to (0, \infty)$,

$$\Gamma(b) := \frac{C}{1 + |b|}, \tag{4.28}$$

where C is a positive constant. The function Γ is needed to ensure that the CV constants remain bounded almost surely. However, we set $\Gamma(b) = 1$ in implementation.

Algorithm 4.2 ***The two time-scale SA algorithm for solving the sensor scheduling problem:***

For conciseness, let

$$\begin{aligned}\theta &= U_{1:N},\\ \tilde{\theta} &= A_{1:N} \quad (= F(\theta)),\\ \omega &= (X_{0:N}, Y_{1:N}).\end{aligned}$$

$$\theta_{k+1} = \theta_k - \alpha_{k+1}\Gamma(b_k)\nabla F(\theta_k)(h_{1,\tilde{\theta}_k}(\omega_{k+1}) + h_{2,\tilde{\theta}_k}(\omega_{k+1}) - S_{\tilde{\theta}_k}(\omega_{k+1})b_k), \tag{4.29}$$

$$b_{k+1} = b_k - \beta_{k+1}S^2_{\tilde{\theta}_k}(\omega_{k+1})b_k + \beta_{k+1}S_{\tilde{\theta}_k}(\omega_{k+1})h_{1,\tilde{\theta}_k}(\omega_{k+1}), \tag{4.30}$$

$$\omega_{k+1} \sim \mathbf{P}_{\tilde{\theta}_k}, \tag{4.31}$$

$$\tilde{\theta}_k = F(\theta_k), \qquad k \geq 0, \tag{4.32}$$

where

$$S_{A_{1:N,k}}(X_{0:N,k+1}, Y_{1:N,k+1}) = \mathit{diag}([S(Y_{l,k+1}, X_{l,k+1}, A_{l,k})]_{l=1,\ldots,N}). \tag{4.33}$$

(Note that $\theta_k = U_{1:N,k}$, $\tilde{\theta}_k = A_{1:N,k}$, $\omega_{k+1} = (X_{0:N,k+1}, Y_{1:N,k+1})$.)

As for the algorithm presented in Section 4.3, the cost of sampling ω_{k+1} and computing $\hat{\pi}_{0:N}$ is $O(NL)$. The only difference is that we sample the state trajectories that form the approximate posterior density $\hat{\pi}_{0:N}$ (which also gives us $\{\hat{\pi}_n\}_{n=0}^N$) at the start and do not change them thereafter. Also, no resampling is employed. The storage requirement does however increase. Computing $h_{2,\tilde{\theta}_k}$ is the most expensive step, specifically $O(N^2L)$, and subsumes all other costs, matrix multiplications included. Thus the total cost of the algorithm is $O(N^2L)$. Note that the cost is still linear in L, which the most important point since the horizon is typically very small compared to the number of state trajectory samples L.

Assumption 4.4.1 *The step-size sequences $\{\alpha_k\}$ and $\{\beta_k\}$ are non-negative, sum to infinity, are squared summable, and for some $p > 0$ satisfy $\sum_k \left(\frac{\alpha_k}{\beta_k}\right)^p < \infty$.*

Typically, the step-sizes are

$$\alpha_k = k^{-\alpha}, \qquad \beta_k = k^{-\beta}, \tag{4.34}$$

where $\alpha > \beta > 0.5$. Thus, $\sum_k \left(\frac{\alpha_k}{\beta_k}\right)^p < \infty$ may only be satisfied for a large positive p. Since α_k tends to zero more quickly than β_k, the recursion for the actions (4.29) is said to evolve on a slower time-scale than that for the CV constants (4.30). By having $U_{1:N,k}$ evolve more slowly than b_k, we allow b_k to 'track' the optimal CV constants, which depend on the point at which the gradient $\nabla \hat{J}$ is evaluated (see (4.25)). In [148], we establish the convergence of algorithm (4.29)-(4.32) for the choice of step-sizes in Assumption 4.4.1. However, in the numerical example presented in Section 4.6, we set function $\Gamma(b) = 1$, use constant step-sizes $\alpha_k = \alpha$ and $\beta_k = \beta$ and still demonstrate convergence. For SA in general, decreasing step-sizes are essential for almost sure convergence. If fixed step-sizes are used, then we may still have convergence but now the iterates 'oscillate' about their limiting values with variance proportional to the step-size.

The convergence of a two time-scale SA algorithm related to (4.29)-(4.32) was studied in [89]. We may write the slow time-scale process in a more general form than (4.29) as,$\theta_{k+1} = \theta_k + \alpha_{k+1} H_{k+1}$. If the parameter θ_k does not change, say $\theta_k = \theta$ for all k, the process $\{b_k\}$ would converge to some $\bar{b}(\theta)$. When θ_k varies slowly, we would like the process $\{b_k\}$ to track $\bar{b}(\theta_k)$. Under certain regularity assumptions on the process $\{H_k\}$ [89], it can be shown that this would be the case when θ_k did change. As for the convergence of $\{\theta_k\}$, we may use the line of proof in [23] to show $\liminf_k \left|\nabla(\hat{J} \circ F)(\theta_k)\right| = 0$. Details are in [148].

4.5 Application to Observer Trajectory Planning

In observer trajectory planning, we wish to track a maneuvering target for N epochs. At epoch n, X_n denotes the state of the target, A_n the position of the observer and Y_n the partial observation of the target state, i.e., $Y_n = g(X_n, A_n, V_n)$, where V_n is measurement noise. Typically, the observer has its own motion model and we let X_n^o denote state of the observer. The observer state descriptor usually includes its position and therefore A_n corresponds to certain components of X_n^o. The aim of OTP is to adaptively maneuver the observer to optimise the tracking performance of the target.

In this section, we formalise the OTP problem for a bearings-only observation model as

an instance of the sensor scheduling problem in Section 4.2. We also give results concerning convergence of algorithm (4.29)-(4.32) for this application. As we show below, while most existing work in the literature concerns OTP for one observer only, our proposed framework can handle multiple observers simultaneously. There are some convergence issues though, as we point out in the numerical examples in Section 4.6. Adding more observers can result in an increased number of local minima of the cost function J while gradient based algorithms are only guaranteed to converge to a local minimum. Also, J can be increasingly flat at the minima, which means varying the trajectory of the observers at any minima will result in only small changes to J. In practice, this can slow down the convergence of SA.

We do not need to specify the target model explicitly. Our only concern is that we can sample from the model. Maneuvering targets are often modelled as a *jump Markov linear system* (JMLS) [49]. The state of the target is comprised of continuous and discrete valued variables, i.e., $X_n = [r_{x,n}, v_{x,n}, r_{y,n}, v_{y,n}, \xi_n]^T \in \mathbb{R}^4 \times \Xi$, where $(r_{x,n}, r_{y,n})$ denotes the target's (Cartesian) coordinates at time n, $(v_{x,n}, v_{y,n})$ denotes the target velocity in the x and y directions, and ξ_n denotes the mode of the target, which belongs to a finite set Ξ. The target switches discontinuously, as indicated by ξ_n, between constant velocity maneuvers. In Section 4.6, we consider a maneuvering target in the examples.

As indicated in (4.9), we require an observer model of the form $A_{1:N} = F(U_{1:N})$, where we exert control on the observer positions $A_{1:N}$ through the variables $U_{1:N}$. For instance, the accelerations of the observer could be determined from the input $U_{1:N}$, which will in turn determine the observer trajectory. The convergence results of Propositions 4.5.1 and 4.5.2 below do not depend on the specific form of F but only that this function is sufficiently regular. We now give an example of F which is adopted in the Numerical Example section.

Example 4.5.1 *Let the state of the observer be* $X_n^o = \left[r_{x,n}^o, v_{x,n}^o, r_{y,n}^o, v_{y,n}^o\right]^T$, *with* $A_n = \left[r_{x,n}^o, r_{y,n}^o\right]^T$. *Assume a kinematic model for the evolution of the state,*

$$X_{n+1}^o = \underbrace{\begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{=: G} X_n^o + \underbrace{\begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}}_{=: H} \times C \times \arctan(U_{n+1}) \tag{4.35}$$

where the initial state X_0^o is fixed, T is the sampling interval, and $U_{n+1} \in \mathbb{R}^2$ determines the acceleration in the x and y directions. We have included the function $\arctan$ *and the positive diagonal matrix C. The function* $\arctan$ *and its first two derivatives are bounded. Also,* $\arctan$ *is linear around zero and makes a nice choice of saturating function for the acceleration; naturally the acceleration cannot be unbounded. The matrix C alters the saturation behaviour of the acceleration. The observer trajectory is completely determined once X_0^o and $U_{1:N}$ are given,*

$$A_n = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix} \times \left(G^n X_0^o + \sum_{i=1}^{n} G^{n-i} HC \arctan(U_i) \right). \tag{4.36}$$

The function F in (4.8) is now implicitly defined by (4.36).

In the bearings-only model, the observation process $\{Y_n\}_{n\geq 0}$ ($\subset \mathbb{R}$) is generated according to

$$Y_n = \arctan\left(\frac{r_{x,n} - A_n(1)}{r_{y,n} - A_n(2)}\right) + V_n, \tag{4.37}$$

where $V_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_Y^2)$. In our simulation-based framework, we require the observation process density to be known and differentiable w.r.t. A_n. The bearings-only case is one such example. To present the convergence results of Proposition 4.5.1 and 4.5.2 below, we will assume that the x and y position of the target corresponds to the first and third component of the state descriptor X_n,

$$X_n = [r_{x,n}, \cdot, r_{y,n}, \cdots]^T, \tag{4.38}$$

which is usual convention in the literature. The score is then given by

$$\begin{aligned} &S(y,x,a)^T \\ &= [\nabla_{a(1)} q(y|x,a), \qquad \nabla_{a(2)} q(y|x,a)] \times q(y|x,a)^{-1} \\ &= -\sigma_Y^{-2} \frac{y - \arctan(\frac{x(1)-a(1)}{x(3)-a(2)})}{1 + (\frac{x(1)-a(1)}{x(3)-a(2)})^2} [\frac{1}{x(3)-a(2)}, -\frac{x(1)-a(1)}{(x(3)-a(2))^2}]. \end{aligned} \tag{4.39}$$

For the case of multiple observers, say p of them, let the position of observer l at epoch n be denoted by $A_n^{(l)}$. Also, assume that each observer measures a bearings angle according to (4.37) independently of the other observers, i.e., observer l receives the measurement $Y_n^{(l)}$ generated according to the model (4.37) based on its own position $A_n^{(l)}$. We stack these observations together as a vector-valued observation $Y_n = [Y_n^{(1)}, \ldots, Y_n^{(p)}]^T$. Likewise, we stack the positions to form the effective position $A_n = [(A_n^{(1)})^T, \cdots, (A_n^{(p)})^T]^T$, The observation density for this

multiple observer case is $q(Y_n|X_n, A_n) = q(Y_n^{(1)}|X_n, A_n^{(1)}) \times \cdots \times q(Y_n^{(p)}|X_n, A_n^{(p)})$. It is apparent that we are now effectively in the original single observer setting and may proceed to solve the multiple OTP problem as above. Note that we are now optimising the tracking performance criterion over the space of possible trajectories for p observers, which implies that the observers cooperate. In the examples of Section 4.6, we study the two observer case.

4.5.1 Convergence For Bearings-Only Tracking

For the bearings-only observation model, we have the following sufficient conditions for the convergence of the slow and fast time-scale. The sufficient conditions (4.40)-(4.43) are only restrictions on the target state transition model, and the range of the function F that is used to map the sequence $U_{1:N}$ (which could be accelerations) to the observer positions for all N epochs. We omit the proof and refer to the reader to the technical report [148], which is available online, for full details.

The following result concerns the cost function (4.17) with $\lambda = 0$ and can be generalised to any $\lambda \in [0, 1]$.

Proposition 4.5.1 *Consider algorithm (4.29)-(4.32) for the bearings-only observation model (4.37). Suppose the following assumptions hold:*

$$\sup_{A_{1:N} \in range(F)} \mathbf{E}_{A_{1:N}} \left\{ \left| \frac{1}{X_n(3) - A_n(2)} \right|^p \right\} < \infty,$$
$$1 \leq n \leq N, \quad p > 0, \tag{4.40}$$
$$\sup_{A_{1:N} \in range(F)} \max_l \left| \frac{1}{X_n^{(l)}(3) - A_n(2)} \right| < \infty,$$
$$1 \leq n \leq N, \tag{4.41}$$
$$\inf_{1 \leq n \leq N} \inf_{A_{1:N} \in range(F)} \mathbf{E}_{A_{1:N}} \left\{ \sigma_Y^{-2} \frac{\frac{1}{(X_n(3) - A_n(2))^2}}{\left[1 + \left(\frac{X_n(1) - A_n(1)}{X_n(3) - A_n(2)}\right)^2\right]^2} \right\} > 0, \tag{4.42}$$
$$\inf_{1 \leq n \leq N} \inf_{A_{1:N} \in range(F)} \mathbf{E}_{A_{1:N}} \left\{ \sigma_Y^{-2} \frac{\left[\frac{X_n(1) - A_n(1)}{(X_n(3) - A_n(2))^2}\right]^2}{\left[1 + \left(\frac{X_n(1) - A_n(1)}{X_n(3) - A_n(2)}\right)^2\right]^2} \right\} > 0. \tag{4.43}$$

Then, almost surely, $\sup_k |b_k| < \infty$ *and*

$$\lim_k \left| b_k - \overline{S^2}(A_{1:N,k})^{-1} \overline{S \times h_1}(A_{1:N,k}) \right| = 0.$$

Furthermore, if F has bounded second order derivatives then, almost surely, $\liminf_k \left| \nabla(\hat{J} \circ F)(U_{1:N,k}) \right| = 0$.

Proof. See [148]. ■

Recall that the expectation operator $\mathbf{E}_{A_{1:N}}\{\cdot\}$ above is an abbreviation for $\mathbf{E}_{(X_{0:N},Y_{1:N})\sim\mathbf{P}_{A_{1:N}}}\{\cdot\}$. Condition (4.41) relates to the samples used to approximate the posterior density in (4.14)-(4.15). Also, the first and third component of the target state is its x and y component respectively. Note that the proposition does not limit the specific form of function F that relates inputs $U_{1:N}$ to actions $A_{1:N}$. It only restricts range(F) and requires F to be sufficiently regular as specified by the last assumption concerning bounded second order derivatives. For F defined implicitly by (4.36), this assumption is satisfied.

The next result gives the conditions under which assumptions (4.40)-(4.43) hold. This result basically says that if the support of $X_{0:N}$ and the range of function F do not intersect, then the assumptions hold and we have the desired convergence of two time-scale SA for OTP. It is interesting to note that the scenario in which the support of $X_{0:N}$ and the range of F do not intersect is a standard setting studied by previous works on OTP for bearings-only observations [25, 108, 163] and hence the conditions of Proposition 4.5.1 are not restrictive for the application.

Proposition 4.5.2 *Write the mapping $F : R^{2N} \to R^{2N}$ as $F = [F_{1,1}, F_{1,2}, \cdots, F_{N,1}, F_{N,2}]^T$. (Note that $A_n(j) = F_{n,j}(U_{1:N})$.) Suppose that the density of $X_{0:N}$, $f(x_{0:N})$, has a compact support $\mathcal{K}_f \subset R^{4(N+1)}$. Furthermore, suppose that for each $1 \leq n \leq N$, the compact set $\mathcal{K}_{f,n} := \{x_n(3) |\, x_{0:N} \in \mathcal{K}_f\}$ does not intersect with the closure of the set range($F_{n,2}$), i.e., there exists a compact set $\mathcal{K}_{A,n}$ such that range($F_{n,2}$) $\subset \mathcal{K}_{A,n}$, and $\mathcal{K}_{f,n} \cap \mathcal{K}_{A,n} = \varnothing$. Then, conditions (4.40)-(4.43) are satisfied.*

Proof. We prove the result for (4.43) while the rest follow similar arguments. Note that

$$\inf_{X_{0:N}\in\mathcal{K}_f, A_{1:N}\in\text{range}(F)} \frac{\frac{1}{(X_n(3)-A_n(2))^4}}{\left[1+\left(\frac{X_n(1)-A_n(1)}{X_n(3)-A_n(2)}\right)^2\right]^2} \geq \frac{\frac{1}{\left(\sup_{X_n(3)\in\mathcal{K}_{f,n}, A_n(2)\in\mathcal{K}_{A,n}} |X_n(3)-A_n(2)|\right)^4}}{\left[1+\left(\frac{\sup_{X_{0:N}\in\mathcal{K}_f, A_{1:N}\in\text{range}(F)} |X_n(1)-A_n(1)|}{\inf_{X_n(3)\in\mathcal{K}_{f,n}, A_n(2)\in\mathcal{K}_{A,n}} |X_n(3)-A_n(2)|}\right)^2\right]^2} = C \quad (> 0).$$

The equality in the second line above follows since we are maximising, and minimising, continuous functions over compact sets; for each problem, at least one solution from the compact

set exists. Thus,

$$\mathbf{E}_{A_{1:N}}\left\{\frac{\left[\frac{X_n(1)-A_n(1)}{(X_n(3)-A_n(2))^2}\right]^2}{\left[1+\left(\frac{X_n(1)-A_n(1)}{X_n(3)-A_n(2)}\right)^2\right]^2}\right\}=\mathbf{E}_{A_{1:N}}\left\{I_{\mathcal{K}_f}(X_{0:N})\frac{\left[\frac{X_n(1)-A_n(1)}{(X_n(3)-A_n(2))^2}\right]^2}{\left[1+\left(\frac{X_n(1)-A_n(1)}{X_n(3)-A_n(2)}\right)^2\right]^2}\right\}$$

$$\geq \mathbf{E}_{A_{1:N}}\left\{\begin{array}{c} I_{\mathcal{K}_f}(X_{0:N})\left(X_n(1)-A_n(1)\right)^2 \\ \times \inf_{X_{0:N}\in\mathcal{K}_f, A_{1:N}\in\text{range}(F)} \frac{\frac{1}{(X_n(3)-A_n(2))^4}}{\left[1+\left(\frac{X_n(1)-A_n(1)}{X_n(3)-A_n(2)}\right)^2\right]^2} \end{array}\right\}$$

$$\geq C\times\mathbf{E}_{A_{1:N}}\left\{\left(X_n(1)-A_n(1)\right)^2\right\}$$

$$\geq C\times\mathbf{var}\left\{X_n(1)\right\},$$

where in the last line, the density of $X_{0:N}$ is independent of the sequence of actions $A_{1:N}$ and hence we write $\mathbf{var}\{\cdot\}$ omitting reference to the actions. For more details, see [148]. ■

4.6 Numerical Example

The aim of this section is to demonstrate the utility of the proposed simulation-based algorithm for the OTP problem. In addition to demonstrating various convergence aspects of the algorithm, we will also solve for the optimal open-loop observer trajectory under a variety of tracking scenarios, namely, with a fast observer, a slow observer and cooperating observers. Open loop feedback control is also implemented for the cooperating observers case.

All examples below for OLFC concern a maneuvering target where the target follows a linear Gaussian model between maneuvers. However, when solving for the open loop trajectory we assume a linear Gaussian model $X_{n+1}=GX_n+HW_n$ with increased acceleration noise to account for the un-modelled maneuvers. This is more robust in practise as one usually knows little about the target's precise model. The particle cloud in Figure 4.1(a) are independent samples from the target's dynamic model (4.3) to help visualise it. The target starts at $(0,0)$ and moves northeast. Shown in Figure 4.1(c) is the actual maneuvering target when the OLFC trajectory is constructed. The maneuvering target also starts at $(0,0)$ and moves northeast. For the maneuvering target, at time $t=3$, the velocity of the target in the y direction is increased to induce the maneuver. Note that the target maneuver was intentionally chosen to be far more drastic than that predicted by its model. This was done to contrast the constructed open-loop and OLFC trajectories. In all the examples below, the problem horizon N is 7 and we choose function ψ to be $\psi(X_n)=w_1X_n(1)+w_2X_n(3)$ where weights $w_1,w_2\in[0,1]$ are selected to

trade-off accuracy in tracking the x and y coordinates.

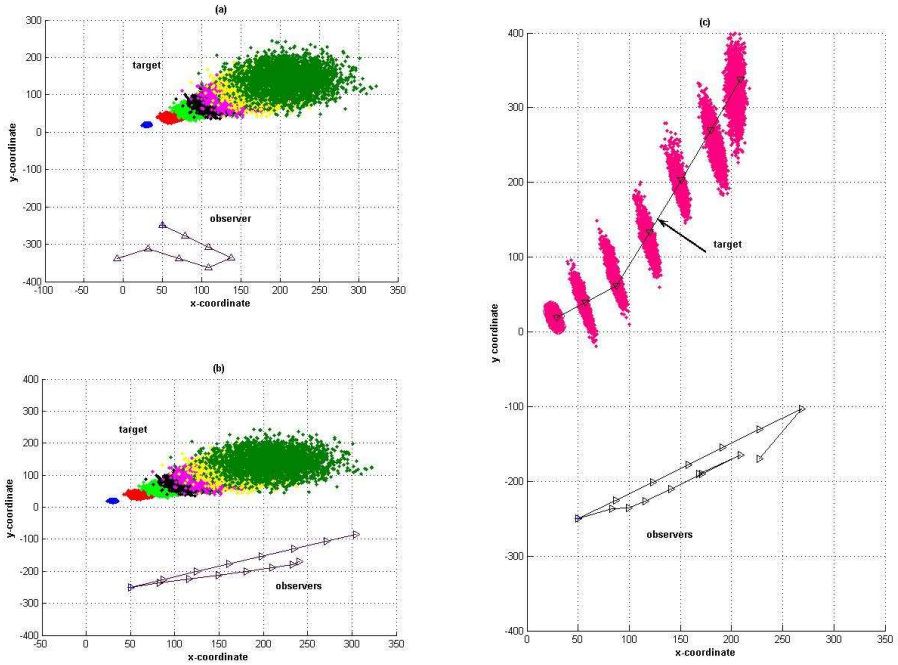


Figure 4.1: *(a) Single fast observer open loop trajectory. Particle clouds are trajectory samples from the target's dynamical model in (4.3). Shown are many trajectory samples* $(X_1, X_2, \ldots, X_N)$ *where the various* X_1 *samples is the blue cloud, the red cloud is* X_2*, lime is* X_3*, black* X_4 *and so on. Target moving northeast but observer moves southeast and does a hook-turn. (b) Two cooperating fast observers open loop trajectory; both moving northeast. Particle cloud as in part (a). (c) The OLFC trajectory of two cooperating fast observers moving northeast. A maneuvering target is being tracked. Particle cloud are samples from the target filtering density* $\{\widehat{\pi}_n\}_{n=0}^N$*. All observers commence at* $(50, -250)$*. The open loop trajectories of figures (a) and (b) took several hours to compute with a 2.8 GHz Pentium 4 CPU.*

4.6.1 Fast Observers

The setting for this example is a maneuvering target that is to be tracked by a single fast observer and two cooperating fast observers. The term fast is to be understood in the sense that the observer in the subsequent example is significantly more constrained in its motion. The observer motion model is given by (4.35), with a fast or slow observer defined by choosing constant matrix C appropriately. In Figure 4.1(a) the optimal open-loop trajectory of the single fast observer is plotted for a horizon 7 problem. Figure 4.1(b) shows the difference in the op-

timal open-loop trajectory obtained when there are two cooperating fast observers. The single fast observer commences at coordinate (50,-250) and travels towards the southeast while making a hook turn. This is in contrast to the two fast observers, both commencing at (50,-250), and travelling in straight lines towards the northeast. Figure 4.1(c) shows the OLFC trajectory obtained for the same two cooperating fast observers. The cloud of particles shown here are samples from the filtering density at each time, which is implemented using a particle filter. We note that the OLFC trajectory performs more maneuvers than the equivalent open loop one in order to respond to the actual maneuvers of the target. Also, the open loop trajectory of a single observer differs greatly from that of two cooperating observers.

4.6.2 Slow Observers

We now contrast the optimal trajectory of the fast observers above with that of slow observers. Figure 4.2(a) shows the optimal open-loop trajectory of one slow observer, and Figure 4.2(b) that of two cooperating slow observers. All observers commence at coordinate (250,-250). Note that a single slow observer is obliged to do more maneuvers to improve the tracking performance since it is significantly more constrained in motion. Figure 4.2(c) shows the OLFC trajectory obtained for two cooperating slow observers. A more detailed plot of the OLFC trajectory is shown in Figure 4.2(d). Figure 4.2(c) also shows the actual target maneuver and the particle cloud surrounding it are samples from its filtering density.

Figure 4.3 shows a running plot of the performance criterion, as the (open loop) trajectories are computed by algorithm (4.29)-(4.32), for the single and two cooperating slow observers of Figure 4.2(a) and 4.2(b). The cost was estimated using Monte Carlo simulation. As Figure 4.3 indicates, two observers cooperating during tracking outperforms one. In Figure 4.4, the convergence of the trajectory $A_{1:N,k}$ computed using algorithm (4.29)-(4.32) for the two slow cooperating observers of Figure 4.2(b) is shown. Note that although it converges slowly, there is little gain in performance beyond iteration 1.5×10^6 as indicated by the running cost plot in Figure 4.3. All these results are very similar for the fast observers case and are omitted.

4.6.3 Variance Reduction

Here we illustrate the importance of using the control variate variance reduction scheme. In algorithm (4.29)-(4.32), we do not update the actions, i.e., $A_{1:N,k} = A_{1:N,0}$ for all k. In Figure

4.5(a) we show the gradient estimates without the control variate while in Figure 4.5(b) we show the gradient estimate as the control variate iterated by (4.30) converges. The convergence of the control variate iterated by (4.30) is shown in Figure 4.5(c). Note the significant variance reduction achieved which will speed up the convergence of the actions iterated by (4.29). The plots only show the gradient of the performance criterion with respect to x-coordinate of the action at time 1, i.e., $A_1(1)$. The effect of the control variate on the remaining components of the performance criterion gradient is similar.

4.7 Conclusion

In this chapter we proposed a novel simulation-based method to solve the sensor scheduling problem for the case in which the state, observation and action spaces are continuous valued vectors. This general continuous state-space case is important as it is the natural framework for many applications, like observer trajectory planning. We avoided restrictive modelling assumptions on the continuous state HMM, such as assuming a linear and (or) Gaussian system, by recourse to simulation-based methods. This chapter solved the sensor scheduling problem with continuous action space directly, and not a surrogate problem defined through the PCRLB or otherwise, which is the approach adopted in other works. The novel simulation-based method presented used a two timescale Stochastic Approximation algorithm to find the optimal actions. We presented general convergence results for convergence to a local minima of the cost function.

As an instance of the sensor scheduling problem, we studied the observer trajectory planning problem for a bearings-only application. We established that in the standard scenario where the observer and the target are well separated, our simulation-based algorithm converged to a local minimum of the cost function. As a generalisation, it is also possible to extend our work to sensor scheduling based on an information criterion like the Kullback-Leibler criterion [108]. In future work we plan to apply our simulation-based method to related problems in Robotics and Sensor Networks.

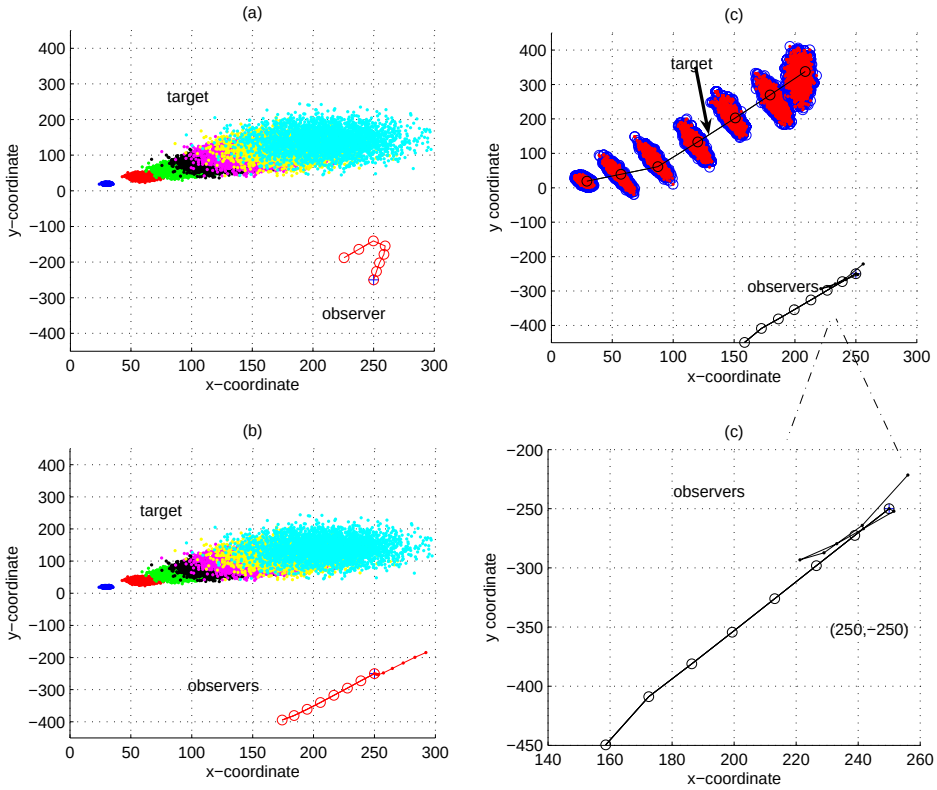


Figure 4.2: *(a) Single slow observer open loop trajectory. Particle clouds are trajectory samples from the target's dynamical model in (4.3). Shown are many trajectory samples* $(X_1, X_2, \ldots, X_N)$ *where the various* X_1 *samples is the blue cloud, the red cloud is* X_2*, lime is* X_3*, black* X_4 *and so on. Target moving northeast and observer moves southwest while doing a hook-turn. (b) Two cooperating slow observers open loop trajectory; one moving northeast and the other southwest. Particle cloud as in part (a). (c) Two cooperating slow observers OLFC trajectory with a maneuvering target. Particle cloud are samples from the target's filtering density* $\{\widehat{\pi}_n\}_{n=0}^{N}$*. (d) Magnification of the OLFC trajectory of the observers. All slow observers commence from coordinate* $(250, -250)$*. The open loop trajectories of figures (a) and (b) took several hours to compute with a 2.8 GHz Pentium 4 CPU.*

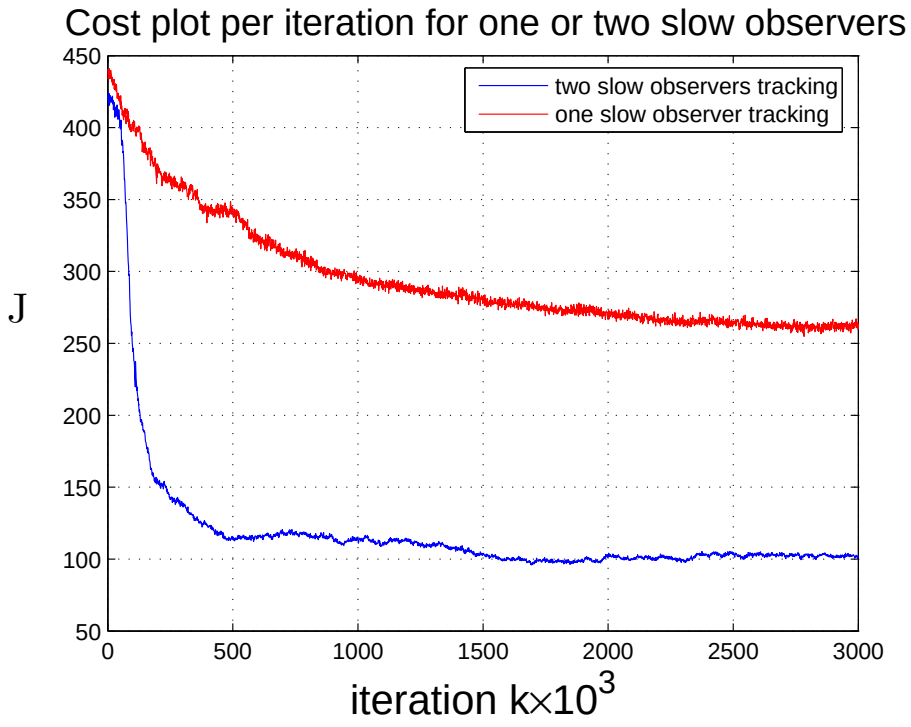
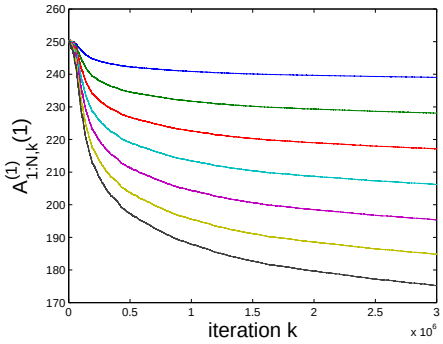
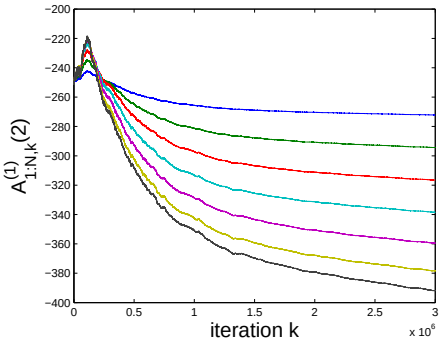


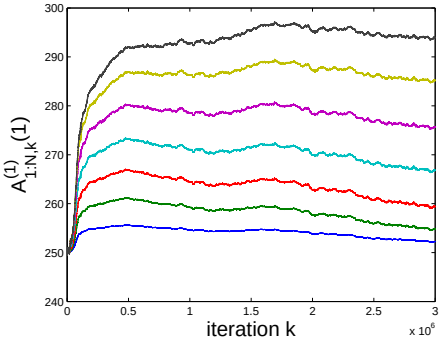
Figure 4.3: *Plot of performance criterion as actions are iterated by algorithm (4.29)-(4.32), for the single and two cooperating slow observers of Figure 4.2(a) and 4.2(b). Performance criterion was estimated using Monte Carlo simulation.*



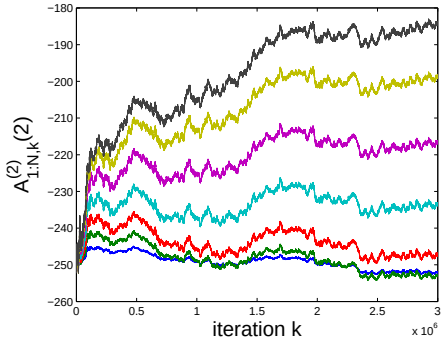
(a) Observer 1, x-axis trajectory



(b) Observer 1, y-axis trajectory



(c) Observer 2, x-axis trajectory



(d) Observer 2, y-axis trajectory

Figure 4.4: *The convergence of the trajectories $A_{1:N,k}$ computed using Algorithm (4.29)-(4.32) for the two slow cooperating observers of Figure 4.2(b) is shown. Figure (a) shows the convergence of the x coordinate of the trajectory for observer 1. The blue line is the x position for epoch 1, green for epoch 2, red for epoch 3, cyan for epoch 4 and so on. There are seven epochs in total. Figure (b) shows the convergence of the y coordinate of the trajectory for observer 1. The blue line is the y position for epoch 1, green for epoch 2, red for epoch 3, cyan for epoch 4 and so on. Figures (c) and (d) are the corresponding plots for observer 2's trajectory computed using Algorithm (4.29)-(4.32).*

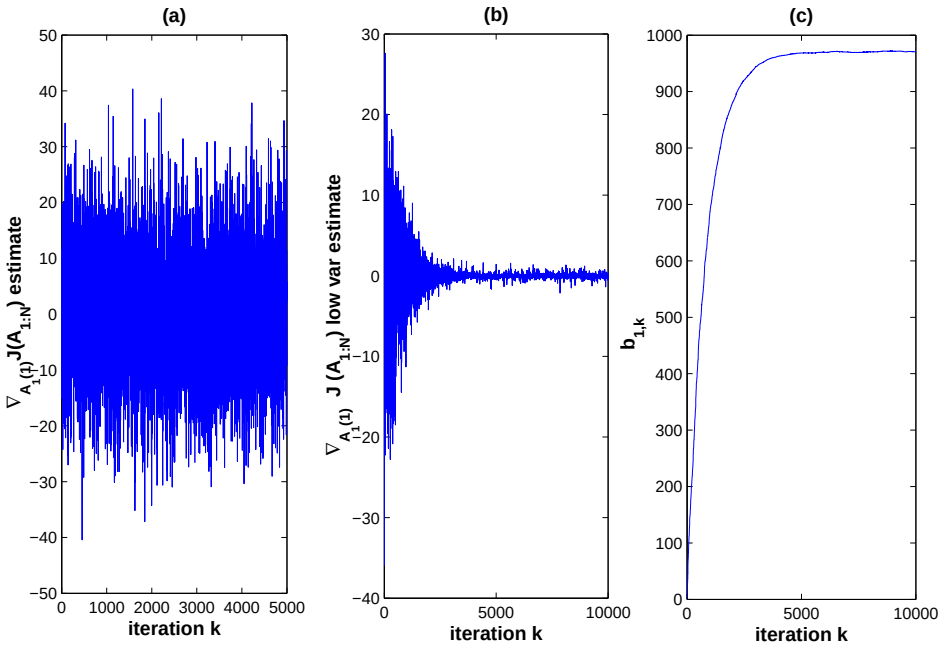


Figure 4.5: *Variance reduction by control variate: Figure 4.5(a) shows gradient estimate without control variate. Figure 4.5(b) shows the gradient estimate variance decreasing as the control variate in Figure 4.5(c) converges.*

5

Risk Sensitive Control using Policy Gradient

Summary. *In this chapter we investigate a general risk sensitive control problem. We aim to design a policy gradient implementation using Sequential Monte Carlo. Initially we use the Feynman-Kac representation of a Markov chain flow to exploit the properties of the logarithmic Lyapunov exponent. This can eventually lead to a policy gradient solution for the parameterised problem. We have also presented a particle algorithm that can be used to compute approximations when analytical expressions are not available. Finally we have shown how our algorithm can be implemented and used by means of a numerical example.*

5.1 Introduction

Consider the controlled Markov chain $\{X_n\}_{n\geq 0}$ on a state space E, with initial distribution ν and transition density $M_n(a, x, dy)$, where a denotes the action variable. Let the policy ζ of the control problem be a sequence of mappings between the state and action space, $\{\Pi_n(X_n)\}_{n\geq 0}$, and consider only the case of deterministic policies, where $A_n = \Pi_n(X_n)$. The infinite horizon risk sensitive control problem consists of minimizing with respect to all admissible policies ζ the cost criterion

$$J(\zeta) = \limsup_{n\to\infty} \frac{1}{\beta n} \log \left(\mathbb{E}_{x_0} \left[\exp \sum_{p=1}^{n} \beta V(X_p, \Pi_p(X_p)) \right] \right), \tag{5.1}$$

where β is a non-zero constant and $V(X_p, \Pi(X_p))$ is a non negative cost function. For $\beta < 0$, the policies that attempt to minimise $J(\zeta)$ are characterised as *risk averse* , otherwise for $\beta > 0$ they are *risk preferring* [169]. In [42] we see that an optimal policy ζ^* exists and is stationary, i.e. satisfies $A_n = \Pi(X_n)$.

Now assume we can parameterise a stationary policy ζ with respect to some parameter θ and obtain Π_θ, M_θ and V_θ. In this case, we could instead minimise

$$J(\theta) = \limsup_{n\to\infty} \frac{1}{\beta n} \log \left(\mathbb{E}_{x_0} \left[\exp \sum_{p=1}^{n} \beta V_\theta(X_p) \right] \right). \tag{5.2}$$

Under certain regularity and aperiodicity conditions on M_n we know from [12, 92] that this limit exists and that

$$\inf_\theta J(\theta) = \sup_\theta \Lambda(\exp(-\beta V_\theta(x))),$$

where Λ is referred as the logarithmic Lyapunov exponent, or the spectral radius of the bounded operator

$$Q'(f) = \int \exp(-\beta V_\theta(y))\, M_\theta(x, dy) f(y), \tag{5.3}$$

for any bounded measurable function f.

