uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Frédéric Mustière

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Particle Filtering Methods for the Enhancement of Speech Corrupted by Additive Noise

TITRE DE LA THÈSE / TITLE OF THESIS

M. Bouchard

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

M. Bolic

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

R. Goubran

T. Aboulnasr

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Particle filtering methods for the enhancement of speech corrupted by additive noise

by

Frédéric Mustière

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

Ottawa-Carleton Institute for
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25811-8
Our file Notre référence
ISBN: 978-0-494-25811-8

Abstract

In this work, we study the application of particle filtering (PF) algorithms to the problem of speech enhancement. The goal of the thesis is to devise PF algorithms that will enhance speech signals corrupted by additive noise, and to evaluate their performance via comparisons with other existing algorithms based on several quality measures. Speech enhancement, or noise reduction, is an important problem in many applications, such as telephony and telecommunications in general, sound recording, human-maching interface (where speech recognition is important), etc. Even though many algorithms already exist for speech enhancement, there is still very much work to do, especially in terms of intelligibility. In many cases, it may be easier to understand the original, noisy speech rather than the processed, "cleaned-out" one. In other cases, the residual noise may be too annoying to carry out a comfortable conversation. In this context, new approaches for the denoising of speech are welcome.

As a first contribution, a practical approach to deriving simple Rao-Blackwellised Particle Filters (RBPFs), which was developed in parallel with a theoretic review of PFs, is presented. In addition, a novel algorithm, called the modified Rao-Blackwellised Particle Filter (RBPF), is proposed to reduce the computational load of regular RBPFs. Several new speech enhancement methods using particle filters are also derived, and shown to outperform some other existing PF-based algorithms. Accessorily, a novel strategy to extend their range of application to colored noise is explained and applied. Comparatively to the other types of enhancement algorithms tested (including spectral subtraction, signal subspace, dual extended Kalman filter, perceptually constrained Kalman filter, dual perceptually constrained unscented Kalman filter) we find that the particle-filter based algorithms presented have the advantage of not introducing any musical noise. Furthermore, in the conditions of our experiments, using several objective measures we find that they are able to compete with and outperform most of the other algorithms tested. Using these measures and based on informal listening, we highlight their advantages – naturalness of the enhanced speech, low intrusiveness of the non-musical residual noise, very good performance at high SNR, flexibility – and their main limitations – intraspeech residual noise "modulated" by the speech, computational burden. Considering how flexible and parametrizable PFs are, there is a strong potential for further improvement.

Acknowledgements

I am sincerely thankful to my supervisors, Dr. Martin Bouchard and Dr. Miodrag Bolić, for all of their support, their feedback, their guidance and comments during my Master's studies. They helped me stay on track whenever I began researching in fruitless directions (which happened often!). They also encouraged me to go and present parts of my work in two conferences.

I would also like to thank my wife, Lisa, for giving me unlimited support, even when I was too captivated in some paper or textbook to decently show my appreciation.

Finally, I would like to thank my dog Clovis for helping me waking up in the morning everyday by jumping on the bed, and by keeping me (relatively) fit by demanding regular walks during the day, encouraging me to get off the eternal computer chair.

Original/Novel ideas contained in the thesis

In the table of contents and in section titles, the novelties (published or not) are marked by a bracketed star ($\star$).

Two publications have emerged from the following work. The first one, presented in section 3.3.1, appeared in the Canadian Conference on Electrical and Computer Engineering, taking place in Ottawa in May 2006, see [42]. The second one, in section 3.4, was presented during the ICASSP conference in Toulouse, in May 2006 as well, see [41]. The thesis also contains unpublished contributions.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and goal of the thesis

For their numerous applications, *Speech enhancement* (or *denoising*) techniques are very important in the field of signal processing. They are employed in many contexts such as hands-free telephony, hearing aids systems, remastering of audio recordings, pre-processing for speech recognition devices in human-machine interfaces, etc. In a speech enhancement problem, we are interested in somehow "removing" the noise from a speech signal, in an attempt to recover the original one.

In the past few decades, this topic has claimed the attention of numerous researchers, and a plethora of methods and algorithms has emerged. These algorithms all have their advantages and drawbacks, although it is difficult to rate them objectively. Some aspects can be unequivocally compared, such as execution time, or the amount of information required a priori for the proper operation of the algorithm. But some other aspects, mainly the quality of the output/enhanced speech, are much more difficult to judge. A given listener may feel that the most important characteristic of the output speech is the lack of annoying, residual noise in between utterances. Another listener may be more sensitive to the distortion applied to certain syllabes. In addition, comparisons are complicated by the fact that a given algorithm may perform very well for a given speaker and noise mixture, but may perform poorly for another set of conditions. In general, the following rule of thumb applies: the more an algorithm removes noise, the more distortion the resulting speech will endure. In this context, the **intelligibility** of the enhanced speech is of foremost importance. In some cases, it may be easier to understand the unprocessed, noisy speech than the output of the speech enhancement algorithm! This simply demonstrates that there is still work to be done in the field. Unfortunately, as natural of a concept as it may be, intelligibility is not only subjective, but it is also very difficult to evaluate

quantitatively. To test and compare new methods with limited time and cost, researchers in speech processing are nevertheless interested in devising mathematical tools intended to *predict* the average opinion of an arbitrary population of listeners on the intelligibility and the quality of enhanced speech signals. However, such predictors are obviously not flawless, and they cannot pretend to replace the results of any large-scale subjective testing. But when aware of their limitations, one can still extract from them some valuable information – an effort is made to do so in this thesis.

Particle Filters (PFs) are a class of algorithms developed over the past decade, which implement a type of Bayesian state estimation, based on a set of available information, which can be the measured data and our a priori knowledge of the state. The so-called Bayesian approach (see 1.3 below) is a probabilistic method, which attempts to evaluate the probability of the state (and maybe some parameters) given the set of observations, using the knowledge of the probability of the observations given the state. The states can then be treated as random processes. Among the available Bayesian filters, PFs are the most demanding in terms of computational load, however they are also the most flexible and general. With the exponential increase in computers' power, particle filters are becoming more and more appealing, and their field of possible applications is under constant investigation. The problem of speech enhancement naturally falls in the range of applications of Bayesian state estimation, since a conceptual solution to this problem is probabilistic: we are interested in being able to infer the original source signal from a set of (corrupted) observations.

In this work, we are interested in exploring the possible particle filtering solutions to speech enhancement problems. We will first present and derive, in chapter 2, a particle filter algorithm. Then, due to its importance and the use that we make of it subsequently, in chapter 3 we present a class of particle filters called Rao-Blackwellised particle filters, along with two examples. In chapter 4, we review some important concepts and methods traditionally used for speech enhancement. In chapters 5 and 6, we then apply the algorithms presented in the two first chapters to the speech denoising problem presented in chapter 4. Chapter 5 is dedicated to existing particle filtering algorithms, nonetheless it does contains original simulation results, comments, and analyses. In contrast, the purpose of chapter 6 is to introduce novel approaches and algorithms, and it also contains comparative simulation results with the existing denoising algorithms presented in chapters 4 and 5. Finally, in the last chapter we conclude, in the light of the results and ideas presented, on the potential of particle filtering methods in signal processing, and we discuss and present ideas to pursue in future work.

1.2 A word on the notation used and on Bayesian inference

1.2.1 Probability density and mass functions

In the Bayesian framework, a certain notation in papers is commonly employed. This notation is more often than not considered to be known by the reader. The fundamental tools that one must deal with in our context are random variables. Traditionally, to denote a random variable, we use the capital letter X, and its realization x. In the case of continuous random variables, a practical characterization of X is its probability density function, denoted by $f_X(x)$. In the case of discrete random variables, we find it more convenient to use the probability mass function, which may be written as $\Pr[X = x]$, or $P(X = x)$. To simplify notations, one will almost surely encounter in today's literature the notation $p(.)$ to denote both the probability density function and the probability mass function, depending on the context. Instead of writing $f_X(x)$ or $P(X = x)$, we simply write $p(x)$. In fact, X could also be a random vector, $X = \{X_1, X_2, ..., X_n\}$, in which case we would have $x = (x_1, x_2, ..., x_n)$. Then, $p(x)$ would denote the joint probability density function of all individual random variables in the random vector, or the joint probability mass function. To simplify further, abusing slightly the language, many authors simply refer to $p(x)$ as the distribution of the random variable X. Additionally, in the case of continuous random variables, it is common to read $p(dx)$ to mean $\Pr[X \in dx]$, i.e., $p(dx) = \int p(x)dx$, in which case $p(dx)$ in fact represents, in rigorous terms, a probability distribution function.

One may wonder how blurring concepts in this way could simplify our task at all if we are to be rigorous in any way. But in fact, using the notation $p(.)$ turns out to be very handy, since the rules that apply to continuous or discrete random variables are mostly identical. For continuous random variables, any summation used for computing expectations or for marginalization can be applied under the form of an integral on the probability density function, and for discrete random variables, a discrete sum on the probability mass function may be employed for the very same purpose. For example, we have:

For continuous random variables X_1,X_2 and X_3 , we could write:

$$f_{X_1}(x_1) = \int f_{X_1X_2}(x_1, x_2)dx_2 \tag{1.1}$$

or:

$$f_{X_1|X_3}(x_1|X_3 = x_3) = \int f_{X_1X_2|X_3}(x_1, x_2|X_3 = x_3)dx_2 \tag{1.2}$$

Or, using the notation $p(.)$, we could also now write, equivalently:

$$p(x_1) = \int p(x_1, x_2)dx_2 \tag{1.3}$$

$$\text{or } p(x_1|x_3) = \int p(x_1, x_2|x_3)dx_2 \tag{1.4}$$

Similarly, we have:

$$p(x_1|x_2) = f_{X_1}(x_1|X_2 = x_2) = \frac{f_{X_1X_2}(x_1, x_2)}{f_{X_2}(x_2)} = \frac{p(x_1, x_2)}{p(x_2)} \tag{1.5}$$

and we can also write:

$$\int p(x_1|x_2)dx_1 = 1 \tag{1.6}$$

Now, the same relations could be written if X_1, X_2 and X_3 were discrete, using a discrete sum symbol:

$$P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2) \tag{1.7}$$

or:

$$P(X_1 = x_1|X_3 = x_3) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2|X_3 = x_3) \tag{1.8}$$

Or, using the notation $p(.)$, we could also now write, equivalently:

$$p(x_1) = \sum_{x_2} p(x_1, x_2) \tag{1.9}$$

$$\text{or } p(x_1|x_3) = \sum_{x_2} p(x_1, x_2|x_3) \tag{1.10}$$

Similarly, we have:

$$p(x_1|x_2) = P(X_1 = x_1|X_2 = x_2) = \frac{P(X_1 = x_1, X_2 = x_2)}{P(X_2 = x_2)} = \frac{p(x_1, x_2)}{p(x_2)} \tag{1.11}$$

and we can also write:

$$\sum_{x_1} p(x_1|x_2) = 1 \tag{1.12}$$

To simplify even further, some sources only use one of the two symbols $\sum$ or $\int$ for both continuous and discrete random variables, explaining that the symbol should be interpreted as an integral or as a discrete sum depending on the context, keeping the notation as simple as possible.

1.2.2 Common symbols and notation used in the thesis

- **Gaussian distributions**

  For a Gaussian distribution with variable $\mathbf{x}$, mean $\mathbf{m}$ and covariance matrix $\mathbf{Q}$, we will write $\mathcal{N}(\mathbf{x}|\mathbf{m};\mathbf{Q}) \triangleq |2\pi\mathbf{Q}|^{-1/2}\exp(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{m}))$. When there is no possible confusion on which variable is being considered, then we may write simply $\mathcal{N}(\mathbf{m},\mathbf{Q})$.

- **Characters convention**

  By convention, we will use bold small case letters to denote vectors. Bold upper case letters will be used to denote matrices. By contrast, regular small case letters will represent scalars, and regular upper case letters will be used for constants.

  The only exceptions to this rule are $\mathbf{w}_k$, the process noise at time k, which can denote both a vector and a scalar, and $\mathbf{X}_k$ and $\mathbf{Z}_k$, which are not matrices but more exactly sequences of vectors (see Table 1.1).

  Additionally, in large tables where many results are reported, to help visualize more rapidly the important points, we will use a simple color coding. Basically, when a result is posted and compared to another result, then the blue-colored number will denote the best of the two, and a red-colored number will indicate that the result is the worse. In addition, if there are more than two algorithms being compared, then more intermediate colors will be used, ranging gradually from blue to red as in the last six words of this sentence.

- **Table of regularly used quantities and symbols**

$\mathbf{x}_k$ (or x_k)	state vector (or scalar) of a system at time k – the quantity to estimate
$\mathbf{X}_k = \{\mathbf{x}_0, \mathbf{x}_1 \dots \mathbf{x}_k\}$	history of states up to time k
$\mathbf{z}_k$ (or z_k)	measurement vector (or scalar) at time k
$\mathbf{Z}_k = \{\mathbf{z}_0, \mathbf{z}_1 \dots \mathbf{z}_k\}$	history of measurements up to time k
$\hat{\mathbf{x}}_k$ (or $\hat{x}_k$)	estimate of the state at time k
$\mathbf{w}_k$	process noise vector or scalar in a state-space model at time k
$\mathbf{v}_k$ (or v_k)	observation noise vector (or scalar) in a state-space model at time k
N	number of particles in a particle filter
$q(.)$	importance function in a particle filter
$w_{k,i}$	the weight associated to the i^{th} particle at time k
$\mathbf{I}$	identity matrix
$\mathbf{K}_{k\|k-1}$	at time k, *a priori* estimate error covariance matrix in a Kalman filter
$\mathbf{K}_k$	at time k, *a posteriori* estimate error covariance matrix in a Kalman filter
$\mathbf{J}_k$	at time k, the Kalman gain in a Kalman filter
$\mathbf{y}_k$ and $\mathbf{T}_k$	at time k, a vector and a matrix defined and used within a Kalman filter
$\mathcal{E}(.)$	expectation operator
$\triangleq$	means "*is defined to be equal to*"
$\equiv$	means "*is identical to*"
M	order of autoregression, or of a Wiener filter

Table 1.1: Some common nomenclature

- **Linear state-space models and the Kalman filter**

  On several occasions, we will be dealing with linear systems. Whenever it is the case, we will be using the following notation:

$$\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{G}_k\mathbf{w}_k \tag{1.13}$$

$$\mathbf{z}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{D}_k\mathbf{u}_k + \mathbf{H}_k\mathbf{v}_k \tag{1.14}$$

  We will consider in such cases that $\mathbf{w}$ and $\mathbf{v}$ are zero mean unit covariance Gaussian random vectors. If a Kalman filter is to be applied to estimate the values of $\mathbf{x}_k$ based on $\mathbf{Z}_k$, then we will use here the notations:

$$\begin{aligned} p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) &\triangleq \mathcal{N}(\mathbf{x}_{k-1}|\hat{\mathbf{x}}_{k-1}, \mathbf{K}_{k-1}) \\ p(\mathbf{x}_k|\mathbf{Z}_{k-1}) &\triangleq \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_{k|k-1}, \mathbf{K}_{k|k-1}) \\ p(\mathbf{x}_k|\mathbf{Z}_k) &\triangleq \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_k, \mathbf{K}_k) \end{aligned}$$

and the Kalman filter equations are the following [28]:

$$\begin{aligned} \mathbf{K}_{k|k-1} &= \mathbf{G}_k\mathbf{G}_k{}^T + \mathbf{A}_k\mathbf{K}_{k-1}\mathbf{A}_k{}^T \\ \mathbf{T}_k &\triangleq \mathbf{H}_k\mathbf{H}_k{}^T + \mathbf{C}_k\mathbf{K}_{k|k-1}\mathbf{C}_k{}^T \\ \hat{\mathbf{x}}_{k|k-1} &= \mathbf{A}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\mathbf{u}_k \\ \mathbf{y}_k &\triangleq \mathbf{C}_k\hat{\mathbf{x}}_{k|k-1} + \mathbf{D}_k\mathbf{u}_k \\ \mathbf{J}_k &\triangleq \mathbf{K}_{k|k-1}\mathbf{C}_k{}^T\mathbf{T}_k^{-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{J}_k(\mathbf{z}_k - \mathbf{y}_k) \\ \mathbf{K}_k &= (\mathbf{I} - \mathbf{J}_k\mathbf{C}_k)\mathbf{K}_{k|k-1} \end{aligned}$$

On several occasions in the thesis, the Kalman filter will be used as a "subalgorithm" of a larger particle filter. The same notation will be kept, except for $\mathbf{x}_k$ and $\hat{\mathbf{x}}_k$ which will be defined and replaced according to the context (these larger particle filters, introduced in chapter 3, use Kalman filters operating on "substates").

1.3 On Bayesian inference

Let us now clarify the very basics of the Bayesian approach. If we want to estimate a certain quantity x given an observation z, our initial knowledge of the variable x (i.e., our knowledge without any observation) takes the form of a distribution $p(x)$. This distribution is called a *prior distribution*, or simply a prior. In addition, from our knowledge of the system observed, we should be able to define the so-called *likelihood* distribution, which is defined as the distribution of the observation z conditioned upon x, denoted $p(z|x)$. The key idea is then the following: with the use of Bayes' rule, the prior and the likelihood, then we can determine the *posterior density*, or the posterior, defined as the probability of x conditioned upon z, $p(x|z)$, as follows:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} = \frac{p(z|x)p(x)}{\int p(z|x')p(x')dx'} \tag{1.15}$$

The determination of the posterior is the goal of Bayesian methods, and we will use a similar argument in the introduction of particle filters in the next section. From the posterior, we can deduce an estimate for x, for example $\mathcal{E}\{x|z\}$.

In the context of sequential problems, this may become confusing if care is not taken to always make sure that no ambiguities are possible. More precisely, suppose that we are now

trying to estimate in time the state of a system, which at discrete time k is denoted x_k, using a sequence of measurements/observations, the observation done at time k being denoted z_k. As we will see in the next section, in this case we can still use Bayes' rule sequentially to determine, for each k, the posterior of the state at time k. However, we do not only require the prior and the likelihood, but also two other important distributions, both of which are usually given a specific name. Below, we list the distributions that intervene and their usual name:

- The prior on x_0, $p(x_0)$
- The likelihood $p(z_k|x_k)$
- The *transitional probability* for x, $p(x_k|x_{k-1})$, sometimes called the *transitional prior*
- The *a priori* state density $p(x_k|z_0, z_1, \ldots, z_{k-1})$ (not to be mistaken with the *prior*).
- The posterior distribution $p(x_k|z_0, z_1, \ldots, z_k)$, sometimes also called the "a posteriori" density.

In this work, we will always try to make sure that there is no doubt about what distributions the words "prior" and "posterior" are pointing to.

Chapter 2

Introduction to Particle Filters

2.1 Particle filters: the problem setting

2.1.1 The generic sequential state estimation problem and its Bayesian solution

The setting of the problem of sequential state estimation given noisy measurements can be summarized in state-space form as the following coupled equations [8, 47]:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \tag{2.1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \tag{2.2}$$

where $\mathbf{x}_k$ represents the state vector at instant k, $\mathbf{z}_k$ represents the measurement (sometimes referred to as the observable), $\mathbf{w}_k$ is the process noise and $\mathbf{v}_k$ is the measurement noise. Functions f and h are functions that are assumed to be known, and may depend on time.

Equation (2.1) is sometimes called the state transition equation, and equation (2.2) the observation equation. In this problem, we would like to be able to estimate the state based on the set of all available measurements up to time k, which we denote $\mathbf{Z}_k = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_k\}$. In the case where functions f and h are linear, $\mathbf{w}$ and $\mathbf{v}$ are independent and Gaussian, then the recursive solution to these equations is optimally represented by the Kalman Filter [28], as a closed form recursion. When f and h are nonlinear, $\mathbf{w}$ and $\mathbf{v}$ still being independent and Gaussian, several approximations exist, such as the Extended Kalman Filter (EKF) [6, 23], which is based on first order Taylor approximation of f and h. However, this linearization may not be accurate enough, in which case the EKF can diverge [24, 47]. Another, more accurate type of approximation is the Unscented Kalman Filter (UKF), which uses at each time step the so-called Unscented Transformation (UT) to propagate a Gaussian approximation for the

posterior density $p(\mathbf{x}_k|\mathbf{Z}_k)$ through the nonlinear function f, which is then updated through h. The obtained mean and covariance matrix are shown to be much more accurately captured, and the UKF generally performs better than the EKF. The main limitation of both the EKF and the UKF is the fact that they approximate all the key distributions as Gaussians. This approximation may not be valid for complex, real-world approximations with multi-modal, non-Gaussian and multidimensional posterior distributions.

The Particle Filters (PF) introduce an approximate sequential solution to equations (2.1) and (2.2) for very weak assumptions: f and h may be non-linear, $\mathbf{w}$ and $\mathbf{v}$ may be non-Gaussian, at the cost of a more computationally expensive implementation [47]. With PFs, an approximation of the entire posterior density $p(\mathbf{x}_k|\mathbf{Z}_k)$ is propagated and updated in time.

The state space equations (2.1) and (2.2) can be seen in a different way, more informative for our approach but equivalent. The state space equations describe the evolution of random processes in time, and in fact, the state $\mathbf{x}$, as described by (2.1), is a Markov process of order one. We are interested in describing the state transition and the observation process using the distributions of the quantities that they describe. To be more specific, we can make use of equation (2.1) to determine the transitional density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and from equation (2.2), we obtain the likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$. The goal of the filtering problem is to determine, at time k, the posterior distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$. Using simple rules from traditional probability theory, we can obtain a two-step recursion on $p(\mathbf{x}_k|\mathbf{Z}_k)$.

Suppose that $p(\mathbf{x}_k|\mathbf{Z}_{k-1})$ is available to us, and that we obtain a new measurement $\mathbf{z}_k$. We have:

$$p(\mathbf{x}_k|\mathbf{Z}_k) = p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) = \frac{p(\mathbf{x}_k, \mathbf{z}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1})p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \tag{2.3}$$

However, from equation (2.2), the measurement $\mathbf{z}_k$ is independent of $\mathbf{Z}_{k-1}$ conditioned upon $\mathbf{x}_k$. Thus, $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$, and $p(\mathbf{z}_k|\mathbf{x}_k)$ is defined by equation (2.2). The only term that is not directly available is $p(\mathbf{z}_k|\mathbf{Z}_{k-1})$. But it is only a marginalized version of the numerator over $\mathbf{x}_k$. Thus, we obtain:

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{\int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_{k-1})d\mathbf{x}_k} \tag{2.4}$$

Equation (2.4) is called the *update* equation – it is a form of Bayes' rule. Once (2.4) is computed, we are ready to use equation (2.1) to predict $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$. But we have:

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int p(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{Z}_{k-1})d\mathbf{x}_{k-1} = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})d\mathbf{x}_{k-1} \tag{2.5}$$

Here, from equation (2.1), we see that $p(\mathbf{x}_k, |\mathbf{x}_{k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k, |\mathbf{x}_{k-1})$. Therefore we get:

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})d\mathbf{x}_{k-1} \tag{2.6}$$

Equation (2.6) is called the *prediction* equation. Together, equations (2.4) and (2.6) form a recursive solution, sometimes called *Bayesian*, to the problem defined by equations (2.1) and (2.2). Unfortunately, the integrals involved are intractable in most practical cases [47]. Particle filtering is a way to approximate the solution. The main idea behind Particle filtering is Monte-Carlo numerical integration techniques. In the next section, we first briefly present the main ideas, and then we see how they can be used to approximate (2.4) and (2.6).

A probabilistic graphical representation for the state-space model

Even though the situation is relatively simple, it is an opportunity to introduce the conventions used here for graphical models. We will in fact more specifically be using Bayesian Networks (a family of graphical models). Figure 2.1 below is a direct graphical representation of the context given by equations (2.1) and (2.2).

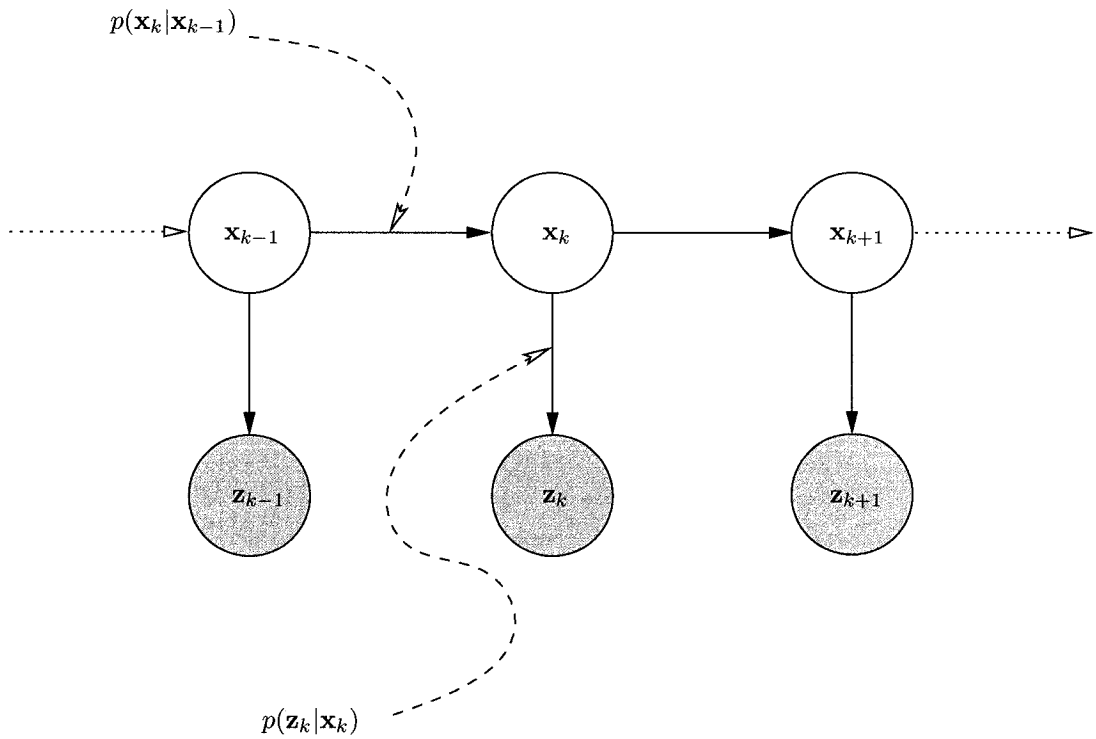


Figure 2.1: Graphical model for the generic state estimation problem

Bayesian Networks (BNs) are directed acyclic graphs, in which each node represents a random variable [27]. The shaded nodes denote observed variables. The arrows define a certain

factorization of the joint probability distribution for all the nodes in the graph: for a given graph, the joint distribution factors as the product of the distribution of each node given its parents. With this interpretation, the joint distribution is conditioned upon the shaded nodes in the graph.

The graph in figure 2.1 is in fact a Dynamic Bayesian Network (DBN), as there is a certain time index k which keeps unfolding the graph as time goes by [8,27]. BNs are a very practical tool for deducing dependencies and independencies: for a set of shaded nodes, certain of these properties arise from the structure of the graph. The Bayes Ball algorithm is a possible tool to infer these dependencies and independencies [48]. In this algorithm, based on the directions of arrows at each node, a "Ball" may or may not be able to reach a certain set of nodes from another set of nodes. If the Ball is blocked, then the random variables in the two set of nodes are independent conditioned upon the shaded nodes. If the Ball can pass, then the random variables are not independent. We do not summarize here the set of rules that define the Bayes Ball algorithm, but we give the following example: if any node $\mathbf{x}_k$ on the top row of the graph in figure 2.1 is shaded, ie, if the random variable it represents is conditioned upon, the "ball" will not be able to pass through the chain from one side to the other: the "future" is then independent on the "past", given that specific node. Of course, these independencies can be deduced analytically as well with consistency, however the graphical approach is sometimes more immediate.

2.1.2 Monte Carlo Integration

Consider the following integral:

$$I = \int g(\mathbf{x})d\mathbf{x} \tag{2.7}$$

We would like to be able to evaluate this integral, where $\mathbf{x}$ is a vector, $g(.)$ is a multivariable function, which in many cases makes it untractable. To be able to approximate I, we may intepret it as an expectation, $\mathbf{x}$ representing a random vector realization of a random variable $\boldsymbol{X}$ [47]. If we write $g(\mathbf{x}) = f(\mathbf{x})h(\mathbf{x})$, where $h(\mathbf{x})$ is interpreted as the probability density function of the random variable $\boldsymbol{X}$ (in the case of continuous random variables), then I could be simply seen as:

$$I = \mathcal{E}\{f(\boldsymbol{X})\} \tag{2.8}$$

Then, from statistics theory, we know that if we are able to obtain N independent observations of $\boldsymbol{X}$, then an unbiased estimate for I is [8]:

$$\hat{I} = \frac{\sum_{i=1}^{N} f(\mathbf{x}_i)}{N} \tag{2.9}$$

where $\mathbf{x}_i$ is the i^{th} observation of $\boldsymbol{X}$.

This estimate is called the *sample mean* estimate. It converges to I almost surely as N tends to infinity [8, 14]. To obtain it, one must be able to draw i.i.d. samples from the random variable $\boldsymbol{X}$, which we said was distributed according to $h(\mathbf{x})$. The problem is that we may not always be able to do so. But suppose that we are able to draw i.i.d. samples according to another distribution $q(\mathbf{x})$. With the condition that $q(\mathbf{x})$ and $h(\mathbf{x})$ have the same support, we can deduce the following: if we write $g(\mathbf{x}) = f(\mathbf{x}).\frac{h(\mathbf{x})}{q(\mathbf{x})}.q(\mathbf{x})$, and if we let $\frac{h(\mathbf{x})}{q(\mathbf{x})} = \widetilde{w}(\mathbf{x})$, which we call the (unnormalized) weights, then $g(\mathbf{x}) = [f(\mathbf{x}).\widetilde{w}(\mathbf{x})].q(\mathbf{x})$, and we can use again the sample mean estimate, but this time drawing samples from the distribution $q(\mathbf{x})$, to get:

$$\hat{I} = \frac{\sum_{i=1}^{N} f(\mathbf{x}_i)\widetilde{w}(\mathbf{x}_i)}{N} \tag{2.10}$$

The distribution $q(\mathbf{x})$ is called the *importance distribution*. We insist here, to avoid confusion later on, that $q(\mathbf{x})$ is *not* necessarily, a priori, related to the distribution of the random variable $\boldsymbol{X}$. It is simply another function of the variable $\mathbf{x}$, which has the properties of a distribution function. By drawing i.i.d. samples from $q(\mathbf{x})$ to evaluate I, we are performing an *importance sampling*. In the context of particle filtering, as we will see in the next section, computations are simplified in such a way that we only know the distribution $h(\mathbf{x})$ up to a normalizing constant. In this case, the estimate of I can still be directly evaluated, provided one performs a normalization of the weights [8, 47]:

$$w(\mathbf{x}_i) = \frac{\widetilde{w}(\mathbf{x}_i)}{\sum_{k=1}^{N} \widetilde{w}(\mathbf{x}_k)} \tag{2.11}$$

Then, the estimate becomes:

$$\hat{I} = \sum_{i=1}^{N} f(\mathbf{x}_i)w(\mathbf{x}_i) \tag{2.12}$$

2.2 Applying Monte Carlo integration to Bayesian filtering

In section 2.1.1, we had pointed out that the computation of $p(\mathbf{x}_k|\mathbf{Z}_k)$ as formulated by equation (2.4) is too complex of a task. But in the light of section 2.1.2, to evaluate $p(\mathbf{x}_k|\mathbf{Z}_k)$ at a certain $\mathbf{x}_k$, we can use Monte Carlo integration techniques. We will here give an intuitive view of the solution. First, denoting $\mathbf{X}_k = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\}$ the set of all (true) states up to time k, the distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$ can be deduced from $p(\mathbf{X}_k|\mathbf{Z}_k)$ by marginalization. (In fact, knowing

$p(\mathbf{X}_k|\mathbf{Z}_k)$ provides us with much more information than just the marginal). We know, from the properties of the (multidimensional) Dirac delta function (and in the case of a continuous random variable - for the discrete version, replace by a discrete sum and by the discrete Delta function):

$$p(\mathbf{X}_k|\mathbf{Z}_k) = \int p(\mathbf{X}|\mathbf{Z}_k)\delta(\mathbf{X}_k - \mathbf{X})d\mathbf{X} \tag{2.13}$$

Compare now with the results from the previous section. Consider that, at time k, $\mathbf{X}_k$ is given. Denoting $f(\mathbf{X}) = \delta(\mathbf{X}_k - \mathbf{X})$ (where $\mathbf{X}$ has here the same dimension as $\mathbf{X}_k$), and choosing an importance density of the form $q(\mathbf{X}_k|\mathbf{Z}_k)$ (using the same support as $p(\mathbf{X}_k|\mathbf{Z}_k)$), we see that if we draw N i.i.d. samples $\mathbf{X}_{k,i} \sim q(\mathbf{X}_k|\mathbf{Z}_k)$, where $i \in \{1, \ldots, N\}$, then we should be able to write, from (2.12):

$$\hat{p}(\mathbf{X}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} f(\mathbf{X}_{k,i})w(\mathbf{X}_{k,i}) \tag{2.14}$$

where $w(\mathbf{X}_{k,i}) \propto \dfrac{p(\mathbf{X}_{k,i}|\mathbf{Z}_k)}{q(\mathbf{X}_{k,i}|\mathbf{Z}_k)}$, which we will now denote simply by $w_{k,i}$.

Observing the previous equation, we see that what we really obtained by performing importance sampling is a weighted, discrete approximation of $p(\mathbf{X}_k|\mathbf{Z}_k)$. This approximation, at time k, is therefore completely determined by the set of samples, or support points $\{\mathbf{X}_{k,i}\}_{i=1}^{N}$, drawn from the importance density, and the set of weights $\{w_{k,i}\}_{i=1..N}$. For a given i, the pair $\{\mathbf{X}_{k,i}, w_{k,i}\}$ is called a *particle*, (or a *particle trajectory* to emphasize the fact that we are considering the entire history of $\mathbf{x}_{k,i}$).

2.2.1 Towards a sequential Monte Carlo filtering

Since the distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$ for all k is what we are looking for as a solution to the generic state estimation problem as defined by (2.1) and (2.2), we now see that we are able to represent the desired solution by a set of particles. But we are not done: suppose that we have, at time k, a set of particles approximating the joint posterior distribution $p(\mathbf{X}_k|\mathbf{Z}_k)$, and a new measurement becomes available. To really make this approach convenient, we would like to find a way to update the weights and support points by incorporating the information carried by the new measurement, so as to obtain a recursive way of solving the problem, in the same way that we go from the LS to the RLS algorithm, for example. Also, it would be even better if we did not have to keep track (in memory) of all the past information.

Let us derive this recursion. We suppose that we are given an approximation of $p(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})$, as represented by $w_{k-1,i} \propto \frac{p(\mathbf{X}_{k-1,i}|\mathbf{Z}_{k-1})}{q(\mathbf{X}_{k-1,i}|\mathbf{Z}_{k-1})}$, which we were able to compute after drawing N

i.i.d. samples, $\{\mathbf{X}_{k-1,i}\}_{i=1..N}$, from the importance density $q(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})$. We would now like to compute the set of new particles, ie, a new set of weights $w_{k,i} \propto \frac{p(\mathbf{X}_{k,i}|\mathbf{Z}_k)}{q(\mathbf{X}_{k,i}|\mathbf{Z}_k)}$, for which we know that every $\mathbf{X}_{k,i}$ should be drawn from the importance density $q(\mathbf{X}_k|\mathbf{Z}_k)$. Let us first consider the numerator of the new weights. We have, using basic probability rules:

$$p(\mathbf{X}_k|\mathbf{Z}_k) = p(\mathbf{X}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}) = \frac{p(\mathbf{X}_k, \mathbf{z}_k, \mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} = \frac{p(\mathbf{z}_k|\mathbf{X}_k, \mathbf{Z}_{k-1})p(\mathbf{X}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \tag{2.15}$$

Which can be further decomposed into:

$$p(\mathbf{X}_k|\mathbf{Z}_k) = \frac{p(\mathbf{z}_k|\mathbf{X}_k, \mathbf{Z}_{k-1})p(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_{k-1})p(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})} \tag{2.16}$$

But the nature of the problem (see equations (2.1) and (2.2)) is such that $p(\mathbf{z}_k|\mathbf{X}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$, and $p(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Recall in the previous section, we stated that the knowledge of the function h, which can be here identified to the distribution $p(\mathbf{X}_k|\mathbf{Z}_k)$, can be known only up to a normalizing constant, still making the approximation possible by normalizing the weights. Thus, we may simply not compute the denominator of (2.16), and simply write:

$$p(\mathbf{X}_k|\mathbf{Z}_k) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1}) \tag{2.17}$$

We have here expressed, up to a proportionality constant, the joint distribution $p(\mathbf{X}_k|\mathbf{Z}_k)$ at time k in terms of the joint distribution at time $k-1$ and known quantities. Consider now the denominator of the new weights, i.e., the importance density. We have:

$$q(\mathbf{X}_k|\mathbf{Z}_k) = q(\mathbf{x}_k, \mathbf{X}_{k-1}|\mathbf{Z}_k) = q(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_k)q(\mathbf{X}_{k-1}|\mathbf{Z}_k) \tag{2.18}$$

Now, as seen in the previous section, we are almost completely free to choose the importance distribution $q(\mathbf{X}_k|\mathbf{Z}_k)$. We are however interested in defining the importance density that will make the resulting procedure tractable. If we pick the importance density such that $q(\mathbf{X}_{k-1}|\mathbf{Z}_k) = q(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})$, then we can write:

$$q(\mathbf{X}_k|\mathbf{Z}_k) = q(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_k)q(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1}) \tag{2.19}$$

The advantage of (2.19) is now as follows: we see that to obtain N samples $\mathbf{X}_{k,i} \sim q(\mathbf{X}_k|\mathbf{Z}_k)$, we could simply keep all the existing samples $\mathbf{X}_{k-1,i} \sim q(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})$, and just draw N new samples $\mathbf{x}_{k,i} \sim q(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_k)$, which would then give samples $\mathbf{X}_{k,i} = (\mathbf{x}_{k,i}, \mathbf{X}_{k-1,i}) \sim q(\mathbf{X}_k|\mathbf{Z}_k)$.

We now substitute (2.17) and (2.19) in the expression $w_{k,i} \propto \frac{p(\mathbf{X}_{k,i}|\mathbf{Z}_k)}{q(\mathbf{X}_{k,i}|\mathbf{Z}_k)}$, and we get:

$$w_{k,i} \propto \frac{p(\mathbf{X}_{k,i}|\mathbf{Z}_k)}{q(\mathbf{X}_{k,i}|\mathbf{Z}_k)} \propto \frac{p(\mathbf{z}_k|\mathbf{x}_{k,i})p(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i})p(\mathbf{X}_{k-1,i}|\mathbf{Z}_{k-1})}{q(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i},\mathbf{Z}_k)q(\mathbf{X}_{k-1,i}|\mathbf{Z}_{k-1})} \tag{2.20}$$

$$\text{and thus} \qquad w_{k,i} \propto w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{x}_{k,i})p(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i})}{q(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i},\mathbf{Z}_k)} \tag{2.21}$$

Finally, if we restrict $q(\mathbf{x}_k|\mathbf{X}_{k-1},\mathbf{Z}_k) = q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k)$, then the previous recursion becomes such that, at each step, we only need to draw samples from $q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k)$, and the weights only depend on single state samples $\mathbf{x}_{k,i}$. Thus we do not have to store the entire history of states, and we also only need the last measurement available. We then obtain the final iterative approximation of $p(\mathbf{x}_k|\mathbf{Z}_k)$:

$$w_{k,i} \propto w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{x}_{k,i})p(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i})}{q(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i},\mathbf{z}_k)} \tag{2.22}$$

where the N samples $\{\mathbf{x}_{k,i}\}_{i=1..N} \sim q(\mathbf{x}_k|\mathbf{x}_{k-1,i},\mathbf{z}_k)$

These weights and support points approximate $p(\mathbf{x}_k|\mathbf{Z}_k)$ as:

$$\hat{p}(\mathbf{x}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} w_{k,i}\delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.23}$$

where the latter is obtained by marginalization of

$$\hat{p}(\mathbf{X}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} \delta(\mathbf{X}_k - \mathbf{X}_{k,i}).w(\mathbf{X}_{k,i}) \tag{2.24}$$

The symbol $\delta(.)$ is used in both cases, for the multidimensional and unidimensional Delta functions, however there should be no ambiguities. This marginalization comes from the property of the multidimensional Delta function that verifies, for $\mathbf{x} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^n$, we have:

$$\delta(\mathbf{x}) = \delta(x_1)\delta(x_2)...\delta(x_n) \tag{2.25}$$

We note again that the approximation (2.23) is the distribution of a random variable that can only take the values $\mathbf{x}_{k,i}$, and for which we have

$$P(\mathbf{x}_k = \mathbf{x}_{k,i}) = w_{k,i} \tag{2.26}$$

Once the weights are computed, then at each step, based on (2.23) we can compute the estimate of our choice for the state at time k. For example, one may compute the expectation of the random variable whose distribution is (2.23). The conditional mean estimate of the state at time k is then equal to:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} w_{k,i}\mathbf{x}_{k,i} \tag{2.27}$$

2.2.2 The resampling step

Unfortunately, as such, (2.22) alone cannot be employed iteratively to implement a so-called particle filter. The problem comes from the fact that as the algorithm iterates, if left alone the particles will keep "dispersing" in the state space at each step, and fewer and fewer will remain plausible candidates for the true states conditioned on the observations, especially if a poorly chosen importance density is used (see 2.2.4) [3,7,8,18,47]. Most of them will have almost zero weight. This means that only a few particles (eventually one of them) will contain the useful information, leading to a very poor approximation of the posterior density. This problem is sometimes called the *degeneracy* problem. To solve this unavoidable practical problem, the common solution is termed the *resampling step* [3,18,47].

Intuitively, to reduce degeneracy one should ensure that at each step, the set of particles are not in a region of the state space that is completely irrelevant. One idea to do so is to resample N support points $\mathbf{x}_{k,i}$ from the distribution (2.23) at the end of each step where we find it necessary, ie, where the degeneracy is found to be too severe. Technically, drawing new such samples for $\mathbf{x}_{k,i}$ will make sure that the particles are "pertinent" at all steps, and are located in a region where the weights are not negligible.

