

# Solving Nonlinear State Estimation Problems Using Particle Filters - An Engineering Perspective

Thomas B. Schön

Division of Automatic Control

E-mail: [schon@isy.liu.se](mailto:schon@isy.liu.se)

10th May 2010

Report no.: LiTH-ISY-R-2953

Address:

Department of Electrical Engineering

Linköpings universitet

SE-581 83 Linköping, Sweden

WWW: <http://www.control.isy.liu.se>

AUTOMATIC CONTROL  
REGLERTEKNIK  
LINKÖPINGS UNIVERSITET



## **Abstract**

The goal of this work is to provide a self contained introduction to the nonlinear state estimation problem and how this can be solved using the particle filter. The objective in nonlinear filtering is to recursively in time estimate the state in a dynamic model. This is a real-time signal processing problem, where new measurement information inherently arrives in a sequential fashion. The particle filter offers a powerful numerical solution to this problem. Besides a theoretical derivation of the particle filter we also provide MATLAB code solving simple examples.

**Keywords:** Nonlinear state estimation, particle filter, nonlinear filtering, dynamic systems

Solving Nonlinear State Estimation Problems  
Using Particle Filters - An Engineering  
Perspective

Thomas B. Schön  
Division of Automatic Control  
Linköping University  
SE-58183 Linköping, Sweden.  
E-mail: [schon@isy.liu.se](mailto:schon@isy.liu.se)

December 9, 2008



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>State Estimation</b>	<b>7</b>
2.1	State-Space Models . . . . .	8
2.2	Conceptual Solution . . . . .	9
2.3	An Important Special Case . . . . .	10
2.4	Local Approximations . . . . .	11
2.5	Point Estimates . . . . .	12
<b>3</b>	<b>Particle Filters</b>	<b>15</b>
3.1	Random Number Generation . . . . .	16
3.1.1	Perfect Sampling . . . . .	16
3.1.2	Importance Sampling . . . . .	18
3.1.3	Sampling Importance Resampling . . . . .	20
3.2	A First Particle Filter . . . . .	21
3.3	A General Particle Filter . . . . .	25
3.3.1	Resampling Algorithms . . . . .	28
3.3.2	Computing the Estimate . . . . .	30
3.3.3	Implementation . . . . .	31
<b>A</b>	<b>Useful Facts from Probability Theory</b>	<b>35</b>



# Chapter 1

## Introduction

The goal of this work is to provide a self contained introduction to the nonlinear state estimation problem and how this can be solved using the particle filter. The objective in nonlinear filtering is to recursively in time estimate the state  $x_t$  in the dynamic model,

$$x_{t+1} = f_t(x_t) + v_t, \quad (1.1a)$$

$$y_t = h_t(x_t) + e_t. \quad (1.1b)$$

This is a real-time signal processing problem, where new measurement information inherently arrives in a sequential fashion.

The main ideas underlying the particle filter were conceived already at the end of the 1940's. However, it was not until 1993 that the first working particle filter was discovered by Gordon *et al.* by introducing the final missing component, the so called resampling step. There are still major ongoing research efforts all over the world when it comes to the particle filter, both in terms of theory and in terms of practical applications.

In Chapter 2 the nonlinear state estimation problem is mathematically defined. This is achieved by introducing the nonlinear state-space models and deriving the conceptual solution to the nonlinear state estimation problem. The particle filter is introduced in Chapter 3. Since the particle filter is really all about random numbers we will start by describing the underlying mechanisms for generating random numbers. We will then move on by adding the concept of time and dynamic models from Chapter 2 to the random number generators. The result is the particle filter. An important part of Chapter 2 is that we also provide MATLAB code for the particle filter, implying that you can implement the filter within five minutes. Finally, Appendix A provides some useful facts from probability theory.



## Chapter 2

# State Estimation

The aim of this chapter is to mathematically define the nonlinear state estimation problem we are considering in this work. The probability density function  $p(x_t|y_{1:s})$  contains all statistical information available about the state variable  $x_t$ , based on the information in the measurements  $y_{1:s}$ . Here,

$$y_{1:s} = \{y_1, y_2, \dots, y_s\}, \quad (2.1)$$

that is the set of all the measurements up to and including time  $s$ . Depending on the relation between  $t$  and  $s$  in  $p(x_t|y_{1:s})$  three different estimation problems are obtained

- The *filtering* problem,  $t = s$ .
- The *prediction* problem,  $t > s$ .
- The *smoothing* problem,  $t < s$ .

This chapter will illustrate how the expressive power of the probability density functions can be used to handle the filtering problem. The filtering problem will be considered solved when we have access to an approximation of the filtering density function  $p(x_t|y_{1:t})$ . Based on this approximation we can then form various state estimates according to

$$I(g(x_t)) \triangleq E_{p(x_t|y_{1:t})} \{g(x_t)\} = \int_{\mathbb{R}^{n_x}} g(x_t) p(x_t|y_{1:t}) dx_t. \quad (2.2)$$

Typically  $g(x_t) = x_t$  is used in the above equations, providing us with an estimate of the state itself. The objective in state estimation is to recursively in time estimate the state in the dynamic model. The dynamic models we are working with are modelled using state-space models, motivating Section 2.1 providing a brief introduction to state-space models. In Section 2.2 we will derive the necessary expressions for the probability density functions that are needed in order to solve the nonlinear filtering problem. Due to its importance, the relation to the Kalman filter is given in Section 2.3. Finally, it is explained how to obtain point estimates using the density functions derived in Section 2.2.

## 2.1 State-Space Models

The dynamical models that we work with are formulated using state-space models. A rather general state-space model is given by

$$x_{t+1} = f_t(x_t, v_t, \theta), \quad (2.3a)$$

$$y_t = h_t(x_t, e_t, \theta), \quad (2.3b)$$

where  $x_t \in \mathbb{R}^{n_x}$  denotes the state,  $y_t \in \mathbb{R}^{n_y}$  denotes the measurement,  $\theta$  denotes possibly unknown parameters,  $v_t$  and  $e_t$  denote the stochastic process and measurement noise, respectively. Furthermore, the dynamic equations for the process (typically referred to as the *process model*) are denoted by  $f_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$  and the equations modelling the sensors (typically referred to as the *measurement model*) are denoted by  $h_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_e} \rightarrow \mathbb{R}^{n_y}$ . Note that any deterministic input signal  $u_t$  is subsumed in the time-varying dynamics. Since we are not concerned with parameter estimation in the present work, we assume that all parameters are known and hence  $\theta$  will not be present in the expressions that follow. Most applied signal processing problems are formulated using a special case of (2.3), where the noise processes are assumed to enter additively according to the formulation given in Model 2.1.

---

### **Model 2.1 (Nonlinear state-space model with additive noise)**

---

*The nonlinear discrete-time state-space model with additive noise is given by*

$$x_{t+1} = f_t(x_t) + v_t, \quad (2.4a)$$

$$y_t = h_t(x_t) + e_t, \quad (2.4b)$$

*with  $v_t$  and  $e_t$  independent and identically distributed i.i.d..*

---

An alternative and useful formulation of the state-space model is the following

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (2.5a)$$

$$y_t \sim p(y_t|x_t). \quad (2.5b)$$

In this formulation we state the model in terms of probability density functions, which describe both the dynamics and the measurement relations. In order to formulate Model 2.1 in the form (2.5) we make the following observations,

$$p(x_{t+1}|x_t) = p_{v_t}(x_{t+1} - f_t(x_t)), \quad (2.6a)$$

$$p(y_t|x_t) = p_{e_t}(y_t - h_t(x_t)). \quad (2.6b)$$

We will in this work only make use of probability density functions  $p(x)$  and not the corresponding probability measures  $p(dx)$ . The reason is to keep things as simple as possible and the use of measure theory is only needed when details on convergence are discussed and such details will not be covered in this introduction.

The most common special case of Model 2.1 is the linear (the functions  $f$  and  $h$  are linear) state-space model, subject to Gaussian noise.

**Model 2.2 (Linear state-space model with Gaussian noise)**

The discrete-time linear state-space model, subject to Gaussian noise is given by

$$x_{t+1} = A_t x_t + w_t, \quad w_t \sim \mathcal{N}(0, Q_t), \quad (2.7a)$$

$$y_t = C_t x_t + e_t, \quad e_t \sim \mathcal{N}(0, R_t). \quad (2.7b)$$

The theory concerning linear state-space models is by now rather mature and there are numerous text books available on the subject.

## 2.2 Conceptual Solution

This section is concerned with the problem of deriving expressions for the probability density functions that are needed for solving the nonlinear filtering problem. We will start by assuming that the model is given in the form (2.5), repeated here for convenience

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (2.8a)$$

$$y_t \sim p(y_t|x_t). \quad (2.8b)$$

Recall the according to (2.6) a model in the form (2.4) can straightforwardly be converted into the form (2.8) above. In the development that follows Bayes' theorem and the Markov property will be instrumental (See Appendix A for details).

Consider the filtering density

$$p(x_t|y_{1:t}) = p(x_t|y_t, y_{1:t-1}), \quad (2.9)$$

which using Bayes' theorem and the Markov property can be rewritten as

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \quad (2.10)$$

In order to handle the denominator in the above equations it has to be expressed using known densities. This can be accomplished by marginalizing the following equation with respect to  $x_t$

$$p(y_t, x_t|y_{1:t-1}) = p(y_t|x_t)p(x_t|y_{1:t-1}), \quad (2.11)$$

which corresponds to integrating (2.11) with respect to  $x_t$ , resulting in

$$p(y_t|y_{1:t-1}) = \int_{\mathbb{R}^{n_x}} p(y_t|x_t)p(x_t|y_{1:t-1})dx_t. \quad (2.12)$$

Furthermore, in order to derive an expression for the one step ahead prediction density  $p(x_{t+1}|y_{1:t})$  we employ the marginalization trick once more by integrating the following equations with respect to  $x_t$ ,

$$p(x_{t+1}, x_t|y_{1:t}) = p(x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t}) = p(x_{t+1}|x_t)p(x_t|y_{1:t}), \quad (2.13)$$

resulting in the following expression

$$p(x_{t+1}|y_{1:t}) = \int_{\mathbb{R}^{n_x}} p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t. \quad (2.14)$$

This equation is sometimes referred to as the Chapman–Kolmogorov equation. These expressions are important, hence we summarize them in the following theorem.

