Resampling in particle filters

Jeroen D. Hol

LiTH-ISY-EX-ET-0283-2004

Linköping 2004
Resampling in particle filters

Internship

performed at
Division of Automatic Control
Department of Electrical Engineering
Linköpings University, Sweden

by
Jeroen D. Hol

LiTH-ISY-EX-ET-0283-2004

Supervisor: Lic., B.Sc. T. Schön
isy, Linköpings universitet
Prof. F. Gustafsson
isy, Linköpings universitet

Examiner: Prof. J.B. Jonker
CTW, University of Twente

Linköping, 31st May 2004
In this report a comparison is made between four frequently encountered resampling algorithms for particle filters. A theoretical framework is introduced to be able to understand and explain the differences between the resampling algorithms. This facilitates a comparison of the algorithms based on resampling quality and on computational complexity. Using extensive Monte Carlo simulations the theoretical results are verified. It is found that systematic resampling is favourable, both in resampling quality and computational complexity.
Abstract

In this report a comparison is made between four frequently encountered resampling algorithms for particle filters. A theoretical framework is introduced to be able to understand and explain the differences between the resampling algorithms. This facilitates a comparison of the algorithms based on resampling quality and on computational complexity. Using extensive Monte Carlo simulations the theoretical results are verified. It is found that systematic resampling is favourable, both in resampling quality and computational complexity.

Keywords: Resampling, Particle filters, sequential Monte Carlo methods, SIR, SISR
Acknowledgements

This report could not have been realised without the help from a number of people, whom I would like to thank.

First of all, I would like to thank my direct supervisor, Thomas Schön, for guiding me during my research and for providing lots of valuable comments on earlier version of this report. I especially appreciated his advice regarding the art of writing a scientific report. I also want to thank Fredrik Gustafsson for generating some ideas on this reports subject. And I would like to thank all the people at RT who made my stay at Linköpings University a pleasant one. Special thanks go to Yvo Boers and Hans Driessen for introducing me to the control group at Linköpings university. Without their help this report would not exist today. Last, but not least, I would like to express my gratitude to my family and Nantje for the love and support they provided me with during my stay abroad.

Linköping, 31st May 2004

Jeroen Hol
Contents

Abstract v

Acknowledgements vii

1 Introduction 1
  1.1 Background .................................................. 1
  1.2 Problem formulation ........................................ 1
  1.3 Outline ....................................................... 2

2 Particle filters 3
  2.1 Recursive state estimation .................................. 4
  2.2 Perfect sampling ............................................. 5
  2.3 Importance sampling ....................................... 6
    2.3.1 Sequential importance sampling ....................... 7
  2.4 Resampling .................................................. 9
  2.5 Basic particle filter ....................................... 9
  2.6 Extensions .................................................. 11
  2.7 Applications ................................................ 11

3 Resampling algorithms 13
  3.1 Introduction ................................................ 13
  3.2 Principles ................................................... 14
    3.2.1 Sample generation .................................... 15
    3.2.2 Set restriction ........................................ 19
  3.3 Overview .................................................... 20
    3.3.1 Simple random resampling ............................ 20
    3.3.2 Stratified resampling ................................ 21
    3.3.3 Systematic resampling ................................ 21
    3.3.4 Residual resampling .................................. 22
  3.4 Comparison .................................................. 22
    3.4.1 Resampling Quality .................................... 23
    3.4.2 Computational complexity .............................. 23
Chapter 1

Introduction

1.1 Background

In many engineering applications one is faced with the following problem: the states describing the system are unknown and they have to be estimated based upon observations or measurements. The solution to this estimation problem is given by the posterior probability density function (PDF), which gives the probability distribution of the states using all available information. Using this PDF, point estimates (e.g., mean) and confidence intervals can be calculated.

In the linear, Gaussian case a parameterisation of these distributions can be derived, known as the Kalman filter. However, in the general, nonlinear case, there is no closed form expression available.

The most common approach to deal with nonlinear situations is to approximate the model with a linearised version of it and than use the optimal Kalman filter with this approximate model. If the nonlinearities are not to strong, this usually works fine.

Recently, a simulation based method has come available which is capable of handling estimation problems in nonlinear and non-Gaussian settings. Using a class of methods referred to as sequential Monte Carlo methods or, more commonly, particle filters, it is possible to numerically approximate the posterior density. An attractive feature is that these methods provide an approximate solution to the real problem instead of an optimal solution to the approximate problem. This is the main reason for the improved accuracy of particle filters over Kalman-filter based methods.

1.2 Problem formulation

The resampling step is a crucial and computationally expensive part of a particle filter. So a well argued choice of resampling method is justified as the entire method benefits from reduced complexity and/or improved quality of the resampling step. In literature quite a few different resampling methods can be found. They are,
however, scattered among many articles and books and, to the best of the author’s knowledge, an overview or discussion with pros and cons is missing. This report aims at filling this gap.

The main objective of this report is to analyse and compare frequently used resampling algorithms and their implementations. The algorithms are compared with respect to resampling quality and computational efficiency, both in theory and with simulations.

1.3 Outline

In this report four frequently used resampling algorithms are analysed and compared. The resampling step is introduced, together with the generic particle filter, in Chapter 2. Chapter 3 discusses the resampling algorithms. A theoretical framework is introduced to be able to understand how the algorithms work and what the differences are. Using this framework a comparison of the resampling algorithms is made on resampling quality and computational complexity. The results from this comparison are verified using Monte Carlo simulations in Chapter 4 and in Chapter 5 some conclusions and recommendations are discussed.
Chapter 2

Particle filters

This chapter provides a short introduction to particle filters, explaining the basic algorithm. More complete and detailed introductions are provided by e.g. [2, 6, 7].

A typical (control) problem is that the states of a system are unknown and have to be estimated using observations. For example, in order to control a robot arm, information about position, velocity and acceleration of various parts is required. Typically only some of these states are measured; the others have to be estimated.

Particle filters are sequential Monte Carlo methods that provide approximations to the solution of this estimation problem for dynamical systems described by the discrete-time nonlinear state-space model

\[
x_{t+1} = f(x_t, n_t) \quad (2.1a)
\]
\[
y_t = h(x_t, \nu_t) \quad (2.1b)
\]

These equations describe the evolution of the states \(x_t\) and the observations \(y_t\). The process noise \(n_t\) and measurement noise \(\nu_t\) are assumed to be white and independent. This state-space model can equivalently be formulated as an infinite dimensional hidden Markov model (HMM), defined by

\[
p(x_0), \, p(x_t|x_{t-1}) \quad (2.2a)
\]
\[
p(y_t|x_t) \quad (2.2b)
\]

In this formulation, the stochastic states \(x_t\) evolve according to a Markov chain with transition probabilities given by (2.2a). In other words, the chain is defined by the initial probability density function (PDF), \(p(x_0)\), combined with the likelihood of the states at some time given the states at the previous time, \(p(x_t|x_{t-1})\). The observations \(y_t\) are distributed according to (2.2b).

A typical estimation problem is to estimate the states \(x_t\) at time \(t\) given the history of measurements or observations up to time \(t\), \(y_{0:t} = \{y_0, \ldots, y_t\}\). The solution to this problem is given by the filtering density \(p(x_t|y_{0:t})\), or in words, the likelihood of the current states given the history of observations.

Particle filters provide approximations to the filtering density for dynamic systems described by (2.1) and (2.2). These equations describe very general, nonlinear
and non-Gaussian, systems. This implies that the assumptions, linear systems with
Gaussian noise, required by the classical Kalman filter can be dropped. The capa-

tibility to handle nonlinear, non-Gaussian systems allows particle filters to achieve
improved accuracy over Kalman filter-based estimation methods.

2.1 Recursion state estimation

In many applications, the observations come available one at a time. This moti-
vates a recursive framework to allow online estimation. Theorem 2.1 shows the
theoretical foundation commonly used to so.

**Theorem 2.1**
The filter density can be recursively updated using the following equations [21]:

\[
p(x_t|y_{0:t}) = \frac{p(y_t|x_t)p(x_t|y_{0:t-1})}{p(y_t|y_{0:t-1})} \quad (2.3a)
\]

\[
p(y_t|y_{0:t-1}) = \int p(y_t|x_t)p(x_t|y_{0:t-1})dx_t \quad (2.3b)
\]

\[
p(x_{t+1}|y_{0:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{0:t})dx_t \quad (2.3c)
\]

This scheme consists of a measurement update according to (2.3a), (2.3b) and a
time update according to (2.3c).