We have stated that this resampling step should be taken whenever the degeneracy is too severe. Thus, there is a need for a measure of the degeneracy of the particles. A common such measure found in the literature is the effective sample size $N_{eff}(k)$, which reflects, at step k, how many of the N particles employed in the process are really meaningful [47]. It is computed as follows:

$$N_{eff}(k) = \left[\sum_{i=1}^{N}(w_{k,i})^2\right]^{-1} \tag{2.28}$$

The best possible case occurs when $N_{eff}(k)$ is equal to N, in which case all particles have equal weights, and a lower value indicates a higher level of degeneracy.

Various resampling algorithms exist [3, 4, 8, 47], one of the most important aspects being efficiency. The common goal is to find a way to keep the effective sample size $N_{eff}(k)$ as large as possible. Resampling may be purely stochastic (where random number generation is involved), deterministic (where the elimination/multiplication of the number of light/heavy particles is performed according to a deterministic function), and some algorithms may combine both approaches. As an introduction, we present here one possible view of a stochastic resampling step.

In the particular resampling procedure that we describe here, the first step consists in build-

ing a cumulative sum of weights for the normalized particles (this is equivalent to constructing the cumulative probability mass function of the random variable characterized by (2.23), ie, the function obtained is the integral of (2.23)). Then, to obtain the i^{th} resampled particle $\mathbf{x}_{k,i}^*$, we generate a uniform random number u_i between 0 and 1. Each of the discontinuities of the cumulative sum of weights have a size that is exactly equal to each of the weights, and the number u_i will fall in one of these discontinuities, say, corresponding to weight $w_{k,j}$, or interval $[a, b]$ with $0 \leq a \leq b \leq 1$ and $b - a = w_{k,j}$. But then, clearly $P(u_i \in [a, b]) = w_{k,j}$, in other words, the larger the weight, the better chance it is selected. Finally, the new sample $\mathbf{x}_{k,i}^*$ is chosen to be equal to $\mathbf{x}_{k,j}$, and we associate to it a weight $1/N$, producing a new particle. To summarize this step, it could be said that the goal is to eliminate particles with low weights, and select several times particles with high weights.

Unfortunately, resampling has a drawback, which we will discuss in section 2.2.4.

2.2.3 A practical algorithm: the SIR particle filter

The practical particle filtering algorithm, referred to as the Sampling Importance Resampling algorithm (SIR) [7,8,18], is presented in Algorithm 1.

Algorithm 1 The Sampling Importance Resampling Particle Filter (SIR PF) algorithm

For every k, do the following:

- For every $i \in 1, 2, .., N$
  - draw $\mathbf{x}_{k,i} \sim q(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$
  - use the $\mathbf{x}_{k,i}$ above to compute the unnormalized weights
    $$\widetilde{w}_{k,i} = w_{k-1,i} \frac{p(\mathbf{z}_k|\mathbf{x}_{k,i}) p(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i})}{q(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i}, \mathbf{z}_k)}$$
- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize each of the weights to obtain $\{w_{k,i}\}_{i=1}^{N}$.
- If needed (as indicated by the effective sample size), resample each of the particles

The SIR particle filter of Algorithm 1 is a fully functional particle filtering algorithm, and it can be directly applied to a multitude of problems. For example, an entire book, [47], is dedicated to tracking applications of particle filters, many of which can directly employ a SIR PF.

Behind the algorithm and the theory described earlier, one may see particle filtering as a

"brute force" simulation, with many trials and errors. At each step, we select a large number of possible candidates for the state. We obtain a measurement, and then we assign scores to the candidates, depending on how well they fit the measurement (and in fact, the sequence of past measurements). Only the fittest candidates survive onto the next step, and the other ones are discarded. This is also why Particle Filtering is also called "Survival of the fittest" [47].

2.2.4 Problems with PF algorithms

Computationally expensive

Even though, in theory, particle filtering offers the most flexibility and the weakest assumptions on the state-space model, one must be aware that it is also the most expensive approach in terms of computational load (compared to, for example, a regular KF, EKF, UKF). In many situations, in order to obtain reliable estimates for the states (i.e., with small variance of estimation error), a large number of particles is required. It has been found to be the case especially when the dimension of the state vector is large. A challenge in implementation is then to try to use as few particles as possible. To reach that goal, a critical step is the choice of an appropriate importance density, as we will see in the next section. An important modification of the generic particle filter, called the Rao-Blackwellised particle filter (RBPF), can reduce drastically the number of particles needed for a certain class of state-space modes. The RBPF is an important improvement to the basic filtering scheme, since the same accuracy as the regular PF can be reached using fewer particles, even though more operations must be done per particle. Globally, the accuracy is improved enough to make the tradeoff to the advantage of RBPFs. Chapter 3 is dedicated to this type of PF.

As a conclusion, in practice, it is highly recommended to experiment on the performance of traditional filters before implementing a particle filter. For example, for a linear system but with non-Gaussian noise, a pure Kalman filter may perform acceptably, depending on the problem requirements.

Choice of the importance density

At the heart of the particle filter algorithm, the importance density directly impacts, at each step, the variance of the importance weights, as shown by (2.22). Recall from our previous discussion that the only reason we must pick an importance density $q(.)$ is for sampling purposes, since we assume that we cannot draw from the posterior density. As it can be intuitively seen, it turns out that this posterior density $p(\mathbf{X}_k|\mathbf{Z}_k)$ is the optimal one, that is, the one that minimizes the variance of the importance weights. Following the derivation of the SIR PF algorithm (Algorithm 1) with importance density $q(\mathbf{X}_k|\mathbf{Z}_k)$, we see that at each step, optimally one

should then draw N new samples from $p(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$. If that were possible, the computation of the unnormalized importance weights would then reduce to:

$$\widetilde{w}_{k,i} = w_{k-1,i} p(\mathbf{z}_k|\mathbf{x}_{k-1,i}) \tag{2.29}$$

This means that one must be able not only to sample from $p(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$, but also to evaluate $p(\mathbf{z}_k|\mathbf{x}_{k-1,i})$. It is unfortunately not usually the case, and then it is still possible to use the filter with a suboptimal choice. It is however possible to use the optimal importance density when the state-space model has the following form (called *Partial Gaussian* [7]):

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1} \tag{2.30}$$

$$\mathbf{z}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k \tag{2.31}$$

where both the process noise $\mathbf{w}_k$ and the measurement noise $\mathbf{v}_k$ are zero-mean Gaussian.

It can be shown that $p(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$ and $p(\mathbf{z}_k|\mathbf{x}_{k-1,i})$ are Gaussian, and can be expressed in terms of the covariance matrices of the noise processes [7].

In most other cases, a suboptimal importance density must be picked. Many particle filter implementations use $q(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1,i})$ [8,47]. The use of this simple distribution simplifies the algorithm, however it can lead to disappointing results, because it means that we are drawing at each step some particles *regardless* of the current observation. If possible, it is preferable to incorporate the current measurement z_k somehow in the importance function.

Issues related to basic resampling

Even though the algorithm presented earlier is operational, the resampling step in turn introduces non-negligible problems. The resampling phase is such that the particle filter will mostly operate on the same particles in time, since these are more likely to be re-selected at each step. This means that the diversity of the set of particles is poor, and is likely to get poorer and poorer [8,47]. In other words, the set of particles, which is sometimes termed the *particle cloud*, may collapse to very few different points. This is sometimes called *sample depletion* or *sample impoverishment*. Different approaches have been suggested to solve that problem, such as the introduction of a Markov Chain Monte Carlo step, or a regularization step (see 2.3).

2.2.5 A more general form for the SIR PF algorithm

In the SIR PF derivation, recall that we had used the initial state-space model to simplify our task. More specifically, we had stated that $p(\mathbf{z}_k|\mathbf{X}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$, and $p(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_{k-1}) =$

$p(\mathbf{x}_k|\mathbf{x}_{k-1})$ from the nature of the problem. It is possible not to make any of these assumptions, and still derive a more general particle filtering algorithm. As we will see in chapter 3, it is sometimes advantageous to only consider certain elements of the state vector, and these elements alone do not follow the original state-space model. Recall also that we had restricted, during the derivation of the algorithm, the importance density to verify $q(\mathbf{x}_k|\mathbf{X}_{k-1}, \mathbf{Z}_k) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{Z}_k)$. Again, it may not always be wise to do so. In the case where only certain elements of the state vector are taken into account, the optimal importance density would not necessarily verify this restriction. The only constraint that we keep for the derivation is that $q(\mathbf{X}_{k-1}|\mathbf{Z}_k) = q(\mathbf{X}_{k-1}|\mathbf{Z}_{k-1})$. In practice, this restriction does apply for any reasonable importance density we might think of.

Following the exact same steps for the SIR algorithm derivation, but with these less restrictive assumptions, we find that the weights, at each step, should be updated as:

$$\widetilde{w}_{k,i} = w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{X}_{k,i}, \mathbf{Z}_{k-1})p(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_{k-1})}{q(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)} \tag{2.32}$$

We can rewrite the SIR PF algorithm in the general form in Algorithm 2.

Algorithm 2 A more general form for the SIR PF algorithm

For every k, do the following:

- For every $i \in 1, 2, .., N$
  - draw $\mathbf{x}_{k,i} \sim q(\mathbf{x}_k|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)$
  - set $\mathbf{X}_{k,i} = \{\mathbf{x}_{k,i}\ ; \mathbf{X}_{k-1,i}\}$
  - use the $\mathbf{X}_{k,i}$ above to compute the unnormalized weights
    $\widetilde{w}_{k,i} = w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{X}_{k,i}, \mathbf{Z}_{k-1})p(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_{k-1})}{q(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)}$
- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize each of the weights to obtain $\{w_{k,i}\}_{i=1}^{N}$
- If needed, resample each of the particles

For the general SIR PF of Algorithm 2, the optimal importance density is then $p(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)$, which reduces to $p(\mathbf{x}_{k,i}|\mathbf{x}_{k-1,i}, \mathbf{Z}_k)$ for the state-space model described in (2.1) and (2.2). Finally, we conclude with another remark that is also applicable for the "reduced" SIR PF of Algorithm 1: if resampling is chosen to be done at each step, then it is not necessary to multiply each of the new unnormalized weights by the previous values of the weights, since they are all equal after the resampling step.

2.3 Particle Filters improvements

2.3.1 Markov Chain Monte Carlo moves

Before explaining why introducing such methods is beneficial to the particle filter algorithm, let us recall certain facts about Markov chain theory.

By definition, a Markov chain is a discrete random process $\mathbf{X}_n$ such that for every i, $\mathbf{X}_i$ is independent of all $\{ \mathbf{X}_j \mid j \leq (i-2) \}$ conditioned on $\mathbf{X}_{i-1}$. The evolution in time of a Markov chain is determined by its *transition kernel*, which is defined as $P(\mathbf{X}_{n+1} \in A \mid \mathbf{X}_n = x)$, and denoted $K_n(x, A)$, for every measurable set A. For a *homogeneous* Markov chain, we have $K_n(x, A) = K(x, A)$ is independent of n. Assume from now on that the Markov chains we deal with are homogeneous. We may then conveniently express the transition kernel in terms of its density $k(x, y) = K(x, dy)$ which is such that:

$$K(x, A) = \int_{y \in A} k(x, y) dy = \int_{y \in A} K(x, dy) \tag{2.33}$$

A distribution $\pi(x)$ is said to be *invariant* or *stationary* under $K(x, dy)$ if:

$$\pi(dy) = \int K(x, dy)\pi(x)dx, \text{ or } \pi(A) = \int K(x, A)\pi(x)dx \tag{2.34}$$

The distribution $\pi(x)$ is of major interest: it is such that if a certain state is itself drawn from it, then the Markov chain with $\pi(x)$ as an invariant distribution and transition kernel $K(x, dy)$ will simply generate new state sequences that are also drawn from $\pi(x)$. Moreover, under some conditions, a Markov chain is said to "converge" to its stationary distribution, that is, from almost every initial state, after a certain number of steps (which is called the *burn-in* time), the states appear to be drawn from the stationary distribution. We state here a sufficient condition on the Markov chain with kernel $K(x, dy)$ for $\pi(x)$ to be a stationary distribution:

$$\pi(x)k(x, y) = \pi(y)k(y, x) \tag{2.35}$$

The previous condition is called the *reversibility* property.

Markov Chain Monte Carlo (MCMC) methods are often employed to generate samples from an arbitrary distribution $p(x)$. In the light of the previous talk, the idea is to create a Markov chain (i.e., design a transition Kernel) whose stationary distribution is $p(x)$. Then, given an initial state on the chain, the following states will converge to samples of $p(x)$. One drawback is the burn-in time, which could make the samples generation expensive. First, convergence may take several iterations, and second, there may not be ways to ensure that convergence has actually been achieved. However, from one state to the next, applying the transition kernel will

move towards region of the state space that is "closer" to the region supporting distribution $p(x)$.

There exist several methods for "designing" chains given a target stationary distribution. We mention two of them here: the *Metropolis-Hastings* algorithm and the *Gibbs sampling* algorithm.

In the context of particle filters, an MCMC move step can be introduced after the resampling step, in order to reduce the problem of sample impoverishment. The idea is to apply to the resampled set of particles a Markov chain transition Kernel with stationary distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$. Since the particles are already approximately distributed according to the posterior distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$, then this procedure will have generated a new set of particles, also distributed according to $p(\mathbf{x}_k|\mathbf{Z}_k)$. It may seem that there is nothing changed, but in fact there are two major benefits. First, it will effectively "jitter" the set of particles, thereby making the set of particles more diverse and healthier. Secondly, again it can be shown that the application of an MCMC step can only move the particles closer to the target distribution, increasing the pertinence of each of the particles.

Unfortunately, there is a drawback from using MCMC steps: in general they introduce a considerable additional computational burden, which is already a problem for plain particle filters.

2.3.2 Regularized particle filters

The goal of regularized particle filters is also to minimize sample impoverishment arising during the resampling step of the SIR PF algorithm. The idea is simple: instead of resampling the particles from the discrete approximation:

$$\hat{p}(\mathbf{x}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} w_{k,i}\delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.36}$$

we use a continuous approximation for this density, as follows:

$$\tilde{p}(\mathbf{x}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} w_{k,i}\psi(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.37}$$

where $\psi(\mathbf{x}) = \frac{1}{h^n}K(\frac{\mathbf{x}}{h})$, n is the dimension of the state $\mathbf{x}$, h is a positive number called the Kernel bandwidth, and $K(.)$, the Kernel density, is a symmetric density satisfying the two criteria [8]:

$$\begin{aligned} \sum \mathbf{x} K(\mathbf{x}) d\mathbf{x} &= 0 \\ \sum |\mathbf{x}|^2 K(\mathbf{x}) d\mathbf{x} &< \infty \end{aligned}$$

The Kernel density and the bandwidth must be chosen to try to minimize the mean integrated squared error between the true density $p(\mathbf{x}_k|\mathbf{Z}_k)$ and $\tilde{p}(\mathbf{x}_k|\mathbf{Z}_k)$ [47]. Some possible discusses choices for them are presented in [47]. Figure 2.2 illustrates the effect of the regularization step on a discrete distribution (such as $\hat{p}(\mathbf{x}_k|\mathbf{Z}_k)$).

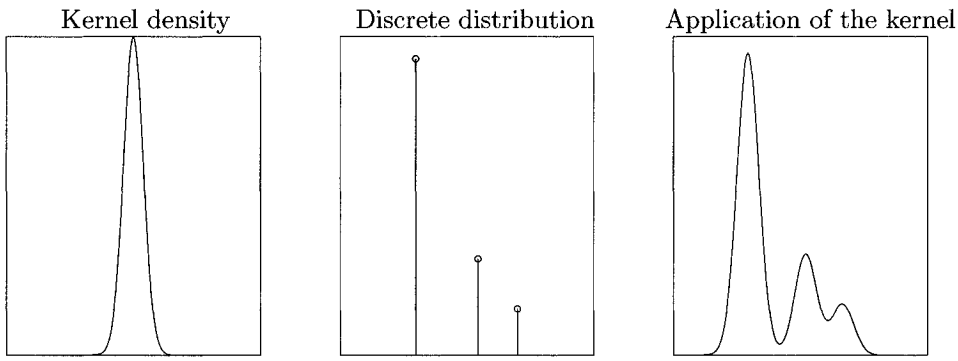


Figure 2.2: Effect of the regularization step on a discrete distribution
Effectively, this step consists of a convolution of the discrete posterior approximation with a continuous, symmetric Kernel density.

The benefits of using such a regularized resampling step is case-dependent, but it is often found to improve the estimates to a certain extent. The main drawback is there is no theoretic proof that the particles resampled from the regularized posterior density are indeed a valid approximation for the true posterior. But since it only introduces a small additional computational load and it is easy to add/remove from the SIR PF algorithm, it is usually worthwhile to implement the regularized PF, at least for testing purposes.

2.3.3 Smoothing techniques

We mention here two beneficial techniques to improve the quality of the estimates returned by particle filters. As we mentioned in the introduction, smoothing consists of estimating a sequence of states not only given the sequence of measurements up to the present time index, but also given a certain amount of measurements in the future. In concrete terms, the pros and cons of smoothing can be formulated in the following reasonable statement: the use of extra

information can only help the estimation, although processing more information requires more computations.

Among the several smoothing strategies in the literature, we choose to present the one given in [16]: it is a theoretically validated, fairly efficient and simple procedure to draw independent realizations from the joint, *smoothing* density $p(\mathbf{X}_k|\mathbf{Z}_k)$ (as opposed to the *filtering* density $p(\mathbf{x}_k|\mathbf{Z}_k)$).

Recall that in our presentation of the rationale for the PF algorithm (specifically in section 2.2.1), we had stated that the entire history of particles does not need to be stored in memory, and that we can only keep the most recent particles if our goal is the estimation of $p(\mathbf{x}_k|\mathbf{Z}_k)$. However, going through section 2.2.1, there is nothing theoretically preventing us from keeping this entire history, thereby obtaining an estimate for $p(\mathbf{X}_k|\mathbf{Z}_k)$. This is true, but it is not realistically feasible in practice due to the problem of *sample impoverishment*, related to the necessary resampling step in PFs (see 2.2.4). This problem is all the more severe for the history, or the trail of particles. Indeed, as time goes by, the particles $\{\mathbf{X}_{k,i}, w_{k,i}\}_{i=1}^N$ (corresponding to the trajectories $\{\mathbf{x}_{0,i}, \mathbf{x}_{1,i}, \ldots, \mathbf{x}_{k,i}\}_{i=1}^N$) are resampled many times: rapidly all of the past components of the particles collapse to a single trajectory, and only the recent components of the particles are able to retain a certain degree of diversity. It is therefore unrealistic to represent the entire distribution $p(\mathbf{X}_k|\mathbf{Z}_k)$ with such non-diversity (just like it would be unrealistic to represent $p(\mathbf{x}_k|\mathbf{Z}_k)$ with only a few *distinct* particles). This phenomenon is illustrated in [16].

The approach used in the algorithm of [16] is a "forward-filtering backwards smoothing" procedure. This means that, to be able to draw sample realizations from $p(\mathbf{X}_K|\mathbf{Z}_K)$, $K > 0$, we must have already completed a forward pass of (particle) filtering, and we have at hand for every $k \in \{0, \ldots, K\}$ and every $i \in \{1, \ldots, N\}$, a pair $\{\mathbf{x}_{k,i}; w_{k,i}\}$. The article then shows that an approximation of $p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{Z}_K)$ is given by:

$$p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{Z}_K) \simeq \sum_{i=1}^{N} w_{k|k+1,i}\delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.38}$$

where the modified weights $w_{k|k+1,i}$ are functions of $\mathbf{x}_{k+1}$ and can be obtained via:

$$w_{k|k+1,i} = \frac{w_{k,i}p(\mathbf{x}_{k+1}|\mathbf{x}_{k,i})}{\sum_{j=1}^{N} w_{k,j}p(\mathbf{x}_{k+1}|\mathbf{x}_{k,j})} \tag{2.39}$$

Now, we can also write:

$$\begin{aligned} p(\mathbf{X}_K|\mathbf{Z}_K) &= p(\mathbf{x}_K|\mathbf{Z}_K) \prod_{k=1}^{K-1} p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_K, \mathbf{Z}_K) \\ &= p(\mathbf{x}_K|\mathbf{Z}_K) \prod_{k=1}^{K-1} p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{Z}_K) \end{aligned} \tag{2.40}$$

The procedure to obtain one sample realization from $p(\mathbf{X}_K|\mathbf{Z}_K)$ now takes place as follows. Starting with a sample $\widetilde{\mathbf{x}}_K$ approximately drawn from $p(\mathbf{x}_K|\mathbf{Z}_K)$ (using the results from the forward filtering pass), we use equation (2.38) to draw a sample $\widetilde{\mathbf{x}}_{K-1}$ from (an approximation of) $p(\mathbf{x}_{K-1}|\mathbf{x}_K, \mathbf{Z}_K)$. Equation (2.40) then shows that $\{\widetilde{\mathbf{x}}_{K-1}, \widetilde{\mathbf{x}}_K\}$ is draw approximately from $p(\mathbf{x}_{K-1}, \mathbf{x}_K|\mathbf{Z}_K)$. We can continue this procedure backwards in time until we reach the initial instant. The resulting algorithm is given in Algorithm 3.

Algorithm 3 Sampling of a realization from the smoothing posterior density

To obtain a sample realization $\widetilde{\mathbf{X}}_K$ from $p(\mathbf{X}_K|\mathbf{Z}_K)$, $K > 0$:

- Perform particle filtering for $k = 1$ to K, using Algorithm 1, storing all the history of weights and particles.
- Draw $\widetilde{\mathbf{x}}_K$ from the discrete approximation for $p(\mathbf{x}_K|\mathbf{Z}_K)$ returned by the PF algorithm.
- For $k = K - 1$ to 1,
  - $\forall i \in \{1, \ldots, N\}$, compute $w_{k|k+1,i} \propto w_{k,i} p(\widetilde{\mathbf{x}}_{k+1}|\mathbf{x}_{k,i})$
  - Normalize the modified weights $\{w_{k|k+1,i}\}_{i=1}^N$.
  - Using the set of normalized weights, draw $\widetilde{\mathbf{x}}_k$ from the discrete approximation for $p(\mathbf{x}_k|\widetilde{\mathbf{x}}_{k+1}, \mathbf{Z}_K)$

Algorithm 3 can be repeated as many times as desired, to obtain as many sample realizations as needed. In [16], to improve the PF estimates, several realizations are drawn and then an average is computed. In Figure 2.3, we show a simulation screenshot illustrating this situation.

In [16], simulation results also show that the average smoothed results are better than the filtered estimates. The major drawback from this type of method is the fact that it is much more demanding computationally than just plain particle filtering. In algorithm 3, to draw a single realization, in the loop iterating from $K - 1$ to 1, there is also a subloop going through every i. Moreover, our experience is that a fairly large number of particles is required for the improvement to be really worthwhile. Still, if the goal is to study a complex multivariate joint distribution, then as stated in [16], generating sample realizations is an insighful and intuitive

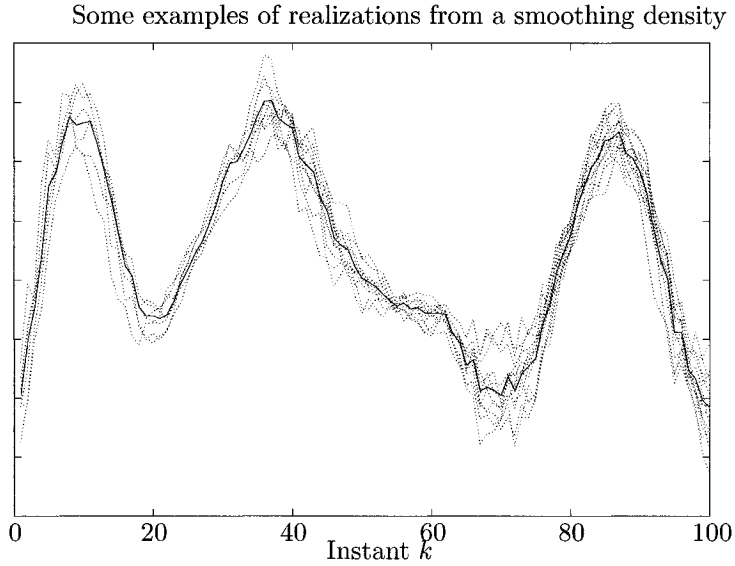


Figure 2.3: An example of the utilization of Algorithm 3
The blue, dotted lines are independent draws from a smoothing density. The red line is the average of these realizations, which is found to be closer to the true value (in the mean-square sense) than the direct filtering estimates.

approach.

There is a simple alternative, which does not allow as such the generation of smoothed realizations, but still uses information from the future to compute its estimates. It still improves the quality of the estimates, but there is no additional computational cost (the only cost is counted in terms of memory). This very simple method is called *fixed-lag smoothing* [9, 50]. Recall the previous discussion in this section, where we said that an estimate for $p(\mathbf{X}_k|\mathbf{Z}_k)$ can be theoretically obtained with the very same method as the one described in 2.2.1, although we also stated the poorness of the resulting approximation, due to the problem of sample impoverishment. This estimate is the following, given here at a given time K:

$$p(\mathbf{X}_K|\mathbf{Z}_K) \simeq \sum_{i=1}^{N} w_{K,i}\delta(\mathbf{X}_K - \mathbf{X}_{K,i}) \tag{2.41}$$

but again, the problem is that the history of $\mathbf{X}_{K,i}$ rapidly collapses to a single trajectory. The idea of fixed-lag smoothing is that the recent trail is hopefully still diverse enough. Suppose that, given a problem context, it is found that the particle diversity is still maintained for the last L values $\{\mathbf{x}_{m,i}\}_{m=K-L+1}^{K}$. Then, by marginalizing (2.41) and picking $k = K - L$, we get:

$$p(\mathbf{x}_k|\mathbf{Z}_{k+L}) \simeq \sum_{i=1}^{N} w_{k+L,i}\delta(\mathbf{x}_k - \mathbf{x}_{k,i}) \tag{2.42}$$

Equation (2.42) is the fixed-lag smoothing equation. For the reasons discussed above, it tends to perform poorly as L increases [9], however for small enough L (the value depends on many parameters: the number of particles, the frequency of resampling, and of course the problem itself) we can expect an improved performance. For example, in chapter 5 we use this approach almost systematically in the context of speech enhancement for $L = 8$.

2.3.4 The extended and the unscented particle filters

We only briefly mention these methods here, since we have chosen not to utilize them in the next chapters. Moreover, a detailed description of these algorithms is available in [24]. In the Extended Particle Filter (EPF) and the Unscented Particle Filter (UPF), the goal is to approximate the optimal importance density $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$ when it is not tractable, rather than using the transition prior $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. The resulting approximation is Gaussian, and it is propagated using the EKF (for the EPF) or the UKF (for the UPF).

The UPF (or the EPF) is therefore simply a particle filter coupled with a UKF (or an EKF) whose goal is to generate a (Gaussian) approximation to the optimal importance function. In many cases, this has been shown to improve significantly the resulting estimates [24]. Unfortunately the computational overhead is relatively important, especially if the state to estimate has a high dimension, and if the number of particles is large. Indeed, a separate UKF/EKF must be used to generate, for all i and at each time k, the distribution $q(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$. In the UPF, this also means that a large amount of memory must be temporarily allocated for this purpose, since the UKF requires several deterministically chosen draws for the state to operate.

Again, we do not exclude the use of these techniques in future work, but we have not made their implementation the primary goal of this thesis.

Chapter 3

Rao-Blackwellised Particle Filters

3.1 The main idea

As we noted in section (2.2.4), particle filtering methods are by nature computationally expensive. It is particularly true when the dimension of the state vector is large, as it may be the case in speech processing applications for example (as will be seen later on). In such cases, a large number of particles is typically required to achieve a performance that would justify the use of particle filtering. Rao-Blackwellised Particle Filters (RBPF) can be seen as a form of "enhanced" PF which can allow, if applicable, the use of much less particles to obtain similar performance, even though more computations must be done per particle. The idea is to exploit the structure of the state vector. If some conditional dependencies between elements of the state vector can be analytically explicited, then there is no need to sample all of these elements. We will here present the main idea, and a more precise discussion will follow. Consider the d^{th}-dimensional state vector $\mathbf{X}_k$, which we partition into two substates $\{\mathbf{X_{1}}_k \, ; \mathbf{X_{2}}_k\}$. From Bayes' rule, we can always write:

$$p(\mathbf{X}_k|\mathbf{Z}_k) = p(\mathbf{X_{1}}_k, \mathbf{X_{2}}_k|\mathbf{Z}_k) = p(\mathbf{X_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)p(\mathbf{X_{1}}_k|\mathbf{Z}_k) \tag{3.1}$$

If it is possible, from the structure of the problem, to compute analytically the density $p(\mathbf{X_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)$, then we can run a particle filter on $p(\mathbf{X_{1}}_k|\mathbf{Z}_k)$ while updating $p(\mathbf{X_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)$ to obtain an estimate for the "joint" posterior $p(\mathbf{X}_k|\mathbf{Z}_k)$. As we pointed out in section (2.2.4), in general the number of particles needed to obtain reliable estimates increases as the dimension of the state vector increases. But using the idea presented here, since the dimension of $\mathbf{X_{1}}_k$ will be smaller than d, we are sampling in a smaller space, and thus we can expect to require less particles than for a regular PF. In addition, a "portion" of the posterior, namely $p(\mathbf{X_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)$, is computed analytically. This procedure, known as Rao-Blackwellisation, has been shown to sharply reduce the variance of the error estimates (one can refer to Rao-Blackwell's theorem,

showing conditions for finding a reduced-variance unbiased estimator given another unbiased estimator [10]).

A graphical model describing a generic RBPF setting

The setting for applying RBPF can be described by the Dynamic Bayesian Network of Figure 3.1.

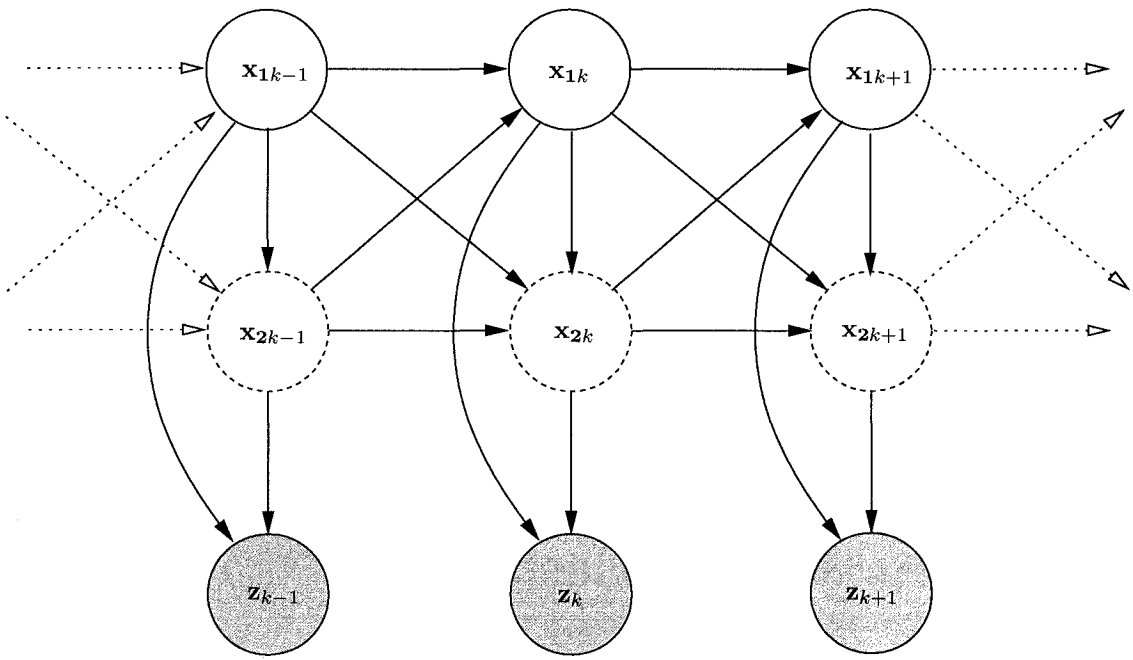


Figure 3.1: Graphical model for the dynamic system where RBPF can be applied
The nodes on the top row are sampled from, and the nodes on the bottom row are observed.

In this graph, note that applying the Bayes Ball algorithm can lead to certain interesting relations. For example, suppose we are interested in computing $p(\mathbf{x}_{1k}|\mathbf{X}_{1k-1}, \mathbf{x}_{2k-1}, \mathbf{Z}_{k-1})$. We then directly see from the graph that, conditioning on these nodes yields:

$$p(\mathbf{x}_{1k}|\mathbf{X}_{1k-1}, \mathbf{x}_{2k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_{1k}|\mathbf{x}_{1k-1}, \mathbf{x}_{2k-1}) \tag{3.2}$$

Other relations of the same type can be deduced.

3.2 The RBPF algorithm

3.2.1 The generic algorithm

The RBPF algorithm consists of running a particle filter on a subspace of the original state-space, but also an extra step must be included. For every particle of $\mathbf{x_{1}}_{k,i}$ that is drawn from the importance distribution, we must compute the entire corresponding conditional density $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$. Clearly, for this to be worthwhile, it is strongly preferable that this computation can be carried out online (that is, that the distribution can be easily updated from $p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1})$), and that the conditional densities $\{p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)\}_{i=1}^N$ can be parametrized by as small of a set as possible in order to reduce memory requirements.

The ultimate goal is to obtain an estimate of the state, as we did using equation (2.27). Using the generic SIR PF (Algorithm 2) which was presented in section 2.2.5, we are able to find a set of N particles $\{\mathbf{X_{1}}_{k,i}\ ;\ w_{k,i}\}$ that approximate $p(\mathbf{X_{1}}_k|\mathbf{Z}_k)$ as:

$$\hat{p}(\mathbf{X_{1}}_k|\mathbf{Z}_k) = \sum_{i=1}^{N} w_{k,i}\delta(\mathbf{X_{1}}_k - \mathbf{X_{1}}_{k,i}) \tag{3.3}$$

Using our knowledge of the structure of the model, for every "incomplete" particle $\{\mathbf{X_{1}}_{k,i}\ ;\ w_{k,i}\}$, we can compute $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$. This is then enough to determine an approximation of $p(\mathbf{x_{2}}_k|\mathbf{Z}_k)$, since we have:

$$p(\mathbf{x_{2}}_k|\mathbf{Z}_k) = \int p(\mathbf{X_{1}}_k, \mathbf{x_{2}}_k|\mathbf{Z}_k)d\mathbf{X_{1}}_k \tag{3.4}$$

$$= \int p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)p(\mathbf{X_{1}}_k|\mathbf{Z}_k)d\mathbf{X_{1}}_k \tag{3.5}$$

And therefore, from equation (3.3), we can now write:

$$\begin{aligned} p(\mathbf{x_{2}}_k|\mathbf{Z}_k) &\approx \int p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k) \sum_{i=1}^{N} w_{k,i}\delta(\mathbf{X_{1}}_k - \mathbf{X_{1}}_{k,i})d\mathbf{X_{1}}_k \\ &= \sum_{i=1}^{N} w_{k,i} \int p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)\delta(\mathbf{X_{1}}_k - \mathbf{X_{1}}_{k,i})d\mathbf{X_{1}}_k \\ &= \sum_{i=1}^{N} w_{k,i}p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k) \end{aligned} \tag{3.6}$$

Thus, the conditional mean estimate of $\mathbf{x_{2}}_k$ can be computed as:

$$\mathcal{E}\{\mathbf{x_{2}}_k|\mathbf{Z}_k\} \approx \sum_{i=1}^{N} w_{k,i}\mathbf{x_{2}}_{k,i} \tag{3.7}$$

where $\mathbf{x}_{\mathbf{2}k,i} = \mathcal{E}\{\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i}, \mathbf{Z}_k\} = \int \mathbf{x}_{\mathbf{2}k}\, p(\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i}, \mathbf{Z}_k)\, d\mathbf{x}_{\mathbf{2}k}$.

Recall that we also have, from (3.3):

$$\mathcal{E}\left\{\mathbf{x}_{\mathbf{1}k}|\mathbf{Z}_k\right\} \approx \sum_{i=1}^{N} w_{k,i}\mathbf{x}_{\mathbf{1}k,i} \tag{3.8}$$

Therefore, we can still compute the conditional mean estimate of the whole state as:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} w_{k,i}\mathbf{x}_{k,i} \tag{3.9}$$

In summary, to implement the Rao-Blackwellised PF, we must not only run a particle filter for the estimation of $p(\mathbf{x}_{\mathbf{1}k}|\mathbf{Z}_k)$, but also be able to recursively update the distributions $p(\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i}, \mathbf{Z}_k)$ for every i (if this is done as new particles $\mathbf{x}_{\mathbf{1}k,i}$ and new measurements $\mathbf{z}_k$ arrive, then we do not necessarily need to keep the history $\mathbf{X}_{\mathbf{1}k}$ if we are only interested in filtering). The update of $p(\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i}, \mathbf{Z}_k)$ that must be analytically carried out in the RBPF algorithm is precisely the additional step that must be conducted compared to the regular particle filtering method, and it is called the *exact step*.

The generic RBPF algorithm can now be formulated in Algorithm 4, and more details of the implementation will follow. Note that the first part of the algorithm is identical to the generic SIR PF presented in Algorithm 2, in section 2.2.5.

3.2.2 Updating the weights

When running the regular PF to estimate $p(\mathbf{X}_{\mathbf{1}k}|\mathbf{Z}_k)$, to update the weights we see that we must be able to compute $p(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1})$.

We have:

$$p(\mathbf{x}_{\mathbf{1}k}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{x}_{\mathbf{1}k}, \mathbf{x}_{\mathbf{2}k-1}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x}_{\mathbf{2}k-1} \tag{3.10}$$

Which leads to:

$$p(\mathbf{x}_{\mathbf{1}k}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{x}_{\mathbf{1}k}|\mathbf{x}_{\mathbf{2}k-1}, \mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) p(\mathbf{x}_{\mathbf{2}k-1}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x}_{\mathbf{2}k-1} \tag{3.11}$$

But we assume that we know $p(\mathbf{x}_{\mathbf{2}k-1}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1})$, so that the only term that needs to be explicited is $p(\mathbf{x}_{\mathbf{1}k}|\mathbf{x}_{\mathbf{2}k-1}, \mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1})$. But we can here use equation (3.2), and we then obtain:

$$p(\mathbf{x}_{\mathbf{1}k}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{x}_{\mathbf{1}k}|\mathbf{x}_{\mathbf{2}k-1}, \mathbf{x}_{\mathbf{1}k-1,i}) p(\mathbf{x}_{\mathbf{2}k-1}|\mathbf{X}_{\mathbf{1}k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x}_{\mathbf{2}k-1} \tag{3.12}$$

Algorithm 4 The generic RBPF algorithm

For every k, do the following:

- For every $i \in 1, 2, .., N$
  - draw $\mathbf{x_{1}}_{k,i} \sim q(\mathbf{x_{1}}_k | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_k)$
  - set $\mathbf{X_{1}}_{k,i} = \{\mathbf{x_{1}}_{k,i} \; ; \mathbf{X_{1}}_{k-1,i}\}$
  - use the $\mathbf{X_{1}}_{k,i}$ above to compute the unnormalized weights

$$\widetilde{w}_{k,i} = w_{k-1,i} \frac{p(\mathbf{z}_k | \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) p(\mathbf{x_{1}}_{k,i} | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1})}{q(\mathbf{x_{1}}_{k,i} | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_k)}$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize each of the weights
- If needed (as indicated by $N_{eff}(k)$), resample each of the particles
- **Exact step**: For every i, update $p(\mathbf{x_{2}}_k | \mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ using $p(\mathbf{x_{2}}_{k-1} | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1})$, $\mathbf{x_{1}}_{k,i}$, $\mathbf{x_{1}}_{k-1,i}$, and $\mathbf{z}_k$.
- **MCMC move step** (Optional): Apply a Markov transition kernel, with stationary distribution $p(\mathbf{X_{1}}_k | \mathbf{Z}_k)$ to update the support points $\mathbf{X_{1}}_{k,i}$.

The term $p(\mathbf{x_{1}}_k | \mathbf{x_{2}}_{k-1}, \mathbf{x_{1}}_{k-1,i})$ can be deduced from marginalization of the state-space model transition equation. However, in general it is computationally expensive to evaluate the sum in equation (3.12). It is therefore common to assume that $\mathbf{x_{1}}_k$ is independent of $\mathbf{x_{2}}_{k-1}$, conditioned upon $\mathbf{x_{1}}_{k-1,i}$. In other words, in the graph of Figure 3.1, there is no arc from $\mathbf{x_{2}}_{k-1}$ to $\mathbf{x_{1}}_k$. In this case, equation (3.12) reduces to:

$$p(\mathbf{x_{1}}_k | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{x_{1}}_k | \mathbf{x_{1}}_{k-1,i}) p(\mathbf{x_{2}}_{k-1} | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_{k-1} \quad (3.13)$$

$$= p(\mathbf{x_{1}}_k | \mathbf{x_{1}}_{k-1,i}) \int p(\mathbf{x_{2}}_{k-1} | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_{k-1} \quad (3.14)$$

Finally:

$$p(\mathbf{x_{1}}_k | \mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1}) = p(\mathbf{x_{1}}_k | \mathbf{x_{1}}_{k-1,i}) \quad (3.15)$$

The restriction represented by (3.15) may be perfectly applicable given certain problem models, and may represent an approximation given other models which in fact intrinsically would require the lack of independence between $\mathbf{x_{1}}_k$ and $\mathbf{x_{2}}_{k-1}$, conditioned upon $\mathbf{x_{1}}_{k-1,i}$.

Another common simplification applied to RBPF models is to assume that $\mathbf{x_{2}}_k$ is independent of $\mathbf{x_{1}}_{k-1}$, conditioned upon $\mathbf{x_{2}}_{k-1,i}$. In other words, in the graph of figure 3.1, there is

no arc from $\mathbf{x_{1}}_{k-1}$ to $\mathbf{x_{2}}_k$. In this case, the algorithm is in fact the same, except in the exact step, where for every i, the update of $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ only necessitates $p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1})$, $\mathbf{x_{1}}_{k,i}$, and $\mathbf{z}_k$, but not $\mathbf{x_{1}}_{k-1,i}$.