**Theorem 1** *If the dynamic model is given by*

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (2.15a)$$

$$y_t \sim p(y_t|x_t), \quad (2.15b)$$

*the filtering density  $p(x_t|y_{1:t})$  and the one step ahead prediction density  $p(x_{t+1}|y_{1:t})$  are given by*

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \quad (2.16a)$$

$$p(x_{t+1}|y_{1:t}) = \int_{\mathbb{R}^{n_x}} p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t, \quad (2.16b)$$

where

$$p(y_t|y_{1:t-1}) = \int_{\mathbb{R}^{n_x}} p(y_t|x_t)p(x_t|y_{1:t-1})dx_t. \quad (2.16c)$$

Given the complexity of the problem it is actually quite remarkable that we are able to derive a result as the one given in Theorem 1 above. However, there is a severe problem with this solution, the multidimensional integrals involved only permit an analytical solution in a few special cases. The most important special case is when the dynamic model is linear and the involved stochastic variables are normal, Theorem 1 is then reduced to the Kalman filter (see the subsequent section). This is due to the fact that the mathematics involved is tractable, but most importantly it hinges on the fact that there are a vast amount of real world applications where this special case has been successfully applied. However, most applications would perform better if the nonlinear estimation problem could be properly solved. More importantly this would also allow us to tackle more complicated applications, which do not lend themselves to linear algorithms. The particle filter has proved to be a very successful technique for solving the nonlinear state estimation problem.

## 2.3 An Important Special Case

An important property of the linear state-space model, see Model 2.2, is that all density functions involved are Gaussian. This is due to the fact that a linear transformation of a Gaussian random variable will result in a new Gaussian random variable. Another important fact is that a Gaussian density function is completely parameterized by two parameters, the first and second order moments, i.e., the mean and the covariance. This implies that if it is assumed that the underlying model is given by (2.7) the recursions in Theorem 1 can

be recast as recursive relations for the mean values and the covariances of the involved probability density functions. This leads to an important corollary to Theorem 1, the Kalman filter. Before stating the corollary the notation  $\hat{x}_{t|s}$  is introduced, which denotes the estimate of the state  $x$  at time  $t$  using the information available in the measurements up to and including time  $s$ . In other words,  $\hat{x}_{t|s} = \mathbb{E}_{p(x_t|y_{1:s})} \{x_t\}$ .

**Corollary 1 (Kalman filter)** *Consider (2.7) and assume that the initial state is distributed as  $x_0 \sim \mathcal{N}(\bar{x}_0, \bar{P}_0)$ . Then, the estimates for the filtering density function and the one step ahead prediction density function are both normal, according to*

$$\hat{p}(x_t|y_{1:t}) = \mathcal{N}(x; \hat{x}_{t|t}, P_{t|t}), \quad (2.17a)$$

$$\hat{p}(x_{t+1}|y_{1:t}) = \mathcal{N}(x; \hat{x}_{t+1|t}, P_{t+1|t}), \quad (2.17b)$$

where

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - C_t\hat{x}_{t|t-1}), \quad (2.18a)$$

$$P_{t|t} = P_{t|t-1} - K_t C_t P_{t|t-1}, \quad (2.18b)$$

$$\hat{x}_{t+1|t} = A_t \hat{x}_{t|t}, \quad (2.18c)$$

$$P_{t+1|t} = A_t P_{t|t} A_t^T + Q_t, \quad (2.18d)$$

$$K_t = P_{t|t-1} C_t^T (C_t P_{t|t-1} C_t^T + R_t)^{-1}, \quad (2.18e)$$

with initial values  $\hat{x}_{0|-1} = \bar{x}_0$  and  $P_{0|-1} = \bar{P}_0$ .

There are many different ways in which this result can be proved. A good way to practise the theory introduced above is to derive Corollary 1 based on Theorem 1 and Model 2.2. Theorem 3 in Appendix A is useful.

## 2.4 Local Approximations

The idea employed in local methods is to approximate the nonlinear model by a linear, Gaussian model. This model is only valid locally, but the Kalman filter can readily be applied. The first approach along those lines was to linearize the model along a nominal trajectory, resulting in the *linearized Kalman filter*. An improvement to this was suggested by S. F. Schmidt *et al.* They suggested that the linearization should be performed around the current estimate, rather than around a nominal trajectory. The result is the *extended Kalman filter* (or perhaps more appropriately the Schmidt EKF). More specifically, the local approximation is obtained by linearizing the nonlinear model (2.4) by applying a first-order Taylor expansion around the current estimate,

$$f(x_t, t) \approx f(\hat{x}_{t|t}, t) + \left. \frac{\partial f(x, t)}{\partial x} \right|_{x=\hat{x}_{t|t}} (x_t - \hat{x}_{t|t}), \quad (2.19a)$$

$$h(x_t, t) \approx h(\hat{x}_{t|t-1}, t) + \left. \frac{\partial h(x, t)}{\partial x} \right|_{x=\hat{x}_{t|t-1}} (x_t - \hat{x}_{t|t-1}). \quad (2.19b)$$

Using this approximation in (2.4) gives

$$x_{t+1} = f(\hat{x}_{t|t}, t) - F_t \hat{x}_{t|t} + F_t x_t + w_t, \quad (2.20a)$$

$$y_t = h(\hat{x}_{t|t-1}, t) - H_t \hat{x}_{t|t-1} + H_t x_t + e_t, \quad (2.20b)$$

where

$$F_t \triangleq \left. \frac{\partial f(x, t)}{\partial x} \right|_{x=\hat{x}_{t|t}}, \quad H_t \triangleq \left. \frac{\partial h(x, t)}{\partial x} \right|_{x=\hat{x}_{t|t-1}}. \quad (2.21)$$

The approximate model given in (2.20) is a linear, Gaussian model in  $x_t$ , which implies that the Kalman filter given in Corollary 1 can be applied. The result is the extended Kalman filter, given in Algorithm 2.1.

---

**Algorithm 2.1 (Extended Kalman Filter (EKF))**

---

Consider Model 2.1. An approximate sub-optimal estimate for the filter density function  $p(x_t|Y_t)$ , obtained by linearization, is recursively given according to

$$\hat{p}(x_t|Y_t) = \mathcal{N}(x; \hat{x}_{t|t}, P_{t|t}), \quad (2.22a)$$

$$\hat{p}(x_{t+1}|Y_t) = \mathcal{N}(x; \hat{x}_{t+1|t}, P_{t+1|t}), \quad (2.22b)$$

where

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (y_t - h(\hat{x}_{t|t-1}, t)), \quad (2.23a)$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1}, \quad (2.23b)$$

$$\hat{x}_{t+1|t} = f(\hat{x}_{t|t}, t), \quad (2.23c)$$

$$P_{t+1|t} = F_t P_{t|t} F_t^T + Q_t, \quad (2.23d)$$

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}, \quad (2.23e)$$

with initial values  $\hat{x}_{1|0} = \bar{x}_1$  and  $P_{1|0} = \bar{\Pi}_1$ . Furthermore,  $F_t$  and  $H_t$  are defined by

$$F_t = \left. \frac{\partial f(x, t)}{\partial x} \right|_{x_t=\hat{x}_{t|t}}, \quad H_t = \left. \frac{\partial h(x, t)}{\partial x} \right|_{x_t=\hat{x}_{t|t-1}} \quad (2.24)$$


---

One of the problems inherent in the EKF is that it might diverge. The literature contains several more or less *ad hoc* methods trying to counteract this phenomenon and to further enhance the general performance of the EKF. To mention a few examples we have, the *iterated EKF* and higher-order Taylor expansions.

## 2.5 Point Estimates

The task of finding a point estimate can, in abstract terms, be cast as a problem of finding a transformation  $m_t$ , which makes use of the information in the measurements and the known input signals to produce estimates of the states of interest.

$$m_t : u_{1:t} \times y_{1:t} \rightarrow \hat{x}_t \quad (2.25)$$

All information available in the measurements has been processed and inferred into the density function  $p(x_t|y_{1:t})$ . This density function can then be used to derive various point estimates, which is normally what the user would expect from the estimation algorithm. Typically, the application does not need the

entire probability density function. Instead it is sufficient to know how the values of the various states evolve over time and a quality assessment of these values. It is reasonable to claim that an estimate is useless, if we do not know how good it is. Since a probabilistic framework is employed, this opens up for using the tools available in probability theory and statistics for assessing the quality of estimates, such as covariances, confidence regions, tests, etc.

This section is concerned with some of the most common mappings (2.25) present in the literature. Most of the estimates are indeed based on approximations of the probability density functions  $p(x_t|y_{1:t})$ , but the estimates can also be based on deterministic considerations.