Most of the work so far dealing with the risk sensitive control problem is inspired by the theory of large deviations involving countable state spaces [12, 27, 92]. By taking advantage of multiplicative ergodicity of the Markov chain and the structure of the multiplicative Poisson equation, Value and Policy iteration algorithms within the Dynamic Programming concept were derived [31, 58, 75]. This approach has been generalised to general state spaces in [41, 42],

where the existence of a deterministic stationary optimal policy ζ^* is proven under mild conditions for a cost criterion like (5.1). In this chapter we shall aim to develop a policy gradient algorithm for general state spaces based on a parameterisation of the policy. This approach is influenced by the work in Reinforcement learning or Neurodynamic Programming [22], leading to policy gradient algorithm for Markov decision processes [16] for the average and discount cost criterion. There are a large number of papers for the case of discrete state and action spaces, but Sequential Monte Carlo methods have been used previously successfully for the average and discount cost criterion for the general state space cases [52]. We shall be proposing a Sequential Monte Carlo implementation for solving the risk sensitive control problem using policy gradient.

Our approach is based on the work of Del Moral and Doucet [35] involving Markov motions in absorbing media. There the long term behaviour of the Markov motion in absorbing media is analysed and the authors use Feynman-Kac distributions [34] to characterise at each time the distribution of motion conditioned on survival and the probability that motion is terminated. Under weak conditions, quantities like the operator $Q'(f)$ are shown to be linear Feynman-Kac operators and characterise the Lyapunov exponent as in equation (5.2). Assuming a suitable parameterisation of the policy exists, we can use the results of [35] to develop a stochastic approximation based algorithm to learn the parameter θ, as done in [52] for the average cost case. Subsequently, we will be able to compute the optimal policy Π_θ as in the standard the reinforcement learning approach described in [16, 22].

A similar approach has been carried out in online parameter estimation problems for Hidden Markov Models, where the long run average of the log likelihood admits a structure similar to $J(\theta)$, [135, 136]. Parallel to our work, more recently in [33] an interacting particle algorithm was developed to estimate the gradients of Feynman-Kac flows propagated in time. We aim to provide an alternative method for estimating these gradients and at the same time consider problems involving controlled Markov chains, or Markov decision processes, with a risk sensitive criterion. The idea of developing policy gradient algorithms for the infinite horizon risk sensitive Markov decision problem using Feynman-Kac models is to our best knowledge novel.

The organisation of this chapter is as follows. In Section 5.2 we formulate the decision problem using Feynman-Kac models. In Section 5.3 we develop a generic gradient algorithm to learn the parameters of the policy and present its particle implementation in Section 5.4. Finally,

in Section 5.5 we present a numerical example to show the effectiveness of our approach.

5.2 Problem Formulation

In Section 2.8 we introduced the Feynman-Kac model for a Markov chain. In this chapter, we shall attempt to use the model for the case where the Markov chain $\{X_n\}_{n\geq 0}$ is the state of a decision process, controlled by an action sequence $\{A_n\}_{n\geq 0}$, where each action $A_n \in \mathbb{A}$. In this case, the transition density changes to $M_n(a_{n,}x_{n-1}, dx_n)$, so as to show the dependence of the state X_n on the current action. In the previous section we defined Feynman-Kac models, but made no assumption on the particular structure of M_n. Therefore all the definitions and results above are eligible to be used for a controlled Markov chain, just by substituting $M_n(a_{n,}x_{n-1}, dx_n)$ instead of $M_n(x_{n-1}, dx_n)$. In this chapter we shall restrict ourselves to the fully observable Markov decision problem, i.e. the state sequence $\{X_n\}_{n\geq 0}$ is fully observable.

We will now present the problem more formally. Let $\{X_n\}_{n\geq 0}$ be a Markov process on some measurable space $(E, \mathcal{E})$ with initial distribution ν and a family of transition kernels $\{M_n\}_{n\geq 0}$ such that

$$\mathbb{P}(X_n \in dx_n | X_{0:n-1} = x_{0:n-1}, A_{1:n} = a_{1:n}) = M_n(x_{n-1}, a_n, dx_n).$$

In addition, let the policy ζ be the sequence of mappings $\{\Pi_n\}_{n\geq 0}$. The policy is called randomised if each Π_n is a kernel with domain $E \times \mathbb{A} \to \mathcal{P}(\mathbb{A})$ and such that

$$\mathbb{P}(A_n \in da | X_n = x_n, A_{n-1} = a_{n-1}) = \Pi_n(x_n, a_{n-1}, da).$$

Alternatively, we may set $\zeta = \{\Pi_n\}_{n\geq 0}$, where each Π_n is deterministic mapping from $E \to \mathbb{A}$ such that

$$A_n = \Pi_n(X_n).$$

Then the policy will be characterised as deterministic. In any case, the class of policies such that the sequence $\{\Pi_n\}_{n\geq 0}$ are time invariant, i.e. $\Pi_n = \Pi$, are called stationary. In this chapter we will consider only stationary deterministic policies.

At each time n we associate each state and action pair (X_n, A_n) with a non-negative cost $V(X_n, A_n)$. The aim of the decision problem is to find the optimal policy ζ^* such that a long term cost is minimised respectively. For example, in equation (5.1) if we let $\beta \to 0$, the resulting

long term cost to be minimised will tend to the following infinite horizon average cost

$$\limsup_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\mathbb{E}_{x_0}\left[V(X_k,\Pi_k(X_k))\right],$$

which is also referred in the literature as the *risk neutral* cost [169]. Another example of a long term risk neutral cost is the finite horizon criterion found in Chapter 4. However we shall not examine any of these cases. Instead we will consider only the case where $\beta \neq 0$ and shall be examining policies minimising the infinite horizon risk sensitive cost defined as

$$J(\zeta)=\limsup_{n\to\infty}\frac{1}{\beta n}\log\left(\mathbb{E}_{x_0}\left[\exp\sum_{p=1}^{n}\beta V(X_p,\Pi_p(X_p))\right]\right),$$

where β is a constant. So the risk sensitive control problem is to find a policy ζ^* such that

$$\zeta^*=\arg\inf_{\zeta}J(\zeta).$$

In many real world problems it is possible to parameterise the policy through a parameter $\theta\in\Theta$. In this case we can write the policy ζ as ζ_θ. Similarly the instantaneous cost $V(X_n,A_n)$ can be expressed as $V(X_n,\Pi_\theta(X_n))$ or more simply as $V_\theta(X_n)$. In the same context we write $J(\zeta)$ as

$$J(\theta)=\limsup_{n\to\infty}\frac{1}{\beta n}\log\left(\mathbb{E}_{x_0}\left[\exp\sum_{p=1}^{n}\beta V_\theta(X_p)\right]\right),$$

and transform the decision problem to a minimisation problem seeking the optimal parameter θ^*

$$\theta^*=\arg\inf_{\theta}J(\theta).$$

Similarly, the state's transition density M_n can be written as $M_n(\theta,x_{n-1},dx_n)$.

5.2.1 Properties of $J(\theta)$

For the sake of simplicity only, we will from now on assume the Markov chain $\{X_n\}_{n\geq 0}$ is time homogeneous and write the transition kernel as $M_\theta(x,dy)$ on some measurable space acting from E into E. Note that our framework can also tolerate the inhomogeneous case with the only restriction being that M_n obeys assumption (A1) , or the accessibility condition (A2) presented earlier in Section 2.8. Therefore the methodology and algorithms to be presented are can be directly applied any M_n that does not violate one of (A1) or (A2).

Inspired by [35] we represent the controlled Markov chain as a Feynman-Kac flow, which is characterised by means of a potential function G. Define the potential function as the unnormalised Boltzmann-Gibbs measure on E, taking values in $(0, 1]$,

$$G_\theta(X_n) = \exp(\beta V_\theta(X_n))$$

Since now G_n and M_n do not vary with time any more, the time subscripts for them have been omitted and instead we put θ at the subscript to explicitly denote their dependance on that parameter. Let also $S = G_\theta^{-1}((0, 1))$. In the discussion that follows we shall exploit some properties of these Feynman-Kac models found in [35], that will eventually lead to a policy gradient algorithm. Note that in this section an the remainder of this chapter the material presented in Section 2.8 will be used extensively.

We wish to consider the Feynman-Kac models for the pair (G_θ, M_θ). For any bounded measurable function $f \in B_b(E)$, from $E \to \mathbb{R}$, the distribution flows of the Feynman-Kac functional representation formula associated with the pair (G_θ, M_θ) are,

$$\eta_n(f) = \frac{\gamma_n(f)}{\gamma_n(1)} = \frac{\mathbb{E}_{x_0}\left(f(X_n)\exp(\beta\sum_{p=1}^{n-1} V_\theta(X_p))\right)}{\mathbb{E}_{x_0}\left(\exp(\beta\sum_{p=1}^{n-1} V_\theta(X_p))\right)},$$

$$\mu_n(f) = \frac{\lambda_n(f)}{\lambda_n(1)} = \frac{\mathbb{E}_{x_0}\left(f(X_n)\exp(\beta\sum_{p=1}^{n} V_\theta(X_p))\right)}{\mathbb{E}_{x_0}\left(\exp(\beta\sum_{p=1}^{n} V_\theta(X_p))\right)}.$$

So far we have not made any discussion regarding the sign of β apart from the distinction between the risk averse case for $\beta < 0$ and the risk preferring for $\beta > 0$. An important requirement for the distribution flows of the above Feynman-Kac model is G_θ to be bounded measurable functions. For the risk averse case this is not a problem as G_θ is obviously bounded and takes values in $(0, 1)$. However, this is not true in general for the risk preferring case except in the simple case when V_θ is bounded or when dealing with compact state spaces. To tackle this we shall require that Assumption (A1) is satisfied. Therefore, for $\beta > 0$ we can use the transformed Feynman-Kac model (G'_θ, M'_θ) presented in (2.29)-(2.30) and require that G'_θ is bounded instead. In the remainder of this section we shall focus more on the problem that is formulated using (G_θ, M_θ), but naturally all the results can be reproduced for the (G'_θ, M'_θ)

case. For more details, see [35]. An alternative approach is to use a different model instead of (G'_θ, M'_θ). This can be according to an Importance Sampling approach, where instead of using M_θ one uses a different kernel Q_n, and considers the Feynman-Kac model of the pair $(Q_n, \overline{G}_n)$, where $\overline{G}_n = G_\theta(x_n)\frac{dM_\theta(x_{n-1},x_n)}{dQ_n(x_{n-1},x_n)}$. In order for the model to be well defined we shall require instead that $\overline{G}_n$ is bounded. We will present more details on this approach later in this chapter. Having made this discussion, for the remainder of this chapter we shall consider the model (G_θ, M_θ) and assume that it is well defined for the problem with $G_\theta \in (0, 1)$ without worrying about the sign of β.

We shall proceed now with a few definitions in order to examine the relation of the unnormalised prediction and updated flows $\gamma_n(f)$ and $\lambda_n(f)$ with the cost criterion $J(\theta)$ to be minimised.

Definition 5.2.1 *From [34, Def. 2.7.2]. Let Q'_θ be the bounded operator on the Banach space of bounded measurable function $f \in B_b(E)$ defined for any $x \in E$ by*

$$Q'_\theta(f)(x) = \int M_\theta(x, dy)G_\theta(y)f(y) = M_\theta(G_\theta f)(x) = G'_\theta(x)M'_\theta(f)(x).$$

We denote by $Q'^{(n)}_\theta$ the semigroup on $B_b(E)$ associated to the operator Q'_θ and defined by $Q'^{(n)}_\theta = Q'^{(n-1)}_\theta Q'_\theta$. We initialise using $Q'^{(0)}_\theta = I$, where I is an appropriate identity matrix. The expression for $Q'^{(n)}_\theta$ is

$$\begin{aligned} &Q'^{(n)}_\theta(1)(x_0) \\ &= \int_{E^n} M_\theta(x_0, dx_1)G_\theta(x_1)...M_\theta(x_{n-1}, dx_n)G_\theta(x_n) \\ &= \mathbb{E}_{x_0}\left(\prod_{p=1}^{n} G_\theta(X_p)\right) \end{aligned}$$

Note that at the subscript θ of Q'_θ has been added in order to emphasise on the dependance of $Q'^{(n)}_\theta(1)$ on the parameter. When $v = \delta_{x_0}$, we have

$$Q'^{(n)}_\theta(1)(x_0) = \lambda_n(1)$$

Using (2.21)-(2.23) and (2.31)-(2.32) from Section 2.8, we get

$$\lambda_n(1) = \mathbb{E}_{x_0}\left(\prod_{p=1}^{n} G_\theta(X_p)\right) = \prod_{p=1}^{n} \eta_n(G_\theta) = \prod_{p=0}^{n-1} \eta'_n(G'_\theta).$$

Hence

$$Q_\theta'^{(n)}(1)(x_0) = \prod_{p=1}^{n} \eta_n(G_\theta) = \prod_{p=0}^{n-1} \eta_n'(G_\theta'). \tag{5.4}$$

We shall now define the spectral radius or logarithmic Lyapunov exponent of the semigroup operator Q_θ'.

Definition 5.2.2 *[35, Eqn. (3)]The logarithmic Lyapunov exponent or spectral radius of Q_θ' on $B_b(E)$ is defined by*

$$\begin{aligned}\Lambda(G_\theta) &\equiv \log Lyap(Q_\theta') \\ &= \lim_{n\to\infty} \frac{1}{n} \sup_{x_0\in S} \log Q_\theta'^{(n)}(1).\end{aligned}$$

Now, rewrite Λ as

$$\Lambda(G_\theta) = \lim_{n\to\infty} \sup_{x_0\in E} \Lambda_n(G_\theta),$$

where $\Lambda_n(G)(x_0)$ is given by

$$\Lambda_n(G_\theta) = \frac{1}{n} \log Q_\theta'^{(n)}(1).$$

In [35, p. 1184] this was combined with (5.4) to give

$$\Lambda_n(G) = \frac{1}{n} \sum_{p=1}^{n} \log \eta_n(G_\theta) = \frac{1}{n} \sum_{p=0}^{n-1} \log \eta_n'(G_\theta').$$

This definition starts to reveal that the asymptotical properties of the semigroup $Q_\theta'^{(n)}$ closely match the structure of $J(\theta)$, which can be written as

$$J(\theta) = \limsup_{n\to\infty} \beta^{-1} \Lambda_n(G).$$

Note that since $G_\theta \in (0,1)$, we have $\Lambda_n \in (-\infty, 0)$. It is therefore clear that the minimisers of J with respect to θ are the same for any x_0 and therefore coincide with the maximisers of Λ giving

$$\theta^* = \arg\inf_\theta J(\theta) = \arg\sup_\theta \Lambda(G_\theta).$$

The problem has now been transformed into a problem of finding parameter θ^* that maximises Λ instead of minimising J directly. For more details on the properties of logarithmic Lyapunov exponent and other similar type operators we refer the reader to [36]. In [35] we find the fundamental theoretical justification for our approach, giving asymptotic results and

particle approximations. We reproduce a theorem that shows both existence of Λ and investigates the how long term behaviour of the Feynman-Kac flow μ_n can be used to approximate it.

Theorem 5.2.1 *From [35, p. 1185]. Assume that there exists an integer parameter $m \geq 1$ and a pair $(r', \delta') \in [0, 1]$ such that for each $x, y \in S$,*

$$G'_\theta(x) \geq r' G'_\theta(y) \text{ and } Q'^{(m)}_\theta(x, \cdot) \geq \delta' Q'^{(m)}_\theta(y, \cdot).$$

Then there exists a unique distribution $\mu_\infty \in \mathcal{P}(S)$ such that

$$\mu_n \xrightarrow[n\to\infty]{a.s.} \mu_\infty.$$

Also, μ_∞ is such that

$$\Lambda(G_\theta) = \log \mu_\infty M_\theta(G_\theta), \tag{5.5}$$

and for any $f \in B_b(S)$

$$\mu_\infty M_\theta(G_\theta f) = e^{\Lambda(G_\theta)} \mu_\infty(f). \tag{5.6}$$

In addition we have the uniform estimates

$$\sup_{x_0 \in S} ||\mu_n - \mu_\infty||_{TV} \leq \frac{2}{\delta'^2}(1 - \delta'^2)^{[\frac{n}{m}]},$$
$$\sup_{x_0 \in S} |\Lambda_n(G_\theta) - \Lambda(G_\theta)| \leq \frac{2m}{r' \delta'^3} \frac{1}{n},$$
$$\sup_{x_0 \in S} |\log \mu_n M_\theta(G_\theta) - \Lambda(G_\theta)| \leq \frac{4}{\delta'^2}(1 - \delta'^2)^{[\frac{n}{m}]}.$$

The particular Feynman-Kac flow μ_n leads to a stationary distribution μ_∞, which in turn guarantees that

$$\Lambda(G_\theta) = \lim_{n\to\infty} \Lambda_n(G_\theta).$$

We refer the interested reader to [35] for a particle methods to approximate $\Lambda_n(G_\theta)$ and asymptotic convergence results. As it is not of our interest to approximate $\Lambda_n(G_\theta)$ directly, but instead to maximise it we omit further discussion on that topic. Finally, it is important to add that the assumption of Theorem 5.2.1 can be replaced with similar condition for M_θ instead of that for the operator Q'_θ. This condition is that there exists $\varphi' \in [0, 1]$ such that for each $x, y \in S$,

$$M'_\theta(\, Q'^{(m)}_\theta)(x, \cdot) \geq \varphi' M'_\theta(Q'^{(m)}_\theta)(y, \cdot).$$

In the next section we will propose a gradient method to find the maximiser of $\Lambda(G_\theta)$ that could be regarded as a general way for performing policy gradient for risk sensitive control problems.

5.3 Policy Gradient Search for the Policy's Parameter θ

So far we have examined the long time behaviour of a risk sensitive controlled Markov chain assuming one can obtain a suitable parameterisation for the policy ζ. In this section we shall see how the optimal policy ζ^* can be computed. To do so, we shall propose a stochastic gradient method on the parameter space Θ. This method is based on the work done in [136] for parameter estimation for general state space models using Recursive Maximum Likelihood. Our criterion to be maximised $\Lambda(G_\theta)$ admits very similar properties as the recursive likelihood of [136], which enables the use of stochastic approximation for an gradient ascent algorithm.

It is important to note that having obtained (5.6)-(5.5) the effect of x_0 can be ignored and we can rewrite $\Lambda(G_\theta)$ as

$$\Lambda(G_\theta) = \lim_{n\to\infty} \Lambda_n(G_\theta),$$

where $\Lambda_n(G_\theta) = \frac{1}{n}\sum_{p=1}^{n} \log \eta_n(G_\theta)$. Using same ergodicity assumptions as in [157] the authors in [35] show that

$$\Lambda(G_\theta) = \log(\eta_\infty(G_\theta)),$$

where $\eta_\infty = \mu_\infty M_\theta$. Under additional regularity assumptions, the assumptions and results found in [157] can be extended for defining the gradient of the limit as

$$\begin{aligned}\nabla_\theta \Lambda(G_\theta) &= \lim_{n\to\infty} \nabla_\theta \Lambda_n(G_\theta) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{p=1}^{n} \nabla_\theta \log \eta_n(G_\theta).\end{aligned}$$

In [157] the authors also show that $\nabla_\theta \log \eta_n(G_\theta)$ is an asymptotically unbiased estimate of $\nabla_\theta \Lambda(G_\theta)$. This allows us to use $\nabla_\theta \Lambda(G_\theta)$ propose to perform a stochastic gradient search on the space of $\theta \in \Theta$ using stochastic approximation,

$$\theta_{n+1} = \theta_n + \alpha_n [\nabla_\theta \log \eta_n(G_\theta)]_{\theta=\theta_n}.$$

where the step-size α_n is a non-increasing positive sequence tending to zero.

As we rely on a gradient search to compute the parameter of the optimal policy, θ^*, this is a policy gradient method. For discrete state-space, everything can be done analytically. For continuous state-spaces we can see that the iteration can be written as

$$\theta_{n+1} = \theta_n + \alpha_n [\frac{\nabla_\theta \eta_n(G_\theta)}{\eta_n(G_\theta)}]_{\theta=\theta_n},$$

and it may not be possible to solve analytically the integrals found in $\frac{\nabla_\theta \eta_n(G_\theta)}{\eta_n(G_\theta)}$. We shall resort to Sequential Monte Carlo methods and use an interacting particle algorithm to approximate these expressions. Before presenting a complete algorithm for solving the parameterised problem, we first have to consider how to propagate the gradient of the flows $\mu_n(f)$ and $\eta_n(f)$ with respect to θ.

5.3.1 Propagating the Gradient of a Feynman Kac Flow

In this section we shall consider how to compute the gradients of the Feynman-Kac representation. The problem of computing gradients of distributions or measures has already received a lot of attention, but mostly in the literature dealing with parameter estimation for hidden Markov models. In [136] in the context of online maximum likelihood parameter estimation, the long run average of the log likelihood admits a structure similar to $J(\theta)$ and there has been a detailed study on how to approximate the gradients of optimal filters. In the control literature, for the case of infinite average cost criterion and Partially Observed Markov Decision Processes, in [52] we have seen how the gradient of the optimal filter has been approximated in a policy gradient algorithm.

More recently there have been developments to extend the ideas found in these papers for the case of Feynman-Kac flows. In [33] we find two ways to approximate the gradient of the flows $\mu_n(f)$ and $\eta_n(f)$ with respect to a parameter θ. One uses an infinitesimal perturbation analysis (IPA) method and the other a score method. We will first illustrate the core of their ideas and then proceed to propose a different approach that extends the work of [136].

Recently proposed methods

It is of interest to find a method to approximate $\nabla_\theta \eta_n(f)$, so we have that

$$\nabla_\theta \eta_n(f) = \nabla_\theta \frac{\gamma_n(f)}{\gamma_n(1)} = \frac{\nabla_\theta \gamma_n(f)}{\gamma_n(1)} - \eta_n(f)\frac{\nabla_\theta \gamma_n(1)}{\gamma_n(1)}.$$

Therefore calculating the gradient of the normalised flow η_n reduces to the calculation of the unnormalised flow $\nabla_\theta \gamma_n(f)$. We write

$$\nabla_\theta \gamma_n(f) = \nabla_\theta \mathbb{E}_\nu \left(f(X_n) \prod_{p=0}^{n-1} G_\theta(X_p) \right)$$

$$= \mathbb{E}_\nu \left(\underbrace{[\nabla_\theta f(X_n) + \nabla_{x_n} f(X_n) \nabla_\theta X_n]}_{Z_n} \prod_{p=0}^{n-1} G_\theta(X_p) \right)$$

$$+ \mathbb{E}_\nu \left(f(X_n) \underbrace{[\sum_{p=0}^{n-1} \frac{\nabla_{x_n} G_\theta(X_p) \nabla_\theta X_p + \nabla_\theta G_\theta(X_p)}{G_\theta(X_p)}]}_{R_n} \prod_{p=0}^{n-1} G_\theta(X_p) \right)$$

In [33] an Infinitesimal Perturbation Analysis (IPA) approach is used to compute Z_n and R_n for each particle in order to use them for an approximation of $\nabla_\theta \eta_n(f)$. We also find in that paper an alternative method using a score approach, [131]. For the score method let us further assume M_θ poses a particular structure suited for many examples and that $M_\theta(x_{n-1}, dx_n) = m_\theta(x_{n-1}, x_n) h(dx_n)$. Starting from

$$\nabla_\theta \gamma_n(f) = \nabla_\theta \int_{E^{n+1}} \nu(dx_0) G_\theta(x_0) M_\theta(x_0, dx_1) G_\theta(x_1) ... G_\theta(x_{n-1}) M_\theta(x_{n-1}, dx_n) f(x_n),$$

and assuming that all functions are smooth so that we can interchange the integral with the derivative operators, we get

$$\begin{aligned}
&\nabla_\theta \gamma_n(f) \\
&= \int_{E^{n+1}} (\sum_{p=0}^{n-1} [\frac{\nabla_\theta G_\theta(x_p)}{G_\theta(x_p)} + \frac{\nabla_\theta m_\theta(x_p, x_{p+1})}{m_\theta(x_p, x_{p+1})}] + \frac{\nabla_\theta f(x_n)}{f(x_n)}) \\
&\times \nu(dx_0) G_\theta(x_0) M_\theta(x_0, dx_1) ... G_\theta(x_{n-1}) M_\theta(x_{n-1}, dx_n) f(x_n) \\
&= \mathbb{E}_\nu(\varphi(X_{0:n}) \prod_{p=0}^{n-1} G_\theta(X_p))
\end{aligned}$$

where $\varphi(X_{0:n}) = f(X_n) \sum_{p=0}^{n-1} [\frac{\nabla_\theta G_\theta(X_p)}{G_\theta(X_p)} + \frac{\nabla_\theta m_\theta(X_p, X_{p+1})}{m_\theta(X_p, X_{p+1})}] + \nabla_\theta f(X_n)$ is the score. Again an unbiased estimate of the score can be computed for each particle that is used to approximate $\nabla_\theta \eta_n(f)$. Note both these approaches, although they both use a recursive method to approximate $\nabla_\theta \eta_n(f)$, they inherently rely on the path of $X_{0:n}$ and so this might bring up the degeneracy issue concerning the variance of particle methods used to approximate path integrals [47]. Furthermore we remark that in the context of [16] in order to prevent the variance of the gradients from growing infinitely large, a discount factor will have to be added to their algorithm at the cost of adding a bias. This discount factor approach has already been used in [52] successfully for the infinite horizon average cost case.

Our proposed method

Although the attempt to approximate gradients of Feynman Kac flows in [33] seems to be a computationally attractive approach, it is clear from the previous work in [16, 52] that it would not be possible to implement it for infinite horizon problems without using an appropriate discount factor, which introduces bias in the approximation. In addition, numerical comparisons using a finite but large horizon for parameter estimation examples[1] between the methods in [33] and the method of [136] show that the latter is more accurate, although computationally more demanding. We will therefore adopt the approach of [136] and extend it to the Feynman-Kac framework. We feel in general that method suits our framework better and is more generic.

Our method essentially is to propagate in time the gradients of flows $\nabla_\theta \mu_n(f)$ and $\nabla_\theta \eta_n(f)$ as well as $\mu_n(f)$ and $\eta_n(f)$. This should can be done in parallel as illustrated below:

$$\begin{array}{ccccc} \eta_n(f) & \overset{\text{update}}{\longrightarrow} & \mu_n(f) & \overset{\text{prediction}}{\longrightarrow} & \eta_{n+1}(f) \\ \nabla_\theta \eta_n(f) & & \nabla_\theta \mu_n(f) & & \nabla_\theta \eta_{n+1}(f) \end{array}$$

Initially we remark that

$$\nabla_\theta \mu_n(f) = \nabla_\theta \int_E \mu_n(dy) f(y)$$

and assuming that all functions are smooth and continuous, we can change the order of the integral and the differentiation operator to get

$$\nabla_\theta \mu_n(f) = \int_E \nabla_\theta \mu_n(dy) f(y) + \int_E \mu_n(dy) \nabla_\theta f(y). \tag{5.7}$$

This means that in order to propagate the flow it is sufficient to propagate only the signed measure $\nabla_\theta \mu_n(dx)$. Similarly the same holds for $\nabla_\theta \eta_n(dx)$. Moreover, as far as the computation of $\nabla \Lambda_n(G_\theta)$ is concerned, which was the motivation for propagating the gradients in the first place, we observe that

$$\nabla \Lambda_n(G_\theta) = \frac{\nabla_\theta \eta_n(G_\theta)}{\eta_n(G_\theta)} = \frac{\nabla_\theta \mu_n(1)}{\mu_n(1)}.$$

Then it seems that for our problem, propagating just the measures $\nabla_\theta \eta_n(dx)$ and $\nabla_\theta \mu_n(dx_n)$ and not the entire gradients of flows $\nabla_\theta \mu_n(f)$ and $\nabla_\theta \eta_n(f)$ for some defined function f is sufficient. Of course, equation (5.7) is useful for the sake of completeness and generality.

[1] The horizon size is determined by the number of observations used to estimate the static parameter sequentially.

To implement the recursion for the gradients, we can take advantage of (2.27) and differentiate with respect to θ,

$$\begin{aligned}\nabla_\theta \eta_n &= \nabla_\theta \Phi_\eta(\eta_{n-1}) \\ &= \nabla_\theta \Psi_{n-1}(\eta_{n-1}) M_\theta + \Psi_{n-1}(\eta_{n-1}) \nabla_\theta M_\theta,\end{aligned}$$

where $\nabla_\theta \eta_n$ stands for the signed measure $\nabla_\theta \eta_n(dx)$. We could present this recursion as a one step procedure but for the sake of clarity we shall follow the two step pattern shown earlier. So we separate Φ_η to its prediction and update steps, and rewrite the gradients. For the update step we start by assuming that $\nabla_\theta \eta_n$ is available and then derive $\nabla_\theta \mu_n$ (or $\nabla_\theta \mu_n(dx)$) as a function of G_θ, μ_n, η_n, $\nabla_\theta \eta_n$.

$$\begin{aligned}\nabla_\theta \mu_n &= \nabla_\theta \Psi_n(\eta_n) \\ &= \nabla_\theta \left(\frac{G_\theta(x_n)\eta_n}{\eta_n(G_\theta)} \right) \\ &= \frac{\nabla_\theta G_\theta(x_n)\eta_n + G_\theta(x_n)\nabla_\theta \eta_n - \mu_n(dx_n)\left(\eta_n(\nabla_\theta G_\theta) + \nabla_\theta \eta_n(G_\theta)\right)}{\eta_n(G_\theta)}.\end{aligned}$$

This expression can be also written as

$$\nabla_\theta \mu_n(dx_n) = \frac{\nabla_\theta \pi_n(dx_n)}{\int\limits_E \pi_n(dx_n)} - \pi_n(dx_n) \frac{\int\limits_E \nabla_\theta \pi_n(dx_n)}{\int\limits_E \pi_n(dx_n)}, \tag{5.8}$$

where $\pi_n = G_\theta(x_n)\eta_n$ and $\int\limits_E \pi_n(dx_n) = \eta_n(G_\theta)$. We shall be using this expression later to derive smooth approximations for $\nabla_\theta \mu_n$.

Similarly, for the prediction step we use $\nabla_\theta \mu_n$ that has just been computed to derive $\nabla_\theta \eta_{n+1}$ as a function of G_θ, μ_n, η_n, $\nabla_\theta \mu_n$.

$$\begin{aligned}\nabla_\theta \eta_{n+1} &= \nabla_\theta(\mu_n M_\theta) \\ &= (\nabla_\theta \mu_n) M_\theta + \mu_n \nabla_\theta M_\theta\end{aligned}$$

Note that using the notation $\nabla_\theta \eta_n$ and $\nabla_\theta \mu_n$ may consist of some abuse, so to avoid any confusion we rewrite the equations properly with some integrals expressed in full.

$$\begin{aligned}&\nabla_\theta \mu_n(dx_n) \\ &= \mu_n(dx_n) \left(\frac{\nabla_\theta G_\theta(x_n)}{G_\theta(x_n)} + \frac{\nabla_\theta \eta_n(x_n)}{\eta_n(x_n)} - \frac{\eta_n(\nabla_\theta G_\theta)}{\eta_n(G_\theta)} - \int\limits_E \frac{G_\theta(x_n)}{\eta_n(G_\theta)} \nabla_\theta \eta_n(dx_n) \right)\end{aligned}$$

$$= \mu_n(dx_n)\left[\left(\frac{\nabla_\theta G_\theta(x_n)}{G_\theta(x_n)} + \frac{\nabla_\theta \eta_n(x_n)}{\eta_n(x_n)}\right) - \mu_n\left(\frac{\nabla_\theta G_\theta(x_n)}{G_\theta(x_n)} + \frac{\nabla_\theta \eta_n(x_n)}{\eta_n(x_n)}\right)\right] \tag{5.9}$$

$$\nabla_\theta \eta_{n+1}(dx_{n+1})$$

$$= \int_E \nabla_\theta \mu_n(dx_n) M_\theta(x_n, dx_{n+1}) + \int_E \mu_n(dx_n) \nabla_\theta M_\theta(x_n, dx_{n+1}) \tag{5.10}$$

Of course, if these equations could be solved analytically there would be no reason to use any form of approximation. Unfortunately the integrals in equations (5.9)-(5.10) do not have closed form expressions in general. The same holds for the methods in [33] presented earlier. Therefore, we shall use equations (5.9)-(5.10) to derive particle approximations for $\nabla_\theta \mu_n$, $\nabla_\theta \eta_n$, and $\nabla \Lambda_n(G_\theta)$. This will lead to a particle algorithm for policy gradient for parameterised risk sensitive control problems.

5.4 Particle Approximations for Policy Gradient Search

In this section we shall derive particle approximations for the distribution flows and their gradients described earlier in Section 5.3. We will base these approximations on the particle algorithm described on Section 2.8.4, but will modify this to address a couple of limitations. The first is that in general it might not be possible in some cases to sample from M_θ. Secondly it is possible that an a particle approximation of $\nabla \Lambda_n(G_\theta)$ could suffer from high variance when a low number of samples is used. Both of these limitations indicate that a particle system based on Importance Sampling should be proposed. The main idea would be that instead of sampling from $M_\theta(x_n, \cdot)$, we could use a proposal kernel $Q_{n+1}(x_n, \cdot)$ proportional to $M_\theta(x_n, x_{n+1}) G_\theta(x_{n+1})$ or some distribution as close as possible to it. This would improve the approximations to the flows and their gradients without altering in any fashion how we use these for searching in the parameter's space to improve our policy.

Of course, this means we should also modify the recursion to propagate $\widehat{\eta}_n$ and $\widehat{\mu}_n$ as seen in Section 2.8.3. As before, the particle system is initialised by sampling N independent samples or particles, from common initial density ν,

$$\xi_0^i \sim \nu(\cdot),$$

where i is the sample's index and $1 \leq i \leq N$. At each time n we have a set of N independent, uniformly weighted, particles ξ_n^i. Define the discrete collection of particles as $\xi_n = (\xi_n^1, ..., \xi_n^N)$, where $\xi_n \in E_n^N$.

We will now present the recursion from time n to time $n+1$. At time n, the prediction flow approximation $\widehat{\eta}_n$ is given from the previous iteration and takes the form

$$\widehat{\eta}_n(dx_n) = \sum_{i=1}^{N} \rho_n^i \delta_{\xi_n^i}(dx_n).$$

Then we can derive an approximation to use for the distribution flow μ_n as

$$\widehat{\mu}_n(dx_n) = \Psi_n(\widehat{\eta}_n) = \sum_{i=1}^{N} \widehat{\rho}_n^i \delta_{\xi_n^i}(dx_n),$$

where

$$\widehat{\rho}_n^i = \frac{G(\xi_n^i)\rho_n^i}{\sum_{k=1}^{N} G(\xi_n^k)\rho_n^k}. \tag{5.11}$$

To propagate the particles ξ_n further we shall use the following Markov transition in the same spirit as (2.33) in Section 2.8.3

$$\begin{aligned}\mathbb{P}_\nu(\xi_{n+1} \in dx_n^{1:N} | \xi_n) &= \mathcal{K}_{n+1,\widehat{\eta}_n}(\xi_n, dx_{n+1}^{1:N}) \\ &= \int \mathcal{S}_{n,\widehat{\eta}_n}(\xi_n, dx_n^{1:N}) \mathcal{Q}_{n+1}(x_n^{1:N}, dx_{n+1}^{1:N}).\end{aligned}$$

The transition kernel $\mathcal{K}_{n+1,\widehat{\eta}_n}$ for the Markov chain $\{\xi_n\}_{n\geq 0}$ is decomposed to an update and prediction step resembled by $\mathcal{S}_{n,\widehat{\eta}_n}$ and $\mathcal{Q}_{n+1}$ respectively. These are given by

$$\begin{aligned}\mathcal{S}_{n,\widehat{\eta}_n}(\xi_n, dx_n^{1:N}) &= \prod_{i=1}^{N} S_{n,\widehat{\eta}_n}(\xi_n^i, dx_n^i), \\ \mathcal{Q}_{n+1}(x_n^{1:N}, dx_{n+1}^{1:N}) &= \prod_{i=1}^{N} Q_{n+1}(x_n^i, dx_{n+1}^i)\end{aligned}$$

For the selection kernel in (2.36) and (2.37)we set $\varepsilon_n = 0$ and have

$$S_{n,\widehat{\eta}_n}(\xi_n^i, dx_n^i) = \Psi_n(\widehat{\eta}_n) = \sum_{j=1}^{N} \frac{G(\xi_n^j)\rho_n^j}{\sum_{k=1}^{N} G(\xi_n^k)\rho_n^k} \delta_{\xi_n^j}(dx_n^j).$$

Compared to Section 2.8.3, we now use an instrumental kernel $Q_{n+1}(x_n^i, dx_{n+1}^i)$ for each particle, which aims to resemble the properties of $M_\theta(x_n, x_{n+1})G_\theta(x_{n+1})$ closely. For the general inhomogeneous case with transition density M_{n+1}, Q_{n+1} has to be such that the Radon-Nikodym derivative $\frac{dM_n}{dQ_n}$ of $(M_n)_{n\geq 0}$ with respect to $(Q_n)_{n\geq 0}$ is well defined obeying

$$\forall n \geq 0, \forall x_n \in E_n, Q_{n+1}(x_n, \cdot) \ll M_{n+1}(x_n, \cdot).$$

We can then sample the next particle sets according to $\widehat{\xi}_n^i \sim S_{n,\widehat{\eta}_n}(\xi_n^i, \cdot)$ and then $\xi_{n+1}^i \sim Q_{n+1}(\widehat{\xi}_n^i, \cdot)$ to approximate μ_n and η_{n+1} respectively as

$$\widehat{\eta}_{n+1}(dx_{n+1}) = \sum_{i=1}^{N} \rho_{n+1}^i \delta_{\xi_{n+1}^i}(dx_{n+1}),$$

where

$$\rho_{n+1}^i = \frac{1}{N} \frac{dM_\theta}{dQ_{n+1}}(\widehat{\xi}_n^i, \xi_{n+1}^i).$$

Before proceeding with deriving approximations for the propagation of the gradient measures seen in (5.9)-(5.10) we shall take the opportunity to comment on how the choice of Q_{n+1} affects the risk sensitive problem for the risk preferring case, where $\beta > 0$. When modelling the propagation of measures η_n, μ_n, one has to be careful when selecting each Q_n so that the flow is properly defined, especially for the risk preferring case. We can pose the resulting controlled Markov chain with transition kernel Q_n as a Feynman-Kac model with the associated pair being $(Q_n, \overline{G}_n)$, where $\overline{G}_n = G_\theta(x_n)\frac{dM_\theta(x_{n-1},x_n)}{dQ_n(x_{n-1},x_n)}$. In order for the model to be well defined we require that $\frac{dM_\theta(x_{n-1},x_n)}{dQ_n(x_{n-1},x_n)}$ is well defined, i.e. $Q_n(x_{n-1}, \cdot) \ll M_\theta(x_{n-1}, \cdot)$ and $\overline{G}_n$ is bounded. This should be taken into account when designing Q_n for particular problems. Also, all the particle approximations in this chapter can be used in the same fashion to derive particle approximations for the measure flow defined by $(Q_n, \overline{G}_n)$.