Both these assumptions simplify the other important part of the update of the weights, ie, the evaluation of $p(\mathbf{z}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1})$. To compute it, notice that:

$$p(\mathbf{z}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k, \mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_k \tag{3.16}$$

Thus:

$$p(\mathbf{z}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x_{2}}_k, \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_k \tag{3.17}$$

But then we have:

$$p(\mathbf{z}_k|\mathbf{x_{2}}_k, \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x_{2}}_k, \mathbf{x_{1}}_{k,i}) \tag{3.18}$$

And therefore:

$$p(\mathbf{z}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x_{2}}_k, \mathbf{x_{1}}_{k,i}) p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_k \tag{3.19}$$

We still need to explicit the term $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1})$. We have:

$$\begin{aligned} p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) &= \int p(\mathbf{x_{2}}_k, \mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_{k-1} \\ &= \int p(\mathbf{x_{2}}_k|\mathbf{x_{2}}_{k-1}, \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_{k-1} \\ &= \int p(\mathbf{x_{2}}_k|\mathbf{x_{2}}_{k-1}, \mathbf{x_{1}}_{k,i}) p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1}) d\mathbf{x_{2}}_{k-1} \end{aligned} \tag{3.20}$$

At this point, the model should provide us with $p(\mathbf{x_{2}}_k|\mathbf{x_{2}}_{k-1}, \mathbf{x_{1}}_{k,i})$, and we are updating online the distribution $p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_{k-1})$, so we have here everything we need for the derivation of the algorithm.

3.3 When and how to use RBPFs

3.3.1 A RBPF for conditionally linear-Gaussian problems

There exist a family of state-space models for which it is particularly advantageous to use the RBPF approach: when the distribution $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k, \mathbf{Z}_k)$ is found to be Gaussian, and when the evolution of $\mathbf{x_{2}}_k$, conditioned upon $\mathbf{X_{1}}_k$ and $\mathbf{Z}_k$, is described by linear equations. This type of situation is called **conditionally linear-Gaussian**. When this occurs, then for each

i, $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ can be parametrized by a mean vector and a covariance matrix, and the sequential update of $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ can be incorporated easily to the particle filter recursion under the form of a Kalman filter. Because of this, conditionally linear-Gaussian systems cover almost all of the practical range of applications of RBPFs. In fact, many authors in the literature only describe the RBPF in the case where $\mathbf{x_{2}}_k$ is a conditionally linear Gaussian model, even though theoretically, RBPFs are not restricted to this case. In this section, we derive an RBPF that is tailored to conditionally linear-Gaussian situations. A subcase of the following work was summarized in a paper which appeared in [42]. In [42], the algorithm is presented only for the case where the importance density is the transition prior of the substate $\mathbf{x_{1}}_k$.

In practical terms, as a basis for the algorithm derivation, suppose that we are able to model the evolution of some state $\mathbf{x_{2}}_k$ of a system using the following time-varying equations:

$$\mathbf{x_{2}}_k = \mathbf{A}_k\mathbf{x_{2}}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{G}_k\mathbf{w}_k \tag{3.21}$$

$$\mathbf{z}_k = \mathbf{C}_k\mathbf{x_{2}}_k + \mathbf{D}_k\mathbf{u}_k + \mathbf{H}_k\mathbf{v}_k \tag{3.22}$$

but with one or more of the quantities $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, $\mathbf{D}_k$, $\mathbf{G}_k$, $\mathbf{H}_k$ or $\mathbf{u}_k$ being unknown and possibly evolving under non-linear, non-Gaussian conditions. In (3.21) and (3.22), $\mathbf{w}$ and $\mathbf{v}$ are zero mean unit covariance Gaussian random vectors, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{G}$, and $\mathbf{H}$ have the dimension of matrices, and $\mathbf{u}$ that of a vector. This situation is a typical conditionally linear-Gaussian case since, conditioned upon the values of $\mathbf{x_{1}}_k$, (3.21) and (3.22) are linear and Gaussian.

A possibility to solve the problem using particle filtering (RBPF or not) is to define a complete state vector, that is, to first form another set of variables describing the time-varying parameters, say $\mathbf{x_{1}}_k$, and then to apply the algorithm on the whole state $\mathbf{x}_k = \{\mathbf{x_{1}}_k; \mathbf{x_{2}}_k\}$.

Note here, as an important aside, that particle filters have been found to be not well suited for *fixed*-parameter estimation [1]. However if this is necessary, in [1] it is stated that a possible alternative is to introduce in the model some artificial variation of the parameters over time. Such an artificial drift can take the form of a *Gaussian random walk*, which is equivalent to a Markov process with a Gaussian transitional probability. In a Gaussian random walk, the current value observed is drawn normally around the previous value. The resulting quantity drifts randomly in time, but it does so with a certain form of "continuity". This necessitates a hyperparameter, fixed and predetermined: the variance of the Gaussian random walk. Of course, Gaussian random walks can also be used to model parameters that are actually time-varying, and we will in fact be using them very often throughout this work. Note also that there exists a more advanced scheme, presented in [8], which defines a different type of evolution specifically for the evaluation of fixed parameters (the method is presented in the context of a combined

fixed-parameter and time-varying parameter estimation).

In the following, for generality we consider that the set of time-varying parameters includes all of $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, $\mathbf{D}_k$, $\mathbf{G}_k$, $\mathbf{H}_k$ and $\mathbf{u}_k$. The relationship between $\mathbf{x}_{\mathbf{1}k}$ and these parameters must be one-to-one: for example, for given i, the matrix $\mathbf{A}_{k,i}$ can be obtained uniquely from $\mathbf{x}_{\mathbf{1}k,i}$.

A simple yet generic and directly applicable algorithm can be derived in the case where $\mathbf{x}_{\mathbf{1}k}$ is independent of $\mathbf{x}_{\mathbf{2}k-1}$, conditioned upon $\mathbf{x}_{\mathbf{1}k-1,i}$, and where in addition $\mathbf{x}_{\mathbf{2}k}$ is independent of $\mathbf{x}_{\mathbf{1}k-1}$, conditioned upon $\mathbf{x}_{\mathbf{2}k-1,i}$ (these are the simplifications that were discussed in 3.2.2). In this case, recall from 3.2.2 that we can use equations (3.15), (3.16), (3.19) and (3.20) in the weight update.

The weight update equation, in this case, takes the following form, using equation (3.15):

$$\begin{aligned}\widetilde{w}_{k,i} &= w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1})p(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{X}_{\mathbf{1}k-1,i},\mathbf{Z}_{k-1})}{q(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{X}_{\mathbf{1}k-1,i},\mathbf{Z}_k)} \\ &= w_{k-1,i}\frac{p(\mathbf{z}_k|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1})p(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{x}_{\mathbf{1}k-1,i})}{q(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{X}_{\mathbf{1}k-1,i},\mathbf{Z}_k)}\end{aligned}$$

Firstly, this shows that we must define the density $p(\mathbf{x}_{\mathbf{1}k,i}|\mathbf{x}_{\mathbf{1}k-1})$ to apply the algorithm. This must be done with the physical constraints of the problem in mind: this distribution is equivalent to the description of the *evolution in time* of the variable $\mathbf{x}_{\mathbf{1}k}$. Moreover, we must be able to evaluate this distribution easily. A good starting point, when the a priori knowledge of the system is limited, is to employ a Gaussian random walk on the elements of $\mathbf{x}_{\mathbf{1}k}$ to model their evolution.

Secondly, we see that we must be able to compute $p(\mathbf{z}_k|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1})$. For this, we will refer to the explanations provided in 3.2.2, and particulary to equations (3.19) and (3.20). For clarity, let us repeat here equations (3.19), and (3.20):

$$p(\mathbf{z}_k|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_{\mathbf{2}k},\mathbf{x}_{\mathbf{1}k,i})p(\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1})d\mathbf{x}_{\mathbf{2}k} \tag{3.23}$$

$$p(\mathbf{x}_{\mathbf{2}k}|\mathbf{X}_{\mathbf{1}k,i},\mathbf{Z}_{k-1}) = \int p(\mathbf{x}_{\mathbf{2}k}|\mathbf{x}_{\mathbf{2}k-1},\mathbf{x}_{\mathbf{1}k,i})p(\mathbf{x}_{\mathbf{2}k-1}|\mathbf{X}_{\mathbf{1}k-1,i},\mathbf{Z}_{k-1})d\mathbf{x}_{\mathbf{2}k-1} \tag{3.24}$$

We begin by noting from (3.22) that:

$$p(\mathbf{z}_k|\mathbf{x}_{\mathbf{2}k},\mathbf{x}_{\mathbf{1}k,i}) = \mathcal{N}(\mathbf{z}_k|\mathbf{C}_{k,i}\mathbf{x}_{\mathbf{2}k} + \mathbf{D}_{k,i}\mathbf{u}_{k,i};\mathbf{H}_{k,i}\mathbf{H}_{k,i}{}^T) \tag{3.25}$$

Moreover, from equation (3.21), we have:

$$p(\mathbf{x}_{2k}|\mathbf{x}_{2k-1}, \mathbf{x}_{1k,i}) = \mathcal{N}(\mathbf{x}_{2k}|\mathbf{A}_{k,i}\mathbf{x}_{2k-1} + \mathbf{B}_{k,i}\mathbf{u}_{k,i}; \mathbf{G}_{k,i}\mathbf{G}_{k,i}{}^T) \tag{3.26}$$

According to the algorithm, at instant k we are given the distribution $p(\mathbf{x}_{2k-1}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_{k-1})$ – it is precisely the one that we are updating online. From the Gaussianness of the system, this distribution is Gaussian. Let us define it as:

$$p(\mathbf{x}_{2k-1}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_{k-1}) \triangleq \mathcal{N}(\mathbf{x}_{2k-1}|\mathbf{x}_{2k-1,i}; \mathbf{K}_{k-1,i}) \tag{3.27}$$

To proceed with the derivation, we require the following technical result:

Lemma: Combination of Gaussian density functions. *If $\mathcal{N}(\mathbf{x}|\mathbf{y}; \mathbf{Q})$ represents the density of a Gaussian random vector with mean $\mathbf{y}$ and covariance matrix $\mathbf{Q}$, then:*

$$\int \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y}; \mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z}; \mathbf{K})\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{z}; \mathbf{Q} + \mathbf{F}\mathbf{K}\mathbf{F}^T) \tag{3.28}$$

The proof is given in appendix A.

From this result, which we apply to equation (3.24) using (3.26) and (3.27), we obtain:

$$p(\mathbf{x}_{2k}|\mathbf{X}_{1k,i}, \mathbf{Z}_{k-1}) = \mathcal{N}(\mathbf{x}_{2k}|\mathbf{x}_{2k|k-1,i}; \mathbf{K}_{k|k-1,i}) \tag{3.29}$$

where:

$$\begin{aligned} \mathbf{x}_{2k|k-1,i} &\triangleq \mathbf{A}_{k,i}\mathbf{x}_{2k-1,i} + \mathbf{B}_{k,i}\mathbf{u}_{k,i} \\ \mathbf{K}_{k|k-1,i} &\triangleq \mathbf{G}_{k,i}\mathbf{G}_{k,i}{}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1,i}\mathbf{A}_{k,i}{}^T \end{aligned}$$

We can now apply once again the result in (3.28), this time to equation (3.23) using (3.25) and (3.29) to obtain the final expression for the distribution in the weight update equation:

$$p(\mathbf{z}_k|\mathbf{X}_{1k,i}, \mathbf{Z}_{k-1}) = \mathcal{N}(\mathbf{z}_k|\mathbf{y}_{k,i}; \mathbf{T}_{k,i}) \tag{3.30}$$

where:

$$\begin{aligned} \mathbf{y}_{k,i} &\triangleq \mathbf{C}_{k,i}\mathbf{x}_{2k|k-1,i} + \mathbf{D}_{k,i}\mathbf{u}_{k,i} \\ \mathbf{T}_{k,i} &\triangleq \mathbf{H}_{k,i}\mathbf{H}_{k,i}{}^T + \mathbf{C}_{k,i}\mathbf{K}_{k|k-1,i}\mathbf{C}_{k,i}{}^T \end{aligned}$$

Observe that as a byproduct, we have carried out part of the exact step, the goal of which being the update of the Gaussian distribution $p(\mathbf{x}_{2k-1}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_{k-1})$ to $p(\mathbf{x}_{2k}|\mathbf{X}_{1k,i}, \mathbf{Z}_k)$. This can be done via a Kalman filter, but some of the previous equations intervening in the computations of the weight are precisely part of the classical KF equations. In RBPFs for conditionally

linear-Gaussian systems, the KF equations and the weight computations are thus intertwined. To complete the derivation of the algorithm, we must now obtain $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ to complete the derivation of the algorithm. From [28], we directly have:

$$p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k) = \mathcal{N}(\mathbf{x_{2}}_k|\mathbf{x_{2}}_{k,i}; \mathbf{K}_{k,i}) \tag{3.31}$$

where, using $\mathbf{J}_{k,i} = \mathbf{K}_{k|k-1,i}\mathbf{C}_{k,i}{}^T\mathbf{T}_{k,i}^{-1}$, we have:

$$\begin{aligned} \mathbf{x_{2}}_{k,i} &= \mathbf{x_{2}}_{k|k-1,i} + \mathbf{J}_{k,i}(\mathbf{z}_k - \mathbf{y}_{k,i}) \\ \mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C}_{k,i})\mathbf{K}_{k|k-1,i} \end{aligned}$$

We are now ready to write a complete algorithm, tailored to conditionally linear-Gaussian situations. The result is shown in Algorithm 5. In this algorithm, we suppose that resampling is done at every step. Note also that, as we had previously discussed, the algorithm simplifies at the weight update equation if the importance density is chosen to be equal to $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1})$. Algorithms of this type have been successfully applied to different problems. For example, in [19], an application to integrated navigation systems demonstrates the clear benefits of such RBPFs over PFs (the article also contains simulation results in which PFs outperform state-of-the-art IMM methods). In this thesis, Algorithm 5 is central since it is the main building block of several algorithms presented in Chapter 5. We will frequently refer to it thereafter.

3.3.2 Example of RBPF applicable to signal processing

Problem formulation and application of the algorithm

We now take an example that can be applied in a signal processing problem, in which a signal x_k is passed through an unknown, time-varying FIR filter. We are given a set of noisy measurements at the output of the filter. We do not know in advance the values of the signal, however its evolution in time is described by a known function (which is not restricted to be linear). Similarly, the coefficients are unknown, however we suppose that they evolve in a linear fashion (this linear behavior ensures that Rao-Blackwellisation is possible).

Let $\mathbf{x_{2}}_k \in \mathbb{R}^p$ represent the p coefficients of the FIR filter. We then define $\mathbf{x_{1}}_k \in \mathbb{R}^p$ as the last p values of the signal to be estimated, from time $k-p+1$ to time k. This way, $\mathbf{x_{1}}_k(1) \equiv x_k$ and $\mathbf{x_{1}}_k(l) \equiv x_{k-l+1}$ (for $l \leq p$). In addition, let $g(.)$ be a function from $\mathbb{R}^p$ to $\mathbb{R}^p$ defined as follows:

$$g(\mathbf{x_{1}}_k) = [f(\mathbf{x_{1}}_k) \;\; \mathbf{x_{1}}_k(1) \;\; \mathbf{x_{1}}_k(2) \;\; \ldots \;\; \mathbf{x_{1}}_k(p-1)]^T \tag{3.32}$$

In equation (3.32), $f(.)$ represents a function from $\mathbb{R}^p$ to $\mathbb{R}$, which is used to describe the evolution of the signal $x_k = \mathbf{x_{1}}_k(1)$. As a sidenote, note that we have arbitrarily chosen to restrict the form of function $g(.)$ so that it reflects a more common practical situation. In theory,

any function can be handled by the particle filter. Finally, we let $\mathbf{A}_k$ be a transition matrix for the FIR coefficients, and define the following noises: $\mathbf{w_{1}}_k$, $\mathbf{w_{2}}_k$ and v_k are all $\sim \mathcal{N}(0, \mathbf{I})$, and we choose to constrain the multiplicative matrix $\mathbf{S} = \text{diag}\{\sigma_1, \mathbf{0}_{p-1\times p-1}\}$, such that only the first component of $\mathbf{x_{1}}_k$ is stochastic. The problem setting can then be summarized by the following equations:

$$\mathbf{x_{1}}_k = g(\mathbf{x_{1}}_{k-1}) + \mathbf{S}\mathbf{w_{1}}_k \tag{3.33}$$

$$\mathbf{x_{2}}_k = \mathbf{A}_k\mathbf{x_{2}}_{k-1} + \mathbf{G}\mathbf{w_{2}}_k \tag{3.34}$$

$$z_k = {\mathbf{x_{1}}_k}^T\mathbf{x_{2}}_k + \sigma_v v_k \tag{3.35}$$

It is readily seen that conditioned upon the values of the substate $\mathbf{x_{1}}_k$, the system formed by equations (3.34) and (3.35) is linear-Gaussian. We can therefore solve this problem using RBPFs, by directly applying algorithm 5, in which we use $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1})$ as the importance density, and where we set $\forall\{k,i\}, \mathbf{A}_{k,i} = \mathbf{A}_k$, $\mathbf{B}_{k,i} = \mathbf{0}$, $\mathbf{C}_{k,i} = {\mathbf{x_{1}}_{k,i}}^T$, $\mathbf{D}_{k,i} = \mathbf{0}$, $\mathbf{u}_{k,i} = \mathbf{0}$, and $\mathbf{H}_{k,i} = \sigma_v$. In addition, we have $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1}) = \mathcal{N}(\mathbf{x_{1}}_k|g(\mathbf{x_{1}}_{k-1}); \mathbf{S}\mathbf{S}^T)$.

At the end of every step, we must store in memory the set $\{\mathbf{x_{1}}_{k,i}, \mathbf{x_{2}}_{k,i}, \mathbf{K}_{k,i}\}_{i=1}^N$ to re-use for the next step. The result is presented in Algorithm 6.

Simulation results

We present results here for a very simple case, in which a signal is passed through a time-varying gain. We thus have $p = 1$ and $g(.) \equiv f(.)$. We define:

$$f(\mathbf{x_{1}}_{k-1}) = \cos(\mathbf{x_{1}}_{k-1}) + \sin(\mathbf{x_{1}}_{k-1}) \tag{3.36}$$

with $\mathbf{x_{1}}_k \in \mathbb{R}$, and we choose $\mathbf{A}_k = \mathbf{I}$. In this simple case, $\mathbf{w_{1}}_k$ and $\mathbf{w_{2}}_k$ are scalars (as opposed to vectors), and we let $\mathbf{S}\mathbf{S}^T = 0.09$ and $\mathbf{G}\mathbf{G}^T = 0.04$. We also define $\sigma_v^2 = 10^{-3}$.

This sub-case is summarized by the following equations:

$$\mathbf{x_{1}}_k = \cos(\mathbf{x_{1}}_{k-1}) + \sin(\mathbf{x_{1}}_{k-1}) + \mathbf{S}\mathbf{w_{1}}_k \tag{3.37}$$

$$\mathbf{x_{2}}_k = \mathbf{x_{2}}_{k-1} + \mathbf{G}\mathbf{w_{2}}_k \tag{3.38}$$

$$z_k = {\mathbf{x_{1}}_k}^T\mathbf{x_{2}}_k + \sigma_v v_k \tag{3.39}$$

$$\tag{3.40}$$

For the simulation, we choose $N = 500$ particles, which are initialized at random as well. The results obtained for the estimates of the substate $\mathbf{x_{1}}_k$ are shown in Figure 3.2. For the

estimates of $\mathbf{x_{2}}_k$, refer to Figure 3.2. On both figures, we observe that the filter is able to track quite well the evolution of each substate, with a reasonable amount of particles. For more precision, more particles can be used, but the implementation then becomes more computationally expensive. For $p > 1$, we find that the algorithm requires a good initialization and an increasingly large amount of particles to perform correctly. To improve the performance of the algorithm, it may become necessary to try a different importance distribution taking into account the current measurement.

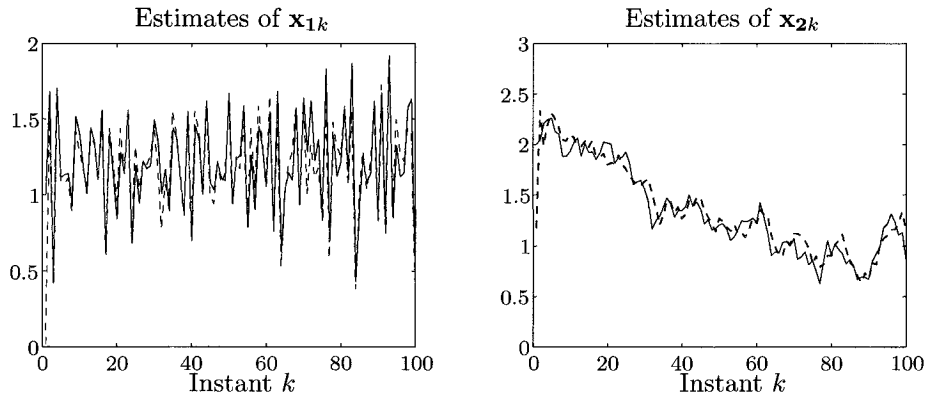


Figure 3.2: Simulation results for the first example
The blue lines are the true values and the red, dotted lines are the estimates.

3.3.3 Example of RBPF for tracking

Problem formulation and application of the algorithm

Consider the problem of tracking a maneuvering target, whose position and velocity at instant k is given by a continuous random vector $\mathbf{x_{2}}_k$, and where the regime[1] of the target is represented by the discrete random variable $\mathbf{x_{1}}_k$. The state to be estimated is $\mathbf{x}_k = \{\mathbf{x_{1}}_k \ ; \ \mathbf{x_{2}}_k\}$. The model is as follows:

$$\mathbf{x_{2}}_k = \mathbf{A}\mathbf{x_{2}}_{k-1} + \mathbf{B}\mathbf{x_{1}}_k + \mathbf{G}\mathbf{w}_k \tag{3.41}$$

$$\mathbf{z}_k = \mathbf{C}\mathbf{x_{2}}_k + \mathbf{H}\mathbf{v}_k \tag{3.42}$$

Additionally, $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1})$ is given, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{G}$ and $\mathbf{H}$ are given matrices. $\mathbf{w}$ and $\mathbf{v}$ are zero-mean, unit variance Gaussian noises. Again, we note that if a certain sequence of states $\mathbf{X_{1}}_{k,i}$ is given, then we can find the corresponding values of $\mathbf{x_{2}}_{k,i}$ with the Kalman filter, since

[1]The regime of a maneuvering target is defined as the type of maneuver being executed. For example, a change of regime can mean a sudden change of speed, direction, acceleration, etc.

then $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ is Gaussian.

If we choose again to use the importance density $q(\mathbf{x_{1}}_k|\mathbf{X_{1}}_{k-1,i}, \mathbf{Z}_k) = p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1,i})$, then we can directly apply the method and results from the previous example, and we obtain algorithm 7.

Simulation results

In our model, we use $\mathbf{x_{2}}_k = \begin{pmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{pmatrix}^T$ where (x_k, y_k) is the position of the target in a cartesian plane at instant k. We take:

$$A = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

and

$$\mathbf{B} = \begin{pmatrix} 1.25 & -1.25 & 0.35 & -0.35 \end{pmatrix}^T$$

We only observe the noisy position x and y of the object to be tracked. Also, we suppose that $\mathbf{x_{1}}_k$ can only take values $\{-1, 0, 1\}$, and we define the transition probabilities as $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1}) = 0.8$ if $\mathbf{x_{1}}_k = \mathbf{x_{1}}_{k-1}$, and $p(\mathbf{x_{1}}_k|\mathbf{x_{1}}_{k-1}) = 0.1$ if $\mathbf{x_{1}}_k \neq \mathbf{x_{1}}_{k-1}$.

Initialization is done randomly as follows: for the regime, each particle is assigned a value of -1, 0, or 1 with probability 1/3. For the position and velocity, we set the true initial values as $\begin{pmatrix} x_0 & y_0 & \dot{x}_0 & \dot{y}_0 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 1 & 4 \end{pmatrix}^T$, and we draw each initial particle as $\mathbf{x_{2}}_{0,i} = \alpha_i + \beta_i + \mathbf{x}_0$, where $\alpha \sim \mathcal{N}(\mathbf{0}; 50\mathbf{I})$, all the components of β are drawn from a uniform distribution $\mathcal{U}(0, 50)$, and where $\mathbf{x}_0 = \begin{pmatrix} 20 & 30 & 1 & 4 \end{pmatrix}^T$. With $T = 0.3$, $N = 100$, $\{\sigma_{\mathbf{w}}, \sigma_{\mathbf{v}}\} = \{0.3, 4\}$, where $\mathbf{G}\mathbf{G}^T = \sigma_{\mathbf{w}}^2\mathbf{I}$ and $\mathbf{H}\mathbf{H}^T = \sigma_{\mathbf{v}}^2\mathbf{I}$, we obtain the results presented in figures (3.3), (3.4), and (3.5).

As seen on the figures, the RBPF strategy efficiently estimates the true state values, including the regime. We begin with Figure 3.3, on which we can observe that the overall estimated trajectory follows the true trajectory. On Figure 3.5, we see that the velocity estimates, which we recall are a part of the state that is not directly observe, quickly converge, even though the initial guesses are very rough. On Figure 3.4, we find that the estimated regime, after an incorrect first guess, becomes correct most of the time, with the majority of errors occuring at the transitions (the filter cannot track changes instantaneously).

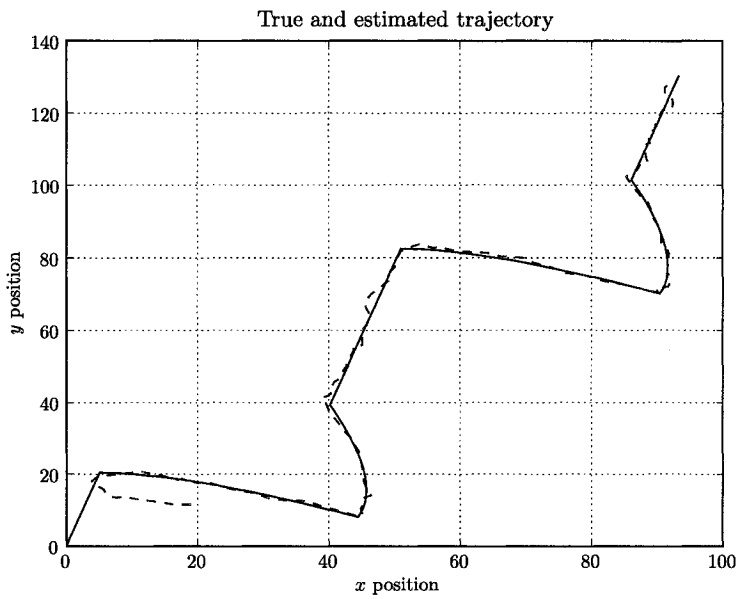


Figure 3.3: Simulation results for the second RBPF example: trajectory estimates
The estimates are the red dotted lines, and the true values are the solid, blue lines. The simulation begins in the lower left part of the graph.

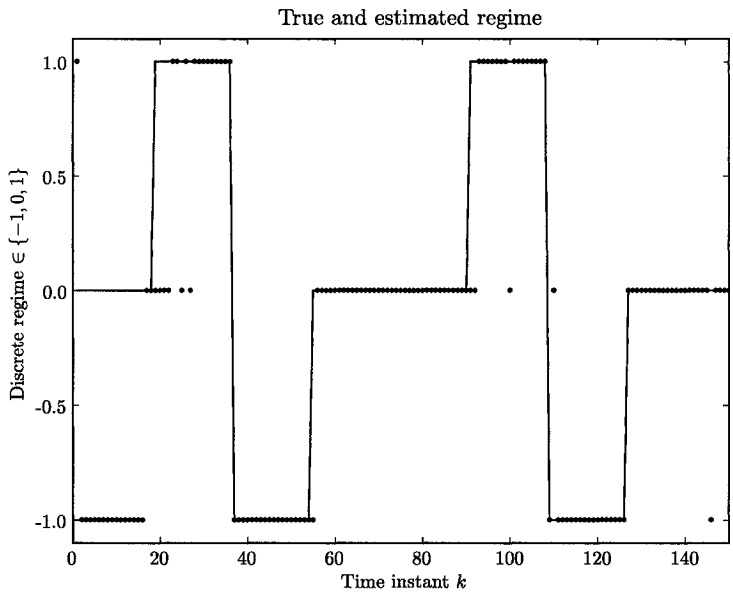


Figure 3.4: Simulation results for the second RBPF example: regime estimates
The estimates are represented by the red dots, and the true values are the straight, solid blue lines. In this example, there are only three possible regime values.

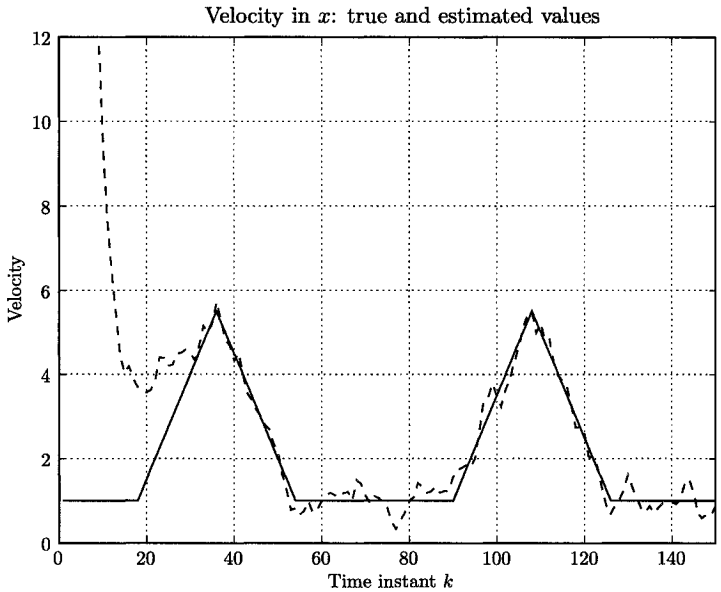


Figure 3.5: Simulation results for the second RBPF example: velocity in x estimates
At first, the estimates (red dotted lines) are completely off, but the algorithm is able to quickly gear the estimates to the true values (solid, blue lines).

3.4 The modified Rao-Blackwellised Particle Filter ($\star$)

The exact computation of one part of the state makes Rao-Blackwellised Particle Filters more robust in terms of convergence properties, and allows the use of less particles to achieve a desired performance. However, the computational load may still be too high for certain applications, especially if real-time operation is a requirement. In this section, we present an altered version of the generic RBPF algorithm (described in Algorithm 4 in subsection 3.2.1) with the motivation of making it computationally less intensive. In the case where a Kalman filter can be used within the RBPF (i.e., in conditionally linear-Gaussian situations), we will see that using a modified RBPF requires a single KF iteration per input sample instead of N. In this section we also show experiments results, and find that while good convergence can still be obtained, computational efficiency is always increased, making this algorithm an option to consider for real-time implementations. This work was presented in [41].

3.4.1 Presentation of the algorithm ($\star$)

As previously stated, we are interested in making the RBPF algorithm lighter. In the generic RBPF algorithm (presented in Algorithm 4), we have seen that once a single particle of $\mathbf{x_1}$ is drawn, we must update the entire corresponding conditional distribution of $\mathbf{x_2}$. At each

step, this must be done for every particle. In conditionally linear-Gaussian situations, we have also seen that the update of each conditional distribution can be carried out more efficiently using Kalman filters. As a seemingly radical simplification, we would like to think of a way of updating, at every k, a single distribution for a "global" $\mathbf{x_{2}}_k$, chosen to be as meaningful as possible. Within one iteration of the algorithm, given the updated set of particles $\{\mathbf{x_{1}}_{k,i}\}_{i=1}^N$, we propose to update only the probability distribution of $\mathbf{x_{2}}_k$ given a particular estimate of $\mathbf{x_{1}}_k$, denoted by $\hat{\mathbf{x}}_{\mathbf{1}k}$, obtained from $\{\mathbf{x_{1}}_{k,i}\}_{i=1}^N$. In many cases, $\{\mathbf{x_{1}}_{k,i}\}_{i=1}^N$ is unimodal, and therefore a reasonable candidate for the particular estimate used to update the single distribution of $\mathbf{x_{2}}_k$ could simply be the mean of $\{\mathbf{x_{1}}_{k,i}\}_{i=1}^N$. This is what is done in the modified RBPF.

In the modified algorithm, the main difference from the RBPF approach lies in the way that the particles $\{\mathbf{x_{2}}_{k,i}\}_{i=1}^N$ are propagated to the next iteration of the algorithm. Suppose that we have a conditionally linear-Gaussian model. Let us first describe how the algorithm operates, and how the two "subfilters" – Particle and Kalman – interact. At step k, the PF will compute its own estimate, $\hat{\mathbf{x}}_{\mathbf{1}k}$. This estimate is passed to a single KF to update the mean $\hat{\mathbf{x}}_{\mathbf{2}k}$ and error covariance matrix $\mathbf{K}_k$ of an estimate for $\mathbf{x_{2}}_k$. In turn, $\{\hat{\mathbf{x}}_{\mathbf{2}k}$ and $\mathbf{K}_k\}$ are passed to the PF at the beginning of the next step, which still carries the particles $\mathbf{x_{1}}_{k,i}$. A new procedure then takes place: as new particles $\{\mathbf{x_{1}}_{k+1,i}\}_{i=1}^N$ are drawn at the beginning of the simulation part, they are completed by a set of N samples $\mathbf{x_{2}}_{k,i} \sim \mathcal{N}(\mathbf{x_{2}}_k|\hat{\mathbf{x}}_{\mathbf{2}k}, \mathbf{K}_k)$, since this Gaussian distribution is the one that reflects best the precision of our knowledge of $\hat{\mathbf{x}}_{\mathbf{2}k}$. With this completed set of particles, the PF can now carry on, and the algorithm continues. We formally present the modified RBPF in Algorithm 8.

Observe here that although the explanation above is given in the case where Kalman filters can be used, theoretically the modified RBPF is not limited to this case. In the general case, in the "exact" step of Algorithm 8, rather than obtaining $\mathcal{N}(\mathbf{x_{2}}_k|\hat{\mathbf{x}}_{\mathbf{2}k}, \mathbf{K}_k) = p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k = \hat{\mathbf{X}}_{\mathbf{1}k}, \mathbf{Z}_k)$ using a Kalman Filter and $\hat{\mathbf{x}}_{\mathbf{1}k}$, $\hat{\mathbf{x}}_{\mathbf{2}k-1}$, $\mathbf{K}_{k-1}$ and $\mathbf{z}_k$, we could only request the update of $p(\mathbf{x_{2}}_{k-1}|\mathbf{X_{1}}_{k-1} = \hat{\mathbf{X}}_{\mathbf{1}k-1}, \mathbf{Z}_{k-1})$ from $\hat{\mathbf{x}}_{\mathbf{1}k}$, and then obtain $\mathbf{K}_k$ from it. Nevertheless we have chosen to understate in Algorithm 8 the possibility of using Kalman filters.

Note that in the KF part, we are updating the distribution $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k = \hat{\mathbf{X}}_{\mathbf{1}k}, \mathbf{Z}_k)$, and we are taking its mean as our estimate for $\mathbf{x_{2}}_k$. In contrast, in the regular RBPF, we must update $p(\mathbf{x_{2}}_k|\mathbf{X_{1}}_k = \mathbf{X_{1}}_{k,i}, \mathbf{Z}_k)$ for all i (all particles) to form $\{\mathbf{x_{2}}_{k,i}\}_{i=1}^N$, and only then are we able to form an estimate for $\mathbf{x_{2}}_k$, using the weights computed by the PF. In the simulation part, the weight computation is identical to that carried out during a regular RBPF, but in particular, we have $\forall i, \mathbf{K}_{k,i} = \mathbf{K}_k$. Finally, we consider that resampling is done at each step, so that the available particles of $\mathbf{x_{1}}_{k,i}$ are as relevant as possible (only a few will have negligible weight). It

would not serve any purpose to resample the $\mathbf{x_{2}}_{k,i}$ particles, however the $\mathbf{x_{1}}_{k,i}$ are still subject to degeneracy.

3.4.2 Practical derivation from a regular RBPF algorithm ($\star$)

In this section we give some details on how to derive an efficient modified RBPF from a regular RBPF. Assume again a conditionally linear-Gaussian model, so that we can use the RBPF algorithm shown in Algorithm 5. According to the modified RBPF algorithm (presented in Algorithm 8), in the main loop of the PF step, the modified algorithm begins with the draw of a "regenerated" sample of $\mathbf{x_{2}}_{k-1,i}$. Practically, how is the rest of the loop in the regular RBPF affected? The key is to spot in the initial RBPF algorithm all the computations that can be taken out of this main loop. There are two steps in this procedure:

- Starting from the RBPF algorithm shown in Algorithm 5, we can (and must) always remove the lines that are directly involved with the update of the mean and covariance matrix of the substate $\mathbf{x_{2}}_k$, since the KF update is done after the loop in the modified algorithm.
- Secondly, it is important to examine the rest of the loop so as to remove any computations which do not depend on i anymore, due to the fact that $\forall i, \mathbf{K}_{k,i} = \mathbf{K}_k$: these computations can be done before entering the loop. The amount of computations that can be removed in this second step is case-dependent. As an illustration to this claim, among the two examples described in 3.4.4 below, we find that in the second example, the main loop is more significantly lightened than in the first example.

For the RBPF for conditionally linear-Gaussian systems described in 3.3.1 (Algorithm 5), the first step above consists in removing the following three lines from the end of the loop:

$$\begin{aligned}
\mathbf{J}_{k,i} &= \mathbf{K}_{k|k-1,i}{\mathbf{C}_{k,i}}^T\mathbf{T}_{k,i}^{-1} \\
\mathbf{x_{2}}_{k,i} &= \mathbf{x_{2}}_{k|k-1,i} + \mathbf{J}_{k,i}(\mathbf{z}_k - \mathbf{y}_{k,i}) \\
\mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C}_{k,i})\mathbf{K}_{k|k-1,i}
\end{aligned}$$

These computations must be conducted after the loop, along with the necessary steps for a complete KF update.

The second step is then a matter of examining the rest of the loop, that is:

$$
\begin{aligned}
\mathbf{K}_{k|k-1,i} &= \mathbf{G}_{k,i}\mathbf{G}_{k,i}{}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1,i}\mathbf{A}_{k,i}{}^T \\
\mathbf{T}_{k,i} &= \mathbf{H}_{k,i}\mathbf{H}_{k,i}{}^T + \mathbf{C}_{k,i}\mathbf{K}_{k|k-1,i}\mathbf{C}_{k,i}{}^T \\
\mathbf{x}_{\mathbf{2}k|k-1,i} &= \mathbf{A}_{k,i}\mathbf{x}_{\mathbf{2}k-1,i} + \mathbf{B}_{k,i}\mathbf{u}_{k,i} \\
\mathbf{y}_{k,i} &= \mathbf{C}_{k,i}\mathbf{x}_{\mathbf{2}k|k-1,i} + \mathbf{D}_{k,i}\mathbf{u}_{k,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(\mathbf{z}_k|\mathbf{y}_{k,i}, \mathbf{T}_{k,i})
\end{aligned}
$$

As it can be seen, setting $\mathbf{K}_{k-1,i} = \mathbf{K}_{k-1}$ does not in itself mean that the loop will be lighter, and what can be done depends on the problem setting. For example, if in the model $\mathbf{A}_k$ and $\mathbf{G}_k$ are known in advance, then the first line above can be removed, i.e., $\mathbf{K}_{k|k-1,i} = \mathbf{K}_{k|k-1}$ and this matrix can be computed outside of the loop. Similarly, if in addition $\mathbf{C}_k$ and $\mathbf{H}_k$, then the matrix $\mathbf{T}_{k,i} = \mathbf{T}_k$ can also be computed before entering the loop, as well as its inverse which is necessary to compute the unnormalized weight, as seen above. Similar considerations can be made regarding $\mathbf{B}_{k,i}$, $\mathbf{D}_{k,i}$ and $\mathbf{u}_{k,i}$.

3.4.3 Advantages and drawbacks ($\star$)

As the main advantage, since only 1 KF is used instead of N, the efficiency is always increased (as observed by the execution time), especially if the number of particles used is large. The gain in efficiency is case dependent, but there are always at least some elements that can be removed from the main loop iterating on all i, and done before or after (see section 3.4.2 above). In general, even if only one single operation is removed from the loop, it is really removed N times, thus any possible simplification can have significant consequences for the execution time. By contrast, we are only adding a single KF step after the loop in the modified algorithm, and its effects on the overall execution time is likely to be negligible, especially if there are many particles. In the conditionally linear-Gaussian situation, as shown in 3.4.2 there are at least three equations that can be removed from the loop, each potentially containing a large number of operations. We see that the modified RBPF will always execute faster.

Another advantage that is not negligible in practical implementations is the fact that there is less memory usage. In the regular RBPF, one needs to store the N means and covariance matrices that are used in the KFs. If $\mathbf{x}_{\mathbf{2}k} \in \mathbb{R}^n$, then $N \times (n^2+1)$ memory locations must be used, instead of n^2 for the algorithm presented here (since only $\mathbf{K}_k$ must be stored). Note that this is even less then what a regular PF would require, since the PF would store the N values of $\mathbf{x}_{\mathbf{2}k}$.

There are also some other, accessory advantages. For example, compared to a basic PF implementation, we have access to the covariance matrix of $\mathbf{x}_{2k}$, and thus we can have more control over the initialization (like in the RBPF case). In fact, through experimentation we have found that for some situations, the robustness of RBPFs over PFs is preserved, in the

sense that there are less situations where the algorithm fails, returning all-zero weights at a given time.

On the other hand, we can expect that there will be a loss in performance. Specifically, for the modified algorithm, we are only passing a single estimate of $\mathbf{x_{1}}_k$ at step k to a single KF, and not the entire approximated distribution to a bank of KFs. If this estimate is an expectation, then the algorithm will not perform as well if the current measure $\{\mathbf{x_{1}}_{k,i}\,,\, w_{k,i}\}_{i=1}^{N}$ is not unimodal. Similarly, if we decide to only return the particle with maximum weight to the KF, then we are taking the risk of selecting an outlier. Moreover, in the regular RBPF, the analytical relationship that exists between every particle of $\mathbf{x_1}$ and $\mathbf{x_2}$ (and their history) is only approximated in the modified algorithm. Indeed, at the beginning of the simulation part, each $\mathbf{x_{2}}_{k-1,i}$ is drawn from a Gaussian distribution instead of being the "exact" match for $\mathbf{x_{1}}_{k-1,i}$ (i.e., the value that would be returned by a KF iterated on $\mathbf{x_{1}}_{k-1,i}$). For these reasons, we also expect that the difference of performance between the two algorithms will increase with larger measurement and process noises. Intuitively, since the single estimate of $\mathbf{x_{1}}_k$ is the only source of information that the KF can rely on, repeated inaccuracies could potentially gear the KF in wrong directions, damaging in turn the quality of estimates of $\mathbf{x_{2}}_k$.