**Proof**
Using the Markov property of model (2.2) and Bayes’ theorem

\[
p(x|y) = \frac{p(y|x)p(x)}{p(y)} \quad (2.4a)
\]

\[
p(y) = \int p(x, y)dx = \int p(y|x)p(x)dx \quad (2.4b)
\]

it can be shown that

\[
p(x_t|y_{0:t}) = p(x_t|y_t, y_{0:t-1}) = \frac{p(y_t|x_t)p(x_t|y_{0:t-1})}{p(y_t|y_{0:t-1})} = \frac{p(y_t|x_t)p(x_t|y_{0:t-1})}{p(y_t|y_{0:t-1})} \quad (2.5)
\]

Also,

\[
p(y_t, x_t|y_{0:t-1}) = p(y_t|x_t, y_{0:t-1})p(x_t|y_{0:t-1}) = p(y_t|x_t)p(x_t|y_{0:t-1}) \quad (2.6)
\]

\[
p(x_{t+1}, x_t|y_{0:t}) = p(x_{t+1}|x_t, y_t)p(x_t|y_{0:t}) = p(x_{t+1}|x_t)p(x_t|y_{0:t}) \quad (2.7)
\]

implying that (integrating w.r.t. \( x_t \) on both sides)

\[
p(y_t|y_{0:t-1}) = \int p(y_t|x_t)p(x_t|y_{0:t-1})dx_t \quad (2.8)
\]

\[
p(x_{t+1}|y_{0:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{0:t})dx_t \quad (2.9)
\]
Note that the conditional probability density functions \( p(x_{t+1}|x_t) \) and \( p(y_t|x_t) \) occurring in (2.3) are known from model (2.2).

Theorem 2.1 looks deceptively simple. However, the recursion is a rather theoretical result since the (multidimensional) integrals involved typically don’t permit an analytical solution. The linear, Gaussian case is an exception and its solution is known as the Kalman filter.

Particle filters numerically approximate the posterior PDF \( p(x_{0:t}|y_{0:t}) \) using Monte Carlo (MC) techniques. The approximation of the filtering density \( p(x_t|y_{0:t}) \) is obtained by marginalisation. The techniques that are used will be discussed in the subsequent sections.

### 2.2 Perfect sampling

Perfect sampling forms the basis of all Monte Carlo techniques. It is a way of approximating PDFs. In this section perfect sampling is introduced using an intuitive derivation.

The cumulative density function (CDF) \( D(\cdot) \) is related to the PDF \( p(\cdot) \) by

\[
D(x) = P(X < x) = \int_{-\infty}^{x} p(s)ds
\]  

(2.10)

Clearly this function is piecewise continuous. It is also bounded, for by definition probabilities are limited to \([0, 1]\), so it can be arbitrary closely approximated using a stair function \( \hat{D}_N(\cdot) \). An approximate density \( \hat{p}_N(\cdot) \) of \( p(\cdot) \) is constructed by differentiating \( \hat{D}_N(\cdot) \). That is, \( p(\cdot) \) is approximated by \( N \) point masses which are modelled using the Dirac function \( \delta(\cdot) \).

Now, choosing as the positions of the steps \( N \) independent and identically distributed (i.i.d.) samples \( \{x_i\}_{i=1}^N \) from \( p(\cdot) \) in combination with step sizes of \( \frac{1}{N} \), the approximated density is given by

\[
\hat{p}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i)
\]  

(2.11)

This approximation method is illustrated by Figure 2.1.

A probability integral

\[
I(g) = E_{p(\cdot)}[g] = \int g(x)p(x)dx
\]  

(2.12)

is estimated (using the approximate density) by

\[
\hat{I}_N(g) = \int g(x)\hat{p}_N(x)dx = \frac{1}{N} \sum_{i=1}^{N} g(x_i)
\]  

(2.13)

This estimate converges almost sure to \( I(g) \) by the strong law of large numbers, see Section 3.2.1. Note that it is possible, by choosing \( g(s) = H(s - x) \) with
Figure 2.1: A 50 point approximation of the standard normal CDF, $D(\cdot)$, and PDF, $p(\cdot)$, (dotted).

$H(\cdot)$ the Heaviside step function, to let $I(g) = D(x)$. So, the CDF indeed is well approximated using the suggested step function.

Unfortunately, perfect sampling cannot be directly applied in the particle filter. Samples from $p(x_{0:t}|y_{0:t})$ have to be drawn, but that density is only known up to a proportionally constant. This problem can be circumvented using importance sampling.

2.3 Importance sampling

Importance sampling is a modification of perfect sampling (Section 2.2). The samples are drawn from the importance function $\pi(\cdot|\cdot)$, which can be chosen almost arbitrarily. The discussion which function to use is postponed to Section 2.5. The importance sampling method is based on the following argument:

If $p(x_{0:t}|y_{0:t}) > 0$ implies $\pi(x_{0:t}|y_{0:t}) > 0$ it is possible to rewrite (2.12) as

$$I(g) = \int g(x_{0:t})p(x_{0:t}|y_{0:t})dx_{0:t}$$

$$= \int g(x_{0:t}) \frac{p(x_{0:t}|y_{0:t})}{\pi(x_{0:t}|y_{0:t})} \pi(x_{0:t}|y_{0:t})dx_{0:t}$$

$$= \int g(x_{0:t}) \frac{p(x_{0:t}, y_{0:t})}{p(y_{0:t})\pi(x_{0:t}|y_{0:t})} \pi(x_{0:t}|y_{0:t})dx_{0:t}$$

$$= \frac{\int g(x_{0:t}) \frac{p(x_{0:t}, y_{0:t})}{\pi(x_{0:t}|y_{0:t})} \pi(x_{0:t}|y_{0:t})dx_{0:t}}{\int p(y_{0:t}, x_{0:t})dx_{0:t}}$$
\[
\frac{\int g(x_{0:t}) \frac{p(x_{0:t},y_{0:t})}{\pi(x_{0:t}|y_{0:t})} \pi(x_{0:t}|y_{0:t}) dx_{0:t}}{\int \frac{p(x_{0:t},y_{0:t})}{\pi(x_{0:t}|y_{0:t})} \pi(x_{0:t}|y_{0:t}) dx_{0:t}} = \frac{\int g(x_{0:t}) \tilde{w}(x_{0:t}) \pi(x_{0:t}|y_{0:t}) dx_{0:t}}{\int \tilde{w}(x_{0:t}) \pi(x_{0:t}|y_{0:t}) dx_{0:t}}
\]

(2.14)

where the importance weight, \( \tilde{w}(\cdot) \), is defined as

\[
\tilde{w}(x_{0:t}) = \frac{p(x_{0:t},y_{0:t})}{\pi(x_{0:t}|y_{0:t})}
\]

(2.15)

Now, use perfect sampling (Section 2.2) to approximate \( \pi(x_{0:t}|y_{0:t}) \): draw \( N \) i.i.d. samples, \( \{x_{0:t,i}\}_{i=1}^{N} \), from the importance function \( \pi(x_{0:t}|y_{0:t}) \) instead of the complex \( p(x_{0:t}|y_{0:t}) \). That is

\[
\hat{\pi}_{N}(x_{0:t}|y_{0:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta(x_{0:t} - x_{0:t,i})
\]

(2.16)

and using (2.13) a Monte Carlo estimate of \( I(g) \) is given by

\[
\hat{I}_{N}(g) = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}(x_{0:t,i}) g(x_{0:t,i}) = \frac{1}{N} \sum_{i=1}^{N} w_{t,i} g(x_{0:t,i})
\]

(2.17)

where the normalised importance weights \( w_{t,i} \) are defined as

\[
w_{t,i} = \frac{\tilde{w}(x_{0:t,i})}{\sum_{i=1}^{N} \tilde{w}(x_{0:t,i})}
\]

(2.18)

This estimate is biased (a ratio of two expectations), but asymptotically it converges almost sure to \( I(g) \).

The importance sampling method just explained is thus identical to using the following approximation of the posterior PDF

\[
\hat{p}_{N}(x_{0:t}|y_{0:t}) = \sum_{i=1}^{N} w_{t,i} \delta(x_{0:t} - x_{0:t,i})
\]

(2.19)

where \( x_{0:t,i} \sim \pi(x_{0:t}|y_{0:t}) \) for \( i = 1, \ldots, N \). This is a weighted version of (2.11).

This method is still unfit for use in a particle filter. It requires recalculation of the importance weights when a new observation comes available, which makes the method unfit for online estimation.

### 2.3.1 Sequential importance sampling

Importance sampling can be made recursively: it is possible to update the importance weights when a new observation comes available. The key step [7] is to
Assume \( \pi(x_{0:t}|y_{0:t}) \) to be of the following form:

\[
\pi(x_{0:t}|y_{0:t}) = \pi(x_t|x_{0:t-1}, y_{0:t}) \pi(x_{0:t-1}|y_{0:t-1}) = \pi(x_0) \prod_{s=1}^{t} \pi(x_s|x_{0:s-1}, y_{0:s})
\] (2.20)

Now,

\[
p(x_{0:t}, y_{0:t}) = p(x_t, y_t, x_{0:t-1}, y_{0:t-1}) = p(y_t|x_t, x_{0:t-1}, y_{0:t-1}) p(x_t|x_{0:t-1}, y_{0:t-1}) p(x_{0:t-1}, y_{0:t-1})
\] (2.21)

and inserting (2.20) and (2.21) in (2.15) we get a recursive expression for the importance weight

\[
\tilde{w}(x_{0:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{\pi(x_t|x_{0:t-1}, y_{0:t})} \tilde{w}(x_{0:t-1})
\] (2.22)

The steps of sequential importance sampling are summarised in Algorithm 2.1.