From a probabilistic point of view a rather appealing point estimate is provided by choosing the value that minimizes the variance of the estimation error, referred to as the *minimum variance* (MV) estimate

$$\hat{x}_t^{MV} \triangleq \arg \min_{\hat{x}_t} \mathbb{E} \left\{ \|x_t - \hat{x}_t\|^2 \mid y_{1:t} \right\} \quad (2.26)$$

where  $\|x_t\|^2 = x_t^T x_t$ . It is in fact possible to derive an explicit expression for this estimate.

$$\begin{aligned} \mathbb{E} \left\{ \|\hat{x}_t - x_t\|^2 \mid y_{1:t} \right\} &= \mathbb{E} \left\{ (x_t - \hat{x}_t)^T (x_t - \hat{x}_t) \mid y_{1:t} \right\} \\ &= \mathbb{E} \left\{ x_t^T x_t \mid y_{1:t} \right\} - 2\hat{x}_t^T \mathbb{E} \left\{ x_t \mid y_{1:t} \right\} + \hat{x}_t^T \hat{x}_t \\ &= \|\hat{x}_t - \mathbb{E} \left\{ x_t \mid y_{1:t} \right\}\|^2 + \mathbb{E} \left\{ \|x_t\|^2 \mid y_{1:t} \right\} - \|\mathbb{E} \left\{ x_t \mid y_{1:t} \right\}\|^2 \end{aligned} \quad (2.27)$$

The two last terms in (2.27) are independent of  $\hat{x}_t$  and hence (2.27) is minimized by

$$\hat{x}_t^{MV} = \mathbb{E} \left\{ x_t \mid y_{1:t} \right\} = \int x_t p(x_t | y_{1:t}) dx_t. \quad (2.28)$$

The above calculation explains the name, *minimum mean square error* (MMSE), which is commonly used as an alternative name for the estimate (2.28).

Another point estimate which suggests itself, within the probabilistic framework, is the point where the probability is most dense.

$$\hat{x}_t^{MAP} \triangleq \arg \max_{x_t} p(x_t | y_{1:t}) = \arg \max_{x_t} p(y_t | x_t) p(x_t | y_{1:t-1}), \quad (2.29)$$

which is referred to as the *maximum a posteriori* (MAP) estimate. In the second equality of (2.29) Bayes' theorem and the Markov property are employed, together with the fact that the maximization is performed over  $x_t$ .



## Chapter 3

# Particle Filters

The *key idea* underlying the particle filter is to approximate the filtering density function as a weighted set of samples

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - x_t^i), \quad \sum_{i=1}^N w_t^i = 1, \quad w_t^i \geq 0, \quad \forall i, \quad (3.1)$$

where  $\delta(\cdot)$  is the Dirac delta<sup>1</sup> and  $w_t^i$  denotes the weights associated with the sample  $x_t^i$ . Here it is worth noting that the samples  $x_t^i$  are also referred to as particles, which explains the name *particle* filter. The approximation (3.1) allows the otherwise intractable integrals in (2.2) and (2.16) to be transformed into tractable sums. For example, if we use (3.1) in order to evaluate (2.2) with  $g(x_t) = x_t$  and  $s = t$  we straightforwardly have the following approximation

$$\hat{I}_N(x_t) \approx \int_{\mathbb{R}^{n_x}} x_t \hat{p}_N(x_t|y_{1:t}) dx_t = \int_{\mathbb{R}^{n_x}} x_t \sum_{i=1}^N w_t^i \delta(x_t - x_t^i) dx_t = \sum_{i=1}^N w_t^i x_t^i. \quad (3.3)$$

The representation (3.1) is fundamentally different from the representation used in for instance the Kalman filter, where a specific functional form of the density function is assumed (Gaussian). There the estimate is then represented by the parameters (the mean and the covariance) parameterizing this density. In the particle filter the filtering density is represented as a set of random samples approximately distributed according to this density.

The particle filter is really all about random numbers, motivating Section 3.1, where we introduce the concept of importance sampling. In Section 3.2 we will

---

<sup>1</sup>The Dirac delta is not a function, but it can formally be defined as a distribution. However, for many purposes it can be manipulated as if it is a function. Informally, the Dirac delta  $\delta(x - a)$  is a function that has zero value everywhere, except at  $x = a$  where it is infinitely large, such that its total integral is 1. The most important property of the Dirac delta is the so called sifting property,

$$\int_{-\infty}^{\infty} f(x) \delta(x - a) dx = f(a) \quad (3.2)$$

for a test function  $f$ . The property will be heavily used in the discussion to come in order to form the actual state estimates based on the particle approximations.

the add the concept of dynamic models and time to the random number generators, providing us with a first particle filter. There are lots of embellishments available for this first particle filter, some of which are introduced in Section 3.3, where a more general particle filter is introduced.

### 3.1 Random Number Generation

The particle filter represents the filtering density function using a large number of samples approximately drawn from the true filtering PDF. Hence, the ability to generate random numbers distributed according to nonstandard probability density functions is essential in order to devise a particle filter. Before we start our brief exposition on random number generation it is worth noting that throughout this section there will be no notion of time, since that would only complicate things without adding any additional understanding.

Here, the problem under consideration is how to generate samples from some known probability density function, referred to as the *target density*  $t(x)$ . If the target density is a standard PDF, for example Gaussian, it is by now well-known how to generate these random numbers. This leads to perfect samples, in the sense that the samples are drawn exactly from the target density. This case is further discussed in Section 3.1.1 below. However, in most problems we typically cannot generate samples from  $t(x)$  directly. This calls for an alternative solution and the idea is then to employ an alternate density that is simple to draw samples from, referred to as the *proposal density*  $q(x)$ . The only restriction imposed on  $q(x)$  is that its support should include the support of  $t(x)$ <sup>2</sup>. When a sample  $\tilde{x} \sim q(x)$  is drawn the probability that it was in fact generated from the target density can be calculated. This probability can then be used to decide whether  $\tilde{x}$  should be considered to be a sample from  $t(x)$  or not. This probability is referred to as the *acceptance probability*, and it is typically expressed as a function of  $w(\tilde{x})$ , defined by the following relationship,

$$t(\tilde{x}) \propto w(\tilde{x})q(\tilde{x}), \quad (3.4)$$

or equivalently

$$w(\tilde{x}) \propto \frac{\text{target density}}{\text{proposal density}} \quad (3.5)$$

Depending on the exact details of how the acceptance probability is computed different methods are obtained. The three most common methods used to derive particle filters are, importance sampling, acceptance-rejection sampling and Metropolis-Hastings independence sampling. The most well-used strategy is to make use of the importance sampling idea and that will be done here as well. However, we will first discuss perfect sampling.

#### 3.1.1 Perfect Sampling

This section is concerned with the problem of calculating estimates (2.2) based on the assumption that we have access to  $N$  i.i.d. samples,  $\{x^i\}_{i=1}^N$  from the target density  $t(x)$ . This assumption is unrealistic from a practical point of

---

<sup>2</sup>The support of  $q(x)$  includes the support of  $t(x)$  if  $\forall x \in \mathbb{R}^{n_x}, t(x) > 0 \Rightarrow q(x) > 0$ .

view. Nevertheless, it will allow us to illustrate the key idea underlying the particle filter. Using the samples  $\{x^i\}_{i=1}^N$  an empirical estimate of the density function  $t(x)$  can be formed according to

$$\hat{t}_N(x) = \sum_{i=1}^N \frac{1}{N} \delta(x - x^i). \quad (3.6)$$

Using this empirical density an estimate of  $I(g(x))$ , defined in (2.2), is obtained as

$$\hat{I}_N(g(x)) = \int g(x) \hat{t}_N(x) dx = \sum_{i=1}^N \frac{1}{N} g(x^i). \quad (3.7)$$

This estimate is unbiased and according to the strong law of large numbers we have that

$$\lim_{N \rightarrow \infty} \hat{I}_N(g(x)) \xrightarrow{\text{a.s.}} I(g(x)), \quad (3.8)$$

where  $\xrightarrow{\text{a.s.}}$  denotes almost sure (a.s.) convergence. If we assume that  $\sigma^2 = I(g^2(x)) - I^2(g(x)) < \infty$  the central limit theorem can be applied, which gives

$$\lim_{N \rightarrow \infty} \frac{\sqrt{N}}{\sigma} (\hat{I}_N(g(x)) - I(g(x))) \xrightarrow{d} \mathcal{N}(0, 1), \quad (3.9)$$

where  $\xrightarrow{d}$  denotes convergence in distribution. Hence, using a large number of samples  $\{x^i\}_{i=1}^N$  we can easily estimate any quantity  $I(g(x))$ , according to (3.7). Methods of this type are typically referred to as Monte Carlo methods. In Example 3.1 we will illustrate perfect sampling from a Gaussian mixture.

**Example 3.1 (Perfect Sampling from a Gaussian mixture)**

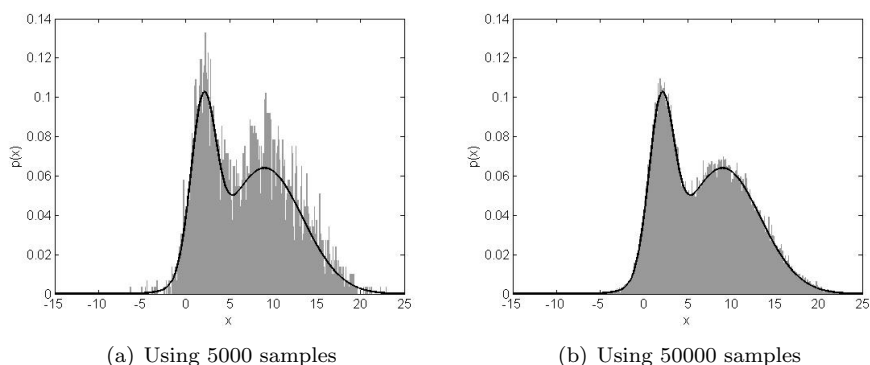
This example will illustrate perfect sampling of the following Gaussian mixture

$$t(x) = 0.3\mathcal{N}(x; 2, 2) + 0.7\mathcal{N}(x; 9, 19). \quad (3.10)$$

Since the target PDF  $t(x)$  given in (3.10) is made up of normal PDF's we can generate samples from this using standard functions. That is, we can generate realizations from the stochastic variable with the density function given in (3.10). Using these realizations an empirical estimate  $\hat{t}_N(x)$  of the true PDF can then be formed according to (3.6). The result is given in Figure 3.1 using  $N = 5000$  and  $N = 50000$  samples.

The fact that the quality of the empirical density improves as the number of samples increase should come as no surprise. The example gets more interesting as we change to importance sampling in the subsequent section.