For the propagation of the gradient approximations, we assume that a particle approximation $\widehat{\nabla_\theta \eta}_n(dx_n)$ is available for the recursion at time n,

$$\widehat{\nabla_\theta \eta}_n(dx_n) = \sum_{i=1}^{N} \rho_n^i b_n^i \delta_{\xi_n^i}(dx_n).$$

First we will consider how obtain a particle approximation gradient of the update measure, $\nabla_\theta \mu_n$. We will use equation (5.8) to derive a smooth approximations for π_n and $\nabla_\theta \pi_n$. Let $\widetilde{\pi_n}$ and $\widetilde{\nabla_\theta \pi_n}$ denote the pointwise approximations resulting from substituting $\widehat{\eta}_n$ and $\widehat{\nabla_\theta \eta}_n$ instead of η_n and $\nabla_\theta \eta_n$ in the following expressions for π_n and $\nabla_\theta \pi_n$

$$\pi_n = G_\theta(x_n)\eta_n, \nabla_\theta \pi_n = \nabla_\theta G_\theta(x_n)\eta_n + G_\theta(x_n)\nabla_\theta \eta_n.$$

This gives for $\widetilde{\pi_n}$ and $\widetilde{\nabla_\theta \pi_n}$

$$\widetilde{\pi_n}(dx_n) = \sum_{i=1}^{N} G_\theta(x_n)\rho_n^i \delta_{\xi_n^i}(dx_n), \tag{5.12}$$

$$\widetilde{\nabla_\theta \pi_n} = \sum_{i=1}^{N} \left(\nabla_\theta G_\theta(x_n)\rho_n^i + G_\theta(x_n)\rho_n^i b_n^i\right) \delta_{\xi_n^i}(dx_n), \tag{5.13}$$

and for the corresponding integrals we have

$$\int_E \widetilde{\pi_n}(dx_n) = \sum_{k=1}^{N} G(\xi_n^k)\rho_n^k, \tag{5.14}$$

$$\int_E \widetilde{\nabla_\theta \pi_n}(dx_n) = \sum_{k=1}^{N} \left(\nabla_\theta G_\theta(\xi_n^k)\rho_n^k + G_\theta(\xi_n^k)\rho_n^k b_n^k \right). \tag{5.15}$$

Substituting (5.12)-(5.15) and (5.11) into (5.8) leads to the following pointwise approximation for $\nabla_\theta \mu_n$

$$\begin{aligned}
\widetilde{\nabla_\theta \mu_n} &= \frac{\widetilde{\nabla_\theta \pi_n}(dx_n)}{\int_E \widetilde{\pi_n}(dx_n)} - \widetilde{\pi_n}(dx_n) \frac{\int_E \widetilde{\nabla_\theta \pi_n}(dx_n)}{\int_E \widetilde{\pi_n}(dx_n)} \\
&= \sum_{i=1}^{N} \left(\frac{G_\theta(x_n)\rho_n^i}{\sum_{k=1}^{N} G(\xi_n^k)\rho_n^k} - \left(\nabla_\theta G_\theta(x_n)\rho_n^i + G_\theta(x_n)\rho_n^i b_n^i \right) \frac{\int_E \widetilde{\nabla_\theta \pi_n}(dx_n)}{\int_E \widetilde{\pi_n}(dx_n)} \right) \delta_{\xi_n^i}(dx_n).
\end{aligned}$$

Evaluating at the particle set ξ_n gives the following particle approximation

$$\widehat{\nabla_\theta \mu_n}(dx_n) = \sum_{i=1}^{N} \widehat{b}_n^i \widehat{\rho}_n^i \delta_{\widehat{\xi}_n^i}(dx_n),$$

where

$$\widehat{b}_n^i = \left(\frac{\nabla_\theta G_\theta(\xi_n^i)}{G_\theta(\xi_n^i)} + b_n^i \right) - \sum_{j=1}^{N} \widehat{\rho}_n^i \left(\frac{\nabla_\theta G_\theta(\xi_n^j)}{G_\theta(\xi_n^j)} + b_n^j \right).$$

Note that this resembles (5.9) since one can show that $b_n^i = \widetilde{\frac{\nabla_\theta \eta_n}{\eta_n}}(\xi_n^i)$, where $\widetilde{\frac{\nabla_\theta \eta_n}{\eta_n}}$ is the pointwise approximation of $\frac{\nabla_\theta \eta_n}{\eta_n}$.

We can apply the same procedure for the gradient of the prediction measure, $\nabla_\theta \eta_{n+1}(dx_{n+1})$. Using (5.10) the resulting smooth approximation will be

$$\int_E \widetilde{\nabla_\theta \mu_n}(dx_n) M_\theta(x_n, dx_{n+1}) + \int_E \widetilde{\mu_n}(dx_n) \nabla_\theta M_\theta(x_n, dx_{n+1}),$$

where $\widetilde{\mu_n}(dx_n) = \frac{\widetilde{\pi_n}(dx_n)}{\int_E \widetilde{\pi_n}(dx_n)}$. Note that now we have used Q_{n+1} to generate particle set ξ_{n+1}, so using Importance Sampling we write $\widetilde{\nabla_\theta \eta_{n+1}}(dx_{n+1})$ as

$$\widetilde{\nabla_\theta \eta_{n+1}}(dx_{n+1}) = \int_E \widetilde{\nabla_\theta \mu_n}(dx_n) \frac{dM_\theta}{dQ_{n+1}}(x_n, dx_{n+1}) + \int_E \widetilde{\mu_n}(dx_n) \frac{d(\nabla_\theta M_\theta)}{dQ_{n+1}}(x_n, dx_{n+1}).$$

Evaluating at the particle set ξ_n gives the following particle approximation

$$\widehat{\nabla_\theta \eta}_{n+1}(dx_{n+1}) = \sum_{i=1}^{N} b_{n+1}^i \rho_{n+1}^i \delta_{\xi_{n+1}^i}(dx_{n+1}),$$

where the weights are given by

$$b_{n+1}^i \rho_{n+1}^i = \sum_{j=1}^{N} \widehat{b}_n^j \widehat{\rho}_n^j \frac{M_\theta(\widehat{\xi}_n^j, \xi_{n+1}^i)}{Q_{n+1}(\widehat{\xi}_n^j, \xi_{n+1}^i)} + \sum_{j=1}^{N} \widehat{\rho}_n^j \frac{\nabla_\theta M_\theta(\widehat{\xi}_n^j, \xi_{n+1}^i)}{Q_{n+1}(\widehat{\xi}_n^j, \xi_{n+1}^i)}.$$

Finally, we are now ready to obtain an estimate of the gradient of $\nabla \Lambda_n(G_\theta)$, given by

$$\widehat{\nabla \Lambda}_n(G_\theta) = \frac{\widehat{\nabla_\theta \mu}_n(1)}{\widehat{\mu}_n(1)} = \frac{\sum_{i=1}^{N} \widehat{b}_n^i \widehat{\rho}_n^i}{\sum_{i=1}^{N} \widehat{\rho}_n^i}.$$

5.4.1 Particle Based Policy Gradient Algorithm

We summarise all the approximations in this section to present the following algorithm, which can be used to solve the risk sensitive control problem using particle methods, when a suitable parameterisation of the policy is possible.

Algorithm 5.1 ***Policy Gradient for Risk Sensitive Control using particle methods:***

Initialisation $\xi_0^i \sim \nu(\cdot)$

For each time n , proceed with the following steps.

Update distribution flow *For* $i = 1, .., N$

- *Compute weights* $\widehat{\rho}_n^i = \frac{G(\xi_n^i)\rho_n^i}{\sum_{k=1}^{N} G(\xi_n^k)\rho_n^k}$,

  $\widehat{b}_n^i = \left(\frac{\nabla_\theta G_\theta(\xi_n^i)}{G_\theta(\xi_n^i)} + b_n^i\right) - \sum_{j=1}^{N} \widehat{\rho}_n^i \left(\frac{\nabla_\theta G_\theta(\xi_n^j)}{G_\theta(\xi_n^j)} + b_n^j\right).$

Selection *For* $i = 1, .., N$

- *Sample* $\widehat{\xi}_n^i \sim \sum_{j=1}^{N} \frac{G_{\theta_n}(\xi_n^j)}{\sum_{k=1}^{N} G_{\theta_n}(\xi_n^k)} \delta_{\xi_n^j}(dx_n^j)$.

Update θ

- $\theta_{n+1} = \theta_n + \alpha_n \frac{\sum_{i=1}^{N} \widehat{b}_n^i \widehat{\rho}_n^i}{\sum_{i=1}^{N} \widehat{\rho}_n^i}$.

Mutation *For* $i = 1, .., N$

- *Sample* $\xi_{n+1}^i \sim Q_{n+1}(\widehat{\xi}_n^i, \cdot)$,
- *Compute weights* $\rho_{n+1}^i = \frac{1}{N} \frac{M_\theta(\widehat{\xi}_n^i, \xi_{n+1}^i)}{Q_{n+1}(\widehat{\xi}_n^i, \xi_{n+1}^i)}$,
  $b_{n+1}^i \rho_{n+1}^i = \sum_{j=1}^N \widehat{b}_n^j \widehat{\rho}_n^j \frac{M_\theta(\widehat{\xi}_n^j, \xi_{n+1}^i)}{Q_{n+1}(\widehat{\xi}_n^j, \xi_{n+1}^i)} + \sum_{j=1}^N \widehat{\rho}_n^j \frac{\nabla_\theta M_\theta(\widehat{\xi}_n^j, \xi_{n+1}^i)}{Q_{n+1}(\widehat{\xi}_n^j, \xi_{n+1}^i)}$..

Continue until convergence.

Typically, the step-sizes for the iteration of θ are $\alpha_k = k^{-\alpha}$,where $\alpha > 0.5$. Note that he have chosen $\varepsilon_n = 0$ in the selection step, which results to a standard resampling mechanism. The approximations found in the algorithm require $\mathcal{O}(N^2)$ amount of computation, which is quite high, but on the other hand the computation can be parallelised as in most cases when SMC is used. We refer the reader to [86] for a treatment on how to implement fast SMC algorithms with $\mathcal{O}(N^2)$ computation. We would like to remind the reader that during the formulation of the problem and derivation of the particle approximations, we have not made any restrictive assumptions for the state evolution, such as linearity or the use of particular distributions, e.g. Gaussian noise. So our generic algorithm can be suitable for any nonlinear non-Gaussian problem in general state spaces

5.5 Numerical Example

In this section we shall demonstrate the proposed methodology by means of a simple numerical example. We will be using a linear Gaussian example with a quadratic instantaneous cost. This simple model has been well studied in the control literature [169] and there exist analytical expressions for the optimal policy of the risk sensitive problem. The risk neutral problem is commonly referred as the Linear Quadratic Gaussian (LQG) control, [169]. At each time n, consider a the state X_n evolving according to

$$X_{n+1} = HX_n + BA_n + W_n,$$

where A_n is the action, H, B are known matrices and $W_n \overset{i.i.d.}{\sim} \mathcal{N}(0, \Sigma)$. Let also

$$V(X_n, A_n) = \frac{1}{2} X_n^T Q X_n + \frac{1}{2} A_n^T R A_n,$$

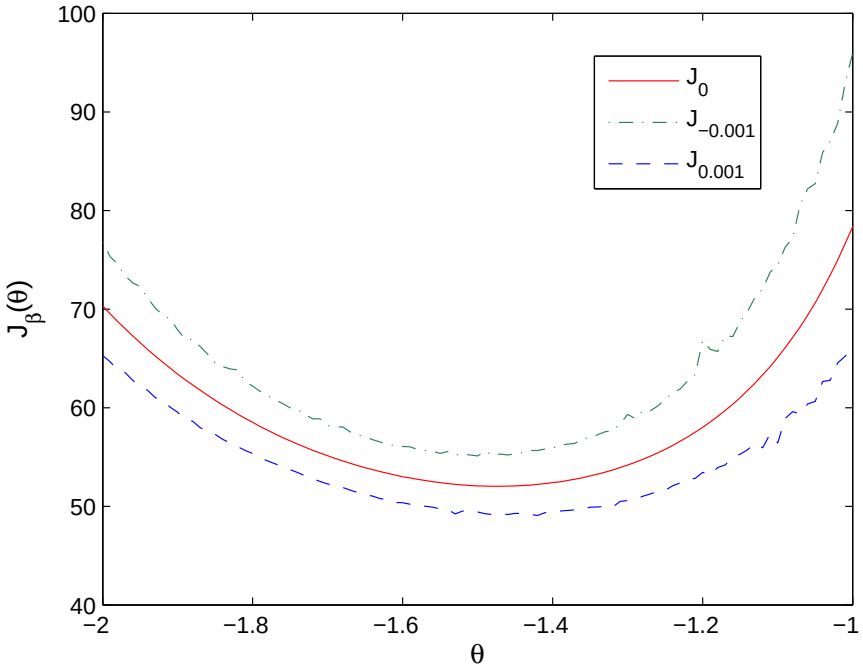


Figure 5.1: *Plot $J_\beta(\theta)$ against θ for the cases when $\beta = \{-0.001, 0, 0.001\}$.*

where Q, R are positive definite matrices. For the risk sensitive LQG problem the optimal policy can be computed analytically and is given by

$$A_n = F_\beta X_n,$$

To see how matrix F_β can be obtained we will follow the approach found in [169]. First we need to solve the general discrete-time algebraic Riccati equation for P,

$$H^T PH - P + H^T PB(B^T PB + R)^{-1} B^T PH + Q = 0, \tag{5.16}$$

and then use

$$P_\beta = (P^{-1} + \beta\Sigma)^{-1}, \tag{5.17}$$

$$F_\beta = -(B^T P_\beta B + R)^{-1} B^T P_\beta H, \tag{5.18}$$

where β is the risk sensitivity constant. For $\beta = 0$ this the solution coincides with that of the risk neutral LQG problem found in [3, 20].

The parameterisation for this problem is obvious. One can use F_β as parameter θ. We will consider the scalar case, where $\theta = F$ with $H = 1.7$, $B = 1$, $\Sigma = 4$, $Q = 4$ and $R = 1$. To

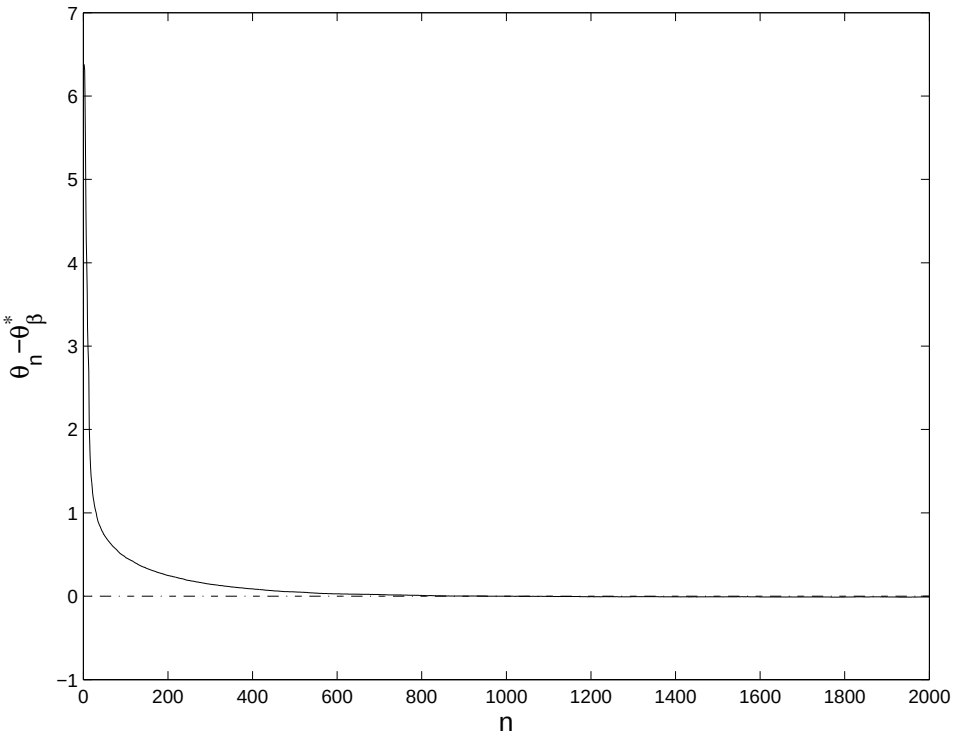


Figure 5.2: *Plot of the error $\theta_n - \theta^*_\beta$ against n for $\beta = 0.001$, $\alpha_n = 0.01$, $L = 1000$, $\theta_0 = 5$.*

distinguish between different choices of β, we shall denote the long term cost of equation (5.2) as $J_\beta(\theta)$ and the parameter of the optimal policy as θ^*_β. In Figure 5.1 we plot $J_\beta(\theta)$ for three cases where $\beta = \{-0.001, 0, 0.001\}$. For each β the corresponding optimal parameters given by equations (5.16)-(5.18) are $\theta^*_\beta = \{-1.4942, -1.4735, -1.4533\}$ respectively. In Figure 5.1, it is apparent that for the risk averse case with $\beta = -0.001$ the long term cost is higher than the risk neutral one. This leads to more pessimistic policies with a higher value of $|\theta^*_\beta|$. The opposite holds for the risk preferring case when $\beta = 0.001$.

In order to illustrate that Algorithm 5.1 converges to the optimal parameter θ^*_β, we plot in Figure 5.2 the error at each iteration, $\theta_n - \theta^*_\beta$, for $L = 1000$, $\theta_0 = 5$, $\beta = -0.001$. We have used a constant step size $\alpha = 0.01$. The same plot for $\beta = 0.001$ is very similar and is therefore omitted. In Table 5.1 we also record the total mean squared error (MSE) of the particle algorithm as well as the absolute value of the resulting bias in estimating θ^*_β for $\beta = \{0.001, -0.001\}$ and different values of L. When using more than 500 particles the algorithm seems to perform well. As we expect, increasing L improves the accuracy of the particle approximations and subsequently the optimal parameter's estimate.

L	bias for $\beta = 0.001$	bias for $\beta = -0.001$	MSE for $\beta = 0.001$	MSE for $\beta = -0.001$
100	0.0263	0.0196	0.345	0.313
200	0.0141	0.0102	0.184	0.181
500	0.0067	0.0060	0.163	0.147
1000	0.0046	0.0039	0.123	0.109
2000	0.0036	0.0027	0.097	0.098
5000	0.0024	0.0018	0.081	0.080

Table 5.1: *Observed absolute bias and total mean squared error (MSE), when using a single run of Algorithm 5.1 to compute θ^*_β for $\beta = \{0.001, -0.001\}$.*

5.6 Conclusions

In this chapter we solved a risk sensitive control problem, by using the Feynman-Kac representation of a Markov chain. We exploited the properties of the logarithmic Lyapunov exponent, which lead to a policy gradient solution for the parameterised problem. We have also presented an SMC algorithm that can be used to compute approximations when analytical expressions are not available. Finally we have shown how our algorithm can be implemented and used by means of a numerical example.

Part III

Online Distributed Parameter Estimation for Graphical Models

6

Self Localisation of Sensor Networks for Target Tracking

Summary. *In this chapter we present an overview of the self localisation problem for sensor networks deployed for target tracking. We shall propose a novel formulation using distributed state space models with multiple frames of reference. We will then cast the problem as one of static parameter inference using Maximum Likelihood (ML). We describe how a completely decentralized version of Recursive Maximum Likelihood (RML) can be implemented through the propagation of suitable messages that are exchanged between neighbouring nodes of the network. In the case when the network follows a tree topology, an exact implementation is given for dynamic linear Gaussian models. If loops are present, a loopy version of the algorithm is described. We conclude by discussing the disadvantages of the proposed algorithm and provide motivation for the material in the subsequent chapters.*

6.1 Introduction

The standard approach in centralised sensor management is to transmit measurements from all sensor nodes to a fusion node which performs most of the computation [13]. This limits the number of expensive sensors used [77]. However, the spatial deployment of the network is limited to a restricted range relative to the central fusion node. Moreover, the (large) communication bandwidth requirements also limit the maximum number of potentially utilised sensors [1]. This motivates the adoption of decentralised or distributed approaches, such as distributed sensor networks. In such networks neighbouring sensors-trackers exchange information between themselves in order to track the target optimally as in the centralised approach [28].

The use of distributed sensors forming a sensor network has influenced a great deal of recent developments in the areas of control and signal processing. In a distributed architecture, each node of the network obtains measurements of a common monitored state and the distribution of the state is updated at each node using all the measurements of all the nodes in the network instead of the observation of each individual sensor. Thus, the filtering distribution of the state can be considered as a marginal of the joint distribution of all the measurements and the common state. Problems in which the jointly filtered state is of great interest include environmental monitoring [123], such as greenhouse control, where one controls the shutters, air inflow, and the humidifier to regulate light intensity temperature and humidity [128], or distributed tracking [59,159], where the self calibration of the network is important for performing joint sensor fusion.

Furthermore, in many distributed sensor networks it is also important for each node to "learn" the position or coordinates of its neighbours relative to itself. This is known as sensor self-localisation or self-calibration [1]; this problem also often appears in the computer vision literature [79]. For example, let a sensor network consist of a collection of nodes: each node is a tracker and considers itself as the origin of its own coordinate system. In a target tracking scenario it is important for each sensor-tracker node to be able to translate the relative frame of reference of its neighbour to its own in order to utilise and fuse the measurements. This is essentially equivalent to knowing with precision the coordinates of its neighbours w.r.t. itself. If these coordinates are imprecise or unknown, it might not be possible to perform distributed sensor fusion at all.

A related problem is that of sensor registration, where all sensors have to learn some parameters of its neighbouring nodes so as to be able to use their measurements for tracking. The network should also be able to adapt to potentially changing parameters and/or network topology. In particular, sensors' measurements are known to exhibit errors, which in some cases can be decomposed into random noise and systematic biases. The problem of identifying these biases is known as sensor registration [124,125,165]. In a centralised approach this can be solved at the fusion center by augmenting the state with this bias and maximising the resulting measurement likelihood. In a distributed context, the problem is somewhat more complex and each node needs to estimate/register the biases of its neighbours. We want to achieve this using only local messages between a node and its neighbours. Solving this problem is essential in many real-world systems as a heterogeneous collection of sensors is often used and their associated sensor biases can differ significantly [125].

In this chapter, we show that these problems -sensor registration and self-localisation for sensor networks- can be cast as recursive static parameter estimation for dynamic Graphical models. We shall be adopting here a Recursive Maximum Likelihood (RML) approach seen in Section 3.5.3, where the (average) log-likelihood of the unknown parameters is maximized on-line using a stochastic gradient approach. Belief Propagation ideas have been widely used to perform statistical inference in undirected and directed graphs using message passing [130]. The novelty of our implementation relies on a fully decentralized calculation of the log-likelihood gradient on graphs. In this respect, it can be interpreted as a generalization of Belief Propagation to undirected graphs. Received messages at a node shall contain sufficient statistics sent from the rest of the network, in order for that node to infer the optimal parameters of its surrounding neighbourhood.

The rest of this Chapter is organized as follows: in Section 6.2 we present our framework, which uses multiple frame of references for tracking and RML for self localisation. In Section 6.3 we propose a generic algorithm to solve the self-localisation problem. Finally, in Sections 6.4 and 6.5 we illustrate our ideas on an example and make some conclusive remarks, which should motivate the work contained in the subsequent chapters.

6.2 Problem Formulation

We consider a sensor network deployed for the tracking of moving targets. We will assume that at each time only a single target is present and every node in the sensor network observes the same target. Let the set of nodes of the network be indexed by the finite set $\mathcal{V}$ while the connectivity of the network is specified by the set of edges $\mathcal{E}$. We will deal with an undirected Graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We assume that two nodes i and j communicate provided the edge $e = (i, j)$ (or equivalently (j, i)) belongs to $\mathcal{E}$.

6.2.1 Distributed State Space Models and Collaborative Filtering

The state of a node r is a random variable X_n^r that represents the state of the target . The state of the target X_n^r would comprise of the cartesian position and velocities of the target being tracked at time n, as measured with respect the local coordinate frame of reference of node r. The only assumption on the target dynamic model is

$$X_{n+1}^r \big| \, X_n^r = x^r \sim f^r \left(\cdot | \, x^r \right)$$

A local measurement Y_n^r of the target's state is generated based on the sensing capabilities of node r. Let the measurement be generated according to the probability density function

$$Y_n^r | \, X_n^r = x^r \sim g_{\theta_*^r} \left(\cdot | \, x^r \right)$$

where θ_*^r is the unknown calibration parameter of sensor node r.

Since all nodes are engaged in tracking the same target, it is possible to utilise the measurement of all nodes to enhance the tracking performance. For instance, for a completely decoupled implementation, each node will maintain its own local filtering distribution

$$\pi_n^r(dx_n^r) = P_{\theta_*^r}(X_n^r \in dx_n^r | \, Y_1^r, \ldots, Y_n^r),$$

where it will only process its own measurements $Y_1^r, \ldots, Y_n^r$. We are interested instead in a collaborative filtering approach, where all nodes are fully coupled by means of the sensor network and process all the measurements received from all the sensors. In a *fully coupled* implementation, each node will also use the measurements of all other nodes in the graph to update the Bayes recursion, i.e. it propagates the filtering distribution

$$\pi_n^r(dx_n^r) = P_{\theta_*}(X_n^r \in dx_n^r | \, Y_1, \ldots, Y_n)$$

where Y_n denotes the vector of stacked observations $[Y_n^v]_{v\in\mathcal{V}}$ and θ_* denotes the vector of stacked parameters $[\theta_*^v]_{v\in\mathcal{V}}$.

Similarly, the prediction distribution will be

$$\pi^r_{n|n-1}(dx^r_n) = P_{\theta_*}(X^r_n \in dx^r_n | Y_1, \ldots, Y_{n-1})$$

It is clear that a fully coupled implementation is advantageous since nodes whose observations are poor, due to distance or sensing capabilities, can benefit from other sensors with better quality observations. In principle, even in the case where the target was out of the range of a specific sensor, it could then utilise measurements from other nodes to know where it is, although this case will not be considered further in this thesis.

The prediction step can be directly done in a decentralized way at each node. However in the update step, each node receives an observation and all nodes in the network have to share their information by passing messages around the network in order to compute their filtering distribution. In addition, we have to account for the case when θ_* is not known and has to be estimated. We emphasise that ϑ_* is a collection of static parameters that are distributed around the network.

The present formulation is similar to that of Section 3.7.5, although here potentials have not been explicitly defined. One interpretation of this framework is that $\theta_*$is a collection of measurement biases or calibration parameters of each sensor. With reference to the notation of Section 3.7.5, in this case each θ^r_* is the same as parameter θ^{r^*} in Section 3.7.5. Thus, the present formulation can be used for bias registration or calibration problems. In the beginning of Section 3.7.6, the discussion explains how such problems can be solved locally at each node. Therefore we shall not discuss these problems further in this thesis. Another interpretation of the present formulation is that θ^r_* corresponds to coordinate transformations between the frame of references of two different nodes. In this case it can be shown that for a specific choice of potentials θ^r_* of the present formulation resembles ϑ^{e^*} in Section 3.7.5, where e is a particular edge. This is explained in detail later in this chapter. Hence, we could use this framework for the sensor self localisation problem as well. In the remainder of this thesis, we will mainly consider the problem of self localisation, but it is clear that our framework is generic and that the proposed methodology can be used for other problems as well.

6.2.2 The Sensor Self Localisation Problem

We will concentrate on the sensor self localisation problem formulated as in the previous section. For sake of clarity, we make the following assumption which we stress is not essential for the framework we propose.

Assumption 6.2.1 *All nodes maintain a 2D-cartesian coordinate system and maintain as the state of the target its position and velocity in the relevant directions. Also, each node regards itself as located at the origin of its own coordinate system.*

We refer to a particular node r and would like to use the measurements of the rest of the nodes to compute π_n^r. Obviously the measurement likelihood depends on the coordinate of each node w.r.t. node r. We denote $\theta_*^{r,j}$ as the coordinate transformation from node r to j, i.e. $\theta_*^{r,j}$ is the origin of node r in the coordinate frame of node j. As regard to the notation the previous section, we will use from now on $\theta_* = [\theta_*^{r,j}]_{(r,j)\in\mathcal{E}}$ as the self localisation parameter. Also, due to Assumption 6.2.1 we have $\theta_*^{r,r} = 0$.

We denote the observation likelihood of node r by $g^r(\cdot|\, x^r)$. The Bayes recursion applied independently to each node r would yield

$$\pi_{n+1}^r(x_{n+1}^r) \propto g^r(Y_{n+1}^r|\, x_{n+1}^r) \int f^r(x_{n+1}^r|x_n^r)\pi_n^r(x_n^r)dx_n^r.$$

In a coupled implementation of only two nodes, node r incorporates the observation of adjacent node j (connected by an edge),

$$\pi_{n+1}^r(x_{n+1}^r) \propto g^r\left(Y_{n+1}^r|\, x_{n+1}^r\right) g^j(Y_{n+1}^j|x_n^r + \theta_*^{r,j}) \int f^r(x_{n+1}^r|x_n^r)\pi_n^r(x_n^r)dx_n^r.$$

It is clear that sharing the observations is only possible when the coordinate transformation variable $\theta_*^{r,j}$ is available. In this chapter, we will extend this two node coupling for an entire network of nodes, in which each node aims to use the observations from the entire network to update its filtering density. The problem is that each node r will need to know $\theta_*^{r,j}$, where j will be any neighbouring node. The problem that arises is estimating $\theta_*^{r,j}$ distributively. Using the RML approach presented in Section 3.5.3, we will propose in the remainder of this thesis different methods such that each node r can simultaneously estimate $\theta_*^{r,j}$ and perform collaborative filtering to track x_n^r.

Assumption 6.2.2 *All nodes have a* consistent *transition density for the target being tracked and let this density for node r be denoted by $f^r(x^{r\prime}|\,x^r)$. By consistent we mean that for any two nodes r and j,*

$$f^r(x^{r\prime}\big|\,x^r) = f^j(x^{r\prime} + \theta_*^{r,j}\big|\,x^r + \theta_*^{r,j}).$$

This assumption is necessary for the problem to be formulated without any ambiguity when translating the coordinate system from one node to another. All nodes should maintain the same transition density or prediction model with respect to an arbitrary coordinate scheme. This is important since we want all nodes to share the same filtering and prediction densities. Moreover the transition density should in turn obey the physical coordinate transformation of the state from one node to another. Using Assumption 6.2.2 we can also show that the prediction and filtering density at each node can be propagated consistently:

Property 6.2.1 *Assume π_n^r and π_n^j satisfy*

$$\pi_n^r(x^r) = \pi_n^j(x^r + \theta_*^{r,j})$$

for all x_n^r. This implies also $\pi_n^r(x^j + \theta_*^{j,r}) = \pi_n^j(x^j)$ for all x^j. Then one also has $\pi_{n+1|n}^r(x^r) = \pi_{n+1|n}^j\left(x^r + \theta_*^{r,j}\right)$.

In the fully coupled implementation, each node r will incorporate the observations of all other nodes in the network too. In the filtering step implemented by a node r, after all nodes take a measurement, we have

$$\pi_{n+1}^r(x_n^r) \propto \left(\prod_{v\in\mathcal{V}\setminus\{r\}} g^v(Y_{n+1}^v\big|\,x_n^r + \theta_*^{r,v})\right) g^r(Y_{n+1}^r\big|\,x_n^r)\pi_{n+1|n}^r(x_n^r)$$

where $\theta_*^{r,v}$ for non adjacent nodes is defined as follows: for any path that connects nodes r and v then

$$\theta_*^{r,v} = \theta_*^{r,j_1} + \theta_*^{j_1,j_2} + \ldots + \theta_*^{j_{n-1},j_n} + \theta_*^{j_n,v}. \tag{6.1}$$

Since we start with the same prior distribution and all nodes have the same transition densities, it is obvious that the filtering density shared in the network will be the same for all nodes on it. The coordinate transformations are consistent at each filtering step and hence can be used when one node needs to pass a message to another.

Property 6.2.2 *After the update step, for any two pair of nodes r and j, it follows that*

$$\pi_n^r(x^r) = \pi_n^j(x^r + \theta_*^{r,j})$$

for all x^i.

Note that for the localisation problem the coordinate transformation function $h_{\theta_*^{r,j}}$ of the state from node r to node j can be generalised to

$$x^j = h_{\theta_*^{r,j}}(x^r) = \alpha_*^r H_*^{r,j} x^r + \theta_*^{r,j},$$

where α^r is a scaling factor and $H^{r,j}$ a rotation matrix. This could be very useful in many computer vision or surveillance applications, where networks of cameras are used and observe targets at different orientations. Our method to follow can handle this without any alteration, and can even be extended to estimate $\alpha_*^r H_*^{r,j}$. For the sake of simplicity only, we shall ignore the case where rotation appears, and set $\alpha_*^r H_*^{r,j} = 0$. In the general sensor registration case $h_{\theta_*^{r,j}}$ could admit any nonlinear form, but only if Assumption 6.2.2 and is true then we could possibly generalise our results without further considerations.

In addition, the framework presented in Section 6.2.1 is equivalent to Section 3.7.6, if the prediction step is carried out at each node separately and for the potentials we set

$$\phi_n^r(x_n^r) = g^r(Y_n^r | x_n^r),$$
$$\psi_n^{rj}(x_n^r, x_n^j) = \delta_{x_n^j + \theta_*^{j,r}}(x_n^r).$$

A standard Belief Propagation message passing after the prediction step would yield the same filtering density. Unfortunately, the RML recursion described in Section 3.7.6 cannot be used for estimating the static parameter of the formulation of Section 6.2.1 directly, because it would be impossible to differentiate $\psi_n^{rj}(x_n^r, x_n^j) = \delta_{x_n^j + \theta_*^{j,r}}(x_n^r)$. This might not look convenient to start with, but later in this chapter we will to design appropriate algorithmic modifications to bypass this problem.

6.2.3 Other Proposed Approaches for Collaborative Filtering and Localisation

In this section we shall try to justify why we choose to use the framework presented earlier in Section 6.2.2 together a with maximum likelihood (ML) approach for estimating θ_*. We shall compare it with a joint state parameter filtering approach, when the same or alternative settings

for the coordinate framework are used. At the end of this discussion we aim to show that our choice of framework is very sensible and show how it can be equivalent to a Graphical model approach.

We shall consider first the case when each sensor node has a coordinate relative to some fixed point 0. Let $\theta^0 = [\theta^{i,0}]_{i\in\mathcal{V}}$ and consider the problem of jointly estimating $p(x_n^0, \theta^0)$, where x_n^0 is the coordinate of the observed state at time n with respect to point 0. In a standard Bayesian setting we would recursively update a prior using the likelihood to obtain a posterior. Each sensor's position can be derived by marginalising $p(x_n^0, \theta)$ and obtaining $p(x_n^0, \theta^{i,0}|Y_n)$. As far as the likelihood is concerned, we have already assumed that each node measures the state independently. Hence the joint likelihood will always take the following form

$$p(Y_n|x_n^{\mathcal{V}}) = \prod_{v\in\mathcal{V}} g^v(Y_n^v \,|\, x_n^v),$$

where Y_n denote the vector of stacked observations $[Y_n^v]_{v\in\mathcal{V}}$. It is trivial to spot that the likelihood does not give sufficient information about θ^0 and also

$$p(Y_n|x_n^0, \theta) = p(Y_n|x_n^0 + c, \theta^{1,0} + c, ..., \theta^{M,0} + c),$$

where M is the total number of nodes in the graph. In other words, whatever we use for $\theta^{v,0}$ and x_n^0, we will always get the same measurement Y_n regardless of where point 0 is located. Furthermore, each node cannot possess any information about any other node's locations or the network size, therefore we could not pose any bound on the support (x_n^0, θ^0). In a Bayesian framework this means we would have to use uninformative flat priors. Given also that the likelihood is only informative about differences of the variables in (x_n^0, θ^0), we would end up with improper posterior. This can be attributed also to the fact that there is no measurement taken from a sensor at node 0. As a result, any selected reference point should be an arbitrary node on the network. On the other hand, a choice of such a node would limit the effectiveness of any solution to the localisation problem to many applications in distributed environments. This motivated the idea of each node assuming itself as the origin and performing relative localisation between the nodes.

On the other hand, one could consider jointly filtering $p(x_n^1, \theta^1)$, where $\theta^1 = [\theta^{i,1}]_{i\in\mathcal{V}}$ denotes the relative coordinate transformation with respect to node 1. Note that node 1 is an arbitrary choice and it could be replaced by any node on the graph. This was done in [59] for an example of a chain of twelve cameras. By the term chain, we mean a graph which could be though of a

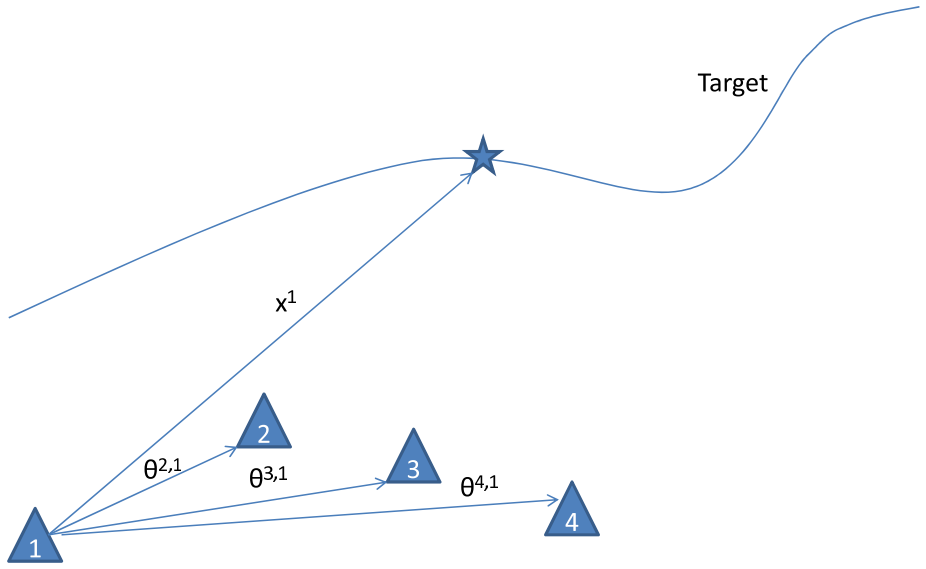


Figure 6.1: *Four nodes jointly observing a target using the frame of reference of node 1 only.*

single walk between a starting node to a finishing one, see a 4 node example drawn on Figure 6.2. In [59, 159] the authors explored possible factorisations of $p(x_n^1, \theta^1)$ in order to remove "weaker" dependencies between variables and relieve the computation. They constructed a *junction tree*, which is a tree graph of cliques, and performed a filtering on $p(x_n^1, \theta^1)$ following an Extended Kalman filtering approach.

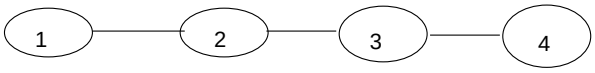


Figure 6.2: *An example of a 4 node chain.*

We will investigate their methodology for a four node example drawn in Figure 6.1, where $\theta^1 = [\theta^{2,1^T}, \theta^{3,1^T}, \theta^{4,1^T}]^T$.The joint filtering problem consists of performing sequentially in time

the prediction and update steps on the joint vector containing x_n^1 and θ^1:

$$\textbf{Prediction } p_n(x_n^1, \theta^1 | Y_{1:n-1}) = \int f(x_n^1 | x_{n-1}^1) p_{n-1}(x_{n-1}^1, \theta^1 | Y_{1:n-1}) dx_{n-1}^1, \tag{6.2}$$

$$\textbf{Update} \quad p_n(x_n^1, \theta^1 | Y_{1:n}) \propto p(Y_n | x_n^1, \theta^1) p_n(x_n^1, \theta^1 | Y_{1:n-1}), \tag{6.3}$$

where $Y_n = [Y_n^{1^T}, Y_n^{2^T}, Y_n^{3^T}, Y_n^{4^T}]^T$ and

$$p(Y_n | x_n^1, \theta^1) = p(Y_n^1 | x_n^1) p(Y_n^2 | x_n^1, \theta^{2,1}) p(Y_n^3 | x_n^1, \theta^{3,1}) p(Y_n^4 | x_n^1, \theta^{4,1}). \tag{6.4}$$

We will assume that at $n = 0$ independence holds and that the initial distribution $\pi_0(x_0^1, \theta^1)$ is given by

$$\pi_0(x_0^1, \theta) = \pi_0(\theta^{2,1}) \pi_0(\theta^{3,1}) \pi_0(\theta^{4,1}) \pi_0(x_0^1).$$

Although this can be posed as a centralised problem and becomes straightforward to formulate and solve, this approach is not useful in environments where distributed synchronised sensors or trackers are used. We assume that each sensor has considerable computational abilities, but does not access directly the measurements or estimates of x_n^1, $\theta^{2,1}, \theta^{3,1}, \theta^{4,1}$ that other nodes may possess. In order each node to utilise all the measurements of the network to be able to perform filtering jointly on x_n^1 and a small subset of $\theta^{2,1}, \theta^{3,1}, \theta^{4,1}$ we have to define messages between the nodes. This has to be done in such a scalable way that the proposed solution can be extended to a tree network of any size.