**Algorithm 2.1 (Sequential importance sampling (SIS))**

The approximate density \( \hat{p}_N(x_{0:t}|y_{0:t}) \) is recursively updated when a new observation comes available by performing the following steps for \( i = 1, \ldots, N \):

- update the particles
  
  \[
x_{0:t,i} = \{x_{0:t-1,i}, x_{t,i}\} \text{ with } x_{t,i} \sim \pi(x|x_{0:t-1,i})
  \] (2.23a)

- update the importance weights (up to a normalising constant)
  
  \[
w_{t,i} = \frac{p(y_t|x_{t,i})p(x_{t,i}|x_{t-1,i})}{\pi(x_{t,i}|x_{0:t-1,i}, y_{0:t})} w_{t-1,i}
  \] (2.23b)

- normalise the importance weights
  
  \[
w_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{N} w_{t,i}}
  \] (2.23c)

The sequential importance sampling method suffers a serious drawback: the variance of the importance weights can only increase over time when using an importance function of the form of (2.20) as shown by [15]. Practically, this means that after a few iteration steps all but one of the normalised weights are very close to zero. Hence it is impossible to avoid deterioration of the quality of the estimated density and ultimately the filter will diverge. The solution to this problem is a resampling step.
2.4 Resampling

A resampling step modifies the weighted approximate density $\hat{p}_N$ to an unweighted one $\tilde{p}_N$ by eliminating particles having low importance weights and by multiplying particles having high importance weights. More formally:

$$
\sum_{i=1}^{N} w_{t,i} \delta(x_{0:t} - x_{0:t,i}) \Rightarrow \sum_{i=1}^{N} \frac{n_{t,i}}{N} \delta(x_{0:t} - x_{0:t,i}) = \frac{1}{N} \sum_{j=1}^{N} \delta(x_{0:t} - x_{0:t,j})
$$

(2.24)

where $n_{t,i}$ is the number of offspring of particle $x_{t,i}$. If the resampled density is ‘close’ to the original one, in the sense that for any function $g(\cdot)$ it holds that

$$
E \left[ \left( \int g(x)\hat{p}_N(x)dx - \int g(x)\tilde{p}_N(x)dx \right)^2 \right] \xrightarrow{N \to \infty} 0
$$

(2.25)

one can prove convergence [5, 6]. There are many different methods to generate the $N_{t,i}$. An overview is provided in Chapter 3.

The resampling step solves the divergence problem, but it introduces other problems. A phenomenon known as sample impoverishment occurs. This means that the diversity of the particles is reduced due to the fact that particles with a high importance weight will be selected many times. This is especially problematic for smoothing applications, where one is interested in the density at time $t' \ll t$, since the particles there have been resampled many times.

Because of this negative effect, one wants to avoid a resampling step when it is not necessary. Besides a deterministic scheme (resample every $k$-th iteration), the effective sample size, defined by

$$
N_{\text{eff}} = \left( \sum_{i=1}^{N} w_{t,i}^2 \right)^{-1}
$$

(2.26)

can be used as a criterion [21]: apply a resampling step when it is smaller than a certain threshold.

2.5 Basic particle filter

In the previous sections all the ingredients of the basic particle filters are discussed. It consists of applying a resampling step after SIS (Algorithm 2.1) and is known in literature as Sequential Importance Sampling with Resampling (SISR) or as Bayesian bootstrap filtering.

However, one key element of the particle filter is not yet discussed: one has to choose an importance function $\pi(x_s|x_{0:s-1}, y_{0:s})$. This decision determines the performance of the method.
It is commonly assumed that \( \pi(x_s|x_{0:s-1}, y_{0:s}) = \pi(x_s|x_{s-1}, y_s) \), which implies that not the entire history of states \( \{x_{0:t-1,i}\}_{i=1}^N \) and measurements \( y_{0:t} \) has to be stored. It suffices to know \( \{x_{t-1,i}\}_{i=1}^N \) and \( y_t \). Furthermore, the resampling step is introduced to prevent the variance of the weights from increasing. Hence, it is a natural decision to choose an importance function which minimises the variance of the importance weights [7]. In many cases this optimal importance function cannot be used and one has to resolve to suboptimal functions. However, the performance of the particle filter can be greatly improved by using a good importance function.

A simple, though not optimal, choice for the importance function is to use the prior distribution, that is
\[
\pi(x_t|x_{t-1}, y_t) = p(x_t|x_{t-1}) \quad (2.27)
\]
Using this importance function, the weight update equation (2.22) can be simplified
\[
\tilde{w}(x_{0:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{\pi(x_t|x_{0:t-1}, y_{0:t})} \tilde{w}(x_{0:t-1}) = p(y_t|x_t)\tilde{w}(x_{0:t-1}) \quad (2.28)
\]
Applying a resampling step at every time step, a basic particle filter is now given by Algorithm 2.2.

**Algorithm 2.2 (Basic particle filter (SISR))**
The filtering density approximation is updated when a new observation comes available by applying the following steps: First, for \( i = 1, \ldots, N \) do

- update the particles
  \[
x_{t,i} \sim \pi(x|\cdot) \quad (2.29a)
  \]
- update the importance weights (up to a normalising constant)
  \[
w_{t,i} = p(y_t|x_{t,i})w_{t-1,i} \quad (2.29b)
  \]
- normalise the importance weights
  \[
w_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{N} w_{t,i}} \quad (2.29c)
  \]
Use the approximated density specified by \((x_{t,i}, w_{t,i})\) to calculate point estimates, e.g., mean, using (2.17).

- Apply a resampling step: replace the weighted approximation with an unweighted one by making \( n_i \) copies of particle \( x_{t,i} \) and reset the importance weights \( w_{t,i} \) to \( N^{-1} \).
2.6 Extensions

In the previous section the basic particle filter is introduced. There exist numerous extensions, which can be roughly split into two groups. Without going into detail, some common extensions are mentioned in this section.

First, there are some techniques to address the sample impoverishment problem. Regularised resampling [6] tries to decrease the sample impoverishment by using a continuous kernel approximation instead of a discrete one. There are also suggestions to replace the resampling step by Markov Chain Monte Carlo moves [6, 8] in order to generate completely new particles.

Furthermore, there are a number methods to improve the particle filter performance. Auxiliary particle filters [6, 18] modify the resampling weights to incorporate measurement information, which comes down to using an importance function that takes measurement data into account. Marginalisation (Rao-Blackwellisation) [6, 21] splits the system in a nonlinear and a linear part and applies the optimal Kalman filter to the linear part. This allows the particle filter to estimate the nonlinear part only, resulting in better estimates at lower computational costs.

2.7 Applications

Although particle filters are a relatively new method, they have been successfully applied to many different subjects. The main reason for this is that the method is applicable to very general dynamic systems. Some examples are listed below.

The first working particle filter was developed in the context of tracking objects using radars [3, 10]. In this area the method is now used in industrial applications.

Car positioning [9, 11] is another commercialised area. Here, relative position information from wheel speed sensors is combined with digital map information to obtain the position of a car. This setup gives similar or better functionality than when using GPS.

Particle filters have been used for audio source separation [1], in econometrics, see e.g. [6, 18] and in visualisation [6].

This list is far from complete, but it gives an indication of the wide range of possible applications for particle filters.
Particle filters
Chapter 3

Resampling algorithms

The resampling step in particle filters is a computational intensive one. Therefore, a justified decision regarding which resampling algorithm to use might result in a reduction of the overall computational effort. However, the quality of the resampling should not be influenced in a negative way. In this chapter implementations of different resampling algorithms found in literature are compared in order to facilitate a good decision of which resampling algorithm to choose.

3.1 Introduction

In particle filter literature quite a few resampling algorithms are used, however they are scattered among different papers and a thorough comparison is missing.

From the literature four ‘basic’ resampling algorithms can be identified, all of which can be used in the basic (SISR) particle filter. Modifications to these four basic algorithms, like regularised resampling (see Section 2.6), are not discussed here.

These four resampling algorithms are all based on multinomial selection of $n_i$, which is equivalent to selecting with replacement $N$ particles $x_j^*$ from the original set of particles $\{x_i\}$ where $P(x_j^* = x_i) = w_i$, $i, j = 1, \ldots, N$. In probability theory the selection of $x_j^*$ is called an event, the result of the $j$-th experiment. It is denoted by $I_j = i$. The standard method to generate an event [19] is given by Algorithm 3.1 below.