3.4.4 Applications and performance ($\star$)

In some cases, it is more desirable to use an efficient, cheaper solution, provided it still meets the problem requirements. The alteration presented here to the regular RBPF constitutes a tradeoff between performance and efficiency. In this context, the modified RBPF is applicable to the same range of problems as the regular RBPF, provided the results obtained are satisfactory, according to the designer's criteria. In our opinion, it is probably more suited for small-scale projects, in which fewer resources (computational but also of memory) are available. In chapter 5, we will implement a modified RBPF in the context of speech enhancement.

We show in this section comparative results between the regular and modified RBPFs, using the same two examples that were presented in sections 3.3.2 and 3.3.3. We first rework algorithms 6 and 7 using the method described in 3.4.2, so as to match Algorithm 8. We obtain respectively Algorithms 9 and 10.

In algorithms 9 and 10, the key improvement in efficiency can be noticed by observing the "for" loop, running from 1 to N, that is, on every particle. In these modified algorithms, fewer lines are present in this loop, making the execution faster. It is clear that the computation load will be lightened, with an increasing difference if the sample size N is large. We now compare the performance obtained with the regular and the modified RBPF algorithm, in terms of ex-

ecution time, and in terms of the actual results obtained, given the same initial conditions.

Execution time comparisons

The execution time depends of course on many factors: the type of computer, the style of coding, the programming language, etc. The following experiments were conducted on a PC with an Intel® Xeon™ CPU at 3.2 GHz, with 4 GB of RAM, running the Linux command-line version of MATLAB®. The results are gathered in Table 3.1, where the only varying parameter is the number of particles, N.

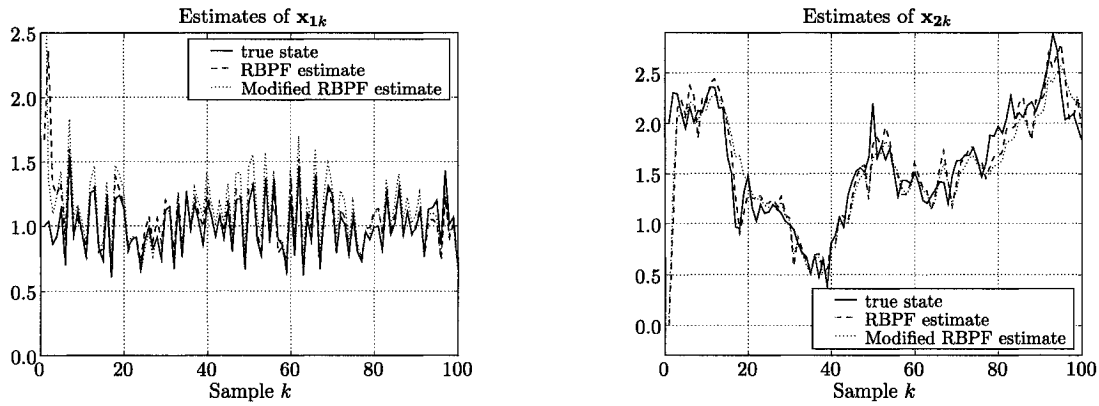
Execution time, seconds	*Number of particles*								
	500	1000	1500	2000	3000	5000	10000	50000	100000
Example 1: Regular RBPF	6.4	12.7	20.4	25.6	39.3	64.5	130.1	1213.4	4025.3
Example 1: Modified RBPF	4.1	9.3	12.5	17.9	25.9	43.5	89.6	942.6	3346.3
% Improvement	36	27	39	30	34	33	31	22	17
Example 2: Regular RBPF	11.4	22.9	34.5	46.8	70.8	120.8	265.3	4843.6	–
Example 2: Modified RBPF	4.8	9.9	14.6	19.3	30.8	52.1	110.6	3626.5	–
% Improvement	58	57	58	59	56	57	58	25	–

Table 3.1: Execution time comparisons: regular and modified RBPF

From Table 3.1, we observe that the improvement is always substantial, especially for the second example. This difference comes from the fact that in the second example, the substate $\mathbf{x}_{2k}$ has a dimension of 4, and thus the amount of computations saved is particularly significant. Also note that for the second example, the values corresponding to $N = 50,000$ are outliers: we assume that there is not enough RAM in our system to handle 50,000 particles of dimension 4, as well as to save the history of estimates (of dimension 5), and therefore at some point during the execution of the algorithm, the computer must begin to use Linux's swap partition, considerably slowing down the process. However this case does not necessarily represent the targeted application of the modified RBPF, which is in our minds aimed at implementations with lesser particles. The fact that an application requires in the first place a very large number of particles understates that convergence is an important issue – implementing the modified RBPF may simply not be an appropriate option.

Results comparison

We now show some convergence results, starting with the first example. For convenience reasons, a thorough comparison was conducted for the second example only; we will only show some



(a) Estimates of $\mathbf{x}_{1k}$ (b) Estimates of $\mathbf{x}_{2k}$

Figure 3.6: Estimates comparisons for regular and modified RBPF, first example
The regular RBPF estimates are the red dotted lines, and the modified RBPF estimates are green dotted lines.

selected simulation results for the first example, followed by qualitative comments. Overall, the second example requires less particles (less than 200) than the first one for a satisfactory convergence, and thus extensive testing/simulating is less time-consuming (extensive testing with less particles than necessary for basic convergence could appear pointless). The first batch of experiments were conducted using a given, static set of measurements, and we use the same sub-case as the one used for the simulations in section 3.3.2, represented by equations (3.37), (3.38), and (3.39). The noises used are $\sigma_1^2 = 0.2$, $\sigma_2^2 = 0.2$, and $\sigma_v^2 = 10^{-4}$. Figure 3.6 show the results of a simulation run with $N = 2000$ particles. These first results were chosen to illustrate the following points, which were observed as general occurrences:

- The modified and regular algorithm offer comparable performance for "small" variances noises
- The performance of the modified algorithm degrades faster when the noise variances increase, especially the observation variance σ_v^2
- In general, the modified algorithm has a more difficult time tracking abrupt changes: for example, observe Figure 3.6 (b): around samples 15-20, there is a sharp decrease and then an increase in the gain $\mathbf{x}_{2k}$, which the regular algorithm tracks, but not the modified one.

For the second example, to compare the performance, a range of values for the covariance of $\mathbf{w}$ and $\mathbf{v}$ was defined. For a given pair $\{\sigma_{\mathbf{w}}, \sigma_{\mathbf{v}}\}$, where $\mathbf{Q} = \sigma_{\mathbf{w}}^2\mathbf{I}$ and $\mathbf{R} = \sigma_{\mathbf{v}}^2\mathbf{I}$, we generated a sequence of true states and measurements shared by both algorithms. Initially the particles for the regular RBPF are drawn from a specific normal distribution (which remains the same

throughout all tests), from which the modified RBPF also draws its initial state estimates. To measure the performance, we use the following metric:

- For the sub-state $\mathbf{x_{1}}_k$, we add the absolute value of the differences between the true and estimated values, and divide the result by the number of observations.
- For the sub-state $\mathbf{x_{2}}_k$, we compute $\sum_k \left|\mathbf{x_{2}}_{k\ (\text{true})} - \hat{\mathbf{x}}_{2k}\right|$, which we then divide elementwise by the empirical standard deviation of the true vector $\mathbf{x_{2}}_k$. Finally, we add the 4 values obtained.

Note that we must use different metrics for $\mathbf{x_{1}}_k$ and $\mathbf{x_{2}}_k$. Since $\mathbf{x_{1}}_k$ is a discrete variable which can take only the values -1, 0, and 1, the only possible absolute errors are 0, 1, and 2, and thus there is no need for normalization. This is of course different for $\mathbf{x_{2}}_k$, the variance of which depends on the situation. To measure the amount of error relatively to the true $\mathbf{x_{2}}_k$, we must therefore take into account its variance.

The average performance computed over 200 experiments for each pair $\{\sigma_{\mathbf{w}}, \sigma_{\mathbf{v}}\}$, using 120 particles is shown in Figure 3.7. The modified algorithm offers a performance that is close to the regular algorithm, but with increasing degradation when both noises become large, as conjectured in 3.4.3.

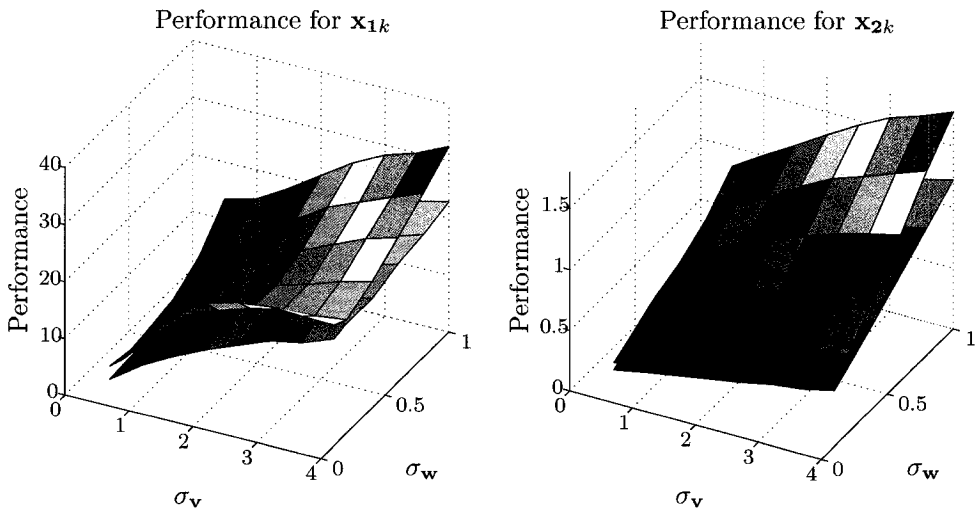
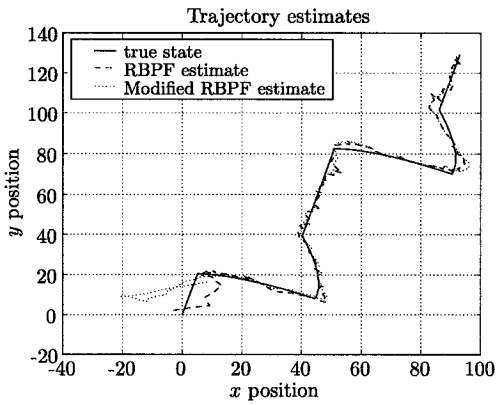
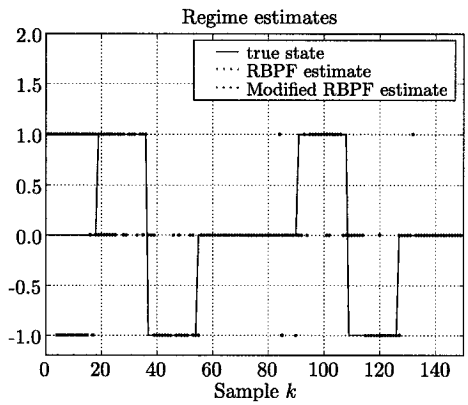


Figure 3.7: Average performance: regular vs. modified RBPF, second example
The performance for the modified RBPF is the upper surface. Convergence appears especially close for small noise variances.



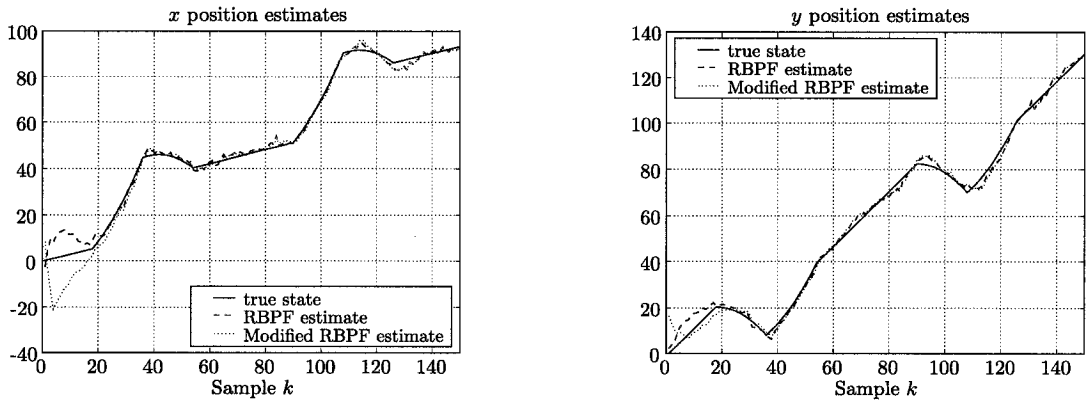
(a) Trajectory estimates (part of the substate $\mathbf{x}_{2k}$)



(b) (discrete) regime estimates

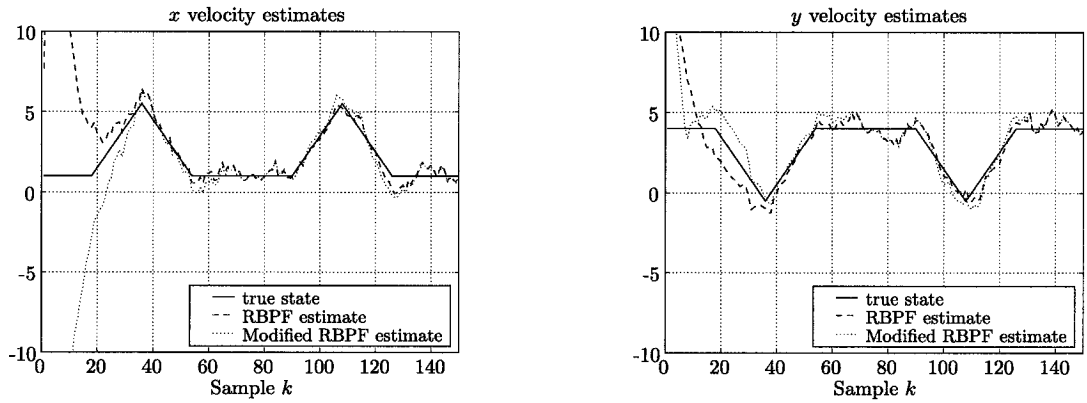
Figure 3.8: Trajectory and regime estimates: regular vs. modified RBPF, second example
The regular RBPF estimates are the red dotted lines, and the modified RBPF estimates are green dotted lines.

We now show in Figures 3.8, 3.9, and 3.10 the results associated with one run of a simulation, with the same parameters as in the simulations of section 3.3.3, except that we choose to use 500 particles. Observe in this situation that again, the regular and modified algorithms demonstrate similar convergence properties. We emphasize that this is no longer the case when noise variances become larger.



(a) x position estimates (part of the substate $\mathbf{x}_{2k}$)(b)y position estimates (part of the substate $\mathbf{x}_{2k}$)

Figure 3.9: Position estimates comparisons for regular and modified RBPF, second example. *The regular RBPF estimates are the red dotted lines, and the modified RBPF estimates are green dotted lines.*



(a) x velocity estimates (part of the substate $\mathbf{x}_{2k}$)(b) y velocity estimates (part of the substate $\mathbf{x}_{2k}$)

Figure 3.10: Velocity estimates comparisons for regular and modified RBPF, second example. *The regular RBPF estimates are the red dotted lines, and the modified RBPF estimates are green dotted lines.*

Algorithm 5 A RBPF algorithm for conditionally linear-Gaussian systems

1. **Define** $p(\mathbf{x}_{1k}|\mathbf{x}_{1k-1})$**, define** $q(\mathbf{x}_{1k,i}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_k)$ **and choose the number of particles** N

2. **Define and initialize the value of** $\{\mathbf{x}_{10,i}\,;\,\mathbf{x}_{20,i}\,;\,\mathbf{K}_{0,i}\}_{i=1}^{N}$**, where** $\mathbf{K}$ **is the covariance matrix of** $\mathbf{x}_2$**, according to a priori belief.**

3. **For every** k**, update the set** $\{\mathbf{x}_{1k-1,i}\,;\,\mathbf{x}_{2k-1,i}\,;\,\mathbf{K}_{k-1,i}\}_{i=1}^{N}$ **as follows:**

- For every $i \in \{1, 2, \ldots, N\}$:
  - Draw $\mathbf{x}_{1k,i} \sim q(\mathbf{x}_{1k,i}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_k)$
  - From $\mathbf{x}_{1k,i}$, obtain the corresponding $\mathbf{A}_{k,i}$, $\mathbf{B}_{k,i}$, $\mathbf{C}_{k,i}$, $\mathbf{D}_{k,i}$, $\mathbf{G}_{k,i}$, $\mathbf{H}_{k,i}$, and $\mathbf{u}_{k,i}$ as applicable.
  - Compute the following:

$$
\begin{aligned}
\mathbf{K}_{k|k-1,i} &= \mathbf{G}_{k,i}{\mathbf{G}_{k,i}}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1,i}{\mathbf{A}_{k,i}}^T \\
\mathbf{T}_{k,i} &= \mathbf{H}_{k,i}{\mathbf{H}_{k,i}}^T + \mathbf{C}_{k,i}\mathbf{K}_{k|k-1,i}{\mathbf{C}_{k,i}}^T \\
\mathbf{x}_{2k|k-1,i} &= \mathbf{A}_{k,i}\mathbf{x}_{2k-1,i} + \mathbf{B}_{k,i}\mathbf{u}_{k,i} \\
\mathbf{y}_{k,i} &= \mathbf{C}_{k,i}\mathbf{x}_{2k|k-1,i} + \mathbf{D}_{k,i}\mathbf{u}_{k,i} \\
\widetilde{w}_{k,i} &= \frac{\mathcal{N}(\mathbf{z}_k|\mathbf{y}_{k,i}, \mathbf{T}_{k,i})p(\mathbf{x}_{1k,i}|\mathbf{x}_{1k-1,i})}{q(\mathbf{x}_{1k,i}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_k)} \\
\mathbf{J}_{k,i} &= \mathbf{K}_{k|k-1,i}{\mathbf{C}_{k,i}}^T\mathbf{T}_{k,i}^{-1} \\
\mathbf{x}_{2k,i} &= \mathbf{x}_{2k|k-1,i} + \mathbf{J}_{k,i}(\mathbf{z}_k - \mathbf{y}_{k,i}) \\
\mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C}_{k,i})\mathbf{K}_{k|k-1,i}
\end{aligned}
$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize the weights (obtain $\{w_{k,i}\}_{i=1}^{N}$)
- Using the weights, resample the set $\{\mathbf{x}_{1k,i}\,;\,\mathbf{x}_{2k,i}\,;\,\mathbf{K}_{k,i}\}_{i=1}^{N}$
- Obtain the state estimates:
  - $\hat{\mathbf{x}}_{1k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{1k,i}$
  - $\hat{\mathbf{x}}_{2k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{2k,i}$

Algorithm 6 Application of a RBPF algorithm: a first example

- For every $i \in 1, 2, .., N$

  - draw $\mathbf{x}_{\mathbf{1}k,i} \sim \mathcal{N}(\mathbf{x}_{\mathbf{1}k}|g(\mathbf{x}_{\mathbf{1}k-1,i}), \mathbf{S}\mathbf{S}^T)$

  - Compute the following:

$$\begin{aligned}
\mathbf{K}_{k|k-1,i} &= \mathbf{G}\mathbf{G}^T + \mathbf{A}_k\mathbf{K}_{k-1,i}{\mathbf{A}_k}^T \\
t_{k,i} &= \sigma_v^2 + {\mathbf{x}_{\mathbf{1}k,i}}^T\mathbf{K}_{k|k-1,i}\mathbf{x}_{\mathbf{1}k,i} \\
\mathbf{x}_{\mathbf{2}k|k-1,i} &= \mathbf{A}_k\mathbf{x}_{\mathbf{2}k-1,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(x_k|{\mathbf{x}_{\mathbf{1}k,i}}^T\mathbf{x}_{\mathbf{2}k|k-1,i}, t_{k,i}) \\
\mathbf{G}_{k,i} &= \mathbf{K}_{k|k-1,i}\mathbf{x}_{\mathbf{1}k,i}t_{k,i}^{-1} \\
\mathbf{x}_{\mathbf{2}k,i} &= \mathbf{x}_{\mathbf{2}k|k-1,i} + \mathbf{G}_{k,i}(z_k - {\mathbf{x}_{\mathbf{1}k,i}}^T\mathbf{x}_{\mathbf{2}k|k-1,i}) \\
\mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{G}_{k,i}{\mathbf{x}_{\mathbf{1}k,i}}^T)\mathbf{K}_{k|k-1,i}
\end{aligned}$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize the weights and resample the set $\{\mathbf{x}_{\mathbf{1}k,i}, \mathbf{x}_{\mathbf{2}k,i}, \mathbf{K}_{k,i}\}_{i=1}^{N}$

Algorithm 7 Application of a RBPF algorithm: a second example

- For every $i \in 1, 2, .., N$

  - draw $\mathbf{x}_{\mathbf{1}k,i} \sim p(\mathbf{x}_{\mathbf{1}k}|\mathbf{x}_{\mathbf{1}k-1,i})$

  - Compute the following:

$$\begin{aligned}
\mathbf{K}_{k|k-1,i} &= \mathbf{G}\mathbf{G}^T + \mathbf{A}\mathbf{K}_{k-1,i}\mathbf{A}^T \\
\mathbf{T}_{k,i} &= \mathbf{H}\mathbf{H}^T + \mathbf{C}\mathbf{K}_{k|k-1,i}\mathbf{C}^T \\
\mathbf{x}_{\mathbf{2}k|k-1,i} &= \mathbf{B}\mathbf{x}_{\mathbf{1}k,i} + \mathbf{A}\mathbf{x}_{\mathbf{2}k-1,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(\mathbf{z}_k|\mathbf{C}\mathbf{x}_{\mathbf{2}k|k-1,i}, \mathbf{T}_{k,i}) \\
\mathbf{J}_{k,i} &= \mathbf{K}_{k|k-1,i}\mathbf{C}^T\mathbf{T}_{k,i}^{-1} \\
\mathbf{x}_{\mathbf{2}k,i} &= \mathbf{x}_{\mathbf{2}k|k-1,i} + \mathbf{J}_{k,i}(\mathbf{z}_k - \mathbf{C}\mathbf{x}_{\mathbf{2}k|k-1,i}) \\
\mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C})\mathbf{K}_{k|k-1,i}
\end{aligned}$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$
- Normalize each of the weights
- Resample the set $\{\mathbf{x}_{\mathbf{1}k,i}, \mathbf{x}_{\mathbf{2}k,i}, \mathbf{K}_{k,i}\}_{i=1}^{N}$

Algorithm 8 The modified RBPF algorithm

Initialization

o Obtain N initial particles of $\mathbf{x}_{10}$

o Choose initial mean $\hat{\mathbf{x}}_{20}$ and error covariance matrix $\mathbf{K}_0$ for an estimate of $\mathbf{x}_{20}$.

Then, for every k, execute the following PF and KF ("exact") parts:

PF part

o For every $i \in 1, 2, .., N$

- Draw $\mathbf{x}_{1k,i} \sim q(\mathbf{x}_{1k}|\mathbf{X}_{1k-1,i}, \mathbf{Z}_k)$

  and set $\mathbf{X}_{1k,i} = \{\mathbf{x}_{1k,i}\ ; \mathbf{X}_{1k-1,i}\}$

- Draw $\mathbf{x}_{2k-1,i} \sim \mathcal{N}(\mathbf{x}_{2k-1}|\hat{\mathbf{x}}_{2k-1}, \mathbf{K}_{k-1})$

- Compute the unnormalized weights

  $\widetilde{w}_{k,i} = \frac{p(\mathbf{z}_k|\mathbf{X}_{1k,i},\mathbf{Z}_{k-1})p(\mathbf{x}_{1k,i}|\mathbf{X}_{1k-1,i},\mathbf{Z}_{k-1})}{q(\mathbf{x}_{1k,i}|\mathbf{X}_{1k-1,i},\mathbf{Z}_k)}$

o Compute the normalizing factor $\sum_i \widetilde{w}_{k,i}$ and obtain normalized weights $w_{k,i}$

o Resample **only** the particles $\mathbf{x}_{1k,i}$, with replacement.

KF/"Exact" part

o Compute $\hat{\mathbf{x}}_{1k} = N^{-1} \sum_{i=1}^{N} \mathbf{x}_{1k,i}$

o Obtain $\mathcal{N}(\mathbf{x}_{2k}|\hat{\mathbf{x}}_{2k}, \mathbf{K}_k) = p(\mathbf{x}_{2k}|\mathbf{X}_{1k} = \hat{\mathbf{X}}_{1k}, \mathbf{Z}_k)$ using a Kalman Filter and $\hat{\mathbf{x}}_{1k}$, $\hat{\mathbf{x}}_{2k-1}$, $\mathbf{K}_{k-1}$ and $\mathbf{z}_k$.

Algorithm 9 Application of the modified RBPF algorithm: a first example

Initialization

- Obtain N initial particles of $\mathbf{x}_{10}$
- Choose initial mean $\hat{\mathbf{x}}_{20}$ and error covariance matrix $\mathbf{K}_0$ for an estimate of $\mathbf{x}_{20}$.

Main algorithm

For every k, do the following:

- Compute $\mathbf{K}_{k|k-1} = \mathbf{G}\mathbf{G}^T + \mathbf{F}_k \mathbf{K}_{k-1} {\mathbf{F}_k}^T$
- For every $i \in 1, 2, .., N$
  - Draw $\mathbf{x}_{\mathbf{1}k,i} \sim \mathcal{N}(\mathbf{x}_{\mathbf{1}k} | g(\mathbf{x}_{\mathbf{1}k-1,i}), \mathbf{S}\mathbf{S}^T)$ and $\mathbf{x}_{\mathbf{2}k-1,i} \sim \mathcal{N}(\mathbf{x}_{\mathbf{2}k-1} | \hat{\mathbf{x}}_{\mathbf{2}k-1}, \mathbf{K}_{k-1})$
  - Compute the following:

$$\begin{aligned} t_{k,i} &= \sigma_v^2 + {\mathbf{x}_{\mathbf{1}k,i}}^T \mathbf{K}_{k|k-1} \mathbf{x}_{\mathbf{1}k,i} \\ \mathbf{x}_{\mathbf{2}k|k-1,i} &= \mathbf{F}_k \mathbf{x}_{\mathbf{2}k-1,i} \\ \widetilde{w}_{k,i} &= \mathcal{N}(z_k | {\mathbf{x}_{\mathbf{1}k}}^T \mathbf{x}_{\mathbf{2}k|k-1,i}, t_{k,i}) \end{aligned}$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$, normalize the weights and resample the set $\{\mathbf{x}_{\mathbf{1}k,i}\}_{i=1}^{N}$
- Compute the following:

$$\begin{aligned} \hat{\mathbf{x}}_{\mathbf{1}k} &= N^{-1} \sum_{i=0}^{N} \mathbf{x}_{\mathbf{1}k,i} \\ t_k &= \sigma_v^2 + {\hat{\mathbf{x}}_{\mathbf{1}k}}^T \mathbf{K}_{k|k-1} \hat{\mathbf{x}}_{\mathbf{1}k} \\ \mathbf{J}_k &= \mathbf{K}_{k|k-1} \hat{\mathbf{x}}_{\mathbf{1}k} t_k^{-1} \\ \hat{\mathbf{x}}_{\mathbf{2}k|k-1} &= \mathbf{F}_k \hat{\mathbf{x}}_{\mathbf{2}k-1} \\ \hat{\mathbf{x}}_{\mathbf{2}k} &= \hat{\mathbf{x}}_{\mathbf{2}k|k-1} + \mathbf{J}_k (\mathbf{z}_k - {\hat{\mathbf{x}}_{\mathbf{1}k}}^T \hat{\mathbf{x}}_{\mathbf{2}k|k-1}) \\ \mathbf{K}_k &= (\mathbf{I} - \mathbf{J}_k {\hat{\mathbf{x}}_{\mathbf{1}k}}^T) \mathbf{K}_{k|k-1} \end{aligned}$$

Algorithm 10 Application of the modified RBPF algorithm: a second example

Initialization

- Obtain N initial particles of $\mathbf{x_{10}}$
- Choose initial mean $\hat{\mathbf{x}}_{\mathbf{20}}$ and error covariance matrix $\mathbf{K}_0$ for an estimate of $\mathbf{x_{20}}$.

Main algorithm

For every k, do the following:

- Compute:

$$\begin{aligned}
\mathbf{K}_{k|k-1} &= \mathbf{G}\mathbf{G}^T + \mathbf{A}\mathbf{K}_{k-1}\mathbf{A}^T \\
\mathbf{T}_k &= \mathbf{H}\mathbf{H}^T + \mathbf{C}\mathbf{K}_{k|k-1}\mathbf{C}^T \\
\mathbf{J}_k &= \mathbf{K}_{k|k-1}\mathbf{C}^T\mathbf{T}_k^{-1}
\end{aligned}$$

- For every $i \in 1, 2, .., N$
  - Draw $\mathbf{x}_{\mathbf{1}k,i} \sim p(\mathbf{x}_{\mathbf{1}k}|\mathbf{x}_{\mathbf{1}k-1,i})$ and $\mathbf{x}_{\mathbf{2}k-1,i} \sim \mathcal{N}(\mathbf{x}_{\mathbf{2}k-1}|\hat{\mathbf{x}}_{\mathbf{2}k-1}, \mathbf{K}_{k-1})$
  - Compute the following:

$$\begin{aligned}
\mathbf{x}_{\mathbf{2}k|k-1,i} &= \mathbf{B}\mathbf{x}_{\mathbf{1}k,i} + \mathbf{A}\mathbf{x}_{\mathbf{2}k-1,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(\mathbf{z}_k|\mathbf{C}\mathbf{x}_{\mathbf{2}k|k-1,i}, \mathbf{T}_k)
\end{aligned}$$

- Compute the normalizing factor $\sum_{i=1}^{N} \widetilde{w}_{k,i}$, normalize the weights and resample the set $\{\mathbf{x}_{\mathbf{1}k,i}\}_{i=1}^{N}$
- Compute the following:

$$\begin{aligned}
\hat{\mathbf{x}}_{\mathbf{1}k} &= N^{-1}\sum_{i=0}^{N}\mathbf{x}_{\mathbf{1}k,i} \\
\hat{\mathbf{x}}_{\mathbf{2}k|k-1} &= \mathbf{B}\hat{\mathbf{x}}_{\mathbf{1}k} + \mathbf{A}\hat{\mathbf{x}}_{\mathbf{2}k-1} \\
\hat{\mathbf{x}}_{\mathbf{2}k} &= \hat{\mathbf{x}}_{\mathbf{2}k|k-1} + \mathbf{J}_k(\mathbf{z}_k - \mathbf{C}\hat{\mathbf{x}}_{\mathbf{2}k|k-1}) \\
\mathbf{K}_k &= (\mathbf{I} - \mathbf{J}_k\mathbf{C})\mathbf{K}_{k|k-1}
\end{aligned}$$

Chapter 4

Selected Speech Processing Background

4.1 A brief summary of the physiology of speech production

In an attempt to provide the reader with a physical rationale for the mathematical models used later on, a basic anatomical and physiological overview of speech production will first be presented, along with some necessary vocabulary to describe the different categories of speech sounds.

We first refer the reader to Figure 4.1, showing a cross-sectional view of the human upper respiratory tract [38]. The main organs involving speech production are the lungs, the larynx, and the vocal tract, which includes the pharynx, the oral and nasal cavities, and the lips, nose and tongue [46]. We begin with a topological description, following Figure 4.1. The lungs are connected to the trachea via the bronchi, and the trachea links the lungs to the larynx. The larynx superiorly opens to the laryngopharynx (the lower part of the pharynx). In addition to the larynx, the pharynx is also attached inferiorly to the esophagus. Superiorly, it opens to the nasal and oral cavities.

At the origin of speech production, a flow of air is expired from the lungs. This flow circulates through the trachea, up to the larynx. The larynx has three main functions [38]: first, it is to provide a path for respiration, from the oral and nasal cavities to the lungs. Secondly, it acts as a router for air and food, which must respectively be traveling in the respiratory tract (through the trachea) and in the digestive tract (through the esophagus). This is accomplished via a cartilage called the epiglottis (see Figure 4.1). The trachea is the path that has the priority, and the larynx has the epiglottis opened during respiration. During swallowing, the

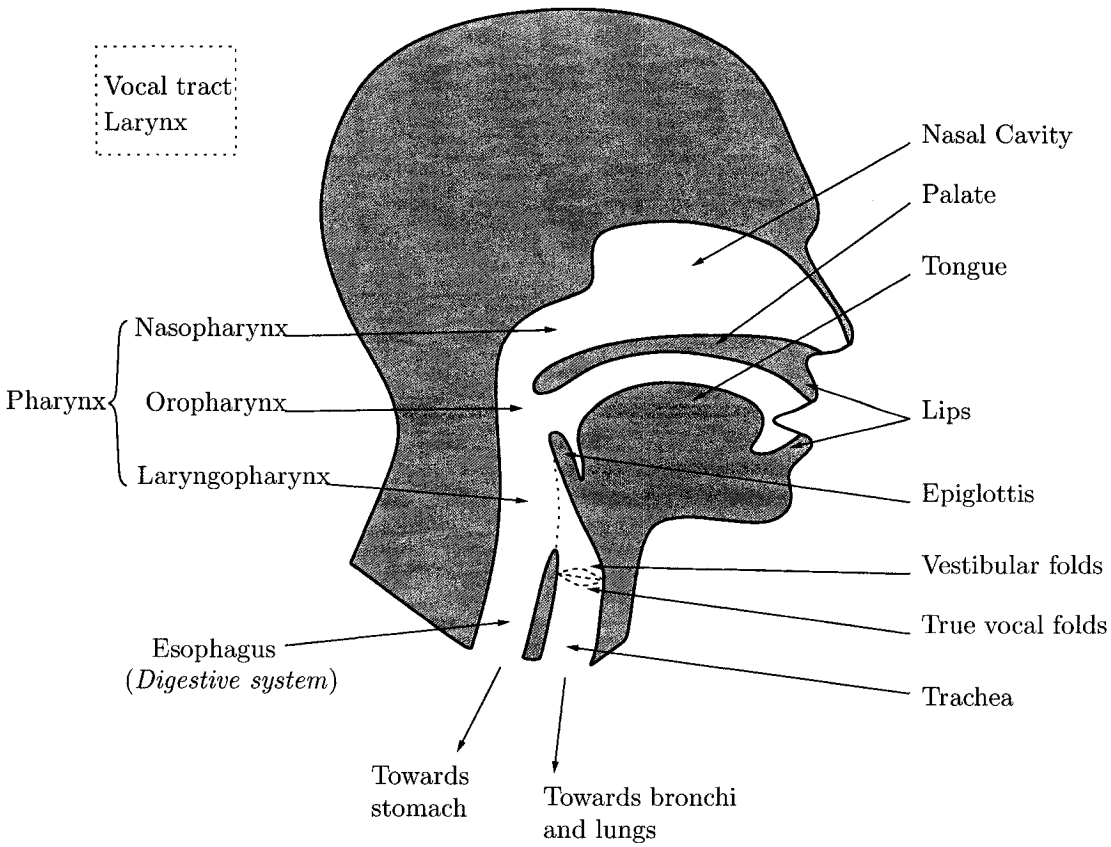


Figure 4.1: Basic anatomy of speech production: upper respiratory tract

larynx moves upwards, causing the epiglottis to close the respiratory tract. Finally, the third function, which is the one pertaining to speech production, is to process the flow of air in order to produce sound. The main intervenants to that purpose are an elastic compound of ligaments and muscle called the *true vocal folds* [38, 46].

The true vocal folds are attached to the surrounding cartilage in the larynx in such a way that their freedom of movement can be constrained in different manners, and that the width of the medial opening between them, called the *glottis*, can be regulated [38, 46]. The air coming from the lungs must pass through the glottis, and the type of configuration of the vocal folds then determines the type of vibration that they are subjected to. The vibration causes the creation of a sound wave, which we also will call a sound source, or a glottal excitation – we will discuss the types of sound sources below. The sound source is then propagated through the pharynx which, with its funnel shape, serves mainly as an amplifier. The rest of the vocal tract shown in Figure 4.1 has the purpose of "shaping" the incoming sound. Effectively, the oral and nasal cavities, amongst other factors, act as resonators. The lips, teeth, and tongue

also contribute greatly in rendering the output speech [38, 46].

During breathing, the glottis is wide and the muscles in the vocal folds are relaxed, offering very low impedance to the airflow, and generating no sound. In contrast, during speech, the glottis is made thinner and the airflow through it is constrained. The possible sound sources coming from the glottis area correspond to two main states for the vocal folds: the *voiced* state and the *unvoiced* state [46]. The voiced state corresponds to a configuration of the larynx causing the vocal folds to oscillate. The glottis is forced to close periodically, and the air flows up the pharynx as a periodic (or quasi-periodic) series of small bursts [46]. In the unvoiced state, the vocal folds do not oscillate, but a turbulence is created at the glottis; the resulting glottal excitation can be roughly described as a white noise. We have thus described two possible glottal sound sources: a quasi-periodic series of small air bursts, and a white form of noise [46]. Besides the sound sources coming from the glottis area, a third type of source can be identified: some impulsive sounds can be generated by elements of the vocal tracts themselves, such as the lips and tongue (for example, for the pronunciation of the letters 'p' or 'k') [38, 46]. This third type can also be seen as unvoiced, in the sense that the vocal cords do not vibrate to produce the sound source.

Typically, the speech sounds resulting from a quasi-periodic series of air bursts in the glottis area (that is, a voiced state of the vocal folds) correspond to when vowels or nasals are spoken: in musical terms, these sounds are pitched, or tonal. As in [46], we conveniently refer to these speech sounds as "voiced". We call "unvoiced" the speech sounds that include plosive sounds ('k', 't', etc.) and fricative sounds ('f', 'sh', etc), which can be qualified as atonal. The unvoiced speech sounds can be generated by unvoiced vocal folds configuration, possibly combined with additional vocal tract impulsive sources [46].

4.2 Linear prediction models of speech signals

The first step in speech processing consists in choosing a model for speech signals. A popular generic model for speech production is the so-called *Source/Filter* model. In simple terms, the speech sound x_k is produced by a source signal u_k passing through a filter with transfer function $H(z)$, as depicted in Figure 4.2. Note that we are here using the term *filter* to describe a system that convolves deterministically its input with a given impulse response to produce its output – we are not using it in the sense of a state estimator.

An important foundation for this model is the belief that the vocal tract is effectively a resonator for the excitation generated by the glottis. We discuss possible models for glottal excitations below, but first we formalize the vocal tract model, i.e. the filter, in further details.

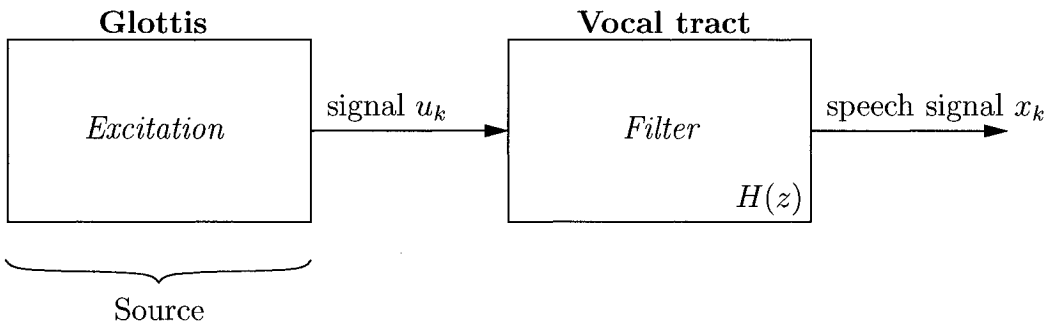


Figure 4.2: Source/Filter model for speech production

The source emits an excitation signal which passes through a filter to produce the speech signal. The source is the model for the glottis, and the filter attempts to model the vocal tract.

As hinted by the physiological summary of speech production presented earlier, experience has shown that the effect of the vocal tract on these sources may be summarized as that of a set of resonators, towards the production of the speech signal. The resonant frequencies of the vocal tract are called *formant frequencies* or *formants* [17, 46]: in this context, the filter can thus be seen as a combination of formant filters. From digital signal processing theory, we know that resonant frequencies correspond to poles in the transfer function $H(z)$. Therefore, a possible realization of a formant filter is an "all-pole" filter, that is, a filter with all of its zeros at the origin of the complex plane. The transfer function of such a filter can be written as:

$$H(z) = \frac{G}{\sum_{m=0}^{M} \alpha_m z^{-m}} \tag{4.1}$$

With the convention $\alpha_0 = 1$, we choose to rewrite the latter equation using coefficients $a_m = -\alpha_m$, as:

$$H(z) = \frac{G}{1 - \sum_{m=1}^{M} a_m z^{-m}} \tag{4.2}$$

For a given excitation input u_k and an output y_k, with z-transforms $U(z)$ and $X(z)$, we thus have:

$$\frac{X(z)}{U(z)} = \frac{G}{1 - \sum_{m=1}^{M} a_m z^{-m}} \tag{4.3}$$

In the time domain, we can write the latter as:

$$x_k = \sum_{m=1}^{M} a_m x_{k-m} + G u_k \tag{4.4}$$

The speech process, as described by equation (4.4), is called *auto-regressive*, for the value of the speech signal value at instant k depends on a linear combination of its previous values[1]. We expected the speech values to depend on past values (the process is obviously not white), however the interesting aspect of equation (4.4) is the fact that it is linear. The fact that the next value of the signal can be predicted from its past values with a linear equation justifies the name for this type of *linear prediction model.* If we let:

$$\begin{cases} \mathbf{x}_k = [x_k \;\; x_{k-1} \;\; \ldots \;\; x_{k-M+1}]^T \\ \mathbf{u}_k = [u_k \;\; 0 \;\; \ldots \;\; 0]^T \\ \mathbf{A} = \begin{bmatrix} a_1 & a_2 & \ldots & a_{M-1} & a_M \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{cases}$$

then, we can rewrite equation (4.4) as:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{u}_k \tag{4.5}$$

There are two important aspects of the model that remain to be discussed: the excitation signal u_k, and the possible evolution of matrix $\mathbf{A}$ over time.