Funiak et al. in [59] have proposed a decentralised solution for a 12 node chain using inference on junction trees. We shall illustrate the key elements of their approach using a 4 node chain drawn in Figure 6.2. The inference graph they use is separate from the communication graph. The communication graph is simply the 4 node chain of Figure 6.2 and they use as an inference graph the junction tree of Figure 6.3. This relies on using following factorisation for the joint distribution including all the states and parameters:

$$p(x_n^1, \theta) = \frac{p(x_n^1, \theta^{2,1}) p(x_n^1, \theta^{2,1}, \theta^{3,1}) p(x_n^1, \theta^{3,1}, \theta^{4,1}) p(x_n^1, \theta^{4,1})}{p(x_n^1, \theta^{2,1}) p(x_n^1, \theta^{3,1}) p(x_n^1, \theta^{4,1})} \tag{6.5}$$

Note that this is an approximation since it does not include all dependencies between all the localisation parameters of the nodes. The authors in [59] claim that the omitted dependencies do not contribute much in further reducing the variance of the estimate. We verified this claim using a simple linear Gaussian numerical example, but do not present that here as it does not seem important.

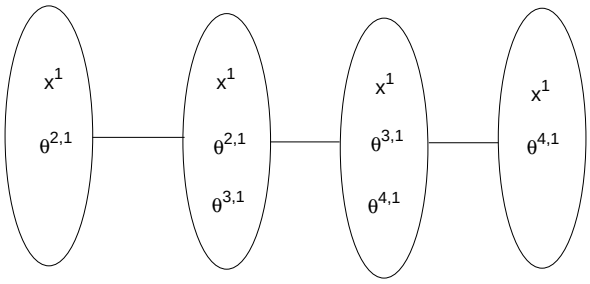


Figure 6.3: *Junction tree for 4 node chain used in Funiak et al. [59] as an inference graph for distributed computation.*

There is an inherent drawback in the methodology of [59, 159]. Each sensor tracks the target using the frame of reference of a certain pin pointed reference node. So far we have used node 1. It is not apparent how reference nodes can be chosen in many applications. It might be impossible or cost inefficient to implement such an approach. In our framework proposed earlier in Section 6.2.2, we used multiple frames of reference. Instead of using the location of each node in the sensor network relative to a reference or anchor node, we used the relative locations between adjacent nodes and showed how to perform distributed filtering. This scenario is described by Figure 6.4. Each node maintains a its own frame of reference and tracks state x^i where x^i is the state of the target relative to node i. For the remainder of this section we drop the time subscripts on X^i and Y for the sake of simplicity. For each node i, j, $\theta^{i,j}$ is the relative position of node j with respect to node i. So since we are examining a chain this time the localisation parameters can be stacked as $\theta = [\theta^{1,2^T}, \theta^{2,3^T}, \theta^{3,4^T}]^T$.

If we were to implement the algorithm of Funiak et al. [59], we would have to choose to use the junction tree of Figure 6.5 for inference. This gives the following factorisation for the joint distribution including all the states and parameters.

$$p(x^{1:4}, \theta) = \frac{p(x^1, \theta^{1,2})p(x^2, \theta^{1,2}, \theta^{2,3})p(x^3, \theta^{2,3}, \theta^{3,4})p(x^4, \theta^{3,4})}{p(\theta^{1,2})p(\theta^{2,3})p(\theta^{3,4})} \tag{6.6}$$

This factorisation seems to be the only sensible choice given that each node should process only its own state, the relative location of its neighbours and incoming messages. We have found

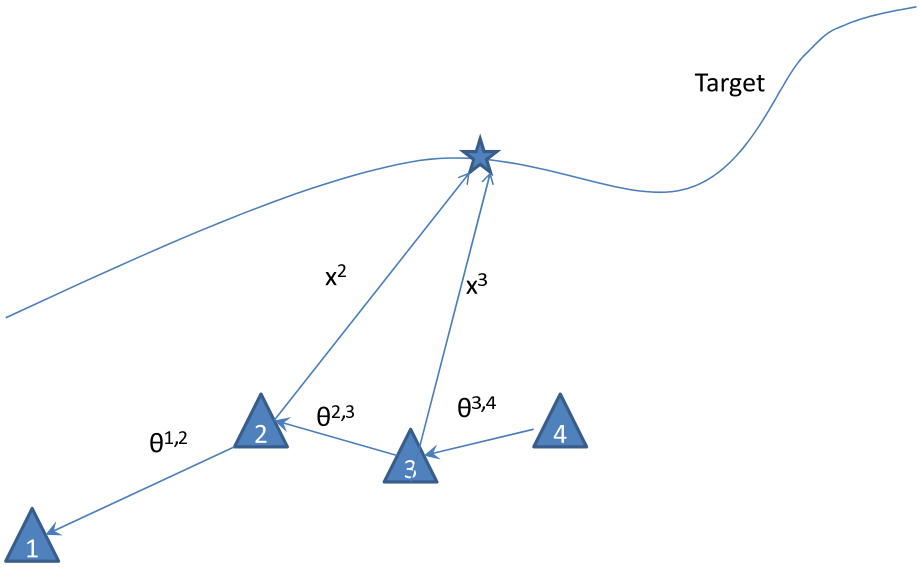


Figure 6.4: *Four nodes jointly observing a target using multiple frame of references. We show only the state w.r.t. node 2 and 3.*

that the algorithm in [59] cannot be used in the multiple frame of reference case, since it will not be able to produce informative posterior distributions of θ when we start with each element in θ being independent from each other and also independent to each x^i.

To make this clear, we shall examine a single prediction and update step after initialising all variables to be independent. Let the initial distribution be

$$\pi_0(x_0, \theta) = \pi_0(\theta^{1,2})\pi_0(\theta^{2,3})\pi_0(\theta^{3,4})\pi_0(x^1)\pi_0(x^2)\pi_0(x^3)\pi_0(x^4).$$

After initialisation the centralised prediction step is given by

$$\begin{aligned} p_1(x^{1:4}, \theta) = \int \pi_0(\theta^{1,2})\pi_0(\theta^{2,3})\pi_0(\theta^{3,4})\pi_0(x^{1\prime})\pi_0(x^{2\prime})\pi_0(x^{3\prime})\pi_0(x^{4\prime}) \\ \times f(x^1|x^{1\prime})f(x^2|x^{2\prime})f(x^3|x^{3\prime})f(x^4|x^{4\prime})dx^{1:4\prime}, \end{aligned}$$

We will use the junction tree of Figure 6.5 to perform the prediction step distributively. For each state x^i its corresponding clique can perform a local prediction step independently. In each clique i the prediction step will be given for $i = 1, ..., 4$ by

$$p_1^i(x^i) = \int \pi_0(x^{i\prime})f(x^i|x^{i\prime})dx^{i\prime},$$

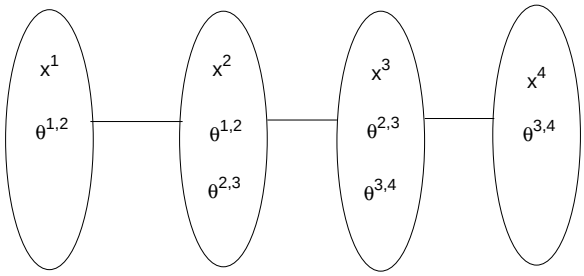


Figure 6.5: *Junction tree for the multiple frame of reference problem.*

Then, for example in clique 2 the joint distribution of the clique variables will be simply

$$\widetilde{p}_1^2(x^2, \theta) = p_1^2(x^2)\pi(\theta^{1,2})\pi(\theta^{2,3}),$$

and one can obtain similar expressions for the rest. For the global prediction step a simple message passing algorithm would yield the joint distribution of the variables given by

$$\widetilde{p}_1(x^{1:4}, \theta) = \frac{p_1^i(x^1)\pi(\theta^{1,2})p_1^i(x^2)\pi_0(\theta^{1,2})\pi_0(\theta^{2,3})p_1^i(x^3)\pi_0(\theta^{2,3})\pi_0(\theta^{3,4})p_1^i(x^4)\pi_0(\theta^{3,4})}{\pi_0(\theta^{1,2})\pi_0(\theta^{2,3})\pi_0(\theta^{3,4})}$$

This is the same factorisation as in (6.6) . Note that $\widetilde{p}_1(x^{1:4}, \theta)$ agrees completely with the centralised prediction step. Also, independence between all variables is preserved.

Assume now that an observation Y becomes available and likelihood is given by

$$p(Y|x^{1:4}) = p(Y^1|x^1)p(Y^2|x^2)p(Y^3|x^3)p(Y^4|x^4).$$

For the centralised update step we have

$$p(x^{1:4}, \theta|Y) \propto p(Y|x^{1:4})p_1(x^{1:4}, \theta).$$

For performing the update step using the junction tree of Figure 6.5, in each clique i a local update step can done as

$$\widetilde{p}_1(x^{1:4}, \theta|Y) \propto p(Y^i|x^i)\widetilde{p}_1^i(x^i, \theta)$$

and then the global update will be given by the product of local updates as

$$\widetilde{p}_1(x^{1:4}, \theta|Y)$$
$$\propto p(Y^1|x^1)p(Y^2|x^2)p(Y^3|x^3)p(Y^4|x^4)p_1^1(x^1)p_1^2(x^2)p_1^3(x^3)p_1^4(x^4)\pi_0(\theta^{1,2})\pi_0(\theta^{2,3})\pi_0(\theta^{3,4}).$$

Again $\widetilde{p}_1(x^{1:4}, \theta|Y)$ agrees with the centralised update $p(x^{1:4}, \theta|Y)$ and independence is preserved. Each x^i and θ will remain decoupled if the recursion is propagated further in time. Unfortunately this is an issue, because any attempt to perform filtering recursively might benefit the marginal distributions of x^i, but the marginal distribution of θ will remain as in the initial condition. This problem has not appeared in the formulation of Funiak et al [59] because the independence is not preserved by the prediction step and the likelihood appears as a function of the localisation parameters explicitly (and not by means of transformation as in our case,) conveying some information on θ. This is a direct consequence of using a single frame of reference.

We feel that Maximum Likelihood (ML) estimation is the most sensible approach to perform distributed parameter inference for our framework. Choosing ML for estimating the localisation parameters was strongly influenced by the choice of multiple frames of reference. The formulation of Section 6.2.2 is to the best of our knowledge is novel and can address some of the problems not seen at previous attempts to solve the self localisation problem. In addition, it can be used for simultaneous collaborative filtering and parameter estimation in a fully decentralised way.

6.3 Distributed Recursive Maximum Likelihood for Self Localisation

We will propose a distributed implementation of RML for learning all the coordinate transformations $\theta_*^{r,j}$. Since there is one parameter per edge, namely $\theta^{r,j}$ for the edge $(j, r) \in \mathcal{E}$, a particular node, say r, will take ownership of the parameter and recursively update it as observations are received in the network. This node shall be referred to as root node for that edge, since it shall be the node to collect messages from the rest of the network in order to iterate $\theta_n^{r,j}$ as the RML parameter update in (3.23) as presented in Section 3.5.3. The messages received by each node r from its neighbours should be sufficient to iterate the parameter $\theta_n^{r,j}$ and perform the update step of the Bayesian recursion yielding the filtering density π_n^r. Since the prediction

step can be performed locally at each node, we shall be able to estimate $\theta^{r,j}$ while propagating π_n^r.

We now formulate the RML problem for node r as the reference node. Note this is an arbitrary choice since any node in the network can become root node. For a fixed parameter $\theta = [\theta^{i,j}]_{(i,j)\in\mathcal{E}}$, the recursive log likelihood $J^r_{\theta^{r,j}} = \log p_{\theta^{r,j}}(Y_n|Y_{1:n-1})$ can be written as

$$J^r_{\theta^{r,j}} = \log \int \prod_{v\in\mathcal{V}} g^v(Y_n^v|\, x_n^r + \theta^{r,v})\pi^r_{n|n-1}(x^r_{n-1})dx^r_n$$

where $\theta^{r,r} = 0$. We will now take the gradient of this quantity with respect to $\theta^{r,j}$, i.e., we are assuming that node (r,j) is a valid edge and that node r has ownership of parameter $\theta^{r,j}$, i.e. node r is the controlling node in this case,

$$\begin{aligned}\nabla_{\theta^{r,j}} J^r_{\theta^{r,j}} = &\left(\int \prod_{v\in\mathcal{V}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})\pi^r_{n|n-1}(x^r_n)dx^r_n\right)^{-1} \\ &\times \{\int \left(\sum_{v\in\mathcal{V}} \frac{\nabla_{\theta^{r,j}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})}{g^v(Y_n^r|\, x_n^r + \theta^{r,v})}\right) \prod_{v\in\mathcal{V}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})\pi^r_{n|n-1}(x^r_n)dx^r_n + \\ &\int \prod_{v\in\mathcal{V}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})\nabla_{\theta^{r,j}}\pi^r_{n|n-1}(x^r_n)dx^r_n\} \end{aligned} \tag{6.7}$$

In the context of recursive distributed parameter estimation $\pi^r_{n|n-1}(x^r_n)$ and its gradient $\nabla_{\theta^{r,j}}\pi^r_{n|n-1}(x^r_n)$ should be propagated locally at node r. It also appears necessary to pass $\prod_{v\in\mathcal{V}\setminus\{r\}} g^v(Y_n^v|\, x_n^r+\theta^{r,v})$ and $\sum_{v\in\mathcal{V}} \frac{\nabla_{\theta^{r,j}} g^v(Y_n^r|x_n^r+\theta^{r,v})}{g^v(Y_n^r|x_n^r+\theta^{r,v})}$ using appropriate messages from the network to node r in order to be able to update $\theta^{r,j}_n$ using RML. It can be shown that these messages are sufficient to propagate the filtering and prediction densities as well as their gradients.

6.3.1 Propagating the Filtering and Prediction Densities and its Derivatives

For node r we would like to implement a recursion for $\pi^r_n(x^r_n)$ and $\nabla_{\theta^{r,j}}\pi^r_n(x^r_n)$ given that $\pi^r_{n-1}(x^r_{n-1})$ and $\nabla_{\theta^{r,j}}\pi^r_{n-1}(x^r_{n-1})$ are available locally from previous epoch $n-1$. As the choice of node r as a root node is a completely arbitrary choice, the derivation for any other node is identical. Consider the prediction and update stage of the filter

$$\pi^r_{n|n-1}(x^r_n) = \int f^r(x^r_n|x^r_{n-1})\pi^r_{n-1}(x^r_{n-1})dx^r_{n-1}, \tag{6.8}$$

and

$$\pi^r_n(x^r_n) = \frac{\prod_{v\in\mathcal{V}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})\pi^r_{n|n-1}(x^r_n)}{\int \prod_{v\in\mathcal{V}} g^v(Y_n^r|\, x_n^r + \theta^{r,v})\pi^r_{n|n-1}(x^r_n)dx^r_n} \tag{6.9}$$

We aim to propagate not only these distributions but also their derivatives. For the derivatives we have:

$$\nabla_{\theta^{r,j}}\pi^r_{n|n-1}(x^r_n) = \int f^r(x^r_n|x^r_{n-1})\nabla_{\theta^{r,j}}\pi^r_{n-1}(x^r_{n-1})dx^r_{n-1}, \tag{6.10}$$

and

$$\begin{aligned}&\nabla_{\theta^{r,j}}\pi^r_n(x^r_n) = \\ &\pi^r_n(x^r_n)\left(\begin{array}{c}\frac{\nabla_{\theta^{r,j}}\pi^r_{n|n-1}(x^r_n)}{\pi^r_{n|n-1}(x^r_n)} + \sum\limits_{v\in\mathcal{V}}\frac{\nabla_{\theta^{r,j}}g^v(Y^r_n|x^r_n+\theta^{r,v})}{g^v(Y^r_n|x^r_n+\theta^{r,v})} \\ -\int\left(\frac{\nabla_{\theta^{r,j}}\pi^r_{n|n-1}(x^r_n)}{\pi^r_{n|n-1}(x^r_n)} + \sum\limits_{v\in\mathcal{V}}\frac{\nabla_{\theta^{r,j}}g^v(Y^r_n|x^r_n+\theta^{r,v})}{g^v(Y^r_n|x^r_n+\theta^{r,v})}\right)\pi^r_n(x^r_n)dx^r_n\end{array}\right).\end{aligned} \tag{6.11}$$

Note that if $\prod\limits_{v\in\mathcal{V}\setminus\{r\}} g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ and $\sum\limits_{v\in\mathcal{V}}\frac{\nabla_{\theta^{r,j}}g^v(Y^r_n|x^r_n+\theta^{r,v})}{g^v(Y^r_n|x^r_n+\theta^{r,v})}$ are available at the root node r, then the filtering and prediction distributions together with their derivatives can be computed locally. These quantities should become available to node r by messages received from its neighbours, which in turn receive messages from theirs. We aim to define an appropriate message passing scheme so that all possible nodes can act as roots and update any parameters associated with its adjacent edges.

6.3.2 Defining Message Passing in the Sensor Network

To derive a distributed implementation, consider first how for each $v\in\mathcal{V}$, $g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ and $\nabla_{\theta^{r,j}}g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ can be communicated to node r via a sequence of messages. Assume the directed path from node v to r traverses the edges (v,j_l), (j_l,j_{l-1}), $\ldots$, (j_2,j_1), (j_1,r).

In order to pass $g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ from node v to r, the incoming message to node j_k from node j_{k+1} should be $g^v(Y^v_n|\, x^{j_k}_n+\theta^{j_k,v})$. Then node j_k should forward to node j_{k-1} the message

$$g^v(Y^v_n|\, x^{j_k}_n+\theta^{j_k,v})\big|_{x^{j_k}_n=x^{j_{k-1}}_n+\theta^{j_{k-1},j_k}} = g^v(Y^v_n|\, x^{j_{k-1}}_n+\theta^{j_{k-1},v})$$

So starting from $g^v(Y^v_n|\, x^v_n)$ at node v, after using repetitive coordinate transformations from each node j_{k-1} to node j_k and taking advantage of (6.1), $g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ can reach root node r.

In order to show how $\nabla_{\theta^{r,j}}g^v(Y^v_n|\, x^r_n+\theta^{r,v})$ could be passed from node v to r in a similar fashion, we note from (6.1) that for any path connecting node r and v, which does not include the edge (r,j), then $\nabla_{\theta^{r,j}}g^v(Y^v_n|\, x^v_n+\theta^{r,v})=0$. If $\theta^{r,v}$ is defined over a path that includes edge (r,j) then

$$\nabla_{\theta^{r,j}}g^v(Y^v_n|\, x^r_n+\theta^{r,v}) = \nabla_z g^v(Y^v_n|\, z)|_{z=x^r_n+\theta^{r,v}}$$

since $\nabla_{\theta^{r,j}}(x_n^r+\theta^{r,v})=1$. As before, each node j_k should forward to node j_{k-1} the message

$$\nabla_{x_n^{j_k}+\theta^{j_k\to v}} g^v(Y_n^v|\, x_n^{j_k}+\theta^{j_k,v})|_{x_n^{j_k}=x_n^{j_{k-1}}+\theta^{j_{k-1},j_k}} = \nabla_{x_n^{j_{k-1}}+\theta^{j_{k-1},v}} g^v(Y_n^v|\, x_n^{j_{k-1}}+\theta^{j_{k-1},v})$$

so that $\nabla_{\theta^{r,j}} g^v(Y_n^v|\, x_n^v+\theta^{r,v})$ travels along the path $(v,j_l),(j_l,j_{l-1}),\ldots,(j_2,j_1),(j_1,1)$.

Previously we showed that $\prod_{v\in\mathcal{V}\setminus\{r\}} g^v(Y_n^v|\, x_n^r+\theta^{r,v})$ and $\sum_{v\in\mathcal{V}} \frac{\nabla_{\theta^{r,j}} g^v(Y_n^r|x_n^r+\theta^{r,v})}{g^v(Y_n^r|x_n^r+\theta^{r,v})}$ are needed to be available to root node r as the sufficient statistics to compute $\theta_n^{r,j}$ as well as propagating the densities in (6.8)-(6.11). Therefore, we could define messages from each node in a way that it is not necessary to transmit each $\nabla_{\theta^{r,j}} g^v(Y_n^v|\, x_n^v+\theta^{r,v})$ and $g^v(Y_n^v|\, x_n^v+\theta^{r,v})$ separately in all different possible paths $(v,j_l),(j_l,j_{l-1}),\ldots,(j_2,j_1),(j_1,r)$.

For this reason we inherit a generalized version of Belief Propagation, where messages are defined as $m_n^{i,j}$ and $\mathring{m}_n^{i,j}$ from node i to j for all $(i,j)\in\mathcal{E}$

$$m_n^{i,j}(x_n^j) = g^i(Y_n^i\big|\, x_n^j+\theta^{j,i}) \prod_{p\in \text{ne}(i)\setminus\{j\}} \left(m_n^{p,i}(x_n^i)|_{x_n^j+\theta^{j,i}}\right) \tag{6.12}$$

$$\mathring{m}_n^{i,j}(x_n^j) = \frac{\nabla_{\theta^{j,i}} g^i(Y_n^i\big|\, x_n^j+\theta^{j,i})}{g^i(Y_n^i\big|\, x_n^j+\theta^{j,i})} + \sum_{p\in \text{ne}(i)\setminus\{j\}} \left(\mathring{m}_n^{i,j}(x_n^i)|_{x_n^j+\theta^{j,i}}\right) \tag{6.13}$$

where $\text{ne}(i)$ is the set of neighbouring nodes of i in $\mathcal{V}$.

Equations (6.7)-(6.11) can be reproduced for any other root node r and parameter $\theta^{r,j}$, $(j,r)\in\mathcal{E}$. In addition, we can define for each root node r an appropriate message scheduling, where we choose to use only messages $m_n^{i,j}$ and $\mathring{m}_n^{i,j}$ only for (i,j) directed towards that root node, i.e. start from the outer branches of the Graphical model and leading each time to node r. So after the root node receives its messages from its neighbours it will have available:

$$\prod_{p\in\text{ne}(r)} m_n^{p,r}(x_n^r) = \prod_{v\in\mathcal{V}\setminus\{r\}} g^v(Y_n^v|\, x_n^r+\theta^{r,v}),$$
$$\mathring{m}_n^{j,r}(x_n^r) = \sum_{v\in\mathcal{V}} \frac{\nabla_{\theta^{r,j}} g^v(Y_n^v|\, x_n^r+\theta^{r,v})}{g^v(Y_n^v|\, x_n^r+\theta^{r,v})},$$

as desired.

6.3.3 Distributed RML Algorithm for the Self Localisation problem

In this section we demonstrate how the general distributed parameter estimation problem for sensor localisation can be solved using RML. Using the results of Section 6.3 we estimate $\theta^{r,j}$ for any edge (r,j) using now an arbitrary root node r. At each iteration n all edges should

be updated in a cyclic fashion using a valid root node and hence $\theta^{i,j}$ will be updated for all $(i,j) \in \mathcal{E}$.

Algorithm 6.1 ***Distributed RML Algorithm for the Self Localisation problem:***

For each edge $(j,r) \in \mathcal{E}$ assign a valid root node, say r, so as to update parameter $\theta_n^{r,j}$. ***At time*** *n,*

Prediction Update for all nodes: *For all nodes $r \in \mathcal{V}$ propagate the prediction densities $\pi^r_{n|n-1}(x^r_n)$ as in (6.8) and their derivatives $\nabla_{\theta_n^{r,j}} \pi^r_{n|n-1}(x^r_n)$ as in (6.10).*

Propagate messages: *After Y_n is received pass all possible messages $m_n^{i,j}$ and $\overset{\circ}{m}{}_n^{i,j}$ given by (6.12) and (6.13) for all edges $(i,j) \in \mathcal{E}$ in the network.*

Use messages to compute sufficient statistics: *At each node r compute $\prod_{p \in ne(r)} m_n^{p,r}(x^r)$ and $\overset{\circ}{m}{}_n^{j,r}(x^r)$.*

Update the parameter $\theta^{r,j}$: *At each root node r set*

$$\theta^{r,j}_{n+1} = \theta^{r,j}_n + \gamma^r_n \nabla_{\theta^{r,j}} J^r_{\theta^{r,j}_n},$$

where $\nabla_{\theta^{r,j}} J^r_{\theta^{r,j}_n}$ is given by evaluating expression (6.7) at $\theta^{r,j}_n$.

Update Filtering density and derivative: *Using incoming messages, update at all nodes the current $\pi^r_n(x^r_n)$ as in (6.9) and its derivative $\nabla_{\theta_n^{r,j}} \pi^r_n(x^r_n)$ as in (6.11).*

Typically, the step-sizes are selected as $\gamma^r_n = n^{-\gamma^r}$, where $\gamma^r > 0.5$, so that $\sum_n \gamma^r_n = \infty$ and $\sum_n {\gamma^r_n}^2 < \infty$.

As for standard Belief Propagation [130], this algorithm will only be exact when the graph or network admits a tree structure. Applying belief propagation algorithms to non-tree topologies is generally referred as Loopy Belief Propagation (LBP). In some cases LBP can lead to very good approximations [167]. In the next section, we apply this algorithm to an example using a Gaussian Markov Random field, whose graph topology has a single loop and we demonstrate in simulations that this loopy version can exhibit very good performance.

6.4 Numerical Examples

In this section we shall demonstrate how to solve the sensor self-localisation problem using our framework. We will be using the ideal algorithm of Section 6.3.3 to solve the sensor network

self localisation problem for the linear Gaussian case and a nonlinear example using a particle implementation.

6.4.1 A Linear Gaussian Example

We consider an M node sensor network. At each node r, the target being tracked yields observation Y_n^r and obeys the following dynamics

$$X_n^r = AX_{n-1}^r + BV_n^r,$$
$$Y_n^r = CX_n^r + DW_n^r$$

with $V_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, Q)$ and $W_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, R)$.

Thanks to the linear and Gaussian assumptions, we have at time n

$$\pi_{n|n-1}^r(x_n^r) = \mathcal{N}_{x_n^r}(\mu_{n|n-1}^r, \Sigma_{n|n-1}^r)$$
$$\pi_n^r(x_n^r) = \mathcal{N}_{x_n^r}(\mu_{n|n}^r, \Sigma_{n|n}^r)$$

whose parameters can be computed using a distributed Kalman filter recursion for each node r

$$\begin{aligned}
\mu_{n|n-1}^r &= A\mu_{n-1|n-1}^r \\
\Sigma_{n|n-1}^r &= A\Sigma_{n-1|n-1}^r A^T + BQB^T \\
m_n^r &= C\mu_{n|n-1}^r \\
S_n^r &= C\Sigma_{n|n-1}^r C^T + M^{-1}DRD^T \\
K_n^r &= \Sigma_{n|n-1}^r C^T S_n^{r-1} \\
\mu_{n|n}^r &= \mu_{n|n-1}^r + K_n^r(M^{-1}\sum_{i\in\mathcal{V}}(Y_n^i - C\theta^{r,i}) - m_n) \\
\Sigma_{n|n}^r &= \Sigma_{n|n-1}^r - K_n^r C\Sigma_{n|n-1}^r
\end{aligned}$$

As far as the collaborative filtering part is concerned, it is only necessary to propagate the mean and covariance of these densities. For the localisation part of the problem, given $\nabla_{\theta^{r,j}}\mathcal{N}_{x^r}(m,\Sigma) = (\nabla_{\theta^{r,j}}m)^T\mathcal{N}_{x^r}(m,\Sigma)\Sigma^{-1}(x-m)$, we only need to propagate $\nabla_{\theta^{r,j}}\mu_{n|n-1}^r$ and $\nabla_{\theta^{r,j}}\mu_{n|n}^r$ in order to propagate the derivatives of the densities. To update each edge parameter $\theta^{r,j}$, we can use the analytical expression of $p(Y_n|Y_{0:n-1})$ to obtain the derivative of $\log p(Y_n|Y_{0:n-1})$. We present all analytical calculations for deriving $\nabla_{\theta^{r,j}}J_{\theta^{r,j}}^r$ in the appendix of this chapter.

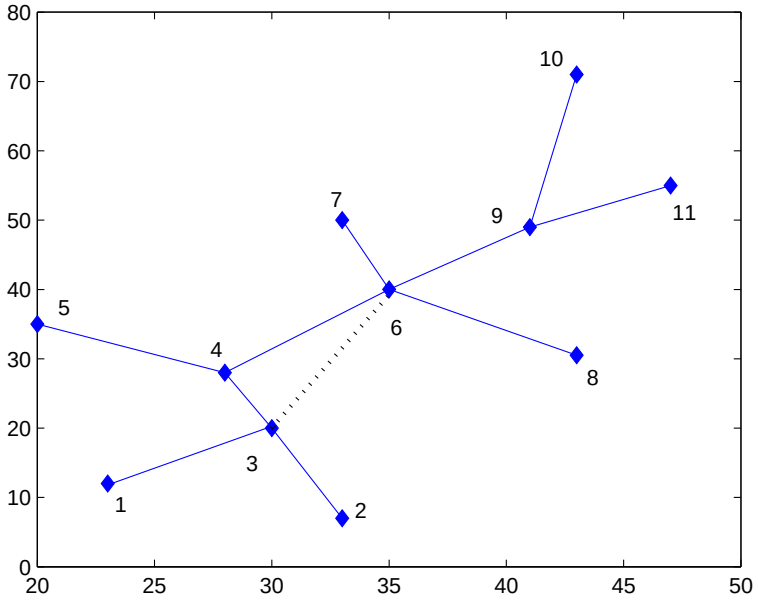


Figure 6.6: *Sensor Network used for target tracking; in the numerical example its localisation parameter θ is estimated.*

We can define message m as

$$m_n^{i,j} = [Y_n^i - C\theta_n^{j,i}] + \sum_{p \in \text{ne}(i) \setminus \{j\}} m_n^{p,i}$$

in order to be able to obtain $\sum_{i \in \mathcal{V}} (Y_n^i - C\theta^{r,i})$ at each node r. Also we define message $\overset{\circ}{m}$ as

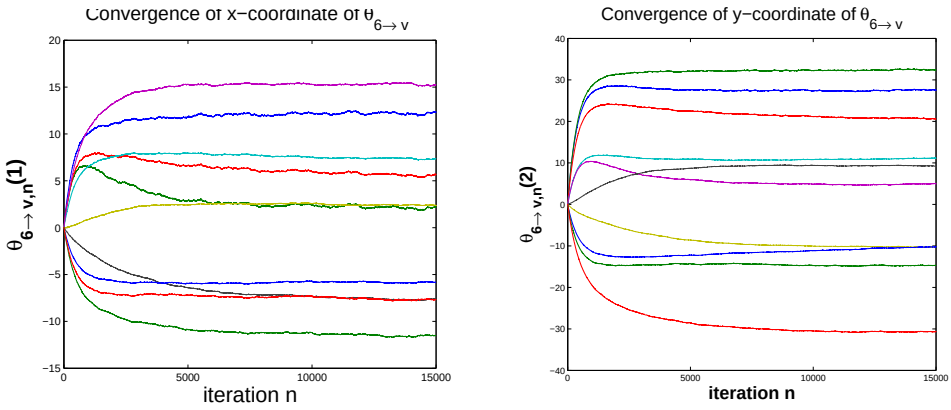
$$\overset{\circ}{m}{}_n^{i,j} = 1 + \sum_{p \in \text{ne}(i) \setminus \{j\}} \overset{\circ}{m}{}_n^{p,i}.$$

The cardinality of the graph, M, is given at each node i by $\sum_{p \in \text{ne}(i)} \overset{\circ}{m}{}_n^{p,i}$.

For the linear Gaussian example we shall be investigating we use

$$A = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{\tau^2}{2} & 0 \\ \tau & 0 \\ 0 & \frac{\tau^2}{2} \\ 0 & \tau \end{bmatrix}, Q = 3I, C = I, D = I, R = 5I,$$

where I is an appropriate identity matrix. The choice of A, B, Q are a common practice modeling the target dynamics in the target tracking literature, [13]. The measurement parameters,



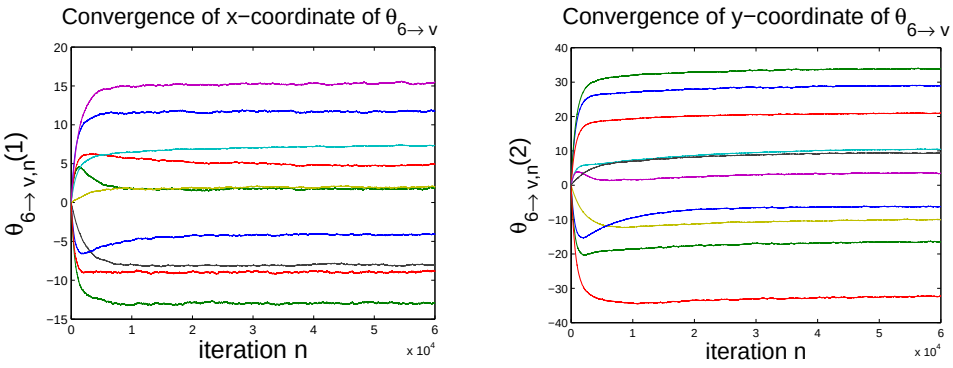
(a) X-coordinate of $\theta_n^{6,v}$ (b) Y-coordinate of $\theta_n^{6,v}$

Figure 6.7: *Convergence of x- and y- coordinates of $\theta_n^{6,v}$ for the sensor net with dotted line not connected.*

C, D, Q, should actually depend on the particular specifications of each sensor used, but for simplicity we choose these values.

For the sensor network in Figure 6.6 ignore first the dotted line between nodes 3 and 6. In this case, the graph has a tree structure and we can solve for any $\theta^{i,j}$ exactly. We choose nodes $\{3, 4, 6, 9\}$ as root nodes and update at each iteration their adjacent edges. Also, we use $\tau = 0.1$. For practical implementation reasons we choose to use a constant step size $\gamma_n = 10^{-3}$. For stochastic approximation in general, decreasing step-sizes are essential conditions of convergence. If fixed step-sizes are used, then we may still have convergence, but now the iterates "oscillate" about their limiting values with variance proportional to the step-size. We also initialise $\theta^{i,j} = 0$ for all $(i, j) \in \mathcal{E}$. In Figures 6.7(a), 6.7(b) we illustrate the convergence to the correct values of all the localisation parameters relative to node 6, $\theta_n^{6,v}$, for all $v \in \mathcal{V}$.

Next we solve for the sensor network in Figure 6.6 but with the dotted line connected. Now the sensor network has a loop in its topology defined by $\mathcal{E}$ at $\{3, 4, 6\}$. We choose again nodes $\{3, 4, 6, 9\}$ as root nodes and update at each iteration their adjacent edges. As before, we choose to use a constant step size $\gamma_n = 10^{-3}$ and initialise $\theta^{i,j} = 0$. In Figures 6.8(a), 6.8(b) we illustrate the convergence of all the localisation parameters relative to node 6, $\theta_n^{6,v}$ when LBP is used. Note that the errors of the solution for $\theta_n^{6,v}$ are very small, but the convergence rate is now slower.



(a) LBP, X-coordinate of $\theta_n^{6,v}$

(b) LBP, Y-coordinate of $\theta_n^{6,v}$

Figure 6.8: *Convergence of y-coordinate of $\theta_n^{6,v}$ when LBP is applied to the sensor net with the dotted line connected.*

6.4.2 A Nonlinear Example using SMC

The tree version of the same sensor network of Figure 6.6 will be used. We would like to solve the self localisation problem using bearings only tracking. At each node r, the target being tracked yields observation Y_n^r and obeys the following dynamics

$$X_n^r = AX_{n-1}^r + BV_n^r,$$
$$Y_n^r = tan^{-1}(X_n^r(1)/X_n^r(3)) + W_n^r$$

with $V_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, Q)$ and $W_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_y^2)$. At each time all measurements at each node are independent, so at each node r we have $f(x_n^r|x_{n-1}^r) = \mathcal{N}_{x_n^r}(Ax_{n-1}^r, BQB^T)$ and $g^r(Y_n^r|x_n^r) = \mathcal{N}_{Y_n^r}(tan^{-1}(x_n^r(1)/x_n^r(3)), \sigma_y^{-2})$.

Filtering with bearing only tracking is a well known problem. Observing a target using only bearings measurements can lead in some cases to certain problems concerning the observability of the target, especially when abrupt maneuvers take place. This was studied in [121]. As we are using multiple measurements from different sensors, we shall not face such or similar problems, such as ambiguities of where the target could be located. On the other hand we are more interested whether these measurements provide enough information on the relative location of the nodes. This has appeared in the wireless networks literature, in [129]. In [146] there has been a study on how nodes of network can be localised with the aid of a mobile beacon of known positions. It was found that the motion of that beacon should have an adequate number of turns so that all nodes could be discovered without any ambiguity. As the scope of our

work is to show how our setting can be used to perform collaborative filtering and localisation for sensor networks on a difficult nonlinear scenario, such as bearings only tracking, and not to present complex methods for resolving any observability issues, we opt for using successive targets that pass through the monitored area instead of having to design a single target of desired properties. These targets should not coincide in time and we shall limit ourselves to the single target at each time scenario avoiding any data association problems. This approach could be viewed as a realistic representation of the problem of using sensor networks to track unmanned aerial vehicles (UAVs). Note that we shall use the same approach in the next two chapters when dealing with the bearings only tracking problem.

The algorithm of Section 6.3.3 relies on the evaluation of complex multi-dimensional integrals given in equations (6.7)-(6.11). In the general non linear non Gaussian case this is impossible and one must rely on approximation. It is important to emphasise that we have not made any linearity or Gaussianity assumption in our framework. In [133, 134] a centralised implementation of a particle filter for RML has been derived. This implementation can be extended to our distributed framework. In the above algorithm π_n^r and $\pi_{n|n-1}^r$ will have to be replaced by their particle approximations. Particle approximations can be derived for the gradients as well. Unfortunately, implementing a particle based message passing algorithm is not trivial. In this section only, we will assume that each node r has $[Y_n^v]_{v\in\mathcal{V}}$, $[\theta^{r,v}]_{v\neq j,(r,v)\in\mathcal{E}}$ available to itself, and can solve for $\theta^{r,j}$, where $(r,j)\in\mathcal{E}$ without the need of message passing. This means that at each node r we are solving for the parameter of its adjacent edge assuming that r has knowledge of all the necessary variables. Thus, the centralised problem is solved at each node with respect its own frame of reference. We we will implement the algorithm of [133, 134] for this approach. Of course this approach does not consist of a proper decentralised solution and lacks scalability. The main reason of presenting this here is that it can be used as motivation for the development of proper decentralised particle methods based on message passing, which will be presented later in this thesis.