The assumption underlying the above discussion is that it is possible to obtain i.i.d. samples from  $t(x)$ . However, in practise this assumption is very seldom valid. Nevertheless, the ideas sketched above can still be used if we are able to generate random numbers from more complicated distributions.



**Figure 3.1:** Illustration of perfect sampling of the Gaussian mixture given in (3.10). The dark gray shows the histogram obtained by perfect sampling and the black line shows the true PDF.

### 3.1.2 Importance Sampling

The importance sampling idea can be used to generate random variables from complicated densities. In discussing this algorithm the proposal density  $q(x)$  is typically referred to as the *importance density*. Recall that the support of the proposal density  $q(x)$  should include the support of the target density  $t(x)$ . To understand the idea behind importance sampling, note that integrals in the form (2.2) can be rewritten as

$$I(g(x)) = \int_{\mathbb{R}^{n_x}} g(x) \tilde{w}(x) q(x) dx, \quad (3.11)$$

where

$$\tilde{w}(x) \triangleq \frac{t(x)}{q(x)} \quad (3.12)$$

is referred to as the *importance weight*. We can now obtain an approximation of the target density by first generating  $N \gg 1$  i.i.d. samples from the proposal density, call these  $\{\tilde{x}^i\}_{i=1}^N$ . These samples can then be used to evaluate the corresponding importance weights (3.12) according to

$$\tilde{w}^i = \frac{t(\tilde{x}^i)}{q(\tilde{x}^i)}, \quad i = 1, \dots, N, \quad (3.13)$$

where we have used the notation  $\tilde{w}^i$  rather than  $\tilde{w}(\tilde{x}^i)$  for brevity. This results in the following approximation of the target density

$$\tilde{t}_N(x) = \sum_{i=1}^N \tilde{w}^i \delta(x - \tilde{x}^i). \quad (3.14)$$

Based on the discussion in Section 3.1.1 it is now straightforward to make use of the approximation (3.14) to obtain an estimate of  $I(g(x))$  according to

$$\tilde{I}_N(g(x)) = \int_{\mathbb{R}^{n_x}} g(x) \tilde{t}_N(x) dx = \int_{\mathbb{R}^{n_x}} g(x) \sum_{i=1}^N \tilde{w}^i \delta(x - \tilde{x}^i) dx = \sum_{i=1}^N \tilde{w}^i g(\tilde{x}^i). \quad (3.15)$$

Note that in order to carry out the above idea, the subtle assumption that the target density  $t(x)$  can be exactly evaluated has to be fulfilled. In many applications this is not the case, due to the fact that the target density is only known up to a normalizing constant, which means that the importance weights are only given by the following proportionality

$$\tilde{w}(x) \propto \frac{t(x)}{q(x)}, \quad (3.16)$$

rather than by equality as in (3.12). This has to be dealt with, otherwise the importance sampling idea breaks down, since the importance weights (3.12) will be wrong. Luckily, this can easily be handled within the importance sampling framework.

Intuitively, this should be possible to solve simply by using the *normalized* importance weights

$$w^i = \frac{\tilde{w}^i}{\sum_{j=1}^N \tilde{w}^j}, \quad i = 1, \dots, N, \quad (3.17)$$

rather than the importance weights  $\{\tilde{w}_t^i\}_{i=1}^N$  in the approximation (3.14). That this is indeed true can be realized in the following way. First of all, note that (2.2) can be rewritten as

$$I(g(x)) = \frac{\int g(x) \tilde{w}(x) q(x) dx}{\int \tilde{w}(x) q(x) dx} \quad (3.18)$$

since

$$\int \tilde{w}(x) q(x) dx = \int t(x) dx = 1. \quad (3.19)$$

Inserting the samples from the proposal density and the corresponding importance weights into (3.18) results in

$$\hat{I}_N(g(x)) = \frac{\sum_{i=1}^N \tilde{w}^i g(\tilde{x}^i)}{\sum_{j=1}^N \tilde{w}^j} = \sum_{i=1}^N w^i g(\tilde{x}^i), \quad (3.20)$$

where we have used the normalized importance weights (3.17) in the last equality. Before importance sampling is summarized in Algorithm 3.1 it is worth noting that the target density approximation is given by

$$\hat{t}_N(x) = \sum_{i=1}^N w^i \delta(x - \tilde{x}^i). \quad (3.21)$$

The only difference from (3.14) is that here we are using the normalized rather than the un-normalized importance weights, allowing us to handle un-normalized target densities as well.

---

**Algorithm 3.1 (Importance sampling)**


---

1. Generate  $N$  i.i.d. samples  $\{\tilde{x}^i\}_{i=1}^N$  from the proposal density  $q(x)$  and compute the importance weights

$$\tilde{w}^i = t(\tilde{x}^i)/q(\tilde{x}^i), \quad i = 1, \dots, N. \quad (3.22)$$

2. Form the acceptance probabilities by normalization,

$$w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j, \quad i = 1, \dots, N. \quad (3.23)$$


---

Similarly to what was done in the previous section we can prove both a central limit theorem and a strong law of large numbers for the importance sampling algorithm. However, this falls outside the scope of the present text

### 3.1.3 Sampling Importance Resampling

The topic in the present section is to explain how to compute an unweighted empirical density approximation from the weighted approximation given in (3.21).

The importance weights contains information about how probable it is that the corresponding sample was in fact generated from the target density. Hence, the importance weights can be used as acceptance probabilities, which allows us to generate approximately independent samples  $\{x^i\}_{i=1}^N$  from the target density function. The approximation  $\hat{t}_N(x)$  given in (3.21) is defined using a finite number of samples  $\{\tilde{x}^i\}_{i=1}^N$ . This implies that the process of generating the samples from the target density function is limited to these samples. More specifically this is realized by *resampling* among the samples. The resampling step consists in, for each  $i = 1, \dots, N$  generating a new set of samples  $\{x^i\}_{i=1}^N$  by drawing with replacement according to

$$P(x^i = \tilde{x}^j) = w^j, \quad j = 1, \dots, N. \quad (3.24)$$

The above equation states that the probability of drawing sample  $\tilde{x}^j$  is given by the corresponding weight  $w^j$ . Algorithms and MATLAB code for achieving this give in Section 3.3.1. For now it is sufficient to accept that it can be done. After the resampling (3.24) has been performed the approximation of the target density has been changed from (3.21) to  $\hat{t}_N(x) = \sum_{i=1}^N 1/N \delta(x - x^i)$ .

In Algorithm 3.2 the above discussion is summarized by describing how to approximately generate  $N$  samples from the target density according to the so called sampling importance resampling (SIR) idea.

---

**Algorithm 3.2 (Sampling Importance Resampling (SIR))**


---

1. Generate  $N$  i.i.d. samples  $\{\tilde{x}^i\}_{i=1}^N$  from the proposal density  $q(x)$  and compute the importance weights

$$\tilde{w}^i = t(\tilde{x}^i)/q(\tilde{x}^i), \quad i = 1, \dots, N. \quad (3.25)$$

2. Compute the acceptance probabilities by normalization

$$w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j, \quad i = 1, \dots, N. \quad (3.26)$$

3. For each  $i = 1, \dots, N$  draw a new particle  $x_t^i$  with replacement (resample) according to,

$$P(x^i = \tilde{x}^j) = w^j, \quad j = 1, \dots, N. \quad (3.27)$$

---

The sampling importance resampling algorithm is closely related to the *bootstrap* procedure, introduced by Efron in 1979. The SIR algorithm can in fact be interpreted in terms of a weighted bootstrap procedure. It is worthwhile to note that the resampling step (3.27) is a very important part of the particle filter. Without this step everything breaks down. Example 3.2 will illustrate the importance of using a good importance density  $q(x)$  for generating the samples in Algorithm 3.2.

---

**Example 3.2 (Sampling importance resampling with a Gaussian mixture)**

This example is concerned with the Gaussian mixture (3.10) previously used to illustrate perfect sampling in Example 3.1. We will now assume that we are unable to generate samples from the target density (3.10) and hence we have to decide which importance density  $q(x)$  to use in Algorithm 3.2. In Figure 3.2 we illustrate the approximation from Algorithm 3.2 using the following importance densities

$$q_1(x) = \mathcal{N}(x; 5, 20), \quad (3.28a)$$

$$q_2(x) = \mathcal{N}(x; 1, 20). \quad (3.28b)$$

From the figure it is obvious that the approximation is better if we make use of  $q_1(x)$  rather than  $q_2(x)$ , which is rather natural, since  $q_1(x)$  covers more of the target density than  $q_2(x)$ .

---

## 3.2 A First Particle Filter

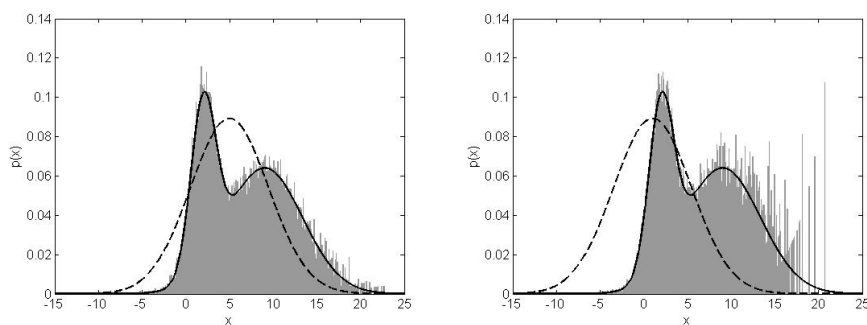
Let us consider the nonlinear filtering problem introduced in Section 2.2 in the light of the random number generation ideas presented in the previous sections. Hence, the concept of time and dynamic models from Chapter 2 will be combined with the random number generators previously discussed.