Recall that we had stated that there are three main types of sound sources, but only two of them are glottal. The two possible glottal sound sources are a quasi-periodic train of bursts, and a white noise. To model those, the signal $\mathbf{u}_k$ can be represented as either an impulse train, or a white noise. The third source that we had mentioned was impulsive and originating from the vocal tract. For the sake of simplicity, this source is amalgamated with the glottal sources, and the model for the impulsive vocal tract sound sources are chosen to be a single Dirac

[1] Observe that equation (4.4), written as such with constant values for a_m, can be identified to well-known *Wold representation* for stationary processes $\{x_k\}$. In its broadest definition, a Wold representation for a stationary stochastic process is comprised of a linear, causal filter to which a white noise is fed. As a special case, if the power spectral density of $\{x_k\}$ is a rational function of $z = e^{j2\pi f}$, then $\{x_k\}$ is an autoregressive, moving average process (ARMA). If in addition, the filter is all-pole, the process is simply autoregressive.

impulsion for the signal $\mathbf{u}_k$. For these three main possible excitations, in the context of speech processing, a decision must be made as to which one to use in the model, or whether to allow combinations of the three (and what type of combination). This type of decision is often called the *voicing* decision [17]. In this thesis, we choose to only use for $\mathbf{u}_k$ a white Gaussian noise model, which has been successfully used to model audio and speech in many applications (for example [2, 43, 50]). We keep in mind that to model speech, the excitation noise $\mathbf{u}_k$ cannot be stationary: a possible approach is to use a "white" Gaussian noise with time-varying standard deviation.

In equation (4.5) and in the previous discussion, we have assumed that the filter coefficients (represented by $\mathbf{A}$) are fixed over time. But the vocal tract is able to continuously change shape during speech, and we must thus allow the resonant center frequencies to shift. We must therefore make $\mathbf{A}$ time-varying. We obtain the following model:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{u}_k \tag{4.6}$$

where each of the coefficients a_m are allowed to vary – we write them $a_{m,k}$. We thus have:

$$\mathbf{A}_k = \begin{bmatrix} a_{1,k} & a_{2,k} & \dots & a_{M-1,k} & a_{M,k} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The way that the coefficients are allowed to vary is a subject of debate of its own. We will discuss more on this subject later on, when required. Equation (4.6) is important in our discussion, because it will be the basic building block for most of the proposed particle filtering algorithms in the next chapter, and it will also be referred to in section 4.3.3.

4.3 Some existing speech enhancement techniques

In this section, we will briefly review some existing speech enhancement techniques. We will begin by reviewing a basic Wiener filtering approach, since this is not only one of the most academic approaches, but it is also widely used in practical applications, due to the fact that it does not require heavy computational power. Next, we will introduce classical spectral subtraction schemes, and then we will present some Kalman filter based methods. The main ideas behind signal subspace techniques are explained next, and finally, at the end of this section we

describe in general terms some perceptual enhancement approaches.

In all cases, we are interested in recovering a signal x_k corrupted by additive noise v_k, from the measurements z_k:

$$z_k = x_k + v_k \tag{4.7}$$

4.3.1 Wiener filtering

In this approach, the goal is to determine the coefficients of an FIR filter which, applied to the measurements, would yield an estimate of the original signal. In the context of Wiener filtering, the estimate is chosen to be optimal in the least mean-square sense, that is, the estimate $\hat{x}_k$ is the one that minimizes $\epsilon = \mathcal{E}\{(x_k - \hat{x}_k)^2\}$.

Let us make the assumption that the signal and the noise are uncorrelated and stationary. Let $\mathbf{b} \in \mathbb{R}^{M\times 1}$ represent the weights of the FIR filter, and $\mathbf{z}_{M,k} = [z_k \;\; z_{k-1} \;\; \cdots \;\; z_{k-M+1}]^T$ denote the last M measured values. With this notation, $\hat{x}_k = \mathbf{b}^T\mathbf{z}_{M,k}$, and we have:

$$\begin{aligned} \epsilon &= \mathcal{E}\{(x_k - \mathbf{b}^T\mathbf{z}_{M,k})^2\} &(4.8)\\ &= \mathcal{E}\{(x_k)^2\} - 2\mathbf{b}^T\mathbf{p} + \mathbf{b}^T\mathbf{R}\mathbf{b} &(4.9)\end{aligned}$$

where $\mathbf{p} = \mathcal{E}\{\mathbf{z}_{M,k}x_k\} \in \mathbb{R}^{M\times 1}$ and $\mathbf{R} = \mathcal{E}\{\mathbf{z}_{M,k}\mathbf{z}_{M,k}{}^T\}$.

By solving $\dfrac{d\epsilon}{d\mathbf{b}} = 0$, we obtain the so-called Wiener solution to the problem:

$$\mathbf{b} = \mathbf{R}^{-1}\mathbf{p} \tag{4.10}$$

Observe that this theoretic solution requires the a priori knowledge of $\mathbf{R}$ and $\mathbf{p}$. Alternatively, there is a way to solve this problem in the frequency domain, in order to obtain a frequency response $B(f)$ for the Wiener filter. Let $S_x(f)$ represent the mean power spectral density function of x_k, and $S_{x\hat{x}}(f)$ be the mean cross-spectral density function between x_k and $\hat{x}_k$. For a wide-sense stationary signal, recall that:

$$\mathcal{E}\{|x_k|^2\} = \int_1 S_x(f)df \tag{4.11}$$

If $B(f)$ is the discrete-time Fourier transform of the desired Wiener filter, we can write the

following:

$$\mathcal{E}\{|x_k - \hat{x}_k|^2\} = \mathcal{E}\{|e_k|^2\} \tag{4.12}$$

$$= \int_1 S_e(f)df \tag{4.13}$$

$$= \int_1 \big(S_x(f) + |B(f)|^2 S_z(f) - 2\mathrm{Re}\{S_{x\hat{x}}(f)\}\big)df \tag{4.14}$$

Note now that since we assume in equation (4.7) that x_k and v_k are uncorrelated, the cross-correlation between x_k and $\hat{x}_k$ is:

$$R_{x\hat{x}}(l) = b(l) * R_x(l), \text{ where } l \in \mathbb{Z} \tag{4.15}$$

and with $b(l)$ being the impulse response of the filter. As a result, $S_{x\hat{x}}(f) = |B(f)|S_x(f)$, so that (4.14) becomes:

$$\mathcal{E}\{|x_k - \hat{x}_k|^2\} = \int_1 \big(S_x(f) + |B(f)|^2 S_z(f) - 2|B(f)|S_x(f)\big)df \tag{4.16}$$

$$= \int_1 \big(S_x(f)(1 - 2|B(f)|) + |B(f)|^2(S_x(f) + S_v(f))\big)df \tag{4.17}$$

The integrand is a second degree equation on $|B(f)|$, with coefficients $a = (S_x(f) + S_v(f))$, $b = -2S_x(f)$, and $c = S_x(f)$. For each f, there is a minimum at $|B(f)| = \frac{-b}{2a}$, which is equal to $\frac{4ac-b^2}{4a} = \frac{S_x(f)S_v(f)}{S_x(f)+S_v(f)} \geq 0$. Therefore, from this positiveness the mean-square error $\mathcal{E}\{|x_k - \hat{x}_k|^2\}$ will be minimized if and only if the integrand is minimized for every f. In addition, the value of $|B(f)| = \frac{-b}{2a}$ is:

$$|B(f)| = \frac{S_x(f)}{S_x(f) + S_v(f)} \tag{4.18}$$

The right-hand side of (4.18) is formed of real, nonnegative and even functions of f. Thus if we are looking for a real solution for $B(f)$, we can simply remove the absolute value in (4.18), and we obtain[1]:

$$B(f) = \frac{S_x(f)}{S_x(f) + S_v(f)} \tag{4.19}$$

It can be seen that if the mean PSD of the noise is $S_v(f) = 0$, then $B(f) = 1$ and the right-hand side of (4.17) vanishes. As expected from the result in the time-domain, we require substantial information about the signal and the noise to apply a Wiener filter. In practice, the spectra of the signal and of the noise are a priori unknown and time-varying, and therefore

[1]The resulting ideal time-domain filter is then zero-phase, non-causal and has an infinite length. To obtain a causal, FIR filter, the usual methods of delay/truncation can be used. In practice, including a delay means that a linear-phase term $e^{j\theta}$ where θ is a linear function of the frequency, is introduced ($B(f)$ is then not constrained to be real, but its modulus still follows (4.18)). A truncation respecting the coefficients symmetry will then ensure a linear-phase result. Note that the choice of the "truncation width" amounts to choosing, initially, the value of M in the temporal derivation of the Wiener filter presented at the beginning of this section.

it is common to use some windowing on the data to be filtered (i.e., to apply the filter only on frames), and consider that during the period of observation, the signals are almost stationary, so that rather than estimating the ensemble mean PSDs, we estimate them based on truncated realizations. Obviously, some additional techniques must be employed to estimate the signal and noise spectra in order to be able to use the filter.

Although there are important differences, Wiener filtering is closely related to spectral subtraction, as we will see in the next section.

4.3.2 Spectral subtraction

The main idea behind spectral subtraction is very simple: given a signal corrupted by additive noise, estimate the spectrum of the noise and subtract it from the spectrum of the observed mixture, and then revert to time-domain to obtain the cleaned signal. Figure 4.3 shows the high-level steps of basic spectral subtraction.

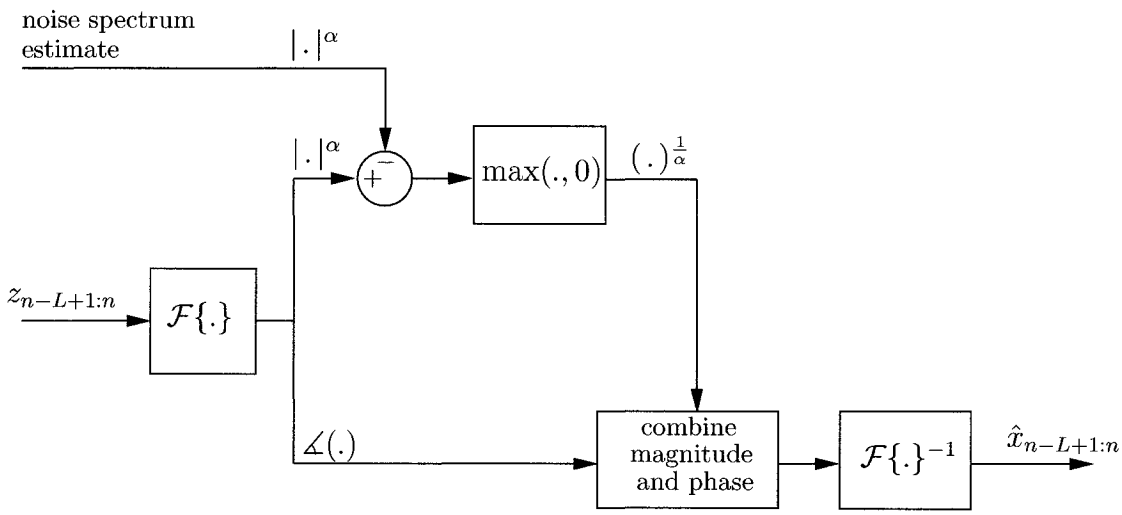


Figure 4.3: Basic spectral subtraction

The major steps of spectral subtraction, applied to a frame of length L, are shown above. The noise spectrum estimate may be obtained from previous frames. At the output, an overlap-add type scheme accompanying the inverse Fourier transform is understated if short-time, window processing is used.

As for the practical version of Wiener filtering, spectral subtraction is typically performed on frames of windowed data. The assumptions that we make are similar: the noise and the signal are uncorrelated, and within a frame (or hopefully several frames), the noise is almost

stationary. In addition, it is assumed that the effect of additive noise on the phase of the spectrum of the original signal is negligible to the human ear, so that the phase of the spectrum of the noisy signal is practically considered to be clean.

Following Figure 4.3, the first step is to convert the windowed measurements z_k over a frame to the frequency domain. Denote by $Z_w(f)$ the obtained spectrum for the windowed data. We separate the magnitude $|Z_w(f)|$ and the phase $\angle Z_w(f)$. We then require an estimate for the magnitude of the noise spectrum $|\hat{V}_w(f)|$ to be subtracted over the frame. There are different ways to do so: some techniques detect the absence of speech to update this estimate as frequently as possible (this requires the aid of so-called VADs, or voice activity detectors). Other simpler schemes form the estimate from initial or offline measurements, where it is known that only noise is being recorded. The spectral subtraction can then take place as follows:

$$|\hat{X}_w(f)| = |Z_w(f)| - |\hat{V}_w(f)| \tag{4.20}$$

It was found that it may be beneficial to perform the subtraction in a more general way, as follows:

$$|\hat{X}_w(f)|^\alpha = |Z_w(f)|^\alpha - \beta|\hat{V}_w(f)|^\alpha \tag{4.21}$$

where α and β are positive numbers. In practical implementations, it is wise to ensure that the right-hand side remains positive, and so we may rather write:

$$|\hat{X}_w(f)|^\alpha = \max\{|Z_w(f)|^\alpha - \beta|\hat{V}_w(f)|^\alpha, 0\} \tag{4.22}$$

To complete the procedure, we use our assumption that the phase of the clean signal is close enough to that of the observed signal, so that the final expression for the estimated clean signal is:

$$\hat{x}_{w,k} = \mathcal{F}^{-1}\Big\{\max\{\big(|Z_w(f)|^\alpha - \beta|\hat{V}_w(f)|^\alpha\big)^{\frac{1}{\alpha}}, 0\} \times e^{j\angle Z_w(f)}\Big\} \tag{4.23}$$

We mentioned that there is a close relationship between Wiener filtering (equation (4.19)) and spectral subtraction. With $\alpha = 2$ and $\beta = 1$, equation (4.21) can be rewritten as:

$$|\hat{X}_w(f)|^2 = |Z_w(f)|^2 \left(1 - \frac{|\hat{V}_w(f)|^2}{|Z_w(f)|^2}\right) \tag{4.24}$$

$$= |Z_w(f)|^2 \left(\frac{|Z_w(f)|^2 - |\hat{V}_w(f)|^2}{|Z_w(f)|^2}\right) \tag{4.25}$$

If the estimate of the magnitude of the noise spectrum is perfect, then (4.25) becomes:

$$|\hat{X}_w(f)|^2 = |Z_w(f)|^2 \left(\frac{|X_w(f)|^2}{|X_w(f)|^2 + |V_w(f)|^2}\right) \tag{4.26}$$

Therefore, the effect of spectral subtraction can be identified to that of a filter with frequency response $B_{\text{sub}}(f)$ such that:

$$B_{\text{sub}}(f) = \left(\frac{|X_w(f)|^2}{|X_w(f)|^2 + |V_w(f)|^2} \right)^{\frac{1}{2}} \tag{4.27}$$

The relationship to equation (4.19) is now more apparent. With much language abuse and with the relatively strong assumption that the mean power spectral density of the signals coincide with the windowed power spectra of (4.27), then we could say that "spectral subtraction is the square-root of Wiener filters". In pratice though, both methods use short-time spectrum estimation techniques, and are considered to be of the same family of "subtractive-type" algorithms.

We must mention here that in general, subtractive-type algorithms suffer from an important drawback: the enhanced signal contains artefacts, including a sort of musical, robotic noise, sometimes described as a "synthetic bird-song" in the background. This phenomenon is due to the zero-flooring (the maximum operation in equation (4.23)) that is applied to the short-time spectra of the frame under consideration, after subtraction. In the short-time spectrum of the actual true noise, some peaks can appear "randomly" in frequency. After subtraction and flooring, some of the narrower remaining peaks can be identified to time-varying tones, which constitute the musical noise. As we will see in 4.3.5, some techniques exist to further process the signal to reduce these artefacts.

4.3.3 Kalman Filter-Based methods

These methods are in fact a subclass of the so-called "model-based" speech enhancement algorithms. For these techniques, the procedures are derived based on a proposed model for speech signals. In section 4.2, we presented a linear prediction model which naturally calls for an algorithm of the type of the Kalman filter. Let us rewrite here equation (4.6):

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{u}_k \tag{4.28}$$

where the matrix $\mathbf{A}_k$ contains the auto-regressive coefficients (see section 4.2). Let us consider the valid case where $\mathbf{u}_k$ is white and Gaussian (although with a time-varying variance). There exists a method, called the *autocorrelation* method or the *Yule-Walker* method, to estimate, given a segment of a signal assumed to be stationary (which mean here that $\mathbf{A}_k$ is constant and $\mathbf{u}_k$ has a constant variance), the autoregressive coefficients as well as the variance of the prediction error (which is equivalent to the variance of the Gaussian noise $\mathbf{u}_k$). This method is based on the fact that there is a relationship between the autocorrelation function and the autoregressive coefficients. This relationship is linear (the equations for the resulting

system are called the *Yule-Walker* equations), and has a special form as the matrix intervening is Toeplitz, and therefore one can use algorithms such as the Levinson-Durbin recursion to solve it efficiently.

In practical terms, the approach consists of observing a frame of the signal x_k, short enough to be considered stationary. Typically, for a speech signal we may consider a frame length of a couple of milliseconds at most. Then, we compute an estimate for the (statistical) autocorrelation function, and we use it to solve the Yule-Walker equations and obtain in turn an estimate for the matrix $\mathbf{A}_k$ and the input noise variance over the frame.

We can then use a Kalman filter running on the following system:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{u}_k \tag{4.29}$$

$$z_k = \mathbf{C}\mathbf{x}_k + v_k \tag{4.30}$$

where $\mathbf{C} = [1\ 0 \ldots 0]$.

A KF-based algorithm must generally obtain sequentially estimates for $\mathbf{A}_k$, as well as estimates of the statistics of the noise v_k. In practice, the KF may call every couple of milliseconds an external function to update the value of $\mathbf{A}_k$, and as we have seen this external function can be an implementation of the autocorrelation method. However, if such a method is to be used, then the argument of the external function should theoretically be the clean speech signal that we are trying to estimate!

To circumvent this problem, there exist techniques to estimate the parameters for the speech model from noisy data, for example [13, 15, 33, 43]. In the family of KF-based methods, one of the simplest approaches is to apply the Yule-Walker method using the current estimate of clean speech, such as in [33]. Our experience is that although it usually performs fairly well, the resulting algorithm is less robust and has a tendency to fail (diverge). Generally, in many sources related to KF-based speech enhancement, an external algorithm attempts to estimate the speech parameters from the enhanced signal, and then returns them to the KF. In [15], these two steps are effectively unified with the introduction of an Expectation-Maximization algorithm, which simultaneously estimates the additive noise parameters.

In the basic context of white Gaussian measurement noise, where estimating the additive noise statistics is considerably simplified, we have found that if the current frame of corrupted measurements is directly used for the Yule-Walker method, then the resulting enhanced speech compares favourably to such results as those obtained from using [15]. To support that claim,

comparisons were made using audio samples downloaded from the webpage of the author of [15], `www.eng.biu.ac.il/~gannot/examples1.html`. We used three of the quality measures described in section 4.4 to do the comparisons: overall signal-to-noise ratio (overall SNR), average segmental SNR (ASSNR), and PESQ scores. In the website above, we consider the two cases where white Gaussian noise is added (the Kalman filter based algorithm is denoted by "kem" in the page). On the other hand, we run a Kalman filter using directly the corrupted measurements as the basis for the autocorrelation method, and we abbreviate this simplistic method by KF+YW (the code used for the simulations corresponding to the third column in the table below is available at `www.site.uottawa.ca/~mustiere/`). The results are shown in Tables 4.1 and 4.2[2].

Female speaker	noisy signal	[15]	KF+YW
SNR (dB)	6.51	9.95	10.32
ASSNR	-1.05	0.49	0.35
PESQ	1.885	2.130	2.138

Table 4.1: Performance comparison between two Kalman filter-based algorithms, case 1
The measurement noise is white Gaussian, and the speaker is a female.

Male speaker	noisy signal	[15]	KF+YW
SNR (dB)	9.50	12.75	12.13
ASSNR	0.90	2.85	2.60
PESQ	2.296	2.564	2.523

Table 4.2: Performance comparison between two Kalman filter-based algorithms, case 2
The measurement noise is white Gaussian, and the speaker is a male.

Observe that the results are in the same range, and in fact the two pieces of processed speech sound extremely close – subjectively, we are forced to conclude that they seem to be "enhancing" in a similar manner. One may wonder why feeding corrupted mesurements to the Yule-Walker method performs that well. We believe that this is only the case for additive white Gaussian measurement noise, because of its reduced impact on the autocorrelation function of

[2]Note that before processing, we converted all the sounds presented in the webpage from 44.1 kHz to 8 kHz. From a basic spectral analysis under Matlab, it is confirmed that the sounds are bandlimited to 4 kHz (we can suspect that they were later upsampled from 8 kHz to 44.1 kHz). Therefore, this conversion is quasi lossless and a comparison is allowed. Without this conversion, the comparison is not possible since the additive noise is no longer white after upsampling.

the speech (we can expect most of this impact to be found only around the zero lag). In addition, such a simple method should work well as the SNR improves, as the impact of the Gaussian noise itself on the speech decreases.

We thus decide to use our simple implementation as a basis for comparisons in the next chapter, since it appears to be a good representation of what type of results KF-based methods can yield in AWGN conditions. We insist now that our claim is certainly not that this simple implementation is better than the ones presented in [15] or [13,33,43], since it can only be used as such for white Gaussian environment noises. The scheme to estimate the noise statistics is much more basic in the simulation above: the noise variance is empirically computed initially from the first few samples of data, and then fed to the Kalman filter. Nevertheless, our implementation is sufficient to provide an illustration about how this family of methods behave in additive white Gaussian noise.

4.3.4 Signal subspace

The signal subspace approach, introduced in [11], is another different class of method used for speech enhancement. It is based on a *linear model* for speech signals. Suppose here that $\mathbf{x}$ represents a frame of length M for the clean speech signal. In the linear model, it is assumed that $\mathbf{x}$ can be written as[1]:

$$\mathbf{x} = V\mathbf{s} \tag{4.31}$$

where $V \in \mathbb{R}^{M\times K}$ is full rank, $\mathbf{s} \in \mathbb{R}^K$ is a zero mean complex random vector with (nonsingular) covariance matrix $\mathbf{R_s}$, and where we impose $K < M$.

First recall that for zero mean processes, the covariance matrix is equal to the correlation matrix. In the following, we may use either terms intechangeably.

Since $K < M$, then the range of V, denoted here $\mathcal{R}(V)$ is a proper subspace of $\mathbb{R}^M$. This representation implies several assumptions:

- First, clearly $\mathcal{E}\{\mathbf{x}\} = 0$ since $\mathcal{E}\{\mathbf{s}\} = 0$.
- The covariance/correlation matrix of $\mathbf{x}$ is given by $\mathbf{R_x} = \mathcal{E}\{\mathbf{x}\mathbf{x}^T\} = \mathcal{E}\{V\mathbf{s}\mathbf{s}^T V^T\} = V\mathcal{E}\{\mathbf{s}\mathbf{s}^T\}V^T = V\mathbf{R_s}V^T$
- $\mathbf{R_x} \in \mathbb{R}^M$ has thus rank K only, and thus $\mathbf{R_x}$ only has K nonzero eigenvalues.

[1]Such a model for speech signals has been validated by successful experimentations and practical implementations [11].

Suppose that a frame of white noise $\mathbf{v}$ is added to the frame $\mathbf{x}$. In contrast with the speech signal, since $\mathbf{v}$ is a white random vector, its covariance matrix $\mathbf{R}_\mathbf{v}$ is (diagonal and) nonsingular and all of its eigenvalues are nonzero – they are in fact all equal to the variance of the noise. Therefore, $\mathbf{v}$ lies in the entire space $\mathbb{R}^M$, but $\mathbf{x}$ only lies in $\mathcal{R}(V) \subset \mathbb{R}^M$.

$\mathcal{R}(V)$ is termed the *signal subspace*. The orthogonal complement of $\mathcal{R}(V)$ is called the *noise subspace* (their union forms $\mathbb{R}^M$). Thus, in a frame of a noisy speech signal, the noise lies in both the signal and the noise subspaces, but the clean speech is only confined to the signal subspace.

From these considerations, we can summarize the signal subspace approach in two steps as follows:

- Given a frame of noisy speech, first try to determine the signal and the noise subspaces, and compute the orthogonal projection of the noisy speech onto the signal subspace.
- Secondly, from the resulting vector, obtain an estimate for the clean speech.

Determination of the signal subspace

We will not go into the full derivation of the signal subspace method, but we still present the fundamental tool used for the subspaces determination: it is the Karhunen-Loève transformation/expansion (KLT or KLE). Suppose that an M-dimensional vector $\mathbf{x}$ is an observation of a zero mean wide-sense stationary discrete random process with correlation matrix $\mathbf{R}_\mathbf{x}$. Since by definition $\mathbf{R}_\mathbf{x}$ is Hermitian and positive semi-definite, its Schur decomposition has the following form:

$$\mathbf{R}_\mathbf{x} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H \tag{4.32}$$

where $\mathbf{Q}$ is the matrix of eigenvectors $\{\mathbf{q}_i\}_{i=1}^M$ associated with the diagonal matrix of eigenvalues $\mathbf{\Lambda}$. $\mathbf{Q}$ is necessarily unitary if $\mathbf{R}_\mathbf{x}$ is strictly positive definite (if all its eigenvalues are strictly positive), otherwise the subset of $\{\mathbf{q}_i\}_{i=1}^M$ corresponding to zero eigenvalues, although still mutually orthogonal, may be arbitrarily scaled. The KLT then states that:

$$\begin{aligned} \mathbf{x} &= \mathbf{Q}\mathbf{Q}^H\mathbf{x} \\ &= \sum_{i=1}^{M} \mathbf{q}_i^H \mathbf{x} \mathbf{q}_i \\ &= \sum_{i=1}^{M} c_i \mathbf{q}_i \end{aligned} \tag{4.33}$$

Furthermore, the coefficients in the expansion of (4.33) are zero-mean and uncorrelated. The interest for the KLT is now clear: since we had stated that $\mathbf{R_x} \in \mathbb{R}^M$ has rank K only, and that $\mathbf{R_x}$ only has K nonzero eigenvalues, then we can apply the KLT assuming that the speech is a zero mean wide-sense stationary discrete random process, and (4.32) may be written as:

$$\mathbf{R_x} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H \tag{4.34}$$

$$= [\mathbf{Q}_1\ \mathbf{Q}_2] \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\mathbf{Q}_1\ \mathbf{Q}_2]^H \tag{4.35}$$

And therefore, effectively the matrix $\mathbf{Q}_1\mathbf{Q}_1^H$ is the orthogonal projector onto the signal subspace that we are looking for. Moreover, we can write from (4.33) that:

$$\mathbf{x} = \sum_{i=1}^{K} c_i \mathbf{q}_i \tag{4.36}$$

where it is understated that $\{\mathbf{q}_i\}_{i=1}^K$ corresponds to the columns of the matrix $\mathbf{Q}_1$. Note also that even if we choose $K' < K$, the resulting representation has the property of still being optimal in the mean-square sense.

In the case of additive zero mean noise $\mathbf{v}$ with covariance matrix $\mathbf{R_v} = \sigma_v^2\mathbf{I}$, then we only observe the corrupted frame $\mathbf{z}$. But we have, from the uncorrelatedness of the signal and the noise:

$$\mathbf{R_z} = \mathbf{R_x} + \sigma_v^2\mathbf{I} \tag{4.37}$$

Since the KLT method requires the knowledge of $\mathbf{R_x}$, we must compute empirically an estimate for $\mathbf{R_z}$, and also estimate σ_v^2. Practical techniques are explained in [11].

Estimation of the clean speech

Once a projection of the noisy speech onto an estimate for the signal subspace is obtained, there remains to estimate the clean speech. Several estimators are presented in [11,31]; these estimators are principally concerned with minimizing the signal distortion while removing/reducing the remaining contribution from the noise present in the signal subspace. We refer the reader to these sources for more details on the subject.

Signal subspace methods are recognized as being superior to classical spectral subtraction schemes, in the sense that they introduce less musical noise in the enhanced speech, although they still do and therefore the resulting enhanced speech may still sound unnatural. One drawback is that they require an estimate for the clean autocorrelation matrix to operate,

which may be not be straightforward to obtain in certain conditions, and especially at low SNR.

4.3.5 On perceptual approaches in general

Researchers in the field of speech enhancement have been naturally led to study the human beings auditory system from a psychoacoustical standpoint. It is legitimate to assume that an algorithm taking into account the properties of listeners has the potential to perform better than another algorithm that only relies on mathematical concepts.

We have stated at the beginning of this section that in general, the more an algorithm suppresses noise, the higher the risk of distortion on the resulting speech is, and in fact the higher the likelihood of appearance of artefacts becomes. The idea of psychoacoustical or perceptual approaches is that if the human listener cannot perceive the noise that is contained in a component of the corrupted signal, then there is no need to try to suppress that noise, and we can expect less artefacts to appear in the resulting signal by doing so. The intelligibility of the enhanced speech should therefore be more preserved.

This idea comes from the fact that it was found that the mere presence of noise in parts of the spectrum of a signal does not automatically mean that the listener will perceive it. The noise component may indeed be effectively masked by the speech component. To picture this phenomenon, imagine being in a parked car, with the engine off, playing some perfectly audible radio music. Later, on the highway, the radio is still on at the same level, but for the driver it is as if it was off, since the car's engine is completely masking it. If a hypothetical algorithm were to try to suppress the background music anyways in order to hear better the engine noises, it would only make the resulting engine noises sound less realistic because of the distortion that it would introduce. Note that this masking occurs not only in the frequency domain, but also in the time domain: one sound occuring very close in time to the next or previous noise may be masked by the latter.

Let us consider here the frequency domain masking only. In order to be able to decide whether a given spectral component in a noisy speech signal is *perceptually* noise-free, there is a method to compute the so-called *masking threshold* of a signal, which is a real function of frequency (and obviously time), and has the units of power (it may be expressed in dB). The masking threshold should theoretically be computed from the clean, original speech signal, and basically indicates, for a given frequency, what level (in terms of spectral power) is required for an additive noise to start impacting the listener's perception of the signal. If the noise is located below the masking threshold, it is virtually impossible to detect its presence.

The steps for computing the masking threshold for a speech signal are summarized in [51]. We will not go into the detailed description of the computation of the masking threshold. These steps were determined and adjusted from the analysis of the frequency selectivity of the human ear.

In [51], the author presents a perceptual noise reduction scheme applicable to spectral subtraction. The method is based on the estimation of the masking threshold described above. Recall the basis formula for spectral subtraction, that we had written in 4.22:

$$|\hat{X}_w(f)|^\alpha = \max\{|Z_w(f)|^\alpha - \beta|\hat{V}_w(f)|^\alpha, 0\} \tag{4.38}$$

It is also possible to reformulate spectral subtraction as a filter, with gain $G(f)$ in the frequency domain. Equation (4.38) becomes:

$$|\hat{X}_w(f)| = |Z_w(f)| \times G(f) \tag{4.39}$$

$$= |Z_w(f)| \times \max\left\{\left(1 - \beta\frac{|\hat{V}_w(f)|^\alpha}{|Z_w(f)|^\alpha}\right)^{\frac{1}{\alpha}}, 0\right\} \tag{4.40}$$

Equation (4.40) is in fact not the most flexible and general formula that can be used for spectral subtraction. In [51], instead of forcing a minimum of 0, this minimum is allowed to be parametrized by the value of γ. The formula for the gain $G(f)$ proposed is of the form:

$$G(f) = \begin{cases} \left(1 - \beta\frac{|\hat{V}_w(f)|^\alpha}{|Z_w(f)|^\alpha}\right)^{\frac{1}{\alpha}} & \text{if } \frac{|\hat{V}_w(f)|^\alpha}{|Z_w(f)|^\alpha} < \frac{1}{\beta+\gamma} \\ \left(\gamma\frac{|\hat{V}_w(f)|^\alpha}{|Z_w(f)|^\alpha}\right)^{\frac{1}{\alpha}} & \text{otherwise} \end{cases} \tag{4.41}$$

In [51], it is proposed that the amount of suppression as a function of frequency, as represented by (4.41), should be adapted depending on the value of the estimated noise masking threshold of speech, according to the principles outlined in this section. To do so, β and γ are made functions of frequency and time, and an adaptation rule is derived. Simulation results show improvements in several situations and with several types of quality measures.

4.4 Some tools used for the assessment of speech quality

To test the performance of a speech enhancement algorithm, the researcher's dream would be to be able to obtain, instantly, the opinion of as many individuals as possible about the way the enhanced speech sounds. This is clearly impossible, although there are ways to obtain such opinion scores via standardized panel tests. There exists a widely recognized recommendation

from the ITU-T (International Telecommunication Union, Telecommunication Standardization sector) thoroughly describing a methodology for the subjective assessment of speech quality. This is the P.800 recommendation, and it is used to create so-called *Mean Opinion Scores*, or MOS tests. Unfortunately, usually in practice the population available is limited, the results are highly variable and sometimes ambiguous, and the whole process may take too long to be practical, especially for initial testing purposes. Therefore some mathematical, objective quality measures are necessary. Ideally, these measures should be as highly correlated as possible to a standard MOS. In other words, we would like the measures to be good predictors of average subjective preferences. In this work, we chose to use a few different measures, each having advantages and disadvantages. We claim, however, that none of the currently existing speech quality assessment tools are entirely satisfactory. However, it does not mean that they are unsignificant, in fact it is the contrary: all of them bear some important information, and they could all be seen as "indicators" or "symptoms" of a type of degradation in the enhanced signal, which can then be strongly suspected, and possibly confirmed by subjective listening tests.

In this section we pave the way for the comparisons in the next chapter by introducing the different measures of speech quality that we will be using.

4.4.1 Overall SNR

The classic overall signal-to-noise ratio (SNR) is simply the ratio between a (clean) signal's energy and the energy contained in the noise or error. Recall that the energy of a signal x_k is defined as $E = \sum_k |x_k|^2$. In the context of speech enhancement, the signal is the original, clean speech signal x_k, and the noise is the difference between x_k and its estimate $\hat{x}_k$. The SNR, expressed in dB, is defined as:

$$\text{SNR}_{dB} = 10\log_{10}\frac{\sum_k x_k^2}{\sum_k (x_k - \hat{x}_k)^2} \tag{4.42}$$

The SNR is clearly higher if the squared difference between the estimated speech and the original speech signal is smaller. In the context of speech enhancement, it is considered that the SNR is not a reliable indicator of intelligibility and speech quality. Concretely, the human auditory system simply does not base its quality assessment on the average error squared.

Still, the overall SNR can be an interesting "first glance" at the performance of an algorithm: a very low SNR is usually not a good sign! In addition, it is very inexpensive computationally, and thus it can be used as a simple online indicator that, for example, an algorithm is not on a divergent path.

4.4.2 Average segmental SNR

The average segmental SNR (ASSNR) is based on the same principle as the overall SNR, except that it is computed as the average of the SNR of segments, or frames (possibly overlapping and windowed). In this work, where we are focusing on speech sampled at 8 kHz, we use a Hanning window of width 30 milliseconds (240 samples), and 75% overlaps (that is, the window is anchored every 7.5 milliseconds (60 samples). So, if L frames are used to cover a signal, and if SNR_l is the SNR_{dB} computed on the l^{th} frame, then we have:

$$\text{ASSNR} = \frac{1}{L}\sum_{l=1}^{L}\text{SNR}_l \tag{4.43}$$

The ASSNR is a more insightful quality measure than the overall SNR, in the sense that it is a better MOS predictor [22, 29]. Based on the multiple experiments and informal listening tests conducted in this work, we find that in general a higher ASSNR means less residual background noise. This finding is in accordance to a study presented in [26], in which objective measures for speech enhancement are evaluated. In [26], listeners are given an enhanced signal, and are asked to give three scores from 1 to 5. They must first rate the distortion on the speech signal itself ("SIG" score, in terms of naturalness), then their must rate the background noise (the "BAK" score, in terms of intrusiveness), and finally they must give an overall score (the "OVRL" score). Then, the correlation between several objective measures (including the ASSNR) and these three scores is inferred. The ASSNR is found to be much more correlated to the "BAK" score than to the "SIG" score.

The ASSNR has its limitations: we have observed several cases where a very high value corresponded to a speech signal with very small residual noise but with significant distortion, such as the beginning of some words effectively "cut-off", resulting in reduced intelligibility.

In the computation of the signal-to-noise ratio, no voice-activity detection (VAD) was used in the thesis, and it was not considered necessary for the following reasons. First, the different test speech signals used for the comparisons in chapter 6 are only single sentences, and therefore there are no significant pauses. More specifically, among the three different clean signals used, only one of them contains a complete silence of approximately 200 samples (which is less than the width of the Hanning window) in between utterances. Secondly, in our implementation, the average segmental SNR values are lower bounded by -10 dB, as suggested in [21] and [44] in order to dampen the influence of speech pauses; accessorily this avoids the necessity of using VADs when pauses are brief enough.

4.4.3 PESQ scores

We mentioned earlier the ITU-T group, who proposed a methodology for developing mean opinion scores. The same group later approved the PESQ (Perceptual Evaluation of Speech Quality) algorithm as an objective method to predict the results of subjective MOS tests (in a P.800 listening setup), designed purposely for handset telephony speech codecs. This is the ITU-T recommendation P.862, approved in 2001. Although PESQ scores were not designed for speech enhancement algorithms evaluation, they are still found to provide a meaningful indication of performance and they are frequently used by researchers for this purpose.

The PESQ algorithm compares the original, clean speech signal to the output of the enhancement algorithm, and penalizes the final score based on measures of the distortion. The PESQ is perceptual in the sense that the amount of distortion is measured in the context of a model for the human auditory system. The specific term to describe distortion in the human auditory domain is the *disturbance*.

A high-level description of the PESQ algorithm is now presented.

- The first step consists of a **level alignment**, so that both the clean and the processed signals are brought to the same power level.
- Next, the level aligned signals are **filtered** through a filter which models the receive path of standard telephone handsets.
- Once the filtering is done, the signals must be **time aligned**, since the PESQ takes into account the fact that the (telephony) system that must be tested may introduce variable delays. The scheme used employs a voice activity detector, so that speech utterances can be overlapped in the two signals. Note that PESQ is able to align the signals even if the enhanced speech is missing parts (to account for the fact that telephony systems may suffer from packet losses).
- The next processing applied to the signals is an **auditory transform**. This places the signals in a perceptual context, for this transformation extracts from the signals an estimate of the perceived loudness as a function of both time and frequency. This function is called the *sensation surface*.
- Following the auditory transformation, the PESQ algorithm can move on to the **disturbance processing** part. The difference between the *sensation surfaces* forms an *error surface*, from which two average disturbance parameters are computed.
- Finally, the two disturbance parameters are converted to a single MOS score.

PESQ scores range from -0.5 to 4.5 (even though MOS scores can go from 1 to 5, the PESQ algorithm is trying to simulate an average subjective result, and for most cases PESQ scores are located between 1 and 4). According to the ITU-T, PESQ scores demonstrate a good correlation with what one could expect from subjective tests. Even though this is generally the case, the PESQ cannot be blindly trusted. For example, in `http://microtronix.ca/pesq-disc.html`, audio examples are shown where two degraded signals obtain the same PESQ score, even though one of them is of significantly lower quality.

In addition, in some of our experimentations, we have found that if the enhancement algorithm is such that the enhanced speech signal's amplitude is a modulated version of the amplitude of the original signal (i.e., if the amplitude of the processed signal is distorted in time), then the PESQ score of the output might be very penalized, even if intelligibility is improved. If only speech is of concern, then such an amplitude distortion, as long as it is not excessive, does not have a significant impact on the perceived speech quality. It is of course more crucial if music is being tested.

For these reasons, the PESQ, like the overall SNR and the segmental SNR, cannot be used alone. But these three indicators still contain some information, and we will therefore monitor all of them when doing comparisons in the next chapter. Next, we present two additional speech quality measures, the Log-Area Ratio and the Weighted Spectral Slope, both of which focus on the evaluation of the *spectral distortion* which may occur in the enhanced speech.

4.4.4 Log-Area Ratio scores

The Log-Area Ratio (LAR) score is another objective speech quality measure; it is recommended in [21] for the evaluation of speech enhancement algorithms. As opposed to the previous three measures, the LAR measures a distance, and therefore it increases as distortion increases. The distance measured is based on the reflection coefficients of the corrupted (or enhanced) speech, $\{\rho_{\hat{x}}(m)\}_{m=1}^{M}$, and those of the clean speech, $\{\rho_x(m)\}_{m=1}^{M}$ (M is the order of the linear prediction analysis. The reflection coefficients are in one-to-one correspondence with the linear prediction coefficients – see 4.2). For a given frame, it is computed as follows:

$$LAR_l = \left| \frac{1}{M} \sum_{m=1}^{M} \left(\log \frac{1+\rho_x(m)}{1-\rho_x(m)} - \log \frac{1+\rho_{\hat{x}}(m)}{1-\rho_{\hat{x}}(m)} \right)^2 \right|^{\frac{1}{2}} \tag{4.44}$$

This measure can be seen as a way to estimate efficiently the differences between the logarithms of the spectra of the clean and the corrupted speech signals, (the power spectra are directly related to the reflection coefficients). The LAR measure has been used for testing the

performance of speech enhancement strategies (for example [20, 40]), or of vocoders in [52].

The LAR is also a better indicator than the overall SNR, as it is better correlated with expected opinion scores [45]. In [45], it is observed that the LAR scores are more correlated to subjective listening test results than the Itakura-Saito measure, which is another popular spectral distortion measure based on the same linear prediction principle. In our experience, the LAR distance is much more focused on the signal's intelligibility and naturalness than on the background residual noise, This observation is based on the large amount of informal listening made during the preparation of this thesis. It is in accordance with the results reported in [39], where several objective measures are assessed (including the PMF, the PSQM, the Itakura-Saito distance, the log-likelihood ratio, the ASSNR and the weighted spectral slope distance). [39] reports that the LAR has the highest correlation with speech naturalness, and even with overall subjective ratings.

4.4.5 Weighted Spectral Slope distance

The Weighted Spectral Slope distance (WSS), introduced in [30], is a spectral distortion measure which is based on a perceptual model, for the original and enhanced signals are compared with respect to a *critical band* segmentation of frequencies. A critical band, relative to a given center frequency f_c, is a bandwidth which is formed by all the frequencies around f_c that are resolved by the same part of the basilar membrane[1] in the human ear.