We choose nodes $\{3,4,6,9\}$ as root nodes and update at each iteration their adjacent edges. We use A, B, Q be as before. We choose to use a constant step size $\gamma_n = 10^{-4}$ and also initialise $\theta^{i,j} = 0$ for all $(i,j)\in\mathcal{E}$. In Figure 6.9 (a), (b) we present the plot $\theta_*^{i,j} - \theta_n^{i,j}$ against the iteration number n, i.e. the error at each iteration between the true parameter and the value of the current iteration of the localisation parameter for all edges $(i,j)\in\mathcal{E}$, when $L = 3000$ number of

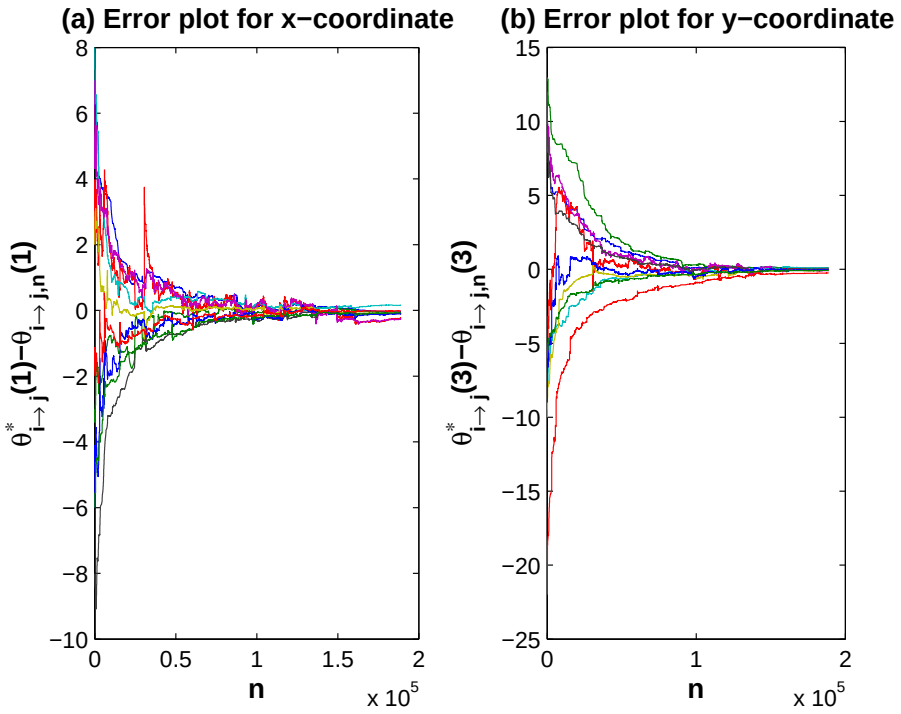


Figure 6.9: *Error at each iteration between the true parameter and current update of the localisation parameter for all edges of the sensor net.*

particles are used. In Table 1 we present the total mean squared error (MSE) of all the estimated

Number of Particles	Total MSE
1000	0.1228
3000	0.0437
5000	0.0284

Table 6.1: *Mean squared errors after convergence when using different number of particles.*

localisation parameters for each of the edges in the graph after convergence is met for different number of particles used.

6.5 Conclusion

In this chapter, we have presented a method for performing recursive static parameter estimation in dynamic Graphical models. We describe how a completely decentralised version of RML can be implemented in dynamic Graphical models through the propagation of suitable messages that are exchanged between neighbouring nodes of the graph. The resulting algorithm can be interpreted as a generalization of the Belief Propagation algorithm to compute likelihood gradients. We have used this approach to formulate the sensor registration problem and proposed an algorithm to solve the sensor localisation problem. For linear Gaussian graphs, our algorithm can be implemented exactly using a distributed version of the Kalman filter and its derivative. In the general non linear and non Gaussian case, Sequential Monte Carlo can be used.

As it was commented on Section 6.4, the generic algorithm to solve the localisation problem in Section 6.3.3 relies on a certain substitution when computing the messages, see equations (6.12)-(6.13). This seems to be a great disadvantage as we cannot obtain a general solution, where each node computes the messages to be passed on to its neighbours using only its local parameters and its received messages. This unfortunately limits the use of this algorithm for large scale sensor networks, with hundreds or possibly thousands of nodes. In order to improve this we have further studied the problem in next two Chapters and propose improvements to obtain proper scalable algorithms, where each node uses solely functions of its received messages and local variables to pass messages, estimate the localisation parameters of its neighbours relative to itself, enabling in this manner collaborative filtering.

Another issue that has not been investigated in the numerical examples is the use of different observation densities at each node. This is quite important as our framework does not make any assumptions like all nodes performing the same type of measurements. The use of different sensors with different types of measurements is an important aspect for any real application. Moreover, we are also interested to show that our methodology can be used also in time varying state space models. We will address these issues in the next two Chapters.

6.A Appendix - Linear Gaussian Example

In this section we shall derive the analytical expressions for $J^r_{\theta^{r,j}}$ and $\nabla_{\theta^{r,j}} J^r_{\theta^{r,j}}$ for the linear gaussian problem of Section 6.4. We start from the definition of the recursive likelihood as

$$\begin{aligned} J^r_{\theta^{r,j}} &= p(Y_n|Y_{0:n-1}) \\ &= \int \prod_{v\in\mathcal{V}} g^v(Y^v_n|\, x^r_n + \theta^{r,v}) \pi^r_{n|n-1}(x^r_n) dx^r_n, \end{aligned}$$

where we have dropped the time subscript on x^r for simplicity. For the linear Gaussian example described in Section 6.4 we have $\pi^r_{n|n-1}(x^r_n) = \mathcal{N}_{x^r}(\mu^r_{n|n-1}, \Sigma^r_{n|n-1})$ and $g^v(Y^v_n|x^r_n + \theta^{r,v}) = \mathcal{N}_{Y^v_n}(C(x^r_n + \theta^{r,v}), DRD^T)$. Let $\Sigma^Y = DRD^T$. Therefore

$$\begin{aligned} p(Y_n|Y_{0:n-1}) = \frac{1}{Z} \int \exp(-\frac{1}{2} \sum_{v\in\mathcal{V}} (Y^v_n - C(x^r_n + \theta^{r,v}))^T \Sigma^{Y^{-1}} (Y^v_n - C(x^r_n + \theta^{r,v})) \\ -\frac{1}{2}(x^r_n - \mu^r_{n|n-1})^T \Sigma^{r^{-1}}_{n|n-1}(x^r_n - \mu^r_{n|n-1}))dx^r_n, \end{aligned} \tag{6.14}$$

where Z is the appropriate normalisation constant. For the terms inside the integral, we shall use the following identity

$$\int \exp(-\frac{1}{2} x^T \mathcal{A} x + b^T x) dx = \sqrt{\det(2\pi\mathcal{A}^{-1})} \exp(\frac{1}{2} b^T \mathcal{A}^{-1} b).$$

It is trivial to show that in our case we have

$$\begin{aligned} \mathcal{A} &= MC^T \Sigma^{Y^{-1}} C + \Sigma^{r^{-1}}_{n|n-1}, \\ b &= C^T \Sigma^{Y^{-1}} \sum_{v\in\mathcal{V}} (Y^v_n - C\theta^{r,v}) + \Sigma^{r^{-1}}_{n|n-1} \mu^r_{n|n-1}, \end{aligned}$$

where M is the number of nodes in the graph. Note that even if they are not needed for the integration, we should not abandon the two remaining quadratic terms that come from the expansion of the lhs of (6.14). So, for the likelihood we can write

$$p(Y_n|Y_{0:n-1}) \propto \exp(\frac{1}{2} b^T \mathcal{A}^{-1} b - \frac{1}{2} \mu^{r^T}_{n|n-1} \Sigma^{r^{-1}}_{n|n-1} \mu^r_{n|n-1} - \frac{1}{2} \sum_{v\in\mathcal{V}} (Y^v_n - C\theta^{r,v})^T \Sigma^{Y^{-1}} (Y^v_n - C\theta^{r,v})),$$

and for the log likelihood

$$\log p(Y_n|Y_{0:n-1}) = \frac{1}{2} b^T \mathcal{A}^{-1} b - \frac{1}{2} \mu^{r^T}_{n|n-1} \Sigma^{r^{-1}}_{n|n-1} \mu^r_{n|n-1} - \frac{1}{2} \sum_{v\in\mathcal{V}} (Y^v_n - C\theta^{r,v})^T \Sigma^{Y^{-1}} (Y^v_n - C\theta^{r,v}).$$

Note also

$$\begin{aligned} \nabla_{\theta^{r,j}} J^r_{\theta^{r,j}} &= \nabla_{\theta^{r,j}} \log p(Y_n|Y_{0:n-1}) \\ &= (\nabla_{\theta^{r,j}} b)^T \mathcal{A}^{-1} b - (\nabla_{\theta^{r,j}} \mu^r_{n|n-1})^T \Sigma^{r^{-1}}_{n|n-1} \mu^r_{n|n-1} - M^r C^T \Sigma^{Y^{-1}} \sum_{v\in\mathcal{V}} (Y^v_n - C\theta^{r,v}), \end{aligned}$$

with

$$\nabla_{\theta^{r,j}} b = M^r C^T \Sigma^{Y^{-1}} C + \Sigma_{n|n-1}^{r^{-1}} \nabla_{\theta^{r,j}} \mu_{n|n-1}^r,$$

where M^r is the number of nodes in the subgraph commencing from j away from node r. Also, note that

$$M^r = \overset{\circ}{m}_n^{j,r},$$

so there is no need to define an extra message around the network to obtain M^r.

7

Distributed Localisation and Tracking for Linear Gaussian Sensor Networks

Summary. *Recursive Maximum Likelihood (RML) and Expectation Maximisation (EM) are a popular methodologies for estimating unknown static parameters in state-space models. We describe how a completely decentralized version of RML and EM can be implemented in dynamic Graphical models through the propagation of suitable messages that are exchanged between neighboring nodes of the graph. The resulting algorithm can be interpreted as an extension of the Belief Propagation algorithm to compute likelihood gradients. This algorithm is applied to solve the sensor localisation problem for sensor networks without loops.*

7.1 Introduction

Figure 7.1 depicts a sensor network that is deployed to perform target tracking. The network is comprised of sensor-trackers where each node in the network has the processing ability to perform the computations needed for target tracking. The lines joining the nodes indicate communication links and defines the neighborhood structure of the network. It is assumed that a sensor can only communicate with its neighboring nodes. A moving target will be simultaneously observed by more than one sensor. If the target is within the field-of-view of a sensor, then that sensor will collect measurements of the target . In a centralized architecture all the sensors transmit their measurements to a central fusion node, which then combines them and computes the estimate of the target's trajectory. The interest however is to perform *collaborative tracking* but without the need for a central fusion node. Loosely speaking, in such networks nodes collaborate by exchanging appropriate messages between neighboring nodes to achieve the same effect as they would by communicating with a central fusion node.

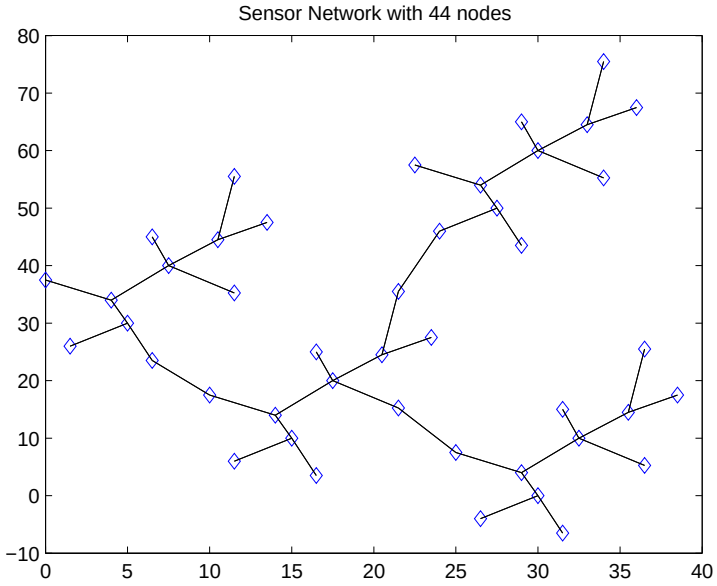


Figure 7.1: *Sensor Network used for target tracking*

Distributed collaborative tracking can be achieved if each node is able to accurately determine the position of its neighboring nodes in its local frame of reference. (More details in Section 7.2.) This is essentially an instance of the *self-localisation* problem [1]. In this chapter we

solve the localisation problem without the need of a Global Positioning System (GPS) or direct measurements of the distance between neighboring nodes. The latter is usually estimated from the Received Signal Strength (RSS) when each node is equipped with a wireless transceiver. The method we propose is significantly different. Essentially, the very task that the network is deployed for, which is collaborative tracking, will be exploited to achieve self-localisation in a completely decentralized manner. Initially as nodes are not localised they behave as independent trackers. As the tracking task is performed on objects that traverse the field of view of the sensors, information is shared between nodes in a way that allows them to self-localise. Even though the target's true trajectory is not known to the sensors, localisation can be achieved in this manner because the same target is being simultaneously measured by the sensors. This simple fact, which seems to have been largely overlooked in the literature, is the basis of our solution.

This idea of using measurements of a common target to solve the self localisation problem has also been independently developed by [59, 159]. However, our work differs from these in the application studied as well as the inference scheme. Both [59, 159] formulate the localisation as a posterior inference problem and approximate the distributions of interest with simple Gaussians. In this work we develop fully *decentralized* versions of the two most common likelihood inference techniques, namely Recursive Maximum Likelihood (RML) [134] and Expectation-Maximisation (EM) [40]. Maximum Likelihood is the most popular approach to parameter estimation in Hidden Markov Models [30] and these techniques have not been previously developed for the self-localisation problem. Our work addresses this shortfall. We validate our approach in simulation with a large network of bearings-only trackers. The RML algorithm we develop admits an online implementation which is important for the application studied.

Most tracking problems are essentially non-linear non-Gaussian filtering problems and Sequential Monte Carlo (SMC) methods, also known as Particle Filters, provide very good approximations to the filtering densities [47]. While it is possible to develop SMC versions of our algorithms, in the interest of execution speed, we use instead a linearization procedure similar to the Extended Kalman filter when dealing with a non-linear system. The decentralized solution to the self-localisation and collaborative tracking problem necessitates the use of *belief propagation*, which is a message passing algorithm widely used in the computer science liter-

ature to perform inference on graphs [130]. However belief propagation computes posterior distributions only while we also require a fully decentralized algorithm for calculating the gradient of the log-likelihood function. We propose a message passing scheme similar to belief propagation for doing so which is, to the best of our knowledge, novel.

There is a sizeable literature on the self-localisation problem. Furthermore, the topic has been independently pursued by researchers working in different application areas, most notably wireless communications [79, 129, 137] and sensor networks for environmental monitoring [123]. Although all these works tend to be targeted for the application at hand and differ in implementation specifics, they may however be broadly summarized as follows. There are many works that rely on direct measurements between neighboring nodes [79, 129, 137]. Given such measurements, it is then possible to solve for the geometry of the sensor network but with ambiguities in translation and rotation of the entire network remaining. These ambiguities can be removed if the absolute position of certain nodes, referred to as anchor nodes, are known. There are non-statistical based approaches of this idea, like least squares [137], and statistical ones based on statistical inference [79, 129]. There are also methods that utilize *beacon* nodes which have either been manually placed at precise locations or are equipped with a GPS. The un-localised nodes will use the signal broadcast by these beacon nodes to self-localise [129].

The related problem of *sensor registration* which aims to compensate for systematic biases in the sensors has been studied by the target tracking community [124, 165]. The algorithms devised therein are centralized. There is also the related problem of average consensus [170]. The value of a global static parameter is measured at each node via a linear Gaussian observation model and the aim is to obtain a maximum likelihood estimate in a distributed fashion. This is not a distributed localisation and tracking task. Moreover, the work of [15] is based on the principle shared by our approach and [59, 159] . The authors exploit the correlation of the measurements made by the various sensors of a hidden spatial process to perform self-localisation. However for reasons concerned with the applications being addressed, which is not distributed target tracking, their method is not on-line and centralized in nature.

The structure of the chapter is as follows. We begin with the specification of the statistical model for the localisation and tracking problem in Section 7.2. In Section 7.3 we show how message passing may be utilized to perform distributed filtering and smoothing. In Section 7.4 we derive the distributed RML and EM algorithms. Section 7.5 presents several numer-

ical examples on small and medium sized networks. The Appendix contains more detailed derivations of the distributed versions of RML and EM.

7.2 Problem Formulation

We consider the sensor network $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of nodes of the network and $\mathcal{E}$ is the set of edges (or communication links between nodes.) In this chapter we assume that the sensor network has no loop. Nodes $i, j \in \mathcal{V}$ are connected provided the edge $(i, j) \in \mathcal{E}$ exists. Also, we will assume that if $(i, j) \in \mathcal{E}$ holds, then $(j, i) \in \mathcal{E}$ holds as well. The nodes observe the same physical target at discrete time intervals $n \in \mathbb{N}$. Note though that the target may only be in the field of view of a limited number of sensors at any given time. The hidden state, as is standard in target tracking, is defined to comprise of the position and velocity of the target,

$$X_n^r = [X_n^r(1), X_n^r(2), X_n^r(3), X_n^r(4)]^{\mathrm{T}},$$

where $X_n^r(1)$ and $X_n^r(3)$ is the target's x and y position while $X_n^r(2)$ and $X_n^r(4)$ is the velocity in the x and y direction. Subscript n denotes time while superscript r denotes the coordinate system w.r.t. which these quantities are defined. For generality we assume that each node maintains a local coordinate system (or frame of reference) and regards itself as the origin (or center of) its coordinate system.

As a specific example, consider the following linear Gaussian model:

$$X_n^r = A_n X_{n-1}^r + b_n^r + V_n^r, \quad n \geq 1, \tag{7.1}$$

where V_n^r is zero mean Gaussian additive noise with variance Q_n and b_n^r are deterministic inputs. The measurement Y_n^r made by node r is also defined relative to the local coordinate system at node r. For a linear Gaussian observation model the measurement is generated as follows:

$$Y_n^r = C_n^r X_n^r + d_n^r + W_n^r, \quad n \geq 1, \tag{7.2}$$

where W_n^r is zero mean Gaussian additive noise with variance R_n^r and d_n^r is deterministic. Note that the time varying observation model $\{(C_n^r, d_n^r, R_n^r)\}_{n\geq 1}$ is different for each node. A time varying state and observation model is retained for an Extended Kalman Filter (EKF) implementation for the non-linear setting to be defined in the next chapter. It is in this setting that the need for sequences $\{b_n^r\}_{n\geq 1}$ and $\{d_n^r\}_{n\geq 1}$ arises.

The more general non-linear non-Gaussian setting can be expressed with the following Hidden Markov Model (HMM),

$$X_n^r | X_{n-1}^r = x_{n-1}^r \sim f_n^r(.|x_{n-1}^r), \tag{7.3}$$

$$Y_n^r | X_{n-1}^r = x_n^r \sim g_n^r(.|x_n^r). \tag{7.4}$$

where X_n^r is the hidden state and $Y_n^r \in \mathbb{R}^{d_y^r}$ is the measurement made by node r at time n. The target's transition model $f_n^r(.|.)$ is assumed time varying for generality. The observation model (7.4) is time varying and is different for each node. Also, the dimension of the observation vector Y_n^r need not be the same for different nodes since each node may be equipped with a different sensor type. For example, node r may obtain measurements of the target's position while node v measures bearing.

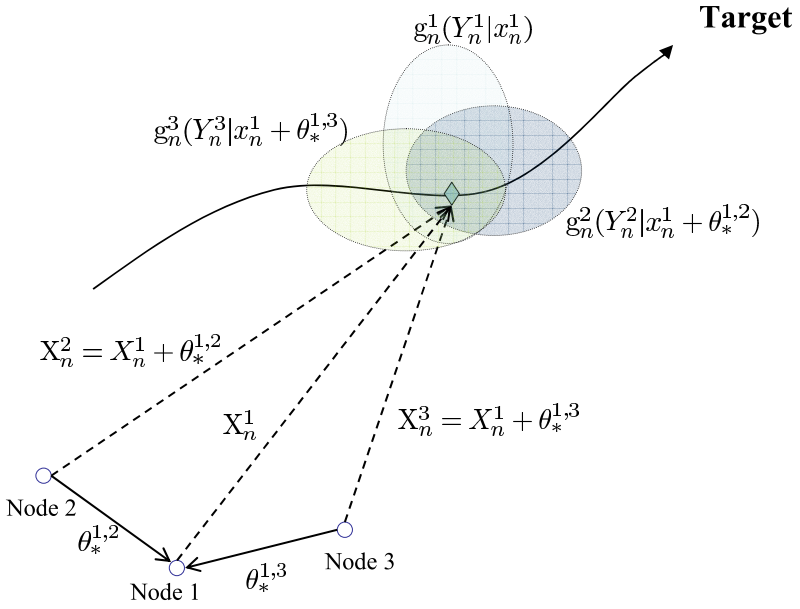


Figure 7.2: *A three node network tracking a target traversing its field of view. The trajectory of the target is shown with the solid line. Each node regards itself as the center of its local coordinate system. At time n a measurement is registered by all three nodes. The ellipses show the support of the observation densities for the three nodes, i.e. the support of $g_n^1(Y_n^1|.)$ is defined as all x_n^1 such that $g_n^1(Y_n^1|x_n^1)$; similarly for the rest. The filtering update step at node 1 will clearly benefit from the observations made by nodes 2 and 3. The localization parameters $\theta_*^{1,2}$, $\theta_*^{1,3}$ are the coordinates of node 1 in the local coordinate systems of node 2 and 3 respectively. While X_n^r was defined to be the state of the target, which includes its velocity, for this illustration only, X_n^r is to be understood as the position of the target at time n w.r.t. the coordinate system of node r.*

Figure 7.2 illustrates a three node setting where a target is being jointly observed and

tracked by three sensors. (Only the position of the target is shown.) At node 1, X_n^1 is defined relative to the local coordinate system of node 1 which regards itself as the origin. Similarly for nodes 2 and 3. We define $\theta_*^{i,j}$ to be the position of node i *in the local coordinate system* of node j. This means that the vector X_n^i relates to the local coordinate system of node j as follows (see Figure 7.2):

$$X_n^j = X_n^i + \theta_*^{i,j}.$$

The *localisation* parameters $\{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}$ are static as the nodes are not mobile. We note the following obvious but important relationship: if nodes i and j are connected through intermediate nodes $j_1, j_2, \ldots, j_m$ then

$$\theta_*^{i,j} = \theta_*^{i,j_1} + \theta_*^{j_1,j_2} + \theta_*^{j_2,j_3} + \ldots + \theta_*^{j_{m-1},j_m} + \theta_*^{j_m,j}. \tag{7.5}$$

This relationship is exploited to derive the distributed filtering and localisation algorithms in the next section. We define $\theta_*^{i,j}$ so that the dimensions are the same as the target state vector. When the state vector is comprised of the position and velocity of the target, only the first and third components of $\theta_*^{i,j}$ are relevant while the other two are redundant and set to $\theta_*^{i,j}(2) = 0$ and $\theta_*^{i,j}(4) = 0$. Let

$$\theta_* \equiv \{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}, \quad \theta_*^{i,i} \equiv 0, \tag{7.6}$$

where $\theta_*^{i,i}$ for all $i \in \mathcal{V}$ is defined to be the zero vector.

Let Y_n denote all the measurements received by the network at time n, i.e. $Y_n \equiv \{Y_n^v\}_{v\in\mathcal{V}}$. We also denote the sequence $(Y_1, ..., Y_n)$ by $Y_{1:n}$. In the *collaborative* filtering problem, each node r computes the local filtering density:

$$p_{\theta_*}^r(x_n^r|Y_{1:n}) \propto p_{\theta_*}^r(Y_n|x_n^r) p_{\theta_*}^r(x_n^r|Y_{1:n-1}), \tag{7.7}$$

where $p_{\theta_*}^r(x_n^r|Y_{1:n-1})$ is the predicted density and is related to the filtering density of the previous time through the following prediction step:

$$p_{\theta_*}^r(x_n^r|Y_{1:n-1}) = \int f_n^r(x_n^r|x_{n-1}^r) p_{\theta_*}^r(x_{n-1}^r|Y_{1:n-1}) dx_{n-1}^r. \tag{7.8}$$

The likelihood term is

$$p_{\theta_*}^r(Y_n|x_n^r) = \prod_{v\in\mathcal{V}} g_n^v(Y_n^v|x_n^r + \theta_*^{r,v}), \tag{7.9}$$

where $\theta_*^{v,v}$ for all $v \in \mathcal{V}$ is defined to be the zero vector by default. The superscript on the densities indicate the coordinate system they are defined w.r.t. (or node the density belongs to)

while the subscript makes explicit the dependence on the localisation parameters. The prediction step in (7.8) can be implemented locally at each node without exchange of information but the update step in (7.7) incorporates all the measurements of the network. Figure 7.2 shows the support of the three observation densities as ellipses where the support of $g_n^1(Y_n^1|\cdot)$ is defined to be all x^1 such that $g_n^1(Y_n^1|\cdot) > 0$; similarly for the rest. The filtering update step at node 1 can only include the observations made by nodes 2 and 3 provided the localisation parameters $\theta_*^{1,2}$ and $\theta_*^{1,3}$ are known locally to node 1, since the likelihood $p_{\theta_*}^1(Y_n|x_n^1)$ defined in (7.9) is

$$g_n^1(Y_n^1|x_n^1)g_n^2(Y_n^2|x_n^1+\theta_*^{1,2})g_n^3(Y_n^3|x_n^1+\theta_*^{1,3}).$$

The term collaborative filtering is used since each sensor benefits from the observation made by all the other sensors. As is shown in Section 7.3, it is possible to implement collaborative filtering in a truly distributed manner, i.e., each node executes a message passing algorithm (with communication limited only to neighboring nodes) that is scalable with the size of the network. However collaborative filtering hinges on knowledge of the localisation parameters $\{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}$ which are unknown *a priori*. We propose estimation algorithms based on RML and EM to learn the localisation parameters. RML is an online algorithm and refines the parameter estimates as new data arrives while EM is a batch algorithm that runs once a batch of observations have been acquired. These proposed algorithms in this context are to the best of our knowledge novel.

We end this section with a simple (but key) lemma concerning the aggregation of sufficient statistics locally at each node. Consider a sequence of localisation parameters $\{\theta_n\}_{n\geq 0}$ where it is assumed that $\theta_n^{i,j}$ is known to nodes i and j only. Let $(\{F_n^v\}_{v\in\mathcal{V}})_{n\geq 0}$ be a sequence of matrices where F_n^v is known to node v only.

Lemma 7.2.1 *Consider the task of computing $\sum_{v\in\mathcal{V}} F_n^v$ and $\sum_{v\in\mathcal{V}} F_n^v\theta_n^{r,v}$ at each node r of the network. Define the following messages which are to be communicated between all pairs of neighboring nodes in both directions at each time n:*

$$m_n^{i,j} = F_n^i + \sum_{p\in ne(i)\setminus\{j\}} m_n^{p,i}, \tag{7.10}$$

$$\ddot{m}_n^{i,j} = m_n^{i,j}\theta_n^{j,i} + \sum_{p\in ne(i)\setminus\{j\}} \ddot{m}_n^{p,i}. \tag{7.11}$$

It follows then that $\sum_{v\in\mathcal{V}} F_n^v = F_n^r + \sum_{j\in ne(r)} m_n^{j,r}$ while $\sum_{v\in\mathcal{V}} F_n^v\theta_n^{r,v} = \sum_{j\in ne(r)} \ddot{m}_n^{j,r}$.

Here ne(i) denote the neighbors of node i excluding node i itself. A message from node i to j (the source node is indicated by the first letter of the superscript) can be sent once node i has received message from all its neighbors except node j. Thus the leaf nodes of the network (or nodes with only one neighbor) will originate the messages.

7.3 Distributed Collaborative Filtering and Smoothing

For a linear Gaussian system, the collaborative filter $p_\theta^v(x_n^v|Y_{1:n})$ at node v is a Gaussian distribution with mean vector μ_n^v and covariance Σ_n^v. The derivation of the Kalman filter to implement $p_\theta^v(x_n^v|Y_{1:n})$ is standard upon noting that the measurement model at node v can be written as $Y_n = C_n X_n^v + \tilde{d}_n + W_n$ where the r-th block of Y_n, Y_n^r, satisfies $Y_n^r = C_n^r(X_n^v + \theta^{v,r}) + d_n^r + W_n^r$. However, there will be "non-local" steps due to the requirement that quantities $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i$, $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i$ and $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i\theta^{v,i}$ be available locally at node v. To solve this problem, we may use Lemma 7.2.1.

We summarise the result of these steps with the following distributed Kalman filter which is to be implemented at every node of the network. To aid the development of the distributed RML algorithm in Section 7.4, we assume the localisation parameter $\{\theta_n\}_{n\geq 1}$ is time varying but known to the relevant nodes they belong to at time n.

Algorithm 7.1 ***Distributed Filtering***

At time n, let the localization parameter be θ_n and the set of collected measurements be $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$. Exchange the messages $(m_n^{i,j}, \dot{m}_n^{i,j}, \ddot{m}_n^{i,j})$ and $(m_n^{j,i}, \dot{m}_n^{j,i}, \ddot{m}_n^{j,i})$ defined below between all neighboring nodes $(i,j)\in\mathcal{E}$:

$$m_n^{i,j} = (C_n^i)^T(R_n^i)^{-1}C_n^i + \sum_{p\in ne(i)\setminus\{j\}} m_n^{p,i}, \tag{7.12}$$

$$\dot{m}_n^{i,j} = (C_n^i)^T(R_n^i)^{-1}Y_n^i + \sum_{p\in ne(i)\setminus\{j\}} \dot{m}_n^{p,i}, \tag{7.13}$$

$$\ddot{m}_n^{i,j} = m_n^{i,j}\theta_n^{j,i} + \sum_{p\in ne(i)\setminus\{j\}} \ddot{m}_n^{p,i}, \tag{7.14}$$

Update the local filtering densities at each node $r \in \mathcal{V}$:

$$\mu^r_{n|n-1} = A_n \mu^r_{n-1}, \quad \Sigma^r_{n|n-1} = A_n \Sigma^r_{n-1} A_n^T + Q_n, \tag{7.15}$$

$$M^r_n = (\Sigma^r_{n|n-1})^{-1} + (C^r_n)^T (R^r_n)^{-1} C^r_n + \sum_{i \in ne(r)} m^{i,r}_n \tag{7.16}$$

$$z^r_n = (\Sigma^r_{n|n-1})^{-1} \mu^r_{n|n-1} + (C^r_n)^T (R^r_n)^{-1} Y^r_n + \sum_{i \in ne(r)} \dot{m}^{i,r}_n - \ddot{m}^{i,r}_n, \tag{7.17}$$

$$\Sigma^r_n = (M^r_n)^{-1}, \quad \mu^r_n = \Sigma^r_n z^r_n, \tag{7.18}$$

Note that messages (7.12)-(7.14) are matrix and vector valued quantities and require a fixed amount memory for storage regardless of the number of nodes in the network. Also, the same rule for generating and combining messages are implemented at each node.

For collaborative smoothing, once a batch of T observations have been obtained, each node r aims to implement

$$p^r_\theta(x^r_n | Y_{1:T}) \propto \int p^r_\theta(x^r_{1:T}, Y_{1:T}) dx^r_{1:T\setminus\{n\}}$$

where $dx^r_{1:T\setminus\{n\}}$ means integration w.r.to all variables except x^r_n. The standard Kalman smoother is implemented with a forward pass (7.15)-(7.18) first to compute the filtering densities, and then followed by a backward pass which is summarized by the following equations [139]:

$$J^r_{n-1} = \Sigma^r_{n-1} A_n^{\mathrm{T}} (\Sigma^r_{n|n-1})^{-1} \tag{7.19}$$

$$\mu^r_{n-1|T} = \mu^r_{n-1} + J^r_{n-1} (\mu^r_{n|T} - A_n \mu^r_{n-1}) \tag{7.20}$$

$$\Sigma^r_{n-1|T} = \Sigma^r_{n-1} + J^r_{n-1} (\Sigma^r_{n|T} - \Sigma^r_{n|n-1}) \left(J^r_{n-1}\right)^{\mathrm{T}} \tag{7.21}$$

The backward pass is performed commencing with $n = T$ until $n = 2$ and the smoothed density is $p^r_\theta(x^r_n | Y_{1:T}) = \mathcal{N}(\mu^r_{n|T}, \Sigma^r_{n|T})$. It is an entirely local procedure with no exchange of information between neighboring nodes.

7.4 Distributed Collaborative Localisation

Our sensor localisation problem is a static parameter estimation problem as discussed in Section 7.2. To solve the localisation problem, we propose to use likelihood inference techniques, namely Recursive Maximum Likelihood and Expectation-Maximisation, which have been presented in Section 3.5. The RML and EM algorithm described in Section 3.5 centralized and here we will derive distributed RML and EM counterparts.

7.4.1 Distributed RML

Every edge is assigned to one node and all edge-controlling nodes will implement a RML algorithm to learn the θ-parameter for its edge. For example in the three node network of Figure 7.2, edge (1,2) could be assigned to node 2 and edge (1,3) to node 3. Let $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true parameter θ_* given the available data $Y_{1:n-1}$. At a given node r that controls edge (r,j) the following RML algorithm is implemented,

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r \left[\nabla_{\theta^{r,j}} \log \int p_\theta^r(Y_n|x_n^r) p_\theta^r(x_n^r|Y_{1:n-1}) dx_n^r\right]_{\theta=\theta_n}$$

where γ_{n+1}^r is the step-size and should satisfy the same conditions that were specified for (**??**). In practice a constant step-size sequence is selected to ensure continual adaptation. The gradient is computed w.r.t. $\theta^{r,j}$, the local collaborative *predicted* density $p_\theta^r(x_n^r|Y_{1:n-1})$ at node r was defined in (7.8) and is a function of $\theta = \{\theta^{i,j}\}_{(i,j)\in\mathcal{E}}$, and likelihood term is given in (7.9). Node r updates $\theta_n^{r,j}$ in the direction of ascent of $p_\theta^r(Y_n|Y_{1:n-1})$, which is its *local* conditional likelihood of $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ given all the measurements received in the network from time 1 to $n-1$. Also, the gradient is evaluated at $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ while only $\theta_n^{r,j}$ is available locally at node r. The remaining values θ_n are stored across the network. All nodes of the network that control an edge parameter will implement such a local gradient algorithm.

The distributed RML algorithm is given as follows.

Algorithm 7.2 ***Distributed RML***

At time n, let the current parameter estimate be θ_n. Upon obtaining measurements $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ the following filtering and parameter update steps are to be performed.

Filtering step*: Perform all steps in Algorithm 7.1.*

Parameter update*: Each node $r \in \mathcal{V}$ of the network will update the following quantities for every edge (r,j) controlled by it:*

$$\nabla_{r,j}\mu_{n|n-1}^r = A_n \nabla_{r,j}\mu_{n-1}^r, \tag{7.22}$$

$$\nabla_{r,j} z_n^r = (\Sigma_{n|n-1}^r)^{-1} \nabla_{r,j}\mu_{n|n-1}^r - m_n^{j,r}, \tag{7.23}$$

$$\nabla_{r,j}\mu_n^r = (M_n^r)^{-1} \nabla_{r,j} z_n^r. \tag{7.24}$$

Upon doing so the localization parameter is updated:

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r[-(\nabla_{r,j}\mu_{n|n-1}^r)^T(\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r + (\nabla_{r,j}z_n^r)^T(M_n^r)^{-1}z_n^r + \dot{m}_n^{j,r} - \ddot{m}_n^{j,r}].$$

The derivation of the algorithm is presented in the Appendix. The intermediate quantities (7.22)-(7.24) take values in $\mathbb{R}^{4\times 4}$ and may be initialized to zero matrices. The step-size sequences should satisfy $\sum_n \gamma_n^r = \infty$ and $\sum_n {\gamma_n^r}^2 < \infty$, but in practice a constant step-size can be used instead (see Section 7.5).

7.4.2 Distributed EM

The basic idea behind a distributed implementation of the EM is as follows. Let $\theta_k = \{\theta_k^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the current estimate of θ_* after $k-1$ distributed EM iterations on the batch of observations $Y_{1:T}$.[1] Each edge controlling node r will execute the following E and M steps to update the estimate of the localisation parameter for its edge:

$$Q^r(\theta_k, \theta) = \int \log p_\theta^r(x_{1:T}^r, Y_{1:T}) p_{\theta_k}^r(x_{1:T}^r|Y_{1:T}) dx_{1:T}^r,$$
$$\theta_{k+1}^{r,j} = \arg\max_{\theta^{r,j}} \quad Q^r(\theta_k, (\theta^{r,j}, \theta_k^{-(r,j)})),$$

where $\theta_k^{-(r,j)} = \{\theta_k^e\}_{e\in\mathcal{E}\setminus(r,j)}$. These steps can be performed simultaneously by all edge controlling nodes r using an exchange of messages as detailed in Algorithm 7.1. Let

$$l^r(\theta_k) = \log p_{\theta_k}^r(Y_{1:T})$$

denote the log-likelihood of node r. From the general discussion of the EM algorithm in the start of Section 7.4 it is evident that

$$l^r(\theta_{k+1}^{r,j}, \theta_k^{-(r,j)}) \geq l^r(\theta_k).$$

In fact, we show in the simulations in Section 7.5 that the sequence of iterates θ_k converges to θ_*.

[1] Each iteration of the EM involves all the observations from $n = 1$ to T. We use a different subscript k to denote the k-th EM step.

To show how the E-step can be computed we write $p^r_\theta(x^r_{1:T}, Y_{1:T})$ as,

$$p^r_\theta(x^r_{1:T})p^r_\theta(Y_{1:T}|x^r_{1:T}) = \prod_{n=1}^{T} f^r_n(x^r_n|x^r_{n-1})p^r_\theta(Y_n|x^r_n),$$

where $p^r_\theta(Y_n|x^r_n)$ was defined in (7.9). Note that $p^r_{\theta_k}(x^r_{1:T}|Y_{1:T})$ is a function of $\theta_k = \{\theta^{i,i'}_k\}_{(i,i')\in\mathcal{E}}$ (and not just $\theta^{r,j}_k$) and the θ-dependance of $p^r_\theta(x^r_{1:T}, Y_{1:T})$ arises through the likelihood term only as $p^r_\theta(x^r_{1:T})$ is θ-independent. This means that it is sufficient to maintain the smoothed marginals $p^r_{\theta_k}(x^r_n|Y_{1:T})$, $1 \le n \le T$, to compute the E-step. The M-step is solved by setting the derivative of $Q^r(\theta_k, (\theta^{r,j}, \theta^{-(r,j)}_k))$ w.r.t. $\theta^{r,j}$ to zero. The derivation of the EM is presented in the Appendix and the main result is,

$$\nabla_{\theta^{r,j}} \int \log p^r_\theta(Y_n|x^r_n)p^r_{\theta_k}(x^r_n|Y_{1:T}) = \dot{m}^{j,r}_n - \ddot{m}^{j,r}_n - (m^{j,r}_n)^{\mathrm{T}}\mu^r_{n|T}$$

where $(m^{j,r}_n, \dot{m}^{j,r}_n, \ddot{m}^{j,r}_n)$, defined in (7.12)-(7.14), are propagated with localisation parameter θ_k for all observations from time 1 to T and $\mu^r_{n|T}$ is the mean of x^r_n under $p^r_{\theta_k}(x^r_n|Y_{1:T})$ as given by (7.19)-(7.21). Only $\ddot{m}^{j,r}_n$ is a function of $\theta^{r,j}$. To perform the M-step, the following equation is solved for $\theta^{r,j}$

$$(\sum_{n=1}^{T} m^{j,r}_n)\theta^{r,j} = \sum_{n=1}^{T}(\dot{m}^{j,r}_n - (m^{j,r}_n)^{\mathrm{T}}\mu^r_{n|T} - \sum_{p\in\mathrm{ne}(j)\setminus\{r\}} \ddot{m}^{p,j}_n) \tag{7.25}$$

Note that $\theta^{r,j}$ is a function of quantities available locally to node r and j only.