In the nonlinear filtering problem, the target density is given by the filtering density,

$$t(x_t) = p(x_t | y_{1:t}). \quad (3.29)$$

In order to use the SIR idea outlined in the previous section it is necessary to choose an appropriate proposal density  $q(x_t)$  and a corresponding importance weight. This is in fact quite simple, since according to (2.16a) we have

$$p(x_t | y_{1:t}) = \frac{p(y_t | x_t) p(x_t | y_{1:t-1})}{p(y_t | y_{1:t-1})} \quad (3.30)$$



(a) Using 50000 samples from  $q(x) = q_1(x) = \mathcal{N}(x; 5, 20)$ .

(b) Using 50000 samples from  $q(x) = q_2(x) = \mathcal{N}(x; 1, 20)$ .

**Figure 3.2:** Illustration of importance sampling resampling of the Gaussian mixture given in (3.10). The dark gray shows the histogram obtained by SIR, the black solid line shows the true PDF and the black dashed line shows the PDF of the proposal densities employed.

which suggests the following choices

$$\underbrace{p(x_t|y_{1:t})}_{t(x_t)} \propto \underbrace{p(y_t|x_t)}_{w(x_t)} \underbrace{p(x_t|y_{1:t-1})}_{q(x_t)}. \quad (3.31)$$

The resemblance with (3.4) and (3.16) is obvious. Hence, we can employ Algorithm 3.2 to obtain samples approximately distributed according to the target density. This provides a good framework to start our discussion of the particle filtering algorithms.

The particle filter is typically derived completely within an importance sampling framework. However, it is interesting, at least from a conceptual point of view, to note that we could just as well have used acceptance-rejection sampling, Metropolis-Hastings independence sampling or some other method to generate random numbers in order to obtain alternative particle filtering algorithms.

Based on the appealing properties of the sampling importance resampling idea we will choose to employ this principle in deriving the particle filter, which will be done in an inductive manner. Let us start at time  $t := 0$  by initializing the particles and their corresponding weights according to

$$x_0^i \sim p(x_0), \quad i = 1, \dots, N, \quad (3.32a)$$

$$w_0^i = \frac{1}{N}, \quad i = 1, \dots, N, \quad (3.32b)$$

resulting in the following approximation

$$\hat{p}_N(x_0) = \sum_{i=1}^N \frac{1}{N} \delta(x_0 - x_0^i). \quad (3.33)$$

At time  $t$  we will now assume that the following approximation

$$\hat{p}_N(x_{t-1}|y_{1:t-1}) = \sum_{i=1}^N \frac{1}{N} \delta(x_{t-1} - x_{t-1}^i) \quad (3.34)$$

is available from time  $t-1$ . According to Algorithm 3.2 we should now generate  $N$  i.i.d. samples  $\{\tilde{x}_t^i\}_{i=1}^N$  from the importance density  $q(x_t)$ . From (3.31) we have that

$$q(x_t) = p(x_t|y_{1:t-1}), \quad (3.35)$$

which is rather natural. Given that we have an approximation of the filtering density at time  $t-1$  and are interested in samples of the state at time  $t$ , it should not come as a surprise that we are generating them simply by sampling from the one step ahead prediction density, according to (3.35). The question is now how to generate samples from  $p(x_t|y_{1:t-1})$ . In order to answer this question we will make use of the time update (2.16b) in Theorem 1 in the following way,

$$\begin{aligned} q(x_t) &= p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \\ &\approx \int p(x_t|x_{t-1}) \sum_{i=1}^N \frac{1}{N} \delta(x_{t-1} - x_{t-1}^i) dx_{t-1} \\ &= \sum_{i=1}^N \frac{1}{N} \int p(x_t|x_{t-1}) \delta(x_{t-1} - x_{t-1}^i) dx_{t-1} \\ &= \sum_{i=1}^N \frac{1}{N} p(x_t|x_{t-1}^i). \end{aligned} \quad (3.36)$$

This implies that the proposal density can be chosen as

$$q(x_t^i) = p(x_t|x_{t-1}^i), \quad (3.37)$$

Hence, according to (3.37) the predicted particles  $\{\tilde{x}_t^i\}_{i=1}^N$  are obtained simply by passing the filtered particles from the previous time instance  $\{x_{t-1}^i\}_{i=1}^N$  through the process dynamics (2.4b), i.e.,

$$\tilde{x}_t^i \sim p(x_t|x_{t-1}^i), \quad i = 1, \dots, N. \quad (3.38)$$

Alternatively, using the notation in (2.4a), this can be formulated as

$$\tilde{x}_t^i = f_{t-1}(x_{t-1}^i) + v_{t-1}^i, \quad i = 1, \dots, N, \quad (3.39)$$

where  $v_{t-1}^i$  is a realization from the process noise  $p_{v_{t-1}}(v_{t-1})$ . The next step in Algorithm 3.2 is to compute the importance weights, which according to (3.31) are given by

$$\tilde{w}_t^i = p(y_t|\tilde{x}_t^i), \quad i = 1, \dots, N. \quad (3.40)$$

The acceptance probabilities are then found simply by normalizing the importance weights, step 2 in Algorithm 3.2,

$$w_t^i = \frac{\tilde{w}_t^i}{\sum_{i=1}^N \tilde{w}_t^i} = \frac{p(y_t|\tilde{x}_t^i)}{\sum_{j=1}^N p(y_t|\tilde{x}_t^j)} \quad (3.41)$$

We have now derived the following approximation for the filtering density,

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - \tilde{x}_t^i), \quad (3.42)$$

where  $w_t^i$  is given in (3.41). According to (3.41) the acceptance probabilities  $\{w_t^i\}_{i=1}^N$  are affected by the likelihood function  $p(y_t|x_t)$ . This makes sense, since the likelihood reveals how likely the obtained measurement  $y_t$  is, given the present state  $x_t$ . The better a certain particle explains the received measurement, the higher the probability that this particle was in fact drawn from the true density. Following Algorithm 3.2 (step 3), a new set of particles  $\{x_t^i\}_{i=1}^N$  approximating  $p(x_t|y_{1:t})$  is generated by resampling with replacement among the predicted particles  $\{\tilde{x}_t^i\}_{i=1}^N$ , distributed according to the importance density. Hence, for each  $i = 1, \dots, N$  draw a new particle  $x_t^i$  according to,

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \quad j = 1, \dots, N. \quad (3.43)$$

If this procedure is recursively repeated over time the following approximation

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^N \frac{1}{N} \delta(x_t - x_t^i) \quad (3.44)$$

is obtained and we have in fact derived a particle filter algorithm, which is given in Algorithm 3.3. Note that this algorithm can be generalized. This will be the topic for the subsequent section.

---

**Algorithm 3.3 (A first particle filter)**

---

1. Initialize the particles,  $\{x_0^i\}_{i=1}^N \sim p(x_0)$  and let  $t := 1$ .
2. Predict the particles by drawing  $N$  i.i.d. samples according to

$$\tilde{x}_t^i \sim p(x_t|x_{t-1}^i), \quad i = 1, \dots, N. \quad (3.45)$$

3. Compute the importance weights  $\{\tilde{w}_t^i\}_{i=1}^N$ ,

$$\tilde{w}_t^i = p(y_t|\tilde{x}_t^i), \quad i = 1, \dots, N. \quad (3.46)$$

and normalize  $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$ .

4. For each  $i = 1, \dots, N$  draw a new particle  $x_t^i$  with replacement (resample) according to,

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \quad j = 1, \dots, N. \quad (3.47)$$

5. Set  $t := t + 1$  and repeat from step 2.
- 

The point estimates are computed after step 3 in the above algorithm using the approximation available (3.42). According to (3.3) we can now form the following state estimate,

$$\hat{x}_t = \sum_{i=1}^N w_t^i \tilde{x}_t^i. \quad (3.48)$$

In Section 3.3.2 we will elaborate a bit on forming estimates based on the approximations provided by the particle filter.

In an attempt to add some intuition for the particle filter presented in Algorithm 3.3 we will now draw some parallels to the Kalman filter. The filter is initialized by generating  $N$  particles from the initial density function  $p(x_0)$  (which is considered to be a design choice). In the Kalman filter it is sufficient to choose a mean and a covariance, since the density function has to be Gaussian. The second step is a pure prediction, using the process model. This corresponds to the time update (2.18c) – (2.18d) in the Kalman filter, where the process model is used to predict the state and covariance one time step forward in time.

In the measurement update (step 2 – 3 in Algorithm 3.3) the new measurement is used to assign a probability, represented by the normalized importance weight, to each particle. This probability is calculated using the likelihood function, which describes how likely it was to obtain the measurement given the information available in the particle. The normalized importance weights and the corresponding particles constitute an approximation of the filtering density. This corresponds to the measurement update (2.18a) – (2.18b) in the Kalman filter, where the information in the new measurement is used to update the estimate. Finally, the resampling step will generate particles which are equally probable. The resampled particles will then form the starting point for another iteration of the algorithm. The resampling step has no interpretation in the Kalman filter.

### 3.3 A General Particle Filter

The bootstrap particle filter derived in the previous section is just a particular special case, corresponding to certain design choices of the more general particle filter that will be introduced in the present section.

Before we start the derivation it is instructive to think about Algorithm 3.3 a bit more. In step 2 the particles are predicted without using the information present in the current measurement  $y_t$ , see (3.45). Since the new measurement contains useful information about the state this information could probably be used to make better predictions. This idea can indeed be utilized simply by using an importance density that makes use of the new measurement  $y_t$  in generating the predicted particles. Hence, if we could construct an importance density in the following form

$$q(x_t|x_{t-1}, y_t), \tag{3.49}$$

rather than just using  $p(x_t|x_{t-1})$ , that would probably allow us to produce better predictions. Once again, the intuition behind this stems from the fact that there is additional information available in the new measurement  $y_t$ , which will allow for more informed predictions. Recall Example 3.2, where it was illustrated that it is important to have as good an importance density as possible.