As a summary of this method, the signals' short-time spectra are first analyzed in a critical-bands scale. For each signal, a set of *spectral slopes*, defined as the absolute difference between the energies contained in two frequency bands, is computed. For a given short-time spectra (typically computed on frames), the spectral slopes of the original and the enhanced signals are then compared, taking into account the fact that each critical band is weighted in order to give all bands the same *perceptual* weight. Then, an average score is computed as an average over all the frames of the signals to be compared.

Like the LAR distance, the WSS is also recommended in [21]. It has been fairly widely used to assess the quality of speech (for example in [20, 25]). Again, it is said to be usually well correlated to subjective opinion scores [25, 30, 45], and at least much more than the overall SNR. It is difficult to interpret concretely the indication given by a high WSS score if no other indicator is present. Spectral distortion measures mostly indicate to what extent the spectrum of the

[1]The basilar membrane is one of the main hearing/sensing organs of the human ear. The incoming sound waves are geographically dispersed on the membrane depending on their frequency.

resulting speech signal was distorted in the enhancement process, with respect to the original, clean signal. But this does not necessarily mean that the intelligibility is penalized. In the context of the experiments conducted in the next chapter, informal listening tests yielded the conclusion that the WSS distance is mainly an indicator of perceptible spectral artefacts within the resulting speech. Specifically, if the naturalness of the speech is affected, then the WSS tends to be larger. Again, these findings agree with those of [26], which we already mentioned while describing the ASSNR measure. This time, the WSS measure is much more correlated to the "SIG" score than to the "BAK" score.

In all our comparisons, we directly use the LAR and WSS implementations available from the website of the authors of [21], that is `http://cslr.colorado.edu/software/rspl.html`.

Chapter 5

Applying Particle Filtering to Speech Enhancement: existing methods

We will now introduce some particle filtering algorithms for speech denoising. In trying to apply PFs to this problem, we are hopeful that they will provide a worthy alternative to existing algorithms, which all have some advantages and drawbacks. We know that particle filters are theoretically able to approximate arbitrarily well the exact posterior distribution corresponding to a given state-space formulation of a problem. Thus, we are led to believe that provided a good model is used to formulate the speech enhancement problem, then PFs have the potential to offer very good results.

In this chapter, we present some existing and previously published approaches. In section 5.1, we review some existing PF algorithms for enhancement of speech corrupted by white Gaussian noise, as described in [8,12,16,50]. In the next section, we discuss two existing cases where the noise is not white-Gaussian: one where the noise standard deviation is time-varying [50], and one where the observation noise is modelled as an α-stable symmetric noise [34].

Even though these algorithms have been previously published, so far no emphasis has been made on a proper analysis of the quality of resulting speech. Throughout the chapter, one general novelty is the effort made for speech quality assessment, as simulation results are always detailed using the quality measures described in 4.4. None of [8,12,16,50] do this type of assessment or comparison – only the overall SNR is used.

5.1 Particle filtering algorithms for speech enhancement: the AWGN case

5.1.1 Problem setting

In this chapter, the model for speech production is the one presented in 4.2, specifically with equation (4.6). The observation model (additive noise) is represented by equation (4.7). As in section 4.3.3, where we discussed Kalman filter-based methods, the autoregressive coefficients and the excitation noise variance are allowed to vary in time, but this time we do not "frame" the data, and we consider that they can vary from one time instant to the next.

The summary for the model used is the following:

$$\mathbf{x_{2}}_k = \mathbf{A}_k \mathbf{x_{2}}_{k-1} + \mathbf{G}_k \mathbf{w}_k \tag{5.1}$$

$$z_k = \mathbf{C}\mathbf{x_{2}}_k + \sigma_{v,k} v_k \tag{5.2}$$

where:

$$\begin{cases} \mathbf{x_{2}}_k = [x_k \;\; x_{k-1} \;\; \dots \;\; x_{k-M+1}]^T \\ \mathbf{A}_k = \begin{bmatrix} a_{1,k} & a_{2,k} & \dots & a_{M-1,k} & a_{M,k} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_k^T & \\ \mathbf{I}_{M-1} & \mathbf{0}_{M-1\times 1} \end{bmatrix} \\ \mathbf{G}_k = [\sigma_{\mathbf{w},k} \;\; 0 \;\; \dots \;\; 0]^T \\ \mathbf{C} = [1 \;\; 0 \;\; \dots \;\; 0] \\ \mathbf{w}_k \sim \mathcal{N}(0,1) \quad \text{and} \quad v_k \sim \mathcal{N}(0,1) \end{cases}$$

The white Gaussian noise $\mathbf{w}_k$ is not a vector, but we keep the bold font to distinguish it from the weights of the particle filter.

We want to allow the autoregressive coefficients and the standard deviations $\sigma_{\mathbf{w},k}$ and $\sigma_{v,k}$ to vary, as they would in the speech production model of section 4.2. This makes the system formed by equations (5.1) and (5.2) a *Conditionally linear-Gaussian* system (see section 3.3.1), and it is a particular case of the broader system formed by (3.21) and (3.22).

To solve this problem, as stated in 3.3.1, whether a RBPF or a simple PF is chosen to be used, we must first define a set of variables describing the time-varying parameters, say $\mathbf{x_{1}}_k$, and then to apply the algorithm on the whole state $\mathbf{x}_k = \{\mathbf{x_{1}}_k; \mathbf{x_{2}}_k\}$. Clearly here, $\mathbf{x_{1}}_k$ will be in relation to $\mathbf{A}_k$, $\sigma_{\mathbf{w},k}$ and $\sigma_{v,k}$. For generality we still allow $\sigma_{v,k}$ to vary for the derivation of the algorithm, but in this section we will only observe results in the case where $\sigma_{v,k}$ is constant and predetermined (i.e., it is not contained in $\mathbf{x_{1}}_k$). In subsection 5.2.1 below, we focus on the case where it is possibly time-varying and part of the state to estimate.

There are different ways to define $\mathbf{x_{1}}_k$ from $\mathbf{A}_k$, $\sigma_{\mathbf{w},k}$ and $\sigma_{v,k}$. To ensure that the standard deviations are never negative, [12,34,50] model the log-variances, $\log\left(\sigma^2_{\mathbf{w},k}\right)$ and $\log\left(\sigma^2_{v,k}\right)$, as quantities evolving in a Gaussian random walk fashion[1]. We then have:

$$\log\left(\sigma^2_{\mathbf{w},k}\right) \sim \mathcal{N}\left(\log\left(\sigma^2_{\mathbf{w},k-1}\right); \phi^2_{\mathbf{w}}\right) \tag{5.3}$$

$$\log\left(\sigma^2_{v,k}\right) \sim \mathcal{N}\left(\log\left(\sigma^2_{v,k-1}\right); \phi^2_{v}\right) \tag{5.4}$$

For clarity we introduce the notation $\mathcal{L}_{\mathbf{w},k} \triangleq \log\left(\sigma^2_{\mathbf{w},k}\right)$ and $\mathcal{L}_{v,k} \triangleq \log\left(\sigma^2_{v,k}\right)$. In the problem setting, $\phi_{\mathbf{w}}$ and ϕ_v are determined before the algorithm begins.

Considering now the case of the autoregressive coefficient vector $\mathbf{a}_k$ which forms the first row of $\mathbf{A}_k$, we point that there are two major ways appearing in the literature to simulate their evolution. [50] directly uses a Gaussian random walk on each of the AR coefficients, but still constrains it in such a way that the resulting AR vector corresponds to a filter whose poles lie within the unit circle (thereby ensuring stability). In this scheme, the substate $\mathbf{x_{1}}_k$ contains the vector $\mathbf{a}_k$ with no necessary transformation. In [12], it is stated that a constrained Gaussian random-walk on the partial correlation coefficients, rather than the AR coefficients $\mathbf{a}_k$, yields better results (although the article only uses the overall SNR for comparison, and shows relatively modest improvements and for a limited set of test data). The one-to-one relation is then defined by the Levinson-Durbin algorithm, and the substate $\mathbf{x_{1}}_k$ contains a vector of partial correlation coefficients ρ_k.

In this chapter, we choose to use a constrained Gaussian random walk on the autoregressive coefficients, as in [50]. Each time, we draw a new AR vector normally around the previous one, and we compute the partial correlation coefficients (which is equivalent to, but much cheaper than computing all the complex roots, and verifying that they are in the unit circle). If they

[1] In a Gaussian random walk, the current value observed is drawn normally around the previous value. The resulting quantity drifts randomly in time, but it does so with a certain form of "continuity". This necessitates a hyperparameter, fixed and predetermined: the variance of the Gaussian random walk.

are all absolutely inferior to 1, then we accept the new draw. If not, then we simply redraw another AR vector. If we do it this way, then the computational load is not much heavier than if a Gaussian random walk on the partial correlation coefficients is used, since it is still necessary to convert these back to autoregressive coefficients.

5.1.2 On possible PF algorithms

We only mention briefly here some possible PF algorithms (as opposed to RBPF implementations).

First, if we use the transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ as the importance function, our experience is that the algorithm performs relatively poorly, unless a very large number of particles is used. For example, it may enhance correctly and very well a portion of speech, but after a pause it can have a difficult time "catching up" with the next utterance. We can witness this phenomenon as it happens, either by using a form of online SNR computations, or by simply listening to the resulting enhanced speech. Of course, the overall SNR, the ASSNR and the PESQ (see section 4.4) are then strongly penalized, as well as any random listener's perception of the enhanced speech.

In [8, 16], a better (but still subpotimal) importance function is used, since it incorporates the current measurement. We refer the reader to these references for more details. We have experimented with this scheme, and in our resulting implementation the algorithm is indeed found to be more robust and accurate than the PF using the mere transition density, however we still find that unless quite a large number of particles is used, this robustness is not sufficient for some situations. For example, when a male and a female interact in a noisy speech recording, we find that the algorithm sometimes has a difficult time tracking the first few milliseconds of the utterances of each speaker, resulting in a distortion that is not unsignificant to the listener. Moreover, if a very large number of particles is chosen, then we were only able to partially reduce this problem using this method.

In addition, for both the PF algorithms mentioned above, in a few cases and for a wide range of the number of particles, we have found that the PF fails (in the sense that at a given point, all weights have collapsed to zero), although this happened less often for the one presented in [8,16].

For these reasons, we have chosen not to include these PF implementations in the rest of the chapter. Because of the nature of the artifacts affecting directly the objective measures used here, it would have been difficult to compare them with other algorithms. In addition, they sometimes failed during our experiments, whereas the ones presented next only very rarely

did.

5.1.3 Development of a RBPF algorithm

We can use directly the results from section 3.3.1, applying Algorithm 5 with $\mathbf{B}_k = \mathbf{0}$, $\mathbf{u}_k = \mathbf{0}$, $\mathbf{C}_k = \mathbf{C}$, $\mathbf{D}_k = \mathbf{0}$, and $\mathbf{H}_k = \sigma_{v,k}$. We obtain Algorithm 11 shown below, which is the same basis algorithm that is presented in [50], except that we omit here the MCMC move step (see 2.3). We do so for computational reasons: the RBPF is already a heavy algorithm, and ultimately our goal would be to keep the procedure light enough to be implemented in real-time. MCMC moves are very expensive, and do not always improve the results (see, for example, [49]), and sometimes only improves them slightly.

In Algorithm 11, we must define the variances $\phi^2_{\mathbf{a}}$, $\phi^2_{\mathbf{w}}$ and ϕ^2_v for the Gaussian random walk models for the AR vector $\mathbf{a}_k$, the excitation noise $\mathbf{w}_k$ and the observation noise v_k prior to the execution. The values chosen for these parameters are given in 5.1.4. Note that we apply resampling at each step, simply because we have tried otherwise to only resample if the effective sample size is smaller than a fraction of the number of particles N, but in this case we have not found a significant advantage in doing so. In fact, on average, for the number of particles used throughout our simulations, we have witnessed a slight decrease in overall performance.

To improve the estimates, the smoothing techniques explained in 2.3.3 were implemented. For the reasons given in the latter section, we decided to only employ fixed-lag smoothing. Simulation results will be given in 5.1.4.

5.1.4 Simulation results

We show here simulation results for the basic RBPF for the AWGN case of Algorithm 11, with and without fixed-lag smoothing.

Experimental conditions

In this work, we are focusing on speech signals sampled at 8 kHz, normalized to $[-1, +1]$. More information about the speech signals used are given below. The parameters of the simulations are the following:

- $N = 500$. This was found to be a value which guarantees a good robustness over the different speech signals tested.
- The order of the autoregression M is set to 6.

Algorithm 11 A basic RBPF algorithm for speech enhancement

1. **Define the appropriate variances $\phi_{\mathbf{a}}^2$, $\phi_{\mathbf{w}}^2$ and ϕ_v^2, and choose the number of particles N**

2. **Define and initialize the value of $\{\mathbf{x}_{10,i}\,;\,\mathbf{x}_{20,i}\,;\,\mathbf{K}_{0,i}\}_{i=1}^N$, where $\mathbf{K}$ is the covariance matrix of $\mathbf{x}_2$. Draw the initial values of the N AR vectors so that they correspond to a stable filter.**

3. **For every k, update the set $\{\mathbf{x}_{1k-1,i}\,;\,\mathbf{x}_{2k-1,i}\,;\,\mathbf{K}_{k-1,i}\}_{i=1}^N$ as follows:**

○ For every $i \in \{1, 2, \ldots, N\}$:

- Draw $\mathbf{x}_{1k,i}$ as:

$$\begin{aligned}
\mathbf{a}_{k,i} &\sim \mathcal{N}(\mathbf{a}_{k-1,i}; \phi_{\mathbf{a}}^2 \times \mathbf{I}), \text{ accept the draw if it is a stable filter, otherwise draw another one} \\
\mathcal{L}_{\mathbf{w},k,i} &\sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w},k-1,i} | \phi_{\mathbf{w}}^2\right) \\
\mathcal{L}_{v,k,i} &\sim \mathcal{N}\left(\mathcal{L}_{v,k-1,i} | \phi_v^2\right)
\end{aligned}$$

- From $\mathbf{a}_{k,i}$, $\mathcal{L}_{\mathbf{w},k,i}$ and $\mathcal{L}_{v,k,i}$, obtain the corresponding $\mathbf{A}_{k,i}$, $\mathbf{G}_{k,i}$, and $\sigma_{v,k,i}$.
- Compute the following:

$$\begin{aligned}
\mathbf{K}_{k|k-1,i} &= \mathbf{G}_{k,i}{\mathbf{G}_{k,i}}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1,i}{\mathbf{A}_{k,i}}^T \\
T_{k,i} &= \sigma_{v,k,i}^2 + \mathbf{C}\mathbf{K}_{k|k-1,i}\mathbf{C}^T \\
\mathbf{x}_{2k|k-1,i} &= \mathbf{A}_{k,i}\mathbf{x}_{2k-1,i} \\
y_{k,i} &= \mathbf{C}\mathbf{x}_{2k|k-1,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(z_k | y_{k,i}, T_{k,i}) \\
\mathbf{J}_{k,i} &= \mathbf{K}_{k|k-1,i}\mathbf{C}^T T_{k,i}^{-1} \\
\mathbf{x}_{2k,i} &= \mathbf{x}_{2k|k-1,i} + \mathbf{J}_{k,i}(z_k - y_{k,i}) \\
\mathbf{K}_{k,i} &= (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C})\mathbf{K}_{k|k-1,i}
\end{aligned}$$

○ Compute the normalizing factor $\sum_{i=1}^N \widetilde{w}_{k,i}$, and normalize the weights (obtain $\{w_{k,i}\}_{i=1}^N$)

○ Using the weights, resample the set $\{\mathbf{x}_{1k,i}\,;\,\mathbf{x}_{2k,i}\,;\,\mathbf{K}_{k,i}\}_{i=1}^N$

○ Obtain the enhanced speech estimate $\hat{x}_k = \mathbf{C}\frac{1}{N}\sum_{i=1}^N \mathbf{x}_{2k,i}$

- $\phi_{\mathbf{a}}^2$ and $\phi_{\mathbf{w}}^2$ are both set to 5×10^{-3}, according to the results of [50]. Other values were tested, but any value in the 10^{-3} range seems to yield about the same result. Below and above that range, the performance begins to decrease.
- The additive white Gaussian noise's standard deviation ϕ_v is set to the following values for each experiment: $\{0.025\,;\ 0.05\,;\ 0.075\,;\ 0.1\,;\ 0.125\,;\ 0.15\}$, and this value is given and considered known to the RBPF, and thus there is no need to update $\mathcal{L}_{v,k}$ in Algorithm 11 (in section 5.2.1 we investigate the case where this value is a priori unknown and continuously estimated). The AWGN was generated by a computer, and for every simulation, unless otherwise stated a different sequence of random numbers was obtained.

The objective measures used are the ones described in 4.4, which are the overall SNR (OSNR), the average segmental SNR (ASSNR), the PESQ score, the log-area ratio (LAR), and the weighted spectral slope distance (WSS). For clarity, a rough summary of the observations made in 4.4 to describe these measures now follows. The OSNR is merely an indicator that some enhancement (i.e. noise reduction) is taking place. A very high value may or may not indicate a very high speech quality, but a low value likely indicates a low speech quality. The ASSNR is majoritarily correlated with the background noise intrusiveness, or with interspeech residual noise. Next, the PESQ is well correlated with the overall speech quality. Finally, the LAR and WSS are mostly correlated with the speech naturalness.

The speech signals used are whole sentences, lasting a few seconds. Initially, we had begun testing individually a male speaker and then a female speaker (in separate files), but then we decided to use a single file with both a male and a female alternating. The reason for this choice is that we have found that even though some algorithms would perform fairly well in the individual case, they would struggle (and sometimes fail) when two very different voices are present in the file (see 5.1.2). For this reason, the subsequent experiments were chosen to be conducted only on speech files with mixed speakers (In section 6.4.2, where the RBPF algorithms are compared to other algorithms, different speakers and noise levels are used). Both the clean sound files for the individual male and the female speakers can be retrived from Dr. E. Wan's demonstration webpage: `http://cslu.ece.ogi.edu/nsel/demos/index.html`. The sentence spoken by the male speaker is: "*primitive tribes have an upbeat attitude*", and the one said by the female speaker is: "*A lathe is a big tool. Grab every dish of sugar*". The clean speech signal used is a mixture of some sections of each sentence.

In Table 5.1, experiment results are shown. The scores reported are an average over 20 experiments. For each of the 20 experiments and for each standard deviation σ_v, we save the noise sequence generated, in order to use it subsequently for comparisons. As stated in the introduction, in this table, and in all the tables containing such results, a simple color code

will be used to help the visualization of the tables. Basically, among results from the same objective measure, the best value will be in blue, and the worst in red. If there are more than two algorithms being compared, then more colors will be used, ranging gradually from blue to red as in the last six words of this sentence.

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.65**	**2.29**	**4.19**	**6.63**	**10.24**	**16.22**
	ASSNR	-1.56	-0.78	0.31	1.91	4.64	9.76
	PESQ	1.42	1.51	1.63	1.74	1.97	2.39
	LAR	8.80	8.54	8.08	7.60	6.66	5.22
	WSS	61.53	60.29	53.29	49.71	38.91	26.47
RBPF (Alg. 11) no smoothing	OSNR	6.15	7.23	8.63	10.12	12.78	17.64
	ASSNR	1.79	2.68	3.84	5.23	7.47	11.77
	PESQ	1.62	1.73	1.91	2.07	2.29	2.76
	LAR	7.37	6.99	6.53	6.07	5.41	4.26
	WSS	62.90	61.06	55.12	50.15	38.82	26.35
RBPF (Alg. 11) fixed-lag smoothing with $L = 8$	OSNR	6.81	7.94	9.35	11.01	13.44	17.92
	ASSNR	2.38	3.47	4.50	6.02	8.13	12.28
	PESQ	1.70	1.79	2.03	2.24	2.42	2.88
	LAR	7.18	6.85	6.31	5.81	5.18	4.10
	WSS	62.96	60.91	54.88	49.39	38.94	26.16

Table 5.1: Comparison of different RBPF algorithms for speech enhancement
The additive noise is white and Gaussian with variance σ_v^2. Each entry in the table was obtained by computing the average result of 20 runs. In bold is the average input SNR (over the 20 noisy signals).

Comments on the results

First, we mention that, as expected with RBPF algorithms, the robustness of the algorithm is clear, as observed from the low variability of the results and the very low likelihood of divergence over the 20 experiments conducted.

Comparing the performance of the fixed-lag smoothed RBPF and the non-smoothed one, we find that the benefits of smoothing are quite significant according to all of the quality measures, except for the WSS measure, for which the difference is minimal. Listening tests clearly confirm these findings. We conjecture that a significantly better score in more than one of the metrics is a good basis for comparisons.

Overall, the quality of the enhanced speech suffers from one drawback: the residual noise

$x_k - \hat{x}_k$ could be roughly described as a white Gaussian noise with a standard deviation being modulated by the amplitude of the speech. This is consistent with our observation that during speech pauses, in the enhanced signal there is only a very small amount of noise remaining, even for larger initial observation noises. Unfortunately, during speech this noise is not completely masked, especially at low SNR.

As an aside, we have conducted another experiment. Given one *single* noisy signal for each standard deviation, we ran each algorithm 10 times, and then computed the average of their output. The results are reported in table 5.2. We found that the resulting average signal has a better quality than each of the individually enhanced signals (whose average scores are close to those reported in Table 5.1). Although these results are theoretically interesting, they have very limited practical value: since PF algorithms are by nature very heavy, it is not appealing to impose ten times more computations to improve the results.

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.69**	**2.24**	**4.16**	**6.71**	**10.25**	**16.19**
	ASSNR	-1.56	-0.81	0.33	1.95	4.62	9.69
	PESQ	1.43	1.49	1.61	1.75	1.99	2.41
	LAR	8.77	8.46	8.14	7.59	6.69	5.20
	WSS	63.06	62.00	54.92	48.01	39.29	26.53
RBPF (Alg. 11) no smoothing	OSNR	6.24	7.34	8.72	10.33	13.07	17.91
	ASSNR	1.87	2.75	3.93	5.35	7.61	11.85
	PESQ	1.68	1.79	1.98	2.12	2.34	2.79
	LAR	7.34	6.92	6.47	6.00	5.34	4.22
	WSS	62.48	60.57	54.85	49.94	38.50	26.16
RBPF (Alg. 11) fixed-lag smoothing with $L = 8$	OSNR	6.95	8.08	9.49	11.13	13.73	18.35
	ASSNR	2.48	3.57	4.62	6.13	8.30	12.42
	PESQ	1.76	1.86	2.11	2.26	2.46	2.92
	LAR	7.10	6.78	6.19	5.72	5.05	4.02
	WSS	62.65	60.23	54.32	49.04	38.41	25.70

Table 5.2: Comparisons of RBPF algorithms using average signals

The additive noise is white and Gaussian with variance σ_v^2. Each entry in the table was obtained by first running each algorithm 10 times on a single *noisy signal, and then taking the average of the 10 resulting signals. In bold is the input SNR (of the single noisy test signal).*

5.2 RBPF algorithms for speech enhancement: non-white noise case

In the previous section, we have conducted some experiments with RBPF-based speech enhancement algorithms in simple experimental conditions, where the additive noise was white and Gaussian. In this section, we consider the case in which the additive noise is essentially a white and Gaussian noise but with a time-varying standard deviation – in other words, we consider that we have an "amplitude-modulated" white Gaussian noise. Next we also briefly recapituale the work of [35], where an interesting extension of the RBPF-based enhancement algorithm to the case where the observation noise is drawn from a symmetric α-stable distribution. It is shown in [35] that such an α-stable noise can be used to model certain types of real-world noises in a more accurate way than white Gaussian noises.

5.2.1 Gaussian noise with time-varying standard deviation

In this section, we simply apply the RBPF algorithm for speech enhancement shown in Algorithm 11 to the case where the observation noise variance $\sigma_{v,k}^2$ is unknown, and is thus part of the state to estimate.

In this case, Algorithm 11 can be directly applied, provided the user specifies the variance ϕ_v^2 of the Gaussian random walk on $\sigma_{v,k}^2$. The choice of ϕ_v^2 depends on the a priori knowledge for the noise. For example, if the real additive noise is unknown, but known to be stationary, then ϕ_v^2 can be made very small (although it cannot be set to 0 for the proper operation of the RBPF). If in addition we do not have very much information about the possible range of ϕ_v^2, then we can draw the initial particles from a heavy-tailed prior. In the case where we believe that the variance of the noise is very volatile, then we may want to either increase ϕ_v^2 and use more particles, or think about implementing a RBPF algorithm using a different importance function (see the generic RBPF algorithm for conditionally linear-Gaussian problems, shown in Algorithm 5) or a different model than a Gaussian random walk. In fact, in the general case where the probability density function of the observation noise is known in advance, then Algorithm 11 can be accomodated to any situation via small modifications. We present an example of such a situation in the next section.

We have conducted one first set of experiments in the same conditions as those presented previously in 5.1.4, except that this time the noise variance is unknown. We have chosen not to report the results, since on average they are almost identical in every way to those reported in 5.1.4, where the fixed noise variance was predetermined.

A more interesting, second test was performed on a sentence of length $T \simeq 2$ seconds. The noise added is defined as follows:

$$v_k = K \left| \sin\left(\frac{2k\pi}{T}\right) \right| \times \nu_k \tag{5.5}$$

with $\nu_k \sim \mathcal{N}(0,1)$ and K being a constant. For example, with $K = 0.1$, the additive noise is shown on Figure 5.1. In Figure 5.2, we show the superimposed clean speech and the additive noise.

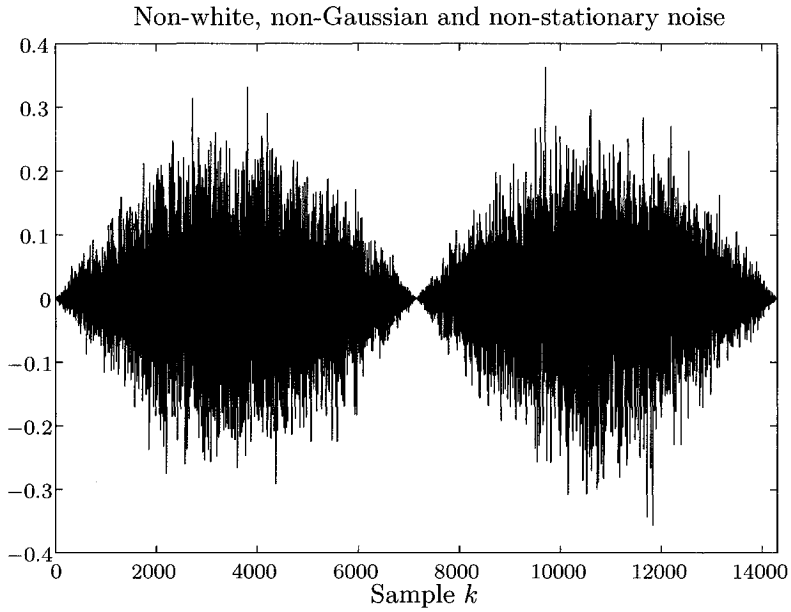


Figure 5.1: A non stationary, non Gaussian, and non white additive noise example

For the simulations, we directly apply Algorithm 11 where we set $\phi_v^2 = 10^{-3}$ and we uniformly draw initial variances from $[0, 0.1]$ in order to reflect a poor a priori knowledge. We obtain the following results:

These results show that the algorithm is able to track the statistics of the environment noise. All of the quality measures show some improvement, which are confirmed by listening tests.

5.2.2 Symmetric α-stable environment noise

In this section, we briefly present the main idea behind the algorithm proposed by [35]. In this paper, the authors are considering the enhancement of signals corrupted by vynil or gramophone-type noise. Such a noise can be typically described as a form of superimposition of a quasi-stationary white Gaussian noise, and of a sequence of sudden (and ample)

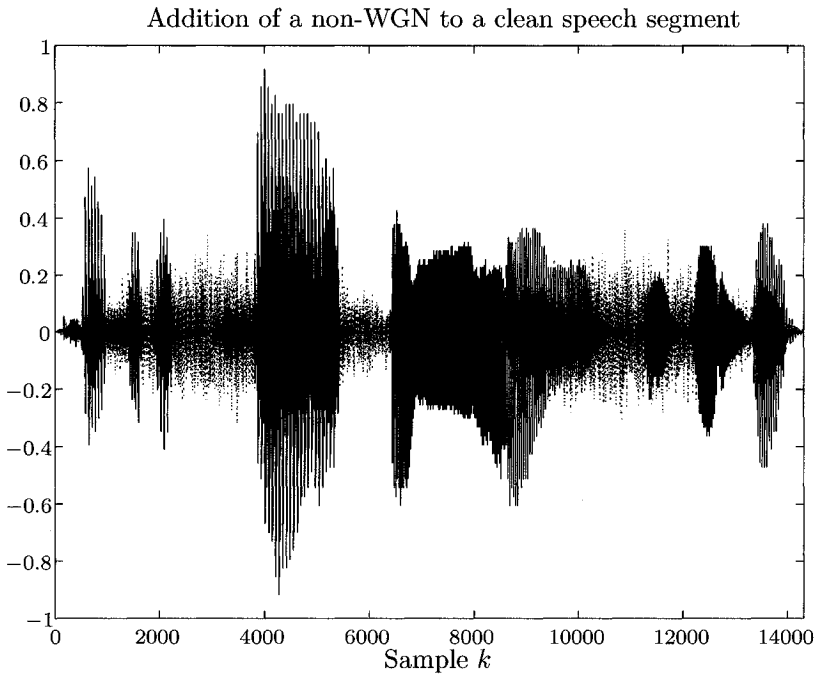


Figure 5.2: Addition of a non white-Gaussian noise to a clean speech segment.

clicks. In addition, a satisfactory model for this type of noise consists of a sequence of draws from a symmetric α-stable distribution.

Let $\mathcal{S}(\alpha, \beta)$ denote a standard α-stable distribution (with unit scale and zero location). The parameters α and β determine the "heavy-tailness" and the symmetry of the distribution. Although in general, the density function for α-stable random variables does not admit a closed-form expression, there still exists a method to simulate them (see [5]). In addition, it is possible to express a symmetric α-stable random variable $v \sim \mathcal{S}(\alpha, 0)$ as [35]:

$$v = \sqrt{\gamma}\mu \tag{5.6}$$

where $\gamma \sim \mathcal{S}(\frac{\alpha}{2}, 0)$ and $\mu \sim \mathcal{N}(0, 1)$.

These considerations pave the way for the application of a RBPF. Suppose that the additive noise v_k is modeled as a sequence of samples from $\mathcal{S}(\alpha, 0)$, up to a scaling parameter λ_k. We can write $v_k = \lambda_k \sqrt{\gamma_k}\mu_k$ where $\gamma_k \sim \mathcal{S}(\frac{\alpha}{2}, 0)$ and $\mu_k \sim \mathcal{N}(0, 1)$. Then, conditioned upon the knowledge of the sequences of λ_k and γ_k, the observation equation $z_k = x_k + \lambda_k \sqrt{\gamma_k}\mu_k$ is linear and Gaussian.

Quality measure	Noisy signal	Enhanced signal
OSNR	6.60	10.92
ASSNR	3.61	6.77
PESQ	1.91	2.13
LAR	6.82	5.92
WSS	42.62	42.02

Table 5.3: RBPF algorithm performance for the observation noise in equation (5.5)
The standard deviation of the noise added varies quite rapidly, as seen in Figure 5.1, and the overall input SNR is 6.60 dB.

To apply the generic RBPF for conditionally linear-Gaussian problems (shown in Algorithm 5) to audio enhancement in these conditions, we can follow the very same steps as for the derivation of the RBPF algorithm for speech enhancement presented in Algorithm 11, but we simply need to include in the vector $\mathbf{x_{1}}_k$ the additional parameters λ_k and γ_k. Initially, several candidate values for α are drawn according to any prior knowledge. These candidates are associated with the remainder of the particles during the initialization process. The value of α is considered fixed, but an artificial evolution is introduced for the proper operation of the PF [35] (we had mentioned the existence of a method for combined fixed parameters and time-varying ones in 3.3.1 – the method used in [35] is the one presented in [8]). Additionally, at each step, γ_k is drawn from $\mathcal{S}(\frac{\alpha}{2}, 0)$, and λ_k may follow a simple Gaussian random walk. The resulting algorithm is shown to perform very well on both artificial and real data.

Chapter 6

Applying Particle Filtering to Speech Enhancement: novel methods

This chapter introduces new particle filtering methods for speech enhancement, and it then also focuses on the comparison of some of these methods to existing denoising algorithms, such as the ones presented in the last chapters. The organization of the chapter is as follows: In section 6.1, we apply the modified RBPF algorithm (presented in its generic form in Algorithm 8) to speech enhancement, and we compare the results obtained to those of the regular RBPF tailored for speech enhancement (Algorithm 11) presented in the previous chapter. Next, in section 6.2, we introduce a technique to deal with colored, autoregressive observation noise. Then, section 6.3 is dedicated to the introduction of two new algorithm extensions to improve the quality of the enhanced speech obtained by applying Algorithm 11. Both are "perceptually motivated", and both are shown, through simulation results, to improve the estimated clean speech. Finally, in 6.4 we compare RBPF algorithms to other speech enhancement algorithms, including (but not limited to) the ones presented in section 4.3 and chapter 5.

Again, as in the previous chapter, we use the quality measures described in 4.4, both for the assessment of each individual algorithm but also for comparisons with other speech enhancement algorithms.

6.1 Application of the modified RBPF to speech enhancement ($\star$)

6.1.1 Derivation of the algorithm from a regular RBPF ($\star$)

The modified RBPF was presented in 3.4. Recall that we had stated that to derive a proper modified RBPF algorithm from a regular RBPF, we must begin by spotting all the steps that can be taken out of the main loop iterating over all i (see 3.4.2). First, we can always remove the computations directly related to the update of the new mean and covariance of $\mathbf{x_2}$, and do these computations after the loop via a single KF. Secondly, in the leftover contents of the loop, we use the fact that $\forall i, \mathbf{K}_{k,i} = \mathbf{K}_k$ in the modified algorithm to remove any computations which do not depend on i anymore, and perform them before the loop. We had also stated that in this second step, the number of operations that can be removed from this main loop is case dependent.

Observing the regular RBPF for speech enhancement shown in Algorithm 11, we find that unfortunately, this case is typically one where only a few lines can be removed. Nevertheless, recall that a single operation removed from the loop amounts to N operations per input sample in the overall algorithm. In the modified RBPF for speech enhancement, shown in Algorithm 12 below, the last three lines, which were in the original RBPF directly related to the KF update, are moved outside of the loop.

Algorithm 12 A basic modified RBPF algorithm for speech enhancement

1. **Define the appropriate variances $\phi_{\mathbf{a}}^2$, $\phi_{\mathbf{w}}^2$ and ϕ_v^2, and choose the number of particles N**

2. **Define and initialize the value of $\{\mathbf{x}_{10,i}\}_{i=1}^N$, and that of $\{\hat{\mathbf{x}}_{20}\,;\,\mathbf{K}_0\}$ where $\mathbf{K}$ is the covariance matrix of $\hat{\mathbf{x}}_2$. Draw the initial values of the N AR vectors so that they correspond to a stable filter.**

3. **For every k, update the set $\{\{\mathbf{x}_{1k-1,i}\}_{i=1}^N\,;\,\hat{\mathbf{x}}_{2k-1}\,;\,\mathbf{K}_{k-1}\}$ as follows:**

○ For every $i \in \{1, 2, \ldots, N\}$:

- Draw $\mathbf{x}_{2k-1,i} \sim \mathcal{N}(\hat{\mathbf{x}}_{2k-1}|\mathbf{K}_{k-1})$
- Draw $\mathbf{x}_{1k,i}$ as:

$$\begin{aligned}
\mathbf{a}_{k,i} &\sim \mathcal{N}(\mathbf{a}_{k-1,i}; \phi_{\mathbf{a}}^2 \times \mathbf{I}), \text{ accept the draw if it is a stable filter, otherwise draw another one} \\
\mathcal{L}_{\mathbf{w},k,i} &\sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w},k-1,i}|\phi_{\mathbf{w}}^2\right) \\
\mathcal{L}_{v,k,i} &\sim \mathcal{N}\left(\mathcal{L}_{v,k-1,i}|\phi_v^2\right)
\end{aligned}$$

- From $\mathbf{a}_{k,i}$, $\mathcal{L}_{\mathbf{w},k,i}$ and $\mathcal{L}_{v,k,i}$, obtain the corresponding $\mathbf{A}_{k,i}$, $\mathbf{G}_{k,i}$, and $\sigma_{v,k,i}$.
- Compute the following:

$$\begin{aligned}
\widetilde{\mathbf{K}}_{k,i} &= \mathbf{G}_{k,i}{\mathbf{G}_{k,i}}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1}{\mathbf{A}_{k,i}}^T \\
T_{k,i} &= \sigma_{v,k,i}^2 + \mathbf{C}\widetilde{\mathbf{K}}_{k,i}\mathbf{C}^T \\
\mathbf{x}_{2k|k-1,i} &= \mathbf{A}_{k,i}\mathbf{x}_{2k-1,i} \\
y_{k,i} &= \mathbf{C}\mathbf{x}_{2k|k-1,i} \\
\widetilde{w}_{k,i} &= \mathcal{N}(z_k|y_{k,i}, T_{k,i})
\end{aligned}$$

○ Compute the normalizing factor $\sum_{i=1}^N \widetilde{w}_{k,i}$, and normalize the weights (obtain $\{w_{k,i}\}_{i=1}^N$)

○ Using the weights, resample the set $\{\mathbf{x}_{1k,i}\}_{i=1}^N$

○ Obtain an estimate for $\mathbf{x}_{1k}$ as $\hat{\mathbf{x}}_{1k} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_{1k,i}$, and obtain the corresponding $\widehat{\mathbf{A}}_k$, $\widehat{\mathbf{G}}_k$, and $\widehat{\sigma}_{v,k}$.

○ Perform the following KF step:

$$\begin{aligned}
\mathbf{K}_{k|k-1} &= \widehat{\mathbf{G}}_k{\widehat{\mathbf{G}}_k}^T + \widehat{\mathbf{A}}_k\mathbf{K}_{k-1}{\widehat{\mathbf{A}}_k}^T \\
T_k &= \widehat{\sigma}_{v,k}^2 + \mathbf{C}\mathbf{K}_{k|k-1}\mathbf{C}^T \\
\hat{\mathbf{x}}_{2k|k-1} &= \hat{\mathbf{A}}_k\hat{\mathbf{x}}_{2k-1} \\
y_k &= \mathbf{C}\hat{\mathbf{x}}_{2k|k-1} \\
\mathbf{J}_k &= \mathbf{K}_{k|k-1}\mathbf{C}^T T_k^{-1} \\
\hat{x}_{2k} &= \hat{\mathbf{x}}_{2k|k-1} + \mathbf{J}_k(z_k - y_k) \\
\mathbf{K}_k &= (\mathbf{I} - \mathbf{J}_k\mathbf{C})\mathbf{K}_{k|k-1}
\end{aligned}$$

○ Obtain the enhanced speech estimate $\hat{x}_k = \mathbf{C}\hat{x}_{2k}$

6.1.2 Simulation results ($\star$)

Experimental conditions

The experimental conditions are exactly the same as those used to conduct experiments to test the regular RBPF for speech enhancement, and they are described in details in section 5.1.4. In Table 6.1, simulation results are shown. To facilitate the comparison with the regular RBPF, we repeat the experiment results already reported in Table 5.1 from the previous chapter. Note that because its nature, it is not possible to apply fixed-lag smoothing to the modified RBPF.

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.65**	**2.29**	**4.19**	**6.63**	**10.24**	**16.22**
	ASSNR	-1.56	-0.78	0.31	1.91	4.64	9.76
	PESQ	1.42	1.51	1.63	1.74	1.97	2.39
	LAR	8.80	8.54	8.08	7.60	6.66	5.22
	WSS	61.53	60.29	53.29	49.71	38.91	26.47
RBPF (Alg. 11) no smoothing	OSNR	6.15	7.23	8.63	10.12	12.78	17.64
	ASSNR	1.79	2.68	3.84	5.23	7.47	11.77
	PESQ	1.62	1.73	1.91	2.07	2.29	2.76
	LAR	7.37	6.99	6.53	6.07	5.41	4.26
	WSS	62.90	61.06	55.12	50.15	38.82	26.35
Modified RBPF (Alg. 12)	OSNR	4.74	5.84	7.46	8.84	12.30	16.63
	ASSNR	1.71	2.48	3.74	4.96	7.30	11.48
	PESQ	1.36	1.47	1.77	2.02	2.35	2.76
	LAR	7.27	6.80	6.24	5.85	5.13	4.10
	WSS	63.42	62.72	55.95	50.81	38.97	27.04

Table 6.1: Regular vs. modified RBPF for speech enhancement

The additive noise is white and Gaussian with variance σ_v^2. Each entry in the table was obtained by computing the average result of 20 experiments. In bold is the average input SNR (over the 20 noisy signals).

Comments on the results

First of all, we observe that overall, the modified RBPF closely approximates the regular RBPF (when no smoothing scheme is used), although on average the values for the overall SNR, the ASSNR and the PESQ obtained by the modified RBPF are all inferior to those associated with the regular RBPF, especially for low SNR. Similarly, the WSS scores are all slightly higher for the modified algorithm. The modified algorithm has the benefit of executing faster (in this case, approximately 20 to 25% faster). We already knew from section 3.4 that the modified RBPF could perform similarly to the regular RBPF, but we also noted that an increase of

observation noise tends to deteriorate the results – according to the specific performance measures used then. This tendency is only replicated, in the context of speech enhancement, when the OSNR and PESQ scores are used as metrics. In contrast, the ASSNR, the LAR and the WSS all remain fairly close. This suggests that the amount of residual noise (ASSNR) and the global/perceptual effects on the spectra (LAR/WSS) of the enhanced signals are close to what can be expected from a regular RBPF.

Strangely at first, although the LAR measures for both the regular and the modified algorithms are very close, it appears that the modified algorithm tends to obtain on average a (very) slightly better score – in fact even than that of the smoothed RBPF for some of the experimental conditions. In Figure 6.1, we show a plot of the LAR scores obtained for a regular RBPF and a modified RBPF. The plot shows the LAR scores for each frame, and the graphs are time-aligned. It is interesting to note that the two graphs are almost identical, except around $k = 8000$, where the regular RBPF is sanctioned. More interestingly, this experiment is repeatable, and for many of the 20 simulations made for several input noises, we observe this "glitch". It occurs in the "l" of the words "a lathe" (pronouced from approximately $k = 6000$ to 11000). We fail to explain this phenomenon rationally, but we observe that the LAR is sensitive to this very part of the speech segment.