A summary of the distributed EM is as follows:

Algorithm 7.3 ***Distributed EM***

Consider a fixed batch of measurements $Y_{1:T} = \{Y^v_{1:T}\}_{v\in\mathcal{V}}$. At the k-th EM iteration ($k \ge 0$ with θ_0 arbitrary, the zero vector say) execute the following E and M steps.

***E-step**: For each node r, compute the smoothed marginals $\{p^r_{\theta_k}(x^r_n|Y_{1:T})\}_{n=1:T}$ by executing Algorithm 7.1 for time $n = 1$ to T with fixed localization parameter θ_k. Then perform the backward pass by executing (7.19)-(7.21) for $n = T$ until $n = 2$.*

***M-step**: For each node $r \in \mathcal{V}$, for all edges (r, j) controlled by node r, set $\theta^{r,j}_{k+1}$ to be the solution to (7.25).*

7.5 Numerical Examples

The performance of the distributed RML and EM algorithms are studied first for the Linear Gaussian case in a numerical example. The model for the hidden target is given in (7.1) with $V_n^r = B\widetilde{V}_n^r$, where $\widetilde{V}_n^r$ is zero mean Gaussian additive noise with variance $\widetilde{Q}_n$, and

$$A_n = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{\tau^2}{2} & 0 \\ \tau & 0 \\ 0 & \frac{\tau^2}{2} \\ 0 & \tau \end{bmatrix}, \quad \widetilde{Q}_n = 3I,$$

and I is the identity matrix. This choice for $A_n, B, \widetilde{Q}_n$ is common in the target tracking literature [13]. The observation model is given by (7.2) with

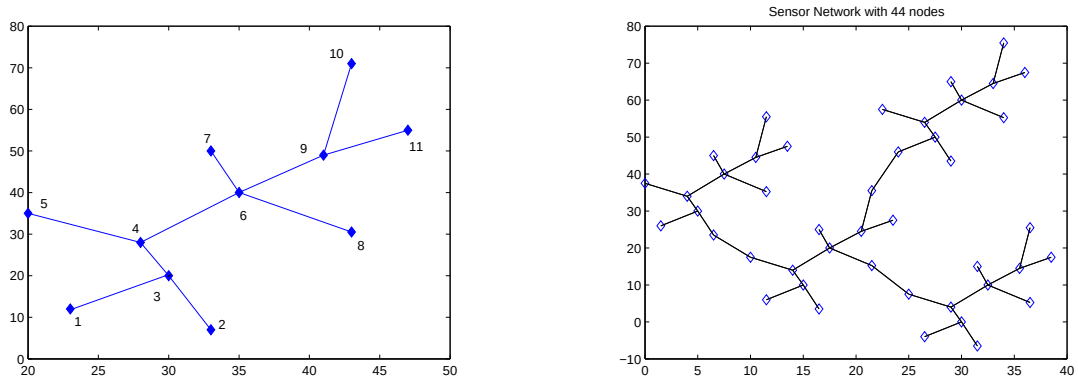
$$C_n^r = \alpha_n^r \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad R_n = \beta_n^r I,$$

where α_n^r, β_n^r are known time varying scaling parameters sampled at each time n from the uniform distribution defined over $[1, 3]$.

Two networks of different sizes and configurations were considered as shown in Figures 7.3(a) and 7.3(b). The settings for the remaining parameters are as follows: $\tau = 0.1$, a constant step size $\gamma_k = 10^{-3}$ for the RML, $T = 1000$ for the EM, $\theta_0^{r,j} = 0$ for all $(r, j) \in \mathcal{E}$ for the RML and EM. In Figures 7.4(a) and 7.5(a) we plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ against iteration n for RML and for the EM in Figures 7.4(b) and 7.5(b). All errors converge to zero.

Figure 7.6 plots $l^r(\theta_k) = \log p_{\theta_k}^r(Y_{1:T})$ with respect to k for nodes $r \in \{3, 4, 6, 9\}$, where θ_k is obtained using distributed EM for the sensor network of Figure 7.3(a). We observe that overall each iteration of the distributed EM is increasing the log-likelihood of all nodes.

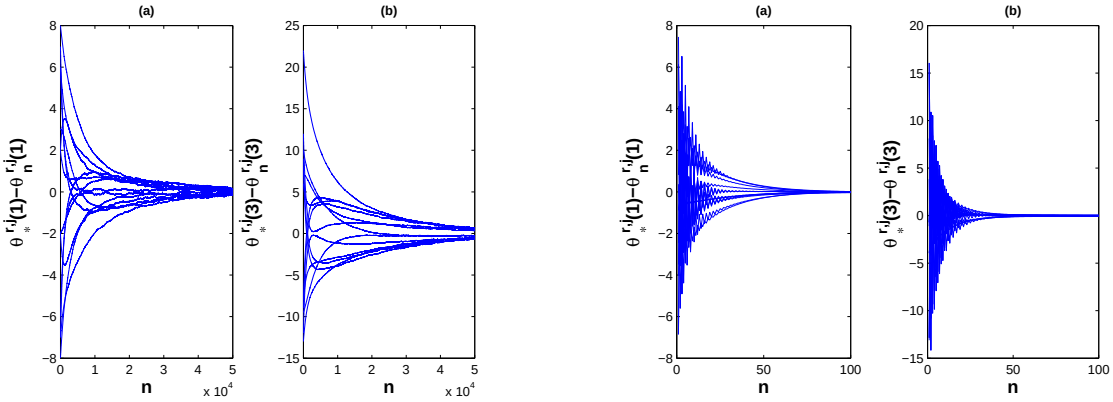
We observe that overall as θ_k approaches its true value θ^* each l^r converges to $\log p(Y_{1:T})$. In Figure 7.7 we plot contour plots of $l^r(\theta)$ for $r \in \{3, 4, 6, 9\}$ with respect to $\theta^{3,1}$, when the other $\theta^{i,j}$s are set to zero vectors. Again as in Figure 7.6 one can see that when $\theta = \theta_*$ then $\arg\max_\theta l^i(\theta) = \arg\max_\theta l^j(\theta)$, but this does not hold in any other case. Fortunately, in Figure 7.7 we observe that $\arg\max_{\theta^{3,1}} l^3(\theta)$ remains the same regardless the value the other localisation parameters have.



(a) 11 node sensor network

(b) 44 node sensor network

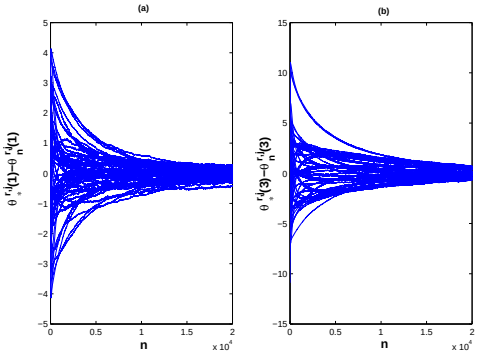
Figure 7.3: *Sensor networks of different sizes used for target tracking; in each case the localisation parameters θ_* are estimated.*



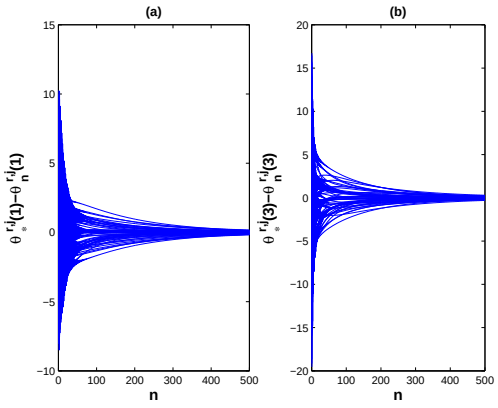
(a) Parameter error after each RML iteration for each edge of the sensor network. (a) and (b) show the errors in the x- and y- coordinates respectively.

(b) Parameter error after each EM iteration for each edge of the sensor network. (a) and (b) show the errors in the x- and y- coordinates respectively.

Figure 7.4: *Simulation results for the 11 node sensor network of Figure 7.3(a). The convergence of the localization parameter estimate to θ_* is demonstrated using appropriate error plots for the distributed RML and.*



(a) Parameter error after each RML iteration for each edge of the sensor network. (a) and (b) show the errors in the x- and y- coordinates respectively.

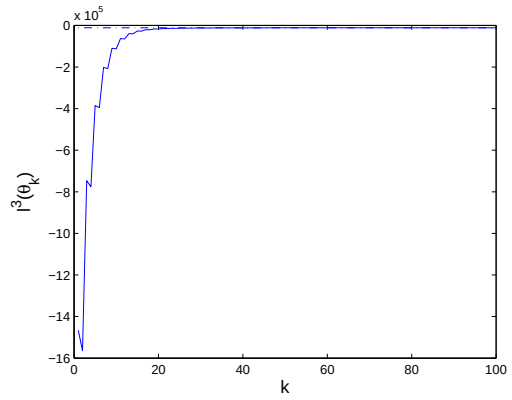


(b) Parameter error after each EM iteration for each edge of the sensor network. (a) and (b) show the errors in the x- and y- coordinates respectively.

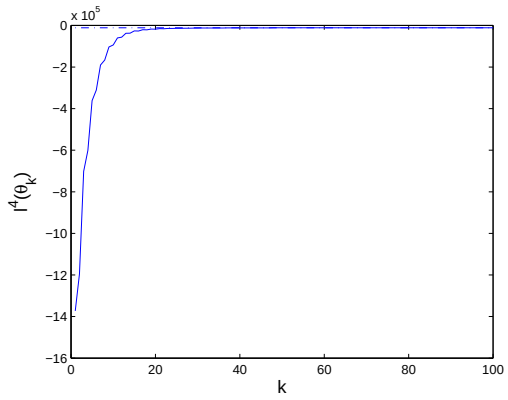
Figure 7.5: *Simulation results for the 44 node sensor network of Figure 7.3(a). The convergence of localization parameter estimate to θ_* is demonstrated using appropriate error plots for the distributed RML and EM.*

7.6 Conclusions

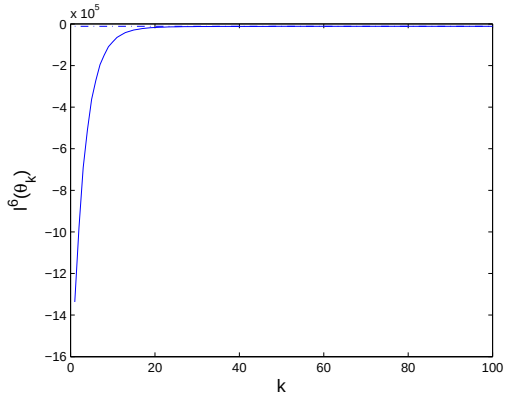
In this chapter, we have presented a general framework to perform recursive static parameter estimation in dynamic Graphical models. We derive fully decentralized algorithms using RML and EM to solve the sensor localisation problem. For linear Gaussian graphs, our algorithm can be implemented exactly using a distributed version of the Kalman filter and its derivative. In the non linear case, a solution based on linearisation and the Extended Kalman Filter can be proposed. A Sequential Monte Carlo algorithm to solve the general nonlinear and non Gaussian problem is presented in the next chapter.



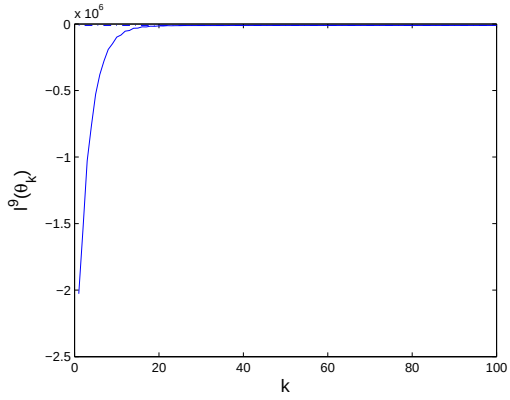
(a) $l^3(\theta_k)$ with respect to k.



(b) $l^4(\theta_k)$ with respect to k.

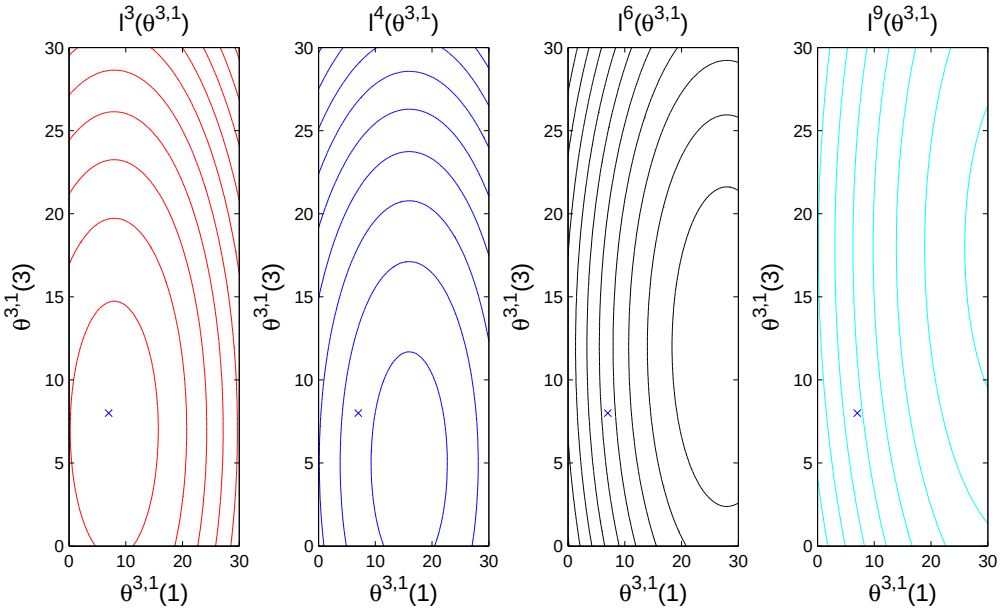


(c) $l^6(\theta_k)$ with respect to k.



(d) $l^9(\theta_k)$ with respect to k.

Figure 7.6: *Likelihood plots of $l^r(\theta_k)$ with respect to k, for the sensor network of Figure 7.3(b) and nodes $r = 3, 4, 6, 9$. For each sub-figure, the dotted line indicates $\log p^r_{\theta_*}(Y_{1:T})$. Note how (generally) each iteration is increasing the likelihood of all nodes.*



.

Figure 7.7: *Contour plots of $l^r(\theta^{3,1})$ for $r = 3, 4, 6, 9$. The rest of the localisation parameters $\theta^{i,j}$, with $i \neq 3$ and $j \neq 1$, are set to initial values $[0, 0, 0, 0]^T$. In each subfigure, the cross is at $\theta_*^{3,1}$*

7.A Distributed RML derivation

For the linear Gaussian case, we have

$$\begin{aligned}
&\log p_\theta^r(Y_n|Y_{0:n-1}) \\
&= -\frac{1}{2}\sum_{i\in\mathcal{V}}(Y_n^i - C_n^i\theta^{r,i})^T {R_n^i}^{-1}(Y_n^i - C_n^i\theta^{r,i}) \\
&- \frac{1}{2}{\mu_{n|n-1}^r}^{\mathrm{T}}(\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r \\
&+ \frac{1}{2}(z_n^r)^{\mathrm{T}}(M_n^r)^{-1}z_n^r + const
\end{aligned}$$

where all θ-independent terms have been lumped together in the term 'const'. Differentiating this expression w.r.t. $\theta^{r,j}$ yields

$$\begin{aligned}
&\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n|Y_{0:n-1}) \\
&= -(\nabla_{\theta^{r,j}}\mu_{n|n-1}^r)^{\mathrm{T}}(\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r \\
&+ (\nabla_{\theta^{r,j}} z_n^r)^{\mathrm{T}}(M_n^r)^{-1}z_n^r \\
&+ \sum_{i\in\mathcal{V}}(\nabla_{\theta^{r,j}}\theta^{r,i})^{\mathrm{T}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}(Y_n^i - C_n^i\theta^{r,i}).
\end{aligned}$$

Using the recursion of (7.15)-(7.18) we can to propagate terms $\nabla_{\theta^{r,j}}\mu^r_{n|n-1}$, $\nabla_{\theta^{r,j}}z^r_n$ and $\nabla_{\theta^{r,j}}\mu^v_n$ locally at each node r as follows,

$$\nabla_{\theta^{r,j}}\mu^r_{n|n-1} = A_n\nabla_{\theta^{r,j}}\mu^r_{n-1}, \tag{7.26}$$

$$\nabla_{\theta^{r,j}}z^r_n = (\Sigma^r_{n|n-1})^{-1}\nabla_{\theta^{r,j}}\mu^r_{n|n-1} - \sum_{i\in\mathcal{V}}(C^i_n)^{\mathrm{T}}(R^i_n)^{-1}C^i_n\nabla_{\theta^{r,j}}\theta^{r,i}, \tag{7.27}$$

$$\nabla_{\theta^{r,j}}\mu^r_n = (M^r_n)^{-1}\nabla_{\theta^{r,j}}z^r_n. \tag{7.28}$$

Using property (8.3) we note that for the set of vertices i for which the path from r to i includes edge (r,j), $\nabla_{\theta^{r,j}}\theta^{r,i} = I$ (the identity matrix) whereas for the rest $\nabla_{\theta^{r,j}}\theta^{r,i} = 0$. For all the nodes i for which $\nabla_{\theta^{r,j}}\theta^{r,i} = I$, let them form a sub tree $(\mathcal{V}'_{rj}, \mathcal{E}'_{rj})$ branching out from node j away from node r. Then the last sum in the expression for $\nabla_{\theta^{r,j}}\log p^r_\theta(Y_n|Y_{0:n-1})$ evaluates to,

$$\sum_{i\in\mathcal{V}'_{rj}}(C^i_n)^{\mathrm{T}}(R^i_n)^{-1}(Y^i_n - C^i_n\theta^{r,i}) = \dot{m}^{j,r}_n - \ddot{m}^{j,r}_n.$$

Similarly, we can write the sum in the expression for $\nabla_{\theta^{r,j}}z^r_n$ as $m^{j,r}_n$ to obtain

$$\nabla_{\theta^{r,j}}z^r_n = (\Sigma^r_{n|n-1})^{-1}\nabla_{\theta^{r,j}}\mu^r_{n|n-1} - m^{j,r}_n. \tag{7.29}$$

7.B Distributed EM derivation

For the EM approach, once a batch of T observations have been obtained, each node r of the network that controls an edge will execute the following E and M step iteration n,

$$Q^r(\theta_k,\theta) = \int \log p^r_\theta(x^r_{1:T}, Y_{1:T})p^r_{\theta_k}(x^r_{1:T}|Y_{1:T})dx^r_{1:T},$$

$$\theta^{r,j}_{k+1} = \arg\max_{\theta^{r,j}\in\Theta} Q^r(\theta_k, (\theta^{r,j}, \{\theta^e, e\in\mathcal{E}\backslash(r,j)\})),$$

where it is assumed that node r controls edge (r,j). The quantity $p^r_{\theta_k}(x^r_{1:T}|Y_{1:T})$ is the joint distribution of the hidden states at node r given all the observations of the network from time 1 to T and is given up to a proportionality constant,

$$p^r_{\theta_k}(x^r_{1:T})p^r_{\theta_k}(Y_{1:T}|x^r_{1:T}) = \prod_{n=1}^{T} f^r_n(x^r_n|x^r_{n-1})p^r_{\theta_k}(Y_n|x^r_n),$$

where $p^r_{\theta_k}(Y_n|x^r_n)$ was defined in (7.9). Note that $p^r_{\theta_k}(x^r_{1:T}, Y_{1:T})$ (and hence $p^r_{\theta_k}(x^r_{1:T}|Y_{1:T})$) is a function of $\theta_k = \{\theta^{i,i'}_k\}_{(i,i')\in\mathcal{E}}$ and not just $\theta^{r,j}_k$. Also, the θ-dependance of $p^r_\theta(x^r_{1:T}, Y_{1:T})$ arises

through the likelihood term only as $p^r_\theta(x^r_{1:T})$ is θ-independent. Note that

$$\begin{aligned}
&\sum_{v\in\mathcal{V}} \log g^v_n(Y^v_n|x^r_n+\theta^{r,v}) \\
&= \sum_{v\in\mathcal{V}} c^v_n - \frac{1}{2}\sum_{v\in\mathcal{V}}(Y^v_n - C^v_n\theta^{r,v})^{\mathrm{T}}(R^v_n)^{-1}(Y^v_n - C^v_n\theta^{r,v}) \\
&+ (x^r_n)^{\mathrm{T}}\sum_{v\in\mathcal{V}}(C^v_n)^{\mathrm{T}}(R^v_n)^{-1}(Y^v_n - C^v_n\theta^{r,v}) \\
&- \frac{1}{2}(x^r_n)^{\mathrm{T}}\left[\sum_{v\in\mathcal{V}}(C^v_n)^{\mathrm{T}}(R^v_n)^{-1}C^v_n\right]x^r_n
\end{aligned}$$

where c^v_n is a constant independent of θ. Using the fact that $x^{\mathrm{T}}Ax = trace(Axx^{\mathrm{T}})$ and $E(x^{\mathrm{T}}Ax) = trace(AE(xx^{\mathrm{T}}))$, taking the expectation w.r.t. $p^r_{\theta_n}(x^r_n|Y_{1:T})$ gives

$$\begin{aligned}
&\int \log p^r_\theta(Y_n|x^r_n)p^r_{\theta_k}(x^r_n|Y_{1:T}) \\
&= -\frac{1}{2}\sum_{v\in\mathcal{V}}\left[(Y^v_n - C^v_n\theta^{r,v})^{\mathrm{T}}(R^v_n)^{-1}(Y^v_n - C^v_n\theta^{r,v})\right] \\
&- (\mu^r_{n|T})^{\mathrm{T}}\sum_{v\in\mathcal{V}}(C^v_n)^{\mathrm{T}}(R^v_n)^{-1}C^v_n\theta^{r,v} + \text{const}
\end{aligned}$$

where all terms independent of $\theta^{r,j}$ have been lumped together as 'const' and $\mu^r_{n|T}$ is the mean of x^r_n under $p^r_{\theta_k}(x^r_n|Y_{1:T})$. Taking the gradient w.r.t. $\theta^{r,j}$ we get and following the steps in the derivation of the distributed RML we obtain

$$\nabla_{\theta^{r,j}}\int \log p^r_\theta(Y_n|x^r_n)p^r_{\theta_k}(x^r_n|Y_{1:T}) = \dot{m}^{j,r}_n - \ddot{m}^{j,r}_n - (m^{j,r}_n)^{\mathrm{T}}\mu^r_{n|T}$$

where $(m^{j,r}_n, \dot{m}^{j,r}_n, \ddot{m}^{j,r}_n)$ is defined in (7.12)-(7.14). Only $\ddot{m}^{j,r}_n$ is a function of $\theta^{r,j}$. Now to perform the M-step, we solve

$$\left(\sum_{n=1}^{T} m^{j,r}_n\right)\theta^{r,j} = \sum_{n=1}^{T}\left(\dot{m}^{j,r}_n - (m^{j,r}_n)^{\mathrm{T}}\mu^r_{n|T} - \sum_{p\in\mathrm{ne}(j)\setminus\{r\}}\ddot{m}^{p,j}_n\right)$$

and $\theta^{r,j}$ can recovered by standard linear algebra. Note that $\theta^{r,j}$ is solved by quantities available locally to node r and j only.

8

Distributed Localisation and Tracking for Nonlinear Non-Gaussian Sensor Networks

Summary. *Sequential Monte Carlo methods have been used successfully for non-gaussian and non-linear filtering and inference problems. When used within the Graphical models context, they can be combined with Nonparametric Belief Propagation at each node of a graph. This has already been developed for problems involving dynamic filtering for distributed environments. We aim to extend this methodology for the static parameter inference problem using a distributed implementation of Recursive Maximum Likelihood (RML). The resulting algorithm can be thought as an extension of Nonparametric Belief Propagation to compute likelihood gradients. This algorithm is applied to solve the sensor localisation problem for sensor networks.*

8.1 Introduction

In the previous chapter we proposed to use distributed likelihood inference techniques, namely Recursive Maximum Likelihood (RML) and Expectation-Maximization (EM), to solve the self localisation problem when collaborative filtering is used. We focused on the linear gaussian case and proposed an extended Kalman filter approach for the nonlinear case. The main motive was computational efficiency in order to be able to apply our methodology for large scale sensor networks. In this chapter we shall concentrate on how to develop fully distributed Sequential Monte Carlo (SMC) methods using RML for static parameter estimation problems in dynamic Graphical Models. We will mainly focus on solving the self localisation problem for sensor networks considered earlier in Chapters 6 and 7. Note that we are dealing with static parameters because we assume that sensors' positions remain fixed.

Most tracking problems are essentially non-linear non-Gaussian filtering problems and Sequential Monte Carlo (SMC) methods, also known as Particle Filters, provide very good approximations to the filtering densities under weak assumptions. Posterior inference for static parameters using SMC is a widely studied problem. A common approach is to include the unknown parameter and cast the problem as a filtering one. This has appeared in the Sequential Monte Carlo literature in [56, 105, 154], but has proved to be inefficient due to a degeneracy problem inherent in the standard SMC algorithm [10]. It is well known that posterior inference for static parameters with SMC suffers from the *degeneracy* problem. As more and more observations are made over time, the initial diversity of samples for the unknown static parameters will be lost as result of the resampling step in a particle filter. Several methods have been proposed in the literature to overcome this limitation. They include introducing artificial dynamics for the static parameter or Markov Chain Monte Carlo (MCMC) moves to re-introduce diversity lost during resampling. The first method will alter the problem and yields imprecise estimates of the static parameters. Furthermore, there is ambiguity on how much "dynamics"should be introduced. Incorporating MCMC steps does not alter the problem being solved but additional complications arise during implementation. The MCMC steps rely on sufficient statistics that are based on a particle approximation to the path posterior density $p_\theta\left(x_{0:n}|Y_{0:n}\right)$. In Chapter 3 of [136] it was demonstrated that this density cannot be properly approximated using SMC methods, for a fixed number of particles, and the sufficient statistics degrade over time due to error accumulation [10], which will result in the static parameter estimates to diverge.

To circumvent all these problems, some gradient approaches that use Recursive Maximum Likelihood (RML) have been proposed, [98]. They approximate the gradient of the optimal filter using standard path-based particle filtering methods [32, 51, 72], but suffer from similar limitations. Recently in [136], particle methods have been developed to approximate the first derivative of the optimal filter with respect to the unknown parameters of the dynamic model, in order to solve the problem with a RML approach. We aim to extend this methodology for Graphical Models, in order to benefit from the merits of SMC approximations and be able to derive parameter estimation algorithms for distributed environments.

8.1.1 Inference in Graphical Models

Collaborative filtering for Graphical models is a well studied area, [82], [166], [59], [159]. However, a large proportion of the literature is dedicated to a finite valued hidden state and observation process. Recently, driven by applications in Computer Vision, several works have been dedicated to inference in Graphical models for continuous hidden state and observation models [29], [81], [155]. All these works are SMC versions of the Belief Propagation algorithm but differ in implementation. The Particle Message Passing (PAMPAS) algorithm of [81] is a SMC algorithm for graphs where each node has a small number of neighbours. The Nonparametric Belief Propagation (NBP) algorithm of [155] can handle more dense graphs by incorporating MCMC steps. In [29] the author proposes a purely Importance Sampling based alternative to NBP. NBP is perhaps the most general of these and is more suited to our sensor network localisation and tracking problem. In this section, we extend the NBP algorithm to propagate not just the filters $(\pi_{n|n-1}, \pi_n)$ but their derivatives as well. Doing so allows us to implement RML for estimating the localisation parameters. The development of NBP for propagating the derivatives $(\pi_{n|n-1}, \pi_n)$, denoted as $(\mathring{\pi}_{n|n-1}, \mathring{\pi}_n)$, is based on the work [136], which studies a centralised implementation of filter derivatives only.

The organisation of this chapter shall be as follows. In Section 8.2 we formulate the collaborative filtering problem for sensor networks as a distributed Hidden Markov Model and adopt a Recursive Maximum Likelihood approach for the localisation problem. In Section 8.3 we show how the algorithms of Chapter 7 can be extended for nonlinear models by appropriate linearisations. In Section 8.4, we analyse how message passing can be used around the network to perform distributed filtering and localisation and also show how we can approximate

the messages so that they can be used for a particle implementation. In Section 8.5 we present the particle approximations for the filters, their derivatives and the likelihood gradients. In Section 8.6 we present the particle algorithm to solve the problem. Finally, in Section 8.7 we use this algorithm to solve the problem for a particular numerical example.

8.2 Problem Formulation

In this section we shall formulate the distributed filtering problem suited to the collaborative filtering and sensor self localisation problem. As this has been done in much detail in the previous two chapters we shall only define the variables and parameters that are necessary for this chapter. We understand that this might involve some repetition of the material presented in Chapters 6 and 7, but this was done so that the material in this chapter is self contained. In any case, we shall assume that the reader is familiar with the problem, and keep the repetition of previously made definitions and comments to a minimum.

8.2.1 Parameter Estimation for Hidden Markov Models using Recursive Maximum Likelihood

A HMM suited for parameter estimation is defined in Section 3.3. RML is an online algorithm that refines the parameter estimates as new data arrives using a stochastic gradient ascent approach. A detailed description is found in Section 3.5.3. In summary, a recursion is maintained for computing the quantity $[\nabla \log p(Y_n|Y_{1:n-1})]|_{\theta=\theta_n}$ needed by the gradient ascent algorithm and this recursion needs to propagate $p_{\theta_n}(x_n|Y_{1:n-1})$ and $p_{\theta_n}(x_n|Y_{1:n})$ as well as the gradient of these filters with respect to the parameter θ_n.In the spirit of [98, 136] we provide the following definitions to assist the exposition of this chapter:

$$\begin{aligned}
\pi_{n|n-1}(x_n) &= p_{\theta_n}(x_n|Y_{1:n-1}) \\
&= \int f(x_n|x_{n-1})\pi_{n-1}(x_{n-1})dx_{n-1}, \\
\mathring{\pi}_{n|n-1}(x_n) &\equiv \nabla_{\theta_n} p_\theta(x_n|Y_{1:n-1}) \\
&= \int \nabla_{\theta_n} f(x_n|x_{n-1})\pi_{n-1}(x_{n-1})dx_{n-1} + \int f(x_n|x_{n-1})\mathring{\pi}_{n-1}(x_{n-1})dx_{n-1},
\end{aligned}$$

$$\begin{aligned}\pi_n(x_n) &= p_{\theta_n}(x_n|Y_{1:n}) \\ &= \frac{g_{\theta_n}(Y_n|x_n)\pi_{n|n-1}(x_n)}{\int g_{\theta_n}(Y_n|x_n)\pi_{n|n-1}(x_n)dx_n}, \\ \mathring{\pi}_n(x_n) &\equiv \nabla_{\theta_n} p_\theta(x_n|Y_{1:n}) \\ &= \left(\frac{\nabla g_{\theta_n}(Y_n|x_n)}{g_{\theta_n}(Y_n|x_n)} + \frac{\mathring{\pi}_{n|n-1}(x_n)}{\pi_{n|n-1}(x_n)} - \int \left[\frac{\nabla g_{\theta_n}(Y_n|x_n)}{g_{\theta_n}(Y_n|x_n)} + \frac{\mathring{\pi}_{n|n-1}(x_n)}{\pi_{n|n-1}(x_n)}\right] \pi_n(x_n)dx_n\right)\pi_n(x_n).\end{aligned}$$

The recursion for $(\mathring{\pi}_{n|n-1}, \mathring{\pi}_n)$ can be derived using the standard product rule for differentiation and interchanging derivatives and integrals where required. The actual form of the steepest ascent used in [136] is

$$\theta_{n+1} = \theta_n + \frac{\gamma_{n+1}}{\int g_{\theta_n}(Y_n|x_n)\pi_{n|n-1}(x_n)dx_n}\left[\int \left(\nabla g_{\theta_n}(Y_n|x_n)\pi_{n|n-1}(x_n) + g_{\theta_n}(Y_n|x_n)\mathring{\pi}_{n|n-1}(x_n)\right)dx_n\right],$$

8.2.2 Collaborative Filtering of Distributed Hidden Markov Models defined for Sensor networks

In Section 3.7.5 and 3.7.6 we provide a generalisation of HMM and RML for distributed problems using Graphical Models. In this chapter, we will use a slightly different choice of potential functions and follow the problem formulation of Chapter 7. In summary, we consider the sensor network $(\mathcal{V}, \mathcal{E})$ which is an undirected Graphical model with a tree topology where $\mathcal{V}$ are the set of nodes of the network and $\mathcal{E}$ are the set of edges. Nodes $i, j \in \mathcal{V}$ are connected provided the edge $(i, j) \in \mathcal{E}$ exists. All nodes are time synchronised distributed trackers and observe a single physical target at discrete time intervals $n \geq 0$.

The non-linear non-Gaussian setting can be expressed with the following distributed Hidden Markov Model (HMM),

$$X_n^r|X_{n-1}^r = x_{n-1}^r \sim f_n(.|x_{n-1}^r), \tag{8.1}$$

$$Y_n^r|X_n^r = x_n^r \sim g_n^r(.|x_n^r). \tag{8.2}$$

where $X_n^r \in \mathbb{R}^{d_x}$ is the hidden state and $Y_n^r \in \mathbb{R}^{d_y^r}$ is the measurement made by node r at time n. A common state-space $\mathbb{R}^{d_x}$ and transition model $f_n(.|x_{n-1}^r)$ is adopted for all nodes $r \in \mathcal{V}$ since they track the same physical target. The target's transition model f_n and is assumed time varying for generality.

The vector X_n^r at node r is defined relative to the local coordinate system of node r which regards itself as the origin. Similarly for nodes the rest of the nodes. In a standard target

tracking setup the state-space is $\mathbb{R}^4$,

$$X_n^r = [X_n^r(1), X_n^r(2), X_n^r(3), X_n^r(4)]^{\mathrm{T}} \in \mathbb{R}^4,$$

where $X_n^r(1)$ and $X_n^r(3)$ is the target's x and y position while $X_n^r(2)$ and $X_n^r(4)$ is the velocity in the x and y direction. The measurement Y_n^r made by node r is also defined relative to the local coordinate system at node r. The observation model $g_n^r(.|x_n^r)$ is time varying and is different for each node. Also, the length of the observation vector Y_n^r need not be the same for different nodes, since each node may be equipped with a different sensor type. For example, node r may obtain measurements of the target's position while node v measures bearing.

We define $\theta_*^{i,j}$ to be the position of node i in the local coordinate system of node j. This means that the vector X_n^i relates to the local coordinate system of node j as follows

$$X_n^j = X_n^i + \theta_*^{i,j}.$$

By default, $\theta_*^{v,v} = 0$ for all $v \in \mathcal{V}$. We assume the nodes are not mobile. Therefore the *localisation* parameters $\theta_* := [\theta_*^{i,j}]_{(i,j)\in\mathcal{E}}$ are static. We note the following obvious relationship: if nodes i and j are connected through intermediate nodes $j_1, j_2, \ldots, j_m$ then

$$\theta_*^{i,j} = \theta_*^{i,j_1} + \theta_*^{j_1,j_2} + \theta_*^{j_2,j_3} + \ldots + \theta_*^{j_{m-1},j_m} + \theta_*^{j_m,j}. \tag{8.3}$$

As in the previous chapters we define $\theta_*^{i,j}$ so that the dimensions are the same as the target state vector. This means that only the first and third component of $\theta_*^{i,j}$ is relevant while for the velocity components of $\theta_*^{i,j}$ we have $\theta_*^{i,j}(2) = \theta_*^{i,j}(4) = 0$. In general, in the *collaborative* filtering problem, each node r propagates

$$\text{prediction step:} \qquad \pi_{n|n-1}^r(x_n^r) = \int f_n(x_n^r|x_{n-1}^r)\pi_{n-1}^r(x_{n-1}^r)dx_{n-1}^r, \tag{8.4}$$

$$\text{update step:} \qquad \pi_n^r(x_n^r) \propto p_n^r(Y_n|x_n^r)p_n^r(x_n^r|Y_{1:n-1}), \tag{8.5}$$

where $Y_n \equiv [Y_n^v]_{v\in\mathcal{V}}$, and

$$p_n^r(Y_n|x_n^r) = \prod_{v\in\mathcal{V}} g_n^v(Y_n^v|x_n^r + \theta_*^{r,v}).$$

The prediction step can be implemented locally at each node without exchange of information, but the update step is replaced with

$$\pi_n^r(x_n^r) \propto \pi_{n|n-1}^r(x_n^r) \prod_{v\in\mathcal{V}} g_n^v(Y_n^v|x_n^r + \theta_*^{r,v}). \tag{8.6}$$

The gradients of the distributed filters are denoted as $(\mathring{\pi}^r_{n|n-1}, \mathring{\pi}^r_n)$. The derivation of the recursion for $(\mathring{\pi}^r_{n|n-1}, \mathring{\pi}^r_n)$ resembles the one above for $(\mathring{\pi}_{n|n-1}, \mathring{\pi}_n)$ and therefore will be omitted.

We will derive a truly distributed implementation of the collaborative filtering problems, i.e. each node executes the filtering update using its own variables and parameters as well as incoming messages. The message passing algorithm should be scalable with the size of the network. However, collaborative filtering hinges on knowledge of the localisation parameters $[\theta^{i,j}_*]_{(i,j)\in\mathcal{E}}$ which are unknown *apriori*.

8.2.3 Distributed Recursive Maximum Likelihood Applied to Sensor Localisation

We consider first the RML procedure. Let $\theta_n = [\theta^{i,j}_n]_{(i,j)\in\mathcal{E}}$ be the estimate of the true parameter θ_* given the available data $Y_{1:n-1}$. At a given node r that controls edge (r, j) the following RML algorithm is implemented,

$$\theta^{r,j}_{n+1} = \theta^{r,j}_n + \gamma^r_{n+1} \left[\nabla_{\theta^{r,j}} \log \int \left(\prod_{v\in\mathcal{V}} g^v_n(Y^v_n | x^r_n + \theta^{r,v}) \right) \pi^r_{n|n-1}(x^r_n) dx^r_n \right] \Bigg|_{\theta=\theta_n},$$

where we have used property (8.3) to obtain $\{\theta^{r,v}\}_{v\in\mathcal{V}}$ from $\theta = \{\theta^{i,j}\}_{(i,j)\in\mathcal{E}}$. We remark that $p^r_\theta(Y_n|x^r_n)$ is not a time homogeneous likelihood and perhaps should have been explicit with an additional subscript n as follows $p^r_{\theta,n}(Y_n|x^r_n)$. We write $p^r_\theta(Y_n|x^r_n)$ to simplify the notation.

Furthermore, note that node r updates $\theta^{r,j}_n$ to ascend the log-likelihood of $[Y^v_n]_{v\in\mathcal{V}}$ given all the measurements received in the network from time 1 to $n-1$. Thus the RML procedure is truly collaborative. Also, the gradient is evaluated at θ_n, while only $\theta^{r,j}_n$ is available locally at node r. The remaining values θ_n are stored across the network. All nodes of the network that control an edge parameter will implement the RML recursion shown above.

8.2.4 Contribution of this Chapter

For a linear Gaussian state-space model, closed-form expressions exist for $(\pi^r_{n|n-1}, \pi^r_n)$ and $(\mathring{\pi}^r_{n|n-1}, \mathring{\pi}^r_n)$. We have already presented these in Chapter 7. In the next section, we shall show how to implement local linearisation steps, so that the algorithms in Chapter 7 can be used for nonlinear Gaussian models. The remainder of the chapter shows how a truly distributed implementation may be obtained for nonlinear non-Gaussian models. We shall propose this time a Sequential Monte Carlo estimation algorithm based on RML that allows simultaneous estimation of the localisation parameters and collaborative filtering. We will derive distributed

SMC approximations for the recursion for $(\mathring{\pi}^r_{n|n-1}, \mathring{\pi}^r_n)$ in the spirit of [136]. These distributed SMC approximations to $(\pi^r_n, \mathring{\pi}^r_n)$ build on Nonparametric Belief Propagation that has already been proposed for Graphical models in [155].