Algorithm 3.1 (Event generation)
The generation of an event $I_k$ with $P(I_k = i) = w_i$ can be carried out by

- Simulate a standard uniform random number $u_k \sim U[0, 1)$
- Assign $I_k$ the value $i$ according to $Q_{i-1} < u_k \leq Q_i$ where $Q_i = \sum_{s=1}^{i} w_s$.

Note that this procedure uses the generalised inverse of the cumulative density function (CDF), see also Section 3.2.1 to map standard uniform numbers to events. The event generation is graphically illustrated by Figure 3.1 which shows how the inverse CDF is used to generate events.
Events generated using this method can easily be used to determine the resampled particles by treating the events as indices of the old particles.

Before discussing the resampling algorithms, the principles that form the basis for these resampling algorithms are introduced. This theory provides a framework to be able to understand how the selected resampling algorithms work and what their differences are.

### 3.2 Principles

In Section 2.4 the resampling step is introduced. There it is argued that the resampled density should be close to the original one:

$$\sum_{i=1}^{N} w_i \delta(x - x_i) \approx \sum_{i=1}^{N} \frac{n_i}{N} \delta(x - x_i) \quad (3.1)$$

The measure $E[(n_i - Nw_i)^2]$ indicates the distance between the densities for one peak. So, intuitively, if these distances can be reduced for all peaks the densities are closer to each other and hence the resampling is improved. When the $n_i$'s are unbiased, that is $E n_i = Nw_i$, then the measure is identical to $\text{Var} n_i$. All resampling algorithms discussed are unbiased, but differ in terms of variance. Based on the above argument, low variance of $n_i$ is favourable for a resampling algorithm.

The resampling algorithms use two different approaches to achieve variance reduction. Both approaches are discussed in the subsequent sections.
3.2 Principles

3.2.1 Sample generation

For resampling algorithms based on Algorithm 3.1, the number of offsprings $n_i$ from particle $x_i$ is determined by counting the times $x_i$ is selected:

$$n_i = \sum_{j=1}^{N} \mathbb{1}_{x_j^*=x_i} = \sum_{j=1}^{N} g_i(x_j^*) = N\hat{I}(g_i)$$  \hspace{1cm} (3.2)

where $g = \mathbb{1}(\cdot)$ is the indicator function and $\hat{I}(g_i)$ is a MC estimate of the integral

$$I(g_i) = \mathbb{E}[g_i] = \sum_{j=1}^{N} g_i(x_j^*)P(X = x_j^*) = P(X = x_i) = w_i$$  \hspace{1cm} (3.3)

If $\hat{I}(g_i)$ is an unbiased MC estimate then the $n_i$ are that as well since $\mathbb{E}[n_i] = N\mathbb{E}[\hat{I}(g_i)] = N I(g_i) = N w_i$.

The observation that the $n_i$ are MC estimates allows for variance reduction by means of sample generation techniques. The applicability of these techniques to process noise generation is discussed in Appendix A. The techniques are very general methods and will be discussed as such.

Introduction

In the next sections a number of algorithms to generate sequences of random samples are discussed. More specific, algorithms to generate $N$ random vectors $x_i \in \mathcal{D} \subseteq \mathbb{R}^d, \; i = 1 \ldots N$ distributed according to a given probability density $p(x)$ on the domain $\mathcal{D}$. These generated sequences should represent the given distribution as closely as possible. More formally, for any function $g(\cdot)$, the difference between

$$I(g) = \mathbb{E}[g] = \int_{\mathcal{D}} g(x)p(x)dx$$  \hspace{1cm} (3.4)

and an estimate $\hat{I}(g)$ based on random samples should be minimal. When $\hat{I}$ is unbiased, the Monte Carlo variance, $\text{Var} \hat{I} = \mathbb{E}[(\hat{I} - I)^2]$, is a measure of this distance. So, sample generation is closely related to Monte Carlo integration theory. Note also the similarity to survey sampling, where samples are draw from a population in order to estimate properties of an unknown distribution.

Typically, samples according to some density are generated using the following method: generate samples from the uniform distribution in the unit cube $[0, 1)^d$ and subsequently apply a transformation. This requires that the dimensions should satisfy $d' \geq d$ and also the domain of (3.3) has to be modified to $[0, 1)^{d'}$. Further should $g$ be modified to incorporate the transformation. Although for a general distribution this transformation method might not be possible, almost all common distributions can be generated this way, see for instance [19], and when $d = 1$ the generalised inverse $D^-(u) = \inf\{x; D(x) \geq u\}$ of the cumulative density function $D(x) = P(X \leq x)$ can be used, at least theoretically, as transformation function. Considering this, the subsequent sections concentrate on generating samples $x_i \sim \text{U}(0,1)^d$.  

Simple random sampling

Using simple random sampling, $N$ independent random samples are generated from $U[0,1)^d$. Then, using an approximation of the probability distribution as in Section 2.2, the estimate of the integral (3.4) is given by

$$\hat{I}_{sr}(g) = \frac{1}{N} \sum_{i=1}^{N} g(x_i), \quad x_i \sim p(x)$$

This estimate is unbiased since

$$E[\hat{I}_{sr}(g)] = \frac{1}{N} \sum_{i=1}^{N} E[g(x_i)] = \frac{N}{N} E[g(x)] = I(g)$$

and its variance is given by

$$\text{Var} \hat{I}_{sr}(g) = \frac{1}{N^2} \text{Var} \sum_{i=1}^{N} g(x_i) = \frac{1}{N^2} \sum_{i=1}^{N} \text{Var} g(x_i) = \frac{\text{Var} g}{N}$$

Hence, by the strong law of large numbers, the estimate $\hat{I}(g)$ converges almost sure to $I(g)$ as $N \to \infty$.

These results are derived using the additive and multiplicative properties of the expectation and variance operators for independent variables and they form the basis of Monte Carlo integration theory.

Stratified sampling

Simple random sampling can been viewed as a random sequence of values from a random set. This perspective allows for methods that might reduce MC variance as the values in the set might be chosen differently. A method based on this point of view is stratified sampling.

Stratified sampling is a method that originated from survey sampling [4] that leads to MC variation reduction. The domain is partitioned into different strata, that is $\mathcal{D} = \bigcup_{j=1}^{p} \mathcal{D}_j$ where $\mathcal{D}_k \cap \mathcal{D}_l = \emptyset$ for $k \neq l$. This allows (3.4) to be decomposed into

$$I(g) = \int_{\mathcal{D}} g(x)p(x)dx = \sum_{j=1}^{p} \int_{\mathcal{D}_j} g(x)p(x)dx = \sum_{j=1}^{p} \rho_j \int_{\mathcal{D}_j} g(x)p_j(x)dx = \sum_{j=1}^{p} \rho_j I_{p_j}(\cdot)(g)$$

where the weights $\rho_j$ are the probabilities of the regions $\mathcal{D}_j$ (hence $\sum \rho_j = 1$) and $p_j(\cdot)$ the normalised restrictions of the density $p(\cdot)$ to these regions. Combining
independent simple random samples within every stratum, an estimator is given by
\[
\hat{I}_s(g) = \sum_{j=1}^{p} \rho_j \hat{I}_{sr_j}(g) = \sum_{j=1}^{p} \frac{\rho_j N_j}{N} \sum_{i=1}^{g(x_{ji})} (3.9)
\]
where \(\sum N_j = N\). As the estimator within every strata is unbiased, see (3.6), so is the combined one, for
\[
E[\hat{I}_s(g)] = \sum_{j=1}^{p} \rho_j E[\hat{I}_{sr_j}(g)] = \sum_{j=1}^{p} \rho_j E[\hat{I}_{sr_j}(g)|x \in D_j]P(x \in D_j)
\]
\[
= E\left[E[\hat{I}_{sr_j}(g)|x \in D_j]\right] = E[\hat{I}_{sr_j}(g)] = I(g) (3.10)
\]
Its variance is given by
\[
\text{Var} \hat{I}_s(g) = \sum_{j=1}^{p} \rho_j^2 \text{Var} I_{sr_j}(g) = \sum_{j=1}^{p} \frac{\rho_j^2 N_j}{N} \text{Var}_{p_j(\cdot)} g
\]
Here we have used the property that every stratum is independently sampled. Typically, the number of samples allocated in each stratum \(N_j\) is subject to optimisation. Optimal allocation is achieved with the Neyman allocation \([4]\]
\[
N_j \propto \rho_j \text{Var}_{p_j(\cdot)} g
\]
However, unlike survey sampling or MC integration, the function \(g(\cdot)\) is arbitrary in sample generation. Therefore, consider proportional allocation \(N_j \propto \rho_j\). A very general performance result can be derived for this allocation.