We will not discuss exactly how to draw samples from  $q(x_t|x_{t-1}, y_t)$ , that falls outside the scope of this introductory text. For the purposes of the present section it is sufficient to assume that this can, at least approximately, be done. Note that we are perfectly free to make the following approximation  $q(x_t|x_{t-1}, y_t) \approx p(x_t|x_{t-1})$ , as we did in the previous section.

Let us now start the derivation, which will again be done in an inductive manner. The result is then summarized in Algorithm 3.4. Assume that we have

access to the initial density function  $p(x_0)$ . Then, at time  $t := 0$  we initialize the particles and their weights according to

$$x_0^i \sim p(x_0), \quad i = 1, \dots, N, \quad (3.50a)$$

$$w_0^i = \frac{1}{N}, \quad i = 1, \dots, N, \quad (3.50b)$$

resulting in the following approximation

$$\hat{p}_N(x_0) = \sum_{i=1}^N \frac{1}{N} \delta(x_0 - x_0^i). \quad (3.51)$$

At time  $t$  we will now assume that the following approximation

$$\hat{p}_N(x_{t-1}|y_{1:t-1}) = \sum_{i=1}^N w_{t-1}^i \delta(x_{t-1} - x_{t-1}^i) \quad (3.52)$$

is available from time  $t - 1$ . The task is now to produce an approximation of  $p(x_t|y_t)$  using the importance sampling principle. Hence, the target density is given by  $p(x_t|y_t)$  and the importance density is given by  $q(x_t|x_{t-1}, y_t)$ . According to Algorithm 3.2 the first step is now to generate  $N$  i.i.d. samples from the importance density according to

$$\tilde{x}_t^i \sim q(x_t|x_{t-1}^i, y_t), \quad i = 1, \dots, N. \quad (3.53)$$

Furthermore, the weights are computed according to (3.25), which in the present setting corresponds to

$$\tilde{w}_t^i = \frac{p(\tilde{x}_t^i|y_{1:t-1})}{q(\tilde{x}_t^i|x_{t-1}^i, y_t)}, \quad i = 1, \dots, N \quad (3.54)$$

In order to evaluate this expression we make use of Theorem 1, which states that

$$p(x_t|y_{1:t-1}) = \int_{\mathbb{R}^{n_x}} p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}. \quad (3.55)$$

Using the assumption (3.52) together with (3.55) and (3.54) results in

$$\begin{aligned} \tilde{w}_t^i &= \frac{\int_{\mathbb{R}^{n_x}} p(\tilde{x}_t^i|x_{t-1}) \sum_{i=1}^N w_{t-1}^i \delta(x_{t-1} - x_{t-1}^i) dx_{t-1}}{q(\tilde{x}_t^i|x_{t-1}^i, y_t)} \\ &= \frac{p(\tilde{x}_t^i|x_{t-1}^i)}{q(\tilde{x}_t^i|x_{t-1}^i, y_t)} w_{t-1}^i. \end{aligned} \quad (3.56)$$

We now have the following approximation of the one step ahead prediction

$$\hat{p}_N(x_t|y_{1:t-1}) = \sum_{i=1}^N \tilde{w}_t^i \delta(x_t - \tilde{x}_t^i), \quad (3.57)$$

where the weights are given by (3.56) and the particles are generated according to (3.53).

In order to obtain an approximation of  $p(x_t|y_{1:t})$  we make use of

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \quad (3.58)$$

where

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t \quad (3.59)$$

The above expressions are from Theorem 1. The one step ahead prediction density is approximated according to (3.57), implying that (3.59) is straightforwardly approximated according to

$$\begin{aligned} \hat{p}_N(y_t|y_{1:t-1}) &= \int p(y_t|x_t) \sum_{i=1}^N \tilde{w}_t^i \delta(x_t - \tilde{x}_t^i) dx_t \\ &= \sum_{i=1}^N p(y_t|\tilde{x}_t^i) \tilde{w}_t^i. \end{aligned} \quad (3.60)$$

Inserting (3.57) and (3.60) into (3.58) provides the following approximation

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^N \frac{p(y_t|\tilde{x}_t^i) \tilde{w}_t^i}{\sum_{j=1}^N p(y_t|\tilde{x}_t^j) \tilde{w}_t^j} \delta(x_t - \tilde{x}_t^i) \quad (3.61)$$

Finally, according to step 3 in Algorithm 3.2 the resampling is performed. Now, the resampling step does not have to be performed at each point in time. It is common to construct a measure of sample diversity, in order to detect when resampling should be performed. This is included in Algorithm 3.4 below and further discussed in Section 3.3.1.

---

**Algorithm 3.4 (Particle filter)**

---

1. Initialize the particles,  $x_0^i \sim p(x_0)$ ,  $i = 1, \dots, N$  and the weights,  $w_0^i = 1/N$ ,  $i = 1, \dots, N$  and let  $t := 1$ .
2. Generate  $N$  i.i.d. new particles by drawing from the importance density

$$\tilde{x}_t^i \sim q(x_t|x_{t-1}^i, y_t), \quad i = 1, \dots, N. \quad (3.62)$$

and update the weights accordingly

$$\tilde{w}_t^i = \frac{p(\tilde{x}_t^i|x_{t-1}^i)}{q(\tilde{x}_t^i|x_{t-1}^i, y_t)} w_{t-1}^i, \quad i = 1, \dots, N. \quad (3.63)$$

3. Update the weights according to,

$$w_t^i = \frac{p(y_t|x_t^i) \tilde{w}_t^i}{\sum_{j=1}^N p(y_t|x_t^j) \tilde{w}_t^j}, \quad i = 1, \dots, N. \quad (3.64)$$

4. Resample according to Algorithm 3.5 or Algorithm 3.7.
  5. Set  $t := t + 1$  and iterate from step 2.
-

It is interesting to note that if we choose the importance density according to  $q(x_t) = p(x_t|x_{t-1})$ , Algorithm 3.4 is reduced to Algorithm 3.3.

---

**Algorithm 3.5 (Resampling 1)**

---

1. For each  $i = 1, \dots, N$  draw a new particle  $x_t^i$  with replacement (resample) according to,

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \quad j = 1, \dots, N. \quad (3.65)$$

---

There are a lot of design choices to discuss when it comes to the particle filter. The literature is full of “new” particle filtering algorithms. However, most of them are just minor variations of the main algorithm presented above. All the important mechanisms of the particle filter has been explained above.

### 3.3.1 Resampling Algorithms

The resampling step is absolutely essential for the particle filter to work, since without resampling the variance of the importance weights will grow to infinity and the filter will diverge. We will in this section describe what the resampling step does and provide an efficient algorithm for implementing it.

The resampling step performs a *re-weighting* of an empirical probability density function. More specifically, it changes the non-uniformly weighted empirical density

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - \tilde{x}_t^i) \quad (3.66)$$

into an uniformly weighted empirical density

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^N \frac{1}{N} \delta(x_t - x_t^i) \quad (3.67)$$

by generating  $N$  i.i.d. samples  $\{x_t^i\}_{i=1}^N$  from (3.66). This is accomplished by drawing a new sample  $x_t^i$  with replacement for each  $i = 1, \dots, N$ , according to

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \quad j = 1, \dots, N. \quad (3.68)$$

In Figure 3.3 an illustration of the resampling (3.68) procedure is provided.

A direct implementation according to the explanation in association with Figure 3.3 is not recommended. There are more efficient (both in terms of computational complexity and the quality of the samples) algorithms available. Here, we will only introduce one of the most commonly used algorithms, called systematic resampling. Other popular algorithms include stratified resampling and residual resampling.

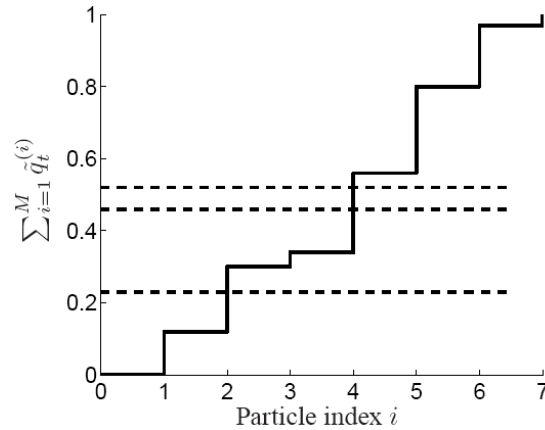
---

**Algorithm 3.6 (Resampling – Systematic sampling)**

---

1. Generate  $N$  ordered numbers according to

$$u_k = \frac{(k-1) + \tilde{u}}{N}, \quad \tilde{u} \sim \mathcal{U}(0, 1). \quad (3.69)$$



**Figure 3.3:** Illustration of the resampling step. The new set of samples is obtained by first generating  $N$  sorted uniformly distributed random numbers, three of which are shown by the dashed lines in the figure. These are then associated with a particle guided by the cumulative sum of the normalized importance weights. For the situation illustrated in this figure, particle number 2 is chosen once and particle number 4 is chosen twice.

2. The resampled particles are obtained by producing  $n_i$  copies of particle  $x^i$ , where

$$n_i = \text{the number of } u_k \in \left( \sum_{s=1}^{i-1} w_t^s, \sum_{s=1}^i w_t^s \right]. \quad (3.70)$$

The MATLAB code for implementing the systematic resampling algorithm given in Algorithm 3.6 is given in Code 3.1 below. This code will later be used by the particle filter code.

---

**Code 3.1 (Systematic Resampling (Algorithm 3.3))**

---

```
function i=sysresample(q)
qc=cumsum(q); M=length(q);
u=(0:M-1)+rand(1))/M;
i=zeros(1,M); k=1;
for j=1:M
    while (qc(k)<u(j))
        k=k+1;
    end
    i(j)=k;
end
end
```

As an alternative to Algorithm 3.5 we will now give an algorithm which does not require the resampling step to be carried out at every time instant. The particle filtering algorithm with this resampling algorithm is sometimes referred to as the *sampling importance sampling (SIS)* particle filter.