8.3 Linearisation for Distributed Filtering and RML

An Extended Kalman Filter (EKF) implementation for distributed RML can be derived a based on local linearisation. Let the distributed tracking system be given by the following model:

$$X^r_n = \varphi_n(X^r_{n-1}) + V^r_n, \tag{8.7}$$

$$Y^r_n = \phi^r_n(X^r_n) + W^r_n. \tag{8.8}$$

where noise V^r_n and W^r_n are iid, zero mean and Gaussian.

We will now illustrate how the distributed filtering and distributed RML algorithms of Chapter 7 can be appropriately modified by employing linearisations of the nonlinear state space model in (8.7)-(8.8). At time n, prior to receiving $[Y^v_n]_{v\in\mathcal{V}}$, let $\theta_n = \{\theta^{i,j}_n\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true localisation parameter θ_* given the available data $Y_{1:n-1}$. Each node will linearize its state and observation model about the filtered and predicted mean respectively. Specifically, a given node r will implement:

$$X^r_n = \varphi_n(\mu^r_{n-1}) + \nabla\varphi_n(\mu^r_{n-1})(X^r_{n-1} - \mu^r_{n-1}) + V^r_n, \tag{8.9}$$

$$Y^r_n = \phi^r_n(\mu^r_{n|n-1}) + \nabla\phi^r_n(\mu^r_{n|n-1})(X^r_n - \mu^r_{n|n-1}) + W^r_n. \tag{8.10}$$

where for a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, $\nabla f \equiv [\nabla f_1, \ldots, \nabla f_d]^{\mathrm{T}}$ with each ∇f_i being the vector of partial derivatives.

Note that after linearisation extra additive terms appear as seen in the setting described by equations (7.1)-(7.2). The appropriate modifications to Algorithm 7.1 are as follows. In (7.15), to the right hand side of $\mu^r_{n|n-1}$, the term $\varphi_n(\mu^r_{n-1}) - \nabla\varphi_n(\mu^r_{n-1})\mu^r_{n-1}$ should be added. Furthermore, all instances of Y^r_n should be replaced with $Y^r_n - \phi^r_n(\mu^r_{n|n-1}) + \nabla\phi^r_n(\mu^r_{n|n-1})\mu^r_{n|n-1}$. Algorithm 7.2 remains the same.

8.4 Belief Propagation and Message Passing

In Chapter 6 we have introduced the Belief Propagation (BP) algorithm in its general message passing form to perform inference on graphs. Before showing how nonparametric approxima-

tions of the messages used in BP can be derived, we shall outline the BP algorithm in a way better suited for our problem.

The BP algorithm is a message passing algorithm for computing the collaborative filtering densities

$$\pi_n^r(x_n^r) = p_n^r(x_n^r|Y_{1:n}), \qquad r \in \mathcal{V}, n \geq 1,$$

At time n, upon receiving the observation $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ the following set of messages are generated for each $(j,r) \in \mathcal{E}$,

$$\widetilde{m}_n^{j,r}(x_n^r) = g_n^j(Y_n^j|x_n^j) \prod_{i\in \text{ne}(j)\setminus\{r\}} \widetilde{m}_n^{i,j}(x_n^j)\Big|_{x_n^j = x_n^r + \theta^{r,j}}. \tag{8.11}$$

Each node will send a different m-message to each of its neighbours. The message sent from node j to r is the product of the m-messages received by node j from all its neighbours, except node r, with its local likelihood $g_n^j(Y_n^j|x_n^j)$.

Thus the message $m_n^{j,r}$ can only be generated once node j has received all the messages from its remaining set of neighbouring nodes. This implies that the messages are initiated by the leafs of the tree, i.e., nodes with only one neighbour. This product is then converted into a function of x_n^r by the coordinate transformation $x_n^j = x_n^r + \theta^{r,j}$. Note that we could have defined instead undirected Graphical model potentials as

$$\psi_n(x_n^r, x_n^j) = \delta_{x_n^r+\theta^{r,j}}(x_n^j),$$
$$\phi_n(x_n^j) = g_n^j(Y_n^j|x_n^j),$$

and perform the standard BP algorithm presented in Section 3.7.1. For the sake of simplicity we preferred to use the notation presented in equation (8.11).

We introduce an extra step to normalise each message

$$m_n^{j,r}(x_n^r) = \frac{\widetilde{m}_n^{j,r}(x_n^r)}{\int \widetilde{m}_n^{j,r}(x_n^r)dx_n^r}, \tag{8.12}$$

so that $m_n^{j,r}$ integrates to one. In standard BP this normalisation step is actually redundant. Here it is added because Nonparametric Belief Propagation interprets the product of messages $\prod_{j\in\text{ne}(r)/j} m_n^{j,r}(x_n^r)$ as a product of normalised distributions, from which it is possible to sample from. Also, this implies that the messages should be integrable.

Once node r has received a message from all its neighbours, the desired collaborative filtering distribution at node r is

$$\pi_n^r(x_n^r) \propto \pi_{n|n-1}^r(x_n^r) g_n^r(Y_n^r|x_n^r) \prod_{j \in \text{ne}(r)} m_n^{j,r}(x_n^r), \tag{8.13}$$

where $\pi_{n|n-1}^r$ is given by a local prediction step at node r. We remark that (8.13) agrees with (8.6).

8.4.1 Nonparametric Belief Propagation for Approximating the Messages

In [155], the NBP approximation to the message $m_n^{j,r}$ is described as a simple two step procedure. The first step is to use $\prod_{i \in \text{ne}(j) \setminus \{r\}} m_n^{i,j}(x_n^j)$ as the instrumental distribution for sampling from $m_n^{j,r}$. This will require that the samples[1] $\{X_n^{j,(l)}\}_{l=1}^N$ from this instrumental be corrected with a weight proportional to $g_n^j(Y_n^j|X_n^{j,(l)})$. The actual step of generating these samples can be achieved by Gibbs sampling assuming that each term in the product that defines the instrumental is a mixture of Gaussians. The second step is to smooth the importance sampling approximation to $m_n^{j,r}$ using a Gaussian kernels, which will ensure that the approximate message is again a mixture of Gaussians and hence the same algorithm can be applied by node r when generating messages to its neighbours.

8.4.1.1 NBP and Sampling from Products of Mixtures

We represent the nonparametric approximation of each message as a mixture of Gaussian kernels

$$\widehat{m_n^{j,r}}(x_n^r) = \sum_{l=1}^N \alpha_n^{r,(l)} \mathcal{N}(x_n^r; \mu_n^{r,(l)}, \Sigma_n^r),$$

where $\mathcal{N}(x^r; \mu_n^{r,(l)}, \Sigma_n^r)$ is the multivariate Gaussian kernel with mean $\mu_n^{r,(l)}$ and covariance Σ_n^r. Each weight $\alpha_n^{(l)}$ is associated to each lth kernel. The weights are normalised,

$$\sum_{l=1}^N \alpha_n^{r,(l)} = 1,$$

so that $\widehat{m_n^{j,r}}(x^r)$ integrates to one. The covariance Σ_n^r is also referred as the *bandwidth* or *smoothing parameter.* Other possible kernels may be used as well, see [147], but for our approach we shall consider only nonparametric representations of Gaussian mixtures.

[1] In this chapter any superscript enclosed in brackets denotes sample index while superscripts not enclosed in brackets, j in the current example, implies the samples are defined w.r.t. the local coordinate system of node j.

In equation (8.11), we see that $m_n^{j,r}(x_n^r)$ is a function of $\prod_{i \in \text{ne}(j)\backslash\{r\}} m_n^{i,j}(x_n^j)$. Before computing $\widehat{m_n^{j,r}}(x^r)$ we assume that all $\widehat{m}_n^{i,j}(x_n^j)$ are available for all $i \in \text{ne}(j)\backslash\{r\}$. Each $\widehat{m}_n^{i,j}(x_n^j)$ is itself a mixture of Gaussians. In order to obtain $\alpha_n^{(l)}, \mu_n^{r,(l)}, \Sigma_n^r$ and thus $\widehat{m_n^{j,r}}$ at each r, a sampling mechanism will be used. At node r, we first sample from the product of the mixtures attributed to all the incoming messages. The sample from this product is then corrected by reweighing to take into account the effect of the local likelihood. Then, we then use a smooth kernel density approximation so that the approximated message $\widehat{m_n^{j,r}}(x^r)$ is guaranteed to be a mixture of Gaussians. Then, it can be passed as an incoming message to the next node, so that the next message passing iteration takes place.

This iterative BP scheme is referred as Nonparametric Belief propagation (NBP), since the underlying messages are approximated by nonparametric representations. The procedure of NBP is outlined in detail as follows. Before each node j passes a message to its neighbour r, it samples N times from a density proportional to the product of mixtures of the neighbouring nodes of j. Let each sample be denoted as

$$X_n^{j,(l)} \overset{i.i.d}{\sim} \prod_{i \in \text{ne}(j)\backslash\{r\}} m_n^{i,j}(\cdot),$$

where $l = 1, ..., N$. Then node j reweighs these samples according to $g_n^j(Y_n^j | x_n^j)$. The sample set $\left\{X_n^{j,(l)}\right\}_{l=1}^N$ is passed to node r, which in turn uses it to produce N samples from the pairwise potential function $\psi_n(x_n^r, x_n^j)$ as follows

$$X_n^{r,(l)} \overset{i.i.d}{\sim} \psi_n(\cdot, X_n^{j,(l)}),$$

where $l = 1, ..., N$. In our case, sampling from $\psi_n(x_n^r, x_n^j)$ is equivalent to applying the coordinate transformation. Of course the last step can be done also at node j. Finally, $\alpha_n^{r,(l)}, \mu_n^{r,(l)}, \Sigma_n^r$ are computed to get a smooth approximation of $m_n^{j,r}$.

It is apparent that NBP inherently relies on being able to sample from a product of mixture distributions. Since the publication of the seminal work of [155] there has been increased attention to the problem of sampling from a product of mixture distributions. Moreover, the same authors in [80] proposed an improved intelligent sampling mechanism for sampling from a product of Gaussian mixtures. We shall now present a Gibbs sampling routine presented for the case of Gaussian mixtures, which is of interest and has also appeared in [81]. For simplicity in the description of the algorithm we drop the time subscript n.

Algorithm 8.1 ***Sampling from a product of Gaussian mixtures as in [155]:*** *Given d mixtures of N Gaussians, we aim to obtain M iid samples, $\{X^{(p)}\}_{p=1}^{M}$, from*

$$\prod_{k=1}^{d}\sum_{l=1}^{N}\omega^{k,(l)}\mathcal{N}(x;\mu^{k,(l)},\Sigma^{k})$$

- *For $j=1,...,d$,*
  - *For the jth mixture, choose a starting label $l^j \in [1,...,N]$, by sampling such that $P(l^j = l) \propto \omega^{j,(l)}$.*
- *For $j=1,...,d$,*
  - *Calculate the mean $\widehat{\mu}$ and $\widehat{\Sigma}$ of the product $\prod_{k\neq j}\mathcal{N}(x;\mu^{k,(l^k)},\Sigma^{k})$ using*

$$\widehat{\Sigma}^{-1} = \sum_{k\neq j}\Sigma^{k^{-1}} \tag{8.14}$$

$$\widehat{\Sigma}^{-1}\widehat{\mu} = \sum_{k\neq j}\Sigma^{k^{-1}}\mu^{k,(l^k)} \tag{8.15}$$

  - *For $i=1,...,N$,*
    - *Use (8.14)-(8.15) to calculate the mean $\overline{\mu}^{j,(i)}$ and covariance $\overline{\Sigma}^{j,(i)}$ of $\mathcal{N}(x;\mu^{j,(i)},\Sigma^{j})\mathcal{N}(x;\widehat{\mu},\widehat{\Sigma})$.*
    - *Using any convenient x compute the weight*

$$\overline{\omega}^{j,(i)} = \omega^{j,(i)}\frac{\mathcal{N}(x;\mu^{j,(i)},\Sigma^{j})\mathcal{N}(x;\widehat{\mu},\widehat{\Sigma})}{\mathcal{N}(x;\overline{\mu}^{j,(i)},\overline{\Sigma}^{j,(i)})}.$$

  - *Sample a new label l^j such that $P(l^j = l) \propto \overline{\omega}^{j,(l)}$.*
- *Repeat the previous step κ times until convergence.*
- *Using (8.14)-(8.15) compute mean μ and covariance Σ of $\prod_{k=1}^{d}\mathcal{N}(x;\mu^{k,(l^k)},\Sigma^{k})$.*
- *For $p=1,...,M$, sample $X^{(p)} \overset{i.i.d.}{\sim} \mathcal{N}(x;\mu,\Sigma)$.*

Note that it is possible to generalise this Gibbs sampling approach to products of non-Gaussian mixtures. Whilst the above Gibbs sampling approaches presented is very elegant,

there have been expressed slight considerations about executing an MCMC procedure within an iterative algorithmic scheme such as Belief Propagation. MCMC within a particle filtering context has been used [24], it sometimes referred to as being inefficient [132]. Assuming we run κ Gibbs sampling iterations, the computational cost of such an algorithms is $O(\kappa d N^2)$. In the numerical example we observed that κ was quite small and Gibbs sampling converged very fast. In addition, the computational cost can be reduced by employing the auxiliary particle filtering approach of [132]. This was done in [29] for the problem of filtering within dynamic Graphical Models. Our methodology can be extended to follow the framework of [29] without any problem.

8.4.1.2 Approximating the m-message

We shall now show how $\alpha_n^{r,(l)}, \mu_n^{r,(l)}, \Sigma_n^r$ can be computed in order to obtain $\widehat{m_n^{j,r}}(x^r)$. Assume that all $m_n^{i,j}(x_n^j)$ integrate to one and we can approximate them as a mixtures of Gaussians. We can generate samples $\{X_n^{j,(l)}\}_{l=1}^N$ from a density proportional to $\prod_{i \neq r} m_n^{i,j}(x_n^j)$ using the Gibbs sampling approach detailed in earlier in Algorithm 8.1, where the product $\prod_{i \neq r}$ denotes $\prod_{i \in ne(j) \setminus \{r\}}$. The samples $\{X_n^{j,(l)}\}_{l=1}^N$ are then assigned the weight

$$\varrho_n^{r,(l)} := \frac{g_n^j(Y_n^j | X_n^{j,(l)})}{\sum_{l=1}^N g_n^j(Y_n^j | X_n^{j,(l)})}$$

Moreover, the weighted samples $\{(\varrho^{r,(l)}, X_n^{j,(l)})\}_{l=1}^N$ are transformed to $\{(\varrho^{r,(l)}, \widetilde{X}_n^{r,(l)} := X_n^{j,(l)} - \theta^{r,j})\}_{l=1}^N$ by using the appropriate coordinate transformation.

The next step is to smooth the approximation for $m_n^{j,r}(x^r)$. We compute the empirical covariance of $\{(\varrho^{r,(l)}, \widetilde{X_n}^{r,(l)})\}_{l=1}^N$,

$$\overline{\Sigma}_n = \sum_{l=1}^N \varrho_n^{r,(l)} (\widetilde{X}_n^{r,(l)} - \overline{\mu}_n)(\widetilde{X}_n^{r,(l)} - \overline{\mu}_n)^T$$

where

$$\overline{\mu}_n = \sum_{l=1}^N \varrho_n^{r,(l)} \widetilde{X}_n^{r,(l)}.$$

We then set

$$\Sigma_n^r = diag(\overline{\Sigma}_{n_{ii}})_{i=1}^{n_x},$$

$$\mu_n^{r,(l)} = \widetilde{X}_n^{r,(l)} = X_n^{j,(l)} - \theta^{r,j},$$

$$\alpha_n^{r,(l)} = \varrho_n^{(l)},$$

where $diag(\overline{\Sigma}_{n_{ii}})_{i=1}^{n_x}$ denotes the diagonal matrix composed from the elements of $\overline{\Sigma}_n$. The resulting smooth approximation of $m_n^{j,r}$ can be equivalently written as

$$\widehat{m_n^{j,r}}(x_n^r) = \frac{1}{|\Sigma_n^r|^{1/2}} \sum_{l=1}^{N} \alpha_n^{r,(l)} \mathcal{N}(\sqrt{\Sigma_n^r}(x_n^r - \mu_n^{r,(l)})) \tag{8.16}$$

where $\mathcal{N}(x)$ is the pdf of a multivariate Gaussian random variable with mean zero and variance I. The m-message from node j to node r at iteration n may thus be summarised as $\left(\{(\alpha_n^{(l)}, \mu_n^{r,(l)})\}_{l=1}^N, \Sigma_n^r\right)$. In fact, the kernel smoothing step, i.e. computing Σ_n^r, can actually be performed at the destination node r if desired. Note that this smoothing step using the empirical covariance of $\{(\alpha_n^{(l)}, \mu_n^{r,(l)})\}_{l=1}^N$ is a standard approach for constructing nonparametric approximations of a density. For more details, see [147].

8.4.2 Nonparametric Belief Propagation for Approximating the Gradients of the Messages

In the remainder of this section, we will see how the NBP algorithm can provide smooth approximations for the gradient of $m_n^{j,r}(x_n^r)$. This message, which is not part of the usual BP algorithm, is needed for the propagation of the derivatives of the collaborative filters $(\mathring{\pi}_{n|n-1}, \mathring{\pi}_n)$. The gradient propagation of the gradients is a necessary for estimating the localisation parameters using RML.

At time n, the messages $\mathring{m}_n^{j,r}(x_n^r)$ are defined for each $(j,r) \in \mathcal{E}$ as

$$\mathring{m}_n^{j,r}(x_n^r) \equiv \nabla_{x^r} m_n^{j,r}(x_n^r). \tag{8.17}$$

Let $\widehat{\mathring{m}_n^{j,r}}$ denote the smooth approximation of $\mathring{m}_n^{j,r}(x_n^r)$. Once $\widehat{m_n^{j,r}}$ has been generated, $\widehat{\mathring{m}_n^{j,r}}$ can be generated without any further need of sampling or smoothing. We will develop an Importance Sampling (IS) approximation to $\mathring{m}_n^{j,r}(x^r)$ using $m_n^{j,r}(x^r)$ as the instrumental. This requires the use of a re-weighting step.

The $\mathring{m}_n^{j,r}$-message from node j to r can be written as

$$\mathring{m}_n^{j,r}(x_n^r) = C^{-1} \left(\frac{\nabla g_n^j(Y_n^j|x_n^j)}{g_n^j(Y_n^j|x_n^j)} + \sum_{i \neq r} \frac{\mathring{m}_n^{i,j}(x_n^j)}{m_n^{i,j}(x_n^j)} \right) g_n^j(Y_n^j|x_n^j) \prod_{i \neq r} m_n^{i,j}(x_n^j) \Bigg|_{x_n^j = x_n^r + \theta^{r,j}}, \tag{8.18}$$

where C^{-1} is an unknown normalizing constant. We will use $C^{-1} g_n^j(Y_n^j|x^j) \prod_{i \neq r} m_n^{i,j}(x^j)$ as

the importance sampling distribution. Before in order to derive an approximation for message $m_n^{j,r}$ above we generated weighted samples $\{X_n^{j,(l)}\}_{l=1}^N$ from $C^{-1}g_n^j(Y_n^j|x_n^j)\prod_{i\neq r} m_n^{i,j}(x_n^j)$ by sampling from $\prod_{i\neq r} m_n^{i,j}(x_n^j)$ then re-weighting according to $g_n^j(Y_n^j|x_n^j)$. This way we obtained $\{(\alpha_n^{r,(l)}, X_n^{j,(l)})\}_{l=1}^N$ as a weighted sample from $C^{-1}g_n^j(Y_n^j|x_n^j)\prod_{i\neq r} m_n^{i,j}(x_n^j)$. Now, we will re-weight $\{(X_n^{j,(l)}, \alpha_n^{r,(l)})\}_{l=1}^N$ according to $\left(\frac{\nabla g_n^j(Y_n^j|x_n^j)}{g_n^j(Y_n^j|x_n^j)} + \sum_{i\neq r}\frac{\mathring{m}_n^{i,j}(x_n^j)}{m_n^{i,j}(x_n^j)}\right)$ to a particle approximation for $C^{-1}\left(\frac{\nabla g_n^j(Y_n^j|x_n^j)}{g_n^j(Y_n^j|x_n^j)} + \sum_{i\neq r}\frac{\mathring{m}_n^{i,j}(x_n^j)}{m_n^{i,j}(x_n^j)}\right) g_n^j(Y_n^j|x_n^j)\prod_{i\neq r} m_n^{i,j}(x_n^j)$ as

$$\sum_{l=1}^N \alpha_n^{r,(l)}\left(\frac{\nabla g_n^j(Y_n^j|X_n^{j,(l)})}{g_n^j(Y_n^j|X_n^{j,(l)})} + \sum_{i\neq r}\frac{\mathring{m}_n^{i,j}(X_n^{j,(l)})}{m_n^{i,j}(X_n^{j,(l)})}\right)\delta_{X_n^{j,(l)}}(x_n^j). \tag{8.19}$$

To get $\mathring{m}_n^{j,r}(x_n^r)$ the coordinate transformation must be applied, i.e. as done in the previous section for the sample set $\{X_n^{j,(l)}\}_{l=1}^N$, we set x_n^r to $x_n^j - \theta^{r,j}$ to get: $\mu_n^{r,(l)} = X_n^{j,(l)} - \theta^{r,j}$.

Also, the same kernel bandwidth that is used to smooth $m_n^{j,r}$ is also used to smooth $\mathring{m}_n^{j,r}$, which gives the nonparametric approximation for $\mathring{m}_n^{j,r}$ as

$$\widehat{\mathring{m}_n^{j,r}}(x^r) = \frac{1}{|\Sigma_n^r|^{1/2}}\sum_{l=1}^N \alpha_n^{r,(l)}\left(\frac{\nabla g_n^j(Y_n^j|X_n^{j,(l)})}{g_n^j(Y_n^j|X_n^{j,(l)})} + \sum_{i\neq r}\frac{\mathring{m}_n^{i,j}(X_n^{j,(l)})}{m_n^{i,j}(X_n^{j,(l)})}\right)\mathcal{N}(\sqrt{\Sigma_n^r}(x_n^r - \mu_n^{r,(l)})). \tag{8.20}$$

8.5 Particle Approximations for the filter derivatives and the likelihood gradients

In the previous section we established how to generate mixtures of Gaussian kernels that approximate the incoming messages at each node. The message passing algorithm has been reduced to a sampling procedure where nonparametric approximations of the messages received by neighbouring nodes are used to compute a nonparametric approximation of each incoming message and its gradient at each node. This enables us to use these approximations for $m_n^{j,r}(x_n^r)$ and $\mathring{m}_n^{j,r}(x^r)$ when deriving SMC approximations for the collaborative filtering and prediction density, their derivatives and the likelihood gradients needed to update the parameter θ_n using distributed RML.

At time $n-1$, let $\{(\omega_{n-1}^{r,(i)}, X_{n-1}^{r,(i)})\}_{i=1}^N$ and $\{(\mathring{\omega}_{n-1}^{r,(i)}, X_{n-1}^{r,(i)})\}_{i=1}^N$ be the particle approximations at node r of π_{n-1}^r and $\mathring{\pi}_{n-1}^r$ respectively:

$$\widehat{\pi_{n-1}^r}(x_{n-1}^r) = \sum_{i=1}^N \omega_{n-1}^{r,(i)}\delta_{X_{n-1}^{r,(i)}}(x_{n-1}^r)$$

$$\widehat{\mathring{\pi}^r_{n-1}}(x^r_{n-1}) = \sum_{i=1}^{N} \mathring{\omega}^{r,(i)}_{n-1} \delta_{X^{r,(i)}_{n-1}}(x^r_{n-1})$$

We assume these particle approximations are available at time n, and we aim to propagate them in time. In addition, we assume that (j, r) is a valid edge, and also node r, controls or updates $\theta^{r,j}$ recursively. We want to derive a particle approximation for the likelihood gradient

$$J^r_{\theta^{r,j}_n} = \nabla_{\theta^{r,j}} \log p(Y_n|Y_{1:n-1})|_{\theta^{r,j}_n} .$$

At the same time the rest of the nodes propagate their collaborative filters together with their gradients and also update the localisation parameter defined by their adjacent edges. At node r, we shall consider the approximations for the filters, their gradient and the likelihood gradient separately.

8.5.1 Collaborative Filters

At time n the collaborative filters have to be computed for each node. The prediction density at each node is given by

$$\pi^r_{n|n-1}(x^r_n) = \int f_n(x^r_n|x^r_{n-1})\pi^r_{n-1}(x^r_{n-1})dx^r_{n-1}.$$

Replacing π^r_{n-1} in the above integral with the particle approximation $\widehat{\pi^r_{n-1}}(x^r_n)$ gives the following particle approximation for the prediction density

$$\widehat{\pi^r_{n|n-1}}(x^r_n) = \sum_{i=1}^{N} \omega^{r,(i)}_{n-1} f_n(x^r_n|X^{r,(i)}_{n-1}). \tag{8.21}$$

Moreover, the collaborative filter at node r at time n is,

$$\pi^r_n(x^r_n) \propto \pi^r_{n|n-1}(x^r_n) g^r_n(Y^r_n|x^r_n) \prod_l m^{l,r}_n(x^r_n).$$

A particle approximation of π^r_n is implemented as follows. For each $1 \leq i \leq N$, sample $X^{r,(i)}_n$ from a user specified importance density $q(.|X^{r,(i)}_{n-1}, Y^r_n)$ and assign it the weight

$$\widetilde{\omega}^{r,(i)}_n = \frac{\omega^{r,(i)}_{n-1} f_n(X^{r,(i)}_n|X^{r,(i)}_{n-1}) g^r_n(Y^r_n|X^{r,(i)}_n) \prod_l \widehat{m^{l,r}_n}(X^{r,(i)}_n)}{q(X^{r,(i)}_n|X^{r,(i)}_{n-1}, Y^r_n)}. \tag{8.22}$$

The weights are then normalised to sum to one

$$\omega^{r,(i)}_n = \frac{\widetilde{\omega}^{r,(i)}_n}{\sum_{i=1}^{N} \widetilde{\omega}^{r,(i)}_n} \tag{8.23}$$

A resampling step can be then used if the effective sample size is small, to obtain $\widehat{\pi_n^r}(x_n^r)$. This follows a SISR approach presented in Chapter 2. Note that now we have used $\widehat{m_n^{l,r}}$ to approximate each incoming message $m_n^{l,r}$.

Many methods of optimally designing $q(.|X_{n-1}^{r,(i)}, Y_n^r)$ can be found in [47]. We shall not make a detailed discussion on this as it is a well known topic, but rather sketch the most common approach. Consider for example the nonlinear state-observation model in (8.7)-(8.8) of Section 8.3, The state transition density $f_n(X_n^{r,(i)}|X_{n-1}^{r,(i)})$ is Gaussian and we can obtain a Gaussian approximation to the likelihood by linearising $\phi_n^r(x^r)$ about $\varphi_n(X_{n-1}^{r,(i)})$. In this manner a Gaussian approximation to $q(.|X_{n-1}^{r,(i)}, Y_n^r)$ is constructed. Alternatively, a Laplace or an Unscented approximation to $f_n(x_n^r|X_{n-1}^{r,(i)})g_n^r(Y_n^r|x^r)$ can be constructed as a candidate for $q(.|X_{n-1}^{r,(i)}, Y_n^r)$. For more details, see [47].

8.5.2 Filter Derivatives

Our framework is designed so that each node can update or control any parameter defined by its adjacent edges. The particular choice of which parameters are controlled by which nodes is flexible, but for simplicity we assume that each parameter is controlled only by a specific node. This restriction is not necessary, but it serves some practical purposes related with implementation. For example, if a particular node r controls no edges then it is not necessary to keep $\mathring{\pi}_n^r$.

Assume node r controls edge (r, j). For the gradient of the prediction density we have

$$\mathring{\pi}_{n|n-1}^r(x_n^r) = \int f_n(x_n^r|x_{n-1}^r)\mathring{\pi}_{n-1}^r(x_{n-1}^r)dx_{n-1}^r.$$

Particle approximations to $\mathring{\pi}_{n|n-1}^r$, as for $\pi_{n|n-1}^r$, are obtained by replacing $\mathring{\pi}_{n-1}^r$ in the above integral with its particle approximation, giving

$$\widehat{\mathring{\pi}_{n|n-1}^r}(x_n^r) = \sum_{i=1}^N \mathring{\omega}_{n-1}^{r,(i)} f_n(x_n^r|X_{n-1}^{r,(i)}). \tag{8.24}$$

For the gradient of the collaborative filtering density we require

$$\mathring{\pi}_n^r(x_n^r) = \left(\frac{\mathring{m}_n^{j,r}(x_n^r)}{m_n^{j,r}(x_n^r)} + \frac{\mathring{\pi}_{n|n-1}^r(x_n^r)}{\pi_{n|n-1}^r(x_n^r)} - \int \left[\frac{\mathring{m}_n^{j,r}(x_n^r)}{m_n^{j,r}(x_n^r)} + \frac{\mathring{\pi}_{n|n-1}^r(x_n^r)}{\pi_{n|n-1}^r(x_n^r)}\right] \pi_n^r(x_n^r)dx_n^r\right)\pi_n^r(x_n^r).$$

Let $\{(\omega_n^{r,(i)}, X_n^{r,(i)})\}_{i=1}^N$ be samples from π_n^r. We can obtain an importance sampling approxima-

tion of $\mathring{\pi}_n^r$, with π_n^r as the instrumental distribution as follows,

$$\widehat{\mathring{\pi}_n^r}(x_n^r) = \sum_{i=1}^{N} \mathring{\omega}_n^{r,(i)} \delta_{X_n^{r,(i)}}(x_n^r)$$

where the weight is given by

$$\mathring{\omega}_n^{r,(i)} = \omega_n^{r,(i)} \left(\frac{\widehat{\mathring{m}_n^{j,r}}(X_n^{r,(i)})}{\widehat{m_n^{j,r}}(X_n^{r,(i)})} + \frac{\widehat{\mathring{\pi}_{n|n-1}^r}(X_n^{r,(i)})}{\widehat{\pi_{n|n-1}^r}(X_n^{r,(i)})} - \sum_{i'=1}^{N} \omega_{r,n}^{(i')} \left[\frac{\widehat{\mathring{m}_n^{j,r}}(X_n^{r,(i')})}{\widehat{m_n^{j,r}}(X_n^{r,(i')})} + \frac{\widehat{\mathring{\pi}_{n|n-1}^r}(X_n^{r,(i')})}{\widehat{\pi_{n|n-1}^r}(X_n^{r,(i')})} \right] \right). \tag{8.25}$$

Again as done for propagating $\widehat{\pi_n^r}$, when computing the weight $\mathring{\omega}_n^{r,(i)}$we use the ratio of the nonparametric versions of the messages $\frac{\widehat{\mathring{m}_n^{j,r}}}{\widehat{m_n^{j,r}}}$ to approximate $\frac{\mathring{m}_n^{j,r}}{m_n^{j,r}}$.

8.5.3 Likelihood Gradients

Upon making measurements at time n, the localisation parameters $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ are updated to $\{\theta_{n+1}^{i,j}\}_{(i,j)\in\mathcal{E}}$ as follows. Again, assuming node r controls edge (r,j), the localisation parameter update given by RML can be written as

$$\begin{aligned}
\theta_{n+1}^{r,j} &= \theta_n^{r,j} + \gamma_{n+1} J_{\theta_n^{r,j}}^r \\
&= \theta_n^{r,j} + \gamma_n^r \begin{pmatrix} \int g_n^r(Y_n^r|x_n^r) \mathring{m}_n^{j,r}(x_n^r) \prod_{l\neq j} m_n^{l,r}(x_n^r) \pi_{n|n-1}^r(x_n^r) dx^r \\ + \int g_n^r(Y_n^r|x_n^r) \prod_l m_n^{l,r}(x_n^r) \mathring{\pi}_{n|n-1}^r(x_n^r) dx_n^r \end{pmatrix}.
\end{aligned}$$

Given the particle approximations $\widehat{\pi_{n-1}^r}$, $\widehat{\mathring{\pi}_{n-1}^r}$, and the nonparametric approximations $\widehat{m_n^{j,r}}$, $\widehat{\mathring{m}_n^{j,r}}$, we can approximate the above quantities that multiply the step-size γ_{n+1}^r as follows. For $1 \leq i \leq N$, sample $X_n^{r,(i)}$ from $f_n(.|X_{n-1}^{r,(i)})$ and form the estimate of $J_n^{r,j}$ as

$$\widehat{J_{\theta_n^{r,j}}^r} = \sum_{i=1}^{N} g_n^r(Y_n^r|X_n^{r,(i)}) \prod_{l\neq j} \widehat{m_n^{l,r}}(X_n^{r,(i)}) \left(\omega_{n-1}^{r,(i)} \widehat{\mathring{m}_n^{j,r}}(X_n^{r,(i)}) + \mathring{\omega}_{n-1}^{r,(i)} \widehat{m_n^{j,r}}(X_n^{r,(i)}) \right). \tag{8.26}$$

8.6 Distributed Particle Algorithm for Sensor Self Localisation

In this section we shall summarise how to solve the self localisation problem using a completely decentralised RML approach that combines NBP and SMC. NBP is used to provide smooth approximations of the messages around the network and SMC to compute the integrals needed to implement the filtering recursion, the recursion of the filter derivatives and the parameter update using the gradients of the likelihood.

For each edge $(r, j) \in \mathcal{E}$ assign a valid controlling node, say r, so as to update parameter $\theta^{r,j}$. At each iteration n all edges should be updated in a cyclic fashion using a valid root node and hence $\theta^{r,j}$ will be updated for all $(r, j) \in \mathcal{E}$. The distributed algorithm is as follows.

Algorithm 8.2 ***Particle Algorithm for Sensor Self Localisation using RML and NBP:*** *At time n, we start having $\{(\omega_{n-1}^{r,(i)}, X_{n-1}^{r,(i)})\}_{i=1}^N$ and $\{(\mathring{\omega}_{n-1}^{r,(i)}, X_{n-1}^{r,(i)})\}_{i=1}^N$ available from time $n-1$.*

Prediction Filters: *For all nodes $r \in \mathcal{V}$ propagate the prediction filters $\widehat{\pi_{n|n-1}^r}(x_n^r)$ and their derivatives $\widehat{\mathring{\pi}_{n|n-1}^r}(x_n^r)$ as in equations (8.21) and (8.24) respectively.*

Propagate messages: *After Y_n is received, at node j, sample $\{\mathcal{X}_n^{j,(l)}\}_{l=1}^N$ from a density proportional to $\prod_{i \in ne(j)\setminus\{r\}} m_n^{i,j}(x_n^j)$ using Algorithm 8.1. Assign each sample the weight $\alpha_n^{j,(l)} := \frac{g^j(Y_n^j|\mathcal{X}_n^{j,(l)})}{\sum_{l=1}^N g^j(Y_n^j|\mathcal{X}_n^{j,(l)})}$ and pass the weighted sample at node r. At node r, set $\mu_n^{r,(l)} = \mathcal{X}_n^{j,(l)} - \theta_n^{r,j}$ and compute $\widehat{m_n^{j,r}}$, $\widehat{\mathring{m}_n^{j,r}}$ as given by (8.16),(8.20) respectively.*

Update the parameter $\theta^{r,j}$: *At each node r sample $\overline{X}_n^{r,(i)}$ from $f_n(.|X_{n-1}^{r,(i)})$ and update the localisation parameter as*

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r \widehat{J_{\theta_n^{r,j}}^r},$$

where $\widehat{J_{\theta_n^{r,j}}^r}$ is given by (8.26).

Collaborative Filters: *At each node r sample $X_n^{r,(i)}$ from $q(.|X_{n-1}^{r,(i)}, Y_n^r)$ and assign it the weights $\omega_n^{r,(i)}$ and $\mathring{\omega}_n^{r,(i)}$ given by equations (8.23) and (8.25) respectively.*

- ***Resampling**: At all nodes r resample only for the filter. Resample particles $\widetilde{X}_n^{r,(i)}$ with respect to weights $\omega_n^{r,(i)}$ to obtain N new particles and denote the new particle by $\varphi_n(i)$, where $\varphi_n(i)$ is determined by the resampling mechanism. The particle approximations are now given by $\widehat{\pi_n^r}(x_n^r) = \frac{1}{N}\sum_{i=1}^N \delta_{X_{v,n}^{r,(\varphi_n(i))}}(x_n^r)$ and $\widehat{\mathring{\pi}_n^r}(x_n^r) = \sum_{i=1}^N \mathring{\omega}_n^{r,(i)} \delta_{X_n^{r,(i)}}(x_n^r)$.*

Typically, the step-sizes are selected as $\gamma_n^r = n^{-\gamma^r}$, where $\gamma^r > 0.5$, so that $\sum_n \gamma_n^r = \infty$ and $\sum_n {\gamma_n^r}^2 < \infty$. As for standard Belief Propagation [130], this algorithm is designed for a sensor network that admits a tree structure.

8.7 Numerical Examples

We would like to solve the self localisation problem using Algorithm 8.2 and Algorithm 7.2 when linearisation is used together with a distributed Extended Kalman filter obtained by Algorithm 7.1. For a bearings-only tracking example, we will demonstrate in both cases the convergence of the localisation parameters to their true value. As in Chapter 6 we will be observing a sequence of targets passing through the field of view of the sensor network. Each target will originate from a different point selected randomly from the boundary of the field of view and will be moving towards a different direction. Only one target will appear at any instant and we shall therefore avoid any association problems. The reason for observing many targets are discussed in Chapter 6 and involves observability issues for θ, and also the practical issue of having to use a new target when one has left the field of view. Each time a new target appears the filter is initialised by sampling uniformly from the area around the target defined by the region $[x^r - d, x^r + d]$. We shall use $d = 5$ for both the x- and y- direction.