As the stratum of \(x\) is a random variable itself, a classical result in probability theory can be used
\[
\text{Var} g = E[\text{Var}(g(x)|x \in D_j)] + \text{Var}(E[g(x)|x \in D_j])
\]
\[
= \sum_{j=1}^{p} \rho_j \text{Var}_{p_j(\cdot)} g + \sum_{j=1}^{p} \rho_j (E_{p_j(\cdot)} g - I(g))^2 (3.12)
\]
Thus, using (3.12) for proportional allocation, that is \(N_j = N \rho_j\), the following inequality holds
\[
\text{Var} \hat{I}_s(g) = \sum_{j=1}^{p} \frac{\rho_j^2}{N_j} \text{Var}_{p_j(\cdot)} g = \frac{1}{N} \sum_{j=1}^{p} \rho_j \text{Var}_{p_j(\cdot)} g \leq \frac{\text{Var} g}{N} = \text{Var} \hat{I}_{sr}(g) (3.13)
\]
In words, the MC variance when using stratified sampling with proportional allocation does not increase with respect to simple random sampling. A reduction, however, is possible. Note that these statements hold regardless of the sample space dimension and the number of samples.

When applying stratified sampling to \(U[0,1]^d\), the theory of uniform distribution, see for instance \([17]\), gives a very intuitive explanation why this method works. This theory is based on the Koksma-Hlawka inequality \([13]\]
\[
|\hat{I}(g) - I(g)| \leq D_N^*(x_1, \ldots, x_N)V_{HK}(g) (3.14)
\]
where $V_{HK}(g)$ is the total variation of $g$ in the sense of Hardy and Krause and the star discrepancy $D^*$ is defined as

$$D^*_N(x_1, \ldots, x_N) = \sup_{a \in [0,1)^d} \left| \frac{1}{N} \sum_{i=1}^{N} 1_{(0,a]}(x_i) - |[0,a)| \right|$$

(3.15)

where $1$ is the indicator function defined by

$$1_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

(3.16)

and $|\cdot|$ denotes volume. The discrepancy $D^*$ is a numerical measure of how uniform a set of points is distributed in the unit cube. It compares the fraction of points in an anchored box to the volume of this box. Clearly this difference will be smaller when the points are more uniform distributed.

Now, the Koksma-Hlawka inequality (3.14) relates a smaller discrepancy to better integration accuracy. In practice this inequality cannot be used to determine accuracy bounds since both the variation and the discrepancy are almost impossible to evaluate. However it provides an explanation to why stratified sampling improves the MC variance: compared to simple random sampling, the expected discrepancy of the set of integration points when using stratified sampling is smaller.

Using this alternative explanation, stratified sampling can be enhanced by choosing the strata such that the discrepancy is reduced. For the one dimensional case this can be achieved by dividing the interval in $N$ strata of equal size $\left[ \frac{j}{N}, \frac{j+1}{N} \right)$ for $j = 0, \ldots, N-1$ and drawing a single random sample from each interval. The returned sequence is a random permutation of the stratified samples. That is

$$x_j = \pi_{j-1} + \frac{u_j}{N}, \quad u_j \sim U[0,1), \ j = 0, \ldots, N-1$$

(3.17)

where $\pi$ is a uniform random permutation of $0, \ldots, N-1$, where uniform means that all $n!$ possible orderings have the same probability.

When the stratification is applied to the multidimensional case it becomes less efficient. Typically, dividing the $d$ dimensional unit cube in $N$ strata gives $\sqrt{N}$ intervals per dimension. Hence the discrepancy reduction decreases with higher dimensions, especially for ‘low’ $N$. Latin hypercube sampling [17, 20] and its extensions provides a way to reduce this problem.

**Systematic sampling**

The stratification method in (3.17) tries to reduce the discrepancy of the samples by choosing the strata and their number of samples effectively. Using the following method the discrepancy is minimised:

$$x_j = \frac{\pi_{j-1} + u}{N}, \quad u \sim U[0,1), \ j = 0, \ldots, N-1$$

(3.18)
This method is known in survey sampling as systematic sampling [4]. The interval is divided in $N$ strata and one sample is taken from every stratum as in stratified sampling, but the samples are no longer independent: all the samples have the same position within a stratum. This gives the minimal discrepancy for $N$ samples.

Note that the variance results from (3.11) and (3.13) don’t hold for systematic sampling since the strata are not sampled independently, though the estimates are unbiased. This prevents a variance based comparison, but the systematic sampling is expected to perform very good as it has minimal discrepancy and hence is optimal according to (3.14).

Overview

Some different methods to generate sequences of uniform random numbers have been discussed. It is argued that the methods differ in discrepancy $D^*$, a measure of uniformity, and hence by (3.14) in how well the samples represent the standard uniform distribution.

Samples from the different methods - simple random sampling, stratified sampling and systematic sampling - are shown in Figure 3.2.

![Figure 3.2: 10 standard uniform samples generated using simple random sampling (x), stratified sampling (+) and systematic sampling (·).](image)

The effect of stratification on the discrepancy is clearly shown. The stratified and systematic samples are more uniform than the simple random one. Note also the difference between the stratified sample and the systematic sample: although (3.17) and (3.18) differ only subtle, the systematic samples are more uniform.

Variance results (3.13) for the estimates using the sampling methods confirm the improvement of stratified sampling over simple random sampling. Though not confirmed by a variance analysis, systematic sampling is better than stratified sampling according to (3.14).

3.2.2 Set restriction

Set restriction is the second technique used to reduce the variance of $n_i$. The reasoning is very simple: variance is defined by

$$\text{Var } x = E[(x - E x)^2]$$  \hspace{1cm} (3.19)
which applied to $n_i$ gives

$$\text{Var } n_i = \sum_{n \in S_{n_i}} (n - E n_i)^2 P(n_i = n)$$

(3.20)

In the case that the $n_i$ are distributed according to the multinomial distribution, the set of possible values for every $n_i$ is given by $S_{n_i} = \{0, \ldots, N\}$. By restricting this set to values which lie closer to the (original) $E n_i$, the variance is reduced. Changing the possible values of $n_i$ implies that the density is also changed. Therefore, the modification should such that $E n_i$ remains constant.

### 3.3 Overview

In the particle filter literature four ‘basic’ resampling algorithms can be identified which are commonly used. These are simple random resampling, stratified resampling, systematic resampling and residual resampling. They are all unbiased, but as different principles from Section 3.2 are used they differ in terms of variance.

In the next sections, an overview of the resampling algorithms is given.

#### 3.3.1 Simple random resampling

Simple random resampling was introduced in the original bootstrap filter by [10]. This algorithm applies Algorithm 3.1 $N$ times to generate $N$ new particles. Thus the events $I_k$ are generated using i.i.d. standard uniform samples, which can be classified as simple random sampling (Section 3.2.1). Hence the name of this resampling algorithm.

To implement the find operation of Algorithm 3.1 efficiently for $N$ random numbers, these have to be ordered. A basic algorithm is now given by

**Algorithm 3.2 (Simple random resampling)**

- Draw $N$ uniform random numbers
  $$\tilde{u}_k \sim U[0, 1]$$

- Obtain ordered random numbers $u_k$ by sorting the $\tilde{u}_k$

- Allocate $n_i$ copies of the particle $x_i$ to the new distribution

  $$n_i = \text{the number of } u_k \in \left( \sum_{s=1}^{i-1} w_s, \sum_{s=1}^i w_s \right)$$

The ordering in random numbers for this algorithm gives rise to an ordering in the resampled distribution, but that influences neither the particle filter nor the values of $n_i$. An efficient sorting algorithm requires $O(N \log N)$ operations to order the random numbers. One can generate ordered uniform random numbers using the inverse CDF method in $O(N)$ operations [7]. This method is preferable since it is computationally more efficient from a certain number of samples.
Ordered uniform numbers are obtained by sorting the i.i.d. variables \( v_k \sim U[0, 1) \). The CDFs of \( u_k \) are given by

\[
P(u_N < u) = P(v_1 < u \cap \ldots \cap v_N < u) = u^N \tag{3.21a}
\]

\[
P(u_k < u|u_{k+1}) = P(\tilde{v}_1 < u \cap \ldots \cap \tilde{v}_k < u) = \left( \frac{u}{u_{k+1}} \right)^k \tag{3.21b}
\]

Here \( \tilde{v} \) are the \( k \) variables satisfying \( v < u_{k+1} \). So, \( \tilde{v} \sim U[0, u_{k+1}) \). The inverse CDF method is used to generate ordered random uniform numbers \( u_k \) from i.i.d. uniform random numbers \( \tilde{u}_k \). Then the \( u_k \) are transformed to an index of a particle using Algorithm 3.1. The efficient simple resampling algorithm requires \( \mathcal{O}(N) \) operations and consists of the following steps:

**Algorithm 3.3 (Simple random resampling (efficient))**

- Draw \( N \) ordered uniform random numbers
  \[
u_k = u_{k+1} \tilde{u}_k^i, \ u_N = \tilde{u}_N^i \text{ with } \tilde{u}_k \sim U[0, 1)\]

- Allocate \( n_i \) copies of the particle \( x_i \) to the new distribution
  \[
n_i = \text{the number of } u_k \in \left( \sum_{s=1}^{i-1} w_s, \sum_{s=1}^{i} w_s \right)\]

### 3.3.2 Stratified resampling

Stratified resampling \([6, 14]\) applies stratified sampling as described in Section 3.2.1. More specifically, the implementation uses \( (3.17) \).