Nevertheless, overall the difference is minimal enough not to draw any strong conclusion, and we can claim that the speech enhanced using a modified RBPF shares several properties with the equivalent speech enhanced by a regular RBPF.

The modified RBPF does not suffer drastically from the comparison with the regular RBPF, especially at relatively high SNR, however we still observe some increasingly significant discrepancies in some of the quality metrics as the amount of noise increases. During listening tests, it may be concluded that the modified RBPF belongs to the same class of algorithm as the regular RBPF (which may be predicted by the WSS and the LAR), and that the inter-speech residual noise is reduced in a similar way (as the ASSNR seems to indicate), however the degradation in intelligibility is more perceivable as the background noise intensifies (this is likely related to the tendency observed with PESQ scores).

6.2 Time-varying autoregressive environment noise ($\star$)

We propose here a different approach from Algorithm 11 to accomodate colored observation noises. In the next paragraphs, we first mathematically describe the situation, we propose a RBPF algorithm, and then we show some simulation results. The novelty lies in the way the

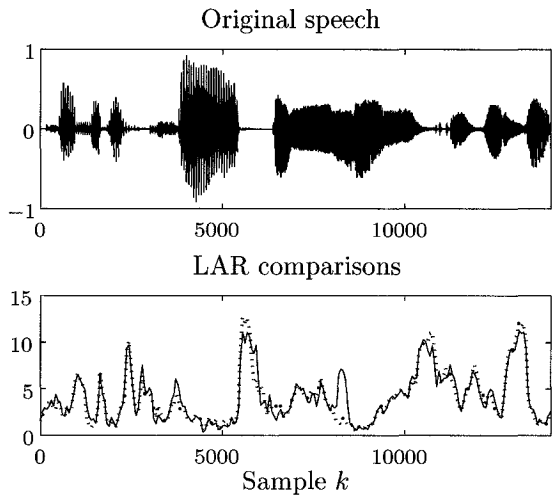


Figure 6.1: Comparisons of LAR scores obtained on each frame: regular vs. modified RBPF
The scores for the regular RBPF are in blue, and those for the modified RBPF are the red dots. These specific LAR scores were obtained for $\sigma_v = 0.025$.

problem is posed and applied – the generic RBPF is not modified here but rather used as a tool.

Problem setting

We keep here the same AR speech production model as before, but this time we also model the environment noise as an autoregressive process. This approach is frequently used to model colored noise (for example, see [32]).

Suppose that $\mathbf{y}_k$ and $\mathbf{v}_k$ respectively denote the autoregressive speech and noise in consideration. Let M be the order of the speech autoregression, and M_v be that of the noise model. The new problem setting then takes the following form:

$$\mathbf{y}_k = \mathbf{A}_{\mathbf{y},k}\mathbf{y}_{k-1} + \mathbf{G}_{\mathbf{y},k}\mathbf{w}_{\mathbf{y},k} \tag{6.1}$$

$$\mathbf{v}_k = \mathbf{A}_{\mathbf{v},k}\mathbf{v}_{k-1} + \mathbf{G}_{\mathbf{v},k}\mathbf{w}_{\mathbf{v},k} \tag{6.2}$$

$$z_k = \mathbf{y}_k[1] + \mathbf{v}_k[1] \tag{6.3}$$

where:

$$
\begin{cases}
\mathbf{y}_k & = [x_k \;\; x_{k-1} \;\; \dots \;\; x_{k-M+1}]^T \\
\mathbf{v}_k & = [v_k \;\; v_{k-1} \;\; \dots \;\; v_{k-M_v+1}]^T \\
\mathbf{A}_{\mathbf{y},k} & = \begin{bmatrix} \mathbf{a}_k^T \\ \mathbf{I}_{M-1} \quad \mathbf{0}_{M-1\times 1} \end{bmatrix} \quad \text{and} \quad \mathbf{A}_{\mathbf{v},k} = \begin{bmatrix} \mathbf{p}_k^T \\ \mathbf{I}_{M_v-1} \quad \mathbf{0}_{M_v-1\times 1} \end{bmatrix} \\
\mathbf{G}_{\mathbf{y},k} & = [\sigma_{\mathbf{w}_\mathbf{y},k} \;\; 0 \;\; \dots \;\; 0]^T \\
\mathbf{G}_{\mathbf{v},k} & = [\sigma_{\mathbf{w}_\mathbf{v},k} \;\; 0 \;\; \dots \;\; 0]^T \\
\mathbf{w}_{\mathbf{y},k} & \sim \mathcal{N}(0,1) \quad \text{and} \quad \mathbf{w}_{\mathbf{v},k} \sim \mathcal{N}(0,1) \\
\mathcal{L}_{\mathbf{w}_\mathbf{y},k} & \triangleq \log\left(\sigma^2_{\mathbf{w}_\mathbf{y},k}\right) \sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w}_\mathbf{y},k-1}|\phi^2_{\mathbf{w}_\mathbf{y}}\right) \\
\mathcal{L}_{\mathbf{w}_\mathbf{v},k} & \triangleq \log\left(\sigma^2_{\mathbf{w}_\mathbf{v},k}\right) \sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w}_\mathbf{v},k-1}|\phi^2_{\mathbf{w}_\mathbf{v}}\right)
\end{cases}
$$

In the model of this very section, our only assumption to distinguish the two AR vectors $\mathbf{a}_k \in \mathbb{R}^M$ and $\mathbf{p}_k \in \mathbb{R}^{M_v}$ is that the AR noise model is slowly time-varying, whereas the speech AR model is more rapidly time-varying. In the model, again we still consider that the AR vectors evolve according to a Gaussian random walk, however to account for this assumption, the variance of the Gaussian random walk for $\mathbf{a}_k$ (denoted again $\phi^2_\mathbf{a}$) is made much larger than that of $\mathbf{p}_k$ (denoted $\phi^2_\mathbf{p}$). Similarly, we choose $\phi_{\mathbf{w}_\mathbf{v}} \ll \phi_{\mathbf{w}_\mathbf{y}}$.

Development of a RBPF algorithm

We would now like to use an algorithm of the form of Algorithm 11. To be able to do so, we simply define:

$$\mathbf{x}_{1k} = \{\mathbf{a}_k \; ; \; \mathbf{p}_k \; ; \; \mathcal{L}_{\mathbf{w}_\mathbf{y},k} \; ; \; \mathcal{L}_{\mathbf{w}_\mathbf{v},k}\} \tag{6.4}$$

$$\mathbf{x}_{2k} = \{\mathbf{y}_k \; ; \; \mathbf{v}_k\} \tag{6.5}$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{A}_{\mathbf{y},k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathbf{v},k} \end{bmatrix} \tag{6.6}$$

$$\mathbf{C} = [1 \;\; 0 \;\; \dots \;\; 0 \;\; 1 \;\; 0 \;\; \dots \;\; 0] \quad \text{where the second "1" is at position } M+1 \tag{6.7}$$

$$\mathbf{G}_k = \begin{bmatrix} \mathbf{G}_{\mathbf{y},k} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\mathbf{v},k} \end{bmatrix} \tag{6.8}$$

$$\mathbf{w}_k = \begin{bmatrix} \mathbf{w}_{\mathbf{y},k} \\ \mathbf{w}_{\mathbf{v},k} \end{bmatrix} \tag{6.9}$$

The resulting model can then be written as:

$$\mathbf{x_{2}}_k = \mathbf{A}_k \mathbf{x_{2}}_{k-1} + \mathbf{G}_k \mathbf{w}_k \tag{6.10}$$

$$z_k = \mathbf{C}\mathbf{x_{2}}_k \tag{6.11}$$

The model as posed is now clearly conditionally linear-Gaussian. Having defined all parameters, we can thus directly apply the generic RBPF algorithm for conditionally linear-Gaussian systems shown in Algorithm 5. We obtain Algorithm 13 shown below, which is formally simpler than Algorithm 11 since equation (6.11) is "noise-free".

Algorithm 13 A basic RBPF algorithm for speech enhancement in colored noise

1. **Define the appropriate variances $\phi^2_{\mathbf{a}}$, $\phi^2_{\mathbf{p}}$, $\phi_{\mathbf{w_y}}$ and $\phi_{\mathbf{w_v}}$, and choose the number of particles N.**

2. **Define and initialize the value of $\{\mathbf{x}_{10,i}\,;\,\mathbf{x}_{20,i}\,;\,\mathbf{K}_{0,i}\}_{i=1}^N$, where $\mathbf{K}$ is the covariance matrix of $\mathbf{x_2}$. Draw the initial values of the N AR vectors $\mathbf{a}_k$ and $\mathbf{p}_k$ so that they correspond to a stable filter.**

3. **For every k, update the set $\{\mathbf{x}_{1k-1,i}\,;\,\mathbf{x}_{2k-1,i}\,;\,\mathbf{K}_{k-1,i}\}_{i=1}^N$ as follows:**

○ For every $i \in \{1, 2, \ldots, N\}$:

- Draw $\mathbf{x}_{1k,i}$ as:

$\mathbf{a}_{k,i} \sim \mathcal{N}(\mathbf{a}_{k-1,i}; \phi^2_{\mathbf{a}} \times \mathbf{I})$, accept the draw if it is a stable filter, otherwise draw another one

$\mathbf{p}_{k,i} \sim \mathcal{N}(\mathbf{p}_{k-1,i}; \phi^2_{\mathbf{p}} \times \mathbf{I})$, accept the draw if it is a stable filter, otherwise draw another one

$\mathcal{L}_{\mathbf{w_y},k,i} \sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w_y},k-1,i} | \phi^2_{\mathbf{w_y}}\right)$

$\mathcal{L}_{\mathbf{w_v},k,i} \sim \mathcal{N}\left(\mathcal{L}_{\mathbf{w_v},k-1,i} | \phi^2_{\mathbf{w_v}}\right)$

- From $\mathbf{a}_{k,i}$, $\mathbf{p}_{k,i}$, $\mathcal{L}_{\mathbf{w},k,i}$ and $\mathcal{L}_{\mathbf{w_v},k,i}$, obtain the corresponding $\mathbf{A}_{k,i}$, and $\mathbf{G}_{k,i}$.
- Compute the following:

$$\mathbf{K}_{k|k-1,i} = \mathbf{G}_{k,i}{\mathbf{G}_{k,i}}^T + \mathbf{A}_{k,i}\mathbf{K}_{k-1,i}{\mathbf{A}_{k,i}}^T$$

$$T_{k,i} = \mathbf{C}\mathbf{K}_{k|k-1,i}\mathbf{C}^T$$

$$\mathbf{x_{2}}_{k|k-1,i} = \mathbf{A}_{k,i}\mathbf{x_{2}}_{k-1,i}$$

$$y_{k,i} = \mathbf{C}\mathbf{x_{2}}_{k|k-1,i}$$

$$\widetilde{w}_{k,i} = \mathcal{N}(z_k | y_{k,i}, T_{k,i})$$

$$\mathbf{J}_{k,i} = \mathbf{K}_{k|k-1,i}\mathbf{C}^T T^{-1}_{k,i}$$

$$\mathbf{x_{2}}_{k,i} = \mathbf{x_{2}}_{k|k-1,i} + \mathbf{J}_{k,i}(z_k - y_{k,i})$$

$$\mathbf{K}_{k,i} = (\mathbf{I} - \mathbf{J}_{k,i}\mathbf{C})\mathbf{K}_{k|k-1,i}$$

○ Compute the normalizing factor $\sum_{i=1}^N \widetilde{w}_{k,i}$, and normalize the weights (obtain $\{w_{k,i}\}_{i=1}^N$)

○ Using the weights, resample the set $\{\mathbf{x}_{1k,i}\,;\,\mathbf{x_{2}}_{k,i}\,;\,\mathbf{K}_{k,i}\}_{i=1}^N$

○ Obtain the enhanced speech estimate $\hat{x}_k$ as the first component of $\frac{1}{N}\sum_{i=1}^N \mathbf{x_{2}}_{k,i}$

Simulation results

In the simulations conducted, a clean speech signal was corrupted with an artificial order 5 autoregressive colored noise. In one simulation, the AR coefficients are made slowly time-varying, and in another, they are fixed. We choose the AR noise coefficients to be very obviously audibly colored. They are initialized as $\mathbf{p}_0 = [0.2 \quad -0.4 \quad 0.2 \quad -0.1 \quad 0.7]^T$ (and they remain at this value in the second simulation). The original clean signal is different from the one used in the previous section.

We only report here simulation results where the order $M_v = 5$ is known[1]. The vector $\mathbf{p}_0$ and the variance $\sigma^2_{\mathbf{w_v},0}$ are initialized from an estimation conducted using the Yule-Walker equations (see 4.3.3) over a small initial frame, where it is known that no speech is present, and particles are drawn around the obtained value.

For the two simulations, the variances for the Gaussian random walk directing the evolution of $\mathbf{p}_k$ and $\mathbf{w}_{\mathbf{v},k}$ are set to a tenth of the corresponding values used for $\mathbf{a}_k$ and $\mathbf{w}_{\mathbf{y},k}$. The results for the first simulation (where the true AR noise coefficients are slowly time-varying, according to a Gaussian random walk with variance 10^{-3}) are shown in Table 6.2, and those for the stationary AR noise are shown in Table 6.3.

Quality measure	Noisy signal	Enhanced signal
OSNR	4.48	10.71
ASSNR	0.06	5.79
PESQ	2.60	2.61
LAR	6.14	4.90
WSS	49.70	35.42

Table 6.2: RBPF algorithm performance for non-stationary colored observation noise
In the RBPF, the observation noise is modelled as a slowly time-varying autoregressive signal of order $M_v = 5$. In the simulation, only the order M_v is a priori known to the algorithm. An artificial, slightly non-stationary noise is added to the clean speech signal such that the overall input SNR is 4.48 dB.

We observe that the results, in terms of all of the quality measures, are quite significantly improved. The exception is the PESQ score, which is only very slightly improved. Neverthe-

[1]We have tried to run the same simulations with a different value for M_v than the true one. We have basically found that if M_v is too small, then there will possibly be substantial residual noise. On the other hand, if M_v is larger than the true value, the enhanced speech appears to be of the same quality as that obtained with the ideal, true value. The only drawback is that the algorithm is heavier since the state has a larger dimension.

Quality measure	Noisy signal	Enhanced signal
OSNR	4.30	10.82
ASSNR	-0.54	5.92
PESQ	2.70	2.72
LAR	6.30	4.61
WSS	50.29	38.66

Table 6.3: RBPF algorithm performance for stationary colored observation noise
In the RBPF, the observation noise is modelled as a slowly time-varying autoregressive signal of order $M_v = 5$. In the simulation, only the order M_v is a priori known to the algorithm. An artificial, stationary noise is added to the clean speech signal such that the overall input SNR is 4.30 dB.

less, informal listening tests show that the resulting speech is reasonably more intelligible, but especially we note that the very annoying-sounding colored noise has considerably faded in the enhanced speech.

The results are only slightly inferior when the AR noise is slowly time-varying. Unfortunately, the more the AR noise varies, the poorer the algorithm behaves. This is due to the fact that the variability of the AR noise is the only distinctive property given to the RBPF algorithm to separate it from the speech. For example, if used as presented here, this algorithm cannot separate two speech signals. By curiosity, we have tried to do so, but the resulting "separated" signals both sound like the original mixture, with both amplitudes modulated appearingly randomly. To be able to succeed, we would need to give more information to the RBPF concerning the independence of the two signals.

6.3 Improvements to the basic RBPF algorithm for speech enhancement ($\star$)

In this section, we present two novel approaches to improve the quality of the estimates obtained with the basic RBPF algorithm for speech enhancement presented in Algorithm 11. In the first approach, the state of the RBPF is augmented with a single extra component, which has roughly the effect of an adaptive temporal, short-time smoother. In the second approach, we use the information contained in the covariance matrices propagated by the Kalman filters in the RBPF to perform a simple frame-based post-filtering scheme, in an attempt to reduce the perceived noise.

6.3.1 The FIR-augmented RBPF algorithm (F-RBPF) ($\star$)

Recall in section 5.1.4, during our analysis of the simulation results from the application of the basic RBPF algorithm for speech enhancement, we had observed that the enhanced speech is polluted by a residual noise which seems modulated in amplitude by the enhanced speech. Unfortunately, this noise is ample enough not to always be masked by the speech components of the enhanced signal. But on the reasonable assumption that the speech is more correlated than the modulated noise, to improve the quality of the output signal, as a first try it might be tempting to smooth out the parts of the estimated signal which contain this noise, by convolving the enhanced speech with a smoothing function, small enough to avoid distortion if possible. Ideally, this function should itself adapt autonomously to the signal.

In the context of sequential estimation as performed by RBPFs, there is a very simple way to include such ideas within the algorithm, in fact as part of the estimation and at a minimal extra computational cost.

Let us rewrite here the speech model that we have been using so far:

$$\mathbf{x_{2}}_k = \mathbf{A}_k \mathbf{x_{2}}_{k-1} + \mathbf{G}_k \mathbf{w}_k \tag{6.12}$$

$$z_k = \mathbf{C}\mathbf{x_{2}}_k + \sigma_{v,k} v_k \tag{6.13}$$

The idea is to modify $\mathbf{C}$ into a time-varying, length M vector $\mathbf{h}_k^T$. Then, according to this model, we are not observing an autoregressive signal, but a filtered autoregressive signal, $\widetilde{\mathbf{x}}_{\mathbf{2}k} \triangleq \mathbf{h}_k^T \mathbf{x_{2}}_k$, as a representation of clean speech. The shape of the filter is controlled by a separate mechanism. In order to achieve a simple temporal smoothing, consider $\mathbf{h}_k$ to be a length M low-pass FIR filter, with cut-off frequency ω_k. If we use a standard windowing method to design the FIR filter, then the entire symmetric vector $\mathbf{h}_k \in \mathbb{R}^M$ is determined by the single number $\omega_k \in (0, 1)$.

If ω_k is close to 1, then the vast majority of the contribution of $\mathbf{x_{2}}_k$ to the observation z_k are due to its central elements. If ω_k is small, then the observed signal is a more even combination of the elements of the vector $\mathbf{x_{2}}_k$. To illustrate this, we have plotted a few graphs of possible resulting filters for different values of ω in Figure 6.2.

In addition, note that the filter coefficients are in this context a continuous function of ω. This makes a Gaussian random walk type of evolution plausible for the cut-off frequency ω_k. Figure 6.3 shows the coefficients of a small low-pass FIR filter as ω is varied from 0 to 1.

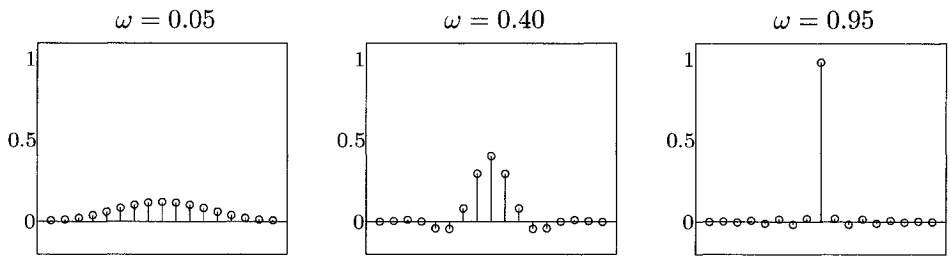


Figure 6.2: Low-pass FIR filter coefficients for different values of the cut-off frequency
The filter coefficients are more evenly distributed if the cut-off frequency is low. The filter shown has an odd length – for an even length, the two central coefficients are equal and tend to 0.5 as ω tends to 1.

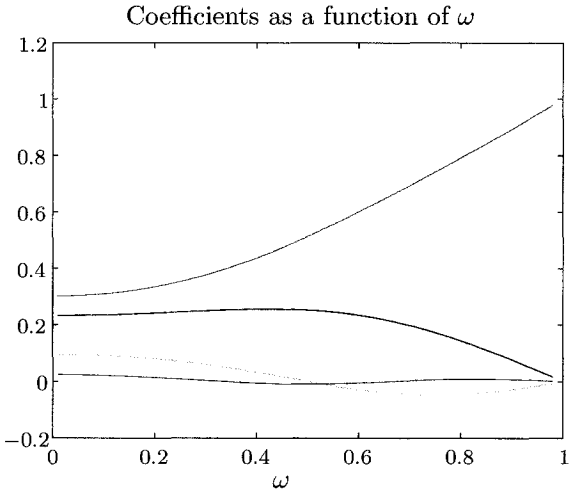


Figure 6.3: Low-pass FIR filter coefficients as a function of the cut-off frequency ω
Here, the filter coefficients are a continuous function of ω. Again, here the filter shown has an odd length.

The physical dependence between the cutoff frequency and the desired clean speech is not a priori obvious, although we clearly would expect them to be correlated. Our choice is to simply allow the cutoff frequency ω_k to drift according to a Gaussian random walk constrained in $[0+\epsilon, 1-\epsilon]$ where ϵ is a small number, and then to let the RBPF algorithm estimate the optimal values of the sequence ω_k.

What we expect to witness is the following: if the measurements and the previous state estimates indicate that the clean signal is likely "sharp" and "abrupt", then the estimated ω should be close to 1. In contrast, if the expected speech signal is either weak (drowned in noise)

or varying more slowly, then the optimal cutoff frequency should decrease. As a byproduct, this should help reducing the residual noise, and in general we foresee that this scheme will encourage the RBPF to look for estimates yielding to a smoother enhanced signal, but still allowing, if necessary, some occasional abrupt, high frequency changes. We can further anticipate an improvement by considering that a speech signal generally lies in the lower frequency range. We can hope that the use of such an adaptive low-pass filter should most often leave the crucial parts of the speech signal (in terms of intelligibility) unaffected.

The resulting algorithm is obtained very simply from Algorithm 11, by adding to the vector $\mathbf{x_1}_k$ the number ω_k, defining a Gaussian random walk on it, and each time obtaining a corresponding vector $\mathbf{h}_k^T \equiv \mathbf{C}_k$ to return to the Kalman filters of the RBPF. The speech estimates are given by $\hat{\mathbf{x}}_{\mathbf{2}k} \triangleq \hat{h}_k^T \hat{\mathbf{x}}_{\mathbf{2}k}$ for every k. In the algorithm, note that using this procedure makes the resulting speech directly dependent on both the recent past and the close future samples. At the heart of the RBPF, state candidates are drawn according to their history, and after a few iterations (depending on the filter lag/delay), they are confirmed or discarded.

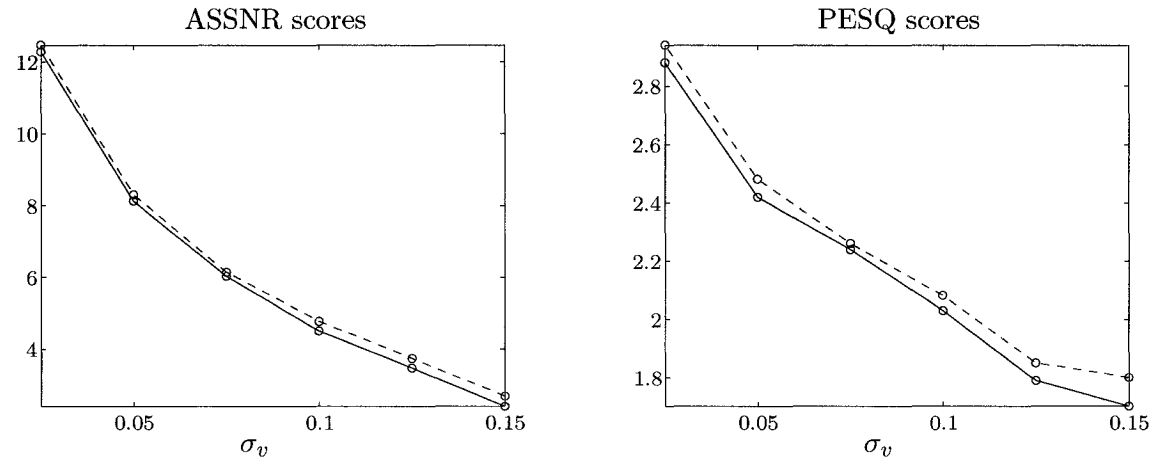
To differentiate this algorithm from the other algorithms, we call it the "FIR-augmented RBPF", or simply F-RBPF.

6.3.2 Simulation results

For the implementation of the algorithm, we choose that the variance of the Gaussian random walk on ω_k to be 5×10^{-3}. We choose the other parameters to be all equal to those used in 5.1.4. In addition, to reduce the computational load, we first determine a polynomial fit for each of the filter coefficients as functions of ω (i.e., a fit for the type of curves shown in Figure 6.3). We then use these functions to map for every particle $\omega_{k,i}$ to the vector $\mathbf{h}_{k,i}^T$. By doing so, we indeed observe a smaller execution time: instead of calling the "sinc" function and multiplying it by a Hamming window for every particle, a few additions and multiplications are performed. Note that from the symmetry of the filter, the computations must be done only for half of the coefficients.

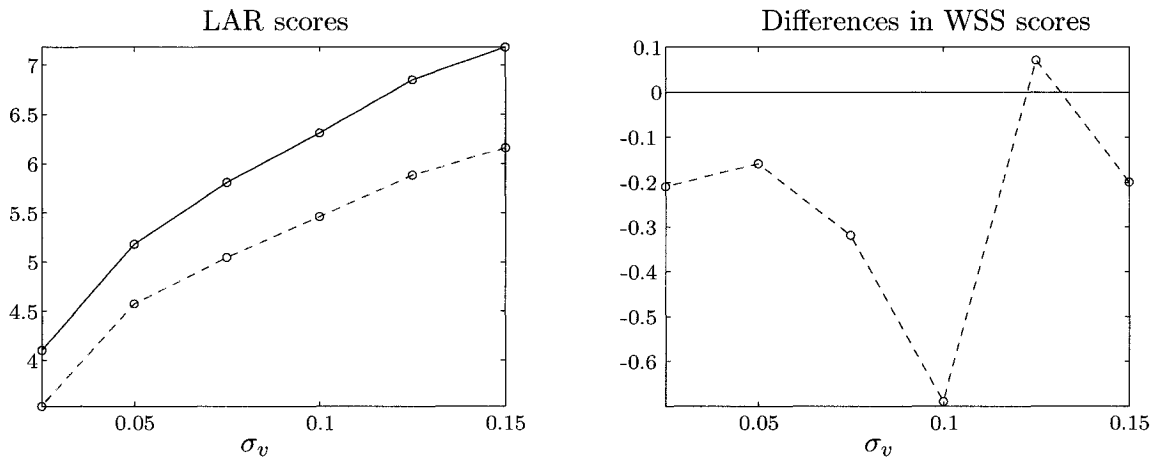
Simulation results (averaged over 20 trials) are shown in Table 6.4, and are represented graphically in Figures 6.4 and 6.5. The summary of the comparisons is very simple: according to all of the speech quality measures used, the F-RBPF is better than the basic RBPF, and this improvement is observed for all of the observation noises tested. The best improvement is observed in terms of the LAR score.

We also find that the residual noise appears to have faded, although it is disappointedly



(a) Average segmental SNR comparisons (b) PESQ comparisons

Figure 6.4: Comparisons of ASSNR and PESQ for RBPF vs. F-RBPF
The blue curves are the results obtained with a regular RBPF, and the red dashed curves are for the F-RBPF.



(a) LAR comparisons (b) WSS differences

Figure 6.5: Comparisons of LAR and WSS for RBPF vs. F-RBPF
For these measures, a lower scores means an improvement. The blue curves are the results obtained with a regular RBPF, and the red dashed curves are for the F-RBPF. On graph (b), only the score differences are shown: the WSS score of the regular RBPF is subtracted from both curves (which is why the blue curve is at 0). For example, at $\sigma_v = 0.1$*, referring to Table 6.4, the value shown is* $54.19 - 54.88 = -0.69$.

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.65**	**2.29**	**4.19**	**6.63**	**10.24**	**16.22**
	ASSNR	-1.56	-0.78	0.31	1.91	4.64	9.76
	PESQ	1.42	1.51	1.63	1.74	1.97	2.39
	LAR	8.80	8.54	8.08	7.60	6.66	5.22
	WSS	61.53	60.29	53.29	49.71	38.91	26.47
RBPF (Alg. 11) fixed-lag smoothing with $L = 8$	OSNR	6.81	7.94	9.35	11.01	13.44	17.92
	ASSNR	2.38	3.47	4.50	6.02	8.13	12.28
	PESQ	1.70	1.79	2.03	2.24	2.42	2.88
	LAR	7.18	6.85	6.31	5.81	5.18	4.10
	WSS	62.96	60.91	54.88	49.39	38.94	26.16
F-RBPF fixed-lag smoothing with $L = 8$	OSNR	7.13	8.14	9.56	11.10	13.62	18.17
	ASSNR	2.67	3.73	4.76	6.13	8.30	12.46
	PESQ	1.80	1.85	2.08	2.26	2.48	2.94
	LAR	6.16	5.88	5.46	5.04	4.57	3.53
	WSS	62.76	60.98	54.19	49.07	38.78	25.95

Table 6.4: Comparison of the F-RBPF algorithm to a regular RBPF
The additive noise is white and Gaussian; the F-RBPF is tested on the very same set of noisy signals as in the initial RBPF experiments (the first rows of the table are the same as those of Table 5.1), and the results shown are an average over the 20 runs per σ_v.

still present. By comparing the waveforms of the enhanced signals produced by the RBPF and by the F-RBPF, we observe that some areas benefit more than others from the effects of the FIR filter. In Figure 6.6, we show a plot of the errors $e_k = x_k - \hat{x}_k$ superimposed for both algorithms. In this plot, both algorithms were initialized exactly in the same way, and were working on the very same segment of noisy speech.

We are also interested in observing the shape of the resulting estimates for the cutoff frequency ω_k of the FIR filter as a function of time, and whether or not this shape follows the enhanced speech in some way. We show some typical plots in Figure 6.7. First, it is clear that the estimates of ω and the speech signal are highly correlated. On the right-hand side, we can observe the results obtained when a vowel is spoken. It appears that both the rate of change and the amplitude of the signals have an influence on the resulting ω. In fact, other observations indicate that there is a strong similarity between the curves of the estimates of ω and the sequence of successive slopes (the "derivative") of the original speech signal. We also see that as an overall tendency, the cutoff frequency is on average raised during an utterance, and lowered during pauses.

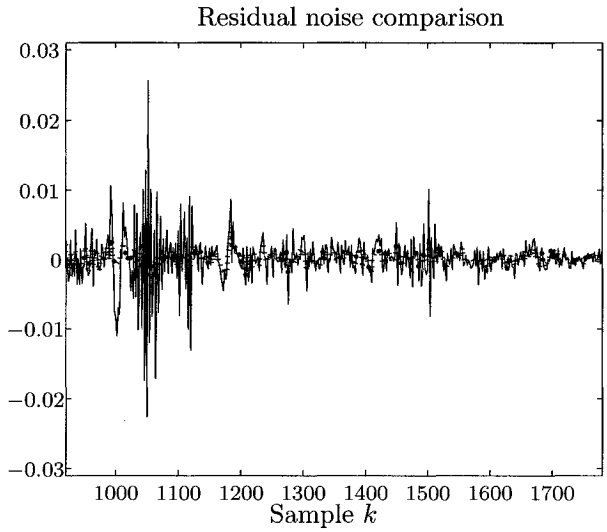


Figure 6.6: Comparison of residual noises on a speech segment: RBPF vs. F-RBPF
The residual noise obtained by the RBPF is here in blue, and in red dots is the residual noise for the F-RBPF.

6.3.3 The RBPF with FIR-based Postfiltering (RBPF-P) ($\star$)

In this section, we present another type of improvement for the RBPF algorithm, although we may see a relationship between the F-RBPF and the one presented here.

In the following, the main idea comes from the following assumption: in the noisy speech, the spectral components of the additive noise that are located in the higher frequency range have a tendency to be more perceived (or to be less masked by the clean speech). Thus, if we could remove from the enhanced signal these higher frequencies whenever we can assume that this removal will have a smaller impact on the clean signal, then we should expect a better sounding result – although the improvement is expected to depend on the actual frequency content of the noise.

Practically, we are looking for speech frames where the estimated "signal-to-estimation error" ratio is small. If this ratio is small, then the frame still contains a large amount of noise, and based on our assumption the listener will likely be more sensitive to its high frequency components. We can take the risk of distorting the speech contained in the frame by filtering out the higher frequencies. This risk is not necessarily very high, since in general, that the speech signal is more likely to lie in the lower frequency regions. In addition, intelligibility may be improved, even if some higher frequency parts are removed on occasional frames.

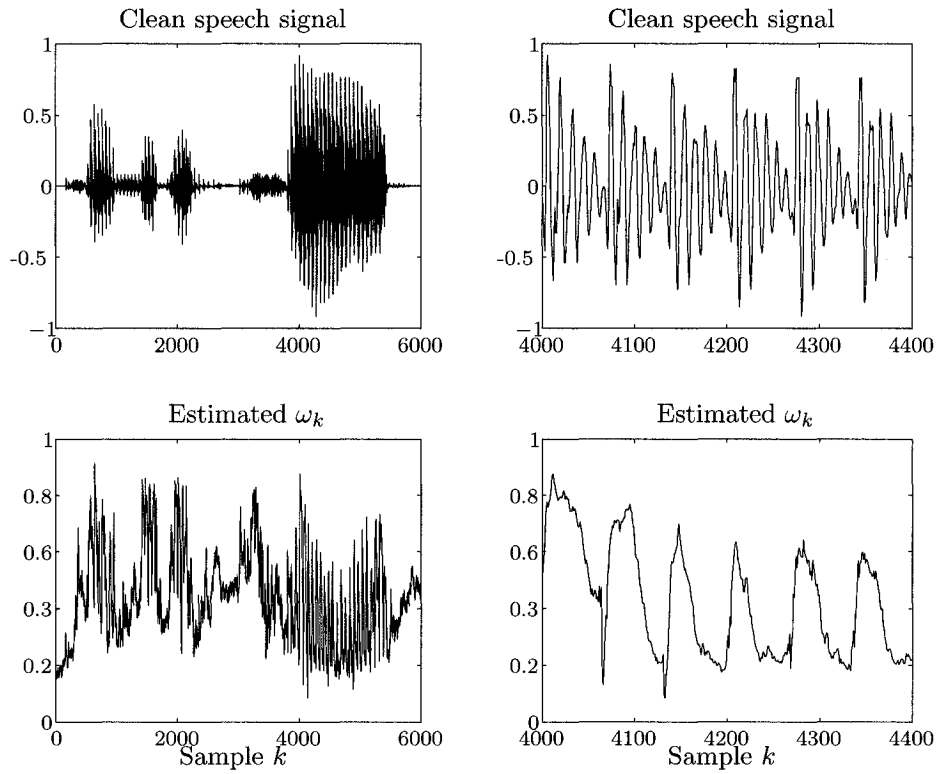


Figure 6.7: F-RBPF: estimates of the cutoff frequency as a function of time
The two top graphs show the original clean speech signal. The right-hand side graph is a zoomed version of the left one, specifically on a vowel. At the bottom, the estimates for ω_k are plotted.

We mentioned the need for an estimation of the "signal-to-estimation error" ratio. In an RBPF, this information can be conveniently obtained from the covariance matrices of the internal Kalman filters. As part of the estimation process, the RBPF returns not only a global estimated $\hat{\mathbf{x}}_{2k}$, but also a global estimated error covariance matrix $\hat{\mathbf{K}}_k$. Under the assumption that the signals are zero mean, then since $\hat{\mathbf{x}}_{2k} \triangleq [\hat{x}_k \ \hat{x}_{k-1} \ \ldots \ \hat{x}_{k-M+1}]^T$ (where $\hat{x}_k$ is the estimated clean speech signal), therefore supposing that the enhanced signal is "short-time WSS", then the i^{th} element of the first row of $\hat{\mathbf{K}}_k$ is an estimate of the autocorrelation function of x_k at a lag $i-1$. Using the Wiener-Khinchin theorem, taking the DFT of this function (constructed from lags $-M+1$ to $M-1$ since the autocorrelation function is even) yields an estimate of the power spectral density of the estimation error, from which an average power can be obtained. In turn, directly from the most recent frame (of length L – we chose $L = 128$ in our implementation), we can obtain the average power of the (estimated) clean signal. From

these two powers, an estimate for the current "signal-to-estimation error" ratio is computed – let us denote it $\widehat{SNR}_k$. (The reasoning to compute $\widehat{SNR}_k$ is close to that presented in [37], except that we only use the first row of $\hat{\mathbf{K}}_k$ at the basis of our computations).

Now, we said that we would like to filter out the higher frequency components of the enhanced signal depending on the value of $\widehat{SNR}_k$. Practically, we decide to compute $\widehat{SNR}_k$ and filter out the enhanced speech every L samples, so that the procedure is a form of post-filtering on frames of data. It remains to decide how the filtering will take place from the value of $\widehat{SNR}_k$. We require a mapping from $\widehat{SNR}_k$ to a cut-off frequency ω_k. We determined heuristically that a mapping yielding good results is the following:

$$\omega_k = \frac{2 \times \arctan\left(\widehat{SNR}_k\right)}{\pi} \tag{6.14}$$

so that ω_k belongs to $(0;1)$. The low-pass filter that we use is a simple FIR filter of (small) order N_{FIR}.

To summarize, the algorithm can be directly applied from Algorithm 11, which does not require any changes except that we need, for every frame of L samples, to:

- compute an estimate $\hat{\mathbf{K}}_k$ for $\mathbf{K}_k$
- using $\hat{\mathbf{K}}_k$ and the most recent frame of enhanced speech, estimate $\widehat{SNR}_k$
- map $\widehat{SNR}_k$ to ω_k, and
- filter the enhanced speech with a low-pass filter.

Note that for a valid FIR filtering, we must first take into account the filter delay (to keep the enhanced speech and the post-filtered enhanced speech signals time-aligned), but also to appropriately discard the first few post-filtered values (for which the FIR filter is not full yet) according to the filter order. Therefore, overlapping frames must be used.

We call the resulting algorithm "RBPF-P", for "RBPF with (FIR)-Postfiltering".

6.3.4 Simulation results

We show simulation results in Table 6.5, where 20 simulations were averaged. We use an FIR filter with a small order (6), and we perform filtering on frames of length 128.

We can draw several conclusions from the observation of the results:

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.65**	**2.29**	**4.19**	**6.63**	**10.24**	**16.22**
	ASSNR	-1.56	-0.78	0.31	1.91	4.64	9.76
	PESQ	1.42	1.51	1.63	1.74	1.97	2.39
	LAR	8.80	8.54	8.08	7.60	6.66	5.22
	WSS	61.53	60.29	53.29	49.71	38.91	26.47
RBPF (Alg. 11) fixed-lag smoothing with $L = 8$	OSNR	6.81	7.94	9.35	11.01	13.44	17.92
	ASSNR	2.38	3.47	4.50	6.02	8.13	12.28
	PESQ	1.70	1.79	2.03	2.24	2.42	2.88
	LAR	7.18	6.85	6.31	5.81	5.18	4.10
	WSS	62.96	60.91	54.88	49.39	38.94	26.16
F-RBPF fixed-lag smoothing with $L = 8$	OSNR	7.13	8.14	9.56	11.10	13.62	18.17
	ASSNR	2.67	3.73	4.76	6.13	8.30	12.46
	PESQ	1.80	1.85	2.08	2.26	2.48	2.94
	LAR	6.16	5.88	5.46	5.04	4.57	3.53
	WSS	62.76	60.98	54.19	49.07	38.78	25.95
RBPF-P fixed-lag smoothing with $L = 8$	OSNR	7.25	8.32	9.54	11.01	13.11	16.30
	ASSNR	3.38	4.33	5.37	6.70	8.55	11.92
	PESQ	1.86	1.97	2.18	2.35	2.54	3.00
	LAR	7.12	6.89	6.72	6.40	6.00	5.37
	WSS	64.49	62.24	55.57	49.92	39.50	26.74

Table 6.5: Comparison of the RBPF-P algorithm to the F-RBPF and the regular RBPF *The additive noise is white and Gaussian. The RBPF-P is also tested on the very same set of noisy signals as the initial RBPF and the F-RBPF (the first rows of the table are the same as those of Table 6.4), and the results shown are an average over the 20 runs per* σ_v.

- For larger observation noise variance, the overall SNR (OSNR) for the RBPF-P is on average larger than that obtained with the F-RBPF and the RBPF. For smaller variance however, the OSNR is smaller than that of the other algorithms, with in fact a quite low value for the smallest observation variance tested.
- Except for the smallest observation variance in the table, **all** of the ASSNR values are better to much better than those obtained with the other RBPF algorithms.
- **all** of the PESQ scores obtained are better than with the other algorithms, especially when the observation noise is large.
- The LAR and the WSS scores are, however, all worse than even with the regular RBPF algorithm. This seems to indicate that the post-filtering scheme denatures the spectrum of the enhanced speech in a way which affects these measures more specifically than the other measures.

From subjective listening tests, our impressions are that the RBPF-P produces more intelligible enhanced speech signals than the other two RBPF algorithms when the observation noise is large. One of the strongest advantages is the important reduction of high-frequency residual noise. Unfortunately, this comes at a cost: potentially there are larger parts of the true speech signal that are also being altered – more often than with the F-RBPF – since the cutoff frequency is only updated at every frame (the F-RBPF was able to adapt its FIR cut-off frequency very fast, from one sample to the next). We believe that this is one of the reasons why the LAR and WSS scores are poor for the RBPF-P. Additionally, overall the F-RBPF-enhanced speech signals sound more natural than with both the RBPF-P (and the regular, basic RBPF), which may explain the very good LAR/WSS scores for the F-RBPF.

An example of the values of ω_k computed as part of the RBPF-P algorithm is shown in Figure 6.8. For this example, the observation noise was 0.05, and we see that the values of omega seem to follow quite closely the speech utterances, only fully "opening" the filter when judged necessary. However, note how the part of the original speech, from $k = 3200$ to 3800, will be severely affected by the postfilter, for the center frequency of the filter is then of the order of 0.15. For the RBPF-P, this situation is more likely to happen for unvoiced speech, such as the part from $k = 3200$ to 3800.