**Algorithm 3.7 (Resampling 2)**


---

Compute the effective sample size,

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^N (w_t^i)^2} \quad (3.71)$$

*if*  $\hat{N}_{\text{eff}} < N_{\text{th}}$

Generate a new set of particles  $\{x_t^i\}_{i=1}^N$  using Algorithm 3.5 and  
set  $w_t^i := 1/N$ , for  $i = 1, \dots, N$ .

*else*

Keep the current particles and their corresponding weights.

*end*

---

**3.3.2 Computing the Estimate**

We have this far only explained how to compute estimates of the filtering density function  $p(x_t|y_{1:t})$ . Typically, we are interested in some particular property of the underlying state variable, such as for instance a point estimate and its associated quality, provided by the covariance. The present section will describe how these estimates can be obtained using the approximated densities. The approach can readily be extended to other interesting features of the underlying state variable.

An minimum mean square error estimate of the mean value of the current state is obtained by inserting  $g(x_t) = x_t$  in (2.2), resulting in

$$\mathbb{E}_{p(x_t|y_{1:t})} \{x_t\} = \int x_t p(x_t|y_{1:t}) dx_t = \hat{x}_{t|t}. \quad (3.72)$$

Using the following estimate of the probability density function,

$$\hat{p}_N(x_t|Y_s) = \sum_{i=1}^M w_t^i \delta(x_t - x_t^i), \quad (3.73)$$

results in

$$\hat{x}_{t|t} = \int x_t \hat{p}_M(x_t|y_{1:t}) dx_t = \int x_t \sum_{i=1}^N w_t^i \delta(x_t - x_t^i) dx_t = \sum_{i=1}^N w_t^i x_t^i. \quad (3.74)$$

The reason for using the weighted approximation of the filtering density, rather than the unweighted, is simply that the resampling step will slightly increase the variance of the weights. This will in turn degrade the estimate.

Similarly to what was done above, an estimate of the covariance is obtained using

$$g(x_t) = (x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})^T \quad (3.75)$$

in (2.2), which after some calculations results in

$$\hat{P}_{t|t} = \sum_{i=1}^N w_t^i \left( x_{t|t}^i - \hat{x}_{t|t} \right) \left( x_{t|t}^i - \hat{x}_{t|t} \right)^T. \quad (3.76)$$

From the two expressions (3.74) and (3.76) it is clear how the estimates are affected by the information in the normalized importance weights  $w_t^i$ . The more likely a certain particle is, the more it influences the estimate.

### 3.3.3 Implementation

We have this far derived the particle filter using mathematical tools and we have also provided an algorithm statement. In order to make the particle filter more accessible for those previously not familiar to it we will now provide a MATLAB implementation of the particle filter given in Algorithm 3.3. Having read this section the reader will be able to implement a particle filter from scratch within five minutes. Before the implementation is given there are a few steps in the algorithm that deserves some attention.

In step 2 we are asked to generate samples according to

$$\tilde{x}_t^i \sim p(x_t | x_{t-1}^i), \quad i = 1, \dots, N. \quad (3.77)$$

Since the model is given by (2.4) this can be accomplished according to

$$\tilde{x}_t^i = f(x_{t-1}^i) + v_t^i, \quad i = 1, \dots, N, \quad (3.78)$$

where  $v_t^i \sim p_{v_t}(v_t)$ . Furthermore, the importance weights are given by evaluating the likelihood function, which according to (2.6b) is provided by

$$\tilde{w}_t^i = p(y_t | \tilde{x}_t^i) = p_{e_t}(y_t - h(\tilde{x}_t^i)), \quad i = 1, \dots, N. \quad (3.79)$$

The MATLAB code for Algorithm 3.3 is given in Code 3.2 below. The commented numbers 1 – 4 indicate the connection to the corresponding step in Algorithm 3.4. Recall that the MATLAB code (`sysresample`) for systematic resampling was previously given in Code 3.1.

---

#### **Code 3.2 (Particle filter (Algorithm 3.3))**

---

```
function [xhat] = PF(f,h,pe,B,Q,x0,PO,N,y)
    nx=size(PO,2);
    x = repmat(x0,1,N)+sqrtm(PO)*randn(nx,N); % 1.
    xhat = zeros(nx,size(y,2));
    for t=1:size(y,2)
        x = feval(f,x,t) + B*sqrtm(Q)*randn(1,N); % 2.
        e = repmat(y(:,t),1,N) - feval(h,x);
        w = feval(pe,e); % 3.
        w = w/sum(w);
        xhat(:,t) = sum(repmat(w,nx,1).*x,2);
        index = sysresample(w); % 4.
        x = x(:,index);
    end;
end
```

In Example 3.3 we make use of the particle filter provided in Code 3.2 in order to estimate a one dimensional motion. Since this is a linear Gaussian problem the Kalman filter applies, which allows us to compare the performance to the optimal filter. We will then in Example 3.4 move on to a much trickier nonlinear time-varying system, which cannot be handled by the Kalman filter, whereas the particle filter can directly be used. The purpose of the first example is simply to illustrate that the particle filter will provide the same result as the optimal filter when it is available and the number of particles tends to infinity. The idea behind the second example is to show that the particle filter can indeed handle more complicated nonlinear, non-Gaussian examples as well.

---

**Example 3.3 (One dimensional motion)**

Let us study a one dimensional motion, where we assume that we can measure the position and the acceleration. The task is to compute estimates of the position and the velocity. Hence, we will devise a very simple positioning system. The state vector is given by

$$x_t = \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix}, \quad (3.80)$$

where  $p_t$  denotes the position,  $v_t$  denotes the velocity and  $a_t$  denotes the acceleration. The discrete-time state-space model is given by a sampled double integrator according to,

$$x_{t+1} = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + \begin{pmatrix} T^3/6 \\ T^2/2 \\ T \end{pmatrix} v_t, \quad v_t \sim \mathcal{N}(0, Q_t), \quad (3.81)$$

$$y_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + e_t, \quad e_t \sim \mathcal{N}(0, R_t), \quad (3.82)$$

where  $T$  denotes the sampling time. This is a linear system subject to Gaussian noise, which implies that the Kalman filter will provide the optimal (in the sense of a minimum variance) solution. Hence, we can compare the particle filter to the best available solution. The MATLAB code for the solving this problem is provided in Code 3.3, save for the Kalman filter code.

---

**Code 3.3 (Solution to example 3.3)**

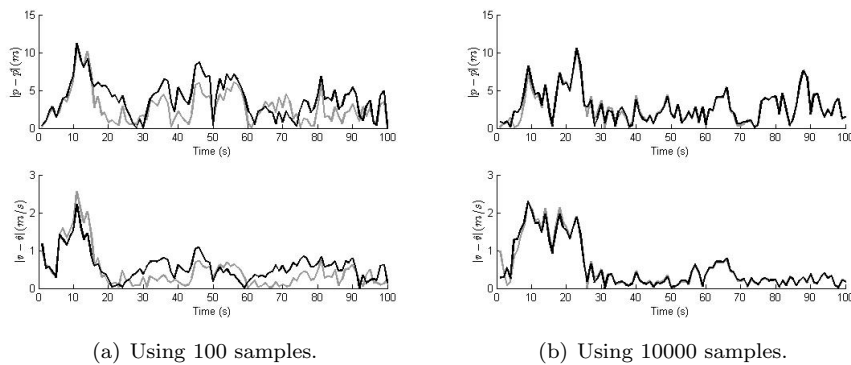
```
N = 100; % Number of particles
x0 = [0 0 0]'; % Initial guess of the state value
P0 = 1*eye(3); % Initial covariance
B = [1/6 1/2 1]';
Q = 0.01^2; % Process noise
R = [10^2 0; 0 0.5^2]; % Measurement noise
f = inline('[1 1 0.5; 0 1 1; 0 0 1]*x', 'x', 't');
h = inline('[1 0 0; 0 0 1]*x', 'x');
pe = inline('exp(-(1/2)*(sum(e.*(inv([10^2 0; 0 0.5^2])*e))))');
```

```

x(:,1) = [0 1 0.1]'; % Initial state used in simulation
for t=1:100 % Simulate the model
    x(:,t+1) = feval(f,x(:,t),t) + B*sqrtm(Q)*randn(1);
    y(:,t) = feval(h,x(:,t)) + sqrtm(R)*randn(2,1);
end;
xHat = PF(f,h,pe,B,Q,x0,P0,N,y);

```

In Figure 3.4 we provide a particular realization illustrating the result. From this figure it can be seen that using only 100 particles provides a reasonable result. When the number of particles is increased up to 10000, the result is almost identical to the optimal filter.



**Figure 3.4:** Illustration of the position and velocity estimation error, when  $N = 100$  and  $N = 10000$  particles are used, respectively. The estimation error in acceleration is not shown for brevity. Note that this is one particular realization. The black line corresponds to the estimation error from the particle filter the grey line corresponds to the estimation error from the Kalman filter.

---

#### Example 3.4 (Nonlinear system)

In this example the particle filter will be used to estimate the state in the following discrete-time, nonlinear, non-Gaussian, time-varying state-space model (Model 2.1),

$$x_{t+1} = \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8 \cos(1.2t) + v_t, \quad (3.83a)$$

$$y_t = \frac{x_t^2}{20} + e_t, \quad (3.83b)$$

where  $x_0 \sim \mathcal{N}(0,5)$ ,  $v_t$  and  $e_t$  are mutually independent white noise sequences,  $v_t \sim \mathcal{N}(0,10)$  and  $e_t \sim \mathcal{N}(0,1)$ . In the literature this system has become a favourite when it comes to illustrating the particle filter.

The fact that we measure the square of the state variable implies that the measurement does not contain any information about the sign of the state. This makes the state estimation problem especially challenging. Since there is no closed form solution for the optimal filter for this system, we can only compare the particle filter performance to the true state. In Code 3.4 the MATLAB code for this example is provided.