As far as the dynamics of each target are concerned, at each node r the state follows a linear Gaussian update as follows

$$X_n^r = A_n X_{n-1}^r + B_n V_n^r,$$

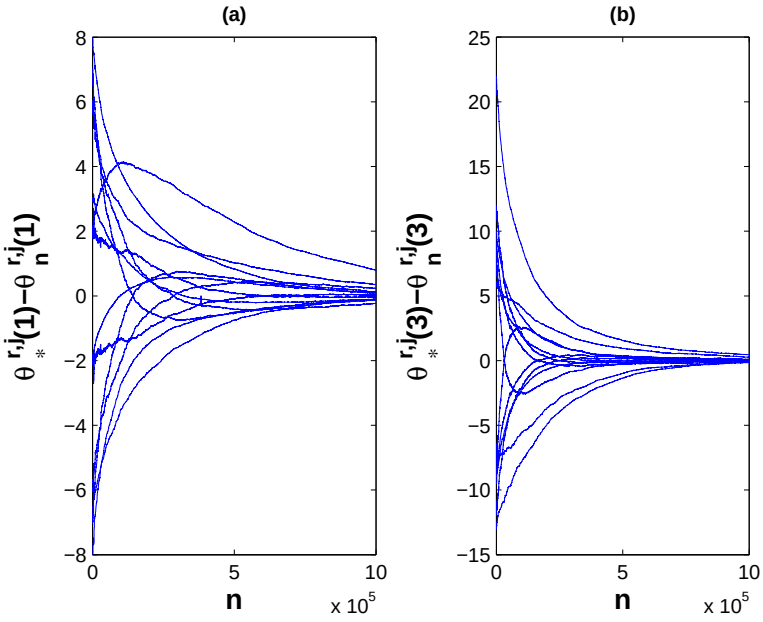
with $V_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, Q_n)$, where A_n, B_n, Q_n are the same as in Chapter 7. At each node r, the target being tracked yields observation $Y_{r,n}$ and obeys the following dynamics

$$Y_n^r = tan^{-1}(X_n^r(1)/X_n^r(3)) + W_n^r.$$

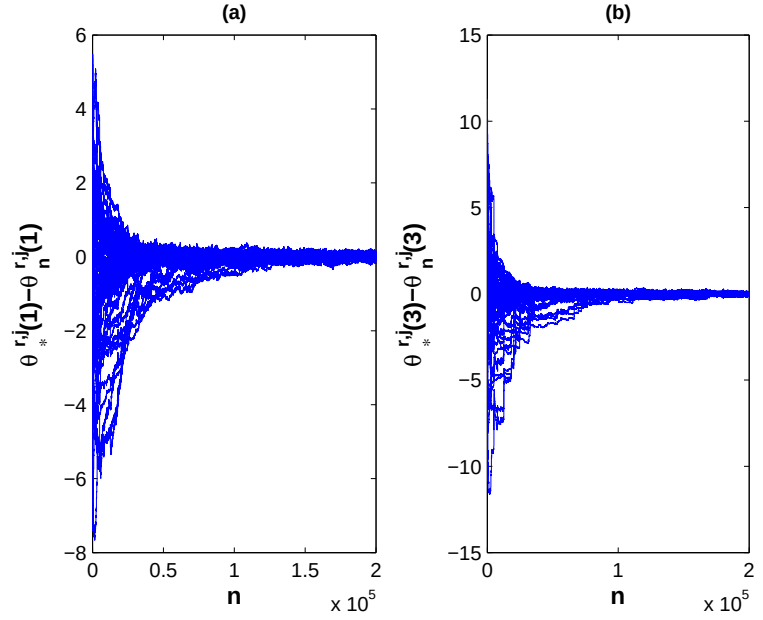
with $W_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, (\sigma_n^r)^2)$, where at each time n, σ_n^r is randomly chosen for each node r from the uniform distribution over the values $[0.1, 0.5]$ so that a time varying observation can be implemented. Note that σ_n^r is made known only to node r. Also, for the RML step sizes, we use a constant step size $\gamma_r = 10^{-3}$ and we initialize $\theta^{r,j} = 0$ for all $(r, j) \in \mathcal{E}$.

8.7.1 EKF Implementation

We we will use the two different sensor networks we used earlier in Chapter 7. The deployment of the sensors is as shown in Figures shown in Figures 7.3(a) and 7.3(b) (Chapter 7). We plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ for each edge in Figure 8.1(a) and Figure 8.1(b) respectively.



(a) Convergence of Algorithm 7.2 using the linearised observation model for the sensor network of Figure 7.3(a)



(b) Convergence of Algorithm 7.2 using the linearised observation model for the sensor network of Figure 7.3(b)

Figure 8.1: *Parameter error after each RML iteration between the true localization parameter and current estimate for all edges of each sensor network used. In each plot (a) and (b) show the errors in the x- and y- coordinates respectively.*

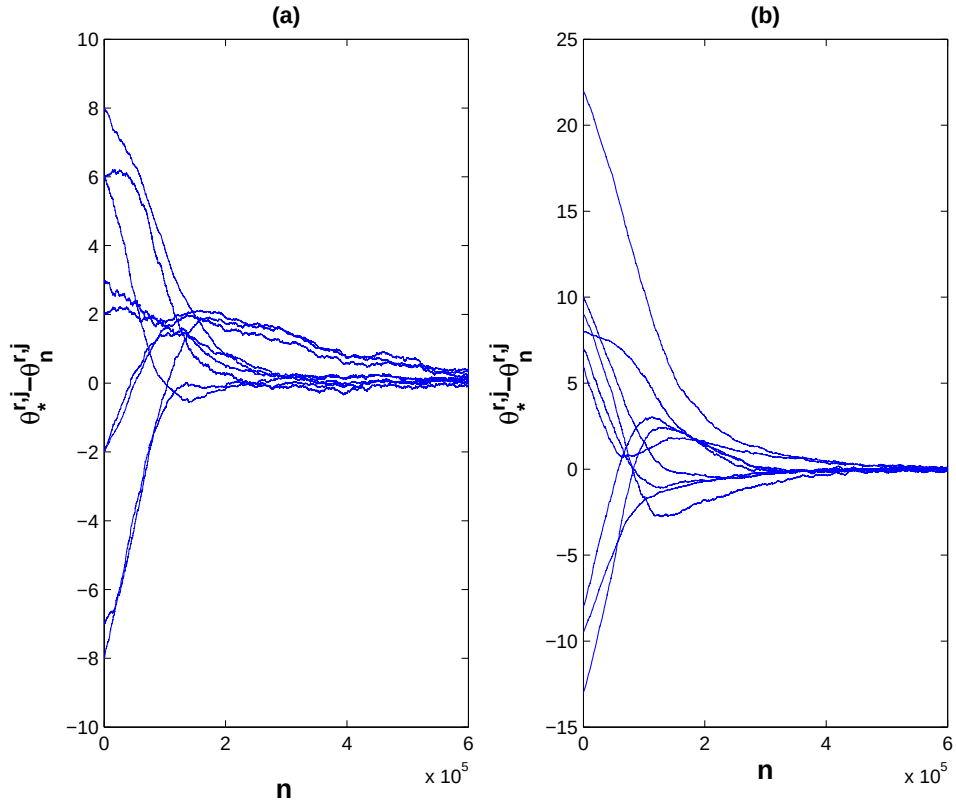


Figure 8.2: *Convergence of Algorithm 8.2 using the nonlinear observation model for the sensor network of Figure 7.3(a). Error at each iteration between the true parameter and current update of the localization parameter for all edges of the sensor net. (a) and (b) show the errors in the x- and y- coordinates respectively.*

8.7.2 SMC Implementation with NBP

For the sensor network shown in Figures 7.3(a), we choose nodes $\{3, 4, 6, 9\}$ as root nodes and update at each iteration their adjacent edges. In Figure 8.2 we plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ against iteration n. We see that eventually all errors converge to zero. We observe that this implementation converges faster and with smaller errors than the one obtained earlier through linearisation and Extended Kalman filtering for the same sensor network.

8.8 Conclusions

In this chapter we have demonstrated how particle methods can be can be combined with Nonparametric Belief propagation, when used within the Graphical models context. Furthermore, we have extended this methodology for the static parameter inference problem using a dis-

tributed implementation of Recursive Maximum Likelihood (RML). The resulting algorithm can be thought as an extension of the Nonparametric Belief Propagation to compute likelihood gradients. In addition, we have extended the algorithms of Chapter 7 for nonlinear models by using appropriate linearisations and Extended Kalman Filtering. All algorithms were applied to solve the sensor localisation problem for sensor networks, when bearings only tracking is used.

9

Conclusions

Sequential Monte Carlo (SMC) methods is a rapidly growing area of research in many branches of science, engineering and computational statistics. In this thesis we aimed to use SMC for constructing particle approximations that would enable gradient methods to be used in optimisation problems regarding control, on line parameter estimation and Graphical models. We shall conclude by making some remarks on the contribution of our work to SMC methodolgy and some possible future directions.

9.1 Contributions

In this section we shall summarize the contributions and novelties of the work presented earlier in each chapter of the thesis.

Chapter 4 focuses on solving the sensor scheduling problem when cast as a controlled Hidden Markov Model. We consider the case in which the state, observation and action spaces are continuous. This general case is important as it is the natural framework for many applica-

tions. In sensor scheduling, our aim is to minimise the variance of the estimation error of the hidden state with respect to the action sequence. We present a novel SMC method that uses a stochastic gradient algorithm to find optimal actions. This is in contrast to existing works in the literature that only solve approximations to the original problem.

In Chapter 5 we presented how an SMC can be used to solve a risk sensitive control problem. We adopt the use of the Feynman-Kac representation of a controlled Markov chain flow and exploit the properties of the logarithmic Lyapunov exponent, which lead to a policy gradient solution for the parameterised problem. The resulting SMC algorithm follows a similar structure with the Recursive Maximum Likelihood (RML) algorithm presented for online parameter estimation.

In Chapters $6, 7$ and 8, dynamical Graphical models were combined with with state space models for the purpose of online decentralised inference. We have concentrated more on the distributed parameter estimation problem using two Maximum Likelihood techniques, namely Recursive Maximum Likelihood (RML) and Expectation Maximization (EM). The resulting algorithms can be interpreted as an extension of the Belief Propagation (BP) algorithm to compute likelihood gradients. In order to design an SMC algorithm, in Chapter 8 we have approximated the messages of BP using Nonparametric Belief propagation. The algorithms were successfully applied to solve the sensor localisation problem for sensor networks of small and medium size.

9.2 Future Directions

In this final section, we discuss of any possible extensions of the work presented in this thesis. First of all, a natural limitation of using gradient methods for optimisation problems is that they guarantee convergence only to local optimum. Appropriate initialisation is therefore important. We have not considered this at all so far, as there are many heuristics to tackle this issue. Of course, one can also consider other optimisation methods suited to general state spaces, such as MCMC or SMC algorithms for Simulated Annealing. A comparison would be very interesting. From our point of view though, we felt that since in all our problems it was possible to perform differentiation to the cost or reward function, gradient methods seemed as the most sensible choice to start from.

Moreover, in the work of Chapter 4 we have considered the minimisation of a variance cri-

terion. Without any alterations the results can apply also for a Kullback-Leibler (KL) information criterion. Another useful extension may be to derive gradient free algorithms using Finite Difference Stochastic Approximation or Simultaneous Perturbation Stochastic approximation. This might be of a benefit as far as the high computational cost required is concerned.

The work of Chapter 5 is still at its very first stages and more complex numerical examples have to be considered. The main purpose of including it in this thesis was to illustrate how the underlying methodology found in RML can be useful in difficult control problems and register some early results for risk sensitive control. This work may be very useful in other applications, such as optimising portfolios in finance.

Chapters $6, 7$ and 8 describe how Graphical models can be integrated with state space model for the purpose of decentralised inference. We have particulary focused on the sensor network localisation problem. Of course our methodology is generic and we could have included also the effect of rotation, which would be useful for many computer vision problems, where cameras of different orientations are used. Furthermore, in Chapter 8 the use of an Gibbs sampling step to perform Nonparametric belief propagation can be replaced by sequential belief propagation using an auxiliary SMC filter as done in [29].

Bibliography

[1] Akyildiz I.F., Su W. , Sankarasubramaniam Y., and Cayirci E. (2002) "A Survey on Sensor Networks ", *IEEE Communications Magazine*, Vol.40(8), pp:102–114.

[2] Anderson B.D.O. and Moore J.B. (1979) *Optimal Filtering*, Prentice-Hall, New York.

[3] Anderson B.D.O. and Moore J.B. (1989) *Optimal Control: Linear Quadratic Methods*, Prentice-Hall, New York.

[4] Andrieu C. and Doucet A. (2002) Particle filtering for partially observed Gaussian state space models. *J. Royal Statist. Soc.* B, **64**, 827-836.

[5] Andrieu C., and Doucet A. (2003). Online expectation maximization type algorithms for parameter estimation in general state space models, *In Proc. IEEE ICASSP*.

[6] Andrieu, C., Doucet, A. and Fitzerald, W. J.(2000) An introduction to Monte Carlo Methods for Bayesian Data Analysis, In *Nonlinear Dynamics and Statistics* (eds Mees, A. I.) Birkhauser.

[7] Andrieu, C ., Doucet, A. and Punskaya. E. (2001) Sequential Monte Carlo methods for optimal filtering. In *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N, and Gordon, N. J.) Springer-Verlag.

[8] Andrieu C., Doucet A., Singh S.S., Tadić V., (2004) Particle Methods for Change Detection, Identification and Control, *Proceedings of the IEEE*, vol. 92, pp. 423 - 438.

[9] Andrieu C., A. Doucet and Tadic. V.B. (2005) Online simulation-based methods for parameter estimation in non linear non Gaussian state-space models, *In Proc. IEEE CDC*, Sevilla.

[10] Andrieu C., Doucet A. and Tadic V.B. (2005) Online parameter estimation in general state space models, *Proc. IEEE CDC/ECC*.

[11] Atar, R. and Zeitouni, O. (1997) Exponential stability for nonlinear filtering, *Ann. Inst. Henri Poincare, Probab. Statist.* 33 697–725.

[12] Balaji S., and Meyn S.P. (2000) Multiplicative Ergodicity and Large Deviations for an Irreducible Markov Chain *Stochastic Processes and Their Application*. Vol. 1, pp. 123-144.

[13] Bar-Shalom Y. and Li X. R. (1993) *Estimation and Tracking: Principles, Techniques, and Software,* Artech House Inc.

[14] Bartusek J.D. and Makowski A.M. (1994) On Stochastic Approximations driven by sample averages: convergence results via the ODE method, Technical Report ISR T.R. 94-4, University of Maryland, Institute for Systems Research, URL:http://www.isr.umd.edu/CSHCN/.

[15] Baryshnikov Y. and Tan J. (2007) Localization for Anchoritic Sensor Networks, *In Proc. 3rd IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS '07), Santa Fe, New Mexico, USA, June 18-20, 2007.

[16] Baxter J., and Barlett P. (1999) Direct Gradient-Based Reinforcement Learning. Technical report, Research School of Information Sciences and Engineering, Australian National University.

[17] Benveniste, A., Métivier, M. and Priouret, P. (1990) *Adaptive Algorithms and Stochastic Approximation*. New York: Springer-Verlag.

[18] Bernardo J. M. and Smith A. F. M. (1994) *Bayesian Theory*. Wiley Series in Probability and Statistics. Wiley.

[19] Bergman N., Doucet A. and Gordon N., Optimal estimation and Cramér-Rao bounds for partial non-Gaussian state space models, *Annals of the Institute of Statistical Mathematics*, vol. 53, no. 1, pp. 97-112, 2001.

[20] Bertsekas D.P. (1995) *Dynamic programming and optimal control*. Belmont: Athena Scientific.

[21] Bertsekas D. (1999) *Nonlinear Programming*, 2nd Edition, Athena Scientific.

[22] Bertsekas D.P., and Tsitsiklis J.N. (1996) *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[23] Bertsekas D.P., and Tsitsiklis J.N. (2000) Gradient Convergence in Gradient Methods with Errors. *SIAM Journal on Optimization*, 10(3), 627-642.

[24] Berzuini C., Best N., Gilks W., and Larizza C. (1997) Dynamic conditional independence models and markov chain Monte Carlo methods. *Journal of the American Statistical Association* 92: 1403–1412.

[25] Blackman S. and Popoli R. (1999) *Modern Tracking Systems.* Artech House.

[26] Blom H.A.P. and Bar-Shalom Y. (1988) The interacting model algorithm for systems with Markovian switching coefficients, *IEEE Trans. Automat. Contr.*, vol. 33, pp. 780–783.

[27] Borkar V., and Meyn S.P. (2002) Risk Sensitive Optimal Control: Existence and Synthesis for Models with Unbounded Cost *Math. Oper. Res.* Vol. 1, pp. 192–209.

[28] Brehard T. and Le Cadre J.-P. (2005) Distributed Target Tracking for Nonlinear Systems: Application to Bearings-only Tracking, *In Proceedings Conference Information Fusion (FUSION'05)*, Vol. 40(2), Philadelphia, USA

[29] Briers M. (2007). *Improved Monte Carlo Methods for State-Space Models*, University of Cambridge, PhD Thesis.

[30] Cappé, O., Moulines, E. and Rydén, T. (2005) *Inference in Hidden Markov Models*. Springer.

[31] Cavazos-Cadena R. and Hernandez-Hernandez D. (2005) A Characterization of the optimal risk-Sensitive average cost in finite controlled Markov chains, *Annals of Applied Probability*, Vol. 15, No. 1A, 175-212.

[32] Cérou F., LeGland F. and Newton N. J. (2001) Stochastic particle methods for linear tangent equations. in *Optimal Control and PDE's - Innovations and Applications* (eds. J. Menaldi, E. Rofman & A. Sulem), pp. 231-240, IOS Press, Amsterdam.

[33] Coquelin P.A, Deguest R., and Munos R. (2007) Gradient Feynman-Kac flows Technical Report, INRIA Futur Lille.

[34] Del Moral P. (2004). *Feynman-Kac formulae: genealogical and interacting particlesystems with applications*. New York: Springer Verlag.

[35] Del Moral P., and Doucet A. (2004). Particle Motions in Absorbing Medium with Hard and Soft Obstacles. *Stochastic Analysis and Applications*. vol. 22, no. 5.

[36] Del Moral P., and Miclo L. (2003) Particle approximations of Lyapunov exponents connected to Schrödinger operators and Feynman-Kac semigroups *ESAIM: Probabilités et Statistique*.

[37] Del Moral P., and Lezaud P. (2006) Branching and interacting particle interpretation of rare event probabilities. S*tochastic Hybrid Systems: Theory and Safety Critical Applications*, eds. H. Blom and J. Lygeros. Springer-Verlag, Heidelberg.

[38] Delyon B. (1996) General results on the Convergence of Stochastic Algorithms, IEEE Trans. of Automatic Control, vol41, No9, p.1245-1255.

[39] Delyon B. (2000) Stochastic approximation with decreasing gain: Convergence and asymptotic theory, Technical Report, IRISA-INRIA.

[40] Demster N.P., Lair N.M., and Rubin D.B. (1977) Maximum likelihood from incomplete data via the E.M. algorithm, *Journal of the Royal statistical society of London*, Series B 39:1-38.

[41] Di Masi G. B., Stettner L. (1999) Risk sensitive control of discrete time partially observed Markov processes with infinite horizon, *Stochastics and Stochastics Rep. 67*, 309 - 322.

[42] Di Masi G. B., Stettner L. (2000) Risk sensitive control of discrete time Markov processes with infinite horizon *SIAM J. Control Optimiz*, 38, 61 - 78.

[43] Doherty L., Pister K., and Ghaoui L., (2001) Convex position estimation in wireless sensor networks. *In Proceedings of IEEE INFOCOM*.

[44] Douc, R., Cappé, O. and Moulines, E. (2005) Comparison of Resampling Schemes for Particle Filtering. *International Symposium on Image and Signal Processing and Analysis*.

[45] Douc, R. and Matias, C. (2001). Asymptotics of the Maximum Likelihood Estimator for general Hidden Markov Models. *Bernoulli*, 7 (3) 381–420.

[46] Doucet, A. (1998) On sequential simulation-based methods for Bayesian fltering. Technical Report CUED/F-INFENG/TR310, Department of Engineering, Cambridge University.

[47] Doucet A., de Freitas J.F.G. and Gordon N.J., *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.

[48] Doucet A., Godsill S.J. and Andrieu C. (2000) On sequential Monte Carlo sampling methods for Bayesian filtering, *Statist. Comput.*, vol. 10, pp.197-208 .

[49] Doucet A. , Gordon N.J. and Krishnamurthy V. (2001) "Particle filters for state estimation of jump Markov linear systems," *IEEE Trans. Signal Processing*, vol. 49, no. 3, pp. 613–624.

[50] Doucet A., and Tadić V. (2002). On-line optimization of sequential Monte Carlo methods using stochastic approximation. *Proceedings of the American Control Conference* (pp. 2565-2570).

[51] Doucet, A. and Tadić, V.B. (2003). Parameter estimation in general state-space models using particle methods. *Ann. Inst. Stat. Math.*, **55**, 409-422.

[52] Doucet A., Tadić V., and Singh S.S. (2002). Particle methods for average cost control in nonlinear non-Gaussian state-space models. Technical report, Department of Electrical and Electronic Engineering, University of Melbourne.

[53] Durbin, J. and Koopman, S. J. (2000) Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives (with discussion). *J. R. Statist. Soc.* B, **62**, 3-56.

[54] Eagle J.N. (1984) The optimal search for a moving target when the search path is constrained. *Oper. Res.*, vol. 32, pp. 1107–1115.

[55] Evans R.J. , Krishnamurthy V. and Nair G. (2001) Sensor adaptive target tracking over variable bandwidth networks, *Model Identification and Adaptive Control*, G.C. Goodwin (Ed.), Springer-Verlag, London, pp. 115-124.

[56] Fearnhead P. (2002) MCMC, sufficient statistics and particle filter, *J. Comp. Graph. Stat.*, vol. 11, pp. 848-862.

[57] Fitzgerald, W. J. (2001) Markov chain Monte Carlo methods with applications to signal processing, *Signal Processing*, 81: 3 - 18.

[58] Fleming W.H., Hernandez-Hernandez D. (1997) Risk sensitive control of finite state machines on an infinite horizon, *SIAM J. Control Optimiz.*, 35 , pp. 1790-1810.

[59] Funiak S., Guestrin C., Pashkin M., and R. Sukthankar, (2006) Distributed localization of networked cameras, *In Proc. of 5th IPSN*, Nashville, Tennessee, USA.

[60] Galstyan A., Krishnamachari B., Lerman K. and PattemS. (2004) Distributed online localization in sensor networks using a moving target, *In Proc. IPSN*, pp:61–70, Berkeley, California, 2004

[61] Gelman, A., Carlin, J. B., Stem, H. S. and Rubin, D. B. (1995) *Bayesian Data Analysis,* Chapman & Hall, London.

[62] Geweke, J. (1989) Bayesian Inference in Econometric Models Using Monte Carlo Integration, *Econometrica*, 57, 1317-1340.

[63] Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1996) *Markov Chain Monte Carlo in practice* Chapman & Hall.

[64] Glasserman P. (1990) *Gradient Estimation via Perturbation Analysis*, Kluwer Academic Publisher.

[65] Glasserman P. (2004) *Monte Carlo Methods in Financial Engineering*, Springer.

[66] Glynn, P. W. (1990) Likelihood Ratio Gradient Estimation for Stochastic Systems, *Communications of the ACM,* Vol. 33, 75-84.

[67] Glynn P. W. (1986) Stochastic approximation for Monte Carlo optimization, *Proceedings of the 18th conference on Winter simulation*, p.356-365.

[68] Glynn P.W. and Szechtman R. (2002) "Some new perspectives on the method of control variates," in *Monte Carlo and Quasi-Monte Carlo Methods 2000*, K.T. Fang, F.J. Hickernell and H. Niederreiter, Eds., Springer-Verlag, Berlin, pp. 27–49.

[69] Godsill, S. J. and Clapp, T. C. (2001) Improvement strategies for Monte Carlo particle filters, In *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N, and Gordon, N. J.) Springer-Verlag.

[70] Godsill, S. J., Doucet, A. and West, M. (2004) Monte Carlo smoothing for non-linear time series. *Journal of the American Statistical Association*, vol. 99, no. 465.

[71] Gordon, N. J., Salmond, D. J. and Smith, A. F. M. (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proc. F*, vol. 140, pp.107-113.

[72] Guyader, A., LeGland, F. and Oudjane, N. (2003) A particle implementation of the recursive MLE for partially observed diffusions. *Proceedings of the 13th IFAC Symposium on System Identification*, 1305-1310.

[73] Hastings W. K. (1970) Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika*, 52:97-109.

[74] Hauskrecht M. (2000) Value-function approximations for partially observable Markov decision processes, *Journal of Artificial Intelligence Research*, vol.13, pp. 33-94.

[75] Hernandez-Hernandez D. and Marcus S.J. (1996) Risk sensitive control of Markov processes in countable state space, *Sys. Control Letters*, 29 , pp. 147-155.

[76] Hernandez M.L. (2004) Optimal sensor trajectories in bearings-only tracking, in *Proceedings of the 7th International Conference on Information Fusion*, pp. 893-900, Stockholm, Sweden.

[77] Hernandez M.L., Kirubarajan T., Bar-Shalom, Y. (2004) Multisensor resource deployment using posterior Cramer-Rao bounds, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, April.

[78] O. Hernandez-Lerma, *Adaptive Markov Control Processes*. New York: Springer, 1989.

[79] Ihler A. T. and Fisher J. W., Moses R. L. and WillskyA. S. (2004) Nonparametric Belief Propagation for Self-Localisation in Sensor Networks, *IPSN'04: Proceedings of 3rd International Symposium on Information processing in sensor networks*, pp: 225–233, Berkeley, California.

[80] Ihler A. T., Sudderth E., Freeman W., and Willsky A. S. (2003). Efficient multiscale sampling from products of Gaussian mixtures. In *Neural Information Processing Systems*, vol. 17a.

[81] Isard M. (2003). Pampas: Real-Valued Graphical Models for Computer Vision. *In Proc. Computer Vision and Pattern Recognition*, vol. 1 613-620.

[82] Jordan M.I. (1999). *Learning in Graphical Models*, MIT Press, Cambridge, MA.

[83] Julier S. J. and Uhlmann J. K. (1997) A new extension of the Kalman filter to nonlinear systems. *In Proceedings of SPIE*.

[84] Kershaw D.J. and Evans R.J. (1994) "Optimal waveform selection for target tracking," *IEEE Trans. Inform. Theory*, vol. 40, no. 5, pp. 1536–1550.

[85] Kitagawa G. (1996) Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. Comput. Graph. Statist.*, **5**, 1-25.

[86] Klaas, M., Lang, D., Hamze, F. and de Freitas, N. (2004) Fast probability propagation: Beyond belief(s). Technical report, CS Department, University of British Columbia.

[87] Kleinman, N. L., Spall, J. C. and Naiman, D. Q. (1999) Simulation-based optimisation with stochastic approximation using common random numbers, *Management Science*, vol. 45, no. 11, pp. 1571-1578.

[88] Kloeden, P. E., Platen, E. and Schurz, H. (1997) *Numerical Solution of SDE Through Computer Experiments*. Springer-Verlag, Berlin, 1st Edition.

[89] Konda V.R., and Tsitsiklis J.N. (2003). Linear stochastic approximation driven by slowly varying Markov chains. *Systems and Control Letters*, 50(2), 95-102.

[90] Konda, V. R. and Tsitsiklis, J. N. (2004). Convergence Rate of Linear Two-Time Scale Stochastic Approximation, *Annals of Applied Probability*, Vol. 14, No. 2.

[91] Kong A. , Liu J. S. and Wong W. H. (1994) Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278– 288.

[92] Kontoyiannis, I., and Meyn S.P. (2003). Spectral Theory and Limit Theorems for Geometrically Ergodic Markov Processes. *Annals of Applied Probability*, Volume 13, pp. 304-362, 2003.

[93] Kushner, H. J. and Yin, G. (2003) *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd Edition, Springer-Verlag, New York.

[94] Langendoen K. and Reijers N. (2003) Distributed localization in wireless sensor networks: a quantitative comparison, *Computer Networks*, Vol. 43, Issue 4, pp:499-518, Nov.

[95] Lauritzen S. (1996). *Graphical Models*,Oxford Statistical Science Series.

[96] Lee H. W.J., Teo K.L. and Lim E.B., (2001) Sensor scheduling in continuous time, *Automatica*, vol. 37, pp. 2017-2023.

[97] LeGland F. and Mevel L. (1997), Asymptotic Properties of the MLE in HMM's, *Proc. of the 4th European Control Conference*, Bruxelles.

[98] F. LeGland, and L. Mevel, (1999) Recursive Identification in Hidden Markov Models *Proceedings of 36th IEEE Conference Decision and Control*, pp:3468-3473.

[99] Le Gland F. and Mevel L.(2000) Exponential forgetting and geometric ergodicity in hidden Markov models. *Mathematics of Control, Signals and Systems 13*, 63-93.

[100] LeGland F. and Oudjane N. (2004), Stability and uniform approximation of nonlinear filters using the Hilbert metric, and application to particle filters, *Ann. Appl. Probab.* 14, 144–187.

[101] Liu, J.S. (2001)*Monte Carlo Strategies in Scientific Computing*, Springer.

[102] Liu, J. S. (1996) Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statist. Comput.* 6, 113–119.

[103] Liu J.S. and Chen R. (1998) Sequential Monte Carlo methods for dynamic systems, *J. Am. Statist. Ass.*, vol. 93, pp. 1032-1044.

[104] Liu, J., Chen, R. and Logvinenko, T. (2001) A theoretical framework for sequential importance sampling and resampling. In *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N, and Gordon, N. J.) Springer-Verlag.

[105] Liu J. and West M. (2001) Combined parameter and state estimation in simulation-based filtering, In *Sequential Monte Carlo Methods in Practice* (eds Doucet A., de Freitas J.F.G. and Gordon N.J. New York: Springer-Verlag.

[106] Liu J. , Zhao F. and Petrovic D., (2003) Information directed routing in ad hoc Sensor Networks, in *Second ACM International Workshop on Wireless Sensor Networks and Applications*, San Diego, California.

[107] Ljung L. and Söderström T. (1983), *Theory and Practice of Recursive Identification*, MIT Press, Cambridge.

[108] Logothetis A. and Isaksson A. (1999) On sensor scheduling via information theoretic criteria, in *IEEE Conf. Decision Control*, pp. 2402–2406.

[109] Logothetis A., Isaksson A. and Evans R.J. (1997) An information theoretic approach to observer path design for bearings-only tracking, in *IEEE Conf. Decision Control*, pp. 3132–3137.

[110] Lovejoy W.S. (1991) A survey of algorithmic methods for partially observed Markov decision processes," *Annals Oper. Res.*, vol. 28, pp. 47–66.

[111] Mazliaka, L. (1999) Approximation of a partially observable stochastic control problem. *Markov Processes and related fields*, **5**, 477-486.

[112] Meier L. , Perschon J.and Dressler R.M. (1967) Optimal control of measurement systems, *IEEE Trans. Automat. Contr.*, vol. 12, no. 5, pp. 528–536, Oct.

[113] Mevel, L. and Finesso, L. (2000) Bayesian estimation of hidden Markov models, 14^{th} *Intern. Symposium on Mathematical Theory of Networks and Systems*, Perpignan.

[114] Meyn, S. and Tweedie R. (1993) *Markov Chains and Stochastic Stability*, Springer-Verlag.

[115] Metropolis N., Rosenbluth A. W., Rosenbluth M. N., and A. H. Teller. (1953) Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087-1092.

[116] Moallemi C.C. and Van Roy B.(2006) "Consensus propagation." *IEEE Transactions on Information Theory*, 52(11): 4753–4766.

[117] Moore D., Leonard J., Rus D., and Teller S., (2004) Robust distributed network localization with noisy range measurements," *in Second Int. Conf. on Embedded Networked Sensor Systems (SenSys 2004)*, Nov.

[118] Moses R., Krishnamurthy D., and Patterson R. (2003) A Self-Localization Method for Wireless Sensor Networks, *Eurasip Journal on Applied Signal Processing*, Special Issue on Sensor Networks, Vol. 2003, No. 4, pp. 148-158, Mar.

[119] Murphy K. P., Weiss Y., and Jordan M.(1999) Loopy-belief propagation for approximate inference: An empirical study. *In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, July.

[120] Musso, C., Oudjane, N. and Le Gland, F. (2001) Improving Regularised Particle Filters. In *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N, and Gordon, N. J.) Springer-Verlag.

[121] Nardone, S.C. and Aidala, V.J. (1981) Observability Criteria for Bearings-Only Target Motion Analysis", *IEEE Transactions on Aerospace and Electronic Systems*, Volume AES-17, Issue 2, pp. 162 - 166.

[122] Niculescu D. and Nath B., (2001) Ad hoc positioning system (aps), *IEEE GLOBECOM '01*, vol. 5, pp. 2926–2931.

[123] Nowak R. , (2003) Distributed EM Algorithms for Density Estimation and Clustering in Sensor Networks, *IEEE Transaction of Signal Processing in Networking*, August.

[124] Okello N. N. , and Challa S. (2003) Joint Sensor Registration for Distributed Trackers, *IEEE Transactions on Aerospace and Electronic Systems*, March.

[125] Okello N. N., and Ristic B. (2003) Maximum Likelihood Registration for Dissimilar Sensors, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 39(3), pp:1074–1083, July.

[126] Olfati-Saber R. (2005) Distributed Kalman Filter with Embedded Consensus Filters, *In Proc. IEEE Conf. on Decision and Control*.

[127] Paris S. , Le Cadre J.-P. (2002) Planification for terrain-aided navigation, in *Proceedings of the 5th International Conference on Information Fusion*, pp. 1007-1014, Annapolis, Maryland.

[128] Paskin M. and Guestrin C. (2003) A robust architecture for distributed inference in sensor networks. Technical Report IRBTR-03-039, Intel Research.

[129] Patwari N. ,Ash J. , Kyperountas S., Hero A.O., Moses R.L., and Correal N.S. (2005) Locating the nodes, cooperative localisation in wireless sensor networks, *IEEE Signal Processing,* Vol. 54, July.

[130] Pearl J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan-Kauffman.

[131] Pflug G.Ch. (1996) *Optimization of stochastic models: the interface between simulation and optimization*. Boston: Kluwer.

[132] Pitt M.K. and Shephard N. (1999). Filtering via simulation: auxiliary particle filter. *J. Amer. Statist. Assoc*. 94 590-599.

[133] Poyiadjis G. , Doucet A. and Singh S.S., (2005) Particle Methods for Optimal Filter Derivative: Application to Parameter Estimation, *Proceedings IEEE ICASSP*.

[134] Poyadjis G., Doucet A. and Singh S.S., (2005) Maximum Likelihood Parameter Estimation using Particle Methods, *In Proceedings of the American Statistical Association*, 2005.

[135] Poyiadjis G., Singh S.S., and Doucet A. (2006). Particle filter as a controlled Markov chain for on-line parameter estimation in General State Space Models. *Proceedings ICASSP*, 2006.

[136] Poyiadjis G. (2006) Parameter estimation in General State Space Models. *PhD Thesis*. University of Cambridge.

[137] Priyantha N. , Balakrishnan H., Demaine E., and Teller S. (2005) Mobile-Assisted Localization in Wireless Sensor Networks", *IEEE INFOCOM*, Miami, FL, March.

[138] Rabiner L.R. (1989) A tutorial on hidden Markov models and selected applications inspeech recognition, *Proceedings of the IEEE*, Vol. 77, Issue 2, pp 257 – 286.

[139] Rauch H. E.,Tung F. and Striebel C. T. (1965) Maximum likelihood estimates of linear dynamic systems, *J. Amer. Inst Aeronautics and Astronautics,* 3 (8), pp 1445–1450.

[140] Robert, C. P. and Casella, G. (2004) *Monte Carlo Statistical Methods*. 2nd Ed., Springer.

[141] Rydén T. (1997) On recursive estimation for hidden Markov models. *Stochastic Processes and their Applications* 66, 79-96.

[142] Salmond, D. and Gordon, N.J. (2001) Particles and mixtures for tracking and guidance. In *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N, and Gordon, N. J.) Springer-Verlag.

[143] Savarese C. and Rabaey J. (2002) Robust positioning algorithms for distributed ad-hoc wireless sensor networks, "*Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pp. 317–327.

[144] Savvides A., Park H., and Srivastava M. B., (2002) The bits and flops of the n-hop multilateration primitive for node localization problems, *International Workshopon Sensor Networks Application*, pp. 112– 121.

[145] Shiryaev A. N. (1995) *Probability*. Graduate Texts in Mathematics, Number 95. Springer Verlag, New York.

[146] Sichitiu M.L. and Ramadurai V., (2003) Localization of wireless sensor networks with a mobile beacon, Center for Advanced Computing and Communications (CACC), Raleigh, NC, Tech. Rep. TR-03/06, July .

[147] Silverman. S. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

[148] Singh S. S., Kantas N., Vo B.,Doucet A. and Evans R. (2005) On the convergence of a stochastic optimisation algorithm for optimal observer trajectory planning, Technical Report CUED/F-INFENG/TR 522, University of Cambridge, Department of Engineering. URL: www-sigproc.eng.cam.ac.uk/˜nk234/.

[149] Singh S.S. , Vo B., Doucet A. and Evans R. (2004) Variance Reduction for Monte Carlo Implementation of Adaptive Sensor Management, In *Proceedings of the 7th International Conference on Information Fusion*, pp. 901-908, Stockholm, Sweden.

[150] Singh S.S., Vo B., Doucet A. and Evans R.J., (2003) Stochastic approximation for optimal observer trajectory planning in bearings-only tracking," *IEEE Conf. Decision Control*.

[151] Smallwood R.D. and Sondik E.J., (1973) The optimal control of partially observable Markov processes over a finite horizon, *Oper. Res.*, vol. 21, pp. 1071–1088.

[152] Spall J. C. (2000), Adaptive stochastic approximation by the simultaneous perturbation method,*IEEE Trans. Autom. Contr.,* vol. 45, pp. 1839–1853.

[153] Spanos D., Olfati-Saber R. and Murray R. M. (2005) Distributed sensor fusion using dynamic consensus. *In Proc. IFAC*.

[154] Storvik G. (2002), Particle filters in state space models with the presence of unknown static parameters, *IEEE. Trans. Signal Processing*, vol. 50, pp. 281–289.

[155] Sudderth E., Ihler A., Freeman W., and Willsky A. (2002). Nonparametric Belief Propagation. MIT LIDS Technical Report 2551.

[156] Tadic, V. (2000) Asymptotic analysis of stochastic approximation algorithms under violated Kushner-Clark conditions with applications, *IEEE Conference on Decision and Control*, Sydney, Australia.

[157] Tadić V.B. and Doucet A. (2005) Exponential forgetting and geometric ergodicity for optimal filtering in general state-space models. *Stochastic Processes and Their Applications*, vol. 115, pp. 1408-1436.

[158] Tatikonda S. and Jordan M. I. (2002) Loopy belief propagation and Gibbs measures. *Uncertainty in Artificial Intelligence,* D. Koller and A. Darwiche, editors.

[159] Taylor C., Rahimi A., Bachrach J., Shrobe H., and. Grue A, (2006) Simultaneous localization, calibration, and tracking in an ad hoc sensor network, *In Proc. IPSN*, Nashville, Tennessee.

[160] Thrun S. (2000) Monte Carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, 1064-1070. MIT Press.

[161] Tichavsky P., Muravchik C.H. and Nehorai A.,(1998) "Posterior Cramer-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1386–1396.

[162] Tierney L. (1994) Markov Chains for exploring posterior distributions, *The Annals of Statistics*, 22:1701-1762.

[163] Tremois O. and LeCadre J.-P. (1999) Optimal observer trajectory in bearings-only tracking for manoeuvring sources, *IEE Proc. Radar, Sonar Navig.*, vol. 146, no. 1, pp. 31–39, Feb.

[164] Van der Merwe R., de Freitas N., Doucet A. and Wan E. (2001) The Unscented Particel Filter. In *Advances in Neural Information Processing Systems* 13.

[165] Vermaak J., Maskell S. and Briers M.,(2006) Online Sensor Registration, *IEEE Aerospace Conference*, Big Sky, MT.

[166] Wainwright M. J. and Jordan M. I. (2005) A variational principle for graphical models. *New Directions in Statistical Signal Processing: From Systems to Brain*. Cambridge, MA: MIT Press.

[167] Weiss Y. (2000) Correctness of Local Probability Propagation in Graphical Models with Loops, *Neural Computation*, Vol. 12 pp:1–41.

[168] Weiss Y. and Freeman W.T. (2001) Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology, *Neural Computation*, Vol. 13 pp:2173–2200.

[169] Whittle, P. (1990) *Risk-Sensitive Optimal Control.* John Wiley and Sons Ltd.

[170] Xiao L., Boyd S. and Lall S., (2005) A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus, " *IPSN'05*, April, pp:63-70.

[171] Yedidia, J.S., Freeman, W.T. and Weiss, Y., (2005) Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms, *IEEE Transactions on Information Theory*, Vol. 51, Issue 7, pp. 2282-2312, July.

[172] Zhao F., Shin J. and Reich J. (2002) Information-driven dynamic sensor collaboration for tracking applications. " *IEEE Signal Processing Magazine*, Vol. 19(2):61–72, March.