As with simple random resampling, this algorithm is based on Algorithm 3.1. By replacing the random permutation \( \pi \) in \( (3.17) \) with the sequence \( 0, \ldots, N - 1 \) the random numbers are ordered, which allows an \( \mathcal{O}(N) \) implementation using the same argument as in Section 3.3.1. The algorithm consists of

**Algorithm 3.4 (Stratified resampling)**

- Draw \( N \) ordered random numbers
  \[
u_k = \frac{(k - 1) + \tilde{u}_k}{N} \text{ with } \tilde{u}_k \sim U[0, 1)\]

- Allocate \( n_i \) copies of the particle \( x_i \) to the new distribution
  \[
n_i = \text{the number of } u_k \in \left( \sum_{s=1}^{i-1} w_s, \sum_{s=1}^{i} w_s \right)\]

### 3.3.3 Systematic resampling

Systematic resampling \([2, 14]\) uses algorithm 3.1 in combination with systematic sampling \( (3.18) \). As this sampling method is very similar to stratified sampling
Resampling algorithms

(3.17), the same steps as with stratified resampling are made to construct this resampling algorithm: the random permutation \( \pi \) is replaced with the sequence \( 0, \ldots, N-1 \) to allow an \( \mathcal{O}(N) \) implementation. The resulting algorithm is therefore also very similar to stratified sampling. It is given by

**Algorithm 3.5 (Systematic resampling)**

- Draw \( N \) ordered numbers
  
  \[
  u_k = \frac{(k-1) + \tilde{u}}{N} \text{ with } \tilde{u} \sim \text{U}[0,1]
  \]

- Allocate \( n_i \) copies of the particle \( x_i \) to the new distribution
  
  \[n_i = \text{the number of } u_k \in \left(\sum_{s=1}^{i-1} w_s, \sum_{s=1}^{i} w_s\right)\]

The differences between stratified and systematic sampling are discussed in Section 3.2.1.

### 3.3.4 Residual resampling

Residual resampling [16] uses a somewhat different approach to resample. Residual sampling is based on set restriction instead of sampling methods. The idea is that a large part of the number of offspring \( n_i \) can be determined without resorting to random numbers. This can be achieved by taking the integer part of \( N w_i \). To retain the original population size some more copies need to be made. These residual particles are randomly selected with replacement from the original particles using modified weights. The method just described consists of

**Algorithm 3.6**

- Allocate \( n_i' = \lfloor N w_i \rfloor \) copies of particle \( x_i \) to the new distribution

- Additionally, resample \( m = N - \sum n_i' \) particles from \( \{x_i\} \) by making \( n_i'' \) copies of particle \( x_i \) where the probability for selecting \( x_i \) is proportional to \( w_i' = N w_i - n_i' \) using one of the resampling schemes mentioned earlier.

This algorithm indeed gives an unbiased resampled distribution since

\[
E[n_i] = E[n_i' + n_i''] = n_i' + N w_i - n_i' = N w_i
\]  \hspace{1cm} (3.22)

Residual resampling can be implemented in \( \mathcal{O}(N) \), since all the previously mentioned resampling algorithms and the additional operations are of \( \mathcal{O}(N) \).

### 3.4 Comparison

In this section the resampling algorithms are compared. Two important factors in determining the performance of a resampling algorithm are its resampling quality and its computational complexity. Both factors will be discussed in the next sections.
3.4.1 Resampling Quality

Resampling quality can be defined in a number of ways. From a theoretical argument (see Section 3.2), the variance of \( n_i \) is a quality indicator for a resampling algorithm. A more practical approach is to relate resampling quality to the quality of the estimates of a particle filter. This relation is very complex since many other factors influence the quality of the estimates as well. Intuitively, improved variance should also give improved estimates, but that remains to be confirmed using simulations (see Chapter 4).

The resampling algorithms differ in terms of variance. Using the theoretical framework in Section 3.2, some statements regarding this variance can be made.

Simple random resampling, stratified resampling and systematic resampling apply different sampling techniques. As the \( n_i \) are identified as MC estimates, the theory of Section 3.2.1 can be used to compare the algorithms. Stratified resampling and systematic resampling have lower variance than simple random resampling. Also, systematic resampling is expected to perform better than stratified resampling.

Residual resampling uses set restriction (see Section 3.2.2) to reduce variance compared to simple random resampling. The probability space is modified in such a way that now \( n_i \in \{ \lfloor Nw_i \rfloor, \ldots, N \} \subset \{0, \ldots, N\} \), but still \( \mathbb{E}[n_i] = Nw_i \).

Set restriction is used in systematic resampling as well. A line segment of length \( \ell \) always contains \( \lfloor N\ell \rfloor \) points placed a distance \( N^{-1} \) apart and at most one point more. Hence, systematic resampling restricts the set values of \( n_i \) to \( \{ \lfloor Nw_i \rfloor, \lfloor Nw_i \rfloor + 1 \} \). As systematic resampling is unbiased (see Section 3.2.1), it has minimal variance for \( n_i \).

By the argument that reduced variance of \( n_i \) gives better resampling, stratified resampling, systematic resampling and residual resampling should be better than simple resampling as they use variance reducing methods. Systematic resampling has minimal variance and is therefore the most favourable resampling algorithm.

Note that these conclusions hold independently of the number of particles \( N \).

3.4.2 Computational complexity

Comparing algorithms based on computational complexity is very difficult. This is due to system architecture, memory management and programming language, just to mention a few factors in the complex computer environment. Therefore only a qualitative analysis is made.

The simple, stratified and systematic resampling algorithms are very similar. They only differ in how the ordered sequence of numbers is generated. As all the algorithms are \( \mathcal{O}(N) \), it is sufficient to compare them based on the complexity of the operations for one element. Table 3.1 shows which operations are required to obtain one element of these sequences. Clearly the algorithms have different computational complexity, especially when taking into account that fractional power and random number generation are more complex operations than addition/subtraction or multiplication/division. Simple random resampling is the most expensive resampling algorithm, followed by stratified resampling and finally systematic resampling.
Resampling algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Formula</th>
<th>$a + b$</th>
<th>$a \times b$</th>
<th>$a^b$</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple random</td>
<td>$u_k = u_{k+1} \tilde{u}_k^{k-1}$</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stratified</td>
<td>$u_k = ((k - 1) + \tilde{u}_k)N^{-1}$</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Systematic</td>
<td>$u_k = ((k - 1) + \tilde{u})N^{-1}$</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1: The number of operations required for the generation of one $u_k$ for different resampling algorithms.

Residual resampling is more difficult to place. Experiments show that approximately $N/2$ particles are determined deterministically, leaving the other half to be determined using one of the algorithms discussed before. This complexity reduction is cancelled by the recalculation of the weights and other preparations. So, simulations have to point out which position residual resampling has.
Chapter 4

Simulations

In Chapter 3, four resampling algorithms are introduced and, based on a theoretical framework, a comparison is made in terms of resampling quality and computational complexity. The results of this comparison are in this chapter validated using simulations. To do so, the behaviour of the resampling algorithms is studied for:

1. The variance of $n_i$
2. The quality of particle filter estimates
3. The computational complexity

In the next sections the simulation results are discussed. The simulations are performed with MATLAB using the implementations of the resampling algorithms shown in Appendix B.

4.1 Resampling variance

The variance of $n_i$ is an indicator of the resampling quality for a resampling algorithm. In Section 3.4, the variance properties of the resampling algorithms are compared based on the theoretical framework of Section 3.2. To verify the results of this comparison the variance properties of the algorithms are investigated using the method discussed in the next section.

4.1.1 Method

The resampling variance of the different resampling algorithms can be compared by means of the likelihood function of the multinomial distribution [12]. The multinomial distribution (similar to the binomial distribution) arises by counting the number of times the mutually exclusive events $I_k = i$ occur in $N$ independent experiments. These events have probability $P(I_k = i) = w_i$. The joint distribution of $N_i$ (the number of times the $i$ is selected) is then given by

$$P(N_1 = n_1, \ldots, N_n = n_n) = \frac{N!}{n_1! \cdots n_n!} w_1^{n_1} \cdots w_n^{n_n}$$  (4.1)
where $\sum_{i=1}^{n} n_i = N$. This likelihood function has a useful property which can be used to compare the different resampling algorithms.

**Theorem 4.1**
The multinomial likelihood function \( (4.1) \) attains its global maximum at \( n_i = N w_i \) for \( i = 1, \ldots, N \).