---

*Code 3.4 (Solution to example 3.4)*

---

```

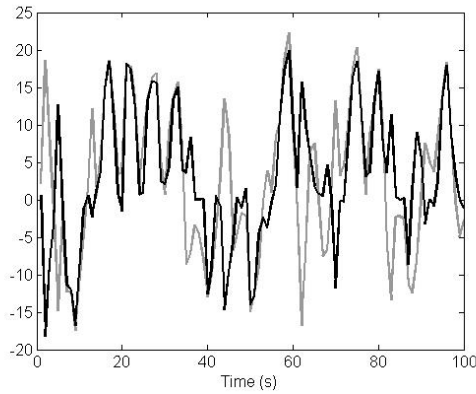
N = 5000;                % Number of particles
P0 = 5;                  % Initial covariance
x0 = sqrt(P0)*randn(1); % Initial guess of the state value
Q = 10;                  % Process noise
R = 1;                   % Measurement noise
f = inline('x./2 + 25*x./(1+x.^2)+8*cos(1.2*t)', 'x', 't');
h = inline('(x.^2)/20', 'x');
pe = inline('exp(-(1/2)*(e.^2))', 'e');

x(:,1) = x0;             % Initial state used in simulation
for t=1:100              % Simulate the model
    x(:,t+1) = feval(f,x(:,t),t) + sqrtm(Q)*randn(1);
    y(:,t)   = feval(h,x(:,t)) + sqrtm(R)*randn(1);
end;
xHat = PF(f,h,pe,1,Q,x0,P0,N,y);

```

---

In Figure 3.5 the particle filter estimate using 5000 particles, implemented in Code 3.4, is shown together with the true state. The particle filter has obvious problems from time to time, due to the fact that information about the sign of the state is not available in the measurement. Increase the number of particles.



**Figure 3.5:** The black line corresponds to the particle filter estimate of the state  $x_t$  in the system (3.83). The grey line corresponds to the true state. The error is mainly due to the square in the measurement equation.

---

# Appendix A

## Useful Facts from Probability Theory

In this appendix we list some facts from probability theory that are important to understand in order to read the main text. For a detailed treatment of probability theory we refer to one of the many textbooks on the subject.

**Theorem 2 (Bayes' theorem)** *Given two random variables  $a$  and  $b$  the conditional probability density function  $p(a|b)$  is given by*

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}. \quad (\text{A.1})$$

**Definition 1 (Markov property)** A discrete-time stochastic process  $\{x_t\}$  is said to possess the Markov property if

$$p(x_{t+1}|x_1, \dots, x_t) = p(x_{t+1}|x_t). \quad (\text{A.2})$$

**Definition 2 (Multivariable Normal Density)** *A random variable  $a$  with  $E\{a\} = \mu_a$  and  $\text{Cov}\{a\} = \Sigma_a$ , such that  $\det \Sigma_a > 0$  is  $\mathcal{N}(\mu_a, \Sigma_a)$  if and only if the probability density function for  $x$  is*

$$p(x) = \frac{1}{(2\pi)^{n_x/2} \sqrt{\det \Sigma_a}} e^{-\frac{1}{2}(x-\mu_a)^T \Sigma_a^{-1} (x-\mu_a)} \quad (\text{A.3})$$

In order to have a practical notation for stating that a probability density function is normal with a certain mean value and covariance we will define the following

$$\mathcal{N}(x; \mu_a, \Sigma_a) = \frac{1}{(2\pi)^{n_x/2} \sqrt{\det \Sigma_a}} e^{-\frac{1}{2}(x-\mu_a)^T \Sigma_a^{-1} (x-\mu_a)}. \quad (\text{A.4})$$

This notation will allow us to write  $\mathcal{N}(x; \mu_a, \Sigma_a)$  rather than the entire expression in (A.4), which is of course convenient.

**Theorem 3 (Marginalization of Gaussian densities)** *Let*

$$p(a|b) = \frac{1}{(2\pi)^{n_a/2} \sqrt{\det \Sigma_{a|b}}} e^{-\frac{1}{2}(a-Cb-\bar{a})^T \Sigma_{a|b}^{-1} (a-Cb-\bar{a})}, \quad (\text{A.5a})$$

$$p(b) = \frac{1}{(2\pi)^{n_b/2} \sqrt{\det \Sigma_b}} e^{-\frac{1}{2}(b-\mu_b)^T \Sigma_b^{-1} (b-\mu_b)}, \quad (\text{A.5b})$$

where  $\tilde{a}$  is a known constant. Then

$$p(a) = \mathcal{N}(a|C\mu_b + \tilde{a}, C\Sigma_b C^T + \Sigma_{a|b}). \quad (\text{A.6})$$

**Proof 1**

$$\begin{aligned} p(a) &= \int p(a, b)db = \int p(a|b)p(b)db \\ &= \int \frac{1}{(2\pi)^{(n_a+n_b)/2} \sqrt{\det \Sigma_{a|b} \det \Sigma_b}} e^{-\frac{1}{2}E} db, \end{aligned} \quad (\text{A.7})$$

where

$$E = (a - Cb)^T \Sigma_{a|b}^{-1} (a - Cb) + (b - \mu_b)^T \Sigma_b^{-1} (b - \mu_b). \quad (\text{A.8})$$

Introduce the following variables

$$e = b - \mu_b, \quad (\text{A.9})$$

$$f = a - C\mu_b - \tilde{a}, \quad (\text{A.10})$$

allows us to write the exponent (A.8) according to

$$E = (f - Ce)^T \Sigma_{a|b}^{-1} (f - Ce) + e^T \Sigma_b^{-1} e \quad (\text{A.11})$$

$$= \begin{pmatrix} e \\ f \end{pmatrix}^T \begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & -C^T \Sigma_{a|b}^{-1} \\ -\Sigma_{a|b}^{-1} C & \Sigma_{a|b}^{-1} \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} \quad (\text{A.12})$$

The middle term in (A.12) can, using a block diagonal factorization, be written according to

$$\begin{aligned} &\begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & -C^T \Sigma_{a|b}^{-1} \\ -\Sigma_{a|b}^{-1} C & \Sigma_{a|b}^{-1} \end{pmatrix} \\ &= \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^T \begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}, \end{aligned} \quad (\text{A.13})$$

where

$$K = \Sigma_b C^T (C\Sigma_b C^T + \Sigma_{a|b})^{-1}, \quad (\text{A.14a})$$

$$S = C\Sigma_b C^T + \Sigma_{a|b}. \quad (\text{A.14b})$$

Inserting this in (A.12) results in

$$\begin{aligned} E &= \begin{pmatrix} e \\ f \end{pmatrix}^T \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^T \begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} \\ &= (e - Kf)^T (C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1}) (e - Kf) + f^T S^{-1} f \end{aligned} \quad (\text{A.15})$$

Furthermore, the determinant in (A.7) can be written

$$\frac{1}{\det \Sigma_{a|b} \det \Sigma_b} = \det \Sigma_{a|b}^{-1} \det \Sigma_b^{-1} = \det \begin{pmatrix} \Sigma_{a|b}^{-1} & 0 \\ 0 & \Sigma_b^{-1} \end{pmatrix} \quad (\text{A.16})$$

The determinant of a triangular matrix with unit diagonal equals one. This implies that the matrix in (A.16) can be multiplied with any such matrix without changing the value of the expression. For example it can be written as

$$\begin{aligned} & \det \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^{-T} \begin{pmatrix} I & 0 \\ -C & I \end{pmatrix}^T \begin{pmatrix} \Sigma_{a|b}^{-1} & 0 \\ 0 & \Sigma_b^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -C & I \end{pmatrix} \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^{-1} \\ &= \det \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^{-T} \begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & -C^T \Sigma_{a|b}^{-1} \\ -\Sigma_{a|b}^{-1} C & \Sigma_b^{-1} \end{pmatrix} \begin{pmatrix} I & -K \\ 0 & I \end{pmatrix}^{-1} \end{aligned} \quad (\text{A.17})$$


which allows us to use the block diagonal factorization (A.13) to write the determinant as

$$\det \begin{pmatrix} C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} = \frac{1}{\det(C^T \Sigma_{a|b}^{-1} C + \Sigma_b^{-1})^{-1} \det S} \quad (\text{A.18})$$

Finally, inserting (A.15) and (A.18) into (A.7), results in

$$p(a) = \frac{1}{(2\pi)^{n_a/2} \sqrt{\det S}} e^{-\frac{1}{2} f^T S^{-1} f} \quad (\text{A.19})$$

$$p(a) = \mathcal{N}(a|C\mu_b + \tilde{a}, C\Sigma_b C^T + \Sigma_{a|b}). \quad (\text{A.20})$$

	<b>Avdelning, Institution</b> Division, Department  Division of Automatic Control Department of Electrical Engineering	<b>Datum</b> Date  2010-05-10
	<b>Språk</b> Language  <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English  <input type="checkbox"/> _____	<b>Rapporttyp</b> Report category  <input type="checkbox"/> Licentiatavhandling <input type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input checked="" type="checkbox"/> Övrig rapport <input type="checkbox"/> _____
<b>URL för elektronisk version</b>  <a href="http://www.control.isy.liu.se">http://www.control.isy.liu.se</a>		LiTH-ISY-R-2953
<b>Titel</b> Solving Nonlinear State Estimation Problems Using Particle Filters - An Engineering Per- Title                      spective   <b>Författare</b> Thomas B. Schön Author		
<b>Sammanfattning</b> Abstract  <p>The goal of this work is to provide a self contained introduction to the nonlinear state estimation problem and how this can be solved using the particle filter. The objective in nonlinear filtering is to recursively in time estimate the state in a dynamic model. This is a real-time signal processing problem, where new measurement information inherently arrives in a sequential fashion. The particle filter offers a powerful numerical solution to this problem. Besides a theoretical derivation of the particle filter we also provide MATLAB code solving simple examples.</p>		
<b>Nyckelord</b> Keywords              Nonlinear state estimation, particle filter, nonlinear filtering, dynamic systems		