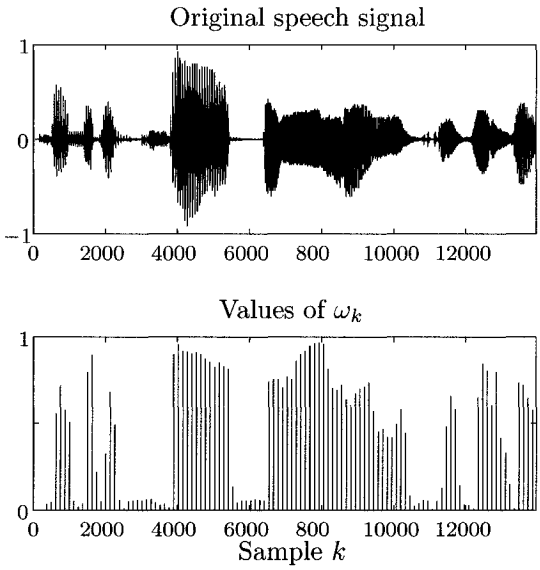


Figure 6.8: RBPF-P: computed values of the cutoff frequency for each frame
The values of ω_k are computed only for each frame – hence the discrete graph.

6.4 Comparisons with other existing speech enhancement algorithms

In this section, we will compare the RBPF-based algorithms to different, previously published algorithms. The goal is to try to have an idea of what the pros and cons of RBPF enhanced speech are compared to other possible solutions.

In order to make the section readable, we separate this section into two parts. The first part deals with the basic algorithms presented in 4.3, and the second part deals with other algorithms for which we obtained demonstration audio examples, and thus for which comparison was also directly possible.

6.4.1 Comparisons with the algorithms of section 4.3

In this first subsection, we report and comment the results of our comparisons with the following algorithms, for which we have our own implementation:

- A basic spectral subtraction scheme, abbreviated by SSUB (section 4.3.2)
- The "perceptually enhanced" spectral subtraction of [51], PSSUB (section 4.3.5)
- A basic Kalman filter-based method employing the Yule-Walker equations, KFYW (section 4.3.3)
- A signal subspace method, KLT (section 4.3.4)

For the comparisons reported in this section, we used the very same batch of speech/noisy signals which we had used in the previous sections for the RBPF algorithms (recall that we had stated in section 5.1.4 that each noise sequence was saved), and we computed the average performance for each of the 20 experiments per value of σ_v. Note that we have also, in parallel, conducted other random experimentations on different speakers and sets of data, and we have found that the observations made (and the hierarchies established) in this section globally apply for the majority of situations observed. In section 6.4.2 below, we will compare the performance of RBPF methods to other types of algorithms based on two other datasets (different speakers and noise levels), for which the RBPF algorithms are also very competitive, demonstrating thereby that we have not "tuned" the algorithms specifically for the experimental conditions of this section. This is why, for clarity and simplicity we only report here the results obtained using the same set of conditions as in the previous section.

The comparisons are only done for the additive white Gaussian noise case, where the variance is considered to be known, although we do realize that this may be a strong initial advantage for certain algorithms, and only a small one for others. As mentioned in 5.2.1, our experience is that for RBPF methods, the initial knowledge of the observation variance can be very approximative without impacting the results. Since their implementation only introduces an almost negligible amount of computations, the RBPF methods considered for comparisons are the F-RBPF and the RBPF-P. Note also here that for a given noisy signal, the SSUB, PSSUB, KFYW, and the KLT yield a unique output signal, whereas RBPF-based signals always give different results. By using 20 different noise sequences for each σ_v and averaging the output performance, we hope to give a correct idea of what results can be expected.

Simulation results are shown in Table 6.6. There are several interesting points to discuss from the consultation of this table.

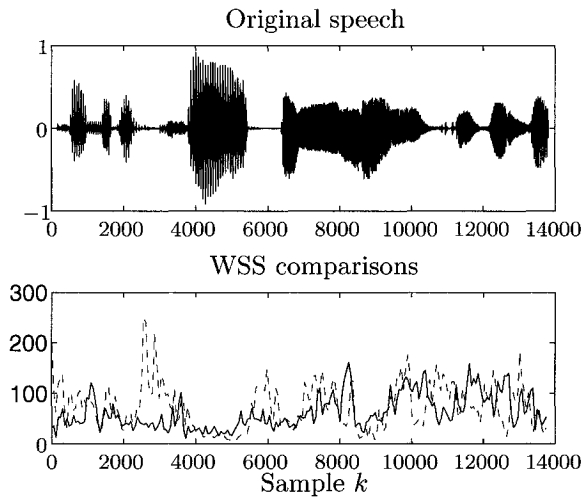


Figure 6.9: Comparisons of the WSS scores obtained on each frame: F-RBPF vs PSSUB
The WSS for each frame is plot against the clean signal. The F-RBPF scores are in blue, and the PSSUB ones are in dashed green. These WSS scores were obtained for $\sigma_v = 0.05$.

- The PSSUB is superior to the SSUB in terms of OSNR, ASSNR, and PESQ, but its LAR is very slightly penalized, while its WSS is extremely penalized. In Figure 6.9, we show a plot of the WSS scores obtained for each frame, time-aligned with the original speech signal. It appears that the PSSUB method is only really penalized from $k = 2000$ to 4000. In fact, if we had only computed the average WSS score for $k > 6000$, we would have obtained a score of 78.35 for thr F-RBPF and 80.03 for the PSSUB. Our conjecture to explain this phenomenon is that the spectral distortion in the output signal of the

PSSUB may be perceived as being less important than they truly are (several spectral manipulations occur within the PSSUB algorithm), and the WSS measure is by nature very sensitive to spectral differences.

- Comparing only the PSSUB/SSUB to the RBPF methods, we see that for the quasi-totality of the quality measures used, RBPF methods obtain better scores, especially in terms of ASSNR, and LAR for the F-RBPF. (In fact, the F-RBPF consistently obtains the best LAR scores of all the algorithms tested.)

- A similar conclusion can be drawn from the comparison of the RBPF methods to the KFYW one used in this work. In fact, for smaller observation variance ($\sigma_v = 0.025$), the KFYW is doing relatively well, although the results are on average much lower in PESQ (2.60 versus 2.94 for the F-RBPF and 3.00 for the RBPF-P). But the problem is that the scores quickly drop as σ_v increases, at a much faster rate than for the RBPF algorithms. Note finally that surprisingly, the WSS scores are not suffering from this quick drop – in fact the KFYW algorithm is given the best WSS scores of all algorithms for $\sigma_v = 0.15$, 0.1, and 0.075.

- The comparison with the signal subspace algorithm results appears more balanced. We observe first that the signal subspace is consistently given the best ASSNR scores compared to the average score obtained by the RBPF methods. On the other hand, the KLT's enhanced speech is rated with a much worse LAR and WSS score than the RBPFs, especially when σ_v is large (see Figure 6.10). As for the PESQ, observe Figure 6.11: the RBPF-P obtains the best results for the tests where $\sigma_v \leq 0.1$, with an increasing difference in performance (at $\sigma_v = 0.025$, the signal subspace's PESQ is only 2.69). The F-RBPF also seems to perform better, in terms of PESQ, for lower observation noises. However, for larger σ_v, the signal subspace is given a better PESQ score.

Our conclusions are that first, the RBPF methods are clearly superior to the classic spectral subtraction algorithms, as well as to our implementation of the classic Kalman filter-based methods. A direct hierarchical comparison with the signal subspace approach, on the other hand, is more difficult, because of the fact that the quality measures appear to "contradict" one another. According to the LAR, for example, the RBPF methods are clearly superior. But according to the ASSNR scores, then the KLT algorithm is the best method. This is in our opinion a typical case demonstrating the limits of objective quality measures, and where only subjective listening may clearly differentiate the two. Let us describe here some of our impressions concerning the performance of the two algorithms.

Very roughly, the RBPF methods sound more intelligible, more natural and clearer when the SNR is high, but becomes more subject to some residual, intraspeech noise when the SNR is

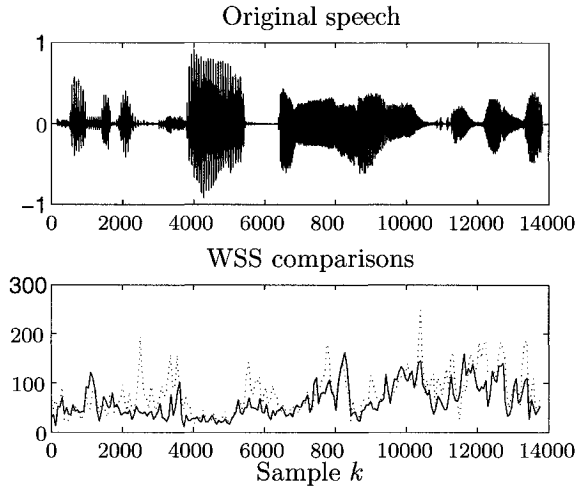


Figure 6.10: Comparisons of the WSS scores obtained on each frame: F-RBPF vs KLT
The WSS for each frame is plot against the clean signal. The F-RBPF scores are in blue, and the KLT ones are in dashed red. These WSS scores were obtained for $\sigma_v = 0.05$.

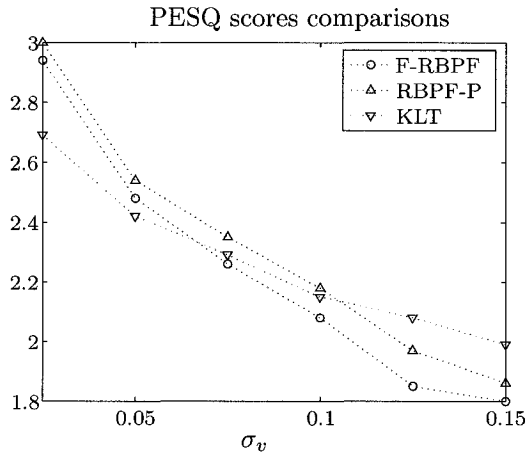


Figure 6.11: Comparisons of the PESQ scores: RBPF methods vs KLT

low, in which case the signal subspace's enhanced signal is of better quality. Our experience is that the signal subspace has the capacity to produce smoother signals than RBPFs at low SNR. However, there is an important factor to consider (which may explain the very bad LAR and WSS scores for the signal subspace): the RBPF methods do not suffer from musical-type noise. The RBPF-enhanced speech is, granted, still corrupted by a residual noise, but this noise can be assimilated to a form of amplitude-modulated white noise. The naturalness of the speech is not perceivably affected. In contrast, the signal subspace, because of its similarities to spectral subtraction, produces speech signals which can sound "robotized" and unnatural. In addition,

there remains in the enhanced signal a small musical noise (although it is much smaller than for classical spectral subtraction agorithms). If a conversation must take place, then it may be more annoying to listen to robotized voices than noisier but more natural sounding ones. This is in fact precisely what we often found to be reported by individuals listening to the output of these algorithms.

The need to further compare the two algorithms, in colored/nonstationary or even unknown noise statistics is clear to the author. We must not forget that for the comparisons conducted here, the signal subspace algorithm is being given, before execution, the value of the noise covariance matrix. It may not be as reliable when no such information is available, and if the noise's statistics are time-varying. This is due to the fact that signal subspace methods require an estimate for the clean speech autocorrelation from the noisy speech signal.

6.4.2 Comparisons with other miscellaneous algorithms

Accessorily, we will also be doing other comparisons using some results retrieved from other authors' demonstrations, without having implemented the algorithms ourselves. These demonstrations are for:

- A perceptually constrained Kalman filter (PCKF) [37]. Audio samples were retrieved from presentation slides written for an illustration of [36, 37].
- Dual perceptually constrained unscented Kalman filter (DCUKF) [36]. Audio samples were retrieved from presentation slides written for an illustration of [36, 37].
- Dual extended Kalman filter (DEKF) [53]. The samples used can be retrieved from Dr. E. Wan's demonstration page: `http://cslu.ece.ogi.edu/nsel/demos/index.html`

The clean and noisy data are common to the PCKF and to the DCUKF, and so comparative results can be agglomerated into a single table, presented in Table 6.7. Since these algorithms do not assume that the noise variance is a priori known, we apply the same constraint on the RBPFs to allow comparisons, and we let the algorithms estimate the noise online. Besides this, no modification is brought to the F-RBPF and the RBPF-P algorithms of section 6.3. The sentence is spoken by a female, saying: "*you were the perfect hostess, he punched viciously at the ball*". We compute the average performance of the RBPF algorithms for 10 experiments.

For the DEKF results shown here, the noise variance σ_v^2 is initially known, so we do the same for the RBPF algorithms. The results are shown in Table 6.8, and for the RBPF algorithms are an average for 10 experiments. The sentence is spoken by a male, saying: "*primitive tribes have an upbeat attitude*".

From the consultation of Table 6.8 only, it appears that the only measure for which the DEKF performs better is the PESQ score. Overall, according to the other measures, the RBPF algorithms are performing better, especially in terms of ASSNR and LAR. Based on listening tests, conducted on a small sample of a dozen different people, we gathered that the RBPF methods are found to be more intelligible, because of the amount and the nature of residual noise in the DEKF-enhanced speech, which is apparently more annoying than in the RBPF case.

Now, for the results shown in Table 6.7, we find that the RBPF algorithms both yield significantly better results in terms of almost all of the measures used, especially for the ASSNR. A single comparison goes in favor of the PCKF and the DCUKF: it is the one made using the PESQ score at 0dB, compared to the F-RBPF.

Type of algorithm	*Quality measure*	AWGN's std σ_v					
		0.15	0.125	0.1	0.075	0.05	0.025
Noisy speech (no processing)	OSNR	**0.65**	**2.29**	**4.19**	**6.63**	**10.24**	**16.22**
	ASSNR	-1.56	-0.78	0.31	1.91	4.64	9.76
	PESQ	1.42	1.51	1.63	1.74	1.97	2.39
	LAR	8.80	8.54	8.08	7.60	6.66	5.22
	WSS	61.53	60.29	53.29	49.71	38.91	26.47
F-RBPF fixed-lag smoothing with $L = 8$	OSNR	7.13	8.14	9.56	11.10	13.62	18.17
	ASSNR	2.67	3.73	4.76	6.13	8.30	12.46
	PESQ	1.80	1.85	2.08	2.26	2.48	2.94
	LAR	6.16	5.88	5.46	5.04	4.57	3.53
	WSS	62.76	60.98	54.19	49.07	38.78	25.95
RBPF-P fixed-lag smoothing with $L = 8$	OSNR	7.25	8.32	9.54	11.01	13.11	16.30
	ASSNR	3.38	4.33	5.37	6.70	8.55	11.92
	PESQ	1.86	1.97	2.18	2.35	2.54	3.00
	LAR	7.12	6.89	6.72	6.40	6.00	5.37
	WSS	64.49	62.24	55.57	49.92	39.50	26.74
SSUB	OSNR	7.30	7.90	9.30	10.71	13.14	16.34
	ASSNR	1.97	2.37	3.80	5.27	7.82	11.15
	PESQ	1.78	1.84	2.04	2.21	2.44	2.91
	LAR	7.40	7.01	6.83	6.16	5.14	4.22
	WSS	77.74	72.11	69.10	61.91	48.03	38.56
PSSUB	OSNR	7.34	8.06	9.48	10.75	13.11	16.65
	ASSNR	2.28	2.71	3.99	5.39	7.88	11.30
	PESQ	1.83	1.92	2.06	2.33	2.46	2.96
	LAR	7.76	7.15	7.13	6.20	5.68	4.67
	WSS	105.44	101.31	95.24	84.66	73.43	65.26
KFYW	OSNR	4.79	6.02	7.73	9.73	12.68	17.80
	ASSNR	-0.12	0.75	2.07	3.78	6.42	11.24
	PESQ	1.57	1.66	1.81	1.95	2.17	2.60
	LAR	8.18	7.81	7.29	6.73	5.92	4.62
	WSS	61.72	61.07	53.44	48.78	39.00	26.38
Signal subspace	OSNR	8.36	9.13	10.14	11.82	13.99	17.19
	ASSNR	4.55	5.14	5.90	7.61	9.55	13.23
	PESQ	1.99	2.08	2.15	2.29	2.42	2.69
	LAR	10.98	10.38	9.51	8.65	7.97	5.82
	WSS	96.15	85.60	74.21	67.82	57.10	38.59

Table 6.6: Comparison of RBPF-based algorithms to other speech enhancement schemes
For all of the algorithms tested, again the set of noisy signals is the same as the one that was used previously for simulations (AWGN). The algorithms were executed for each of the 20 noisy signals with noise variance σ_v^2, and an average performance was computed. Note that all algorithms – including the signal subspace – were given the value of the noise covariance initially. We have already stated in subsection 5.2.1 that this is not a significant advantage for the RBPF methods, but it may however be so for other algorithms.

Type of algorithm	*Quality measure*	Input SNR	
		0	10
No processing	OSNR	0	10
	ASSNR	-3.84	1.31
	PESQ	1.11	1.57
	LAR	8.96	7.61
	WSS	60.40	38.65
F-RBPF fixed-lag smoothing with $L = 8$	OSNR	7.42	14.45
	ASSNR	0.61	5.43
	PESQ	1.42	1.92
	LAR	7.16	6.00
	WSS	60.78	37.40
RBPF-P fixed-lag smoothing with $L = 8$	OSNR	7.84	13.93
	ASSNR	1.33	6.11
	PESQ	1.64	2.06
	LAR	7.31	6.68
	WSS	62.08	37.90
PCKF	OSNR	-0.23	2.73
	ASSNR	-2.37	0.76
	PESQ	1.61	2.04
	LAR	8.33	8.98
	WSS	65.45	46.24
DCUKF	OSNR	6.20	10.50
	ASSNR	-1.00	3.48
	PESQ	1.63	2.00
	LAR	8.68	7.85
	WSS	71.16	46.71

Table 6.7: Comparison of RBPF-based algorithms to perceptually constrained KF schemes *The input signal (a female voice) is corrupted with additive white Gaussian noise. For the two cases tested, the input SNR is respectively* 0 *dB and* 10 *dB. The results obtained from the RBPF-based methods are an average over 10 experiments.*

Type of algorithm	*Quality measure*	Input SNR	
		0	7
No processing	OSNR	0	7
	ASSNR	-4.50	-1.34
	PESQ	1.76	2.16
	LAR	6.65	5.72
	WSS	45.98	36.76
F-RBPF fixed-lag smoothing with $L = 8$	OSNR	7.55	12.23
	ASSNR	0.54	3.14
	PESQ	1.78	2.11
	LAR	5.22	4.63
	WSS	44.83	35.59
RBPF-P fixed-lag smoothing with $L = 8$	OSNR	7.16	11.94
	ASSNR	0.74	3.55
	PESQ	1.83	2.22
	LAR	7.05	6.86
	WSS	45.95	36.52
DEKF	OSNR	7.54	11.72
	ASSNR	0.60	2.72
	PESQ	2.24	2.56
	LAR	6.29	5.47
	WSS	57.38	43.89

Table 6.8: Comparison of RBPF-based algorithms to the DEKF enhancement scheme
The input signal (a male voice) is corrupted with additive white Gaussian noise. For the two cases tested, the input SNR is 0 *dB and* 7 *dB. The results obtained from the RBPF-based methods are an average over 10 experiments.*

Chapter 7

Conclusion and future work

Based on the previous chapter, but also on the rest of this report, we can draw several important conclusions, weighing the pros and the cons of particle filtering algorithms for speech enhancement.

Up to chapter 4, we have explained and discussed particle filter algorithms in general, with an emphasis on Rao-Blackwellised particle filters (in chapter 3). Relatively simple approaches were adopted in the case of conditionally linear-Gaussian systems (in 3.3.1), showing that RBPFs can be readily applied to a wide range of problems. In addition, we proposed an algorithm to address the computational load of regular RBPFs, which is found to represent a compromise between accuracy and execution speed. In certain situations, the modified RBPF is shown to yield results that are very close to what can be expected from a regular RBPF, such as in the two examples of 3.4.4, and in the speech enhancement example shown in chapter 6.

In chapter 5, the techniques proposed in the first chapters were applied to obtain a basic RBPF for speech enhancement, and then a few new techniques were presented in chapter 6 . We first proposed an extension of the basic RBPF to the case of colored environment noise (see 6.2), which is derived via a modification of the state-space model, and we found the resulting scheme to perform well, provided the colored noise is "more" stationary than the speech. In section 6.3 two algorithms, the F-RBPF and the RBPF-P, were introduced to improve the quality of the enhanced speech signal. We found through simulation results that overall, both algorithms perform better than the basic RBPF. Roughly stated, amongst the two new algorithms, the RBPF-P yields the best results in terms of OSNR, ASSNR, and PESQ, but the F-RBPF is quite significantly better with respect to the LAR and the WSS scores. From subjective listening tests, we find that the RBPF-P is best suited for larger additive noise, reducing a large part of the residual noise, whereas the F-RBPF sounds "clearer" at high SNR. Additionally, the

F-RBPF overall sounds more natural (possibly explaining the very good LAR and WSS scores).

It is quite clear that overall, the RBPF algorithms presented in last chapter have the potential to earn a permanent spot among worthy choices of speech enhancement algorithms. The scores obtained using several objective quality measures show that they tend to produce results on the higher end of the quality scale. For some situations (for example, when the SNR is relatively high – see the rightmost columns of Table 6.6), they can obtain the best results among all the algorithms tested, for 4 of the 5 quality measures used. In addition, we have consistently observed that the LAR and WSS scores of the F-RBPF are the best of all the algorithms tested, and so under all of the conditions tested. Moreover, even when other algorithms (such as the DEKF or the signal subspace algorithm) do obtain better scores according to some of the quality measures (ASSNR and/or PESQ), the residual noise in the enhanced speech has often been found to be more acceptable, in the sense that it sounds more white than musical/artificial/robotic. In addition, with minimal adjustments we have seen that they are able to accomodate different types of noises, as in 6.2. In future research, we propose to investigate their performance in real-world noisy environments, and to derive ways to make these algorithms robust to such noises.

Of course, at the present stage they still suffer from several drawbacks, the importance of which we hope to lessen in the future. Among these drawbacks, we find that the most important one is that the amount of intraspeech residual noise is still quite large for low SNRs. But there is still much room for improvement, even for the linear-predictive model used in the previous chapter. Particle filters are very flexible, and there are very many parameters that can be adjusted: for example, the evolution model of the state vector, the choice of the importance density, etc.

The evolution model for the states, coupled with the importance density, are crucial to the performance of particle filters. The Gaussian random walks models are very simple to implement, but they cannot pretend to realistically capture the evolution of the physical quantities that they are trying to model. A possible research idea would be to try to determine from clean speech a better transitional model for the state vector.

For the F-RBPF and the RBPF-P, there are many extra parameters which could be further adjusted to try to reduce the residual noise. In addition, the fact that the LAR and WSS scores are very good for RBPF methods may be an indication that the enhanced signal contains a version of the original clean signal that is "less damaged" than with other methods.

Of course, there is also the disadvantage of the computational complexity of RBPFs. The modified RBPF (3.4) addresses this problem, and it is able to yield acceptable performances (5.1.4) with a faster execution, but it is associated with a drop in overall performance. We have written a RBPF code by means of Mex functions under Matlab, and for about 10 seconds of speech, the processing time is about a minute. But the code is far from optimized, and we believe that a fully optimized C implementation would certainly achieve real-time performance.

Another important area to focus on is that of noise modelling, in order to make the RBPF algorithms more robust to a larger number of situations. One first possibility would be to try to model different types of noise individually, and then to use a discrete state switching to the most appropriate model. Another way would be to incorporate a scheme to estimate the statistics of the noise online (a scheme of the type of 6.2, but of course more advanced).

Appendices

Appendix A

Lemma: Combination of Gaussian density functions

If $\mathcal{N}(\mathbf{a}|\mathbf{b};\mathbf{C}) \triangleq |2\pi\mathbf{C}|^{-1/2}\exp(-\frac{1}{2}(\mathbf{a}-\mathbf{b})^T\mathbf{C}^{-1}(\mathbf{a}-\mathbf{b}))$ *represents the density of a Gaussian random vector with mean* $\mathbf{b}$ *and covariance matrix* $\mathbf{C}$*, then:*

$$\int \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y};\mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z};\mathbf{K})\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{z};\mathbf{Q}+\mathbf{F}\mathbf{K}\mathbf{F}^T) \tag{A.1}$$

where the matrix $\mathbf{F}$, whose dimension is constrained by that of $\mathbf{x}$ and $\mathbf{y}$, can be chosen arbitrarily.

Proof:

We can write:

$$\mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y};\mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z};\mathbf{K}) = |2\pi\mathbf{Q}|^{-1/2}|2\pi\mathbf{K}|^{-1/2}\exp\Big[-\frac{1}{2}\Big((\mathbf{x}-\mathbf{F}\mathbf{y})^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{y})+(\mathbf{y}-\mathbf{z})^T\mathbf{K}^{-1}(\mathbf{y}-\mathbf{z})\Big)\Big] \tag{A.2}$$

From the lemma presented in appendix B, we can rewrite the argument of the exponential in (A.2) as follows:

$$\mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y};\mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z};\mathbf{K}) = |2\pi\mathbf{Q}|^{-1/2}|2\pi\mathbf{K}|^{-1/2}\exp\Big[-\frac{1}{2}\Big((\mathbf{y}-\mathbf{m})^T\mathbf{M}^{-1}(\mathbf{y}-\mathbf{m})+(\mathbf{x}-\mathbf{n})^T\mathbf{N}^{-1}(\mathbf{x}-\mathbf{n})\Big)\Big] \tag{A.3}$$

where:

$$\begin{aligned}\mathbf{M} &= \left[\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{K}^{-1}\right]^{-1}\\ \mathbf{N} &= \mathbf{Q}+\mathbf{F}\mathbf{K}\mathbf{F}^T\\ \mathbf{m} &= \mathbf{z}+\mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{z})\\ \mathbf{n} &= \mathbf{F}\mathbf{z}\end{aligned}$$

Effectively, the variables $\mathbf{x}$ and $\mathbf{y}$ in (A.3) are now separated (they are tied to a different quadratic term), and therefore we can easily proceed with the integration:

$$\int \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y};\mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z};\mathbf{K})\mathrm{d}\mathbf{y} = |2\pi\mathbf{Q}|^{-1/2}|2\pi\mathbf{K}|^{-1/2}|2\pi\mathbf{M}|^{1/2}\exp\left[-\frac{1}{2}\Big((\mathbf{x}-\mathbf{n})^T\mathbf{N}^{-1}(\mathbf{x}-\mathbf{n})\Big)\right] \quad (A.4)$$

and finally $\int \mathcal{N}(\mathbf{x}|\mathbf{F}\mathbf{y};\mathbf{Q})\mathcal{N}(\mathbf{y}|\mathbf{z};\mathbf{K})\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{x}|\mathbf{n};\mathbf{N})$ which is the desired result. □

Appendix B

Lemma: Separation of Quadratic Terms

Suppose that matrices $\mathbf{Q}$ and $\mathbf{K}$ are positive definite symmetric. Then, we have the following equality (where each of the vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ and the matrix $\mathbf{F}$ have appropriate dimensions):

$$(\mathbf{x}-\mathbf{F}\mathbf{y})^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{y})+(\mathbf{y}-\mathbf{z})^T\mathbf{K}^{-1}(\mathbf{y}-\mathbf{z})=(\mathbf{y}-\mathbf{m})^T\mathbf{M}^{-1}(\mathbf{y}-\mathbf{m})+(\mathbf{x}-\mathbf{n})^T\mathbf{N}^{-1}(\mathbf{x}-\mathbf{n})$$

where:

$$\begin{aligned}\mathbf{M} &= \left[\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{K}^{-1}\right]^{-1}\\ \mathbf{N} &= \mathbf{Q}+\mathbf{F}\mathbf{K}\mathbf{F}^T\\ \mathbf{m} &= \mathbf{z}+\mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{z})\\ \mathbf{n} &= \mathbf{F}\mathbf{z}\end{aligned}$$

Proof:

First, note that:

$$(\mathbf{a}-\mathbf{b})^T\mathbf{C}(\mathbf{a}-\mathbf{b})=\mathbf{a}^T\mathbf{C}\mathbf{a}+\mathbf{b}^T\mathbf{C}\mathbf{b}-\mathbf{b}^T\mathbf{C}\mathbf{a}-\mathbf{a}^T\mathbf{C}\mathbf{b} \tag{B.1}$$

We expand the left-hand side of the equation to be proven:

$$\begin{aligned}&(\mathbf{x}-\mathbf{F}\mathbf{y})^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{y})+(\mathbf{y}-\mathbf{z})^T\mathbf{K}^{-1}(\mathbf{y}-\mathbf{z})\\ &\quad=\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x}-\mathbf{y}^T\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x}-\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{y}+\mathbf{y}^T\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{y}+\mathbf{y}^T\mathbf{K}^{-1}-\mathbf{z}^T\mathbf{K}^{-1}\mathbf{y}-\mathbf{y}^T\mathbf{K}^{-1}\mathbf{z}+\mathbf{z}^T\mathbf{K}^{-1}\mathbf{z}\\ &\quad=\mathbf{y}^T\left[\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{z}^T\mathbf{K}^{-1}\right]\mathbf{y}-(\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{K}^{-1})\mathbf{y}-\mathbf{y}^T(\mathbf{K}^{-1}\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x})+\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x}+\mathbf{z}^T\mathbf{K}^{-1}\mathbf{z}\end{aligned}$$

Consider now the term $\mathbf{K}^{-1}\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x}$. We have:

$$\begin{aligned}\mathbf{K}^{-1}\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x} &= \mathbf{K}^{-1}\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{z}-\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{z}\\ &= (\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{K}^{-1})\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x}-\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{z}\\ &= (\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}+\mathbf{K}^{-1})\mathbf{z}+\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{z})\end{aligned}$$

Letting $\mathbf{M} = \left[\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F} + \mathbf{K}^{-1}\right]^{-1}$, we obtain:

$$\begin{aligned}\mathbf{K}^{-1}\mathbf{z} + \mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x} &= \mathbf{M}^{-1}\mathbf{z} + \mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{F}\mathbf{z}) \\ &= \mathbf{M}^{-1}(\mathbf{z} + \mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{F}\mathbf{z}))\end{aligned}$$

Now, with $\mathbf{m} = \mathbf{z} + \mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{F}\mathbf{z})$, we can write $\mathbf{K}^{-1}\mathbf{z} + \mathbf{F}^T\mathbf{Q}^{-1}\mathbf{x} = \mathbf{M}^{-1}\mathbf{m}$.

Therefore, we can go back to the expansion of the left-hand side of the main equation and rewrite it as:

$$\begin{aligned}&(\mathbf{x} - \mathbf{F}\mathbf{y})^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{F}\mathbf{y}) + (\mathbf{y} - \mathbf{z})^T\mathbf{K}^{-1}(\mathbf{y} - \mathbf{z}) \\ &\quad = \mathbf{y}^T\mathbf{M}^{-1}\mathbf{y} - \mathbf{m}^T\mathbf{M}^{-1}\mathbf{y} - \mathbf{y}^T\mathbf{M}^{-1}\mathbf{m}^T + \mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x} + \mathbf{z}^T\mathbf{K}^{-1}\mathbf{z} \\ &\quad = \mathbf{y}^T\mathbf{M}^{-1}\mathbf{y} - \mathbf{m}^T\mathbf{M}^{-1}\mathbf{y} - \mathbf{y}^T\mathbf{M}^{-1}\mathbf{m}^T + \mathbf{m}^T\mathbf{M}^{-1}\mathbf{m} - \mathbf{m}^T\mathbf{M}^{-1}\mathbf{m} + \mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x} + \mathbf{z}^T\mathbf{K}^{-1}\mathbf{z}\end{aligned}$$

Now, using equation (B.1), we get:

$$(\mathbf{x}-\mathbf{F}\mathbf{y})^T\mathbf{Q}^{-1}(\mathbf{x}-\mathbf{F}\mathbf{y})+(\mathbf{y}-\mathbf{z})^T\mathbf{K}^{-1}(\mathbf{y}-\mathbf{z}) = (\mathbf{y}-\mathbf{m})^T\mathbf{M}^{-1}(\mathbf{y}-\mathbf{m})-\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}+\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x}+\mathbf{z}^T\mathbf{K}^{-1}\mathbf{z} \tag{B.2}$$

We have shown the first part of the lemma. Consider now the expression $-\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}+\mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x}+\mathbf{z}^T\mathbf{K}^{-1}\mathbf{z}$ in (B.2), and especially the term $\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}$.

Let $\mathbf{n} = \mathbf{F}\mathbf{z}$. We have:

$$\begin{aligned}\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m} &= \left[(\mathbf{z}^T + \mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n})\right]^T\mathbf{M}^{-1}\left[\mathbf{z} + \mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n})\right] \\ &= \mathbf{z}^T\mathbf{M}^{-1}\mathbf{z} + (\mathbf{x} - \mathbf{n})^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{z} + \mathbf{z}^T\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n}) + (\mathbf{x} - \mathbf{n})^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{M}^T\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n})\end{aligned}$$

Define $\mathbf{N} = \mathbf{Q} + \mathbf{F}\mathbf{K}\mathbf{F}^T$. Since $\mathbf{Q}$ and $\mathbf{K}$ are positive definite symmetric, then so is $\mathbf{N}$, and therefore, by the matrix inversion lemma, we have:

$$\begin{aligned}\mathbf{N}^{-1} &= \mathbf{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{F}(\mathbf{K}^{-1} + \mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F})^{-1}\mathbf{F}^T\mathbf{Q}^{-1} \\ &= \mathbf{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{F}\mathbf{M}\mathbf{F}^T\mathbf{Q}^{-1}\end{aligned}$$

Note also that $\mathbf{M} = \mathbf{M}^T$ from the definition of $\mathbf{M}$ and from the fact that $\mathbf{Q}$ and $\mathbf{K}$ are symmetric. As a result, we can write:

$$\begin{aligned}\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m} &= \mathbf{z}^T(\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F} + \mathbf{K}^{-1})\mathbf{z} + (\mathbf{x} - \mathbf{n})^T\mathbf{Q}^{-1}\mathbf{F}\mathbf{z} + \mathbf{z}^T\mathbf{F}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n}) + (\mathbf{x} - \mathbf{n})^T(\mathbf{Q}^{-1} - \mathbf{N}^{-1})(\mathbf{x} - \mathbf{n}) \\ &= \mathbf{n}^T\mathbf{Q}^{-1}\mathbf{n} + \mathbf{z}^T\mathbf{K}^{-1}\mathbf{z} + (\mathbf{x} - \mathbf{n})^T\mathbf{Q}^{-1}\mathbf{n} \\ &\quad + \mathbf{n}^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n}) + (\mathbf{x} - \mathbf{n})^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{n}) - (\mathbf{x} - \mathbf{n})^T\mathbf{N}^{-1}(\mathbf{x} - \mathbf{n}) \\ &= \mathbf{z}^T\mathbf{K}^{-1}\mathbf{z} + \mathbf{x}^T\mathbf{Q}^{-1}\mathbf{x} - (\mathbf{x} - \mathbf{n})^T\mathbf{N}^{-1}(\mathbf{x} - \mathbf{n})\end{aligned}$$

The final result is obtained by replacing the latter expression for $\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}$ in (B.2) □

Bibliography

[1] C. Andrieu, N. De Freitas, and A. Doucet. Sequential mcmc for bayesian model selection. In *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 130–4, Caesarea, Israel, 1999.

[2] B.S. Atal and S.L. Hanauer. Speech analysis and synthesis by linear prediction of the speech wave. 50(2):637–55, August 1971.

[3] M. Bolić, P.M. Djurić, and S. Hong. New resampling algorithms for particle filters. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP'03)*, pages 589–92, Hong Kong, China, 2003.

[4] M. Bolić, P.M. Djurić, and S. Hong. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*, 53(7):2442–2450, July 2005.

[5] J.M. Chambers, C. Mallows, and Stuck B.W. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71:340–344, 1976.

[6] G. Chui, C. K. Chen. *Kalman filtering: with real-time applications.* Springer-Verlag, Berlin, 2nd edition, 1991.

[7] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering, 1998.

[8] A. Doucet, J.F.G. Freitas, and N. Gordon, editors. *Sequential monte carlo methods in practice.* Springer-Verlag, New York, 2001.

[9] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000.

[10] C. Elkan. Cse 291: Statistical learning course notes: Rao-blackwell theorem: Intuition, lemmas, and start of proof. University of California, San Diego, USA, CA., 2005.

[11] Y. Ephraim and H. L. Van Trees. Signal subspace approach for speech enhancement. volume 3, pages 251–266, July 1995.

[12] W. Fong, S.J. Godsill, A. Doucet, and M. West. Monte carlo smoothing with application to audio signal enhancement. *IEEE Transactions on Signal Processing*, 50(2):438–49, Feb 2002.

[13] M. Gabrea. Adaptive kalman filtering-based speech enhancement algorithm. In *Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 521–526, Toronto, Ont., May 2001.

[14] P. Galko. Elg 5119/92.519 stochastic processes course notes. Faculty of Engineering, University of Ottawa, Canada, ON, 2005.

[15] S. Gannot, D. Burshtein, and E. Weinstein. Iterative and sequential kalman filter-based speech enhancement algorithms. *IEEE Transactions on Speech and Audio Processing*, 6(4):373–385, July 1998.

[16] S. J. Godsill, A. Doucet, and M. West. Monte carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

[17] R. G. Goldberg and L. Riek. *A practical handbook of speech coders.* CRC Press LLC, FL, USA, 2000.

[18] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings, Part F: Radar and Signal Processing*, 140(2):107–113, April 1993.

[19] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–37, Feb 2002.

[20] H. Gustafsson, S.E. Nordholm, and I. Claesson. Spectral subtraction using reduced delay convolution and adaptive averaging. *IEEE Transactions on Speech and Audio Processing*, 9(8):799–807, November 2001.

[21] J.H.L. Hansen and B.L. Pellom. An effective quality evaluation protocol for speech enhancement algorithms. In *ICSLP, International Conference on Spoken Language Processing*, volume 7, pages 2819–2822, Sydney, Australia, December 1998.

[22] K. Hasan and L. Akter. Quality improvement of enhanced speech in DCT domain using modified a priori SNR. In *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*, pages 733–6, Darmstadt, Germany, 2004.

[23] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, New Jersey, 1999.

[24] S. Haykin. *Kalman filtering and neural networks.* John Wiley & Sons, Inc., New York, 2001.

[25] Y. Hu, M. Bhatnagar, and P.C. Loizou. A cross-correlation technique for enhancing speech corrupted with correlated noise. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 1, pages 673–676, Salt Lake, UT, May 2001.

[26] Y. Hu and P. C. Loizou. Evaluation of objective measures for speech enhancement. In *International Conference on Spoken Language Processing, INTERSPEECH - Proceedings*, Pittsburg, PA, USA, September 2006.

[27] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

[28] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[29] N. Kitawaki, K. Itoh, M. Honda, and K. Kakehi. Comparison of objective speech quality measures for voiceband codes. In *Proceedings of ICASSP 82. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1000–3, Paris, France, 1982.

[30] D.H. Klatt. Prediction of perceived phonetic distance from critical-band spectra: a first step. In *Proceedings of ICASSP 82. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1278–81, Paris, France, 1982.

[31] M. Klein. Signal subspace speech enhancement with perceptual post-filtering. Master's thesis, McGill University, Montreal, QC, Canada, 2002.

[32] K. Y. Lee and K. Shirai. Efficient recursive estimation for speech enhancement in colored noise. *IEEE Signal Processing Letters*, 3(7):196–199, July 1996.

[33] Jae S. Lim and Alan V. Oppenheim. All-pole modeling of degraded speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(3):197–210, June 1978.

[34] M. Lombardi. *Simulation-based Estimation Methods for α-Stable Distributions and Processes.* PhD thesis, Universita Degli Studi Di Firenze, 2004.

[35] M. Lombardi and S. Godsill. Online bayesian estimation of signals in α-stable noise. *IEEE Transactions On Signal Processings*, 53(6):775–779, February 2006.

[36] N. Ma, M. Bouchard, and R.A. Goubran. Dual perceptually constrained unscented kalman filter for enhancing speech degraded by colored noise. In *2004 7th International Conference on Signal Processing Proceedings*, pages 2522–5, Beijing, China, 2004.

[37] Ning Ma, M. Bouchard, and R.A. Goubran. Speech enhancement using a masking threshold constrained kalman filter and its heuristic implementations. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):19–32, January 2006.

[38] E.N. Marieb. *Human anatomy and physiology.* Benjamin-Cummings Publishing Company, Menlo Park, CA, USA, 1998.

[39] M. Marzinzik. *Noise Reduction Schemes for Digital Hearing Aids and their Use for the Hearing Impaired.* PhD thesis, Carl von Ossietzky University Oldenburg, 2000.

[40] I.A. McCowan, D.C. Moore, and S. Sridharan. Near-field adaptive beamformer for robust speech recognition. *Digital Signal Processing*, 12(1):87–106, January 2002.

[41] F. Mustiere, M. Bolić, and M. Bouchard. A modified rao-blackwellised particle filter. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[42] F. Mustiere, M. Bolić, and M. Bouchard. Rao-blackwellised particle filters: examples of applications. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Ottawa, Canada, May 2006.

[43] K. K. Paliwal and Anjan Basu. Speech enhancement method based on kalman filtering. In *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 177–180, Dallas, TX, USA, 1987.

[44] P.E. Papamichalis. *Practical approaches to speech coding.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[45] S.R. Quackenbush, T.P. Barnwell III, and M.A. Clements. *Objective Measures of Speech Quality.* Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[46] T. Quatieri. *Discrete-time speech signal processing: principle and practice.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[47] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: particle filters for tracking applications.* Artech House, London, Feb 2004.

[48] R.D. Shachter. Bayes-ball: the rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of Uncertainty in Artificial Intelligence (UAI-98)*, pages 480–487, Madison, WI, USA, 1998.

[49] R. van der Merwe, A. Doucet, J.F.G. Freitas, and E. Wan. The unscented particle filter. Technical Report CUED/F-INFENG/TR380, Cambridge University Engineering Department, Cambridge, UK, August 2000.

[50] J. Vermaak, C. Andrieu, A. Doucet, and S.J. Godsill. Particle methods for bayesian modeling and enhancement of speech signals. *IEEE Transactions on Speech and Audio Processing*, 10(3):173–85, March 2002.

[51] N. Virag. Single channel speech enhancement based on masking properties of the human auditory system. *IEEE Transactions on Speech and Audio Processing*, 7(2):126–137, March 1999.

[52] R. Viswanathan, W. Russell, and J. Makhoul. Objective speech quality evaluation of narrowband LPC vocoders. In *Proceedings of the 1978 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 591–4, Tulsa, OK, USA, 1978.

[53] E. A. Wan and A. T. Nelson. Removal of noise from speech using the dual EKF algorithm. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 1, pages 381–384, Seattler, WA, USA, May 1998.