**Proof**
First the logarithm of the likelihood is taken, as this simplifies things drastically. As the logarithm is a strictly increasing function on \( R^+ \) the position of the maximum doesn’t change using this transformation. So,

\[
f = \ln \left( \frac{N!}{n_1! \cdots n_n!} w_1^{n_1} \cdots w_n^{n_n} \right) = \ln N! + \sum_{i=1}^{n} \left( -\ln(n_i!) + n_i \ln(w_i) \right)
\]

(4.2)

Now, substitute \( \ln n! \approx n \ln n - n \). Then, using the method of Lagrange multipliers (see any textbook on optimisation) under the constraint \( g = \sum n_i = N \), for \( i = 1, \ldots, N \)

\[
\frac{\partial f}{\partial n_i} + \lambda \frac{\partial g}{\partial n_i} = 0 \\
- \ln n_i + \ln w_i + \lambda = 0 \\
n_i = e^\lambda w_i = N w_i
\]

(4.5)

In the last step \( \lambda \) is determined using the constraint equation.

□

Although only the \( n_i \) from simple random resampling are distributed according to the multinomial distribution, the multinomial likelihood can be used to compare all resampling algorithms. Algorithms with lower MC variance have, on average, their \( n_i \) closer to \( N w_i \). Hence, their average values of the multinomial likelihood are also higher. This provides an easy method to compare the MC variances of the different resampling algorithms.

**4.1.2 Results**
The likelihood value of the resampled density resulting from the resampling algorithms is calculated. Normalised weight sequences of uniform random numbers are used as original densities. Each algorithm is applied 100 times to a sequence and the mean value of the likelihood is calculated.

Figure 4.1 clearly indicates the difference in likelihood between simple resampling and the other algorithms. Simple random resampling has always a lower likelihood value than the other resampling algorithms. This implies a variance reduction for stratified, systematic and residual resampling, which confirms the theory as discussed in Section 3.4.1. Note that systematic resampling is always the upper line, which implies that it has the lowest variance. Similar results are observed using 10, 40 or 80 particles (not shown). The factorials in (4.1) make it impossible to calculate the likelihood accurately for larger number of particles.
4.2 Particle filter estimates

Besides resampling variance, there is another quality indicator for resampling quality. As argued in Section 3.4.1, the quality of the particle filter estimates can be used as an indicator for resampling quality. The relation is complex and lots of other factors influence the particle filter estimates as well, which makes a theoretical comparison of the resampling algorithms very hard. Intuitively, improved resampling is likely to give better particle filter estimates. At the very least, no deterioration of the quality of the estimate should occur. Using simulations these expectations are verified using the method discussed in the next section.

4.2.1 Method

As measure of the quality of the particle filter estimates of the posterior PDF is given by the root mean square error (RMSE). That is the distance between a point estimate of the states of the system, calculated using the approximated density, and the actual states, which are known in a simulation.

Two systems, both known in the literature, are used to compare the different resampling algorithms on this measure.

Figure 4.1: The mean value of the multinomial likelihood over 100 simulations is shown for 20 random weight sequences of 20 particles.

However, as the simulations confirm the theory for the mentioned number of particles and as the theory is derived independently of the number of particles, it is argued that the variance reduction holds also for higher number of particles.
System A (Univariate non-stationary growth model)
Consider the following nonlinear system

\[
\begin{align*}
x_{t+1} &= \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8 \cos 1.2 t + n_t \\
y_t &= \frac{x_t^2}{20} + \nu_t
\end{align*}
\] (4.6a)

with \(x_0 \sim N(0, 5), n_t \sim N(0, 10)\) and \(\nu_t \sim N(0, 1)\) mutually independent Gaussian noise sequences. This system has been analysed in many publications, e.g. [2, 10], mostly as an example where the (Extended) Kalman filter fails.

Figure 4.2: A typical run of System A. It shows the true state (line) and the evolution in time of the probability distribution (pixels) according to the particle filter.

Figure 4.2 clearly shows the bimodal character of the probability density, which can be attributed to the squaring in equation (4.6b). Hence ‘classical’ point estimators like the mean or maximum a posteriori perform badly, since they either indicate a point somewhere between the two modes or they indicate a point in one of the modes. Therefore, as the density is more or less symmetric around zero for this system, the quality indicator is defined as the Root Mean Square Error (RMSE) ignoring sign. That is, the error in the RMSE is defined as \(e = |x| - E[|x|]\).
4.2 Particle filter estimates

System B (Tracking)

Consider the following 2D tracking model of an aircraft

\[
x_{t+1} = \begin{bmatrix}
1 & 0 & T & 0 & 0 & 0 \\
0 & 1 & 0 & T & 0 & 0 \\
0 & 0 & 1 & 0 & T & 0 \\
0 & 0 & 0 & 1 & 0 & T \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} x_t + n_t
\]

\[(4.7a)\]

\[
y_t = \left[ \sqrt{p_x^2 + p_y^2} \right] + \nu_t
\]

\[(4.7b)\]

with \( x = [p_x \ p_y \ v_x \ v_y \ a_x \ a_y]^T \), \( n_t \sim N(0, Q) \) and \( \nu_t \sim N(0, R) \) mutually independent Gaussian noise sequences. The simulation parameters are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>1</td>
<td>sample time</td>
</tr>
<tr>
<td>( x_0 )</td>
<td>[2000, 2000, 20, 20, 0, 0]^T</td>
<td>initial position</td>
</tr>
<tr>
<td>( P_0 )</td>
<td>diag[4, 4, 16, 16, 0.04, 0.04]</td>
<td>Cov ( x_0 )</td>
</tr>
<tr>
<td>( Q )</td>
<td>diag[4, 4, 4, 4, 0.01, 0.01]</td>
<td>Cov ( n_t )</td>
</tr>
<tr>
<td>( R )</td>
<td>diag[100, 10^{-6}]</td>
<td>Cov ( \nu_t )</td>
</tr>
</tbody>
</table>

Table 4.1: Parameters for System B

This system differs from System A with respect to dimensionality. Both the states and the measurements are multidimensional. The a posteriori distribution has only one mode, and hence the RMSE of \( E[x] \) is used as a performance indicator.

4.2.2 Results

The resampling algorithms are applied in the standard SISR particle filter (Algorithm 2.2) on the two systems described in Section 4.2.1. Each resampling algorithm has been run with different numbers of particles while using the same simulated measurement data. All combinations have been simulated 100 times.

Figure 4.3 shows the average RMSE for the two systems discussed in Section 4.2.1. For System A (Figure 4.3(a)), there is minimal difference between the estimates of the different resampling algorithms. However, for System B (Figure 4.3(b)), simple random resampling, which has highest variance, performs on average worse than the other resampling algorithms and this phenomenon is not only observed in the velocity estimates, but also in the position and acceleration estimates (not shown).

These simulations show that using a different resampling algorithm than simple random resampling might increase the quality of the estimates of the particle filter. Although improvements are not guaranteed using theoretical favourable resampling algorithms, some simulations show on average better estimates and a
Figure 4.3: Average RMSE values as a function of the number of particles for the two systems described in Section 4.2.1.
deterioration of quality of the estimates is not encountered. These results confirm the argumentation that better resampling quality in the sense of reduced variance also gives improved estimates.

4.3 Computational complexity

The computational complexity is the second factor the resampling algorithms are compared on. Some qualitative statements are made in Section 3.4.2. In this section these statements are verified using simulations.

4.3.1 Method

The complexity of the algorithms can easily be compared using their execution times. These are of course machine and implementation dependent, so the values should be treated with caution. However, general complexity trends can still be identified.

4.3.2 Results

The computational complexity of the resampling schemes is investigated by measuring the time required to perform resampling of a random weight sequence. To minimise computer system intervention, the minimum value found is used.

Figure 4.4 shows the measured times each resample algorithm requires on two computer systems. The complexity of all the implementations is indeed $O(N)$ since all lines are linear in $N$. Furthermore, simple random resampling and residual resampling take significantly longer than stratified resampling and systematic resampling, as anticipated in Section 3.4.2. Thus, to reduce computational complexity, the stratified resampling and systematic resampling algorithms are favourable, where the latter is slightly better.
Figure 4.4: Computational effort as a function of the number of particles.
Chapter 5

Concluding remarks

5.1 Conclusions

In the particle filter literature four resampling algorithms are commonly used. These are: simple random resampling, stratified resampling, systematic resampling and residual resampling. In this report, these resampling algorithms are compared with respect to resampling quality and computational complexity. Based on a theoretical analysis statements can be made about the performance of the resampling algorithms on both factors. These statements are verified using MC simulations and they are found to hold.

Theoretically, better resampling relates to lower variance of the number of copies from a particle. Systematic resampling is the algorithm with the lowest resampling variance. Particle filter estimates may improve slightly from lower resampling variance and a deterioration is not encountered.

The resampling algorithms differ a lot in computational complexity. Systematic resampling has the lowest computational complexity.

5.2 Recommendations

Considering resampling quality and computational complexity, systematic resampling is favourable. Compared to the other resampling algorithms, it reduces computational complexity while giving identical or even better particle filter estimates. Also, from a theoretical perspective this resampling algorithm is superior.

This report doesn’t discuss whether the resampling algorithms meet the necessary conditions for convergence of the particle filter. For simple random resampling convergence can be proved. Since the other resampling algorithms are derived from this algorithm and since they have been applied successfully many times, it is likely that these conditions are met. However, a formal proof might be a relevant addition to this report.

Another direction for future work might be to investigate other resampling algorithms.
Bibliography


Notation

Operators and functions

diag(·) diagonal elements of a matrix
\( f(\cdot) \) system equation
\( h(\cdot) \) measurement equation
\( \inf(\cdot) \) infimum
\( p(\cdot) \) probability density function
\( p(\cdot, \cdot) \) joint probability density function
\( p(\cdot|\cdot) \) conditional probability density function
\( \hat{p}_N(\cdot) \) \( N \) point approximation of \( p(\cdot) \)
\( \tilde{p}_N(\cdot) \) \( \hat{p}_N(\cdot) \) after resampling
\( \sup(\cdot) \) supremum
\( \tilde{\omega}(\cdot) \) importance weight function
\( D(\cdot) \) cumulative density function
\( D_N(\cdot) \) \( N \) point approximation of \( D(\cdot) \)
\( E[\cdot] \) expectation
\( H(\cdot) \) Heaviside step function
\( I(\cdot) \) probability integral
\( \hat{I}_N(\cdot) \) \( N \) point MC approximation to \( I(\cdot) \)
\( \mathcal{O}(\cdot) \) order symbol
\( P(\cdot) \) probability
\( \text{Var}(\cdot) \) variance
\( \delta(\cdot) \) Dirac delta function
\( \pi(\cdot|\cdot) \) importance function
\( \mathbb{1}(\cdot) \) indicator function
\( |\cdot| \) absolute value or volume
\( \lfloor \cdot \rfloor \) integer part
\( \in \) belongs to
\( \subset \) subset of
Symbols

d  dimension

d'  dimension before transformation

n_i  number of copies from particle x_i

n_t  process noise vector at time t

u_k  k-th ordered random number

w_k  k-th random number

w_i  normalised importance weight of particle x_i

x_i  i-th particle

x^*_j  j-th particle after resampling

x_t  state vector at time t

x_{0:t}  history of states up to time t

y_t  observation vector at time t

y_{0:t}  history of observations up to time t

D  integration domain

I_k  result of k-th selection

N  number of particles

N(\mu, \sigma)  normal distribution with mean \mu and standard deviation \sigma

Q_i  sum of normalised importance weights

\mathbb{R}  real values

\mathbb{R}^+  positive real values

U[0,1)  standard uniform distribution

\pi  random permutation vector

\nu_t  observation noise vector at time t

Abbreviations and acronyms

i.i.d.  independent and identical distributed

w.r.t.  with respect to

CDF  Cumulative Density Function

HMM  Hidden Markov Model

MC  Monte Carlo

PDF  Probability Density Function

RMSE  Root Mean Square Error

SIS  Sequential Importance Sampling

SISR  Sequential Importance Sampling with Resampling
Appendix A

Process noise generation

In this appendix it is investigated whether the sampling techniques from Section 3.2.1 can be used to generate samples from the process noise distribution (2.1). If these variance reducing methods can be applied, the quality of estimates from the particle filter might be improved.

The particle filter recursively updates the approximate filter density \( \hat{p}_N(x_t|y_{0:t}) \) according to Algorithm 2.2. Estimates are calculated using the density before resampling by

\[
\hat{I}(g) = \sum_{i=1}^{N} w_{t,i} g(x_{t,i}) = \frac{\sum_{i=1}^{N} \tilde{w}(x_{t,i}) g(x_{t,i})}{\sum_{i=1}^{N} \tilde{w}(x_{t,i})}
\]

(A.1)

with

\[
x_{t,i} \sim p(x_t|x_{t-1,i}) = p_t(x)
\]

(A.2)

Both the numerator and denominator of (A.1) can be identified as MC approximations of the form of (3.5). However, the samples \( x_i \) are distributed according to \( p_t(x) \) instead of \( p(x) \). Thus the \( N \) samples are not drawn from one PDF but from \( N \) PDFs. Therefore, the variance reducing methods from Section 3.2.1 cannot be applied for process noise generation.
Appendix B

Implementations

In this appendix the MATLAB implementations of the resampling algorithms discussed in Chapter 3 are listed. These codes have been used for the simulations in Chapter 4.
B.1 Simple random resampling (Alg. 3.3)

```matlab
function [x, w] = rs_simple(x, w, N)
% RS_SIMPLE simple resampling
%
% [X,W] = RS_SIMPLE(X,W) resamples the states X
% and their associated weights W using simple
% random sampling.
%
% [X,W] = RS_SIMPLE(X,W,N) returns a resampled
% distribution of size N.
%
% check inputs
if nargin<3
    N = length(w);
end
%
% generate ordered uniform random numbers
u = cumprod(rand(1,N).^(1./[N:-1:1]));
u = u(N:-1:1);
%
% generate cumulative weights (ordered)
wc = cumsum(w);
%
% indices of the selected states
ind = zeros(1,N);
k = 1;
for i = 1:N
    while(wc(k)<u(i))
        k = k + 1;
    end
    ind(i) = k;
end;
%
% create new states and corresponding weights
x = x(:,ind);
w = ones(1,N)./N;
```
function \( [x, w] = \text{rs\_stratified}(x, w, N) \)

% RS\_STRATIFIED stratified resampling
% 
% \([X, W] = \text{RS\_STRATIFIED}(X, W)\) resamples the states \(X\) and their associated weights \(W\) using stratified sampling.
% 
% \([X, W] = \text{RS\_STRATIFIED}(X, W, N)\) returns a resampled distribution of size \(N\).
% 
% check inputs
if nargin<3
    N = length(w);
end
% generate ordered uniform numbers
\(u = ([0:N–1]+(\text{rand}(1,N)))/N;\)
% generate cumulative weights (ordered)
\(wc = \text{cumsum}(w);\)
% indices of the selected states
\(\text{ind} = \text{zeros}(1,N);\)
\(k = 1;\)
for \(i = 1:N\)
    while (wc(\(k\))<u(\(i\)))
        \(k = k + 1;\)
    end
    \(\text{ind}(i) = k;\)
end
% create new states and corresponding weights
\(x = x(:,\text{ind});\)
\(w = \text{ones}(1,N)./N;\)
B.3 Systematic resampling (Alg. 3.5)

```matlab
function [x,w] = rs_systematic(x,w,N)
  %RS_SYSTEMATIC systematic resampling
  % [X,W] = RS_SYSTEMATIC(X,W) resamples the states X and their associated weights W using systematic sampling.
  % [X,W] = RS_SYSTEMATIC(X,W,N) returns a resampled distribution of size N.
  % check inputs
  if nargin<3
    N = length(w);
  end
  % generate ordered uniform numbers
  u = ([0:N-1]+rand(1))/N;
  % generate cumulative weights (ordered)
  wc = cumsum(w);
  % indices of the selected states
  ind = zeros(1,N);
  k = 1;
  for i = 1:N
    while wc(k)<u(i)
      k = k + 1;
    end
    ind(i) = k;
  end;
  % create new states and corresponding weights
  x = x(:,ind);
  w = ones(1,N)./N;
```
B.4 Residual resampling (Alg. 3.6)

```matlab
function [x,w] = rs_regularized(x,w,N,rs)
%RS_REGULARIZED residual resampling
% [X,W] = RS_REGULARIZED(X,W) resamples the states X and their 
% associated weights W using residual resampling.
% [X,W] = RS_REGULARIZED(X,W,N) returns a resampled distribution of size N.
% [X,W] = RS_REGULARIZED(X,W,N,RS) uses RS as resample function to sample the remaining points from 
% (defaults to RS_STRATIFIED).
% check inputs
if nargin<3
    rs = @rs_stratified;
    N = length(w);
elseif nargin<4
    rs = @rs_stratified;
end
% calculate the number of copies
wa = N*w;
 nk = floor(wa);
Nk = cumsum(nk);
K = Nk(end);
% indices of the particles to copy
ind = zeros(1,K);
k=1;
for i = 1:K
    while(Nk(k)<i)
        k = k + 1;
    end
    ind(i) = k;
end
% calculate the number of additional copies and the weights for random resampling
wb = wa-nk;
w = wb / sum(wb);
 if n>0
     x = [x(:,ind), feval(rs,x,wb,n)];
 else
     x = x(:,ind);
 end
 w = ones(1,N)./N;
```
Copyright

På svenska

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådan sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet — or its possible replacement — for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Jeroen D. Hol